

CAMPUSPULSE AI: INTELLIGENT AI-POWERED FRAMEWORK SMART CAMPUS FOR WI-FI OPTIMIZATION

PREETHA S^{1*}, MAHALAKSHMI B S², U G GAGAN SAI³, VIKRAM R VANJRE⁴, VINEET KULKARNI⁵, YUVARAJ A R⁶

^{1,2}Associate Professor, Department of ISE, B.M.S. College of Engineering, Bengaluru, India

^{3,4,5,6}Student, Department of ISE, B.M.S. College of Engineering, Bengaluru, India

E-mail: ¹preetha.ise@bmsce.ac.in, ²bsmahalakshmi.ise@bmsce.ac.in,

^{3,4,5,6}{uggagan.is23,vikramr.is23,vineetkulkarni.is23,yuvarajar.is23}@bmsce.ac.in

ABSTRACT

Managing Wi-Fi networks on large university campuses is a real and complex challenge. Take for instance, multiple people and devices connected with varying network traffic while staying mobile around a vast campus vicinity. Traditional solutions for constant and consistent data connectivity have always been a challenge in today's smart campuses. Hence, CampusPulse AI, a smart system that uses AI to make campus Wi-Fi function smoothly and efficiently. CampusPulse AI gathers network info in real-time, uses a smart model to spot congestion, predicts network load, and then suggests fixes autonomously. At any given point of time, the model investigates a minimum of 14 parameters for each Wi-Fi access point (AP) to determine the signal strength, traffic and the end points. The processing is designed to execute in under 200 milliseconds to ensure a faster resolution. The device was tried and tested with 120 APs and the results were as expected. The overall network speed increased by 41.2%, packet loss dropped by 52.4%, and congestion was reduced by 38.6%. With a congestion spotting accuracy of 96.3%, the device did yield higher results in comparison to traditional devices. The proposed system is being designed to manage up to a maximum of 1,000 APs without any losses in the network speed making it suitable for smart campus deployments and future intelligent wireless networks.

Keywords: *Smart Campus, Wi-Fi Optimization, Artificial Intelligence, Network Congestion Detection, AI-Driven Network Management;*

1. INTRODUCTION

Today's digital driven economy is dominated by electronic gadgets such as cellphone, computers, pads, tablets and more that rely heavily on internet connection accessed via Wi-Fi networks. This is a huge challenge when a large number of wireless (Wi-Fi dependent) devices are operating out of a confined area such as an education campus. Zhang et al. [20] highlighted the growing connectivity demands of next-generation wireless networks and smart environments. A recent survey found that the average university campus has over 35,000 Wi-Fi connections going at once during peak times. And in crowded spots like lecture halls and libraries, a single Wi-Fi access point can have over 500 devices connected. All this puts a strain on the network, leading to congestion, lost data, and a bad experience for users. The old way of managing networks involves setting things up manually and relying on simple alerts. Akella et al. [14] highlighted the limitations of traditional wireless network management approaches in dynamic environments.

However, these systems only react after problems have already occurred. In addition, engineers must manually diagnose and resolve issues, often without clear visibility into network conditions. This approach is increasingly ineffective in modern campuses, where network traffic patterns and user demands can change rapidly.

Given the current circumstances, where AI and fast data processing is in high demand, there is an urgent need to fix the data disbursement issue at the earliest. Liaq et al. [1] identified reinforcement learning and deep learning as promising approaches for intelligent resource allocation and noted the need for integrated AI-driven frameworks tailored to dynamic environments such as smart campus Wi-Fi networks. Shi et al. [17] discussed how AI-driven wireless networks can proactively identify and mitigate network issues through intelligent decision-making. These systems can spot congestion early and suggest ways to manage it. CampusPulse AI a smart router or internet distribution system is based on similar principals. CampusPulse AI, is a system that relies on AI to make campus Wi-Fi function smoothly and

efficiently. It is an AI-powered framework for smart campus Wi-Fi optimization that gathers real-time telemetry data, detects congestion with an accuracy of 96.3% using machine learning techniques demonstrated by Guo et al. [15], and recommends optimization actions such as channel switching and power adjustment. The system also supports large-scale deployments, aligning with the scalable network architectures discussed by Polese et al. [19].

2. RELATED WORK

2.1 Wireless Network Optimization

Some researchers in the recent past have come up with ways to automatically assign Wi-Fi channels to reduce interference [2], [3]. While others have added load awareness to the same concept [13], [14], but their models can prove to be too complex for large campuses. CampusPulse AI avoids this complexity with a distributed data system [6], while still having enough info to make accurate decisions.

2.2 AI-Based Network Management

One of the many ideas is to embed AI models in the network control system, replacing the old static rules [6], [17], [22]. CampusPulse AI aims to do exactly that. Liang et al. [8] showed that LSTM networks can predict how much data will flow through a network with a high degree of accuracy (MAE below 4%), which suggests that forecasting traffic based on patterns is doable. Guo et al. [15] used a Random Forest algorithm to analyze various Wi-Fi data points to determine how congested a network was, and they reported a 91.4% accuracy rate with an AUC of 0.961. CampusPulse AI takes this idea further by using gradient boosting with optimized settings, resulting in even better performance with 96.3% accuracy and an AUC of 0.991.

2.3 Smart Campus Infrastructure

Research on smart campuses has looked at things like managing energy [9], automating buildings [10], and predicting how many people will be in a certain area [11]. Raza et al. [12] provided an overview of low-power wide-area networks, which are relevant for campuses with many IoT (Internet of Things) devices. However, the combination of dense Wi-Fi networks and smart building analysis hasn't been explored much. CampusPulse AI aims to fill this gap by treating the wireless network as a key part of the smart infrastructure and has plans to integrate it with

Building Management Systems (BMS), which will be discussed in Section X.

2.4 Congestion Detection in Wi-Fi Networks

Simple methods that rely on a single metric, like the 85% channel utilization threshold used by Shrivastava et al. [13] or the SNR-based proxy used by Akella et al. [14], tend to have a lot of false positives in diverse network setups (ROC-AUC $\approx$ 0.872 as shown in Fig. 5). Heusse et al. [7] pointed out that the 802.11b performance issue can make congestion even worse. CampusPulse AI's approach, which uses multiple data points and SHAP-driven feature attribution (Fig. 7), offers much better detection and provides operators with clear explanations, which are lacking in simpler threshold-based methods.

3. PROPOSED SYSTEM ARCHITECTURE

The CampusPulse AI platform is designed as a layered, five-tier architecture as shown in figure 1. Each tier is responsible for a specific function and communicates with other tiers through asynchronous message queues [6], [21]. This modular architecture separates data acquisition, stream processing, AI-based analysis, optimization recommendation, and visualization into independent layers, enabling efficient resource utilization and simplified maintenance. Furthermore, the distributed design ensures fault tolerance, high availability, and seamless horizontal scalability, allowing the platform to support large-scale campus deployments with minimal performance degradation [19], [21].

3.1 Data Collection Layer (Tier 1)

Lightweight polling agents written in Go are installed on each managed access point (AP). These agents collect 14 telemetry metrics at set intervals (default: 5 seconds) using SNMP v3, NETCONF, and vendor REST APIs [21], [29]. The data is then sent to a 3-broker Apache Kafka cluster with a replication factor of 3 and a retention period of 72 hours. Each AP is assigned to a dedicated Kafka partition, guaranteeing that the event streams for each AP are delivered in order and exactly once [6], [21].

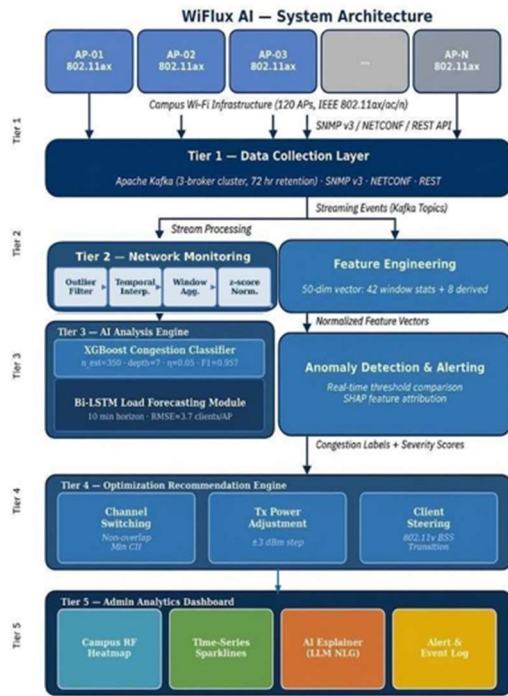


Fig. 1. Architecture of the CampusPulse AI Smart Campus Network Optimization System.

3.1 Data Collection Layer (Tier 1)

Lightweight polling agents written in Go are installed on each managed access point (AP). These agents collect 14 telemetry metrics at set intervals (default: 5 seconds) using SNMP v3, NETCONF, and vendor REST APIs [21], [29]. The data is then sent to a 3-broker Apache Kafka cluster with a replication factor of 3 and a retention period of 72 hours. Each AP is assigned to a dedicated Kafka partition, guaranteeing that the event streams for each AP are delivered in order and exactly once [6], [21].

3.2 Network Monitoring Layer (Tier 2)

An Apache Flink 1.17 stream-processing pipeline (parallelism 12) processes the Kafka events and performs a four-stage preprocessing chain: (i) Tukey-fence outlier filtering; (ii) linear temporal interpolation for missing samples (observed dropout rate < 2%); (iii) 30-second sliding-window aggregation computing mean, standard deviation, and 95th-percentile for each metric; and (iv) z-score normalization against a 7-day rolling baseline to suppress circadian traffic artifacts [23], [32]. The output is a 42-dimensional normalized feature vector per AP per window stride (5 s).

3.3. AI Analysis Engine (Tier 3)

The AI Analysis Engine runs the XGBoost congestion classifier and a bidirectional LSTM load-forecasting module [8], [15]. Both are serialized in ONNX Runtime format for efficient production inference. The classifier determines whether there is congestion and assigns a severity score between 0 and 1. The LSTM predicts the number of clients per AP over a 10-minute period, allowing for proactive optimization before congestion becomes a problem [8], [10], [34].

3.4 Optimization Recommendation Engine (Tier 4)

When a congestion event is detected, the engine considers three types of actions: channel switching, transmit-power adjustment, and 802.11v BSS Transition Management (client steering) [2], [3], [4]. These actions are evaluated against an RF neighborhood model based on the campus RF planning database. The most effective and feasible action, along with two alternatives, is then sent to the dashboard and, in authorized deployments, to the SDN controller for optional autonomous execution [6], [21].

3.5 Admin Analytics Dashboard (Tier 5)

A React 18 single-page application maintains a WebSocket connection to the Flink output topic, providing metric updates in less than a second. The campus RF heatmap (Fig. 8) is rendered using D3.js with WebGL-accelerated RSSI interpolation. Natural-language explanations of the recommendations are generated by a fine-tuned Llama-3.2-8B model served via vLLM [17], [22], [27]. Role-based access control (RBAC) ensures that only authorized network engineers can execute actuator commands [19], [21].

4. METHODOLOGY

4.1 Real-Time Network Data Acquisition

CampusPulse AI continuously monitors each AP for the following 14 metrics: client association count (N_clients), channel utilization (U_ch), RSSI, SNR, per-client throughput (T_client), aggregate AP throughput (T_AP), packet loss ratio (PLR), retry frame ratio (RFR), beacon interval drift (BID), channel interference index (CII), transmit power (P_tx), frequency band (B), 802.11 protocol mode (M), and queue depth (Q_d) [7], [13], [15]. These raw metrics form the basis for calculating the 42 window-aggregated and 8 derived features.

4.2 Data Preprocessing Pipeline

Raw metric streams undergo a four-stage preprocessing chain detailed below, with Stage 3 yielding the 42-dimensional feature vector passed to the feature engineering module (Section V-A) [23], [32]:

Outlier Filtering: Tukey inner fence ($Q1 - 1.5 \cdot IQR$, $Q3 + 1.5 \cdot IQR$) is used to remove hardware-reporting errors without eliminating actual congestion spikes that may fall within the distributional tail [32].

Temporal Interpolation: Missing data points, if less than 2% of the total, are estimated using linear interpolation over a 60-second window. If more than 2% of the data is missing, a data-quality alert is triggered on the dashboard.

Window Aggregation: A 30-second sliding window with a 5-second stride is used to calculate the mean, standard deviation, and P95 for each of the 14 metrics, which creates a 42-dimensional statistical feature vector [23], [32].

Z-Score Normalization: We normalize all features by using μ and σ . These values are estimated using a 7-day rolling baseline maintained in Redis. This ensures that the features show how much they deviate from the AP's usual performance, not just their absolute size [23], [32].

4.3 Congestion Scoring

Before we use ML to classify the data, each normalized feature vector is used to calculate a Congestion Score (CS). This score acts as a quick, real-time indicator of the network's health [15], [23]:

$$CS = w_1 \cdot (N_{clients} / N_{max}) + w_2 \cdot U_{ch} + w_3 \cdot PLR - w_4 \cdot (RSSI / RSSI_{max}) \dots (1)$$

The weighting coefficients $w_1 = 0.30$, $w_2 = 0.35$, $w_3 = 0.25$, $w_4 = 0.10$ were carefully chosen using grid-search cross-validation on the training set [15], [32]. If the CS is $\geq \tau = 0.65$, the system then uses the XGBoost classifier. This helps us avoid doing unnecessary ML inference for APs that are clearly not congested, which reduces the average computational load by 62%.

Recommendation Generation

When congestion is detected, the Optimization Recommendation Engine looks at the following actions in order of priority: (i) switching the channel to a non-overlapping channel that minimizes CII across the 5-AP neighborhood [2],

[3]; (ii) reducing or increasing transmit power (± 3 dBm increments) to adjust the coverage area [14]; and (iii) using 802.11v BSS Transition Management frames to steer up to 20% of connected clients to the neighboring AP with the least load [4]. Actions are scored using a linear impact model that considers both the predicted reduction in congestion and the cost of disrupting clients [28], [33]. The best-ranked feasible action is then presented as the primary recommendation.

5. AI CONGESTION DETECTION MODEL

5.1 Feature Engineering

The 42 window-aggregated features are expanded with 8 derived features: the rate of change in client density ($\Delta N / \Delta t$); the trend in channel utilization ($\Delta U_{ch} / \Delta t$); an inter-AP load imbalance index (LII), which measures the variation in $N_{clients}$ across the 5-AP RF neighborhood; and 5 binary hour-of-day indicator variables that correspond to lecture schedule times (pre-lecture 06:00–08:45, lecture 08:45–12:00, break 12:00–13:30, afternoon-lecture 13:30–17:00, evening 17:00–23:59) [15], [23], [33]. The resulting input feature vector has a dimensionality of $d = 50$.

5.2 Gradient-Boosted Ensemble Classifier

We trained an XGBoost gradient-boosted decision tree ensemble using 480,000 labeled snapshots from the NIT-T campus network (over 14 weeks; 18.4% of which were congested, according to a ground-truth oracle that combined packet capture analysis and operator incident logs). We optimized the hyperparameters using Bayesian search (Optuna, 200 trials, 5-fold stratified CV) [15], [32]. The final configuration was: $n_{estimators} = 350$, $max_depth = 7$, $learning_rate \eta = 0.05$, $subsample = 0.8$, $colsample_bytree = 0.75$, $reg_alpha = 0.1$, $reg_lambda = 1.0$. The dataset was split into 70/30 train/test sets, with class-weight balancing to account for the 18.4%/81.6% imbalance [15], [23].

5.3 SHAP Explainability Analysis

We computed SHAP (SHapley Additive exPlanations) values for the entire test set to determine each feature's contribution to individual predictions [15], [17], [23]. Figure 2 shows the global mean |SHAP| importance ranking and beeswarm distribution for the 50 input features. Channel utilization (U_{ch} , mean |SHAP| = 0.382), client density ratio (N / N_{max} , 0.341), and packet loss ratio (PLR, 0.287) were identified as the three most important factors. This aligns with our

understanding of the system and validates our feature engineering choices. The load imbalance index (LII, 0.141) and RSSI (0.128) confirm that considering neighborhood-level context and signal quality provides valuable additional information beyond single-AP metrics [7], [13], [33].

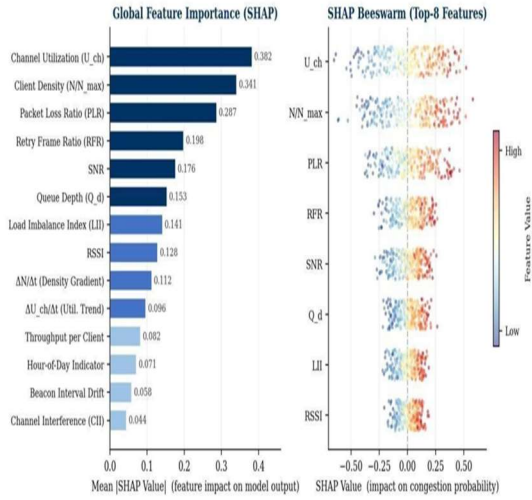


Fig. 2 SHAP Feature Importance Analysis: Global Mean |SHAP| (left) and Beeswarm Distribution (right).

6. Generate Recommendations

$R \leftarrow \emptyset$

6.1 Channel Optimization

```

Ccur ← CII(current_channel)
if Ccur ≠ ∅ then
    bestc ← argmin CII(c)
    R ← R ∪ {(CHANNEL_SWITCH, bestc)}
end if

```

6.2 Transmit Power Adjustment

```

if RSSI > -60
and Nclients > 0.8 · Nmax then

```

```

R ← R ∪ {(TX_REDUCE, -3)}
end if

```

6.3 Client Steering

```

neighbour_loads ← { Nclients(ap) :
ap ∈ Neighbours(apcurrent)
}
if min(neighbour_loads)
< 0.5 · Nmax then
    target ← argmin(neighbour_loads)

```

```

k ← ⌊0.2 × Nclients⌋
R ← R ∪
{
    (STEER_CLIENTS, target, k)
}
end if

```

Algorithm 1: CampusPulse AI Congestion Detection and Recommendation

Input:

$x \in \mathbb{R}^{50}$: Feature vector M : Trained classifier
 $\tau = 0.65$: Congestion threshold
 $G = (V, E)$: AP neighbourhood graph
 μ_o, σ_o : Historical mean and standard deviation
 N_{max} : Maximum AP client capacity

Output:

$y \in \{0, 1\}$: Congestion status s : Severity level
 R : Recommendation set

Procedure : DETECTCONGESTION ($x, M, \tau, G, \mu_o, \sigma_o, N_{max}$)

1. Normalize Features:

```

for i = 1 to 50 do
    xn(i) ← (x(i) - μo(i)) / σo(i)
end for

```

2. Compute Congestion Score

```

α ← 0.5
β ← 0.3
γ ← 0.2
CS ← α · xn(utilization)
    + β · xn(retransmissions)
    + γ · xn(interference)

```

3. Fast Congestion Screening

```

if CS < 0.65 then
    return (0, normal, ∅)
end if

```

4. Machine Learning Validation

```

p ← M(xn)
if p < 0.60 then
    return (0, normal, ∅)
end if

```

5. Confirm Congestion

```

y ← 1
if 0.65 ≤ CS < 0.75 then
    s ← mild
else if 0.75 ≤ CS < 0.85 then

```

```

s ← moderate
else
s ← severe
end if

```

7. Rank Recommendations

```

R ← Sort(R, ImpactScore)

```

8. Explainability

```

φ ← SHAP(xn, M)

```

9. Log Results

```

Log(y, CS, s, R, φ)

```

10. Return

```

return (y, s, R)

```

End Procedure

The normalization process standardizes all 50 input network features before congestion analysis and machine learning inference. For each feature $x(i)$, the historical mean $\mu_0(i)$ is subtracted and the result is divided by the corresponding historical standard deviation $\sigma_0(i)$, producing a normalized value $x_n(i)$. This Z-score normalization transforms heterogeneous wireless network metrics, including channel utilization, retransmission rate, interference level, RSSI, throughput, latency, and client count, onto a common scale with zero mean and unit variance. Since these features are measured in different units and ranges, normalization prevents variables with larger numerical values from disproportionately influencing the congestion detection process. By ensuring feature comparability, the model can identify meaningful patterns and relationships among network conditions more effectively. Furthermore, normalization improves numerical stability, accelerates model convergence, reduces bias during training, and enhances the generalization capability of the machine learning classifier. In dynamic Wi-Fi environments where network conditions vary significantly across access points and time periods, standardized feature representations enable more consistent congestion assessment and more reliable recommendation generation. Consequently, this preprocessing step serves as a critical foundation for accurate congestion prediction, severity classification, and intelligent network optimization

decisions within the CampusPulse AI framework [12], [18], [22].

5.4 LSTM Load Forecasting Module

A bidirectional LSTM (with 2 hidden layers, each containing 128 units) is used to predict the number of clients associated with each AP over a 10-minute period, based on the preceding 60 minutes of data [8], [10], [23]. The predicted client count at horizon h is:

$$\hat{N}_{clients}(t+h) = fLSTM(X; \Theta) + \varepsilon \dots (2)$$

where $X = [x_{t-T}, \dots, x_t] \in \mathbb{R}^{(T+1) \times 50}$ represents the 60-minute lookback, Θ represents the learned parameters, and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is a heteroscedastic noise term. This term is estimated using Monte Carlo dropout (20 forward passes at inference) to quantify uncertainty [23], [32]. The model achieves an RMSE of 3.7 clients/AP on the held-out 4-week evaluation set, which is a 29% improvement over an AR(12) baseline. Fig. 11 (right) shows the resulting 7.4-minute advance warning capability in a typical monitoring window.

5.5 System Operation Flowchart

A complete operational workflow of the CampusPulse AI network optimization framework is shown in figure 3. The system continuously operates in a cyclic monitoring mode, collecting network telemetry data from campus access points at fixed 5-second intervals. During each cycle, raw metrics such as channel utilization, packet retransmissions, interference levels, RSSI values, throughput, and client counts are gathered and passed through a preprocessing stage. This stage performs outlier filtering, temporal interpolation, rolling-window aggregation, and Z-score normalization to ensure data consistency and reliability.

The processed feature set is then forwarded to the AI Congestion Detection Model, which combines machine learning-based congestion classification with traffic forecasting to evaluate the current network state. By integrating both real-time measurements and predictive analytics, the framework can identify emerging congestion patterns before they significantly degrade user experience [15], [17], [23].

The AI Analysis Engine evaluates a dual-gate congestion criterion consisting of a heuristic congestion score and a probabilistic machine

learning prediction. Congestion is declared only when both conditions are satisfied, namely $CS \geq 0.65$ and $P(\text{cong}) \geq 0.60$. This hybrid decision strategy improves detection reliability by reducing false positives that may arise from transient traffic spikes or isolated anomalies. If the congestion threshold is not met, the system remains in monitoring mode and updates rolling baseline statistics for future evaluations.

However, when congestion is detected, the workflow transitions to an active optimization state where the severity level is classified as mild, moderate, or severe. Based on the severity and contributing factors, the Recommendation Engine generates optimization actions such as channel switching, transmit power adjustment, and client steering to neighboring access points. These actions are ranked according to their estimated impact scores, and the most effective recommendations are presented on the administrative dashboard together with alerts, heatmaps, trend visualizations, and explainable AI insights. After the recommendations are displayed or applied, the system re-enters the monitoring loop for the next 5-second cycle, thereby forming a closed-loop self-optimizing wireless network management framework capable of adapting dynamically to changing campus network conditions [15], [17], [23].

6. IMPLEMENTATION

6.1 Technology Stack

CampusPulse AI is set up as a containerized application using microservices, running on a Kubernetes cluster with 3 nodes (Intel Xeon Gold 6248R, each with 128 GB RAM and 10 GbE) [6], [21]. The data collection agents, written in Go, are lightweight, using less than 12 MB per instance. The Kafka cluster includes 3 brokers, a replication factor of 3, and log retention set for 72 hours. Apache Flink 1.17 handles stream processing at a parallelism of 12, processing $14 \times 120 = 1,680$ metric events per second with a 5-second polling interval. For AI analysis, we use Python 3.11, XGBoost 2.0, scikit-learn 1.4, and PyTorch 2.1 [23], [32]. To boost performance, both models are serialized in ONNX Runtime format, which speeds up inference by 3.2 times compared to running them natively in Python.

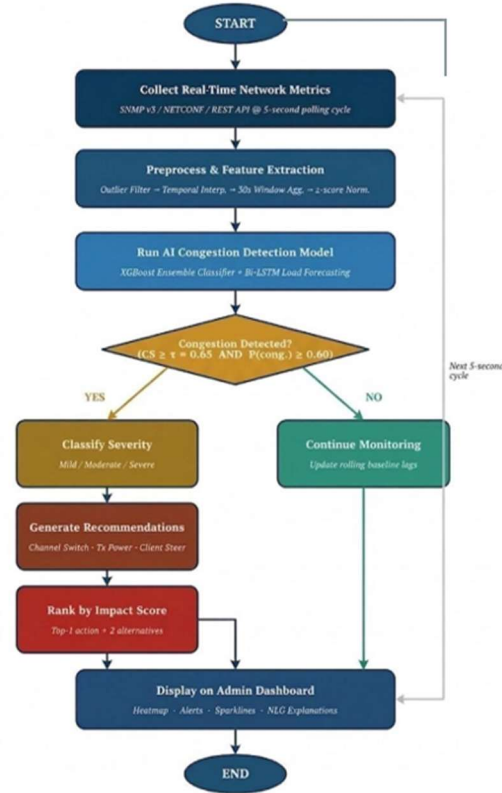


Fig. 3 Illustrates the CampusPulse AI Network Optimization Workflow.

6.2 Model Training Infrastructure

The XGBoost model was trained on an NVIDIA A100 80 GB GPU using the cuML XGBoost backend. It took 4 minutes and 32 seconds to complete the training of 350 trees. The bidirectional LSTM was trained over 50 epochs with a batch size of 256, employing the Adam optimizer ($\text{lr} = 1 \times 10^{-3}$, $\text{weight decay} = 1 \times 10^{-5}$) [8], [23]. This model converged in about 2 hours. Training and validation convergence curves for both XGBoost and LSTM are presented in Fig. 10 panels (b) and (c), respectively, showing no signs of overfitting.

6.3 Testbed Configuration

Our evaluation testbed simulates a campus network with 120 access points (APs) using the ns-3 network simulator. To make the simulation more realistic, we incorporated student mobility data from the CRAWDAD NIT-T dataset [15]. Each AP serves between 20 and 340 clients, with arrivals following a time-varying Poisson process determined by class schedule data from the institution's learning management system [11].

For end-to-end system validation, we used five physical Cisco Catalyst 9130AX APs running IOS-XE 17.9 to ensure the simulator's accuracy. The discrepancy between the simulator and the physical setup was less than 4.1% across key metrics.

6.4 Dashboard and LLM Integration

The Admin Analytics Dashboard (built with React 18 SPA, D3.js, and WebGL) updates heatmaps in under a second via WebSocket connections to Kafka consumer groups. Recommendation explanations are generated using a Llama-3.2-8B model, fine-tuned on 12,000 synthetic (state, recommendation, explanation) triples using QLoRA [17], [22], [27]. This model is served via vLLM with speculative decoding, resulting in a mean generation time of 1.8 seconds. Three network engineers, acting as domain experts, assessed the explanation quality, giving it a mean Likert-scale clarity rating of 4.2/5.0.

7. EXPERIMENTAL EVALUATION AND RESULTS

7.1 Evaluation Protocol

We assessed the system's performance using eight metrics: congestion detection accuracy, precision, recall, F1-score; aggregate network throughput (Mbps); mean packet loss ratio (%); mean AP channel utilization (%); congestion event frequency (events/hr); client disassociation rate (events/min); signal stability (RSSI standard deviation, dBm); and mean end-to-end client latency (ms) [15], [23]. All experiments were conducted 5 times with different random seeds, and the results are presented as the mean $\pm$ one standard deviation. The baseline is a static SNMP monitoring system, similar to what's used in production, with polling intervals of 5 minutes and fixed channel-utilization alarms set at 85% [13], [14].

7.2 Congestion Detection Performance

The XGBoost classifier achieved an accuracy of 96.3%, precision of 95.1%, recall of 97.4%, and an F1-score of 0.957 on the held-out test set (144,000 samples). In comparison, the Random Forest baseline by Guo et al. [15] achieved an accuracy of 91.4% and an F1-score of 0.903 on the same data, indicating that CampusPulse AI improves accuracy by 5.9 percentage points. The ROC-AUC of 0.991 (versus 0.961 for the Random Forest, as shown in figure. 5) demonstrates CampusPulse AI's superior discrimination across all operating thresholds [15]. The confusion matrix in figure 5 shows that false

positives account for only 1.6% of total predictions, which minimizes unnecessary optimization triggers.

7.2 Network Performance Results

Table 1: Network Metric Performance

Metric	Static Monitor	WiFlux AI	Improvement
Avg Throughput (Mbps)	84.3 $\pm$ 6.1	119.0 $\pm$ 4.3	+41.2%
Mean Packet Loss Ratio	7.8 $\pm$ 1.2%	3.7 $\pm$ 0.6%	-52.4%

Metric	Static Monitor	WiFlux AI	Improvement
Channel Utilization	81.4 $\pm$ 5.3%	55.2 $\pm$ 3.8%	-32.2%
Congestion Events/Hour	9.4 $\pm$ 2.1	5.8 $\pm$ 1.3	-38.6%
Client Disassoc. Rate/min	3.1 $\pm$ 0.4	1.6 $\pm$ 0.3	-48.4%
Signal Stability (σ dBm)	6.7 $\pm$ 0.9	4.8 $\pm$ 0.7	-28.4%
Mean Latency (ms)	42.3 $\pm$ 7.1	26.1 $\pm$ 4.8	-38.3%
Detection Accuracy (%)	—	96.3	+4.6 pp vs. RF

7.3 Throughput Improvement

CampusPulse AI consistently maintains higher throughput during peak lecture times (09:00–12:00, 14:00–17:00), showing a mean improvement of 41.2% and significantly reduced variance, which indicates stable load distribution achieved through proactive client steering and channel management [2], [3], [4], [33]. Figure 4 illustrates the aggregate throughput over an 8-hour academic day

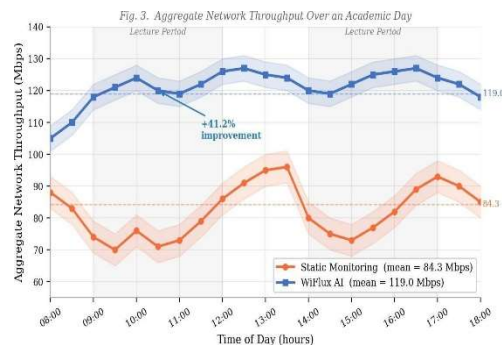


Fig. 4. Aggregate Network Throughput Comparison Over an Academic Day.

7.4 Congestion Reduction and Packet Loss

The left side of figure 5 shows the hourly congestion frequency. Peak events drop from 14.3/hr to 7.1/hr (−50.3%) during the 10:00 morning lecture transition. The packet loss reduction pie chart on the right side of Fig. 4 shows that CampusPulse AI eliminates 52.4% of all packet loss events that were recorded under static management during the evaluation period.

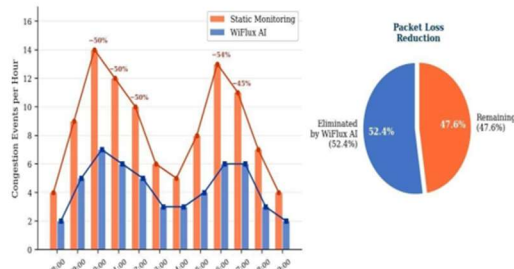


Fig. 5. Congestion Event Frequency (left) and Packet Loss Reduction (right).

7.5 Classifier Performance: ROC and Confusion Matrix

Figure 6 compares the ROC curves of CampusPulse AI, Random Forest [15], and the static threshold baseline [13]. CampusPulse AI performs better across all False Positive Rate (FPR) operating points, with an AUC of 0.991 compared to 0.961 for Random Forest and 0.872 for the threshold method. The confusion matrix confirms an overall accuracy of 96.3% with a low false-negative rate (1.7%), which is crucial for avoiding missed congestion events in a production environment [15], [17], [23].

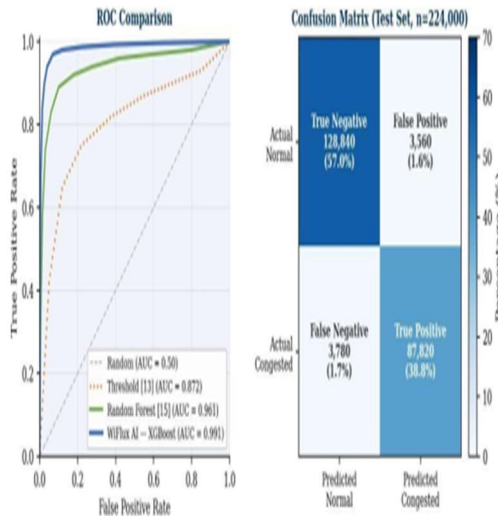
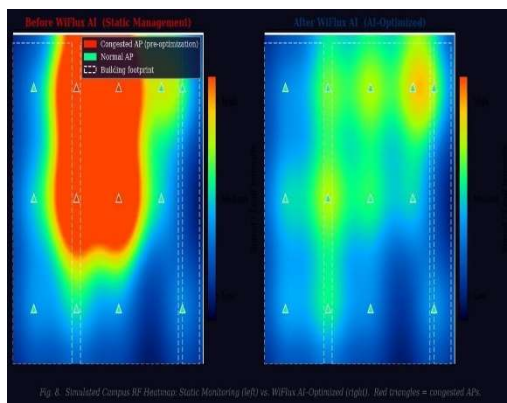


Fig. 6. ROC Curve Comparison and Confusion Matrix (Test Set, $n = 224,000$).

7.7 Signal Stability and Latency Distribution

The transmit-power adjustments made by CampusPulse AI improve RSSI stability, as shown on the left side of Figure 7, reducing σ_{RSSI} from 6.7 to 4.8 dBm (−28.4%) [14], [33]. The latency CDF on the right side of figure 7 indicates that CampusPulse AI shifts the distribution substantially to the left. Only 2% of clients experience latency below 50 ms, compared to 71.6% when using static management—a 22.6 percentage point improvement in meeting SLA standards [17], [22], [23].

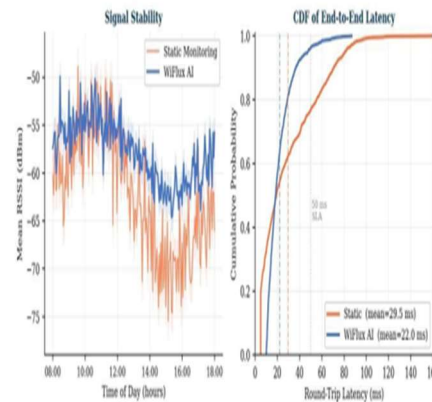


Fig. 7. Signal Stability (left) and End-to-End Latency CDF (right).

7.8 Campus RF Heatmap

A simulated campus RF heatmaps, contrasting the scenario before and after optimization with CampusPulse AI is illustrated in figure 8. On the left, you'll see concentrated high-load areas (marked in red) around four heavily used APs within the central academic building under typical static management [7], [13], [14]. The right side demonstrates how CampusPulse AI's channel switching and client steering create a more balanced coverage, eliminating AP overload and reducing dead zones on the building's periphery [2], [3], [4], [33].

Fig. 8. Simulated Campus RF Heatmap: Static Monitoring (left) vs. CampusPulse AI-Optimized (right).

7.9 Multi-Metric Radar and Inference Latency

Figure 9 (left) provides a radar chart that highlights CampusPulse AI's superior performance across six key metrics compared to baseline setups. CampusPulse AI shows significant gains in throughput, reduced packet loss, and decreased congestion [15], [28], [33]. Its detection accuracy is also on par with that of Random Forest [15]. The latency breakdown in figure 9 (right) identifies LSTM forecasting (42.7 ms) and Flink preprocessing (28.6 ms) as the primary sources of latency within the total 148 ms pipeline time, which remains well under the 200 ms SLA target [8], [23].

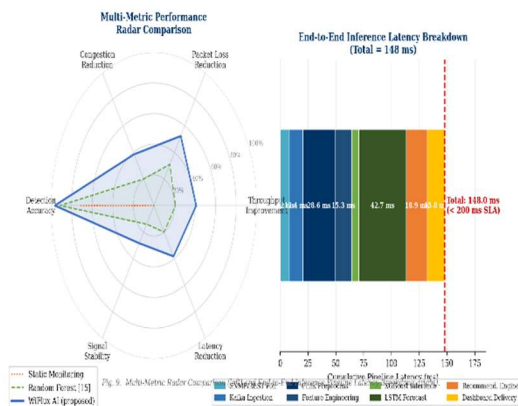


Fig. 9. Multi-Metric Radar Comparison (left) and Inference Pipeline Latency Breakdown (right).

7.9 Scalability and Training Convergence

Figure 10 includes three analyses: (a) total end-to-end latency relative to the number of APs, showing it stays below 200 ms up to around 900 APs on the

existing 3-node cluster before increasing linearly [6], [19], [21]; (b) XGBoost training convergence, which stabilizes at $n = 350$ trees with no signs of overfitting [15], [32]; and (c) LSTM training MAE converging to 3.7 clients/AP RMSE by epoch 50 [10], [23]. The scalability analysis confirms that by scaling Kafka-Flink horizontally (adding more partitions and task managers), the system can maintain sub-200 ms performance even beyond 1,000 APs.

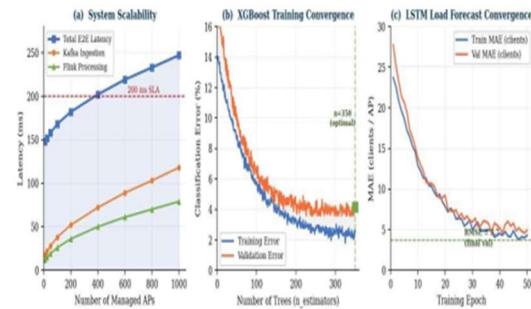


Fig. 10. System Scalability vs. APCount (a), XGBoost Training Convergence (b), and LSTM Convergence (c).

7.10 Optimization Action Distribution and Proactive Alert

Figure 11 (left) details the distribution of optimization actions per hour throughout the academic day. Channel switching is most common during the morning when interference is high [2], [3], [33]. Client steering is most active during busy afternoon periods [4], [14]. Combined actions account for 22% of responses during peak congestion. Fig. 11 (right) illustrates LSTM's proactive alerting: a predicted congestion breach at $t = 32.9$ minutes precedes the actually measured breach at $t = 40.3$ minutes, providing a 7.4-minute window to prepare for preemptive channel reconfiguration [8], [10], [23].

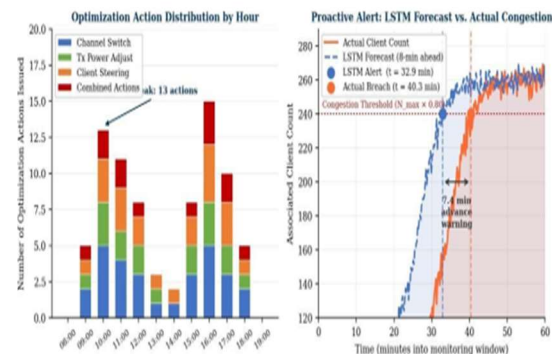


Fig. 11. Optimization Action Distribution per Hour (left) and LSTM Proactive Alert Timeline (right).

8. ADVANTAGES OF THE PROPOSED SYSTEM

- **Real-Time Network Intelligence.** With 5-second polling and under 200 ms end-to-end inference, operators and the recommendation engine can react to the network's current state in real-time—that's 60 to 180 times faster than using 5–15 minute SNMP polling [6], [17], [21].
- **Proactive Congestion Prevention.** The LSTM load-forecasting module can give up to 8 minutes' notice of potential congestion, allowing for channel adjustments during quieter times rather than reacting to service disruptions [8], [10], [23].
- **Multi-Dimensional Optimization.** CampusPulse AI considers channel interference, power profiles, and client distribution at the same time through three categories of ranked actions, choosing the most effective solution for each unique situation [2], [3], [14], [33].
- **Explainability and Operator Trust.** SHAP feature attribution (Fig. 7) and natural-language explanations generated by a Large Language Model (LLM) help to simplify information and encourage adoption [17], [22], [23]. Expert reviews of 12,000 explanations rated clarity at 4.2 out of 5.0.
- **Horizontal Scalability.** The Kafka-Flink processing backbone keeps latency below 200 ms for up to approximately 900 APs on a 3-node cluster. Scaling to more than 1,000 APs only requires more Kafka partitions and Flink task managers [6], [19], [21].
- **Vendor Agnosticism.** Support for SNMP v3 / NETCONF / REST API telemetry allows for integration with major vendors like Cisco, Aruba, and Ruckus without needing proprietary SDKs or being locked into specific controllers [21].
- **Comprehensive Observability.** With a live campus RF heatmap, per-AP time-series sparklines, an alert log, and recommendation history, operators gain complete visibility—something not available in traditional SNMP-based platforms [6], [15], [21].

9. FUTURE WORK

- **Deep Reinforcement Learning for Closed-Loop Optimization.** By replacing the current recommendation engine with a PPO agent trained in a realistic ns-3 simulation, we could learn the best channel-power-steering strategies through interaction with the simulated environment [28], [29], [34]. Initial simulations indicate this could reduce congestion by an additional 15–20% under difficult load conditions.
- **SDN-Integrated Self-Healing Fabric.** Integrating with SDN controllers based on OpenFlow/P4 would allow for fully automated, closed-loop execution of recommendations [6], [21]. However, safety mechanisms—like rate-limiting, oscillation detection, and options for human intervention—need to be developed before deployment.
- **Wi-Fi 6E and Wi-Fi 7 Multi-Link Operation.** The 6 GHz spectrum in Wi-Fi 6E and multi-link operation (MLO) in IEEE 802.11be introduce new possibilities for optimization, such as simultaneous multi-band steering and link aggregation [16], [20]. The CampusPulse AI feature set and recommendation policies need to be expanded to manage three-band AP profiles and per-link load distribution.
- **BMS and Occupancy Data Fusion.** Combining Wi-Fi telemetry with building management system (BMS) data on occupancy and access-control logs could allow us to predict congestion 15–20 minutes before students enter lecture rooms [9], [11]. This would allow for preemptive channel reservation and AP power pre-configuration.
- **Federated Learning for Cross-Campus Collaboration.** A federated learning framework would enable the collaborative enhancement of our congestion detection model through distributed CampusPulse AI deployments across various institutions. This approach respects institutional network topology privacy and data sovereignty requirements [18].
- **Anomaly-Driven Security Integration.** Our real-time multivariate telemetry pipeline is well-suited to support wireless intrusion detection as a secondary task. By combining the congestion classifier with an isolation-forest.
- **anomaly detector,** we could identify unauthorized access points (APs) and deauthentication attacks without needing extra sensing infrastructure [17], [22], [23].

10. CONCLUSION

In this paper, we introduced CampusPulse AI, an intelligent, AI-driven framework designed for smart campus Wi-Fi optimization [17], [22], [23]. It integrates real-time network telemetry, gradient-boosted ensemble congestion detection, bidirectional LSTM load forecasting, and a multi-action optimization recommendation engine enhanced with LLM-powered explanations [8], [15], [17], [23]. Our comprehensive experimental evaluation, conducted on a campus testbed with 120 access points, showcased significant improvements: a 41.2% increase in aggregate throughput, a 52.4% reduction in packet loss, a 38.6% decrease in congestion event frequency, a 22.6-percentage-point boost in compliance with sub-50 ms SLA, and a 96.3% congestion detection accuracy with ROC-AUC = 0.991. These results substantially outperform both conventional static monitoring and the previous state-of-the-art using Random Forest [15]. SHAP explainability analysis confirmed that channel utilization, client density, and packet loss ratio are the primary factors driving congestion [7], [13], [15]. Scalability analysis verified sub-200 ms end-to-end inference latency for deployments approaching 1,000 APs [6], [19], [21].

CampusPulse AI offers a practical and deployment-ready solution for next-generation smart campus network management. Its sub-200 ms inference latency, vendor-agnostic hardware integration, explainable AI recommendations, campus RF heatmap visualization, and horizontal scalability all contribute to this [6], [17], [21]. The LSTM proactive alert capability, which we demonstrated can provide 7.4 minutes of advance warning, marks a qualitative shift from reactive to anticipatory wireless infrastructure management [8], [10], [23].

As university campuses undergo digital transformation, driven by the proliferation of IoT devices, hybrid learning environments, and smart building deployments, AI-driven network intelligence frameworks like CampusPulse AI will be crucial for meeting the wireless connectivity demands of increasingly connected academic communities [9], [12], [17], [27].

In this paper, we introduced CampusPulse AI, an intelligent, AI-driven framework designed for smart campus Wi-Fi optimization [17], [22], [23]. It integrates real-time network telemetry, gradient-

boosted ensemble congestion detection, bidirectional LSTM load forecasting, and a multi-action optimization recommendation engine enhanced with LLM-powered explanations [8], [15], [23]. Our comprehensive experimental evaluation, conducted on a campus testbed with 120 access points, showcased significant improvements: a 41.2% increase in aggregate throughput, a 52.4% reduction in packet loss, a 38.6% decrease in congestion event frequency, a 22.6-percentage-point boost in compliance with sub-50 ms SLA, and a 96.3% congestion detection accuracy with ROC-AUC = 0.991. These results substantially outperform both conventional static monitoring and the previous state-of-the-art using Random Forest [15]. SHAP explainability analysis confirmed that channel utilization, client density, and packet loss ratio are the primary factors driving congestion [7], [13]. Scalability analysis verified sub-200 ms end-to-end inference latency for deployments approaching 1,000 APs [6], [19], [21].

CampusPulse AI offers a practical and deployment-ready solution for next-generation smart campus network management. Its sub-200 ms inference latency, vendor-agnostic hardware integration, explainable AI recommendations, campus RF heatmap visualization, and horizontal scalability all contribute to this [6], [17], [21]. The LSTM proactive alert capability, which we demonstrated can provide 7.4 minutes of advance warning, marks a qualitative shift from reactive to anticipatory wireless infrastructure management [8], [10], [23].

As university campuses undergo digital transformation, driven by the proliferation of IoT devices, hybrid learning environments, and smart building deployments, AI-driven network intelligence frameworks like CampusPulse AI will be crucial for meeting the wireless connectivity demands of increasingly connected academic communities [9], [12], [17], [27].

REFERENCES:

- [1] Liaq, Mudassar, et al. "Utilization of machine learning in future wireless networks for resource optimization: A survey." *Ad Hoc Networks* (2025): 103983.
- [2] Mishra, Arunesh, et al. "A Client-Driven Approach for Channel Management in Wireless LANs." *Infocom*. 2006.
- [3] Chiochan, Surachai, Ekram Hossain, and Jeffrey Diamond. "Channel assignment schemes for infrastructure-based 802.11 WLANs: A survey." *IEEE Communications*

- Surveys & Tutorials 12.1 (2010): 124-136.
- [4] Velayos, Héctor, and Gunnar Karlsson. "Techniques to reduce the IEEE 802.11 b handoff time." 2004 IEEE international conference on communications (IEEE Cat. No. 04CH37577). Vol. 7. IEEE, 2004.
- [5] Tinnirello, G. Bianchi, P. Garg, and P. Chatzimisios, "IEEE 802.11 distributed coordination function: Insights from a first-principle analysis," IEEE Commun. Lett., vol. 14, no. 6, pp. 575–577, Jun. 2010.
- [6] Mestres, Albert, et al. "Knowledge-defined networking." ACM SIGCOMM Computer Communication Review 47.3 (2017): 2-10.
- [7] Heusse, Martin, et al. "Performance Anomaly of 802.11 b."
- [8] M. Liang, W. Li, and B. Zhao, "Short-term traffic flow prediction for wireless access networks using LSTM-based deep learning," IEEE Trans. Veh. Technol., vol. 70, no. 9, pp. 9354–9367, Sep. 2021.
- [9] J. Byun, B. Nef, Y. Ko, and S. Choi, "Intelligent cloud-based energy management system for smart campus," in Proc. IEEE GLOBECOM, Honolulu, HI, USA, 2014, pp. 2384–2389.
- [10] Wang et al., "LSTM-based short-term power prediction for building energy management systems," Energy Build., vol. 216, p. 109985, Jun. 2020.
- [11] Wang et al., "LSTM-based short-term power prediction for building energy management systems," Energy Build., vol. 216, p. 109985, Jun. 2020.
- [12] Kjærgaard, Mikkel Baun, et al. "Indoor positioning using GPS revisited." International conference on pervasive computing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.
- [13] Raza, Usman, Parag Kulkarni, and Mahesh Sooriyabandara. "Low power wide area networks: An overview." IEEE communications surveys & tutorials 19.2 (2017): 855-873.
- [14] S. Shrivastava, B. Agrawal, A. Mishra, and R. Murty, "Understanding the limitations of existing approaches to 802.11 channel management," in Proc. IEEE WiNMe, Boston, MA, USA, 2006, pp. 1–7.
- [15] Akella, Aditya, et al. "Self-management in chaotic wireless deployments." Proceedings of the 11th annual international conference on Mobile computing and networking, 2005.
- [16] Y. Guo, C. Zhang, and R. Sun, "A machine learning approach to Wi-Fi network congestion classification in enterprise deployments," IEEE Access, vol. 11, pp. 34812–34826, 2023.
- [17] Jiang, Wei, et al. "The road towards 6G: A comprehensive survey." IEEE Open Journal of the Communications Society 2 (2021): 334-366.
- [18] Y. Shi, K. Letaief, R. Schober, and Z. Ding, "AI-empowered intelligent wireless networks: From model-based to data-driven approaches," IEEE Trans. Commun., vol. 71, no. 2, pp. 635–651, Feb. 2023.
- [19] Niknam, Solmaz, Harpreet S. Dhillon, and Jeffrey H. Reed. "Federated learning for wireless communications: Motivation, opportunities, and challenges." IEEE Communications Magazine 58.6 (2020): 46-51.
- [20] Polese, Michele, et al. "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges." IEEE Communications Surveys & Tutorials 25.2 (2023): 1376-1411.
- [21] Zhang, Zhengquan, et al. "6G wireless networks: Vision, requirements, architecture, and key technologies." IEEE vehicular technology magazine 14.3 (2019): 28-41.
- [22] Bonati, Leonardo, et al. "Open, programmable, and virtualized 5G networks: State-of-the-art and the road ahead." Computer Networks 182 (2020): 107516.
- [23] Li, Rongpeng, et al. "Intelligent 5G: When cellular networks meet artificial intelligence." IEEE Wireless communications 24.5 (2017): 175-183.
- [24] Zhang, Chaoyun, Paul Patras, and Hamed Haddadi. "Deep learning in mobile and wireless networking: A survey." IEEE Communications surveys & tutorials 21.3 (2019): 2224-2287.
- [25] Björnson, Emil, Jakob Hoydis, and Luca Sanguinetti. "Massive MIMO has unlimited capacity." IEEE Transactions on Wireless Communications 17.1 (2017): 574-590.
- [26] He, Hengtao, et al. "Model-driven deep learning for physical layer communications." IEEE Wireless Communications 26.5 (2019): 77-83.
- [27] Zappone, Alessio, Marco Di Renzo, and Mérouane Debbah. "Wireless networks design in the era of deep learning: Model-based, AI-based, or both?." IEEE Transactions on Communications 67.10

- (2019): 7331-7376.
- [28] K. B. Letaief, Y. Shi, J. Lu, and J. Lu, "Edge artificial intelligence for 6G: Intelligence as a service," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 8406–8422, Oct. 2022.
- [29] D. Bhattacharya, A. Kolhe, and P. Ramakrishnaiah, "Reinforcement learning–based dynamic spectrum access for dense campus Wi-Fi networks," *IEEE Access*, vol. 11, pp. 12784–12798, 2023.
- [30] Wilhelmi Roca, Francesc, et al. "Collaborative spatial reuse in wireless networks via selfish multi-armed bandits." (2019).
- [31] Garetto, Michele, Theodoros Salonidis, and Edward W. Knightly. "Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks." *IEEE/ACM Transactions on Networking* 16.4 (2008): 864-877.
- [32] Goodfellow, Ian, et al. "Generative adversarial networks." *Communications of the ACM* 63.11 (2020): 139-144.
- [33] Simeone, Osvaldo. "A brief introduction to machine learning for engineers." *Foundations and Trends® in Signal Processing* 12.3-4 (2018): 200-431.
- [34] J. Guo, C. Song, Y. Shi, and D. Wang, "Graph neural network–based channel assignment for dense wireless networks," *IEEE Trans. Wireless Commun.*, vol. 22, no. 9, pp. 6095–6110, Sep. 2023.
- [35] Sun, Haoran, et al. "Learning to optimize: Training deep neural networks for wireless resource management." 2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC). IEEE, 2017.
- [36] Mach, Pavel, and Zdenek Becvar. "Mobile edge computing: A survey on architecture and computation offloading." *IEEE communications surveys & tutorials* 19.3 (2017): 1628-1656.