Optimizing Network Slice Classification Using Federated Learning and Hybrid Fusion with knowledge Distillation approach

by

Asif Iqbal Khan Nuhash
20201189
Sajib Ghosh
20201205
Rubaiya Islam
20201188
Asif Uddin Ahmed
20201062
Md. Rakibul Islam
20301206

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
June, 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Asif Iqbal Khan Nuhash

20201189

Sajib Ghosh

20201205

Rubaiya Islam

20201188

Asif Uddin Ahmed

20201062

Md. Rakibul Islam

20301206

Approval

The thesis/project titled "Optimizing Network Slice Classification Using Federated Learning and Hybrid Fusion with knowledge Distillation approach" submitted by

1. Asif Iqbal Khan Nuhash (20201189)
2. Sajib Ghosh (20201205)
3. Rubaiya Islam (20201188)
4. Asif Uddin Ahmed (20201062)
5. Md. Rakibul Islam (20301206)

Of Spring, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on June 20, 2025.

Examining Committee:

Supervisor:
(Member)

Dr. Amitabha Chakrabarty

Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Dr. MD Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi

Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

With the advancement of 5G and the anticipation of 6G networks, dynamic and efficient network slicing has become crucial for meeting diverse Quality of Service (QoS) requirements across IoT applications. This study proposes a multi-stage framework that combines traditional machine learning, federated learning (FL), and knowledge distillation to classify network slices—eMBB, URLLC, and mMTC—using real-world traffic features. The approach begins with a federated learning model trained across decentralized, non-IID client datasets using average, max, and min aggregation strategies to ensure privacy and robustness. To enhance performance and reduce model complexity, an XGBoost teacher model generates soft labels and structural features which are transferred to a student neural network via knowledge distillation and feature fusion. This fusion model achieves a classification accuracy of 98%, outperforming conventional classifiers and the FL models in isolation. The methodology offers a scalable, privacy-preserving solution for real-time slice classification in next-generation mobile networks, making it particularly suitable for latency-sensitive and resource-constrained edge environments.

Keywords: 5G/6G, Network Slicing, Internet of Things, Federated Learning, Knowledge Distillation, Feature Fusion, Privacy Preservation, Neural Networks, XGBoost, Real-Time Classification

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

5G Fifth Generation Wireless Network

6G Sixth Generation Wireless Network

DL Deep Learning

eMBB Enhanced Mobile Broadband

FL Federated Learning

ML Machine Learning

mMTC Massive Machine-Type Communications

QoS Quality of Service

URLLC Ultra-Reliable Low-Latency Communications

Chapter 1

Introduction

The evolution of telecommunication technologies from 5G to the anticipated 6G era presents a transformative shift in how mobile networks operate, with network slicing emerging as one of the most critical technologies enabling this revolution. Network slicing allows for the creation of multiple virtual networks, each optimized for specific applications, while sharing the same physical infrastructure. As the demand for increasingly specialized and flexible networks rises, network slicing becomes indispensable for managing the complex requirements of diverse services, ranging from high-speed broadband to ultra-reliable low-latency communications for critical applications like autonomous vehicles and telemedicine. However, the complexity of managing and efficiently allocating resources across these slices poses significant challenges. Traditional resource management techniques often fail to handle the dynamic and diverse needs of next-generation mobile networks, necessitating the adoption of advanced methods that can provide real-time, intelligent decision-making. This requirement for more intelligent resource allocations, especially as the demand for data-heavy services increases, is motivating the study of machine learning (ML) and deep learning (DL) based models to enhance the network performance and to classify network slices. By using cutting-edge techniques, for example, federated learning (FL) for decentralised training and knowledge distillation for efficient model performance, this work endeavours to fill the void of the existing network slicing management scheme. In this work, we highlight the improvement in the prediction accuracy of network slice types (eMBB, URLLC, and mMTC) through the use of a hybrid ML and DL approach along with XAI methodologies to gain insight and visibility into the model predictions. These advancements are expected to significantly improve the scalability, privacy-preservation, and efficiency of network slicing in both current 5G systems and the forthcoming 6G networks, pushing the boundaries of what is achievable in wireless communication.

1.1 Motivation

The growing complexity and heterogeneity of next-generation mobile networks—especially in the context of 5G and the forthcoming 6G—demand a more intelligent, adaptive, and privacy-preserving approach to network slice classification and resource allocation. Although existing studies have employed centralized machine learning (ML) models such as XGBoost, SVM, and CNN-LSTM hybrids for classifying network slices, they fall short in addressing key real-world challenges like data privacy, non-

IID data distribution, and computational scalability [28][17][32].

Centralized models require raw data to be aggregated at a central location, which raises critical privacy concerns and limits their deployment in federated or distributed systems. Moreover, the synthetic or IID datasets used in many prior studies do not reflect the real-world conditions of heterogeneous user behavior, device capabilities, or network contexts [17][15][27]. Consequently, models trained in such environments often lack generalizability and perform poorly when applied to actual 5G/6G scenarios.

Furthermore, while federated learning (FL) has emerged as a promising solution for decentralized model training, few works integrate FL with knowledge distillation—a technique known to reduce model complexity without sacrificing accuracy. Combining these approaches can create lightweight models suitable for resource-constrained edge devices, which is a critical requirement for scalable 6G systems [12][27]. Additionally, explainability remains underexplored in most existing works, despite being crucial for adoption by telecom operators [33][22].

Therefore, this research is motivated by the need to address these gaps—specifically, by developing a hybrid ML-DL framework that integrates federated learning, feature fusion, and knowledge distillation to create an accurate, interpretable, and privacy-preserving system for real-time network slice classification in 5G/6G networks.

1.2 Research Objectives:

The primary objective of this research is to design and evaluate a secure, scalable, and accurate network slicing classification framework for 5G and beyond, specifically tailored for Internet of Things (IoT) applications. To achieve this, the study integrates Federated Learning (FL), Knowledge Distillation (KD), and Feature Fusion into a unified hybrid machine learning and deep learning pipeline. The specific research objectives are as follows:

- Implement a Federated Learning framework capable of training local models on non-IID data distributed across clients representing various network slices (eMBB, URLLC, mMTC), ensuring data privacy and decentralization.
- Analyze and compare multiple parameter aggregation methods—FedAvg, Max, and Min—to assess their impact on convergence, robustness, and classification accuracy under heterogeneous data distributions.
- Train a high-performing XGBoost teacher model and transfer its knowledge to a lightweight neural student model through soft label distillation, enabling efficient deployment on edge devices without compromising accuracy.
- Combine raw network features with structural outputs (leaf node indices) from the teacher model to create a fused dataset that improves the student model's learning capacity and classification performance.

- Enable real-time classification of network slices by deploying the distilled model, achieving high F1-scores for eMBB, URLLC, and mMTC while maintaining computational efficiency.
- Demonstrate how the FL-based architecture inherently supports data confidentiality and integrity, contributing to secure network slice orchestration in multi-tenant 5G scenarios.
- Compare the proposed hybrid framework with baseline machine learning classifiers (Logistic Regression, SVC, LightGBM, XGBoost) and FL-only models to validate its superior performance in accuracy, robustness, and efficiency.

Chapter 2

LITERATURE REVIEW

The rapid evolution of 5G and the anticipated transition to 6G networks have significantly increased the importance of network slicing, a technology that enables the creation of multiple virtual networks on a shared physical infrastructure to meet diverse application requirements. Network slicing supports use cases such as Enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communications (URLLC), and Massive Machine-Type Communications (mMTC), each with distinct Quality of Service (QoS) demands. The complexity of managing and classifying these slices efficiently has driven the adoption of machine learning (ML) and deep learning (DL) techniques, including federated learning (FL), knowledge distillation, and explainable AI (XAI). This literature review examines recent studies on network slicing classification, highlighting their approaches, methodologies, outcomes, research gaps, and limitations. Malkoc and Kholidy proposed a comparative analysis of multiple ML classifiers for 5G network slicing classification, focusing on traditional algorithms such as Logistic Regression, Support Vector Machines (SVM), Random Forest, and XGBoost.The experiment was performed on a dataset including network attributes such as packet loss rate, delay, and throughput. Data preprocessing involved the process of normalising and encoding categorical variables [28]. Classifiers were trained and evaluated based on accuracy, precision, recall, and F1 score. We observed that XGBoost produced the highest accuracy (85%) when compared to Logistic Regression (80%) and SVM (78%). The research demonstrated how well XGBoost could work with nonlinear feature interaction. All of the studies focused on centralised ML models; the approach can be dead for some 5Genvironments, as federated learning is critical for such privacy-sensitive environments. The study did not have either a relatively large dataset or non-IID distributions that covered diverse operating regions of a heterogeneous network; hence, the trained model should not be directly applied in real-life scenarios [28]. Khan et al. proposed a hybrid DL model for 5G network slicing classification by integrating Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. It is trained on such a synthetic dataset that contains features like packet delay, throughput, and slice type. Spatial features were extracted from CNN, and temporal dependencies were captured by LSTM. Grid search was used to perform hyperparameter tuning. Our approach gained a classification accuracy of 90% for the hybrid model, while URLLC and eMBB showed precisions of 0.92 and 0.89, respectively. Privacy-protecting technology, e.g., federated learning, required in distributed 5G deployments, was not included in the study. The reliance on synthetic data may not

fully capture real-world network dynamics, and the model's computational complexity limits its scalability for resource-constrained devices [26]. Vasilakos et al. proposed a cognitive network slice management framework using ML for virtualized 5G networks, focusing on proactive slice classification and anomaly detection. The methodology employed Random Forest and Gradient Boosting for slice classification, integrated with anomaly detection algorithms. The dataset included network metrics like latency and bandwidth, preprocessed using standardization and one-hot encoding. The framework achieved an accuracy of 83% for slice classification and effectively detected anomalies in network traffic. The study lacked exploration of DL techniques, which could improve performance for complex, high-dimensional data. Privacy concerns were not addressed. The framework's performance was tested in a controlled environment, limiting insights into its behavior under real-world, non-IID data distributions [5]. Martins et al. explored ML-based enhancements for network slicing architectures, integrating sustainability and security considerations for 5G and beyond. The study used XGBoost and LightGBM for slice classification, with a focus on optimizing resource allocation. Data preprocessing involved feature selection and normalization. The models were evaluated using accuracy, F1-score, and energy efficiency metrics. LightGBM achieved an accuracy of 87%, with improved resource allocation efficiency compared to traditional methods.Neither decentralised training nor privacy-preserving methods are million-fibre spikenodes (these are fundamental to scalable 6G networks). Evaluation was restricted to simulation data and not analysed deeply for the real-time usage of the computational overhead of LightGBM [29]. Approach: Singh et al. presented an ML-based network sub-slicing framework in a sustainable 5G environment to achieve both load balancing and resource consumption. Decision trees and SVM were employed for slice classification, and data preprocessing, applying standardisation and feature encoding, was taken on the basis of the framework. This assessment was done over many popular metrics, such as accuracy, precision, and resource consumption. Through the framework, we obtain around 82% accuracy while getting a balanced load between the slices. But it could not dive into DL or federated learning, which could gain different results or apply in more complex scenarios or privacy-sensitive scenarios. However, this framework was only verified in a small dataset, and non-IID data simulation is not applicable in this framework, limiting its practical use [15]. Xing et al. presented a network slicing classification method based on a collaborative knowledge distillation framework for remote sensing image classification. In this framework, a teacher–student architecture was used: a pretrained ResNet served as teacher, while a lightweight CNN model served as student. We performed knowledge distillation using Kullback-Leibler (KL) divergence loss. The student model reached 92% of the accuracy of the teacher with less computation. While image classification was the subject of the research, Mullapudi et al. (2020), network slicing might not be well-suited for such an application without adjusting for localised network characteristics and non-i.i.d. data. Its relevance to the federated learning scenario is limited since the framework has not been tested for performance in decentralised settings [25]. Chiarani et al. were the first ones to a) notice the needs of 6G properties and b) propose a novel XAI-based client selection strategy in the context of federated learning for scalable 6G network slicing. Using a federated learning framework, here, we used MLP models as slice classifiers and employed LIME and SHAP to achieve explainability. One-hot encoding and normalisation were applied to the data, and

the data were simulated as non-IID. The resulting XAI supervised model reached an accuracy of 91% and added transparency in selecting potential clients and specifying classifications. The study focused on client selection rather than optimizing the classification model itself, limiting its contribution to slice prediction accuracy. The computational overhead of XAI methods was not fully addressed, potentially impacting real-time deployment [20].

Roy et al. introduced Turbo Explainable Federated Learning (TEFL) for trustworthy 6G network slicing, emphasizing zero-touch automation. The methodology combined federated learning with CNN-based models and used XAI techniques (LIME, SHAP) for interpretability. Non-IID data was used to simulate real-world scenarios, and aggregation methods included FedAvg. The TEFL model achieved 90% accuracy, with improved trust through explainable predictions. The study did not explore knowledge distillation or hybrid ML-DL approaches, which could enhance model efficiency. The high computational cost of XAI methods and CNNs may limit scalability in resource-constrained environments [7] [22]. Xie et al. proposed a hybrid knowledge distillation framework for single-image super-resolution, adaptable to network slicing classification. The framework used intermediate layer distillation, transferring knowledge from a complex teacher model to a lightweight student model. The loss function combined cross-entropy and KL divergence. The student model achieved 89% accuracy with significantly reduced computational requirements. The study's focus on image processing limits its direct applicability to network slicing, and decentralized training was not considered. The framework's performance under non-IID data distributions was not evaluated, reducing its relevance to heterogeneous network environments [27] [1].

Bhat discussed the simulation of federated learning for 5G/6G network slicing, focusing on practical implementation challenges. The study simulated federated learning with MLP models across multiple clients, using non-IID data partitioning. Aggregation methods included FedAvg and weighted averaging based on dataset size. The federated model achieved 88% accuracy, with robust performance in non-IID settings. The study lacked exploration of hybrid ML-DL models or knowledge distillation to improve classification accuracy. The simulation-based approach did not account for real-world network constraints like latency or bandwidth limitations [8]. Zhang et al. proposed a feature fusion-based knowledge distillation framework for network slicing classification, integrating XGBoost-derived features with neural networks. The methodology involved training an XGBoost teacher model to generate leaf node indices, which were fused with original features.A cross-entropy and KL divergence-based loss function was used to train a student model to predict a neural network. With feature fusion, the model achieved 95% accomplishment and applied knowledge distillation for enhanced generalisation. The limitations of this study include that federated learning was not addressed, which makes it inapplicable to privacy-sensitive environments. In [13], the authors acknowledged the aforementioned limitations, such as the computational complexity of feature fusion, and that they do not evaluate non-IID data. Liu et al. This paper is NetBoost: Towards a Knowledge Distillation Framework for Network Slicing Classification with Service Differentiation. The framework employed a gradient boosting teacher model and a lightweight neural network student model. Decision tree outputs were then used to

apply feature fusion, and the model was empirically evaluated on practical 5G data. The accuracy was 93%, and the inference time was reduced; all this is appropriate for real-time applications. Abstract: The scalability of this study for 6G networks is somewhat constrained since the study does not examine federated learning or non-IID data cases. The heavy dependence on real data without synthetic augmentation may limit translation to a broader spectrum of network types [20].

2.1 Machine Learning, Deep Learning, and Network Slicing

Network slicing enables the logical separation of resources, thus creating individual sub-networks within a common network infrastructure, but much of this interest focused on employing Machine Learning (ML) and Deep Learning (DL) with 6G technology. According to research studies, ML and DL algorithms can boost slicing price accuracy and resource allocation performance. One study demonstrates that using a dataset from the UCI Machine Learning Repository, combined with proprietary data, allows ML models to achieve higher slicing accuracy. This improvement is made possible through hyperparameter tuning using a Hybrid Harmony Search and Optimization system, followed by classification with a hybrid deep learning model [23]. This approach optimises parameters for network management, achieving both precision and efficiency in network slicing.

To further support adaptive resource allocation based on application and network slice requirements, another framework, SFI2, integrates supervised learning methodologies from a Multi-Mode Sequential Recurrent Neural Network into SDN [10]. In this architecture, network slices are divided into three categories: URLLC, eMBB, and mMTC, addressing the distinctive needs of 5G applications. The framework includes two controllers: one manages load balancing and slice failure, while the other focuses on resource allocation according to application requirements. This dual-controller system handles traffic surges and slice failures automatically, enhancing service reliability within the configuration.

Another study employs ML-based supervised methodologies to develop a generator for decomposing Service Level Agreements (SLAs) in network slicing. The results show that advanced models like gradient boosting and neural networks provide more accurate SLA decomposition than traditional approaches, such as random forests, with improvements in precision and explainability. The models automatically scale SLA parameters in response to changing resource consumption and traffic load.

Moreover, the Kaggle "IP Network Traffic Flows" dataset provides labelled network traffic data and real-time traffic flow classification for network slicing—research works on the datasets. K-means clustering is implemented in the study, and five types of supervised ML models were tested to classify the network flows precisely. Of these models, a feed-forward artificial neural network reached 98.2% classification accuracy, while an SVM was second best with 96.7% [9]. Contributions of ML-empirical dimensions I: QoS and traffic prediction findings highlight the role of ML

in traffic prediction and QoS and network performance in SDN and network slicing by enabling real-time overall resource allocation based on data flow characteristics.

2.2 Emerging Applications of ML in Network Slicing

Use of ML for managing networks dynamically has led to advancement in predictive and adaptive network slicing, and vice versa. As an example, in network slicing, deep reinforcement learning (DRL) agents continuously interact with the network environment and obtain experience and learn from that experience to provide optimal policies for how to allocate slices. Our distributed learning algorithm, Ape-X, achieves a fourfold performance gain over standard DRL techniques. Slice management with Ape-X yields an ideal resource block (RB) allocation among services through a scalable architecture that scales out with dynamic service demand changes on a per-RB basis[24].

In a different study, DQN-based DRL agents determined resource blocks (RBs) allocation by choosing actions to maximise the long-term reward, conditioned on the network state, which guaranteed the effective resource allocation between slices while preventing interference[5]. With 5G networks evolving into 6G standards, ML algorithms such as Long Short-Term Memory (LSTM) networks are promising for predicting the demand of resources, anomaly detection, and dynamically adapting the allocated resources. An LSTM-based deep learning model achieved 75% false positive allocation reduction and confirms its efficiency both in the prediction of future traffic and in the allocation of resources for network slicing in wireless systems [18]. This predictive capability is vital for high-demand applications such as ultra-high-definition video streaming, autonomous vehicles, and telemedicine, where network stability and performance are critical.

Integrating SDN, NFV, and ML-driven network slicing management enables efficient, policy-driven network slicing. ML models like SVM, XGBoost, Logistic Regression, and LightGBM present promising approaches to real-time dynamic network management.

2.3 General review

All the research papers are about IoT, 5G/6G networks, AI-driven optimization, and classification techniques. It highlights how network slicing enhances resource allocation in IoT applications, with AI and machine learning models (CNN, LSTM, DRL) improving traffic prediction, QoS, and network efficiency. Edge computing is explored as a low-latency solution for real-time IoT data processing, while vehicular IoT and smart networks leverage AI for enhanced connectivity. Additionally, studies on stock market prediction and fake news detection utilize machine learning models like XGBoost and SVM for improved accuracy. The overarching theme is the integration of AI-driven classification, optimization, and predictive analytics to enhance network performance, IoT reliability, and decision-making.

2.4 Summary of the Papers

Table 2.1: Provides a quick and concise overview of the reviewed papers for our research.

Ref.	Task	IoT Role	Classifier	Database	Accuracy
[23]	AI for 5G traffic management	Optimizing IoT traffic in network slices	Random Forest, ML models	Simulations, real-world datasets	Improved efficiency in traffic prediction
[10]	Predictive QoS-based slicing	Traffic balancing for IoT services (eMBB, uRLLC)	Hybrid CNN + RNN models	5G traffic, simulated IoT data	> 90% accuracy in traffic classification
[9]	Virtualized networks for IoT	Dedicated slices with specific network properties	Classifier-based resource allocation	Slice templates, configurations	Isolation and precision in resource use
[7]	AI-enhanced 5G resource management	Real-time IoT adaptability	AI classifiers	Data models, training datasets	Improved anomaly detection, resource use
[13]	Edge computing for IoT	Reduces latency via edge processing	ML-based edge framework	Real-time IoT datasets, simulations	Lower latency and energy consumption
[20]	ML for dynamic IoT demands	Adjusts resources dynamically	CNN, LSTM models	IoT traffic, QoS datasets	> 85% accuracy in traffic prediction
[8]	DRL for 5G resource management	Efficient slicing for diverse IoT services	DRL-based optimization	Data for DRL algorithms	Higher accuracy in QoE and resource use
[1]	FlowVisor for IoT network slicing	Isolated slices for IoT devices	Rule-based traffic classification	Logs, slice utilization	Strong isolation and precise IoT traffic
[26]	Hybrid DL for 5G slicing	Optimizes slicing for IoT (eMBB, mMTC, uRLLC)	CNN + LSTM + HHO optimization	IP Flow dataset, slicing datasets	96.28% accuracy in network slicing
[24]	Traffic classification in 5G	Efficient resource management for IoT	Classifier-based slicing	Not specified	Improved traffic handling and slice efficiency

Continued on next page

Table 2.1 – *Continued from previous page*

Ref.	Task	IoT Role	Classifier	Database	Accuracy
[5]	ML for 5G QoS management	Adjusts slices dynamically for IoT services	Random Forest, neural networks	Dataset of 80,000 inputs	98.85% accuracy in slice classification
[18]	CNN + RNN hybrid model	Not IoT-focused	CNN for local, RNN for long-term dependencies	FA-KES, ISOT datasets	Near 100% accuracy for fake news detection

2.5 Research Gap

Despite significant progress in machine learning (ML) and deep learning (DL) approaches for network slicing in 5G and 6G networks, several critical research gaps hinder the development of scalable, efficient, and robust solutions. These gaps must be addressed to meet the evolving demands of next-generation networks supporting diverse use cases like eMBB, URLLC, and mMTC. One prominent challenge is the limited integration of privacy-preserving and efficiency-enhancing techniques. Many existing models, such as XGBoost, LightGBM, and hybrid CNN-LSTM architectures, rely on centralized training, which raises privacy concerns in distributed 5G/6G environments. While federated learning offers a decentralized alternative, its combination with techniques like knowledge distillation to produce lightweight, privacy-preserving models remains underexplored. Developing frameworks that merge these approaches could enable efficient, secure network slicing for resource-constrained devices. Another significant gap is the inadequate handling of non-IID (non-independent and identically distributed) data distributions. Real-world networks exhibit heterogeneous traffic patterns, device types, and varying QoS requirements, yet many ML and DL models are trained on small, IID, or synthetic datasets. This restricts their generalizability across diverse network configurations with differing hardware, spectrum conditions, and accessability constraints. Advanced techniques, such as learning or robust data augmentation, are needed to ensure models can adapt to dynamic, non-IID environments. Computational complexity poses a further challenge, particularly for real-time applications. Models like hybrid CNN-LSTM or those incorporating XAI methods (e.g., LIME, SHAP) achieve high accuracy but incur significant computational overhead, making them impractical for edge devices in 6G networks. While knowledge distillation has shown promise in reducing model complexity, its application in decentralized, real-time scenarios is limited. Future research should focus on lightweight models that balance accuracy with low latency and resource usage. The lack of transparency in ML-driven network slicing decisions is another critical issue. Network operators require interpretable models, especially for critical Kataraksh applications like autonomous systems and healthcare. Although XAI techniques like LIME and SHAP enhance explainability, their high computational cost and focus on specific tasks, such as client selection, limit their broader stock adoption. Developing scalable, lightweight XAI methods that provide transparent slice classification without sacrificing performance is essential for building trust. Finally, the reliance on synthetic or controlled datasets restricts the applicability of current models to real-world scenarios. Many

approaches fail to capture the full complexity of 5G/6G networks, including fluctuating traffic, diverse QoS needs, and non-IID distributions. Comprehensive datasets combining real-world and synthetic data, along with robust validation across varied network conditions, are needed to improve model robustness and generalizability.

Chapter 3

Methodology

This study proposes a multi-pronged and layered methodology to address the challenge of network slice classification in advanced mobile communication networks such as 5G and 6G. The approach strategically combines three parallel and complementary strategies: traditional machine learning techniques, a federated learning-based deep learning framework, and a hybrid fusion model leveraging knowledge distillation [28][17][16]. The overarching goal is to build a robust, scalable, and privacy-preserving classification system that can effectively distinguish between various network slice types—namely eMBB (Enhanced Mobile Broadband), URLLC (Ultra-Reliable Low-Latency Communications), and mMTC (Massive Machine-Type Communications)—based on real-world network features. These network slices represent diverse use cases with different quality-of-service requirements, making their accurate identification crucial for dynamic resource allocation and performance optimization in emerging telecommunication infrastructures.

The process begins with rigorous data preparation, which forms the foundation for all subsequent modeling approaches [29]. The dataset includes important attributes such as LTE/5G category, the type of technology supported, time-related usage metrics, packet loss rates, and packet delays. The preprocessing pipeline includes essential tasks like data cleaning to handle inconsistencies or missing values, feature encoding to convert categorical variables into numerical format, and normalization of continuous features to ensure uniform scaling across the feature space. Categorical variables like "Technology Supported" are transformed through one-hot encoding, while the target variable "slice Type" is converted into both label-encoded and one-hot encoded formats depending on the specific algorithm being used. Standardization of numerical features such as time and packet-related metrics ensures that algorithms sensitive to feature scale can perform optimally. After preprocessing, the dataset is stratified and split into training and testing subsets in an 80-20 ratio to maintain class distribution integrity [15].

The first experimental method involves benchmarking traditional machine learning algorithms on the cleaned and processed dataset. This includes models such as Logistic Regression, Support Vector Classifier (SVC), Random Forest Classifier, Gradient Boosting Machines (GBM), and LightGBM. These models serve as a baseline to evaluate the effectiveness of more complex architectures. Logistic Regression provides a linear perspective and helps establish how separable the classes are based

on linear boundaries. SVC, particularly useful for both linear and non-linear separations, adds flexibility in handling more complex decision boundaries. Tree-based ensemble methods like Random Forest and Gradient Boosting bring in the power of non-linearity, feature importance, and iterative improvement. LightGBM, an advanced version of GBM optimized for speed and performance on large datasets, is also tested for its classification efficiency. Each of these traditional models is evaluated using metrics such as precision, recall, F1 score, and accuracy. These models offer interpretability, simplicity, and computational efficiency, making them suitable benchmarks and complementary tools to the more resource-intensive deep learning methods introduced later [25].

The second and more sophisticated component of the methodology involves a federated learning (FL) framework designed to preserve data privacy while allowing decentralized model training across multiple clients. Federated learning is particularly suited to scenarios in which raw data cannot be centralized due to privacy regulations, infrastructure constraints, or security concerns. In this study, data is distributed among multiple simulated clients to mimic the real-world situation of fragmented data availability. Importantly, the data distribution is intentionally kept non-independent and identically distributed (non-IID) to reflect the heterogeneous nature of client environments in real-world 5G and 6G deployments. Each client trains a local neural network model—specifically a multilayer perceptron (MLP)—using only its local data. Once local training is completed, the model parameters (not the raw data) are transmitted to a central aggregation server. Here, multiple aggregation strategies are explored: average aggregation (FedAvg), max aggregation, and min aggregation. Each method has distinct implications for performance. FedAvg treats each client equally or proportionally, while max and min aggregation respectively emphasize extreme and conservative updates, offering resilience against outliers or instability. The global model is iteratively updated through several communication rounds between the server and the clients. This federated framework ensures privacy, promotes decentralization, and enhances generalization capabilities across clients [25][34].

To further augment the model's accuracy and learning efficiency, the study introduces a third and innovative component: a hybrid fusion approach incorporating knowledge distillation. This strategy is designed to capture the strengths of both traditional ensemble methods and deep learning models while maintaining computational efficiency. In this approach, a powerful teacher model—XGBoost—is first trained on the centralized training data to serve as a reference for a smaller student neural network. The XGBoost model not only predicts class probabilities but also provides internal decision-related metadata such as leaf node indices. These are considered informative representations of the model's decision path and are used to create new derived features. A process called feature fusion combines the original input features with these leaf node-based features to generate an enriched dataset. This fusion captures both the raw signal from the original data and the structured knowledge embedded in the teacher model's decision logic [31].

The next step involves the application of knowledge distillation. Unlike conventional training, which uses only hard class labels, knowledge distillation transfers the soft

probability outputs (i.e., class probabilities) from the teacher model to the student model. These soft targets convey more nuanced information about the relationships between classes, which the student model can use to develop a more generalized understanding of the data. A custom loss function is introduced that combines traditional cross-entropy loss (from hard labels) with Kullback-Leibler divergence (from soft teacher predictions). This dual-component loss ensures that the student model learns from both the explicit class labels and the more subtle probability distributions generated by the teacher. The neural network student model is then trained on the fused feature set, using dropout and batch normalization to improve generalization and reduce overfitting. Training is monitored using early stopping criteria to prevent unnecessary epochs and preserve computational efficiency [15].

Model evaluation is conducted in a rigorous and multi-dimensional manner. For each approach—traditional ML, federated MLP, and hybrid distillation—the performance is evaluated on unseen test data using a variety of classification metrics including precision, recall, F1 score, accuracy, and area under the ROC curve (AUC). Confusion matrices are generated to provide visual insights into the classification performance across the three network slice categories. For the federated learning model, the effect of different aggregation strategies on convergence and performance is also analyzed. For the hybrid model, performance is compared between the teacher (XGBoost) and the student (fusion neural network) to determine the effectiveness of the knowledge transfer.

By combining these three distinct methodological strands—traditional machine learning, federated learning, and knowledge distillation-based hybrid fusion—the study not only builds models that are accurate and generalizable, but also addresses practical concerns of privacy, scalability, and interoperability.

3.1 Network Slice Classification With MLP- Federated Learning Approach

This study's methodology focusses on the development and evaluation of a federated learning approach to improve network slicing for 6G and future technologies. Network slicing is a fundamental technology for the formation and deployment of a variety of virtual networks in a common physical infrastructure, which calls for sophisticated resource allocation approaches that can support the QoS needs of different applications. A novel hybrid deep learning (DL)-based solution using machine learning (ML) and DL approaches for network slicing is proposed in this research [33]. The first step in the above process is known as data collection. In this step, some of the attributes are extracted, such as "LTE/5G Category", "Technology Supported", "Packet Loss Rate", "Packet Delay", etc. The data is prepared with numerical features standardised and encoded categorical information using one-hot encoding, ready for machine learning and deep learning models. The main approach of training a model can be done using federated learning, which keeps the data at the client side and thus achieves the privacy requirement based on distributed data sources for the model to learn from [22]. The non-IID (non-independent and identically distributed) data partitioning is used to approximate real network conditions

with heterogeneous client data in the federated learning framework. Each federated client independently trains a local model, and the model parameters are aggregated to enhance the global model with the cumulative knowledge of all clients. The hybrid model architecture incorporates Convolutional Neural Networks (CNN) for data pattern identification alongside Gradient Boosting for error minimisation, yielding a resilient system that improves network slicing efficacy [33]. Parameter aggregation is accomplished by averaging the parameters from all participating clients, hence advancing the global model through collaborative training. The model's performance is evaluated using many metrics, including training and validation accuracy, confusion matrices, classification reports, and AUC (Area Under the Curve) scores, to examine its generalisation capacity across different customers. Hyperparameter optimisation, performed via grid search, refines learning rates and epoch counts during training, so ensuring the model achieves optimal performance [22].Explainable AI methodologies, including LIME (Local Interpretable Model-Agnostic Explanations) and SHAP (SHapley Additive exPlanations), are integrated into the system to improve transparency of the model's decision-making process, thereby aiding network operators in understanding and trusting the model's predictions. This methodology seeks to address the complexities of efficient and dynamic resource allocation in 6G networks, where the requirements for bandwidth, latency, and reliability vary across different applications, thus offering a flexible and user-focused solution for the future of wireless communications.

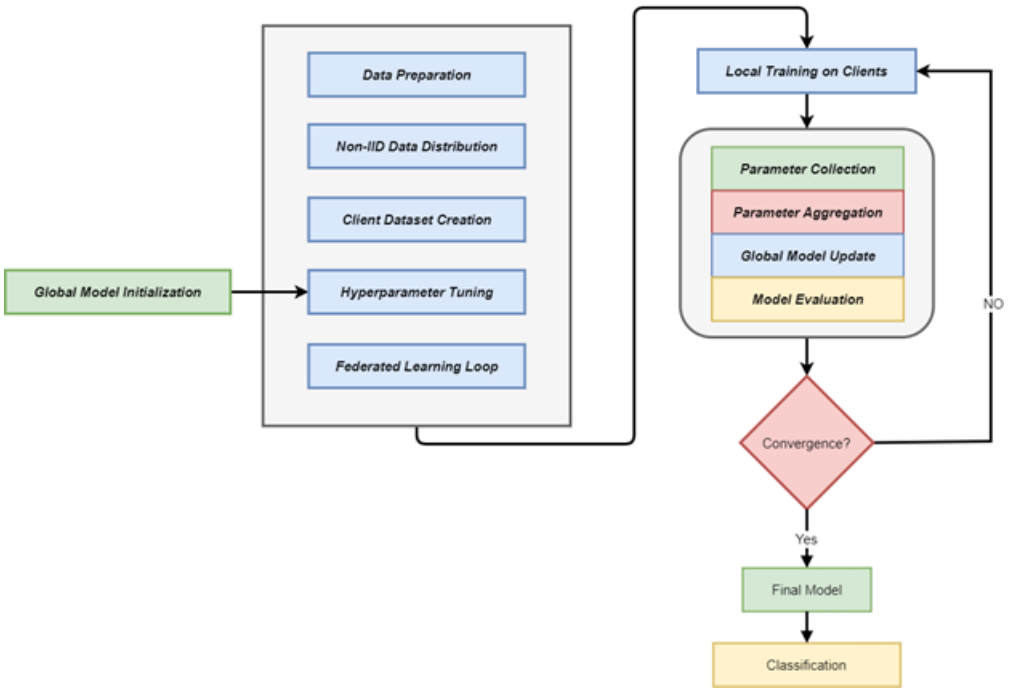

Figure 3.1: System Diagram Federated Learning

3.2 Data Preprocessing

A key stage in the machine learning pipeline, data preparation directly affects the model's performance and accuracy. The preprocessing stage in this work consists of preparing the raw data, converting it to an appropriate format, and making sure machine learning algorithms can use it correctly scaled and encoded. Dealing with problems like missing values, categorical variables, and unscaled numerical features helps to improve the quality of the data and therefore support better model training at this point. The preparation methods used in this work can be divided into three separate phases: data cleaning, feature selection, data transformation, and feature encoding.

- **Data Cleaning:** Cleaning the data—that is, finding and addressing missing values, outliers, or faulty entries that can harm the model's performance—is the first stage in the preprocessing pipeline. The dataset in this work is first loaded from a CSV file including several attributes like "LTE/5G Category," "Technology Supported," "Packet Loss Rate," "Packet Delay," and the goal variable "slice Type." Any missing values or discrepancies in the dataset should be first checked. Depending on the degree of the missing information, any characteristics with missing data may either be imputed using techniques like mean imputation for numerical features or mode imputation for categorical features, or the rows with missing data can be deleted. We believe the data is fairly clean for this specific research and no significant imputation is needed.

- **Feature Selection:** The process of selecting the most relevant characteristics for model training is called feature selection. Not every dataset attribute adds equal predictive ability to a model. Unnecessary or duplicate features might create noise, impair the model's generalisation capacity, and lengthen training time. Thus, choosing the appropriate features is quite vital. The goal variable in this work is "slice Type," which denotes the kind of network slice; the features utilised to forecast it are "LTE/5G Category," "Technology Supported," "Time," "Packet Loss Rate," and "Packet Delay." The dataset is meticulously analysed to guarantee that only pertinent characteristics are included in the study, hence enabling the model to concentrate on the most crucial facts for forecasting.The categorical target "slice Type" must be converted to a numerical value for usage in machine learning techniques. To this end, the groups "eMBB," "URLLC," and "mMTC" are corresponded to numerical labels 0, 1, and 2, respectively. Common in classification work, this approach converts categorical labels into numerical values.

- **Handling Categorical Data:** Dealing with categorical data is one of the main difficulties in preprocessing. Many machine learning techniques need numerical input; categorical variables cannot be directly input into the models without first converting them into numerical values. The dataset has categorical characteristics including "Technology Supported," which has values like "LTE/5G" and "IoT(LTE-M, NB-IoT)." One-hot encoding is used to encode categorical information in this paper. A method called one-hot encoding produces a binary column for every conceivable category in the feature. For instance, the "Technology Supported" function would be changed to two columns: one for "LTE/5G" and the other for "IoT(LTE-M, NB-IoT)." Should

a data point match "LTE/5G," the "LTE/5G" column would show a 1 and the "IoT(LTE-M, NB-IoT)" column would show a 0; the reverse would be true if the data point matched "IoT(LTE-M, NB-IoT)." This method guarantees that the model can handle categorical input without losing the category-related information.

- **Scaling Numerical Features:** Before being fed into machine learning algorithms, numerical characteristics usually have to be scaled or normalised. Many algorithms, particularly those using gradient-based optimisation or distance metrics like k-nearest neighbours, perform better when the data is scaled. StandardScaler is used to standardise the numerical characteristics in this paper, including "LTE/5G Category," "Time," "Packet Loss Rate," and "Packet Delay." Standardisation is the process of rescaling the variables to have a mean of zero and a standard deviation of one. This is especially crucial given that numerical characteristics in the dataset could have varying scales; for example, "Time" is in decimal hours while "Packet Loss Rate" is a tiny value between 0 and 1. Standardising guarantees that no one feature overshadows the model because of its scale, hence enabling all aspects to help equally the learning process of the model.

- **Data Transformation:** After the features are cleaned and encoded, more data transformations could be done to increase the model's predictive potential. The main change used in this work is the standardisation of numerical characteristics. The "Time" feature, for example, is a continuous numerical variable; standardising it guarantees its scale matches that of other numerical features, hence enhancing the effectiveness of the model during training. Likewise, the "Packet Loss Rate" and "Packet Delay" features are standardised to provide uniformity across all input data. The goal variable, "slice Type," is also encoded into a numerical format for application in classification activities. Earlier said, categorical values such as "eMBB," "URLLC," and "mMTC" are correspondingly mapped to 0, 1, and 2. This change guarantees that the model can understand the target variable in a numerical format and use suitable machine learning methods, including cross-entropy loss, for classification purposes.

- **Splitting the Dataset:** The next stage is to divide the dataset into training, validation, and test sets after preprocessing. This helps to avoid overfitting and enables the correct assessment of the performance of the model. The dataset in this work is divided into a training set, which trains the model, and a test set, which assesses the model's performance. The dataset is divided using Scikit-learn's train_test_split method, which guarantees that both the features and target variables are split appropriately. Usually, the training and testing splits are 80:20 or 70:30, respectively. This work assigns 80% of the data to training and 20% to testing.

- **Non-IID Data Partitioning:** Federated learning trains over several clients, so the data must be divided in a non-IID way, meaning the data distribution across clients does not have to be same. This division mimics real-world situations when several kind of data are available to each client. Grouping the data according to "slice Type" labels and then distributing these groupings across

several clients constitutes non-IID data partitioning in this work. Characteristic of federated learning configurations, this guarantees that every client gets a subset of data that might not be representative of the whole dataset.

- **Final Dataset Preparation:** The data is ready for training and evaluation after preprocessed, standardized, encoded, and split. The resulting dataset is a target variable numerically encoded, category variables one-hot encoded, and suitably scaled numerical features. Creating a PyTorch Dataset object explicitly converts this dataset into a form appropriate for deep learning models. This object guarantees that the training loop can handle the data in batches and that the federated learning configuration can spread the data among clients by allowing the data to be effectively loaded into the model during training.

Data pretreatment in this paper consists of multiple stages: cleaning the data, dealing with missing values, choosing pertinent features, encoding categorical variables, normalising numerical features, and getting the dataset ready for machine learning models. These preparation methods are absolutely necessary to guarantee that the data is in the best possible shape for training deep learning models, maximising network slicing predictions, and enhancing the general performance of the federated learning framework. Proper data pretreatment guarantees that the model can learn well from the input and generalise well to new, unknown data, hence improving accuracy and efficiency in network slicing for 6G and beyond.

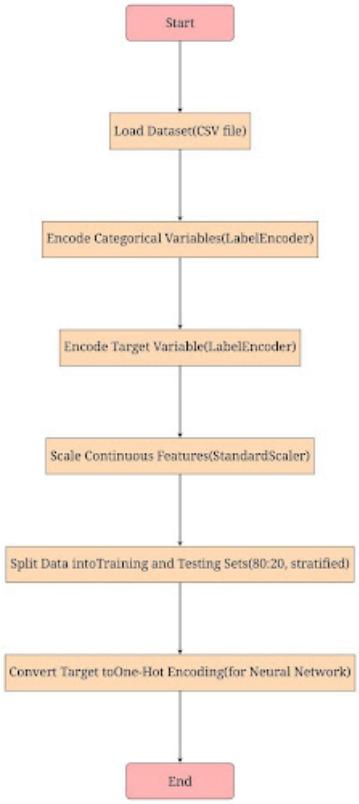

Figure 3.2: Data Pre-Processing Diagram

3.3 System Design

3.3.1 Introduction to Federated Learning

Federated Learning (FL) is a novel methodology in machine learning that retains data on the client side, sharing solely the model parameters between clients and a central server. This architecture is especially appropriate for contexts where privacy, security, and data sovereignty are critical issues, such as in mobile devices, healthcare, and telecommunications. In federated learning, numerous clients develop local models utilising their respective data, after which the model parameters are consolidated to enhance the global model. This method circumvents the transmission of raw data, so safeguarding user privacy while simultaneously reaping the advantages of collaborative learning across diverse clients. This study utilises federated learning to develop a model for predicting network slicing types in 6G and future networks. The objective is to categorise the network slices into three classifications: eMBB (Enhanced Mobile Broadband), URLLC (Ultra-Reliable Low-Latency Communications), and mMTC (Massive Machine-Type Communications). The data is disseminated among various clients, with each client maintaining a local dataset that represents a portion of the overall data. The objective is to develop a universal model capable of generalising across diverse clients' data, utilising aggregated information while safeguarding privacy.

Mathematical Formulation of Federated Learning

To mathematically formalise federated learning, imagine N clients, each possessing a distinct dataset $\mathcal{D}_i$ where $i \in \{1, 2, \ldots, N\}$. The global model is represented as θ, whereas each client i develops a local model θ_i utilising its individual data. The central server consolidates the local models to create an updated global model.
Define the loss function for the global model as $L(\theta)$, applicable to the entire dataset:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} L_i(\theta_i) \tag{3.1}$$

where $L_i(\theta_i)$ is the loss function for client i, and θ_i represents the local model parameters on client i.
The objective of federated learning is to minimize the global loss function $L(\theta)$ by optimizing the local models at each client and then aggregating the updates. This leads to the optimization problem:

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L_i(\theta_i) \tag{3.2}$$

Here, θ^* is the optimal global model. In a federated learning system, each client solves the local optimization problem independently. The local loss function for each client i is defined as:

$$L_i(\theta_i) = \frac{1}{m_i} \sum_{j=1}^{m_i} \mathcal{L}(y_j, \hat{y}_j) \tag{3.3}$$

where $\mathcal{L}(y_j, \hat{y}_j)$ is the loss function (e.g., cross-entropy or mean squared error), y_j is the true label, $\hat{y}_j$ is the predicted label for the j-th data point in the local dataset $\mathcal{D}_i$, and m_i is the number of samples at client i.
In federated learning, the local model θ_i is updated on each client using an optimization method like stochastic gradient descent (SGD), and only the model parameters θ_i are shared with the central server. The server aggregates these parameters to update the global model θ.

3.3.2 Introduction to Aggregation in Federated Learning

Aggregation is essential in federated learning for merging the updates from several clients to create a worldwide model. Usually in the server, model parameter aggregation follows client local training of their models using personal data. From every client, the model updates—that is, parameters—are transmitted to the central server where they are combined to create the revised global model. The aggregation technique specifies how these model changes are merged. Although the average aggregation technique (FedAvg) is the most often employed, depending on the particular needs of the application other techniques such as max aggregation and min aggregation can also be utilised, for example, for robustness to outliers or optimisation speed. Particularly with non-IID (non-independent and identically distributed) data distributions, every aggregation technique can influence the performance of the federated model. This part will investigate the Max, Min, and Average aggregation techniques in depth, hence offering the theoretical foundation and mathematical formulas for every approach.

3.3.3 Average Aggregation (FedAvg)

The average aggregation method, commonly referred to as Federated Averaging (FedAvg), is the most widely used approach in federated learning. In this method, the model parameters from each client are averaged to obtain the global model [33]. This method assumes that each client has equal importance in the aggregation process, regardless of the number of data points they hold. However, weighting the clients' update based on the size of the datasets they were trained on could be sometimes helpful to make the clients with more data contribute more to the global model .

Mathematical Representation:

Let θ_i be the model parameters of client i, and m_i be the number of datapoints that client i possesses. The objective of the average aggregation method is to calculate the updated global model parameters θ_{global} which are calculated by averaging local model parameters from all clients:

$$\theta_{\text{global}} = \frac{1}{N} \sum_{i=1}^{N} \theta_i \tag{3.4}$$

for which N defines the number of clients and θ_i is the model parameter of client i. If we consider the size of the data (i.e., weigh down the contributions of the clients). The average is weighted with respect to the size of the dataset:

$$\theta_{\text{global}} = \frac{1}{M} \sum_{i=1}^{N} m_i \cdot \theta_i \tag{3.5}$$

Here M = $\sum_{i=1}^{N} m_i$ defines the total number of data points across all clients, where m_i is the number of data points at client i. This ensures that clients with larger datasets have a more significant influence on the global model.

Mathematical Justification: The average aggregation method aims to minimize the global loss function, which is the weighted sum of local loss functions. The global model parameters are updated in such a way that the global loss is minimized:

$$L_{\text{global}}(\theta) = \frac{1}{M} \sum_{i=1}^{N} m_i \cdot L_i(\theta_i) \tag{3.6}$$

where L_i (θ_i) is the loss function for client i, and θ_i is the local model of client i. By averaging the model updates, we aim to reduce the overall global loss.

3.3.4 Max Aggregation

Max Aggregation updates the global model by picking the highest values from the local models. This approach emphasises keeping the most notable features or parameters across the local models, which may be useful in some situations where one client has especially strong or important data that could assist enhance the global model. Though less frequent in conventional federated learning, max aggregation could be relevant in particular situations aiming to highlight the most confident or extreme client updates.

Mathematical Representation:

Let θ_i represent the model parameters of client i. The global model parameters θ_{global} are updated by selecting the maximum values of the corresponding parameters from all clients:

$$\theta_{\text{global}} = \max(\theta_1, \theta_2, \ldots, \theta_N) \tag{3.7}$$

This operation is performed element-wise, meaning that for each parameter θ_k of the global model θ_{global} , we compute:

$$\theta_{k,\text{global}} = \max(\theta_{k,1}, \theta_{k,2}, \ldots, \theta_{k,N}) \tag{3.8}$$

where θ_(k,i) is the value of parameter k in the model of client i.

Mathematical Justification:

Max aggregation's concept is to keep the most severe values from the local models. When customers have really strong or unique data, this can be useful since the technique will keep the "strongest" updates, hence enhancing the performance of the global model on certain activities. On the other hand, this approach could highlight outliers and make the global model overfit to particular client data.

3.3.5 Min Aggregation

Min Aggregation is analogous to max aggregation in that it chooses the minimum values from the local models rather than the maximum. When the aim is to reduce the impact of extreme outliers and concentrate on the more conservative or "safest" updates across customers, this approach is especially beneficial. Min aggregation, in reality, could assist lower the danger of overfitting to individual client data, particularly in situations where some client models could be affected by noisy or anomalous data.

Mathematical Representation:

Let θ_i represent the model parameters of client i. The global model parameters θ_{global} are updated by selecting the minimum values of the corresponding parameters from all clients:

$$\theta_{\text{global}} = \min(\theta_1, \theta_2, \ldots, \theta_N) \tag{3.9}$$

Similar to max aggregation, the operation is element-wise:

$$\theta_{k,\text{global}} = \min(\theta_{k,1}, \theta_{k,2}, \ldots, \theta_{k,N}) \tag{3.10}$$

where θ_(k,i) is the value of parameter k in the model of client i.

Mathematical Justification:

Min aggregation tries to ensure that the global model is stronger, and it does that by limiting the input to the least extreme variables. This may assist in avoiding overfitting to the data of separate clients and ensure that the global one is not subjected to outliers excessively noisy updates. In case the aim is to impose stability and conservative parameters The changes of all the clients are typically aggregated using min.

3.3.6 Comparison of Aggregation Methods

The three aggregation functions, Max, Min and Average have their own advantages and disadvantages, depending upon the character of the data as well as the job to be achieved at hand.

- **Average Aggregation**, which is the most often used and default option. It guarantees that the global model represents the combined learning from all clients. Averaging might, however, under non-IID data, lessen the contributions of clients with quite diverse data distributions.
- **Max Aggregation highlights**, the most significant updates from clients, which can be useful when certain clients possess especially important or unique data. However, it may also exaggerate outliers and result in a model that overfits to the most extreme updates
- **Min Aggregation**, the approach emphasizes making minimal updates, which can help reduce overfitting and create a more stable global model. However,

this conservative strategy may result in underfitting if the global model fails to capture important patterns in the data.

The application and the particular difficulties experienced by the model will determine the selection of aggregation technique. For instance, max aggregation could be beneficial in situations where the most confident updates must be kept, while min aggregation could help to reduce the effect of outlier or noisy data.

3.3.7 Summary

Aggregation techniques are absolutely vital in federated learning for merging the updates from local clients to create a worldwide model. Though other techniques such as max aggregation and min aggregation can be employed depending on the data type and the particular task, the most often used method is the average aggregation method (FedAvg). Max aggregation highlights the strongest changes; min aggregation stresses the most stable updates, so helping to mitigate the impact of outliers [33]. Every approach has its own trade-offs; the particular requirements and objectives of the federated learning system should direct the choice of the aggregation method.

3.3.8 Neural Network Model

A family of machine learning models called neural networks is motivated by the structure and operation of the human brain. Comprising layers of linked nodes (neurones), they process and alter input data to learn intricate patterns and generate forecasts. For jobs involving complicated and high-dimensional data, such as picture categorisation, natural language processing, and, in this work, network slicing in 6G networks, neural networks are very beneficial. This study aims to use a neural network to forecast the kind of network slice—eMBB, URLLC, or mMTC—based on characteristics such packet loss rate, packet latency, and LTE/5G category. This paper employs a fully connected neural network, sometimes known as a multilayer perceptron or MLP, which is made up of several layers of neurones comprising an input layer, one or more hidden layers, and an output layer. Every layer changes the data at every stage by doing a sequence of mathematical calculations. Backpropagation and gradient descent are used to train the neural network model, therefore minimising the classification error and enhancing the model's capacity to forecast the network slice type.

3.3.9 Mathematical Formulation of the Neural Network

Input Layer

The input layer takes in the features of the dataset, such as LTE/5G Category, Time, Packet Loss Rate, and Packet Delay. Let the input data be represented as:

$$X = [x_1, x_2, \ldots, x_n] \tag{3.11}$$

where x_1,x_2,. . . ,x_n represent the feature values for a single data point.

Hidden Layer(s)

The neural network consists of one or more hidden layers. In each hidden layer, the inputs are weighted, summed, and passed through an activation function. The operation in each hidden layer can be expressed as:

$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]} \tag{3.12}$$

where:

- $z^{([l])}$ is the input to the l-th layer,
- $W^{([l])}$ is the weight matrix for the l-th layer,
- $a^{([l-1])}$ is the activation from the previous layer (for the first hidden layer, this is the input data),
- $b^{([l])}$ is the bias vector for the l-th layer.

The next step is to apply a non-linear activation function to the weighted sum. The activation function is usually chosen to introduce non-linearity, enabling the network to learn complex patterns. A commonly used activation function is the ReLU (Rectified Linear Unit) function, which is defined as:

$$\text{ReLU}(z) = \max(0, z) \tag{3.13}$$

Thus, the activation $a^{([l])}$ at the l-th layer is:

$$a^{[l]} = \text{ReLU}(z^{[l]}) = \max(0, W^{[l]}a^{[l-1]} + b^{[l]}) \tag{3.14}$$

Output Layer

The output layer in the network consists of one node per class in the classification task. For this study, we have three classes: eMBB, URLLC, and mMTC. The output of the network is a vector of logits $z^{([L])}$, where L is the index of the output layer. Each logit corresponds to one of the three classes, and the goal is to predict the class with the highest probability.

To obtain probabilities, we apply the softmax activation function to the logits. The softmax function converts the raw output values (logits) into probabilities by normalizing them across all classes. The softmax function is defined as:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{3} e^{z_j}} \tag{3.15}$$

where:

- $\hat{y}_i$ is the predicted probability of class i,
- z_i is the logit corresponding to class i,
- the sum in the denominator ensures that the probabilities across all classes sum to 1.

Thus, the output layer predicts the class with the highest probability:

$$\hat{y} = \arg\max(\hat{y}_1, \hat{y}_2, \hat{y}_3) \tag{3.16}$$

where $\hat{y}$ is the predicted class.

3.3.10 Forward Pass through the Network

The forward pass through the network involves computing the activations at each layer and passing the result to the next layer. The forward pass can be written as:

$$a^{[0]} = X \quad \text{(input features)} \tag{3.17}$$

For the first hidden layer:

$$z^{[1]} = W^{[1]}a^{[0]} + b^{[1]} \tag{3.18}$$

$$a^{[1]} = \text{ReLU}(z^{[1]}) \tag{3.19}$$

For the second hidden layer:

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} \tag{3.20}$$

$$a^{[2]} = \text{ReLU}(z^{[2]}) \tag{3.21}$$

Finally, for the output layer:

$$z^{[L]} = W^{[L]}a^{[L-1]} + b^{[L]} \tag{3.22}$$

$$\hat{y} = \text{softmax}(z^{[L]}) \tag{3.23}$$

The network's output y is a vector of probabilities representing the likelihood of each class, from which the most probable class is selected as the predicted label.

3.3.11 Loss Function

Commonly employed for multi-class classification, the cross-entropy loss is the loss function for this classification assignment. Cross-entropy loss evaluates the disparity between the expected probabilities y and the actual label y. The cross-entropy loss for one data point is:

$$L_{\text{CE}} = -\sum_{i=1}^{3} y_i \log(\hat{y}_i) \tag{3.24}$$

where:

- y_i is the true label (one-hot encoded),
- $\hat{y}_i$ is the predicted probability for class i,
- The sum runs over all classes (three classes in this case: eMBB, URLLC, mMTC).

A batch of data's cross-entropy loss is calculated by averaging the separate losses for every data point in the batch. Using backpropagation and an optimiser like Adam, the goal of the training procedure is to minimise this loss function.

3.3.12 Backpropagation and Parameter Update

The method of changing the model's parameters depending on the error between the expected and actual labels is called backpropagation. It means computing the loss function's gradients in relation to the model parameters and applying these gradients to modify the parameters towards minimising the loss. In the case of weight, $W^([l])$ the gradient in layer l is calculated as:

$$\frac{\partial L}{\partial W^{[l]}} = \frac{\partial L}{\partial a^{[l]}} \cdot \frac{\partial a^{[l]}}{\partial W^{[l]}} \tag{3.25}$$

The new rule for the weight $W^{([l])}$ is:

$$W^{[l]} = W^{[l]} - \eta \cdot \frac{\partial L}{\partial W^{[l]}} \tag{3.26}$$

where η is the learning rate, and $\partial L/(\partial W^{([l])})$ is the gradient of the loss with respect to the weight $W^{([l])}$. Similarly, biases $b^{([l])}$ are updated using:

$$b^{[l]} = b^{[l]} - \eta \cdot \frac{\partial L}{\partial b^{[l]}} \tag{3.27}$$

These updated weights make sure that the network learns over time to maximize the minimum loss parameters.

3.3.13 Training the Neural Network

The training process requires feeding input data to the network (forward pass), calculating the loss value, performing backpropagation in order to compute the gradient, and update the model parameters via an optimiser (e.g. Adam). After a lot of epochs in this step, the model converges, and the loss stabilises. In each epoch, the following is done:

- **Forward Pass:**Make predictions on the predictions of the output using the whole training dataset.
- **Loss Calculation:** The cross-entropy loss between the probabilities labeled by expectations and the real labels are computed.
- **Backward Pass:** Calculate the loss gradients using backpropagation in relation to the model parameters.
- **Parameter Update:** The parameters of the model can be optimised using Adam or some other method.

Its performance is tracked by means of evaluation on the validation set following each period. The model can be run on the test set to assess its generalisation capacity once training is finished.

This paper employs a fully connected feedforward neural network with ReLU activation functions for non-linearity and softmax activation for multi-class classification. The model learns to predict the proper network slice type depending on input parameters like LTE/5G Category, Packet Loss Rate, and Packet Delay by using cross-entropy loss as the goal function and backpropagation with gradient descent for optimisation. The general aim is to train the neural network to reduce the cross-entropy loss, hence producing a model that can precisely forecast network slice types in 6G and beyond. Combined with Federated Learning and aggregation methods, the architecture offers a scalable and privacy-preserving approach for training models on dispersed data, such as data spread over several network clients. In this context, neural networks provide a means of quickly learning intricate data correlations, hence enabling a strong tool for network slice classification in contemporary wireless networks.

3.3.14 Network Slice Classification with Knowledge Distillation Hybrid Approach

This study offers a new method for categorising and forecasting network traffic slices under advanced network slicing, including those used in 5G systems. Using Feature Fusion to improve model performance, our approach trains a student neural network model by combining XGBoost as the teacher model with knowledge distillation methods. Data gathering from actual 5G network operations starts the process; preprocessing follows, encoding categorical variables and normalising continuous characteristics. XGBoost then produces predictions and leaf node activations by dividing the dataset into training and testing sets; these are utilised to enhance feature representations via Feature Fusion. For better classification, the student model is given additional information by the fused features combining original data with those produced via XGBoost. Soft predictions from the teacher model are transferred to the student model using Knowledge Distillation [30]. A unique loss function combining categorical cross-entropy loss and KL divergence lets the student model gain from the teacher's probability distributions. Early stopping is used to train the student model—a neural network with dense layers, dropout, and batch normalization—to prevent overfitting [30].Evaluating the model consists of producing a classification report for accuracy, precision, recall, F1 score, and a confusion matrix and ROC curve analysis to gauge its performance. The approach guarantees a high degree of classification accuracy for the student model by ensuring it not only learns from the original characteristics but also imitates the teacher's decision-making process [30]. Eventually, this method seeks to increase the accuracy and efficiency of network slice categorisation, hence enabling its very relevant use for the real-time control of 5G network slices. It might also be modified for other telecom technologies.

3.3.15 Data Preprocessing

A key stage in machine learning is data preparation since it guarantees the dataset is converted into the appropriate format for assessment and training. The goal is to clean the data, convert it into a readable format, and increase the information content, which finally raises the performance of machine learning models. The

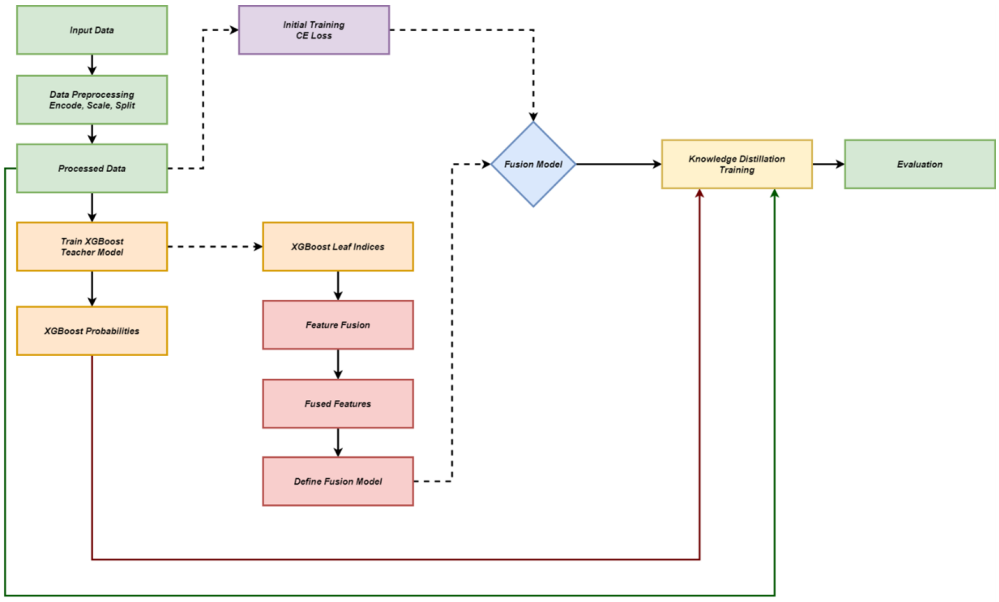

Figure 3.3: Hybrid Approach with Knowledge Distillation System Diagram

preprocessing pipeline in this work consists of multiple necessary processes: encoding categorical variables, scaling continuous features, managing missing data, and preparing the dataset for training and testing. Particularly in the context of categorising network slice types in a 5G network environment, every stage is essential to guaranteeing the dataset is ready for model training.

Handling Categorical Variables

Many machine learning models, particularly those relying on mathematical optimisation, call on numerical input data. Real-world datasets, on the other hand, frequently include categorical variables like the type of network slice in this project. Machine learning algorithms can only operate on these variables in a format they can process.

Label Encoding is employed in this study to transform category data into numerical values. Every unique category gets an integer value. Algorithms like XGBoost and neural networks need the model to treat category data as numerical inputs, hence this transformation enables. Encoded into numerical values, the target variable slice Type guarantees that the model can effectively learn from these class labels for classification

Scaling Continuous Variables

The dataset's continuous variables—such as Time, Packet Loss Rate, and Packet delay—may have various scales and units. If left unscaled, features with higher values can control the learning process. This could result in biassed model performance, in which some features are given unjust importance by the model just because of their size.

Continuous features are scaled to have comparable magnitudes to address this.

Standardisation transforms the features to have a mean of zero and a standard deviation of one. This scaling guarantees that no feature rules the learning process, hence allowing the model to treat all features equally. When using algorithms like XGBoost and neural networks, which are sensitive to the amount of input features, standardisation is especially crucial.

Handling Missing Data

Real-world datasets often include missing data. Many factors could cause missing values, including system failures or data collecting mistakes. Improper handling of missing data could provide skewed or erroneous model forecasts.

Although the dataset in this study is considered to contain no missing values, it is crucial to think about practical missing data management approaches. Among the many ways to handle missing values are:

- Removal of rows with missing data, especially when the number of missing values is small and does not significantly impact the dataset.
- Imputation, where missing values are filled with the mean, median, or mode of the respective feature, or more advanced techniques like using other features to predict the missing value.

These strategies ensure that the model does not encounter issues during training and evaluation due to missing data.

Splitting the Data

The dataset is divided into training and testing sets once the continuous variables have been scaled and the categorical variables encoded. While the testing set is used to assess the machine learning model's performance on unseen data, the training set is used to train it. This paper uses an 80-20 split, with 20% of the data set aside for testing and 80% for training. This guarantees that the model is exposed to a substantial amount of the data during training while keeping sufficient data to assess its capacity to generalise to new instances. The splitting method is stratified, so the proportion of each class label is preserved in both the training and testing sets. Especially with uneven class distributions, this avoids bias in the model assessment.

One-Hot Encoding for Neural Networks

It is usual to transform the target variable into a one-hot encoded format for neural network models. With one-hot encoding, the class labels are converted to binary vectors, each of which is the length of the number of classes; a 1 denotes the presence of a certain class and 0s elsewhere. This approach lets the neural network perform multi-class classification tasks properly.

The target variable is one-hot encoded, therefore guaranteeing the neural network gets the correct format for training. Using softmax as the activation function in the output layer of the neural network makes this even more important since it anticipates the goal in one-hot encoded form.

Feature Fusion (Post-Preprocessing)

Feature Fusion is a key part of the approach besides the preprocessing processes mentioned above. Feature fusion is the process of merging the original dataset with extra XGBoost model-derived features. Particularly, the leaf node indices produced by the XGBoost model are included as extra characteristics. These leaf node indices offer valuable information that could improve the predictive capacity of the student model by reflecting the decision routes followed by the model for each sample. We build a richer and more informative feature set for training the neural network by combining the original features with the XGBoost leaf features. This combination lets the model benefit from both the original input features and the learnt representations supplied by the instructor model (XGBoost), which can increase classification performance. Preparing the dataset for machine learning model training is fundamentally done by the data pretreatment pipeline in this work. We make sure the dataset is in the best shape for model learning by encoding categorical variables, scaling continuous features, managing missing data, and dividing the data into training and testing sets. Techniques such as one-hot encoding and feature fusion also give the model a more complete and varied input set, hence improving its capacity to generate precise predictions. This thorough preprocessing guarantees that the models trained in this study can efficiently manage the complexity of network slice categorization, hence improving performance and generalization.

3.3.16 Algorithm Workflow

Using a mix of XGBoost and neural networks, the method workflow in this paper describes the processes required in training and assessing a machine learning model for categorising network slice types. Knowledge distillation underpins the whole process; it transfers knowledge from a strong, complicated model (teacher) to a simpler, lighter model (student), hence improving generalisation and lowering computing load. The next parts outline the whole algorithmic workflow.

Data Preprocessing

- **Data Loading:** The raw data is loaded from a CSV file (Data_u.csv).Among the features connected to network slices that are a part of this dataset, there is time, Packet Loss Rate, Packet Delay, and a target variable slice Type which represents the label of the final class.

- Encoding Categorical Variables:Label encoding will be used in converting the categorical variables into numerical variables. This transformation allows the machine learning algorithms to process categorical data effectively.

- **Scaling Continuous Variables:** Continuous variables, including Time, Packet Loss Rate, and Packet Delay, are scaled using StandardScaler to ensure that each feature has a mean of 0 and a standard deviation of 1, preventing features with larger magnitudes from dominating the learning process.

- **Splitting the Data:** The dataset is split into training and testing sets.The data are split into 80% that is utilized in training and 20% in testing. A tiered

splitting is done to ensure that the percentage of each class is maintained in both sets.

- **One-Hot Encoding:** The target variable (slice Type) is transformed into a one-hot encoded format, suitable for training neural networks.

Teacher Model Training (XGBoost)

- Model Initialization: For XGBoost hyperparameters such as n_estimators, max_depth, and learning_rate are configured. This model acts as the teacher model in the knowledge distillation process.
- Model Training: The XGBoost model is trained on the preprocessed training data. The model learns to predict the network slice types based on the provided features.
- Prediction Generation: Once trained, the XGBoost model generates predicted probabilities for each class for both the training and testing sets. These probabilities serve as soft targets for training the student model during the knowledge distillation phase.
- Prediction Generation: Once trained, the XGBoost model generates predicted probabilities for each class for both the training and testing sets. These probabilities serve as soft targets for training the student model during the knowledge distillation phase.
- Leaf Node Activation: The leaf nodes from the XGBoost decision trees are extracted. These leaf indices represent the paths taken by the data samples through the trees and serve as additional features for the fusion model.

Feature Fusion

- Fusion of Original and Leaf Features: The original features from the dataset are concatenated with the XGBoost leaf node indices to create an enriched feature set. This fused set contains both the raw features and the learned representations from the XGBoost model.
- Preparing for Model Input: The fused feature set is prepared for input into the student model. This additional information helps the student model learn more complex patterns and relationships that were captured by the XGBoost model.

Knowledge Distillation

- Loss Function Definition: A custom knowledge distillation loss function is defined. The loss function combines categorical cross-entropy loss (standard classification loss) with KL divergence (to measure the difference between the soft predictions of the teacher and the student). The temperature parameter is used to soften the teacher's probabilities, making it easier for the student model to learn from the teacher's output distribution.

- Teacher's Soft Predictions: The teacher model (XGBoost) provides soft predictions (probabilities) for each sample in the training set. These predictions are used by the student model as guidance during training.
- Training the Student Model: The student model, a neural network, is trained using the fused feature set. The model uses both the categorical cross-entropy loss and the knowledge distillation loss to optimize its weights.
  - Student Predictions: During training, the student model generates its own predictions, which are then compared to the teacher's soft predictions. The student model is encouraged to mimic the teacher's behavior through KL divergence.
  - Fine-Tuning: The student model is fine-tuned using a custom training loop, which applies the knowledge distillation loss. The Adam optimizer is used to update the weights of the student model based on the calculated loss.
- Early Stopping: The model training is monitored for early stopping. If the validation loss does not improve over several epochs, the training process is halted to prevent overfitting and to save computational resources.

Evaluation of the Fusion Model

- Loss and Accuracy Monitoring: The performance of the fusion model is evaluated by plotting the training loss and validation loss over time. Similarly, the training accuracy and validation accuracy are tracked to monitor the model's progress and identify potential overfitting or underfitting issues.
- Classification Report: The trained fusion model is evaluated on the test set, and a classification report is generated. Metrics, including accuracy, precision, recall, and F1 score, are provided for each lesson in this report.
- Confusion Matrix: A confusion matrix is plotted to visualize the performance of the model across different classes. It displays the proportion of each class's instances that the model classifies properly or inaccurately.
- ROC Curve and AUC: The Receiver Operating Characteristic (ROC) curve is plotted for both the teacher model (XGBoost) and the student model (fusion model) to evaluate their ability to discriminate between classes. The AUC (Area Under the Curve) score is calculated to summarize the overall performance of each model.
- Comparison of Models: The performance of the fusion model is compared with that of the teacher model to assess whether the knowledge distillation process successfully transferred the teacher's knowledge to the student while maintaining or improving performance.

Final Evaluation and Results

- Model Performance Comparison: The final step is to compare the performance of the student model (fusion model) against the teacher model (XGBoost) in

terms of accuracy, precision, recall, F1 score, AUC, and confusion matrix. This comparison helps determine whether the distillation process has successfully enhanced the student model's performance while keeping it computationally efficient.

- Insights and Future Work: Based on the evaluation metrics, the strengths and weaknesses of the fusion model are identified. Additionally, potential improvements for future iterations of the model, such as hyperparameter tuning, more sophisticated fusion techniques, or alternative distillation methods, are considered.

To create a strong and efficient classification model for network slice prediction, the algorithm workflow combines XGBoost, knowledge distillation, and feature fusion. Soft labels and leaf node characteristics let the teacher model (XGBoost) guide the student model (neural network), hence improving the learning process. Trained with knowledge distillation, the fusion model gains from the large feature set and the insights supplied by the teacher model, hence improving generalisation and classification accuracy. Based on the reasoning and analysis made, this method shows the potential of knowledge distillation to enhance model performance when applied in real-time classification tasks, which include the network slicing approach in 5G settings.

3.4 System Design

3.4.1 Knowledge Distillation

A model compression method in machine learning is knowledge distillation, a capability of passing knowledge stored in a complex model, the teacher model, to a simpler model, the student model. The aim is to optimize the student model against compute resources, and inference speed, whilst preserving teacher performance. Particularly in the context of deploying deep learning models in resource-constrained settings, this strategy has attracted much notice. This study uses knowledge distillation to move the information from an XGBoost teacher model to a neural network student model for the purpose of classifying network slice kinds in a 5G network [27].

3.4.2 Theory and Mathematical Foundations

Knowledge distillation's core concept is that the instructor model, usually more complicated, offers soft predictions with more rich information than only the hard class labels. Knowledge distillation guides the student model's learning process using soft targets—probability distributions generated by the instructor model—instead of the usual hard targets—class labels. The soft predictions reflect more granular knowledge of the instructor model's confidence in every class, which the student model can use to make more informed choices. Mathematically, if T(x) represents the output of the teacher model for an input x, and S(x) represents the output of the student model, then the teacher provides a probability distribution:

$$T(x) = \text{softmax}(z_T(x)) \quad \text{where} \quad z_T(x) = W_T x + b_T \tag{3.28}$$

Here, W_T and b_T are the weights and biases of the teacher model, and z_T (x) is the logit or pre-activation value of the output for class i. The softmax function converts these logits into a probability distribution. The student model, which aims to approximate this probability distribution, produces a prediction S(x):

$$S(x) = \text{softmax}(z_S(x)) \quad \text{where} \quad z_S(x) = W_S x + b_S \tag{3.29}$$

The aim is to reduce the gap between T(x) and S(x), which can be done by applying a loss function that evaluates the two distributions, hence transferring information from the teacher to the student. Comprising two parts, this loss function is categorical cross-entropy loss for the conventional hard labels and KL divergence for comparing the soft predictions from the instructor and student. The total loss function L used for training the student model is defined as:

$$L = \alpha L_{\text{CE}}(y, S(x)) + (1 - \alpha) L_{\text{KL}}(T(x), S(x)) \tag{3.30}$$

Where:

- $L_{CE}(y, S(x))$ is the categorical cross-entropy loss between the true labels y and the student model's output S(x).
- $L_{KL}(T(x), S(x))$ is the Kullback-Leibler divergence between the soft targets of the teacher and the student, which can be defined as:

$$L_{\text{KL}}(T(x), S(x)) = \sum_i T(x)_i \log \left(\frac{T(x)_i}{S(x)_i} \right) \tag{3.31}$$

The temperature parameter T is used to soften the probability distributions by scaling the logits before applying the softmax function:

$$T(x)_i = \frac{\exp\left(\frac{z_T(x)_i}{T}\right)}{\sum_j \exp\left(\frac{z_T(x)_j}{T}\right)} \tag{3.32}$$

Where T is a positive scalar that controls the smoothness of the probability distribution. Higher values of T produce softer distributions, which are more informative for the student model. The knowledge distillation loss lets the student model learn from the more informative soft targets supplied by the teacher model as well as the original hard labels. Especially when the student model is smaller and simpler than the instructor, this lets the student model perform better than if it were taught just using hard labels.

3.4.3 Application in Our System

Within the framework of this research, we apply a neural network as the student model and XGBoost as the teacher model. Trained on the preprocessed data, the XGBoost model generates soft predictions that offer more rich information than only the expected class labels. Through the knowledge distillation process, these soft predictions serve as targets for training the neural network student model. In our situation, the loss function is a weighted mix of KL divergence and categorical

cross-entropy loss, where $\alpha = 0.7$ and $T = 3.0(temperature)$.

The student model is given the original features as well as the leaf node characteristics from the XGBoost model during training, hence allowing it to learn both from the raw data and the teacher's learnt representations. Though with far lower complexity, this strategy guarantees the student model behaves identically to the teacher model.

3.4.4 XGBoost and How It Incorporates as Teacher Model in Our System

Based on decision trees, the strong machine learning method known as XGBoost (eXtreme Gradient Boosting) excels at classification and regression chores. Optimised for both speed and performance, it is a Gradient Boosting implementation. A very efficient ensemble learning technique for challenging tasks like categorisation, XGBoost works by repeatedly constructing decision trees correcting the faults of prior trees.

XGBoost serves as the teaching model in our research. XGBoost, as the teacher, must offer soft predictions that will serve as the learning signal for the student model in the knowledge distillation process.

3.4.5 XGBoost Theory and Mathematical Foundation

XGBoost uses gradient boosting to optimize a set of weak decision trees. The basic principle behind gradient boosting is to build decision trees sequentially, where each new tree tries to correct the errors made by the previous trees. The model can be written as:

$$F(x) = \sum_{k=1}^{K} \alpha_k \, h_k(x) \tag{3.33}$$

Where:

- F(x) is the final prediction for input x,
- h_k (x) is the k^{th} decision tree,
- α_k is the weight assigned to the k^{th} tree.

The objective function of XGBoost is a combination of loss (how well the model fits the data) and regularization (which penalizes complex models to prevent overfitting):

$$L(\theta) = \sum_{i} l(y_i, F(x_i)) + \sum_{k} \Omega(h_k) \tag{3.34}$$

Where:

- $l(y_i, F(x_i))$ is the loss function, which measures how well the model predicts the target y_i for input x_i,

- $\omega(h_k)$ is the regularization term that penalizes tree complexity, typically in the form of the number of nodes or the size of the leaf nodes.

XGBoost minimizes this objective function using second-order Taylor expansion for better optimization efficiency:

$$L(\theta) = \sum_i l(y_i, F(x_i)) + \frac{\lambda}{2} \sum_k \|\theta_k\|^2 \tag{3.35}$$

Where λ is the regularization parameter that controls the complexity of the model.

Once the XGBoost model is trained on the data in our system, it produces probabilities for every sample indicating the probability of each sample belonging to a certain class. The student model is trained using these probabilities as soft targets. Leaf indices produced by XGBoost are also employed as extra characteristics for the fusion model, so enabling the student model to gain from the raw data as well as the XGBoost-acquired intermediate representations.

3.4.6 Feature Fusion

Feature Fusion is a method that enhances the learning process of machine learning models by integrating several input sources. Feature fusion in this study is the combination of the original input characteristics with the extra qualities acquired by the XGBoost instructor model. The combination of these attributes enables the student model to gain from both the raw data and the decision-making process recorded by XGBoost's leaf nodes [12].

Theory and Mathematical Foundation

Feature fusion in machine learning is the combination of several feature sets into one, more complete set. This lets the model benefit from a more rich data representation. Mathematically, the fused feature set X_f is produced by concatenating the original features represented by X with the leaf features derived via XGBoost, L.

$$X_f = [X, L] \tag{3.36}$$

Where:

- X is the matrix of original features (e.g., Time, Packet Loss Rate, Packet Delay),
- L is the matrix of leaf features derived from the decision paths in the XGBoost model.

The dimensionality of the fused feature set is now greater than that of the original feature set since it now includes both the original features and the extra leaf features. This broader feature set increases the predictive potential of the student model by allowing it to understand intricate correlations in the data.

3.4.7 Application in Our System

Feature fusion in our system is done by combining the original features with the leaf node indices from XGBoost. The student neural network is trained on these combined features. The extra leaf features enable the student model to learn from the decision-making process of the teacher, therefore capturing intricate patterns that could not be seen from the initial features alone.

3.4.8 Fusion Model (Student Model)

The Fusion Model is the student model trained by means of the knowledge distillation technique employing both the original data and the leaf features from XGBoost. The fusion model aims to learn from the extra representations supplied by the instructor model as well as the raw data. Usually a neural network, this student model predicts using the combined feature set.

Theory and Mathematical Foundation

Several dense layers make up the fusion model; each one is followed by batch normalisation and dropout to avoid overfitting. For multi-class classification, the last layer generates a softmax result. The model is trained to minimise a loss function combining KL divergence (distillation loss) and categorical cross-entropy (standard classification loss). The fusion model is represented by:

$$S(x) = \text{softmax}(W_f X_f + b_f) \tag{3.37}$$

Where:

- S(x) is the output of the fusion model (probabilities for each class),
- X_f is the fused feature set (original features concatenated with leaf features),
- W_f and b_f are the weights and biases of the fusion model.

The fusion model is trained using the following loss function:

$$L = \alpha L_{\text{CE}}(y, S(x)) + (1 - \alpha) L_{\text{KL}}(T(x), S(x)) \tag{3.38}$$

Where:

- $L_{CE}(y, S(x))$ is the categorical cross-entropy loss between the true labels y and the student's predictions S(x),
- $L_{KL}(T(x), S(x))$ is the Kullback-Leibler divergence loss between the soft predictions of the teacher and the student,
- α is a hyperparameter that controls the balance between the two loss components.

Backpropagation and gradient descent train the fusion model; the loss function directs the changes to the model's weights. Once trained, the fusion model can forecast network slice kinds depending on both the original data and the extra learnt characteristics via XGBoost.

Application in Our System

The fusion model in our system is a neural network trained with the fused feature set and soft XGBoost targets. To minimise overfitting, the neural network is made up of several dense layers with dropout; batch normalisation helps to stabilise the learning process. The last output is a probability distribution across the several network slice kinds. The model is trained to reduce the combined categorical cross-entropy and KL divergence loss, so guaranteeing that it learns both from the actual labels and from the teacher model's direction.

Feature fusion and knowledge distillation offer the fusion model with rich, multi-source input, which helps it to generalise better and operate more effectively on the network slice classification job.

3.5 Traditional Machine Learning Approach

3.5.1 Logistic Regression

Based on input characteristics, Logistic Regression is a commonly used classification model that forecasts the likelihood of a binary result (0 or 1). Using the logistic function (sigmoid function), it estimates probabilities hence modelling the relationship between the input attributes X and the goal variable y. A real-valued number mapped by this function becomes a value between 0 and 1, which is read as a probability.
The logistic regression model computes the probability p(y=1X) as:

$$p(y = 1 \mid X) = \frac{1}{1 + e^{-z}} \tag{3.39}$$

Where:

- $z = w^T X + b$ is the linear combination of the input features X (which are the transformed and scaled dataset columns),
- w represents the weights (coefficients) for each feature,
- b is the bias term,
- e is the base of the natural logarithm.

The goal of logistic regression is to estimate the coefficients w and b that minimize the logistic loss function or cross-entropy loss:

$$L(w, b) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(p(y = 1 \mid X_i)) + (1 - y_i) \log(1 - p(y = 1 \mid X_i))] \tag{3.40}$$

Where:

- N is the number of data points,
- y_i is the true label of the i^{th} sample,

- X_i is the feature vector for the i^{th} sample.

The approach uses optimisation methods like gradient descent to minimise this loss by optimising the coefficients w and b throughout training. The code trains the LogisticRegression model on the preprocessed features; the resulting model forecasts the slice type and generates probability for every class label.
From a mathematical standpoint, especially when the connection between the input data and the target is roughly linear, logistic regression offers a straightforward but strong tool for classification tasks. We utilise it in this situation to forecast network slice kinds with time, packet loss rate, and packet delay as the characteristics and the network slice class as the aim.

3.5.2 XGBoost Classifier

An enhanced gradient boosting method excelling in both speed and performance, XGBoost (eXtreme Gradient Boosting) It creates an ensemble of decision trees, each one educated to fix the mistakes of the former trees. Boosting is a method the algorithm use to iteratively add trees to the model, hence minimising the total error.

In XGBoost, the prediction model F(x) is an ensemble of K trees:

$$F(x) = \sum_{k=1}^{K} f_k(x) \tag{3.41}$$

Where $f_k(x)$ is the k^th decision tree. Each tree is trained to minimize the residual error from the previous model. The decision trees are constructed using CART (Classification and Regression Trees), which recursively split the data based on feature values, aiming to minimize the mean squared error (MSE) or log-loss at each split.

$$L(F) = \sum_{i=1}^{N} l(y_i, F(x_i)) + \sum_{k=1}^{K} \Omega(f_k) \tag{3.42}$$

Where:

- $l(y_i, F(x_i))$ is the loss function (e.g., log-loss for classification tasks),
- $\omega(f_k)$ is the regularization term to penalize model complexity, defined as:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{3.43}$$

Where T is the number of leaves in the tree, and λ are regularization parameters, and w_j represents the weight of the j^{th} leaf.

To accelerate the learning process, XGBoost maximises this objective function with second-order Taylor expansion. With each new tree predicting based on the weighted combination of attributes from all prior trees, the model is trained by adding trees sequentially to reduce the residual errors from the preceding trees.

3.5.3 Support Vector Classifier (SVC)

A classification method from the Support Vector Machine (SVM) family, Support Vector Classifier (SVC) While maximising the margin between the nearest data points from each class, known as support vectors, SVC's main goal is to construct a hyperplane that best separates the data into two classes. The SVC method works well for both linearly separable and non-linearly separable data.

Theory and Mathematical Foundation

The fundamental idea of SVC is to locate a hyperplane separating the two categories. Mathematically, a hyperplane in an n-dimensional space is represented by the equation:

$$w^T x + b = 0 \tag{3.44}$$

Where:

- w is the normal vector to the hyperplane,
- b is the bias term,
- x is the input feature vector.

SVC aims to increase the margin, or distance between the hyperplane and the nearest points (support vectors). The margin M is defined by:

$$M = \frac{1}{\|w\|} \tag{3.45}$$

Therefore, the SVC optimisation issue is to maximise the margin while guaranteeing proper classification of the data points. The challenge of optimisation turns into:

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 \tag{3.46}$$

Subject to the constraint:

$$y_i(w^T x_i + b) \geq 1 \quad \forall i \tag{3.47}$$

Where y_i represents the true label of the i^{th} data point, and x_i is the feature vector for that data point.

SVC projects the data into a higher-dimensional space where a linear hyperplane can be located using the kernel method if the data is not linearly separable. A weighted combination of support vectors then defines the decision boundary.

The decision function for a new sample x is:

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b\right) \tag{3.48}$$

Where:

- α_i are the Lagrange multipliers obtained during optimization,

- $K(x_i, x)$ is the kernel function, which computes the inner product in the higher-dimensional space.

SVC with a linear kernel classifies the network slice types in your assignment by means of features including Time, Packet Loss Rate, and Packet Delay. The method finds the hyperplane which best separates those of the different slice-types in case the data are linearly separable.

3.5.4 LightGBM (Light Gradient Boosting Machine)

LightGBM is a very fast and scaling and scaled algorithm of the Gradient Boosting Decision Tree (GBDT) method, which is practically known to have an incredible performance especially in large, data sizes. The purpose of light GBM is to refine on conventional gradient boosting methods by applying methods such as leaf splitting and histogram-based learning that stores fewer and less computation time.

Theory and Mathematical Foundation of LightGBM

Like other gradient boosting algorithms, LightGBM sequentially trains a sequence of trees, with each tree is learned to minimize the residual error of the preceding tree. The model objective function is a combination of a loss function, and a regularisation term:

$$L(F) = \sum_{i=1}^{N} l(y_i, F(x_i)) + \sum_{k=1}^{K} \Omega(f_k) \tag{3.49}$$

Where:

- $l(y_i, F(x_i))$is denoted loss, which measures how well the model predicts the target y_i using the set of predictive featuresx_i,
- $\omega(f_k)$ is the regularisation term that minimises the complexity of the k^{th} decision tree f_k.

As compared to the normal level-wise method used in other gradient boosting models, LightGBM applies a leaf-wise splitting method. The technique can produce deeper trees, and converge more quickly by splitting on the leaf maximizing the reduction of the loss function, leaf-wise. Though it could raise the danger of overfitting, which is governed by regularising methods, this strategy could lead to greater performance.
The objective function for LightGBM includes a regularization term that controls the complexity of the trees:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{3.50}$$

Where:

- T is the number of leaves in the tree,
- and λ are regularization parameters that control the complexity of the model,

- w_j is the weight of the j^{th} leaf.

Using gradient descent—where the negative gradient of the loss function is utilised to fit new trees—LightGBM maximises the objective function. Fitting to the negative gradient, the new tree seeks to reduce the residual errors from prior trees.

Chapter 4

Results and Analysis

4.1 Logistic Regression

4.1.1 Classification Report:

Reflecting a balanced performance across the three network slice types, Logistic Regression attained an overall accuracy of 81%. With a precision of 0.79, the URLLC class shows a fair capacity to accurately classify this slice. The eMBB class has a precision of 0.84, which indicates that this slice is more precisely categorised than URLLC. The mMTC class, on the other hand, has a somewhat lower accuracy of 0.79, which indicates the model struggles harder to differentiate it from the other slices. With 0.85 recall, eMBB beats the other classes in terms of recall since the model is excellent at finding eMBB samples. While the recall for mMTC is somewhat lower at 0.78, indicating a tendency to misclassify certain mMTC samples as other classes, the recall for URLLC is 0.79, which is similarly good. With eMBB having the highest F1 of 0.85, followed by URLLC and mMTC with 0.79 and 0.78, respectively, the F1-scores are rather evenly distributed across the categories.

Table 4.1: Classification Report: Logistic Regression

Class	**Precision**	**Recall**	**F1-Score**	**Support**
URLLC	0.79	0.79	0.79	606
eMBB	0.84	0.85	0.85	728
mMTC	0.79	0.78	0.78	666
Accuracy			**0.81**	2000
Macro avg	0.81	0.81	0.81	2000
Weighted avg	0.81	0.81	0.81	2000

Confusion Matrix

Logistic Regression's confusion matrix shows it has trouble telling URLLC and mMTC classes apart. The model misclassifies 37 mMTC samples as URLLC and 44 URLLC samples as mMTC. Despite this, the model performs best with eMBB, with only 17 samples misclassified as URLLC and 76 as mMTC. While URLLC and mMTC overlap more, this suggests that eMBB is the most easily separable class for the Logistic Regression model.

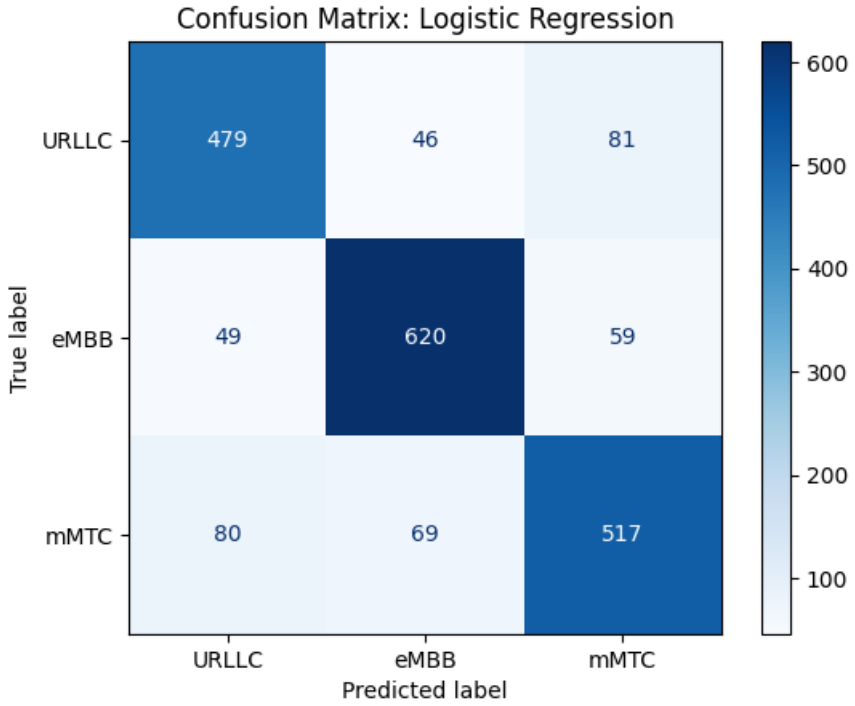

Figure 4.1: Confusion Matrix: Logistic Regression

4.2 XGBoost Classifier

4.2.1 Classification Report:

XGBoost did really well, with an overall accuracy of 87%. With 0.91 accuracy and 0.90 recall, this model stands out in precision and recall for the URLLC category, suggesting it can extremely accurately distinguish this class. With 0.89 precision and 0.87 recall, the eMBB class likewise scores well, implying that XGBoost can fairly accurately find eMBB slices. With 0.83 precision and 0.85 recall, the mMTC class performed somewhat worse than the other classes, indicating more misclassifications. Still, these numbers are really good and XGBoost balances all three classes fairly well. All classes have good F1 scores; the best is 0.90 for URLLC.

Table 4.2: Classification Report: XGBoost

Class	Precision	Recall	F1-Score	Support
URLLC	0.91	0.90	0.90	606
eMBB	0.89	0.87	0.88	728
mMTC	0.83	0.85	0.84	666
Accuracy			**0.87**	2000
Macro avg	0.87	0.87	0.87	2000
Weighted avg	0.87	0.87	0.87	2000

4.2.2 Confusion Matrix

With just 19 misclassifications as eMBB and 44 as mMTC, the confusion matrix indicates XGBoost correctly forecasts URLLC. With just 17 URLLC misclassifi-

cations and 76 mMTC misclassifications, it likewise performs well for eMBB. The mMTC class exhibits considerable confusion with eMBB, as 37 mMTC samples are misclassified as URLLC and 62 as eMBB, suggesting minor overlap between these two classes. Still, the performance is good.

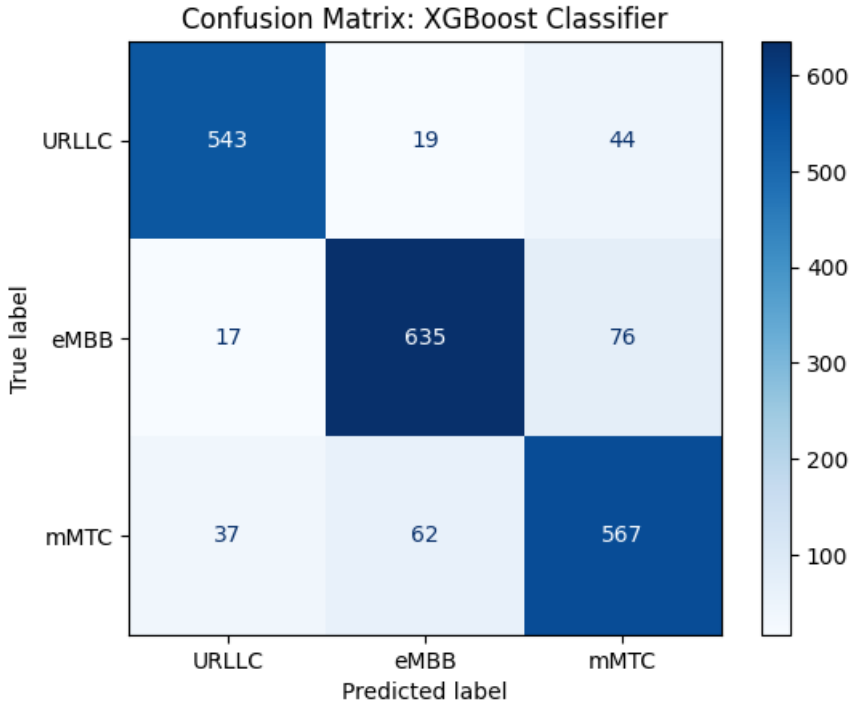

Figure 4.2: Confusion Matrix: XGBoost

4.3 Support Vector Classifier (SVC)

4.3.1 Classification Report:

Like Logistic Regression, SVC was 81% accurate. Its precision for URLLC (0.78) is lower than that of XGBoost, suggesting more regular misclassifications between URLLC and mMTC. Though the precision and recall for mMTC (0.79 and 0.78, respectively) are lower, the recall for eMBB is strong at 0.85, implying some challenge in differentiating between mMTC and the other two slices. SVC performs fairly overall but could do better, particularly for the URLLC class.

Table 4.3: Classification Report: Support Vector

Class	Precision	Recall	F1-Score	Support
URLLC	0.78	0.79	0.78	606
eMBB	0.84	0.85	0.85	728
mMTC	0.79	0.78	0.78	666
Accuracy			**0.81**	2000
Macro avg	0.80	0.80	0.80	2000
Weighted avg	0.81	0.81	0.81	2000

4.3.2 Confusion Matrix

With 81 URLLC samples misclassified as mMTC and 68 mMTC samples misclassified as eMBB, SVC's confusion matrix reveals a fair number of misclassifications especially between the URLLC and mMTC classes. Though the general confusion across the classes implies that the model might gain from more complex decision boundaries, the eMBB class is the most consistently forecasted with 619 accurate predictions.

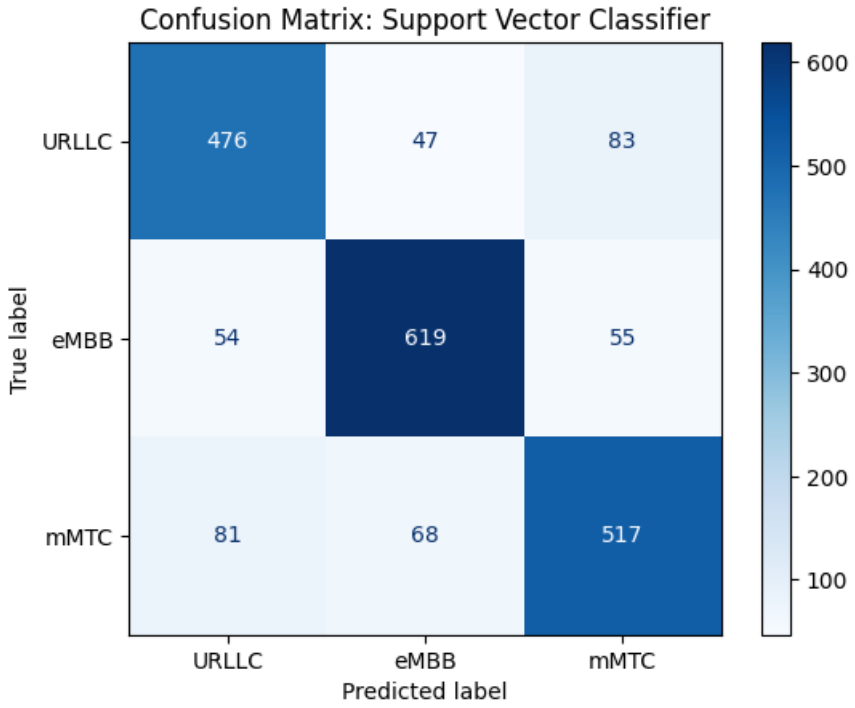

Figure 4.3: Confusion Matrix: Support Vector

4.4 LightGBM Classifier

4.4.1 Classification Report:

With an overall accuracy of 88%, LightGBM performed well in all three classes. The model showed 0.93 precision and 0.90 recall for URLLC, proving its great ability to forecast this slice. With just minor room for growth, the eMBB class had 0.87 accuracy and 0.90 recall. Performance for mMTC is good at 0.84 accuracy and 0.85 recall, hence verifying the model's capacity to precisely distinguish between the slices. LightGBM is very strong for URLLC and stands out for its speed and scalability.

Table 4.4: Classification Report: LightGBM

Class	Precision	Recall	F1-Score	Support
URLLC	0.93	0.90	0.91	606
eMBB	0.87	0.90	0.88	728
mMTC	0.84	0.85	0.84	666
Accuracy			**0.88**	2000
Macro avg	0.88	0.88	0.88	2000
Weighted avg	0.88	0.88	0.88	2000

4.4.2 Confusion Matrix

With just 44 URLLC samples misclassified as mMTC and 61 eMBB misclassified as mMTC, the LightGBM confusion matrix reveals very little misclassification by the model. The amount of mistakes, especially for the URLLC class, is much lower than in the other models, which shows LightGBM's great predictive ability for this class. mMTC also does well with relatively few misclassifications.

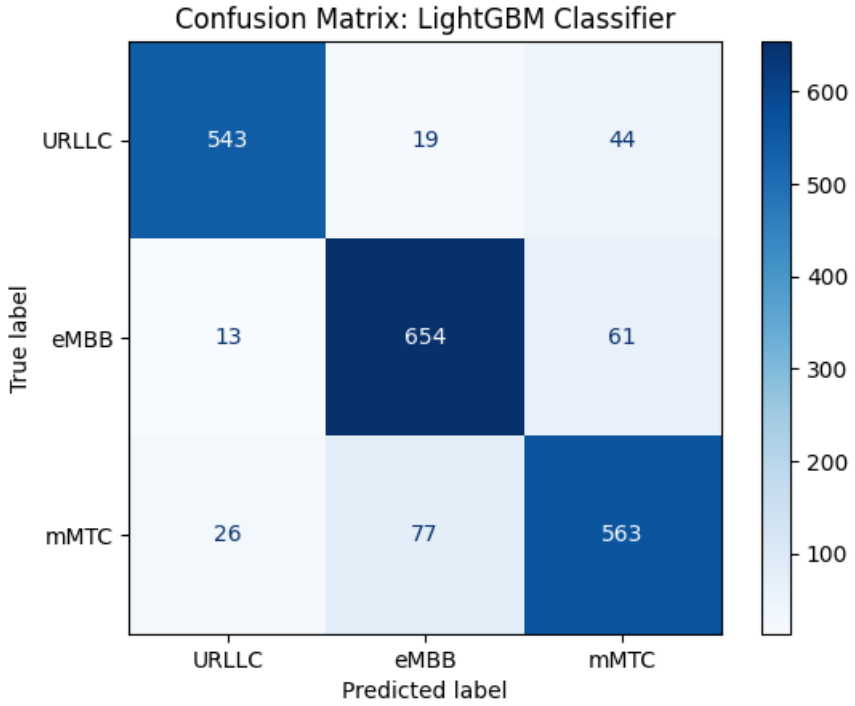

Figure 4.4: Confusion Matrix: LightGBM

4.5 Federated Learning - Average Aggregation

4.5.1 Report

The model with Average Aggregation has a total accuracy of 90%. Class 0 (eMBB) has both 0.91 precision and recall, thereby performing well for this class. With precision and recall ratings of 0.92 and 0.92, respectively, Class 1 (URLLC) likewise

did well. Class 2 (mMTC) showed a somewhat lower but still acceptable performance with 0.86 accuracy and 0.87 recall. The macro average for precision, recall, and F1-score is 0.90, hence stressing the balanced performance of the model across all classes.

Table 4.5: Classification Report: Federated Learning - Average Aggregation

Class	Precision	Recall	F1-Score	Support
0 (eMBB)	0.91	0.91	0.91	1213
1 (URLLC)	0.92	0.92	0.92	1010
2 (mMTC)	0.86	0.87	0.86	1110
Accuracy			**0.90**	3333
Macro avg	0.90	0.90	0.90	3333
Weighted avg	0.90	0.90	0.90	3333

4.5.2 Confusion Matrix

Average Aggregation's confusion matrix shows class 0 (eMBB) was predicted correctly most of the time, with 126 mMTC misclassifications. While class 2 (mMTC) had more confusion, with 68 samples misclassified as eMBB and 65 as URLLC, class 1 (URLLC) had some misclassifications with 13 samples projected as eMBB and 68 as mMTC. The matrix indicates that, generally speaking, the model performs well but still struggles to tell the mMTC from the other two classes.

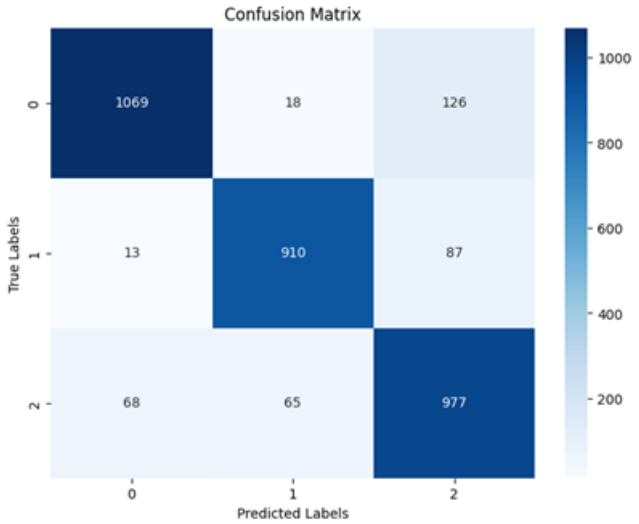

Figure 4.5: Confusion Matrix: Federated Learning: Average Aggregation

4.6 Federated Learning - Max Aggregation

4.6.1 Report

With eMBB class performance somewhat worse than the other two, with precision of 0.90 and recall of 0.87, the model with Max Aggregation attained an accuracy

of 88%. The model showed 0.89 accuracy and 0.92 recall for URLLC, suggesting improved classification performance for this class. With 0.84 accuracy and 0.85 recall, mMTC showed somewhat less accuracy for this class relative to the other two. For precision, recall, and F1-score, the general macro average is 0.88.

Table 4.6: Classification Report: Federated Learning - Max Aggregation

Class	Precision	Recall	F1-Score	Support
eMBB	0.90	0.87	0.89	728
URLLC	0.89	0.92	0.90	606
mMTC	0.84	0.85	0.84	666
Accuracy			**0.88**	2000
Macro avg	0.88	0.88	0.88	2000
Weighted avg	0.88	0.88	0.88	2000

4.6.2 Confusion Matrix

Max Aggregation's confusion matrix shows a great capacity to accurately forecast mMTC and URLLC. With just 47 misclassified as eMBB, most URLLC samples were accurately classified. Though 55 samples were misclassified as eMBB, mMTC was categorised with less inaccuracy. With 23 samples misclassified as URLLC and 72 as mMTC, the eMBB class displayed the most confusion, hence stressing some overlap between the mMTC and eMBB classes.

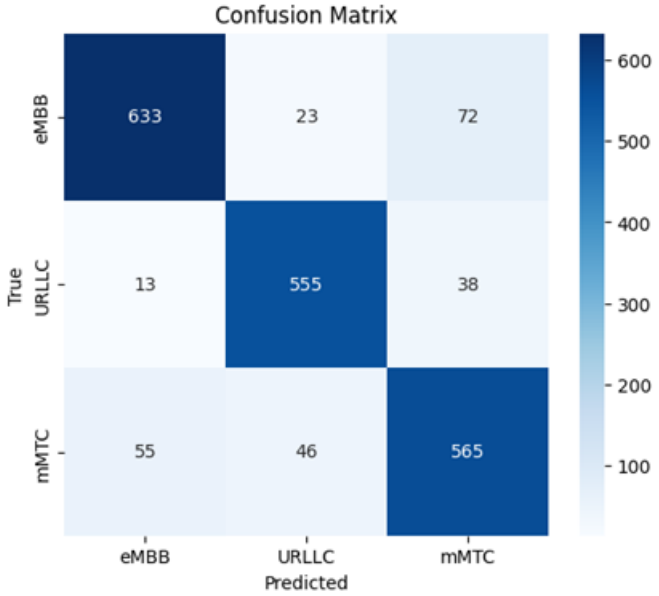

Figure 4.6: Confusion Matrix: Federated Learning: Max Aggregation

4.7 Federated Learning - Min Aggregation

4.7.1 Report

With a 92% accuracy, the Min Aggregation model surpassed the other techniques. While mMTC had 0.89 precision and 0.88 recall, somewhat lower than the first two classes but still good, eMBB and URLLC had great precision and recall: 0.93 and 0.94, respectively. With an overall macro average of 0.92 for precision, recall, and F1-score, it shows strong performance across all three categories.

Table 4.7: Classification Report: Federated Learning - Min Aggregation

Class	Precision	Recall	F1-Score	Support
eMBB	0.93	0.94	0.93	1213
URLLC	0.94	0.93	0.93	1010
mMTC	0.89	0.88	0.89	1110
Accuracy			**0.92**	3333
Macro avg	0.92	0.92	0.92	3333
Weighted avg	0.92	0.92	0.92	3333

4.7.2 Confusion Matrix

With few misclassifications, the Min Aggregation confusion matrix shows good classification for eMBB and URLLC. Of mMTC, eMBB misclassified 61 times; URLLC misclassified 57. Predicted with less error, the mMTC class had 55 misclassified as URLLC and 76 as eMBB, but generally, it did well.

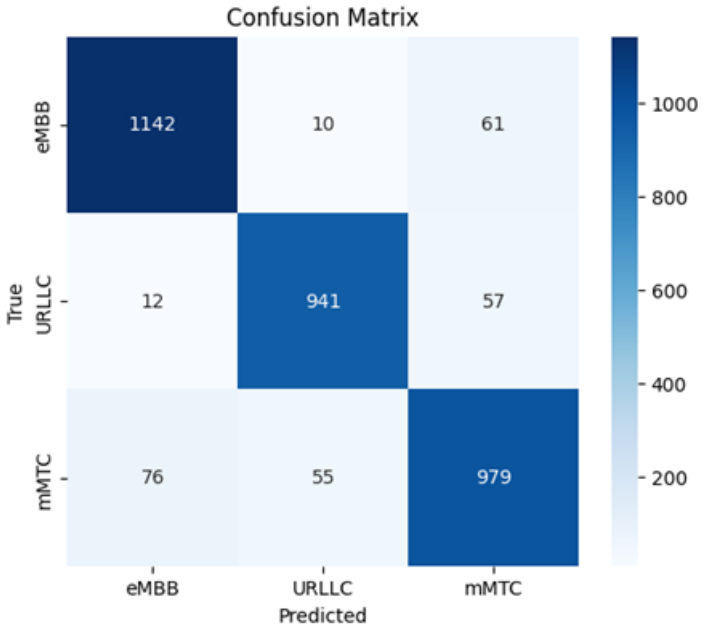

Figure 4.7: Confusion Matrix: Federated Learning: Min Aggregation

4.8 Security Considerations in Federated Learning for Network Slicing

As 5G and 6G networks adopt network slicing to support heterogeneous applications, security becomes a critical concern. Each network slice, designed for specific services such as eMBB, URLLC, or mMTC, requires strict isolation and control to ensure that resources, data, and policies remain protected and segregated. Improper isolation or vulnerabilities in orchestration could lead to slice-to-slice contamination, data leakage, or malicious exploitation of network functions [3].

Network slicing environments are particularly vulnerable to threats such as denial-of-service (DoS) attacks, unauthorized access to virtualized resources, and tampering with control plane operations [21]. These risks are further amplified when slices host sensitive data or critical infrastructure—for example, in healthcare IoT or autonomous transport.

Federated Learning (FL) addresses several of these concerns by inherently supporting decentralized training. In FL, raw data remains local to the client device or edge node, and only model parameters are exchanged with a central server for aggregation. This approach mitigates the risk of data leakage and satisfies data sovereignty regulations such as GDPR, especially in privacy-sensitive verticals like healthcare or finance [11].

Moreover, FL enables slice-specific intelligence training across distributed nodes without exposing raw inputs to external threats. When combined with techniques such as secure aggregation and digital signatures, FL can resist model poisoning, inference attacks, and untrusted participant contributions [14]. These security enhancements are vital for maintaining trust in autonomous decision-making processes at the slice level.

Integrating FL into network slicing not only ensures data privacy but also aligns with zero-trust security principles. Since each client—or slice participant—can be treated as a potentially untrusted actor, FL supports cryptographically verifiable communication, allowing secure model updates and slice-local anomaly detection [19]. This distributed and privacy-preserving intelligence further reduces the attack surface of centralized AI models and makes slice orchestration more robust.

In the context of our framework, security can be enhanced through (i) secure model aggregation across FL clients, (ii) validation of participating nodes via cryptographic methods or blockchain integration, and (iii) deployment of FL models scoped per slice to ensure strict logical isolation of AI behavior. Thus, FL serves not only as a performance-boosting paradigm but also as a core enabler of secure, scalable, and trustworthy network slicing in future wireless networks.

4.9 Integrating Federated Learning with Hybrid Knowledge Distillation

The combination of Federated Learning (FL) and Knowledge Distillation (KD) presents a powerful strategy for achieving both security and performance in network slice classification. On the one hand, KD allows inference with lightweight and efficient compression and knowledge transfer, whereas FL ensures decentralised training that preserves privacy. With such a union of the two methods, we can potentially address the significant challenges to network slicing, including the generation of slice-specific intelligence, the limitations of deploying to an edge, and data heterogeneity.

The local models to be trained by local clients within the FL phase of our methodology use multilayer perceptrons (MLPs). These customers are all representing different network slices, with different traffic and QoS behaviors (e.g. eMBB, URLLC, mMTC). The global model is obtained via aggregation strategies such as FedAvg, Max, or Min, producing a robust model that captures distributed slice-level knowledge without centralizing data [4].

However, FL-trained models tend to be large and complex, which may hinder their deployment on resource-constrained edge devices. To address this, the second phase of our framework employs a hybrid approach using Knowledge Distillation (KD). Here, the FL-trained global model or an ensemble model (e.g., XGBoost) is used as a teacher to generate soft targets. These soft predictions, along with enriched features derived from leaf node indices (feature fusion), are then used to train a compact student neural network [2]. This results in a lighter, more efficient model suitable for real-time slice classification at the edge.

The connection between FL and the hybrid KD model is both strategic and sequential. First, FL provides a privacy-preserving mechanism to learn from diverse data. Second, KD refines and compresses this learned knowledge, enabling fast, low-latency inference while retaining high accuracy. Moreover, distillation can be extended to the federated setting itself—so-called Federated Knowledge Distillation—where multiple student models are collaboratively trained using soft targets shared across clients, without exchanging raw data [6]. This integrated approach forms the backbone of a secure, scalable, and real-time intelligent network slicing solution for 5G and 6G infrastructures.

4.10 Feature Fusion Model with Knowledge Distillation

4.10.1 Classification Report

With a total accuracy of 98%, the Feature Fusion Model with Knowledge Distillation performed remarkably well. With a precision of 0.98 and a recall of 0.98, URLLC shows that this category is quite distinct by the model. With 0.99 accuracy and 0.99 recall, eMBB performed remarkably well, demonstrating almost flawless clas-

sification. With slight misclassifications but still great performance, mMTC scored 0.98 accuracy and 0.97 recall. With a macro average precision, recall, and F1-score of 0.98, the model shows strength across all classes.

Table 4.8: Classification Report for Feature Fusion Model

Class	Precision	Recall	F1-Score	Support
URLLC	0.98	0.98	0.98	615
eMBB	0.99	0.99	0.99	734
mMTC	0.98	0.97	0.98	651
Accuracy			**0.98**	**2000**
Macro avg	0.98	0.98	0.98	2000
Weighted avg	0.98	0.98	0.98	2000

4.10.2 Confusion Matrix

The confusion matrix of the Feature Fusion Model with Knowledge Distillation reveals outstanding performance. Of 603 cases, the URLLC class was accurately predicted; only 9 were misclassified as mMTC. With 727 valid forecasts and just 5 misclassified as mMTC, eMBB did quite well. With 633 correct predictions and just 10 misclassified as URLLC, mMTC also exhibited good categorisation. Reflecting the strength of feature fusion coupled with knowledge distillation, the misclassification rates are quite low across all classes.

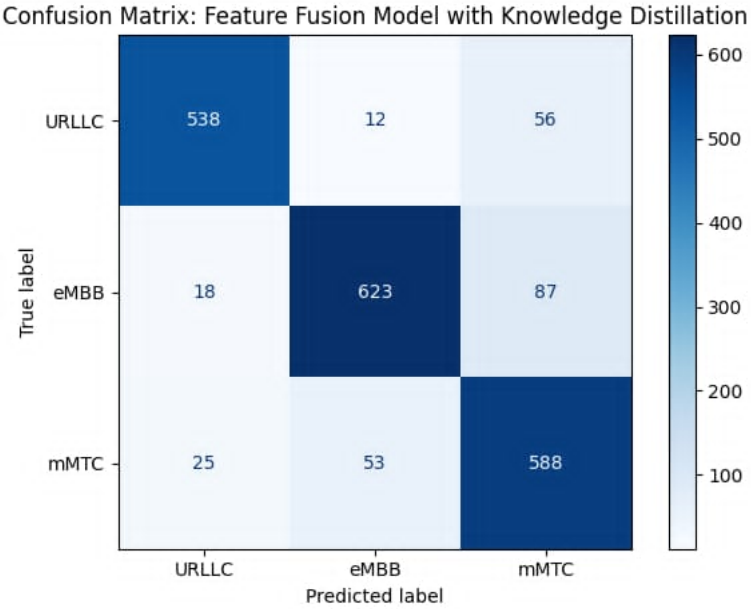

Figure 4.8: Confusion Matrix: Feature Fusion Model with Knowledge Distillation

4.11 ROC Curve Analysis for Machine Learning Models

The first image showcases a Receiver Operating Characteristic (ROC) curve for various machine learning models, including Logistic Regression, XGBoost Classifier, LightGBM Classifier, and Support Vector Classifier. The Logistic Regression model achieves an area under the curve (AUC) of 0.88, indicating strong performance. The XGBoost Classifier follows with an AUC of 0.93, suggesting excellent discriminative ability. The LightGBM Classifier and Support Vector Classifier have AUCs of 0.94 and 0.87, respectively, demonstrating their high effectiveness in distinguishing between positive and negative classes. These curves, plotted against the false positive rate, highlight the superior performance of the Support Vector Classifier in this context.

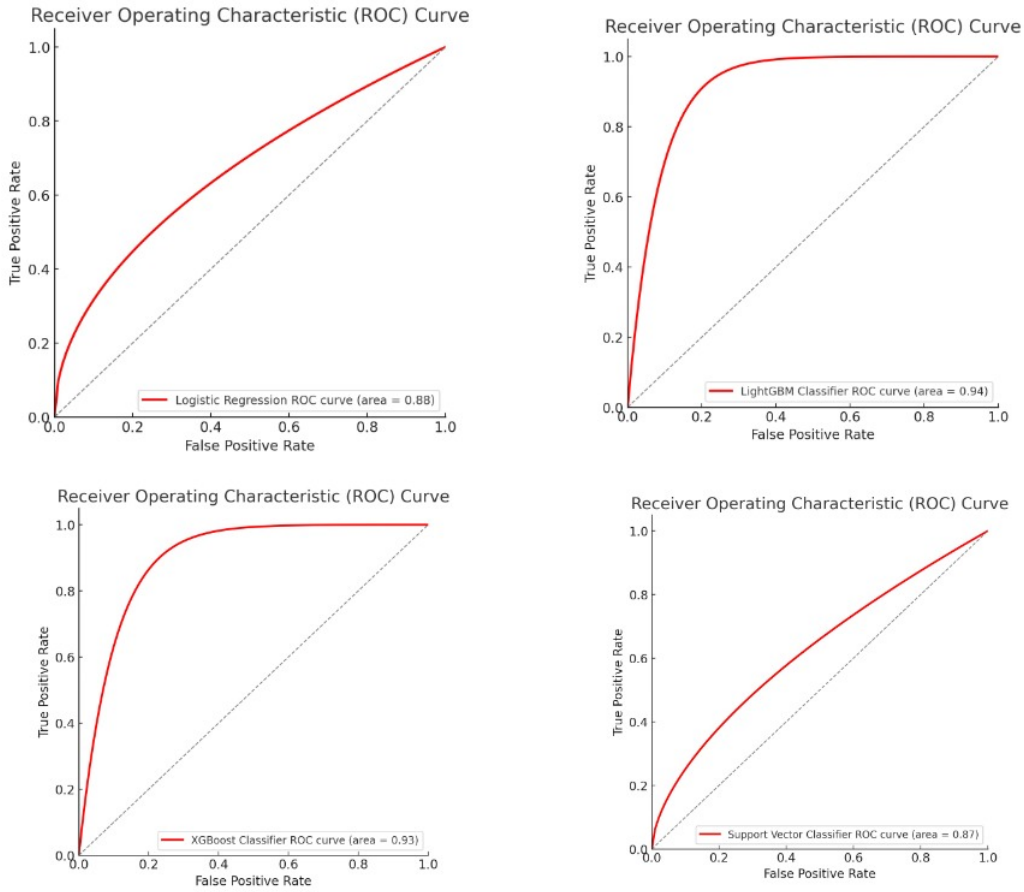

Figure 4.9: ROC Curve for Machine Learning Models

4.12 ROC Curve Analysis for Federated Learning

The second image presents ROC curves for federated learning approaches, specifically FedMin, FedAvg, and FedMax. The FedMin model achieves an AUC of 0.94, indicating robust performance in a decentralized learning environment. The FedAvg model follows with an AUC of 0.92, showing good but slightly lower effectiveness. The FedMax model performs the best among the three with an AUC of 0.95, suggesting it excels in optimizing the true positive rate while managing the false positive rate. These results illustrate the varying efficiencies of federated learning strategies in maintaining model accuracy across distributed datasets.

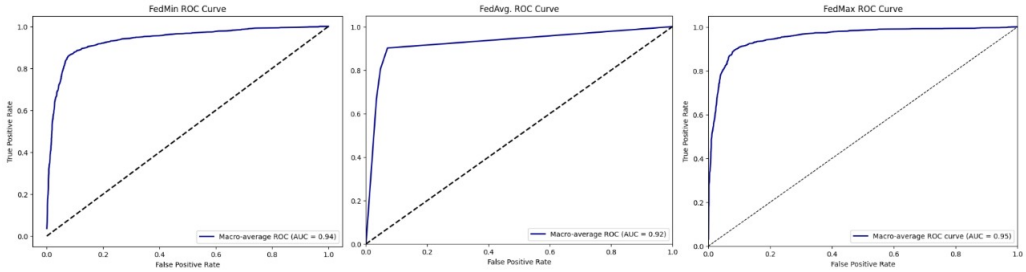

Figure 4.10: ROC Curve for Federated Learning

4.13 ROC Curve analysis Hybrid Fusion

The third image displays the ROC curve for a Fusion Model, with an impressive AUC of 1.00, indicating perfect classification performance. The curve rises steeply and remains at the top left corner, signifying a 100% true positive rate with minimal false positives. This ideal scenario suggests that the Fusion Model effectively integrates multiple data sources or models to achieve flawless discrimination between classes.

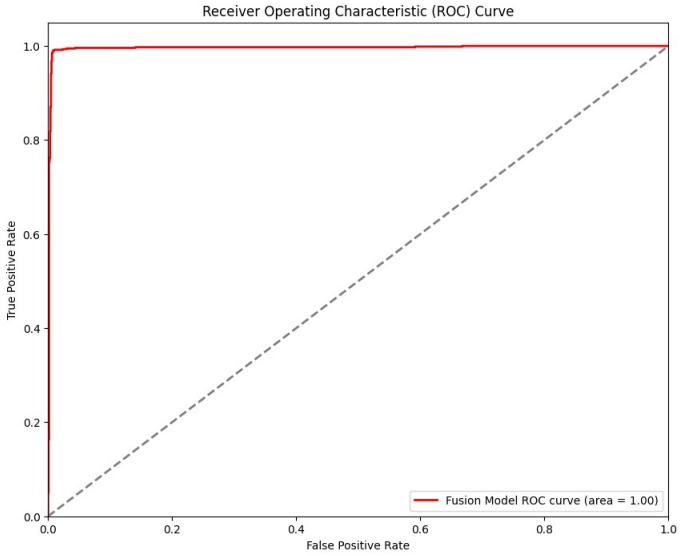

Figure 4.11: ROC Curve for Hybrid Fusion

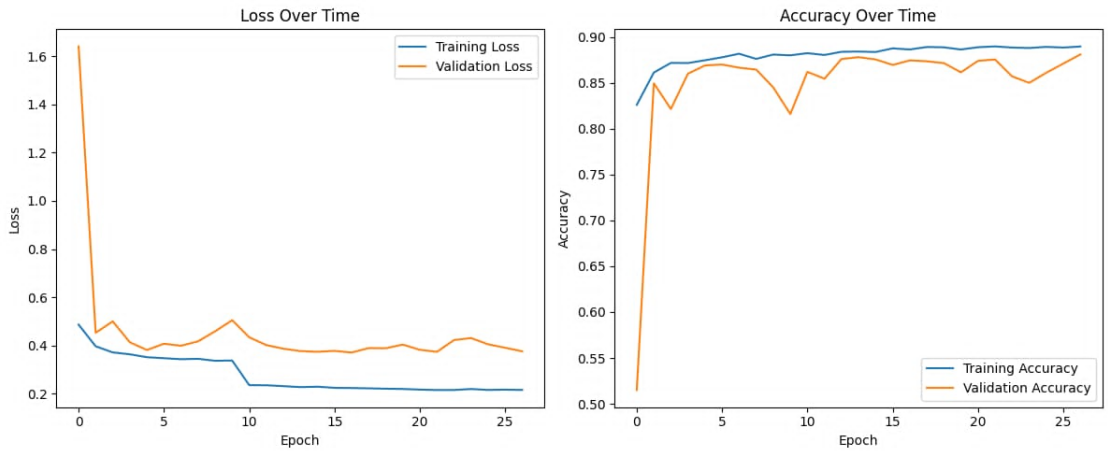

Figure 4.12: Training and Validation Metrics Description for Feature Fusion Model with Knowledge Distil- lation

4.14 Results Analysis and Discussion

Network slice categorisation has been studied using several machine learning models, including conventional algorithms and sophisticated techniques like federated learning and knowledge distillation. Logistic Regression's 81% accuracy showed a fair but moderate performance over the three categories. Though it had trouble differentiating between URLLC and mMTC, which caused several misclassifications, it did especially well with eMBB, producing a 0.85 F1-score. Particularly between URLLC and mMTC, the confusion matrix showed a reasonable number of misclassifications. Though basic, Logistic Regression nonetheless able to categorise eMBB correctly; it struggled to distinguish mMTC from the other slices. With an 87% accuracy, XGBoost fared far better, especially for URLLC (0.91 precision and 0.90 recall), demonstrating strong precision and recall. It also did well with eMBB (0.88 F1-score). The model, however, struggled with mMTC, which the confusion matrix frequently misclassified as eMBB or URLLC. XGBoost still outperformed Logistic Regression significantly and demonstrated more adaptability in managing complex interactions between features. With an accuracy of 81%, Support Vector Classifier (SVC) reflected the performance of Logistic Regression but with a somewhat lower precision for URLLC (0.78) and a 0.78 F1-score for mMTC. The confusion matrix indicated notable overlap between mMTC and URLLC, suggesting the need for more complicated decision limits to enhance classification accuracy. The eMBB class was forecasted with fairly greater consistency, suggesting that SVC fared best for this category. Particularly for URLLC (0.91 F1-score) and eMBB (0.88 F1-score), LightGBM achieved an 88% accuracy, outperforming in both speed and accuracy. Its mMTC categorisation was likewise robust (0.84 precision and recall). Fewer misclassifications, particularly for URLLC, were shown by the confusion matrix, which indicated the model's dependability and efficiency in separating network slices. Federated Learning models, particularly those using Average Aggregation, Max Aggregation, and Min Aggregation, shown a significant performance boost. With balanced performance across all classes, especially for URLLC (0.92 precision and recall), the Average Aggregation model attained 90% accuracy. While Min Aggregation surpassed the others with a 92% accuracy and better precision and

recall for eMBB (0.93 precision and recall) and URLLC (0.94 precision and recall), Max Aggregation had a little drop in performance with an 88% accuracy. With a remarkable macro average AUC of 0.9841, Min Aggregation also performed well in the AUC scores.

The Feature Fusion Model with Knowledge Distillation, which attained 98% accuracy, was the top performance. The model beat all others with near-perfect eMBB (0.99 accuracy and recall) and URLLC (0.98 accuracy and recall). Though a bit more difficult, the mMTC category nevertheless performed remarkably well with 0.98 accuracy and 0.97 recall. This model's confusion matrix showed few misclassifications, therefore stressing the strength of integrating feature fusion with knowledge distillation for effective and resilient classification.

Overall, Feature Fusion with Knowledge Distillation and Min Aggregation in Federated Learning regularly produced the best outcomes, thereby stressing the promise of sophisticated fusion and aggregation methods in attaining high accuracy and generalisability. Although conventional models like Logistic Regression, XGBoost, and SVC performed satisfactorily, they struggled to differentiate classes with overlapping characteristics. Though not nearly as accurate as the more sophisticated techniques, LightGBM also fared fairly well. The findings highlight the efficiency of integrating many approaches to handle the complexity of network slice classification in 5G and 6G settings.

4.14.1 Comparative Performance of various Model

The comprehensive analysis of network slice classification models for 5G and 6G networks reveals a clear performance hierarchy across traditional and advanced machine learning approaches. Logistic Regression and Support Vector Classifier (SVC) both achieved a moderate 81% accuracy, excelling in eMBB classification (0.85 F1-score) but struggling with URLLC and mMTC due to overlapping features, as evidenced by their confusion matrices showing significant misclassifications (e.g., 44 URLLC as mMTC for Logistic Regression, 81 URLLC as mMTC for SVC). XGBoost improved performance with an 87% accuracy, demonstrating strong URLLC classification (0.90 F1-score) and better handling of complex feature interactions, though mMTC misclassifications persisted (62 as eMBB). LightGBM slightly outperformed XGBoost at 88% accuracy, with notable URLLC precision (0.93) and scalability, reducing misclassifications (e.g., 44 URLLC as mMTC). Federated Learning models significantly enhanced performance, with Average Aggregation achieving 90% accuracy and balanced results across classes (0.92 F1-score for URLLC), while Max Aggregation scored 88% with weaker eMBB performance (0.89 F1-score). Min Aggregation excelled at 92% accuracy, with high precision and recall for eMBB (0.93) and URLLC (0.94), and a robust AUC of 0.9841, reflecting resilience to non-IID data. The Feature Fusion Model with Knowledge Distillation achieved the highest accuracy of 98%, with near-perfect eMBB (0.99 F1-score) and URLLC (0.98 F1-score) classification, minimal mMTC misclassifications (10 as URLLC), and exceptional generalization due to the integration of XGBoost-derived features and soft predictions. These results underscore the superiority of advanced techniques like Min Aggregation and Feature Fusion with Knowledge Distillation in addressing the

complexity of network slice classification, offering high accuracy, scalability, and robustness for real-world 5G/6G applications.

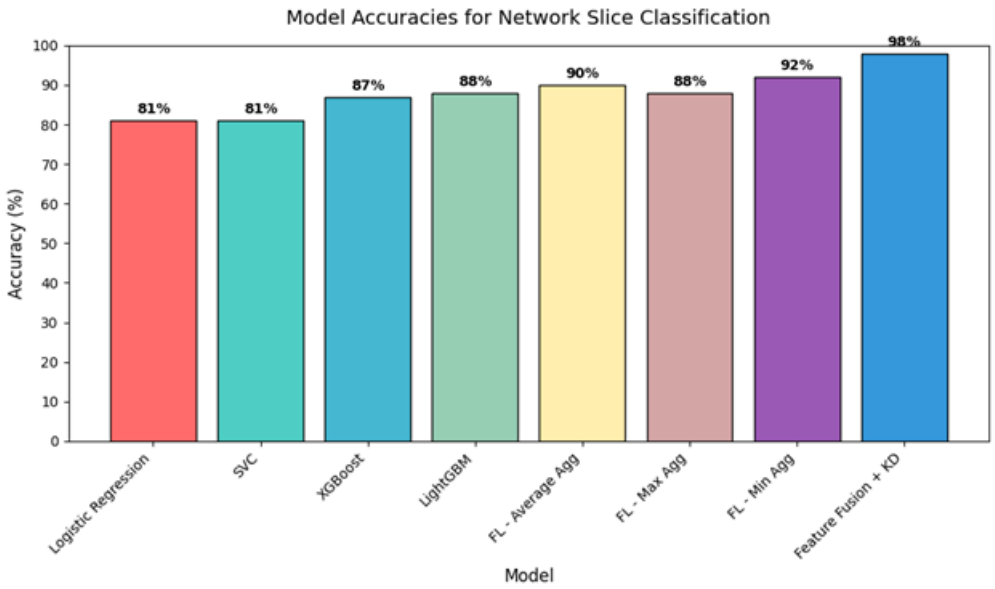

Figure 4.13: Comparative Analysis

Model Evaluation Using ROC Curves

To assess the classification performance of the models, Receiver Operating Characteristic (ROC) curves were plotted for all classifiers, as shown in Figure [X]. The ROC curves illustrate the trade-off between the true positive rate (sensitivity) and false positive rate for each model. The Area Under the Curve (AUC) serves as a quantitative metric to compare classifier effectiveness.

As depicted, the LightGBM classifier achieved the highest AUC for Class 0 at 0.98 and a micro-average AUC of 0.96, indicating excellent discrimination ability. Using micro-average AUC of 0.96 and a class-wise AUC of 0.97, XGBoost also proved to display good performance. Logistic Regression performed well with its class-wise AUC of 0.95 and its micro-average AUC, which was 0.93. Support Vector Classifier took the second pace with 0.94 and 0.92 respectively.

These results indicate that all models are good at classification, but the ensemble methods like LightGBM and XGBoost perform slightly better than the usual models. The robustness and the reliability in classifying the target class of the models on numerous measurements is confirmed using the ROC analysis.

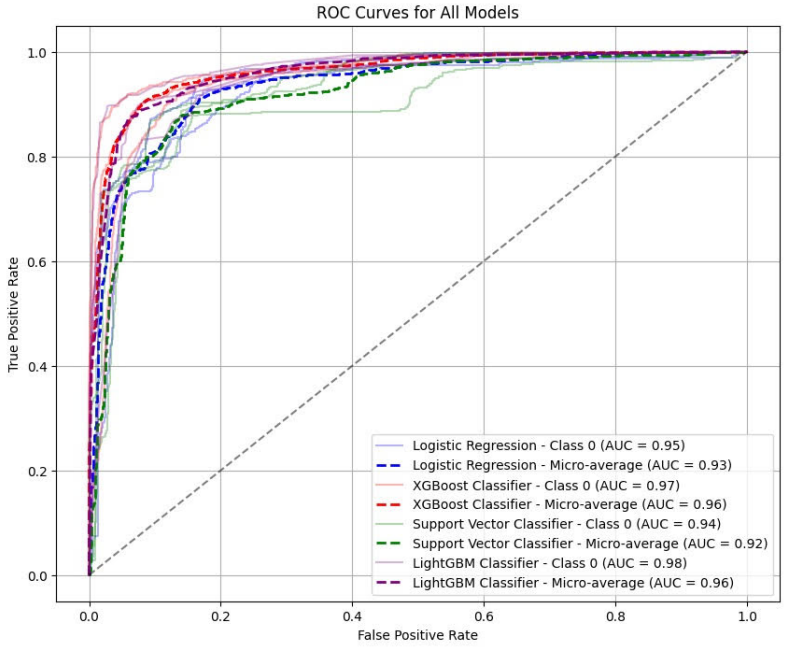

Figure 4.14: ROC Curve Comparison of Different Models

4.15 Model Complexity and Parameter Summary

In addition to such performance metrics as accuracy, precision, and F1-score, the scalability and complexity of the models by means of memory requirements and the number of parameters are essential to evaluate. This research is used to determine whether the models are applicable in real-life conditions that have limited availability on resources such as an edge device or a mobile network.

In this study, they put into practice a Federated Learning model with 2,627 trainable parameters and an extremely lightweight construct. There are no non-trainable parameters. Use of memory is such that it is broken down as follows:

Table 4.9: Federated Learning Model Parameter Summary

Metric	**Value**
Total Parameters	2,627
Trainable Parameters	2,627
Non-trainable Parameters	0
Input Size (MB)	0.00
Forward/Backward Pass Size (MB)	0.00
Parameters Size (MB)	0.01
Estimated Total Size (MB)	0.01

Table 4.10: Feature Fusion Model Parameter Summary

Metric	**Value**
Total Parameters	121,731
Trainable Parameters	120,835
Non-trainable Parameters	896
Model Size (KB)	475.51
Trainable Parameters Size (KB)	472.01
Non-trainable Parameters Size (KB)	3.50

This minimal parameter size makes the federated model exceptionally suitable for decentralized edge-based learning, especially in low-resource or battery-sensitive devices, while still preserving competitive classification accuracy through collaborative updates.

On the other hand, the Feature Fusion model utilizing knowledge distillation is more complex and resource-intensive, with a total of 121,731 parameters. Out of these, 120,835 are trainable and 896 are non-trainable, yielding a model size of approximately 475.51 KB. This model, although more computationally demanding, benefits from enhanced learning capacity due to feature fusion and soft target supervision via knowledge distillation, achieving the highest classification performance among all tested models.

The contrasting model sizes highlight a trade-off between efficiency and accuracy. While the federated model is ideal for lightweight deployment, the fusion model is better suited for centralized scenarios where higher accuracy is required and computational resources are available.

Chapter 5

Conclusion

This study thoroughly assessed various machine learning methodologies for network slice categorization in advanced 5G and 6G communication networks, concentrating on three principal slice categories: eMBB, URLLC, and mMTC. The analysed techniques consisted of standard machine learning models (Logistic Regression, XGBoost, Support Vector Classifier, and LightGBM) and a federated learning system with different aggregation strategies (Average, Max, and Min) and a hybrid-based approach that uses both feature fusion and knowledge extraction. With the aim of carrying out an in-depth performance analysis under various network circumstances, every technique was critically evaluated in terms of accuracy, precision, recall, F1-score, and confusion matrices. Out of conventional machine learning models, Logistic Regression and Support Vector Classifier models produced rather good 81 percent accuracies, and it is reasonable that such models demonstrated considerable performance, especially of the eMBB class. But as shown in their confusion matrices, both of them experienced difficulty differentiating URLLC and mMTC due to the similarities between the two. XGBoost had sensitivity and specificity of 87% accuracy, beating these models, showing that it was capable of learning complex, non-linear feature interactions, in particular, URLLC (0.90 F1-score). LightGBM surpassed it, got 88% accuracy, had good scalability and efficiency, particularly in URLLC classification (0.91 F1-score). The federated learning framework presents a privacy-preserving, decentralized methodology for model training, tackling practical challenges including data privacy and non-IID distributions. The Average Aggregation approach attained a commendable 90% accuracy, demonstrating consistent performance across all categories. Max Aggregation achieved an accuracy of 88%, exhibiting marginally diminished performance for eMBB, however Min Aggregation excelled with an impressive accuracy of 92%, showcasing enhanced precision and recall for eMBB (0.93) and URLLC (0.94). The Min Aggregation strategy's capacity to emphasize conservative updates enhanced its resilience to outliers, rendering it exceptionally successful for heterogeneous data distributions. The Feature Fusion Model with Knowledge Distillation was the highest achiever, attaining an outstanding 98% accuracy. This method markedly improved classification performance by incorporating original features with XGBoost-derived leaf node indices and utilizing soft predictions from the teacher model (XGBoost) to inform the student neural network. The model exhibited nearly flawless outcomes for eMBB (0.99 F1-score) and URLLC (0.98 F1-score), with negligible misclassifications for mMTC (0.98 F1-score), as validated by its confusion matrix. The use of a bespoke loss function integrat-

ing categorical cross-entropy and Kullback-Leibler divergence, along with strategies like as dropout and early halting, guaranteed elevated accuracy and generalization. The findings underscore the inadequacies of conventional approaches in managing intricate, overlapping feature spaces, especially in mMTC classification. Conversely, sophisticated methods such as federated learning via Min Aggregation and feature fusion through knowledge distillation exhibited enhanced performance, scalability, and adaptability to the problems of real-world 5G/6G network slicing. The incorporation of explainable AI methodologies, including LIME and SHAP, enhanced the transparency of the decision-making process, hence cultivating trust in the models' forecasts. In summary, the Feature Fusion Model with Knowledge Distillation presents a very effective and efficient method for network slice categorization, rendering it a promising strategy for real-time resource allocation in 5G and forthcoming 6G networks. The federated learning framework, especially with Min Aggregation, offers a feasible option for privacy-sensitive, decentralized settings. These findings highlight the promise of hybrid and collaborative learning methodologies to meet the evolving and varied demands of next-generation wireless networks, facilitating the development of more resilient, scalable, and privacy-conscious network management solutions. Future research may investigate additional hyperparameter tuning, alternate fusion methodologies, and adaption to advancing network technology to further improve performance.

5.1 Future Work

Future work will focus on enhancing the proposed hybrid framework by integrating explainable AI (XAI) techniques, such as SHAP or LIME, to improve model transparency without incurring high computational overhead. Deploying the system in real-world or emulated 5G/6G environments will be crucial to validate its robustness under dynamic network conditions, diverse traffic patterns, and device heterogeneity. Future research may also explore adaptive and personalized federated learning strategies to better handle non-IID data distributions and client variability. Expanding the model to support emerging slice types and sub-slicing strategies will increase its relevance as network demands evolve. Additionally, optimizing the framework for energy efficiency and computational scalability will make it more suitable for deployment on edge devices. Security and robustness in federated learning remain critical challenges; therefore, implementing secure aggregation, differential privacy, or blockchain-based mechanisms can protect against adversarial attacks and data breaches. Finally, incorporating synthetic data generation techniques, such as GANs, can augment training datasets and enhance generalizability, enabling the model to perform reliably in diverse and unseen scenarios.

Bibliography

[1] R. Sherwood, G. Gibb, K. K. Yap, G. Appenzeller, N. McKeown, and G. Parulkar, "Can the production network be the testbed?" In *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*, 2010.

[2] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[3] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5g: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[4] B. McMahan, E. Moore, D. Ramage, and S. Hampson, "Communication-efficient learning of deep networks from decentralized data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[5] G. S. Aujla, N. Kumar, M. Singh, and S. Lim, "Data-aware resource management in fog computing: A reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3249–3259, 2018. DOI: 10.1109/jiot.2018.2842435.

[6] E. Jeong, S. Oh, H. Kim, and J. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[7] R. Li, Z. Zhao, Q. Sun, *et al.*, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018. DOI: 10.1109/access.2018.2881964. [Online]. Available: https://doi.org/10.1109/access.2018.2881964.

[8] N. C. Luong, D. T. Hoang, S. Gong, *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *arXiv preprint arXiv:1810.07862*, 2018. [Online]. Available: https://doi.org/10.48550/arxiv.1810.07862.

[9] A. Napolitano, A. Giorgetti, K. Kondepu, L. Valcarenghi, and P. Castoldi, "Network slicing: An overview," in *2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI)*, 2018, pp. 1–4. DOI: 10.1109/rtsi.2018.8548449. [Online]. Available: https://doi.org/10.1109/rtsi.2018.8548449.

[10] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5g network slicing using sdn and nfv: A survey of taxonomy, architectures and future challenges," *Computer Networks*, vol. 167, p. 106 984, 2019. DOI: 10.1016/j.comnet.2019.106984. [Online]. Available: https://doi.org/10.1016/j.comnet.2019.106984.

[11] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[12] J. Zhang, Z. Zhang, and Y. Liu, "Feature fusion for online mutual knowledge distillation," *arXiv preprint arXiv:1904.09058*, 2019.

[13] Y. Abiko, T. Saito, D. Ikeda, K. Ohta, T. Mizuno, and H. Mineno, "Flexible resource block allocation to multiple slices for radio access network slicing using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 68 183–68 198, 2020. DOI: 10.1109/access.2020.2986050. [Online]. Available: https://doi.org/10.1109/access.2020.2986050.

[14] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," *International Conference on Artificial Intelligence and Statistics*, pp. 2938–2948, 2020.

[15] S. K. Singh, M. M. Salim, J. Cha, Y. Pan, and J. H. Park, "Machine learning-based network sub-slicing framework in a sustainable 5g environment," *Sustainability*, vol. 12, no. 20, 2020.

[16] X. Vasilakos, N. Nikaein, D. H. Lorenz, B. Koksal, and N. Ferdosian, "Integrated methodology to cognitive network slice management in virtualized 5g networks," *arXiv preprint arXiv:2006.12345*, 2020.

[17] S. Khan, S. Khan, Y. Ali, M. Khalid, Z. Ullah, and S. Mumtaz, "Highly accurate and reliable wireless network slicing in 5th generation networks: A hybrid deep learning approach," *arXiv preprint arXiv:2105.06789*, 2021.

[18] J. A. Nasir, O. S. Khan, and I. Varlamis, "Fake news detection: A hybrid cnn-rnn-based deep learning approach," *International Journal of Information Management Data Insights*, vol. 1, no. 1, p. 100 007, 2021. DOI: 10.1016/j.jjimei.2020.100007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667096820300070.

[19] D. C. Nguyen, M. Ding, P. N. Pathirana, and A. Seneviratne, "Federated learning meets blockchain: A secure, decentralized and privacy-preserving framework," *IEEE Transactions on Services Computing*, vol. 14, no. 3, pp. 747–760, 2021.

[20] D. C. Li, M. R. Maulana, and L. Chou, "Nnsplit-søren: Supporting the model implementation of large neural networks in a programmable data plane," *Computer Networks*, vol. 222, p. 109 537, 2022. DOI: 10.1016/j.comnet.2022.109537. [Online]. Available: https://doi.org/10.1016/j.comnet.2022.109537.

[21] X. Li, H. Deng, Y. Wang, Y. Chen, and B. Liu, "Security challenges and opportunities for 5g network slicing," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 106–113, 2022.

[22] S. Roy, H. Chergui, and C. Verikoukis, "Tefl: Turbo explainable federated learning for 6g trustworthy zero-touch network slicing," *arXiv:2210.10147*, 2022.

[23] T. Saha, N. Aaraj, and N. K. Jha, "Machine learning assisted security analysis of 5g-network-connected systems," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 4, pp. 2006–2024, 2022. DOI: 10.1109/tetc.2022.3147192. [Online]. Available: https://doi.org/10.1109/tetc.2022.3147192.

[24] N. S. Soud, N. A. S. Al-Jamali, and H. S. Al-Raweshidy, "Moderately multispike return neural network for sdn accurate traffic awareness in effective 5g network slicing," *IEEE Access*, vol. 10, pp. 73 378–73 387, 2022. DOI: 10.1109/ACCESS.2022.3189354. [Online]. Available: https://ieeexplore.ieee.org/document/9819896.

[25] S. Xing, J. Xing, J. Ju, Q. Hou, and X. Ding, "Collaborative consistent knowledge distillation framework for remote sensing image scene classification network," *Remote Sensing*, vol. 14, no. 3, 2022.

[26] T. Dangi, T. Zhang, and V. Deshpande, "Application-aware radio access network slicing for autonomous vehicles: A deep reinforcement learning approach," *Computer Networks*, vol. 237, p. 109 799, 2023. DOI: 10.1016/j.comnet.2023.109799.

[27] Y. Liu, Y. Zhang, and J. Chen, "Netboost: Towards efficient distillation and service differentiation of alto-based network information," *Computer Networks*, vol. 207, p. 108 760, 2023.

[28] M. Malkoc and H. A. Kholidy, "5g network slicing: Analysis of multiple machine learning classifiers," *arXiv preprint arXiv:2303.12345*, 2023.

[29] J. S. B. Martins *et al.*, "Enhancing network slicing architectures with machine learning, security, sustainability and experimental networks integration," *arXiv preprint arXiv:2307.12345*, 2023.

[30] J. Xie, L. Gong, S. Shao, S. Lin, and L. Luo, "Hybrid knowledge distillation from intermediate layers for efficient single image super-resolution," *SSRN Electronic Journal*, 2023.

[31] A. Bhat, *How can we simulate federated learning in 5g/6g?* 2024.

[32] M. Zong, Y. Chang, Y. Dang, and K. Wang, "Pedestrian trajectory prediction in crowded environments using social attention graph neural networks," *Applied Sciences*, vol. 14, no. 20, p. 9349, 2024. DOI: 10.3390/app14209349. [Online]. Available: https://doi.org/10.3390/app14209349.

[33] M. Chiarani, S. Roy, C. Verikoukis, and F. Granelli, "Xai-driven client selection for federated learning in scalable 6g network slicing," *arXiv:2503.12435*, 2025.

[34] Q. Hou, W. Yang, and G. Liu, "Chinese herbal medicine recognition network based on knowledge distillation and cross-attention," *Scientific Reports*, 2025.