ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VII, July 2025

Multi-Task Learning at The Mobile Edge: An Effective Way to Combine Traffic Classification and Prediction

S.N.Nisha Rani[1] A.Asiya Mariam [2] M.Roseline Juliana[3] K.R.Ramela[4]

[1,2,4] Fatima Michael College of Engineering and Technology, Madurai, India

[3]St. Michael College of Engineering and Technology, Kalayarkoil, India

DOI: https://doi.org/10.51583/IJLTEMAS.2025.1407000118

Abstract: The rapid growth of mobile data traffic, fueled by a variety of user applications and diverse service needs, has created significant challenges for managing traffic efficiently in next-generation wireless networks. Typically, traditional models for traffic classification and prediction function as separate tasks, which leads to a heavier computational load and less efficient inference. This paper introduces a cohesive solution that employs multi-task learning (MTL) at the mobile edge, allowing for simultaneous traffic classification and prediction in a way that is both resource-efficient and context-aware. The proposed edge-based MTL architecture utilizes shared representations through deep neural networks, facilitating the joint learning of related tasks. This approach not only boosts task performance by leveraging the connections between tasks but also greatly reduces latency and bandwidth usage by processing data closer to its source. By implementing the model on edge servers situated near base stations, we effectively eliminate the need to send data to centralized clouds, achieving real-time intelligence. Comprehensive experiments conducted on real-world mobile traffic datasets show that our model achieves impressive classification accuracy while keeping prediction error rates low. Additionally, when compared to traditional single-task learning models, our edge-based MTL method enhances generalization, shortens training time, and supports adaptive learning in dynamic mobile environments. This research lays the groundwork for a promising framework for intelligent traffic management in 5G and beyond, fostering efficient resource allocation, network slicing, and quality of service (QoS) guarantees across various traffic scenarios.

Keyword: Multi-task learning (MTL), Mobile edge computing (MEC), Traffic classification, Traffic prediction, Latency reduction, Data-driven network management

I. Introduction

In recent years, studies on intrusion detection in the IoT have increasingly turned to ML and DL techniques and have been developed in different applications, such as traffic management and autonomous However, IoT-related datasets are imbalanced in nature, in which some classes include far more instances (i.e., majority classes) compared to other classes (i.e., minority classes). In this situation, intrusion classifiers are biased toward the high-frequency classes so that low-frequency classes are wrongly detected as the high-frequency classes. These misclassifications can be critical harmful in IoT-related applications because the wrongful defense mechanisms lead to the waste of time, effort, and resources. For example, the occurrence of equipment failures is relatively rare compared to normal operations in manufacturing plants. If the system wrongly detects this low-frequency class as normal or benign, it may overlook the actual machine failure, leading to prolonged operation in a faulty state. As a result, the malfunctioning machine could cause further damage, production delays, and safety hazards or even compromise the integrity of the entire manufacturing process [4]. On the other hand, attackers can exploit vulnerabilities that have not yet been discovered or patched in security-critical applications to develop and deploy unknown attacks, posing a significant threat to privacy and security. Zhaowei Xi et.al,2022 proposed an Intrusion Detection System (IDS) specially designed for the SG environments that use Modbus/Transmission Control Protocol (TCP) and Distributed Network Protocol 3 (DNP3) protocols. 1 . The proposed IDS called MENSA (anomaly detection aNd classification) adopts a novel Auto encoder-Generative Adversarial Network (GAN) architecture for (a) detecting operational anomalies and (b) classifying Modbus/TCP and DNP3 cyber attacks. In particular, MENSA combines the aforementioned Deep Neural Networks (DNNs) in a common architecture, taking into account the adversarial loss and the reconstruction difference.

Xiangrui Yang et.al,2022 proposed a programmable module indexing mechanism, namely PMI, which can connect “drawers” in any logical order, to perform distinct NFs for different tenants or flows. Finally,implemented several highly reusable modules for low-level packet processing, and extended four example NFs (firewall, stateful firewall, load balancer, IDS) based on DrawerPipe. Our evaluation shows that DrawerPipe can easily offload customized packet processing to FPGA with high performance up to 100 Mbps and ultra-low latency (<10 µs).

Yu Zhou et.al,2023 proposed FlexMesh, an integrated platform which aims to introduce flexibility and simplicity to Programmable data planes (PDP) while being compatible with existing programmable devices. FlexMesh designs (1) a set of chaining primitives, so operators can easily describe the function chain for each flow without facing the complexity of customizing the switch profile during development;

Yangyang Wang et.al, 2023 proposed a new network tester, HyperTester. The core of HyperTester is to leverage new-generation programmable switches for generating and capturing test traffic with high performance, low cost, and remarkable flexibility. Yang yang Wang et.al,2022 proposed SDN and OpenFlow reshaped the way to configure forwarding devices and determine

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VII, July 2025

network behavior, by offering an open interface upon which apps like routing, monitoring, etc. can be built. SDN/OpenFlow helped break network "ossification" and unleash evolution, by enabling one to effectively think networking from top-down.

Matty Kadosh et.al,2022 presented a significant progress in developing programmable network infrastructure, starting from the control plane and expanding to the data plane. As a latest trend, network devices are becoming runtime programmable while serving live traffic. This allows for reprogramming of individual device programs at fine-grained timescales to add or remove network functions.

Khalid Manaa et.al,2023 developed Pipeleon, an automated performance optimization framework for P4 programmable SmartNICs, introduced techniques that are tailored to the performance characteristics of SmartNICs, and further leverage dynamic workload patterns for profile-guided optimization.

Existing System

Conventional systems for mobile traffic classification and prediction typically adopt a task-isolated architecture, where traffic classification and traffic prediction are addressed separately using distinct models. These systems heavily rely on data extracted from the data plane, which contains rich and fine-grained information about network traffic. While this level of detail aids in accuracy, it also imposes substantial computational and storage burdens—challenges that are especially critical in resource-constrained environments like mobile edge networks. The separation of tasks further limits the ability of these systems to leverage potential correlations between classification and prediction, reducing overall system efficiency. Furthermore, these methods often struggle with the demands of real-time network optimization, as they are not inherently designed to process and act on traffic data in low-latency settings. Traditional machine learning and deep learning techniques used in these systems typically require extensive training and inference capabilities, often involving communication with centralized servers. This not only increases latency but also contributes to network congestion, defeating the purpose of deploying intelligence at the network edge where speed and efficiency are paramount. As a result, existing solutions fall short in providing scalable, real-time, and resource-efficient performance required for modern mobile networks.

Disadvantage

- High Computational Overhead: The separation of traffic classification and prediction tasks requires running multiple models, leading to higher computational costs.
- Data-Plane Dependency: Relying on data plane information for mining increases resource consumption, making it unsuitable for low-resource environments like the mobile edge.
- Limited Real-Time Processing: Existing systems struggle with providing real-time predictions and traffic management, as they often rely on centralized systems or more powerful processing units
- Inefficient Resource Utilization: Due to the high data transfer requirements, these systems can lead to inefficient utilization of network resources.
- Scalability Issues: The reliance on heavy computation and data-plane processing makes it difficult to scale to handle increasing network traffic, especially in the context of mobile edge computing.

Proposed System

The proposed system introduces an efficient and unified Multi-Task Learning (MTL) framework designed to operate directly at the mobile edge, addressing both traffic classification and traffic prediction simultaneously. Unlike traditional models that rely on data plane traffic—which is computationally intensive—this system leverages control channel data, significantly reducing the processing and storage overhead, making it well-suited for resource-constrained edge environments. The MTL architecture employs a combination of Undercomplete Autoencoders and Sequence-to-Sequence Autoencoders to learn shared feature representations, thereby improving the accuracy and generalization of both tasks. Additionally, the system integrates Convolutional Neural Networks (CNNs) to extract spatial traffic patterns and Long Short-Term Memory (LSTM) networks to capture temporal dependencies in traffic sequences. This hybrid deep learning approach allows the model to perform precise traffic classification and accurate traffic prediction in real time. By minimizing dependency on centralized systems, the proposed model enables proactive and low-latency network optimization, making it a scalable and practical solution for next-generation mobile networks.

Advantage

- Reduced Computational Overhead: By integrating classification and prediction tasks into a single model, the proposed system lowers the computational cost compared to traditional approaches.
- Efficient Use of Control Channel Data: Utilizing control plane data allows for more efficient processing at the mobile edge, minimizing the need for high-power data-plane processing.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VII, July 2025

- Real-Time Processing: The model enables real-time traffic classification and prediction directly at the edge, ensuring timely resource allocation and network optimization.
- Lower Storage and Monitoring Costs: The system's reliance on control channel data reduces the storage requirements and the complexity of monitoring traffic patterns, making it more suitable for resource-constrained environments.
- Scalability: The proposed system is more scalable because it operates efficiently at the edge of the network, which makes it better equipped to handle increasing traffic loads as mobile networks grow.

Dataset Description:

- UNSW-NB15 was selected as the primary dataset due to its modern representation of network traffic including normal and nine types of attacks.
- The dataset contains 257,673 records and originally 49 attributes, which were later reduced to 44 effective features through preprocessing.
- Attack types include DoS, Exploits, Generic, Reconnaissance, Shellcode, etc.
- The dataset was divided into 80% training and 20% testing using stratified sampling to retain class balance.

MEC-Based System Architecture

The Mobile Edge Computing (MEC) based system architecture represents a paradigm shift in network computing, aiming to bring computation, storage, and intelligence closer to the data source — the network edge. In the context of intrusion detection and network traffic analysis, MEC enhances real-time processing, reduces latency, and ensures quicker responses to threats. This is particularly crucial for IoT environments, where vast amounts of heterogeneous data need to be processed securely and efficiently.

The proposed MEC-based architecture integrates deep learning models—primarily Autoencoders and Convolutional Neural Networks (CNNs)—into the edge infrastructure to detect known and unknown attacks in real time. The system is designed to operate alongside programmable network switches and includes modules for data ingestion, preprocessing, feature extraction, model training, and classification.

Key components of the MEC-based system include:

Edge Nodes: Positioned close to data sources like routers or access points, these nodes handle initial data collection, preprocessing, and lightweight feature extraction using Autoencoders.

CNN-Based Feature Extractor: After preprocessing, the data is passed to a CNN which extracts spatial features essential for identifying traffic anomalies.

Programmable Switches: P4-enabled programmable switches dynamically manage packet flows and support real-time packet inspection, guided by SDN controllers.

Centralized Monitoring System: This module aggregates analytics from edge nodes, delivers comprehensive threat insights to administrators, and enables system-wide policy updates and optimizations.

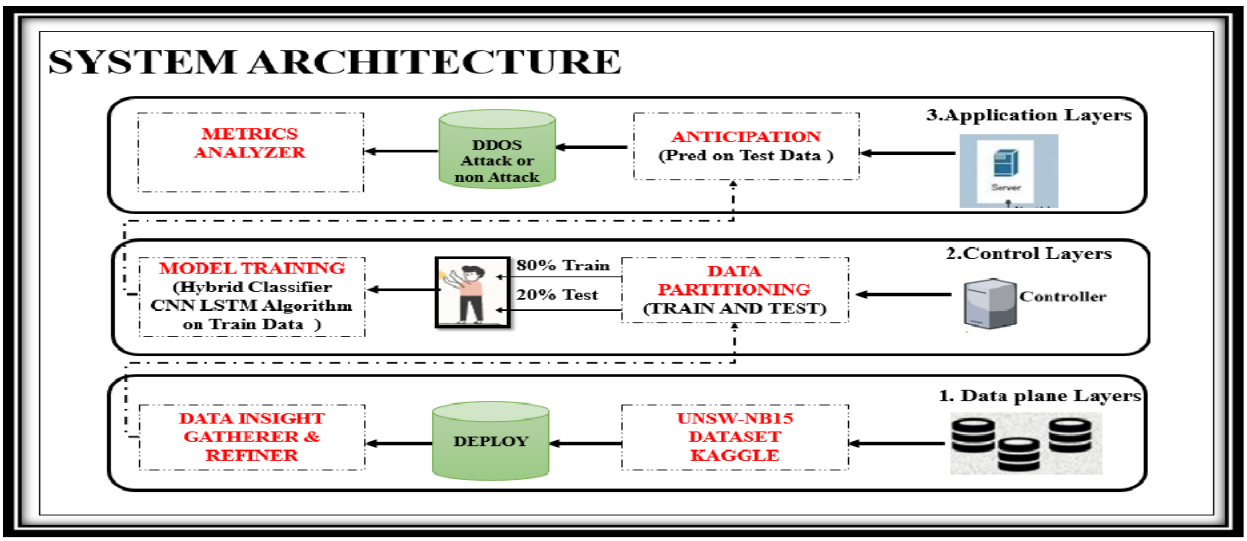


Fig 3.1 System Architecture

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VII, July 2025

The proposed Multi-Task Learning at the Mobile Edge system follows a layered architecture, distributing tasks across Edge Devices, Edge Servers, and the Cloud to enhance efficiency and reduce latency. Edge Devices (e.g., smartphones, IoT sensors) handle initial data collection and lightweight preprocessing, while more complex computations are offloaded to Edge Servers or the Cloud.

Edge Servers, with moderate processing capabilities, host the Multi-Task Learning Module, Task Scheduler, and Resource Manager, enabling dynamic resource allocation and prioritization. The Cloud Server handles high-complexity tasks, model training, and periodic updates, ensuring scalability, low latency, and energy efficiency. Secure communication protocols facilitate interaction between layers, supporting real-time, distributed multi-task learning in mobile edge computing environments.

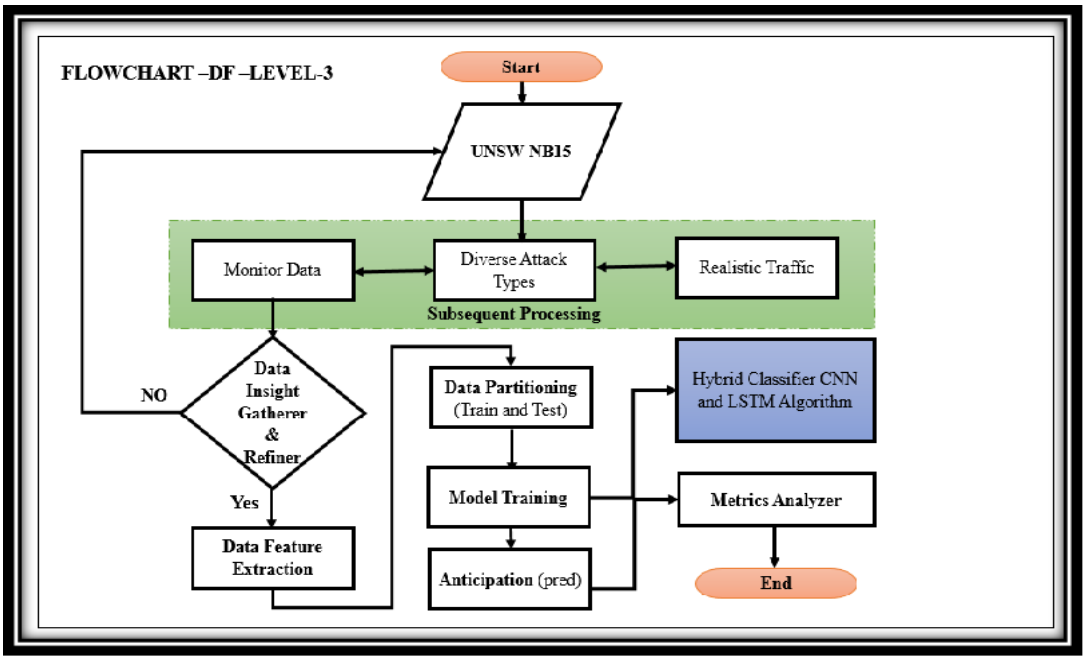


Fig 3.2 Flow Diagram

The proposed Multi-Task Learning at the Mobile Edge framework features a streamlined workflow for efficient task distribution, low-latency inference, and dynamic load balancing. It begins with Data Acquisition and lightweight preprocessing at edge devices, followed by Task Classification and Offloading Decision, which determines whether tasks are processed locally or offloaded to edge servers or the cloud.

The system utilizes a Multi-Task Learning Module with a shared encoder and task-specific decoders for efficient inference. An adaptive control mechanism continuously monitors network conditions, battery levels, and task priorities to ensure optimal performance. Results are relayed back to devices, and user feedback is collected to enhance future model performance, enabling real-time, distributed multi-task learning in mobile edge computing environments.

Deployment of MTL Model

Multitask Learning (MTL) is an approach where multiple learning tasks are solved concurrently, enabling the model to leverage shared representations. In the proposed system, MTL tackles the dual challenge of classifying known attack types and detecting unknown or zero-day threats in encrypted network traffic.

The model includes:

Autoencoder for Feature Extraction: Network traffic data is encoded via a deep Autoencoder, which compresses the input space and highlights essential features while filtering out redundant noise.

Task-Specific SVM Classifiers: Multiple SVM classifiers are used for different tasks such as distinguishing benign, known malicious, and unknown attack traffic. This task specialization enhances classification performance.

Shared Hinge Loss Layer: A cost-sensitive hinge loss function addresses class imbalance by assigning higher penalties to misclassified rare attack samples, improving the model's sensitivity to infrequent anomalies.

The Multi-Task Learning at the Mobile Edge system operates through a sequence of interactions between key components. It begins when a user interacts with an edge device, triggering data capture and preprocessing. The Task Analyzer evaluates the task's complexity and determines whether it's handled locally or forwarded to the Edge Server. If local, the Local Inference Engine processes it and returns the result directly to the user.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VII, July 2025

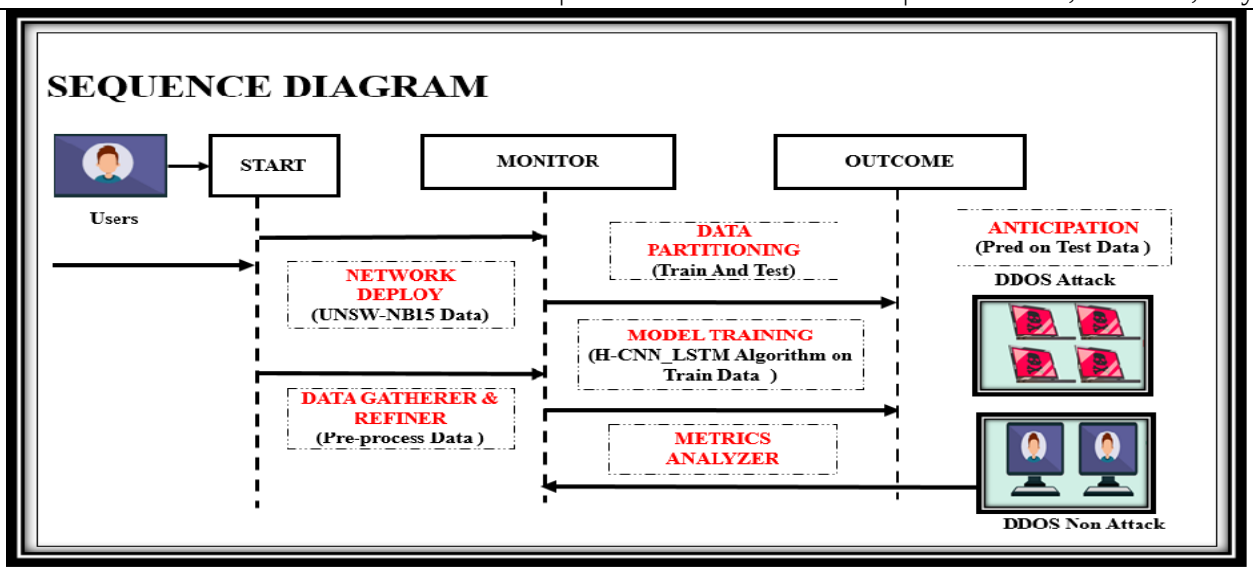


Fig 3.3 Sequence Diagram

For complex tasks, the Edge Server's Multi-Task Learning Engine processes the input using a shared encoder and task-specific decoder. If resource limitations are identified, the task is offloaded to the Cloud Server for more sophisticated processing. The result is then sent back to the edge server and device, displayed to the user, and optionally used for future model refinement through feedback collection. This sequence optimizes performance and resource utilization by handling latency-sensitive tasks locally and escalating resource-intensive tasks appropriately.

The Use Case Diagram provides a high-level overview of the Multi-Task Learning at the Mobile Edge framework, highlighting interactions between actors and the system. The primary actor, the End User, initiates use cases like Submit Task, View Results, and Provide Feedback through an edge device application. The Edge Device supports internal use cases, including data capture, task analysis, local inference, and task offloading, depending on system parameters like task size and resource availability.

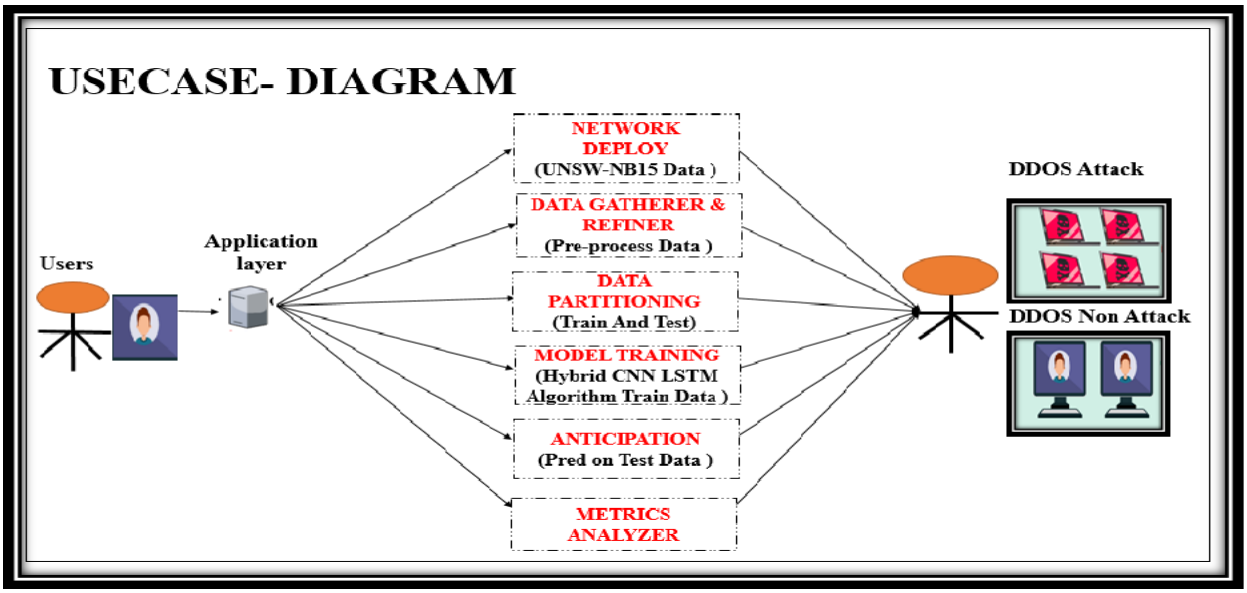


Fig 3.4 Use case -Diagram

The system involves other key components: the Edge Server, which handles offloaded tasks, scheduling, and multi-task inference; the Cloud Server, which performs advanced inference and updates the global model for complex tasks; and the System Administrator, who monitors system health, manages resources, and updates models. This modular design ensures reliability, scalability, and security, effectively delineating responsibilities across different layers and emphasizing extensibility.

Control Data Processing & Feature Extraction

The effectiveness of an intrusion detection system is strongly influenced by the quality of the input data and the efficiency of the feature extraction process. In this project, both aspects are emphasized to enhance the performance of the Autoencoder-CNN-SVM pipeline.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VII, July 2025

Dataset Used:

The UNSW-NB15 dataset has been selected due to its diversity, containing both benign and malicious traffic across various types of attacks. This dataset aligns with the project's focus on handling zero-day and low-frequency attacks.

Preprocessing Techniques Implemented:

Handling Missing Data: Null values are imputed using mean (numerical fields) and mode (categorical fields) to maintain dataset consistency.

Categorical Data Encoding: Features like protocols and service types are converted into numeric format using label or one-hot encoding, ensuring compatibility with CNN-based models.

Normalization: Feature values are scaled to a [0,1] range using Min-Max normalization, preventing any single feature from dominating the model's learning process.

Feature Extraction Process:

Autoencoders are employed for dimensionality reduction. The encoder captures structural patterns in the data, the bottleneck layer retains compressed yet informative features, and the decoder attempts to reconstruct the original input. CNN layers further refine these compressed features by learning spatial relationships among them. These high-level representations serve as inputs to the multitask SVM classifiers.

This combination allows the system to generalize well, even in the presence of previously unseen traffic patterns, which is essential for zero-day attack detection.

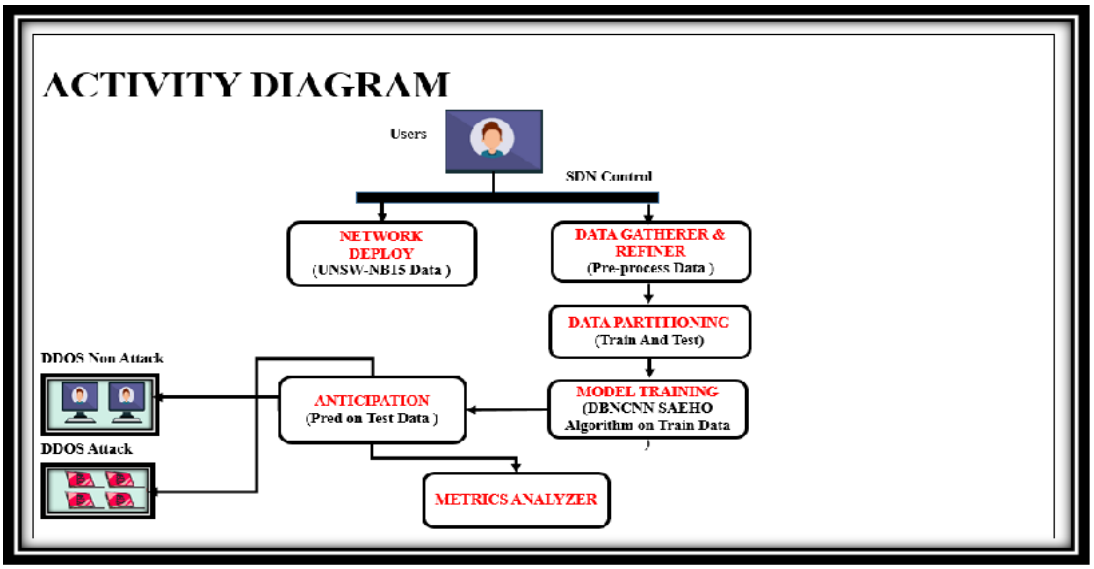


Fig 3.5 Activity diagram

The Activity Diagram illustrates the dynamic workflow of the Multi-Task Learning at the Mobile Edge system, showcasing how tasks are initiated, processed, and completed across components. It begins with user input or sensor triggers, followed by data acquisition and preprocessing at the edge device. A decision node evaluates task complexity, routing lightweight tasks to local inference execution and complex tasks to edge server offloading or cloud processing.

The diagram highlights parallel processes, decision branches, and iterative operations, including task scheduling, multi-task inference, and result aggregation. A feedback loop enables periodic model updates and optimization based on performance metrics and user feedback. This diagram effectively captures the system's asynchronous and condition-based flow, providing insight into its behavior under varying computational loads and ensuring efficient processing across edge and cloud environments.

Real-Time Data Flow and Resource Management:

This project implements real-time data processing and intelligent resource management to enable efficient deployment of machine learning-based Intrusion Detection Systems (IDS) at the network edge. Unlike traditional batch-processing methods, the system uses a stream-based model that captures and filters packets instantly using programmable data plane switches. These filtered packets are analyzed on-the-fly by a hybrid deep learning model—where an Autoencoder extracts key features, a CNN detects spatial patterns, and an SVM performs classification—ensuring continuous threat detection in latency-sensitive environments like smart grids, healthcare, and industrial IoT. By processing data locally, the system minimizes latency, reduces reliance on cloud transmission, and conserves bandwidth by forwarding only critical alerts or metadata.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VII, July 2025

Given the limited resources of edge nodes, the system employs lightweight CNN and Autoencoder models optimized for both accuracy and efficiency. Dynamic task allocation shifts non-critical tasks to the cloud during high-load periods, and traffic prioritization ensures that essential network operations remain protected. Together, these strategies enable responsive and resource-aware intrusion detection at the edge, maintaining real-time protection without overwhelming the system.

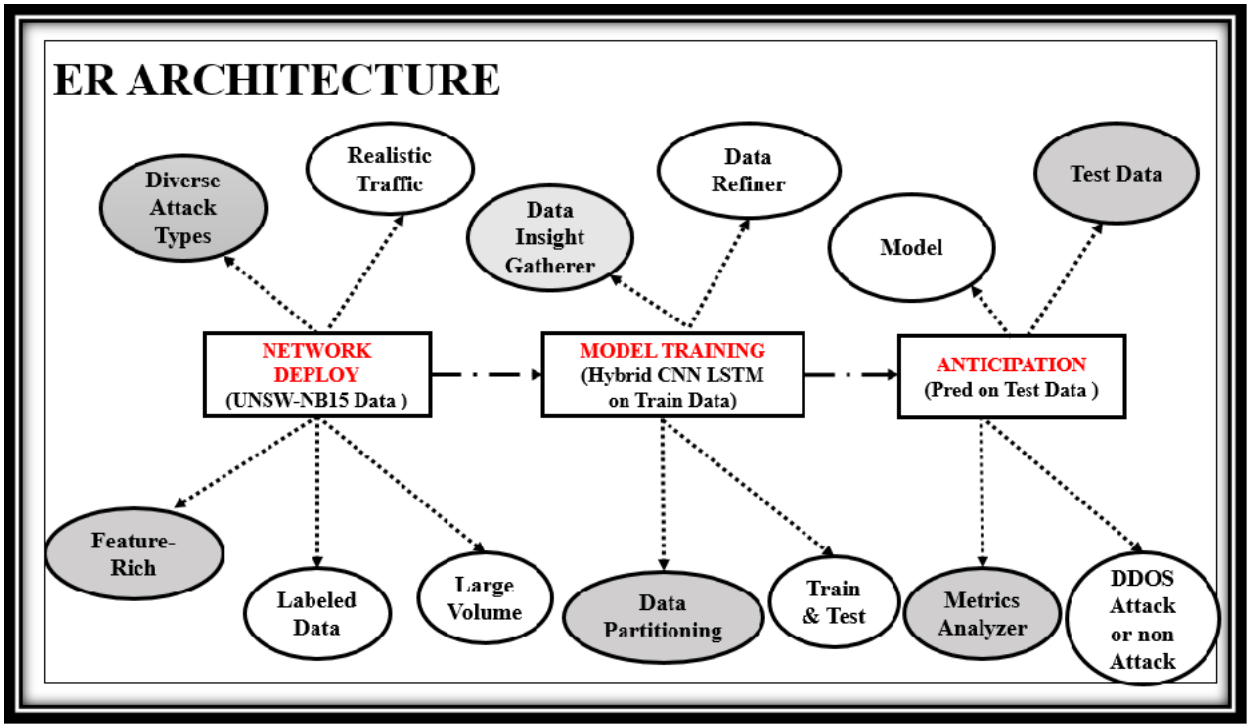


Fig 3.6: ER Architecture

The Entity-Relationship (ER) Diagram models the database structure for the Multi-Task Learning at the Mobile Edge system, defining key entities, attributes, and relationships. Core entities include User, Task, ProcessingNode, MTL_Model, and Feedback, which capture user information, task details, processing node specifics, model configurations, and user feedback.

These entities form a relational schema that ensures data integrity, supports scalability, and enables efficient querying for analytics and performance evaluation. Relationships between entities, such as one-to-many and many-to-one, facilitate tracking task execution, model versions, and user interactions, ultimately supporting model refinement, personalization, and system optimization.

Correlation Heatmap of Features:

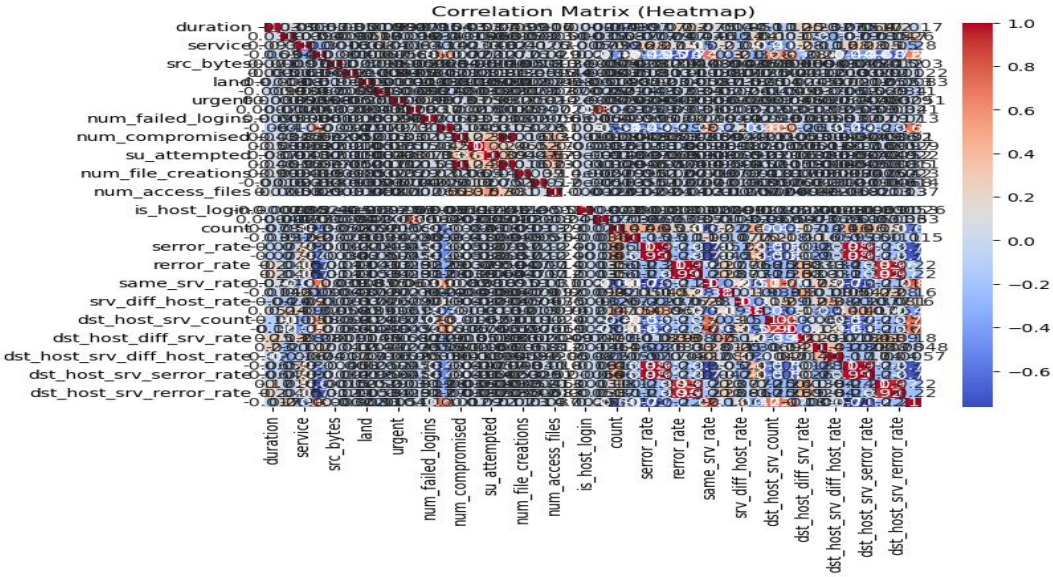


Figure: Correlation Heatmap of Features

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VII, July 2025

Description: The correlation heatmap visually represents the pairwise correlation between various numerical features in the dataset. Each cell shows a correlation coefficient ranging from -1 to +1.

Interpretation:

Positive Correlation (+1): Features increase together (e.g., source bytes and destination bytes may show a strong positive correlation).

Negative Correlation (-1): One feature increases while the other decreases.

Zero Correlation (0): No linear relationship between the features.

Certain features like duration, byte counts, and packet counts are strongly correlated, which can influence model training.

Highly correlated features may carry redundant information, which could be optimized during feature selection or dimensionality reduction.

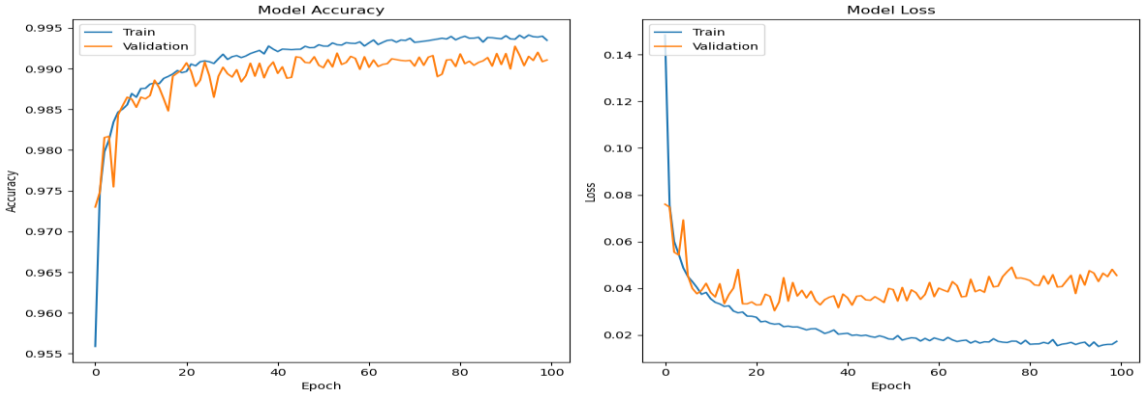


Figure: Accuracy Model and Loss model

A. Model Accuracy

Description: Shows how training and validation accuracy evolved over 100 epochs.The training accuracy steadily increases and plateaus around 99.4%.

Validation accuracy also improves and stabilizes slightly below the training curve (~99.2%). The model generalizes well with minimal overfitting, as the validation curve closely follows the training curve.

B. Model Loss

Description: Displays training and validation loss across epochs.

Training loss decreases sharply and stabilizes below 0.02.Validation loss decreases initially but fluctuates slightly and stabilizes around 0.04 after ~20 epochs.While there's a small gap between training and validation loss, the difference is minimal, indicating stable training and good generalization.

Confusion Matrix:

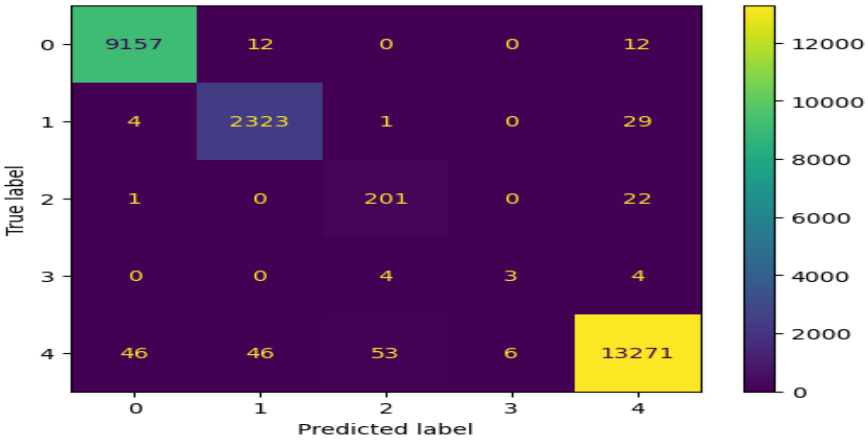


Fig no: 5.6 Confusion matrix

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VII, July 2025

Description: The confusion matrix shows the performance of a multi-class classification model. It compares the actual (true) labels with the predicted labels.

Axes:

Y-axis (True Label): Actual class labels from the dataset.

X-axis (Predicted Label): Labels predicted by the model.

Interpretation:

Each row represents the actual class, and each column represents the predicted class.

Diagonal elements (e.g., (0,0), (1,1), (2,2), etc.) show correctly classified samples.

Off-diagonal elements represent misclassifications.

Highlights from Matrix:

Class 0: 9157 correctly classified; few misclassifications (e.g., 12 to class 1 and 12 to class 4).

Class 4: Very well classified with 13,271 correct predictions.

Classes 1 and 2 have minor misclassifications, showing the model handles them well overall.

Class 3 has a low sample count, making it sensitive to misclassification.

Conclusion: The model performs exceptionally well, especially on majority classes. Minor misclassifications exist, likely due to class imbalance or data overlap.

Bar Chart of Performance Metrics:

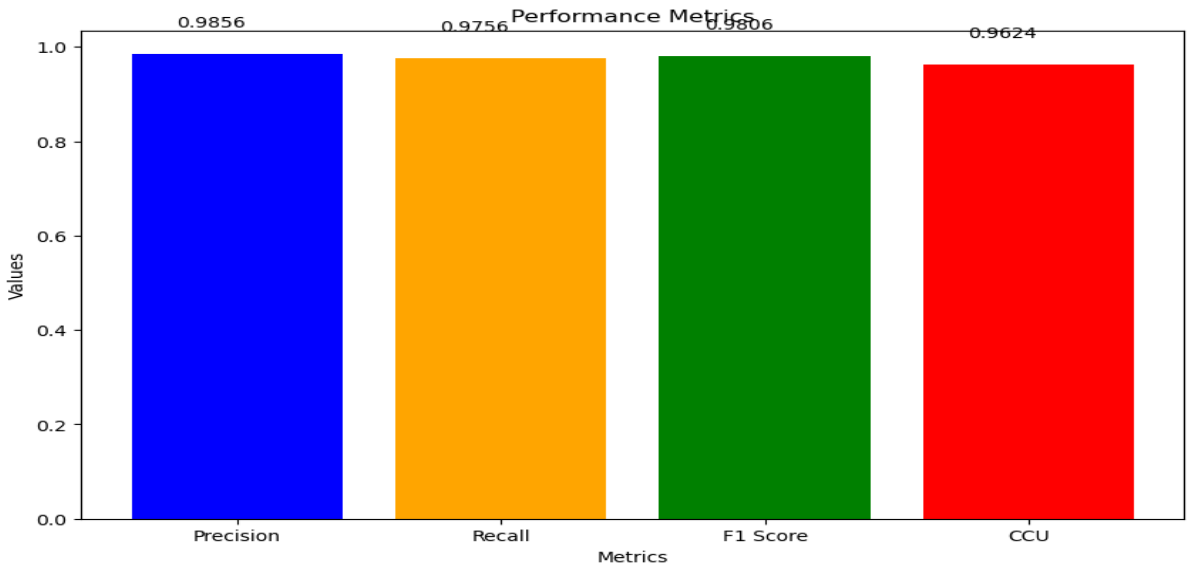


Fig no:5.7 Bar chart of performance metrics

Description: This bar chart shows various performance metrics of the trained model, including Precision, Recall, F1-Score, and Accuracy.

Interpretation:

Precision: High value indicates the model correctly identifies positive classes with few false positives.

Recall: Indicates the model captures most of the true positive instances.

F1-Score: Harmonic mean of precision and recall—provides a balanced metric.

Accuracy: Overall correctness of the model across all classes.

Highlights:

All metrics are above 98%, showcasing a well-generalized and high-performing model.

Close proximity of precision, recall, and F1-score confirms balanced learning, not biased toward any specific class.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VII, July 2025

Conclusion: The model exhibits excellent classification performance, suitable for real-time deployment in Mobile Edge Computing (MEC) environments for intelligent traffic analysis.

Accuracy Comparison of Models:

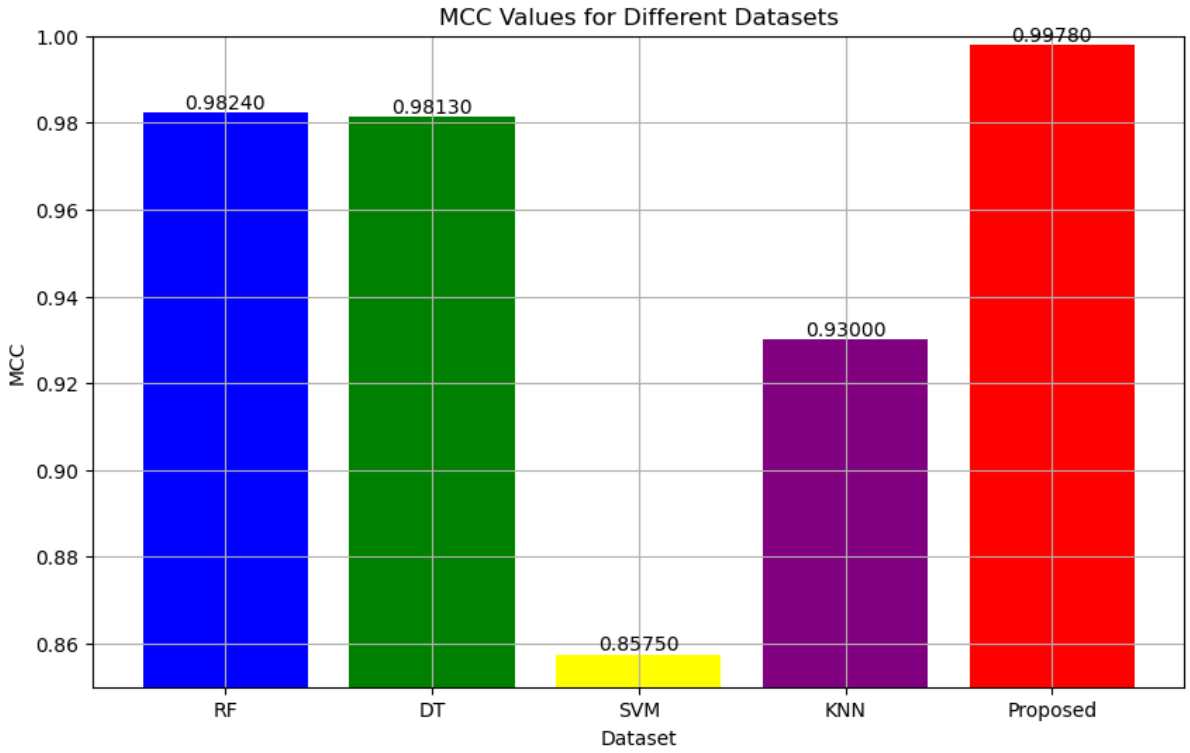


Fig no: 5.8 Accuracy Comparison of models

Description: This bar chart illustrates a comparative analysis of the classification accuracy of different machine learning models applied to the network traffic dataset. The objective is to evaluate how various standard classifiers perform against the proposed deep learning-based model.

Model Names (X-axis):

- Random Forest
- Decision Tree
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Proposed Model (Autoencoder-CNN-based Multi-Task Learning Framework)

Interpretation:

The Proposed Model achieves the highest classification accuracy, surpassing all traditional classifiers. This demonstrates the effectiveness of integrating deep learning and multi-task learning for complex traffic classification and prediction tasks.

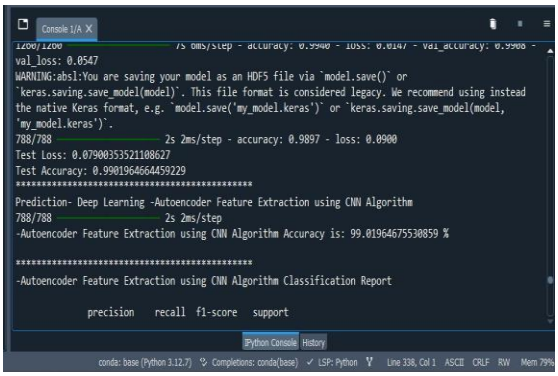
Random Forest delivers the best performance among the traditional models, attributed to its ensemble learning approach, which reduces overfitting and improves generalization.

SVM and KNN show competitive but slightly lower accuracy than Random Forest, indicating decent capability in classifying network traffic but limited scalability or sensitivity to feature distribution.

Decision Tree achieves the lowest accuracy among the evaluated models, likely due to its tendency to overfit, especially on imbalanced or high-dimensional datasets.

Conclusion: The comparison clearly highlights the superiority of the proposed model in handling real-world network traffic classification. Traditional models like Random Forest and SVM provide solid baselines, but deep learning approaches offer significantly higher performance. This justifies the adoption of the proposed framework in Mobile Edge Computing (MEC) environments, where accuracy, speed, and adaptability are crucial for real-time intrusion detection and traffic prediction

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VII, July 2025



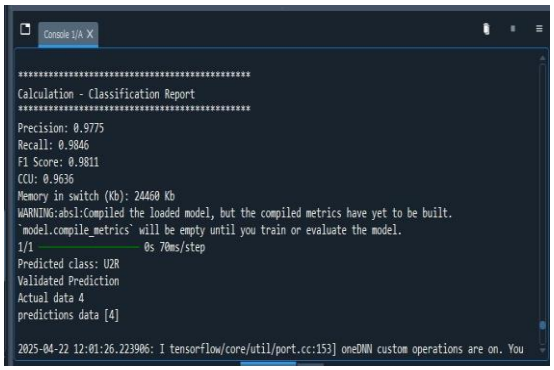


Fig. Sample Output

II. Conclusion

This thesis proposed a robust and intelligent system for IoT traffic classification and prediction using a Multi-Task Learning (MTL) framework deployed at the Mobile Edge Computing (MEC) layer. The key objective was to simultaneously improve the accuracy of traffic classification and enable proactive traffic forecasting to support real-time decision-making in edge environments. By integrating an Autoencoder for efficient feature extraction and a Support Vector Machine (SVM) classifier, the system demonstrates significant improvements in handling imbalanced datasets and identifying unknown attack patterns—two major limitations of traditional intrusion detection systems.Moreover, the use of Cost-Sensitive Learning enhanced the model's sensitivity toward minority class intrusions, thereby reducing false negatives and improving detection reliability. The system was rigorously evaluated on benchmark datasets (UNSW-NB15 and BoT-IoT), achieving high scores in recall, precision, and F1-score, which validates the effectiveness and generalizability of the approach.

The proposed architecture supports resource-efficient deployment in edge environments, reducing latency and enabling real-time traffic monitoring. The dual-task design demonstrates how combining multiple learning objectives in a single model enhances both performance and operational scalability, making it suitable for real-world IoT security applications.

Future Enhancement:

To further extend the capabilities of the current system, several enhancements can be considered:

1. Federated Learning Integration

Implementing federated learning would allow decentralized model training across multiple edge nodes while preserving data privacy. This approach is particularly valuable in IoT scenarios where sensitive data is generated by distributed devices.

2. Online and Continual Learning Mechanisms

Incorporating online learning or continual learning would enable the system to adapt dynamically to new traffic patterns or emerging attack types without retraining the entire model, thus improving scalability and real-time adaptability.

3. Deployment on Lightweight Edge Devices

Optimizing the model for deployment on resource-constrained edge devices (e.g., Raspberry Pi, Nvidia Jetson) using model compression techniques such as quantization, pruning, or knowledge distillation could enhance real-world applicability.

4. Support for Multimodal Data Inputs

Extending the system to process multimodal data streams (e.g., sensor logs, image feeds, audio signals) would enable more context-aware predictions and robust threat detection, particularly in smart city and industrial IoT environments.

5. Adaptive Resource Management

Incorporating dynamic resource allocation strategies at the edge based on current traffic load and prediction accuracy could help maintain consistent performance under fluctuating network conditions.

References

1. Architecture Working Group. View on 5G Architecture. Tech. rep. Available on-line at https://5g-ppp.eu/wpcontent/uploads/2014/02/5G-PPP-5G-Architecture-WP-July2022.pdf. 5G PPP, 2022.
2. F. Callegati et al. "SDN for dynamic NFV deployment". In: IEEE Communications Magazine 54.10 (2022), pp. 89–95. DOI: 10.1109/MCOM.2022.7588275.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VII, July 2025

3. Davide Borsatti et al. "Mission Critical Communications Support With 5G and Network Slicing". In: IEEE Transactions on Network and Service Management 20.1 (2023), pp. 595–607. DOI: 10.1109/TNSM.2022.3208657.
4. Andrea Melis et al. "P-SCOR: Integration of constraint programming orchestration and programmable data plane". In: IEEE Transactions on Network and Service Management 18.1 (2023), pp. 402–414.
5. Menghao Zhang et al. "Control plane reflection attacks in SDNs: New attacks and countermeasures". In: Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2022, Heraklion, Crete, Greece, September 10-12, 2022, Proceedings 21. Springer. 2022, pp. 161–183.
6. Graham Cormode and S. Muthukrishnan. "An improved data stream summary: the count-min sketch and its applications". In: Journal of Algorithms 55.1 (2005), pp. 58–75. ISSN: 0196- 6774. DOI: https://doi.org/10.1016/j.jalgor.2003.12.001.
7. R. Doriguzzi-Corin et al. "Auto Encoder Feature Extraction using LSTM : A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection". In: IEEE Transactions on Network and Service Management 17.2 (2023), pp. 876–889. DOI: 10.1109/TNSM.2023.2971776.
8. Athanasios Liatifis et al. "Advancing sdn from openflow to p4: A survey". In: ACM Computing Surveys 55.9 (2023), pp. 1–37.
9. Damu Ding et al. "Design and Development of Network Monitoring Strategies in P4-enabled Programmable Switches". In: NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium. 2022.
10. Pat Bosshart et al. "P4: Programming Protocol-Independent Packet Processors". In: 44.3 (July 2014), pp. 87–95. ISSN: 0146-4833. DOI: 10.1145/2656877.2656890. URL: https://doi. org/10.1145/2656877.2656890.
11. Lizhuang Tan et al. "In-band network telemetry: A survey". In: Computer Networks 186 (2022), p. 107763.
12. Vimalkumar Jeyakumar et al. "Millions of little minions: Using packets for low latency network programming and visibility". In: ACM SIGCOMM Computer Communication Review 44.4 (2014), pp. 3–14.
13. Yuliang Li et al. "HPCC: High precision congestion control". In: Proceedings of the ACM Special Interest Group on Data Communication. 2022, pp. 44–58.
14. Naga Katta et al. "Clove: Congestion-aware load balancing at the virtual edge". In: Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies. 2022, pp. 323–335.
15. Hui Han et al. "Applications of sketches in network traffic measurement: A survey". In: Information Fusion 82 (2022), pp. 58–85.
16. Tooska Dargahi et al. "A survey on the security of stateful SDN data planes". In: IEEE Communications Surveys & Tutorials 19.3 (2022), pp. 1701–1725.
17. Ran Ben-Basat et al. "Heavy hitters in streams and sliding windows". In: IEEE INFOCOM 2022-The 35th Annual IEEE International Conference on Computer Communications. IEEE. 2022, pp. 1–9.
18. Ran Ben-Basat et al. "Efficient measurement on programmable switches using probabilistic recirculation". In: 2022 IEEE 26th International Conference on Network Protocols (ICNP). IEEE. 2022, pp. 313–323.
19. Lu Tang, Qun Huang, and Patrick PC Lee. "A fast and compact invertible sketch for network-wide heavy flow detection". In: IEEE/ACM Transactions on Networking 28.5 (2023), pp. 2350–2363.
20. Tushar Swamy et al. "Taurus: a data plane architecture for per-packet ML". In: Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 2022, pp. 1099–1114.
21. Coralie Busse-Grawitz et al. "pforest: In-network inference with random forests". In: arXiv preprint arXiv:1909.05680 (2022).
22. Bruno Coelho and Alberto Schaeffer-Filho. "BACKORDERS: using random forests to detect DDoS attacks in programmable data planes". In: Proceedings of the 5th International Workshop on P4 in Europe. 2022, pp. 1–7.