Adaptive Intrusion Detection Using Hybrid Deep Learning Models and Feature Selection

1st Dr. K. Suresh Babu
Dept. of CSE,
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
Email: sureshkunda546@gmail.com

2nd G. Kavya
Dept. of CSE,
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
Email: kavyasujatha49@gmail.com

3rd T. Durga Bhavani
Dept. of CSE,
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
Email: pandu14652@gmail.com

4th Sk. Y. Khajabi
Dept. of CSE,
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
Email: Khajabi2005@gmail.com

5th A. Deepthi Priya
Dept. of CSE,
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
Email: deepuandraju09@gmail.com

6th R. P. Ram Kumar
Dept. of CSE-DS,
Hyderabad, Telangana, India
Email: ramkumar1695@gmail.com

***Abstract*—With the rising complexity and scale of cyber attacks, the need for adaptive and smart Intrusion Detection Systems (IDS) is now a necessity. The present study presents a deep learning-powered IDS system that integrates various neural network architectures along with advanced preprocessing techniques to enhance detection rates. The CICIDS2017 benchmark dataset was preprocessed with data cleaning, Min-Max normalization, and class balancing using SMOTEENN to remove data imbalance. Feature selection was performed using the Boruta algorithm to keep only the significant features. Four deep network models—Autoencoder, GRU, AlexNet, and MiniVGGNet—were employed and experimented with, where the hybrid combination method of feature selection and deep networks demonstrated remarkable improvements in performance. Among them, the most accurate classification results were obtained by Autoencoder with 99.67% accuracy, 98.00% precision, 98.08% recall, and 98.09% F1-score. Training-validation graphs and confusion matrices also asserted the effectiveness of the given multiclass intrusion detection system. The results reflect the importance of the utilization of robust preprocessing, feature extraction, and deep learning strategies for the development of efficient IDS systems.**

***Index Terms*—Intrusion Detection System (IDS), Deep Learning, CICIDS2017 Dataset, SMOTEENN, Boruta Feature Selection, Convolutional Neural Networks (CNN)**

I. Introduction

The internet facilitates global connectivity for e-commerce, data exchange, real-time communication, and other life-critical applications. With digital networks growing in size and complexity, it has become ever more important to secure them. Increasing numbers of cyber interactions, both in volume and sophistication, have produced more damaging and prevalent cyber attacks, typically taking advantage of previously unknown vulnerabilities in network systems [1].

Background and Motivation

Network traffic is the movement of data packets that pass between equipment using the internet. Malicious behavior like data theft, unauthorized access, and denial-of-service (DoS) attacks tend to follow legitimate traffic patterns, hence why traffic analysis is critical [2]. DoS, DDoS, brute-force attacks, botnets, injection exploits, and network scanning are forms of intrusion attempts that significantly threaten data security, system function, and availability [3]. Since threats are dynamic and need to be identified as soon as possible, intrusion detection is still a challenging task.

Research Gap

Conventional intrusion detection systems (IDS) employ signature-matching or rule-based approaches, which are effective to detect known attacks but ineffective to detect unknown or novel threats [4]. Machine learning and deep learning algorithms such as SVM, k-Nearest Neighbors, Random Forests, CNNs, LSTMs, and Autoencoders have been employed in the latest studies to find hidden patterns in vast amounts of data [6]. These approaches, however, continue to be plagued by class imbalance and redundant attributes, leading to degradation of detection precision [**?**].

Objectives

The main goals of this research are:

- Create preprocessing techniques to handle class imbalance so that models can learn efficiently from every class.
- Use the Boruta algorithm for selecting meaningful features so that noise is avoided and model performance is enhanced [12].

- Implement and compare four deep learning models: Autoencoder, GRU, AlexNet, and MiniVGGNet.

Contributions

The key contributions of this research are:

- Use of SMOTEENN [7] for balancing classes and Boruta [?] for selecting features, in place of earlier methods like DSSTE and Random Forest [8].
- Use of Autoencoders to identify anomalous activity that could reveal new unknown attacks and GRU networks to find temporal patterns in traffic data [?].
- Analysis of CNN-based models like MiniVGGNet and AlexNet [11] for spatial pattern extraction, keeping in mind their weakness when it comes to time-series data.
- Suggestion of a hybrid IDS framework that provides enhanced accuracy, interpretability, and scalability across various network environments.

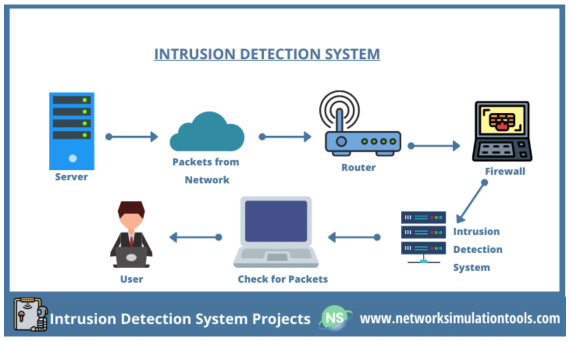
INTRUSION DETECTION SYSTEM
Server
Packets from Network
Router
Firewall
Intrusion Detection System
Check for Packets
User
Intrusion Detection System Projects
NS
www.networksimulationtools.com

Figure 1: Conceptual architecture of a typical Intrusion Detection System (IDS), illustrating the data flow from packet capture to alert generation.

The rest of the paper is outlined as follows: Section II discusses the literature works. Section III outlines the methodology. Section IV gives the result analysis and discussion. Section V discusses the final thoughts and future works. Section VI provides the references.

II. Literature Works

Because cyberattacks are becoming more frequent and sophisticated, it is more important than ever to secure infrastructures connected to the internet in the current digital era. In recent years, academics have been increasingly examining machine learning (ML) and deep learning (DL) techniques to improve intrusion detection systems (IDS). These methods are able to recognize intricate and changing attack patterns by learning from vast amounts of network traffic [?], [14].

The Difficult Set Sampling Technique (DSSTE) was proposed by Li et al. [?] to address the problem of class imbalance, which frequently lowers the effectiveness of IDS. Our strategy makes use of SMOTE-ENN, a hybrid technique that improves the robustness and dependability of the detection process by balancing the dataset and eliminating noisy data points.

Similar to this, Khan et al. [16] used AdaBoost and Decision Trees to create a multi-class IDS that addressed class imbalance in the CICIDS2017 dataset by utilizing SMOTE-Tomek Links. Their system achieved over 96% accuracy and strong F1 scores but lacked support for handling sequential or time-dependent traffic data.

FatimaEzzahra Laghrissi et al. [17] showcased the use of LSTM networks for intrusion detection, improving performance through PCA and Mutual Information-based feature selection on the KDD99 dataset. While LSTM networks are proficient in modeling long-term sequences, reliance on the outdated KDD99 dataset limits real-world applicability. In this work, we instead utilize the CICIDS2017 dataset, which better represents contemporary network environments and attack behaviors.

A hybrid IDS proposed by Xuntao Hu et al. [?] combined ResNet and Bi-LSTM architectures with attention mechanisms, delivering near-perfect binary classification results on both UNSW-NB15 and CICIDS2017 datasets. Despite its high accuracy, the model's computational overhead and complexity reduce its feasibility for real-time use. Our approach favors more efficient architectures, such as MiniVGGNet, and enhances interpretability through Boruta-based feature selection.

Lakshit Sama [19] compared some of the popular deep learning models such as LightGBM, XGBoost, LSTM, and Decision Trees on different datasets and concluded that DL models [20] tend to perform better but are susceptible to high false positive rates.

III. Methodology

A. Experimental Setup

Experiments were performed on the CICIDS2017 dataset. The preprocessing including data cleaning, Min–Max normalization, SMOTEENN-based class balancing, and Boruta-based feature selection were done in Python 3.10. Deep learning models (Autoencoder, GRU, AlexNet, and MiniVGGNet) were coded using TensorFlow 2.12 and Keras libraries, whereas other preprocessing and visualization were carried out using Scikit-learn, Pandas, and Matplotlib.

The training and testing were conducted on the Google Colab environment with the following specification:

- **GPU:** Tesla T4 (12 GB)
- **RAM:** 12 GB
- **Storage:** 100 GB allocation
- **OS:** Linux (Google Colab environment)

Accuracy and loss curves, as well as confusion matrices, were created to confirm and contrast model performance.

B. Dataset

CICIDS2017 dataset, created by the Canadian Institute for Cybersecurity (CIC) [21], is used in this study. It is widely regarded in IDS research for its realistic representation of normal network activity alongside various modern cyberattacks.

The collection includes network traffic associated with different kinds of assaults, such as:

- Attacks known as denial of service (DoS)
- Attacks using Distributed Denial of Service (DDoS)
- Infiltration attempts
- Web-based vulnerabilities including Cross-Site Scripting (XSS) and SQL injection
- Botnet-related activities
- Exploitation of the Heartbleed vulnerability
- Port scanning and reconnaissance operations

Each data instance comprises over **80 extracted features**, generated using CICFlowMeter. These features represent key aspects of network flows, such as duration, protocol, byte and packet statistics, and header details. This diverse information supports thorough training and evaluation of detection models.

The CICIDS2017 dataset's inclusion of realistic, tagged traffic that represents both benign and malevolent activity is one of its main advantages. However, a notable challenge lies in its **imbalanced class distribution**, where benign traffic significantly outweighs attack samples. To reduce bias during model training, we use a resampling method called **SMOTEENN**, which balances the data by generating more samples for underrepresented classes and filtering out noisy or ambiguous records to improve overall classification results.

C. Data Preprocessing

The CICIDS2017 dataset was formed by combining several CSV files into one. Unnecessary columns, such as flow IDs and timestamps, were removed, and any rows containing missing or invalid values were filtered out. Target labels were encoded numerically, and Min-Max scaling was applied to normalize numerical features.

To tackle class imbalance, we used **SMOTEENN**, a hybrid approach combining SMOTE [8] and Edited Nearest Neighbors (ENN). SMOTE creates synthetic samples for minority classes, while ENN removes noisy data near decision boundaries, improving model clarity and reducing overfitting.

To select important features, we applied the **Boruta** algorithm [**?**], which works by comparing actual features with randomly shuffled ones to find those that truly matter. This step improved model performance and reduced computational cost by retaining only the most important features.

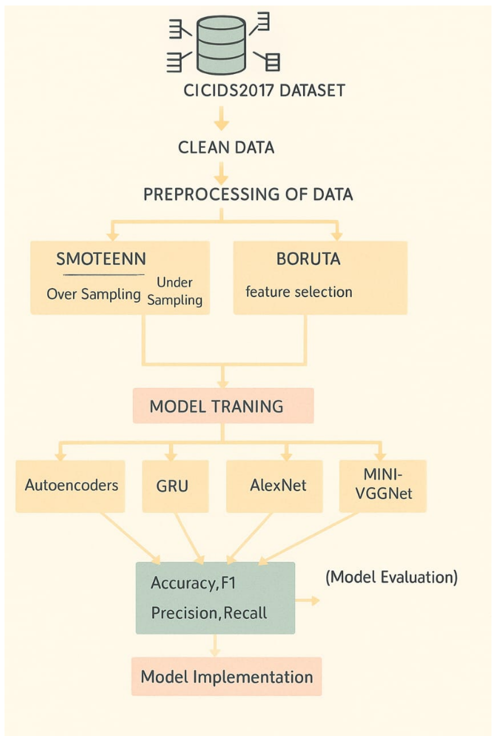

Figure 2: Model architecture of the proposed hybrid intrusion detection system.

D. Deep Learning Models

The deep learning models utilized in this study to identify network intrusions are described in this section. Each model was selected based on its ability to capture spatial, temporal, or anomalous characteristics from high-dimensional traffic data. The models explored include Autoencoder, Gated Recurrent Unit (GRU), AlexNet, and MiniVGGNet. [7]

1) Autoencoder: There is a type of unsupervised neural networks known as autoencoders, which attempts to learn concise, abstract representations of the input and reconstruct the input from the compressed form. They are expert anomaly detectors because they are trained only on normal traffic and generate high reconstruction errors when presented with novel or malicious patterns.

The encoder $f(x)$, which projects input data into a lower-dimensional space, and the decoder $g(z)$, which recovers the input from this compressed representation, are the two main building blocks of an autoencoder. Equation (1) is the encoder function, and Equation (2) is the equivalent decoder reconstruction:

$$z = f(x) = \sigma(W \cdot x + b) \tag{1}$$

$$\tilde{x} = g(z) = \sigma(W' \cdot z + b') \tag{2}$$

The network aims to reduce the reconstruction loss as given in Equation (3), the loss function optimized by the Autoencoder:

$$\mathcal{L}_{AE} = \|x - \tilde{x}\|_2^2 \tag{3}$$

Autoencoders are reported to be especially well-suited for detecting zero-day attacks, when anomalous behavior consistently varies from learned normal patterns [**?**].

2) Gated Recurrent Unit (GRU): A type of recurrent neural network (RNN) referred to as a GRU is constructed to efficiently mimic temporal relations and sequences. They have architectures comprising gating mechanisms for storing context over time steps, hence are well suited for sequential network traffic analysis [**?**].

The fundamental GRU computations are shown in Equations (4)–(7). In particular, Equation (4) specifies the update gate, Equation (5) specifies the reset gate, Equation (6) provides the candidate hidden state, and Equation (7) expresses the final hidden state update:

$$z_t = \sigma(W^{(z)} x_t + U^{(z)} h_{t-1}) \tag{4}$$

$$r_t = \sigma(W^{(r)} x_t + U^{(r)} h_{t-1}) \tag{5}$$

$$\hat{h}t = \tanh(W^{(h)} x_t + U^{(h)} (r_t \circ ht - 1)) \tag{6}$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \hat{h}_t \tag{7}$$

GRUs can learn time-series dependencies effectively, which is crucial for intrusion detection with temporal behavior.

3) AlexNet: AlexNet [11] is a deep convolutional neural network of stacked fully connected, pooling, and convolutional layers that was initially applied for image classification. As computationally costly, it can learn fine spatial features [**?**], [**?**].

The fundamental operations in convolution layers are illustrated in Equations (8) and (9). Equation (8) is the ReLU activation function applied in convolution layers, and Equation (9) gives the Max-Pooling operation:

$$y = \text{ReLU}(W * x + b) \tag{8}$$

$$\text{MaxPool}(x) = \max_{i \in \text{window}} x_i$$

As presented by [11], these operations allow AlexNet to examine traffic flow matrices and recognize spatial feature distributions of various attack types.

4) MiniVGGNet: MiniVGGNet is a slim version of the original VGGNet [**?**] that has been optimized to preserve performance while decreasing computational expense. MiniVGGNet uses a number of small 3×3 convolutional filters and fewer layers compared to its ancestor.

The overall structure of a convolutional layer of MiniVGGNet is given in Equation (10), showing the convolution + ReLU operation performed iteratively over layers:

$$y_l = textReLU(W_l * y_{l-1} + b_l) \tag{10}$$

Equation (10) shows how MiniVGGNet derives hierarchical spatial features without being too heavy for large datasets like CICIDS2017. This renders it real-time friendly because of its efficiency and depth balance.

5) Model Integration: As illustrated in Fig. 2, Our suggested solution integrates the advantages of four different deep learning models:

- The **Autoencoder** serves as an unsupervised detector by learning to reconstruct normal traffic patterns and flagging anomalies based on reconstruction loss. Its compact architecture supports high-throughput, real-time analysis.
- The **GRU** model captures temporal dynamics in traffic sequences, making it well-suited for detecting attacks with time-based patterns, such as brute-force or slow-rate intrusions.
- **AlexNet** and **MiniVGGNet**, In order to extract spatial information from traffic matrices, both convolutional neural networks (CNNs) are used.

IV. Result Analysis and Discussion

A. Training Loss Comparison

Figure 3 shows the training loss trends for each model across several epochs. The Autoencoder quickly reaches a low loss, reflecting its ability to learn normal traffic patterns—useful for spotting anomalies. GRU reduces loss more gradually, benefiting from its memory of sequence data, which helps in identifying long-running or time-based attacks. Both AlexNet and MiniVGGNet show stable learning curves, with MiniVGGNet improving more rapidly, suggesting quicker and more efficient feature learning at a lower computational cost.

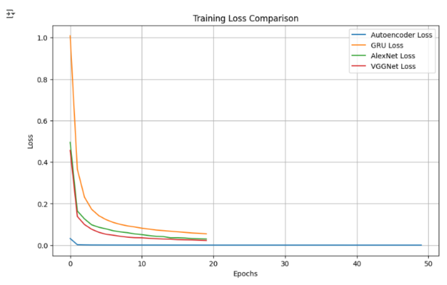

Figure 3: Training Loss Comparison of Autoencoder, GRU, AlexNet, and VGGNet models

B. Validation Accuracy

Figure 4 presents the validation accuracy of all models. Autoencoders achieved the highest score, reaching

99.67%, with MiniVGGNet performing closely behind. Both models benefited from strong feature extraction capabilities that effectively captured complex patterns in the network traffic. Their high accuracy and minimal signs of overfitting suggest they are well-suited for practical intrusion detection applications.

The GRU model also delivered solid results, slightly trailing the CNN models. Its strength lies in processing sequential data, making it particularly effective for detecting time-distributed attacks such as brute-force or SSH intrusions. On the other hand, the Autoencoder, which is inherently designed for unsupervised anomaly detection, showed lower validation accuracy. This is expected, as its architecture focuses on identifying deviations from normal behavior rather than handling multi-class classification tasks.

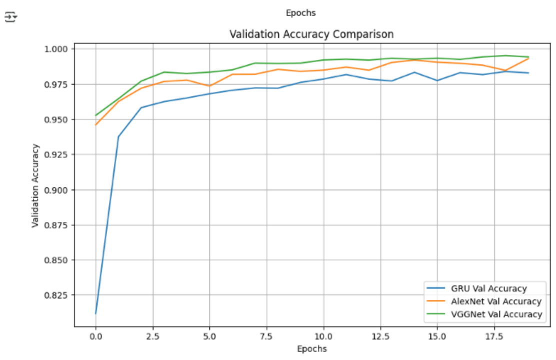

Figure 4: Validation Accuracy Comparison across Models

C. Confusion Matrix Analysis

To gain deeper insight into the classification performance of each model, confusion matrices were generated and are presented in Figures **??** and **??**. The distribution of correctly and incorrectly classified instances across all attack categories is shown by these matrices. We can determine which intrusion types each model manages well and where misclassifications occur more frequently by examining these results. This analysis is essential for assessing the models' applicability and dependability in actual intrusion detection situations.

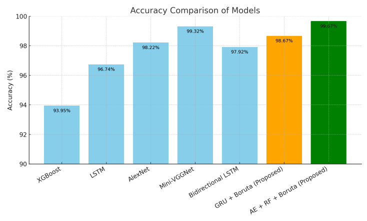

Figure 5: Comparing plots for all models

Overall, MiniVGGNet and AlexNet demonstrated their efficacy in recognizing a broad variety of attack types by achieving high classification accuracy. However, because of their intricate architectures, this performance comes at a higher computational cost. GRU, on the other hand, excels at identifying time-dependent intrusion patterns and achieves a realistic balance between detection capability and resource efficiency. For anomaly-based detection, the Autoencoder combination proved ideal, particularly for detecting zero-day or previously unknown threats.

Model	**Acc**	**F1**	**Prec**	**Rec**
XGBoost	93.95%	0.94	0.94	0.94
LSTM	96.74%	0.97	0.97	0.97
AlexNet	98.22%	0.98	0.98	0.91
Mini-VGGNet	99.32%	0.98	0.98	0.97
Bidirectional LSTM	97.92%	0.98	0.98	0.97
Proposed Model (GRU)	**98.67%**	**0.98**	**0.98**	**0.98**
Proposed Model (AE)	**99.67%**	**0.98**	**0.98**	**0.98**

TABLE I: Performance comparison of models with the proposed model.

Table I provides a comparative evaluation of several models—XGBoost, LSTM, AlexNet, MiniVGGNet, and Bidirectional LSTM—using important criteria including recall, accuracy, precision, and F1 score. With balanced Precision, Recall, and F1 Score values of 0.98 and an amazing accuracy of 98.67%, the suggested model performs better than the others. These results underscore the model's strong generalization capability and its robustness in classifying various network traffic types, outperforming both traditional ensemble methods and deep learning architectures included in the comparison.

D. Discussion

GRU (Proposed): The Gated Recurrent Unit (GRU) model leverages its ability to capture temporal dependencies in network traffic, making it highly effective for detecting time-dependent intrusions such as brute-force login attempts, slow-rate attacks, and persistent scanning activities. Its simplified architecture compared to LSTM ensures faster training while maintaining strong sequential learning capabilities, which is crucial for real-time intrusion detection.

Autoencoders (Proposed): The Autoencoders model excels at learning compact representations of normal network behavior, enabling it to effectively detect anomalies and potential zero-day attacks. By reconstructing input traffic and analyzing reconstruction errors, it can distinguish between normal and malicious patterns with high precision. This makes it particularly valuable for identifying novel threats that do not closely resemble known attack signatures.

Overall, GRU is well-suited for analyzing sequential traffic data, while Autoencoders are effective for uncovering hidden anomalies in large-scale network datasets. Their complementary strengths highlight the potential of combining temporal modeling with unsupervised feature learning for robust and adaptive intrusion detection systems.

V. Conclusion and Future Works

Primary Conclusions and Performance

An adaptive IDS was implemented based on deep learning models—Autoencoder and GRU. The method illustrated the power of using a combination of deep learning and ensemble methods, along with robust preprocessing techniques like SMOTEENN and Boruta, for network intrusion detection employing the CICIDS2017 dataset. Autoencoder and MiniVGGNet models were able to attain very high accuracy because of their ability to perform deep feature extraction, with MiniVGGNet demonstrating better classification performance. GRU was able to capture the sequential patterns essential in detecting ongoing threats, while the hybrid Autoencoders were extremely effective for anomaly-based detection. The system was able to attain high detection accuracy without a high false positive rate, which makes it deployable in real-world situations.

Practical Limitations

While the models executed from top notch on the CICIDS2017 dataset, performance is indirectly tied to training overhead and data quality and variability. Moreover, computational overhead for training deep learning models continues to be large, which would limit deployment to low-resource settings.

Future Work

Future work will concentrate on incorporating attention mechanisms, online data testing, and edge computing and IoT optimization to better enhance detection capabilities and operational efficiency.

References

[1] S. Aljawarneh, M. Aldwairi, and M. Yassein, “Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model,” J. Comput. Sci., vol. 25, pp. 152–160, 2018.

[2] Y. Lin, F. Ye, W. Xu, and J. Hu, “A survey of network traffic analysis and prediction techniques,” Int. J. Comput. Appl., vol. 87, no. 11, 2013.

[3] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, “A deep learning approach to network intrusion detection,” IEEE Trans. Emerg. Topics Comput. Intell., vol. 2, no. 1, pp. 41–50, 2018.

[4] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” Comput. Secur., vol. 28, no. 1–2, pp. 18–28, 2009.

[5] N. Moustafa and J. Slay, ”UNSW-NB15: A comprehensive data set for network intrusion detection systems,” in Proc. MilCIS, 2015, pp. 1–6.

[6] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, ”A deep learning approach for network intrusion detection system,” in Proc. 9th EAI Int. Conf. Bio-inspired Inf. Commun. Technol., 2016, pp. 21–26.

[7] K. S. Babu and Y. N. Rao, “MCGAN: Modified conditional generative adversarial network for class imbalance problems in network intrusion detection system,” Appl. Sci., vol. 13, no. 4, p. 2576, 2023.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.

[9] M. B. Kursa and W. R. Rudnicki, “Feature selection with the Boruta package,” J. Stat. Softw., vol. 36, no. 11, pp. 1–13, 2010.

[10] C. Zhou and R. C. Paffenroth, “Anomaly detection with robust deep autoencoders,” in Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2017, pp. 665–674.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Adv. Neural Inf. Process. Syst., 2012, pp. 1097–1105.

[12] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “An ensemble intrusion detection model based on feature selection and classification algorithms,” J. Big Data, vol. 6, no. 1, pp. 1–18, 2019.

[13] Y. N. Rao and K. S. Babu, ”An imbalanced generative adversarial network-based approach for network intrusion detection in an imbalanced dataset,” Sensors, vol. 23, no. 1, p. 550, 2023.

[14] K. S. Babu, V. Srinivas, Y. Chandana, G. Satish, R. D. Naik, and D. V. Reddy, ”Streamlined network intrusion detection with feature selection and optimization for higher accuracy and efficiency,” in Advances in Elect. Comput. Technol. Boca Raton, FL, USA: CRC Press, 2025, pp. 114–124.

[15] J. Li, R. Wang, and B. Zhang, ”Difficult set sampling for intrusion detection,” IEEE Trans. Depend. Secure Comput., 2021.

[16] A. Khan and N. Ahmed, “Multi-class intrusion detection in network traffic using ensemble learning,” J. Netw. Comput. Appl., vol. 203, 2022.

[17] F. E. Laghrissi, A. Lbath, and T. Ahmed, “Network intrusion detection system using LSTM,” Procedia Comput. Sci., vol. 151, pp. 511–516, 2019.

[18] X. Hu, Q. Liu, Y. Zhang, and K. Han, “A hybrid intrusion detection method based on ResNet and BiLSTM,” IEEE Access, vol. 8, pp. 104119–104130, 2020.

[19] L. Sama, A. Sharma, and A. Arora, “Comparative study of machine learning and deep learning models for intrusion detection in network traffic,” in Proc. Int. Conf. Mach. Learn., Big Data, Cloud Parallel Comput. (COMITCon), 2022, pp. 1–6.

[20] C. Ravindran and K. Sarveshwaran, “A review on deep learning models for intrusion detection systems in cyber security,” Comput. Sci. Rev., vol. 43, p. 100432, 2022.

[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP), 2018, pp. 108–116.