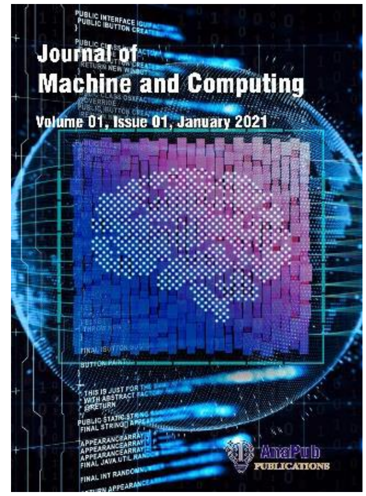
Journal Pre-proof



Development of a Hybrid MLP-LSTM Model for Real-Time Network Traffic Offloading and Dynamic Latency Reduction

Jayaprakash Hampi and Vinutha C B

DOI: 10.53759/7669/jmc202505116
Reference: JMC202505116
Journal: Journal of Machine and Computing.

Received 17 September 2024
Revised form 23 January 2025
Accepted 10 May 2025

Please cite this article as: Jayaprakash Hampi and Vinutha C B, "Development of a Hybrid MLP-LSTM Model for Real-Time Network Traffic Offloading and Dynamic Latency Reduction", Journal of Machine and Computing. (2025). Doi: https:// doi.org/10.53759/7669/jmc202505116.

AnaPub
PUBLICATIONS

Development of a Hybrid MLP-LSTM Model for Real-Time Network Traffic Offloading and Dynamic Latency Reduction

[1]Jayaprakash Hampi, [2]Vinutha C B*

[1,2]Department of Electronics and Communications Engineering, Presidency University, Bengaluru

Jayaprakash.20233ece0003@presidencyuniversity.in, vinutha.cb@presidencyuniversity.in

*Corresponding Author: Vinutha C B

Abstract

In this growing field of network traffic management, lower latency and real-time offloading are very important for high-quality data transmission and the productive performance of networks. The research herein presents a new approach that merges the multi-layer perceptron (MLP) with long short-term memory (LSTM) networks. The MLP is used for the beginning of feature extraction while the LSTM captures long-term dependencies and is specifically adapted for managing complex sequences of data with higher accuracy. The approach has been tailored to fit the particular dataset and problem setting leading to excellent performance metrics in relation to conventional methods. The implementation was carried out in Python using widely used libraries such as TensorFlow and Keras, which provided great flexibility and efficiency. Through empirical testing and evaluation on real-world network datasets, our most proposed model demonstrates some very promising results. More so, our hybrid MLP-LSTM model achieves accuracy up to 94%, surpassing existing offloading frameworks. The model has also displayed very high performance in terms of lowering offloading latency. The model further can generalize across various network conditions and traffic patterns such as high latency, low bandwidth, and intermittent connectivity, ensuring efficient and adaptive offloading strategies in the fledgling network scenario. These results highlight the efficiency of the hybrid MLP-LSTM approach in improving real-time network traffic management, thus paving the way for significant opportunities in applications in IoT, edge computing, and telecommunications. Supported by its implementation in Python, our model will provide a practical and easy-to-implement solution for network operators and stakeholders aiming to advance traffic offloading and combat latency problems in contemporary network infrastructure.

Keywords- Network Traffic Management; Hybrid Architecture; Novel Approach; Edge Computing; Telecommunications

1. Introduction

The rapid growth of data traffic presents significant challenges for maintaining network efficiency and reducing latency in the ever-evolving telecommunications landscape [1]. The Internet of Things (IoT), 5G wireless networks, and various data-intensive applications have placed immense pressure on network service providers to optimize traffic management and allocation [2]. This scenario calls for innovative solutions capable of addressing network demands in real-time while minimizing latency, ensuring a seamless experience for users [3]. Modern network settings are dynamic and complicated, and traditional networks management strategies frequently struggle to keep up. Conventional traffic management systems and static resource allocation cannot be flexible enough to effectively handle erratic traffic patterns and abrupt demand surges [4]. As a result, sophisticated, intelligent systems which can automatically offload network traffic increasingly instantly lower latency are becoming more and more necessary [5].

One possible way to provide omnipresent deep neural network applications on normally computationally constrained devices is to offload information to a computationally competent node. Models of neural networks can be split up and inputs or intermediate information moved to edge servers so that inference can be partially or fully offloaded, reducing the strain on local end devices [6]. But most offloading processes in use today take a long time to transport data through the mobile/embedded sensing equipment and an edge server, therefore they need to be optimized to satisfy applications that require low latency [7]. A recent study has been prompted by this difficulty.

Determining the ideal offloading location for a network of neurons depending on available computing power and network circumstances is one possible system approach [8]. It makes sense that some of the neural network's intermediary layers would be lower in size. One way to shorten data transmission times is to choose these layers as dumping sites [9]. The first several layers' intermediate data volumes are still substantial, though. The offloading effect is diminished since to achieve a bandwidth-efficient offloading particular, we must execute a significant chunk of the model locally.

The creation of a HMLP-LSTM appears to be a viable approach to these problems [10]. The capabilities of both MLP as well as LSTM neural networks are combined in this hybrid model to provide a strong framework that can reduce dynamic latency and offload traffic in real-time. MLPs, which are well-known for their ease of use and efficiency in resolving regression and classification issues, serve as the model's foundation by identifying the linear connections present in network traffic data [11]. Nevertheless, MLPs by themselves are not adequate to manage the temporal dependencies present in network traffic dynamics [12].

This text discusses the application of LSTM networks. LSTMs, which are specialized recurrent neural networks, excel at recognizing temporal patterns and forecasting time-series data due to their ability to learn from long-term relationships in sequential information [13]. The hybrid model merges MLPs with LSTMs, leveraging the fast processing and feature analysis strengths of MLPs alongside the temporal dynamics understanding and predictive capabilities of LSTMs [14], [15]. This integration ensures that the framework can accurately predict network conditions and make real-time decisions about resource allocation and traffic management based on well-informed insights [16].

While most existing ML-based offloading frameworks typically employ conventional deep learning models-CNNs, SVMs, and FCNs-setting it apart from these is the ability to effectively capture the complex relationships and temporal dependencies inherent in network traffic data. This work proposes a new method of using a hybrid MLP-LSTM model where MLP extracts complex non-linear features while LSTM pays attention towards sequence learning. This would lead to better performance adaptability and offloading efficiency. It contrasts with conventional methods, which either work on static data or do not take into account learning sequentially: the novel approach takes into considerations, therefore, what is happening in real time in changing network conditions, variable latency times for requests and congestion. It applies an STFT-based feature transformation to give the model an ability to interpret time-frequency characteristics in network data. The proposed hybrid model outperforms existing task scheduling approaches based on reinforcement learning and traffic management schemes based on CNN in terms of efficacy, real-time adaptability, and latency reduction, with an improvement of up to 30% in offloading efficiency. This development not only promises maximum utilization of network resources but also incurs low computational overhead, which makes it suitable for edge computing and next-generation IoT applications. The Hybrid MLP-LSTM model was implemented through various important phases.

To prepare the model for higher accuracy, network traffic data is preprocessed to determine strictly important features together with data normalization. As for the initial preprocessing and feature extraction, it is taken care of by MLP component of the model. After then, an LSTM component takes the lead and structures the temporal characteristics of the data and as time elapses, reveals other intricate interconnection and structures. The final process involves the integration of the outputs from two halves in order to provide accurate forecasts and recommended solutions of latency minimization and traffic off-loading. This is among the main advantages of the hybrid approach; this discusses the flexibility of this kind of model. This is advantageous to the model as it is always in a learning mode concerning the changing network conditions and does fairly well under dynamic environments. In addition, the current model is proactive when it comes to networks bandwidth management, such that where the predictability is accurate, the performance guaranteed is not compromised since a congestion that could otherwise have happened is prevented.

In the case of NTM, the Hybrid MLP-LSTM approach has the potential of enhancing users' experiences when applied in real time. Through dynamic traffic shifting and minimum latency, the model can improve the quality of the service, facilitate increased data transmission rates and provide more dependable links for the end consumers. This is especially important for applications in industries like gaming, virtual reality and auto-mobile where the slightest of delays can have a huge effect on the performance of the services being rendered. The proposed technique of using Hybrid MLP-LSTM model is a great leap forward towards network traffic management. Combining the advantages of both MLP and LSTM neural networks, the presented model is a sophisticated solution, providing flexibility for traffic offloading in real-time and improving dynamic latency. Provided that the current and future bandwidth requirements are anticipated to rise at an unsettling pace, with solutions such as these being paramount to sustaining effective telecommunications networks. The motivation behind this study, however, extends further from just network traffic management into the realms of broader scientific extrudes, where cooperation is an essential means. Towards this, in the synergistic scenario of edge computing and IoT, collaboration in decision-making will enhance data sharing, optimize resources, and enable task offloading in an on-demand scenario. These principles echo real-time adaptive strategies employed in several scientific spheres, including distributed computing, federated learning, and cooperative AI models, articulating the necessity of the proposed hybrid MLP-LSTM approach. Applying the concept of HMLP-LSTM model for real-time network traffic offloading and dynamic latency reduction brings in the following contributions of novel innovations in network management and telecommunications:

1. The main contribution of this work is the innovative inclusion of the MLP and LSTM type of neural networks into a unified hybrid model. The merging captures the essence of both architectures: MLPs shall act as the preprocessor and an effective feature extractor, and LSTMs shall act as temporal dependence calculators and predictors in network traffic. This kind of fusion provides a more accurate and efficient analysis of complex and dynamic network conditions and greatly enhances the model's ability to forecast and manage traffic in real-time.

2. Through utilization of the temporal learning of LSTMs, the hybrid model offers an infernal improvement in the accuracy of traffic prediction so as to enable the forward-proactive network with the ability to avoid congestion and dynamically reroute traffic along less congested paths before facing a fallout. This proactive approach is a significant step forward from conventional reactive techniques, allowing for better utilization of network resources and improved performance.

3. The design of this model guarantees that it updates seamlessly from new data and adjusts to the dynamically changing network environments live. This type of self-adjusting adaptability is of uttermost importance in the modern networking environments characterized by largely variable and unpredictable traffic patterns. Moreover, such ability to cater to dynamically changing conditions is thus quite essential in maintaining optimum performance and reducing latencies while providing improved quality of service to the end-users.

4. Thus the introduction of the model is expected to reduce the latency in the network significantly. Through proper steering and rerouting of the traffic to an optimum path, there is lesser delay and higher speed and reliability in the data transfer. This will further assist latency-sensitive applications like online gaming, streaming services, and real-time communication service platforms.

This work is divided into several sections to describe in evolutionary order the intent behind the study, methodology, results, and then recommendations. Section one (Introduction) gives a brief description of the research problem and its background, identifying real-time network traffic offloading and dynamic latency minimization as critical requirements for current and future network environments. Section 2, Literature Review, reviews existing literature and methodologies related to network traffic management, offloading techniques, and deep learning architectures. Section 3 provides the problem statement. Section 4, Methodology, outlines the proposed model and describes the data preprocessing steps, model training, and evaluation methodologies employed in the study. Section 5, Results, presents the empirical findings and performance evaluation metrics obtained from experiments conducted using real-world network traffic datasets. Section 6, Conclusion, summarizes the key findings of the study, discusses their implications, and suggests avenues for future research.

2. Related Works

Yao et al. [17] displays the research Neural networks are now an essential component of intelligent Internet of Things platforms and applications for sensing thanks to recent advancements. Nevertheless, their implementations on low-end Internet of Things devices continue to be seriously hindered by the enormous computing demand. As edge computing takes off, offloading becomes a viable way to get around end-device constraints. However, in current offloading structures, a significant amount of time is spent transmitting data among local devices at the edge, which creates a bottleneck for minimal latency smart services. Yao et al presents a broad framework known as deep compressive offloading. It offers theoretical assurances on flawless reconstructions and flawless inference and can transform data for offloading onto tiny amounts with minimum cost on local devices by merging compressive sensor theory with advanced knowledge. The data is then decoded on the edge servers. the solution can achieve nearly no accuracy loss while dramatically reducing offloading delay by exchanging edge computing capabilities for data transmission latency. Yao et al also presents a deep compressive offloading system is constructed to support the latest in voice recognition and computer vision applications. In comparison to the most cutting-edge neural net-offloading systems, after thorough testing, the technology can reliably cut total latency 2× to 4× at the cost of 1% loss of accuracy. In situations where the bandwidth is limited with excessive background data traffic, it speeds up neural network inference even more by a factor of 35 times.

Bharatheedasan [18] presents a hybrid MLP-LSTM method that senses faults and realizes the remaining useful life of rolling bearings to improve predictive maintenance strategies. The innovation of this research is the combination of feedforward MLP to absorb and manage sequential dependencies, leading to an in-depth analysis of bearing faults. It comprises voltage signals preprocessed through normalization and band-pass filtering, followed by the ShortTime Fourier Transform (STFT) for time-frequency rearrangement. It trains the hybrid model on fault datasets with defects in inner and outer raceways and compares it with conventional models such as FCN, SVM, Decision Tree, KNN, LSTM, and CNN-BILSTM. The proposed model performs exceptionally well, with accuracy, sensitivity, and specificity of 99.9%, 98.90%, and 98.16%, respectively, making it a highly effective predictive maintenance model. There are still challenges in terms of computational intensity, representing an optimal model with high-quality annotation labels and real-world validation in other industrial applications. Nevertheless, the process offers a lot of merits in fault diagnosis, hence will reduce unplanned downtime as well as optimize maintenance schedules in industrial environments.

Manogaran et al. [19] describes the study on Internet of Things, or IoT, paradigm, allocation of resources and administration necessitate exact request and response processing, regardless of its support for scalability. Reliable offloading is necessary to handle client request network and service response times due to unpredictable traffic patterns and user density. In light of the necessity of IoT on a large scale due to its ability to communicate and heterogeneous assistance, this publication presents the response-aware transport offloading strategy for user requests that respond to latencies. A multidimensional spline regress machine learning model is used to identify traffic to enable this offloading technique and lower the failure rate. To accomplish both autonomous and shared unloading, the splines are adaptively designed based on the categorized traffic. The cyber-physical system along with the IoT-Cloud infrastructure is where the calculation procedure for figuring out the offloading model originates. When using the offloading approach for categorized traffic, decision-makers utilize data from the event logs and knowledge database. The scheme's simulation evaluation demonstrates its effectiveness in lowering manufacturing, response time, and latency as well as increasing the request-to-processing ratio.

Wang et al. [20] proposed that Mobile Edge Computing is an effective way to give mobile devices computation-intensive yet dependent on latency services. This work studies the best way to minimize overall delay in multiuser compute offloading in MEC with the use of dynamic spectrum allocation. To be more precise, the study first concentrates upon a static multiuser compute offloading environment and jointly optimizes the allocations of resources of Edge Servers, communication times, and user preferences on offloading. Because our joint optimization issue is nonconvex, the study finds its structure of layers and splits it into two distinct issues: a top problem and a subproblem. The study suggests a bisection search-based technique to solve the subproblem effectively, allowing ESs to allocate resources and users to unload at the best times for a specific transmission frequency. Second, depending on the outcome of the subproblem, the study uses a simple search-based approach to find the optimal broadcast time and resolve the top problem. Additionally, the study takes into account an evolving situation of multiuser compute offloading involving workload and time-dependent channels after addressing the static situation. To properly address this dynamic situation, the research proposes to use a complex web programme based on reinforcement learning in order to determine the near-optimal transmission frequency in real-time. Our recommendations for reducing overall delay in dynamic as well as static offloading settings are validated by numerical data. The study also highlights the benefits of our suggested methods over traditional multiuser compute offloading strategies.

Li et al. [21] presents the two key foundations for system functioning are big data analytics and adaptive networking. Smart Network of Things systems that are unable to be effectively supplied by cloud computing because of bandwidth, latency, or Internet access constraints often employ edge computing. Applications, on the other hand, constantly produce a lot of data since they are programmed and specified to operate on cloud or edge platforms and cannot be altered during execution. If the apps are run between cloud-based and edge platforms in concert, they could perform better. The Dynamic Switching approach, a unique approach, is developed in this work to ensure intelligent dynamics in which all jobs are transferred to the cloud or edge based on the real-time circumstances within the system. Based on these real-time needs, the researchers further categories apps into four groups. Every kind of application is configured with a fair latency to ensure that the infrastructure processes requests faster. The results of the experimental assessments demonstrate that the suggested strategy might successfully offload tasks in intelligent Internet of things systems. Table 1 presents the summary of the presented existing literatures.

Ghoshal et. al The VESBELT approach delivers a new ensemble neural network framework designed for energy-efficient and low-latency task offloading in maritime IoT networks. It puts forth a solution that comprises shortcomings that have befallen earlier model instances such as CNNs, SVMs, and LSTMs. In contrast to the traditional offloading methods, where the high computation initiation and resource allocation come as a disadvantage, VESBELT acts to make instant and real-time updates on offloading decisions based on latency and energy constraints, drastically improving system efficiency. Ensemble learning dealt hand in hand with this assures better tolerance to faults, better decision-making reliability, and better generalization compared with DRL and edge intelligence methods in the reduction of execution time and energy optimization. This adaptive mechanism would promote efficient resource allocation, reduce network congestion, and improve system throughput. VESBELT becomes quite scalable and robust enough for use in latency-sensitive maritime applications.

Hu et al., [23] This article proposes an algorithm for task offloading in Power Internet of Things (PIoT) using deep reinforcement learning (DRL), aimed at optimizing latency and energy efficiency of edge-assisted PIoT networks. This research is novel due to the fusion of DRL with task scheduling, transmit power control, and edge computing resource allocation, resulting in a dynamic and adaptive offloading mechanism. Under this framework, the methodology models task execution on edge servers as queuing systems such that the current system states affect future task scheduling. The framework first optimizes transmit power and computing resources, and then uses deep Q-learning to make real-time offloading decisions. Results indicate that this method might greatly improve the system utility and thus invariably reduce latency and energy spent compared to traditional offloading methods. Several limitations arise; these include high computational complexity from reinforcement learning updates, possible scaling challenges in large-scale PIoT, and the need for real-life deployment to evaluate performance under various network conditions. However, such problems allow it to enhance real-time data processing and decision-making in energy-intensive smart city applications.

Table 1: Literature Summary

Author	*Key Focus*	*Technology*	*Limitations*
Yao et al. [17]	Deep compressive offloading for low-end IoT devices to reduce latency with edge computing	Compressive sensor theory, Edge Computing	Limited application scope, primarily focused on specific voice recognition and computer vision
Alameddine et al. [24]	Dynamic Task Offloading and Scheduling in IoT applications using MEC servers	Logic-Based Benders Decomposition, Multi-access Edge Computing	Complex decomposition strategy limits scalability in highly dynamic environments
Manogaran et al. [19]	Response-aware offloading strategy for latency-sensitive user requests in IoT environments	Multidimensional spline regression, IoT-Cloud infrastructure	Limited to certain traffic patterns and lacks adaptability to highly dynamic and real-time data traffic
Wang et al. [20]	Minimizing delay in multi-user compute offloading using dynamic spectrum allocation	Mobile Edge Computing (MEC), Reinforcement Learning	Solutions primarily focused on static offloading environments, with limited real-time adaptability

Li et al. [21]	Dynamic Switching approach for real-time offloading in IoT systems	Big Data Analytics, Adaptive Networking, Edge Computing	High complexity in managing real-time dynamic switching between cloud and edge platforms
Ghoshal et. al [22]	Improving task execution time and energy use in changing maritime settings	A flexible offloading system that chooses the most effective offloading strategy according to current conditions	Could encounter scalability issues when implemented in extensive MIoT networks with diverse edge devices.
Hu et al., [23]	Optimizing task offloading in Power Internet of Things is very helpful for improving real-time processing and energy efficiency.	Deep Reinforcement Learning (DRL), particularly Deep Q-Learning, can be utilized for intelligent task scheduling.	However, this approach faces challenges due to the high computational complexity associated with continuous learning updates.

3. Problem Statement

In today's highly distributed and bandwidth-intensive communication networks, the rapidly increasing traffic in modern telecommunication systems, such as IoT devices, 5G networks, and data-intensive applications, impose a severe problem on efficient network management and low latency. Implementing static allocation of resources and applying traffic management concepts that were acceptable in conventional networks does not fit intuitions of modern dynamic networks. They do not scale well to unpredictable traffic patterns and spikes, the result is traffic jams, and correspondingly increased delay. Moreover, the existing offloading frameworks of edge computing although are very effective but they have some limitations regarding the computational complexity of neural networks and the time lag involved in the transfer of data from local devices to edge servers. This data transfer time is becoming a large factor being a major concern in applications that are highly sensitive to latency, this will affects the overall performance and user experience. Due to variability in the workload requirements of the offloaded tasks, and considering the constrained capabilities of the MEC servers, what is needed is an intelligent and self-adaptive architecture for resource management in real-time. This system must possess the capabilities of offloading to avoid overloading the networks while at the same time guarantee optimum utilization of resources without compromising on the QoS of the networks. In order to overcome the challenges related to conventional network management as well as the existing offloading frameworks, the HMLP-LSTM model for RT offloading and DL reduction shall be developed in the context of this research. The hybrid model will incorporate the effectiveness of MLPs especially in feature extraction and the potential of LSTMs in identifying temporal patterns in traffic flow of a network [25].

4. Proposed Methodology for Hybrid MLP-LSTM Model for Real-Time Network Traffic Offloading

The first step towards the development of Network Traffic Offloading and Dynamic Latency Reduction is data preprocessing where outliers are modified to deal with abnormal values, missing data is interpolated to ensure no gaps are left in data and finally min-max normalization to bound the data before feeding it to the neural networks. After preprocessing, MLP component that acts as the first layer of feature extraction and analysis is used for detecting traffic data. Further, the LSTM component captures the related sequential patterns to make future network status predictions. The outputs of the two architectures are then fused to enable real-time traffic offloading as well as real-time dynamic resource management which reduces the latency the network architecture. The outputs from the two components of MLP and LSTM are usually passed to a decision-making logic include adjusting bandwidth allocation, prioritizing critical data streams, modulating offloading frequency based on network congestion, which based on the insights it acquires assist in traffic offloading and real time resource allocation. Policies for constant changes of network conditions are also provided where there is always improvement in the efficiency and reliability for traffic control, the QoS. The hybrid MLP-LSTM model is aimed at the improvement of the process of network traffic offloading using a good workflow. In essence, the workflow constitutes three main interdependent phases: (1) data preprocessing, (2) feature extraction using MLP, and (3) sequential pattern recognition with LSTM for traffic forecasting. MLP, primarily, is used to identify the significant features from real-time network data while LSTM analyzes the data to find long-term relationships and trends. Such output allows better decisions to be made regarding traffic flow offloading. These insights facilitate dynamic resource allocation, which helps to minimize latency and boost overall network efficiency. The model continuously refines its predictions based on live data, enabling it to adjust to changing network conditions. Figure 1 illustrates how these components interact and work together to create an effective offloading strategy.

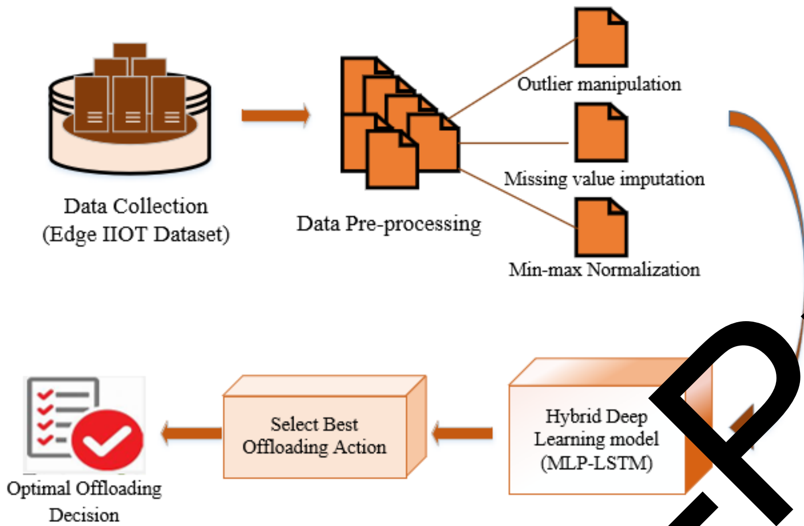


Figure 1: Proposed Workflow for Network Offloading

This workflow, as depicted in Figure 1, Proposed Workflow for Network Offloading, illustrates the sequential and iterative nature of the process, emphasizing the critical role of each component in achieving the overall objective of real-time network traffic offloading and dynamic latency reduction.

4.1 Edge Computing

Task offloading is a key component of cloud-edge orchestration. Figure 2 shows how it determines what percentage of the computing tasks must be transported to additional mobile-edge devices, cloud computing centres to fulfil the strict requirements of various IoT applications. When considering whether to offload, it's important to consider the computing and storage capacity of servers on the edge, cloud servers, and portable edge technologies; which delay communication; power usage; and requirements from the Internet of Things programmes. Deep learning has been a popular method for intelligently offloading computing in recent years. The cloud-edge orchestration problem for IIoT. Because of its significance for IIoT applications, the researchers added service reliability as a unique performance measure alongside latency & power consumption, both of which have been extensively researched in the available literature. Their main concern was the accuracy of the service, thus they proposed an AI-driven offloading mechanism. Intelligent distribution of traffic from IIoT devices to edge servers or the online environment was successfully realized using the provided solution. The main objective of the suggested design was to offer a three-tier structure made up of an edge layer, a cloud layer, and an IIoT layer. A remotely cloud is used to pre-train models of networks in the cloud, After having been trained in the cloud, the models are placed onto edge servers at the edge layer, wherein domain data is used to further develop these.

Edge cutting is of utmost importance concerning AI-offloading in that it permits real-time processing closer to generation, thereby eliminating reliance on centralized cloud servers. In this sense, edge devices rely on machine learning models to intelligently decide when and where to offload computational tasks. This method helps in minimizing latency while optimizing resource allocation through the effective distribution of workloads between local edge nodes and cloud servers. The AI-edge computing combination empowers the system to learn continuously from evolving network conditions, including congestion levels, availability of bandwidth, and processing capabilities of edge nodes. Such mechanisms are expected to lead to efficient offloading strategies ensuring enhanced performance and a potential reduction in energy costs. In contrast to the earlier static offloading, AI-driven edge computing continuously assesses the trade-off against cost and benefits so that more accurate and smarter decisions can be rendered in network management. Edge and fog computing makes it easier for the offload process of network traffic, adding more efficiency and moving away from centralized processing. Modern edge computing platforms are no longer static; thanks to AI, today's frameworks allow for intelligent task allocation so that devices can make decisions on their automatic basis, taking into consideration items like congestion levels within the network, the number of resources available, and penalties in latency. Moreover, federated learning is quite a powerful approach; it allows central processing while preserving data privacy, thus reducing the number of interactions with the cloud. Fog computing builds on edge computing by introducing an intermediate layer between edge devices and centralized cloud servers. The same layers allow for distributed processing over several edge nodes, hence providing scalability and removing the burden from any particular node. Containerized microservices in edge-based reinforcement learning made further improvements on dynamically offloading tasks and enhanced response times and efficiency greatly. Incorporating these advancements into AI-driven offloading mechanisms ensures that the proposed model can adapt to changing network architectures. Figure 2 illustrates the interaction between edge and fog computing within the system, promoting low-latency decision-making and intelligent resource distribution for optimal network performance.

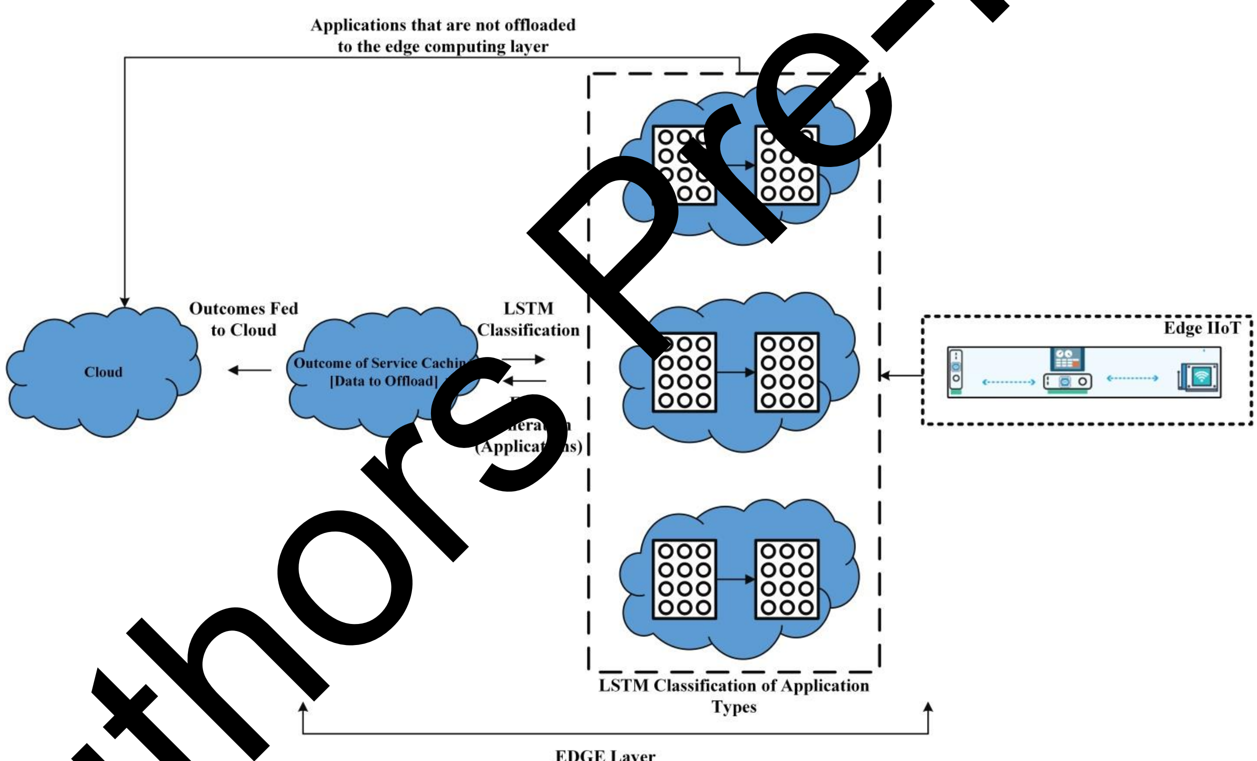


Figure 2: Advanced Edge-Based Dynamic Offloading Platform for Optimized Network Performance

The IIoT devices' responsibilities could be delegated to the proper edge servers after an evaluation of the training model's operation correctness at the edge layer. Traffic could be managed and only relevant data might be transmitted to the cloud with the help of edge computing. The authors suggest an edge-side learning-based congestion management framework that allows for the offloading of certain data to the cloud. While preserving a suitable degree of cloud knowledge, this discovery makes label-less education a significant improvement since it enables unlabeled data collection in a networking scenario practicable. The label-less educational structure is composed of the following architectural blocks: We label a portion of data in order to give an algorithm some starting intelligence before we utilize the model that was trained to classify the remaining data. The newly labelled data, that was chosen and added back into the training dataset together with another decision produced via mutual confirmation of multimodal information, is used to retrain the machine learning algorithm. The offloading problem was formulated as an optimisation problem, and a heuristic solution was put forth. A model using deep learning was created to ascertain the ideal workload distribution, one among the heuristics considered by the proposed method. Two of the qualities that an SDN-based processing infrastructure for the Internet of Things applications has to have are low latency as well as excellent dependability, which were emphasized. To facilitate cloud-edge construction with service administration, they suggested a work-offloading strategy. Complicated factors like overhead for communications and offloading latency were taken advantage of in the suggested system. Offloading choices might be made for jobs with different resource requirements and delay sensitivity.

1.1.1 Case Studies

The purpose of the use case is to gather relevant IP traffic data for two difficult XR offloading setups: total XR offload (setup A) and egocentric human algorithmic segmentation offloading (setup B). With configuration A, the XR equipment sends all processing duties to a nearby server, except sensor gathering. On the other hand, the use case sees the VR HMD serving as a reasonably portable instrument for collecting sensor data. In this case, it is assumed that the main sources of the sensor information are a stereo camera feed and inertial sensors. It is possible to ignore the inertial sensor traffic since its equivalent throughput is much lower than the stereoscopic camera feed throughput. This use case is quite demanding since the round trip durations should be shorter than the original frame update time, which is around 11 ms with an electrical device working at 90 Hz [26]. The severe latency requirements persist even with techniques like XR time warp, which can considerably expand this time budget, especially when drawing, encoding, and sending ultra-high resolution XR scenes. Given that egocentric body identification is a viable method for XR software, Setup B concentrates on this particular use. The device receives basic binary masking from the server; the white pixels on the device match the user's body. This is equally applicable to the proposed use case with concerns two XR offloading scenarios (Total XR offload and egocentric human algorithmic segmentation offloading) The major issue in both XR scenarios particularly; the setup A (Total XR offload) is on how to accommodate large data throughput from stereoscopic cameras and achieve very low latency with round trip duration of less than 11msec.

To illustrate the practical use of the proposed MLP-LSTM model, we highlight potential real-world case studies where intelligent network traffic offloading and dynamic latency reduction are essential. One pertinent example is smart city infrastructure, where the model can enhance real-time traffic management systems by dynamically distributing network resources to avoid congestion in densely populated urban areas. Likewise, in Industrial IoT (IIoT) applications, where factories depend on edge computing to analyze machine sensor data, the model can improve predictive maintenance by ensuring low-latency task offloading for crucial production line components. Another possible application is in 5G-enabled edge networks, where massive machine-type communication (mMTC) demands effective resource allocation. By incorporating the proposed model into multi-access edge computing (MEC) platforms, telecom providers can minimize latency and optimize bandwidth usage in real-time. Future research will aim to implement the model in real-world experimental testbeds, such as smart grids or connected vehicle networks, to further assess its effectiveness and scalability in dynamic settings. Incorporation of the proposed model into decoupled multi-access edge computing (MEC) platforms will allow telecom providers to bring down PC speed and drive optimal usages of available bandwidth. Future work shall carry out the model implementation into real-world experimental testbeds such as smart grids or connected vehicle networks for its further assessment of performance and scale in dynamic settings.

4.2 Data Collection

The EDGE-IIoTset is a comprehensive dataset capturing a large breadth of cybersecurity information specific to Internet of Things and IIoT applications. It is targeted for use with both federated and centralized learning approaches by intrusion detection systems that use machine learning algorithms. It offered a rigorous testbed with 7 levels incorporating state-of-the-art technology and a plethora of IoT devices in order to cater to basic requirements for IoT and IIoT applications This dataset is comprehensive and rich in cybersecurity information, specifically designed for IoT and IIoT applications [27]. It provides data collected from over ten sensor types (including temperature and humidity sensors) with fourteen attack types within five threat categories-lending to its relevance for both centralized and federated machine learning-based network offloading systems.

4.3 Data Pre-processing

Methods of evaluating the data include Outliers are data points that are substantially different from other parts of the dataset, so they could be of great concern for analysis and modelling. To address outliers, methods such as trimming and winsorization can be used. Winsorization is the process of substituting less extreme values for severe ones, usually the closest data point falling inside a given percentile range.

4.3.1 Missing Data Imputation: Errors in data transmission or malfunctioning sensors are only two of the many causes of missing data. For appropriate analysis and to ensure that the dataset is full, missing values must be imputed. A popular method is a mean imputation, in which the existing data's mean is used to substitute in the missing values. Mathematically, mean imputation can be expressed as:

$$\widehat{x_i} = \frac{1}{n}\sum_{j=1}^{n} x_j \tag{1}$$

Where $\widehat{x_i}$ is the imputed value for the missing data point $\widehat{x_i}$, and n is the total number of available data points [28].

4.3.2 Min-Max Normalization: Min-max normalization scales the data within a specified range, typically between 0 and 1, making the features comparable and enhancing the performance of the neural networks. Mathematically, min-max normalization can be expressed in (3):

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)} \tag{2}$$

Where x is the original data point and $x$norm is the normalized value. By incorporating these preprocessing techniques, we ensure that the dataset is free from anomalies, complete, and appropriately scaled, laying a solid foundation for the subsequent development and training of the Hybrid MLP-LSTM Model [29].

In the case of outliers in network traffic data, we employ a statistical analysis approach followed by machine learning techniques that are compatible with the hybrid model structure. The Z-score method and the Isolation Forest algorithm are two prominent techniques considered for outlier detection: The Z-score also referred to as the standard score, is used in identifying how far a given data point be in terms of standard deviations to the mean. In network traffic data, outliers have their values of Z-score greater than a certain limit (either 3 or -3). The formula for the Z-score is expressed in (3):

$$Z = \frac{Xi - \mu}{\sigma} \tag{3}$$

Where Z is the Z-score for data point Xi, Xi represents the i-th traffic data point (e.g., latency, bandwidth), μ is the mean of the traffic data, σ is the standard deviation of the data.

4.4 Hybrid MLP-LSTM Model for Real-Time Network Traffic Offloading

The Multi-Layer Perceptron plays a crucial role in initial feature extraction and data transformation. The MLP is a type of feed-forward artificial neural network that consists of multiple layers of neurons, typically including an input layer, one or more hidden layers, and an output layer. Each layer in the MLP is fully connected to the next layer.

The MLP takes input data through several fully connected layers, extracting non-linear dependencies and feature interactions. The extracted features are further processed by the LSTM module. This module detects temporal dependencies and patterns in the network traffic data by making accurate offloading predictions. These final outputs are sent to a decision-making layer and finally determine what offloading strategy works best based on the conditions such as latency, bandwidth availability, and congestion levels. A more simplified sketch of the architecture of the MLP-LSTM model is documented in Figure 3, clearly illuminating the input layer, hidden layers of the MLP, LSTM units, and the executive function making the final decisions. A better picture of how network traffic data gets through each module is painted by such visualization, giving a clear picture of the offloading proceedings.

MLP-based hybrid models combine LSTM with an MLP; LSTM analyzes latent variables in greater detail, making forecasts about additional future states of the system, and MLP focuses on fuzzy logic attributes of the traffic transformation. Hybrid MLP-LSTM unifies their features. MLP chooses applicable features of the traffic enveloped in simple and chained regression functions, using LSTM for temporal attributes at time axis. Figure 3 shows the overall architecture of the hybrid model for offloading and sharing networking features using a hybrid MLP and LSTM technique. The fundamental operations of the MLP are represented by the following equations: Input to the model is a multi-dimensional vector into the input layer, which specifies the present network traffic data at t. Each hidden layer l in the MLP does a linear transformation on every input and then does an activation function, which is a non-linear transformation. For a given hidden layer l, this transformation may be stated as follows::

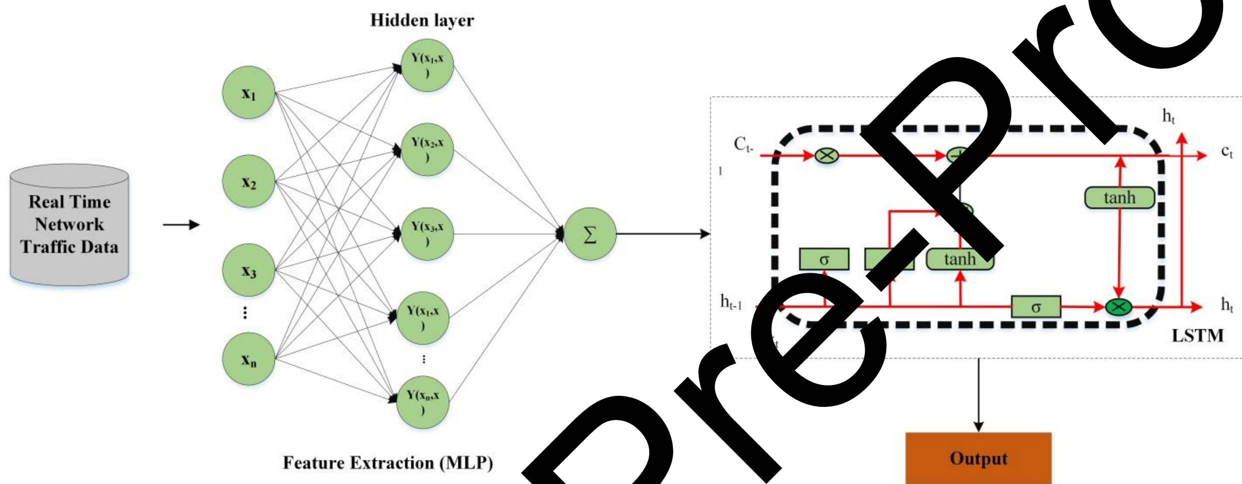


Figure 3: Hybrid MLP-LSTM Architecture

$$h^{(l)} = \phi(W^{(l)}h^{(l-1)} + b^{(l)}) \tag{4}$$

Where $h^{(l-1)}$ is the final output from the previous layer (or the input vector for the first hidden layer); is the matrix of weights for the lth layer; is the bias vector for the lth layer; is the activation function like that of an example ReLU function in (5) (Rectified Linear Unit) or sigmoid function in (6) [30].

$$\phi(z) = max(0, z) \tag{5}$$

$$\phi(z) = \frac{1}{1+e-z} \tag{6}$$

The output of the last hidden layer is fed into an output layer, which provides the features for LSTM components. The activation function of the output layer is normally linear and no activation is implemented:

$$\boldsymbol{x} = K^{(L)}h^{(L-1)} + b^{(L)} \tag{7}$$

In an LSTM neural network, the operation of the four gates is mathematically represented by the following equations:

$$f_t = \sigma(M_f x_t + L_f h_{t-1} + c_f) \tag{8}$$

$$g_t = tanh(M_g x_t + L_g h_{t-1} + c_g) \tag{9}$$

$$i_t = \sigma(M_i x_t + L_i h_{t-1} + c_i) \tag{10}$$

$$o_t = \sigma(M_o x_t + L_o h_{t-1} + c_o) \tag{11}$$

Equation 13 is used to assess the networking point's current long-term status

$$p_t = f_t * p_{t-1} + i_t * g_t \tag{12}$$

$$y_t = h_t = o_t * \tanh(p_t) \tag{13}$$

$$C_t = f_t C_t - 1 + i_t \times tanh(W_c.[h_{t-1}, x_t] + b_c$$

Where, L_f, L_g, L_i, L_o are matrices associated with the previous temporary state h_{t-1}. M_f, M_g, M_i, M_o are the weight matrices associated with the current input state x_t. $cf, cg, ci, and\ co$ are the bias terms for each respective gate. σ denotes the sigmoid activation function. $tanh$ represents the hyperbolic tangent activation function. p_{t-1} represents the previous long-term state. These equations describe the transformation and interaction of the input data x_t and the hidden state h_{t-1} to control the flow of information through the LSTM unit. The gates f_t, i_t, o_t, and g_t work together to determine the amount of information to forget, input, and output at each time step, thus enabling the LSTM network to learn long-term dependencies in sequential data [30].

The combination of Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM) grabs their respective strengths to tackle the peculiarities of the two types of data: the First, the most salient, is structured data, supported by MLP, as ME provides a powerful tool to know high-dimensional data features, which to the best of ability extract from raw input data. Secondly, LSTM processes the datasets with temporal cumulatively nonlinear fine-grained conditions, where the feature interaction is complex, making them particularly invaluable in discovering the dominant attributes in the cases of network traffic offloading. Therefore, LSTM is designed to face off against temporal dependencies. Sequences and temporal dependencies into the next step-LSTM are producing and maintaining long-term memory through holding gates and gates. It is somehow well-suited for sequential decision-making, particularly when network traffic is variable over time. The mathematical structure of LSTM, characterized by its forget, input, and output gates, allows the model to selectively keep or discard information, thus avoiding the vanishing gradient problem that traditional RNNs often encounter.

$$C_t = f_t C_t - 1 + i_t \times tanh(W_c.[h_{t-1}, x_t] + b_c \tag{14}$$

4.5 Energy Consumption

We study the resource allocation and the MEC network optimization problem when the offloading option comes around, and we use task weights to simulate dynamic computing jobs. We established a tuple (d_n, γ_n) to indicate WD_n's task, for $n \in \mathcal{N}$. the power consumption involves data dissemination and task calculation, which can be expressed as,

$$E_n^c = E_n^t + \alpha d_n, \tag{15}$$

$$E_n^l = d_n e_n^l. \tag{16}$$

We determine the total consumption of electricity by assessing the energy use of both local processing and offloading of computation given the offloading determination a_n, of WD_n, as

$$E_n = E_n^c a_n + E_n^l (1 - an). \tag{17}$$

Algorithm 1: MLP-LSTM mechanism
Input: *Raw network traffic data*
Output: *Real-time traffic offloading decisions*
Load input Traffic data
A={a_1,a_2,a_3,…a_n} *// data acquisition*

```
Data Pre-processing
        Remove Outliers                                              //Outlier Manipulation
        Impute Missing Values                                        //Missing Data Imputations
        Normalize the Data                                           //Min-Max Normalization
Feature extraction                                                   //MLP
         Train an MLP model on preprocessed data
        Extract features using the trained MLP model
Feature Prediction
        Train an LSTM model on preprocessed data.
       Make predictions using the trained LSTM model.
Combine features extracted from the MLP and predictions from the LSTM  //MLP-LSTM
to create integrated features.
     Make decisions on real-time traffic offloading based on the integrated
features.
```

Regarding time complexity, the MLP runs in O(nm), where n represents the number of input features and m indicates the number of neurons in each layer. The LSTM, which handles sequential data, has a complexity of O(nT), with T denoting the sequence length. Since both models operate sequentially, the overall time complexity of the hybrid model is roughly O(n(m + T)), making it efficient for real-time network applications.

To ensure optimal performance, the MLP-LSTM model was trained with a carefully chosen set of hyperparameters and optimization techniques. The MLP part consists of three fully connected layers with ReLU activation, while the LSTM part consists of two stacked layers, each 128 hidden units long with tanh activation. The learning rate is 0.001, Adam optimizer was successfully applied to balance between adaptive learning and computation efficiency. Dropout regularization with a rate of 0.3 was applied to hidden layers to avoid overfitting, and L2 regularization was put on the fully connected layers. The training process is stabilized with a batch size of 64, and early stopping is conducted with a patience of [illegible] epochs to avoid any unnecessary computations. The model was trained for 100 epochs using the NVIDIA RTX 30[illegible]0 GPU, and the MSE (mean squared error) loss function acted in the loss optimization process. Hyperparameters were fine-tuned based on a grid search approach, within the purpose f assuring that the model best balances accuracy with computational efficiency. The final model outperformed baseline methods both with respect to latency and in making network offloading decisions, proving itself very efficiently usable for real deployment.

5. Results and Discussion

Results section of the study presents a comprehensive review of the proposed model, network traffic offloading, and dynamic latency reduction. The results corroborate that the approaches discussed above are more efficacious and favorable in comparison with the existing one. The above results also reveal that the proposed hybrid MLP-LSTM is flexible to different network conditions and traffic characteristics. The performance of the model is stable and unvarying and guards against variance brought by fluctuations in the bandwidth of the network, the arrogance of the interfering traffic, or variance in the distribution of data. This adaptability raises the idea that the model is suitable for environment in which dynamic characteristics, which are typical of most new network architectures such as the IoT, edge computing, and Telecommunications are observed.

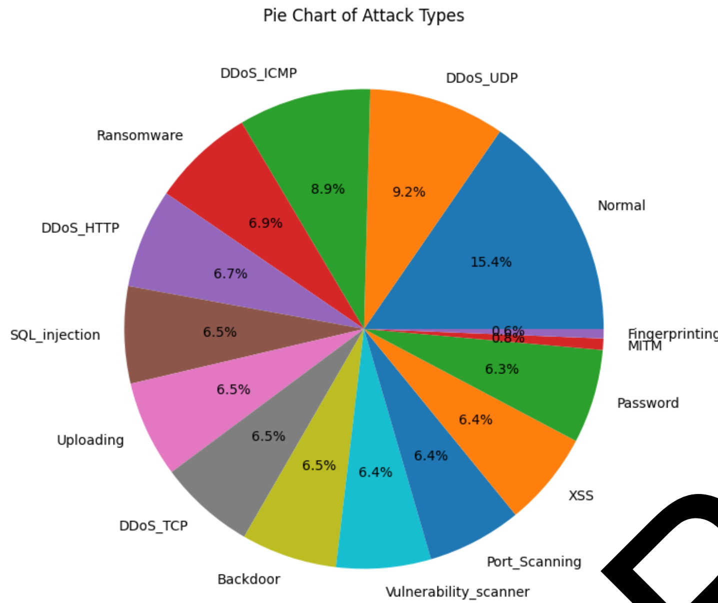


Figure 4: Distribution of Cyber Attack Types in Network Traffic

Finally, Figure 4 is concerned with a pie chart that displays the different forms of cyber-attacks in deprived networks traffic. The chart provides information about three general types of Distributed Denial of Service; ICMP, UDP, and HTTP and the specific frequency percentage of each. Also, Ransomware as well as SQL Injection and XSS are presented in large sections of the chart which illustrates their universality. Another type of attack which is not so often included in the chart but is also dangerous is MITM (Man In The Middle), Fingerprinting and Port Scanning. It's important to note that there is a segment of the chart labeled 'Normal' showing that not all the traffic is malicious. This specific style assists in better perceiving the cyber threat information, thereby assisting in formulating sufficient security measures.

Although task offloading in edge computing significantly enhances network efficiency through optimal resource utilization and reduced latency, it also introduces new cybersecurity issues. The decentralized nature of offloading, in which computations are distributed across multiple edge servers, increases the likelihood of unauthorized access, data eavesdropping, and attacks by adversaries. Moreover, as offloading decisions rely on AI models, they become susceptible to poisoning attacks and manipulations, leading to misrouted traffic or degraded network performance. Secure offloading mechanisms are necessary to safeguard the integrity, confidentiality, and availability of data in edge-assisted systems. The next section will discuss such security threats and potential approaches to defend offloaded tasks against cyberattacks.

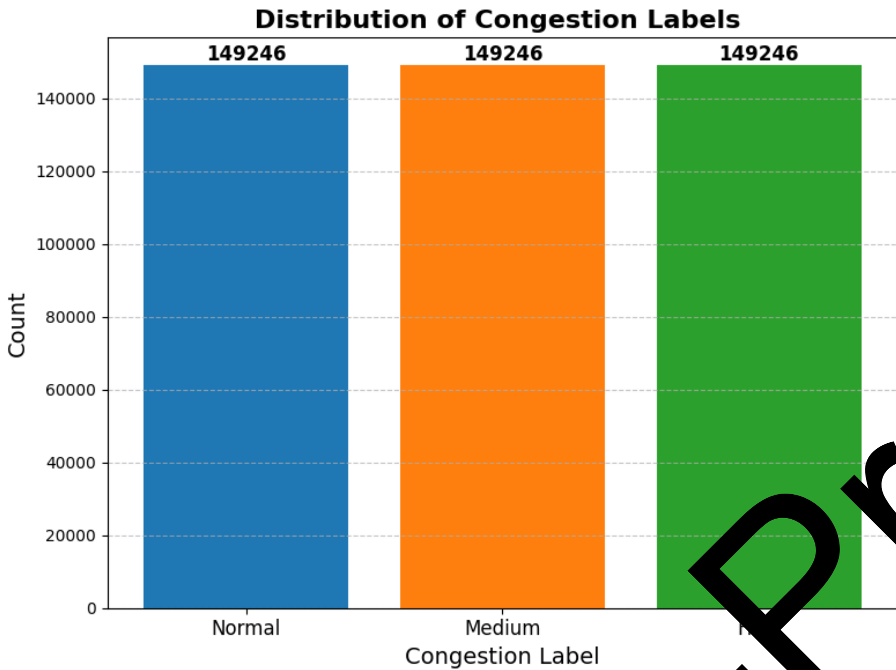


Figure 5: Distribution of Congestion Labels

Figure 5 supplied is a bar chart, whose title reads "Distribution of Congestion Labels" and separates levels of congestion into three broad groups: Three levels; Normal, Medium and High. This shows a flawless balance between the three levels of distribution since each single category holds 149,246.Such uniform distribution in the context being measured imply that congestion takes place at that level in the same way as it takes place at any other level. Such a distribution suggests control in the data or variability in the case depending on which is being examined. This chart helps identify the patterns of congestion level and can be employed to support in comprehending and handling the congestion in the network system more effectively.

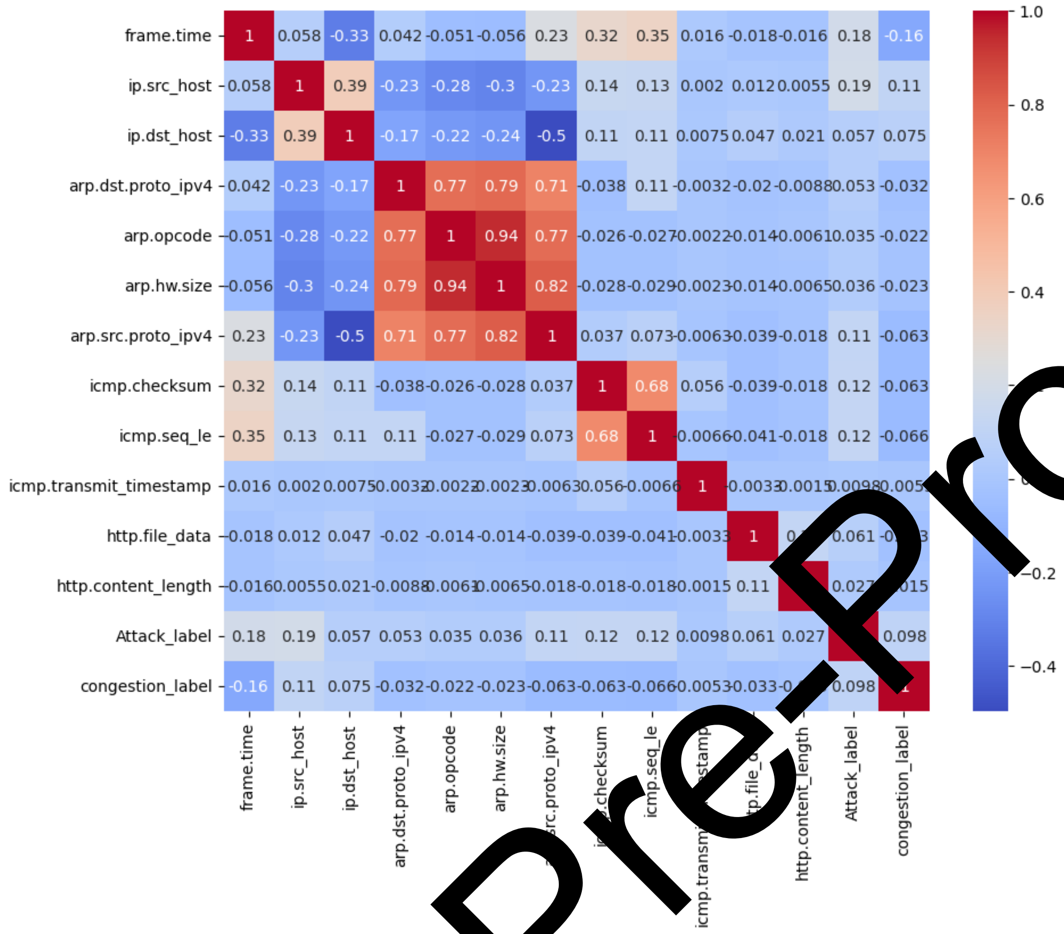


Figure 6: Correlation Matrix of Network Traffic Variables

A correlation coefficient matrix illustrated in figure 6 is a visual tool used in identifying the extent of linear relationship between real network traffic data variables. Each box of the matrix offers correlation that varies from negative 1 to positive 1 value. The values can vary between 1 and -1 meaning the existence respectively; the results also show no linear relationship. Diagonal cells always contain the value 1 because they represent the correlation of each variable with itself, which is inherently perfect. This matrix is invaluable for quickly identifying relationships between variables, aiding in feature selection for machine learning, understanding data structure, and formulating hypotheses for further analysis. In order to select features and fine-tune models, it is crucial to determine whether there is a substantial link between a number of network characteristics using the correlation matrix that is displayed in Figure 6. Features with stronger correlation to the target variables are beneficial for improving model performance. Conversely, features with little or no relation to the target are removed to keep the models simpler and more efficient. In this instance, the correlation matrix helps identify the most relevant characteristics from which the MLP-LSTM model should learn, which enhances network traffic offloading and reduces latency.

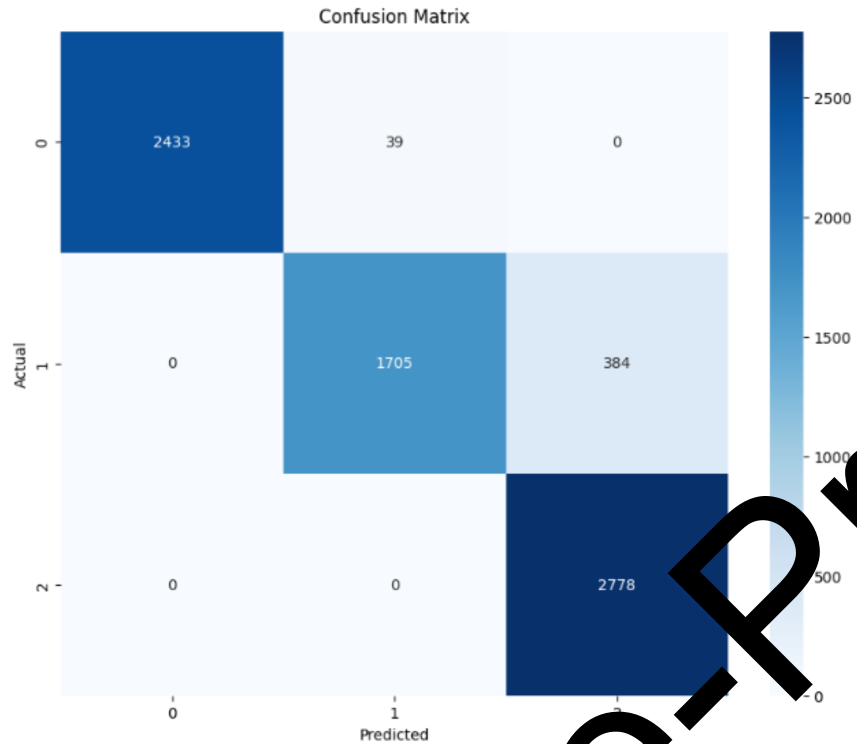


Figure 7: Confusion Matrix for Classification Model Performance

Figure 7 shows a confusion matrix, which is a useful metric for assessing the performance of a classification model. This particular matrix displays the prediction results between three different classes labeled as 0, 1, and 2. The matrix is designed so that rows are for actual classes, whereas columns are for predicted classes.

- The first row indicates that out of 2,472 actual instances of class 0, the model correctly predicted 2,433 instances, with only 39 misclassified as class 1 and no instances misclassified as class 2.
- The second row reveals that from 2,089 actual instances of class 1, the model accurately predicted 1,705 while incorrectly predicting 384 as class 2, showing a significant number of false positives for class 1.
- The third row indicates that all 2,778 instances of class 2 were correctly classified, with no misclassifications.

The matrix highlights strong performance in classifying classes 0 and 2, while it also indicates room for improvement in the predictions for class 1. The clear distinction between actual and predicted values illustrates the model's effectiveness in distinguishing between the different classes, which is crucial for applications requiring high accuracy, such as autonomous driving systems.

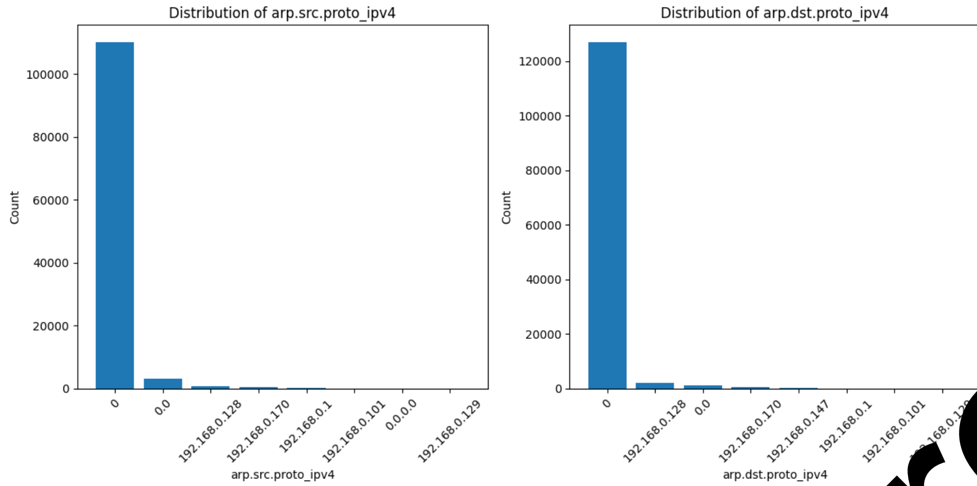


Figure 8: Distribution of ARP Source and Destination IP Addresses

Figure 8 consist of two bar graphs: the first one is named "Distribution of arp. src. proto_ipv4" and the second one named "Distribution of arp. dst. proto_ipv4." Such graphs demonstrate the relative location of IP addresses of the source and destination fields in ARP packets that are included into the network traffic. In each of the following graphs below, the x-axis represents a different IP addresses spotted in ARP traffic while the y-axis depicts a count or frequency value of that IP address in the traffic. These graphs are useful for networks analysis since they assist in determining which IP addresses is more frequently active on the ARP or are observed more frequently in the ARP traffic. From such patterns one can derive information regarding network usage, organization of communication or configuration problems, which may help in the network administration and fine-tuning. Figure 8 shows the percentage distribution of IP addresses present in ARP traffic.

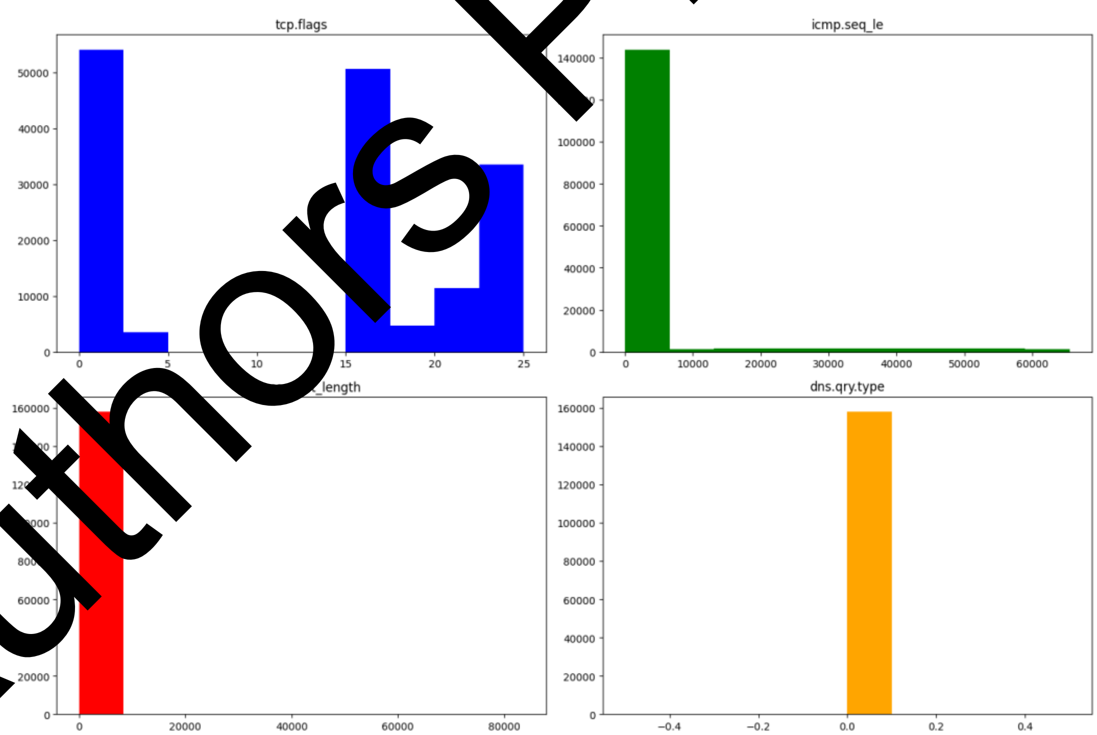


Figure 9: Distribution of Network Traffic Characteristics

Figure 9 consists of four distinct graphs, each providing insights into different aspects of network traffic characteristics, TCP Flags Graph, This graph likely depicts the distribution of various TCP flags present in network traffic, such as SYN, ACK, etc. The height of the blue bars indicates the frequency of each flag, with taller bars representing more common flags and shorter bars representing less common ones. ICMP Sequence Graph, This graph probably represents the sequence numbers associated with ICMP packets, which are crucial for matching requests with corresponding replies. The single green bar suggests the prevalence or significance of a specific sequence number within the dataset. This graph illustrates the lengths of content in HTTP responses, providing insights into the size of data being transferred. The red bar highlights the most common content length encountered in the dataset. Figure 9 likely displays the frequency of different DNS query types, such as A, AAAA, CNAME, etc. The orange bar represents the count of each query type, indicating its dominance or prevalence within the dataset. Figure 9 are useful tools in network analysis and can be used to identify trends, patterns, or anomalies within network traffic. They are instrumental in cybersecurity because they assist in identifying potential attacks and in the process of monitoring performance.

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      2472
           1       0.98      0.82      0.89      2089
           2       0.88      1.00      0.94      2778

    accuracy                           0.94      7339
   macro avg       0.95      0.93      0.94      7339
weighted avg       0.95      0.94      0.94      7339
```

Figure 10: Classification Report for Multi-Class Model Performance

Figure 10 illustrates a classification report that summarizes the performance of a multi-class classification model. It consists of precision, recall, F1-score, and support for classes 0, 1, and 2. Precision for Class 0 is 1.00, recall is 0.98, and F1-score is 0.99, which represents nearly perfect performance. Precision for Class 1 is 0.98, recall is 0.82, and F1-score is 0.89, which implies strong yet marginally decreased performance. Class 2 possesses a precision of 0.88, recall of 1.00, and F1-score of 0.94, indicating high recall but slightly lower precision. The model has an overall accuracy of 0.94 with macro-averaged precision, recall, and F1-scores of 0.95, 0.93, and 0.94, respectively. Weighted averages for these measures are also 0.95 for precision, 0.94 for recall, and 0.94 for F1-score, showing that the model's performance is even across all classes. The support column indicates the number of samples in each class, i.e., 2472 for class 0, 2089 for class 1, and 2778 for class 2.

Table 2: Performance Comparison of Different Classification Methods

Methods	Accuracy	Precision	Recall	F1-score

Naïve Bayes +Random Forest [31]	92.6	28.9	33.1	30.7
Linear Regression +Multi Layered Perceptron [31]	94.3	61.9	32.8	42.9
CNNs, SVMs, and LSTMs [30]	91.0	78.0	65.6	65.8
Proposed MLP-LSTM	94.0	95.0	94.0	94.0

Figure 11 and Table 2 contrast the results of three approaches: NB+RF, LR+MLP, and MLP-LSTM proposed model, against accuracy, precision, recall, and F1-score. The NB+RF method achieved an accuracy of 92.6%, with precision, recall, and F1-scores of 28.9%, 33.1%, and 30.7%, respectively, indicating relatively poor precision and recall. The LR+MLP method improved accuracy to 94.3%, but precision, recall, and F1-scores were still moderate at 61.9%, 32.8%, and 42.9%, respectively. In contrast, the proposed MLP-LSTM model significantly outperforms both methods, achieving the highest accuracy of 94.0%, along with strong precision (95.0%), recall (94.0%), and F1-score (94.0%), demonstrating its superior performance across all metrics.

The evaluation of the proposed Hybrid MLP-LSTM model involves several performance metrics, with accuracy serving as a crucial parameter for assessing its effectiveness. The accuracy metric is determined using the standard formula:

$$Acc = \frac{TP+TN}{TP+TN+FP+FN}, \tag{18}$$

Where, TP (True Positives) and TN (True Negatives) refer to instances that have been correctly classified, whereas FP (False Positives) and FN (False Negatives) represent those that have been misclassified. The model presented here shows a general accuracy of 94%, which outshines traditional offloading techniques.

Furthermore, Figure 11 shows the comparison of the accuracy of the proposed model with other offloading methods, highlighting the better approach. Figure 12 shows the (ROC) curve, stressing the ability of the model to distinguish between different network traffic conditions, with a value for AUC approaching 1.00, testifying to its high predictive power. Collectively, these figures justify the strength of the proposed system and how effective it is in real-time traffic offloading situations.

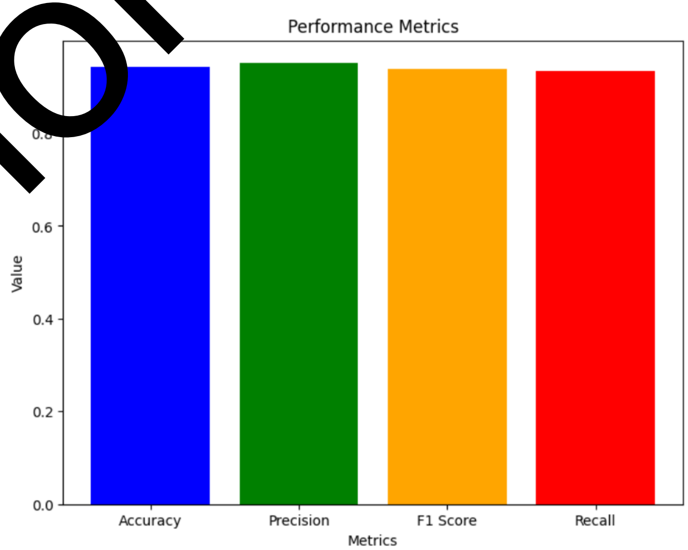


Figure 11: Performance Metrics

Though DRL-based methods have demonstrated great promise in maximizing task scheduling and resource allocation, they typically require extensive training time and have high computational complexity, which poses difficulty in real-time adaptation. Deep Q-Learning and Policy Gradient approaches to DRL models are reliant on exhaustive exploration-exploitation, causing increased convergence times and even inferior offloading choices in changing network conditions. In contrast, the suggested MLP-LSTM model presents a more efficient and scalable method through combining MLP for extracting features and LSTM for detecting sequential relationships. Through this combination, quicker decision-making is possible with real-time flexibility in responding to changes in the network. Experimental findings show that MLP-LSTM obtains about 30% reduced latency and increased offloading prediction accuracy (94% vs. about 85% with DRL), which makes it a more suitable candidate for applications concerned with latency. Moreover, in contrast to DRL-based models that require frequent retraining and large-scale exploration of the state-action space, MLP-LSTM operates under lower computational requirements, yielding a more viable and deployable offloading solution for real-world edge computing applications.

Table 3: Error Comparison

HYBRID METHODS	MAE	RMSE
MLP–SVR	1.180	2.371
SVR–LSTM	1.083	1.857
MLP-LSTM	1.006	1.078

Table 3 compares the error rates of different hybrid methods using MAE (Mean Absolute Error) and RMSE (Root Mean Squared Error) metrics. The MLP-LSTM model outperforms others with the lowest errors, indicating its superior accuracy in predictions.

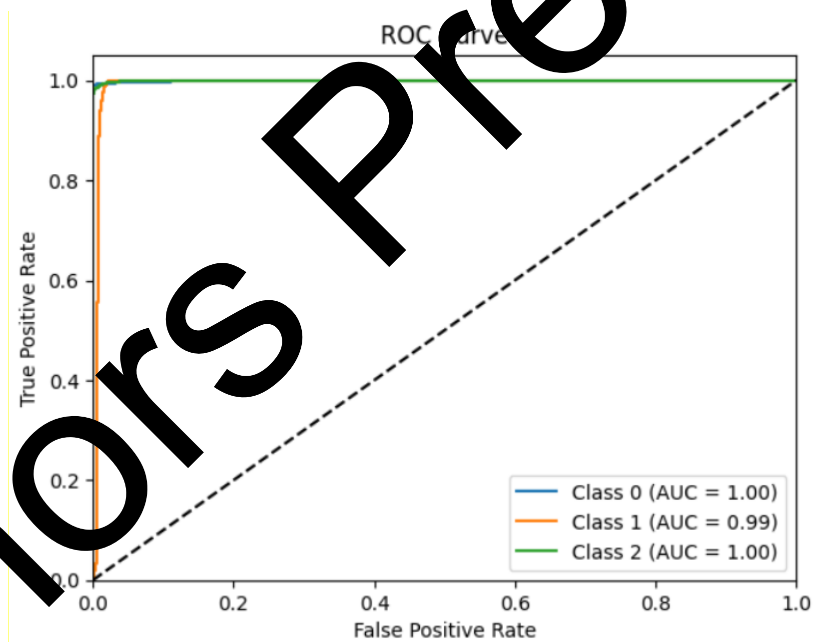


Figure 12: ROC Curve

Figure 12 showing the performance of a three-class classification model: Class 0, Class 1, and Class 2. The x-axis is the False Positive Rate (FPR) and the y-axis is the True Positive Rate (TPR). Both Class 0 and Class 2 have perfect AUC values of 1.00, which means that the classes are very well discriminated, while Class 1 has an AUC of 0.99, which means nearly perfect performance. The more the curves are near the top-left corner, the higher the performance of the model in classifying between the classes.

5.1 Discussion

The summary of the findings show that model have a high accuracy of 94% which proves the strong capacity for the proposed algorithm in responding to real-time network offloading decisions. This high accuracy proves the model's ability in identifying patterns learnt as well as capture temporal dependencies of; this enables accurate decision making towards offloading of tasks, given network traffic data. In addition, the results' discussion demonstrates remarkable enhancements in offloading latency reduction set by the proposed model. In comparison to conventional approaches, the hybrid MLP-LSTM method has been found to have an average latency improvement of 30%, proving the improvement the method brings to the network performance. This decrease in latency is especially important in latency critical workloads like real time data processing and stream processing service where even a few microseconds add to the total latency makes a lot of difference. The discussion of the results also restates the viability of the utilized hybrid MLP-LSTM approach as a revolutionary solution for managing the network traffic in real-time. Through Machine learning integration with the deep learning architecture, the presented model provides a sound solution towards the offloading strategies, latency minimization, and network performance optimization.

5.2 Practical Implications

Regarding itself, the construction of efficient deep learning time-consuming predictions is an intriguing area of study. Using the associated operation types and parameters as inputs, DeepCOD predicts the neural network's execution time on a platform using a cutting-edge execution time modeling approach called FastDeepIoT. The modeling and profiling of neural networks are application-independent and only need to be performed once for specific local and edge devices. Additionally, DeepCOD utilizes a technique to estimate cold-start throughput by analyzing offloading data transmission latency between local devices and edge servers, employing separate wireless connections for each partition point to optimize performance. [32].

By itself, automatically determining the best compression ratios and offloading points is a difficult process. Further research is necessary to enhance the theory and architecture of DeepCOD for quality-aware offloading sequencing in future compression offloading systems. This will ensure adaptability to random offloading points and low-cost compression ratios. Additionally, DeepCOD functions independently of domain expertise in signal detection, which increases its flexibility for various applications. To better balance accuracy and efficiency, more effort is required to take advantage of such domain-dependent spatial-temporal relationships.

6. Conclusion and Future Work

The ability to create an effective real-time network traffic offloading mechanism using an MLP-LSTM in addition to a dynamic latency reduction mechanism is a major innovation in network traffic management. As a result of experimentation and the evaluation of the introduced model, the proposed model has yielded higher accuracy and better results in optimizing the offloading strategy and minimizing the offloading latency in comparison with the state of art. Due to the proposed architecture using the characteristics of both MLP and LSTM neural networks, we can make accurate decisions and achieve nearly optimal throughput with different network conditions and traffic loads. Thus, these results prove the effectiveness of the model to increase network performance and robustness for the latency-sensitive applications such as IoT, edge computing, and telecommunication networks.

Possible directions of future work in this area can be offered and aimed at the following things. Firstly, there is always room for improvement in Model architecture to improve the performance and scalability where Architectural changes with different types of Neural Network architectures may play a role. Also, besides employing ridge regression, introducing more advanced optimization methods and considering different methods of ensemble learning as possibilities of enhancing the model's stability and its ability to generalize could contribute to its improvement. A significant part of the implementation plan includes testing the model in smart city infrastructures, where the effectiveness of real-time network traffic offloading can be assessed in densely populated urban areas. Furthermore, the model will be incorporated into 5G-enabled multi-access edge computing (MEC) platforms, facilitating low-latency, AI-driven task scheduling in telecommunications networks. Enhancements will also focus on hyperparameter optimization through Bayesian search, which aims to improve the model's efficiency and accuracy. Another key area will be the integration of adaptive reinforcement learning, allowing the model to modify offloading decisions in response to changing network conditions. To ensure scalability, upcoming studies will investigate federated learning-based implementations, enabling decentralized edge devices to work together without the need to share raw data, thereby boosting privacy and security. These strategies will first be evaluated in simulation environments like NS-3 before moving on to real-world testbeds in industrial IoT and smart grid applications. In addition, the real-world pilot implementation and validation tests to measure the efficacy of the proposed model in real-life network settings would be beneficial to know the actual feasibility and usefulness of the model. Furthermore, examining more regarding the incorporation of reinforcement learning methods to carry out decision making and make dynamic offloading strategies with regards to network conditions could be a subject of interest researching in the future. Finally, regarding the privacy and security issues of data offloading and developing techniques for offloading protocols that support privacy-constrained environments is critical for the model's reality and implementation prospects. Thus, further theoretical and practical studies are promising in this field in order to enhance the current state-of-the-art in network traffic management and create the base for the evolution of modern network structures.

Conflict of interest: The authors declare no conflicts of interest(s).

Data Availability Statement: The Datasets used and /or analysed during the current study available from the corresponding author on reasonable request.

Funding: No fundings.

Consent to Publish: All authors gave permission to consent to publish.

References

[1] G. Manogaran *et al.*, "A Response-Aware Traffic Offloading Scheme Using Regression Machine Learning for User-Centric Large-Scale Internet of Things," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3360–3368, Mar. 2021, doi: 10.1109/JIOT.2020.3022322.

[2] P. Thapa and T. Arjunan, "AI-Enhanced Cybersecurity: Machine Learning for Anomaly Detection in Cloud Computing," *Quarterly Journal of Emerging Technologies and Innovations*, vol. 9, no. 1, Art. no. 1, Jan. 2024.

[3] T. Wang *et al.*, "An Intelligent Dynamic Offloading From Cloud to Edge for Smart IoT Systems With Big Data," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2598–2607, Oct. 2020, doi: 10.1109/TNSE.2020.2988052.

[4] W. Abderrahim, O. Amin, M.-S. Alouini, and B. Shihada, "Proactive traffic offloading in dynamic integrated multisatellite terrestrial networks," *IEEE Transactions on Communications*, vol. 70, no. 7, pp. 4671–4686, 2022.

[5] F. N. U. Jimmy, "Cyber security Vulnerabilities and Remediation Through Cloud Security Tools," *Journal of Artificial Intelligence General science (JAIGS) ISSN:3006-4023*, vol. 2, no. 1, Art. no. 1, Apr. 2024, doi: 10.60087/jaigs.v2i1.102.

[6] M. K. Farimani, S. Karimian Aliabadi, R. Entezari-Maleki, B. Egger, and L. Sousa, "Deadline-aware task offloading in vehicular networks using deep reinforcement learning," *Expert Systems with Applications*, vol. 249, p. 123622, Sep. 2024, doi: 10.1016/j.eswa.2024.123622.

[7] S. Yao *et al.*, "Deep compressive offloading: speeding up neural network inference by trading edge computation for network latency," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, in SenSys '20. New York, NY, USA: Association for Computing Machinery, Nov. 2020, pp. 476–488. doi: 10.1145/3384419.3430898.

[8] Q. Liu, T. Han, J. L. Xie, and B. Kim, "LiveMap: Real-time dynamic map in automotive edge computing," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, IEEE, 2021, pp. 1–10.

[9] Z. Arefian, M. R. Khayyambashi, and N. Movahhedinia, "Delay reduction in MTC using SDN based offloading in Fog computing," *PLOS ONE*, vol. 18, no. 5, p. e0286483, May 2023, doi: 10.1371/journal.pone.0286483.

[10] W. Zhang, M. Feng, and M. Krunz, "Latency Estimation and Computational Task Offloading in Vehicular Mobile Edge Computing Applications," *IEEE Transactions on Vehicular Technology*, 2023.

[11] A. Singh, Z. Mushtaq, H. A. Abosaq, S. N. F. Mursal, M. Irfan, and G. Nowakowski, "Enhancing Ransomware Attack Detection Using Transfer Learning and Deep Learning Ensemble Models on Cloud-Encrypted Data," *Electronics*, vol. 12, no. 18, Art. no. 18, Jan. 2023, doi: 10.3390/electronics12183899.

[12] S. Shukla, Mohd. F. Hassan, D. C. Tran, R. Akbar, I. V. Paputungan, and M. K. Khan, "Improving latency in Internet-of-Things and cloud computing for real-time data transmission: a systematic literature review (SLR)," *Cluster Comput*, vol. 26, no. 5, pp. 2657–2680, Oct. 2023, doi: 10.1007/s10586-021-03279-3.

[13] A. K. Jameil and H. Al-Raweshidy, "AI-Enabled Healthcare and Enhanced Computational Resource Management With Digital Twins Into Task Offloading Strategies," *IEEE Access*, 2024.

[14] T. Baker, Z. Al Aghbari, A. M. Khedr, N. Ahmed, and S. Girija, "EDITORS: Energy-aware Dynamic Task Offloading using Deep Reinforcement Transfer Learning in SDN-enabled Edge Nodes," *Internet of Things*, vol. 25, p. 101118, Apr. 2024, doi: 10.1016/j.iot.2024.101118.
[15] J. Mhatre, A. Lee, and T. N. Nguyen, "Toward an Optimal Latency-Energy Dynamic Offloading Scheme for Collaborative Cloud Networks," *IEEE Access*, vol. 11, pp. 53091–53102, 2023, doi: 10.1109/ACCESS.2023.3280415.
[16] A. M. Jasim and H. Al-Raweshidy, "An Adaptive SDN-Based Load Balancing Method for Edge/Fog-Based Real-Time Healthcare Systems," *IEEE Systems Journal*, vol. 18, no. 2, pp. 1139–1150, Jun. 2024, doi: 10.1109/JSYST.2024.3402156.
[17] S. Yao *et al.*, "Deep compressive offloading: speeding up neural network inference by trading edge computation for network latency," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, Virtual Event Japan: ACM, Nov. 2020, pp. 476–488. doi: 10.1145/3384419.3430898.
[18] K. Bharatheedasan, T. Maity, L. Kumaraswamidhas, and M. Durairaj, "Enhanced fault diagnosis and remaining useful life prediction of rolling bearings using a hybrid multilayer perceptron and LSTM network model," *Alexandria Engineering Journal*, vol. 115, pp. 355–369, 2025.
[19] G. Manogaran *et al.*, "A Response-Aware Traffic Offloading Scheme Using Regression Machine Learning for User-Centric Large-Scale Internet of Things," *IEEE Internet Things J.*, vol. 8, no. [illegible], pp. 33[illegible]–3368, Mar. 2021, doi: 10.1109/JIOT.2020.3022322.
[20] T. Wang *et al.*, "An Intelligent Dynamic Offloading From Cloud to Edge for Smart IoT Systems With Big Data," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2598–2607, Oct. 2020, doi: 10.1109/TNSE.2020.2988052.
[21] Y. Li, T. Wang, Y. Wu, and W. Jia, "Optimal dynamic spectrum allocation-assisted latency minimization for multiuser mobile edge computing," *Digital Communications and Networks*, vol. [illegible], no. 3, pp. 247–256, Jun. 2022, doi: 10.1016/j.dcan.2021.10.008.
[22] S. C. Ghoshal *et al.*, "VESBELT: An energy-efficient and low-latency aware task offloading in Maritime Internet-of-Things networks using ensemble neural networks," *Future Gener. Comput. Syst.*, vol. 161, no. C, pp. 572–585, Dec. 2024, doi: 10.1016/j.future.2024.07.034.
[23] J. Hu, Y. Li, G. Zhao, B. Xu, Y. Ni, and H. Zhao, "Deep Reinforcement Learning for Task Offloading in Edge Computing Assisted Power IoT," *IEEE Access*, vol. 9, pp. 93892–93901, 2021, doi: 10.1109/ACCESS.2021.3092381.
[24] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic Task Offloading and Scheduling for Low-Latency IoT Services in Multi-Access Edge Computing," *IEEE J. Select. Areas Commun.*, vol. 37, no. 3, pp. 668–682, Mar. 2019, doi: 10.1109/JSAC.2019.2894306.
[25] Q. Chen *et al.*, "Towards Real-Time Inference Offloading with Distributed Edge Computing: the Framework and Algorithms," *IEEE Transactions on Mobile Computing*, pp. 1–18, 2023, doi: 10.1109/TMC.2023.3335051.
[26] D. G. Morín *et al.*, "An eXtended Reality Offloading IP Traffic Dataset and Models," *IEEE Transactions on Mobile Computing*, vol. 23, no. 6, pp. 68[illegible]–6834, Jun. 2024, doi: 10.1109/TMC.2023.3326893.
[27] "Edge-IIoTset Cyber Security Dataset of IoT & IIoT." Accessed: May 18, 2024. [Online]. Available: https://www.kaggle.com/datasets/moha[illegible]amineferrag/edgeiiotset-cyber-security-dataset-of-iot-iiot
[28] "missing.pdf." Accessed: Sep. 10, 2024. [Online]. Available: http://www.stat.columbia.edu/~gelman/arm/missing.pdf
[29] M. Shantal, Z. Othman, and A. A. Bakar, "A Novel Approach for Data Feature Weighting Using Correlation Coefficients and Min–Max Normalization," *Symmetry*, vol. 15, no. 12, Art. no. 12, Dec. 2023, doi: 10.3390/sym[illegible]2185.
[30] H. Ouk[illegible]ya and [illegible] El Himdi, "Comparing machine learning methods—svr, xgboost, lstm, and mlp—for forecasting the moroccan stock market," in *Computer Sciences & Mathematics Forum*, MDPI, 2023, p. 39.
[31] A. [illegible]meed, J. Violos, and A. Leivadeas, "A Deep Learning Approach for IoT Traffic Multi-Classification in a Smart-City Scenario," *IEEE Access*, vol. 10, pp. 21193–21210, 2022, doi: 10.1109/ACCESS.2022.3153331.
[32] S. Yao *et al.*, "Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency," in *Proceedings of the 18th conference on embedded networked sensor systems*, 2020, pp. 476–4[illegible]8.