Information Sciences with Applications, Vol. 4, 2024
https://doi.org/10.61356/j.iswa.2024.4389

Advanced Intrusion Detection in Software-Defined Networks through Ensemble Modeling

Ibrahim M. Elezmazy [1] and Walid Abdullah [1,*]

[1] Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Sharqiyah, Egypt.
Emails: ib.elazmazy024@fci.zu.edu.eg; waleed@zu.edu.eg.

* Correspondence: waleed@zu.edu.eg.

Abstract: In software-defined networks (SDNs), effective intrusion detection is crucial for maintaining network safety and integrity. Traditional intrusion detection systems (IDS) have often failed to identify sophisticated threats due to their limited detection capabilities. To address this challenge, this study introduces an ensemble model that integrates Convolutional Neural Networks (CNN), Support Vector Machines (SVM), and XGBoost. This ensemble approach aims to enhance intrusion detection in SDNs by leveraging the strengths of each model. Using the InSDN dataset for training, our proposed model demonstrates superior performance and significantly outperforms a set of state-of-the-art models achieving a performance of 95% for accuracy, precision, recall, and F1 score exceeding the performance of other methods. Additionally, it significantly reduces false positive rates, highlighting its effectiveness in detecting complex intrusions in SDNs.

Keywords: IDS; Ensemble Learning; Network Security; Deep Learning; CNN; SVM; Anomaly Detection.

1. Introduction

The rapid expansion of computer networks has greatly transformed the Internet and telecommunications sectors, leading to remarkable advancements in connectivity and efficiency [1]. Within this landscape, Software-Defined Networks (SDNs) have emerged as a groundbreaking innovation, offering unprecedented levels of flexibility, control, and centralization that exceed traditional network architectures. However, the rise of SDNs has also introduced new security challenges, making them vulnerable to various forms of unauthorized access and cyber threats. As a result, the development of robust and effective Intrusion Detection Systems (IDS) for SDNs has become a critical area of research [2].

IDS plays a vital role in detecting illicit activities, mitigating potential risks, and safeguarding the integrity and confidentiality of network information [3]. Traditional IDS approaches, which rely on predefined signatures and anomaly detection, have struggled to keep pace with the evolving landscape of cyber threats. The emergence of sophisticated cyber-attacks necessitates more advanced and adaptive solutions. Consequently, there is a pressing need for the integration of machine learning and deep learning techniques to enhance the effectiveness and resilience of IDS in the context of SDNs. This transition to more advanced methodologies aims to address the limitations of conventional systems and improve overall network security [4].

In recent years, the use of machine learning models for intrusion detection has significantly increased due to their ability to identify patterns in data and make accurate predictions [5]. Among the most popular machine learning techniques employed for this purpose are Convolutional Neural Networks (CNNs) [6], Support Vector Machines (SVMs) [7], and XGBoost [8], a prominent example of gradient boosting algorithms. Each of these models offers unique strengths: CNNs excel in feature extraction and pattern recognition, SVMs are highly effective at classifying data in high-dimensional spaces, and XGBoost leverages gradient boosting to construct powerful ensemble models. These

diverse capabilities make them well-suited for the complex task of detecting intrusions in network systems.

Although these individual models have shown considerable success in detecting attacks, their performance can be further enhanced by combining them into ensemble models. Ensemble learning leverages the strengths of different models to improve overall accuracy, robustness, and generalizability. The objective of this study is to develop a hybrid methodology for intrusion detection in SDNs by integrating CNN, SVM, and XGBoost classifiers using a weighted soft voting algorithm. The proposed mode was compared against a set of state-of-arts models, the results of this approach demonstrate impressive performance, it achieved the best performance of 95% for accuracy, precision, recall, and F1 score exceeding the performance of other methods.

The rest of this paper is structured as follows: Section 2 provides the literal review and background needed for this study. Section 3 presents the methodology of this study. The proposed work is shown in Section 4. Section 5 presents experimental results. The conclusion and future directions are presented in Section 6.

2. Related Work

In this section, we provide a literature review on DL-based models for intrusion detection which are summarized in Table 1.

Elsayed et al. [9] proposed a hyper approach based on a Long Short Term Memory (LSTM) autoencoder and One-class Support Vector Machine (OC-SVM) to train the models exclusively with instances of typical classes to identify anomaly-based assaults in an imbalanced dataset. After being trained to recognize the typical traffic pattern and the input data's compressed representation, or latent features, the LSTM-autoencoder feeds the information to an OC-SVM algorithm. The disadvantages of the independent OC-SVM, such as its limited capacity to function in massive and high-dimensional datasets, are addressed by the hybrid model. Furthermore, they used the most recent Intrusion Detection System (IDS) dataset for SDN settings (InSDN) to conduct our tests.

Elsayed et al. [10] investigated the use of CNN for IDSs and proposed a technique to improve its performance by addressing the overfitting issue using two well-liked regularization strategies. The method enhances IDSs' capacity to identify invisible intrusion events. They trained our approach and assessed its performance using the InSDN benchmark dataset. Based on the experimental results, it can be shown that regularization techniques can enhance CNN-based anomaly detection models' performance in an SDN environment.

Thakur et al. [11] proposed a model that classifies the incursions using a deep learning technique after extracting valuable features from the provided features. It should be emphasized that the underlying data points are not representative of the same distribution; rather, they are drawn from two distinct distributions, one of which is specific to the domain and the other general to all network intrusions. Considering this, they developed a novel Generic-particular autoencoder architecture, in which the autoencoder learns features relevant to that domain, while the generic autoencoder learns features common to all types of network intrusions.

Said et al. [12] created a hybrid Convolutional Neural Network (CNN) and bidirectional long short-term memory (BiLSTM) network to improve the use of binary and multiclass classification in network intrusion detection. Using the most widely used datasets (UNSW-NB15 and NSL-KDD), the efficacy of the suggested model was evaluated. We also made use of the InSDN dataset, which is devoted only to SDN.

Liu et al. [13] proposed an edge-intelligent intrusion detection system that can be applied when a WSN encounters a DoS attack. They implemented a parallel technique to increase the population variety in iterations and Lévy flight to enhance the AOA algorithm's capacity to leap out of the local optimum. Then, using the KNN machine learning classifier in conjunction with the enhanced PL-AOA optimization process.

Balyan et al. [14] proposed HNIDS, A PSO technique enhances the vector. A multi-objective function is added to GA to increase its performance. It chooses the best features and produces better fitness outcomes to investigate the important characteristics, minimizing dimensions, increasing the true positive rate (TPR), and reducing the false positive rate (FPR). The following step involves an IRF that removes the less important features, adds a list of decision trees to each iterative process, monitors the classifier's performance, and guards against overfitting problems.

Another recent work [15], proposed an abnormal traffic detection system based on a hybrid deep learning model. The system uses a form of hierarchical detection. To achieve the fine detection of abnormal traffic from the surface, it first completes the rough detection of abnormal traffic in the network based on statistical data from switch ports. Next, it uses deep learning technology and wavelet transform to extract multi-dimensional features of all traffic data flowing through suspicious switches. The experimental findings demonstrate how fast the source of anomalous traffic may be found using the suggested port-information-based detection technique.

Table 1. Summarization of some recent intrusion detection studies based on DL.

Ref.	Year	Dataset	Model	Evaluation (AUC.)
[9]	2020	InSDN	OC-SVM	87.5
			LSTM-Autoencoder-OC-SVM	90.5
[10]	2021	InSDN	Model-Based CNN Technique	93.01
[12]	2023	InSDN	CNN-LSTM	95.03
			LeNet5	92.09
[13]	2022	WSN-DS	KNN	91.16
			kNN_{PSO}	92.89
[14]	2022	NSL-KDD	HNIDS	88.149

3. Methodology

3.1 Convolutional Neural Network

CNN, also known as regularized feed-forward neural networks, uses filter (or kernel) optimization to automatically learn features. By applying regularized weights over fewer connections, backpropagation issues with disappearing gradients and expanding gradients which were observed in previous neural networks are avoided [6].

An input layer, hidden layers, and an output layer compose a CNN. One or more convolution-performing layers are included in the hidden layers of a convolutional neural network. This usually consists of a layer that uses the layer's input matrix to perform a dot product of the convolution kernel. ReLU is typically used as the activation function for this product, which is typically the Frobenius inner product. The convolution procedure creates a feature map as the convolution kernel moves along the layer's input matrix; this feature map then feeds into the input of the layer after it. Other layers including pooling layers, completely connected layers, and normalization layers come after this. The degree to which a convolutional neural network resembles a matched filter should be highlighted here[16]. The CNN architecture is shown in Figure 1.

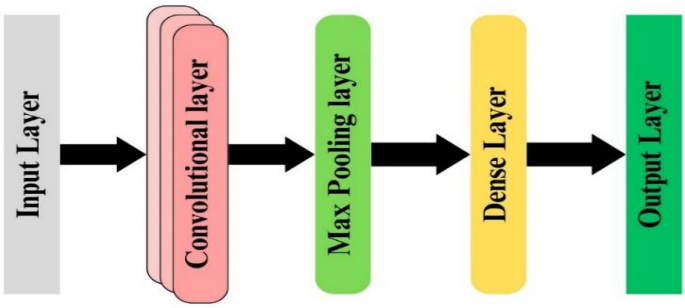

Figure 1. CNN model architecture.

3.2 Support-Vector Machine

The concept behind the support-vector network was previously implemented for the restricted case where the training data can be separated without errors. Here, we extend this result to non-separable training data. The support-vector network is a new learning machine for two-group classification problems. The machine conceptually implements the following idea: input vectors are non-linearly mapped to a very high-dimension feature space. In this feature space, a linear decision surface is constructed. Special properties of the decision surface ensure the high generalization ability of the learning machine [17].

Linear SVM for Binary Classification is used for separating data into two classes using a linear decision boundary. Here's a detailed explanation. Is used to find a hyperplane that best separates two classes in a feature space. The hyperplane is chosen such that the margin between the closest points of the two classes (known as support vectors) is maximized. Given a set of training data $\{(x_i, y_i)\}$, where $x_i \in \mathbb{R}^n$ represents the feature vectors and $y_i \in \{-1, +1\}$ represents the class labels, the hyperplane can be represented as follows:

$$f(x) = w \cdot x + b = 0 \tag{1}$$

Here w is the weight vector and b is the bias. And the distance from a point x_i to the hyperplane can be calculated as follows:

$$\frac{|w \cdot x + b|}{\|w\|} \tag{2}$$

The SVM model for a binary classification problem with a linear separator is shown in Figure 2.

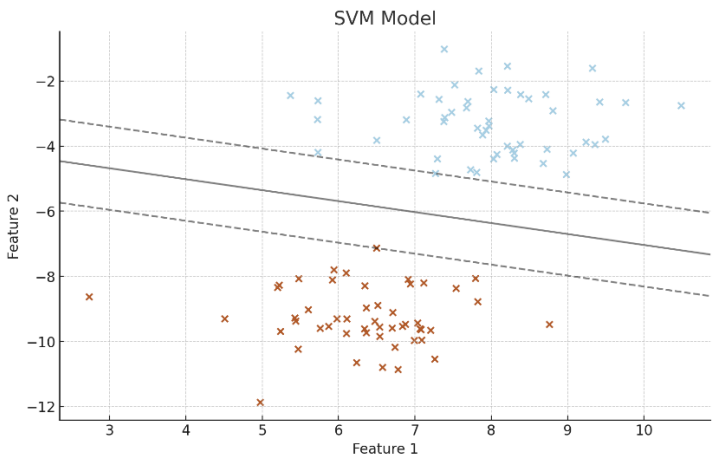

Figure 2. 2D plot illustrating the SVM model.

3.3 Extreme Gradient Boosting (XGBoost)

For data scientists and machine learning engineers, XGBoost has emerged as their preferred machine learning algorithm. Large volumes of data may be used to train and test models using this open-source framework. It has been applied in several fields, such as high-energy physics event classification and ad click-through rate prediction. Large dataset performance, speed, and ease of use are the main goals of XGBoost's design. It can be utilized right away after installation without the need for any additional configuration because it doesn't require parameter tuning or optimization[18]. The mathematical rates involve delving into the convergence rates, learning rates, and the optimization process as shown below:

$$\hat{y}^{(t)} = \hat{y}^{(t-1)} + \eta \cdot f_t(x) \tag{3}$$

$$\mathcal{L}(\emptyset) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega\,(f_k) \tag{4}$$

$$\Omega\,(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{5}$$

Where $\hat{y}^{(t)}$ is the prediction at iteration t, $f_t(x)$ is the prediction from the t tree, η is the learning rate, l is a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i, Ω is a regularization term for the complexity of the model, γ is the penalty for adding a new tree, λ is the L2 regularization term on weights, T is the number of leaves in the tree, w_j is the weight on the j-th leaf.

XGBoost uses second-order Taylor approximation for the objective function to efficiently calculate the gain and cover for the decision tree splits. For each split, it calculates the gradients (g) and Hessians (h) as follows:

$$g_i = \frac{\partial \mathrm{l}(\hat{y}_i, y_i)}{\partial(\hat{y}_i)} \tag{6}$$

$$h_i = \frac{\partial^2 \mathrm{l}(\hat{y}_i, y_i)}{\partial(\hat{y}_i)^2} \tag{7}$$

Finally, the gain from a split is:

$$Gain = \frac{1}{2}\left[\frac{(\sum_{i\epsilon I_L} g_i)^2}{\sum_{i\epsilon I_L} h_i + \lambda} + \frac{(\sum_{i\epsilon I_R} g_i)^2}{\sum_{i\epsilon I_R} h_i + \lambda} - \frac{(\sum_{i\epsilon I} g_i)^2}{\sum_{i\epsilon I} h_i + \lambda}\right] - \gamma \tag{8}$$

where I_L and I_R are the instances in the left and right nodes after the split, respectively.

4. Proposed Work

To develop this problem detection method, we incorporate some machine learning approaches: CNN, SVM, and XGBoost. By fusing these models, we hope to use their different properties as well as enhance the net detection rate. The proposed model architecture is shown in Figure 3. The following is the description of each component of the proposed model.

4.1 Connected Neural Network (CNN)

CNNs can recognize patterns and features in data, especially useful for processing data sets or time series data in intrusion detection. the CNN part consists of four main layers including the Conv2D Layer which uses convolution operations to extract local features from input data, Batch Normalization to normalize the output of previous layers to stabilize and train faster, MaxPooling layer which is commonly utilized after the CNN layers to reduce the space size of the data, focusing

on the most relevant elements, and finally dense layer: A fully connected layer that provides the final output of the CNN, it generates a probabilistic output pCNN(x) of the probability of an attack. The CNN model processes the input X through multiple layers to produce a probabilistic output, this operation is mathematically represented as follows:

$$f_{CNN}(X) = \sigma[w_d \cdot Flatten\left(Conv2D_n \left(.. \left(Conv2D_1(X) \right) \right) \right)] \tag{10}$$

Where $Conv2D_i$ represents the i-th convolutional layer followed by Batch Normalization, max pooling, and Dropout, W_d represents the weights of the dense layer, and σ represents the sigmoid activation function.

4.2 Support Vector Machine (SVM)

SVM performs well in high-level environments and is used for its robustness and accuracy in classification tasks. One known problem with this model is that the values of the variables must be specified precisely. To overcome this problem, the Grid Search method is used for Hyperparameter tuning to find the optimal values for the model's Hyperparameter, which increases the performance of SVM. The SVM model maps the input X to a binary classification output as defined in Eq. (11):

$$f_{SVM}(X) = sign(w \cdot X + b) \tag{11}$$

Where w represents the weights learned by the SVM and b represents the bias term.

4.3 XGBoost

XGBoost is a robust gradient-enhancing algorithm that excels in processing structured data and can capture complex patterns through multiple decision trees. XGBoost model uses the Trees Grow method to aggregate predictions from multiple weak learners (decision trees) and outputs the weighted sum of predictions from multiple decision trees which helps to improve overall accuracy. The mathematical equation of the Trees Grow method is defined as follows:

$$f_{XGB}(X) = \sum_{K=1}^{K} \alpha_k h_k(X) \tag{12}$$

Where α_k represents the weight of the k-th tree and h_k represents the k-th decision tree.

4.4 Ensemble Voting Classifier

The final stage of our approach involves a weighted voting process to aggregate the outputs from the three models: CNN, SVM, and XGBoost. Initially, each model generates a probability estimate indicating the likelihood of an intrusion. Following this, a weighted soft voting mechanism is employed to determine the final prediction. In this process, the individual probability estimates from each model are combined into a single probability score. This is achieved through a weighted sum, where each model's contribution is adjusted according to its assigned weight. The ensemble model thus integrates the probabilistic outputs from all constituent models using this weighted soft voting strategy. Mathematically, this can be represented as follows:

$$p_{ensemble}(X) = \alpha_{CNN} \cdot p_{CNN}(X) + \alpha_{SVM} \cdot p_{SVM}(X) + \alpha_{XGB} \cdot p_{XGB}(X) \tag{13}$$

Where: p_{CNN} , p_{SVM}, p_{XGB} are the predicted probabilities from CNN, SVM, and XGBoost respectively.

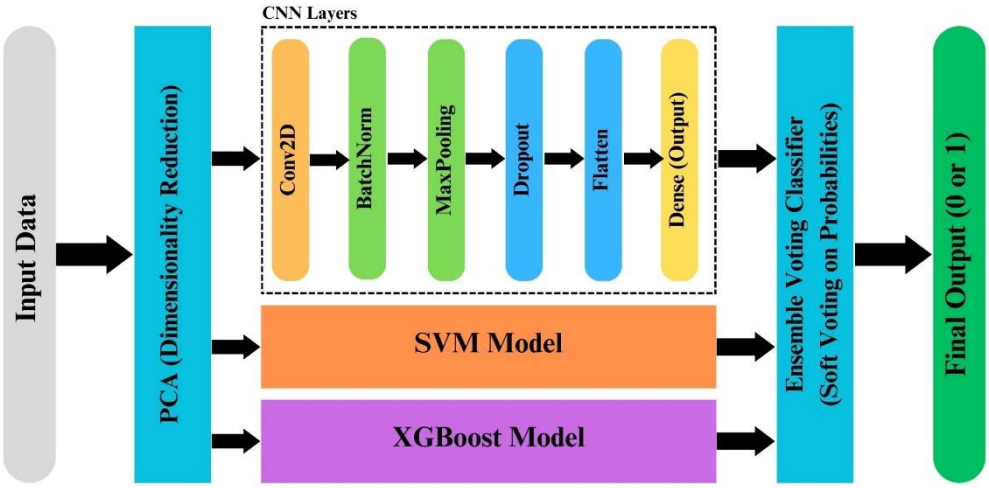

Figure 3. The proposed model architecture.

The proposed model was compiled to determine the loss function, Adam optimizer, and metrics for evaluating performance. The Categorical cross-entropy loss function is used to optimize the initial weights of the proposed model to increase classification accuracy, in addition, it was trained for 50 epochs. In addition, in our experiments, and applied a mini-batch gradient descent technique to decrease the error calculated from the loss function (Binary Cross Entropy). In each epoch, the data is divided into 64 batches so that the weights in each batch are updated. This means that in every epoch, the weights change 64 times, corresponding to the number of batches.

5. Results and Discussion

This section provides a detailed description foe the utilized dataset, evaluation metrics used to assess the proposed model, the key finding, and experimental results for this study.

5.1 Utilized Dataset

In this work, we utilized A public and widely used InSDN dataset designed for intrusion detection research [19]. This dataset contains various objects representing network traffic, which can be used to detect malicious activity. The dataset contains one label that denotes either normalcy or an attack for each of the 30 features that make up its 10000 samples. This set is meant to enhance unique characteristics, and lower repetition hence making it quality for machine learning evaluation. The dataset description is summarized in Table 2.

Table 2. InSDN dataset description.

Number of records	**10000**
Number of features	30
Number of classes	2

5.2 Evaluation Metrics

Our proposed work was evaluated using a set of comprehensive set of evaluation metrics including accuracy, precision, recall, and F1-score.

- **Accuracy (ACC)** measures the proportion of correctly classified instances among all instances:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

where:

- TP = True Positives (correctly identified attacks).
- TN = True Negatives (correctly identified normal instances).
- FP = False Positives (normal instances incorrectly classified as attacks).
- FN = False Negatives (attacks incorrectly classified as normal instances).

- **Precision** measures the proportion of true positive predictions among all positive predictions.

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

- **Recall (Sensitivity or True Positive Rate)**: measures the proportion of true positive instances that are correctly identified.

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

- **F1-score**: **this metric** provides a balanced measure between Recall and Precision

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{17}$$

Together, these metrics examine how well the model can classify network traffic into common and attack categories. Higher contrasts indicate stronger performance in terms of identification while reducing false positives and false negatives. For the Proposed model, these metrics provide quantitative insight into the efficiency and reliability of the IDS.

5.3 Implementation Settings

Python programming language was utilized to develop our deep learning models. The development and experimentation were conducted on the Google Colab platform [20], leveraging its computational resources. In addition, models were built using the Keras API [21]. Table 3 describes the models' configurations and hyperparameters used in training DL models.

Table 3. Implementation settings.

Optimizer	**Adam**
Epochs	50
Batch size	64
Dropout	0.3

5.4 Statistical Analysis

In this section, we present the comprehensive results of our investigation of six deep learning models: Convolutional Neural Networks (CNNs), Deep Neural Networks (DNN), Artificial Neural Networks (ANN), Recurrent Neural Networks (RNN), Deep Belief Networks (DBN), Gated Recurrent Units (GRU), and Long Short-Term Memory networks (LSTM). We evaluate and compare these models based on a set of key performance metrics, namely accuracy, precision, recall, and F1-score. Table 4. Offers significant insights into the classification model's performance, demonstrating its accuracy correctly.

Table 4. Performance of different DL models for intrusion detection.

Model	Accuracy	Precision	Recall	F1-score
ANN	0.92	0.92	0.92	0.92
DNN	0.92	0.90	0.93	0.91
LSTM	0.91	0.91	0.90	0.91
RNN	0.90	0.90	0.90	0.90
DBN	0.90	0.91	0.92	0.91
CNN	0.91	0.91	0.90	0.91
GRU	0.91	0.91	0.90	0.90
Proposed model	0.95	0.95	0.95	0.95

The comparison between the various deep learning models against our proposed model showed that the proposed model consistently outperforms the others across all evaluation metrics. With accuracy, precision, recall, and F1-score for the proposed model are all 0.95, highlighting its balanced and robust performance. In contrast, the other models show slight variations and low performance compared to the proposed model. These results indicate that while the traditional models exhibit competitive performance, particularly ANN and DNN, our proposed model's superior efficiency proves its potential for more accurate and reliable predictions in practical applications.

6. Conclusion and Future Work

This work proposed an enhanced deep learning model for complicated threat detection in software-defined networks (SDNs). This model combines three powerful models named CNN, SVM, and XGBoost classifiers. These models were combined with a weighted soft voting system for the best results. However, before any training could take place various activities and steps were followed during data processing. This included standardization and transformation into a 2D image dummy for the sake of CNN and fully connected or dense layers, batch normalization, and Conv2D layers in step 1&2 with max pooling. Max pooling was then applied on these convolutions after which there was a merge layer between everything else until output nodes where ReLU followed by softmax activation would come into play. In addition, the GridSearch method was utilized to adjust SVM model hyperparameters optimally to increase the model's performance, the ensemble model showed great efficiency in terms of accuracy, precision, recall, F1 score, and AUC, thus being perceived as a stable and well-performing model for identifying intrusions. Improvements in the future will not only enhance the efficiency of the model but also promote its extendibility and trustworthiness in real-world network security environments.

Declarations

Ethics Approval and Consent to Participate

The results/data/figures in this manuscript have not been published elsewhere, nor are they under consideration by another publisher. All the material is owned by the authors, and/or no permissions are required.

Consent for Publication

This article does not contain any studies with human participants or animals performed by any of the authors.

Availability of Data and Materials

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Competing Interests

The authors declare no competing interests in the research.

Funding
This research was not supported by any funding agency or institute.

Author Contribution
All authors contributed equally to this research.

Acknowledgment
The author is grateful to the editorial and reviewers, as well as the correspondent author, who offered assistance in the form of advice, assessment, and checking during the study period.

References

[1] Evans, P.C. and M. Annunziata, Industrial internet: Pushing the boundaries. General Electric Reports, 2012: p. 488-508.
[2] Abdulganiyu, O.H., T. Ait Tchakoucht, and Y.K. Saheed, A systematic literature review for network intrusion detection system (IDS). International journal of information security, 2023. 22(5): p. 1125-1162.
[3] Ashoor, A.S. and S. Gore, Importance of intrusion detection system (IDS). International Journal of Scientific and Engineering Research, 2011. 2(1): p. 1-4.
[4] Saranya, T., et al., Performance analysis of machine learning algorithms in intrusion detection system: A review. Procedia Computer Science, 2020. 171: p. 1251-1260.
[5] Ahmad, Z., et al., Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies, 2021. 32(1): p. e4150.
[6] Vinayakumar, R., K. Soman, and P. Poornachandran. Applying convolutional neural network for network intrusion detection. in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). 2017. IEEE.
[7] Mohammadi, M., et al., A comprehensive survey and taxonomy of the SVM-based intrusion detection systems. Journal of Network and Computer Applications, 2021. 178: p. 102983.
[8] Bhati, B.S., et al., An improved ensemble based intrusion detection technique using XGBoost. Transactions on emerging telecommunications technologies, 2021. 32(6): p. e4076.
[9] Elsayed, M.S., et al., Network Anomaly Detection Using LSTM Based Autoencoder, in Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks. 2020, Association for Computing Machinery: Alicante, Spain. p. 37–45.
[10] Elsayed, M.S., et al. The Role of CNN for Intrusion Detection Systems: An Improved CNN Learning Approach for SDNs. 2021. Cham: Springer International Publishing.
[11] Thakur, S., et al., Intrusion detection in cyber-physical systems using a generic and domain specific deep autoencoder model. Computers & Electrical Engineering, 2021. 91: p. 107044.
[12] Said, R.B., Z. Sabir, and I. Askerzade, CNN-BiLSTM: A Hybrid Deep Learning Approach for Network Intrusion Detection System in Software-Defined Networking With Hybrid Feature Selection. IEEE Access, 2023. 11: p. 138732-138747.
[13] Liu, G., et al., An Enhanced Intrusion Detection Model Based on Improved kNN in WSNs. Sensors, 2022. 22(4): p. 1407.
[14] Balyan, A.K., et al., A Hybrid Intrusion Detection Model Using EGA-PSO and Improved Random Forest Method. Sensors, 2022. 22(16): p. 5986.
[15] Wang, K., et al., Abnormal traffic detection system in SDN based on deep learning hybrid models. Computer Communications, 2024. 216: p. 183-194.
[16] Desai, R. and T. Venkatesh. Robust Network Intrusion Detection Systems for Outlier Detection. in 2022 IEEE 27th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD). 2022. IEEE.
[17] Cortes, C. and V. Vapnik, Support-vector networks. Machine learning, 1995. 20: p. 273-297.
[18] Chen, T. and C. Guestrin. Xgboost: A scalable tree boosting system. in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016.
[19] Yan, M., et al., Bearing remaining useful life prediction using support vector machine and hybrid degradation tracking model. ISA transactions, 2020. 98: p. 471-482.

[20] Sukhdeve, D.S.R. and S.S. Sukhdeve, Google Colaboratory, in Google Cloud Platform for Data Science: A Crash Course on Big Data, Machine Learning, and Data Analytics Services. 2023, Springer. p. 11-34.
[21] Gulli, A. and S. Pal, Deep learning with Keras. 2017: Packt Publishing Ltd.

Received: 07 Mar 2024, **Revised:** 25 Aug 2024,
Accepted: 25 Sep 2024, **Available online:** 01 Oct 2024.

© 2024 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).

Disclaimer/Publisher's Note: The perspectives, opinions, and data shared in all publications are the sole responsibility of the individual authors and contributors, and do not necessarily reflect the views of Sciences Force or the editorial team. Sciences Force and the editorial team disclaim any liability for potential harm to individuals or property resulting from the ideas, methods, instructions, or products referenced in the content.