Adpt-STGIN: An Adaptive Spatio-Temporal Graph Inductive Network for Topology-Robust Traffic Prediction in Data Center Networks

Xuran Chen[1], Jie Hao[1(✉)], Ran Wang[1], Qiang Wu[1] and Yimeng Gao[1]

[1] School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210000, China
haojie@nuaa.edu.cn

Abstract. Accurate network traffic prediction is essential for resource management and congestion control in data center networks. Existing spatio-temporal graph neural network (STGNN) models predominantly employ transductive spatial encoders, such as GCN or GAT, whose parameters are tied to a fixed graph structure, preventing generalization to unseen topologies without full retraining. In this paper, we propose Adpt-STGIN (Adaptive Spatio-Temporal Graph Inductive Network), a topology-robust traffic prediction framework built on two key contributions. First, we design a deeply fused GraphSAGE-GRU cell that embeds independent inductive GraphSAGE(SAmple and aggreGatE) encoders directly into each GRU gate, enabling simultaneous spatio-temporal feature extraction at every time step while remaining topology-agnostic. Second, we develop a topology-robust transfer learning framework with a frozen encoder strategy that adapts pretrained models to new topologies by fine-tuning only the lightweight decoder. Experiments on four data center topologies demonstrate that Adpt-STGIN achieves $R^2 > 0.99$ in pretraining and generalizes to unseen topologies in zero-shot mode with $R^2 > 0.994$, confirming the practical efficiency of the proposed framework.

Keywords: Network Traffic Prediction, Spatio-temporal Graph Neural Network, Transfer Learning.

1 Introduction

The rapid growth of large-scale data centers has made accurate network traffic prediction a critical enabler for resource management, congestion control, and quality-of-service assurance. In modern data centers, topology changes driven by scale-out expansion, scale-in pruning, and workload-induced paradigm shifts are routine, imposing stringent requirements on the adaptability of prediction models. Early approaches relied on classical time series models such as ARIMA [1] and SVR [2], which are limited to linear temporal dependencies. Recurrent architectures such as LSTM [3] and GRU [4] substantially improved non-linear temporal modeling, but treat each node's traffic independently, ignoring the spatial correlations induced by the network topology.

To address the spatial dimension, spatio-temporal (ST) models combining GNNs with recurrent modules, including GCN [5], GAT [6] and DCRNN [7], have demonstrated strong performance on fixed-topology benchmarks. However, these models are fundamentally transductive: their parameters are tied to a fixed adjacency matrix or node-specific embeddings, making them incapable of generalizing to unseen topologies without full retraining, rendering the fixed-graph assumption untenable in practice.

A key observation motivating this work is that inductive spatial encoders have been largely overlooked in ST traffic prediction. GraphSAGE [8] learns aggregation functions over local neighborhoods rather than node-specific embeddings, enabling a model trained on one topology to be directly applied to an entirely different topology without architectural modification. Nevertheless, GraphSAGE alone addresses only the spatial dimension and lacks temporal modeling capability, necessitating its integration with a recurrent architecture to capture the time-varying dynamics of traffic sequences.

In this paper, we propose Adpt-STGIN, a topology-robust traffic prediction framework built on a deeply fused GraphSAGE-GRU architecture and a parameter-efficient transfer learning strategy.

The main contributions of this paper are as follows:

- We propose a deeply fused GraphSAGE-GRU architecture that integrates inductive spatial aggregation directly into GRU gating, enabling topology-agnostic ST feature learning.
- We design a topology-robust transfer learning framework with a frozen encoder strategy that adapts pretrained models to new topologies by fine-tuning only the decoder, achieving competitive performance with minimal data and training cost.
- We construct a dedicated dataset[1] using the ns3-rdma simulator [9], covering four representative data center topologies with realistic Remote Direct Memory Access(RDMA) traffic traces under the Data Center Quantized Congestion Notification(DCQCN) protocol. Despite the importance of traffic prediction in modern data centers, this area remains relatively less explored and lacks sufficient publicly available benchmarks. In this work, the new dataset we present supports more comprehensive experimental studies, including the evaluation of the proposed framework's zero-shot generalization capability.
- We establish a comprehensive evaluation protocol for topology-robust traffic prediction. The results reveal that zero-shot generalization performance is near-optimal for topologies sharing similar architectural patterns, while cross-architecture transfers benefit consistently from lightweight decoder fine-tuning, providing empirical insights into the relationship between topology similarity and transfer effectiveness.

[1] The dataset we proposed is available at https://doi.org/10.5281/zenodo.19865728

2 Related Work

Various strategies for predicting network traffic are divided into three primary categories.

2.1 Classical and Machine Learning Models

Early statistical methods such as ARIMA [1] offer structural simplicity but are confined to linear temporal dependencies and require stationary inputs. Machine learning techniques, including SVR [2] and Bayesian network models [10], extended prediction to non-linear settings by exploiting complex patterns in historical data, yielding more robust results than purely statistical approaches.

2.2 Deep Learning-Based Sequential Models

The advent of deep learning introduced models capable of automatically extracting features from large-scale traffic data. Zhang et al. [11] proposed an adaptive attention mechanism (AAM) to reduce computational complexity while minimizing information loss. Despite their temporal modeling strengths, these approaches treat each node's traffic as an independent time series, ignoring the spatial correlations induced by the underlying network topology.

2.3 STGNN Models

GNNs provide a principled framework for jointly modeling spatial and temporal dynamics in network traffic. Jin et al. [12] integrated spatial attention with GNNs and temporal attention to fuse ST features, constructing adaptive adjacency matrices from node similarity. Huang et al. [13] introduced a source-destination attention mechanism to capture OD-pair spatial dependencies, paired with periodic sparse self-attention for long-range temporal modeling. Sun et al. [14] proposed a framework integrating Graphormer with sequential temporal models, employing dynamic adjacency generation and centrality encoding for effective ST feature extraction. However, these models assume a fixed graph structure and do not address generalization to unseen topologies, limiting their applicability in data center networks where large-scale topology changes are routine.

3 Methodology

We present the proposed Adpt-STGIN framework for network traffic prediction in data center networks. We first formulate the problem and then detail the model architecture, including the GraphSAGE spatial encoder, the deeply fused GraphSAGE-GRU cell, the temporal attention mechanism, and the topology-robust transfer learning strategies.

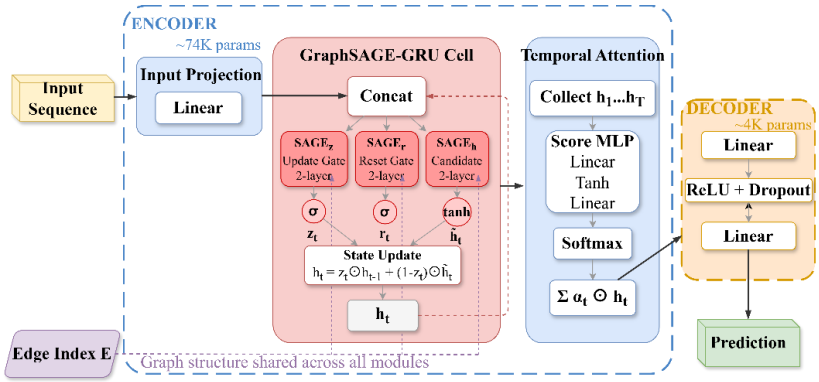

Fig. 1. Detailed architecture of the Adpt-STGIN.

3.1 Problem Formulation

We model a data center network as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V}$ denotes the set of N nodes (switches and hosts), $\mathcal{E}$ denotes the set of edges (physical links), and $\mathbf{A} \in \mathbb{R}^{N\times N}$ is the adjacency matrix. Each edge $e_{ij} \in \mathcal{E}$ carries attributes including bandwidth, propagation delay, and packet error rate. We denote by $\mathcal{N}(v)$ the set of neighbors of node v in $\mathcal{G}$.

At each time step t (corresponding to a 1 ms time bin), we define the node-level traffic feature $x_i^{(t)}$ as the total outgoing packet count of node i, aggregated over all its outgoing links and normalized via per-node z-score standardization. This yields the feature vector $\mathbf{x}_t \in \mathbb{R}^{N\times d_{\text{in}}}$ with $d_{\text{in}} = 1$. Given a historical observation window of T time steps, the input traffic sequence is denoted as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T] \in \mathbb{R}^{T\times N\times d_{\text{in}}}$.

Given the graph structure $\mathcal{G}$ and the historical traffic sequence $\mathbf{X}$, the goal is to learn a mapping functionf_θ that predicts the traffic at the next time step:

$$\hat{\mathbf{x}}_{T+1} = f_\theta(\mathbf{X}, \mathcal{G}), \tag{1}$$

where $\hat{\mathbf{x}}_{T+1} \in \mathbb{R}^{N\times d_{\text{out}}}$ and θ denotes the model parameters.

Considering that topology changes driven by scale-out expansion, scale-in pruning, and workload-induced paradigm shifts are routine, we aim to learn f_θ that generalizes across diverse topologies. Let $\mathcal{G}^{(s)} = (\mathcal{V}^{(s)}, \mathcal{E}^{(s)})$ and $\mathcal{G}^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$ denote the source and target topologies, respectively, where $|\mathcal{V}^{(s)}| \neq |\mathcal{V}^{(t)}|$ in general. The objective is to transfer a model f_θ pretrained on $\mathcal{G}^{(s)}$ to $\mathcal{G}^{(t)}$ with minimal fine-tuning data from the target domain.

3.2 GraphSAGE-GRU Fusion Architecture

The core of Adpt-STGIN is a deeply fused ST architecture that integrates GraphSAGE neighborhood aggregation directly into the GRU gating mechanism. Unlike sequential approaches that apply spatial and temporal modules independently, our design enables

simultaneous ST feature extraction at every time step. The model architecture is shown in Fig. 1.

Overview. We employ a multi-layer GraphSAGE encoder that learns node representations through iterative neighborhood aggregation. For a node v at layer l, the aggregation is defined as:

$$\mathbf{h}_{\mathcal{N}(v)}^{(l)} = \mathrm{AGG}\left(\left\{\mathbf{h}_u^{(l-1)}, \forall u \in \mathcal{N}(v)\right\}\right), \tag{2}$$

$$\mathbf{h}_v^{(l)} = \mathrm{LayerNorm}\left(\mathbf{W}^{(l)} \cdot \mathrm{CONCAT}\left(\mathbf{h}_v^{(l-1)}, \mathbf{h}_{\mathcal{N}(v)}^{(l)}\right)\right), \tag{3}$$

where $\mathrm{AGG}(\cdot)$ is the mean aggregation function, $\mathbf{W}^{(l)} \in \mathbb{R}^{d_h \times 2d_h}$ is the learnable weight matrix at layer l, and LayerNorm denotes layer normalization. ReLU activation and dropout ($p = 0.1$) are applied between layers.

The weight matrices $\mathbf{W}^{(l)}$ depend only on the hidden dimension d_h, not on the number of nodes N. This makes the encoder inherently topology-agnostic: the same parameters can be applied to graphs of arbitrary size and structure, as the neighborhood structure is fully specified by the edge index $\mathbf{E} \in \mathbb{Z}^{2\times|\mathcal{E}|}$, a sparse representation of the adjacency derived from $\mathcal{G}$.

We use a 2-layer GraphSAGE encoder with hidden dimension $d_h = 64$. The first layer maps from d_{in} to d_h dimensions, while the second layer operates at d_h dimensions.

GraphSAGE-GRU Cell. The key innovation of Adpt-STGIN lies in the deep fusion of GraphSAGE spatial aggregation with the GRU temporal gating mechanism. Rather than applying spatial and temporal operations sequentially, we embed independent GraphSAGE encoders into each GRU gate, enabling spatial information to directly influence the temporal state transitions.

At each time step t, given the projected input $\mathbf{x}_t \in \mathbb{R}^{N\times d_h}$ and the previous hidden state$\mathbf{h}_{t-1} \in \mathbb{R}^{N\times d_h}$, the GraphSAGE-GRU cell computes:

$$\mathbf{z}_t = \sigma\left(\mathrm{SAGE}_z([\mathbf{x}_t \| \mathbf{h}_{t-1}],\ \mathbf{E})\right), \tag{4}$$

$$\mathbf{r}_t = \sigma\left(\mathrm{SAGE}_r([\mathbf{x}_t \| \mathbf{h}_{t-1}],\ \mathbf{E})\right), \tag{5}$$

$$\tilde{\mathbf{h}}_t = \tanh\left(\mathrm{SAGE}_h([\mathbf{x}_t \| \mathbf{r}_t \odot \mathbf{h}_{t-1}],\ \mathbf{E})\right), \tag{6}$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t, \tag{7}$$

where Eq.4 denotes update gate, Eq.5 denotes reset gate, Eq.6 denotes candidate state, Eq.7 denotes hidden state update, $\sigma(\cdot)$ denotes the sigmoid activation, $\odot$ denotes element-wise multiplication, $\|$ denotes concatenation, and SAGE_z, SAGE_r, SAGE_h are three independent 2-layer GraphSAGE encoders with input dimension $(d_h + d_h)$ and output dimension d_h.

This deep fusion design offers two key advantages over sequential spatial-temporal architectures. First, each node's temporal state transition is informed by the spatial context of its neighborhood at every time step, enabling tight coupling between spatial and temporal dynamics. Second, the three independent GraphSAGE encoders allow the model to learn distinct spatial aggregation patterns for different gate functions: the update gate learns which spatial patterns indicate the need to retain or replace historical information, while the reset gate learns which neighborhood configurations warrant forgetting past states.

Temporal Attention Mechanism. To address the information bottleneck in standard recurrent architectures, where the final hidden state must compress all historical information, we introduce a temporal attention mechanism. Given the sequence of hidden states$\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T] \in \mathbb{R}^{T\times N\times d_h}$, the attention weights are computed as:

$$e_t = \mathbf{w}_2^{\top} \cdot \tanh(\mathbf{W}_1\mathbf{h}_t + \mathbf{b}_1) + b_2, \tag{8}$$

$$\alpha_t = \frac{\exp(e_t)}{\sum_{\tau=1}^{T}\exp(e_\tau)}, \tag{9}$$

where $\mathbf{W}_1 \in \mathbb{R}^{(d_h/2)\times d_h}$ and $\mathbf{w}_2 \in \mathbb{R}^{d_h/2}$ are learnable parameters. The final representation is obtained by weighted aggregation:

$$\mathbf{h}_{\text{agg}} = \sum_{t=1}^{T} \alpha_t \odot \mathbf{h}_t. \tag{10}$$

This mechanism allows the model to actively focus on key historical points, rather than just the most recent hidden state. This is particularly advantageous for identifying cyclical traffic trends and understanding long-term time-related patterns.

Output Projection. The aggregated representation $\mathbf{h}_{\text{agg}} \in \mathbb{R}^{N\times d_h}$ is passed through a two-layer MLP decoder to produce the final prediction:

$$\hat{\mathbf{x}}_{T+1} = \mathbf{W}_o^{(2)} \cdot \text{ReLU}\big(\mathbf{W}_o^{(1)}\mathbf{h}_{\text{agg}} + \mathbf{b}_o^{(1)}\big) + \mathbf{b}_o^{(2)}, \tag{11}$$

where $\mathbf{W}_o^{(1)} \in \mathbb{R}^{d_h\times d_h}$,$\mathbf{W}_o^{(2)} \in \mathbb{R}^{d_{\text{out}}\times d_h}$, and dropout ($p = 0.1$) is applied between the two layers. We refer to this module as the decoder, which is structurally decoupled from the encoder (input projection, GraphSAGE-GRU cells, and temporal attention) to facilitate flexible transfer learning strategies.

3.3 Topology-Robust Transfer Learning

The inductive nature of GraphSAGE ensures that all model parameters, including those within the GRU gates, depend solely on the hidden dimensiond_h and are independent of the graph size N. This property enables direct weight transfer between models operating on different topologies without any architectural modification. When transferring

from a source topology $\mathcal{G}^{(s)}$ with N_s nodes to a target topology $\mathcal{G}^{(t)}$ with N_t nodes, we simply replace the edge index $\mathbf{E}^{(s)}$ with $\mathbf{E}^{(t)}$; all weight matrices remain unchanged.

The encoder parameters (input projection, GraphSAGE-GRU cells, and temporal attention) are frozen, and only the decoder (output projection) is fine-tuned on the target domain. This is the most parameter-efficient strategy, requiring optimization of only ∼4K parameters out of ∼78K total. It is best suited for scenarios with limited target domain data. Formally, the optimization objective is:

$$\theta_{\text{enc}}^{(t)} = \theta_{\text{enc}}^{(s)}, \quad \theta_{\text{dec}}^{(t)} = \arg\min_{\theta_{\text{dec}}} \mathcal{L}\left(\theta_{\text{enc}}^{(s)}, \theta_{\text{dec}}; \mathcal{D}^{(t)}\right), \tag{12}$$

where $\mathcal{D}^{(t)}$ denotes the fine-tuning dataset collected from the target topology $\mathcal{G}^{(t)}$.

3.4 Training Objective and Optimization

The model minimizes MSE over all nodes and mini-batch samples, providing a smooth signal for continuous traffic regression. Updates use the Adam optimizer (weight decay 10^{-5}) with gradient clipping (max norm 1.0) to stabilize recurrent training. A plateau-based scheduler halves the learning rate after 5 non-improving validation epochs (min learning rate 10^{-6}), and early stopping with patience of 15/10 epochs for pretraining/fine-tuning prevents overfitting.

4 Experiments

4.1 Datasets

Dataset Generation Framework. Our datasets are generated using the ns3-rdma simulator [9], an open-source network simulator specifically designed for modeling RDMA-enabled data center networks. The simulator implements the DCQCN protocol, which is widely deployed in modern RDMA networks for congestion control.

Network Topologies. We generate four distinct datasets corresponding to four typical data center network topologies.

Fat Tree Topology. The Fat Tree topology is a widely-adopted hierarchical structure that provides full bisection bandwidth and multiple paths between any pair of hosts. This topology consists of multiple layers of switches arranged in a tree-like structure with redundant paths for fault tolerance and load balancing.

Spine-Leaf Topology (Variant 1). The Spine-Leaf topology is an engineered, simplified version of the Clos network architecture for modern data centers (two-tier Clos). This design features a spine layer of high-capacity switches connected to a leaf layer, where end hosts are attached. The Spine-Leaf topology is simple to scale horizontally, and adding a ridge switch can improve the bandwidth.

Spine-Leaf Topology (Variant 2). The second Spine-Leaf topology configuration scales the first horizontally, offering different scale and capacity characteristics to enable comparative analysis across varying network sizes.

Dragonfly Topology. The Dragonfly topology represents a high-radix, low-diameter network design that minimizes the number of hops between nodes while maintaining high bandwidth. This topology is particularly relevant for large-scale data centers and high-performance computing environments.

4.2 Experimental Settings

Implementation Details. We implement Adpt-STGIN using PyTorch 2.8.0 and PyTorch Geometric 2.6.1 with CUDA 12.8. All experiments are conducted on a workstation equipped with an Intel Core i5-13400F processor (2.50 GHz), 16 GB RAM, and an NVIDIA GeForce RTX 4060 Ti GPU.

Hyperparameter Configuration. Table 1 summarizes the hyperparameters used across different training scenarios.

Table 1. hyperparameter configuration

Configuration	Training Stage	
	Pretraining	**Transfer (Frozen Encoder)**
Max Epochs	200	200
Initial Learning Rate	1e-3	1e-4
Batch Size	32	32
Early Stop Patience	15	10
LR Reduction Factor	0.5	0.5
LR Reduction Patience	5	5
Weight Decay	1e-5	1e-5
Gradient Clipping	1.0	1.0

For pretraining, we use an initial learning rate of 10^{-3} with a batch size of 32. The learning rate is reduced by a factor of 0.5 when validation loss plateaus for 5 consecutive epochs, with a minimum threshold of 10^{-6}. Early stopping with a patience of 15 epochs prevents overfitting. For transfer learning with frozen encoder strategy, we use a lower learning rate of 10^{-4} to fine-tune only the decoder, with a reduced patience of 10 epochs to accelerate convergence on the target domain.

Experimental Scenarios. We design two experimental scenarios to evaluate the proposed framework:

Scenario 1: Pretraining. We train independent models on each of the three topologies (Spine-Leaf, Fat-Tree, Dragonfly) to assess the model's capacity to learn topology-specific traffic patterns.

Scenario 2: Topology-Robust Transfer Learning. We evaluate four transfer paths to assess the model's adaptability to new topologies:

- Fat-Tree → Spine-Leaf-v1: Transfer from a hierarchical topology (36 nodes) to a simpler two-tier topology (14 nodes).
- Fat-Tree → Dragonfly: Transfer from a hierarchical topology (36 nodes) to a high-radix topology (60 nodes).
- Spine-Leaf-v1 ↔ Spine-Leaf-v2: Transfer between topologies with similar architecture but different scales (14 nodes → 34 nodes).

For all transfer learning experiments, we employ the frozen encoder strategy, where only the decoder parameters are fine-tuned on the target domain while the encoder remains frozen to preserve learned ST representations.

Evaluation Metrics. Three metrics, namely MSE, MAE and R^2 are employed to assess the model performance. To quantify transfer effectiveness, we compare zero-shot performance (directly applying the pretrained model to the target topology without any fine-tuning) against fine-tuned performance.

4.3 Experimental Results and Analysis

Pretraining Performance. Table 2 presents the performance of Adpt-STGIN and six baseline models trained independently on each topology. The baselines include GRU, which models each node's traffic as an independent time series; GCN+GRU, which sequentially applies a graph convolutional network for spatial encoding followed by a GRU for temporal modeling; STGCN [15], which alternates graph convolution and 1-D convolution layers for ST feature extraction; DCRNN [7], which models traffic as a diffusion process on directed graphs using recurrent architectures; MTGNN [16], which jointly learns the graph structure and temporal dependencies via mix-hop propagation; and STAEformer [17], a Transformer-based model with adaptive ST embedding. All baselines are transductive and are retrained from scratch on each topology.

Through quantitative analysis, we have the following findings. First, Adpt-STGIN achieves competitive performance across all topologies, with R^2 consistently above 0.990. However, its margin over the best-performing baselines (STGCN, DCRNN) is modest, typically within 0.001-0.003 in MSE. Such a result is explainable. For prediction on a fixed graph, transductive models such as STGCN can directly exploit the static graph structure, offering a natural advantage. Second, the competitive accuracy of Adpt-STGIN should be understood alongside a fundamental capability difference. All baseline models are transductive. Their parameters are tied to a specific graph structure, and applying them to a new topology generally requires full retraining from scratch. Adpt-STGIN, by contrast, decouples model parameters from graph size and structure through its inductive GraphSAGE-GRU encoder, enabling transfer to unseen topologies without architectural modification.

Table 2. pretraining results

Model	Spine-Leaf		Fat-Tree		Dragonfly	
	MSE	**R^2**	**MSE**	**R^2**	**MSE**	**R^2**
GRU	0.0015	0.9951	0.0019	0.9912	0.0028	0.9950
GCN+GRU	0.0016	0.9950	0.0019	0.9911	0.0028	0.9949
STGCN	0.0015	0.9954	0.0017	0.9918	0.0026	0.9954
DCRNN	0.0015	0.9953	0.0019	0.9910	0.0029	0.9948
MTGNN	0.0016	0.9948	0.0023	0.9890	0.0048	0.9914
STAEformer	0.0023	0.9926	0.0030	0.9859	0.0032	0.9942
Adpt-STGIN	0.0015	0.9953	0.0021	0.9900	0.0027	0.9950

Topology-Robust Transfer Learning. Fig. 2(a)--(d) visualize the zero-shot versus fine-tuned performance for each transfer path. All pretrained Adpt-STGIN models achieve $R^2 > 0.994$ on target topologies without any fine-tuning, demonstrating that the inductive GraphSAGE-GRU encoder captures transferable ST representations that generalize across diverse network architectures, a capability no baseline model possesses.

For cross-architecture transfers (Fat-Tree → Spine-Leaf-v1: +7.60%; Fat-Tree → Dragonfly: +5.32%), fine-tuning only the lightweight decoder yields consistent MSE improvements, validating the encoder-decoder decoupling design where the encoder learns topology-agnostic representations while the decoder adapts to target-specific output distributions. The total fine-tuning time is 127 s for Fat-Tree → Spine-Leaf-v1 and 96 s for Fat-Tree → Dragonfly, which is substantially lower than the cost of retraining a model from scratch on the target topology.

Transfers between Spine-Leaf variants (v1 ↔ v2) show negligible fine-tuning gain (-0.10% to +0.44%), indicating that topologies sharing similar architectural patterns require minimal adaptation and that zero-shot performance is already near-optimal in such cases.

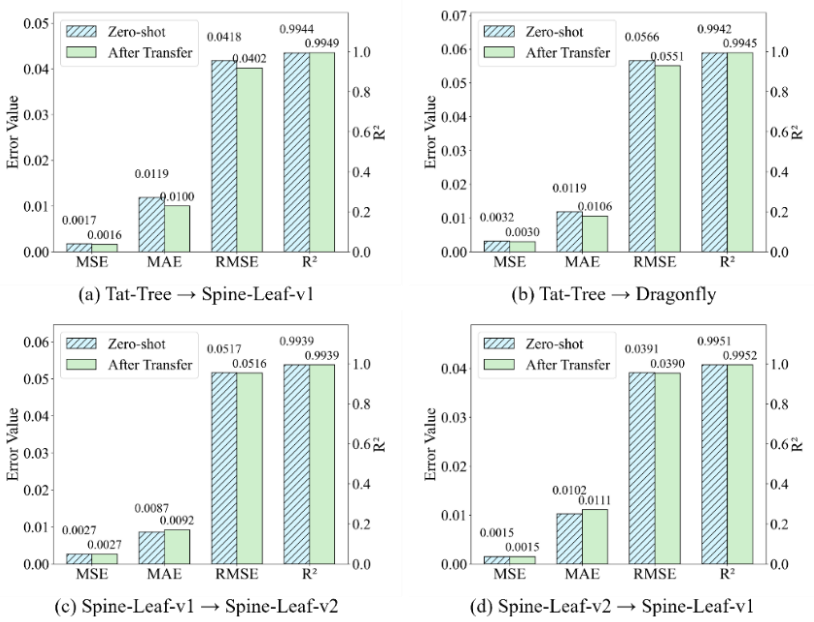

Fig. 2. Zero-shot vs. fine-tuned performance across different transfer paths.

5 Conclusion

In this paper, we proposed Adpt-STGIN, an adaptive ST graph inductive network for topology-robust data center traffic prediction. By deeply integrating an inductive GraphSAGE encoder with GRU temporal gating, the model decouples its parameters from graph topology, enabling direct transfer to unseen networks.

Experiments on four data center topologies show that Adpt-STGIN achieves $R^2 > 0.99$ during pretraining and maintains strong zero-shot generalization ($R^2 > 0.994$) on unseen topologies with rapid convergence. These results validate both the prediction accuracy and transferability of the proposed framework. Future work will investigate the relationship between traffic pattern similarity and transfer effectiveness across diverse data center architectures.

Acknowledgments. This paper is sponered by National Key Research and Development Program of China under grant No.2023YFB2904200 and Frontier Technologies R&D Program of Jiangsu under grant No.BF2024067.

References

1. Kumar, S.V., Vanajakshi, L.: Short-term traffic flow prediction using seasonal ARIMA model with limited input data. European Transport Research Review **7**(3), 21 (2015)

2. Hong, W.-C.: Traffic flow forecasting by seasonal SVR with chaotic simulated annealing algorithm. Neurocomputing **74**(12–13), 2096–2107 (2011)
3. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
4. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734 (2014)
5. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)
6. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. In: International Conference on Learning Representations (ICLR) (2018)
7. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: data-driven traffic forecasting. In: International Conference on Learning Representations (ICLR) (2018)
8. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in Neural Information Processing Systems **30** (2017)
9. Zhu, Y., Eran, H., Firestone, D., Guo, C., Lipshteyn, M., Liron, Y., Padhye, J., Raindel, S., Yahia, M.H., Zhang, M.: Congestion control for large-scale RDMA deployments. ACM SIGCOMM Computer Communication Review **45**(4), 523–536 (2015)
10. Zhang, Z., Liu, F., Zeng, Z., Zhao, W.: A traffic prediction algorithm based on Bayesian spatio-temporal model in cellular network. In: 2017 International Symposium on Wireless Communication Systems (ISWCS), pp. 43–48 (2017)
11. Zhang, Z., Shu, Z., Yang, S., Chen, S., Guo, T.: Network traffic prediction study based on the adaptive attention mechanism. In: 2023 3rd International Conference on Electronic Information Engineering and Computer Communication (EIECC), pp. 500–506 (2023)
12. Jin, Z., Qian, J., Kong, Z., Pan, C.: A mobility aware network traffic prediction model based on dynamic graph attention spatio-temporal network. Computer Networks **235**, 109981 (2023)
13. Huang, B., Ruan, K., Yu, W., Xiao, J., Xie, R., Huang, J.: ODformer: spatial–temporal transformers for long sequence origin–destination matrix forecasting against cross application scenario. Expert Systems with Applications **222**, 119835 (2023)
14. Sun, X., Xiong, R., Shen, D., Luo, J.: Enhancing network traffic prediction by integrating graph transformer with a temporal model. In: Proceedings of the 9th Asia-Pacific Workshop on Networking, pp. 150–156 (2025)
15. Yu, B., Yin, H., Zhu, Z.: Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), pp. 3634–3640 (2018)
16. Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., Zhang, C.: Connecting the dots: multivariate time series forecasting with graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 753–763 (2020)
17. Liu, H., Dong, Z., Jiang, R., Deng, J., Deng, J., Chen, Q., Song, X.: Spatio-temporal adaptive embedding makes vanilla transformer SOTA for traffic forecasting. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM), pp. 4125–4129 (2023)