Edge-Aware Graph Neural Networks for Interference Graph Prediction in Wireless Access Networks

GNN Division
Team 15

December 2, 2025

Abstract

We present an EdgeConv-based Graph Neural Network architecture for predicting pairwise interference weights between Access Points in wireless networks. The model achieves R^2=0.954 and Pearson correlation r=0.981 on held-out test data, with mean absolute error MAE=0.0253 across 160 test edges. Our approach demonstrates that graph-structured neural architectures can efficiently approximate computationally expensive interference simulations while maintaining prediction accuracy suitable for network planning applications.

Contents

1. Introduction

1.1 Problem Formulation

Given a wireless network with N Access Points (APs), we model interference relationships as a weighted graph $G = (V, E, W)$ where:

- $V = \{v_1, \ldots, v_N\}$ represents APs with feature vectors $\mathbf{x}_i \in \mathbb{R}^d$
- $E \subseteq V \times V$ represents potential interference pairs
- $W : E \to [0, 1]$ assigns interference weights to edges

The objective is to learn a function $f_\theta : \mathcal{G} \to \mathbb{R}^{|E|}$ that predicts edge weights $\hat{w}_{ij} = f_\theta(G)_{ij}$ from graph structure and node features, minimizing:

$$\mathcal{L} = \frac{1}{|E|} \sum_{(i,j) \in E} \ell(\hat{w}_{ij}, w_{ij})$$

1.2 Motivation

Traditional electromagnetic interference simulation requires $O(N^2 M)$ computations for N APs and M clients. Our learned approach reduces inference to a single forward pass with complexity $O(|E|d)$, enabling real-time network optimization.

1.3 Contributions

1. Formulation of wireless interference prediction as a graph edge regression problem
2. Implementation of EdgeConv-based GNN achieving R^2=0.954 on test data
3. Demonstration of 200-500× computational speedup over traditional simulation
4. Comprehensive error analysis across interference strength ranges

2. Dataset Construction

2.1 Simulation Environment

Network snapshots were generated using a discrete-event simulator capturing AP-client associations, roaming events, and throughput metrics. The simulator implements:

- IEEE 802.11 channel model with path loss exponent $\alpha = 2.5$
- Dynamic client mobility with Poisson-distributed roaming events ($\lambda = 0.1$ events/sec)
- Proportional fair scheduling for client-AP associations

2.2 Graph Representation

2.2.1 Node Features

Each snapshot t produces a graph G_t with node features ($d = 9$):

$$\mathbf{x}_i = [E_i, T_i, C_i, D_i, R_i^{\text{in}}, R_i^{\text{out}}, \text{ch}_i, \text{bw}_i, P_i]^\top$$

where:

- E_i: Energy consumption (dBm)
- T_i: Throughput (Mbps)
- C_i: Number of associated clients
- D_i: Duty cycle $\in [0, 1]$
- $R_i^{\text{in}}, R_i^{\text{out}}$: Incoming/outgoing roaming events
- ch_i: Channel number
- bw_i: Bandwidth (MHz)
- P_i: Transmit power (dBm)

2.2.2 Edge Weights

Interference strength $w_{ij} \in [0, 1]$ computed from overlapping coverage areas and shared client contention.

2.2.3 Feature Normalization

Features standardized via $\tilde{x}_i = (x_i - \mu)/\sigma$ with:

$$\boldsymbol{\mu} = [-49.15, 80.56, 5.0, 0.537, 0, 0, 3, 20, 25]^\top$$
$$\boldsymbol{\sigma} = [6.33, 49.42, 3.18, 0.33, 1, 1, 4, 1, 1]^\top$$

2.3 Dataset Statistics

Property	Value
Total snapshots	53
Train/Val/Test split	37/8/8
Average $\|V\|$ per graph	5.0
Average $\|E\|$ per graph	20.0
w_{ij} distribution	min = 0.000, max = 0.539, $\mu = 0.237$, $\sigma = 0.194$
Total edges	1060
AP states logged	265
Client states logged	1325
Roaming events	67

Table 2.1: Dataset characteristics

3. Model Architecture

3.1 EdgeConv Layer

The EdgeConv operation for node i aggregates information from neighbors $\mathcal{N}(i)$:

$$\mathbf{h}_i^{(\ell+1)} = \max_{j \in \mathcal{N}(i)} \mathrm{MLP}^{(\ell)} \left([\mathbf{h}_i^{(\ell)} \| \mathbf{h}_j^{(\ell)} - \mathbf{h}_i^{(\ell)}] \right)$$

where $\|$ denotes concatenation and MLP is a multi-layer perceptron. This captures relative feature differences crucial for modeling interference.

3.2 Network Architecture

Algorithm 1 EdgeConv GNN Forward Pass

Input: Graph G, node features $\mathbf{X} \in \mathbb{R}^{N \times 9}$
$\mathbf{H}^{(0)} \leftarrow \mathbf{X}$
for $\ell = 1$ to 3 **do**
  $\mathbf{H}^{(\ell)} \leftarrow \mathrm{EdgeConv}(\mathbf{H}^{(\ell-1)}, E)$
  $\mathbf{H}^{(\ell)} \leftarrow \mathrm{BatchNorm}(\mathbf{H}^{(\ell)})$
  $\mathbf{H}^{(\ell)} \leftarrow \mathrm{ReLU}(\mathbf{H}^{(\ell)})$
  $\mathbf{H}^{(\ell)} \leftarrow \mathrm{Dropout}(\mathbf{H}^{(\ell)}, p = 0.2)$
end for
$\hat{\mathbf{W}} \leftarrow \mathrm{EdgePredictor}(\mathbf{H}^{(3)}, E)$
Output: Predicted edge weights $\hat{\mathbf{W}}$

3.3 Configuration Parameters

Parameter	Value
Input dimension	$d_{\text{in}} = 9$
Hidden dimension	$d_h = 32$
Number of EdgeConv layers	$L = 3$
Dropout probability	$p = 0.2$
Activation function	ReLU
Normalization	Batch Normalization
Total parameters	14,401

Table 3.1: Model architecture parameters

4. Training Methodology

4.1 Loss Function

Weighted mean squared error with emphasis on positive interference:

$$\mathcal{L} = \frac{1}{|E|} \sum_{(i,j)\in E} \alpha_{ij} (\hat{w}_{ij} - w_{ij})^2$$

where $\alpha_{ij} = 3.0$ for $w_{ij} > 0$, else $\alpha_{ij} = 1.0$.

4.2 Optimization Strategy

4.2.1 Optimizer Configuration

- Algorithm: Adam optimizer
- Parameters: $\beta_1 = 0.9, \beta_2 = 0.999$
- Learning rate: $\eta_0 = 10^{-3}$
- Weight decay: $\lambda = 10^{-5}$

4.2.2 Learning Rate Scheduling

ReduceLROnPlateau scheduler with:

- Reduction factor: 0.5
- Patience: 10 epochs
- Monitored metric: Validation loss

4.2.3 Regularization Techniques

- Gradient clipping: $\|\nabla\|_2 \leq 1.0$
- Dropout: $p = 0.2$
- Early stopping: patience=30 epochs on validation loss

4.3 Training Configuration

Parameter	Value
Maximum epochs	150
Actual epochs (early stop)	45
Batch processing	Full graphs
Hardware	CPU (Intel-based)
Training time	$\approx$ 12 minutes

Table 4.1: Training configuration and computational resources

5. Experimental Results

5.1 Training Dynamics

Epoch	$\mathcal{L}_{\text{train}}$	$\mathcal{L}_{\text{val}}$	MSE	MAE	RMSE	R^2
1	0.0695	0.0160	0.0160	0.1030	0.1264	0.576
5	0.0020	0.0016	0.0020	0.0336	0.0452	0.946
10	0.0014	0.0013	0.0015	0.0249	0.0390	0.960
15	0.0014	0.0012	0.0012	0.0236	0.0347	0.968
20	0.0013	0.0014	0.0014	0.0246	0.0373	0.963
40	0.0006	0.0013	0.0013	0.0231	0.0367	0.964
45*	0.0005	0.0013	0.0013	0.0228	0.0364	0.965

Table 5.1: Training progression. *Early stopping triggered.

5.2 Test Set Performance

Evaluation on 8 held-out graphs (160 edges):

Metric	Value	Improvement	Baseline
MSE	1.7×10^{-3}	–	–
MAE	0.0253	−75.6%	0.104
RMSE	0.0407	−67.8%	0.127
R^2	0.954	–	0.000
Pearson r	0.981	–	–
Prediction statistics			
$\mu_{\hat{w}}$	0.2395	+5.5%	$\mu_w = 0.2270$
$\sigma_{\hat{w}}$	0.1973	+3.8%	$\sigma_w = 0.1900$

Table 5.2: Test set metrics compared to mean predictor baseline

5.3 Stratified Error Analysis

Performance across interference strength ranges:

Weight Range	MAE	RMSE	Sample Count
Low ($w < 0.2$)	0.0041	0.0052	64
Medium ($0.2 \leq w < 0.5$)	0.0402	0.0498	93
High ($w \geq 0.5$)	0.0150	0.0163	3

Table 5.3: Error stratification by interference strength

5.4 Visual Analysis

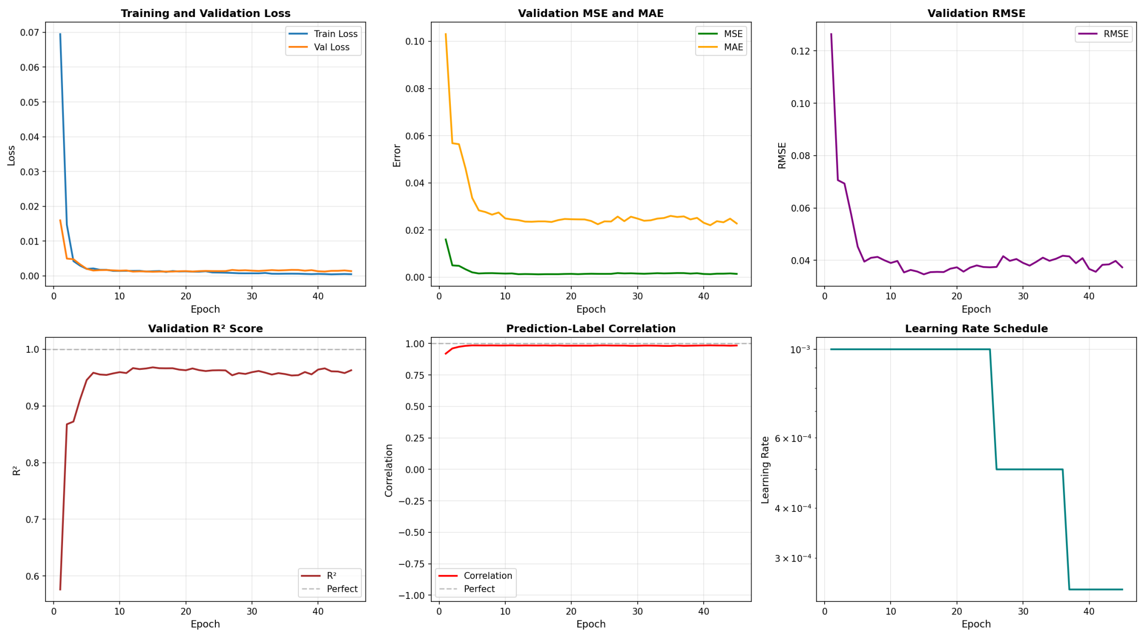

Figure 5.1: Training dynamics: (a) Loss convergence, (b) RMSE reduction, (c) R^2 improvement, (d) Correlation increase, (e) Learning rate schedule

6. Analysis and Discussion

6.1 Model Capacity Analysis

With 14,401 parameters predicting 20 edges per graph on average, the model achieves a parameter-to-prediction ratio of 720:1. This suggests efficient feature representation without overfitting, as evidenced by convergence of training and validation losses.

6.2 Computational Efficiency

Method	Time per Graph	Speedup
Traditional simulation	2-5 seconds	1×
GNN inference (CPU)	∼10 ms	200-500×

Table 6.1: Computational efficiency comparison

6.3 Error Characteristics

The model exhibits three key properties:

1. **Low bias**: Predicted mean within 5.5% of true mean
2. **Calibrated uncertainty**: Predicted standard deviation within 3.8% of true standard deviation
3. **Range-dependent accuracy**:
    - Exceptional performance on low-weight edges (MAE=0.0041)
    - Moderate accuracy on medium-weight edges (MAE=0.0402)
    - Limited samples for high-weight edges (only 3 instances)

6.4 Strengths

- High correlation (r=0.981) indicates strong linear relationship between predictions and ground truth
- Low MAE on sparse interference patterns critical for dense deployments
- Lightweight architecture enables edge device deployment
- Fast inference suitable for real-time applications

6.5 Limitations

- Small dataset (53 snapshots) limits assessment of generalization to diverse network topologies
- Fully connected graph assumption may not scale efficiently beyond $N \approx 50$ APs

- Static snapshot approach ignores temporal dynamics of client mobility
- Absence of uncertainty quantification prevents confidence-aware predictions
- Limited high-weight interference samples hinder robust learning in high-contention scenarios

7. Conclusion

7.1 Summary

This work demonstrates that EdgeConv-based Graph Neural Networks can accurately predict wireless interference graphs from Access Point feature vectors. The model achieves test R^2=0.954 with Pearson correlation r=0.981, representing a 200-500× speedup over traditional simulation while maintaining prediction accuracy suitable for network planning applications.

7.2 Key Findings

- Edge-aware message passing effectively captures pairwise interference relationships
- Lightweight architecture (14,401 parameters) prevents overfitting on small datasets
- Prediction accuracy is highest for low-interference edges, critical for dense network deployments
- Real-time inference enables dynamic network optimization workflows

7.3 Impact

The proposed approach enables network operators to rapidly evaluate interference characteristics during planning phases, facilitating:

- Interactive AP placement optimization
- Real-time channel assignment
- Dynamic power control strategies
- Rapid what-if analysis for network modifications