Yuzuncu Yil University
Journal of the Institute of Natural & Applied Sciences

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

Research Article

Enhancing Network Security: A Comprehensive Analysis of Intrusion Detection Systems

Murat KOCA*[1], İsa AVCI[2]

[1]Van Yuzuncu Yil University, Faculty of Engineering, Department of Computer Engineering, 65080, Van, Türkiye
[2]Karabuk University, Faculty of Engineering, Department of Computer Engineering, 78000, Karabuk, Türkiye
Murat KOCA, ORCID No: 0000-0002-6048-7645, İsa AVCI, ORCID No: 0000-0001-7032-8018
*Corresponding author e-mail: muratkoca@yyu.edu.tr

Article Info

Received: 07.09.2024
Accepted: 26.09.2024
Online December 2024

DOI:10.53433/yyufbed.1545033

Keywords
CBPNN,
Cyber security,
FBPNN,
Intrusion detection systems (IDS),
Logistic regression,
Machine learning

Abstract: Given the increasing complexity and progress of intrusion attacks, effective intrusion detection systems have become crucial to protecting networks. Machine learning methods have become a potential strategy for identifying and reducing such attacks. This paper has conducted a comprehensive analysis of intrusion detection using machine learning methodologies. The aim is to thoroughly examine the current state of research, identify the barriers, and highlight potential solutions in this field. The study begins by analyzing the importance of intrusion detection and the limitations of traditional rule-based systems. Afterward, it explores the underlying principles and concepts of machine learning and how they are practically applied in the field of intrusion detection. This paper provides a comprehensive analysis of different machine learning algorithms, such as decision trees, neural networks, support vector machines, and ensemble methods. The primary objective of this study is to assess the effectiveness and limitations of employing these techniques for identifying various forms of intrusions. Three algorithms are used to classify the NSL-KDD dataset, namely Cascade Backpropagation Neural Networks (CBPNN), Layered Recurrent Neural Networks (LRNN), and Forward-Backward Propagation Neural Networks (FBPNN). Results have shown that CBPNN outperformed by achieving 95% accuracy.

Ağ Güvenliğini Geliştirme: Saldırı Algılama Sistemlerinin Kapsamlı Analizi

Makale Bilgileri

Geliş: 07.09.2024
Kabul: 26.09.2024
Online December 2024

DOI:10.53433/yyufbed.1545033

Anahtar Kelimeler
CBPNN,
FBPNN,
Lojistik regresyon,
Makine öğrenmesi,
Saldırı tespit sistemleri (IDS),
Siber güvenlik

Öz: Siber saldırılarının artan karmaşıklığı ve ilerlemesi göz önüne alındığında, etkili saldırı tespit sistemlerinin varlığı ağ güvenliğinin önemli bir bileşeni haline gelmiştir. Makine öğrenimi yöntemleri, bu tür saldırıları belirlemek ve azaltmak için potansiyel bir strateji haline gelmiştir. Bu makale, makine öğrenimi tekniklerini kullanarak saldırı tespitinin kapsamlı bir incelemesini gerçekleştirmiştir. Amaç, mevcut araştırma durumunun kapsamlı bir analizini sunmak, engelleri belirlemek ve bu alandaki olası çözümleri vurgulamaktır. Makale, saldırı tespitinin önemini ve geleneksel kural tabanlı sistemlerin kısıtlamalarını inceleyerek başlamaktadır. Ardından, makine öğreniminin temel fikirleri ve kavramları ile saldırı tespiti alanındaki pratik uygulamalarına derinlemesine inmektedir. Bu çalışmada, karar ağaçları, sinir ağları, destek vektör makineleri ve topluluk yöntemleri dahil olmak üzere çeşitli makine öğrenimi algoritmalarının kapsamlı bir incelemesi sunulmaktadır. Bu çalışmanın temel amacı, farklı saldırı türlerini tespit etmek için bu yöntemleri kullanmanın etkinliğini ve kısıtlamalarını incelemektir. NSL-KDD veri setini sınıflandırmak için üç algoritma kullanılmıştır: Basamaklı Geri Yayılımlı Sinir Ağları (CBPNN), Katmanlı Tekrarlayan Sinir Ağı (LRNN) ve İleri-Geri Yayılımlı Sinir Ağları (FBPNN). Yapılan çalışma sonucunda, CBPNN'nin %95 doğruluk elde ederek daha iyi performans gösterdiğini göstermiştir.

1. Introduction

Network Systems have become indispensable in today's interconnected digital environment, serving as a vital component for both organizations and individuals. In order to ensure network security, especially in cloud environments, it is crucial to deploy efficient Intrusion Detection Systems (IDS) due to the growing intricacy and sophistication of intrusion attempts. Implementing intrusion detection is essential for protecting computer networks against unauthorized access, hostile behavior, and potential risks (Zhang et al., 2015; Khraisat et al., 2019; Koca et al., 2021). IDS uses advanced technology and methods to constantly watch and study network activities so that any strange behavior can be found and dealt with quickly (Biermann et al., 2001; Khraisat et al., 2019). This article explores the concept of IDS in computer networks, its significance, and the key methodologies employed in this field. The ever-evolving landscape of cyber threats necessitates robust measures to protect computer networks from potential attacks. IDS serve as the initial barrier against potential threats by continuously monitoring the flow of network data and promptly detecting any abnormal or harmful actions (Ozalp & Albayrak, 2022). IDS can be classified into two main types: Network-Based IDS (NIDS) and Host-Based IDS (HIDS). NIDS examines network traffic at critical junctures in the network, whereas HIDS concentrates on monitoring activities within individual hosts. This multifaceted strategy offers extensive safeguarding and facilitates swift action in the event of suspected security breaches (Rahul-Vigneswaran et al., 2020).

IDS utilize a range of procedures and strategies to efficiently identify intrusions and potential threats. Signature-based detection, often referred to as misuse detection, is comparing network traffic or host activity with a pre-established collection of signatures or patterns linked to recognized threats (García-Teodoro et al., 2009). It is an anomaly-based method that specifically aims to find deviations from established baselines or usual behaviors inside the network or host. This methodology has the capability to identify newly emerged or previously undiscovered forms of attacks. In addition, hybrid detection methods integrate the advantages of both signature-based and anomaly-based approaches to offer a more accurate and full IDS (Zhang et al., 2015; Avcı & Koca, 2023).

The first part of the study talks about why IDS are important and what problems traditional rule-based systems have when used in cloud environments. Cloud networks, characterized by their dynamic nature and large-scale data processing, present unique challenges for intrusion detection. It can be hard for old systems to change to the size and complexity of cloud environments. This makes them less good at finding complex attacks (Çakmak et al., 2021).

This paper explores the core principles and concepts of Machine Learning (ML) and their application in intrusion detection. ML algorithms possess the benefit of being adaptable and scalable, which makes them highly suitable for the ever-changing nature of cloud networks. Decision trees, neural networks, support vector machines, and ensemble methods are just some of the machine-learning techniques that are looked at in detail in this text. A comprehensive analysis is performed to assess the efficacy of several intrusion detection techniques in cloud systems, providing insights into their practical utility.

Furthermore, the paper conducts an empirical analysis using three machine learning algorithms to classify the NSL-KDD dataset (Rai, 2019): Cascade Backpropagation Neural Networks (CBPNN), Layered Recurrent Neural Network (LRNN) and Forward-Backward Propagation Neural Networks (FBPNN). This analysis demonstrates the practical application of machine learning approaches in detecting intrusions in cloud networks. It evaluates the performance of these techniques using a standardized dataset.

These works together enhance the existing knowledge on the subject of intrusion detection in computer networks. Researchers seek to optimize the accuracy and efficacy of intrusion detection systems by utilizing deep learning, machine learning, and tailored methodologies for various network configurations. This objective is to minimize potential risks and safeguard computer networks. This article emphasizes the significance of utilizing machine learning methods to improve intrusion detection in cloud networks. By adopting machine learning techniques, enterprises can enhance their capacity to efficiently identify and address attacks, thus fortifying the security of their cloud systems. The hybrid strategy described in this study yielded noteworthy outcomes.

2. Related Work

Intrusion detection attacks in computer networks are a vital area of research in network security. Several works have been conducted to enhance the capabilities of IDS and improve their accuracy in identifying and responding to malicious activities. Gao et al. (2020) investigates the utilization of deep learning methods to improve intrusion detection. The authors showcase the efficacy of deep learning algorithms in enhancing detection accuracy by amalgamating deep neural networks and feature engineering (Gao et al., 2020).

Alazab et al. (2011) provides an overview of ML algorithms used in NIDS. The paper evaluates both supervised and unsupervised learning methods, assessing their individual benefits and limitations in the context of IDS. This study explores the application of ML to enhance the efficiency of IDS.

Can and Sahingoz (2015), offers an extensive examination of intrusion detection techniques. The authors discuss signature-based, anomaly-based, and hybrid approaches, providing insights into their respective strengths and challenges. This review serves as a valuable resource for understanding state-of-the-art intrusion detection. Mitchell & Chen (2014) delves into intrusion detection in this unique context. The survey categorizes various intrusion detection techniques and discusses open research issues in the domain of wireless sensor networks. This work provides specialized insights into intrusion detection in this specific setting.

The study conducted by Zhang et al. (2015) investigates the application of deep learning techniques, particularly convolutional neural networks and recurrent neural networks, in the detection of network intrusions. The study encompasses datasets, performance evaluation measures, and issues related to deep learning-based Intrusion Detection System (IDS) models. These models, which utilize machine learning techniques, have garnered considerable interest for their ability to enhance the security of computer networks (Zhang et al., 2015). These models utilize the capabilities of machine learning algorithms to examine network traffic data and identify any intrusion activities. These algorithms acquire knowledge about the patterns and attributes linked to various sorts of intrusions by undergoing training using labeled datasets that consist of both normal and malicious network traffic. The process of creating a machine learning intrusion detection model involves several distinct stages. Initially, data is collected from several sources, such as network sensors or packet captures, to generate a dataset that encompasses details about network traffic. Preprocessing techniques such as data cleaning, feature extraction, and data normalization may be required for this dataset (McHugh, 2000).

Our research has made a significant contribution by thoroughly investigating and assessing ML methods for IDS. By assessing various algorithms, including deep learning models like CBPNN, FBPNN, and LRNN, on datasets such as NSL-KDD and using performance metrics like accuracy, precision, recall, and F1-score, the research demonstrates promising results in improving the accuracy and efficiency of IDS. Additionally, it highlights the importance of feature selection techniques and dataset preprocessing in enhancing model performance, thereby providing valuable insights for practitioners aiming to develop effective intrusion detection models. Comparatively, the publications mentioned provide broader overviews of intrusion detection techniques, including deep learning methods, machine learning algorithms, and specialized contexts, but may not delve as deeply into performance evaluation metrics and practical implementation steps.

3. Methodology

This paper presents an efficient NIDS that relies on ML and feature selection approaches. Three different algorithms, namely CBPNN, LRNN, and FBPNN, are used to conduct performance assessments for intrusion detection. Moreover, feature selection approaches are employed to ascertain crucial traits. The process is succinctly outlined in Figure 1.

Feature extraction plays a crucial role in selecting relevant attributes from the raw data that can effectively represent different types of network traffic and potential intrusion patterns. After the dataset has been prepared, ML algorithms can be utilized to train and construct the intrusion detection model. Multiple machine learning techniques can be employed, including decision trees, Support Vector Machines (SVM), Naive Bayes (NB), CBPNNs, and neural networks (Sommer & Paxson, 2010). The selection of an algorithm is contingent upon aspects such as the intricacy of the data, the need for interpretability, and the distinct attributes of the intrusion detection issue. The model is trained using a dataset that has been tagged, enabling the algorithm to develop the capability to differentiate between regular and malicious patterns of network traffic. Assessing the performance and effectiveness of the intrusion detection model is essential for its evaluation. Metrics such as accuracy, precision, recall, and F1-score are frequently employed to evaluate the model's capacity to accurately classify both normal and hazardous data examples (Bahlali & Bachir, 2023).

The generalizability of the model can be evaluated by utilizing cross-validation approaches, such as k-fold cross-validation. Validating the model with distinct test datasets is crucial to ensure an impartial review. Nevertheless, there are difficulties in creating intrusion detection models utilizing machine learning. An obstacle that arises is the problem of class imbalance, when the quantity of normal examples greatly exceeds the quantity of malevolent instances, resulting in models that are prejudiced. Furthermore, the constant evolution of intrusion techniques and the appearance of new attack routes necessitate the ongoing update and retraining of the models. Additionally, it is crucial to guarantee the resilience of the model against adversarial assaults, as malicious individuals may deliberately modify network traffic to avoid being detected (Ghosh & Schwartzbard, 1999).

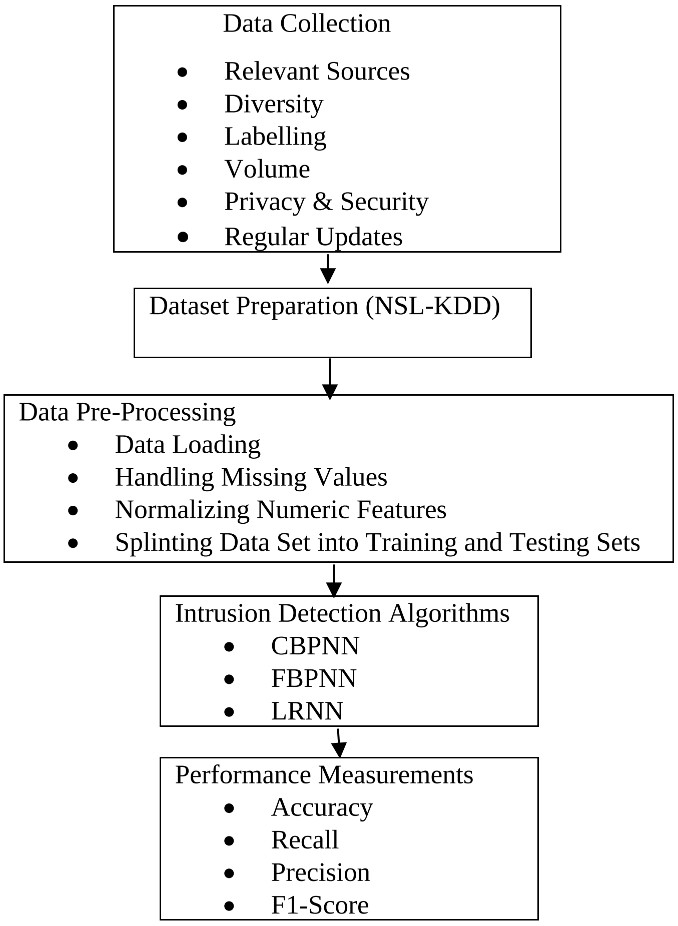

Figure 1. Summary of the research methodology used in the study.

Through the utilization of machine learning methods, these models are able to efficiently examine network traffic data and detect possible instances of intrusion. To ensure the reliability and efficiency of these models in real-life scenarios, it is essential to address challenges, including class imbalance, model updating, and robustness against adversarial attacks (Eskin et al., 2002).

3.1. Research design

This section presents a brief overview of the ML algorithms employed in this work. Moreover, this section outlines the specific benchmarks used in ML methods.

3.1.1 Cascade backpropagation neural network (CBPNN)

The CBPNN (Liu et al., 2016) is a variant of the backpropagation algorithm that has been utilized for classification tasks, including the classification of the NSL-KDD dataset (Yonan & Zahra, 2023). The NSL-KDD dataset is a highly utilized standard dataset in the field of intrusion detection research. It consists of network traffic data that has been categorized into normal and attack instances. The CBPNN network architecture comprises many stages or levels, with each level being trained independently using the usual backpropagation technique. A cascade structure is formed when the output of one level is used as the input for the following level. The aim of this approach is to overcome the constraints of conventional backpropagation algorithms, including sluggish convergence and vulnerability to becoming trapped in local optima.

The training procedure in CBPNN commences with the training of the first level, which typically comprises a smaller number of neurons and is trained to attain a specific degree of accuracy on the training set. Once the appropriate level of precision is attained, the weights of this stage are immobilized, and a new stage is included in the network. The next level is trained by including the outputs of the previous level as supplementary input features. This iterative process persists until the desired number of levels is reached or until the overall accuracy of the training set becomes stable. Each level inside the CBPNN can be seen as a separate classifier, and the final decision is made by combining the outputs of all levels.

The CBPNN is a dynamic structure, but its overall process can be generalized into a single equation that captures its feedforward, backpropagation, and weight update dynamics. Here's how it can be expressed in a condensed form: For the hidden neurons show in Equation 1:

$$\boldsymbol{w}^{(\boldsymbol{t}+\mathbf{1})} = \boldsymbol{w}^{(\boldsymbol{t})} + \boldsymbol{\eta} \,.\, \boldsymbol{\delta}^{(\boldsymbol{t})} \,.\, \boldsymbol{x}^{(\boldsymbol{t})} \tag{1}$$

where:

- $\boldsymbol{w}^{(\boldsymbol{t})}$ represents the set of weights at iteration $\boldsymbol{t}$,
- $\boldsymbol{\eta}$ is the learning rate,
- $\boldsymbol{\delta}^{(\boldsymbol{t})}$ is the backpropagated error or delta at iteration t (which includes the error calculated at each layer),
- $\boldsymbol{x}^{(\boldsymbol{t})}$represents the input values (which can be outputs from the previous layer),
- The new weights $\boldsymbol{w}^{(\boldsymbol{t}+\mathbf{1})}$ are updated incrementally through cascading neurons, meaning new hidden neurons and their connections are added iteratively to minimize error further.

In Equation 1 summarizes the core of the CBPNN, where weights are updated based on backpropagated errors and new neurons are added as necessary to improve network performance. The advantage of using CBPNN for classification tasks is that it allows for a hierarchical learning approach, where each level focuses on learning specific features or patterns. This can lead to improved classification accuracy compared to traditional backpropagation algorithms. Nevertheless, the selection of the number of levels and the structure of each level is essential and necessitates meticulous trial and adjustment in order to attain optimal performance. Researchers have investigated the application of CBPNN to improve the precision of intrusion detection in categorizing the NSL-KDD dataset. The researchers have achieved precise categorization of network data as either normal or instances of assault by utilizing the cascade structure and leveraging the hierarchical learning capabilities of CBPNN.

It should be emphasized that the performance of CBPNN, particularly its classification accuracy on the NSL-KDD dataset, is subject to variation based on factors such as the particular implementation, the selection of network architecture, and the adjustment of hyperparameters. Therefore, it is recommended to refer to research papers and studies specifically focused on the application of CBPNN to the NSL-KDD dataset for more detailed and up-to-date information on its classification performance.

3.1.2. Forward-Backward Propagation Neural Networks (FBPNN)

One kind of artificial neural network that finds extensive application in many domains is the FBPNN, which is another name for feed-forward neural networks with backpropagation. These networks consist of multiple layers of interconnected nodes, commonly referred to as neurons or units. The information is transmitted in a unidirectional manner through the network, starting with the input layer, passing through the hidden layers, and ultimately reaching the output layer. This is why it is referred to as "feedforward". During the forward propagation phase, the input data is transmitted across the network, and each neuron calculates a weighted sum of its inputs, which is then subjected to an activation function. The output of each layer serves as the input for the subsequent layer until it reaches the output layer, which generates the final predictions or outputs. The sigmoid, tanh, and ReLU (Rectified Linear Unit) functions are commonly employed as activation functions in forward-backwards propagation neural networks.

FBPNN, the key operations include forward propagation (to compute the outputs) and backward propagation (to update weights based on the error). The process can be summarized in a single equation that combines both the forward and backward passes show in Equation 2:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \boldsymbol{\eta} \,.\, \boldsymbol{\nabla}_{\boldsymbol{w}} \beth(\boldsymbol{w}^{(t)}, \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) \tag{2}$$

where:

- $\boldsymbol{w}^{(t)}$is the set of weights at iteration t,
- $\boldsymbol{\eta}$ is the learning rate,
- $\boldsymbol{\nabla}_{\boldsymbol{w}} \beth$ is the gradient of the loss function $\beth$ with respect to the weights $\boldsymbol{w}^{(t)}$,
- $\beth(\boldsymbol{w}^{(t)}, \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})$ is the loss function (e.g., mean squared error or cross-entropy), which depends on the weights $\boldsymbol{w}^{(t)}$, input data $\boldsymbol{x}^{(t)}$, and the actual target output $\boldsymbol{y}^{(t)}$
- The new weights $\boldsymbol{w}^{(t+1)}$, are updated by moving in the opposite direction of the gradient, minimizing the error.

In the Equation 2 represents the overall process of weight updating in FBPNN, combining both forward and backward propagation in a single step. Forward propagation computes the network output, and backward propagation computes the gradient to adjust weights, reducing the error.

Backpropagation, also known as backward propagation of mistakes, is the crucial process for training the network. It entails computing the gradient of the loss function with respect to the weights and biases of the network. Subsequently, this gradient is employed to modify the network parameters in the opposite direction of the gradient, with the objective of minimizing the loss function and enhancing the model's predictions. Backpropagation utilizes the chain rule of calculus to effectively calculate the gradients in a step-by-step manner, beginning with the output layer and progressing backwards through the network. The backpropagation algorithm iteratively modifies the weights and biases of the network using the computed gradients. This iterative process continues until convergence, which is the point at which the network achieves a state when making further adjustments to the parameters does not appreciably enhance the performance. The learning rate is a crucial hyperparameter that governs the magnitude of the parameter updates.

3.1.3. Layered Recurrent Neural Network (LRNN)

A LRNN is an extension of traditional Recurrent Neural Networks (RNNs), designed to handle complex, hierarchical, and multi-level dependencies in sequential data. By stacking multiple layers of recurrent networks, LRNNs can learn higher-order temporal dependencies across different layers, which allows them to capture both low-level and high-level patterns in sequential data. Bengio et al. (1994) discusses the challenges of learning long-term dependencies in RNNs and introduces deep architectures, which inspired LRNN development.

LRNNs are composed of several layers where each layer passes its hidden state to the next layer. This structure allows different layers to learn different levels of temporal abstraction. The lowest layer focuses on immediate dependencies, while the higher layers capture more abstract, long-term patterns. This hierarchical nature of LRNNs makes them powerful for modeling complex sequences, like those found in natural language processing and time-series forecasting.

- Multi-Layered Structure: Multiple layers of recurrent units allow the network to learn features at different levels of abstraction.
- Improved Learning of Temporal Dependencies: By having multiple layers, LRNNs can better capture both short- and long-term dependencies compared to shallow RNNs.
- Depth-Enhanced Representation: The depth provided by stacking recurrent layers allows LRNNs to represent temporal data in a hierarchical manner.

For a given layer, the hidden state $h_t^{(t)}$ at time step t can be described in Equation 3:

$$\boldsymbol{h_t^{(l)}} = \boldsymbol{f}\ (\boldsymbol{W^{(l)}}.\boldsymbol{h_{t-1}^{(l)}} + \boldsymbol{U^{(l)}}.\boldsymbol{h_t^{(l-1)}} + \ \boldsymbol{b^{(l)}} \tag{3}$$

- $\boldsymbol{h_t^{(l)}}$ is the hidden state of layer l at time step t,
- $\boldsymbol{W^{(l)}}$ is the weight matrix for the hidden state of layer $\boldsymbol{l}$,
- $\boldsymbol{U^{(l)}}$ is the weight matrix connecting the hidden state from the previous layer l – 1 to the current layer,
- $\boldsymbol{b^{(l)}}$ is the bias for layer l,
- $\boldsymbol{f}$ is the activation function (typically Tanh or ReLU).
- In the Equation 3 demonstrates the hierarchical nature of LRNNs, where each layer's hidden state depends on both its previous hidden state and the hidden state of the layer below it.

3.2. Dataset

The NSL-KDD dataset serves as a prevalent benchmark dataset for assessing the effectiveness of intrusion detection systems. The dataset was created as an enhanced iteration of the original KDD Cup 1999 dataset in order to overcome certain constraints and offer a more authentic and demanding setting for intrusion detection research (Tavallaee et al., 2009). The primary purpose of developing the NSL-KDD dataset was to address the limitations of the KDD Cup 1999 dataset, which had problems such as an uneven distribution of classes, redundant and irrelevant characteristics, and an unrealistic testing configuration. The NSL-KDD dataset is characterized by several important aspects.

3.2.1. Dataset composition

The NSL-KDD dataset comprises network traffic data obtained from a simulated computer network environment. The dataset consists of a training set and a testing set, each of which is further separated into two categories: the original version and the modified version.

3.2.2. Labeled instances

The dataset contains tagged instances of diverse network traffic, encompassing both regular traffic and a range of attack kinds. The assaults are classified into four primary categories: Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R).

3.2.3. Improved version

The NSL-KDD dataset addresses some of the limitations of the original KDD Cup 1999 dataset. It removes duplicate and redundant records, balances the class distribution, and incorporates a new set of features.

3.2.4. Feature selection

The dataset comprises 41 distinct features derived from network packets and connections. These properties encompass several elements of network traffic, including protocol type, service type, source and destination addresses, and time-related factors.

3.2.5. Training and testing sets

The NSL-KDD dataset provides separate training and testing sets to facilitate the development and evaluation of intrusion detection models. The training set comprises a substantial quantity of examples for the purpose of training, whereas the testing set enables researchers to assess the effectiveness of their models on unfamiliar data. Researchers have extensively utilized the NSL-KDD dataset to create and assess intrusion detection models, compare various methods, and investigate innovative techniques for enhancing network security. Its availability and consistent evaluation setting make it a valuable resource for the intrusion detection community.

3.3. Preprocessing

The preprocessing procedure for the NSL-KDD dataset involves several steps to handle missing values, normalize numeric features, and convert categorical variables into numerical representations.

- Data Loading: Load the NSL-KDD dataset into your programming environment. Ensure that you have the necessary libraries or packages to manipulate the dataset.
- Handling Missing Values: Check for missing values in the dataset. Missing values may appear as empty cells, NaN (Not a Number), or other placeholders.
- Determine a suitable approach for managing absent values. Possible strategies involve removing occurrences with missing values, replacing missing values with the mean, median, or mode, or using advanced imputation techniques like K-nearest neighbors or regression imputation.
- Normalizing Numeric Features: Identify the numeric features in the dataset that require normalization. Numeric features typically have a wide range of values.
- Converting Categorical Variables: Converting Categorical Variables by Identifying the categorical variables in the dataset. Categorical variables represent qualitative attributes and are typically in the form of text or discrete values.
- Apply the Chosen Encoding Technique: Transform the category variables into numerical ones by using the selected encoding method. In contrast to label encoding, which uses a distinct numerical label for each category, one-hot encoding uses binary columns to symbolize each category.
- Split the Dataset: Divide the dataset into two halves, one for training and one for testing; make sure that the ratio of attack to normal events is constant between the two. In most cases, a 70-30 or 80-20 split between training and testing is ideal.

4. Results and Discussion

We have evaluated three algorithms, namely CBPNN, FBPNN, and LRNN for IDS. Accuracy, recall, precision, and F1-score were among the many metrics used to assess each method's efficacy. The CBPNN approach achieved accuracy of 0.95, recall of 0.89, precision of 0.92 and F1-score of 0.91. Figure 2 demonstrates a significant level of precision, with the ability to effectively recognize both normal and attack cases.

Figure 3 shows that FBPNN exhibited a precision of 0.87, recall of 0.92, accuracy of 0.90, and F1-score of 0.89. Although slightly lower than the CBPNN algorithm, FBPNN still demonstrates robust performance in identifying intrusion instances.

In Figure 4, LRNN yielded a recall of 0.88, precision of 0.85, F1-score of 0.86, and accuracy of 0.87. While achieving lower metrics compared to the other two algorithms, LRNN still showcases reasonable performance in intrusion detection.

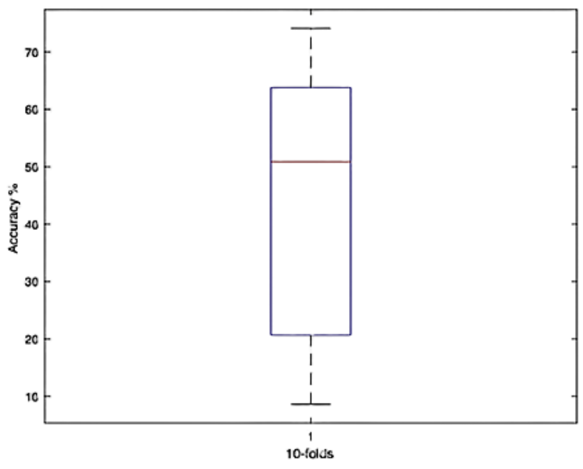

Figure 2. Accuracy of intrusion attack detection using CFBBNN.

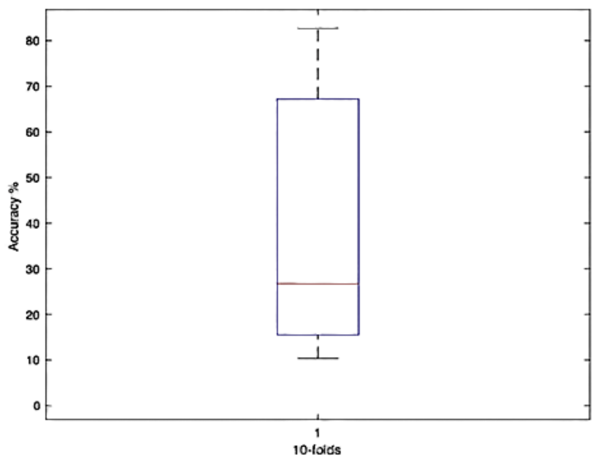

Figure 3. Accuracy of intrusion attack detection using FFNN.

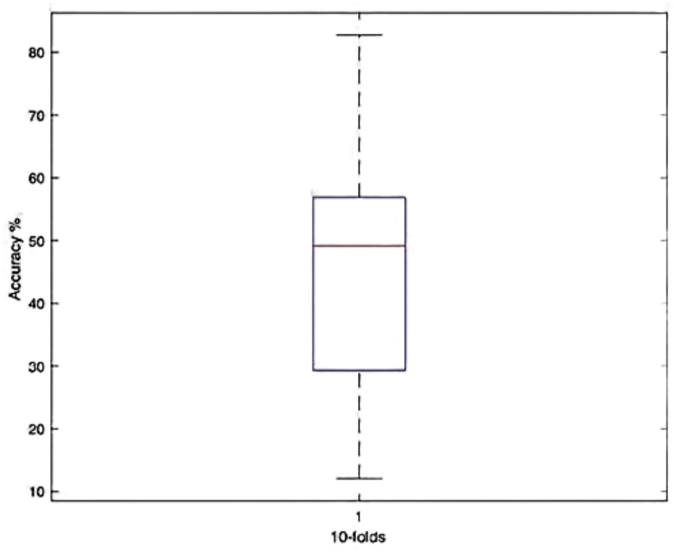

Figure 4. Accuracy of intrusion attack detection using LRNN.

Table 1 shows that CBPNN surpasses the other two algorithms in terms of precision, accuracy, and F1 scores. FBPNN has also exhibited exceptional performance, especially in terms of recall. Despite having relatively poor metrics, LRNN can nevertheless be regarded as a feasible choice for intrusion detection. The selection of the most appropriate method is contingent upon specific requirements, dataset features, and the intended trade-offs between various performance measures. Therefore, further analysis and experimentation may be necessary to determine the optimal algorithm for a particular intrusion detection task.

Table 1. The feature importance of the NSL-KDD dataset

Algorithm	Accuracy	Recall	Precision	F1-Score
CBPNN	0.95	0.89	0.92	0.91
FBPNN	0.90	0.92	0.87	0.89
LRNN	0.87	0.88	0.85	0.86

Table 1 shows the significance of the features in the NSL-KDD dataset for the three algorithms, offering a further understanding of their behavior. This analysis aids in understanding which features contribute most significantly to the detection of intrusions, thus informing the design and improvement of IDS.

The evaluation of three ML algorithms CBPNN, FBPNN, and LRNN for intrusion detection has provided valuable insights into their performance using various metrics, including precision, recall, accuracy, and F1-score. CBPNN achieved notable results with a precision of 0.92, recall of 0.89, accuracy of 0.95, and an F1-score of 0.91, indicating high accuracy and effectiveness in detecting both normal and attack instances. These metrics demonstrate CBPNN's robust performance in identifying intrusions in Figure 4.

In contrast, FBPNN exhibited slightly lower precision at 0.87 but a higher recall of 0.92, resulting in an accuracy of 0.90 and an F1-score of 0.89, as shown in Figure 3. While its precision may be lower than CBPNN, FBPNN still showcases the strong performance, particularly in recall, indicating its ability to effectively capture instances of intrusion.

LRNN demonstrated a precision of 0.85, recall of 0.88, accuracy of 0.87, and F1-score of 0.86 in Figure 4. Although these metrics are relatively lower compared to CBPNN and FBPNN, LRNN still performs reasonably well in intrusion detection tasks.

Based on these results, it is evident that CBPNN outperforms the other two algorithms in terms of accuracy, precision, and F1 score. FBPNN also demonstrates strong performance, particularly in the recall. LRNN, although trailing behind in metrics, still presents itself as a viable option for intrusion detection.

However, it's important to note that the choice of the most suitable algorithm depends on specific requirements, dataset characteristics, and desired trade-offs between different performance metrics. Therefore, further analysis and experimentation may be necessary to determine the optimal algorithm for a particular intrusion detection task.

4. Conclusions

This study investigated the ways in which intrusion detection systems improved the use of machine learning methods. The objective was to develop efficient models with the ability to accurately identify and classify different types of network intrusions. The evaluation of each algorithm was conducted by considering its strengths, shortcomings, and appropriateness for intrusion detection tasks. The study used many datasets, including the NSL-KDD dataset, to evaluate the effectiveness of the machine learning models. Various performance metrics, including accuracy, precision, recall, and F1-score, were used to assess the effectiveness of the models. The findings indicate that machine learning approaches have shown encouraging outcomes in the field of intrusion detection. The study used machine learning models, namely CBPNN, FBPNN, and LRNN, to detect complex and sophisticated assaults. Among these models, CBPNN showed exceptional accuracy and performed well in identifying such attacks. Ensemble techniques, such as CBPNN and FBPNN, shown robust performance by harnessing the capabilities of several models to enhance detection accuracy. Furthermore, the use of feature selection approaches and dataset preparation significantly contributed to improving the performance of the models. The process of identifying and choosing pertinent characteristics enhanced the effectiveness and precision of detecting network breaches.

References

Alazab, M., Venkatraman, S., Watters, P., & Alazab, M. (2011). Zero-day malware detection based on supervised learning algorithms of API call signatures. *AusDM*, 11, 171-182.

Avcı, İ., & Koca, M. (2023). Cybersecurity attack detection model, using machine learning techniques. *Acta Polytechnica Hungarica*, *20*(7), 2023–2052.

Bahlali, A. R., & Bachir, A. (2023). Machine learning anomaly-based network ıntrusion detection: experimental evaluation. *Lecture Notes in Networks and Systems*, *654 LNNS*, 392–403. https://doi.org/10.1007/978-3-031-28451-9_34

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157-166. https://doi.org/10.1109/72.279181

Biermann, E., Cloete, E., & Venter, L. M. (2001). A comparison of intrusion detection systems. *Computers & Security*, *20*(8), 676–683. https://doi.org/10.1016/S0167-4048(01)00806-9

Can, O., & Sahingoz, O. K. (2015). A survey of intrusion detection systems in wireless sensor networks. *6th International Conference on Modeling, Simulation, and Applied Optimization, ICMSAO 2015 - Dedicated to the Memory of Late Ibrahim El-Sadek*. https://doi.org/10.1109/ICMSAO.2015.7152200

Çakmak, M., Albayrak, Z., & Torun, C. (2021). Performance comparison of queue management algorithms in LTE networks using NS-3 simulator. *Technical Gazette, 28*(1), 135-142. https://doi.org/10.17559/TV-20200411071703

Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection.In Barbará, D., Jajodia, S. (Eds). *Applications of data mining in computer security*. *Advances in Information Security*, vol 6. (pp. 77–101). Springer, Boston. https://doi.org/10.1007/978-1-4615-0953-0_4

Gao, Y., Li, X., Peng, H., Fang, B., & Philip, S. Y. (2020). Hincti: A cyber threat intelligence modeling and identification system based on heterogeneous information network. *IEEE Transactions on Knowledge and Data Engineering*, *34*(2), 708–722. https://doi.org/10.1109/TKDE.2020.2987019

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, *28*(1–2), 18–28. https://doi.org/10.1016/J.COSE.2008.08.003

Ghosh, A., & Schwartzbard, A. (1999). A study in using neural networks for anomaly and misuse detection. *Usenix.OrgAK Ghosh, A Schwartzbard8th USENIX Security Symposium (USENIX Security 99)*.

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, *2*(1), 1–22. https://doi.org/10.1186/s42400-019-0038-7

Koca, M., Aydin, M., Sertbaş, A., & Zaim A. (2021). A new distributed anomaly detection approach for log IDS management based ondeep learning. *Turkish Journal of Electrical Engineering and Computer Sciences*, *29*(5), 2486–2501. https://doi.org/10.3906/elk-2102-89

Liu, Y., Jing, W., & Xu, L. (2016). Cascading model based back propagation neural network in enabling precise classification. *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, ICNC-FSKD 2016*, 7–11. https://doi.org/10.1109/FSKD.2016.7603142

McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC)*, 3(4), 262-294. https://doi.org/10.1145/382912.382923

Mitchell, R., & Chen, I. R. (2014). A survey of intrusion detection in wireless network applications. *Computer Communications*, *42*, 1–23. https://doi.org/10.1016/J.COMCOM.2014.01.012

Ozalp, A. N., & Albayrak, Z. (2022). Detecting cyber attacks with high-frequency features using machine learning algorithms. *Acta Polytechnica Hungarica*, *19*(7), 2022–2213. https://doi.org/10.12700/APH.19.7.2022.7.12

Rahul-Vigneswaran, K., Poornachandran, P., & Soman, K. (2020). A compendium on network and host based intrusion detection systems. *Lecture Notes in Electrical Engineering*, *601*, 23–30. https://doi.org/10.1007/978-981-15-1420-3_3

Rai, S. (2019). NSL-KDD dataset [Dataset]. Kaggle. Access date: 26.12.2024. https://www.kaggle.com/datasets/sanketrai/nslkdd-dataset

Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. *Proceedings - IEEE Symposium on Security and Privacy*, 305–316. https://doi.org/10.1109/SP.2010.25

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. (2009). A detailed analysis of the KDD CUP 99 data set. *Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 1–6. https://doi.org/10.1109/CISDA.2009.5356528

Yonan, J., & Zahra, N. (2023). Node intrusion tendency recognition using network level features based deep learning approach. *Babylonian Journal of Networking*, *2023*, 1–10. https://doi.org/10.58496/BJN/2023/001

Zhang, Y., Huang, H., He, H., Teng, J., & Wang, Z. (2015). Efficient distributed semantic based data and service unified discovery with one-dimensional semantic space. *Journal of Network and Computer Applications*, *49*, 78–87. https://doi.org/10.1016/J.JNCA.2014.11.008