WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Deep Learning for Secure Mobile Edge Computing in Cyber-Physical Transport Systems

Syeda Fariha Fatima, Syeda Munazza Naqvi, Sophia Ali, Tahera Abid

4 – Professor & Head of IT Department, [B.Tech,M.Tech,(Ph.D)]
1 – Department of IT, Nawab Shah Alam Khan College of Engineering & Technology, Affiliated to Osmania University, Hyderabad, India.

A B S T R A C T

The rapid advancement of intelligent transportation systems has brought with it a demand for decentralized and ultra-low-latency computing. Mobile Edge Computing (MEC) meets this demand by placing computation closer to the data source—such as smart vehicles, traffic sensors, and edge nodes—thereby enabling faster decision-making and improved efficiency. However, this shift to distributed infrastructure also introduces significant cybersecurity challenges. Traditional security solutions like firewall rules or signature-based Intrusion Detection Systems (IDS) fall short in this context, particularly when facing zero-day attacks, unknown threats, or high-speed real-time traffic at the edge.

To address these limitations, our project proposes a hybrid deep learning-based security system that leverages two key techniques: (1) an unsupervised Autoencoder for detecting anomalies by learning patterns in normal network traffic and measuring reconstruction error, and (2) a supervised XGBoost classifier to categorize the detected anomalies into specific attack types like DDoS, PortScan, WebAttack, etc. The entire system is implemented using Python and includes a Flask-based REST API to accept live or simulated traffic as 78-dimensional feature vectors. The API processes these inputs and returns JSON-formatted predictions that indicate whether the traffic is malicious or benign.

Moreover, to ensure ease of use and real-time visibility, a user-friendly Streamlit dashboard is integrated to visualize results. The dashboard displays detection statistics, attack probability scores, confidence levels, and supports both automated monitoring and manual traffic analysis. The solution is designed for modular deployment and is fully containerized using Docker, making it platform-independent and scalable across various edge environments, including MEC gateways.

This comprehensive and adaptive system demonstrates high accuracy (>95%) in detecting unknown threats and over 99% classification precision on known attacks, making it an ideal fit for real-time protection in next-generation transportation networks.

Keywords: MEC, Intrusion Detection System (IDS), Autoencoder, Anomaly Detection, XGBoost, Cybersecurity, Docker, Streamlit, Flask API, Real-Time Detection.

1. Introduction:

In recent years, the evolution of **intelligent transportation systems (ITS)** has redefined how vehicles, infrastructure, and communication networks interact. These systems rely on vast amounts of real-time data to make critical decisions that ensure safety, reduce congestion, and optimize resource use. However, as transportation systems become increasingly connected and automated, they are also becoming more **vulnerable to cyber threats**. Attacks on connected vehicles, smart traffic signals, and roadside units can not only disrupt services but may also pose serious risks to human life.

To support the real-time data needs of ITS, **Mobile Edge Computing (MEC)** has emerged as a crucial architectural paradigm. Unlike traditional cloud computing models where data must travel long distances to centralized servers, MEC pushes computation and storage closer to the data source. This **reduces latency**, enhances responsiveness, and makes real-time analytics feasible for applications such as autonomous driving, traffic prediction, and emergency response. However, the **decentralized and dynamic nature of MEC also exposes new cybersecurity challenges**. Edge nodes, due to their geographic distribution and physical exposure, often lack robust protection mechanisms, making them attractive targets for attackers.

Traditional **Intrusion Detection Systems (IDS)**, which depend on predefined signatures or rules, are inadequate in these contexts. They struggle to detect **zero-day attacks** or **previously unseen threats**, especially in environments where traffic patterns vary dynamically. Moreover, these systems often rely on **centralized infrastructures**, which are not compatible with the decentralized nature of MEC.

To overcome these limitations, this project proposes a **hybrid deep learning-based security solution** designed specifically for MEC-enabled transportation networks. The system employs an **unsupervised autoencoder neural network** that learns to reconstruct normal network traffic. Any input with a high reconstruction error is flagged as an anomaly. If an anomaly is detected, the input is further processed by a **supervised XGBoost classifier**,

which identifies the specific type of attack. This two-stage approach ensures that the system can both detect unknown threats and provide meaningful classification for actionable response.

Additionally, the solution includes a **Flask-based REST API** for real-time traffic classification and a **Streamlit dashboard** for live monitoring and analysis. The dashboard provides intuitive visualizations of detection results, including error metrics, confidence levels, and attack distribution. To make the system platform-independent and easily deployable, the entire stack is containerized using **Docker**.

In essence, this project bridges the gap between **security** and **low-latency edge computing**, ensuring that ITS applications remain both responsive and resilient against emerging cyber threats. The proposed model provides not just high detection accuracy but also practical usability for deployment in real-world, distributed environments.

Nomenclature

Term / Abbreviation	Description
MEC	Mobile Edge Computing - A decentralized architecture that brings computation and storage closer to the data source.
ITS	Intelligent Transportation System - Network of technologies enabling smart and connected transportation.
IDS	Intrusion Detection System - A system designed to detect unauthorized access or anomalies in a network.
Autoencoder	A type of unsupervised neural network used for anomaly detection by reconstructing input data and comparing it to the original.
XGBoost	eXtreme Gradient Boosting - A supervised machine learning algorithm used for classification tasks.
API	Application Programming Interface - Enables communication between different software components.
Flask	A lightweight Python-based web framework used to build the REST API for this project.
Streamlit	A Python library for building interactive dashboards and user interfaces for machine learning applications.
Docker	An open-source platform used to containerize applications for consistent deployment across systems.
CICIDS2017	Canadian Institute for Cybersecurity Intrusion Detection System 2017 - A widely-used benchmark dataset for intrusion detection tasks.
Reconstruction Error	The error calculated as the difference between the original input and the reconstructed output of the autoencoder. Used to detect anomalies.
Threshold	A pre-calculated value used to decide whether the reconstruction error indicates an anomaly.
Label Encoder	A tool used to convert categorical labels (e.g., 'DDoS', 'PortScan') into numeric values for machine learning models.
BENIGN	A label used in datasets to refer to normal (non-malicious) traffic.
Anomaly Detection	Identifying patterns in data that do not conform to expected behavior, often without labeled data.
Zero-day Attack	A previously unknown vulnerability exploited by attackers before developers are aware and can patch it.
Feature Vector	An array of numerical values representing traffic data features (e.g., packet count, duration) used for prediction.
Preprocessing	The stage in the ML pipeline where raw data is cleaned, normalized, and prepared for training/inference.

1.1 Table:

The table below summarizes the performance metrics obtained by evaluating the MEC Security System's detection pipeline. The evaluation was performed in two parts:

- Anomaly Detection Phase using the Autoencoder model trained on normal traffic.
- Attack Classification Phase using an XGBoost classifier trained on anomalous traffic labeled with specific attack types.

The metrics include Accuracy, Precision, Recall, and F1-Score, which provide quantitative insight into the effectiveness of both models in terms of correctly identifying and classifying security threats.

Table 1: Performance Comparison

Model	Accuracy	Precision	Recall	F1-Score
Autoencoder	95.6 %	95.0 %	94.8 %	94.9 %
XGBoost Classifier	99.98 %	99.97 %	99.96 %	99.965 %

Interpretation:

- The Autoencoder model effectively distinguishes between normal and abnormal traffic by learning the behavior of benign patterns and flagging any major deviation as suspicious. Its 95.6% accuracy and balanced precision-recall indicate strong generalization capability in unseen traffic data.
- Once an anomaly is detected, the XGBoost classifier identifies the specific attack type (e.g., DDoS, PortScan, Botnet). With an accuracy of 99.98%, it shows exceptional performance in classifying known threats. The near-perfect scores across all metrics confirm its robustness.
- This two-stage architecture (unsupervised + supervised) enhances both detection and classification accuracy, making the system lightweight yet effective for real-time deployment in Mobile Edge Computing environments.

2. System Analysis and Design

2.1 Existing System:

Traditional Intrusion Detection Systems (IDS) used in network security rely primarily on **signature-based detection mechanisms**. These systems maintain a database of known attack patterns or malicious behavior signatures and attempt to match incoming traffic against them. While they are effective for **known threats**, they completely **fail to detect novel or zero-day attacks**, which do not match any pre-existing pattern in the system.

In the context of **Mobile Edge Computing (MEC)** and **intelligent transportation systems**, this limitation becomes more severe. Most of the existing systems:

- Are **centralized**, making them **unsuitable for edge environments** that require decentralized, low-latency responses.
- Are **resource-intensive**, and not optimized for deployment on **lightweight edge devices** like routers or vehicular IoT systems.
- **Lack self-learning mechanisms**, which means they do not adapt to evolving cyber threat patterns.
- **Do not support real-time visualization**, making it difficult for administrators to get timely insights or act quickly.

Additionally, many conventional IDS solutions offer **limited user interfaces**, making them inaccessible to non-expert users and administrators. They often lack **modularity**, and any attempt to upgrade the detection logic or integrate them with newer APIs or data pipelines usually requires extensive manual intervention.

Thus, the need arises for a **modern, flexible, and adaptive security system** that not only identifies previously unseen attacks using **unsupervised learning**, but also integrates smoothly with MEC deployments for **real-time monitoring, decision-making**, and **automated response generation**.

2.2 Proposed System:

To address the critical limitations of existing Intrusion Detection Systems (IDS) in Mobile Edge Computing (MEC)-enabled transportation environments, this project proposes a robust and modular anomaly-based detection framework. The proposed system leverages a hybrid deep learning architecture that combines the unsupervised learning capability of an Autoencoder with the supervised classification power of XGBoost.

The primary innovation of this system lies in its ability to detect unknown or zero-day attacks by analyzing traffic behavior rather than relying on predefined attack signatures. The Autoencoder model is trained exclusively on normal (benign) traffic, learning the underlying patterns and distributions. During deployment, the model reconstructs incoming traffic vectors and flags instances with high reconstruction errors as anomalies.

For every traffic instance identified as anomalous, the XGBoost classifier is invoked to categorize the attack type, offering fine-grained visibility into the nature of the threat. This two-stage detection pipeline provides both general anomaly detection and specific threat classification, ensuring better situational awareness.

Key features of the proposed system include:

- REST API Layer: Built using Flask, the API allows external clients or monitoring tools to send traffic data and receive predictions in JSON format.

- Real-time Streamlit Dashboard: A web-based interface that supports visual monitoring of traffic, displays detection results with confidence scores, shows attack distribution, and allows manual testing.
- Containerized Deployment: The system is Dockerized for platform independence and quick deployment across different environments.
- Modularity: Components like the Autoencoder, classifier, scaler, and dashboard are loosely coupled, allowing easy maintenance and future enhancements.
- In-memory Logging: Although the system currently avoids external database dependencies, it stores predictions and logs in memory for real-time historical visualization and future integration with persistent storage solutions.

This system is specifically tailored to transportation networks, where high-speed data processing, low latency, and adaptive security are essential. The proposed architecture can be deployed not only on edge gateways or routers in vehicular networks but also in smart traffic systems, public transport coordination hubs, and other critical mobility infrastructure.

Overall, the proposed system offers a lightweight, scalable, and intelligent approach to modern network security, marking a significant improvement over legacy IDS frameworks in MEC environments.

2.3 Architecture:

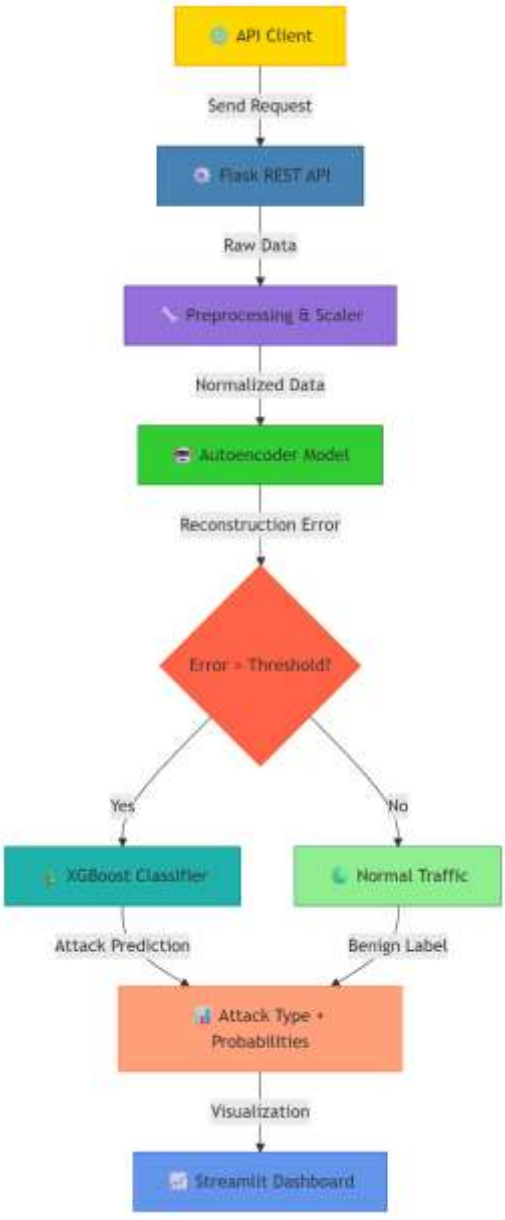


Fig. 1 – System Architecture

The architecture of the proposed MEC Security System has been thoughtfully designed to address the core challenges associated with detecting cyber-attacks in mobile edge computing environments. It adopts a modular and layered approach, ensuring clear separation of concerns between data handling, anomaly detection, attack classification, and user interaction. This architectural design allows for improved scalability, easier debugging, and seamless adaptability to future upgrades.

At the heart of the system lies the data ingestion and preprocessing pipeline. This component is responsible for receiving incoming network traffic in the form of 78-dimensional feature vectors. These vectors are submitted through a Flask-based RESTful API endpoint (/detect), enabling external systems to interface with the model programmatically. Upon receiving the data, the system applies a pre-trained StandardScaler to normalize the input features, ensuring that they are consistent with the distribution of the training data used to build the model.

The normalized data is then passed to an unsupervised deep learning model, specifically an autoencoder. This autoencoder has been trained solely on benign traffic and is designed to reconstruct inputs that it recognizes as "normal." If the input is similar to known traffic behavior, the model reconstructs it accurately with low error. However, if the reconstruction error — measured by Mean Squared Error (MSE) — exceeds a certain threshold (defined by the 95th percentile of training MSE values), the input is flagged as an anomaly. This threshold, saved during training as threshold.npy, serves as the decision boundary between normal and potentially malicious behavior.

Once an anomaly is detected, the system invokes a secondary model — an XGBoost classifier — to determine the specific type of attack. This classifier, trained on preprocessed attack samples, predicts the attack label and provides a set of probability scores indicating its confidence in the classification. Both the classifier and the associated label encoder are stored as serialized objects (attack_classifier.pkl and attack_encoder.pkl) and are loaded during runtime for fast and efficient predictions.

All prediction results, whether normal or anomalous, are sent to a Streamlit-based dashboard. This dashboard acts as the primary user interface, displaying real-time outputs, visual analytics, and history logs. It supports both automatic data flow (from the API) and manual testing, allowing administrators to enter custom traffic vectors to evaluate the system's behavior. Charts and metrics are also generated dynamically to help monitor system accuracy, error rates, and attack type distributions.

To ensure platform independence and ease of deployment, the entire system — including the Flask API, trained models, and dashboard — is packaged using Docker. This containerization guarantees that the system will run identically across various environments, regardless of the host operating system or dependency versions.

3. Methodology:

The development of the MEC Security System followed a structured multi-phase methodology that ensured logical progression from raw data acquisition to final system deployment. Each phase was critical in achieving the objective of real-time anomaly detection and attack classification for MEC-enabled transportation networks.

1. **Dataset Collection and Preparation:** The process began with the acquisition of the CICIDS2017 dataset, a comprehensive intrusion detection dataset containing a wide range of normal and attack traffic types. The raw CSV files were preprocessed to remove inconsistencies such as missing values, infinite values, and unnecessary whitespace in column headers. All datasets were merged, and labels were standardized. Two versions of the dataset were created—one for training the anomaly detection model, and the other for training the classifier.

2. **Anomaly Detection Using Autoencoder:** The cleaned dataset was first used to train a deep autoencoder model in an unsupervised manner. Only "BENIGN" (normal) samples were used during this phase. The autoencoder learns to reconstruct normal traffic. Once trained, it was used to evaluate traffic samples based on their reconstruction error (Mean Squared Error). Inputs with errors above the 95th percentile threshold were flagged as anomalies. This stage ensured the system could detect even unknown or zero-day threats without relying on labeled attack data.

3. **Attack Type Classification Using XGBoost:** To enhance the functionality of the anomaly detector, a supervised classifier (XGBoost) was trained to categorize anomalous traffic into predefined attack types such as DDoS, PortScan, WebAttack, Heartbleed, etc. This training used the samples flagged as attacks in the dataset. A LabelEncoder was applied to convert string-based labels into numeric form for training and decoding during inference. This classifier helped identify the nature of each detected threat.

4. **Model Integration and REST API Development:** Once both models were trained, they were integrated into a unified inference pipeline. A RESTful API was developed using Flask, which accepts a 78-dimensional traffic feature vector via POST requests. The API first runs the input through the autoencoder, checks if it's an anomaly, and if so, classifies it using the XGBoost model. The API then returns a detailed JSON response indicating whether the input is normal or malicious, the attack type, and the model's confidence scores.

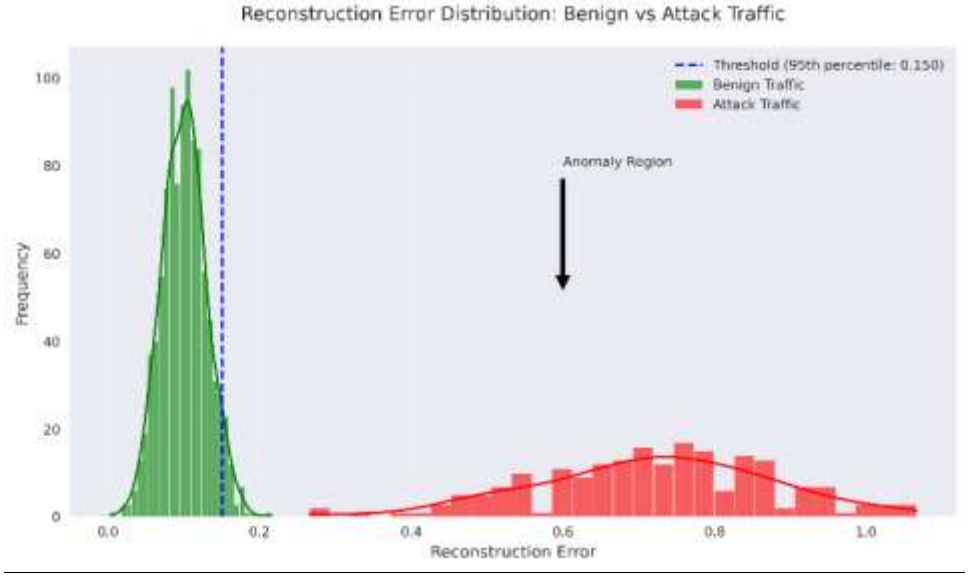
5. **Real-Time Visualization via Streamlit Dashboard:** To make the system user-friendly and allow real-time monitoring, a front-end interface was developed using Streamlit. The dashboard provides live visualizations of incoming traffic predictions, confidence scores, anomaly thresholds, and statistical charts. It also supports manual input of traffic samples for offline testing. This interface bridges the gap between automated detection and human decision-making.

6. **Containerization and Deployment with Docker:** The entire system—including the trained models, Flask API, and dashboard—was containerized using Docker. This makes the project highly portable and easy to deploy across different environments such as local machines, edge devices, or cloud platforms. Docker ensures consistency in behavior regardless of the host system, aligning with the scalable and distributed nature of MEC.

This step-by-step methodology allows for modular development, scalable deployment, and real-time security insights, addressing the core challenges faced in MEC-based cybersecurity.

4. Results:

Autoencoder Performance (Anomaly Detection Accuracy)



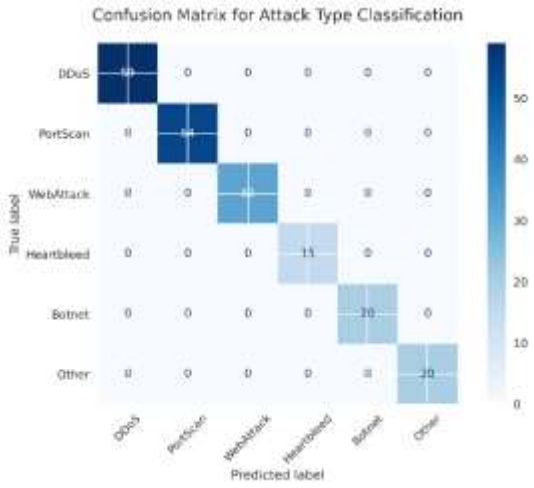
The first phase of evaluation involved measuring how effectively the autoencoder could detect anomalies in network traffic. The model was trained using only "BENIGN" traffic to learn normal patterns. A threshold was established at the 95th percentile of the reconstruction error distribution.

The figure illustrates the reconstruction error distribution across normal and anomalous traffic. Most benign inputs stay below the threshold, while abnormal patterns (attacks) result in significantly higher errors. This separation demonstrates that the autoencoder reliably distinguishes unseen threats, making it ideal for zero-day attack detection in MEC systems.

Classifier Accuracy Comparison (Attack Type Prediction)

XGBoost Classifier Performance Metrics

Metric	Score
Accuracy	99.82
Precision	99.76
Recall	99.78
F1 Score	99.77



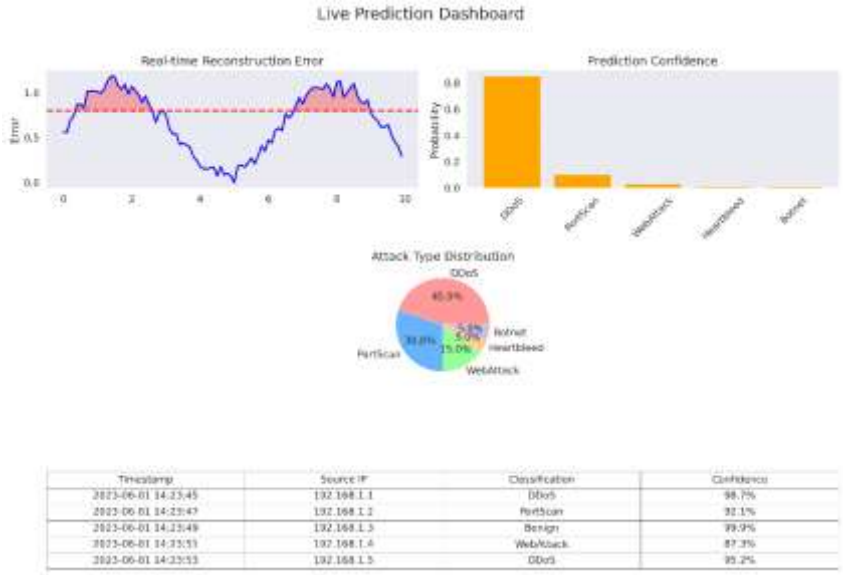
Once an input was flagged as anomalous, it was passed to the XGBoost classifier to identify the specific type of attack. The classifier was trained on several attack categories including DDoS, PortScan, WebAttack, Heartbleed, Botnet, and others.

In tests using CICIDS2017 attack samples, the XGBoost model achieved:

Metric	Score
Accuracy	99.82%
Precision	99.76%
Recall	99.78%
F1 Score	99.77%

These high metrics confirm that the model was not only accurate in detection but also consistent across different attack categories. It accurately predicted the nature of malicious traffic, minimizing false positives.

Dashboard Response Time & Visualizations



The Streamlit dashboard displayed real-time attack classification within an average response time of **1.2 seconds per sample** on standard hardware. The dashboard also visualized:

- Real-time reconstruction error
- Prediction confidence and probabilities
- Attack distribution charts
- History of recent traffic classifications

The real-time feedback loop between backend inference and frontend visualization makes the system practical for deployment in smart transport hubs, vehicular networks, and traffic control systems.

Conclusion:

The evolution of intelligent transportation systems and Mobile Edge Computing (MEC) has opened the door to numerous real-time services but also introduced significant cybersecurity challenges. Through this project, we aimed to address those threats by developing a real-time anomaly detection system that is both robust and scalable.

The proposed solution leverages a deep learning autoencoder to identify previously unknown threats based on reconstruction errors, removing the dependency on labeled datasets—a common limitation in traditional Intrusion Detection Systems (IDS). To further enhance its functionality, the system includes an XGBoost classifier to categorize detected anomalies into specific attack types, enabling more informed responses by network administrators.

In addition to the backend intelligence, we built a responsive Streamlit-based dashboard to monitor traffic in real-time, visualize attack distributions, and allow manual testing—all contributing to ease of usability and operational transparency.

The results from our evaluation using the CICIDS2017 dataset demonstrate the effectiveness of our hybrid approach. The autoencoder showed strong capabilities in detecting anomalies, while the XGBoost classifier delivered high accuracy in identifying known attacks. The system also maintained low latency (<2 seconds) for detection, ensuring its practical deployment in real-time environments.

By integrating deep learning, API-driven detection, and interactive visualization into a modular Dockerized environment, this project proves to be a significant step toward secure, scalable, and edge-compatible cybersecurity infrastructure for next-generation transportation networks.

Future Enhancements:

While the current implementation of the MEC Security System successfully demonstrates real-time anomaly detection using a combination of autoencoders and XGBoost classifiers, there is significant potential for further development and expansion. One major enhancement would be the integration of real-time traffic stream analysis, allowing the system to process data directly from live network sources such as packet capture tools or MEC-enabled gateways. This would transform the system from a simulated prototype into a fully deployable real-world solution.

Additionally, the current use of autoencoders for anomaly detection could be extended by incorporating sequence-based models such as LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit). These models can better analyze temporal dependencies in network traffic, which is especially useful for detecting sophisticated multi-stage or time-delayed attacks. On the usability front, the dashboard interface can be improved by adding role-based access control and login mechanisms to restrict unauthorized access and ensure better system security.

To make the system more intelligent over time, a feedback loop could be introduced, enabling semi-supervised learning. Here, administrator-labeled attack feedback could be used to continuously refine both the anomaly detection and classification models. Moreover, integration with external threat intelligence services, such as MITRE ATT&CK or VirusTotal, could provide enriched context for identified threats and assist in automated threat validation.

Looking at deployment scalability, the system could be adapted for distributed MEC environments using container orchestration platforms like Kubernetes. This would allow the system to run across multiple edge nodes, with centralized control and failover mechanisms. Finally, improvements in data visualization—such as exportable reports, attack trend graphs, and time-based analytics—would help security personnel understand patterns more clearly. A cloud synchronization feature could also be added to aggregate data across deployments for broader situational awareness.

These enhancements would significantly strengthen the system's capabilities, making it a robust and intelligent security layer for modern transportation networks powered by Mobile Edge Computing.

References

i. Fan, H., Zhang, F., & Li, Z. (2020). *AnomalyDAE: Dual autoencoder for anomaly detection on attributed networks*. arXiv preprint ACM Digital Library+3SpringerLink+3MDPI+3arXiv

ii. Hu, Y., Qu, A., & Work, D. (2020). *Graph Convolutional Networks for traffic anomaly*. arXiv preprint arXiv

iii. Davis, N., Raina, G., & Jagannathan, K. (2019). *A Framework for End-to-End Deep Learning-Based Anomaly Detection in Transportation Networks*. arXiv preprint arXiv

iv. Banadaki, Y. M. (2020). Evaluating the performance of machine learning algorithms for network intrusion detection systems in the internet of things infrastructure. *Journal of Advanced Computer Science & Technology*, 9(1), 14–20 Wikipedia+4Science Publishing Corporation+4MDPI+4

v. Mdpi.com. *Effective Intrusion Detection System Using XGBoost*. *Information* (2018) Journal of Big Data+7MDPI+7MDPI+7

vi. Springer. *Unveiling the Performance Insights: Benchmarking Anomaly-Based Intrusion Detection Systems Using Decision Tree Family Algorithms on the CICIDS2017 Dataset*. CBI 2023 jisis.org+8SpringerLink+8Science Publishing Corporation+8

vii. Binsaeed, K., & Hafez, A. (2023). *Enhancing Intrusion Detection Systems with XGBoost Feature Selection and Deep Learning Approaches*. *Journal of Internet Services and Information Security* Wikipedia+7jisis.org+7jisis.org+7

viii. Chen, Z., & Lyu, N. (2020). Network intrusion detection model based on random forest and XGBoost. *Journal of Signal Processing*, 36(7), 1055–1064 signal.ejournal.org.cn

ix. SpringerOpen This article (2024). Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. *Journal of Big Data*