Towards a Network Intrusion Classification System using Machine Learning

Luna dos Santos Almeida[1] **, Fabiola Spredemann**[1] **, Analucia Schiaffino Morales**[1] **, Roberto Rodrigues-Filho**[2] **, Alison R. Panisson**[1]

[1] Department of Computing (DEC) – Federal University of Santa Catarina (UFSC)
Araranguá, Brazil

[2]Department of Computer Science (CIC) – University of Brasília (UnB)
Brasília, Brazil

lalmeida.rr@gmail.com, fabiola.spredemann@grad.ufsc.br,
analucia.morales@ufsc.br, roberto.filho@unb.br, alison.panisson@ufsc.br

Abstract. *With the expansion of the Internet and digital transformation, cybersecurity threats have increased, including attacks such as Denial of Service (DoS), ransomware, phishing, and trojans. Intrusion Detection Systems (IDS) play a crucial role in mitigating these threats, often leveraging artificial intelligence for enhanced detection. This study proposes a machine learning-based approach for cyber-attack classification, designed for integration into IDS. Using a multilayer neural network, the model successfully classified nine attack types and normal network activities, achieving an accuracy of 90.49% through optimized intermediate layer configurations.*

1. Introduction

In April 2025, internet access reached 5.64 billion individuals, encompassing 68.7% of the world's population, according to DataReportal[1]. The connected population increased by 144 million over the previous year, representing an annual growth rate of 2.6%. This sustained expansion positions internet users as a substantial majority, significantly outnumbering those without access. Digital transformation initiatives have further accelerated the development of interconnected systems, making the accurate assessment of this growth imperative. However, it is important to acknowledge that published statistics may underestimate the actual growth rate due to inherent reporting delays. While these advancements foster unprecedented connectivity, they concurrently expose computers and data to heightened privacy and security risks from cyberattacks [Saheed et al. 2022].

Cyberattacks, often driven by automated scripts, exploit system vulnerabilities, occurring approximately every 39 seconds. This reality means the current cybersecurity landscape is characterized by the relentless emergence of novel cyber threats and trends, presenting increasingly sophisticated and diverse challenges to both individuals and organizations [Aslan et al. 2023, Admass et al. 2024]. Common types of cyberattacks include Denial of Service (DoS), ransomware, phishing, spoofing, backdoor exploitation, Trojan horses, and brute-force attacks [Buczak and Guven 2016, Du et al. 2023]. Furthermore, attackers frequently leverage network vulnerabilities through methods such

[1]https://datareportal.com/

as scanning, Remote to Local, and User to Root exploits [Jang-Jaccard and Nepal 2014, Hesham et al. 2024, Makhdoomi et al. 2025].

The ongoing expansion of technology, encompassing diverse platforms, Big Data analytics, and novel communication protocols, significantly complicates the task of maintaining robust cybersecurity. To address these complex threats, the field of cybersecurity is being transformed by advanced technologies: artificial intelligence (AI) and machine learning (ML) are actively employed to refine threat detection and response, while cloud computing and quantum computing are also emerging as powerful tools in this critical domain [Admass et al. 2024]. In this context, Intrusion Detection Systems (IDS) are essential components of cybersecurity frameworks, responsible for identifying and mitigating potential threats in network environments. Among the advantages of employing ML solutions for IDS tasks are the reduction of false alarms, improvements in threat detection accuracy, and the ability to provide immediate responses to security events [Ali et al. 2024, Makhdoomi et al. 2025].

Machine learning is widely utilized in intrusion detection for its capacity to analyze large datasets, recognize patterns, and predict or classify threats [Golande et al. 2024]. Key applications include machine learning algorithms capable of being trained for anomaly detection, identifying normal network traffic patterns, and detecting deviations that may indicate potential intrusions [Hesham et al. 2024]. Additionally, machine learning models can classify network traffic into various categories of attacks, facilitating the implementation of targeted defense mechanisms. By analyzing historical data, machine learning models can predict potential future attacks, thereby enabling proactive security measures [Golande et al. 2024]. Several machine learning algorithms, especially ensemble methods like Random Forest and Gradient Boosting, are adept at handling imbalanced datasets where instances of attacks are scarce compared to normal traffic [Fathima et al. 2023, Ali et al. 2024].

This study proposes a machine learning-based approach to classify cyberattacks using an artificial neural network, which can be integrated into an IDS for autonomous decision-making, such as blocking attacks or notifying administrators [Sabahi and Movaghar 2008, Liao et al. 2013, Kumar and Gutierrez 2025]. Activities like searching for network vulnerabilities and performing integrity tests are not covered here. This study introduces a machine learning-based network intrusion detection system capable of classifying nine distinct attack types and normal traffic, thereby providing detailed and actionable outputs for IDS integration. The proposed multilayer neural network, optimized with five hidden layers and specific neuron configurations, achieved an accuracy of 90.49%. Performance evaluation using standard metrics demonstrated the model's robustness, with a precision of 91%, recall of 90%, and an F1-score of 90%. These results highlight the system's effectiveness in enhancing the granularity and reliability of cyberattack detection, representing a significant advancement over traditional binary IDS approaches.

The present paper is organized into six sections. Section 2 reviews the theoretical foundations of artificial intelligence and artificial neural networks, with a focus on multilayer perceptrons for classification tasks. It covers the architecture comprising input, hidden, and output layers, training through backpropagation, and addresses potential issues such as overfitting. Section 3 describes the application of machine learning prin-

ciples to cyberattack detection, utilizing 40 network features to classify nine attack types and normal traffic. This section also details the use of the KDD-CUP-99 dataset, preprocessing techniques to mitigate class imbalance, and the conversion of textual data into numerical features. Section 4 presents the neural network architecture, including the number of layers and neurons, and specifies training parameters for the MLPClassifier from scikit-learn. It also discusses stopping criteria implemented to prevent overfitting. The model verification strategy is outlined, employing metrics such as accuracy, confusion matrix, precision, recall, and F1-score to evaluate performance. The results demonstrating the classifier's effectiveness across various attack types are discussed. Finally, Section 5 surveys alternative machine learning methods for intrusion detection, followed by conclusions and references.

2. Multilayer Perceptron

Multilayer perceptrons (MLPs) are composed of layered feedforward networks that are typically trained using the static backpropagation method [Russell and Norvig 2016]. They are prevalent in various static pattern classification tasks due to their simplicity and ability to approximate any input-output mapping. In deep learning, networks like MLPs solve classification problems by organizing neurons into layers, where data is processed through weighted inputs and activation functions [Wollowski et al. 2016]. However, MLPs often require a substantial amount of training data, typically around three times the number of network weights, and have a slow training process [Fathima et al. 2023].

Multilayer perceptrons (MLPs) confer significant advantages over conventional machine learning classifiers in intrusion detection system (IDS) applications. Their architectural design, characterized by multiple hidden layers and non-linear activation functions, facilitates the modeling of complex, non-linear patterns within network traffic data. This capability becomes increasingly critical given the escalating sophistication of cyber threats [Palenzuela et al. 2016]. In contrast to linear classifiers such as logistic regression and support vector machines, which often inadequately capture intricate relationships in high-dimensional feature spaces, MLPs demonstrate superior capacity to process large-scale, feature-rich datasets. Consequently, this enhances detection efficacy by effectively addressing the inherent complexities of network traffic analysis [Ali et al. 2024]. Nonetheless, due to the dynamic nature of network environments and the continual emergence of novel cyber threats, MLP-based systems necessitate frequent retraining and updates to sustain reliable performance in operational deployment [Borisenko et al. 2021].

This study is inspired by approaches designed for resource-constrained environments, such as the microcomputer-based solution presented in [de Almeida Florencio et al. 2018], which explores a lightweight multilayer perceptron (MLP) architecture for network intrusion classification. Building on this concept, we are interested in efficient models which can be integrated across multiple layers of an intrusion detection system, enabling broader applicability without compromising performance on limited hardware.

3. Network Intrusion Classification System

The same machine learning principles used for classical problems of classification can be applied to cyberattack detection by analyzing network connection characteristics to

identify normal events or threats [Ali et al. 2024, Admass et al. 2024]. By providing a machine learning algorithm with attack characteristics and classification labels, a model can be trained to classify different types of cyberattacks. This work explores the use of a multilayer perceptron neural network to classify cyberattacks, using 40 network characteristics to distinguish between 9 attack types and 1 class for normal connections.

3.1. Data for Model Training

Data acquisition is a key step in training the classification model. For this work, we used the public KDD-CUP-99 dataset, which was part of the Third International Knowledge Discovery and Data Mining Tools Competition[2]. The dataset contains network connection data labeled as normal or as one of several attack types. The data consists of sequences of TCP packets, each identified by a source and destination IP address and protocol. These connections are classified into four attack categories: DoS (e.g., back), R2L (e.g., warez client), U2R (e.g., buffer overflow), and Probe (e.g., ipsweep). The task is to build a classifier capable of distinguishing between normal connections and these attack types. Table 1 presents a detailed description of the features used in the proposed system.

Table 2 shows the number of records for each attack category in the dataset. The U2R category was excluded due to its small sample size. To address class imbalance, a preprocessing step was performed to balance the dataset, ensuring similar numbers of records for each class. Without this, the network would be biased towards classes with more data. The results of this preprocessing are shown in Table 4.

3.2. MLP Implementation

The multilayer perceptron neural network was developed using Python within the Jupyter Notebook environment, leveraging libraries including Pandas for data manipulation and Scikit-learn for model construction and evaluation. The dataset was imported into a DataFrame, and non-relevant features were subsequently removed to prepare the data for training. To address class imbalance, records were selectively excluded to ensure each class contained between 181 and 200 instances, as detailed in Table 4. Maintaining balanced class distributions is essential to prevent model bias, since disproportionate representation can cause the neural network to favor classes with higher sample counts [Ali et al. 2024].

The textual data was transformed into numerical values using the Label Encoder from the Sklearn library, as neural network algorithms require numerical data for processing. After encoding, the dataset was split into 70% for training and 30% for testing. This division ensures model evaluation with data not used during training, helping to assess the model's generalization ability. For the neural network architecture, the input layer had 40 neurons (corresponding to the dataset's characteristics), and the output layer had 10 neurons (representing 9 attack classes and 1 normal class). The optimal architecture included 5 hidden layers with 9, 9, 9, 10, and 10 neurons, as shown in Figure 1.

To train the model, the *MLPClassifier* algorithm from the Sklearn library was used, with the following parameters [Russell and Norvig 2016]:

- ***activation***: Defines the activation function. The hyperbolic tangent (*'tanh'*) was used, with outputs between -1 and 1, $f(x) = tanh(x)$.

[2]`http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`

Tabela 1. Features used by the proposed intrusion classification system.

Feature	Description
Protocol_type	Type of protocol, e.g., TCP, UDP.
Service	Destination network service, e.g., HTTP, Telnet.
Flag	Connection status: whether an error occurred or it completed normally.
Src_bytes	Number of data bytes from source to destination.
Dst_bytes	Number of data bytes from destination to source.
Land	1 if source and destination IP addresses and port numbers are equal; otherwise 0.
Wrong_fragment	Number of incorrect fragments.
Urgent	Number of urgent packets in this connection (urgent bit set).
Hot	Number of "hot" indicators in the content, such as accessing system directories, creating or executing programs.
Num_failed_logins	Number of failed login attempts.
Logged_in	1 if successfully logged in; otherwise 0.
Num_compromised	Number of compromised conditions.
Root_shell	1 if root shell was obtained; otherwise 0.
Su_attempted	1 if the "su root" command was attempted; otherwise 0.
Num_root	Number of root accesses.
Num_file_creations	Number of file creation operations.
Num_shells	Number of shell prompts.
Num_access_files	Number of operations on access control files.
Num_outbound_cmds	Number of outbound commands in an FTP session.
Is_hot_login	1 if the login belongs to a "hot" list; otherwise 0.
Is_guest_login	1 if the login is a guest login; otherwise 0.
Count	Number of connections to the same host as the current one in the last two seconds.
Srv_count	Number of connections to the same service as the current one in the last two seconds.
Serror_rate	Percentage of connections with flags S0, S1, S2, or S3 among those in Count.
Srv_serror_rate	Percentage of connections with flags S0, S1, S2, or S3 among those in Srv_count.
Rerror_rate	Percentage of connections with "REJ" errors.
Srv_rerror_rate	Percentage of connections with the REJ flag among those in Srv_count.
Same_srv_rate	Percentage of connections to the same service among those in Count.
Diff_srv_rate	Percentage of connections to different services among those in Count.
Srv_diff_host_rate	Percentage of connections to different target machines among those in Srv_count.
Dst_host_count	Number of connections with the same destination IP address.
Dst_host_srv_count	Number of connections with the same destination port number.
Dst_host_same_srv_rate	Percentage of connections to the same service among those in Dst_host_count.
Dst_host_diff_srv_rate	Percentage of connections to different services among those in Dst_host_count.
Dst_host_same_src_port_rate	Percentage of connections with the same source port among those in Dst_host_srv_count.
Dst_host_srv_diff_host_rate	Percentage of connections to different destination machines among those in Dst_host_srv_count.
Dst_host_serror_rate	Percentage of connections with flags S0, S1, S2, or S3 among those in Dst_host_count.
Dst_host_srv_serror_rate	Percentage of connections with flags S0, S1, S2, or S3 among those in Dst_host_srv_count.
Dst_host_rerror_rate	Percentage of connections with the REJ flag among those in Dst_host_count.
Dst_host_srv_rerror_rate	Percentage of connections with the REJ flag among those in Dst_host_srv_count.

- ***solver***: The optimizer for weight optimization. *'Adam'*, a stochastic gradient-based optimizer, was chosen.
- ***learning_rate_init***: Controls the step size for weight updates.
- ***shuffle***: Shuffles data during each iteration.
- ***tol***: The stopping criterion threshold, set to 0.0001.
- ***epsilon***: A stability parameter for *'adam'*, set to 0.00000001 to prevent division-by-zero errors.
- ***n_iter_no_change***: The maximum number of epochs with no improvement based on *tol*, set to 10.

The model was validated by evaluating its logarithmic loss, a measure of classification performance based on predicted probabilities. The loss increases as predictions deviate from the true labels, and a value closer to 0 indicates a better model. In this study, the model achieved a logarithmic loss of 0.2. Subsequently, predictions were made on the test dataset, and the classification results were compared with expected labels. Accuracy

Tabela 2. Number of records in the dataset.

Classes	Number of Records
Normal	97278
DDoS	391458
U2R	22
R2L	1126
Probe	4137

Tabela 3. Number of records per attack subcategory before preprocessing.

Attack Class	Attack Type	Number of Records
DoS	back	196
	neptune	8282
	smurf	529
	teardrop	188
Probe	satan	691
	ipsweep	710
	nmap	301
	portsweep	587
R2L	warezclient	181

metrics and a confusion matrix were then calculated to assess the model's performance, with further details on validation provided in Section 4.

3.3. Stopping Criteria

Stopping criteria are essential in neural network-based models to determine when training should conclude. Various criteria exist, most of which focus on minimizing network error. Common stopping methods include limiting the number of training cycles, monitoring error thresholds, and using validation-based techniques. The number of cycles controls how many times the training set is presented to the network. An excessive number can lead to overfitting, while too few may cause underfitting. Error-based criteria stop training once the mean squared error reaches a predefined threshold, though a small error does not guarantee good generalization. In validation-based stopping, training halts when error on the test set starts increasing, indicating a loss of generalization ability.

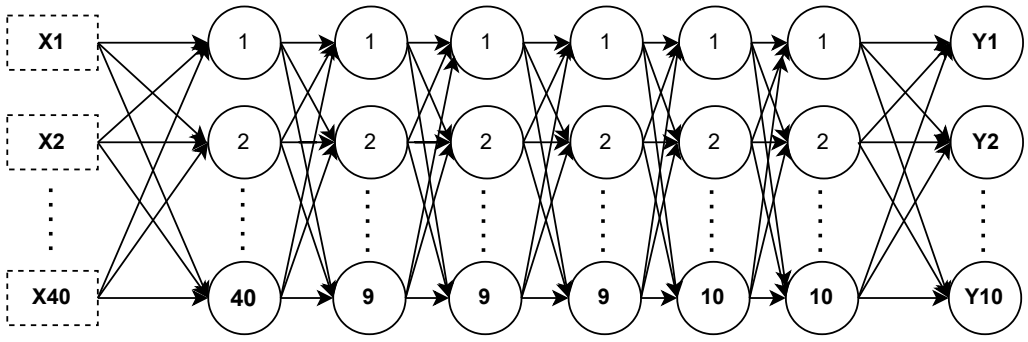

Figura 1. MLP developed architecture.

Tabela 4. Number of records per attack subcategory after preprocessing.

Attack Class	Attack Type	Number of Records
DoS	back	196
	neptune	200
	smurf	200
	teardrop	188
Probe	satan	200
	ipsweep	200
	nmap	200
	portsweep	200
R2L	warezclient	181

For the model in this work, the stopping criterion was based on the 'tol' parameter from the MLPClassifier module of the Sklearn library. This parameter defines the tolerance for optimization, with training halting if the error or score fails to improve by at least 'tol' for 'n_iter_no_change' consecutive iterations. In this case, 'tol' was set to 0.0001, and 'n_iter_no_change' was set to 10, meaning training stops if there is no significant improvement (to the fourteenth decimal place) over 10 epochs.

3.4. Model Verification Strategy

After training, the neural network's performance was evaluated using the accuracy metric. This was calculated using the function `metrics.accuracy_score(test_model, y_test)` from the sklearn metrics library. The accuracy score calculates subset accuracy, meaning that the predicted output set must match exactly with the actual output set in the *test_model*. In addition to accuracy, it is essential to evaluate how well the model classifies data for each individual class. This is achieved by generating a confusion matrix, which compares the predicted labels against the actual labels, providing a detailed view of the classification results. The confusion matrix allows us to observe the frequency of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN), which help to assess the model's performance across different classes.

Based on the confusion matrix, key performance metrics such as Precision, Recall, and F1-score were calculated. Precision is the ratio of True Positives to the sum of True Positives and False Positives, indicating how well the model avoids misclassifying negative samples. Recall, on the other hand, is the ratio of True Positives to the sum of True Positives and False Negatives, highlighting the model's ability to correctly identify all positive samples. The F1-score is the harmonic mean of Precision and Recall, providing a balanced measure of performance, with a best value of 1 and worst value of 0. Additionally, the Support metric indicates the number of occurrences of each class in the test set, providing insight into the distribution of classes. These metrics are essential for evaluating the model's overall effectiveness and identifying areas for improvement in its classification abilities.

4. Results and Discussion

The IDS system proposed in this study utilizes an MLP-based model for classifying nine distinct types of attacks and identifying normal network activities. This is a significant

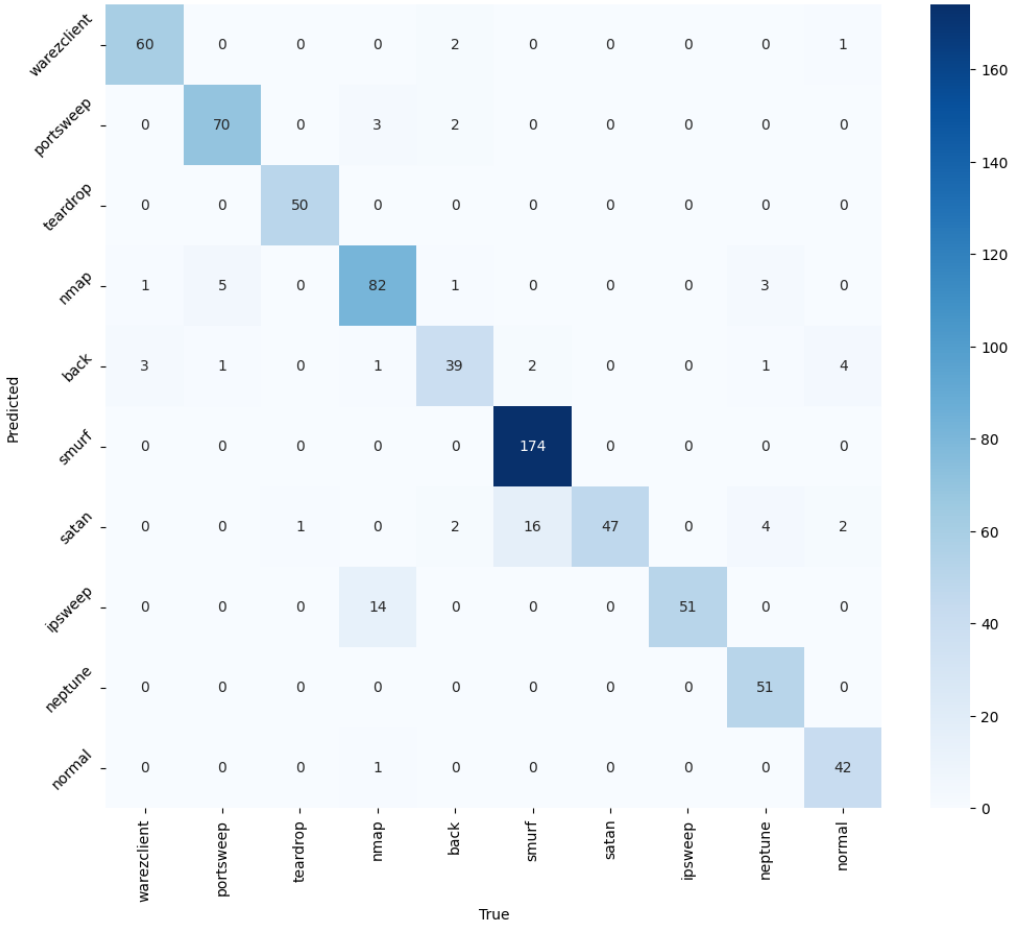

Figura 2. Confusion Matrix.

contribution compared to traditional systems that typically classify only two types of output—attack or normal situation [Buczak and Guven 2016]. The ability to classify specific attack types provides a key advantage, allowing for faster and more accurate decision-making, which enables quicker activation of protective measures once an attack is identified. This helps mitigate potential damage and facilitates prompt system recovery, improving the overall effectiveness of the security system. Figure 2 shows the confusion matrix resulting from the tests performed on the model.

The model achieved an accuracy of 98% using a five-layer configuration with 9, 9, 9, 10, and 10 neurons in each layer. This performance is considered good, as an accuracy between 87.5% and 100% is typically expected for such systems [Alsmadi et al. 2009]. The confusion matrix revealed that the attack types *ipsweep* and *satan* were classified correctly for all tests, while *nmap* and *smurf* exhibited higher error rates. However, despite these misclassifications, the model's ability to detect the presence of network attacks, especially with nine output categories, still demonstrates its potential. Additionally, the model achieved a weighted average recall of 90%, precision of 91%, and an F1-score of 90%, indicating a strong balance between detecting positive samples and minimizing misclassifications. These results highlight the model's robust performance and effectiveness in real-world cybersecurity applications. Table 5 presents the detailed performances.

Tabela 5. Performance by Class

Class	Accuracy	Precision	Recall	F1-score
warezclient	0.99	0.93	0.95	0.94
portsweep	0.98	0.92	0.93	0.92
teardrop	0.99	0.98	1.00	0.99
nmap	0.96	0.81	0.89	0.84
back	0.97	0.84	0.76	0.80
smurf	0.97	0.90	1.00	0.95
satan	0.96	1.00	0.65	0.78
ipsweep	0.98	1.00	0.78	0.87
neptune	0.98	0.86	1.00	0.92
normal	0.98	0.85	0.97	0.91
Mean	**0.98**	**0.91**	**0.90**	**0.90**

5. Related Work

Over the past decades, various studies have investigated multiple machine learning techniques for intrusion detection in computer networks. These studies have focused on optimizing accuracy, efficiency, and scalability. Multi-layer perceptrons have consistently demonstrated their robustness and effectiveness in attack classification within intrusion detection systems, exhibiting high accuracy and reliability in identifying a wide range of network attacks.

The comparative analysis presented in [Mukkamala et al. 2002] evaluates the performance of neural networks and support vector machines (SVMs) for intrusion detection. Results indicate that although SVMs slightly outperform neural networks in detection accuracy, combining both models via ensemble methods yields superior overall performance. Notably, SVMs demonstrated significantly shorter training times (17.77 seconds), compared to neural networks, which required approximately 18 minutes. Their inference times were also considerably lower, underscoring the suitability of SVMs for environments where rapid retraining is essential, such as adapting to emerging attack patterns. While SVMs excel in speed and scalability for binary classification, neural networks are advantageous for multi-class intrusion identification, due to their ability to model complex attack categorizations.

In [Ramos and Santos 2011], a classifier committee approach was proposed, integrating multiple machine learning algorithms to improve anomaly classification accuracy. The combined system enhanced detection precision but incurred high computational costs, limiting practicality when scaling to large datasets.

The work in [Henke et al. 2011] introduces an ensemble of k-nearest neighbors (k-NN) classifiers generated through the random subspace method (RSM). This ensemble outperformed the hybrid Triangle Area based Nearest Neighbor (TANN) method by achieving higher detection rates and reducing false alarms. The results confirm that RSM-based ensemble classifiers significantly enhance network anomaly detection, demonstrating higher accuracy and lower false positive rates compared to individual classifiers or hybrid approaches. Future research could explore the application of this method to more recent attack datasets and evaluate alternative ensemble strategies, such as bagging or

heterogeneous ensembles.

In [Ertam et al. 2017], the authors examine the application of multiple machine learning classifiers to distinguish between normal and abnormal internet traffic. Using the KDD Cup 99 dataset, the study provides insights into the trade-offs between detection accuracy and classification time, informing the development of more efficient intrusion detection systems.

Additionally, [Neto and Gomes 2019] evaluates diverse machine learning algorithms and data resampling techniques using the CICIDS2017 dataset, which includes 78 features and multiple attack types, such as Brute Force, Denial of Service, Heartbleed, and Web Attacks. Classifiers, including Decision Trees, Random Forests, and Multi-layer Perceptrons, exhibited high accuracy and low inference times, underscoring their potential for real-time intrusion detection systems.

The authors describe the use of advanced neural network architectures for effective intrusion detection [Borisenko et al. 2021]. Ensemble and hybrid models combining MLPs with other machine learning or deep learning models have produced notable results. For instance, a hybrid model combining MLP with LSTM achieved high detection accuracy on the CSE-CIC-IDS2018 dataset. These findings could support the development of next-generation IDSs that are capable of classifying network traffic, identifying various cyber attacks, and enhancing network security through hybrid ANN models.

And finally, in [Ali et al. 2024] the authors proposed enhancing network IDS using the Bidirectional Encoder Representations from Transformers (BERT) model, MLP, and Synthetic Minority Over-sampling Technique (SMOTE). It addresses the challenge of class imbalance in network traffic data, critical for effective intrusion detection. Tested on CIC-IDS 2017, UNSW-NB 2015, and NSL-KDD 2009 datasets, the approach shows that combining BERT for feature extraction, SMOTE for balance, and MLP for classification improves intrusion detection in imbalanced traffic. The results highlight its potential to advance network security. Future work will tackle computational intensity and diversify datasets for better applicability.

6. Conclusion

Based on the results, the developed MLP successfully achieved the objective of identifying both network threats and normal activities. The classification task was performed efficiently, as evidenced by the accuracy and other metrics obtained. However, it was also observed that the efficiency of the model's convergence depends on the number of neurons and hidden layers, with certain configurations leading to suboptimal results.

The ability to classify specific attack types highlights the viability of using MLP as a solution for intrusion detection. Nevertheless, model deviations should not be overlooked, as they could lead to potential security risks. It is also recommended to expand the dataset used for training and explore additional classification parameters not considered in this study. Lastly, the promising performance indices suggest that the solution could be deployed on devices with lower processing power. As future work, applying this model to classify attacks on Internet of Things (IoT) devices will be explored.

Referências

Admass, W. S., Munaye, Y. Y., and Diro, A. A. (2024). Cyber security: State of the art, challenges and future directions. *Cyber Security and Applications*, 2:100031.

Ali, Z., Tiberti, W., Marotta, A., and Cassioli, D. (2024). Empowering network security: Bert transformer learning approach and mlp for intrusion detection in imbalanced network traffic. *IEEE Access*, 12:137618–137633.

Alsmadi, M. k., Omar, K. B., Noah, S. A., and Almarashdah, I. (2009). Performance comparison of multi-layer perceptron (back propagation, delta rule and perceptron) algorithms in neural networks. In *2009 IEEE International Advance Computing Conference*, pages 296–299.

Aslan, Ö., Aktuğ, S. S., Ozkan-Okay, M., Yilmaz, A. A., and Akin, E. (2023). A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. *Electronics*, 12(6):1333.

Borisenko, B., Erokhin, S., Fadeev, A., and Martishin, I. (2021). Intrusion detection using multilayer perceptron and neural networks with long short-term memory. In *2021 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO)*, pages 1–6. IEEE.

Buczak, A. L. and Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176.

de Almeida Florencio, F., Moreno, E. D., Macedo, H. T., de Britto Salgueiro, R. J., do Nascimento, F. B., and Santos, F. A. O. (2018). Intrusion detection via mlp neural network using an arduino embedded system. In *2018 VIII Brazilian symposium on computing systems engineering (SBESC)*, pages 190–195. IEEE.

Du, D., Zhu, M., Li, X., Fei, M., Bu, S., Wu, L., and Li, K. (2023). A review on cybersecurity analysis, attack detection, and attack defense methods in cyber-physical power systems. *Journal of Modern Power Systems and Clean Energy*, 11(3):727–743.

Ertam, F., Kilincer, L. F., and Yaman, O. (2017). Intrusion detection in computer networks via machine learning algorithms. In *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pages 1–4.

Fathima, A., Khan, A., Uddin, M. F., Waris, M. M., Ahmad, S., Sanin, C., and Szczerbicki, E. (2023). Performance evaluation and comparative analysis of machine learning models on the unsw-nb15 dataset: a contemporary approach to cyber threat detection. *Cybernetics and Systems*, pages 1–17.

Golande, S. V., Vaidya, S., Pardeshi, A., Katkade, V., and Pawar, V. (2024). An efficient network intrusion detection and classification system using machine learning. *learning*, 4(1).

Henke, M., Costa, C., dos Santos, E. M., and Souto, E. (2011). Detecção de intrusos usando conjunto de k-nn gerado por subespaços aleatórios.

Hesham, M., Essam, M., Bahaa, M., Mohamed, A., Gomaa, M., Hany, M., and Elsersy, W. (2024). Evaluating predictive models in cybersecurity: A comparative analysis of

machine and deep learning techniques for threat detection. In *2024 Intelligent Methods, Systems, and Applications (IMSA)*, pages 33–38. IEEE.

Jang-Jaccard, J. and Nepal, S. (2014). A survey of emerging threats in cybersecurity. *Journal of computer and system sciences*, 80(5):973–993.

Kumar, A. and Gutierrez, J. A. (2025). Impact of machine learning on intrusion detection systems for the protection of critical infrastructure. *Information*, 16(7).

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., and Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24.

Makhdoomi, P. M. S., Ikhlas, M., Khursheed, A., Hilal, F., Ahmad Najar, Z., Hameed, J., and Sharma, S. (2025). Network-based intrusion detection: a comparative analysis of machine learning approaches for improved security. *Journal of Cyber Security Technology*, pages 1–28.

Mukkamala, S., Janoski, G., and Sung, A. (2002). Intrusion detection using neural networks and support vector machines. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, volume 2, pages 1702–1707 vol.2.

Neto, M. S. and Gomes, D. G. (2019). Network intrusion detection systems design: A machine learning approach. In *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 932–945, Porto Alegre, RS, Brasil. SBC.

Palenzuela, F., Shaffer, M., Ennis, M., Gorski, J., McGrew, D., Yowler, D., White, D., Holbrook, L., Yakopcic, C., and Taha, T. M. (2016). Multilayer perceptron algorithms for cyberattack detection. In *2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS)*, pages 248–252. IEEE.

Ramos, A. and Santos, C. (2011). Combinando algoritmos de classificação para detecção de intrusão em redes de computadores. In *Anais do XI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 211–224, Porto Alegre, RS, Brasil. SBC.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. pearson.

Sabahi, F. and Movaghar, A. (2008). Intrusion detection: A survey. In *2008 Third International Conference on Systems and Networks Communications*, pages 23–26. IEEE.

Saheed, Y. K., Abiodun, A. I., Misra, S., Holone, M. K., and Colomo-Palacios, R. (2022). A machine learning-based intrusion detection for detecting internet of things network attacks. *Alexandria Engineering Journal*, 61(12):9395–9409.

Wollowski, M., Selkowitz, R., Brown, L., Goel, A., Luger, G., Marshall, J., Neel, A., Neller, T., and Norvig, P. (2016). A survey of current practice and teaching of ai. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.