FMDB *Transactions on Sustainable Computer Letters*

AI-Powered Cyber Intrusion Detection System with Real-Time Threat Analysis and Web Application Interface for Enhanced Network Security

R. Regin[1,*], G. Hariprasath[2], P. Dhinakaran[3], V. Harishkumar[4], M. Rehena Sulthana[5], S. Silvia Priscila[6]

[1,2,3,4]Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram, Chennai, Tamil Nadu, India.
[5]School of Information Technology and Engineering, Melbourne Institute of Technology, Melbourne, Victoria, Australia.
[6]Department of Computer Science, Bharath Institute of Higher Education and Research, Chennai, Tamil Nadu, India.
regin12006@yahoo.co.in[1], hg8694@srmist.edu.in[2], pp4417@srmist.edu.in[3], hk0837@srmist.edu.in[4], rsulthana@academic.mit.edu.au[5], silviaprisila.cbcs.cs@bharathuniv.ac.in[6]

Abstract: This research paper presents a novel idea for an AI-powered Cyber-intrusion detection system that combines the architecture TabNet with Deep Neural Networks (DNNs) to enhance network security. The Tabnet utilises a sequential attention mechanism for dynamic feature extraction and selection, and the selected features are fed into a deep neural network for accurate prediction. The Tabnet prioritizes important features at every decision point, providing Interpretability through an attention mechanism that improves it by highlighting the most important features that influence predictions. The extracted feature subset is then fed to a deep neural network, which is well-suited for learning and identifying complex patterns in an optimized feature space. The proposed approach works well with high-dimensional network traffic data, improving detection performance. In this proposed approach, the system is also deployed with a frontend web application using Python's Flask library to make predictions using a customised dataset. Users can enter the values of specified features to predict the type of cyber-attack. The model is evaluated on the CICIDS 2018 dataset, which consists of 79 features and 24,67,820 network data, and has also obtained an accuracy of 94.22%. The proposed system outperforms traditional machine learning models in intrusion detection.

Keywords: AI-powered Intrusion Detection System (IDS); Deep Neural Networks (DNNs); Sequential Attention Mechanism; Flask-based Web Application; Network Security; Feature Selection and Extraction.

Received on: 19/08/2024, **Revised on:** 29/10/2024, **Accepted on:** 27/11/2024, **Published on:** 01/03/2025

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSCL

DOI: https://doi.org/10.69888/FTSCL.2025.000358

Cite as: R. Regin, G. Hariprasath, P. Dhinakaran, V. Harishkumar, M. R. Sulthana, and S. S. Priscila, "AI-Powered Cyber Intrusion Detection System with Real-Time Threat Analysis and Web Application Interface for Enhanced Network Security," *FMDB Transactions on Sustainable Computer Letters*., vol. 3, no. 1, pp. 34–49, 2025.

1. Introduction

With the growth in digital technologies and network infrastructure, cybersecurity threats have become a significant concern for individuals as well as organisations worldwide. In today's digital world, networks are growing increasingly complex and interconnected, and they face a proportional increase in cyberattacks. Intrusion detection systems (IDS) serve as an important

*Corresponding author.

line of defence by monitoring network traffic and system activities for malicious actions or security policy violations. Traditional Intrusion detection systems (IDS) rely on rule-based or statistical methods, which often struggle with high-dimensional network traffic data and fail to detect novel attack patterns effectively, thus necessitating more advanced techniques capable of recognizing hidden patterns without prior knowledge of attack signatures. The advancements in machine learning and deep learning have revolutionised the field of Intrusion detection by enabling systems to learn patterns autonomously through training with vast amounts of data. Machine learning and deep learning-based approaches have emerged as promising solutions, leveraging data-driven methods to enhance accuracy and adaptability. Conventional Machine Learning methods require manual feature engineering and struggle with Interpretability, which can be time-consuming and prone to human bias. These conventional methods present a challenge in feature extraction and computational efficiency when using high-dimensional traffic data for model training.

The network dataset comprises features such as packet size, protocol types, and port numbers, among many others, which exhibit high dimensionality and complexity. Feature extraction is a critical preprocessing step that aims to transform the data, reducing computational load and enhancing the system's ability to differentiate normal and cyberattacks. Traditional methods, such as manual feature selection based on statistical techniques like Principal Component Analysis (PCA), often fail to identify complex patterns in data and are considered time-consuming. Tree-based models, such as Random Forest and Gradient Boosting, can handle high-dimensional data and also provide us with feature importance scores. A slight disadvantage with these models is that they treat all features equally, which may not be efficient, as a model capable of dynamic features focuses on a different stage. This paper introduces a novel approach for an Intrusion Detection System that leverages the Tabnet architecture and Deep Neural Network (DNNs) for final classification and prediction. The TabNet classifier is used for feature selection, utilising the selected features from TabNet and a deep neural network for prediction. In this paper, a frontend is also deployed using the Flask Python library. Our proposed system addresses several key limitations of existing approaches and demonstrates superior performance across multiple evaluation metrics on standard benchmark datasets. The dataset comprises various types of cyberattacks, including Distributed Denial of Service (DDoS), Denial of Service (DoS), Brute Force, Botnet attack, Infiltration, and web attacks. The attack types present in the dataset enable the model to learn and generalise across different cyber-attack categories, making the model more effective in real-world scenarios.

Despite the challenges associated with traditional machine learning methods, the TabNet classifier—a deep learning architecture specifically designed for tabular data—offers a promising solution. This Tabnet classifier uses sequential attention mechanisms, which focus on different features at each step, and also combines the high performance of neural networks with the Interpretability of decision trees. This attention mechanism in the tabnet classifier is a useful feature because it prioritises the important features that can be used for predictions. This mechanism enhances detection accuracy and efficiency. Tabnet's architecture comprises an encoder, a decoder, and attentive transformer modules, enabling it to handle both numerical and categorical features commonly found in network traffic logs. Its Interpretability, derived from attention mechanism masks, allows security analysis to understand which features drive each detection, facilitating model validation and operational insights.

Once the key features are extracted using Tabnet, our proposed system employs a Deep Neural Network for prediction. DNNs, with their multiple layers, can learn complex patterns that raw features might obscure. By feeding Tabnet–extracted features into the DNN, the system classifies any cyber-attack, harnessing a network's selection prowess. This hybrid approach is expected to minimize the trade-off between detection accuracy and false positives, a significant challenge in Intrusion Detection Systems. By deploying the model, we achieved an accuracy of 94.22%, demonstrating the model's ability to detect cyberattacks while reducing the percentage of false positives. This high accuracy indicates that the hybrid model, composed of TabNet and a Deep Neural Network for classification, is a robust approach for intrusion detection. To make it more effective and accessible to the general public, this paper also presents a frontend application developed using the Flask library in Python. This frontend web application enables users to interact with the system with ease. The front end, developed using a Python library, gives two functionalities to the users:

- **CSV file upload:** This feature allows users to upload their own CSV file containing network traffic data for specified features. The system will process the data to predict the type of attack for each instance.
- **Manual Entry mode:** The User can enter values manually for the specified features, and the system will predict the type of cyber-attack in real-time.

This frontend application enables users to analyse network traffic and detect potential threats more efficiently. In conclusion, our proposed system combines the strengths of Tabnet and Deep neural networks to improve the efficiency of feature selection and classification accuracy. The proposed system can be used efficiently with high-dimensional network traffic data while maintaining Interpretability, accuracy, and efficiency. The proposed system produced an accuracy of 94.22% and outperformed many traditional approaches, offering a scalable, adaptable, and user-friendly solution for cybersecurity threat detection.

In addition to the proposed system, the frontend application also provides a practical interface for real-time intrusion detection, allowing users to manually upload CSV files or input values for the specified features to accurately predict types of cyberattacks. Future work may also focus on optimising the model's architecture, integrating additional datasets to achieve broader generalisation, and incorporating real-time network monitoring capabilities to enhance its practical deployment in dynamic network environments.

2. Objective

- To create a cyber intrusion detection system based on AI to pinpoint potential cyberattacks with accuracy and efficiency.
- To preprocess a large dataset using techniques such as missing value handling, feature scaling, and encoding for optimal machine learning, thereby facilitating the training of the model.
- For real-time threat analysis to enable proactive cybersecurity solutions.
- To create a frontend web application to ensure effective usability and decision-making monitoring.

3. Literature Survey

Siddiqi and Pak [1] proposed an advanced network intrusion detection system (NIDS) that combines image processing with machine learning. It optimises feature selection and transforms non-image data into enhanced images for improved anomaly detection using a convolutional neural network (CNN). The framework is tested on benchmark datasets, showcasing its effectiveness against cyber threats. Zuniga-Mejia et al. [2] have proposed two intrusion detection techniques for routing in reconfigurable wireless networks (RWNs) based on linear systems theory. These methods address challenges in dynamic routing environments and the need for efficient detection of routing-related attacks. The analysis focuses on node behaviour through linear systems, with a particular emphasis on identifying malicious activities by examining pole locations. Simulations demonstrate high detection accuracy and responsiveness to various attack scenarios. Park et al. [3] have discussed the study of using AI-driven NIDS methodology, which includes preprocessing, generative model training, autoencoder training, and predictive model training. Data preprocessing techniques, such as outlier analysis, one-hot encoding, and feature scaling, are essential for refining the raw data. The system is evaluated using benchmark datasets, including NSL-KDD, as well as real-world data.

Ben Said et al. [4] have conducted a study on CNN-BiLSTM hybrid deep learning methodology for network intrusion detection in SDN. The system architecture includes three components: Anomaly Mitigator, Anomaly Detector, and Flow Collector. It utilises ten distinct features for multi-class classification, thereby enhancing intrusion detection and categorisation. Data transformation and balancing techniques are emphasised to enhance model performance and strengthen SDN resilience against cyber threats. Dietz et al. [5] have conducted a study on the absence of publicly available (pseudo)code, which limits the reproducibility of AI/ML methodologies, hindering practitioners' ability to replicate results and integrate tools. Ensuring model generalizability requires testing across diverse datasets and network variations to evaluate real-world applicability. The study identifies six personas from enterprise scenarios, highlighting distinct roles and challenges in using AI/ML tools. Through a literature review, the research formulates 17 hypotheses addressing challenges in implementing and adopting AI/ML technologies. Qazi et al. [6] present a paper on NIDS for Zero-Touch Networks in smart cities, utilizing deep learning methods, such as CNNs, to enhance security. It addresses the challenges posed by IoT integration, emphasising the need for strong intrusion detection systems. The CICIDS-2018 dataset is utilised for model training, enabling the detection of diverse cyber threats with high accuracy. The goal is to strengthen the resilience and safety of smart city infrastructures against emerging cyber threats.

Halbouni et al. [7] conducted a study on a hybrid deep neural network that combines CNN and LSTM to improve network intrusion detection. It addresses challenges such as increasing cyber threats and high false alarm rates in traditional systems. The model utilises batch normalisation and dropout layers, achieving high accuracy and detection rates on diverse datasets. The approach enhances detection of both known and novel attacks, bolstering overall network security. Wisanwanichthan and Thammawichai [8] present a paper on the Double-Layered Hybrid Approach (DLHA). It aims to enhance the detection of both common and rare attack vectors, addressing the growing cybersecurity threats. PCA is utilised for dimensionality reduction, focusing on relevant features and enabling faster real-time processing. The DLHA aims to provide an efficient solution for identifying and preventing network intrusions. Dutt et al. [9] proposed a model that mimics the human immune system to enhance network intrusion detection. The author has designed a system that not only monitors network traffic but also deploys two layers of detection techniques to effectively identify potential intrusions. The paper also introduces a Statistical Modelling-based Detection (SMAD) for the initial detection phase. The second layer is an Adaptive Immune-based Anomaly Detection (AIAD), which focuses on identifying suspicious packets. Kim and Pak [10] present a study that proposes a classification algorithm combining LSTM, DNN, and GAN for real-time network intrusion detection. Unlike traditional methods, it enables

prompt detection without session-completion delays. Packet data is transformed into image-like features, improving the model's ability to identify malicious traffic.

The approach also creates a new dataset using misclassified data to train a Generative Adversarial Network (GAN). Experimental results demonstrate high detection performance and faster response times, rivalling those of existing systems. The proposed approach aims to reduce false positives and enhance detection accuracy before session termination. Yang and Wang [11] propose a wireless network intrusion detection method based on an improved convolutional neural network (ICNN) that autonomously extracts sample features and optimizes network parameters using a stochastic gradient descent algorithm to enhance detection accuracy and reduce false positive rates. The simulation results of this proposed method demonstrate higher detection accuracy and a higher true positive rate, along with a lower false positive rate, compared to the traditional method. Saikam and Ch [12] proposed a hybrid network intrusion detection system (NIDS) that combines DenseNet 169 and self-attention-based Transformer (SAT-Net) for feature extraction, along with the Enhanced Elman Spike Neural Network (EESNN) for classification. They also address data imbalance issues using hybrid sampling techniques to improve detection performance. The proposed method also utilises a difficult set sampling technique (DSSTE) algorithm to reduce the number of noise samples.

Yang et al. [13] present a real-time intrusion detection mechanism using a Conditional Deep Belief Network (CDBN) to identify attack features in wireless networks. The “SamSelect” algorithm handles dataset imbalance, and a Stacked Contractive Auto-Encoder (SCAE) reduces data dimensionality. The proposed method, which can handle high-dimensional data, has been validated through extensive testing on benchmark datasets. The experimental results of the proposed method demonstrate its capability for feature learning, making it suitable for dynamic and evolving attack patterns. Wali et al. [14] introduce a unified multimodal dataset for network intrusion detection, combining flow data, payloads, and contextual features to improve detection accuracy. Their approach addresses the lack of standardisation and payload data in existing datasets. The proposed dataset enhances the performance of machine learning models and supports cross-dataset validation. This paper introduces a new dataset that integrates various data modalities to improve generalisation across different attack scenarios.

4. Methodology

4.1. Existing Method

4.1.1. Support Vector Machine

The SVM-based method, utilising an RBF kernel, has been adopted for this study to detect cyber intrusions. The entire dataset has been preprocessed by handling infinite values and imputing missing data with the mean. Due to the nature of the labels, they were encoded to fit the machine learning framework. The dataset was too imbalanced; thus, Random Under-Sampling was adopted to balance the recorded different attack types. As mentioned above, the training and testing sets are created, and the features are standardised to improve the model's performance. The SVM is trained with an RBF kernel to catch complex behaviour in the dataset. Finally, the model's performance is evaluated using a classification report, which provides accurate statistics for individual attack types.

4.2. Proposed Methodology

The purpose of this project is to enhance network security by identifying and categorising cyber threats using TabNet and Deep Neural Networks (DNNs). The Hybrid AI-Powered Cyber Intrusion Detection System is an implementation that processes selected features of network traffic through machine learning models and further processes the data to identify various forms of attacks. The use of feature selection (TabNet) in conjunction with deep learning (DNN) enhances the usability and precision of automated intrusion detection systems. A user-friendly web application, developed using Flask, enables users to submit traffic data and receive real-time network intrusion predictions.

4.3. TabNet

TabNet is an advanced model for deep learning in handling tabular data very well. Unlike conventional neural networks, where the inputs are fed all at once, TabNet selects the required features at all time steps by employing attention, except for redundancy. This is one of its significant benefits: due to its individualistic approach to feature selection, the model is more interpretable and produces fewer computations, resulting in increased efficiency and explainability. TabNet enables focusing on only those features that truly matter, while ignoring the rest. This speeds up the model and prevents the generalisation problems associated with overfitting. Therefore, it is widely used in applications that require feature importance, such as cybersecurity, finance, healthcare, and fraud detection. Due to its inherent ability to dynamically identify pertinent features among many related candidates for any given task, TabNet has found wide applicability across diverse fields, ranging from relatively simple to very complex and data-driven tasks. In cybersecurity, TabNet helps detect malicious activities and prevent

cyberattacks by identifying unusual patterns in network traffic. In the banking and finance sector, TabNet is utilised for credit scoring, fraud detection, and risk assessment in decision-making processes. In the healthcare industry, it aids in disease prediction and facilitates the analysis of patient data for early diagnosis and treatment.

The retail and e-commerce space utilises TabNet for customer segmentation, recommendation engines, and sales forecasting, enabling smart marketing. In industries such as manufacturing and IoT, the model is used for predictive maintenance and anomaly detection, therefore ensuring operational excellence and minimising downtime. Additionally, TabNet effectively contributes to natural language processing (NLP) and time series analysis, which derive insights from structured text data and sequential data, respectively. This feature of automatically identifying the most relevant features makes TabNet suitable for datasets that would benefit enormously from intelligent feature selection, resulting in improved performance and Interpretability across various applications.

TabNet has several advantages, which enhance its effectiveness in various machine learning tasks. Primarily, it is interpretable, unlike conventional deep-learning models, which provide insight into the features that drove a particular decision. Furthermore, powerful computations enable the selection of only the most relevant features on an as-needed basis during each step of the decision process, thereby saving unnecessary calculations and accelerating the process. Sparse feature selection reduces the risk of overfitting in TabNet, thereby enabling it to generalise to new data. It also handles missing data effectively and can work with incomplete datasets that require minimal imputation. Thanks to its self-supervised learning capabilities, it can learn from both labelled and unlabeled data, making it useful in a mixed form across various domains. Moreover, it can perform parallel processing, handling large datasets very efficiently. The model takes input data through multiple decision steps, selecting the most relevant features at each step for classification or regression tasks. It uses attention mechanisms to focus on relevant features while avoiding redundant computation selectively. Instead of predicting in a single pass, it sequentially refines its predictions in every step before aggregating the outputs for the final prediction. To extract meaningful patterns, TabNet employs non-linear transformations, such as Gated Linear Units (GLUs), to capture complex relationships within the data. The training of the model is accomplished through backpropagation and gradient descent to minimise the loss function. In the case of classification, such as intrusion detection, the function is categorical cross-entropy.

TabNet performs feature selection and classification within the hybrid AI-powered cyber intrusion detection system. Firstly, the dataset gets preprocessed and normalised, where important features like 'Fwd IAT Mean', 'Bwd Pkt Len Mean', 'Init Fwd Win Byts', 'Pkt Len Mean', 'Flow Pkts/s', 'Fwd Seg Size Min', and 'Timestamp' are selected for study. The next step is to activate the TabNet model, which is parameterised (with learning rate, batch size, and number of decision steps) and will be optimised using gradient descent. The trained model is then saved in tabnet_model.pkl for deployment purposes. Later, evaluation is done on unseen test data to calculate accuracy, precision, recall, F1-score, and ROC curve as performance metrics. The final step involves deploying the trained TabNet model within a Flask web application, aiming for real-time intrusion detection using extracted network traffic data. The configuration of dynamic feature selection, Interpretability, and computational efficiency is what makes TabNet a valid contender in cyber intrusion detection, which means we can handle network traffic to identify security threats robustly with high accuracy and negligible computational expense.

4.4. Deep Neural Network (DNN)

DNNs, or Deep Neural Networks, are a class of machine learning algorithms that are by far the most powerful in terms of multiple-layered neuron connectivity, which allows DNNs to learn complex patterns from data. Unlike traditional machine learning, DNNs can automatically extract raw data without the need for manual feature extraction. Due to high scalability, DNNs can be applied to larger datasets and high-dimensional problems. One of the most powerful aspects of DNNs is their ability to model complex relationships among features using multiple hidden layers with non-linear activation functions.

Training a deep neural network involves forward propagation of input data through layers and backpropagation, adjusting weights using gradient descent to minimise loss function values. Typically, the loss function used in classification is categorical cross-entropy, as it measures the distance between two predicted probabilities and the actual class labels. The DNN model in this hybrid AI-powered cyber intrusion detection system is designed to process selected network traffic features and classify different types of cyber intrusions. The data needs preprocessing, which includes normalisation and selecting the most relevant features for model training. The DNN architecture features multiple dense layers, with ReLU (Rectified Linear Unit) activation in the hidden layers to introduce non-linearity and softmax activation in the output layer, which converts raw logits into probabilities.

Training of this model is possible through backpropagation and the Adam optimiser, which is an efficient weight update mechanism. The trained model acts as a dedicated model for the deployment phase and is saved in dnn_model.h5. Evaluation will include metrics such as accuracy, precision, recall, F1-score, and ROC curve, which enable one to determine how well the model can find intrusions. The trained DNN model can be embedded into a Flask web application, where it will be used for

real-time processing of user-uploaded network traffic data via CSV files or manual entry. Besides being able to process large-scale data, extract complex patterns, and generalise to unseen attacks, DNN becomes the heart of our hybrid AI-powered cybersecurity system, ensuring robust and accurate threat detection.

4.5. Algorithm

BEGIN

- Install pytorch-tabnet, imbalanced-learn, tensorflow, keras
- Import pandas, numpy, sklearn, keras, tensorflow
- Read “CICIDS2018.csv” into dataframe
- Fill missing values
- X = all columns except target
- y = target column (encoded)
- Apply undersampling to balance classes
- Split X, y into train and test sets
- Initialize TabNet
- Train on X_train, y_train
- Select top 7 important features
- Apply StandardScaler
- TabNet_train_features = TabNet.predict_proba(X_train_scaled)
- TabNet_test_features = TabNet.predict_proba(X_test_scaled)
- Define model with input layer, dense layers, dropout, and output layer
- Compile and train model on TabNet_train_features
- Predict attack type
- Save TabNet, DNN, scaler, and label encoder
- Load models
- Create routes for CSV and manual input predictions
- Return results as JSON
- Run Flask app

END

4.6. Mathematical Formulations

4.6.1. Cross-Entropy Loss Function

The loss function for the DNN model is categorical cross-entropy, which measures the difference between the actual and predicted probability distributions.

$$L= - \sum_{i=1}^{N} y_i \log(y_i\hat{})$$

where:

- Yi represents the actual label.
- yi^ represents the predicted probability.
- N is the total number of samples.

4.6.2. Softmax Function

The softmax function converts raw model outputs (logits) into probabilities:

$$\sigma(z_i) = e\hat{}(z_i) / \Sigma_j e\hat{}(z_j)$$

Where Zi represents the logits before the softmax transformation.

4.6.3. TabNet Feature Selection Mask

TabNet applies feature selection dynamically using an attention-based mechanism: $M = f(\theta)$, where M is the feature mask and θ represents the learnable parameter controlling feature selection.

4.6.4. Binary Cross-Entropy Loss

Binary cross-entropy loss measures the error in a binary classification model.

$L = -1/N \sum_i^N [y_i \log(y_i\hat{}) + (1-y_i) \log(1-y\hat{}_i)]$

4.6.5. Accuracy Metric

Accuracy measures the proportion of instances that are correctly predicted.

Accuracy=Total Predictions/Correct Predictions

4.6.6. Precision, Recall, and F1-Score

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

- Precision=TP/TP+FP

Recall measures the proportion of correctly identified positive instances.

- Recall = TP/TP+FN

F1-score is the harmonic mean of precision and recall, balancing both metrics.

- F1 – Score = 2 x Precision x Recall/Precision + Recall

4.7. Architecture Diagram

Figure 1 illustrates the workflow of a Hybrid AI-powered cyber intrusion detection system, comprising backend and frontend components. In the back end, CSV input data is first received for data preprocessing, which includes operations such as handling missing values, label encoding, and data balancing, all of which are important for a distribution dataset.

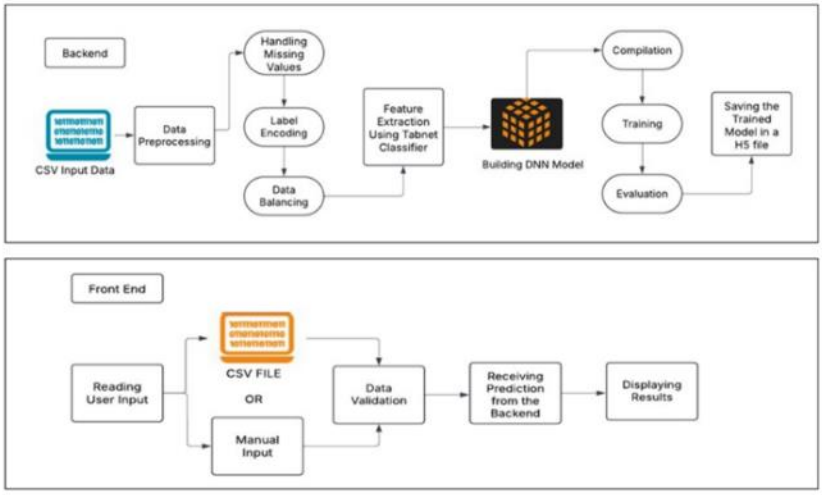

Figure 1: Architecture diagram

Feature selection is performed by the TabNet classifier, which identifies the most important features and then passes the filtered data to the DNN model for training. The DNN model-building cycle consists of the compilation stage, training stage, and evaluation stage. Once trained, the model can be saved in an HDF5 (H5) file for subsequent deployment and use. The front end is designed to include an interface where input data can be provided, either from a CSV file or manually entered. Input data is validated before being forwarded to the backend for processing. The backend processes the input data using the trained model and sends back prediction outputs to be displayed on the user interface. Thus, this architecture facilitates efficient feature selection and model training, as well as real-time intrusion detection, rendering the system highly scalable in identifying cyber threats.

4.8. Flask

Flask, a lightweight and flexible Python web framework, can be effectively used to build web applications. It is such a simple and scalable framework that lends itself especially well to deploying machine-learning and deep-learning models in applications. Moreover, this framework enables backend logic to be merged and interfaced with frontend components, making it a suitable candidate for real-time systems that involve data processing and prediction. In this Hybrid AI-powered cyber Intrusion Detection System, Flask serves as a backend framework, enabling the combination of trained machine learning models (TabNet and DNN) with the user interface. Here's how it is being utilised:

- Model Loading-Finally, the pre-trained TabNet model (tabnet_model.pkl) and DNN model (dnn_model.h5) have been loaded by Flask for real-time intrusive detection.
- User Input Handling-The front end allows the user to upload a CSV file or input data manually. Flask processes this and passes it to the prediction functions.
- Data Validation and pre-processing must be done before making predictions. Flask utilises this mechanism to ensure that the data is in the appropriate format, preprocessed as required, and features are selected according to TabNet.
- Making Predictions-Takes the input that has been processed and enters it into the prediction models. The models then return the predicted attack type (Normal, DoS, Probe, R2L, U2R, etc.).
- Returning Results-The predictions are sent by Flask to the frontend, where they are presented to the user.
- Deployment & API Creation-The intrusion detection is consumed as a web-based Flask application, allowing interaction through a browser.

With this, the integration of machine learning use cases and data processing, along with a friendly web interface, becomes possible, making this intrusion detection system efficient and easy to use in live and participatory cybersecurity applications.

5. Implementation

5.1. Data and Preprocessing

The collection of data specified in this Hybrid AI-Powered Cyber Intrusion Detection System will serve as an exploratory feature of cyber traffic, enabling the classification and identification of various types of cyberattacks. The data preprocessing pipeline will initiate data cleaning, which includes handling missing values to maintain data accuracy and integrity. After the data is cleaned, label encoding will be applied to the categorical variables to convert them into a numerical format suitable for model training. Then, to reduce class imbalances of the data, it will also undergo balancing techniques such as oversampling or undersampling, to ensure adequate representation in each of the attack types.

The importance of feature selection procedures will be reduced by inspecting the retained attributes after using the TabNet classifier to include only the most important ones. The features that will be retained for model training are considered the 'Fwd IAT Mean,' 'Bwd Pkt Len Mean,' 'Init Fwd Win Byts,' 'Pkt Len Mean,' 'Flow Pkts/s,' 'Fwd Seg Size Min,' and 'Timestamp.' Once the preprocessing of the data proceeds past cleaning and optimization of the original collected data, normalization will also be used to standardize the data processing procedure and normalize all numerical values, ultimately improving the efficiency of model prediction. The processed data will then be added to the DNN (Deep Neural Network) model, which has been trained to classify network traffic into related intrusion types. Overall, the preprocessing of data in the AI-Powered Cyber Intrusion Detection System will ensure that the models receive higher-quality input data, improving the prediction models and the overall reliability of the system.

5.2. Data Visualisation

Figure 2 illustrates the relationships between the selected features and how different attack types (labels) are distributed across them. The diagonal plots display the distribution of each feature, using a kernel density estimate (KDE). Although they may not appear effective individually for differentiating between attack types, some distinct peaks suggest that certain features can

effectively differentiate between them. For instance, “RST Flag Cnt” and “Init Fwd Win Byts” have quite different values for the different types. The other scatter plots display pairwise relationships and indicate cluster formations or patterns.

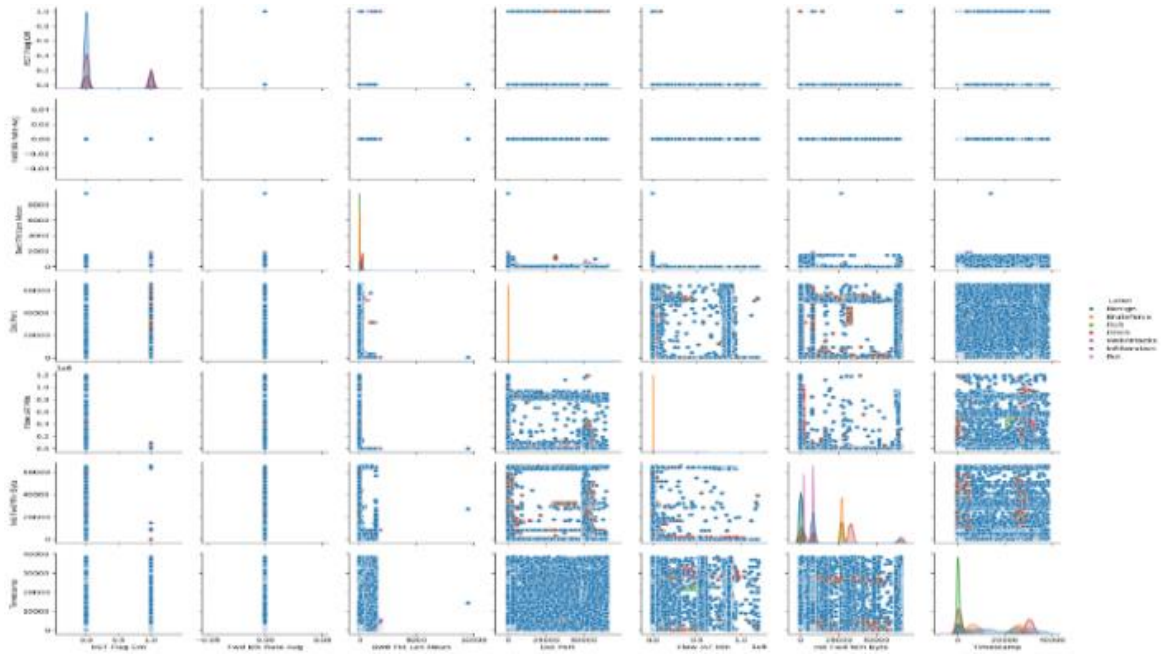

Figure 2: Pair plot

A few features, such as “Dst Port” and “Flow IAT Min,” form very close point clusters, implying that they share similar values under varying labels, while others, like “Bwd Pkt Len Mean,” have wider spreads. Where there are shared points, it suggests that the features are correlated and could complicate the classification of attacks.

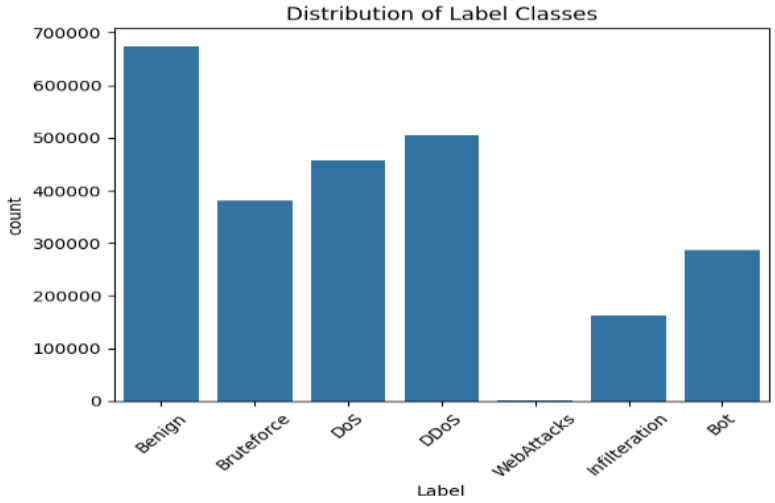

Figure 3: Distribution of label classes bar plot

Figure 3 illustrates the distribution of distinct label classes within the dataset, indicating the number of instances for each category type. The benign class has the highest number of samples, indicating that normal network activity is the predominant type. Concerning the attacks, DDoS and DoS attacks are represented in significant numbers, suggesting that these types of attacks occur frequently in the dataset. Additionally, Brute Force attacks also have relatively significant amounts, highlighting the prevalence of these attacks as a cyber threat. Both bot and infiltration attacks occur much less frequently, indicating a relatively low occurrence. The web attack type has the fewest instances, making it a rare class in this dataset. This redistribution will be useful for understanding class imbalances, which is important when training intrusion detection capabilities, whether that involves utilising methods such as DNN and TabNet.

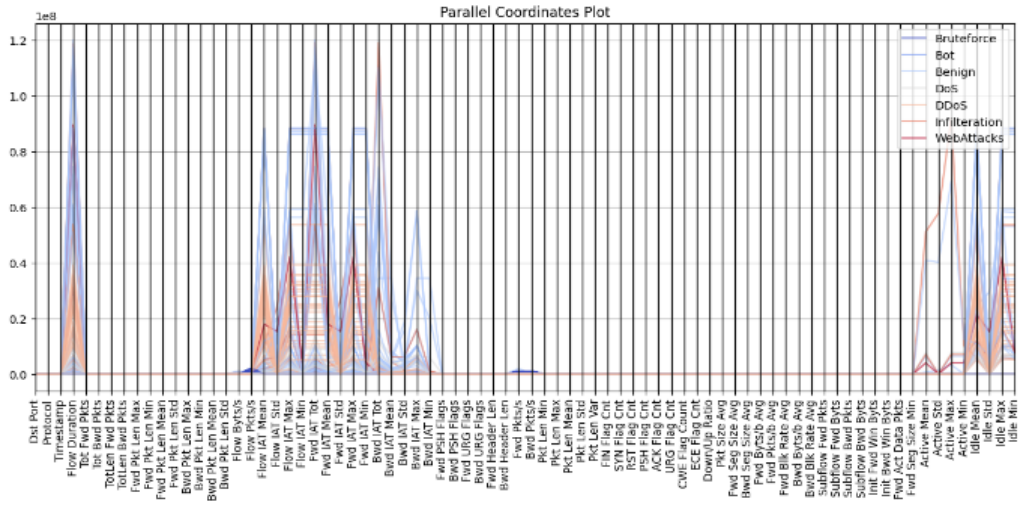

Figure 4: Parallel coordinates plot

Figure 4 illustrates the distribution of distinct label classes in the dataset, highlighting the number of instances for each category. The benign class has the highest count of samples, suggesting that normal network activity is predominant. Regarding the attacks, DDoS and DoS attacks are represented in significant numbers, indicating that these types of attacks occur frequently in the dataset. Additionally, Brute Force attacks also have relatively significant amounts, highlighting the prevalence of such attacks as a cyber threat. Both bot and infiltration attacks occur infrequently and hence tend to show a relative absence of occurrence. The web attack type has the fewest instances, thus making it a rare class in this dataset. This redistribution will help understand class imbalances, which is an important factor when training intrusion detection capacity, whether using techniques like DNN or TabNet.

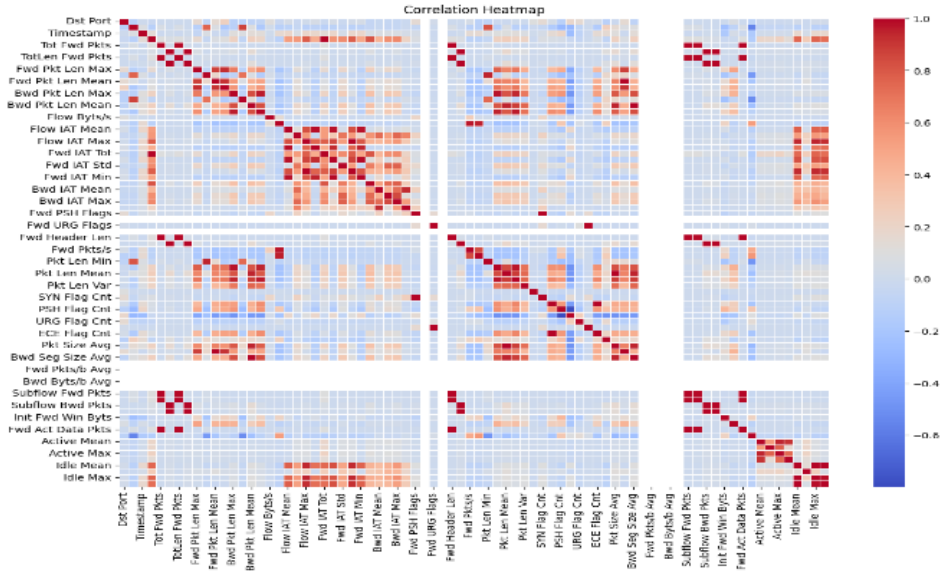

Figure 5: Heat map

Figure 5 displays the relationship between numerical features within the dataset. A strong positive correlation is indicated by areas coloured dark red, where any two features tend to increase or decrease together in tandem. A strong negative correlation is conversely marked by areas of dark blue, where one feature increases while the other decreases. White or light areas signify weak or no correlation. The diagonal line across the heat map represents self-correlation, which, of course, will be 'one'. The heat map, therefore, is apparent in having separate blocks, indicating sets of features possibly related to each other and presenting some redundancy. The identified features reveal significant patterns related to packet size, flow duration, and flag counts. These correlations can further influence feature selection and dimensionality reduction, therefore improving the

intrusion detection system once adequate features are selected while eliminating redundant ones to the advantage of efficiency and performance.

5.3. Model Training

Model training was done using TabNet and a Deep Neural Network (DNN) for the cyber intrusion detection system. First, TabNet was run against the dataset to identify features, after which dimensionality reduction was performed on these features while retaining the patterns. DNN was then trained based on the selected features to capture the complex relationships present in network traffic attributes. Hence, hyperparameter tuning was applied to enhance the model subdivision. The dataset was split into training and testing sets, which will provide a measure of performance. Accuracy, precision, recall, and F1-score are used for assessing the performance of the models. After achieving good performance, the trained models were saved in .pkl and .h5 files. Finally, these models were integrated into a Flask web application for cyber intrusion detection.

5.4. Model Evaluation

In evaluating the model for this cyber intrusion detection system, we focused on performance assessments of TabNet and the Deep Neural Network (DNN). The models were evaluated on a separate validation dataset for their ability to detect various types of cyberattacks. Metrics related to accuracy, precision, recall, and F1-score were measured for performance evaluation, with a confusion matrix providing insight into misclassifications. The TabNet model performed feature extraction, another aid to increasing DNN efficiency, which subsequently proved capable of capturing a wide range of complex network traffic patterns and thereby enhancing actual detection accuracy. The two models were then compared, and the most effective one was selected for deployment. These models were integrated into the Flask web application for live intrusion detection.

6. Result and Discussion

The hybrid AI-based cyber intrusion detection system was implemented and tested on a Windows operating system with an Intel Core i5-12450H processor, 16 GB of RAM, and a GeForce GTX 1650 GPU. It was also run on Google Colab and Kaggle Notebooks. This system utilises a machine learning framework that combines TabNet for feature selection and a Deep Neural Network (DNN) for classification purposes. The dataset was divided into 80% of the total classes for training, while the remaining 20% were allocated to the test data.

Moreover, RandomUnderSampler was used to handle the imbalanced classes and achieve equal sample sizes for different types of attacks. The importance of features was revealed through the TabNet importance score, and the model was trained on this subset of selected features. The final hybrid model comprises TabNet for feature extraction and a DNN for classification. Performance evaluation was based on the following metrics: accuracy, precision, recall, and F1-score. All the trained models, along with the preprocessing tools, were saved for deployment into a web application.

Table 1: Comparison of TabNet + DNN model and SVM model

Metric	TabNet + DNN Model	SVM Model
Accuracy	94.22%	93.34%
Macro Avg Precision	0.91	0.93
Macro Avg Recall	0.91	0.88
Macro Avg F1-Score	0.91	0.90
Weighted Avg Precision	0.94	0.93
Weighted Avg Recall	0.94	0.93
Weighted Avg F1-Score	0.94	0.93

Table 1 indicates that the TabNet + DNN model and the SVM model have nearly identical performance evaluations, with only a few exceptions. Starting with accuracy, TabNet + DNN achieved a slightly higher overall accuracy rate of 94.22% compared to 93.34% for SVM. In terms of classification-wise performance, both models perfectly identified DDoS and Brute-force attacks with an F1-score of 1.00. The recall score for Infiltration attacks was higher for TabNet + DNN (0.86) compared to SVM (0.75), indicating that TabNet + DNN performed better in detecting instances of these attacks. SVM performed better in terms of precision for WebAttacks (0.96 vs. 0.77), implying it generated fewer false positives but more missed detections. The performance was similar for both bot and DoS attacks, whereas DNN performed marginally better in classifying benign traffic, achieving an F1-score of 0.85 compared to 0.81 for SVM. Overall, TabNet + DNN performed better in terms of precision, recall, and F1-score in most categories, making it a more effective option for detecting various types of traffic and attacks.

Table 2: Class-wise comparison

Class	Precision (TabNet)	Precision (SVM)	Recall (TabNet)	Recall (SVM)	F1-Score (TabNet)	F1-Score (SVM)
Benign	0.88	0.80	0.81	0.82	0.85	0.81
Bot	0.99	0.99	0.99	1.00	0.99	0.99
Bruteforce	0.99	1.00	1.00	1.00	1.00	1.00
DDoS	1.00	1.00	1.00	1.00	1.00	1.00
DoS	0.99	1.00	0.98	1.00	0.98	1.00
Infiltration	0.78	0.77	0.86	0.75	0.82	0.76
WebAttacks	0.77	0.96	0.74	0.62	0.76	0.75

Table 2 presents a performance comparison of TabNet + DNN with the SVM model across various attack categories. In terms of DDoS and Brute-force attack detection, both models achieved perfect precision, recall, and F1-scores of 1.00, indicating that they were able to classify all instances of the attacks correctly. Regarding Bot and DoS attacks, both models performed nearly identically. TabNet + DNN slightly outperformed SVM in detecting Benign traffic with an F1 score of 0.85 compared to 0.81.

For Infiltration attacks, we observe that TabNet + DNN performed better in terms of recall (0.86 vs. 0.75), indicating that it detected more instances. In contrast, its counterpart, SVM, maintained higher precision on WebAttacks (0.96 vs. 0.77), resulting in a lower false positive rate but with decreased recall. With most attack classes, TabNet + DNN, therefore, managed to show a more balanced and consistent performance, rendering itself as a slightly more trustworthy option for network intrusion detection.

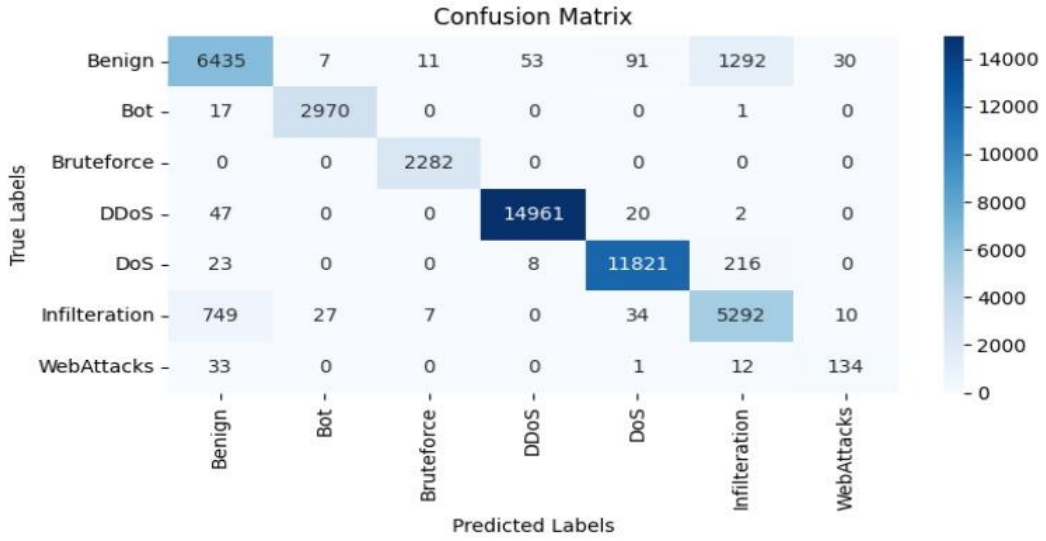

Figure 6: Confusion matrix

Figure 6 presents this performance statistically, differentiating between types of network traffic and cyberattacks. In this case, the rows represent actual (true) labels, while the columns represent predicted labels. Values along the diagonal of the confusion matrix indicate correctly classified instances for that attack type, while values off the diagonal represent instances in which a misclassification occurred. For example, there are 6435 Benign samples correctly predicted as Benign, and 1292 samples that were predicted as "Infiltration," resulting in misclassification.

Similarly, the model classified 14961 DDoS attacks as DDoS attacks with only a few misclassifications. The confusion matrix indicates a strong performance in detecting all attack types, while also showing some challenge in distinguishing "Benign" traffic from "Infiltration" traffic. Additionally, the heatmap shading indicates the density of predictions; darker colours denote higher class prediction values. This analysis will ultimately aid model development, resulting in an improvement to classification performance.

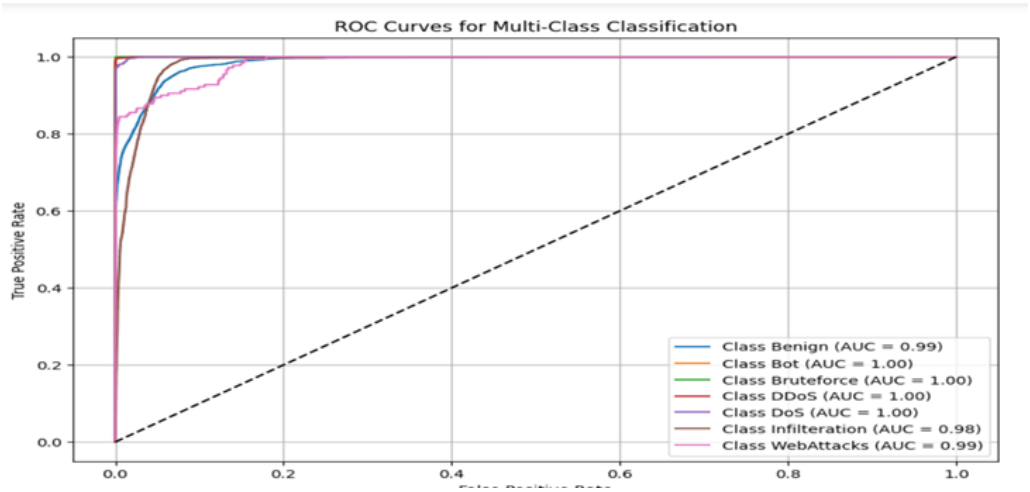

Figure 7: ROC Curves

Figure 7 and Table 3 display the ROC (Receiver Operating Characteristic) curve, which evaluates the model's performance in multi-class classification by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR). Each of the curves represents performance according to a class, while the Area Under the Curve (AUC) score is the metric for the model's ability to effectively distinguish between attack types.

Table 3: ROC Curve (AUC scores for each class)

Class	AUC Score
Benign	0.99
Bot	1.00
BruteForce	1.00
DDoS	1.00
DoS	1.00
Infiltration	0.98
Web Attacks	0.99

A curve that is closer to the top left is a better classification. The model achieves high AUC scores for all classes, with most scores close to or at 1.00, and the remaining scores close to 0.99 or 0.98, indicating nearly perfect classification. The other line on the plot is a dashed diagonal corresponding to random guessing (AUC = 0.5). That all curves are well above this line confirms that the model had good predictive power in properly detecting several cyber threats.

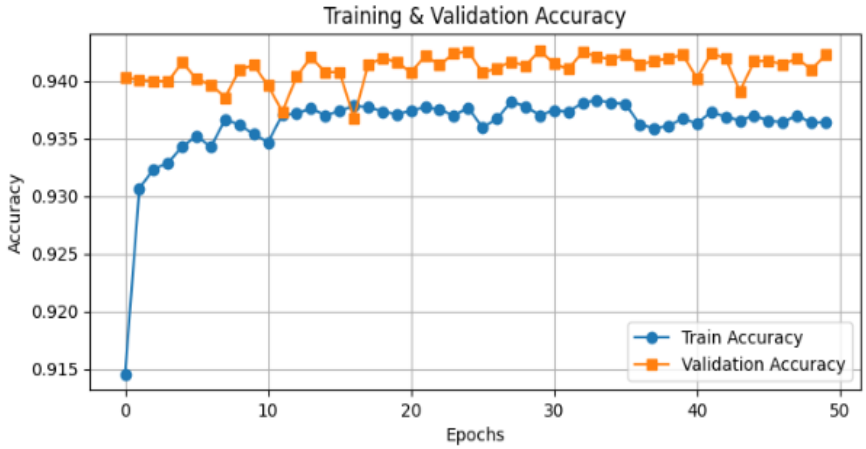

Figure 8: Training and Validation accuracy

The changes in training and validation accuracy of the model are represented by Figure 8 and Table 4 over fifty epochs. The blue line indicates the accuracy in training and starts with an approximate value of 91.5%. It gradually increases and eventually stabilises at around 93.5%.

Table 4: Training and validation accuracy

Epochs	Training Accuracy	Validation Accuracy
0	0.915	0.935
10	0.930	0.940
20	0.935	0.941
30	0.934	0.940
40	0.936	0.941
50	0.937	0.942

Similarly, the orange line indicates validation accuracy, which consistently remains higher than training accuracy and should be around 94% throughout the training. Judging from the closeness of both lines, there is evidence of learning well without major overfitting. However, the small spikes and dips in validation accuracy indicate that there is slightly less consistency in the model's performance across validation batches. Overall, the model seems to generalise quite well.

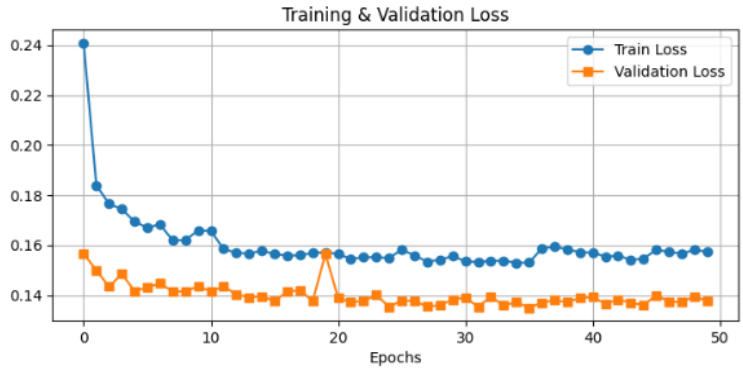

Figure 9: Training and Validation Loss

Figure 9 and Table 5 delineate the training and validation losses of a machine learning model across a span of 50 epochs. The training loss is represented by a blue line, which initially ranges around 0.24 but dips sharply, ultimately settling at approximately 0.16. Its validation loss, as shown by the orange line, on the other hand, is deceptively lower than its training loss and remains relatively stable, hovering around 0.14.

Table 5: Training and Validation Loss

Epochs	Training Loss	Validation Loss
0	0.24	0.16
10	0.18	0.15
20	0.16	0.14
30	0.16	0.14
40	0.16	0.14
50	0.16	0.14

Deviating slightly upward around the 20th epoch yields a minor spike, suggesting a temporary scoliosis but reverberating the general downward movement of loss, indicating learning efficacy. However, as both the training and validation losses are relatively close in value, the model can be inferred to generalise reasonably well without overfitting significantly.

6.1. Flask

Figure 10 illustrates two separate options identified in this section of the web application for users to enter their data. Users can upload a CSV file containing multiple entries by simply selecting the file and pressing the “Upload & Predict” button.

Figure 10: Receiving input from the user flask application

Alternatively, they can enter manual data by filling in the required numerical values in the available input fields and clicking the "Predict" button. With a clean and user-friendly interface, it becomes simple for users across the board to submit data for analysis, making batch processing via CSV and individual entry prediction both parts of a seamless system.

Prediction Results

Row	Predicted Attack
1	Infilteration
2	DoS
3	Infilteration
4	Benign
5	Infilteration
6	Infilteration
7	Infilteration
8	Infilteration
9	Infilteration
10	Benign
Manual Input	Bot

Figure 11: Results of the flask application

Figure 11 illustrates the results from the Intrusion Detection System, showing the predicted attack type for each input row. Each entry in a row represents a distinct data instance extracted from the uploaded CSV file, corresponding to the attack type predicted by the model. Different attack predictions are included here, such as "Infiltration", "DoS", and "Benign", for normal traffic. The last row contains "Manual Input", showing the result for the values the user entered manually; in this case, a "Bot" attack is classified. The result table provides a user-friendly way to visualise the model's predictions and identify potential threats.

7. Conclusion

The proposed approach for the Cyber Intrusion Detection System combines Tabnet with Deep Neural Networks (DNNs) to enhance network security. By using Tabnet's sequential attention mechanism, the model effectively extracts the important features, which improves the Interpretability and performance of the system. The extracted features are then processed by a Deep Neural Network, which is good at learning complex patterns in high-dimensional network traffic data. The proposed approach also comprises a frontend web application built using the Python Flask framework, which enhances usability by allowing users to input custom feature values. Additionally, it enables users to upload a CSV file containing values for the features used in predicting the type of cyber-attack. The model is evaluated on the CICIDS 2018 dataset, and the proposed model achieved an accuracy of 94.22%, outperforming traditional learning models. The proposed approach shows significant improvements in intrusion detection system accuracy and Interpretability, making it a promising solution for predicting the

kind of cyberattacks. Future works may include the integration of additional deep learning techniques and real-time adaptation to evolving cyber threats.

Acknowledgment: The authors gratefully acknowledge the support of their institutions for their academic and infrastructural needs. We appreciate the valuable guidance and encouragement provided by the faculty and staff. Their contributions played a vital role in the successful completion of this research work.

Data Availability Statement: The study utilizes data generated during the development and testing of a Cybersecurity Threat Monitoring System with Integrated Real-Time Alerts and Web-Based Dashboard for Network Protection.

Funding Statement: This research was carried out without any external financial assistance or sponsorship.

Conflicts of Interest Statement: The authors declare no conflicts of interest related to this work. All referenced materials are duly cited.

Ethics and Consent Statement: The study adhered to ethical standards, with informed consent obtained from all participants and institutional approval granted prior to commencement.

References

1. M. A. Siddiqi and W. Pak, "Tier-based optimization for synthesized network intrusion detection system," *IEEE Access*, vol. 10, no. 10, pp. 108530–108544, 2022.
2. J. Zuniga-Mejia, R. Villalpando-Hernandez, C. Vargas-Rosales, and A. Spanias, "A linear systems perspective on intrusion detection for routing in reconfigurable wireless networks," *IEEE Access*, vol. 7, no. 5, pp. 60486–60500, 2019.
3. C. Park, J. Lee, Y. Kim, J.-G. Park, H. Kim, and D. Hong, "An enhanced AI-based network intrusion detection system using generative adversarial networks," *IEEE Internet Things J*., vol. 10, no. 3, pp. 2330–2345, 2023.
4. R. Ben Said, Z. Sabir, and I. Askerzade, "CNN-BiLSTM: A Hybrid Deep Learning Approach for Network Intrusion Detection System in Software-Defined Networking With Hybrid Feature Selection," *IEEE Access*, vol. 11, no. 12, pp. 138732–138747, 2023.
5. K. Dietz, M. Mühlhauser, J. Kögel, S. Schwinger, M. Sichermann, and M. Seufert, "The missing link in network intrusion detection: Taking AI/ML research efforts to users," *IEEE Access*, vol. 12, no. 5, pp. 79815–79837, 2024.
6. E.-U.-H. Qazi, T. Zia, M. Hamza Faheem, K. Shahzad, M. Imran, and Z. Ahmed, "Zero-touch network security (ZTNS): A network intrusion detection system based on deep learning," *IEEE Access*, vol. 12, no. 9, pp. 141625–141638, 2024.
7. A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "CNN-LSTM: Hybrid deep neural network for network intrusion detection system," *IEEE Access*, vol. 10, no. 9, pp. 99837–99849, 2022.
8. T. Wisanwanichthan and M. Thammawichai, "A double-layered hybrid approach for network intrusion detection system using combined naive Bayes and SVM," *IEEE Access*, vol. 9, no. 10, pp. 138432–138450, 2021.
9. I. Dutt, S. Borah, and I. K. Maitra, "Immune system based intrusion detection system (IS-IDS): A proposed model," *IEEE Access*, vol. 8, no. 2, pp. 34929–34941, 2020.
10. T. Kim and W. Pak, "Early detection of network intrusions using a GAN-based one-class classifier," *IEEE Access*, vol. 10, no. 11, pp. 119357–119367, 2022.
11. H. Yang and F. Wang, "Wireless network intrusion detection based on improved convolutional neural network," *IEEE Access*, vol. 7, no. 5, pp. 64366–64374, 2019.
12. J. Saikam and K. Ch, "EESNN: Hybrid deep learning empowered spatial–temporal features for network intrusion detection system," *IEEE Access*, vol. 12, no.1, pp. 15930–15945, 2024.
13. L. Yang, J. Li, L. Yin, Z. Sun, Y. Zhao, and Z. Li, "Real-time intrusion detection in wireless network: A deep learning-based intelligent mechanism," *IEEE Access*, vol. 8, no. 8, pp. 170128–170139, 2020.
14. S. Wali, Y. A. Farrukh, I. Khan, and N. D. Bastian, "Meta: Toward a unified, multimodal dataset for network intrusion detection systems," *IEEE Data Descr*., vol. 1, no. 10, pp. 50–57, 2024.