KEC Conference 2025

ISSN: 3021-940X (Print) DOI: https://doi.org/10.3126/injet.v2i2.78621

Network Intrusion Detection System using Machine Learning

Shyama Mainali[1, *], Pranav Neupane[2], Utsarga Regmi[3], Samrat Chaudhari[4], Suman Shrestha[5]

[1]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, mainalishyama@gmail.com*

[2]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, neupanepranav40@gmail.com*

[3]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, regmiutsarga7@gmail.com*

[4]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, samratchy40770@kec.edu.np*

[5]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, sumanshrestha@kec.edu.np*

Abstract

In modern network environments, security threats such as Denial of Service (DoS) and Distributed Denial of Service (DDoS attacks pose significant risks to network availability and performance. These attacks aim to overwhelm network resources, disrupt services, and make systems inaccessible to legitimate users. A Network Intrusion Detection System (NIDS) plays a crucial role in identifying and mitigating these threats by continuously monitoring network traffic and detecting malicious activities. This project presents a Network Intrusion Detection System (NIDS) utilizing the Fast k-Nearest Neighbors (Fast k-NN) algorithm for detecting Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks in a network environment. The CICIDS2017 dataset is used for model training, involving data preprocessing steps such as cleaning, trans formation, feature selection, extraction, and labeling. The dataset is split into training, validation, and testing sets, where the Fast k-NN algorithm is applied to train the NIDS model. The experimental setup includes a Mininet-based custom topology controlled by the POX SDN controller. DoS and DDoS attacks are simulated using hping3, and network traffic is captured using Wireshark. Features are extracted using NTL FlowLyzer and processed for intrusion detection. The trained model predicts attack patterns, logging detected intrusions for analysis. This approach provides an efficient and scalable intrusion detection system that enhances network security by leveraging machine learning for accurate attack detection.

Keywords: Network Intrusion Detection System (NIDS), Fast k-Nearest Neighbors (Fast k-NN), Denial of Service (DoS), Distributed Denial of Service (DDoS), CICIDS2017, Software-Defined Networking (SDN), POX Controller, Mininet, hping3, Wireshark, NTL FlowLyzer

1. Introduction

The increasing reliance on computers and networks has made cyber security a critical concern, as industries face a growing array of attacks daily. Intrusion detection technologies have become essential for safeguarding computer systems and networks, aiming to enhance detection rates while minimizing false alarms. An Intrusion Detection System (IDS) monitors network traffic for known threats and suspicious activities, generating alerts upon detecting security risks. Since their inception, IDS and Intrusion Prevention Systems (IPS) have evolved significantly. Early systems in the 1970s and 1980s focused on statistical anomaly detection, while the 1990s saw the rise of network-based and signature-based systems. Between 2000 and 2005, IDS gained traction due to emerging threats like SQL injections, leading organizations to favor IDS over IPS initially due to concerns about blocking legitimate traffic. However, from 2006 to 2010, IPS adoption grew as throughput increased and compliance requirements emerged. The period from 2011 to 2015 introduced next-generation IPS (NGIPS) with advanced features in response to high-profile breaches. Major incidents from 2016 to 2020 underscored the need for advanced solutions, resulting in the widespread adoption of next-generation firewalls (NGFW) incorporating IDS/IPS functionalities. Since 2021, the

**Corresponding Author*

integration of machine learning, AI, and Network Detection and Response (NDR) solutions has further enhanced threat detection capabilities, addressing modern challenges posed by IoT and sophisticated cyber threats. Algorithms such as KNN, decision trees, Random Forests, and SVM are commonly employed in this domain. Network Intrusion Detection Systems (NIDS) specifically monitor strategic points within an organization's network to detect malicious traffic and ensure comprehensive security.

- The paper proposed an optimized Fast KNN algorithm for network intrusion detection, enhancing accuracy and efficiency. Using a subset of the KDD CUP99 dataset, the study integrated data preprocessing, feature reduction, and classification. Their approach, incorporating mutual information-based feature selection and data discretization, significantly outperformed traditional KNN, reducing false detection rates and computational time (He, et al., 2023).
- The paper evaluated the Fast k-Nearest Neighbor Classifier (FkNN) for intrusion detection in cloud environments using the CICIDS2017 dataset. The study applied normalization, feature indexing, and classification, comparing KNN, PDS-KNN, and FkNN. Results showed that FkNN achieved high accuracy and reduced computational time by 88%, making it a viable solution for cloud-based NIDS (Krishna, et al., 2020).
- The paper analyzed various machine learning algorithms on the KDD-99 and NSL-KDD datasets, highlighting their effectiveness in detecting attacks. Random Forest achieved the highest accuracy (99.0% on KDD-99 and 99.7% on NSL-KDD) with a low false alarm rate, demonstrating its suitability for intrusion detection systems (Ravipati & Abualkibash, 2019).
- The paper investigated intrusion detection using Support Vector Machine (SVM) and Naïve Bayes on the NSL-KDD dataset. Their results indicated that SVM outperformed Naïve Bayes in detecting attacks. The study also discussed potential enhancements, including integrating an Intrusion Prevention System (IPS) and exploring deep learning models for improved detection (Desai, et al., n.d.).

2. Related Works

In a paper (P. He, G. Feng, H. Li, and L. Yang, 2023), the authors proposed a network intrusion detection method leveraging an optimized fast KNN algorithm to enhance efficiency and accuracy. The study utilized a subset of the KDD CUP99 dataset, selecting 80,056 samples, with 40,000 allocated for training and 40,056 for testing. The intrusion detection model encompasses several stages: data preprocessing, feature reduction, and classification. The Fast KNN algorithm addresses computational inefficiencies found in traditional KNN by incorporating techniques such as continuous data discretization, character data digitization, and a mutual information-based feature reduction algorithm. Evaluation metrics included classification accuracy, misdetection rate, and missed detection rate. Experimental results demonstrated that the Fast KNN algorithm significantly outperformed the traditional KNN method and existing technologies, achieving higher classification accuracy while reducing false detection rates and computational time. This advancement highlights the effectiveness of the Fast KNN algorithm in improving intrusion detection systems' performance (He, et al., 2023).

In a paper (K. V. Krishna, K. Swathi, and B. B. Rao., 2020), the authors investigate the performance of a Fast k-Nearest Neighbor Classifier (FkNN) for Network Intrusion Detection Systems (NIDS) within a cloud environment, utilizing the CICIDS2017 benchmark dataset. This dataset comprises 78 features and includes both benign traffic and the latest common attacks, closely resembling real-world scenarios. The study focuses on the KNN classifier and its variations, beginning with normalization using the min-max technique since all features are numeric. The proposed framework's methodology encompasses data preprocessing, variance-based feature indexing, and classification. After implementing KNN, PDS-KNN, and FkNN for various k values (3, 5, and 7) with 10-fold cross-validation, the authors evaluated metrics such as accuracy, precision, recall, and computational time. The results indicate that FkNN outperforms both traditional KNN and PDS-KNN in terms of classification accuracy and speed, achieving over an 88% reduction in computational time without sacrificing accuracy. The findings suggest that FkNN is a superior machine learning algorithm for

NIDS in cloud environments, effectively addressing modern attack types while minimizing economic losses for Cloud Service Providers (CSPs) (Krishna, et al., 2020).

In a paper (Ravipati & Abualkibash, 2019), the authors experiment with the performance of various machine learning algorithms using two key datasets: KDD-99 Cup and NSL-KDD. The KDD-99 dataset is diverse, containing categorical and integer-based attributes, while NSL-KDD was introduced as an improved version, addressing issues present in the original KDD dataset by selecting only essential records. The KDD dataset primarily includes four types of attacks: Denial of Service (DoS), Remote to Local (R2L), User to Root (U2R), and Probing. The study provides an overview of different machine learning approaches for Intrusion Detection Systems (IDS), categorizing them into Supervised, Semi-Supervised, and Unsupervised Learning. In the case of Supervised Learning, normal data is labeled as +1, while attack data is labeled as -1. The evaluation of various machine learning algorithms highlights the trade-offs between accuracy and false alarm rate (FAR). Among the tested methods, Random Forest achieves the best performance, obtaining 99.0% accuracy with a false alarm rate of 2.43% on the KDD-99 dataset. Similarly, on the NSL-KDD dataset, Random Forest reaches 99.7% accuracy with a false alarm rate of 3.2%, making it the most effective algorithm in both cases (Ravipati & Abualkibash, 2019).

In a paper (Desai, Sonawane, Mane, & Jaiswal, 2023), the authors explore cyber security advancements through the development and implementation of a Network Intrusion Detection System (NIDS**)** using machine learning techniques. With the increasing volume of data transmitted over the Internet, robust security measures are crucial for detecting and mitigating cyber threats. The study enhances Intrusion Detection Systems (IDS) by incorporating Support Vector Machine (SVM) and Naïve Bayes**, chose**n for their classification capabilities. Using the NSL-KDD dataset, the paper evaluates their effectiveness in identifying Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L), and Probe attacks. The findings indicate that SVM outperforms Naïve Bayes in terms of accuracy and misclassification rates, emphasizing the importance of model selection in intrusion detection. The study also explores the potential integration of an Intrusion Prevention System (IPS) to further enhance network security, proposing future research with models such as Convolutional Neural Networks (CNN) and Support Vector Networks (SVN)**.** Additionally, it provides a comprehensive overview of IDS technologies, including honeypots and Snort, and categorizes IDS into five distinct types, highlighting their specific functionalities and applications in modern cyber security (Desai, et al., n.d.).

3. Methodology

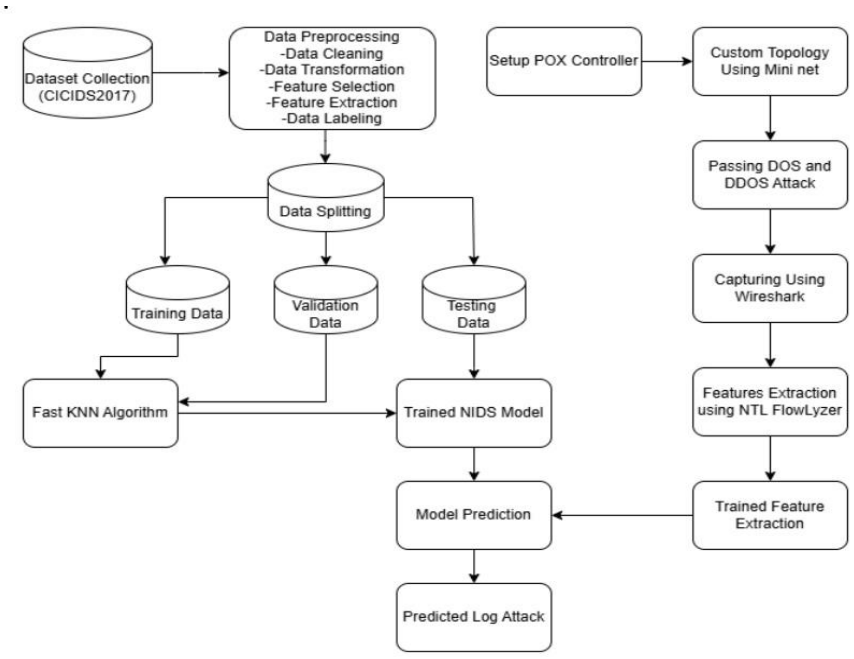

Figure 1. Working Mechanism of NIDS

3.1. Dataset Overview

Preprocessing begins with loading and combining the DOS, DDOS and benign file into single file. The further processes are:

3.1.1. Feature Selection and Correlation Analysis

After loading the dataset, the feature selection process is performed using a Random Forest Classifier. The dataset is split into features (X) and labels (y), followed by a train-test split to ensure that the model is trained and evaluated on different subsets of data. A Random Forest model is trained on the dataset, and feature importance scores are extracted. Features with an importance score greater than 0.01 are selected for further processing.

To understand the relationships between different features, a correlation matrix is computed after encoding the categorical label column using Label Encoding. A heat map is then generated to visualize the correlation between features. A function is implemented to remove highly correlated columns with a correlation threshold of 0.9 to eliminate redundant features. Additionally, feature names are renamed to more meaningful identifiers for clarity

3.1.2. Handling Class Imbalance

Since the dataset contains an imbalance between benign and attack traffic, it is necessary to balance the classes before training the model. Random Under-Sampling (RUS) is applied to reduce the size of the majority class, ensuring that no single class dominates the dataset. Next, SMOTE (Synthetic Minority Over-sampling Technique) is used to generate synthetic samples for the minority class, further balancing the dataset. This ensures that the classifier does not develop a bias toward the dominant class.

3.1.3. Feature Scaling

Once the dataset is balanced, feature normalization is performed using MinMaxScaler to scale all numerical features between 0 and 1. This helps in ensuring that all features contribute equally to the model's learning process, preventing features with larger ranges from dominating others. The trained MinMaxScaler is then saved using joblib, allowing consistent preprocessing for future data during model deployment.

3.2. Model Description

3.2.1. Fast KNN Model

Fast KNN is used to detect network attacks by analyzing traffic data from the CICIDS2017dataset. The process begins by converting each network packet or flow into a feature vector containing important network metrics. These features include Bwd Payload Bytes, Payload Bytes Variance, Bwd Packets Length Total, Fwd Packets Length Total, Fwd Packets IAT Std, Fwd Payload, Fwd Payload Bytes Max, Bwd Total Header Bytes, Fwd Total Header Bytes, Bwd Packets Rate, Packets Rate, Bwd Init Win Bytes, Fwd Packets IAT Mean, and Fwd Init Win Bytes. These feature vectors serve as inputs for the Fast KNN algorithm, allowing it to classify network activity as normal or malicious.

During the training phase, a subset of the CICIDS2017 dataset containing both normal and attack traffic is used. Fast KNN computes distances between feature vectors using metrics such as Euclidean distance. To improve efficiency, it employs optimizations like efficient indexing and approximate nearest neighbor search techniques. These enhancements make Fast KNN faster than traditional KNN while maintaining high accuracy.

In the prediction phase, when a new network packet or flow arrives, it is first converted into a feature vector. Fast KNN then compares this vector to stored training data and identifies the K nearest neighbors. Each of these neighbors votes on the classification of the new packet. The majority vote determines whether the packet is classified as normal or as one of the attack types, such as DDoS, DoS, or Probe. This approach ensures that Fast KNN can quickly and accurately detect network attacks in real-time environments.

$$d(v_{new}, v_i) = \sqrt{\left\{\sum_{j=1}^{m}\left(v_{new,j} - v_{i,j}\right)^2\right\}} \quad (1)$$

Where:
v_{new} is the feature vector of the new network flow.
v_i is the feature vector of the *i*-th flow in the training set.
$v_{new,j}$ and $v_{i,j}$ are the *j*-th feature values of v_{new} and v_i respectively.
m is the number of features used in the dataset.

3.2.2. Simulation using Mininet

Mininet is a network emulator that creates a virtual network on a single computer, enabling the simulation of complex network topologies. It runs virtual hosts, switches, routers, and links using a single Linux kernel. By leveraging lightweight virtualization, Mininet allows a single system to behave like an entire network, running the same software as real networking devices. Virtual hosts in Mininet function like physical machines, allowing users to execute programs, send data through virtual network connections with specific speeds and delays, and interact with virtual switches and routers that process traffic as real network devices would. While the performance of applications such as client and server closely resembles real systems, there may be slight variations in speed. Since Mininet's virtual network components replicate real ones through software, it enables the design and testing of networks that closely mimic physical infrastructures.

The POX Controller is an open-source framework for Software-Defined Networking (SDN) built using Python. As the central intelligence of an SDN network, POX dynamically manages and directs data traffic. It provides network administrators with a programmable interface to configure network devices such as switches and routers, define forwarding rules, and enforce security policies. By utilizing Python's event-driven architecture, POX offers flexibility and scalability, allowing the development of custom network applications and services tailored to specific requirements. It supports various SDN protocols, including Open Flow, ensuring seamless integration with SDN-enabled devices.

The attack simulation workflow in Mininet, as shown in the figure 4, includes creating a custom network topology, setting up hosts as attackers and victims, executing DoS and DDoS attacks, capturing network traffic with Wireshark, extracting features using NTLFlowLyzer, and analyzing the data with a machine learning model for intrusion detection.

After configuring Mininet, the POX controller was deployed to run in the background, enabling dynamic network management. It facilitated the development of custom network applications for traffic control, security enforcement, and monitoring. It managed switch operations and forwarding rules, ensuring efficient packet handling within the defined network topology.

A custom topology was established, connecting hosts (h1, h2, h3, h4, h5, h6, h7) with switches (s1 and s2). In this setup, Host h2 launched a Denial of Service (DoS) attack on Host h1 as shown in figure 2, while Hosts h4, h5, h6, and h7 executed a Distributed Denial of Service (DDoS) attack on Host h3 as shown in figure 3. These attacks were simulated using hping3, a command-line tool for network testing and security assessments. The DoS attack overwhelmed h1 with a high volume of packets from h2, depleting its resources and causing service disruption. In the DDoS attack, multiple hosts (h4, h5, h6, h7) simultaneously transmitted large amounts of packets to h3, intensifying the attack's impact.

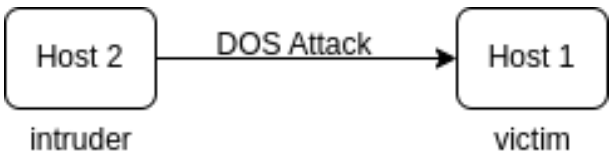

Figure 2. DOS Attack

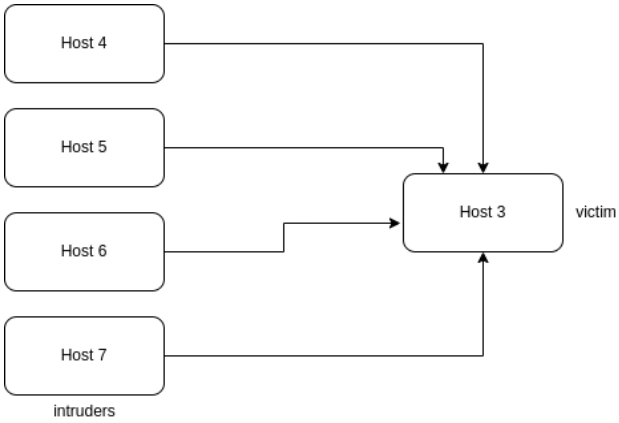

Figure 3. DDOS Attack

To monitor and analyze network activity during the attacks, packets were captured using the tcp dump command and stored in pcap format. These files were further examined with Wireshark to analyze network behavior, detect malicious traffic patterns, and assess the attack's impact. Extracted traffic features were processed using NTLFlowLyzer and analyzed with a fast KNN model for intrusion detection. Additionally, MiniEdit was utilized to provide a graphical representation of the network topology, enhancing visualization and analysis.

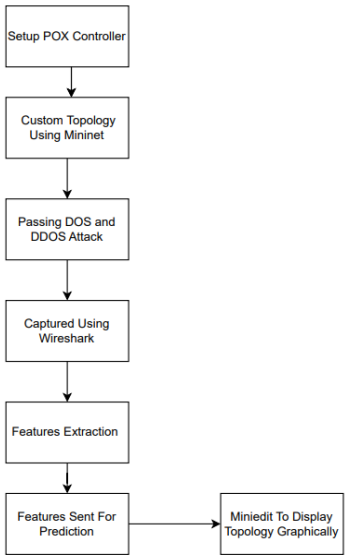

Figure 4. Simulation of attack using Mininet

3.2.3. Model Comparison

Table 1. LSTM Model

Units	**accuracy**	**dropout**	**epoch**	**learning rate**
96	91.2%	0.1	2	0.001
128	92.95%	0.1	2	0.0001
64	93.47%	0.1	2	0.01

Table 2. Fast KNN

	precision	**recall**	**f1-score**	**support**
Benign	0.99	0.99	0.99	29946
DDOS	1.00	1.00	1.00	30039
DOS	0.99	0.99	0.99	30015
Accuracy			0.99	90000

The performance of two different models, Long Short-Term Memory (LSTM) and Fast k-Nearest Neighbors (Fast k-NN), was evaluated for a Network Intrusion Detection System (NIDS) using accuracy, precision, recall, and F1-score as key metrics. The LSTM model, tested with different configurations, achieved a maximum accuracy of 93.47% with 64 units, a dropout rate of 0.1, a learning rate of 0.01, and 2 epochs. In contrast, the Fast k-NN model demonstrated superior classification performance, achieving an overall accuracy of 99% with near-perfect precision, recall, and F1-scores for Benign (0.99), DDoS (1.00), and DoS (0.99) traffic. While LSTM models require careful tuning of hyperparameters such as the number of units and learning rate to optimize performance, Fast k-NN achieved higher accuracy with a more straightforward approach, likely benefiting from its efficiency in handling high-dimensional data. The results suggest that while LSTM can be useful for sequential data modeling, Fast k-NN outperformed it in this intrusion detection task, demonstrating its effectiveness in accurately classifying network traffic. Future work could explore hybrid models that integrate the strengths of both approaches to further enhance detection performance.

4. Discussion

The classification report provides key performance metrics, including precision, recall, and F1-score, for the three classes: Benign, DDoS, and DoS. The F1-score, which balances precision and recall, is nearly 0.99 or 1.00 for all classes, indicating that the model effectively minimizes false positives and false negatives. The high recall values show that the model correctly identifies almost all actual instances of each class, while the high precision confirms that very few incorrect classifications occur. The macro and weighted averages also reinforce the model's robustness, demonstrating consistent performance across all categories. With an overall accuracy of 99%, the model shows exceptional reliability for intrusion detection tasks.

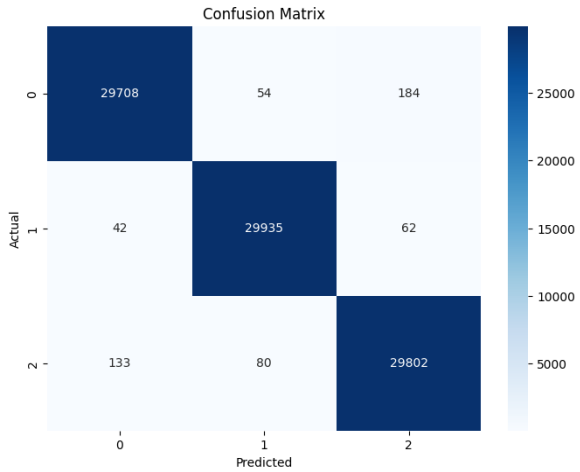

Figure 5. Confusion Matrix

In the process, I also tried using an LSTM model to train my preprocessed dataset. Despite extensive hyper parameter tuning, the best accuracy achieved was 93%. This suggests that while LSTM networks can capture temporal dependencies in the data, they may not always outperform other models for this particular task.

The confusion matrix, as shown in Figure 5, further illustrates the classification performance by displaying the actual versus predicted values. The strong diagonal dominance suggests that the model correctly classifies most instances, with minimal misclassification between classes. The misclassification rates are very low, with only a small number of benign instances being predicted as attacks and vice versa. This confirms that the

model can differentiate well between normal and attack traffic. The low off-diagonal values indicate that there is little overlap or confusion between the classes, making the model highly effective at distinguishing between benign traffic, DDoS attacks, and DoS attacks.

5. Conclusion and Future Enhancement

This project successfully developed a Network Intrusion Detection System (NIDS) using the CICIDS2017 dataset and a Fast k-NN model to classify network traffic as DoS, DDoS, or benign. We improved detection accuracy by selecting the most significant features through feature importance scoring and reducing dimensionality with PCA. To create a realistic attack environment, we used Mininet with a POX controller and OVS switch to simulate DoS and DDoS attacks between hosts. The network was visualized using MiniEdit, and traffic flows were captured using NTLFlowLyzer, allowing us to extract key features for classification. This approach effectively demonstrates the use of machine learning and software-defined networking (SDN) for detecting cyber threats.

Our project stands out from others in a few key ways. First, we used feature importance scoring to select the best features for the given dataset, a step that is often overlooked in other similar projects. This helped us ensure that only the most relevant features were included, which likely contributed to the higher accuracy of our model. Additionally, we used NTLFlowLyzer to extract the features, which is an upgraded version of CICFlowMeter. Unlike other tools, NTLFlowLyzer provides more accurate feature extraction, resulting in more precise classification and ultimately better detection performance.

For further enhancements, we can expand detection capabilities beyond TCP by including UDP and ICMP traffic, making the system more robust against a wider range of attacks. Additionally, incorporating more attack types and increasing the scale and complexity of the virtual network will improve adaptability to real-world scenarios. Advanced techniques, such as deep learning models and adaptive anomaly detection, can also be integrated to enhance detection efficiency. These improvements will strengthen the system's ability to detect and respond to evolving cyber threats in real time.

Acknowledgement

We would like to express our sincere gratitude to everyone who contributed to the completion of this project. First and foremost, we extend our heartfelt thanks to our supervisor, Er. Suman Shrestha, for his invaluable guidance and support throughout this major project. We are also grateful to the Department Head, Er. Rabindra Khati, and Project Coordinator, Er. Suman Shrestha, for their continuous encouragement and assistance. Our appreciation goes to all the faculty members of Kantipur Engineering College for their patience, motivation, and immense knowledge, which greatly aided us in all aspects of research, development, and implementation. Lastly, we would like to thank our family and friends for their unwavering support and encouragement during this journey.

References

Desai, P., Sonawane, A., Mane, T. & Jaiswal, R., n.d. NETWORK BASED INTRUSION DETECTION SYSTEM.

He, P., Feng, G., Li, H. & Yang, L., 2023. A Network Intrusion Detection Method Based on a Fast KNN Algorithm. *Academic Journal of Science and Technology,* Volume 8, pp. 81--83.

Krishna, K. V., Swathi, K. & Rao, B. B., 2020. A novel framework for nids through fast knn classifier on CICIDS 2017 dataset. *International Journal of Recent Technology and Engineering (IJRTE),* Volume 8, pp. 3669--3675.

Ravipati, R. D. & Abualkibash, M., 2019. Intrusion detection system classification using different machine learning algorithms on KDD-99 and NSL-KDD datasets-a review paper. *International Journal of Computer Science \& Information Technology (IJCSIT) Vol,* Volume 11.