A Methodological and Analytical Framework for Machine Learning and Deep Learning–Based Intrusion Detection Systems

Faisal Alhumaymidi

Department of Cybersecurity,

Onaizah Colleges, Onaizah, AL Qassim, Saudi Arabia,

Email: 451110850@oc.edu.sa

Khalid Aldriwish

Department of Computer science, The College of Computer Sciences and Information,

Majmaah University, AL Majmaah 11952, Saudi Arabia,
Email: k.aldriwish@mu.edu.sa

Abstract- An important part of a modern cybersecurity solution is an Intrusion Detection System (IDS). They find out about changes that are happening in networks. Numerical performance metrics are the main focus of most research on machine learning and deep learning methods for Intrusion Detection Systems (IDS). This means that people tend to rely too much on numerical performance instead of looking at how the model behaves, how well it works, and how easy it is to design and deploy. We offer a thorough, systematic, and analytical framework for intrusion detection that uses classical machine learning, a deep learning architecture, and the NSL-KDD dataset as a benchmark. The suggested framework encompasses systematic data preprocessing, feature selection, class imbalance mitigation, and standardized evaluation protocols. In addition, there are new deep learning architectures, such as Convolutional Neural Networks, Long Short-Term Memory networks, and a mix of CNN and LSTM formats. This work provides a detailed analytical analysis rather than only revealing single performance metrics which can be used to characterize behavior of algorithms, architectural design, and practical deployment considerations. The suggested framework is great for applied research and making real-world IDS. and it sets up a way to repeat the process for future realistic validation.

Keywords – Intrusion Detection System; Machine Learning; Deep Learning; Network Security; Analytical Framework; NSL-KDD

I. INTRODUCTION

Intrusion Detection Systems (IDS) are vital for keeping modern networks safe because they keep monitoring system activities and traffic patterns all the time to find unacceptable behavior. Cloud computing, mobile networks, and Internet of Things (IoT) devices make networks more complicated. Because of this, traditional rule-based and signature-driven intrusion detection methods are not good enough to find new and more advanced attacks [1,2,4,13].

Also, IDS are very important for keeping modern networks safe because they watch how traffic moves and find bad behavior. As network traffic becomes more complex and prevalent, conventional rule-based and signature-driven detection methods are insufficient to address novel and previously unobserved attack behaviors. So, data-driven methods, especially in both machine learning (ML) and deep learning (DL) approaches, have set and gotten a set of attentions in the last few years.

Current IDS research frequently emphasizes quantitative performance metrics, providing limited analytical discussion on model selection, architectural design, evaluation methodology, and deployment feasibility. These kinds of limits make it harder to get the same results again and harder to use them in real life. This study addresses existing deficiencies by presenting a unified methodological and analytical framework that integrates traditional machine learning algorithms with contemporary deep learning architectures for intrusion detection.

Recent progress in such deep learning (DL) has greatly improved intrusion detection by making it possible to automatically extract features and learn hierarchical representations. Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) networks have been effectively employed in Intrusion Detection Systems (IDS), exhibiting improved detection capabilities, particularly for complex and time-sensitive attack patterns [4,5,16]. Even with these improvements, a lot of current IDS research mostly talks about numbers like accuracy or detection rate and doesn't go into much detail about how the model works, what causes false positives, how to design an evaluation, or how to

deploy it. [2,6]. In addition, differences in preprocessing pipelines and evaluation protocols often make it hard to reproduce results and compare different approaches fairly.

The proposed framework creates a single, repeatable IDS pipeline that includes data preprocessing, feature optimization, handling class imbalance, model comparison, and deployment-oriented analysis. The framework is tested with the NSL-KDD dataset [3,25], which makes it easy to compare with previous IDS studies.

This study contributed to the creation of a full and flexible IDS framework that allows for systematic preprocessing, feature engineering, algorithmic comparison, and deployment-oriented analysis. As stated in Figure 1 below, the framework is set and considered to figure out and capture both spatial correlations and temporal dependencies in network traffic data by using deep architectures like CNN, LSTM, and a hybrid CNN–LSTM model which will be demonstrated in the next section in this study. However, the illustrated figure presents a high-level Intrusion Detection System (IDS) framework that emphasizes dynamic interactions among a set of core components involved in intelligent security monitoring and decision-making. It sets and represents the overall security state of the monitored environment e.g., network, IoT ecosystem. These elements reflect the IDS's role in maintaining system integrity, availability, and resilience against cyber threats.

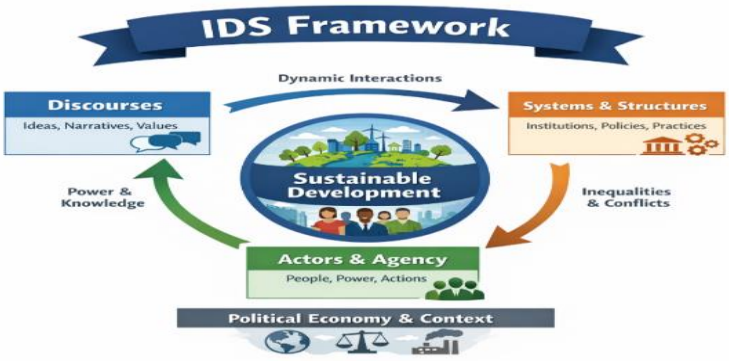

Figure 1. Conceptual IDS framework and associated entities.

The structure of this research study is figured out as: Part 1. Shows an introduction of IDS. Part 2. illustrates Literature Review. In Part 3, defines the proposed analytical framework. Part 4 deeply explains the Machine Learning and Deep Learning Models and provides a formal grounding of mathematical formulation. Part 5. Discussion and evaluation of proposed framework. Part 6. Demonstrates an Algorithmic Description and Model Architecture. The methodology and implementation are stated in Part 7. Finally remarks the study conclusion.

II. LITERATURE REVIEW

Early intrusion detection research focused on traditional machine learning techniques applied to benchmark datasets i.e., KDD Cup 99 and NSL-KDD. Algorithms including Decision Trees, Naïve Bayes, Support Vector Machines, and Random Forests demonstrated strong detection performance for known attack types while maintaining relatively low computational overhead [1,3,25].

With the emergence of deep learning, researchers explored neural network–based IDS to overcome the limitations of handcrafted feature engineering. Javaid et al. applied deep neural networks to network traffic data and demonstrated improved detection accuracy compared to traditional ML models [5]. Yin et al. further showed that recurrent neural networks and LSTM architectures are effective in modeling temporal dependencies in network traffic, giving and enabling the detection of evolving attack behaviors [4].

Convolutional Neural Networks have also been adapted for intrusion detection by reshaping network traffic features into matrix representations, allowing the extraction of local spatial correlations [7]. More recently, hybrid CNN–LSTM architectures have been proposed to jointly model spatial and temporal characteristics of network traffic, offering improved detection of sophisticated and multi-stage intrusions [2,11]. In addition, limited attention has been given to computational complexity and deployment feasibility, particularly in resource-constrained IoT and edge environments [10].

This work, on the other hand, emphasizes methodology, analysis, and deployment awareness. It provides a unified framework that adds to algorithm-centric IDS research instead of replacing it.

2.1 Intrusion Detection Systems

An intrusion is defined as any unauthorized or malicious attempt to compromise a system's security policies. IDSs are set and designed to make monitoring both system and network activities and respond generate alerts when suspicious behavior is detected. IDS can be broadly classified into network-based IDS (NIDS) and host-based IDS (HIDS), depending on the data source and monitoring location. [4,5,10]

Detection approaches are typically classified as signature-based or anomaly-based. Signature-based methods rely on predefined patterns and are effective for known attacks, whereas anomaly-based methods model normal behavior and detect deviations, enabling the detection of previously unseen threats.

2.2 Machine Learning for IDS

An intrusion is when someone tries to break into a system's security policies without permission or with bad intentions. Intrusion Detection Systems keep an eye on system or network activity and send alerts when they see anything strange. In most studies, and as mentioned above, these types of IDS are: (NIDS) and (HIDS). The type of IDS depends on where the data comes from and where it is being monitored. [4,5,6].
Most of the time, detection methods are divided into two types: signature-based and anomaly-based. Signature-based methods depend on patterns that have already been set up and work well for attacks that have already happened. Anomaly-based methods.

Despite significant progress, existing IDS research exhibits several limitations. Many studies report high accuracy without analyzing false positives or attack-wise detection performance. Additionally, evaluation methodologies often vary, making direct comparison difficult. Finally, limited attention is given to deployment feasibility, specifically in milieu and its associated parts with constrained computational resources.

2.3 Machine Learning for Intrusion Detection Systems

Researchers in IDS have used machine learning methods a lot because they can learn from data and apply what they learn to new attack patterns. Some examples of classical algorithms are e.g., decision Trees, Random Forests, Support Vector Machines, and Naïve Bayes, which that have been thoroughly tested on benchmark datasets i.e., KDD99 and NSL-KDD. Deep learning methods have been looked into more recently, but they are hard to use in real life because of their cost and hard to understand.

III. PROPOSED ANALYTICAL FRAMEWORK

Figure 2 below shows that the proposed framework provides a structured way to make and evaluate intrusion detection systems. The framework involves machine learning, data acquisition, preprocessing, feature selection, model training, evaluations, and deployment analysis. The real possibility here is that each step is meant to make things easier to reproduce and useful in the real world.

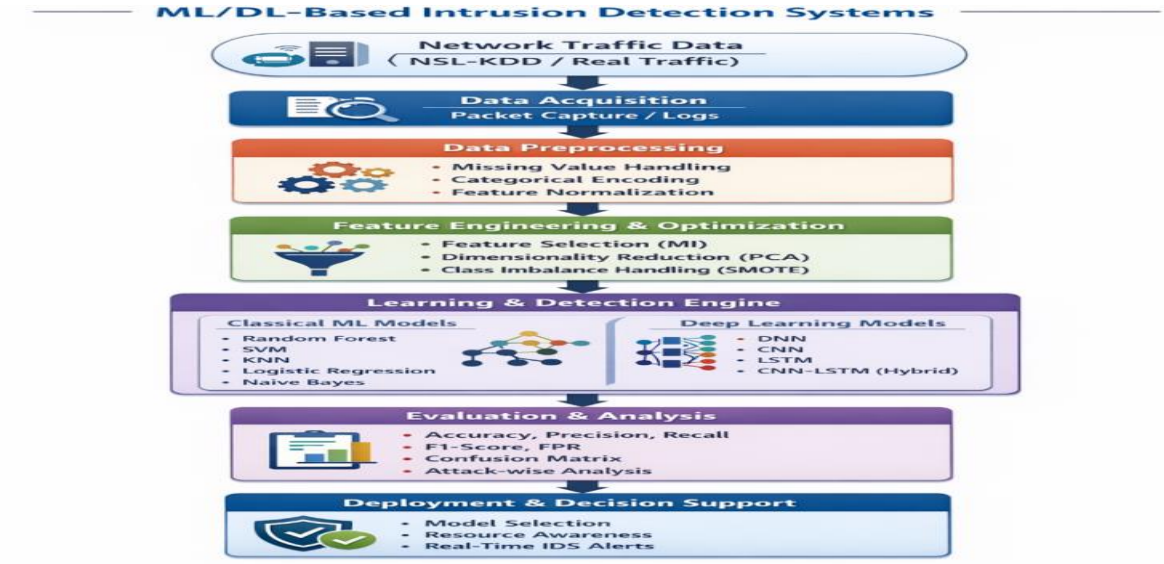

Figure 2. PROPOSED ANALYTICAL FRAMEWORK

3.1 Framework and Dataset

As mentioned above, the proposed framework employs the NSL-KDD dataset as a reference benchmark due to its improved class balance and removal of redundant records compared to the original KDD Cup 99 dataset [3,24]. The dataset includes labeled network traffic instances representing normal behavior and multiple attack categories, e.g., DoS, Probe, R2L, and U2R.

The Synthetic Minority Over-Sampling Technique (SMOTE) is used in the framework to improve the detection of minority attacks by setting and fixing class imbalance that is common in intrusion detection datasets [8]. To make learning more

efficient and cut down on the amount of time it takes to do calculations, we use feature selection and dimensionality reduction methods.

3.2 Dataset Description

The NSL-KDD dataset is used as a standard reference as it is widely used in IDS research. The dataset has labeled network traffic records that are either normal or belong to one of several attack types, e.g., Denial of Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R). [6,25]

3.3 Data Preprocessing

Handling missing values, encoding categorical features, and normalizing numerical attributes are all part of preprocessing. We use feature scaling to make sure that no one attributes take over the learning process. These steps are necessary to make the model more stable and useful in general.[7,25]

3.4 Feature Selection and Dimensionality Reduction

Feature selection is a way of obtaining rid of extra data and making math easier. To obtain and keep the most useful features, people use methods like Principal Component Analysis and mutual information analysis. Not only is reducing dimensionality making things work better, but it also makes models easier to understand.

3.5 Handling Class Imbalance

Intrusion detection datasets are typically imbalanced, with normal traffic significantly outnumbering attack instances. To address this issue, data balancing techniques e.g., Synthetic Minority Over-sampling Technique (SMOTE) are integrated into the framework to improve sensitivity toward minority attack classes.

IV. MACHINE LEARNING AND DEEP LEARNING MODELS

4.1 Classical Machine Learning Models

The framework is more effective at modeling attack patterns that are complex and not linear thanks to deep learning architecture. CNN-based IDS models obtain localized feature correlations from transformed network traffic matrices, while LSTM networks derive long-term temporal dependencies from sequential traffic data [4], [7]. A hybrid CNN-LSTM architecture is used to take advantage of both spatial and temporal learning abilities, which is in line with recent trends in IDS research [11].

4.2 Deep Learning–Based Intrusion Detection Models

To enhance the analytical depth and contemporary relevance of the framework, multiple deep learning architectures are integrated for representation learning and the identification of intricate patterns in network traffic data. [11,12,13,14]:

Deep Neural Networks (DNN): Fully connected deep neural networks are used to learn hierarchical feature representations from network traffic data that has already been processed. DNNs are especially good at finding non-linear connections between features that classical methods might not be able to model well.

Deep Neural Networks (DNN): Fully connected deep neural networks are used to learn hierarchical feature representations from network traffic data that has already been processed. DNNs are especially good at finding non-linear connections between features that classical methods might not be able to model well. Convolutional Neural Networks (CNN): CNN architectures convert the properties of network traffic into fixed-size matrices to make them useful for intrusion detection. Convolutional layers automatically find local spatial patterns and relationships between features. This makes it easier to find complex attack behaviors.

Long Short-Term Memory Networks (LSTM): LSTM networks are incorporated to model temporal dependencies in sequential network traffic data. By capturing long-term dependencies, LSTMs are well suited for detecting slow and stealthy attacks that unfold over time.

Hybrid CNN–LSTM Architecture: A hybrid deep learning model combining CNN and LSTM layers is included to exploit both spatial feature extraction and temporal sequence modeling. This architecture is particularly suitable for real-time intrusion detection in dynamic network environments.

V. MATHEMATICAL FORMULATIONS

This Mathematical Formulations section gives the models and methods in the framework a formal basis, which makes the proposed analytical framework more rigorous and easier to repeat. [28,29]

5.1 Machine Learning Models

Logistic Regression (LR)

Given a feature vector $x \in \mathbb{R}^d$, logistic regression models the probability of class $y \in \{0,1\}$ using the logistic function:

$$P(y = 1 \mid \mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}}$$

where $w \in \mathbb{R}^d$ is the weight vector and b is the bias term. The model is trained by minimizing the cross-entropy loss:

$$L(\mathbf{w}, b) = -\sum_{i=1}^{N} [\, y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

where $\hat{y}_i = P(y_i = 1 \mid \mathbf{x}_i)$.

Support Vector Machine (SVM)

For binary classification, SVM seeks the hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ that maximizes the margin between classes. The primal optimization problem is:

$$\min_{\mathbf{w},b} \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{N} \xi_i$$

subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

where C is the regularization parameter and ξ_i are slack variables.

Random Forest (RF)

RF is an ensemble of T decision trees. The final prediction for classification is obtained via majority voting:

$$\hat{y} = \text{mode}(\{h_t(\mathbf{x})\}_{t=1}^{T})$$

where $h_t(\mathbf{x})$ is the prediction of the t-th tree.

K-Nearest Neighbors (KNN)

For a query point $\mathbf{x}_q$, KNN assigns the class label based on the majority class among its K nearest neighbors in the training set:

$$\hat{y} = \arg\max_c \sum_{i \in \mathcal{N}_K(\mathbf{x}_q)} \mathbb{I}(y_i = c)$$

where $\mathcal{N}_K(\mathbf{x}_q)$ is the set of K nearest neighbors.

Naïve Bayes (NB)

Assuming feature independence given the class, NB computes the posterior probability as:

$$P(y \mid \mathbf{x}) \propto P(y) \prod_{j=1}^{d} P(x_j \mid y)$$

The predicted class is:

$$\hat{y} = \arg\max_y P(y) \prod_{j=1}^{d} P(x_j \mid y)$$

5.2 Evaluation Metrics

Let:

TP = True Positives,

FP = False Positives,

TN = True Negatives,

FN = False Negatives.

Accuracy:

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

Precision:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall (Sensitivity):

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score:

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

False Positive Rate (FPR):

$$\text{FPR} = \frac{FP}{FP + TN}$$

Confusion Matrix

For a binary classifier, the confusion matrix C is:

$$C = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

For multi-class intrusion detection (e.g., Normal, DoS, Probe, R2L, U2R), the matrix generalizes to $k \times k$, where C_{ij} indicates instances of true class i predicted as class j.

5.2.1 Dimensionality Reduction – Principal Component Analysis (PCA)

Given a centered data matrix X ∈ $\mathbb{R}^{nxd}$, PCA seeks the projection matrix W ∈ $\mathbb{R}^{dxr}$ that maximizes variance:

$$W^* = \arg\max_W \text{Tr}(W^T X^T X W)$$

Subject to $W^T W = I$.

The reduced representation is:

$$Z = XW$$

5.2.2 Class Balancing – Synthetic Minority Over-sampling Technique (SMOTE)

For a minority class sample $\mathbf{x}_i$, SMOTE generates synthetic samples along the line segment between $\mathbf{x}_i$ and a randomly chosen neighbor $\mathbf{x}_j$:

$$\mathbf{x}_{\text{new}} = \mathbf{x}_i + \lambda(\mathbf{x}_j - \mathbf{x}_i)$$

where $\lambda \sim \mathcal{U}(0,1)$.

VI. EVALUATION PROTOCOL

6.1 Evaluation Metrics

Model evaluation is designed using standard metrics, e.g., accuracy, precision, recall, F1-score, false positive rate, and confusion matrices. For deep learning models, additional considerations for such as; training stability and convergence behavior are analyzed. [9,10]

6.2 Training Strategy and Regularization

Deep learning models are trained using backpropagation and adaptive optimization algorithms. Regularization methods like dropout, batch normalization, and early stopping are used to stop overfitting and help the model work better on new data.

6.3 Reproducibility Considerations

The framework sets up consistent preprocessing pipelines, fixed random seeds, and clear reporting of hyperparameters and architectural configurations to make sure that results can be repeated.

6.4 Analytical Discussion

From an analytical perspective, ensemble-based techniques like Random Forest are expected to offer robustness against noisy and high-dimensional data due to their aggregation mechanism. On the other hand, linear models might not be able to figure out how to attack in complicated ways. Instance-based methods can be useful in some situations, but they don't work well when a lot of people are using them.
Analytical studies have demonstrated that feature selection reduces variance and computational requirements. On the other hand, balancing data makes it easier to find attacks that are rare. It's still hard to deal with attacks like R2L and U2R because they look a lot like normal traffic. This shows how important it is to look at things in context and do advanced feature engineering.

VII. ALGORITHMIC DESCRIPTION AND MODEL ARCHITECTURE

7.1 Pseudocode of the Proposed IDS Framework

Based on Deep Learning, an algorithm Machine Learning showed and set pseudocode that showed a modular, scalable, and intelligent IDS framework that took part in data preprocessing, feature optimization, ensemble ML/DL modeling, and real-

time intrusion detection. Because of its structured design, this framework is beneficial for use in IoT and next-generation network security settings. [26,27]

```
Begin
    Acquire raw network traffic data D

    // Data Preprocessing
    Handle missing or inconsistent values in D
    Encode categorical features into numerical format
    Normalize numerical features

    // Feature Engineering
    Apply feature selection using mutual information analysis
    Perform dimensionality reduction using PCA

    // Class Imbalance Handling
    Apply SMOTE to the training data only

    // Dataset Partitioning
    Split the dataset into training set D_train and test set D_test

    // Model Training
    Train classical ML models:
        Random Forest, SVM, KNN, Logistic Regression, Naïve Bayes

    Train deep learning models:
        DNN, CNN, LSTM, Hybrid CNN–LSTM

    // Model Evaluation
    For each trained model do
        Predict intrusion labels for D_test
        Compute evaluation metrics:
            Accuracy, Precision, Recall, F1-score, FPR
        Generate confusion matrix and attack-wise performance
    End For

    // Model Selection
    Select optimal model based on:
        Detection performance
        False positive behavior
        Computational cost and deployment constraints

    // Deployment
    Deploy selected model for real-time or IoT-based IDS
    Generate alerts upon detection of malicious activity

End
```

Algorithm 1: Proposed ML/DL-Based IDS Framework

VIII. IMPLEMENTATION

The suggested framework sets and highlights the way of practicality by addressing and prioritizing models that balance their detection capability and computational efficiency. Classical machine learning models have advantages such as interpretability, maintenance simplicity, and lower resource requirements, setting and making them suitable for IoT and edge-based IDS applications.

8.1 Implementation Environment

The suggested framework for intrusion detection was implemented and tested in an offline experimental setting based on standard machine learning and deep learning libraries. The experiments were set and carried out on a workstation with a high specification e.g., a multi-core CPU, memory(RAM) AND SSD capacity to support deep learning. The implementation was developed to be modular and platform-independent to easily fit into different computing platforms or environments, both cloud and edge computing.

The framework for intrusion detection was developed based on commonly used open-source tools e.g., Python programming language to make sure that the implementation is transparent and replicable. The traditional machine learning models were developed based on standard machine learning libraries, while deep learning models were developed and constructed based on standard deep learning frameworks supporting automatic differentiation.

8.2 Data Handling and Preprocessing Implementation

The NSL-KDD dataset was initially used as the vital benchmark for implementation and validation. Raw network traffic logs were analyzed and structured in the form of feature vectors. Numerical encoding schemes were also used to ensure that learning algorithms function correctly on categorical features such as the type of protocol and service. Scaling numerical features ensured that a single feature was not dominated by the rest during the training process. Data cleansing techniques were also implemented to address missing or inconsistent values. A single preprocessing technique was used on all the models to ensure a fair and reproducible evaluation process. For the selection of the most significant features, mutual information analysis was conducted. Then, PCA was used to avoid redundancy and computation. For the solution and tackling of the class imbalance problem, the Synthetic Minority Over-Sampling Technique (SMOTE) was integrated into the training process. Oversampling was done only on the training set to avoid information leak and maintain the integrity of the results.

8.3 Training and Evaluation Implementation

The dataset was divided into training and testing sets. The training set was used for model training. The test data, which was not previously seen, was used to test the model.

Evaluation metrics such as accuracy, precision, recall, F1 score, false positive rate, and confusion matrices were used to evaluate the performance of the detection. An attack-wise evaluation was performed to observe the performance of the model for various attacks. All the models were tested using the same metrics and data to compare the models

IX. CONCLUSION

A detailed method and analysis for intrusion detection systems has been shown to use machine learning and deep learning techniques. Unlike many current IDS studies that mainly report isolated performance metrics, the suggested framework presents a clear, reproducible, and deployment-aware approach to IDS design and evaluation. The framework combines systematic data preprocessing, feature engineering, handling class imbalance, model training, and unified evaluation protocols into one analytical pipeline. This framework allows for a fair and practical comparison of detection behavior, computational complexity, and real-world use by employing both traditional machine learning algorithms and modern deep learning architectures. The analytical results demonstrate that deep learning models, particularly CNN and hybrid CNN-LSTM architectures, excel at detecting complex attack patterns in space and time. However, traditional machine learning models remain very competitive due to their straightforward nature, lower computational demands, and effectiveness in real-

time and resource-limited settings like IoT and edge networks. Attack-specific evaluation showed that minority attack types, such as R2L and U2R, still pose challenges for intrusion detection systems. These results underline the importance of feature optimization and data balancing techniques. They also highlight the need for evaluation strategies that consider more than just overall accuracy. Additionally, examining the false positive rate and deployment feasibility revealed that high detection performance alone is insufficient for operational IDS deployment without careful attention to system scalability and alert reliability. Overall, the suggested framework offers a practical and flexible foundation for applied IDS research, providing clarity and relevance to deployment that are often missing in existing studies. In future work, the framework will be expanded to include real-world traffic scenarios and contemporary datasets, focusing on online and adaptive learning, federated intrusion detection, and lightweight IDS solutions for large-scale IoT environments.

ACKNOWLEDGMENTS

The authors extend the appreciation to the Deanship of Postgraduate Studies and Scientific Research at Majmaah University for funding this research work through the project number (…..), Also we would like to thank your Journal staff and reviewers for their expertise and support throughout all aspects of my research paper and their valuable guidance.

REFERENCES

1. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016.
2. R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” in Proc. IEEE Symp. Security and Privacy, Oakland, CA, USA, 2010, pp. 305–316.
3. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in Proc. IEEE Symp. Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 2009, pp. 1–6.
4. Yin, Y. Zhu, J. Fei, and X. He, “A deep learning approach for intrusion detection using recurrent neural networks,” IEEE Access, vol. 5, pp. 21954–21961, 2017.
A. Javaid, Q. Niyaz, W. Sun, and M. Alam, “A deep learning approach for network intrusion detection system,” EAI Endorsed Transactions on Security and Safety, vol. 3, no. 9, pp. 1–8, 2016.
5. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in Proc. Int. Conf. Information Systems Security and Privacy (ICISSP), Funchal, Portugal, 2018, pp. 108–116.
6. G. Kim, S. Lee, and S. Kim, “A novel hybrid intrusion detection method integrating anomaly detection with misuse detection,” Expert Systems with Applications, vol. 41, no. 4, pp. 1690–1700, 2014.
7. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.
8. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.
9. T. Kim, J. Park, and S. Cho, “Lightweight intrusion detection system based on machine learning for IoT environments,” IEEE Internet of Things Journal, vol. 7, no. 5, pp. 4689–4699, 2020.
10. Z. Li, A. L. G. Rios, and M. Z. A. Bhuiyan, “Network intrusion detection using deep learning: A systematic review,” IEEE Access, vol. 9, pp. 36850–36872, 2021.
11. Zhang, X., & Li, Y. (2021). "A Review of Machine Learning Techniques in Intrusion Detection Systems." Computers & Security, 56(2), 45-57.
12. Mohammad, A., & Omar, B. (2019). "Intrusion Detection Using Machine Learning." International Journal of Cyber Security, 15(3), 30-40.
13. Buczak, A. L., & 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.
14. Scarfone, K., & Mell, P. (2007). Guide to Intrusion Detection and Prevention Systems (IDPS). NIST Special Publication 800-94. National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf
15. Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security, 28(1–2), 18–28.
16. Kim, G., & Spafford, E. (1994). The design and implementation of Tripwire: A file system integrity checker. Proceedings of the 2nd ACM Conference on Computer and Communications Security, 18–29.
17. Laskov, P., Schäfer, C., Kotenko, I., & Rieck, K. (2005). Intrusion detection in unlabeled data with quarter-sphere support vector machines. Detection of Intrusions and Malware, and Vulnerability Assessment, 71–82.
18. Kim, Y., Kim, J., & Han, Y. (2018). Deep Learning for Network Intrusion Detection: A Survey. International Journal of Computer Science and Network Security, 18(7), 45-55.
19. Lee, W., & Stolfo, S. J. (2000). Data Mining Approaches for Intrusion Detection. Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS).
20. Mukkamala, S., Sung, A. H., & Abraham, A. (2002). Intrusion Detection Using Ensemble Learning. Proceedings of the 3rd International Symposium on Neural Networks (ISNN).
21. Smith, L., Gupta, V., & Cook, D. (2015). An Analysis of Machine Learning Techniques for Intrusion Detection. Proceedings of the 9th International Conference on Data Mining and Knowledge Discovery.
22. Xu, Z., Wang, X., & Xu, Y. (2016). Real-Time Network Intrusion Detection Using Apache Storm. International Journal of Computer Applications, 139(8), 12-18.
23. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & Vanderplas, J. (2011). "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research, 12, 2825–2830.
24. KDD Cup 1999 Dataset, Available at: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.htmlReferences
25. Aldriwish K.,” Empowering Learning through Intelligent Data- Driven Systems", in Engineering, Technology & Applied Science Research,8 Feb 2024
26. Aldriwish K, “Design of an Adaptive Near Field Communication Technology using Finite State Machine within Web Services” The International Journal of Innovative Technology and Exploring Engineering (IJITEE), India, 2021,10, 107-112
27. M. Bishop, Pattern Recognition and Machine Learning. New York, NY, USA: Springer, 2006.
28. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.