VOLUME 2 | ISSUE 1 | YEAR 2024

ISSN: 2980-3152

CURRENT TRENDS IN COMPUTING

https://dergipark.org.tr/en/pub/ctc

Current Trends in Computing (CTC)

Advisory Board

Scope

Current Trends in Computing (CTC) is a single-blind, peer-reviewed international scientific journal with an open-access policy. CTC was founded in 2022 by Computer Engineering and Software Engineering Departments, Karabük University (TÜRKİYE). It regularly publishes two issues. The journal does not charge submission and publication fees. The journal accepts submissions of manuscripts in the English language only. CTC is devoted to publishing original research in the niche area of computer sciences.

Contents

Follow this issue and upcoming issues at: https://dergipark.org.tr/en/pub/ctc

Vol: 2, Issue: 1, 2024
ISSN: 2980-3152
Pages:1-12

CURRENT TRENDS IN COMPUTING
https://dergipark.org.tr/en/pub/ctc
RESEARCH ARTICLE

AUTOENCODER-BASED INTRUSION DETECTION IN CRITICAL INFRASTRUCTURES

HAKAN CAN ALTUNAY[1], ZAFER ALBAYRAK[2*] AND MUHAMMET ÇAKMAK[3]

[1] *Department of Computer Technologies, Carsamba Chamber of Commerce Vocational School, Ondokuz Mayis University, Türkiye*
[2] *Department of Computer Engineering, Faculty of Technology, Sakarya University of Applied Sciences, Türkiye*
[3] *Department of Computer Engineering, Faculty of Engineering and Architecture, Sinop University, Türkiye*

ABSTRACT. **Securing critical infrastructure systems such as electricity, energy, health, management, transportation, and production facilities against cyber attacks is the issue on which states spend the most time and money when creating security strategies. Every day, different methods have emerged to prevent attackers who endanger our personal and national security with varying types of attacks. The most important of these methods is intrusion detection systems. This study proposes an autoencoder-based intrusion detection system model to detect security anomalies in critical infrastructures. The accuracy of this proposed model in attack detection has been tested with the current and complex UNSW-NB15 dataset. In the proposed model, training and testing steps were carried out using the attack packages in the data set. These packages are then divided into two: normal or attack. According to the results obtained in the experiments, it has been confirmed that the proposed intrusion detection system can effectively detect attacks with high performance.**

1. INTRODUCTION

Modern societies depend on advanced cyber and physical infrastructures to carry out daily activities [1]. These infrastructures are also called critical assets that protect services not only in the physical world but also in the digital world. Today, the protection of these infrastructures, which cover different areas such as communication, transportation, and energy, has become a national security concern [2]. Ensuring the continuity, control, and security of the services provided by critical infrastructures is a costly and difficult process [3].

Cyber attacks are carried out against SCADA systems in various areas, such as nuclear power plants, electrical networks, and water treatment plants [4]. Figure 1 shows the major attacks on industrial control systems around the world since 2010.

E-mail address: zaferalbayrak@subu.edu.tr (*).
Key words and phrases. Intrusion Detection System, Critical Infrastructure, Autoencoder.

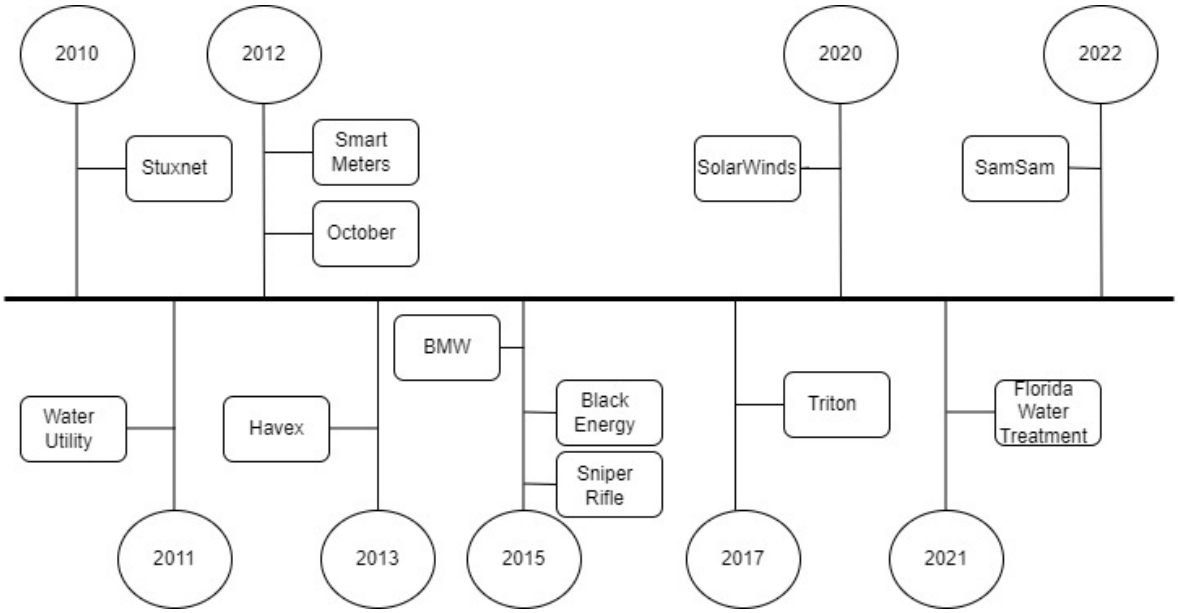

FIGURE 1. **Major attacks on critical infrastructures.**

It is a fact that the use of the Internet in our daily lives increases information sharing, interpersonal communication, and interaction [5]. The Internet of Things (IoT), defined as the intelligent connection of objects that communicate by sensing each other, is frequently used today. With IoT, data communication on devices in the network can be monitored, and this data can be collected using sensors with a wireless network connection [6].

Intrusion detection systems (IDS) are preferred to prevent cyber attacks and reduce their effects. Security is provided by IDS in the transmission of data on the network from the source to the receiving node. Therefore, IDS plays an important role in ensuring network security. Machine learning is a sub-branch of artificial intelligence [7]. With its ability and capacity to improve, it can empower various systems to learn from experience and make decisions without any explicit programming. Machine learning approaches are generally divided into supervised and unsupervised [8]. Classification of traditional IDSs is generally presented as signature-based, anomaly-based, and hybrid IDSs. Signature-based IDSs extract behavioral patterns of intruders [9]. IDSs are generally classified as anomaly-based, signature-based, and hybrid-based. IDSs in which the behavior of attackers trying to enter the network without permission is kept as signatures are called signature-based IDSs [10]. These signatures are compared with the attack types that the network has encountered before [11]. If the signatures match as a result of this comparison, the packet is detected as an attack. This type of IDS does not produce false positive values. They detect any intrusion with a signature pattern. In these IDSs, attacks with unknown signatures cannot be detected, and accordingly, a high rate of false negative values is produced [12]. If the database has an updateable feature, the false negative value rate can be reduced. IDSs that analyze events in the network

are anomaly-based IDSs. In this type of IDS, normal states and abnormal states are distinguished [13]. In anomaly-based IDSs, the behavioral profiles of users in the system are first determined. Behaviors that differ from normal behaviors are defined as abnormal behaviors. The higher the correct detection rate of normal behavior profiles, the higher the correct detection rate of abnormal behaviors. In anomaly-based IDSs, normal behaviors are continuously updated [14]. Some attack types cannot be detected by anomaly-based intrusion detection systems. Therefore, anomaly-based systems may have a high false positive rate. In hybrid-based systems, signature and anomaly-based systems are used together [15]. In this way, a much more reliable network and management system emerges [16].

Considering the above problems, the main subject of this study is the design of an autoencoder-based intrusion detection system to detect attack packets by detecting intrusions with high performance in critical infrastructure systems where the types of attacks and the amount of data increase day by day. The proposed model subjects intrusion packets to binary classification.

The primary motivation and contribution of this study are summarized below. The number of cyber attacks on critical infrastructures is increasing day by day. These attacks cause material and moral losses. Therefore, it is important to detect these attacks and protect the system. In this study, an intrusion detection system for critical infrastructures is implemented using an autoencoder.

Secondly, the proposed model for intrusion detection systems was tested on UNSW-NB15, a complex dataset with a large amount of data and a high number of attributes, taking into account the increasing amount of data in critical infrastructures. The performance of the proposed model was evaluated using the binary classification procedure carried out on the current data set, providing a more reliable and observable process.

2. Related Work

Davis and Clark suggested an in-depth examination of request packets to increase the performance of anomaly-based IDSs developed with the machine learning method and to detect increasing types of attacks and emphasized that data pre-processing has a great impact on the success of anomaly-based IDSs [17].

In their study, Naseer and Saleem tried various categorical data coding methods, chose the most appropriate method for the data set they used, and performed hyperparameter optimization by using the random search method in the models established with the Deep Convolutional Neural Network (DCNN) algorithm. It has been stated that pre-processing and hyper-parameter optimization significantly improve the attack detection rate and speed of the created models [18]. In another study, Hancock and Khoshgoftaar emphasized that stable categorical data coding techniques are suitable for large data sets due to their low running time and low computational complexity [19]. Tang et al. reached an accuracy rate of 89.82% in the data set on which they applied categorical data coding with the one hot encoding method and feature selection pre-processing with the Light Gradient Boosting Machine (LightGBM) algorithm and the attack detection model they created with the Autoencoder (AE) algorithm [20].

Aslan et al. analyzed the malware behavior in the system and proposed a Subtractive Central Behavior Model to detect this malware. In the proposed model, attributes were created by analyzing malware

behaviors and the system in which the behaviors were performed. Additionally, the obtained features were reduced by proposing a new feature selection algorithm. With the proposed model, 99.9%, 0.2%, and 99.8% rates were achieved in detection rate, false positive rate, and accuracy metrics, respectively [21].

Mazini et al. applied hyperparameter optimization to the data set after categorical data coding and scaling pre-processing and made feature selection with the artificial bee colony algorithm. The resulting data set and AdaBoost. In the model created with the M2 classifier, 99.61% detection, 0.01% false detection, and 98.90% correct detection rates were achieved [22]. By selecting features according to the information gain rate, Balakrishnan et al. achieved 99.11% detection success and 2.08s detection time in Denial of Service (DoS) attacks with the data set with the resulting feature subset and the Support Vector Machine (SVM) classification algorithm [23].

Torabi et al. mentioned the importance of using different and up-to-date data sets to prove the generalization success of the developed intrusion detection models [24]. Ozkan Okay et al. proposed a hybrid attack detection model and achieved 99.65% and 99.17% accuracy rates with KDD99 and UNSW-NB15 datasets, which were pre-processed with a feature selection approach (FSAP) algorithm [25].

Ambusaidi et al. achieved 98.90% attack detection success and 0.521% false positive rate with the data sets on which they applied hybrid feature selection using mutual information (MI) and helical sequential forward selection (SFS) methods [26].

Chen et al. have used datasets consisting of different combinations and intersections of features selected by Principal Component Analysis (PCA), C4.5, and Genetic Algorithm (GA) techniques, achieved the most successful results with the features selected jointly by PCA and GA techniques [27].

Song mentioned that since traditional feature selection algorithms are insufficient for variable-size datasets, this problem can be solved with online feature selection algorithms [28].

Kanimozhi and Jacob performed hyperparameter optimization for the number of hidden layers and alpha parameters using the grid search method in the anomaly-based intrusion detection model they created using the Multilayer Perceptrons (MLP) algorithm, and achieved 99.97% accuracy, 0.001% false positive and 99% F-criterion rates [29].

In their study, Latah and Toker used the NSL-KDD dataset for anomaly-based attack detection in software-defined networks (SDN), 12 different classifiers, and the PCA approach for feature extraction from the dataset. As a result of the experiments, the model established with the Decision Tree (DT) algorithm showed the best performance in precision, AUC, F1-measure, McNemar, and accuracy metrics. While bagging and boosting approaches outperform other traditional machine learning methods such as Extreme Learning Machines (ELM), K Nearest Neighbors (KNN), Random Forest (RF), Neural Networks (NN), Latent Dirichlet Allocation (LDA), and SVM with a 99.5% confidence level, the best results were achieved in FAR and recall metrics with LogitBoost. The best test time was obtained with ELM [30].

Uğurlu et al. In their study, they selected 30 attributes through the weighting process from 82 attributes in the CICDarknet2020 data set, which they used to detect and classify darknet traffic. In the study, the grid method was used for hyper-parameter adjustment, and as a result of the experimental studies, an accuracy rate of 93.32% was achieved with the Decision Tree algorithm [31].

3. DATASET

Using the UNSW-NB15 dataset IXIA PerfectStorm tool, a hybrid model was created in Australian cyber security center laboratories, including both real modern normal activity and artificial network traffic attack movements suitable for today's conditions [32].

The developers of the dataset also divided the dataset into two different groups: the training dataset and the testing dataset. This data set was later used by many researchers. The training data set consists of 175341 records, and the test data set consists of 82332 records. The original data set consists of 2540044 records [33]. In this study, the subsample data set, which was created by the developers of the original data set and divided into training and test data sets, which many researchers use in their studies, was used as the data set. The data set used does not contain any unnecessary records. The UNSW-NB15 dataset has a total of 49 features and one target value. The value distribution of the types of attacks in the UNSW NB15 dataset is presented in Table 1.

TABLE 1. **Distribution of attack values in the UNSW - NB15 dataset**

Attack Types	Number
Fuzzers	18184
Backdoor	1746
Analysis	2000
General	40000
Shellcode	1133
Reconnaissance	10491
DoS	12264
Worms	130
Exploits	33393
Benign	56000

In recent years, IDS has been increasing performance using deep learning methods at points where existing traditional security solutions are insufficient. In particular, anomaly-based IDSs play a very important role in detecting attacks known as zero-day attacks. One of the most important factors to evaluate the performance of IDSs and to create more effective and efficient IDSs is the data sets used [32]. The data set used must comply with the age requirements and include current attack types. The UNSW-NB15 data set, which is frequently preferred in the literature, meets modern conditions in a variety of attack types and normal traffic scenarios, and the regular distribution of training and test data sets are the positive aspects of this data set [33].

Deep autoencoders, a specific application of artificial neural networks, are used to perform unsupervised learning. In the deep autoencoder, the data is first compressed and encoded. Then, a representation closest to the input data is obtained from the code whose features are reduced. [34].

The autoencoder learns how to remove noise from the data to reduce data sizes. An autoencoder consists of 3 parts: encoder, code, and decoder. The encoder is where the input data is compressed. In this section, the code is generated. The decoder reconstructs the input data using this code. In other words, an autoencoder cannot be created without an encoder, decoder, and loss function. The loss function is

used in the autoencoder to compare the output with the targeted result [35]. Figure 2 shows a general deep autoencoder architecture.

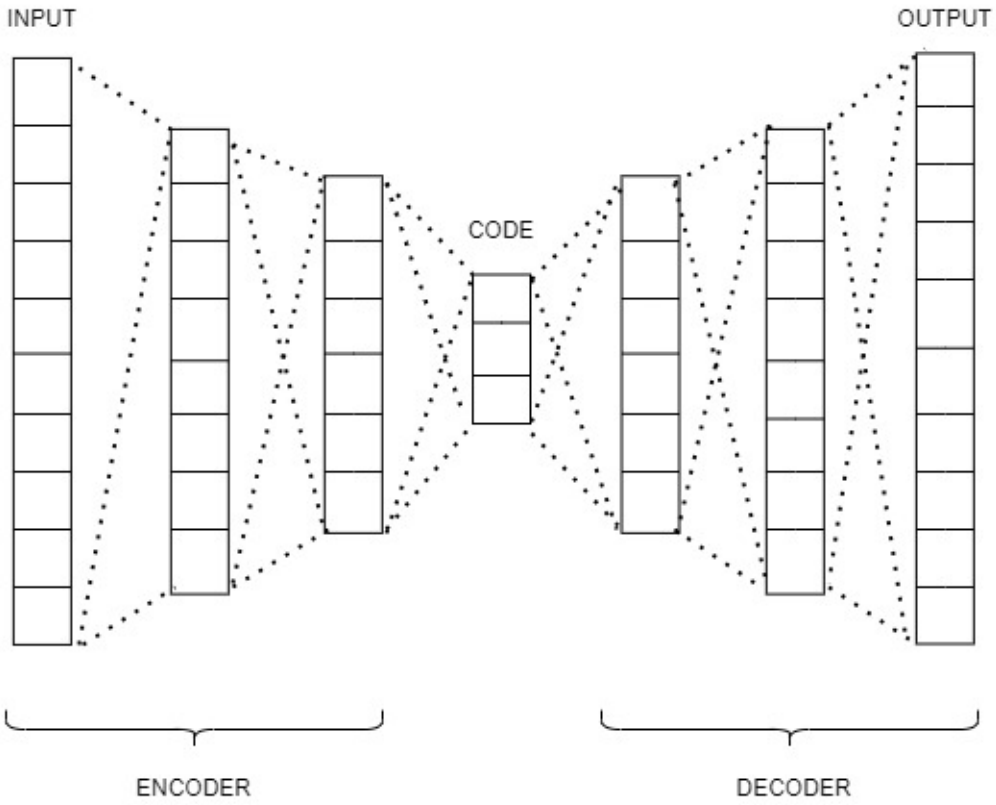

FIGURE 2. **Deep autoencoder architecture.**

The equations of the encoder and decoder sections are shown below.

$$Y = f_{\emptyset}(X) = s(WX + b_x) \tag{1}$$

$$X^{'} = g_{\emptyset'}(Y) = s(W^{'}Y + b_Y) \tag{2}$$

During the autoencoder training process, reconstruction loss is minimized in the dataset. The objective function is used here. The following equation is used to determine the parameters that will minimize the loss value along with the objective function.

$$\emptyset = minL(X, X^{'}) = minL(X, g((f(X)))) \tag{3}$$

Autoencoder is the deep learning model used in this study. Recall, precision, F1-Score, and accuracy were used for evaluation criteria as in [36]. The following equations were used to obtain the relevant metrics.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{4}$$

$$Recall = \frac{TP}{TP+FN} \tag{5}$$

$$Precision = \frac{TP}{TP+FP} \tag{6}$$

$$F1-Score = \frac{2*Precision*Recall}{Precision+Recall} \tag{7}$$

First, data from the training and test sections of the data set were taken. These data were then applied to the Autoencoder model. At the output of the Autoencoder model, the data is classified as attack or normal. Figure 3 shows the architecture of the proposed model.

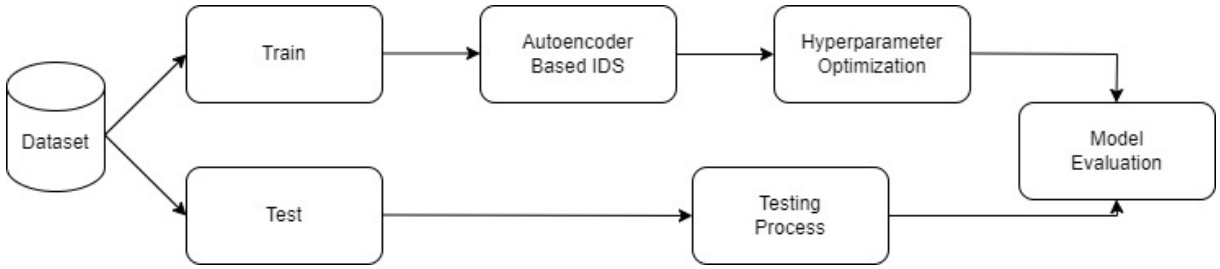

FIGURE 3. **Architecture of the proposed model.**

The experimental study divided the data set into training and testing to prevent overfitting. One of the critical issues of machine learning is the generalization of the algorithms or models we have developed. Generalization is the ability to observe how well the model works with the data you have learned and with new, previously unseen data we will obtain in the future. We can briefly define it as getting good results with the latest data.

Our primary and greatest goal in machine learning studies is to create a model that accurately predicts previously unknown data elements. Therefore, the created learning model must be generalized very well to ensure the accurate classification of future data items. Generalization means our model is good at learning from given data and applying the learned information elsewhere. If it performs well on data it has not seen in training, it generalizes well on the provided data [37]. This study's data set is divided into 80% training and 20% testing.

The multi-layered architectural structures that come with deep learning have brought a series of hyper-parameter groups waiting to be decided by the designer. Some of these parameters are used to select

the basic algorithm to be applied in the model from several algorithm groups, such as the optimization algorithm and activation function. Since the number of algorithms is limited, it is generally relatively easy to select such hyperparameters.

However, the number of layers, neurons, learning coefficients, kernel size, etc. Hyperparameter types also expect us to choose from a set that extends to infinity within certain limits or on the number line. The selection of such hyperparameters is a laborious and time-consuming process. Our first choices regarding hyperparameters when designing a model do not yield the right results. By changing the hyperparameters one after the other iteratively, the model's performance is observed, and the most appropriate hyperparameter group for the model is selected. In addition, some methods automate this selection process.

In this study, the heuristic parameter fitting method was used for hyperparameter optimization. In this method, hyperparameters are estimated using our prior knowledge of the problem, the model is designed according to these hyperparameters, and the results are observed. According to the results, the model is rebuilt and trained by making new hyperparameter estimates that will intuitively increase the model's performance, and the results are observed. This process continues until suitable parameter groups that will give the expected performance are found [38]. Hyperparameters of the model used in the study are shown in Table 2.

TABLE 2. **Hyperparameters of the proposed model**

Hyperparameters	**Value**
Input Neurons	45
Hidden Neurons	32
Output Neurons	45
Iteration	650

4. RESULTS AND DISCUSSION

The study used the autoencoder model with original data without making a feature selection in performance evaluation. Without feature selection, the training dataset and testing dataset were used separately. The results obtained are shown in Table 3 and Figure 4.

According to these results, the proposed autoencoder-based intrusion detection system reached 97.63% accuracy in determining attack packets. The accuracy value obtained was supported by precision, recall, and F1 Score values. Hyperparameter optimization was carried out in the tests carried out with the UNSW-NB15 data set, which is very rich in terms of the number and diversity of attack packages. The ROC curve obtained as a result of the study is shown in Figure 5.

The accuracy rates obtained in studies using different data sets in the literature and the rates obtained in this study are shown in Table 4. The results obtained show that the autoencoder method has a high detection accuracy in attack classification.

TABLE 3. **Performance metrics and results**

Metrics	Value
Accuracy	97.63%
Precision	97.14%
Recall	97.78%
F1 - Score	97.45%

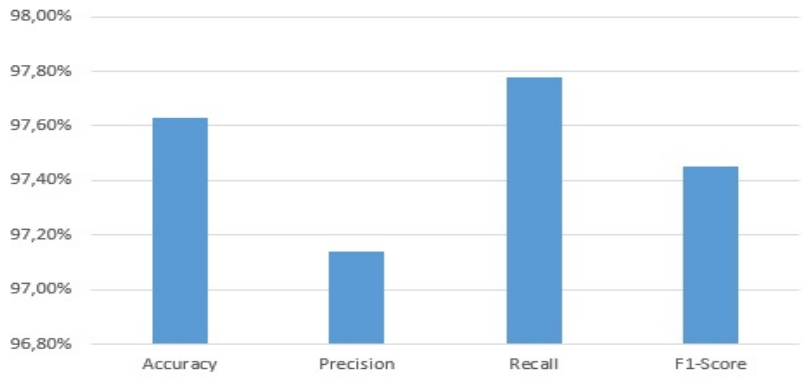

FIGURE 4. **Results obtained in the experimental study.**

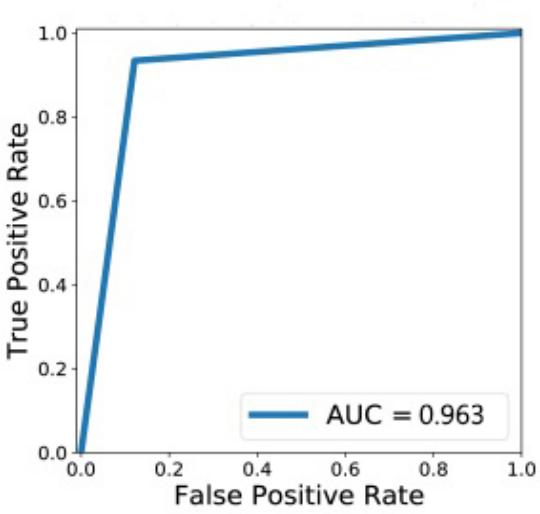

FIGURE 5. **ROC of proposed model.**

TABLE 4. **Comparison of the proposed model with studies in the literature**

Reference	Model	Accuracy
[18]	DCNN	85.22%
[20]	LightGBM + Autoencoder	89.82%
[21]	SCBM + J48	99.8%
[22]	Artificial Bee Colony + AdaBoost	98.90%
[31]	Decision Tree	93.22%
Proposed model	Autoencoder	97.63%

5. CONCLUSION

In our study, an autoencoder-based intrusion detection system is proposed. In this system, which can detect abnormal behavior in the network with high performance, no feature extraction is made from the data set. According to the results, the proposed autoencoder model reached a 97.63% accuracy value. In addition, 97.14% precision, 97.78% recall, and 97.45% F1-Score values were achieved in the model. It is seen that this study achieves higher performance compared to other studies in the literature. The main reason for this situation is the use of an up-to-date data set and hyperparameter optimization. It is planned to prepare algorithms based on feature selection in the future. In this way, the effect of feature selection on classification accuracy will be investigated. In addition, future studies need to examine the detection time of attack symptoms. Considering that the number of institutions and organizations with critical infrastructure is increasing day by day, it is thought that deep learning-based intrusion detection systems will be needed, especially in this field.

DECLARATIONS

- **Conflict of interest:** The authors have not disclosed any competing interests.
- **Data availability:** The data will be shared upon request.

REFERENCES

[1] Kasongo S.M., Sun Y., Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset, J Big Data, 7, 105, 2020.

[2] Yadav M.S., Kalpana R., Data Preprocessing for Intrusion Detection System Using Encoding and Normalization Approaches, 2019 11th International Conference on Advanced Computing (ICoAC), ChennaiIndia, 265-269, 18-20 Aralık, 2019.

[3] Liu H., Zhou M., Liu Q., An embedded feature selection method for imbalanced data classification, in IEEE/CAA Journal of Automatica Sinica, 6 (3), 703-715, 2019.

[4] Alabadi, M., Habbal, A., Wei, X., Industrial internet of things: Requirements, architecture, challenges, and future research directions, IEEE Access, 2022.

[5] Alaca, Y.,Çelik, Y., Cyber attack detection with QR code images using lightweight deep learning models. Computers & Security, 126, 103065, 2023.

[6] Kutluana, G., Turker, I., Classification of cardiac disorders using weighted visibility graph features from ECG signals, Biomedical Signal Processing and Control, 87, 105420, 2024.
[7] Altunay, H. C., Kritik Altyapılara Yönelik Derin Öğrenme Tabanlı Saldırı Tespit Sistemi Tasarımı, (Doctoral dissertation), 2023.
[8] Altunay, H., C., Albayrak, Z., Network Intrusion Detection Approach Based on Convolutional Neural Network, Avrupa Bilim ve Teknoloji Dergisi, (26), 22-29, 2021.
[9] Bharadiya, J. P., Machine learning and AI in business intelligence: Trends and opportunities, International Journal of Computer (IJC), 48(1), 123-134, 2023.
[10] Sharifani, K., Amini, M., Machine Learning and Deep Learning: A Review of Methods and Applications, World Information Technology and Engineering Journal, 10(07), 3897-3904, 2023.
[11] Choi, S., Yoon, S., Energy signature-based clustering using open data for urban building energy analysis toward carbon neutrality: A case study on electricity change under COVID-19, Sustainable Cities and Society, 92, 104471, 2023.
[12] Landauer, M., Wurzenberger, M., Skopik, F., Hotwagner, W., Höld, G., Aminer: A modular log data analysis pipeline for anomaly-based intrusion detection, Digital Threats: Research and Practice, 4(1), 1-16,2023.
[13] Bhavsar, M., Roy, K., Kelly, J., Olusola, O., Anomaly-based intrusion detection system for IoT application, Discover Internet of Things, 3(1), 5, 2023.
[14] Sharma, B., Sharma, L., Lal, C., Roy, S., Anomaly based network intrusion detection for IoT attacks using deep learning technique, Computers and Electrical Engineering, 107, 108626, 2023.
[15] Hnamte, V., Hussain, J., DCNNBiLSTM: An efficient hybrid deep learning-based intrusion detection system, Telematics and Informatics Reports, 10, 100053, 2023.
[16] Yin, Y., Jang-Jaccard, J., Xu, W., Singh, A., Zhu, J., Sabrina, F., Kwak, J., IGRF-RFE: a hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset, Journal of Big Data, 10(1), 1-26, 2023.
[17] Davis J.J., Clark A.J., Data preprocessing for anomaly based network intrusion detection: A review, Computers & Security, 30 (6-7), 353- 375, 2011.
[18] Naseer S., Saleem Y., Enhanced Network Intrusion Detection Using Deep Convolutional Neural Networks, KSII Trans. Internet Inf. Syst, 12 (10), 5159-5178, 2018.
[19] Hancock J.T., Khoshgoftaar T.M., Survey on categorical data for neural networks, Journal of Big Data, 7, 1-41, 2020.
[20] Tang C., Luktarhan N., Zhao Y., An Efficient Intrusion Detection Method Based on LightGBM and Autoencoder. Symmetry, 12 (9), 1458, 2020.
[21] Aslan, Ö., Samet, R., Tanriöver, Ö.Ö, Using a Subtractive Center Behavioral Model to Detect Malware, Secur. Commun. Networks, 7501894, 1-17, 2020.
[22] Mazini M., Shirazi B., Mahdavi I., Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms, Journal of King Saud University - Computer and Information Sciences, 32 (10), 1206-1207, 2019.
[23] Balakrishnan S.M., Venkatalakshmi K., Kannan A., Intrusion Detection System Using Feature Selection and Classification Technique, IJCSA, 3 (4), 145, 2014.
[24] Torabi M., Udzir N.I., Abdullah M.T., Yaakob R.A., Review on Feature Selection and Ensemble Techniques for Intrusion Detection System, IJACSA, 12 (5), 538-553, 2021.
[25] Özkan Okay M., Aslan Ö., Eryiğit R., Samet R., SABADT: Hybrid Intrusion Detection Approach for Cyber Attacks Identification in WLAN, IEEE Access, 9, 157639-157653, 2021.
[26] Ambusaidi M.A., He X., Tan Z., Nanda P., Lu L.F., Nagar U.T., A Novel Feature Selection Approach for Intrusion Detection Data Classification, 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, BeijingChina, 82-89, 24-26 Eylül, 2014.
[27] Chen C.W., Tsai Y.H., Chang F.R., Lin W.C., Ensemble feature selection in medical datasets: Combining filter, wrapper, and embedded feature selection results, Expert Systems, 37 (5), e12553, 2020.

[28] Song J., Feature selection for intrusion detection system, Ph.D. Thesis, Aberystwyth University, Department of Computer Science Institute of Mathematics, Physics and Computer Science, Penglais-UK, 2016.
[29] Kanimozhi V., Jacob P., Artificial Intelligence based Network Intrusion Detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing, ICT Express, 5 (3), 211-214, 2019.
[30] Latah M., Toker L., Towards an efficient anomaly-based intrusion detection for software-defined networks, IET Netw., 7, 453-459, 2018.
[31] Uğurlu M., Doğru İ. A., Arslan R.S., Detection and classification of darknet traffic using machine learning methods, Journal of the Faculty of Engineering and Architecture of Gazi University, 38 (3), 1737-1746, 2023.
[32] Aleesa, A., Younis, M. O. H. A. M. M. E. D., Mohammed, A. A., Sahar, N., Deep-intrusion detection system with enhanced UNSW-NB15 dataset based on deep learning techniques, Journal of Engineering Science and Technology, 16(1), 711-727, 2021.
[33] Choudhary, S., Kesswani, N., Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT, Procedia Computer Science, 167, 1561-1573, 2020.
[34] Yousefi-Azar, M., Varadharajan, V., Hamey, L., Tupakula, U., Autoencoder-based feature learning for cyber security applications. In 2017 International joint conference on neural networks (IJCNN) (pp. 3854-3861), IEEE, 2017.
[35] Basati, A., Faghih, M., M., APAE: an IoT intrusion detection system using asymmetric parallel auto-encoder. Neural Computing and Applications, 35(7), 4813-4833, 2023.
[36] Altunay, H., C., Albayrak, Z., A hybrid CNN+ LSTM-based intrusion detection system for industrial IoT networks, Engineering Science and Technology, an International Journal, 38, 101322, 2023.
[37] Abedi, A., Khan, S. S., Fedsl: Federated split learning on distributed sequential data in recurrent neural networks, Multimedia Tools and Applications, 1-212, 2023.
[38] Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Lindauer, M., Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 13(2), e1484, 2023.

Vol: 2, Issue: 1, 2024
ISSN: 2980-3152
Pages:13-22

CURRENT TRENDS IN COMPUTING
https://dergipark.org.tr/en/pub/ctc
RESEARCH ARTICLE

DEVELOPING LOW-COST TORQUE MEASUREMENT SYSTEM

CENGIZ AYTEN[1] AND FERHAT ATASOY[2*]

[1]*Institute of Postgraduate Education, Karabük University, 78050, Karabük, Turkiye*

[2]*Computer Engineering Department, Karabük University, 78050, Karabük, Turkiye*

ABSTRACT. **Torque measurement is critical in industrial, experimental, and medical applications. Various methods for torque measurement have been introduced in the literature, and application-specific solutions have been developed. The present work proposes a hybrid method using a rotary encoder to measure angular displacement and challenges related to external light conditions and resolution limitations to overcome. Experimental setups demonstrate the system's ability to successfully measure rotary mechanisms' transient (starting and stopping) torque without noise, providing a cost-effective, numerically accurate solution. Experimental studies have shown promising results.**

1. INTRODUCTION

Electric motors are widely used as an environmentally friendly alternative to internal combustion engines in the vehicle industry, conveyors, cranes, and production machines in industrial production, refrigerators, washing machines, fans, and vacuum cleaners in household appliances, wind turbines, propulsion systems of spacecraft, medical devices, curtain, door, window control in smart home automation, drills and saws in electric vehicles [1]. Especially in industrial applications, motor torque measurements are needed for performance optimization by optimizing motor efficiency and output, preventing overload and damage by ensuring operation within safe limits, quality and consistency in production, and energy optimization. In addition, torque measurements are gaining importance for precise control in medical devices and aerospace applications.

Torque can be defined as the measure of the rotational force applied to an object about an axis [2]. Looking at the literature, various methods developed for torque measurement stand out. Mathematical simulation, dynamo-meter, Strain Gauge method, current and voltage measurement, optical methods, piezoelectric sensors, and hall effect sensors are the most widely used methods in the literature [3, 4].

Mathematical simulation is a cost-effective solution that does not require physical components and installation, enables safe testing under extreme conditions, and can model a wide range of scenarios and

E-mail address: ferhatatasoy@karabuk.edu.tr .
Key words and phrases. Torque measure system, Encoder, Arduino.

conditions [4, 5]. However, the results depend on the accuracy of the model and input data. In addition, unexpected real-world conditions are ignored.

The dynamo-meter is one of the most widespread methods because it offers direct physical measurement by connecting to the shaft of the motor, is applicable to a wide range of motors and machines, and provides reliable and repeatable results [6, 7]. However, it is more expensive than many other methods, requiring equipment, physical space, installation, regular maintenance, and calibration.

In the Strain Gauge method, a strain gauge is connected to the motor shaft, and the torque is calculated by measuring the stress in the motor shaft. It stands out with its high accuracy, precise measurement, and small size [8]. However, the disadvantages are that they are easily damaged, require careful installation and calibration, and are affected by environmental factors such as temperature and humidity.

In the current and voltage measurement method, torque is calculated by comparing it with the performance curves of the motor. As such, it is easy to apply and understand and is cheaper than other methods [7]. However, the fact that it requires calibration and works with less accuracy than direct measurement methods prevents it from being used in every field.

The advantages of optical methods are that they measure torque without physical contact, are suitable for high-speed applications, and allow highly accurate measurements. However, they have disadvantages such as more complex setup and calculations, more expensive compared to electrical methods, and being affected by external light conditions [9, 10].

Piezoelectric sensors generate electrical signals under mechanical stress and are used for torque measurement. The high-frequency response makes them suitable for dynamic measurements. Long-term stability in measurements, wide measurement range, and relatively small size are its advantages [11]. However, cost, sensitivity to temperature and vibrations, and the need for specialized and complex electronic circuitry for signal processing make it difficult to use.

The advantages of the Hall effect sensor are that it performs non-contact measurement, is robust against environmental conditions, and can be used in various applications [12], while its disadvantages are that it requires careful calibration, has a limited measurement range, is affected by magnetic fields and its performance changes with temperature.

The proposed method uses a rotary encoder to overcome issues with external light conditions and resolution limitations. The system provides an effective measurement of starting and stopping torque, offering an accurate, cost-effective solution. The disadvantage of the presented method is that it needs an inertial moment of the system.

2. Related Work

In previous studies, there are applications where existing methods are customized according to the application. In the study presented by Ashwindran et al., an Arduino-based system was developed to measure the torque of wind turbines [13]. The system consists of two subsystems, including a photo interrupter (primary) and a load cell (secondary). The developed system has been tested in both laboratory and simulation environments and has been shown to provide reliable results. The study is proposed as a cost-effective solution for the measurement of rotating machine torques.

Brusamarello et al. presented a system mounted on an aluminum alloy wheel for automotive applications [14]. In the developed system, the signal received from the strain gauge is amplified and filtered, and then analog-to-digital conversion is performed and sent to a remote computer via the ZigBee transmission module. The first dynamic tests of the system calibrated with static loads were performed under flat road conditions.

Bayraktar and Güldaş presented a study on the measurement and optimization of thrust and torque forces in unmanned aerial vehicles, especially quadrotors, and applied regression analysis in Matlab/Simulink environment to experimental measurements to minimize errors in trajectory tracking [15]. According to the results of the study, cubic and quadratic force equations give better results in trajectory tracking than other methods. The device sold by Surkon Makine Ltd. has been developed to measure the opening torque of bottle caps [16].

Caruana et al. presented a torque measurement system for use between the crankshaft of an internal combustion engine and an AC motor. A fully blind strain gauge and an electronic amplifier are used in the study. In the developed system, data acquisition can be performed up to 40 kHz by writing to an SD card via an Arduino board. The system was experimentally calibrated and mechanically tested up to 3000 rpm with no data loss [17]. This paper has shown that a low-cost system can be developed to measure torque between internal combustion engines and AC motors.

In the study presented by Sutyasadi, it was aimed to develop an effective control algorithm using a low-cost controller such as Arduino for the control of an aluminum robot arm that can be used in education due to the high cost of industrial robot arms [18]. In this study, computational torque control, PID, and cascade PID control were used to control the shoulder joint of the robot arm. According to the results obtained, computational torque control showed better results than PID and cascade PID control algorithms.

In the reviewed studies, measurement systems are designed according to the area of use and purpose. In this study, we focus on the development of a low-cost system to measure the torque of asynchronous electric motors at start-up.

3. METHODOLOGY

Indirect torque measurement methods can be realized in different ways depending on the specific situation and the available means. One is by measuring the angular displacement or rotational speed of the shaft. This information is then used to calculate the torque from the moment of inertia equation.

In the present work, a hybrid method is proposed. Since optical systems are affected by external light conditions and have low resolution, the angular displacement information is measured with a rotary encoder. The encoder generates 1024 pulses at one revolution. The encoder used is shown in Figure 1 (a).

Accordingly, the total angular displacement between two pulses is calculated as $2\pi/1024$ radians. The encoder has a 3-channel output, and accordingly, the pulse sequence from channels A, B, and Z can also provide information about the direction of rotation of the motor if necessary. Figure 1 (b) shows the relationship between the output information of the channels and the period.

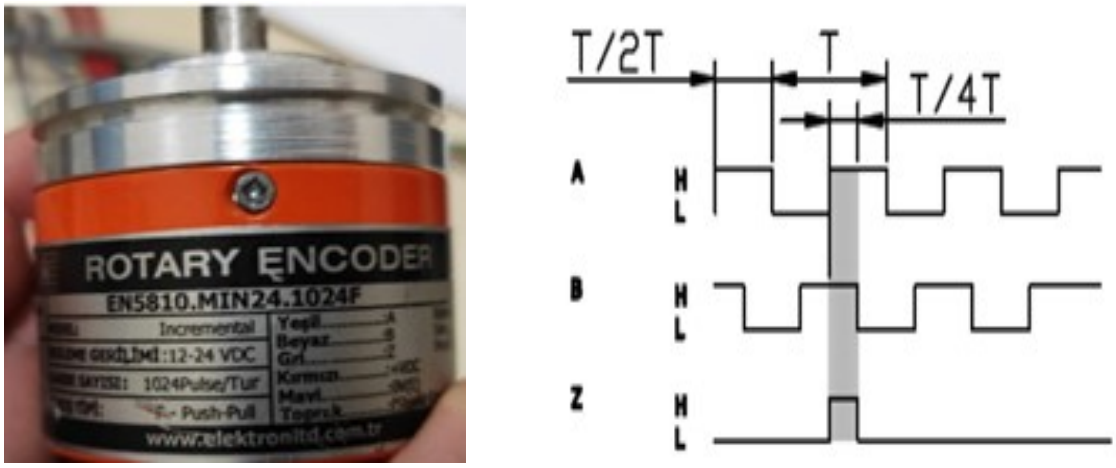

FIGURE 1. **(a). Used rotary encoder, (b) Output pulses and period relation for A, B and Z channels respectively.**

Angular velocity is defined as the change of angular displacement with respect to time and is defined in Eq. 1.

$$\omega = \frac{d\theta}{dt}. \tag{1}$$

The unit is radians/second, where ω (omega) represents the angular velocity. When the starting or stopping torque of an electric motor is to be calculated, the speed is variable. For this reason, the times of logic 1 and logic 0 outputs from channel A or B allow the motor speed to be calculated. Figure **??** shows the angular displacement and speed. In the SI unit system, the unit of angular path is radian and denoted by θ.

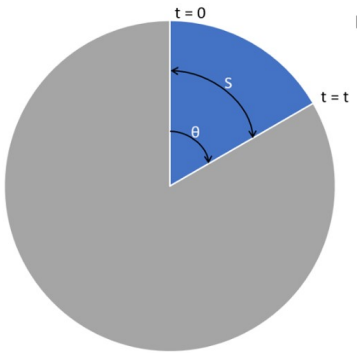

FIGURE 2. **Angular displacement.**

The relation between rotation and radian is given in Eq.2. 1 radian is defined as the central angle on a circle with an arc length equal to the radius, and Eq. 3 defines it.

$$\text{rotation} = 2\pi \text{ radian}, \tag{2}$$

$$\theta = \frac{S}{R}, \tag{3}$$

When the angular velocity is not constant, the rotational velocity variation depends on the rotational moment (torque) acting on the body in circular motion. The change of angular velocity with respect to time is defined as angular acceleration defined in Eq. 4 and denoted by α;

$$\alpha = \frac{d\omega}{dt}. \tag{4}$$

The relationship between angular velocity and torque is expressed by Eq. 5;

$$\tau = l * \alpha, \tag{5}$$

where τ is the torque, I is the moment of inertia and α is the angular acceleration. The moment of inertia measures a body's resistance to rotational motion. The moment of inertia is calculated depending on the geometry of the body and the position of the axis of rotation. For example, the moment of inertia of a cylinder or disk is calculated according to Eq. 6:

$$l = \frac{1}{2}mr^2, \tag{6}$$

where I is the moment of inertia, m is the mass of the cylinder and r is the radius. For bodies with more complex geometries, the moment of inertia is calculated using integration.

Considering the 16 MHz clock speed of the Arduino Uno, it is concluded that a processing cycle is completed in approximately 62.5 nanoseconds. However, considering that the response time of a digital input is completed in 3 processing cycles, a sampling frequency of approximately 5 kHz is obtained. In this case, the times of each logic 0 and logic 1 pulses at the encoder output can be detected fast enough. The flowchart of the algorithm for the measurement software of the designed system is given in Figure **??** in the appendix.

Then, from the sequential information received from the serial port, the period torque is calculated in the computer environment with the angular velocity, angular acceleration, and moment of inertia information obtained from the steady state of the system. The data taken from the serial port is saved as a CSV file and processed in Matlab. In addition to this, the data can be processed using Arduino. Since the data processing process is done after the measurements are completed, it will not pose any problems in terms of performance. This study setup is prepared for just measuring transient torque.

4. Results and Discussions

The connection diagram of the designed system is given in Figure 3. When the existing 1.1 kW asynchronous motor is driven from the line and loaded with a Foucault brake via coupling connection, the speed of the motor can be read optically via a 4-leaf encoder, and the torque generated can be read from the indicators on the control panel via the load cell. However, the measurement system on the control panel cannot perform the relevant measurements due to insufficient response speed in situations where the torque changes dynamically and very quickly, such as starting and stopping.

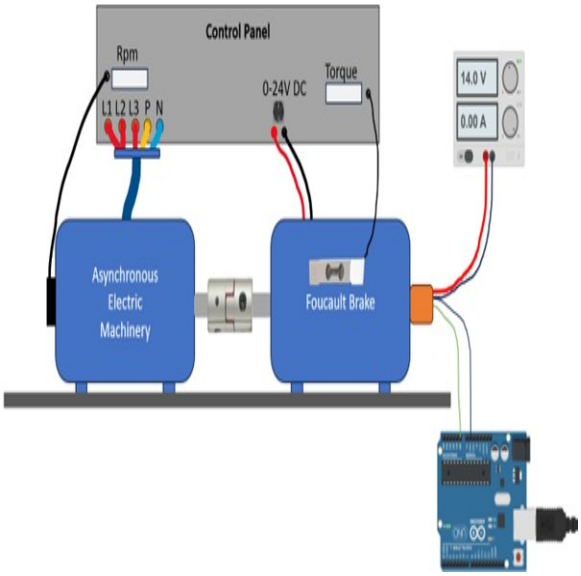

Figure 3. **Designed measurement system schema with real system.**

The bearing and coupling, whose design is shown in Figure 4, was produced on a 3D printer, and the 1024 pulse/revolution encoder shown in orange in Figure 2 was connected to the existing system. The encoder was fed externally with 14 V voltage, and the output was adapted to the 0-5V range with a voltage divider. It was tested separately on Arduino Uno and Mega boards. In cases where the system is required to react very fast, the measurements are saved in the microcontroller's volatile memory to minimize the error that may be caused by delays.

The moment of inertia of the existing system was determined experimentally. The steady-state torque of the system is measured and displayed on the panel via the load cell on the Focault brake. Therefore, the ratio of the torque measured on the panel to the acceleration measured by the 1024 pulse/rotation encoder gives the inertia value of the system. For this purpose, the angular path, velocity, and acceleration were calculated from the encoder, and the steady-state torque was found by proportioning it to the value on the control panel.

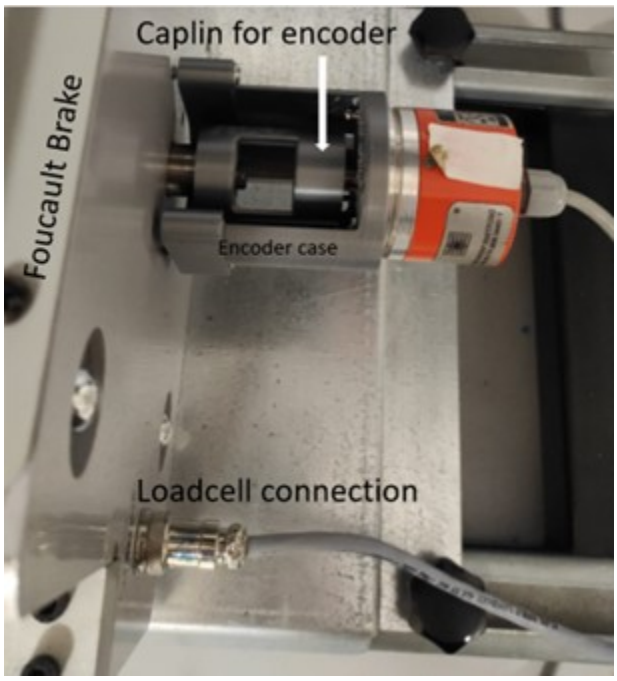

FIGURE 4. **Incremental encoder with foucault brake and connections.**

5. CONCLUSION

The moment of inertia is a measure of an object's resistance to changes in its rotational motion. It plays a crucial role in calculating torque in systems such as electric motors. Typically, it is determined experimentally due to the complexity of theoretical calculations, which may not account for all real-world factors.

Various methods for torque calculations are available in the literature, and a simple comparison of these methods is provided in Table 1.

With the system developed and successfully tested, the transient (start and stop) torque of rotary mechanism systems can be measured silently and successfully. With the proposed method, performance data of electric motors can be realized in a noiseless and numerical way that is cost-effective. The disadvantage of the system is that it requires calibration for measurements of systems with unknown moments of inertia.

The work in [6] has examined some of the situations that cause erroneous results in torque measurement systems. Accordingly, the use of data mining and machine learning methods to analyze the signals obtained in a steady state and to detect possible errors will be discussed in future studies.

TABLE 1. **Comparison of methods for determining moment of inertia**

Method	Advantages	Limitations
Pendulum Method	Simple setup, easy to conduct, good for basic shapes.	Accuracy depends on precise timing and knowledge of center of mass.
Rotary Motion Sensor Method	Precise measurements, suitable for complex shapes, direct data acquisition.	Requires sophisticated equipment like rotary sensors.
Torsional Oscillation Method	Good for symmetrical objects, directly uses torsional properties.	Requires knowledge of the spring constant, precise timing.
Angular Impulse and Momentum Method	Directly uses conservation of angular momentum, good for frictionless environments.	Requires a near-frictionless environment, sophisticated measuring tools.
Acceleration Method Using Known Masses	Flexible for different configurations, measures effect of added masses.	Needs precise measurement of angular acceleration, accurate force application.

DECLARATIONS

- **Conflict of Interest:** The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript.
- **Acknowledgment:** Special thanks to Karabük University Scientific Research Projects Department for supporting the KBÜBAP-21-ABP-115 project; Mehmet Akbaba, an Emeritus Professor at Karabük University, Turkiye, who provided the theoretical background of the entire study; Selim Öncü, a Professor at Karabük University, Turkiye, granted permission for laboratory usage and Ali Uysal, a Ph.D. from Manisa Celal Bayar University, Turkiye, provided additional hardware.

REFERENCES

[1] A. Keleşoğlu et al., "Elektrik motorlarında enerji sınıfları arası tüketim farklılıkları ve verimlilik artırıcı metotlar," Soma Meslek Yüksekokulu Teknik Bilimler Dergisi, vol. 1, no. 35, pp. 1–15, Jul. 2023, doi: 10.47118/somatbd.1238976.

[2] L. Xiao, J. Li, R. Qu, Y. Lu, R. Zhang, and D. Li, "Cogging torque analysis and minimization of axial flux PM machines with combined rectangle-shaped magnet," IEEE Trans Ind Appl, vol. 53, no. 2, pp. 1018–1027, Mar. 2017, doi: 10.1109/TIA.2016.2631522.
[3] B. Skala, "The torque measurement based on various principles", 2004.
[4] H. Işık, "Tork Sensöründe Kullanılan Teknolojiler" European Journal of Science and Technology Special Issue, vol. 28, pp. 622–626, 2021, doi: 10.31590/ejosat.1009173.
[5] T. Zhang, G. Li, R. Zhou, Q. Wang, and L. Wang, "Torque modeling of reluctance spherical motors using the virtual work method," International Journal of Applied Electromagnetics and Mechanics, vol. 71, pp. 199–219, 2023, doi: 10.3233/JAE-220104.
[6] J. Goszczak, "Torque measurement issues," IOP Conf Ser Mater Sci Eng, vol. 148, no. 1, p. 012041, Sep. 2016, doi: 10.1088/1757-899X/148/1/012041.
[7] A. Martyr and M. A. Plint, "Dynamometers: The Measurement of Torque, Speed, and Power," 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:111217151
[8] S. Tarakçı et al., "External torque sensor design providing wireless and real-time data customized for drivetrain", International Journal of Automotive Engineering and Technologies IJAET 11 (1) 18-27 ,2022 doi: 10.18245ijaet.982530.
[9] D. Kvashuk and O. Yashchuk, "Algorithm for determining the torque of electric motors using indirect measurement methods," Herald of Khmelnytskyi National University. Technical sciences, vol. 315, no. 6, pp. 138–146, Dec. 2022, doi: 10.31891/2307-5732-2022-315-6(2)-138-146.
[10] V. Kazakbaev, A. Paramonov, V. Dmitrievskii, V. Prakht, and V. Goman, "Indirect Efficiency Measurement Method for Line-Start Permanent Magnet Synchronous Motors," 2022, doi: 10.3390/math.
[11] J. Liu, L. Yang, and J. Ma, "The state-of-art and prospect of contactless torque measurement methods," in IOP Conference Series: Materials Science and Engineering, Institute of Physics Publishing, Jul. 2019. doi: 10.1088/1757-899X/542/1/012013.
[12] Y. Qin, Y. Zhao, Y. Li, Y. Zhao, and P. Wang, "A high performance torque sensor for milling based on a piezoresistive MEMS strain gauge," Sensors (Switzerland), vol. 16, no. 4, Apr. 2016, doi: 10.3390/s16040513.
[13] S. N. Ashwindran, A. A. Azizuddin, and A. N. Oumer, "A Low-Cost Digital Torquemeter Coordinated by Arduino Board," International Journal of Integrated Engineering, vol. 15, no. 1, pp. 118–130, 2023, doi: 10.30880/ijie.2023.15.01.011.
[14] V. Brusamarello, A. Balbinot, L. Carlos Gertz, and A. Cervieri, "Dynamic torque measurement for automotive application," in 2010 IEEE Instrumentation and Measurement Technology Conference Proceedings, IEEE, May 2010, pp. 1358–1362. doi: 10.1109/IMTC.2010.5488255.
[15] Ö. Bayraktar and A. Güldaş, "Quadrotor itme ve tork katsayılarının optimizasyonu ve matlab/simulink ile simülasyonu," Politeknik Dergisi, vol. 23, no. 4, pp. 1197–1204, Dec. 2020, doi: 10.2339/politeknik.636950.
[16] "Bottle Cap Opening Torque Tester," Sukron Makina. Accessed: Nov. 18, 2023. [Online]. Available: https://www.surkonmakina.com/sise-kapak-acma-tork-olcum-cihazi
[17] C. Caruana, P. Mollicone, and M. Farrugia, "Development of a Simple Instantaneous Torque Measurement System on a Rotating Shaft," in 2019 IEEE International Conference on Mechatronics (ICM), IEEE, Mar. 2019, pp. 382–388. doi: 10.1109/ICMECH.2019.8722854.
[18] P. Sutyasadi, "Control Improvement of Low-Cost Cast Aluminium Robotic Arm Using Arduino Based Computed Torque Control," Jurnal Ilmiah Teknik Elektro Komputer dan Informatika, vol. 8, no. 4, p. 650, Dec. 2022, doi: 10.26555/jiteki.v8i4.24646.

APPENDIX

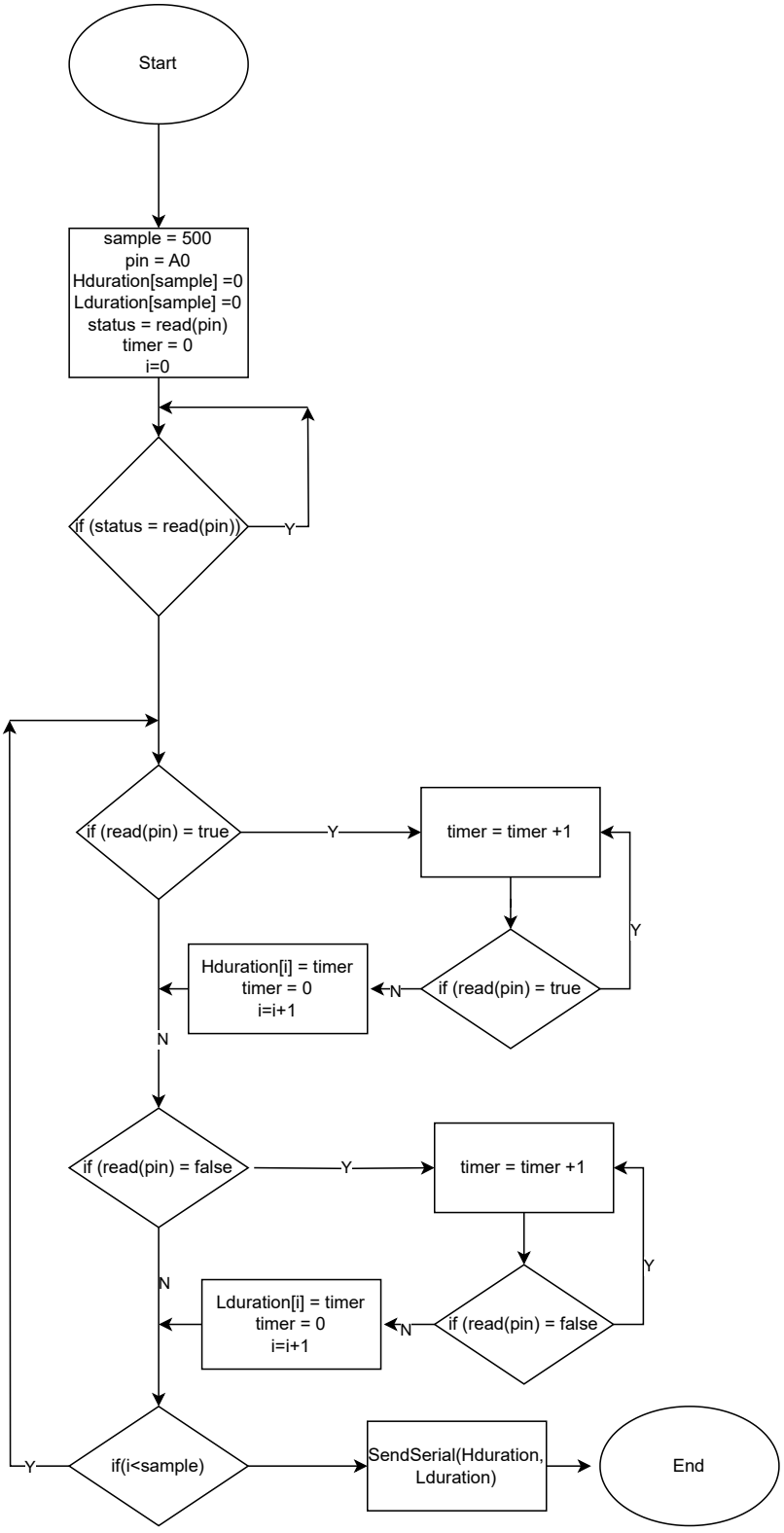

FIGURE 5. Flowchart for algorithm.

Vol: 2, Issue: 1, 2024
ISSN: 2980-3152
Pages:23-35

CURRENT TRENDS IN COMPUTING
https://dergipark.org.tr/en/pub/ctc
RESEARCH ARTICLE

SOLVING STATIC WEAPON-TARGET ASSIGNMENT PROBLEM USING MULTI-START LATE ACCEPTANCE HILL CLIMBING

SELIN ALPARSLAN[1], EMRULLAH SONUÇ[1*]

[1] *Department of Computer Engineering, Karabük University, 78050, Karabük, Türkiye*

ABSTRACT. **A challenging methodology predicted in modern military strategies is the unprotected Weapon-Target Assignment (WTA) problem, where weapons under consideration must be assigned to targets in order to minimize the expected survivability attribute against the targets. In this case, this study is interested in the static WTA (SWTA) scenario, where the assignments are made on a one-time basis. Since the SWTA problem has been found to be of NP-complete nature, the more accurate solution techniques can be considered infeasible due to the escalating complexity. In this paper, it is proposed to extend the library of new methods by implementing the multi-start method and the technique called Late Acceptance Hill Climbing (LAHC). Performance comparisons between the Multi-Start Late Acceptance Hill Climbing (MLAHC) and LAHC algorithms, derived from different examples and problem sizes, prove that the MLAHC algorithm yields better quality solutions and higher reliability than the traditional LAHC algorithm for large problems. This strategy can be seen as a revolution in the process of analyzing military resource allocation towards the optimal level.**

1. INTRODUCTION

The Weapon-Target Assignment (WTA) problem is a complex optimization task that is aimed at allocating weapons to targets with the intended objective of either realizing the maximum anticipated damage of or minimizing the expected survival probability of targets. This problem exists in two main forms: static and dynamic. In the static form, weapons are allocated to the targets once way, and both the weapons and targets remain fixed for the duration of their assignment. On the other hand, the dynamic version allows modification of the assignments over some period of time and may allow many assignments [1]. Nonetheless, the Static WTA (SWTA) frameworkfocuses on minimizing optimization where the goal is to attain the most appropriate weapon to target allocation to deter the enemy's projected impact. This approach is based on the assessment of the organisational defensive environment and focuses on the best ways and means of applying the defensive resources available.

The assessment of expected damage for defense assets is carried out after their involvement in a battlefield scenario. A problem for a defensive mission in SWTA problem can generally be formulated as follows [2]:

E-mail address: selinalparslann@gmail.com (S.Alparslan), esonuc@karabuk.edu.tr(*) (E.Sonuç).
Key words and phrases. Combitonarial Optimization, Late Acceptance Hill Climbing, Weapon-Target Assignment Problem.

$$
\begin{aligned}
f(\pi) &= \min \sum_{i=1}^{n} V_i \prod (1 - p_{ij})^{x_{ij}} \\
s.t. \sum_{i=1}^{n} x_{ij} &= w_i, \quad \text{for } i = 1, \cdots, m \\
x_{ij} &\in \mathbb{Z}_+, \quad \text{for } i = 1, \cdots, m, \quad j = 1, \cdots, n.
\end{aligned} \tag{1}
$$

There are m types of weapons (w_i represented by $i = 1, ..., m$) available to counter n targets, represented by $j = 1, ..., n$. Each weapon type i is associated with a probability p_{ij} of eliminating target j, while each target j has a destruction value denoted by V_j. The decision variables x_{ij} signify the quantity of weapons of type i allocated to target j.

According to [3], SWTA problem is an NP-Complete. Like other assignment problems, e.g. the quadratic assignment problem [4], it is inherently difficult. In the context of SWTA problem, there are n^m potential permutations for allocating m weapons to n targets. The condition is that all weapons must be allocated. As the number of weapons and targets increases, this process becomes increasingly complex. It can be challenging to explore all possible solutions due to the exponential growth of the problem. Exact solution methods are insufficient for solving the SWTA problem due to its computational complexity. Therefore, metaheuristics, known for their efficiency and efficacy in discover the solution space to address complex problems, are preferred for yielding practical and often nearly optimal solutions.

This study proposes an improved methodology for solving the SWTA problem. It suggests combining a strategy that involves multiple starting points with late acceptance hill climbing. This approach has been shown to be an alternative option for obtaining quality solutions within reasonable computational timeframes. The multistart approach increases diversity in the search space, while the hill climbing approach focuses on exploiting local optima. The study is significant because it addresses a crucial obstacle in military mission planning: the optimal allocation of weapons to targets. This allocation is essential for operational success. The research is a significant advancement in the field because it highlights the importance of assigning weapons to targets to achieve operational success. The rest of the paper is structured as follows: Section 2 presents the state-of-the-art methods for solving the WTA problem. Section 3 describes the late acceptance hill-climbing algorithm and the proposed approach with its components. Section 4 reports the experimental results and the last section concludes the study and suggests directions for future work.

2. Related Work

In recent years, a large number of exact and approximate algorithms for solving the WTA problem have been studied [5–7]. Due to its complexity, the WTA problem may be too hard for exact algorithms to solve. However, metaheuristics help us to overcome this problem by producing good solutions in a reasonable time. Metaheuristics, which combine several algorithms, are algorithms designed to solve

more complex optimization problems and can be applied to different optimization problems. Some metaheuristic algorithms, especially preferred for problems with a large number of solutions, can outperform exact methods and provide an optimal or near optimal solution in a reasonable time [8].

Several approaches to WTA have been studied in the literature. These include genetic algorithms, heuristic methods, and optimization techniques [9]. Exact algorithms based on mathematical programming have computational requirements that grow exponentially with the size of the problem [2]. Therefore, these algorithms are limited by some constraints. Recent research has been directed to dynamic situations and heuristic algorithms [10, 11]. In military operations, the efficient solution of the WTA problem is crucial. However, the complexity of the problem makes real-time optimal solutions impossible. Researchers are therefore working on heuristic algorithms such as genetic algorithms, simulated annealing, ant colony optimization, particle swarm optimization [12].

A branch-and-bound algorithm that combines lower bound methods with a search algorithm is proposed to solve the WTA problem. A combination of exact and heuristic algorithms for solving the WTA problem is presented, providing new methods and approaches for solving WTA in defense-related applications. Computational results are presented that demonstrate the ability to solve moderately large instances optimally and to obtain near-optimal solutions for fairly large instances within a few seconds. The ability to obtain optimal solutions for large instances in a short time is a significant achievement. [13]. A new exact algorithm for solving the WTA problem is presented. The algorithm incorporates new methods called weapon number limitation and weapon dominance to reduce the number of columns to be enumerated. The use of stage-dependent probabilities in WTA problems is proposed to optimize the allocation of weapons between different stages and targets. [14].

For the static version of the WTA problem, three approaches from the literature are presented to linearize the problem and transform it into linear optimization problems with complex numbers. The first approach can only be used as an approximation, the second approach fully linearizes the objective function of the WTA problem but is inferior to the solution time of the assignment problem, and the third approach exactly linearizes the objective function of the WTA problem. A special exact algorithm is proposed that avoids the difficulty of large dimensions. When a larger number of weapons are available for each weapon type, the optimization problems become intractable [15]. The modified Crow Search Algorithm (CSA) presents a new approach with a trial mechanism to improve the solution quality in solving WTA. The results show that the modified CSA performs better than the basic CSA and other state-of-the-art algorithms in most problem instances [16]. Another study has improved the previously proposed multi-objective evolutionary optimization algorithm by introducing an innovative approach. The proposed method consists of a Deep Q-Network (DQN) based mutation operator and a greedy-based matching operator.Experimental results show that the DQN-based mutation operator is successful in effectively identifying promising candidate solutions [17].

The WTA problem plays a central role in the improvement of military strategies and security mechanisms, and is characterized by its complicated nature stemming from the imperative requirement of optimal and competent resource allocation. Recent scientific work has witnessed a growing fascination

with metaheuristic methods as a means to address the challenges posed by the WTA problem, as discussed in Kline's study [2]. Metaheuristics are preferred because they provide flexibility and efficiency in solving large and complex problems.

3. The Proposed Method

3.1. **Late Acceptance Hill Climbing:**

The Late Acceptance Hill Climbing (LAHC) Algorithm is a metaheuristic approach designed to address combinatorial optimization problems [18]. It evaluates recent solution history to decide whether to accept a new solution, treating each new solution as an improved version of the current one. The LAHC algorithm has proven effective in various domains, including the traveling salesman problem, scheduling, and timetabling problems. The late acceptance strategy is straightforward. The control parameter for the acceptance condition is derived from the search history. This heuristic resembles Hill Climbing but with a key difference: in Hill Climbing, a candidate solution is compared to the current solution, whereas in LAHC, a candidate solution is compared to a solution from several iterations in the past. LAHC follows an acceptance rule by maintaining a fixed-length list, L_h, which represents the history length and contains previous values of the current cost function. To determine whether to accept a candidate solution, the candidate cost is compared to the final element in the list. If the candidate cost is better, it is accepted. Upon acceptance, the list is updated by inserting the new current cost at the beginning and removing the last element from the end. This process ensures that the added current cost consistently reflects the present cost. The pseudocode of LAHC is outlined in Algorithm 1.

LAHC has a wide range of applications in various domains. Its primary application has been in course scheduling, where it optimizes the quality of schedules by significantly reducing the final solution value, demonstrating an ability to effectively handle complex scheduling constraints [18]. LAHC has been used to solve the unrelated parallel machine scheduling problem [19], the general lot sizing and scheduling problem with rich constraints [20], and the traveling salesman problem [21]. Furthermore, LAHC has been applied in the context of drone trajectory planning algorithms, where it demonstrates superior performance compared to conventional approaches by incorporating local search operators to improve the efficiency of path determination [22]. In addition, the use of LAHC has been integrated into the feature selection process, thereby enhancing the ability to utilize metaheuristic algorithms to reduce dimensionality in machine learning tasks [23]. These examples highlight the adaptability and effectiveness of LAHC in tackling complex optimization and trajectory planning problems.

3.2. **Multi-Start Late Acceptance Hill Climbing:**

The Multistart Late Acceptance Hill Climbing Algorithm (MLAHC) is one of the most advanced optimization techniques which is an enhancement of the existing LAHC as it not only considers single start points of the search space but also in using multiple starts points in the exploration domain. While, in LAHC, new solutions are accepted after a specific time-interval based on their fitness value, and thus, navigate away from local optima, MLAHC enhances this by beginning the search process with different random initial solutions. This, in turn, enhances the prospects of visiting different regions of the exploration space in pursuit of the near-optimum solutions through what has been referred to as the

Algorithm 1 The pseudocode of LAHC.

Input: maxIterations, L (length of history list), initialSolution
Output: bestSolution
Initialisation:
currentSolution $\leftarrow$ initialSolution
bestSolution $\leftarrow$ currentSolution
currentValue $\leftarrow$ Evaluate(currentSolution)
historyList $\leftarrow$ Array of size L initialized with currentValue
Loop for a fixed number of iterations:
for $i \leq maxIterations$ **do**
    neighborSolution $\leftarrow$ GenerateNeighbor(currentSolution)
    neighborValue $\leftarrow$ Evaluate(neighborSolution)
    if neighborValue $\leq$ currentValue or neighborValue $\leq$ historyList[i % L] **then**
        currentSolution $\leftarrow$ neighborSolution
        currentValue $\leftarrow$ neighborValue
    end if
    if currentValue $\leq$ Evaluate(bestSolution) **then**
        bestSolution $\leftarrow$ currentSolution
    end if
    historyList[i % L] $\leftarrow$ currentValue
end for
return bestSolution

multi-start strategy. In other words, MLAHC is different from the basic LAHC in that it have multiple initial solutions as opposed to LAHC's single-start nature, meaning that the exploration of the search space is going to be better and wider with multiple LAHC.

The MLAHC begins by initializing the number of iterations, acceptance period, and restarts, followed by generating an initial solution and setting up the acceptance history. The algorithm then enters the multistart loop, where at each restart it sets the initial solution as the current solution and resets the acceptance history. Within each restart, the iteration loop generates neighboring solutions, calculates their costs, and compares these costs to those in the acceptance history. If a neighboring solution's cost is less than or meets the acceptance criteria, it becomes the new current solution and the acceptance history is updated accordingly. The algorithm tracks the best solution from each restart and updates the global best solution when a superior solution is found. This process continues until all restarts and iterations are complete, ultimately returning the global best solution as the optimal solution found by the algorithm. By using multiple starting points, MLAHC enhances its ability to explore the search space more extensively than traditional LAHC. The flowchart of the MLAHC is shown in Figure 1.

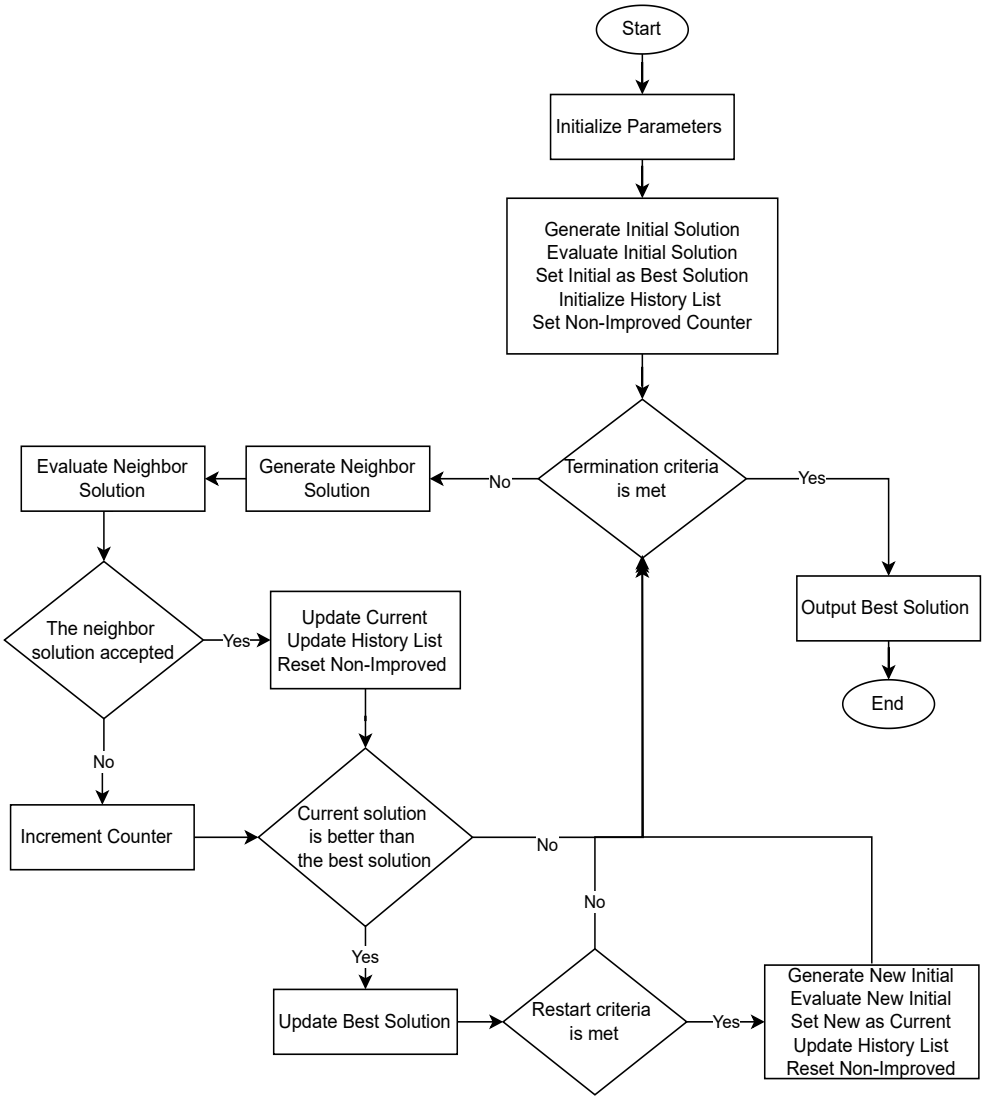

FIGURE 1. The flowchart of MLAHC.

4. EXPERIMENTAL RESULTS

MLAHC is tested on 12 WTA problem instances [7]. The results are given in different metrics: best, mean, median, worst and standard deviation (SD). Sizes of problem instances are in the range 5 and 200 and shown in Table 1. Results were collected from 10 independent runs. The numerical experiments were performed on a PC with 8.00 GB of RAM, MacOS 14.4.1 operating system. The MLAHC codes were written in the C programming language using CLion IDE v2023.3.4.

TABLE 1. WTA problem instances.

Instance	Number of Weapons	Number of Targets
WTA1	5	5
WTA2	10	10
WTA3	20	20
WTA4	30	30
WTA5	40	40
WTA6	50	50
WTA7	60	60
WTA8	70	70
WTA9	80	80
WTA10	90	90
WTA11	100	100
WTA12	200	200

Table 2 presents the results on small-scale WTA instances. The results of the MLAHC show that for WTA1, WTA2, and WTA3, the algorithm consistently achieves identical best, worst, mean, and median objective values, with an SD of 0.0000 across all configurations, indicating highly stable performance. In contrast, WTA4 shows more variability, especially with a history length of 1 and no restarts, resulting in higher and more variable target values and an SD of 6.4358. This means that there is a direct relation between the number of identified parameter configurations and the overall performance of the algorithm, notably when it is faced with more complex instances. Although the algorithm shows consistent performance on small instances (WTA1, WTA2, and WTA3) regardless of historical length and restart values, it requires careful parameter tuning when faced with more complex instances such as WTA4. In particular, adopting larger historical lengths and using restarts as enhancements could be seen as effective on the grounds of stability and optimality to these particular cases.

Table 3 presents the results on medium-scale WTA instances. For the WTA5 instance , the best, worst, mean, and median objective values show some variation across different history lengths and restart values, indicating that the algorithm's performance is somewhat sensitive to these parameters. The lowest mean objective value is 306.8923 for a history length of 1000 and a restart value of 500. For the WTA6 instance, the results also vary across configurations. The lowest mean objective value is 355.6795 for a history length of 1000 and a restart value of 1000. For the WTA7 instance , there is variability in the results, with the lowest mean objective value of 419.5174 achieved with a history length of 1000 and restart value of 1000. For the WTA8 instance, the results show significant variability, especially with a history length of 1 and no restart, leading to much higher objective values and standard deviation. The lowest mean objective value is 502.9574 with a history length of 1000 and a restart value of 500. As a result, for instances WTA5, WTA6, and WTA7, MLAHC consistently achieves better and more stable results with larger history lengths and higher restart values, suggesting that these configurations help the algorithm explore the solution space more effectively. For the WTA8 instance, the variability in results is more pronounced, especially with shorter history lengths and no restarts, resulting in higher objective values

TABLE 2. Experimental results on small-scale problem instances.

Instance	History Length	Restart	Best	Worst	Mean	Median	SD
WTA1	1000	1000	48.3640	48.3640	**48.3640**	48.3640	0.0000
	500	500	48.3640	48.3640	**48.3640**	48.3640	0.0000
	500	1000	48.3640	48.3640	**48.3640**	48.3640	0.0000
	1000	500	48.3640	48.3640	**48.3640**	48.3640	0.0000
	500	-	48.3640	48.3640	**48.3640**	48.3640	0.0000
	1000	-	48.3640	48.3640	**48.3640**	48.3640	0.0000
	1	-	48.3640	48.3640	**48.3640**	48.3640	0.0000
WTA2	1000	1000	96.3123	96.3123	**96.3123**	96.3123	0.0000
	500	500	96.3123	96.3123	**96.3123**	96.3123	0.0000
	500	1000	96.3123	96.3123	**96.3123**	96.3123	0.0000
	1000	500	96.3123	96.3123	**96.3123**	96.3123	0.0000
	500	-	96.3123	96.3123	**96.3123**	96.3123	0.0000
	1000	-	96.3123	96.3123	**96.3123**	96.3123	0.0000
	1	-	96.3123	96.3123	**96.3123**	96.3123	0.0000
WTA3	1000	1000	142.1070	142.1070	**142.1070**	142.1070	0.0000
	500	500	142.1070	142.1070	**142.1070**	142.1070	0.0000
	500	1000	142.1070	142.1070	**142.1070**	142.1070	0.0000
	1000	500	142.1070	142.1070	**142.1070**	142.1070	0.0000
	500	-	142.1070	150.2510	144.7579	144.0702	2.3100
	1000	-	142.1070	144.4690	143.3774	143.2416	0.7843
	1	-	164.5723	178.6062	173.3124	174.3449	4.5964
WTA4	1000	1000	248.0285	248.5817	**248.3479**	248.4051	0.1936
	500	500	248.2730	249.3956	248.5891	248.4222	0.3427
	500	1000	248.3312	249.0275	248.5460	248.4222	0.2646
	1000	500	248.0285	248.8386	248.3717	248.3476	0.2605
	500	-	249.9979	256.5385	253.2718	253.4581	2.2874
	1000	-	250.4865	257.0525	253.6127	253.7566	2.0373
	1	-	327.0574	346.8141	339.7976	340.0045	6.4358

and standard deviations. This suggests that for more complex or larger instances, a longer history length and the ability to restart the search process are critical to achieving optimal and consistent solutions.

Table 4 presents the results on medium-scale WTA instances. For the WTA9 instance, the best, worst, mean, and median objective values show some variation across different history lengths and restart values. The lowest mean objective value is 539.2292 with a history length of 1000 and a restart value of 1000. The SD values are relatively low for most configurations, indicating stable performance, except for configurations with shorter history lengths and no restarts. For the WTA10 instance, the results also vary, with the lowest mean objective value of 599.2728 achieved with a history length of 1000 and restart value of 1000. For the WTA11 instance, the results indicate variability, with the lowest mean objective value of 704.6850 with a history length of 1000 and restart value of 1000. For WTA10 and WTA11 instances, the SD values are also low for most configurations, indicating stable performance, but higher for shorter history lengths and no restarts. For the WTA12 instance, the variability in results is more pronounced, especially for a history length of 1 and no restart, leading to much higher objective values

TABLE 3. Experimental results on medium-scale problem instances.

Instance	History Length	Restart	Best	Worst	Mean	Median	SD
WTA5	1000	1000	306.5564	308.1392	307.2418	307.2133	0.4812
	500	500	306.2859	307.7959	306.9699	307.0481	0.4560
	500	1000	306.1562	308.5417	307.1399	306.9142	0.6765
	1000	500	306.0912	307.5720	**306.8923**	307.1220	0.5953
	500	-	308.0490	319.2780	313.5596	314.6806	3.8503
	1000	-	311.2144	320.5870	314.4265	313.8629	2.5515
	1	-	461.5889	488.0803	476.8728	477.8627	7.9504
WTA6	1000	1000	354.0916	356.8551	**355.6795**	355.6409	0.8638
	500	500	355.3909	356.7601	356.2280	356.3174	0.0536
	500	1000	355.1825	358.1991	356.6363	356.7981	0.8159
	1000	500	354.6224	356.2821	355.6847	355.7897	0.5000
	500	-	360.1330	370.9876	364.8432	364.1965	3.6817
	1000	-	357.4125	364.0606	360.0826	360.0549	1.9652
	1	-	545.3753	596.1006	577.8765	579.7290	14.9065
WTA7	1000	1000	418.5731	420.5899	**419.5174**	419.5085	0.5927
	500	500	417.1001	422.0817	419.6406	419.7593	0.0613
	500	1000	417.1754	421.1842	419.8538	420.4126	1.3062
	1000	500	417.4177	421.3109	419.7452	419.7255	1.1660
	500	-	425.9927	432.3662	428.6032	428.5293	2.0944
	1000	-	422.9075	435.2173	426.7079	426.1469	3.5601
	1	-	700.1578	732.6472	715.8937	713.3675	10.0763
WTA8	1000	1000	500.0615	504.5395	503.0515	503.4774	1.3633
	500	500	499.9063	505.1909	503.3437	503.7197	1.5449
	500	1000	501.4779	507.3864	504.4214	504.8288	2.0883
	1000	500	499.9062	504.4010	**502.9574**	503.6857	1.6086
	500	-	508.2395	519.8273	514.9432	515.8243	3.7444
	1000	-	504.3269	520.1396	511.7618	512.2577	5.7482
	1	-	864.0853	898.3761	884.8409	884.3876	10.5878

and standard deviation. The lowest mean objective value is 1,306.1270 with a history length of 1000 and a restart value of 1000. The SD is higher, especially for the history length of 1 and no restart, where the SD is 20.4028, indicating less stable performance and greater variability in results.

The performance of MLAHC on large instances is strongly influenced by the history length and restart parameters. For instances WTA9, WTA10, and WTA11, MLAHC consistently yields better and more stable results with larger history lengths and higher restart values. This suggests that these configurations help the algorithm explore the solution space more effectively. For the WTA12 instance, the variability in results is more significant, especially with shorter history lengths and no restarts, resulting in higher objective values and standard deviations. This suggests that for more complex or larger instances, using a longer history length and allowing the search process to restart are critical to achieving optimal and consistent solutions. Overall, adjusting these parameters is key to improving the performance of the algorithm. Longer history lengths and restarts are generally recommended for large instances to improve results.

TABLE 4. Experimental results on large-scale problem instances.

Instance	History Length	Restart	Best	Worst	Mean	Median	SD
WTA9	1000	1000	537.7873	541.1868	**539.2292**	539.2745	1.0036
	500	500	538.7035	15.2201	541.2645	541.5898	1.3981
	500	1000	539.4272	542.8226	541.1484	541.3882	1.3585
	1000	500	536.7075	541.1680	539.5932	539.9091	1.4354
	500	-	543.9957	554.6371	548.5667	548.8087	3.2537
	1000	-	540.7142	553.3719	545.6256	544.8100	3.8780
	1	-	935.5836	987.4912	969.9581	976.2804	16.8421
WTA10	1000	1000	598.0171	602.8778	599.8728	599.5441	1.4008
	500	500	597.1049	604.1977	601.0149	601.3492	0.0621
	500	1000	599.9423	602.5693	601.2927	601.2210	0.9187
	1000	500	597.2714	602.1404	**599.4435**	599.5426	1.4863
	500	-	606.5543	614.2934	610.2164	610.1721	2.5238
	1000	-	603.0792	615.5761	607.9658	607.2778	3.9888
	1	-	1,089.2605	1,134.0014	1,118.0351	1,123.5720	14.6743
WTA11	1000	1000	702.4334	706.9329	**704.6850**	704.7220	1.5023
	500	500	705.8282	709.3710	707.8575	707.7180	0.0810
	500	1000	704.9791	708.7655	707.3339	707.3468	1.1818
	1000	500	702.8853	707.9533	705.0482	704.8346	1.5613
	500	-	712.9811	721.9246	716.4566	715.2716	3.3886
	1000	-	704.3536	720.3480	712.7618	713.0143	5.0992
	1	-	1,297.7713	1,338.8655	1,325.7621	1,329.8928	13.7453
WTA12	1000	1000	1,304.0334	1,308.1425	1,306.1527	1,306.2038	1.2910
	500	500	1,306.3751	1,311.0284	1,308.4978	1,308.6416	0.1910
	500	1000	1,305.1196	1,310.4764	1,308.7313	1,309.3100	1.7343
	1000	500	1,300.8688	1,309.1603	**1,304.7562**	1,304.9484	2.4993
	500	-	1,311.4409	1,324.3238	1,319.0374	1,321.3748	4.6725
	1000	-	1,307.7223	1,318.4404	1,312.3279	1,312.8594	3.3418
	1	-	2,664.1240	2,727.3161	2,696.9456	2,698.5258	20.4028

5. DISCUSSION

WTA problem experimentation under small, medium, and large problem instances indicate that the effectiveness of MLAHC depends on the history length and restart parameters. In regard to small scenarios, the specific algorithm seems to be highly stable in terms of yielding near-optimal solutions and is almost inert to these settings. However, these parameters when complex, require adjustments that are more relevant, with the size of the problem advancing. In medium and large scenarios, the use of larger history length and inclusion of restarts, generally enhance and stabilize the performance by attaining better lower objective values with less variability. In the most difficult problem instances, a history length of 1000 combined with frequent restarts consistently produces the best results. Thus, for more complicated and extensive tasks, it regimens stable and longer histories, as well as organize restarts when using the MLAHC algorithm. It also helps give enhanced solution quality and reliability since the best possible solution is chosen from numerous different solutions.

A comprehensive evaluation over a range of problem sizes highlights the effectiveness and stability of the algorithm. However, the study's comparison is limited to traditional LAHC and lacks broader comparisons with other state-of-the-art algorithms. It focuses specifically on SWTA, which may limit generalizability, and does not investigate scalability or computational requirements for extremely large or real-time applications. The performance of the MLAHC algorithm is highly dependent on the history length and restart parameters, which require careful tuning, especially for larger and more complex problem instances. While the algorithm shows consistent performance on small instances, it requires more precise parameter settings to achieve efficient solutions as the problem size increases. In addition, the study focuses primarily on the static version of the WTA problem, and although it suggests potential applications in dynamic WTA problems, these areas are not explored in this paper. Despite these limitations, the practical relevance of the study to military resource allocation and the novel approach presented are significant contributions.

6. Conclusion

This paper aims to develop a new heuristic approach for solving the Static Weapon-Target Assignment (SWTA) problem incorporating the multistart strategy and Late Acceptance Hill Climbing. This new technique called Multistart Late Acceptance Hill Climbing (MLAHC) enhances the search mechanism coupled with optimization in the local optima, and they deliver the best quality solutions with high performance. As one can observe from tests on various scenarios of WTA problems, it can been seen that the MLAHC approach performs well. The simulation results indicate that this technique is also more efficient than conventional versions of LAHC and this becomes more evident when applied to larger and complicated problems.

This research also found that except for the history length and restart parameters, MLAHC has significantly high dependency on these two factors. For the small levels, MLAHC holds an ideal and constant performance in all environment settings. However, as problem size increase, it becomes paramount to tweak these parameters within the system. History length is longer and it has more restarts which prove that it provides better and improved results and it underlines the point that there should be much proper setting required to get the efficient solutions in the complex problems.

Thus, the MLAHC algorithm gives a strong and versatile method to the SWTA problem, which will significantly adds its value to the scopes of computational combinatorial optimization in general and the military operations study in particular. If this algorithm is applied in the dynamic WTA problem and other optimization problems in defence and other fields, then future research can be on these areas. Further enhancement of this algorithm can be done by combining several metaheuristic algorithms and hybridization of the strategies.

Data Statement WTA problem instances are available at `https://doi.org/10.17632/jt2ppwr62p.1`
Conflict of Interest The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript.

REFERENCES

[1] R. K. Ahuja, A. Kumar, K. C. Jha, J. B. Orlin, Exact and heuristic algorithms for the weapon-target assignment problem, Operations research 55 (6) (2007) 1136–1146.

[2] A. Kline, D. Ahner, R. Hill, The weapon-target assignment problem, Computers & Operations Research 105 (2019) 226–236.

[3] S. P. Lloyd, H. S. Witsenhausen, Weapons allocation is np-complete., in: 1986 summer computer simulation conference, 1986, pp. 1054–1058.

[4] E. Sonuc, B. Sen, S. Bayir, A cooperative gpu-based parallel multistart simulated annealing algorithm for quadratic assignment problem, Engineering Science and Technology, an International Journal 21 (5) (2018) 843–849.

[5] Ö. Tolga, E. BOZKAYA, An evaluation on weapon target assignment problem, Journal of Naval Sciences and Engineering 18 (2) (2022) 305–332.

[6] H. Xing, Q. Xing, An air defense weapon target assignment method based on multi-objective artificial bee colony algorithm., Computers, Materials & Continua 76 (3) (2023).

[7] E. Sonuc, B. Sen, S. Bayir, A parallel simulated annealing algorithm for weapon-target assignment problem, International Journal of Advanced Computer Science and Applications 8 (4) (2017) 87–92.

[8] B. Chopard, M. Tomassini, An introduction to metaheuristics for optimization, Springer, 2018.

[9] A. Toet, H. de Waard, The Weapon-Target Assignment Problem, Citeseer, 1995.

[10] C. Wang, G. Fu, D. Zhang, H. Wang, J. Zhao, et al., Genetic algorithm-based variable value control method for solving the ground target attacking weapon-target allocation problem, Mathematical Problems in Engineering 2019 (2019).

[11] D. Guo, Z. Liang, P. Jiang, X. Dong, Q. Li, Z. Ren, Weapon-target assignment for multi-to-multi interception with grouping constraint, IEEE Access 7 (2019) 34838–34849.

[12] M. D. Rezende, B. S. P. De Lima, S. Guimarães, A greedy ant colony system for defensive resource assignment problems, Applied Artificial Intelligence 32 (2) (2018) 138–152.

[13] Exact and heuristic algorithms for the weapon-target assignment problem (2007). doi:10.1287/OPRE.1070.0440.

[14] A new exact algorithm for the weapon-target assignment problem (2021). doi:10.1016/J.OMEGA.2019.102138.

[15] A. C. Andersen, K. Pavlikov, T. A. Toffolo, Weapon-target assignment problem: Exact and approximate solution algorithms, Annals of Operations Research 312 (2) (2022) 581–606.

[16] E. Sonuç, A modified crow search algorithm for the weapon-target assignment problem, An International Journal of Optimization and Control: Theories & Applications (IJOCTA) 10 (2) (2020) 188–197.

[17] S. Zou, X. Shi, S. Song, Moea with adaptive operator based on reinforcement learning for weapon target assignment, Electronic Research Archive 32 (3) (2024) 1498–1532.

[18] E. K. Burke, Y. Bykov, The late acceptance hill-climbing heuristic, European Journal of Operational Research 258 (1) (2017) 70–78.

[19] M. Terzi, T. Arbaoui, F. Yalaoui, K. Benatchba, Solving the unrelated parallel machine scheduling problem with setups using late acceptance hill climbing, in: Asian Conference on Intelligent Information and Database Systems, Springer, 2020, pp. 249–258.

[20] A. Goerler, E. Lalla-Ruiz, S. Voß, Late acceptance hill-climbing matheuristic for the general lot sizing and scheduling problem with rich constraints, Algorithms 13 (6) (2020) 138.

[21] S. Clay, L. Mousin, N. Veerapen, L. Jourdan, Clahc-custom late acceptance hill climbing: First results on tsp, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2021, pp. 1970–1973.

[22] B. S. Shihab, H. N. Abdullah, L. A. Hassnawi, Improved artificial bee colony algorithm-based path planning of unmanned aerial vehicle using late acceptance hill climbing., International Journal of Intelligent Engineering & Systems 15 (6) (2022).

[23] B. Chatterjee, T. Bhattacharyya, K. K. Ghosh, P. K. Singh, Z. W. Geem, R. Sarkar, Late acceptance hill climbing based social ski driver algorithm for feature selection, IEEE Access 8 (2020) 75393–75408.

Vol: 3, Issue: 1, 2024
ISSN: 2980-3152
Pages:36-47

CURRENT TRENDS IN COMPUTING
https://dergipark.org.tr/en/pub/ctc
RESEARCH ARTICLE

EFFECTS OF CHEMICAL AUTAPSE ON INVERSE CHAOTIC RESONANCE IN MORRIS-LECAR NEURON MODEL

ALİ AKÇAY [1], ERGİN YILMAZ [1*]

[1] *Biomedical Engineering Department, Zonguldak Bulent Ecevit University, 67100, Zonguldak, Türkiye*

ABSTRACT. **Inverse chaotic resonance is a phenomenon, in which the mean firing rate of a neuron exhibits a minimum depending on the chaotic signal intensity, which emerges in the firing dynamics of neurons. In this study, we have investigated the effects of inhibitory and excitatory autapses on the inverse chaotic resonance phenomenon in Morris-Lecar (ML) neurons. We show that, for proper constant stimulus current, the ML neurons exhibits inverse chaotic resonance phenomenon in the firing dynamics as a function of the intensity of the chaotic activity. In addition, we find that, at low and medium chaotic activity levels, the ML neuron shows multiple-inverse chaotic resonance phenomenon depending on autaptic time delay for low and intermediate autaptic conductances. Finally, we show that, both excitatory and inhibitory autapse augment the firing rate of the ML neuron, this increase is more in the case of excitatory autapse.**

1. INTRODUCTION

Neurons serve as the building blocks of the nervous system and play a fundamental role in information processing and transmission. The electrical activity occurring in neurons forms the basis for the processing and encoding of information [1–4]. Researchers have proposed various biophysical models to explain information processing and electrical activity in neurons [5–8]. In the literature, it is observed in experimental and theoretical studies that the electrical activity and membrane dynamics in neurons are not deterministic, meaning that neurons do not always generate firing in the same magnitude and timing in response to the same stimulus [9–11].The influence of various environmental factors on the processing of information in the nervous system has been extensively documented in the literature. For instance, the phenomenon of stochastic resonance (SR) allows for the optimization of the detection and transmission of weak signals in neurons under certain conditions [12–18]. Gutkin et al. observed in their studies that at a constant critical current value applied to neurons, the firing rate approaches zero within an optimal noise range and then increases again for increasing noise levels [19].This situation represents a phenomenon that is the opposite of stochastic resonance. This phenomenon, widely investigated in the literature, is known as ”inverse stochastic resonance” (ISR) [20, 21]. In their work, Yu et al. reported the presence of inverse chaotic resonance (ICR), a phenomenon similar to inverse stochastic resonance,

E-mail address: erginyilmaz@yahoo.com (*).
Key words and phrases. Morris-Lecar neuron, inverse chaotic resonance, autapse.

emerges under the influence of chaotic activity in the average firing rate of neurons. This effect manifests itself as a kind of squelch in the average firing rate at a given chaotic signal intensity [22].

Neurons communicate through specialized connections called synapses. In the nervous system, there are two main types of synapses: electrical synapses and chemical synapses. Electrical synapses are synapses where communication between neurons occurs directly through electrical signals. In this type of synapse, nerve impulses are transmitted electrically along the cell of a neuron. On the other hand, chemical synapses are synapses where communication between neurons occurs through chemical signals. In this type of synapse, an excitatory signal from one neuron is transmitted to the receptors of another neuron through chemical signals called neurotransmitters. However, it has been documented in the literature that some neurons form synaptic connections between their own soma and dendrites, creating a feedback structure. This unusual biological structure was first reported by Van der Loos and Glaser and named autapse [23]. In studies using different experimental techniques, the presence of autapses in brain regions has been observed [24–26].

Recently, in addition to the existence of autapse, the effects of autapse on neuron behavior have been extensively examined in numerical studies [27–29]. By inter spike interval histogram analysis, Li et al. showed that electrical autapse reduces the number of spontaneous firings in stochastic Hodgkin-Huxley (H-H) neurons [30]. Qin et al reported that autapse triggers spiral wave formation in an organized network of Hindmarsh-Rose (HR) neurons [31]. Wang at al. demonstrated that autapse provides a transition between silence (no firing state) and periodic and chaotic behavioral patterns in the electrical activity of the Hodgkin-Huxley (H-H) neuron [32]. Baysal at al. examined the effects of chemical autapse on weak signal transmission in scale-free networks and revealed that autapse blocks the weak signal transmission at appropriate parameter values [29]. Yilmaz et al., assuming that only the pacemaker neuron in the small world network has an autapse, demonstrated that the transmission of the local activity of the pacemaker neuron through the network increases significantly at appropriate autapse parameters [33].

In the literature, although the inverse chaotic resonance phenomenon has been studied in single neurons and in neuronal network, the effects of autapse on the this phenomenon are not investigated neither in single neurons nor in complex neuronal networks. To address this gap in the related research topic, in the current paper, we have analyzed the effects of chemical autapse on the inverse chaotic resonance phenomenon in Moris-Lecar (ML) neurons via numerical simulations. Obtained results show that for proper autaptic parameter values, the chemical autapse regardless of it is excitatory or inhibitory induces M-ICR phenomenon at low and medium chaotic activity cases.

2. Materials and Methods

The membrane potential dynamics of the Morris-Lecar neuron, which has an autapse and is exposed to a chaotic signal, is given by the following equations [34, 35]:

$$C\frac{dV}{dt} = -g_{Ca}m_{\infty}(V)(V-E_{Ca})) - g_K w(V-E_K) - g_{leak}(V-E_L) + I_{app} + I_{chaos} + I_{aut} \tag{1}$$

$$\frac{dw}{dt} = \phi\frac{w_{\infty}(V)-w}{\tau_w(V)} \tag{2}$$

$$m_\infty(V) = 0.5[1 + \tanh((V - \beta_m)/\beta_w)] \tag{3}$$

$$w_\infty(V) = 0.5[1 + \tanh((V - \gamma_m)/\gamma_w)] \tag{4}$$

$$\tau_w(V) = 0.5[\cosh(\frac{V - \gamma_m}{2\gamma_w})]^{-1} \tag{5}$$

The meanings of the symbols used in these equations are given in Table 1:

TABLE 1. **Model parameters**

Parameter	**Value**
C: Membrane capacitance of the neuron	$20\mu F/cm^2$
V: Membrane potential	variable
w: Slow recovery variable	variable
g_{Ca}: Conductance of fast Ca++ current	$4.4\mu S/cm^2$
g_K: Conductance of slow K+ current	$8\mu S/cm^2$
g_{leak}: Conductance of leak current	$2\mu S/cm^2$
E_K: Potassium equilibrium potential	$-84mV$
E_L: Leak current equilibrium potential	$-60mV$
E_{Ca}: Sodium equilibrium potential	$120mV$
β_m: the activation midpoint potentials at which the corresponding currents are half activated	$-1.2mV$
β_w: slope factors of the activation	$18mV$
γ_m: the activation midpoint potentials at which the corresponding currents are half activated	$2mV$
γ_w : slope factors of the activation	$30mV$
ϕ : maximum rate constant for K+ channel opening	0.04
m_∞ : the fraction of open calcium channels at steady state	variable
w_∞ : the fraction of open potassium channels at steady state	variable
τ_w :time constant for the activation of potassium channels	variable

The I_{app} current given in Equation 1 represents the constant stimulation current injected externally. I_{aut} represents the autapse current arising from the chemical autapse of the ML neuron and is expressed by the equation given below.

$$I_{aut} = -\kappa(V(t) - V_{syn})S(t - \tau) \tag{6}$$

$$S(t - \tau) = 1/1 + \exp(-k(V(t - \tau) - \theta)$$

where $V(t)$ represents membrane potential of The ML neuron and κ is autaptic conductance,V_{syn} is reverse synaptic potential and τ is autaptic delay. When the reverse synaptic potential is V_{syn}=10mV, the autapse shows excitatory behavior, while when V_{syn}=-65mV, the autapse shows inhibitory behavior. The

other parameters's values are set k= 8 and θ= 0.25 [36]. The chaotic current I_{chaos} whose source is assumed to be the chaotic activity of peripheral neurons, is calculated as $I_{chaos} = \varepsilon.x$, here ε represents the chaotic current intensity and x is the chaotic signal source. In this study, the Lorenz system was used as the chaotic signal source. The equations representing this system are given below [37].

$$dx/dt = \sigma(y-x) \tag{7}$$

$$dy/dt = px - y - xz \tag{8}$$

$$dz/dt = xz - \lambda z \tag{9}$$

where chaotic system parameters are set as β = 8/3, σ = 10, p = 28. The initial values for the (x,y,z) variables are determined randomly. On the other hand, initial values of membrane potential variable V and slow recovery variable W were randomly determined uniformly between related intervals. After distracting 1 second transition time, the firing frequency is calculated during $\delta = 5s$ simulation time. 20mV is assumed as a firing threshold in the membrane potential of neuron in deciding whether a spike is present or not. It is accepted that, at each passing of membrane potential with positive slope from this threshold, a spike occurs. This procedure is repeated $N = 1000$ times and the average is taken to obtain the mean firing rate (MFR). Calculation of mean firing rate (MFR) is given below:

$$MFR = \frac{1}{N\delta}\left(\sum_{k=1}^{N} N_k^{spikes}\right) \tag{10}$$

where N_k^{spikes} is the total number of spikes produced by the neuron in kth realization.

3. Results

3.1. **Inverse Chaotic Resonance in Morris Lecar neuron model.** In this section, we investigate the effect of chaotic activity on the firing rhythms of single ML neuron. To do this, firstly, the time-dependent change of membrane potentials of ML neuron at different chaotic activity levels are given in Figure 3.1.

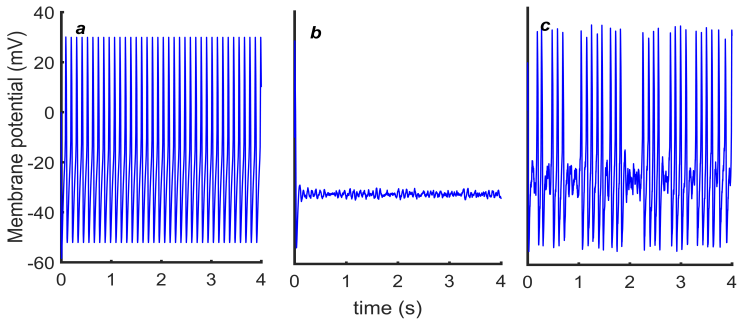

FIGURE 1. Time dependence of membrane potential of ML neuron for different chaotic signal intensities, a) $\varepsilon = 0.01$; b)$\varepsilon = 0.5$; c) $\varepsilon = 2$ ($I_{app} = 89\mu A/cm^2$).

It is seen in Figure that, at a low chaotic current intensity ($\varepsilon = 0.001$), the neuron fires continuously under the effect of constant stimulus current I_{app} and has a constant firing rate (Figure 3.1a). For the chaotic current intensities of moderate level, for example $\varepsilon = 0.5$ (Fig.1b), although the neuron initially emits one spike due to the initial conditions effect, then it emits no spike and remains silent due to the effects of the increasing chaotic activity. At further increase in the chaotic current intensity, as the neuron in Fig.1c in which $\varepsilon = 2$, the neuron stays silent first, and then, it emits spikes in some instants, or vice versa, first emits spikes, and then stays silent, and this pattern of activity, knowing burst type firing, emerges continuously in the electrical activity of the ML neuron. Eventually, the neuron has a constant, non-zero firing rate. From these results presented in Figure 3.1, it can deduced that the chaotic signals can induce inverse chaotic resonance phenomenon (ICR) in ML neurons.

To investigate whether the chaotic activity induces the ICR phenomenon, the average firing frequency of the ML neuron for different constant stimulation currents (I_{app}) is given as a function of the chaotic current intensity in Figure 2.

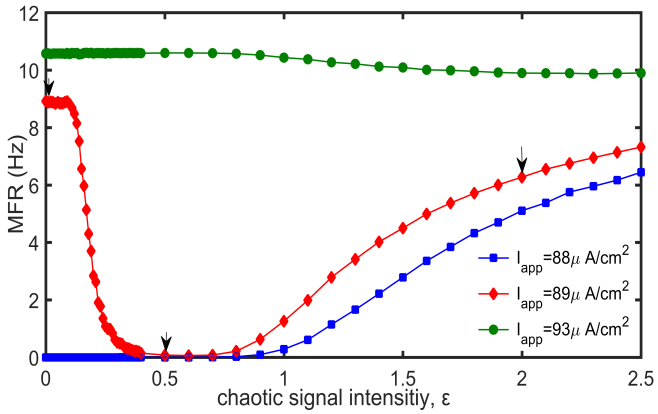

FIGURE 2. The dependence of average firing rate of ML neuron on the chaotic signal intensity for different I_{app} values

When the constant excitation current $I_{app} = 88\mu A/cm^2$, neither the chaotic signal level nor the applied constant current I_{app} is enough to induce spiking activity in ML neuron, and consequently the neuron is silent mode. In addition with the increase in chaotic signal intensity beyond $\varepsilon > 1$, the neuron starts to fire and has a increasing firing rate. On the other hand, for $I_{app} = 93\mu A/cm^2$, the neuron produces spikes at an almost constant firing rate and is unaffected by the chaotic signal levels. But, when we choose $I_{app} = 89\mu A/cm^2$, some different firing patterns emerge. It is observed that the neuron is not affected by chaotic signals at a certain level ($\varepsilon < 0.1$) and maintains its firing rate at a constant value of 9Hz. However, as the chaotic signal intensity increases, the firing frequency of ML neuron decreases swiftly and within a certain range of chaotic signal intensity the neuron becomes completely silent. Then, as the chaotic signal intensity further increases, the neuron starts to fire again and reaches a certain firing rate. These results obtained for $I_{app} = 89\mu A/cm^2$ showed that the inverse chaotic resonance effect can

occur in the ML neuron for the appropriate I_{app} value. That is, under a certain stimulus current value, the neuron is observed to exhibit the inverse chaotic resonance phenomenon depending on chaotic signal level. Finally, the ICR effect was not observed at smaller and larger I_{app} values, indicating that this effect largely depends on the excitability level of the ML neuron.

The mechanism of the occurrence of this phenomenon can be explained by the bifurcation diagram of the deterministic Morris Lecar neuron depending on the I_{app} current [35]. For I_{app} excitation current values of $88.29\mu A/cm^2 < I_{app} < 93.86\mu A/cm^2$, the ML neuron has a bistable attractor. One of them is the fixed point, which represents the resting state, and the other is the limit cycle attractor, which represents the spike formation in the neuron. The membrane potential shows chaotic fluctuations, creating loops around these attractors in certain orbits, and in appropriate cases, it remains under the influence of one of these attractors. Fluctuations caused by the chaotic signal in the range of the externally applied I_{app} current can change the membrane potential from firing to silence, or vice versa, from silence to firing. This triggers to emerge the ICR effect in the ML neuron.

3.2. **Effects of excitatory autapse on inverse chaotic resonance.** In this part of the study, by assuming that the ML neuron has a excitatory chemical autapse, we have analyzed the effects of autapse on the ICR effect in ML neurons. In order to show autapse's effect, first, we keep constant the excitation current as $I_{app} = 89\mu A/cm^2$, as the ICR phenomenon is present, and then depending on autaptic time delay τ, we calculate the MFR for three different chaotic signal intensities, representing low, intermediate and strong chaotic activity levels, which are marked black arrows in Figure 2. The results obtained for weak chaotic signal activity ($\varepsilon = 0.001$) and four different autaptic conductance values κ are given in Figure 3.

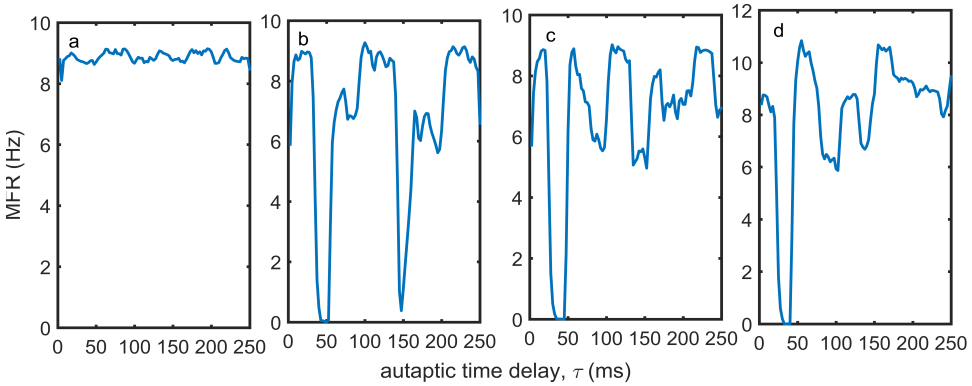

FIGURE 3. Variation of the mean firing frequency (MFR) of the ML neuron with chemical excitatory autapse according to the autaptic time delay (τ) at different autapse conductance values: (a)$\kappa = 0.01$, (b)$\kappa = 0.1$, (c)$\kappa = 0.5$, (d)$\kappa = 0.9$ ($\varepsilon = 0.001$, $I_{app} = 89\mu A/cm^2$).

When we examine Figure 3, for low autaptic conductance level $\kappa = 0.01$ (Figure 3a), the neuron is not affected by τ and has approximately $9Hz$ firing frequency. At a low-intermediate value of κ= 0.1, it exhibits some minimums almost with zero firing rate and maximums with around $9Hz$ depending on τ, implying the occurrence of multi-ICR (M-ICR) phenomenon in the firing activitiy of ML neuron. The ML neuron exhibits complete silent state in firing dynamics at around $\tau = 50, 150ms$ which are closely related to the integer multiple of the half of the intrinsic firing period ($T_{int} \approxeq 110ms$) of ML neuron. In addition, at higher autaptic conductance values (Figure 3c, Figure 3d), it is seen that the M-ICR phenomenon gradually loses its effect depending on increasing autaptic time delay, and the minima reflecting the silent mode in the firing dynamics of the ML neuron disappear to some extent except for the first minima.

The effects of excitatory autapse at an intermediate chaotic activity level are given in Fig.4. As can be seen in Figure 4a, for an intermediate chaotic activity intensity $\varepsilon = 0.5$, the MFR of ML neuron is consistent with the result obtained in Figure 2 (at $\varepsilon = 0.5$ in red curve) at low κ, except for some small peaks. But, when the autaptic conductance is increased to $\kappa = 0.1$, the neuron starting from silent mode shows patterns such as silence-firing-silence-firing, which is a clear indication for M-ICR phenomenon, depending on he autaptic time delay. But, for relatively strong autaptic conductance values (Figure 4c, 4d), The M-ICR phenomenon weakens due to strong autaptic effects.

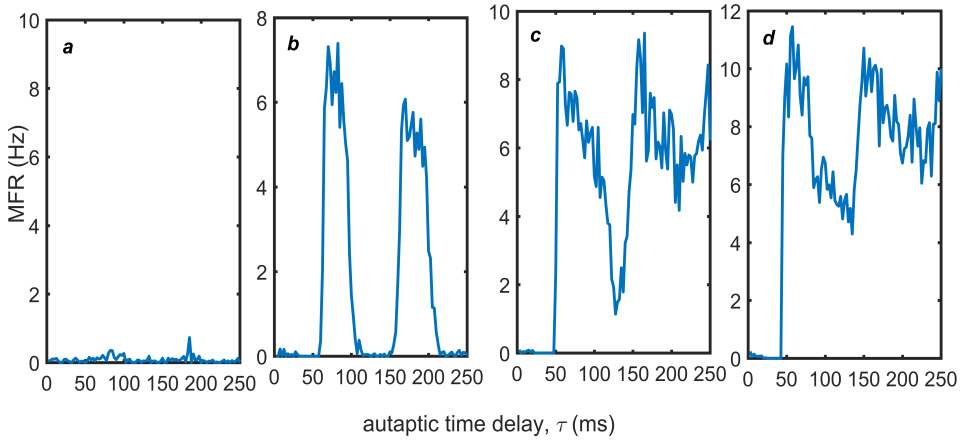

FIGURE 4. Variation of the mean firing frequency (MFR) of the ML neuron with chemical excitatory autapse depending on the autaptic time delay (τ) at different autapse conductance values:(a)$\kappa = 0.01$, (b)$\kappa = 0.1$, (c)$\kappa = 0.5$, (d)$\kappa = 0.9$ ($\varepsilon = 0.5$, $I_{app} = 89\mu A/cm^2$).

Finally, for excitatory autapse, we investigate its effects at high level of high chaotic activity in Figure 5. It can be seen in Figure 5 that when autaptic conductance is low $\kappa = 0.01$ the ML neuron fires around the firing rate of the without autapse case. If the κ is increased to higher values (Figure 5b, Figure 5c and Figure 5d) the neuron's firing rate increases up to $15Hz$ which is two fold increase in firing rate. Also, the MFR curves resemble the curves in the M-ICR phenomenon, but complete quietness in the

firing activity of the ML neuron does not occur. From the above results obtained for excitatory autapse, it can be deduced that, for weak and intermediate chaotic activity levels, the presence of excitatory autapse with moderate autaptic conductance levels can induce the M-ICR phenomenon in the firing activity of the ML neuron. In addition, regardless of the chaotic activity level, although strong autaptic conductance can increase the firing rate of the neuron, it prevents the occurrence of the M-ICR phenomenon.

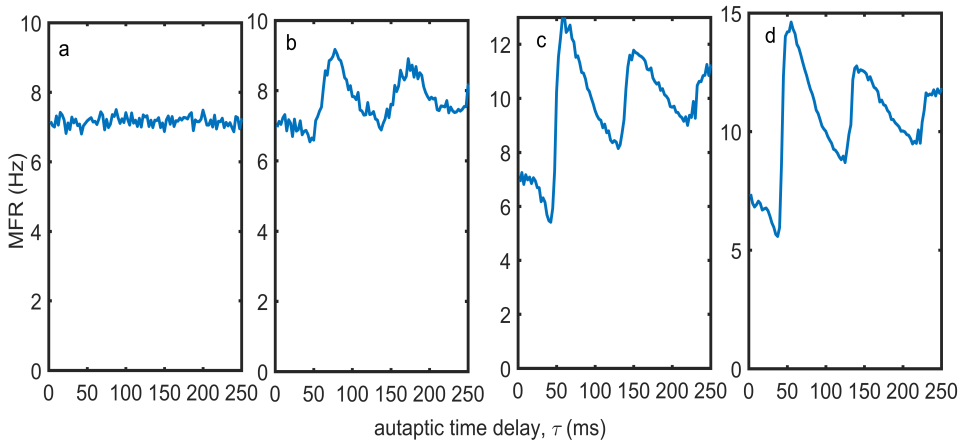

FIGURE 5. Variation of the mean firing frequency (MFR) of the ML neuron with chemical excitatory autapse according to the autaptic time delay (τ) at different autapse conductance values:(a)$\kappa = 0.01$, (b)$\kappa = 0.1$, (c)$\kappa = 0.5$, (d)$\kappa = 0.9$ ($\varepsilon = 2$, $I_{app} = 89\mu A/cm^2$).

3.3. **Effect of inhibitory autapse on inverse chaotic resonance.** In the present paper, finally, we have investigated the effects of inhibitory autapse on the ICR phenomenon in ML neurons. For this purpose, we consider that the ML neuron has an inhibitory autpase instead of excitatory one used in previous case. Then, following the way used in the case of exctitory autapse, we calculate the MFR of the ML neuron for weak, intermediate and strong chaotic activity cases. The results obtained are shown in Figure 6.

When analyzed Figure 6, for weak chaotic activity case $\varepsilon = 0.01$ (top panels of Figure 6), if the autaptic conductance is small (top left panel in Figure 6), there is no autaptic effects on the MFR of ML neuron, and the ML neuron has the firing rate similar to without autapse case. But, when κ is increased to $\kappa = 0.2$ (top-middle panel in Figure 6), due to the inhibiting effect of autapse the neuron becomes silent at some τ values. Besides, the autaptic time delay induced M-ICR phenomenon occurs. With the further increase in κ ($\kappa = 0.7$), the M-ICR effect almost disappears. In the case of intermediate chaotic activity case $\varepsilon = 0.5$ (middle panels in Figure 6 with red colored curves), the ML neuron is in silent mode at small κ due to too weak autaptic effects. When the κ is increased to $\kappa = 0.2$, due to increased autaptic effects some peaks emerge on the MFR of the ML neuron depending on τ. This pattern with multiple peaks in the MFR of ML neurons is a concrete evidence of the M-ICR phenomenon in the firing activities of the neuron. For strong autaptic conductance value ($\kappa = 0.7$), the silent mode corresponding to approximately

zero firing rate disappears at high values of autaptic time. This result implies that under strong autaptic effects, the M-ICR phenomenon can not occur in the firing activity of the ML neuron. On the other hand, for high chaotic activity case ($\varepsilon = 2$) at which the neuron without autapse produces spikes approximately at the frequency of $6.5Hz$, if the κ is low, that is, $\kappa = 0.01$, the neuron continuous to fire without being affected by the autapse. However, for medium and high κ cases, although small amplitude fluctuations occur in the MFR of the neuron, no strong evidence of M-ICR phenomenon is observed. From all the results obtained for inhibitory autapse, we can conclude that, on the one hand an inhibitory autapse, with proper time delay and conductance values, can induce the M-ICR phenomenon at low and medium levels of chaotic activity. On the other hand it increases the firing rate of the ML neuron to some extent.

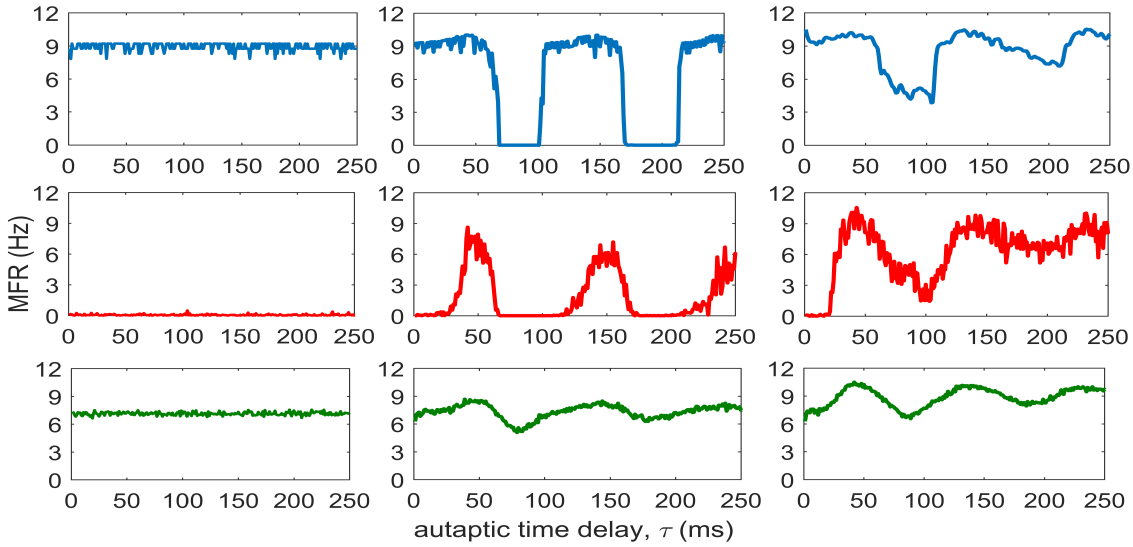

FIGURE 6. The mean firing frequency (MFR) of the ML neuron with chemical inhibitory autapse under different chaotic current intensities: Top panels $\varepsilon = 0.001$, middle panels $\varepsilon = 0.5$, bottom panels $\varepsilon = 2$ with the various autaptic conductances: left column $\kappa = 0.01$, middle column $\kappa = 0.2$ and right column $\kappa = 0.7$ ($I_{app} = 89\mu A/cm^2$).

4. CONCLUSIONS

Autapse is unusual synapse which forms between the dendrite and soma of the same neuron. After the discovery of its presence in different brain regions, its effects on neuronal dynamics have been aroused curiosity in the neuroscience community. Therefore a lot of studies some in numerical other are experimental are dedicated to investigate the effects of autapse on neuron dynamics. In the present paper, we study the effects of chemical autapse, by considering its excitatory and inhibitory types separately, on the ICR phenomenon. In excitatory autapse case, for weak chaotic activity level and at medium autaptic

conductance values, we found that when the autaptic time delay equals to integer multiple of half the intrinsic firing period of ML neuron, the presence of autapse causes the neuron stop firings, and by this way triggers the occurrence of M-ICR effect. In moderate chaotic activity level and at medium autaptic conductances, the presence of excitatory autapse induces firings in silent neuron when the autaptic time delay is proper values, and thus triggers the emergence of M-ICR phenomenon. Also, we obtained that this M-ICR effects gradually disappears as the autaptic conductance is increased regardless of the chaotic activity level. In case of inhibitory autapse, it is revealed that, for high chaotic activity level, the presence of inhibitory autapse does not have prominent effects on the firing dynamics of the ML neuron and not to induce any ICR effect.But, for low and moderate chaotic activity levels, the presence of autapse has distinct effects on the ML neuron dynamics, and causes the appearance of M-ICR phenomenon for medium autaptic conductances. In low chaotic activity level, it leads to the ML neuron, which fires a fixed firing rate, stop firings at some autaptic time delays and thus, induce the emergence of the M-ICR effect. But, in intermediate chaotic activity levels, the presence of inhibitory autapse gives rise to firing in silent neuron at some autaptic time delays and triggers the occurrence of autaptic time delay induced M-ICR phenomenon.

It has prominent importance of understanding the single neuron dynamics in different realistic conditions, since the single neuron is the basic building blocks of complex neuronal networks in brain. On the other hand, it is known that spiking neural networks (SNNs) mimics natural neuronal networks in the brain in a more realistic way than the classic artificial neural networks (ANNs) [38]. In this neuronal networks, the computation unit is a realistic neuron instead of artificial neuron or activation function in classical ANNs. In this context, Zhao et al. have investigated the performance of a deep SNN with autapse on standard data sets such as MNIST, CIFAR10, F-MNIST and N-MNIST, and obtained state-of-the-art performance [39]. Therefore, a clear understanding of the single neuron dynamics is of significant importance in designing more powerful and accurate SNNs model in solving real world problems with the approach of deep learning.

We, here, investigate ICR phenomenon in single ML neuron. However, in the brain, neurons are found in communities with different form of network topologies. Therefore, the investigation of the presented phenomenon in the complex neural networks is worthy. Thus, we want to put our efforts to investigate the ICR phenomenon in complex networks such as scale-free and small-world neural networks, in future studies.

DECLARATIONS

- **Conflict of Interest:** The authors declare no competing financial interests.

REFERENCES

[1] E. D. Adrian, The impulses produced by sensory nerve endings, The Journal of Physiology 61 (1) (1926) 49–72.

[2] A. P. Georgopoulos, A. B. Schwartz, R. E. Kettner, Neuronal population coding of movement direction, Science 233 (4771) (1986) 1416–1419.

[3] M. Abeles, Corticonics: Neural Circuits of the Cerebral Cortex, Cambridge University Press, 1991.

[4] M. Abeles, H. Bergman, E. Margalit, E. Vaadia, Spatiotemporal firing patterns in the frontal cortex of behaving monkeys, Journal of Neurophysiology 70 (4) (1993) 1629–1638.
[5] W. S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, The Bulletin of Mathematical Biophysics 5 (4) (1943) 115–133.
[6] R. H. Adrian, W. K. Chandler, A. L. Hodgkin, Voltage clamp experiments in striated muscle fibres, The Journal of Physiology 208 (3) (1970) 607–644.
[7] J. L. Hindmarsh, R. M. Rose, A. F. Huxley, A model of neuronal bursting using three coupled first order differential equations, Proceedings of the Royal Society of London. Series B. Biological Sciences 221 (1222) (1984) 87–102.
[8] E. Izhikevich, Simple model of spiking neurons, IEEE Transactions on Neural Networks 14 (6) (2003) 1569–1572.
[9] M. J. Chacron, A. Longtin, K. Pakdaman, Chaotic firing in the sinusoidally forced leaky integrate-and-fire model with threshold fatigue, Physica D: Nonlinear Phenomena 192 (1) (2004) 138–160.
[10] H. Hayashi, S. Ishizuka, M. Ohta, K. Hirakawa, Chaotic behavior in the onchidium giant neuron under sinusoidal stimulation, Physics Letters A 88 (8) (1982) 435–438.
[11] Y. Manor, J. Gonczarowski, I. Segev, Propagation of action potentials along complex axonal trees. model and implementation, Biophysical journal 60 (6) (1991) 1411–1423.
[12] L. Gammaitoni, P. Hänggi, P. Jung, F. Marchesoni, Stochastic resonance, Reviews of modern physics 70 (1) (1998) 223.
[13] D. F. Russell, L. A. Wilkens, F. Moss, Use of behavioural stochastic resonance by paddle fish for feeding, Nature 402 (6759) (1999) 291–294.
[14] J. K. Douglass, L. Wilkens, E. Pantazelou, F. Moss, Noise enhancement of information transfer in crayfish mechanoreceptors by stochastic resonance, Nature 365 (6444) (1993) 337–340.
[15] S. Lu, Q. He, J. Wang, A review of stochastic resonance in rotating machine fault detection, Mechanical Systems and Signal Processing 116 (2019) 230–260.
[16] B. McNamara, K. Wiesenfeld, Theory of stochastic resonance, Physical review A 39 (9) (1989) 4854.
[17] A. Palonpon, J. Amistoso, J. Holdsworth, W. Garcia, C. Saloma, Measurement of weak transmittances by stochastic resonance, Optics letters 23 (18) (1998) 1480–1482.
[18] E. Yilmaz, M. Uzuntarla, M. Ozer, M. Perc, Stochastic resonance in hybrid scale-free neuronal networks, Physica A: Statistical Mechanics and its Applications 392 (22) (2013) 5735–5741.
[19] B. S. Gutkin, J. Jost, H. C. Tuckwell, Inhibition of rhythmic neural spiking by noise: the occurrence of a minimum in activity with increasing noise, Naturwissenschaften 96 (2009) 1091–1097.
[20] D. Guo, Inhibition of rhythmic spiking by colored noise in neural systems, Cognitive neurodynamics 5 (2011) 293–300.
[21] H. C. Tuckwell, J. Jost, The effects of various spatial distributions of weak noise on rhythmic spiking, Journal of Computational Neuroscience 30 (2011) 361–371.
[22] D. Yu, Y. Wu, Z. Ye, F. Xiao, Y. Jia, Inverse chaotic resonance in hodgkin–huxley neuronal system, The European Physical Journal Special Topics 231 (22) (2022) 4097–4107.
[23] H. Van Der Loos, E. M. Glaser, Autapses in neocortex cerebri: synapses between a pyramidal cell's axon and its own dendrites, Brain research 48 (1972) 355–360.
[24] M. R. Park, J. W. Lighthall, S. T. Kitai, Recurrent inhibition in the rat neostriatum, Brain research 194 (2) (1980) 359–369.
[25] R. Preston, G. Bishop, S. Kitai, Medium spiny neuron projection from the rat striatum: an intracellular horseradish peroxidase study, Brain research 183 (2) (1980) 253–263.
[26] A. B. Karabelas, D. P. Purrura, Evidence for autapses in the substantia nigra, Brain research 200 (2) (1980) 467–473.
[27] R. Saada, N. Miller, I. Hurwitz, A. J. Susswein, Autaptic excitation elicits persistent activity and a plateau potential in a neuron of known behavioral function, Current Biology 19 (6) (2009) 479–484.
[28] G. C. Sethia, J. Kurths, A. Sen, Coherence resonance in an excitable system with time delay, Physics Letters A 364 (3-4) (2007) 227–230.

[29] V. Baysal, E. Yılmaz, M. Özer, Blocking of weak signal propagation via autaptic transmission in scale-free networks, IU-Journal of Electrical & Electronics Engineering 17 (1) (2017) 3091–3096.
[30] Y. Li, G. Schmid, P. Hänggi, L. Schimansky-Geier, Spontaneous spiking in an autaptic hodgkin-huxley setup, Physical Review E 82 (6) (2010) 061907.
[31] H. Qin, J. Ma, C. Wang, R. Chu, Autapse-induced target wave, spiral wave in regular network of neurons, Science China Physics, Mechanics & Astronomy 57 (2014) 1918–1926.
[32] H. Wang, J. Ma, Y. Chen, Y. Chen, Effect of an autapse on the firing pattern transition in a bursting neuron, Communications in Nonlinear Science and Numerical Simulation 19 (9) (2014) 3242–3254.
[33] E. Yilmaz, V. Baysal, M. Ozer, M. Perc, Autaptic pacemaker mediated propagation of weak rhythmic activity across small-world neuronal networks, Physica A: Statistical Mechanics and its Applications 444 (2016) 538–546.
[34] C. Morris, H. Lecar, Voltage oscillations in the barnacle giant muscle fiber, Biophysical journal 35 (1) (1981) 193–213.
[35] M. Uzuntarla, Inverse stochastic resonance induced by synaptic background activity with unreliable synapses, Physics Letters A 377 (38) (2013) 2585–2589.
[36] E. Yilmaz, M. Ozer, Delayed feedback and detection of weak periodic signals in a stochastic hodgkin–huxley neuron, Physica A: Statistical Mechanics and its Applications 421 (2015) 455–462.
[37] V. Baysal, Z. Saraç, E. Yilmaz, Chaotic resonance in hodgkin–huxley neuron, Nonlinear Dynamics 97 (2019) 1275–1285.
[38] W. Maass, Networks of spiking neurons: the third generation of neural network models, Neural networks 10 (9) (1997) 1659–1671.
[39] D. Zhao, Y. Zeng, Y. Li, Backeisnn: A deep spiking neural network with adaptive self-feedback and balanced excitatory–inhibitory neurons, Neural Networks 154 (2022) 68–77.

Vol: 2, Issue: 1, 2024
ISSN: 2980-3152
Pages:48-59

CURRENT TRENDS IN COMPUTING
https://dergipark.org.tr/en/pub/ctc
RESEARCH ARTICLE

THE GATHERING DECK BUILDER WITH REACT.JS AND CUTTING-EDGE WEB DEVELOPMENT

DANIEL MCCLOY[1], KEVIN BRYANT[1] AND YOUSEF FAZEA[1*]

[1] *Department of Computer Sciences and Electrical Engineering, One John Marshall Dr., Huntington, WV 25704, USA*

ABSTRACT. **This paper comes with a new web-based application "MtGDeckBuild". The main challenge addressed by this application is the integration of complex data management and user authentication systems within a responsive and interactive web interface. MtGDeckBuild leverages React.js, Node.js, and SQL Server to tackle these challenges. It uses Scryfall's full card data and Frontegg's secure user authentication APIs. By using the RAD paradigm, the article was able to successfully include several elements, such as user identification, responsive design, interactivity, and dynamic data display. The main achievements include the seamless incorporation of secure user authentication, efficient data management, and a scalable architecture, which significantly enhances the user experience and application performance. The prototype showcases the team's skill in obtaining and deploying new technologies and highlights the need to efficiently handle third-party dependencies. Improving functionality, increasing efficiency and scalability, and exploring greater IoT integration inside Smart Cities utilizing advanced web development frameworks are some of the prospects.**

1. INTRODUCTION

Although web development is a dynamic field, initiatives such as MtGDeckBuild illustrate how user-centered design and cutting-edge tools can simultaneously accommodate niche audiences and the general public. We will examine the intersection of frictionless user experiences and the rapid expansion of digital systems in this study. The foundations of initiatives like MtGDeckBuild can be traced back to previous research that prioritized user identification, dynamic data, and flexible design [1]. The findings of this study hold significant ramifications for application design across various domains, encompassing strategies for leveraging emerging technologies to enhance application performance and user engagement. React.js, SQL Server, and Node.js comprise the foundation of MtGDeckBuild's architecture. This paper demonstrates how React.js can be effectively used to create robust, scalable, and responsive web applications. Although primarily focused on web development, the study also touches upon the potential integration of IoT systems within Smart Cities, showcasing how web technologies can support advanced

E-mail address: yousef.fazea@marshall.edu (*).
Key words and phrases. JavaScript library, Front-End development, Virtual DOM, React JS..

urban infrastructures. MtGDeckBuild utilizes Frontegg and Scryfall, third-party APIs to ensure a seamless user experience and robust data management [2, 3]. By employing expertise and the most efficient web development methodologies, initiatives like MtGDeckBuild expedite the development of digital infrastructures. To eliminate risks, ensure the stability and scalability of the program, and address the difficulties of managing and regulating dependencies, additional R&D is required. Making this update effortless is the well-known server-side web application framework React.js. An improvement in customer happiness, an improvement in display quality, and a simplification of developer work are the three primary benefits of the product.

This paper is organized as follows: Section 2 introduces the literature, Section 3 presents the methodology and development process, followed by the findings and discussion in Section 4. The paper is concluded in Section 5.

2. Related Work

Web applications are made more useful and faster with the help of React.js since it incorporates capabilities like routing 6. As smaller component-based apps bring this framework online, the architecture, structure, and style of the primary program are retained [4,5]. With its crossover application concept, the React.js framework makes it easy to combine server-side and client-side websites [6–9]. This method allows programmers to construct robust JavaScript web applications without being concerned with the intricacies of the backend. While this approach is simple, it has to be fine-tuned for speed to meet the high expectations of e-commerce customers who want perfect online buying experiences. The second iteration of the Digitalization & Sustainability Review measures the amount of time it takes to provide data, styles, and code to clients, with an emphasis on how crucial it is for developers to use tactics that enhance the efficiency of React.js apps [10, 11]. For instance, sports-related web apps have complex tasks, such as collecting and processing many game videos and generating data during page load, which might hinder optimization and performance. Furthermore, conventional web development approaches can produce inefficient monolithic apps; in contrast, React.js's component-based design is quite different. Architectures based on cutting-edge parts also tend to use antiquated methods of web development, which leads to less performant monolithic apps. React.js has revolutionized online application development by introducing substantial improvements, making it the optimal solution for creating web apps that are scalable, easy to maintain, and deliver exceptional performance [12–14]. Finally, utilizing React.js in online shops and apps overcomes obsolete technology and capitalizes on new buying patterns. User-centered design and cutting-edge technology may boost customer satisfaction and help companies compete in the e-commerce industry [15–17]. This literature analysis emphasizes the drawbacks of traditional methods as well as React.js' advantages.

3. METHODOLOGY AND DEVELOPMENT PROCESS

Rapid Application Development (RAD) is used for the design, testing, and monitoring of study prototypes. The process has a few sequential steps: requirements, design, execution, validation, and maintenance. Efficient team communication is facilitated when everyone has a clear understanding of their

designated job. Consistently document and oversee tasks to guarantee excellence. This website seeks to actively involve and cater to those who have a strong interest in Magic via its goals, methodology, and range of coverage. Figure 1 displays the website for Magic: The Gathering, which is built on the RAD approach.

- Frontegg's user identification Software-as-a-Service (SaaS) is unparalleled. Users may now access and use Software as a Service (SaaS) via internet platforms.
- Frontegg provides authentication services for website administrators. Users have the option to either login or use Single Sign-On (SSO) to access the site.
- Utilize Frontegg's robust authentication system to limit access to site resources. Implement multi-factor and adaptive risk-based authentication to deter illegal access. API calls include the manipulation of dynamic data and encompass several application technologies. Here are the specific details of what we offer:
  - Frontegg is a sophisticated platform for user authentication. Frontegg may be used to authenticate API requests.
  - The card data API is the recommended method for accessing magic-related information on Scryfall. The app can display Scryfall card information by using its API.
  - Axios is an excellent JavaScript library that is interoperable with both browsers and nodes. JavaScript queries sent to websites. Axios can establish communication with external APIs, applications running on the server side, and databases. Frontegg, Scryfall, and Axios are reliable choices for app developers looking to make API calls. These technologies enable the application to verify the identity of users, conduct tests on API requests, get data from other sources, and establish communication with other apps and services.
  - Sign in to create and distribute presentations. The use of a responsive stylesheet enhances the app's performance and facilitates its usability on mobile devices. The app's layout adapts to the size of the viewport and is optimized for mobile devices, which improves the user experience. Adopting a flexible design ensures that mobile users do not experience delays or have difficulties with navigation. Enabling users to use their increasingly prevalent mobile devices to access the internet might enhance the app's value.

3.1. **List of Requirements:**

Two tables were created from the process's results: Table 1 presented the functional requirements of the study, whereas Table 2 delineated the non-functional requirements. The priority and level of relevance of each need are indicated by the same scale in both tables:

- M: Mandatory requirements (essential functions the system must fulfill).
- D: Desirable requirements (functions that are preferred to be included).
- Optional requirements (functions the system may include).

3.2. **Activity and Sequence Diagrams:**

The user proceeds to the login page of the Software-as-a-Service (SaaS) application and enters their credentials. Front-egg verifies the credentials and produces an access token. This token is returned to the SaaS application to verify the user's identity and authorize access to the primary interface. The process

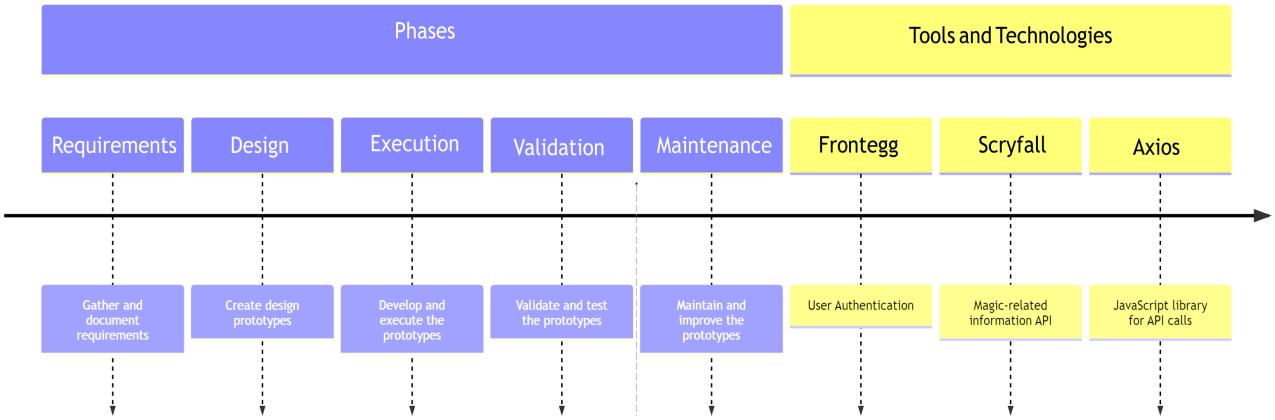

FIGURE 1. **Rapid Application Development Model [18].**

TABLE 1. **Functional Requirements.**

1	MtGDB_01	Login	
	MtGDB_01_02	The User logins in via username and password	M
	MtGDB_01_03	The User logins in via Google Account	M
	MtGDB_01_04	The User logins in via the GitHub Account	M
	MtGDB_01_05	The User logins in via the Microsoft Account	M
	MtGDB_01_06	The User logins in via the Facebook Account	M
2	MtGDB_02	**Deckbuilding**	
	MtGDB_02_01	The User queries a card	M
	MtGDB_02_02	The User adds the queried card to the checklist in a specified quantity	M
	MtGDB_02_03	The User adds the Deck Name and submits the deck	M
3	MtGDB_03	**Deck Viewing**	
	MtGDB_03_01	The User selects a deck from the Recent Decks component	D
	MtGDB_03_02	The User selects "View Decks" from the Navbar component	D
4	MtGDB_04	**Logout**	
	MtGDB_04_01	The User logs out via the "Logout" button from the Navbar component	M

of login is shown in Section (a) and visualized in Figures2 (a) and (b). To conduct a card search, the user enters a string into the designated search field, which initiates an auto-complete feature that produces a compilation of proposed card names. As soon as the user enters a card name, the application loads the

TABLE 2. **Non-Functional Requirements.**

1	MtGDB_05	Security	
	MtGDB_05_01	The system should be secure by specifying the user's tasks and not allowing an unauthorized person to use the system.	M
2	MtGDB_06	**Usability**	
	MtGDB_06_01	The System must be easy to deal with.	D
3	MtGDB_07	**Understandability**	
	MtGDB_07_01	The System must be easy to understand	D
4	MtGDB_08	**Reliability**	
	MtGDB_08_01	The system should present the same selected sequence.	D
5	MtGDB_09	**Performance**	
	MtGDB_09_01	The system must have a reasonable speed according to the technology used to access many Users at the same time.	O
	MtGDB_09_02	The system spends on searching must be less than 2 seconds	D
6	MtGDB_10	**Availability**	
	MtGDB_10_01	The system should be available to all kinds of Users	D
	MtGDB_10_02	The system must not crash and if that happened, it should take less time to be back again.	D

image that corresponds to that card. The process of card searching is shown in Section (b) and visualized in Figures3 (a) and (b).

(a) **Login:**

- The user navigates to the login page of the SaaS application.
- The SaaS application sends a request to Frontegg to initiate the login process.
- Frontegg presents the user with a login page, where they enter their email address and password.
- The user submits their login credentials.
- Frontegg validates the user's credentials and generates an access token for the user.
- Frontegg sends the access token back to the SaaS application.
- The SaaS application uses the access token to authenticate the user and grants access to the application's resources.
- The user is redirected to the SaaS application's main dashboard or home page.
- This activity diagram illustrates how Frontegg's authentication and identity management capabilities can be used to secure access to SaaS applications. By leveraging Frontegg's services, developers can focus on building their core applications, rather than worrying about the complex details of authentication and access control.

(b) **Search Card Activity Diagram:**

- The user navigates to the Search Card component of the application and inputs a string into the search field.
- The application triggers an auto-complete method to generate a list of suggested card names based on the string input by the user.
- The application displays the list of suggested card names in a drop-down menu beneath the search field. The user selects a card name from the list.
- The application loads the corresponding card image associated with the selected card name.
- he application displays the card image on the page.

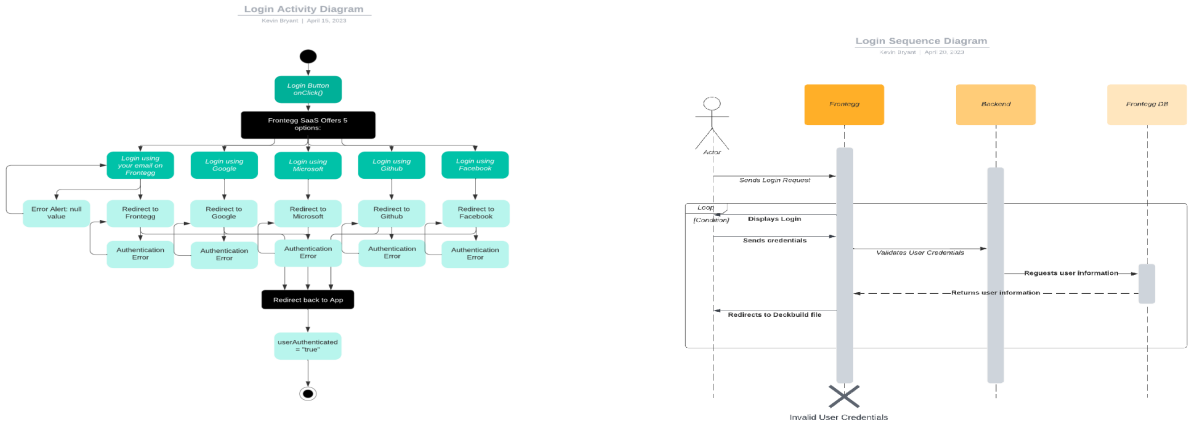

FIGURE 2. **(a) Login activity diagram, (b) Login sequence diagram.**

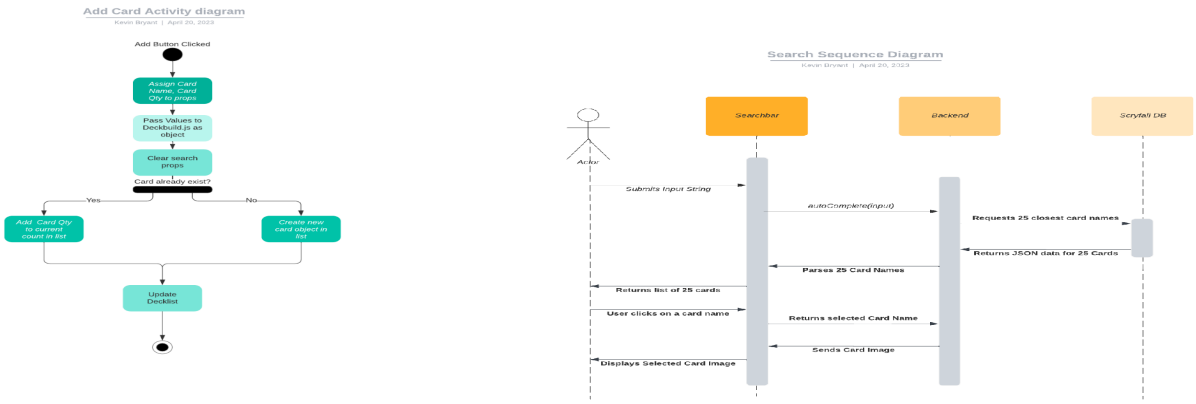

FIGURE 3. **(a) Add and remove card to deck activity diagram, (b) Search card sequence diagram.**

(c) **Case Diagram and Geographical User Interface:**

According to Figure4, a user may do a variety of functions in the system, including logging in, searching for cards, adding cards, removing cards, and disengaging from the system. The fact that these activities are confined inside the system's limits implies that they are system-provided characteristics. The user interacts with each of these skills in line with their needs. Following the completion of the design of each diagram, the next step is to create the website by utilizing React. JS. Figures4,5,6, and 7 illustrate the proof of concept that was found.

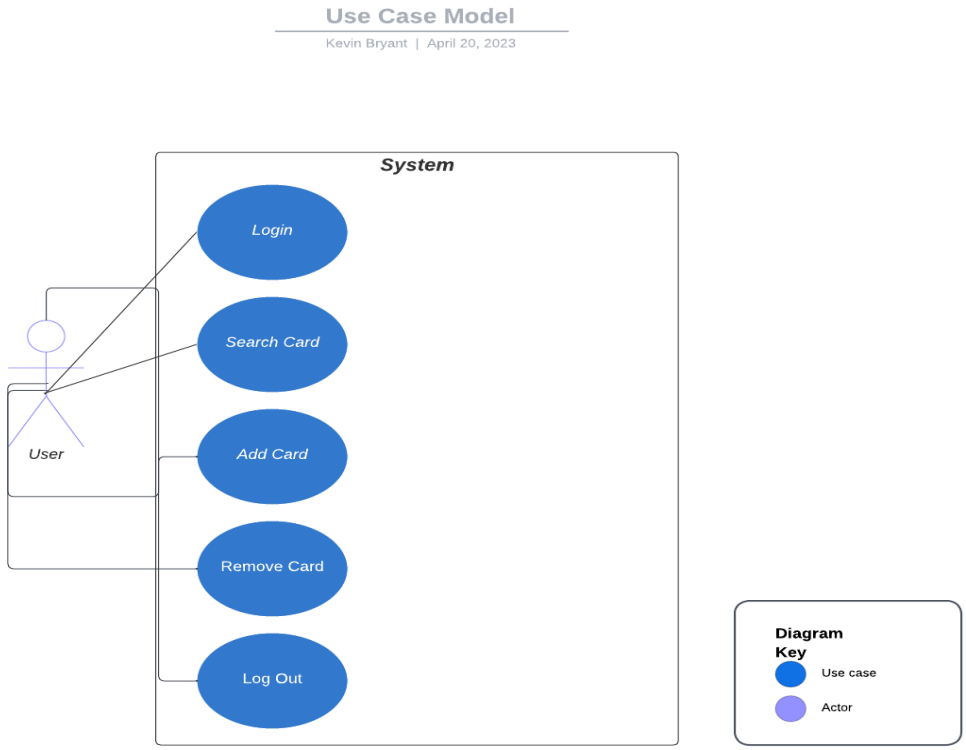

FIGURE 4. **Login page (With Feedback Messages).**

4. RESULTS AND DISCUSSION

To assess the usability of the prototype, data was collected from a sample of thirty distinct participants. We have assessed the Web-based application system utilizing criteria such as Intuitive Design, Easy Navigation, Memorability, and Satisfaction. Figure8 (a) shows the result of a question asked about the

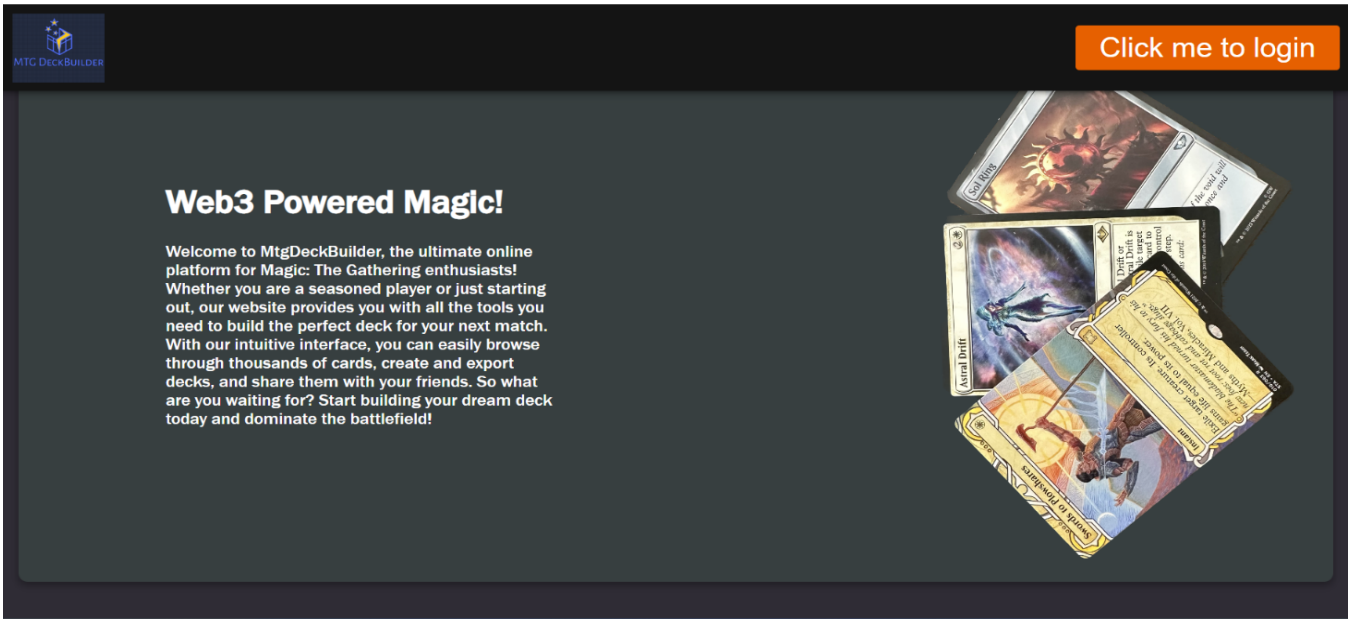

FIGURE 5. **Home page.**

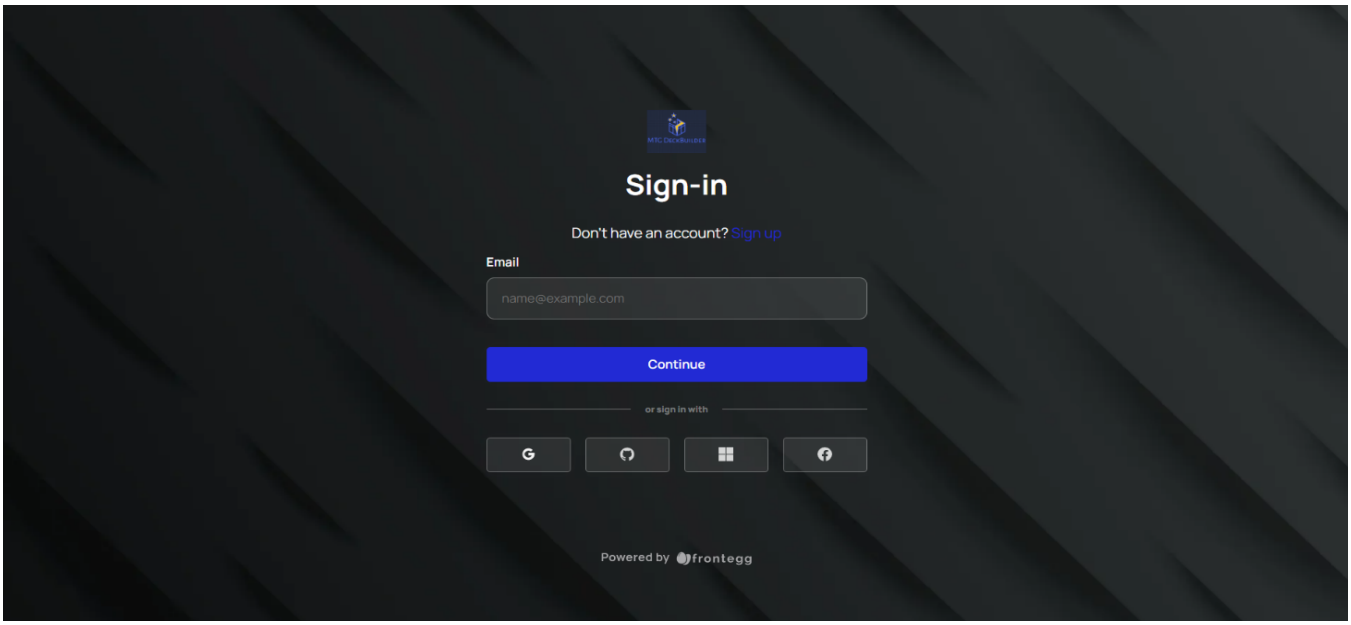

FIGURE 6. **Login page.**

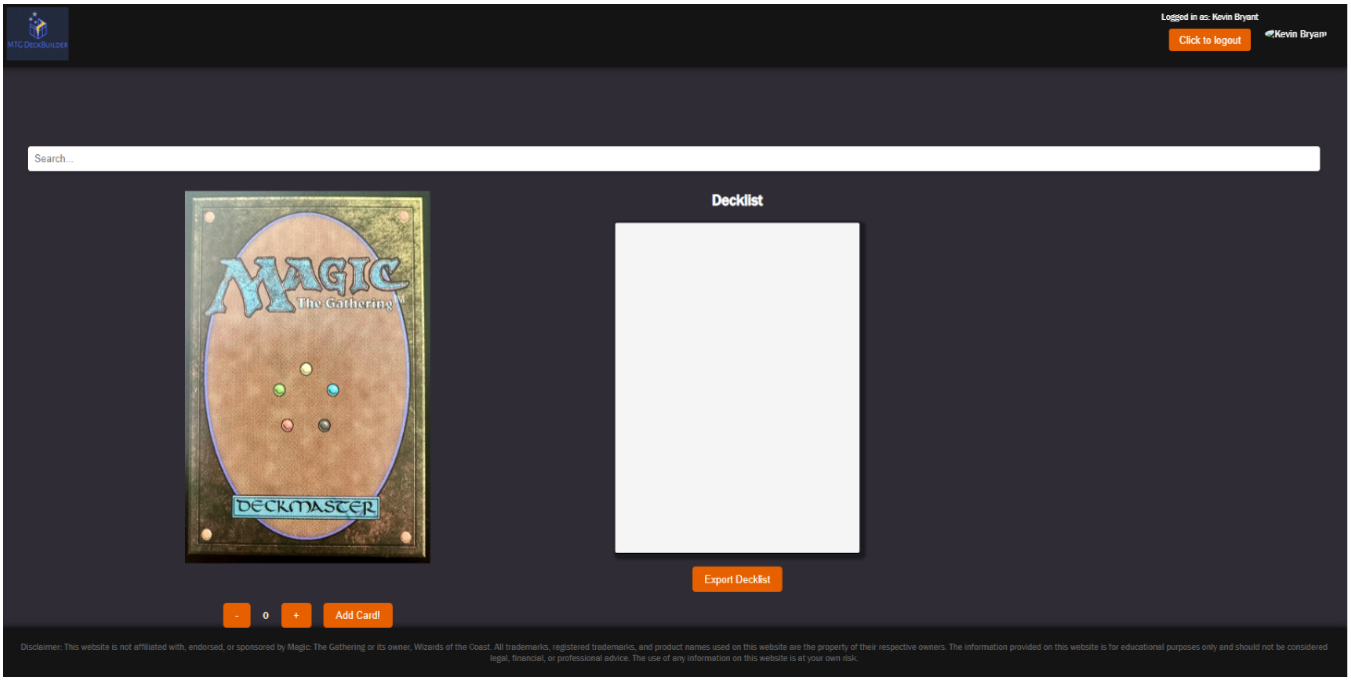

FIGURE 7. **Searching page.**

intuitive design, a nearly effortless understanding of the architecture and navigation of the site. The responses were shown using a Likert scale, where a score of 1 indicated "Strongly Disagree" and a score of 5 indicated "Strongly Agree." The bar chart illustrates the allocation of these scores. The ratings "Strongly Disagree" and "Disagree" both received a score of 0%. Only 5% of the survey respondents selected group 3, indicating their indifference. 20% of the poll respondents agreed with the statement, classifying them into group 4. Category 5 received 75% of the vote, indicating a widespread consensus among most individuals. Figure8 (b) shows the ease of navigation, and how fast users can accomplish tasks. A significant majority of respondents expressed strong agreement with the statement, indicating a largely positive consensus. Figure8 (b) illustrates that no users had any difficulties with navigation. Category 3 represented a minuscule proportion of individuals who had a neutral stance, amounting to just 5%. Unsurprisingly, 20% of the study respondents rated the navigation as straightforward, categorizing it as category 4. The data from Category 5 indicates that most respondents 75% found the navigation to be straightforward. Based on these statistics, most visitors have a positive perception of the site's navigational simplicity.

The bar chart Figure9 (a) measures the site's memorability after visiting the site if a user can remember enough to use it effectively for future visits. Only 5% disagreed, suggesting minimal trouble remembering site characteristics for future visits. 15% of users were unclear if the site was memorable. With a 25% agreement, many consumers remembered the site well enough for future visits. The fact that 55%

of users agreed suggests that most visitors liked the site and could recall things for their future visits. These figures suggest that most users find the site simple to remember and can recall details from past visits. Figure 9 measures user satisfaction where the users were asked if they like using the system and find it friendly. 30% agreed indicating good satisfaction. Most responders 70% strongly agreed, suggesting high site satisfaction. According to these statistics, most users are satisfied with their experience, suggesting a positive user impression.

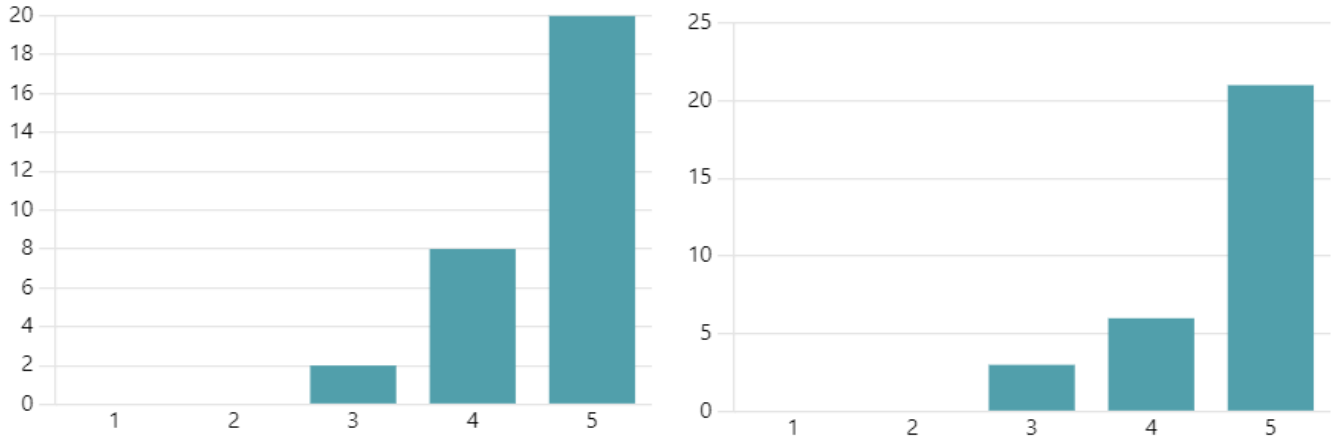

FIGURE 8. **(a) Intuitive design (b) Easy navigation.**

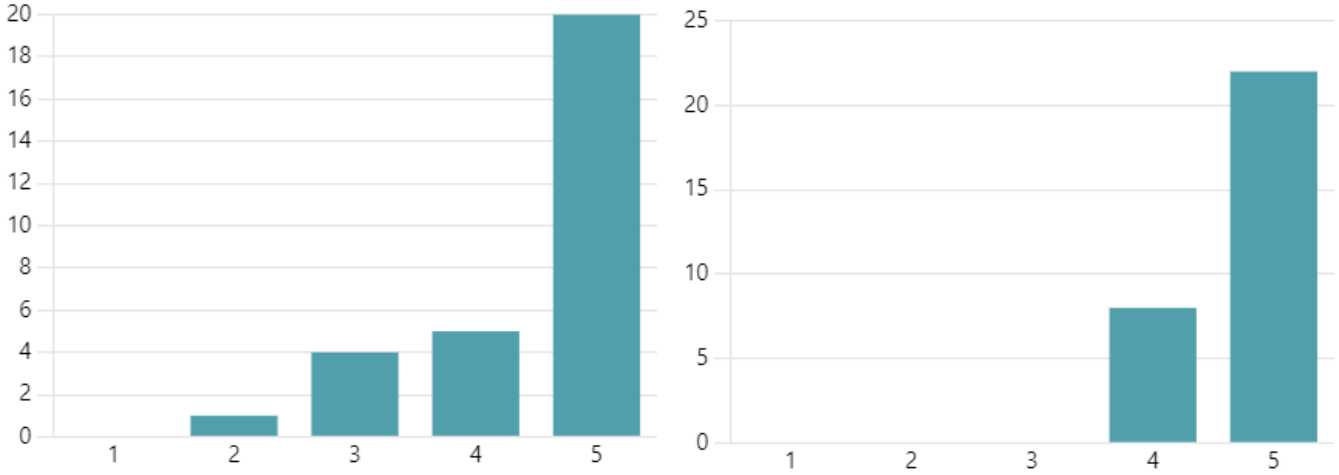

FIGURE 9. **(a)Memorability (b) Satisfaction.**

5. Conclusion

The extensive MtGDeckBuild study investigated contemporary web development methodologies and tools. Frontegg and Scryfall oversaw the administration of user accounts and the enormous Magic: The Gathering card database. Simple access to the API via Axios enhanced the app's usability. Due to the complexity and breadth of the software development process, adaptability and problem-solving skills are essential. Webhooks implemented by Frontegg impeded deck storage and viewing integration. The implementation of these fixes and the maintenance of change logs assisted the group in decreasing reliance on external services and enhancing backend architectural control. Players of Magic: The Gathering derive the greatest benefit from the intuitive and dynamic platform of the study. MtGDeckBuild's innovative technologies and user experience establish it as the industry leader in both casual and competitive play. Enhancing the development, learning, and innovation processes are the objectives of the study team. By enhancing and empowering users, MtGDeckBuild ensures its success in the ever-evolving market for web-based gaming applications.

Declarations

- **Conflict of Interest:** The authors have not disclosed any competing interests.

References

[1] D. Abramov, R. Nabors, Introducing react.dev (2023).
URL https://react.dev/blog/2023/03/16/introducing-react-dev

[2] S. Chen, U. R. Thaduri, V. K. R. Ballamudi, Front-end development in react: an overview, Engineering International 7 (2) (2019) 117–126.

[3] A. Bodepudi, M. Reddy, S. S. Gutlapalli, M. Mandapuram, Voice recognition systems in the cloud networks: Has it reached its full potential, Asian Journal of Applied Science and Engineering 8 (1) (2019) 51–60.

[4] Z. Dinku, React. js vs. next. js (2022).

[5] G.C.Trends, The new experience economy (2022).
URL www.dynata.com/content/GCT_The-New-Experience-Economy.pdf

[6] L. Duy, Web application development (2024).

[7] Q. Odeniran, H. Wimmer, J. Du, Javascript frameworks—a comparative study between react. js and angular. js, in: Interdisciplinary Research in Technology and Management, CRC Press, pp. 319–327.

[8] D. M. Nguyen, Design and implementation of a full stack react and node. js application: simulating driver's license exams (2024).

[9] E. B. Pranata, T. Tony, Utilizing orb algorithm in web-based sales application, Journal of Information Systems and Informatics 6 (1) (2024) 378–398.

[10] O. Lyxell, Server-side rendering in react: When does it become beneficial to your web program? (2023).

[11] C. Minnick, Beginning reactjs foundations building user interfaces with reactjs: an approachable guide, John Wiley & Sons, 2022.

[12] Gaper, How react js is revolutionizing web development (2023).
URL https://gaper.io/how-react-js-is-revolutionizing-web-development/

[13] V. Sahni, A. Chopde, M. Goswami, A. Kumar, Mern (mongodb, express-js, react-js, node-js) stack web-based themefied education platform for placement preparation, Educational Administration: Theory and Practice 30 (5) (2024) 1918–1928.

[14] T. A. Kràlusz, Mobile application development with react native and leveraging third-party libraries (2024).

[15] I. Rizvi, H. Gupta, I. Bharadwaj, et al., Connect easy: Revolutionizing the college experience through innovative web-based solutions (2024).

[16] N. Garg, J. Chopra, V. Kumar, K. Aggarwal, J. Parashar, A. Jain, Applego: React js (web application).

[17] B. Gamage, R. Ranaweera, A. Dilshan, R. Paranagama, D. De Silva, S. Vidhanaarachchi, Centralized platform for managing activities in e-commerce store, International Journal Of Engineering And Management Research 12 (5) (2022) 203–208.

[18] T. Kissflow, What is rapid application development (rad)? an ultimate guide for 2024. (2024).
URL https://kissflow.com/application-development/rad/rapid-application-development/#:~:text=Rapid%20Application%20Development%2C%20or%20RAD,less%20emphasis%20on%20specific%20planning

Vol: 2, Issue: 1, 2024
ISSN: 2980-3152
Pages:60-85

CURRENT TRENDS IN COMPUTING
https://dergipark.org.tr/en/pub/ctc
RESEARCH ARTICLE

ANOMALY DETECTION WITH API CALLS BY USING MACHINE LEARNING: SYSTEMATIC LITERATURE REVIEW

VAROL ŞAHİN[1], FERHAT ARAT[2*], SEDAT AKLEYLEK[3]

[1] *Department of Computer Engineering, Ondokuz Mayis University, Samsun, Türkiye*
[2] *Department of Software Engineering, Samsun University, Samsun, Türkiye*
[3] *Department of Computer Engineering, Istinye University, Istanbul, Türkiye*
[3] *Institute of Computer Science, University of Tartu, Tartu, Estonia*

ABSTRACT. **API, in other words system calls are critical data sources for monitoring the operation of systems and applications, and the data obtained from these calls provides a wealth of information for anomaly detection. API calls are the basic building blocks of the interaction between the operating system and user applications, and analysis of these calls provides important data for securing the system. Anomaly detection is crucial for system security and performance. ML models learn normal and abnormal behaviors by processing large amounts of data and use this information to detect anomalies in new data. When anomaly detection using system calls is combined with ML algorithms, it can make more precise and accurate detections. In this paper, we focus on anomaly detection with machine learning methods using API calls. We present a SLR on the topic as well as a SoK by providing basic knowledge. The main goal is to describe, synthesize, and compare security advancements in anomaly detection using API calls with ML algorithms by examining them through the lens of various research questions. More than 30 research papers were retrieved using search phrases identified from common and reputable databases, and those relevant to the topic were included in the SLR using different screening criteria. In addition, the reviewed studies were compared in terms of different metrics such as dataset, platform, success parameter, used ML method, and features.**

1. INTRODUCTION

Smartphones, computers and other electronic devices are involved in every aspect of our daily lives. With the Internet becoming an indispensable element, system security has become an indispensable requirement in both personal and corporate organizations. A system is a set of components that interact with each other and usually form a complex whole. Alternatively, a system can be defined by the functions and behaviors it contains. The interactions between systems, their interconnections, environmental conditions and human governing factors make questions about system safety complex. for instance,

E-mail address: ferhat.arat@samsun.edu.tr (*).
Key words and phrases. API call, system call, anomaly detection, machine learning.

the need for monitoring, measurement and control are critical elements to consider in system interconnections. System security is an important issue not only at the individual and organizational level, but also at the societal level. Keeping up with rapidly growing and evolving technological developments is inevitable in this context.

Today, information security has become even more crucial with the increase in cyber threats. Cyber attacks can cause a wide range of damages at the user and system level, from the theft of personal data to the collapse of corporate systems. Therefore, security measures need to be handled with a proactive approach, not just a reactive one. Advanced monitoring and intrusion detection systems, artificial intelligence and machine learning techniques are used to identify and prevent potential threats in advance. In addition, user training and awareness is also considered an important component of system security. Human error is one of the fundamental causes of many security breaches. Therefore, making users aware of security protocols and teaching them the necessary applications and systems will improve the overall security posture. Anomaly detection is a critical area that aims to detect security threats in advance by identifying unusual behavior of systems. Anomaly detection is also a technique used to detect data samples that do not fit the data model and is an important research area that is being studied for many applications [1], [2]. The main purpose of anomaly detection is to distinguish between normal and abnormal data. This method is very significant in data analysis as it enables the identification of emerging patterns, trends and anomalies in the data [3].

Anomaly detection is critical for various applications such as security, health, network monitoring [4]. In this context, cybersecurity is an emerging and important research domain with applications across various domains such as healthcare, building management, weather forecasting, etc. [4]. Anomalies were considered important because they can point to very important and rare events, enabling critical measures to be taken in a wide range of application areas.

Anomaly detection using system calls, also known as Application Programming Interface (API) calls, provides a detailed analysis of the internal dynamics and behavior of systems. System calls are the basic building blocks that enable the interaction between the operating system and user applications, and analysis of these calls provides important data to ensure the security of the system. An API is an interface that a software program provides to other programs, users, and in the case of web APIs, to the world via the internet [5].

The accuracy and reliability of the methods used in anomaly detection are directly related to the datasets used and the training of the model. The datasets used in the training process are required to adequately represent normal and abnormal behaviors. In this context, the datasets include various types of attacks and system behaviors and provide ideal test environments to evaluate the performance of anomaly detection models. Various features, such as API calls or system calls, registry modification and network activity, constitute the behavior of malware. API calls and various information related to these calls extracted by dynamic analysis are considered as one of the most important features of behavior-based detection systems. Each API call in the sequence is associated with the previous or next API call. These and similar relationships may contain patterns of destructive functions of malware. Many anomaly detection systems, including ML and deep learning models, have been proposed that use various information about API and system calls as features. In particular, ML methods and deep learning algorithms are

used to improve the performance of such anomaly detection systems. for instance, deep learning models can achieve high accuracy rates on complex datasets and can be effectively used in malware detection. In this context, the integration of MLand deep learning methods is of great importance to improve system security and detect malware effectively. There are three anomaly detection techniques: supervised, unsupervised and semi-supervised.

This paper presents a comprehensive Systematic Literature Review (SLR) focusing on anomaly detection with API calls using ML techniques, utilizing the Systematization of Knowledge (SoK) approach. In addition to ML fundamentals and principles, we offer an expansive framework with a specific emphasis on api calls in anomaly detection, contributing to the systematic organization of information. wUnlike existing reviews, our focus is explicitly directed towards using anomaly detection with API calls. We have initiated a comprehensive literature search methodology. The objective is to describe, synthesize, and compare security developments in terms of anomaly detection with API calls.

TABLE 1. Abbreviations and definitions

Abbreviation	**Definition**	**Abbreviation**	**Definition**
API	Application Programming Interface	ML	Machine Learning
KNN	K-Nearest Neighbors	SVM	Support Vector Machine
LSTM	Long Short-Term Memory	CNN	Convolutional Neural Network
ADFA-LD	Australian Defence Force Academy Linux Dataset	DARPA	Defense Advanced Research Projects Agency
UNM	University of New Mexico	IDS	Intrusion Detection System
DDoS	Distributed Denial of Service	TCP	Transmission Control Protocol
URI	Uniform Resource Identifier	HTTP	Hypertext Transfer Protocol
LR	Logistic Regression	NB	Naive Bayes
RBF	Radial Basis Function	WoS	Web of Science
CPS	Cyber Physical System		

1.1. **Motivation and Contribution.**

In this section, we give our motivation as well as summarize the main contributions of the paper. With the rapid advancement of today's technological developments, the use of interconnected systems and applications by individuals and corporate organizations to facilitate operations, increase productivity and provide a seamless experience to users is increasing in parallel.These systems and applications are made possible by APIs that act as a bridge by enabling communication and data exchange between different software components. The increasing use of APIs also increases the need for security measures to protect against potential threats.

The fact that Internet technologies have been adopted and become an integral part of daily life has brought with it disadvantages such as misuse and vulnerability to abuse. The increasing complexity of large amounts of data circulating on the Internet increases the risk of anomalies, unexpected patterns or behaviors.Anomaly identification through API requests is particularly crucial in this context. Especially monitoring an application's API calls to understand its behavior and logic. As they facilitate data transmission and communication between applications, API calls are critical for functionality.

Anomaly detection helps identify unusual behaviors in the system, enabling timely responses before they cause any damage to the application or the system. It detects abnormal requests and intrusion attempts and in this way it helps to protect the system and reduce the costs of deployment and maintenance.

Anomaly detection gives vital information regarding both operations and security. Differences from standard API behavior might indicate problems. Early identification of anomalies allows applications to run more smoothly and efficiently. For example, if an API call is unusually slow, anomaly detection ensures that the issue is identified and resolved before it negatively impact the user experience. Anomaly detection also help protect against financial losses by identifying illegal activities that produces unusual API requests patterns. Detecting such unusual activities early protects businesses from financial harm. As the number of users and systems increase, the volume of API requests also grows. Anomaly detection helps to monitor these large quantities of requests and identify anomalies, reducing the need for manual monitoring and minimize the human intervention. In this regard, our contributions are as follows:

- Our Systematic Literature Review (SLR) offers guidelines enabling researchers to analyze and address specific questions, presenting a Systematization of Knowledge (SoK) approach focused on anomaly detection methods involving API calls from a ML perspective.
- We only consider anomaly detection approaches with API calls using ML techniques.
- We examine ML methods to detect anomalies in the system which use API call sequences as data features.
- We present a detailed comparison for literature based on technique, used data, peformance metric, and other parameters.

1.2. **Research Methodology.**

A Systematic Literature Review (SLR) is a research method that involves systematically collecting, critically assessing, and synthesising existing studies on a clearly defined topic. SLR provides a methodical, repeatable, and reliable framework for reviewing literature. This approach aims to reduce the complexity of research, improve transparency, and offer a thorough understanding of the current state of knowledge in a specific field. In this paper, we aims to extract anomaly detection approaches which utilize API calls in ML concepts to highlight differences, algorithmic design, data features, environmental requirements of the existing studies. Therefore, our primary key is to prepare literature summation as well as presenting compact, collective, and well-constrained SoK. In this Systematic Literature Review (SLR), we employ a set of key terms pertinent to anomaly detection involving API calls. To conduct the research and gather relevant studies, keywords such as ”anomaly detection,” ”API calls,” and ”Machine Learning” are used to create meaningful and consistent search queries. Additionally, an advanced search mode is utilized across five prominent databases: Scopus, Web of Science (WoS), ACM Digital Library, Science Direct, and IEEE Xplore. Then, we apply three-stage research model defined as bellows:

(1) Definition: In this step, we create different combinations of essential keywords to ensure a reliable and consistent search in databases. We also develop research questions that consider our research focus and key factors, preparing for the SLR.
(2) Determination: In this stage, we determine searching filters and used sentences for advanced search.

(3) Elimination: In the last step, we eliminate obtained studies regarding elimination metrics and main purpose of the SLR. Therefore, we create inclusion and exclusion methodologies as shown in Table 2. And we apply the determined manual filters such as ”research article” and ”computer science research area”.

1.3. **Research questions and planning the review.**

To outline the review's future direction, defining the primary objectives is essential. We set these goals to provide a comprehensive and practical viewpoint. Our SLR results aim to deliver conclusions that are applicable, realistic, and easy for researchers to understand. We review the papers using various criteria, including the datasets used, ML methods employed, performance metrics, and data features. To achieve our objectives, we develop research questions that break the research into sub-phases. We outline the generated research questions as follows:

(1) What datasets and data features are most commonly used in ML methods for anomaly detection utilizing API calls?
(2) How is the performance of ML methods for anomaly detection using API calls evaluated, and which metrics are most commonly utilized?
(3) Which ML methods for anomaly detection using API calls are the most effective, and what characteristics make these methods stand out?

We expand our research by considering the research questions defined above. For each research question, we identify the points of description, comparison, and summarizing that we consider important when reviewing the research papers within the scope of our study. In this context, the first question aims to identify the datasets and data features that are frequently employed in ML methods for anomaly detection using API calls. Understanding which data sources and features are preferred and yield successful results is the goal. The second one focuses on the performance evaluation of studies. Analyzing the performance metrics used helps compare different methods and identify the most commonly preferred metrics. The final question aims to determine the effectiveness of different ML methods. Comparing various methods and analyzing which ones yield better results will be one of the key findings of our study. Figure 2 shows research methodology applied throughout paper.

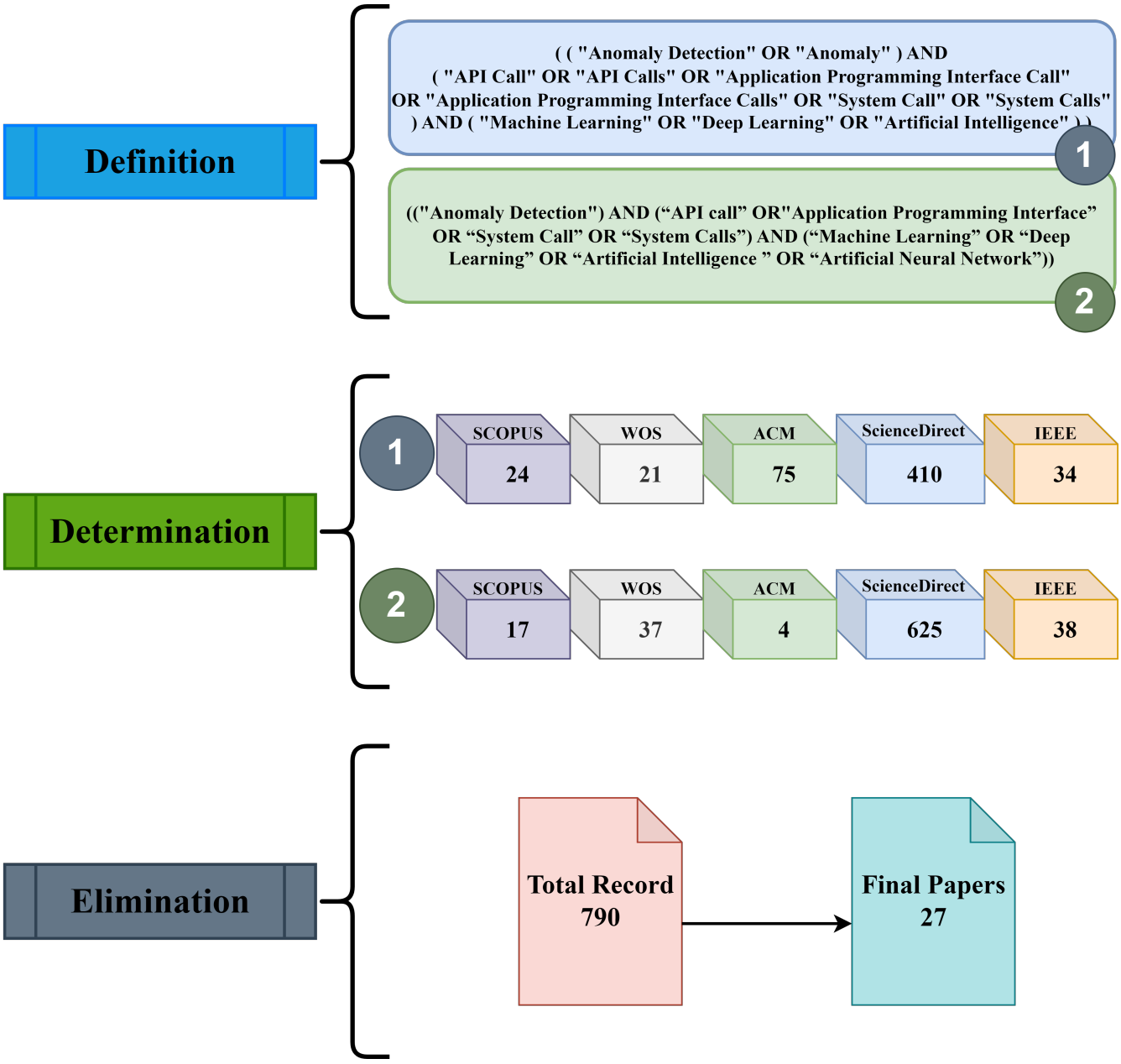

FIGURE 1. Applied research methodology of the paper

1.4. **Determining and performing of the investigation.**

In this section, we address the planning process for selecting studies to be reviewed, following the stages of formulating research questions defined to better specify the focus areas by expanding the SLR

and creating search phrases using keywords. The selection of studies is carried out manually and by using the filtering methods of search engines. In this context, the initial results are subjected to several logical filters to form a study set appropriate for the research. Table 2 shows inclusion and exclusion metrics applied during determining and eliminating the articles.

TABLE 2. Applied research criterion

Exclusion criteria	**Inclusion criteria**
The study must focus on anomaly detection using API calls.	Duplicated papers.
The study must be published in English.	Published in any language other than English.
The study must be related with research fields of computer science or computer engineering.	Review article, conference paper, or another types except research article.
The study should use ML techniques considering API calls for anomaly detection.	Studies using different datasets other than API calls for anomaly detection.

Table 2 outlines the exclusion and inclusion criteria employed in the selection of studies pertinent to anomaly detection using API calls. The exclusion criteria specify that eligible studies must be published in English, ensuring they are accessible to an international audience. Additionally, the studies must be relevant to the fields of computer science or computer engineering, thereby maintaining technical relevance. Methodologically, the studies must utilize ML techniques specifically focusing on API calls for anomaly detection, ensuring consistency with the research objective. Conversely, the inclusion criteria exclude duplicated papers, those published in languages other than English, and studies that are not original research articles, such as review articles or conference papers. Furthermore, studies using datasets other than API calls for anomaly detection are also excluded. These stringent criteria ensure that the selected studies are both methodologically relevant and focused on the specified research area, thereby enhancing the reliability and validity of the research findings.

1.5. **Organization.**

The paper is organized as follows: In Section 1, we explain general definitions of the paper, describing research questions, research structure, and investigation purposes. In Section 2, we give the basics of the topic defining API calls and importance of the anomaly detection using API calls. In Section 3, we present ML concepts giving fundamentals, definitions, and classifications. In Section 4, we highlight and summarize obtained studies in terms of varying research parameters. In addition we compare literature considering specified metrics. In Section 5, we conclude the paper defining open problems and challenges for focused research problems.

2. ANOMALY DETECTION

In this section, we address the topic of anomaly detection, providing general information as well as discussing its significance, the role of API calls in anomaly detection, detection techniques, and the use

of ML methods in anomaly detection. By systematically presenting this framework, we establish the foundational background necessary for the subsequent literature review.

A vital technique in the field of data analytics, anomaly detection plays an important role in various applications such as security, health and network monitoring [4]. Anomalies can indicate critical and often rare events that needs immediate attention. In the rapidly evolving field of cybersecurity, where it is vital for maintaining system integrity and preventing hostile activities, anomaly detection is of a great importance. Anomaly detection is a fundamental component of IDS in cyber security. These systems are able to monitor network traffic and system activity to identify unusual actions that may indicate a security breach. For example, an unexpected increase in data traffic might indicate a DDoS attack, where an attacker overwhelms a network with excessive traffic to disrupt services. Similarly, unusual login attempts from unfamiliar locations or devices could signal potential unauthorized access, requiring further investigation to prevent data theft or system vulnerabilities.

Anomaly detection is a highly important technology with diverse applications across various industries. Its ability to identify unusual events allows for the implementation of preventive measures, thereby it enhences safety, reliability, and efficiency in different sectors. As researches in this field enhences, the integration of advanced machine learning methods and accessible artificial intelligence techniques will be essential for improving the accuracy, reliability, and interpretability of anomaly detection systems. This will ensure their continued relevance and effectiveness in a world where data is essential

2.1. **API calls in anomaly detection.**

The API is a critical element of the operating system and encompasses a set of functions contained in specific libraries. Users utilize these functions to communicate with the operating system reflecting the behavior of various files [6]. In the realm of APIs, the term ”application” generally refers to any software that performs a function. An interface, in this context, acts as a service contract that enables two applications to communicate through requests and responses. Essentially, APIs serve as mechanisms that facilitate communication between two software components using specific protocols. API calls are programming interfaces utilized by applications to interact with each other [7]. During an API call, one server sends a request to another server's web interface via the Transmission Control Protocol (TCP). To make a request, three primary components are necessary: the Uniform Resource Identifier (URI), headers, and the request body. While each API may use a distinct combination of these components, communicate in a different format, or require varying data, the request generally follows the Hypertext Transfer Protocol (HTTP) message structure. APIs provide analysts with a critical foundation for examining a program's behavior and functionality, particularly when direct reverse analysis is challenging. This foundation aids in detecting anomalies within the system's behavior. A system call, on the other hand, is a request made by a program to the kernel for a specific service. Analyzing the trace of such calls can reveal the behavior of the process. These traces are instrumental in classifying the process as either normal or malicious [8].

3. MACHINE LEARNING: FUNDAMENTALS, DEFINITIONS AND CATEGORIZATION

Anomaly detection using ML techniques with API calls is of significant importance in various domains, particularly in cybersecurity, network monitoring, and application performance management. API calls are a valuable source of data that offers detailed information into system behavior and interactions. Unusual patterns and deviations in API call data can be identified by using ML techniques, which may indicate potential security breaches, system problems, or performance issues. This approach simplifies proactive measures, enhancing system reliability, security, and efficiency. Various ML techniques are used for anomaly detection with API calls. Some of these techniques are; supervised, unsupervised, and semi-supervised learning methods. Supervised learning models are trained on labeled datasets that allows them to recognize known anomalies. Unlike supervised methods, unsupervised methods uncover hidden patterns in the data to determine anomalies without labeling. Semi-supervised approaches use both labeled and unlabeled data and improve detection performance by combining both methods. Using these ML techniques enables the analysis of API call data, providing strong and reliable anomaly detection. This approach ensures that systems can reduce issues effectively, maintaining optimal performance and security. Figure 2 illustrates classification of ML methods under varying learning techniques.

3.1. **Logistic Legression.**

Logistic regression (LR) employs the sigmoid function to calculate probability values and perform classification tasks. The sigmoid function produces outputs ranging from 0 to 1. Samples with probability values below 0.5 are classified as belonging to the negative class, while those with values of 0.5 or higher are assigned to the positive class. 3.1 [9].

3.2. **Support Vector Machines.**

SVM are supervised learning algorithms frequently utilized for both binary and multiclass classification tasks. SVM operates by mapping input data points into a high-dimensional space and constructing a hyperplane that is one dimension less to distinguish between different groups of data points [9]. The main objective of SVM is to identify a hyperplane to optimally separate the data into two distinct clusters by maximizing the distance between them. When a linear separation is not possible, a technique known as kernel cheating is used (Muhammad and Yan, 2015). Widely utilized kernel functions include Gaussian, radial basis function (RBF) and polynomial kernels. The most significant advantage of SVM is the ability to avoid overfitting and its non-probabilistic nature [10].

3.3. **Naive Bayes.**

Naive Bayes (NB) classifiers are straightforward probabilistic models [10]. The term ”naive” stems from the assumption that all input features are independent and uncorrelated with each other. This algorithm is fundamentally based on Bayes' theorem and computes the probability of each class for a given set of input features [11].

3.4. **Random Forest.**

Random Forest (RF) is an ensemble learning method composed of multiple decision trees. Each tree in the model employs a decision tree algorithm to choose a subset of features. After the forest is formed using the RF technique, new data is classified by passing it through each tree. The trees vote for the class they believe the instance belongs to, and the forest selects the class with the highest number of votes. RF

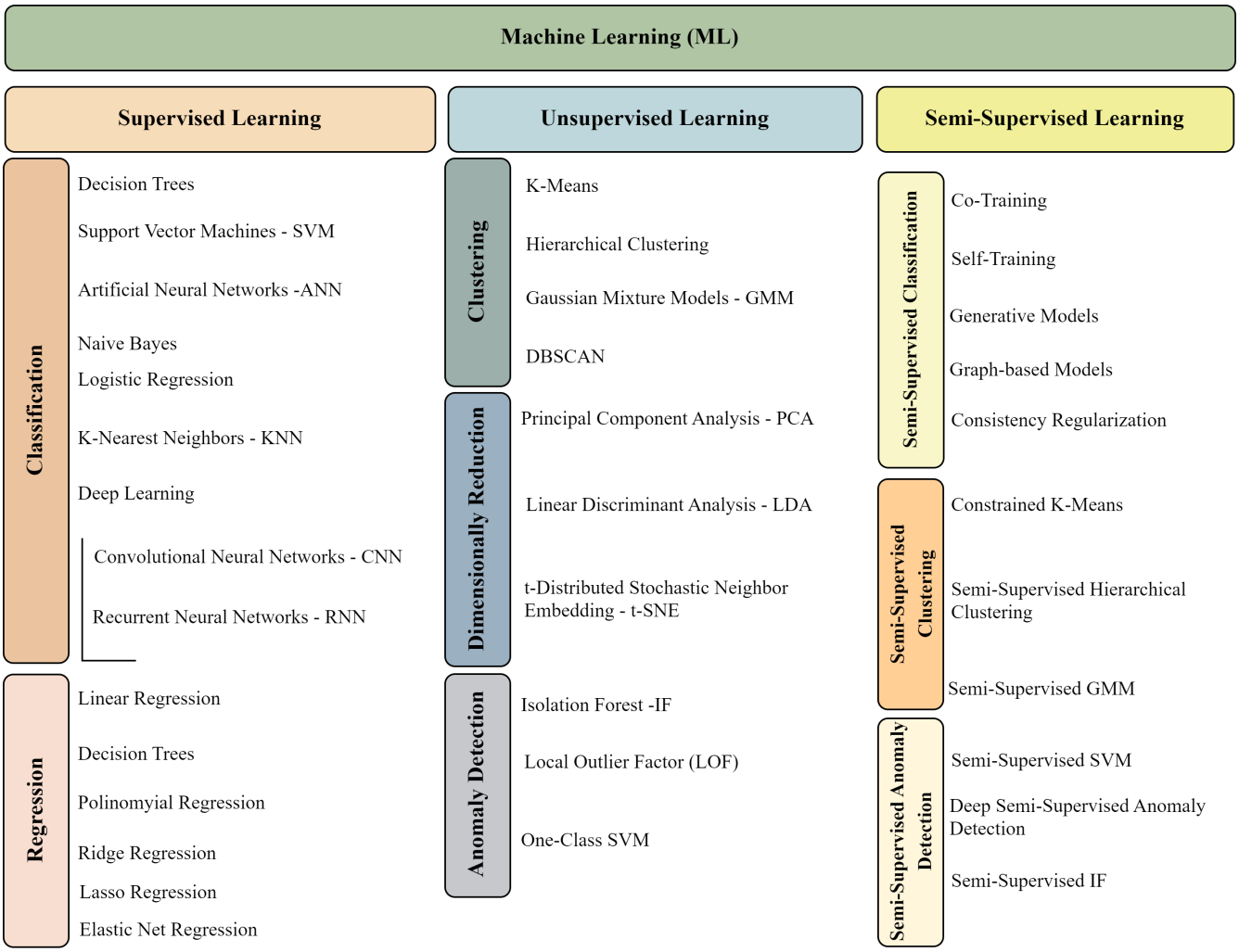

FIGURE 2. Classification of the ML methods

is particularly valued for its noise robustness and lower susceptibility to overfitting compared to other algorithms [12].

3.5. **K-Nearest Neighbor.**

The K-Nearest Neighbor (KNN) algorithm is one of the most basic supervised learning techniques used to classify data into distinct categories [10]. Being non-parametric and probabilistic, KNN is suitable for classification tasks where there is no prior knowledge about the data distribution. The algorithm classifies a new sample based on the majority vote of its k nearest neighbors, determined by a similarity measure (distance). However, its computational complexity increases with data size, often necessitating dimensionality reduction techniques to mitigate the curse of dimensionality before applying KNN [13].

3.6. **Deep Neural Network.**

DNN is one of the most commonly used methods in ML. Unlike traditional ML techniques, it excels when processing large datasets. A notable feature of DNN models is their deep architecture consisting of multiple hidden layers [14]. The design of a DNN reflects the working logic of the human brain and typically consists of an input layer, several hidden layers and output layers. As a result, DNN models encompass many units, making them suitable for classifying non-linear and complex data. However, training DNN models requires more time than other methods due to their complex model structure and size [14].

3.7. **Long Short-Term Memory).**

The LSTM model enhances the capabilities of the RNN (Recurrent Neural Network) model. RNNs, as deep learning models, encounter the vanishing gradient issue when network layers increase, which hampers the network's ability to learn from previous information. LSTMs address this problem with a complex recurrent unit that employs a gating mechanism to regulate information flow. These models incorporate memory cells with fixed weights and self-contained recurrent nodes, enabling the retention of state values over long periods and allowing gradients to pass through numerous time steps without diminishing [15].

Traditional RNNs process sequential data inputs with forward recursive computation, chaining neurons together to integrate past information with current inputs, effectively functioning as a memory to handle time sequences. However, due to their limited memory capacity, RNNs retain less historical information while preserving more recent data. During training, information distortion occurs as it passes through numerous iterative loops, leading to gradient fading.

In contrast, the LSTM model enhances traditional neurons with memory cells, significantly improving the network's information transfer and processing capacity. These memory cells effectively store historical data, and the input gates within the cells autonomously manage the retention time of values, enabling better prediction of crucial information [16]

3.8. **Convolutional Neural Network.**

CNNs have become a cornerstone of deep learning, especially in visual tasks, due to their outstanding performance in computer vision applications such as object recognition, detection, and segmentation. These networks have not only achieved top results in various tasks but also matched human-level performance in recognizing visual objects and in critical medical applications [17].

CNNs are a favored deep learning technique because of their ability to model complex, non-linear relationships. They are more efficient than traditional DNNs and excel at learning abstract image features, which makes them particularly suitable for image processing. A CNN with sparse connections and shared weights has significantly fewer parameters compared to a fully connected network of similar size. The architecture of CNNs consists of three main layers: convolutional layers, pooling layers, and fully connected layers. Unlike standard ANNs where each hidden layer has distinct weights, inputs, and outputs, CNN neurons operate on two-dimensional planes for inputs and outputs, using feature maps (kernels) as weights. Convolutional layers extract features from the input images, organizing the outputs into two-dimensional planes known as feature maps. Each layer's plane is formed by combining outputs from the previous layer. As features are passed to higher layers, their size decreases in proportion to

the filter size used in the convolution and pooling layers, while the number of feature maps increases to improve feature extraction and classification accuracy. Pooling layers typically follow convolution layers. After convolution and pooling, the extracted features are converted into a vector for classification via fully connected layers, which are recognized for their high performance [18].

CNNs, inspired by the visual processing mechanisms of the human brain, are a type of multi-layer perceptron within the feed-forward neural networks category. They are designed to automatically and adaptively learn spatial feature hierarchies through backpropagation using key components such as convolution layers, pooling layers, and fully connected layers [19]. The structure of a CNN includes an input layer, an output layer, and several hidden layers, which can be convolutional, pooling, or fully connected [20].

4. ANOMALY DETECTION USING API CALLS WITH ML MODELS

In this section, we systematically summarize and analyze the studies obtained in response to the research questions posed earlier. This systematic review forms the core of our investigation, building on the general concepts, fundamentals, and definitions provided in the preceding sections. By examining the datasets and data features most commonly used in ML methods for anomaly detection utilizing API calls, we aim to identify the prevalent data sources and attributes that drive effective anomaly detection. Furthermore, we evaluate how the performance of these ML methods is assessed, focusing on the metrics that are most commonly employed in the literature. Lastly, we identify the most effective ML methods for anomaly detection using API calls and explore the characteristics that make these methods particularly successful. Through this structured approach, we not only synthesize existing research but also provide a comprehensive comparison and analysis, offering valuable insights into the state-of-the-art techniques in this domain. Table 4 highlights general contributions of the investigated papers.

TABLE 4. System Call Anomaly Detection Techniques Overview

Ref.	Contribution	Method
[21]	A deep learning approach for detecting anomalies in power grid systems Detection of anomalies in power measurement sets Generation of power system data using a customized module Addressing imbalanced dataset issues in anomaly detection.	Customized real-time cluster formation Anomaly detection and identification using DNN
[22]	Automated system call collector for anomaly detection Large-scale dataset of system calls in Linux kernels Deep learning-based anomaly detection using various models like CNN, LSTM, etc.	Real-time collection of system call logs from Linux systems Anomaly detection using CNN/RNN, LSTM, WaveNet, and ECOD
[23]	Open-source anomaly detection and alert framework for applications Anomaly detection considering different parameters in system calls IDS framework applied to container platforms using machine learning methods	Data reading and writing using Elasticsearch API Anomaly detection using Kubernetes, CNN, and cuDNN
[24]	Frequency and LSTM-focused anomaly detection framework for cloud systems Anomaly detection using LSTM architecture with system calls Attack-based classification based on the frequency of past call sequences	LSTM-based architecture for attack classification Feature selection using TF-IDF and attack classification using NN
[25]	Anomaly detection by analyzing behavioral units Threat analysis at the system call and API levels focusing on critical behaviors	Anomaly detection using a multi-level DL model Attack prevention using a multi-level transformer-based model
[8]	Deep learning-based anomaly detection Improved training time efficiency with an enhanced neural architecture	Anomaly detection using LSTM and CNN with hyperparameter optimization
[26]	Analysis of program behaviors based on characteristic values (CV) Modeling program behavior by altering function parameters with CV sequences Creating a neural network vector using word embedding method and prototype design	Learning method based on CV values with LSTM-RNN
[27]	Real-time dataset analysis using APIs Anomaly detection using machine learning considering traffic flow through APIs	Anomaly detection and classification using STL and iForest
[28]	Design of anomaly detection algorithm for cloud systems Custom classification method for detecting faulty states Scenario design for various subsystems based on fault scenarios	Markov model for state probability estimation Modeling using PPM-C and VMM
[29]	Graph model for modeling state data GCN modeling with network traffic and system state data	Intrusion and anomaly detection using hybrid tGCN-KNN method
[30]	Anomaly detection using system calls for Linux environment End-to-end anomaly classification using machine learning with system logs	Feature vector transformation using Doc2Vec, RNN-AE, and RNN-DAE Anomaly detection using IF, LOF, and 1-SVM
[31]	Design of an ML-based detection system for anomalous pods in Kubernetes clusters Modeling the system using Linux kernel calls	Classification explanation using SHAM and LIME Machine learning implementation using decision trees, ANN, and EL
[32]	Feature-based anomaly detection and classification method	Anomaly detection using hierarchical clustering algorithms
[33]	Multi-anomaly detection system based on application behaviors Weighted graph representation based on system call numbers and frequencies IDS modeling using DNN with system calls	Three-component feature vector method for neural networks with D-MLP Anomaly detection using STIDE, text classification, and system call graph
[34]	Anomaly detection by reducing troubleshooting time for developers Collection of system calls during execution in a Linux environment	Collection of system calls using Toolkit Next Generation Classification using multi-class SVM Clustering using K-means and Dbscan
[35]	Anomaly detection for IoT networks with a customized NIDS method named Panop Feature extraction based on network-related device behaviors Real-time scenario representation using Raspberry Pi devices	Anomaly detection using Kitsune and ANN
[36]	NLP implementation for program behavior analysis using BoSC Anomaly detection based on the sequence of system calls at a specific point in time	Anomaly detection using Cosine Similarity Algorithm (Co-Sim) based on NLP
[37]	ML-based methods for anomaly detection accuracy using normal and attack data	N-gram technique for creating feature vector Classification and prediction using SVM, LR, and KNN methods
[38]	Feature extraction method using system call names Low-cost feature extraction method applicable across different platforms	N-gram technique to convert system calls to frequency sequences Classification using IF, LOC, OCSVM, and KNN
[39]	Anomaly detection using LSTM framework Hybrid detection model using LSTM and unsupervised learning framework	Classification using OC-SVM, LSTM, and SVDD
[40]	Flexible anomaly detection system using system security logs Semantic feature extraction and threat behavior modeling for internal threats	Model using LSTM and GRU

4.1. **Prevalent Datasets and Key Features in API Call Anomaly Detection.**

In this section, we will identify and describe the most commonly used datasets and the key features that are leveraged in ML methods for anomaly detection utilizing API calls. We will provide an overview of the sources of these datasets, the nature of the data they contain, and the specific features that are extracted and used for training anomaly detection models. This analysis will help in understanding the data foundation upon which current research is built and highlight any gaps or opportunities for future dataset development.

In [8] a hybrid anomaly detection system based on deep learning techniques, aiming to enhance both accuracy and efficiency was proposed. The proposed system combines CNN and LSTM to achieve improved detection capabilities. The initial step involves inputting the raw sequence of system call traces into the CNN network to decrease the dimensionality of the traces. Subsequently, the reduced trace vector was passed to the LSTM network to understand the sequence of system calls and generate the final detection result.The hybrid model was implemented and trained using TensorFlow-GPU, and its performance was evaluated on the ADFA-LD dataset. The ADFA-LD host-based intrusion detection dataset was generated by the ADFA. This dataset records Linux system calls, which facilitate communication between user and kernel modes through standard interfaces provided by the Linux kernel. Every system call on the sequence trace has a unique identifier number. The host was designed to profile a recent Linux server that logs system call traces during a specific time period. ADFA-LD dataset was used in the first phase. In the second phase, stratified sampling was applied and the dataset was divided into training, validation, and test. In the second phase, a hybrid deep learning model utilizing CNN and the LSTM algorithm was trained. The CNN consists of two layers: the convolution layer and the pooling layer. The convolution layer applies convolution procedures to the input picture to extract significant features, while the pooling layer decreases image dimensionality and deals with data nonlinearity. ReLu activation function was employed. Finally, a hybrid DL-based CNN with LSTM was presented to detect abnormalities in sequential system calls. The CNN extracted significant features, and the LSTM learned the sequence patterns from the reduced data. Experiments reveal that the suggested technique displayed reduced training time and better anomaly detection rates, hence lowering false alarm rates.

In [41], a novel anomaly recognition and detection framework named AnRAD inspired by biological systems, which utilizes probabilistic inferences was proposed. It investigates feature dependencies and introduces a self-structuring approach that learns an efficient confabulation network from unlabeled data. This network enables fast incremental learning, continuously refining its knowledge base with streaming data. Comparative analysis with existing anomaly detection methods demonstrates competitive detection quality. Moreover, the AnRAD framework leverages parallel processing capabilities, yielding significant speed enhancements when implemented on graphic processing units and Xeon Phi coprocessors compared to sequential execution on standard microprocessors. Versatility of the framework enables real-time processing of concurrent data streams across various knowledge domains, making it applicable to large-scale problems characterized by multiple local patterns. The proposed methodology incorporates the principles of the confabulation theory within a hierarchical cognitive architecture, enabling flexible network configurations tailored to specific applications. Leveraging the computational power of GPUs

and Xeon Phi processors, the notable speed enhancements through both fine-grained and coarse-grained parallelization methods were achieved.

In [24], a novel intrusion detection framework was introduced, which can identify both known and unknown attacks through system call sequence analysis. This framework examines the system call sequences of VMs using a hybrid model that combines LSTM networks with anomaly detection techniques based on system call frequency. The effectiveness of this framework was validated using the ADFA-LD. System call traces are collected and stored as a dataset for offline training, necessitating preprocessing to remove extraneous information and retain only system call sequences. These sequences are then encoded with unique identifiers and labeled as normal or malicious. Frequency-based methods like Bags of n-grams are employed to generate a Feature Vector Matrix (FVM) from the processed traces. These vectors, representing the frequency of distinct n-grams, are stored in a feature-vector database file. Experimental results showed that this framework outperforms existing models in accuracy and has a lower false positive rate.

In [38], a new feature extraction method designed to derive features that are independent of system call names, making the samples directly applicable to cross-platform scenarios. The method converts system call sequences into frequency sequences of n-grams and then extracts a fixed number of statistical features from these sequences. These features are calculated based on the frequency sequences rather than the direct system call sequences, and are used to train a one-class classification model for anomaly detection. The study utilized the ADFA-LD, ADFA-WD, and NGIDS-DS datasets, employing anomaly detection algorithms like Isolation Forest, LOF, OCSVM, and kNN. The method was compared with other feature extraction techniques, such as Bag of System Calls, tf-idf, and subsequence frequency. Even though the proposed method did not always achieve the highest AUC on the same platform, it generally outperformed other methods, especially in cross-platform scenarios.

Lv et al. developed a system-call sequence-to-sequence prediction model by semantically modeling system calls [42]. This model predicts future system calls to monitor system states and detect attack behaviors. An end-to-end system call prediction model was built to predict subsequent system calls based on traces generated during malicious process execution. The RNN model was used to ensure the generation of semantically reasonable sequences. The model was evaluated using the ADFA-LD dataset, which contains traces from both normal and intrusion attempts. Performance was assessed using the BLEU score and Euclidean distance between encoded semantic vectors, with TF-IDF used for sequence similarity evaluation. The predicted sequences, when combined with known system call traces, significantly enhanced intrusion detection performance across various classifiers.

System calls are the primary means for applications to communicate with the Operating System (OS), making them vital for Host-based Intrusion Detection Systems (HIDS). In [22], several existing datasets are outdated, prompting the introduction of a large-scale dataset specifically for anomaly detection in the Linux kernel. The dataset, named DongTing, comprises 85 GB of data, including 18,966 system call sequences labeled as normal or anomalous. It encompasses over 200 kernel versions and 3600 bug-triggering programs from the past five years. Cross-dataset evaluation demonstrated that models trained on this dataset exhibited superior generalization capabilities. The dataset was divided into training, validation, and testing subsets for training deep learning models to detect anomalies in Linux kernels. Four

deep learning models—CNN/RNN, LSTM, WaveNet, and ECOD—were evaluated, showing that models trained on this dataset achieved the highest generalization scores and better performance when trained on abnormal data. This framework significantly reduces the time required to produce the dataset.

In [30], an end-to-end strategy was presented for identifying abnormal activities, merging sequential information preservation through log embedding techniques with anomaly detection algorithms based on ML. Unlike current ML methods for system anomaly detection, which rely on domain experts to extract relevant features from log data, the proposed method converts raw log data into fixed-size continuous vectors regardless of length. These vectors are then utilized to train anomaly detection algorithms. This paper proposes a strong intrusion detection model designed specifically for Linux settings, combining sequential information-preserving log embedding techniques with anomaly detection algorithms. Doc2vec was used to convert system call traces of different durations into fixed-dimensional real-valued vectors, as are recurrent neural network-based auto-encoder (RNN-AE) and recurrent neural network-based denoising auto-encoder (RNN-DAE) approaches that keep sequential information. To validate the detection model, an experiment was carried out using the ADFA-LD dataset, which contains Linux system call traces. Three assessment measures were utilized to assess the performance of anomaly detection algorithms. The ROC was used to assess the performance of each model. In the studies, the performance of anomaly detection systems based on unsupervised learning was assessed across several attack types. After gathering a significant quantity of labeled data, a supervised classification model was trained and its performance was compared against anomaly detection techniques. Finally, the paper provides an unsupervised ML-based system anomaly detection framework that does not require labeled data for model training.

In [30] an end-to-end approach was proposed for detecting abnormal behaviors, integrating sequential information preservation through log embedding algorithms with anomaly detection algorithms based on ML. Unlike other ML models for system anomaly detection that rely on domain experts to extract meaningful features from log data, the proposed approach transforms raw log data into fixed-size continuous vectors regardless of their original length. In this work, a robust intrusion detection model was developed which was specially created for Linux environments, leveraging sequential information-preserving log embedding techniques alongside anomaly detection algorithms. To convert system call traces of varying lengths into fixed-dimensional real-valued vectors, Doc2vec was used, as well as recurrent neural network-based auto-encoder (RNN-AE) and recurrent neural network-based denoising auto-encoder (RNN-DAE) methods, which retain the sequential information. To validate the detection model, an experiment was conducted using the ADFA-LD dataset, which comprises Linux system call traces. In the experiments, firstly the performance of anomaly detection algorithms based on unsupervised learning was evaluated across different attack types. After gathering a significant quantity of labeled data, a supervised classification model was trained and its performance was compared against anomaly detection techniques.

4.2. **Performance Evaluation Metrics for Machine Learning-Based API Anomaly Detection.**

This section will focus on how the performance of ML methods for anomaly detection using API calls is evaluated. We will examine the most commonly utilized metrics, such as accuracy, precision, recall, F1 score, and area under the ROC curve (AUC). By analyzing these metrics, we aim to provide insights into

how researchers measure the effectiveness of their models, the challenges associated with each metric, and the contexts in which certain metrics are preferred over others. This will offer a comprehensive understanding of the evaluation landscape in this field. Figure 3 illustrates dataset and platform summary of the examined studies.

In [34], system calls based anomaly highlighting and detecting framework was proposed to guide developers regarding performance problems in data. The LTTng was used to collect data from the Linux kernel, applications, and libraries. A supervised learning method was utilized in order to manage large amounts of labeled data. In addition, the learning method was improved and modified as semi-supervised. Then, the feature vector was constructed considering the duration of the most significant call sequences. In the detection phase, an automated anomaly detection method was implemented as module-by-module. It was indicated that the proposed method ensures high accuracy and efficiency in large scale systems and distinguishes CPU and memory shortages as well as detecting normal behavior.

In [39], an unsupervised anomaly detection and innovative algorithms based on LSTM neural networks were developed. The proposed structure was started by processing variable length data sequences through an LSTM-based structure, resulting in fixed-length sequences. Decision functions for anomaly detection were then derived using One-Class Support Vector Machines (OC-SVMs) or Support Vector Data Description (SVDD) algorithms.The main contribution of the proposed work was indicated that the simultaneous training and optimization of LSTM architectural parameters as well as OC-SVM or SVDD parameters, facilitated by highly effective gradient and quadratic programming-based training methods.This study extends the unsupervised framework to semi-supervised and fully supervised settings. The resulting anomaly detection algorithms excel in processing variable length data sequences, particularly time series data, offering superior performance compared to conventional methods.

In [43], an anomaly detection model utilizing LSTM networks as well as intra- and inter-trace context vectors was proposed to overcome the challenge of online anomaly detection in CPSs. This dynamic approach allows both identified and unseen anomalies to be addressed while improving the understanding of kernel event execution contexts both horizontally and vertically. A deep context-aware architecture was introduced for anomaly detection in semi-structured sequences specifically focused on system calls or kernel events. Furthermore, the importance of using a context-based attention layer to extract rich semantics that help identify non-linear high-dimensional relationships present in syslog sequences was emphasized. In the simulation, a custom dataset generated by existing work was used. Two parameters were relevant to analyze the complexity of the model. Finally, the proposed approach characterizes the behavior of the system through online execution trace analysis using recurrent neural networks. Experiments show that the proposed model provides effective and robust results in anomaly detection using system call sequences.

In [44], a state summarization and and nested-arc hidden semi-Markov model (NAHSMM) model was proposed to model dynamic usage behavior and identifying anomalies for cloud servers. The model was designed to control the propagation of system call sequences and less usage transitions. In addition, the proposed detection algorithm was generated by integration of NAHSMM and state summarization. The system calls in varying length were summarized using these methods and the NAHMM was utilized to fit time sequences. As fundamental, the proposed system was constructed as a mathematical model. The

FIGURE 3. Comparison of the studies in terms of dataset and platform

Reference	Dataset	Number of Instances	Platform
[34]	Synthetic	946,000 - 3,800,000	Not provided
[25]	Firefox-SD, ADFA-LD, PLAID, Synthetic	18,966	Intel Xeon E5-2640 V4, DDR4 ECC 2400 MHz, 128G memory, 4 GPU
[35]	Synthetic	1,551,006	Linux OS, GeForce RTX 3060
[22]	ADFA-LD	Not provided	Ubuntu 18.04 OS, i5 processor, 8 GB RAM, 500 GB HDD, Xen 4.6
[36]	AndroCT	35,974+	Windows 10 OS, AMD Ryzen 7 5800 8-Core Processor, 3.40 GHz, NVIDIA 3060, 32.0 GB RAM
[8]	ADFA-LD	Not provided	Not provided
[37]	Synthetic	Not provided	Not provided
[38]	Synthetic	5,748 - 15,000,000+	Not provided
[33]	Not provided	Not provided	Intel Xeon server (E5-2630L v3 @ 1.80 GHz), 16 GB RAM, 150 GB, Linux CentOS v7.0
[32]	CSE-CIC-IDS2018, UNSW-NB15, Synthetic	257,673	Intel Xeon (Cascade Lake) Platinum 8269 2.5GHz/3.2GHz 4-core CPU, 8GB RAM
[39]	Synthetic	Not provided	Not provided
[40]	ADFA-LD	5,206	Intel (R) Core (TM) i7-7700K CPU 4.20GHz, NVIDIA GTX 1080Ti 11GB RAM, 32G RAM
[41]	Synthetic	Not provided	Intel Core i5-6200U CPU 2.3 GHz and 2.4 GHz, 8GB RAM
[42]	DARPA, UNM, ADFA-LD	2,766 for UNM, others session-based variable	Intel Core i7-4790 CPUs, 16 GB RAM
[27]	Synthetic	Not provided	Intel Core i7 4 GHz, 8 CPU, 32 GB
[26]	Synthetic	50,000	Not provided
[43]	ADFA-LD	Not provided	Not provided
[23]	ADFA-LD, ADFA-WD, NGIDS-DS	5,951 for ADFA-LD, 7,725 for ADFA-WD, 3,070 for NGIDS-DS	Not provided
[28]	Occupancy, HHE, Http, Alcoa	Not provided	i5-6400, 2.7 GHz CPU, 16 GB RAM
[31]	CMU CERT v6.2	200,000	Not provided

main concept was to use structured numerical models that include summarization of states to better understand the behavior of sequences of system call identifiers. In training and testing phases, IXIA Perfect Storm was used to collect data. As final, the effectiveness of the proposed model in accurately detecting anomalies within machine operating systems was highlighted . By leveraging descriptive features and a streamlined structural framework, the model achieves this with fewer parameters, leading to significant reductions in computational complexity and storage needs. Although the focus is on modeling system call identifier sequences from servers in cloud environments, the method shows promise for application in classifying network traffic and identifying anomalous human behavior.

In [40], a threat detection model was proposed implementing the Word2vec-based approach. The possibility of suspicious behavior was assessed by leveraging Word2vec model trained on a corpus of various security logs.The method consists of three main components: log2text, text2corpus, and anomaly detection. The log2text component standardizes events from diverse security logs into uniform text format. These texts are then merged, sorted chronologically, and processed into a corpus by the text2corpus component. Finally, the anomaly detection component trains a Word2vec model on this corpus to compute the probability of a specific behavior given an event, denoted as p(behavior—user). Events exceeding a certain threshold are flagged as suspicious, potentially indicating malicious insider activity if multiple suspicious events are associated with a user. The dataset was divided into smaller data and selected specified security logs. The TPR and FPR were used as success metrics to determine performance of the anomaly detection. The proposed study was compared with existing ones in terms of cost and complexity.

In [29], a novel anomaly and intrusion detection models were designed. The network data was represented as a graph structure to identify relation features between samples. The graph structure was constructed as a triplet graph CNN and it was used to detect anomalies in the system. In addition Graph Convolutional Network (GCN) was modeled and CSE-CIC-IDS2018 and UNSW-NB15 datasets were used to monitor performance of the model. IThe dataset includes varying attack types and subtypes. In the training phase, the value of the neighborhood number K is modified to achieve optimal detection accuracy and the KNN model was utilized to complete learning. A small traffic data sample was used in the integration phase of the proposed tGCN-KNN. The experiments were performed for a varying number of samples under tGCN and tCNN learning models. As final, according to the comparison of three methods, it was indicated that tGCN-KNN outperforms tCNN and CNN in terms of accuracy.

In [28], a novel approach named fault injection analytics was introduced for analyzing data from fault injection experiments. This approach integrates distributed tracing to gather raw failure data and employs unsupervised ML to identify failure modes within the injected system. The primary objective is to aid human analysts in identifying failure modes more efficiently, especially when managing large volumes of data from fault-injection experiments. A new anomaly detection algorithm was proposed within this framework, designed to pinpoint unusual events in fault injection experiments. This algorithm is resilient to noise inherent in cloud systems, which can stem from non-deterministic timing and event ordering. It is also efficiently trainable with a small set of fault-free executions of the distributed system, leveraging a variable-order Markov Model. The proposed method treats the cloud-computing system as a collection of black-box communicating components, eliminating the need for prior knowledge about their internal workings. Unsupervised ML is applied to execution traces to uncover patterns of failure.

The method detects shared symbols among sequences by calculating the Longest Common Subsequence (LCS) of the sequences. The LCS represents a subset of symbols present in both sequences in the same order, obtained by minimally eliminating symbols from the original sequences. Using this probabilistic model, the method effectively detects anomalies in noisy execution traces, reducing false alarms while maintaining the detection of true anomalies. Results indicate that clustering achieves high accuracy under various conditions.

4.3. **Top Performing Machine Learning Methods for API Call Anomaly Detection: Characteristics and Effectiveness.**

In this section, we will identify the ML methods that have proven to be the most effective for anomaly detection using API calls. We will explore the characteristics that make these methods stand out, such as their ability to handle high-dimensional data, robustness to noise, computational efficiency, and interpretability. By comparing and contrasting these methods, we will highlight their strengths and weaknesses, providing a clear picture of the current best practices and innovative approaches in the field. This analysis will also suggest directions for future research and potential improvements. Table 5 summarizes method and platform based summary of the summarized studies.

In [36], an anomaly detection approach was developed utilizing NLP. The Bags of System Calls (BoSC) was used to analyze application activity on Windows virtual machines operating under the Xen hypervisor. System call traces were retrieved from both regular (benign) and malware-affected (malicious) apps utilizing virtual memory introspection. The retrieved system call sequences were preprocessed to produce valid sequences by filtering and arranging duplicate system calls. The behavior of these sequences was then investigated using NLP-based anomaly detection algorithms. The Cosine Similarity Algorithm (Co-Sim) was used to identify malicious processes on the Virtual Machine (VM). Furthermore, the Point Detection Algorithm was employed to identify the point of breach in the system call sequence. Virtual Machine Introspection (VMI) was identified as the most flexible approach for detecting, monitoring, and evaluating malware threats at the hypervisor level. In the feature extraction step, a technique called angle similarity, which is comparable to text classification for anomaly detection, was applied. In this method, a sequence of system calls was treated as a document, but individual system calls were treated as words. According to the findings, the suggested algorithms have a high detection accuracy for spotting abnormalities.

In [25] an innovative method for detecting anomalies with adversarial robustness was proposed to address vulnerabilities in existing systems against perturbation attacks. The focus of the proposed approach was on analyzing behavior units, which encapsulate representative semantic information of local behaviors to enhance the resilience of behavior analysis. A multilevel deep learning model was leveraged to understand overall semantics and contextual relationships among behavior units, effectively mitigating perturbation attacks targeting both local and large-scale behaviors. Moreover, versatility was demonstrated across different types of behavior logs, including low-level (e.g., API) and high-level (e.g., syscall) logs. The approach assumed limited attacker knowledge and incorporated threat modeling to address potential modifications to behaviors by attackers. Initially, key behavioral actions were identified from behavior sequences, followed by the use of the Longest Common Subsequence (LCS) algorithm to extract related segments that bolstered model robustness. Finally, multilayer transformer models were

implemented for feature extraction from behaviors, enabling behavior classification to determine whether a system sequence was abnormal or normal.

In [26] a novel approach was proposed where program behavior, considered as a sequence of computational steps, was represented by a single Characteristic Value (CV) rather than individual input values. This CV sequence was used as input for neural networks, resulting in improved efficiency in modeling program behavior. Multiple LSTM-RNN models were employed to reduce the neural network's input space, marking a significant advancement in program behavior modeling. The primary focus of the proposed model was on modeling program behavior using CV sequences. These sequences were utilized to represent program behavior after execution steps and were integral to the anomaly detection phase based on the constructed CV models. A custom dataset was employed to evaluate the proposed model, comparing its performance in terms of detection accuracy against existing methods.

In [33], a model for an intrusion detection system was developed that integrated various detection techniques into a single system, aiming to achieve a comprehensive view of application behaviors. The paper proposed a novel modified system calls graph that was designed to integrate and consolidate information from different techniques within a unified data structure. A deep neural network was employed to combine the results from different detection techniques used in the global model. The effectiveness of this approach was validated using three datasets of varying complexity levels. The key contribution of this study was the integration of multiple intrusion detection techniques into a unified system, leading to improved detection accuracy. The architecture of the proposed system was characterized by two main stages: detection and integration. In the detection stage, multiple intrusion detection techniques were utilized concurrently. The study employed several datasets, including DARPA, UNM, and ADFA-LD, each chosen for its distinct complexity levels. Results demonstrated significant improvements in detection accuracy compared to using individual techniques, with higher detection rates and reduced false positives achieved by the proposed model.

TABLE 5. Method and platform based summary of the methods

Reference	ML Method	Applied platform	Success Metrics
[21]	DNN	General	Accuracy, loss rate, TNR, precision, F1-score, FPR, sensitivity, FNR, G-mean
[22]	CNN/RNN, LSTM, WaveNet, ECOD	Linux Kernel	FPR, F1-score, time efficiency, AUC, TPR
[23]	DNN, CNN	Container Platforms (Kubernetes cluster)	Accuracy, NPV, coverage, sensitivity, FPR, F1-score, ROC
[24]	LSTM	Linux, Cloud	Accuracy, loss rate, FPR, F1-score, sensitivity
[25]	Multi-layer DL	Android	Precision, F1-score, ROC
[8]	LSTM, CNN	Linux	Accuracy, loss rate, detection rate, FAR, training time
[26]	LSTM-RNN	General	Accuracy, detection rate, AUROC, AUPR, CPU cycle count, complexity, memory usage
[27]	iForest	Smart Traffic System with Sensor Devices	Detection rate
[28]	Not provided	Cloud Systems	FAR, time efficiency, TPR, computational cost
[29]	tGCN-KNN	General	Accuracy, precision
[31]	DT, ANN	Container Clouds	Loss rate, F1-score, precision, recall
[30]	1-SVM, LOF, iForest	Linux	AUROC, AUC
[32]	Hierarchical Clustering	Kernel Events, Operating System (OS)	Precision, FPR, complexity, FNR
[33]	D-MLP	Linux	Detection rate, FPR, ROC
[34]	SVM, K-means, Dbscan	Linux	Accuracy, time efficiency
[35]	ANN	General, IoT	Accuracy, FPR, TPR
[37]	SVM, LR, KNN	Linux	Accuracy, AUC, ROC
[38]	IF, LOF, OC-SVM, KNN	Linux, Windows	FPR, computational cost, TPR
[39]	OC-SVM, LSTM, SVDD	General	AUC, ROC
[40]	Word2vec	General	FPR, TPR, computational cost

5. OPEN PROBLEMS AND CHALLENGES

While anomaly detection using API calls is very significant and quite functional, there are some open issues that have not been resolved by researchers and application developers. Some of these challenges are as follows:

- **Imbalanced datasets:** The performance of ML methods is significantly affected by the imbalanced datasets used for anomaly detection. Typically, the volume of data representing anomalous behavior is significantly smaller than that representing normal system behavior. This imbalance can lead to inadequate performance of ML methods in both the training and testing phases, resulting in inadequate success metrics. Furthermore, labeling the available dataset is often time-consuming and resource-intensive, especially when dealing with large and diverse data volumes.
- **High data volume in processing:** ML models applied to high-volume datasets may struggle to perform efficiently under heavy loads. The scalability of these models is directly affected by the increasing number of API calls. The literature shows that performance issues arising from high data volumes in distributed and cloud systems represent a critical area for improvement.
- **Open source datasets:** Datasets available in open sources often contain sensitive information within API calls, raising concerns about privacy violations. Therefore, when creating datasets related to application and system behavior, it is crucial to prioritize privacy and data security.
- **Real-time Processing:** The ability of ML models to detect anomalies in real-time remains a significant challenge. Real-time processing of data requires advanced algorithms and infrastructure that can handle large-scale, high-speed data streams without compromising accuracy or speed.
- **Adaptability to Emerging Threats:** Anomaly detection systems must constantly adapt to new and evolving threats. Static models can quickly become obsolete, requiring the development of adaptive learning techniques that can update and evolve in response to new data and threat patterns.
- **Explainability and Interpretability:** The black box nature of many ML models poses a problem for understanding and interpreting results. Developing methods to make anomaly detection models more transparent and interpretable is crucial for their practical application and reliability.
- **Integration with Existing Systems:** It is often difficult to seamlessly integrate anomaly detection systems with existing IT infrastructure and workflows. Ensuring compatibility and minimal disruption to existing processes requires advanced integration strategies and tools.

6. CONCLUSIONS

In this paper, a systematic literature review on anomaly detection using ML methods with API calls is presented by providing a systematization of information. A research methodology is established by selecting appropriate search keywords and the searching sentences are generated with these keywords. Common databases are used in advanced mode to use generated searching terms. Research questions are determined and inclusion and exclusion criteria are defined to filter the studies according to the focus of the literature. Over 30 research papers are summarized and compared based on different criteria. Fundamental concepts related to API calls, machine learning fundamentals, and the scope of our review

are summarized to provide a foundation. Through systematic research and analysis, it is obtained that models such as KNN, SVM, LSTM, and CNN are frequently used for anomaly detection with API calls. Additionally, open-source datasets like ADFA-LD, DARPA, and UNM are generally preferred for classification and detection. It is also observed that custom datasets are often created using various tools from operating systems like Linux. Metrics such as accuracy, F1-score, recall, and precision are used to measure the performance of the models in the studies.

DECLARATIONS

- **Conflict of Interest:** The authors are not affiliated with any entity that has a direct or indirect in the subject matter covered in this paper.

REFERENCES

[1] S. Garg, S. Batra, A novel ensembled technique for anomaly detection, International Journal of Communication Systems 30 (11) (2017) e3248.

[2] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, N. F. Samatova, Anomaly detection in dynamic networks: a survey, Wiley Interdisciplinary Reviews: Computational Statistics 7 (3) (2015) 223–247.

[3] M. Ahmed, A. N. Mahmood, M. R. Islam, A survey of anomaly detection techniques in financial domain, Future Generation Computer Systems 55 (2016) 278–288.

[4] D. Alsalman, A comparative study of anomaly detection techniques for iot security using amot (adaptive machine learning for iot threats), IEEE Access (2024).

[5] B. Jin, S. Sahni, A. Shevat, Designing Web APIs: Building APIs That Developers Love, " O'Reilly Media, Inc.", 2018.

[6] A. Almaleh, R. Almushabb, R. Ogran, Malware api calls detection using hybrid logistic regression and rnn model, Applied Sciences 13 (9) (2023) 5439.

[7] Y. Li, F. Kang, H. Shu, X. Xiong, Y. Zhao, R. Sun, Apiaso: A novel api call obfuscation technique based on address space obscurity, Applied Sciences 13 (16) (2023) 9056.

[8] F. Osamor, B. Wellman, Deep learning-based hybrid model for efficient anomaly detection, International Journal of Advanced Computer Science and Applications 13 (4) (2022).

[9] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, M. Stanley, A brief survey of machine learning methods and their sensor and iot applications, in: 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA), IEEE, 2017, pp. 1–8.

[10] I. Muhammad, Z. Yan, Supervised machine learning approaches: A survey, ICTACT Journal on Soft Computing 5 (3) (2015).

[11] I. Rish, et al., An empirical study of the naive bayes classifier, in: IJCAI 2001 workshop on empirical methods in artificial intelligence, Vol. 3, Citeseer, 2001, pp. 41–46.

[12] E. Min, J. Long, Q. Liu, J. Cui, W. Chen, Tr-ids: Anomaly-based intrusion detection through text-convolutional neural network and random forest, Security and Communication Networks 2018 (1) (2018) 4943509.

[13] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is "nearest neighbor" meaningful?, in: Database Theory—ICDT'99: 7th International Conference Jerusalem, Israel, January 10–12, 1999 Proceedings 7, Springer, 1999, pp. 217–235.

[14] H. Liu, B. Lang, Machine learning and deep learning methods for intrusion detection systems: A survey, applied sciences 9 (20) (2019) 4396.

[15] Y. Liu, X. Hao, B. Zhang, Y. Zhang, Simplified long short-term memory model for robust and fast prediction, Pattern Recognition Letters 136 (2020) 81–86.

[16] S. Yang, A. Jin, W. Nie, C. Liu, Y. Li, Research on ssa-lstm-based slope monitoring and early warning model, Sustainability 14 (16) (2022) 10246.
[17] J. Bernal, K. Kushibar, D. S. Asfaw, S. Valverde, A. Oliver, R. Martí, X. Lladó, Deep convolutional neural networks for brain image analysis on magnetic resonance imaging: a review, Artificial intelligence in medicine 95 (2019) 64–81.
[18] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.
[19] R. Yamashita, M. Nishio, R. K. G. Do, K. Togashi, Convolutional neural networks: an overview and application in radiology, Insights into imaging 9 (2018) 611–629.
[20] G. Yao, T. Lei, J. Zhong, A review of convolutional-neural-network-based action recognition, Pattern Recognition Letters 118 (2019) 14–22.
[21] A. Akagic, I. Džafić, Enhancing smart grid resilience with deep learning anomaly detection prior to state estimation, Engineering Applications of Artificial Intelligence 127 (2024) 107368.
[22] G. Duan, Y. Fu, M. Cai, H. Chen, J. Sun, Dongting: A large-scale dataset for anomaly detection of the linux kernel, Journal of Systems and Software 203 (2023) 111745.
[23] S. L. Rocha, F. L. L. de Mendonca, R. S. Puttini, R. R. Nunes, G. D. A. Nze, Dcids—distributed container ids, Applied Sciences 13 (9301) (2023) 9301.
[24] A. Chaudhari, B. Gohil, U. P. Rao, A novel hybrid framework for cloud intrusion detection system using system call sequence analysis, Cluster Computing (2023) 1–17.
[25] D. Zhan, K. Tan, L. Ye, X. Yu, H. Zhang, Z. He, An adversarial robust behavior sequence anomaly detection approach based on critical behavior unit learning, IEEE Transactions on Computers (2023).
[26] S. Ahn, H. Yi, H. Bae, S. Yoon, Y. Paek, Data embedding scheme for efficient program behavior modeling with neural networks, IEEE Transactions on Emerging Topics in Computational Intelligence 6 (4) (2022) 982–993.
[27] A. Karamanou, P. Brimos, E. Kalampokis, K. Tarabanis, Exploring the quality of dynamic open government data using statistical and machine learning methods, Sensors 22 (24) (2022) 9684.
[28] D. Cotroneo, L. De Simone, P. Liguori, R. Natella, Fault injection analytics: A novel approach to discover failure modes in cloud-computing systems, IEEE transactions on dependable and secure computing 19 (3) (2020) 1476–1491.
[29] Y. Wang, Y. Jiang, J. Lan, Intrusion detection using few-shot learning based on triplet graph convolutional network, Journal of Web Engineering 20 (5) (2021) 1527–1552.
[30] C. Kim, M. Jang, S. Seo, K. Park, P. Kang, Intrusion detection based on sequential information preserving log embedding methods and anomaly detection algorithms, IEEE Access 9 (2021) 58088–58101.
[31] R. R. Karn, P. Kudva, H. Huang, S. Suneja, I. M. Elfadel, Cryptomining detection in container clouds using system calls and explainable machine learning, IEEE transactions on parallel and distributed systems 32 (3) (2020) 674–691.
[32] O. M. Ezeme, A. Azim, Q. H. Mahmoud, Peskea: Anomaly detection framework for profiling kernel event attributes in embedded systems, IEEE Transactions on Emerging Topics in Computing 9 (2) (2020) 957–971.
[33] F. J. Mora-Gimeno, H. Mora-Mora, B. Volckaert, A. Atrey, Intrusion detection system based on integrated system calls graph and neural networks, IEEE Access 9 (2021) 9822–9833.
[34] I. Kohyarnejadfard, D. Aloise, M. R. Dagenais, M. Shakeri, A framework for detecting system performance anomalies using tracing data analysis, Entropy 23 (8) (2021) 1011.
[35] H. Kim, S. Ahn, W. R. Ha, H. Kang, D. S. Kim, H. K. Kim, Y. Paek, Panop: Mimicry-resistant ann-based distributed nids for iot networks, IEEE Access 9 (2021) 111853–111864.
[36] S. K. Peddoju, H. Upadhyay, J. Soni, N. Prabakar, Natural language processing based anomalous system call sequences detection with virtual memory introspection, International Journal of Advanced Computer Science and Applications 11 (5) (2020).
[37] Y. Shin, K. Kim, Comparison of anomaly detection accuracy of host-based intrusion detection systems based on different machine learning algorithms, International Journal of Advanced Computer Science and Applications 11 (2) (2020).

[38] Z. Liu, N. Japkowicz, R. Wang, Y. Cai, D. Tang, X. Cai, A statistical pattern based feature extraction method on system call traces for anomaly detection, Information and Software Technology 126 (2020) 106348.
[39] T. Ergen, S. S. Kozat, Unsupervised anomaly detection with lstm neural networks, IEEE transactions on neural networks and learning systems 31 (8) (2019) 3127–3141.
[40] L. Liu, C. Chen, J. Zhang, O. De Vel, Y. Xiang, Insider threat identification using the simultaneous neural learning of multi-source logs, IEEE Access 7 (2019) 183162–183176.
[41] Q. Chen, R. Luley, Q. Wu, M. Bishop, R. W. Linderman, Q. Qiu, Anrad: A neuromorphic anomaly detection framework for massive concurrent data streams, IEEE transactions on neural networks and learning systems 29 (5) (2017) 1622–1636.
[42] S. Lv, J. Wang, Y. Yang, J. Liu, Intrusion prediction with system-call sequence-to-sequence model, IEEE Access 6 (2018) 71413–71421.
[43] O. M. Ezeme, Q. H. Mahmoud, A. Azim, Dream: deep recursive attentive model for anomaly detection in kernel events, IEEE Access 7 (2019) 18860–18870.
[44] W. Haider, J. Hu, Y. Xie, X. Yu, Q. Wu, Detecting anomalous behavior in cloud servers by nested-arc hidden semi-markov model with state summarization, IEEE Transactions on Big Data 5 (3) (2018) 305–316.