Journal of Scientific Reports 2026, 15(1), 283-303
ISSN: 2708-7085 (Online) & 3079-9317 (Print)
Journal: ijsab.com/jsr

Intelligent QoS Prediction in Software-Defined Networking Via Lightweight Graph-Temporal Machine Learning

Chernor Illah Jalloh [1*], Julliet Josiah [1], Christiana Lebbie [1], Tom Penyikie [1], & Ishmael Sheriff [1,2]

[1] School of Electronics Information and Communication, Huazhong University of Science and Technology, Wuhan, China.
[2] School of Technology, Western Governors University (WGU), 4001 South 700 East, Suite 700 Salt Lake City, Utah 84107 United States.
* Corresponding author: Chernor Illah Jalloh (chernohillahjalloh@gmail.com)

Abstract

Timely Quality of Service (QoS) prediction can enable software-defined networks to identify emerging performance problems before users experience excessive delay, congestion, packet loss, or unstable service. This study presents LiGHT-QoS, a lightweight graph-temporal forecasting framework that combines port-level network topology and telemetry with recent end-to-end QoS measurements. The experimental dataset was generated from 120 independent Mininet and Open vSwitch runs involving linear, tree, and mesh topologies, eight operating conditions, and five repetitions for each topology and condition. Following timing validation and segment-preserving filtering, 33,152 of 36,000 one-second intervals were retained, representing a 92.09% acceptance rate and 375,868 node-level observations. Model performance was evaluated at 1, 3, and 5-second forecasting horizons using five grouped folds and five random seeds, while ensuring that data from the same run did not appear in both training and testing sets. Compared with persistence forecasting, the proposed HGTE model reduced scale-balanced macro means absolute error by 13.30%, 13.40%, and 13.00% at the three respective horizons. HGTE achieved the best overall ranking at 1 and 3 seconds and was only 0.28% behind a QoS-only GRU at 5 seconds. Jitter prediction showed the most consistent improvement, with mean absolute error reductions between 21.07% and 22.11%. The model contains only 12,452 trainable parameters and achieved an 82.63% reduction in median CPU inference latency after export to ONNX FP32. These findings demonstrate that LiGHT-QoS can support efficient controller-assisted network monitoring, although real-network validation and calibrated fallback mechanisms remain necessary before large-scale operational deployment.

Keywords: *Software-defined networking, Quality of service, Graph neural network, Temporal forecasting, Topology generalization, Machine Learning, Edge inference.*

ARTICLE INFO

Research paper

Received: 02 August 2026
Accepted: 07 November 2026
Published: 09 November 2026
DOI: 10.58970/JSR.1251

CITATION

Jalloh, C. I., Josiah, J., Lebbie, C., Penyikie, T., & Sheriff, I. (2026). Intelligent QoS Prediction in Software-Defined Networking Via Lightweight Graph-Temporal Machine Learning, *Journal of Scientific Reports*, 15(1), 283-303.

COPYRIGHT

*Copyright © **2026** by author(s) Papers published by IJSAB International are licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.*

Introduction

Software-defined networking (SDN) separates control decisions from packet forwarding and gives operators a programmable view of the network. Centralized monitoring helps an operator see the network as one system, but it does not remove the delay between detecting a problem and responding to it. If a controller waits until congestion, packet loss, or high latency is already visible to users, an interactive service may have begun to deteriorate. A reliable forecast made a few seconds earlier can support a route review, admission decision, queue adjustment, or operator warning before the problem becomes severe (Keshari et al., 2021; Prasanth & Uma, 2024). Recent QoS studies also show that route state, graph structure, and temporal context can improve prediction when they are represented jointly rather than as unrelated measurements (Li et al., 2024; Liu et al., 2024; Hu et al., 2025). QoS prediction is difficult because network state is both spatial and temporal. A congested link influences neighboring switches and end-to-end paths, while the same measurements can mean different things under different topologies or loads. Classical time-series models can learn recent movement but usually ignore the physical relationships among ports. Graph neural networks represent those relationships directly, and recurrent or attention-based encoders summarize how they evolve (Suárez-Varela et al., 2023; Hu et al., 2025). RouteNet and later graph-based network models demonstrated that learned graph representations can estimate latency, jitter, and loss under varied routing and queueing behavior (Rusek et al., 2020; Ferriol-Galmés et al., 2022, 2023). More recent work has extended graph-based QoS prediction to service-invocation and 5G settings (Liu et al., 2024; Huang et al., 2025). The remaining challenge is to obtain those benefits from controller-observed telemetry with a small model, a leakage-safe evaluation design, and claims that remain credible when the test topology or traffic condition has not been seen during training. LiGHT-QoS addresses this challenge through a Lightweight Hybrid Graph-Temporal Encoder (HGTE). The model uses two complementary inputs: ten seconds of port-level graph telemetry and the corresponding observed QoS history. It predicts mean latency, throughput, packet loss, and jitter one, three, or five seconds ahead. Its graph branch propagates information over topology-specific normalized adjacency matrices; its temporal branches use gated recurrent units (GRUs); and a compact fusion head predicts a correction to the latest QoS state. Padding masks permit one model to process the 10-node linear and tree graphs and the 14-node mesh graph without allowing padded nodes to affect pooled representations.

The empirical contribution extends beyond the model itself. A balanced set of 120 runs covers three topologies, eight network conditions, and five independent repetitions. Whenever the collection process contained a timing gap, the gap remained a segment boundary rather than being replaced by an estimated or imputed observation. Temporal windows were also prevented from crossing those boundaries. Evaluation was grouped by run, repeated across five seeds, and extended with topology holdouts, condition holdouts, neural and non-neural baselines, run-level statistical tests, and a CPU deployment gate. The independent run, rather than each overlapping one-second sample, is therefore the main unit of generalization and statistical inference.

The main finding is encouraging but not absolute. HGTE improved scale-balanced macro-MAE over persistence by about 13% at every horizon and led the six-model comparison at one and three seconds. At five seconds, a QoS-only GRU was slightly better. Jitter gains were large and statistically reliable, while latency and packet-loss behavior depended more strongly on operating condition. Leave-one-topology-out tests also produced mixed target-level outcomes. The evidence therefore supports describing the model as a compact, useful forecasting component rather than a universal predictor. That distinction is important for both scientific reproducibility and deployment decisions. The remainder of the paper reviews relevant graph-temporal and SDN forecasting work, develops testable hypotheses from the identified gaps, describes the experimental design and model, reports grouped and out-of-distribution results, and concludes with theoretical, operational, and research implications.

This study makes a data contribution by releasing a balanced experimental design rather than a collection of convenient traces. The 120 runs cover every combination of three topologies and

eight operating conditions five times. The frozen evidence records both accepted and rejected timing intervals, preserves 632 temporal segments, and links each reported result to check summed inputs and outputs. This makes the dataset suitable for independent re-analysis and for testing models that must distinguish repeatable behavior from one favorable run. The methodological contribution is a topology-aware forecasting pipeline that keeps the physical graph, temporal history, and end-to-end QoS targets aligned without interpolation across timing gaps. A single masked representation accommodates graphs with ten or fourteen port nodes, while grouped folds ensure that windows from one run never appear in both training and testing. The same protocol evaluates three forecast horizons and five random seeds, which exposes sensitivity that a single train-test split would conceal. The theoretical and modeling contribution is a compact residual forecasting design that connects three ideas that are often evaluated separately: message passing over switch-port relationships, recurrent encoding of graph and end-to-end histories, and correction of a strong persistence forecast. This design advances knowledge beyond a benchmark comparison by testing when explicit topology adds value to recent QoS history, when that value becomes small, and how the conclusion changes under unseen network structures and conditions. Its 12,452 trainable parameters also make this theoretical comparison practical for controller-side inference. The evaluation contribution is an evidence hierarchy that moves from grouped interpolation to unseen topology, unseen condition, statistical reliability, and deployment equivalence. Negative findings are reported alongside gains: HGTE does not win every target, a GRU is marginally better at five seconds, some topology transfers fail, and dynamic INT8 is rejected. These boundaries convert the paper's contribution from a broad performance claim into an operationally useful statement of where the model helps, where persistence remains safer, and what must be validated before field control.

Literature Review

Recent SDN research increasingly treats the controller as a source of measurable state for traffic engineering and QoS management. Reviews published since 2020 describe how programmable control can support application-aware routing, congestion management, and differentiated service, while also emphasizing that results obtained in emulation must not be mistaken for evidence from production hardware (Keshari et al., 2021; Prasanth & Uma, 2024). Recent graph-based networking research similarly presents learned network models as tools for planning, online optimization, and troubleshooting, but identifies generalization and interpretability as continuing challenges (Suárez-Varela et al., 2023). Mininet and Open vSwitch were selected because they expose controller and forwarding behavior in a repeatable environment, while the distinction between emulation evidence and production-network evidence remains explicit throughout the paper. Machine learning for communication networks has moved from flat feature vectors toward representations that preserve relationships among links, queues, paths, devices, users, and services. Jiang (2022) reviewed graph-based learning across wired, wireless, and software-defined networks, while Suárez-Varela et al. (2023) explained why communication systems naturally fit graph representations. For QoS forecasting, Ak and Canberk (2021) combined spatial and temporal information for multi-horizon Wi-Fi 6 prediction. RouteNet and its later variants showed that graph models can estimate delay, jitter, and loss under varied routing and queueing behavior (Rusek et al., 2020; Ferriol-Galmés et al., 2022, 2023). These studies support explicit relational modeling, but they do not by themselves establish that a compact graph branch improves short-horizon forecasts made from controller-observed port telemetry. Graph learning is useful here because the same local operation can be applied to ports in different positions, while neighboring port states are combined through the physical topology. Surveys identify such parameter sharing as a reason graph models can operate across structures of different sizes (Wu et al., 2021; Abadal et al., 2022; Waikhom & Patgiri, 2023). In service computing, QoSGNN jointly represents user-service relations and feature embeddings (Liu et al., 2024), while GACL combines a dynamic invocation graph with temporal attention (Hu et al., 2025). Dynamic route estimation has likewise been used to expose network-state changes that a temporal-only model may miss (Li et al., 2024). These findings motivate spatial-temporal fusion, although their service-invocation abstractions differ from the switch-port graph studied here.

Prior evidence leaves four connected gaps. First, many evaluations use sample-level random splits, which can place nearly adjacent windows from the same run in both training and testing and overstate generalization. Second, topology transfer is often inferred from one split or one random seed, despite the sensitivity of learned network models to structure and traffic regime (Suárez-Varela et al., 2023). Third, the incremental value of graph context is not always tested against strong temporal-only and persistence baselines. Fourth, deployment claims are sometimes based on parameter count alone, without numerical-equivalence checks, measured latency, or an explicit fallback when the operating condition is unfamiliar. The present evaluation addresses these gaps through grouped run-level folds, repeated seeds, topology and condition holdouts, matched baselines, run-level inference, and a PyTorch-to-ONNX deployment comparison.

Gaps in Existing Literature

Recent QoS studies demonstrate strong but largely separate advances. QoSGNN jointly model's user-service relations and feature embeddings (Liu et al., 2024), FRLN protects distributed invocation data through personalized federated learning (Zou et al., 2024), and EMD-EmLSTM improves dynamic industrial QoS forecasting through decomposition and residual correction (Chai et al., 2024). Li et al. (2024) connect dynamic QoS prediction with route estimation, Hu et al. (2025) combine graph attention with temporal encoding, and Huang et al. (2025) integrate graph attention and deep reinforcement learning for component-service QoS prediction in 5G. These contributions establish the value of graph structure, temporal modeling, privacy protection, and decision-oriented learning, but they address different data abstractions and deployment settings from controller-observed SDN telemetry. The resulting research gap is therefore not simply the absence of another predictor. What remains uncertain is whether a small port-level graph-temporal residual model can improve a strong persistence forecast under run-disjoint evaluation, whether it remains competitive with non-graph learned predictors, whether any gain is stable across targets and random seeds, and how strongly the conclusion changes under unseen topology or traffic conditions. These questions lead directly to the hypotheses below and to the paper's two central contributions: a compact graph-temporal residual design and an evidence framework that separates in-distribution accuracy from transfer and deployment claims.

Hypothesis Development

Prior work establishes that relational structure can improve prediction when outcomes depend on connected links and paths. Rusek et al. (2020) showed that RouteNet's message-passing formulation estimates delay, jitter, and loss more accurately than non-relational baselines under varied routing and queueing, and Ferriol-Galmés et al. (2022, 2023) extended this to Erlang and Fermi variants with improved fidelity across traffic regimes. Suárez-Varela et al. (2023) synthesize this literature and argue that graph representations suit communication systems because network state is inherently relational. However, these studies share two features that limit their direct bearing on the present question. First, they typically evaluate against weak or non-temporal baselines rather than a persistence anchor tuned to short horizons. Second, they model path-level performance from routing and queueing inputs rather than forecasting from controller-observed port telemetry with a strong recent-history reference. The incremental value of graph structure over persistence at one-to-five-second horizons therefore remains untested in this setting. Residual forecasting offers a principled way to combine both signals. Rather than reconstructing the target, a residual model estimates a correction to the latest observation, preserving the persistence anchor while allowing the learned component to adjust it. This design is conservative: when the correction is near zero, the model degrades gracefully to persistence. No prior graph-based QoS study, to our knowledge, has evaluated a residual formulation anchored to the most recent observation under run-disjoint conditions. The combination of explicit relational modeling and residual anchoring motivates the first hypothesis:

H1. *HGTE will achieve lower standardized macro-MAE than persistence at the 1-, 3-, and 5-second horizons.*

H1 is tested using grouped, run-disjoint out-of-fold predictions at each horizon, with persistence as the matched comparator and standardized macro-MAE as the ranking metric.

H2: Joint Relational-Temporal Representation at Short Horizons

The second hypothesis concerns whether jointly representing relational and temporal information outperforms representations based on either alone. This is a distinct question from H1: a model may beat persistence while still failing to justify its graph branch over a strong temporal-only alternative. Recent QoS prediction studies provide direct evidence for joint representation. Liu et al. (2024) show that QoS-GNN, which jointly encodes user service relations and feature embeddings, improves prediction over feature-only models. Li et al. (2024) connect dynamic QoS prediction with route estimation, demonstrating that route-level relational context carries information a temporal model alone may miss. Hu et al. (2025) combine graph attention with temporal encoding in GACL and report gains over temporal only baselines. Huang et al. (2025) integrate graph attention with deep reinforcement learning for component-service QoS in 5G settings. Together these studies support the general claim that relational and temporal signals are complementary. Critical examination, however, reveals three limitations for the present context. First, these studies operate on service-invocation graphs, user–service or component–service relations rather than on physical switch-port topologies observed through controller telemetry. The relational abstraction differs substantially: service graphs encode invocation affinity, whereas port graphs encode forwarding adjacency and link capacity. Second, the baselines in these studies are often weaker than a well-tuned recurrent model; the incremental value of graph context over a QoS-only GRU is not always isolated. Third, the advantage of relational context is expected to decay with forecast horizon, because current port interactions become less informative about state further in the future. None of these studies tests this horizon dependence explicitly. These considerations imply that any graph-temporal advantage should be most visible at short horizons, where current port interactions remain predictive, and should be tested against strong temporal-only comparators rather than weak baselines. The second hypothesis follows:

H2. *HGTE will outperform the evaluated non-graph learned predictors, ridge regression, the MLP, and the QoS-only GRU on standardized macro-MAE at the 1- and 3-second horizons.*

H2 is deliberately restricted to short horizons; no directional claim is made at five seconds, where the informativeness of topology is expected to diminish. The hypothesis is tested through grouped out-of-fold model ranking at each horizon.

H3: Stability and Target-Level Reliability

Suárez-Varela et al. (2023) identify sensitivity to structure and traffic regime as a continuing challenge for learned network models, and note that generalization claims are often inferred from a single split. Wu et al. (2021) and Waikhom and Patgiri (2023), in their surveys of graph neural networks, emphasize that reported gains can depend substantially on initialization and evaluation protocol. In the QoS forecasting literature specifically, Liu et al. (2024), Hu et al. (2025), and Huang et al. (2025) report strong aggregate results but do not, in general, present repeated-seed dispersion or run-level paired inference. Where confidence intervals are reported, they frequently treat overlapping temporal windows as independent samples, an assumption violated by sliding-window construction, in which adjacent windows share most of their input intervals. This practice can overstate reliability. A model that improves on average across seeds may still fail on some initializations, and a mean improvement driven by a few high-error runs may not reflect a consistent per-run benefit. The appropriate unit of inference is therefore the independent run rather than the one-second sample. Addressing this requires two complementary checks: dispersion of improvement across random seeds, and run-level paired statistical testing with correction for multiple comparisons. The third hypothesis is stated in two parts to reflect these distinct requirements:

H3a. *HGTE's improvement over persistence will remain positive across all five random seeds at every horizon.*

H3b. *At least one QoS target will show a statistically reliable run-level reduction in MAE at every horizon.*

H3a is assessed through repeated-seed evaluation across seeds 42, 52, 62, 72, and 82. H3b is assessed using bootstrap confidence intervals for mean run-level improvement and Holm-corrected two-sided Wilcoxon signed-rank tests on paired run-level differences. Support for H3 requires both components; a positive mean improvement that fails the run-level test, or a significant result confined to a single seed, would not satisfy the hypothesis.

H4: Bounded Transfer Under Unseen Topology and Condition

Prior work offers grounds for both optimism and caution. On the optimistic side, Wu et al. (2021) and Abadal et al. (2022) note that parameter sharing in graph models allows the same local operation to apply across structures of different sizes, and Rusek et al. (2020) and Ferriol-Galmés et al. (2023) demonstrate graph models operating across varied routing configurations. On the cautious side, Suárez-Varela et al. (2023) explicitly identify distribution shift and interpretability as unresolved barriers to operational deployment, and Jiang (2022) observes that graph-based network models are seldom tested on structures outside their training distribution. The service-oriented QoS studies reviewed above (Liu et al., 2024; Li et al., 2024; Hu et al., 2025; Huang et al., 2025) generally evaluate on held-out samples rather than held-out graph structures, leaving topology transfer largely unexamined. A further complication arises in residual formulations. When persistence is nearly perfect, for example, in zero-loss or low-load regimes, a small nonzero correction can appear arbitrarily poor in relative terms, even though the absolute error is negligible. This interaction between residual design and operating regime has not been characterized in prior graph-based QoS work, and it implies that transfer performance may vary sharply by target and condition rather than degrading uniformly. These considerations support a deliberately cautious expectation. Rather than predicting uniform transfer, H4 predicts greater variability under holdouts than under in-distribution evaluation, treating transfer failure as evidence delimiting the validated operating boundary:

H4. *HGTE's advantage over persistence will be more variable under unseen-topology and unseen-condition holdouts than under grouped in-distribution evaluation.*

H4 is not a directional performance claim but a prediction about the dispersion of outcomes. It is tested through leave-one-topology-out and leave-one-condition-out experiments, with results reported at the target and horizon level.

Research Methods

The experiments used a four-switch, four-host SDN testbed implemented with Mininet, Open vSwitch, and an OpenFlow controller. The source and destination hosts were placed at opposite ends of the selected prediction path. The linear topology used three core links, the tree topology changed the branching structure while retaining ten monitored port nodes, and the mesh topology added redundant switch links and increased the monitored graph to fourteen port nodes. Figure 1 summarizes the evidence pipeline, and Figure 2 shows the three physical graph arrangements used in data generation.

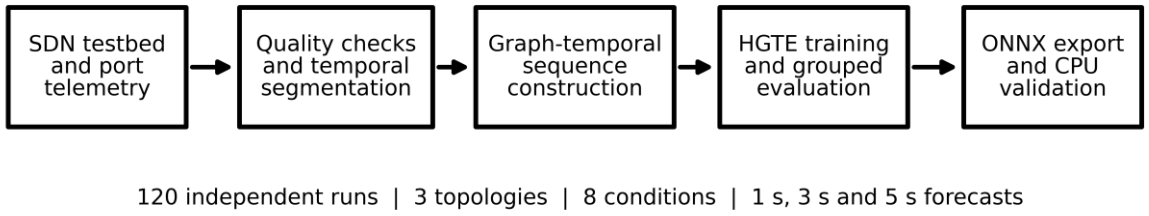


Figure 1: LiGHT-QoS experimental and evaluation pipeline

The matrix crossed three topologies with eight conditions: offered loads of 10, 25, 35, 45, and 60 Mbps; a 25 Mbps case with 5 ms link delay; a 25 Mbps case with 1% configured loss per relevant core interface; and a 45 Mbps case with a 100-packet queue. Five independently executed repetitions produced 120 runs. Each run targeted 300 one-second intervals, giving 36,000 planned intervals. Port counters were sampled approximately twice per second and aligned to interval-level ping and UDP measurements. Table 1 summarizes the balanced factors, levels, and repetitions used in the evaluation.

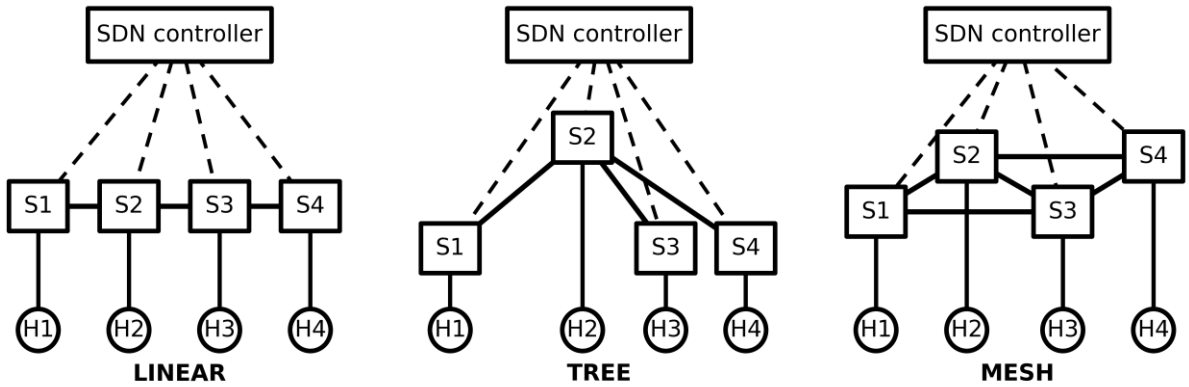


Figure 2: Controller-assisted linear, tree, and mesh test topologies

Table 1: Balanced experimental matrix

Design factor	Levels	Runs per level/cell
Topology	Linear, tree, mesh	40 per topology
Network condition	8 load, delay, loss, and queue cases	15 per condition
Repetition	Five independent executions	5 per topology-condition cell
Forecast horizon	1 s, 3 s, 5 s	All grouped evaluations

Source: *Authors' experimental design.*

Data quality was managed as part of the experimental design rather than as a separate cleaning recipe. Raw artifacts were accepted only when metadata indicated completion, the traffic-generator and ping processes returned successfully, expected switch ports were observed, and telemetry files were non-empty. Interval timing was then checked. A scheduling gap did not trigger synthetic interpolation; it ended one temporal segment and began another. Only consecutive, valid intervals were retained for sequence building, and no input window was permitted to cross a segment boundary. This choice reduced the sample count but preserved the state that the system actually observed. Let a run contain interval timestamps t_i. A new segment begins whenever the observed interval gap exceeds the configured timing tolerance, as expressed by

$$s_i = s_{i-1} + \mathbb{I}[(t_i - t_{i-1}) > \tau] \qquad (1)$$

where s_i is the segment identifier, τ is the gap tolerance, and $\mathbb{I}[\cdot]$ is the indicator function. After filtering, 33,152 intervals were accepted, corresponding to 92.09% retention. The accepted labels aligned exactly with 375,868 node-level model rows across 632 temporal segments. The larger row count for mesh runs reflects their fourteen port nodes, compared with ten for linear and tree runs. Table 2 summarizes the resulting acceptance, segmentation, and interpolation outcomes.

Table 2: Dataset processing outcome

Measure	Value	Interpretation
Independent runs	120	All planned runs passed the run gate
Original intervals	36,000	300 intervals per run
Accepted intervals	33,152	92.09% retained
Node-level rows	375,868	Topology-dependent snapshot width
Temporal segments	632	Gaps preserved as boundaries
Interpolation	None	No synthetic replacement of timing gaps

Source: *Authors' quality-control and acceptance reports.*

Each port is treated as a graph node. Nodes on the same switch are connected to represent local forwarding relationships, and opposing ports on a physical inter-switch link are connected to represent the core topology. Eleven node features describe capacity, sample count, receive and transmit rates, packet rates, drops, errors, and utilization. The graph snapshot at time t is $X_t \in \mathbb{R}^{N\times 11}$, with $N = 10$ or 14. Four end-to-end QoS values form q_t: mean latency, throughput, packet-loss percentage, and jitter. For an undirected topology with adjacency A, self-loops are added and symmetric normalization is applied:

$$\tilde{A} = A + I, \qquad \hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2} \tag{2}$$

A training example uses $W = 10$ consecutive input intervals and a target h seconds after the final input. Thus

$$\mathcal{X}_{t,h} = \{X_{t-W+1}, \dots, X_t\}, \quad Q_{t,h} = \{q_{t-W+1}, \dots, q_t\}, \quad y_{t,h} = q_{t+h} \tag{3}$$

The target interval never appears in either input. The resulting sample counts were 26,832, 25,573, and 24,321 at the 1-, 3-, and 5-second horizons. Longer horizons produce fewer eligible windows because each segment must contain the ten inputs and the future target. Table 3 reports the eligible sample count and tensor shape at each forecast horizon.

Table 3: Temporal datasets

Horizon	Samples	Runs	Input shape	Target shape
1 s	26,832	120	10 × 14 × 11	4
3 s	25,573	120	10 × 14 × 11	4
5 s	24,321	120	10 × 14 × 11	4

Source: *Authors' temporal sequence manifests.*

The resulting sequences were processed by the Lightweight Hybrid Graph-Temporal Encoder (HGTE). Figure 3 presents the evaluated architecture. HGTE contains two lightweight graph layers, a graph GRU, a QoS-history GRU, and a two-layer fusion head. The graph branch represents local and neighboring port states, while the QoS branch summarizes the recent end-to-end trajectory. For layer l, node representations are updated by separate self and neighborhood projections:

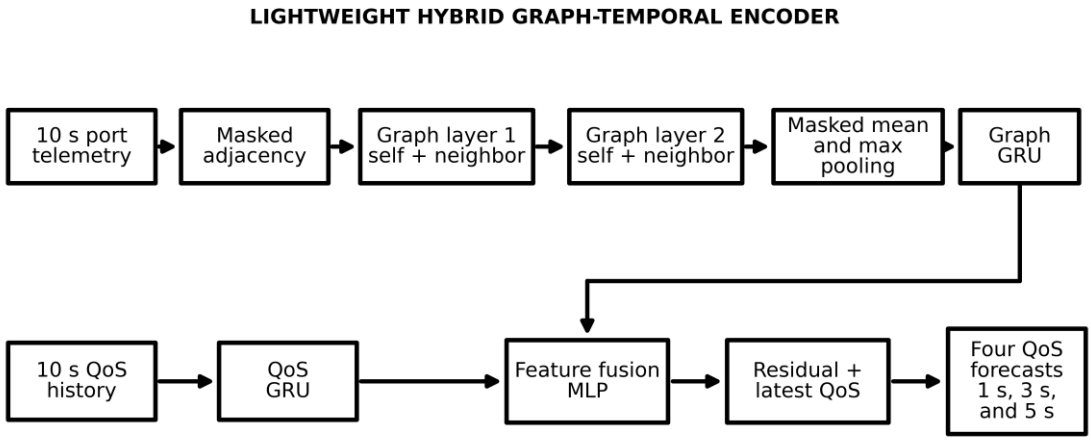


Figure 3: Evaluated HGTE architecture for multi-horizon QoS forecasting

$$H_t^{(l+1)} = \text{Dropout}\left(\text{ReLU}\left(\text{LN}\left(H_t^{(l)}W_s^{(l)} + \hat{A}H_t^{(l)}W_n^{(l)}\right)\right)\right) \quad (4)$$

Masked mean and masked maximum pooling summarize valid nodes only. The pooled graph vector g_t is

$$g_t = \left[\frac{\sum_{n=1}^{N} m_n h_{t,n}}{\sum_{n=1}^{N} m_n} \;\Big\|\; \max_{n:m_n=1} h_{t,n}\right] \quad (5)$$

where m_n is one for a real node and zero for padding. Two recurrent encoders produce z_g and z_q from the graph and QoS sequences, respectively. The fusion head predicts a standardized residual that is added to the latest observed QoS value:

$$\hat{y}_{t,h} = q_t + \sigma_y \odot f_\theta([z_g \,\|\, z_q]) \quad (6)$$

The complete model has 12,452 trainable parameters. Padding-invariance testing measured zero change when padded node values were perturbed, confirming that topology padding does not leak into the pooled representation. Runs were assigned to five grouped folds using repetition identifiers so that each fold contained one repetition from every topology-condition cell. For each test fold, another run-disjoint fold supported validation and epoch selection; the remaining runs supported fitting. This process was repeated for seeds 42, 52, 62, 72, and 82 at all three horizons, producing 75 HGTE models and complete out-of-fold predictions for every sample. The training objective was mean absolute error in standardized target space:

$$\mathcal{L}(\theta) = \frac{1}{4B}\sum_{b=1}^{B}\sum_{k=1}^{4}\left|\frac{y_{b,k} - \hat{y}_{b,k}}{\sigma_k}\right| \quad (7)$$

Five comparators were evaluated under the same grouped folds: persistence, the mean of the ten historical QoS observations, ridge regression, a multilayer perceptron (MLP), and a QoS-only GRU. Mean absolute error (MAE), root mean squared error (RMSE), and R^2 were computed per target. Because the four targets have different units, model ranking used standardized macro-MAE:

$$\text{MacroMAE}_{\text{std}} = \frac{1}{4}\sum_{k=1}^{4}\frac{\text{MAE}_k}{\text{SD}(y_k)} \quad (8)$$

Relative improvement over a comparator c was calculated as

$$\Delta_k(\%) = 100 \times \frac{\text{MAE}_{c,k} - \text{MAE}_{\text{HGTE},k}}{\text{MAE}_{c,k}} \quad (9)$$

For statistical inference, MAE was first calculated within each independent run, after which HGTE was paired with persistence for the same run. Ten thousand bootstrap resamples produced 95% confidence intervals for mean improvement. Two-sided Wilcoxon signed-rank tests were also applied, and the resulting p-values were corrected across the twelve horizon and target comparisons. This run-level analysis avoids treating overlapping windows as independent evidence.

Results and Discussion

Training and validation losses converged without numerical failures across all 75 grouped models. Figure 4 summarizes the fold-level training histories at the 1-, 3-, and 5-second horizons. The curves decline smoothly and remain close, while the longer-horizon traces show greater variability because fewer valid windows remain after segment-preserving construction.

Table 4 and Figure 5 compare the six predictors using seed 42. HGTE had the lowest standardized macro-MAE at 1 second (0.13393) and 3 seconds (0.15631), supporting H2 at both prespecified short horizons. At 5 seconds, GRU scored 0.17253 and HGTE scored 0.17301, a relative difference of only 0.28%. HGTE nevertheless improved on persistence at all three horizons, consistent with H1. The result is scientifically useful because graph context improved the shorter-horizon aggregate score but did not make HGTE uniformly superior to every temporal model.

Figure 6 compares the full one-second out-of-fold error distributions rather than only their means. Absolute errors are divided by the observed standard deviation of the corresponding target, allowing all four QoS measures to be read on a common scale. Curves that rise earlier and remain farther toward the upper-left indicate that a larger share of predictions has low error. The close learned-model curves confirm that HGTE's advantage is modest for latency and throughput, whereas the separation from persistence is clearer for jitter.

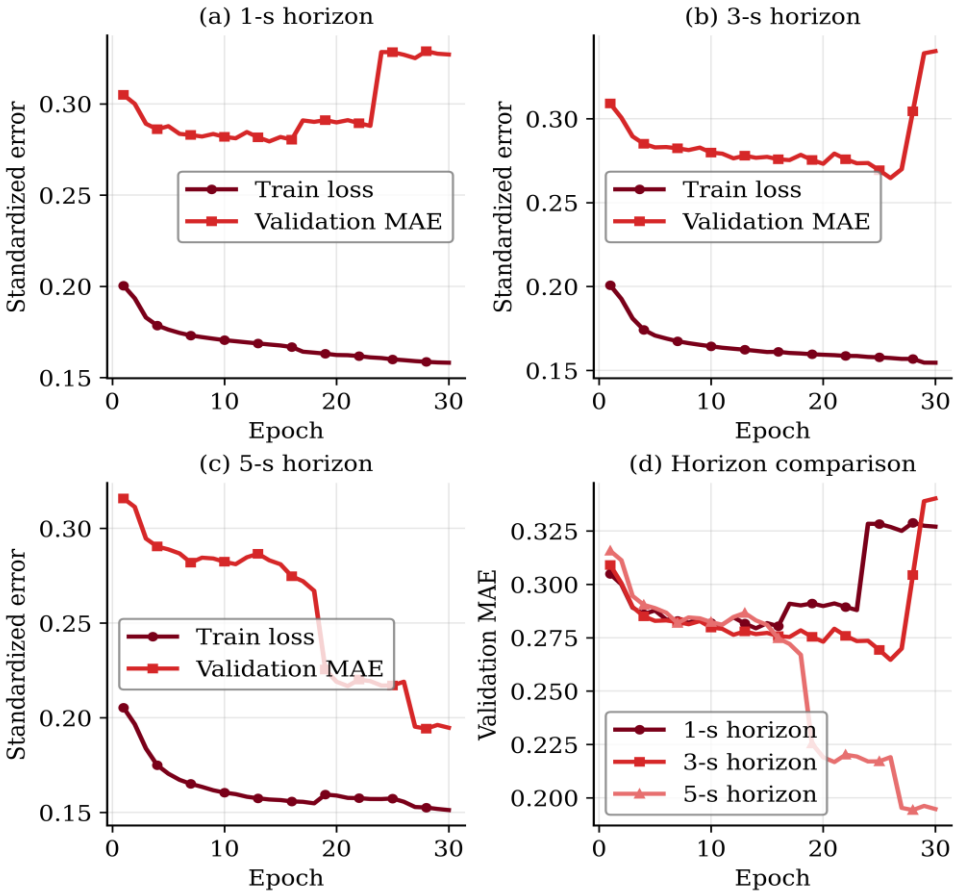


Figure 4: Multi-horizon HGTE convergence across five grouped folds

Table 4: Standardized macro-MAE by model

Model	1 s	3 s	5 s	Overall observation
Persistence	0.15424	0.18090	0.19813	Strong simple baseline
Historical mean	0.15904	0.17162	0.18007	Improves as horizon grows
Ridge	0.14921	0.18403	0.20413	Linear fit weak at longer horizons
MLP	0.13433	0.15854	0.17406	Close to HGTE
GRU	0.13501	0.15820	0.17253	Best at 5 s
HGTE	0.13393	0.15631	0.17301	Best at 1 s and 3 s

Source: Authors' grouped out-of-fold evaluation (seed 42).

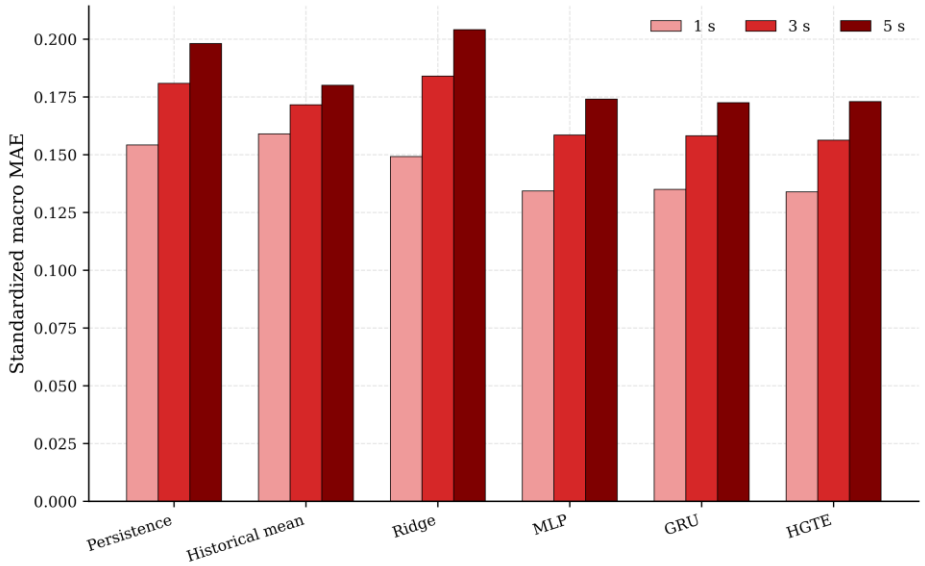


Figure 5: Scale-balanced model comparison across forecast horizons

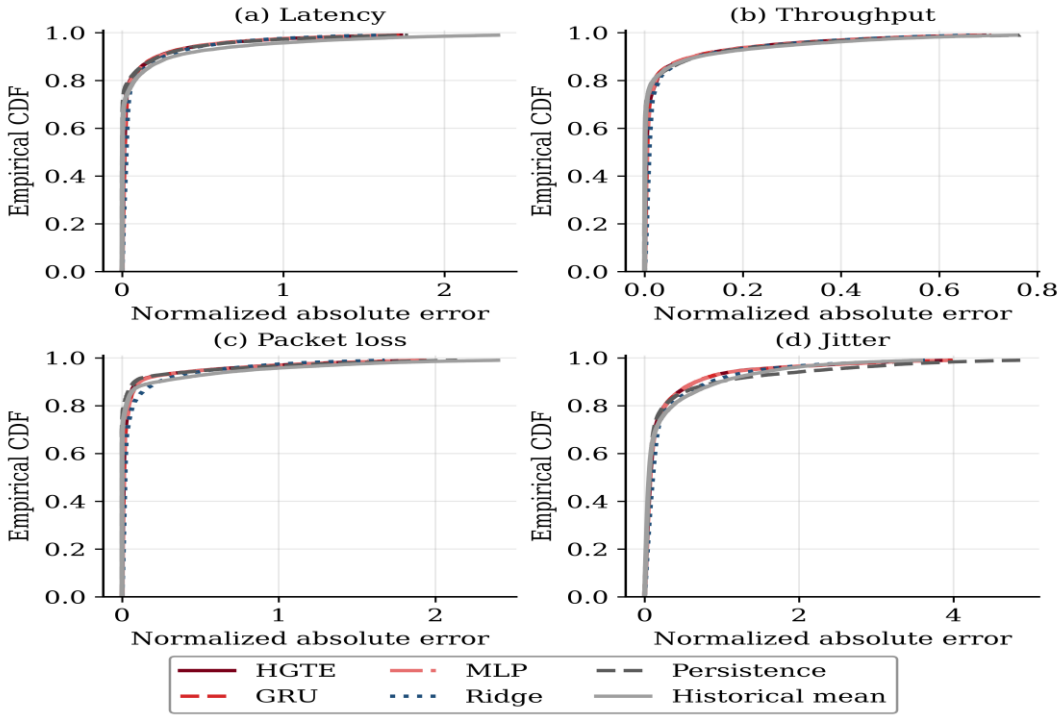


Figure 6: Empirical CDF of normalized absolute error at the 1-second horizon

The repeated-seed analysis provides the first robustness check for H1 and H3.
The repeated-seed analysis produced a stable result. Relative to persistence, the mean standardized macro-MAE improvements were 13.30% ± 0.32% at 1 second, 13.40% ± 0.38% at 3 seconds, and 13.00% ± 0.80% at 5 seconds. Figure 7 shows that all five seeds remained positive and within a narrow range. The standard deviation increased at the longest horizon, but even there the lowest seed-level improvement remained above 11%. Together with the grouped results, this supports the stability component of H3 and reduces the likelihood that the headline result is an artifact of a favorable initialization.

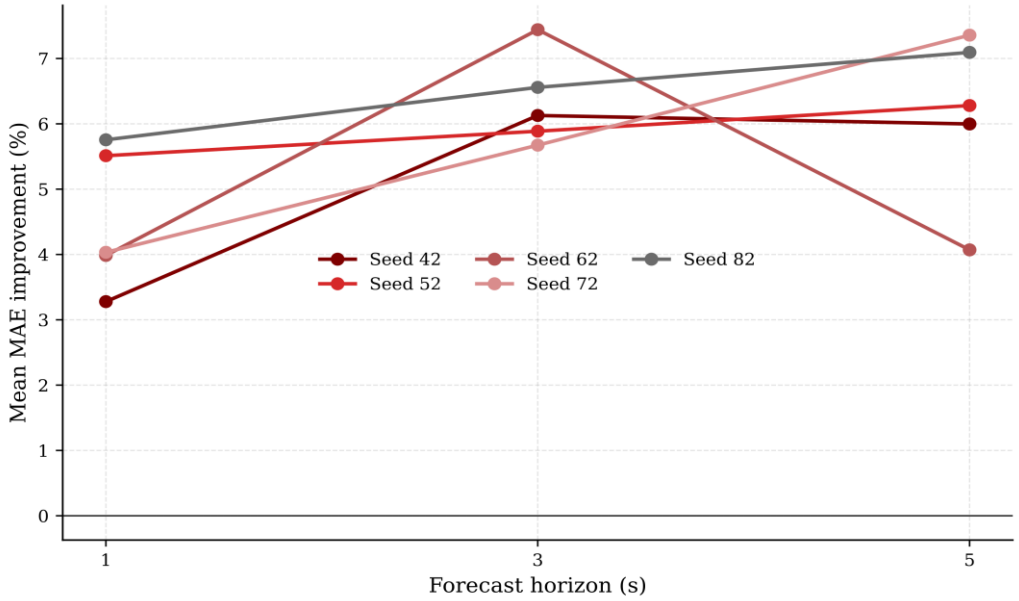


Figure 7: HGTE improvement stability across five random seeds

Target-level results show where the aggregate result originates and complete the run-level test of H3. Scale-balanced aggregation can hide important target differences, so Table 5 and Figure 8 report run-level evidence. Jitter was the clearest strength: mean MAE fell by 21.07%, 22.11%, and 21.83% at 1, 3, and 5 seconds, and every bootstrap interval remained well above zero. Holm-adjusted Wilcoxon p-values ranged from 4.39×10^-8 to 1.11×10^-11. Throughput improved by about 4% on average, and its bootstrap interval was positive at 1 and 3 seconds, but the paired rank test was not significant after correction because improvements were concentrated in larger-error runs rather than distributed uniformly across runs. Latency improvement increased with horizon, from 0.11% at 1 second to 4.55% at 5 seconds, but the confidence intervals included zero. Packet loss was essentially tied at 1 second and slightly worse at 3 and 5 seconds. The packet-loss signed-rank tests were significant in the direction of persistence for many low-loss runs, even though mean differences were small. This apparent tension occurs because percentage improvement is dominated by a few high-error runs, whereas the rank test considers the signs and ranks of paired run-level differences. Reporting both summaries prevent a misleading claim of uniform superiority.

Table 5: Run-level HGTE improvement over persistence

Horizon	Target	MAE improvement	95% bootstrap CI	Holm p	Run wins
1 s	Latency	0.11%	[-2.44, 1.95]	0.0176	28.3%
1 s	Throughput	4.33%	[1.56, 6.37]	1.0000	34.2%
1 s	Packet loss	0.04%	[-6.54, 3.91]	<0.0001	21.7%
1 s	Jitter	21.07%	[18.65, 22.95]	<0.0001	70.8%
3 s	Latency	2.71%	[-0.66, 5.27]	1.0000	35.0%
3 s	Throughput	4.00%	[1.00, 6.57]	1.0000	38.3%
3 s	Packet loss	-0.97%	[-5.96, 2.75]	0.0001	20.0%
3 s	Jitter	22.11%	[20.10, 23.68]	<0.0001	76.7%
5 s	Latency	4.55%	[-1.80, 9.79]	1.0000	35.8%
5 s	Throughput	4.10%	[-0.19, 7.58]	1.0000	37.5%
5 s	Packet loss	-0.62%	[-7.06, 4.26]	0.0003	20.8%
5 s	Jitter	21.83%	[19.32, 23.69]	<0.0001	68.3%

Source: *Authors' repeated-seed run-level statistical analysis.*

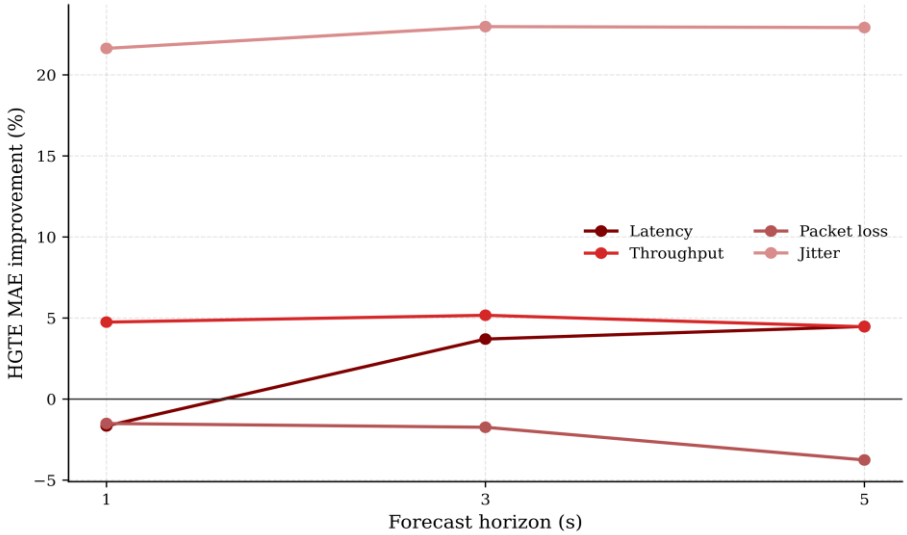


Figure 8: Target-level MAE improvement over persistence

The aggregate improvement in Figure 8 is driven by different target scales and dynamics. Figure 9 therefore shows the absolute jitter MAE for every comparator. Learned temporal models reduce jitter error at all horizons, and HGTE remains competitive with the MLP and GRU even where their macro scores are nearly tied. Figure 10 provides a complementary prediction trace for one-second latency. HGTE follows the broad movement of the observed series but smooths several sharp peaks. Those missed peaks explain why average latency gains are modest despite good aggregate R-squared values, and they identify transient events as a priority for future loss functions.

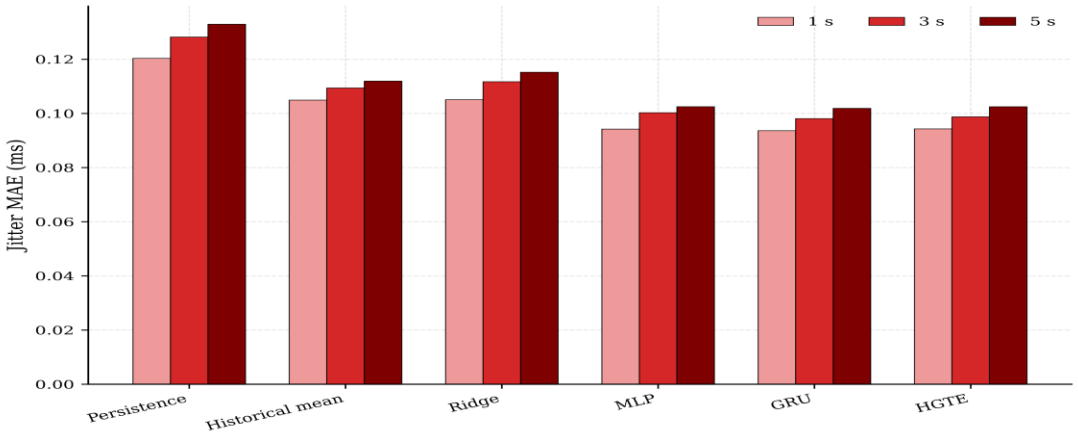


Figure 9: Jitter MAE comparison by model and forecast horizon

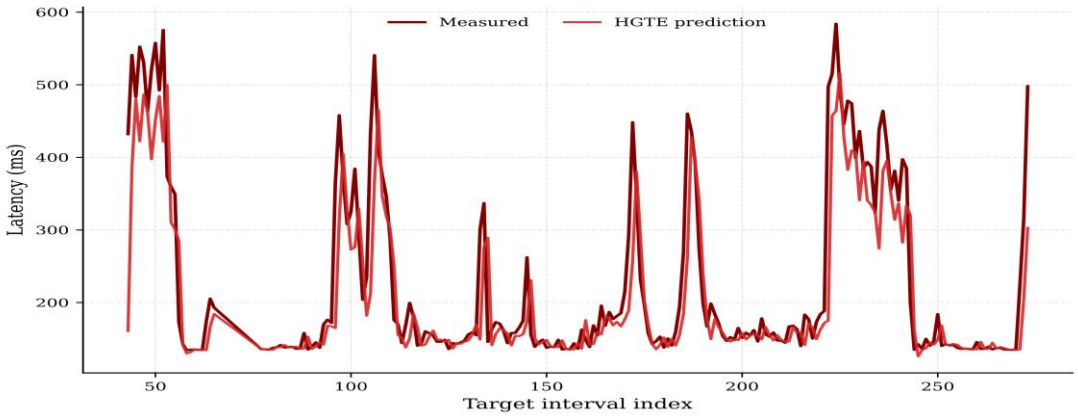


Figure 10: Observed and predicted latency trace at the 1-second horizon

A grouped fold tests new runs but still exposes the model to all topology and condition categories during training. Leave-one-topology-out experiments were therefore added to test H4. Figure 11 compares all three held-out topologies at 1, 3, and 5 seconds. Transfer is target- and horizon-dependent: useful gains remain for several topology-target pairs, but negative bars show that structural novelty can make persistence safer. The greater variability relative to grouped evaluation supports H4 and limits the claim to cautious topology transfer rather than universal out-of-distribution generalization.

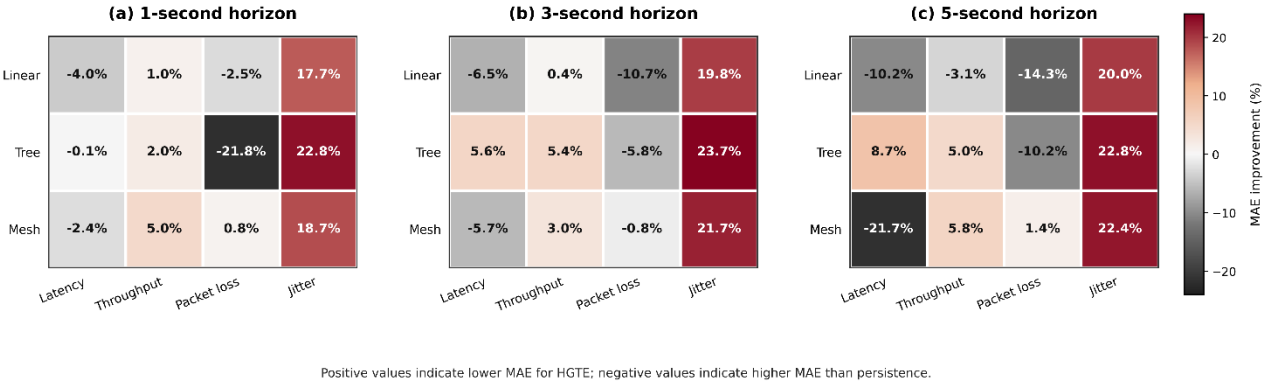


Figure 11: Leave-one-topology-out improvement across forecast horizons

Leave-one-condition-out tests were even more demanding because a withheld operating regime can change both the target scale and the frequency of events. The heat map in Figure 12 shows that the model's behavior varied sharply by condition and target. Extreme overload and zero-loss

or near-zero-loss regimes exposed a weakness of unconstrained residual prediction: a small nonzero error can look arbitrarily poor when persistence is nearly perfect. The result argues for condition-aware calibration and a fallback policy, rather than for deploying a learned correction in every state.

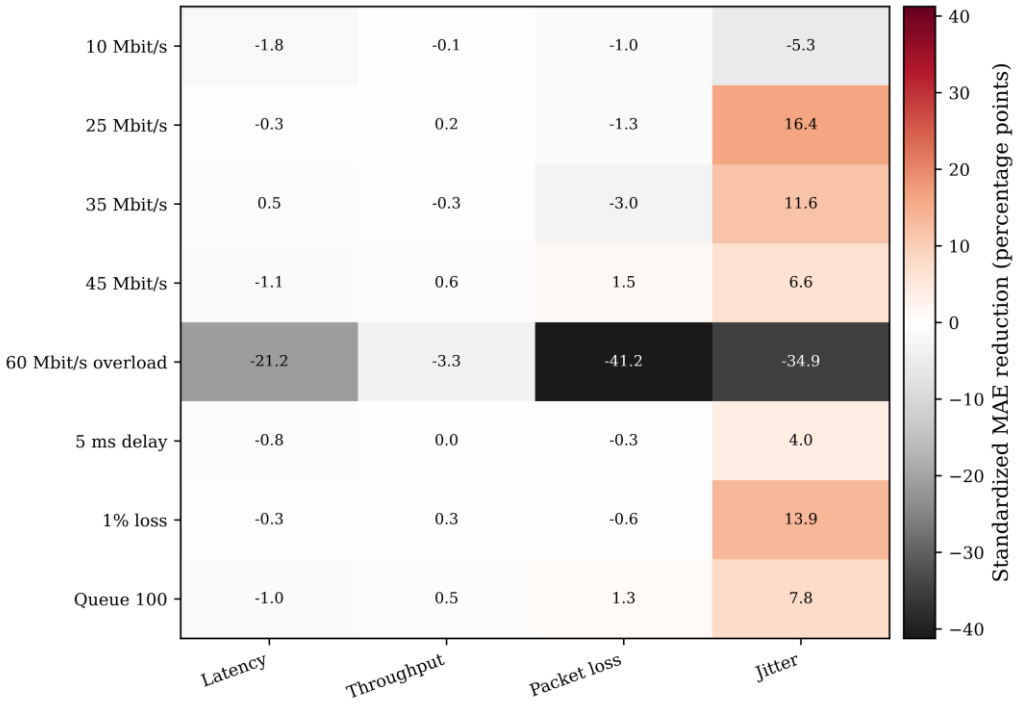


Figure 12: HGTE behavior when one network condition is unseen

A validation-calibrated Safe-HGTE pilot was tested for three one-second holdouts: low load, overload, and configured loss. The gate chose between HGTE and persistence using validation evidence rather than the unseen test labels. It reduced selected failures, but the pilot is too narrow to establish universal safety. It is therefore treated as an engineering direction rather than a completed deployment guarantee. The selected grouped model was exported from PyTorch to ONNX FP32 and dynamically quantized to INT8. Equivalence tests covered all three topology masks. Table 6 and Figure 13 summarize latency, inference throughput, artifact size, and numerical deviation. ONNX FP32 reduced median CPU latency by 82.63%, reached the highest inference throughput, remained close to 60 kB, and differed from PyTorch by at most 1.53×10^{-5}. Dynamic INT8 was rejected because it was 13.89% slower than ONNX FP32, 1.63% larger, and produced a maximum absolute deviation of 3.287. For this compact recurrent model, quantization overhead and drift outweighed its expected benefit.

Table 6: CPU deployment comparison

Format	Median latency	Maximum p95	Throughput	Artifact
PyTorch FP32	1.3295 ms	1.660 ms	752/s	60,708 B
ONNX FP32	0.2309 ms	0.329 ms	4,445/s	60,162 B
ONNX dynamic INT8	0.2630 ms	0.536 ms	3,858/s	61,140 B

Source: *Authors' CPU deployment benchmark.*

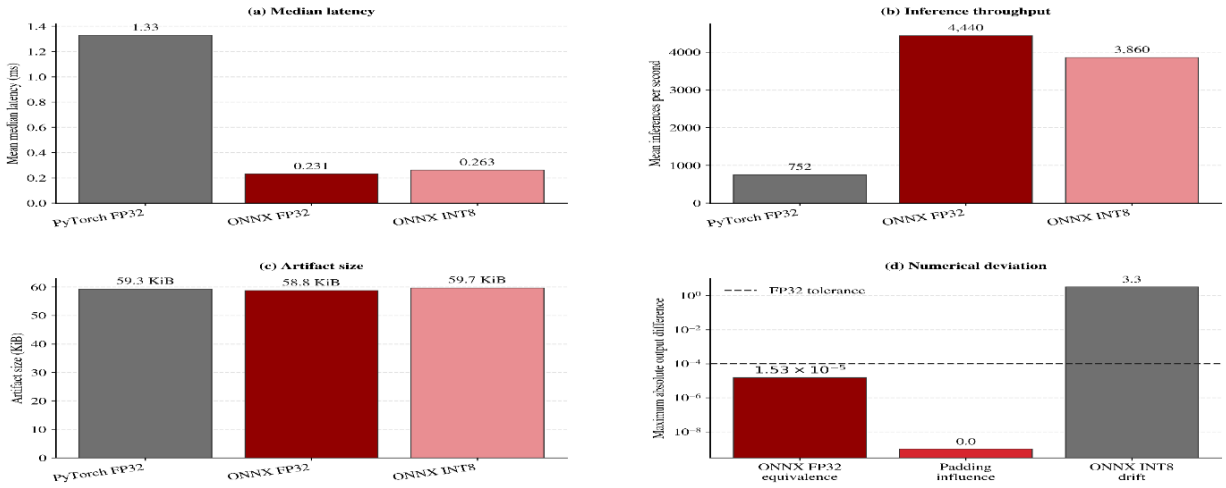


Figure 13: CPU deployment performance, size, and numerical fidelity

The fallback pilot extends the deployment analysis from speed to decision safety.
The unseen-condition results suggested that a learned correction should not be applied blindly. A validation-calibrated Safe-HGTE pilot was therefore examined to select either HGTE or persistence before the unseen test labels were revealed. Figure 14 compares the pilot outcomes for the low-load, overload, and configured-loss holdouts. The gate protected some cases in which HGTE extrapolated poorly, but it did not remove every failure. This evidence supports a fallback mechanism as part of the system design, while also showing that the pilot is not yet a general uncertainty estimator.

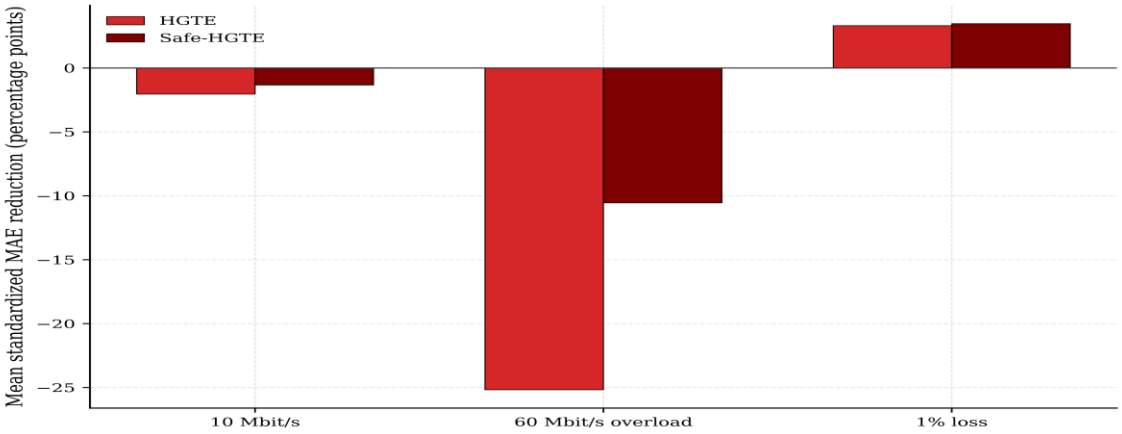


Figure 14: Validation-calibrated Safe-HGTE pilot comparison

A second trace illustrates the practical difficulty at the longer horizon. Figure 15 compares measured and predicted throughput five seconds ahead. The model follows the normal operating level closely, but rapid changes remain harder to reproduce. This behavior is consistent with the higher macro-MAE at five seconds and with the narrow performance difference between HGTE and the QoS-only GRU.

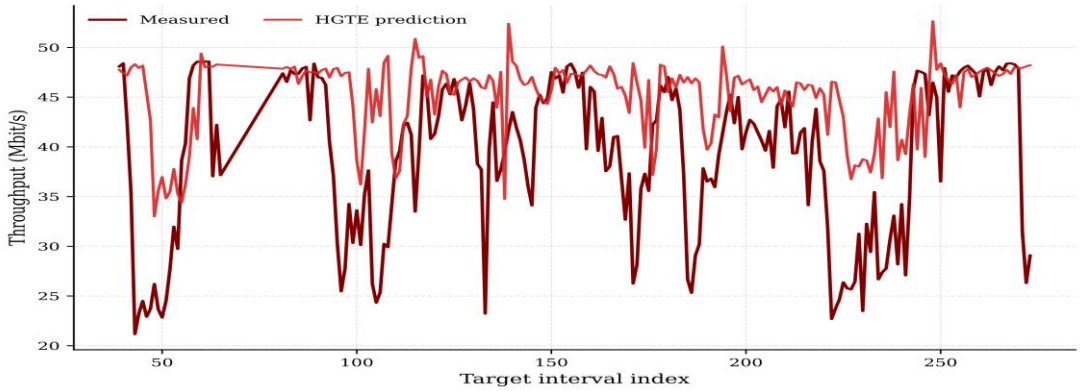


Figure 15: Observed and predicted throughput at the 5-second horizon

Hypothesis Results

H1 (Persistence anchor). HGTE achieved lower standardized macro-MAE than persistence at all three horizons (Table 4), with repeated-seed improvements of 13.30% ± 0.32%, 13.40% ± 0.38%, and 13.00% ± 0.80% at 1, 3, and 5 seconds. All five seeds remained positive, and the lowest seed-level gain exceeded 11% at the longest horizon. Supported.

H2 (Short-horizon graph advantage). HGTE ranked first among six predictors at 1 second (0.13393) and 3 seconds (0.15631), beating ridge, the MLP, and the QoS-only GRU at both horizons. The margins were narrow, however 0.30% over the MLP at 1 second and 1.19% over the GRU at 3 seconds and no run-level significance test against these comparators is reported. At 5 seconds the GRU led by 0.28%, consistent with the expectation that topology becomes less informative over longer horizons. Supported directionally, with narrow margin.

H3a (Stability). All five seeds showed positive improvement at every horizon within a narrow range (Figure 7). Supported.

H3b (Target-level reliability). Jitter improved reliably at every horizon, with MAE reductions of 21.07%, 22.11%, and 21.83% and Holm-adjusted p-values below 10^{-7} (Table 5). Throughput improved by about 4% but failed the corrected rank test; latency confidence intervals included zero; packet loss was tied at 1 second and slightly worse at 3 and 5 seconds. Supported, but carried by jitter alone.

H4 (Bounded transfer). Leave-one-topology-out and leave-one-condition-out results showed greater dispersion than grouped evaluation, with target- and horizon-dependent gains and clear failures in extreme-overload and near-zero-loss regimes (Figures 11-12). Supported as a variability claim.

Table 7: Hypothesis outcomes

Hypothesis	Prediction	Key evidence	Decision	Strength
H1	HGTE < persistence at 1, 3, and 5 s	13.00–13.40% macro-MAE gain; all seeds positive	Supported	Strong
H2	HGTE < ridge, MLP, and GRU at 1 and 3 s	Ranked first at 1 and 3 s; margins of 0.30% and 1.19%	Supported	Directional; narrow
H3a	Positive improvement across all five seeds	All seeds positive; lowest improvement >11% at 5 s	Supported	Strong
H3b	At least one target reliably improved per horizon	Jitter: 21.07–22.11%; Holm-adjusted $p < 10^{-7}$	Supported	Narrow; jitter-driven
H4	Greater variability under holdouts	LOTO and LOCO dispersion exceeded grouped-fold dispersion	Supported	Bounded

Note: *No hypothesis is rejected, but support is uneven. H1 and H3a are robust; H2 rests on small margins without comparator-level significance tests; H3b depends on jitter; and H4 bounds rather than extends the validated range. The aggregate gain should therefore not be read as uniform target-level superiority, and the narrow H2 margin, together with the GRU's lead at 5 seconds, points to the branch ablation identified in Future Directions as the next test.*

Applications and Implications

Theoretical application. The results clarify the role of topology in short-horizon QoS forecasting. Tables 4 and 5 and Figures 5-9 show that a graph-temporal residual design can improve a persistence anchor without requiring a large model, but the benefit is target- and horizon-dependent. The model's first-place ranking at 1 and 3 seconds supports relational-temporal fusion for near-term prediction, whereas the GRU's narrow lead at 5 seconds shows that topology is not automatically beneficial at every horizon. Figures 11 and 12 further establish that structural inductive bias does not guarantee transfer to every unseen topology or condition. This finding refines the broader expectation of graph generalization described by Suárez-Varela et al. (2023) and identifies validated operating range as part of the theoretical claim. Managerial and operational application. LiGHT-QoS can be placed beside an SDN controller as an advisory forecasting service. The controller already collects port statistics and understands topology; the model converts a recent ten-second window into four short-horizon forecasts. Table 5 indicates that jitter is the most dependable target for an early-warning policy, while latency and packet loss require more conservative thresholds. Table 6 and Figure 13 show that ONNX FP32 inference is fast enough not to dominate a one-second monitoring loop in the tested environment. An operator could therefore use the output to request a route review, queue inspection, or admission check, while retaining persistence as a fallback when input support is weak. Governance should follow the evidence boundary. Figures 11, 12, and 14 show why a learned correction should not be applied automatically in every state. Target-specific confidence, input-range monitoring, validation-calibrated fallback, action logging, and human approval are appropriate until field tests establish safe thresholds. The balanced design also has practical value: five repetitions per topology-condition cell separate repeatable effects from isolated runs, and segment-preserving preprocessing keeps timing gaps visible. These choices reduce the headline sample count but make the evidence more useful for engineering decisions.

Limitations

The primary limitation concerns cross-environment generalizability. Mininet and Open vSwitch reproduce important packet, queue, and controller behaviors, but they do not capture every property of hardware switch, network interface cards, cross-traffic, routing policy, or geographically distributed links. The three topologies contain four switches and four hosts; larger and more heterogeneous networks may create graph structures and path interactions outside the present training distribution. This limitation is important because graph-based network models are expected to transfer structurally, yet recent literature also identifies distribution shift and interpretability as unresolved barriers to operational use (Suárez-Varela et al., 2023). The deployment benchmark was performed on a WSL2 CPU, not on a production controller appliance. A second limitation concerns condition diversity. The matrix is balanced and substantially broader than a single-scenario experiment, yet it still uses configured loads, delays, losses, and queue sizes. Real networks combine these factors, experience failures, and exhibit application-dependent traffic bursts. The leave-one-condition-out results show that interpolation among sampled conditions is easier than extrapolation to a truly unseen regime. Future work should therefore add combined impairments, time-varying routing, background flows, link failures, and hardware-in-the-loop traces. A third limitation is metric interpretation. MAE is easy to explain, but a percentage improvement becomes unstable when the comparator error is close to zero. This issue was addressed by reporting raw MAE, scale-balanced macro-MAE, run-level confidence intervals, signed-rank tests, and win rates. Even so, no single statistic captures operational risk. Tail errors and decision-aware costs should be added before closed-loop control. Finally, the Safe-HGTE fallback was evaluated only for three one-second condition holdouts. It demonstrates the feasibility of validation-calibrated selection, not a complete out-of-distribution detector. The pilot cannot yet tell an operator how confident to be in every prediction, and it has not been tested under combined failures or unseen hardware. These boundaries are important because a fallback that works for one isolated impairment may still fail when several conditions change together.

Future Directions

The next stage should move from laptop emulation to hardware-in-the-loop validation. A practical experiment would connect the controller to physical Open vSwitch hosts or programmable switches and replay mixed traffic while collecting the same port and end-to-end measurements. This would show whether operating-system scheduling, network interface buffering, hardware counters, and controller communication delays change the forecasting results. Tests should include larger topologies, multiple simultaneous paths, rerouting events, link failures, and background traffic that was not generated by the primary flow. Future modeling work should focus on uncertainty rather than adding depth without a clear benefit. Repeated runs can support calibrated prediction intervals, while an input-support monitor can detect when utilization, loss, delay, or topology features move outside the training envelope. The model should then return either a forecast with a confidence range or a persistence fallback. The threshold must be selected on validation runs and assessed with decision costs, not with test labels. A further experiment should separate the value of graph telemetry from the value of recent QoS history. The present baseline comparison shows that a QoS-only GRU is highly competitive at five seconds. A formal ablation study can remove the graph branch, remove the QoS branch, replace learned adjacency with identity connections, and test different window lengths. This will reveal which component helps each target and whether the graph branch becomes more valuable on larger or changing topologies. Finally, the forecasting service should be evaluated inside a cautious control loop. Initial trials should generate recommendations without changing the network. Operators can then compare predicted and observed outcomes and define safe action limits. Only after this shadow-mode phase should the system be allowed to request route changes, queue adjustments, or admission control. Every automatic action should include logging, rollback, and a policy that returns control to the operator when uncertainty is high.

Conclusion

LiGHT-QoS was developed and evaluated as a 12,452-parameter graph-temporal model for short-horizon SDN QoS forecasting. The study combined port-level topology with recent end-to-end history and used a residual output anchored to the latest observation. Across 120 independent runs, five grouped folds, and five random seeds, HGTE improved standardized macro-MAE over persistence by 13.30%, 13.40%, and 13.00% at one, three, and five seconds. It ranked first among six predictors at the two shorter horizons and narrowly second to a QoS-only GRU at five seconds. These findings support H1, H2, and the stability component of H3 while showing that the value of graph context depends on forecast horizon. The strongest target-level evidence was for jitter, where run-level MAE fell by about 21%-22% and the paired evidence remained statistically reliable at every horizon, completing the support for H3. Throughput improved modestly on average, while latency and packet loss remained condition-dependent. Tests on unseen topologies and unseen conditions produced both useful transfer and clear failure modes, supporting H4. The theoretical contribution is therefore a bounded one: explicit port topology can improve a persistence-anchored temporal forecast, but structural information does not remove the need to test distribution shift. This conclusion complements recent graph-based QoS research while narrowing its application to the conditions actually measured (Liu et al., 2024; Hu et al., 2025). These conclusions remain limited by the four-switch Mininet/Open vSwitch environment, configured rather than naturally occurring impairments, and the narrow Safe-HGTE pilot. The model should therefore remain advisory until hardware-in-the-loop and broader distribution-shift tests have been completed (Suárez-Varela et al., 2023; Prasanth & Uma, 2024). ONNX FP32 preserved numerical equivalence and reduced median CPU latency by 82.63%, whereas dynamic INT8 was slower, larger, and less accurate. For SDN operators, the immediate application is an advisory forecasting service with target-specific thresholds, input-support monitoring, and a persistence fallback; the evidence does not yet justify autonomous control in broad real networks. For researchers, the grouped run-level protocol shows why overlapping temporal windows should not be treated as independent observations and why transfer claims should include unseen structures and conditions. Hardware-in-the-loop trials, larger graphs, combined impairments, calibrated prediction intervals, formal branch ablations, and shadow-

mode control studies are therefore the specific next steps (Prasanth & Uma, 2024; Suárez-Varela et al., 2023).

References

Abadal, S., Jain, A., Guirado, R., López-Alonso, J., & Alarcón, E. (2022). Computing graph neural networks: A survey from algorithms to accelerators. ACM Computing Surveys, 54(9), Article 191, 1–38. https://doi.org/10.1145/3477141

Ak, E., & Canberk, B. (2021). Forecasting quality of service for next-generation data-driven WiFi6 campus networks. IEEE Transactions on Network and Service Management, 18(4), 4744–4755. https://doi.org/10.1109/TNSM.2021.3108766

Chai, A., Li, M., Yang, H., & Guo, C. (2024). EMD-EmLSTM: A QoS analysis and prediction method for Industrial Internet of Things. IEEE Internet of Things Journal, 11(20), 32730–32744. https://doi.org/10.1109/JIOT.2024.3375855

Ferriol-Galmés, M., Paillisse, J., Suárez-Varela, J., Rusek, K., Xiao, S., Shi, X., Cheng, X., Barlet-Ros, P., & Cabellos-Aparicio, A. (2023). RouteNet-Fermi: Network modeling with graph neural networks. IEEE/ACM Transactions on Networking, 31(6), 3080–3095. https://doi.org/10.1109/TNET.2023.3269983

Ferriol-Galmés, M., Rusek, K., Suárez-Varela, J., Xiao, S., Cheng, X., Barlet-Ros, P., & Cabellos-Aparicio, A. (2022). RouteNet-Erlang: A graph neural network for network performance evaluation. In IEEE INFOCOM 2022—IEEE Conference on Computer Communications. https://doi.org/10.1109/INFOCOM48880.2022.9796944

Hu, S., Zou, G., Zhang, B., Wu, S., Lin, S., Gan, Y., & Chen, Y. (2025). GACL: Graph attention collaborative learning for temporal QoS prediction. IEEE Transactions on Network and Service Management, 22(4), 3388–3402. https://doi.org/10.1109/TNSM.2025.3570464

Huang, H., Wang, Q., Min, G., Wang, M., & Wu, D. O. (2025). QoS prediction for component services in 5G via graph-based deep reinforcement learning. IEEE Transactions on Mobile Computing, 24(12), 13520–13534. https://doi.org/10.1109/TMC.2025.3591783

Jiang, W. (2022). Graph-based deep learning for communication networks: A survey. Computer Communications, 185, 40–54. https://doi.org/10.1016/j.comcom.2021.12.015

Keshari, S. K., Kansal, V., & Kumar, S. (2021). A systematic review of quality of services (QoS) in software defined networking (SDN). Wireless Personal Communications, 116(3), 2593–2614. https://doi.org/10.1007/s11277-020-07812-2

Li, J., Wu, H., He, Q., Zhao, Y., & Wang, X. (2024). Dynamic QoS prediction with intelligent route estimation via inverse reinforcement learning. IEEE Transactions on Services Computing, 17(2), 509–523. https://doi.org/10.1109/TSC.2023.3342481

Liu, M., Xu, H., Sheng, Q. Z., & Wang, Z. (2024). QoSGNN: Boosting QoS prediction performance with graph neural networks. IEEE Transactions on Services Computing, 17(2), 645–658. https://doi.org/10.1109/TSC.2023.3343351

Prasanth, L. L., & Uma, E. (2024). A computationally intelligent framework for traffic engineering and congestion management in software-defined network (SDN). EURASIP Journal on Wireless Communications and Networking, 2024, Article 63. https://doi.org/10.1186/s13638-024-02392-2

Rusek, K., Suárez-Varela, J., Almasan, P., Barlet-Ros, P., & Cabellos-Aparicio, A. (2020). RouteNet: Leveraging graph neural networks for network modeling and optimization in SDN. IEEE Journal on Selected Areas in Communications, 38(10), 2260–2270. https://doi.org/10.1109/JSAC.2020.3000405

Suárez-Varela, J., Almasan, P., Ferriol-Galmés, M., Rusek, K., Geyer, F., Cheng, X., Shi, X., Xiao, S., Scarselli, F., Cabellos-Aparicio, A., & Barlet-Ros, P. (2023). Graph neural networks for communication networks: Context, use cases and opportunities. IEEE Network, 37(3), 146–153. https://doi.org/10.1109/MNET.123.2100773

Waikhom, L., & Patgiri, R. (2023). A survey of graph neural networks in various learning paradigms: Methods, applications, and challenges. Artificial Intelligence Review, 56(7), 6295–6364. https://doi.org/10.1007/s10462-022-10321-2

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

Zou, G., Yu, W., Hu, S., Gan, Y., Zhang, B., & Chen, Y. (2024). FRLN: Federated residual ladder network for data-protected QoS prediction. IEEE Transactions on Services Computing, 17(3), 963–976. https://doi.org/10.1109/TSC.2024.3377100

Published by

IJSAB International