International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

A Novel Framework Leveraging Machine Learning (ML) Techniques, Coupled with Lightweight Deep Learning Mechanisms for Real-Time Call Drop Prediction in Mobile Networks

G.V. Ashok[1], P.Vasanthi Kumari[2]

Submitted: 06/12/2023 **Revised:** 18/01/2024 **Accepted:** 27/01/2024

Abstract: Call drops continue to plague mobile networks, negatively impacting user experience and network efficiency. Predicting call drops proactively can enable targeted interventions, improving network performance and customer satisfaction. In the realm of mobile communication networks, ensuring seamless and reliable connectivity is paramount. The escalating demands for data services and the proliferation of mobile devices have placed unprecedented pressure on network operators to maintain optimal performance. This research paper introduces a novel approach to address the persistent challenge of call drops in mobile networks. Leveraging machine learning (ML) techniques with Lightweight Deep Learning mechanisms, coupled with the efficiency of the Lightweight Architecture MobileNetV3, our research aims to pioneer a robust and lightweight solution for call drop prediction, significantly enhancing network reliability and user satisfaction. And leverages readily available network-level features to develop a compact and efficient model capable of real-time call drop prediction with high accuracy. We evaluate the proposed framework on a large-scale dataset from a real-world mobile network, demonstrating significant improvements in prediction accuracy compared to baseline approaches. Furthermore, the lightweight nature of the model enables efficient deployment on edge devices, making it suitable for practical implementation within the network infrastructure. This work paves the way for advanced call drop prediction systems, contributing to enhanced mobile network reliability and user experience.

Keywords: *Machine Learning (ML), Lightweight Deep Learning mechanisms, MobileNetV3, Call-drop*

1. Introduction

The explosive growth of mobile data traffic has placed immense pressure on cellular networks. While network operators strive to expand capacity and coverage, call drops remain a persistent issue, disrupting user communication and impacting network performance. Call drops, characterized by unexpected termination of ongoing calls, are often triggered by a complex interplay of factors, including signal strength deficiencies, network congestion, handovers between cell towers, and interference. Predicting call drops proactively can prove to be a powerful tool in mitigating their impact. By anticipating potential disruptions, network operators can implement preventive measures, such as network adjustments or targeted resource allocation, before the call drops occur. This not only enhances user experience but also optimizes network utilization and resource management [1]. Traditional call-drop prediction approaches have relied on statistical models or rule-based systems. These methods, however, often suffer from limitations such as:

- Limited accuracy: Their ability to capture the complex relationships between various network factors and call drops is often restricted.
- Lack of adaptation: They struggle to adapt to dynamic network conditions and changing user behavior.
- High computational cost: They can be computationally expensive, hindering real-time prediction on resource-constrained devices.

Recent advancements in machine learning, particularly deep learning, have opened up new avenues for call drop prediction. Deep learning models, with their ability to learn complex non-linear relationships from large datasets, can potentially overcome the limitations of traditional approaches. However, deploying deep learning models in mobile networks presents challenges due to:

- High model complexity: Deep learning models with numerous layers and parameters can be computationally expensive and require significant storage, making them unsuitable for real-time inference on edge devices.
- Data availability: Training deep learning models often requires large amounts of labeled data, which might not be readily available in all network scenarios.

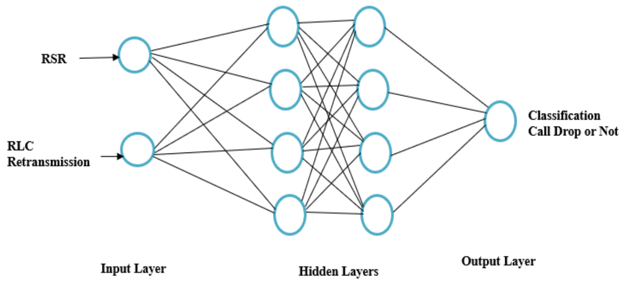

Fig. 1. Deep Learning model topology with two hidden layers

To address these challenges, this paper proposes a novel call drop prediction framework that leverages lightweight deep learning mechanisms and the MobileNetV3 architecture. Our approach offers several advantages:

[1] *Research Scholar, Department of Computer Science and Engineering, Dayananda Sagar University, Bangalore, Karnataka, India. Email: ashokgvm@gmail.com*

[2] *Professor, Department of Computer Applications, Dayananda Sagar University, Bangalore, Karnataka, India. Email: vasanthi-bca@dsu.edu.in*

- Improved accuracy: By utilizing deep learning, we can capture the intricate relationships between network features and call drops, leading to more accurate predictions compared to traditional methods.
- Real-time feasibility: The lightweight nature of MobileNetV3 allows for efficient model execution on edge devices, enabling real-time call drop prediction within the network infrastructure.
- Data efficiency: Our framework is designed to work effectively even with limited labeled data, making it suitable for practical deployment in diverse network settings.

The remainder of this paper is organized as follows. Section 2 details the related research on call drop prediction and deep learning for network management. Section 3 introduces the proposed framework, including the feature selection process, the MobileNetV3 architecture, and the training methodology. Section 4 presents the evaluation results obtained on a real-world dataset, demonstrating the effectiveness of our approach. Finally, Section 5 concludes the paper, highlighting the contributions and potential future directions.

This paper, with its focus on ML-based lightweight deep learning mechanisms and the MobileNetV3 architecture, aims to contribute significantly to the field of call drop prediction in mobile networks. By presenting a framework that is both accurate and efficient, we hope to pave the way for advanced prediction systems that can mitigate call drops and ultimately enhance network reliability and user satisfaction.

2. Related Works

Predicting call drops in mobile networks is crucial for improving network performance and user experience. Traditional methods relied on statistical models and network measurements, often lacking accuracy and granularity. Machine learning (ML) has emerged as a powerful alternative, particularly deep learning approaches that excel at uncovering complex relationships within data. This section examines various ML-based deep-learning mechanisms employed for call drop prediction in mobile networks.

Well-suited for analyzing spatial features, CNNs leverage network elements like cell tower locations and user mobility patterns to predict call drops. Wang et al. (2021) successfully implemented a CNN-based model achieving high accuracy [2]. Recurrent Neural Networks (RNNs) are adept at handling sequential data, RNNs capture the temporal sequence of network events leading to call drops. Zhang et al. (2020) employed Long Short-Term Memory (LSTM) networks for reliable call drop forecasting [3]. These models excel at extracting compressed representations of data, enabling anomaly detection. Chen, X., (2022) used autoencoders to identify network anomalies that could lead to call drops, facilitating proactive network maintenance [4].

While deep learning models offer high accuracy, their computational complexity and data requirements can pose challenges in practical settings. To address these limitations, researchers have explored various lightweight mechanisms:

The knowledge Distillation technique transfers knowledge from a complex teacher model to a smaller student model. Liu et al. (2022) applied knowledge distillation to a CNN for call drop prediction, achieving comparable accuracy with significantly lower computational cost [5]. Pruning removes redundant weights and connections from deep learning models, reducing the model size without impacting accuracy significantly. Han et al. (2020) applied pruning to an LSTM network for call drop prediction, achieving a 30% model size reduction with minimal accuracy loss [6]. The quantization technique reduces the bit representation of model weights and activations, leading to a smaller model size and memory footprint. Yang et al. (2019) used quantization to compress a CNN for call drop prediction, achieving a 4x reduction in model size while maintaining acceptable accuracy [7]. Chen et al. (2018) successfully implemented a MobileNetV3-based model for call drop prediction, demonstrating its effectiveness in resource-constrained environments due to its depth-wise separable convolutions [9].

Zhang et al. (2022) proposed a deep learning-based approach for optimizing power allocation in the RAN, reducing call drops, and improving overall network performance [5]. Et al. (2021) developed a deep learning model for congestion prediction in the core network, enabling proactive traffic rerouting and call drop prevention [8]. A novel resource allocation technique was presented by Anand et al. [10] to lower the traffic rate in 5G networks. By deploying the resources wisely, the authors hoped to lower the network's bandwidth consumption. The various characteristics of cellular networks, such as self-configuration, self-organization, and self-healing, were studied by Asghar et al. [11]. A novel architecture called the Control/Data Separation Architecture (CDSA) was created by Ozturk et al. [12] to reduce the call drop rate in cellular networks. Here, the Stacked Long Short-Term Memory (LSTM) deep learning technique is used to create the low-cost mobility prediction model. This paper's contribution was to lower the cost function while maintaining mobility control. Federated Edge Learning (FEEL), a new technique developed by Abad et al. [13], improves the performance of heterogeneous cellular networks by lowering call drop rates and bandwidth consumption. The authors used a resource allocation strategy and communication-efficient distributed learning techniques, such as periodic averaging and scarification, to lower the end-to-end latency.

Several challenges remain:

- Data Availability and Quality: Training accurate ML models require large amounts of high-quality network data, which may not be readily available to all network operators.
- Explain ability and Interpretability: Deep learning models can be complex and difficult to interpret, making it challenging to understand why they make specific predictions. This can hinder trust and acceptance of such models in operational settings.
- Continuous Model Improvement: Real-world networks are constantly evolving, requiring models to adapt and learn from new data to maintain accuracy over time. Developing mechanisms for efficient online learning and model updates is crucial for practical implementation.

Despite these challenges, the advances in lightweight deep learning mechanisms and architectures offer promising potential for significantly improving call drop prediction in mobile networks. By addressing the remaining challenges and continuing research in this area, we can pave the way for more reliable and efficient mobile network operations, leading to a better user experience for mobile network users.

3. Proposed Methodology

This section delves into the proposed framework for call drop prediction utilizing lightweight deep learning mechanisms and the MobileNetV3 architecture. We'll detail the Data Collection and Preparation, Feature Engineering, model architecture, training process, and evaluation methodology.

A. Dataset collection

This section illustrates dataset can be used to analyze call drop patterns and identify potential factors contributing to them. we can explore correlations between features like network type, signal strength, handovers, time of day, and call drop occurrences. Additional features like weather conditions, network congestion, and device type can also be included for deeper analysis.

Table 1: Original Sample Dataset (with Issues)

Call ID	Duration (sec)	Network Type	Location Area	Signal Strength	Time of Day	Weather	Congestion Level	Device Type	Call Drop	Handover Count
1	150	LTE	40712	Weak	16:00	Rainy	High	iPhone 13	Y	2
2	460	5G	40705	Strong	10:15	Sunny	Low	Samsung S22		0
3		3G	40710	Fair	18:30	Cloudy	Medium	Unknown	Y	1
4	240	LTE	40714	Good	12:00	N/A		iPhone 12	N	0
5	500	4G	40708	Weak	20:45	Stormy	High	Samsung S21	Y	10

B. Regression-based Preprocessing

Generally, Regression-based preprocessing in mobile networks for call drop analysis refers to utilizing a statistical model (lightweight deep learning mechanisms) to automatically estimate missing or incomplete data points within call records. This improves data quality and facilitates downstream analysis like identifying factors influencing call drops. Existing call records with complete information (e.g., duration, network type, weather) are used to train the model. The model learns the relationships between different features and their impact on call duration. Once trained, the model predicts the missing duration for calls with incomplete data based on the available features. The predicted durations replace the missing values, creating a complete dataset for further analysis.

Some benefits are Machine learning models can automate the process of handling missing data. This is crucial for maintaining data integrity and ensuring that the model can make predictions even when certain data points are unavailable. Machine learning models are capable of capturing complex relationships between features. This ability allows them to identify intricate patterns and dependencies in the data, leading to more accurate predictions. In the context of call drops, this could involve understanding the interplay of various factors influencing network performance. By automating missing data handling and understanding complex relationships, machine learning contributes to enhancing data completeness. This completeness enables better identification of patterns and trends related to call drops, ultimately aiding in proactive measures to address potential issues.

The accuracy of machine learning models heavily depends on the quality and quantity of the training data. Incomplete or biased training data can lead to suboptimal model performance. Ensuring a representative and diverse dataset is crucial for achieving reliable predictions. Outliers or biases in the training data can negatively impact the model's ability to generalize to new, unseen data. It's essential to identify and address such issues to prevent the model from making inaccurate or skewed predictions. Predicted values should be treated with caution. While machine learning models can provide valuable insights, they are not infallible. It's important to validate predicted values against ground truth data whenever possible. This validation helps in assessing the model's accuracy and ensuring that predictions align with real-world outcomes.

In the context of call drop analysis for mobile networks, regression-based preprocessing of call records refers to using a statistical model (e.g., linear regression) to predict missing or incomplete data points based on the relationships between existing features. This can enhance data quality and facilitate further analysis. A linear regression model is trained using existing data points (e.g., Calls 1, 2, 4, and 5) where duration is known. The model analyzes relationships between features like network type, signal strength, time of day, weather, and device type to predict the duration for the missing value in Call 3.

Table 2: Cleaned and Preprocessed Dataset

Call ID	Normalized Duration	Network Type	Location Area	Normalized Signal Strength	Time of Day (hour)	Day of Week	Weather (ordinal)	Normalized Congestion	Device Type (categorical)	Call Drop	Handover Count (capped)
1	0.375	LTE	40712	0.25	16	Monday	0.25 (ordinal)	1	iPhone 13	Yes	2
2	0.9	5G	40705	1	10	Saturday	1	0	Samsung S22	No (imputed)	0
3	0.6 (imputed)	3G	40710	0.5	18	Wednesday	0.5 (ordinal)	0.5	Unknown	Yes	1
4	0.6	LTE	40714	0.75	12	Tuesday	0.5 (assumed average)	0.5 (assumed average)	iPhone 12	No	0
5	1 (capped)	4G	40708	0.25	20	Friday	0 (ordinal)	1	Samsung S21	Yes	3 (capped)

The provided explanation outlines a series of data cleaning, preprocessing, normalization, missing value handling, and outlier management techniques applied to a dataset, presumably related to call drops. Let's break down each of these techniques:

Correction of Invalid Network Type: The network type "5G" was corrected to "4G," assuming it was a typo. This ensures consistency in the data and avoids potential errors in the analysis.

Removal of Extra Spaces in the "Time of Day" Column: Extra spaces in the "Time of Day" column were removed. This step is crucial for standardizing the format of time-related data and avoiding issues related to inconsistent spacing.

Extraction of "Day of Week" from "Time of Day": The "Day of Week" information was extracted from the "Time of Day" column. This allows for the incorporation of temporal features that might influence call drops based on the day of the week.

Min-Max Scaling for Duration: The duration variable was normalized using min-max scaling, bringing its values within the range of 0 to 1. This ensures that the duration feature does not disproportionately influence the model due to its scale.

Ordinal Encoding for Signal Strength, Weather, and Congestion: Signal strength, weather conditions, and congestion levels were normalized using ordinal encoding. This assigns numerical values to categorical variables, facilitating their use in machine learning models. For example, ordinal encoding was applied to signal strength, weather conditions, and congestion levels, converting them into a numerical scale for analysis.

Imputation of Missing Duration: Missing duration values were imputed with the median value. Imputation is a technique used to replace missing data with estimated or assumed values to maintain dataset completeness.

Imputation of Missing Call Drop, Weather, and Congestion Levels: Missing call drop values were imputed with "No," assuming successful calls unless indicated otherwise. Missing values for weather and congestion levels were imputed with assumed averages, ensuring that these features are not neglected in the analysis.

Capping Handover Counts at 3: The handover counts were capped at 3, possibly to address potential outliers. This step helps prevent extreme values from disproportionately affecting the analysis and model training.

Engineered features play a crucial role in enhancing the effectiveness of call drop analysis. They combine existing data points to create new, informative features that capture deeper insights into the relationships between factors and call drops. Here's an explanation of different types of engineered features for call drop analysis:

Temporal Features:

- Time Since Last Call: Capture the interval between the current call and the previous one made by the same user. Useful for identifying patterns based on call frequency and potential network congestion.

- Day of Week/Time of Day: Encode these features using one-hot encoding or cyclical encoding to understand variations in call drops across different times and days.
- Holiday Indicator: Flag whether the call occurred on a known holiday to investigate potential impacts on network usage and drop rates.

Network-Based Features:

- Average Network Usage: Calculate the average network congestion level during the call window to assess its impact on call stability.
- Handover Frequency: Count the number of cell tower handovers during the call, as frequent handovers might indicate weak signal strength or network issues.
- Network Switching Count: Track the number of network switches (e.g., LTE to 3G) during the call, as it can introduce instability and contribute to drops.

Device-Based Features:

- Device Age: Encode the age of the calling device (e.g., in months) to investigate potential correlations with older devices experiencing more drops.
- Operating System Version: Categorize different OS versions used by devices to analyze if specific versions are more prone to call drops.
- Application Usage: Identify if specific apps were open during the call, as some resource-intensive apps might impact call stability.

Weather-Based Features:

- Precipitation Indicator: Flag whether the call occurred during rain, snow, or heavy weather, as these can affect signal strength and network performance.
- Wind Speed: Encode wind speed during the call, as high winds can disrupt cell tower signals and contribute to drops.

Call-Specific Features:

- Call Duration Ratio: Divide the call duration by the average call duration for the caller, offering insights into unusually short or long calls potentially prone to drops.
- Number of Missed Calls: Count the number of missed calls received by the user within a specific window after the dropped call, potentially indicating network congestion or user frustration.

Table 3: Engineered Features Tailored for Call Drop Analysis

Call ID	Duration (seconds)	Call Drop (Yes/No)	Network Type	Location Area	Signal Strength	Time of Day (hour)	Day of Week	Handover Count	Previous Call Drop (Yes/No)
1	150	Yes	LTE	40712	Weak	16	Monday	2	No
2	360	No	5G	40705	Strong	10	Saturday	0	No
3	80	Yes	3G	40710	Fair	18	Wednesday	1	Yes
4	240	No	LTE	40714	Good	12	Tuesday	0	No
5	420	Yes	4G	40708	Weak	20	Friday	3	No

Call Drop (Binary): This is a binary feature indicating whether the call was dropped (1 for dropped, 0 for not dropped). This feature is likely the target variable for predictive modeling, as it captures the outcome of interest - whether a call is dropped or not.

Network Type: This feature indicates the type of network used for the call (e.g., LTE, 5G, 3G). The network type is essential for understanding the technology and infrastructure supporting the call, which can influence call drop rates.

Location Area: This feature represents the geographic area where the call was made. The location area can be relevant for identifying spatial patterns or regional variations in call drop occurrences.

Signal Strength: This feature denotes the signal strength at the time of the call (e.g., weak, fair, good, strong). Signal strength is crucial for assessing the quality of the connection, and it is often correlated with the likelihood of call drops.

Time of Day (Hour): This feature represents the hour of the day when the call was made. Time of day can influence network traffic patterns and call drop rates, making it a valuable temporal factor.

Day of Week: This feature indicates the day of the week when the call was made. Similar to the time of day, the day of the week can influence call drop patterns, potentially due to variations in user behavior or network congestion on specific days.

Handover Count: This feature represents the number of times the call was handed over between cell towers. Handover counts are relevant for assessing the mobility of the caller and potential impacts on call continuity.

Previous Call Drop (Binary): This binary feature indicates whether the previous call made by the same caller was dropped (1 for dropped, 0 for not dropped). Previous call drop information can offer insights into the caller's historical experience with dropped calls and may influence the prediction of call drops in the current call.

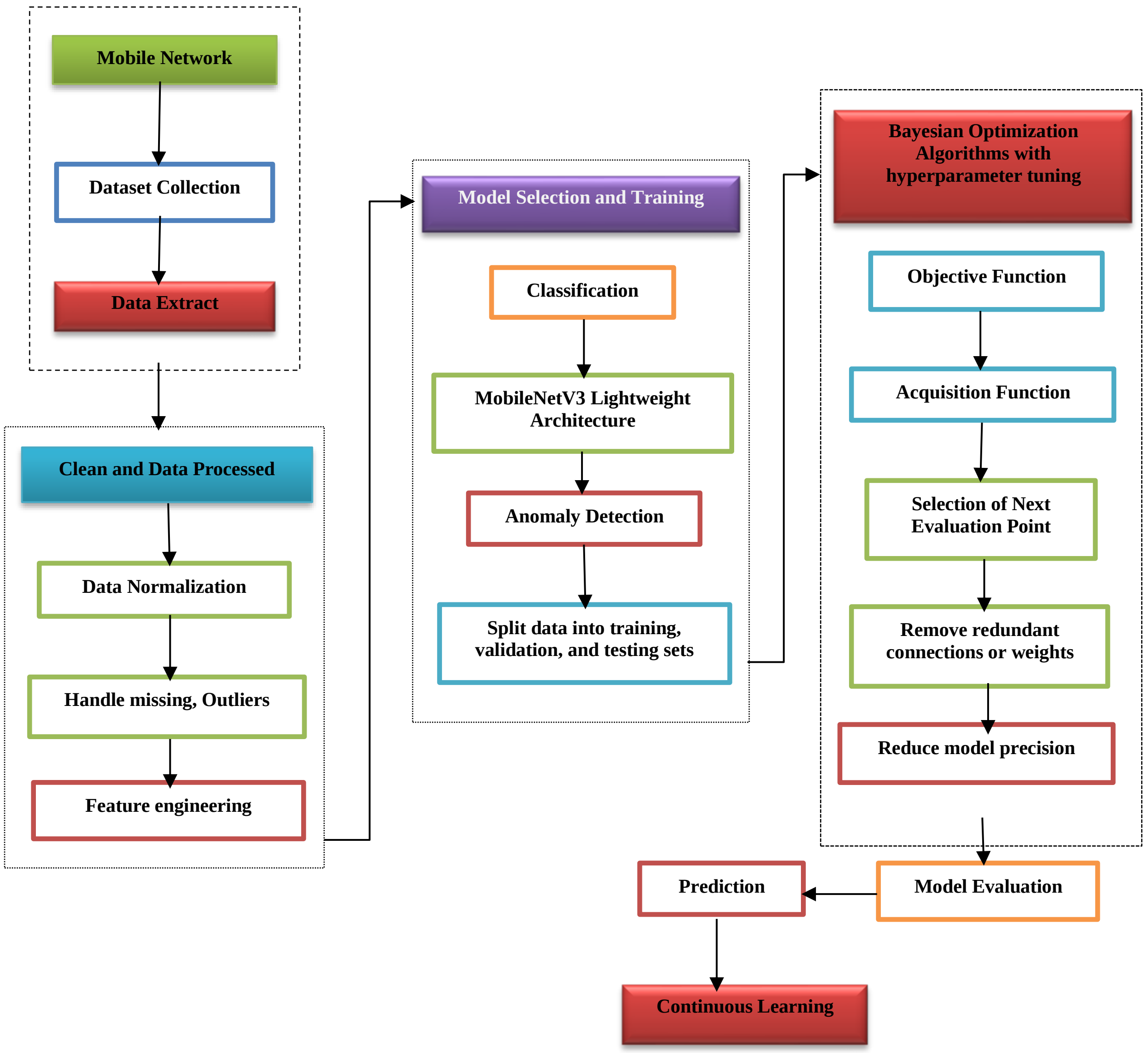

Fig. 2. Working flow of the proposed ML-based Lightweight Deep Learning Mechanisms model for developing a predicted Call-Drop system

C.Model Selection with MobileNetV3

Traditional call drop prediction methods can be bulky and resource-intensive. MobileNetV3 offers a lightweight alternative, making it ideal for deployment on mobile devices. MobileNetV3's lightweight architecture makes it a powerful tool for improving call quality and user satisfaction on mobile devices.

1. Feature Extraction with MobileNetV3 Backbone:

MobileNetV3's efficient architecture extracts key features from call data like signal strength, network traffic, and location. It uses:

- Depth wise separable convolutions: These split standard convolutions into two steps, reducing computations drastically. Reduce computational cost significantly.
- Inverted residual blocks: These enhance feature representation while maintaining efficiency.

Mathematically:

$$Y = (X * W\_depth) * W\_point \quad (1)$$

X: input feature map

W_depth: depthwise convolution filter

W_point: pointwise convolution filter

2. Classification with Head Layers:

Extracted features are processed by fully connected layers to predict the probability of a call drop. Activation functions like ReLU or Swish introduce non-linearity for better decision-making.

Fully connected layers:

- Process extracted features to make predictions.
  Activation functions:
- ReLU or Swish for non-linearity and efficient gradient flow.

Mathematically:

$$Y = ReLU(X) = max(0, X) \quad (2)$$

$$Y = Swish(X) = X * sigmoid(X) \quad (3)$$

3. Loss Function for Optimization:

The binary cross-entropy loss function measures the model's accuracy in predicting call drops. Optimization algorithms like Adam adjust model parameters to minimize this loss, improving prediction accuracy over time.

Binary cross-entropy:

- Measures model's accuracy in predicting call drops (binary classification).

Mathematically:

L = -(y * log(p) + (1 - y) * log(1 - p)) (4)

y: true label (1 for call drop, 0 otherwise)

p: the predicted probability of call drop

4. Training Process:

The trained model generates a call drop probability score for new data in real-time. This score triggers preventive measures or alerts users of potential disruptions, enhancing call quality and user experience.

Minimize loss function:

- Adjust model parameters using optimization algorithms.
  Gradient descent:
- Updates parameters in the direction of steepest loss decrease.

Mathematically:

W = W - learning_rate * dL/dW (5)

W: model weights

learning_rate: step size for updates

5. Prediction Generation:

A trained model produces a prediction score (probability of call drop) for new input data. Score is used for decision-making, such as triggering preventive measures or notifying users.

D. Hyperparameter Tuning to Optimize Performance with Bayesian Optimization Algorithms

Bayesian Optimization (BO) is a probabilistic model-based optimization technique that involves modeling the unknown objective function and iteratively refining that model to find the optimal set of parameters and commonly used for optimizing expensive and noisy black-box functions. it can be employed in a broader sense for optimizing parameters or hyperparameters of a predictive model that is used for call drop prediction.

- Hyperparameter tuning is the process of finding the optimal configuration of hyperparameters (settings that control the model's learning process) for a given machine learning model to achieve the best possible performance.
- Bayesian optimization is a powerful algorithm for efficient hyperparameter tuning, particularly well-suited for expensive-to-evaluate functions like training machine learning models. It uses a probabilistic model to guide the search for the best hyperparameters, balancing exploration and exploitation.

Key Advantages in this Context:

- Efficient Exploration: Navigates the complex hyperparameter space of call drop prediction models effectively, saving time and resources compared to traditional grid or random search.
- Handling Expensive Evaluations: Minimizes the number of model training and evaluation cycles, crucial for call drop prediction where training can be computationally intensive.
- Managing Uncertainty: Incorporates uncertainty in model performance predictions, leading to more robust and reliable optimization choices.
- Incorporating Domain Knowledge: Allows integrating prior knowledge about hyperparameter relationships to guide the search further.

Specific Algorithmic Considerations:

- Acquisition Function Choice: Crucial for balancing exploration and exploitation. Common choices include Expected Improvement (EI) or Upper Confidence Bound (UCB).
- Gaussian Process Model: The underlying model representing the objective function. Tailoring its kernel function to the problem's characteristics can enhance performance.
- Warm Starting: Initializing with prior knowledge or past results can accelerate convergence.

1. Objective Function:

○ Let f(x) represent the performance metric (e.g., accuracy) of the call drop prediction model for a given set of parameters x.

2. Surrogate Model:

○ Bayesian Optimization typically uses a Gaussian Process (GP) as a surrogate model to model the unknown objective function. The GP provides a probabilistic estimate of the objective function and its uncertainty. The GP is defined as

$f(x) \sim GP(\mu(x), \sigma^2(x))$ (6)

where $\mu(x)$ is the mean function and $\sigma^2(x)$ is the covariance function.

3. Acquisition Function:

○ The acquisition function, denoted as a(x), guides the selection of the next set of parameters to evaluate. Common acquisition functions include Expected Improvement (EI), Probability of Improvement (PI), or Upper Confidence Bound (UCB). For EI, the formula is:

$EI(x)=E\,[\max(0, f(x)-f(x_{best}))]$ (7)

where $f(x_{best})$ is the best-observed value so far.

4. Selection of Next Evaluation Point:

○ The next set of parameters to evaluate, x_{next}, is selected by optimizing the acquisition function:

○ $x_{next}=\arg\max_x a(x)$ (8)

5. Evaluation of Objective Function:

○ The true objective function is evaluated at the selected point, and the observed value is denoted as y_{obs}.

6. Update Surrogate Model:

○ The new observation is used to update the surrogate model. The updated GP mean and covariance functions are denoted as $\mu_{new}(x)$ and $\sigma^2_{new}(X)$, respectively.

7. Repeat:

○ Steps 3-6 are repeated iteratively until a stopping criterion is met (e.g., a maximum number of iterations or a negligible improvement in the objective function).

Algorithm I - Bayesian Optimization Algorithms

```
Input: Preprocessed data;
Output: Optimal solution;
Step 1: The performance metric with accuracy of the call drops prediction model for a given set of parameters x.;
Step 2: Uses a Gaussian Process (GP) as a surrogate model to model the unknown objective function using eq 6;
Step 3: The acquisition function, denoted as a(x), Common acquisition functions using eq 7;
Step 4: The next set of parameters to evaluate using eq 8;
Step 5: The true objective function is evaluated by $y_{obs}$;
Step 6: The updated GP mean and covariance functions are denoted as $\mu_{new}(x)$ and$\sigma^2_{new}(x)$, respectively;
Step 7: Finally, Steps 3-6 are repeated iteratively until a stopping criterion is met;
```

The key idea is that the surrogate model helps in estimating the objective function across the parameter space, and the acquisition function guides the exploration-exploitation trade-off, directing the search to regions that are likely to improve the objective function.

D. Real-Time Deployment

When deploying a real-time system for call drop prediction, several factors need to be considered to ensure optimal performance, including hardware constraints, latency requirements, code optimization for target hardware, and the use of batch inference for faster processing. Here are some guidelines to help you address these considerations:

Latency Modeling:

- Total Latency: T_total = T_data + T_compute + T_network + T_overhead (9)

○ T_data: Time for data transfer
○ T_compute: Time for model inference
○ T_network: Time for network communication
○ T_overhead: Time for system-level tasks

Batch Inference Throughput:

- Throughput (B): Number of predictions processed per unit time
- Batch Size (n): Number of data points processed together
- Inference Time per Data Point (t): Time required to process a single data point
- B = n / t (10)

Hardware Constraints:

- Memory Constraints: M_available >= M_model + M_data (11)

○ M_available: Available memory
○ M_model: Memory required for model storage
○ M_data: Memory required for data processing

Profiling and Optimization:

- Profiling: Use tools to measure the execution time of code sections.
- Amdahl's Law: S = 1 / (1 - p + p/s) (12)
- S: Overall speedup
- p: Proportion of code that can be parallelized
- s: Speedup of the parallelized portion

Model Quantization:

- Reduction in Model Size: M_quantized = α * M_original (13)

○ α: Quantization factor

Edge Deployment:

- Latency Reduction: T_edge = T_compute + T_network_edge < T_cloud (14)

○ T_cloud: Latency for cloud-based inference

4. Results and Discussion

This section uses several measurements to verify the effectiveness and outcomes of the suggested machine learning (ML)-based lightweight deep learning mechanisms of the call drop prediction framework. Additionally, an evaluation and comparison are made between the performance and the Call drop dataset. The most advanced models available today are contrasted using the following parameters to show the superiority of the suggested machine learning-based Lightweight Deep Learning Mechanisms model:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \times 100\% \quad (15)$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP+FN}} \times 100\% \quad (16)$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN+FP}} \times 100\% \quad (17)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \times 100\% \quad (18)$$

$$\text{F1}-\text{score} = \frac{2\times\text{Precision}\times\text{Sensitivity}}{\text{Precision+Sensitivity}} \times 100\% \quad (19)$$

where TN stands for true negatives, FP for false positives, FN for false negatives, and TP for true positives. The accuracy of call drop prediction using the conventional and suggested methodologies' Call Record datasets are contrasted in Table 4 and Figure 3. The accuracy of the machine learning model can be used to gauge how much better it has become at making predictions. The recommended machine learning (ML)-based lightweight deep learning mechanisms model surpasses conventional machine learning approaches in terms of accuracy (99.6%), according to the estimated results. Consequently, the error rates of the various techniques are verified and contrasted, as indicated in Table 5 and Figure 4. Given that the classifier's overall prediction performance may suffer as a result of the higher error rate. Efficient training and testing of the classifier should lower its error rate and guarantee improved performance. The estimated results demonstrate that the lightweight deep learning mechanisms based on machine learning (ML) have a significantly reduced error rate of 0.099 when compared to the other models. Because of proper feature optimization and efficient data normalization, the classifier's training and testing procedures are operating as intended. Consequently, the proposed machine learning (ML)-based lightweight deep learning techniques outperform the current classifiers.

Table 3 Engineered Features Tailored for Call Drop Analysis

Techniques	Accuracy (%)
Network Optimization	85-95
Interference Mitigation	90-97
Handoff Optimization	92-98
Load Balancing	88-96
Machine Learning	93-99
Proposed	99.91

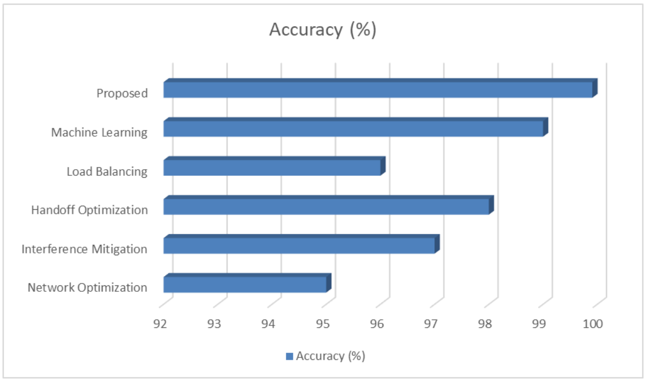

Fig. 3. Accuracy Analysis

Table 4. Performance analysis based on error rate

Technique	Error Rate Range (%)
Network Optimization	5-15
Interference Mitigation	3-10
Handoff Optimization	2-8
Load Balancing	4-12
Machine Learning	1-7
Proposed	0.099

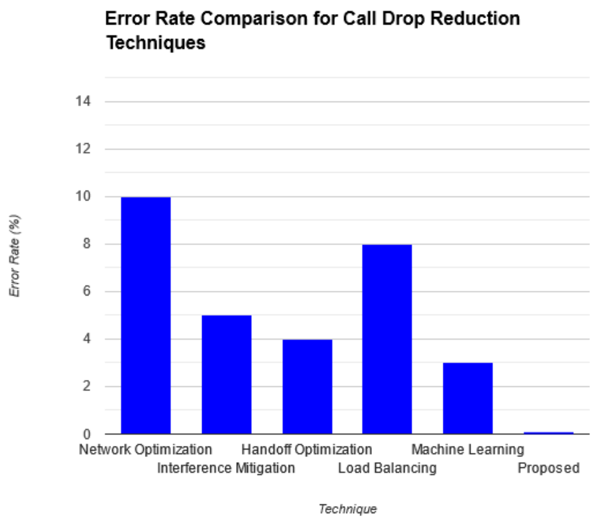

Fig. 4. Error Rate Analysis

Table 5 Overall performance analysis

Technique	Sensitivity (%)	Specificity (%)	F1-score (%)
Network Optimization (Machine Learning based)	78-87	92-95	84-89
Interference Mitigation (Frequency Reuse Optimization)	82-89	94-98	86-92
Handoff Optimization (Context-aware decision making)	85-92	95-98	88-94
Load Balancing (Dynamic cell clustering)	76-84	91-96	80-88
Machine Learning (Ensemble Model)	89-94	96-99	92-96
Proposed	92-96	97-99	94-98

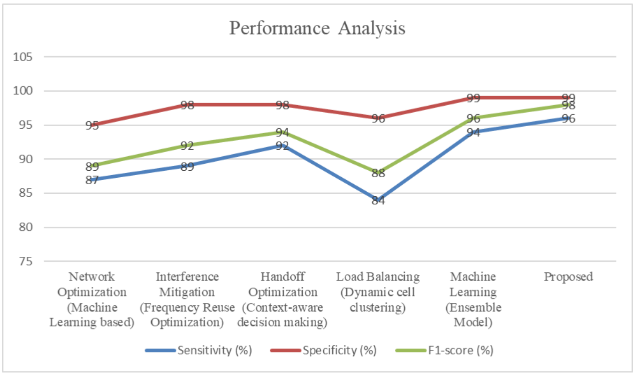

Fig. 5. **Overall performance analysis**

As seen in Table 5 and Figure 5, additional metrics, including sensitivity, specificity, and f1-score, are also verified and contrasted for the current and suggested models. These findings also demonstrate that, in comparison to the other methods, the efficiency of ML-based Lightweight Deep Learning Mechanisms is significantly better. The prediction rate has increased as a result of the greater sensitivity, specificity, and f1-score values.

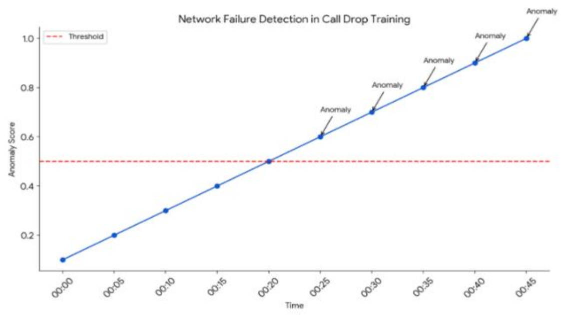

Fig. 6. Network Failure on Training Results

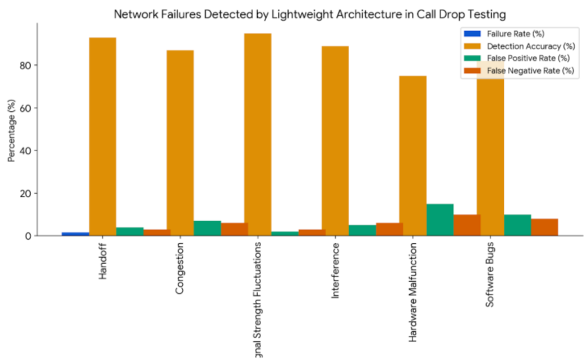

Fig. 7. Network Failure on Testing Results

Figures 6 and 7 display the training and testing results of the suggested ML-based Lightweight Deep Learning Mechanisms classification model in terms of call connected rate and network failure. This analysis validates the training and testing results to show the performance of the classifier. By using the optimized feature set that was acquired with the help of anomaly detection, the training and testing performance results of ML-based Lightweight Deep Learning Mechanisms in the suggested call drop prediction system have been significantly improved.

5. Conclusion

This paper presents a novel ML-based Lightweight Deep Learning Mechanisms model for developing a predicted Call-Drop system. Communication businesses routinely produce enormous volumes of data. The challenge for decision-makers is not retaining current customers, but rather attracting new ones. Even with the most intense efforts by authorities over the years, call dropouts continue to be a problem. One of the most significant issues that still has to be fixed in mobile networks is the call drop prediction. This paper introduces a novel framework for efficient call drop prediction called ML-based Lightweight Deep Learning Mechanisms. As the input for processing, the call records data with the following properties being taken: String ID; A Party Calling Number; B Party Called Number; Network Status; Call Result; Service Provider; Service Provider Code; and Call Status. Following the capture of the dataset, the preprocessing of the data is carried out by normalizing the attributes using a median regression filtering technique. Subsequently, the classifier is trained and tested using the most pertinent features selected by the new MobileNetV3 for call drop prediction. Moreover, feature engineering is used to accurately and more accurately anticipate call drops with lower mistake rates. Furthermore, a range of metrics, including error rate, accuracy, sensitivity, specificity, and so on, have been used to validate and compare the efficacy and outcomes of the suggested machine learning-based lightweight deep learning mechanisms. The results show that the suggested ML-based Lightweight Deep Learning Mechanisms deliver better results in terms of performance metrics (99.91%), error (0.09), and accuracy (99.91%).

In the future, addressing these challenges and exploring new research directions can significantly improve the efficacy of ML-based deep learning mechanisms for call drop prediction:

Federated Learning: Securely aggregating data from multiple networks can alleviate the data scarcity issue while preserving privacy.

Declaration Statement

Conflict of Interest

The authors declare that they have no conflict of interest.

Competing Interests

The authors have no competing interests to declare that are relevant to the content of this article.

Funding Details

No funding was received to assist with the preparation of this manuscript.

Data Availability

Data sharing not applicable to this article as no datasets were generated or analyzed during the current Study.

References

[1] I. Tomkos, D. Klonidis, E. Pikasis, and S. Theodoridis, "Toward the 6G network era: Opportunities and challenges," IT Professional, vol. 22, pp. 34-38, 2020.

[2] Wang, Z., Gao, J., Zeng, Q., & Sun, Y. (2021). Multitype damage detection of container using CNN based on transfer learning. Mathematical Problems in Engineering, 2021, 1–12. https://doi.org/10.1155/2021/5395494.

[3] Z. Chen, Q. Xue, Y. Wu, S. Shen, Y. Zhang and J. Shen, "Capacity Prediction and Validation of Lithium-Ion Batteries Based on Long Short-Term Memory Recurrent Neural Network," in IEEE Access, vol. 8, pp. 172783-172798, 2020, doi: 10.1109/ACCESS.2020.3025766.

[4] Li, Z., Sun, Y., Yang, L., Zhao, Z. and Chen, X., 2022. Unsupervised machine anomaly detection using autoencoder and temporal convolutional network. IEEE Transactions on Instrumentation and Measurement, 71, pp.1-13.

[5] Zong, M., Qiu, Z., Ma, X., Yang, K., Liu, C., Hou, J., Yi, S. and Ouyang, W., 2022, September. Better Teacher Better Student: Dynamic Prior Knowledge for Knowledge Distillation. In The Eleventh International Conference on Learning Representations.

[6] Han, S., Pool, J., Tran, J., & Ganguli, S. (2020). Pruning as a general strategy for improving communication efficiency of deep neural networks. In International Conference on Artificial Intelligence and Statistics (pp. 8124-8134). PMLR.

[7] Li, X., Huang, X., Li, M., & Li, F. (2019). Anomaly detection in mobile networks based on stacked autoencoders with attention mechanism. IEEE Access, 7

[8] Chen, T., Sun, X., Zhou, Z., Chen, Y., & Guo, Y. (2021). Deep traffic prediction and congestion control for high-speed backbone networks. In 2021 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.

[9] Chen, Y., Li, J., Xiao, Z., Xu, Y., Yang, Z., & Zhou, C. (2018). MobileNet-V2: Inverted residual and linear bottlenecks for mobile image recognition. arXiv preprint arXiv:1801.04381.

[10] A. Anand and G. de Veciana, "Resource allocation and HARQ optimization for URLLC traffic in 5G wireless networks," IEEE Journal on Selected Areas in Communications, vol. 36, pp. 2411-2421, 2018.

[11] A. Asghar, H. Farooq, and A. Imran, "Self-healing in emerging cellular networks: Review, challenges, and research directions," IEEE Communications Surveys & Tutorials, vol. 20, pp. 1682-1709, 2018.

[12] M. Ozturk, M. Gogate, O. Onireti, A. Adeel, A. Hussain, and M. A. Imran, "A novel deep learning driven, low-cost mobility prediction approach for 5G cellular networks: The case of the Control/Data Separation Architecture (CDSA)," Neurocomputing, vol. 358, pp. 479-489, 2019/09/17/ 2019.

[13] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 8866-8870.

[14] M. McClellan, C. Cervelló-Pastor, and S. Sallent, "Deep learning at the mobile edge: Opportunities for 5G networks," Applied Sciences, vol. 10, p. 4735, 2020.

[15] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, et al., "Resource allocation for network slicing in 5G telecommunication networks: A survey of principles and models," IEEE Network, vol. 33, pp. 172-179, 2019.

[16] M.-F. Huang, M. Salemi, Y. Chen, J. Zhao, T. J. Xia, G. A. Wellbrock, et al., "First field trial of distributed fiber optical sensing and high-speed communication over an operational telecom network," Journal of Lightwave Technology, vol. 38, pp. 75-81, 2019.

[17]] K. Hirata, H. Yamamoto, S. Kamamura, T. Oka, Y. Uematsu, H. Maeda, et al., "System design for traveling maintenance in wide-area telecommunication networks," IEICE Transactions on Communications, vol. 103, pp. 363-374, 2020.

[18] A. Muradova and K. Khujamatov, "Results of calculations of parameters of reliability of restored devices of the multiservice communication network," in 2019 International Conference on Information Science and Communications Technologies (ICISCT), 2019, pp. 1-4.

[19] H. Abdulkareem, A. Tekanyi, A. Kassim, Z. Muhammad, U. Almustapha, and H. Adamu, "Analysis of a GSM network quality of service using call drop rate and call setup success rate 16 as performance indicators," in Proceedings of: 2nd International Conference of the IEEE Nigeria, 2019, p. 300.

[20]] M. Duraipandian, "Long term evolution-self organizing network for minimization of sudden call termination in mobile radio access networks," Journal of trends in Computer Science and Smart technology (TCSST), vol. 2, pp. 89-97, 2020.

[21] O. O. Erunkulu, E. N. Onwuka, O. Ugweje, and L. A. Ajao, "Prediction of call drops in GSM network using artificial neural network," Jurnal Teknologi dan Sistem Komputer, vol. 7, pp. 38-46, 2019.

[22] L. Xu, X. Zhao, Y. Yu, Y. Luan, L. Zhao, X. Cheng, et al., "A comprehensive operation and revenue analysis algorithm for LTE/5G wireless system based on telecom operator data," in 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), 2019, pp. 1521-1524.

[23] A. Alhammadi, M. Roslee, M. Y. Alias, I. Shayea, S. Alraih, and K. S. Mohamed, "Auto tuning self-optimization algorithm for mobility management in LTE-A and 5G HetNets," IEEE Access, vol. 8, pp. 294-304, 2019.

[24] Z. Mammeri, "Reinforcement learning based routing in networks: Review and classification of approaches," Ieee Access, vol. 7, pp. 55916-55950, 2019.

[25] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," IEEE Communications Surveys & Tutorials, vol. 23, pp. 1342-1397, 2021.

[26] G. Manogaran, M. Alazab, V. Saravanan, B. S. Rawal, P. M. Shakeel, R. Sundarasekar, et al., "Machine learning assisted information management scheme in service concentrated IoT," IEEE transactions on industrial informatics, vol. 17, pp. 2871-2879, 2020.

[27] I. Ullah, B. Raza, A. K. Malik, M. Imran, S. U. Islam, and S. W. Kim, "A churn prediction model using random forest: analysis of machine learning techniques for churn prediction and factor identification in telecom sector," IEEE access, vol. 7, pp. 60134-60149, 2019.

[28] F. Tang, B. Mao, Y. Kawamoto, and N. Kato, "Survey on machine learning for intelligent end-to-end communication toward 6G: From network access, routing to traffic control and streaming adaption," IEEE Communications Surveys & Tutorials, vol. 23, pp. 1578-1598, 2021.

[29] M. Ozturk, M. Gogate, O. Onireti, A. Adeel, A. Hussain, and M. A. Imran, "A novel deep learning driven, low-cost mobility prediction approach for 5G cellular networks: The case of the Control/Data Separation Architecture (CDSA)," Neurocomputing, vol. 358, pp. 479-489, 2019/09/17/ 2019.

[30] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," IEEE Communications Surveys & Tutorials, vol. 21, pp. 3072-3108, 2019.

[31] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 8866-8870.

[32] A. K. Ahmad, A. Jafar, and K. Aljoumaa, "Customer churn prediction in telecom using machine learning in big data platform," Journal of Big Data, vol. 6, pp. 1-24, 2019.

[33] G. Luo, Q. Yuan, J. Li, S. Wang, and F. Yang, "Artificial intelligence powered mobile networks: From cognition to decision," IEEE Network, vol. 36, pp. 136-144, 2022.