TSGFM - Graph Neural Networks for Zero-Shot Time Series Forecasting in Network Monitoring

Hamid Latif-Martínez[†*], Juan Vanerio[*,∞], Pedro Casas[*], José Suárez-Varela[‡]
Albert Cabellos-Aparicio[†], Pere Barlet-Ros[†]
[†]Polytechnic University of Catalonia, [‡]Telefónica Research, Spain
[*]Austrian Institute of Technology, [∞]University of Vienna, Austria
{hamid.latif, alberto.cabellos, pere.barlet}@upc.edu
{name.surname}@ait.ac.at, jose.suarez-varela@telefonica.com

***Abstract*—We present TSGFM, a Time Series Graph Foundation Model for zero-shot network monitoring, leveraging spatiotemporal Graph Neural Networks (GNNs) to extract transferable representations across diverse multivariate time series (MTS) domains. Pretrained on heterogeneous time series datasets, TSGFM enables generalization without task-specific fine-tuning, addressing core challenges in dynamic network environments. TSGFM is benchmarked across five real-world MTS datasets and seven zero-shot forecasting scenarios, outperforming five state-of-the-art baselines in six out of seven tasks. Most notably, in zero-shot network monitoring analysis, TSGFM surpasses all competing models by at least 18%, even without any prior exposure to network monitoring data. We further compare TSGFM against leading Time Series Foundation Models (TSFMs), including TimeGPT and TimesFM. TSGFM achieves performance on par with TimeGPT, occasionally surpassing it, and consistently outperforms TimesFM, while using significantly less pretraining data and relying on a much simpler architecture.**

A detailed analysis of TSGFM's learned spatial attention patterns reveals domain-specific connectivity structures. In particular, lower attention weights in network monitoring tasks suggest that dense spatial graphs may be unnecessary, opening opportunities for efficient spatial pruning without sacrificing accuracy. This challenges prevailing assumptions favoring fully connected spatiotemporal GNNs. To foster transparency and reproducibility, we release the complete implementation of TSGFM as open source, as well as the tested datasets.

***Index Terms*—Graph Neural Networks; Multivariate Time Series; Time Series Foundation Models**

I. Introduction

Modern network infrastructures – spanning cloud-native systems, edge computing, and IoT ecosystems – require intelligent monitoring mechanisms to ensure reliability, performance, and security. A core enabler of such mechanisms is multivariate time series (MTS) analysis, which captures rich, temporally-evolving signals essential for diagnosing anomalies, forecasting behavior, and informing operational decisions [1], [2], [3]. However, detecting anomalies and forecasting future patterns in network settings remains challenging due to the high dimensionality of network measurement data, non-stationary traffic patterns, and the limited availability of ground-truth data [4].

Traditional forecasting models for MTS data often struggle to generalize across diverse network environments, requiring significant retraining and domain-specific adaptations. This makes MTS forecasting particularly challenging, as it requires models capable of capturing complex temporal patterns while maintaining computational efficiency. While traditional statistical models for time series forecasting have been widely explored, they often fall short when applied to the non-stationary, non-linear, and noisy nature of network monitoring data, resulting in suboptimal predictions. As a result, modern deep learning approaches have gained traction in recent years due to their ability to handle complex dependencies and generate more accurate, realistic forecasts [4]. Still, most models in the literature frequently require dataset-specific training, which is both resource-intensive and time-consuming.

To address these challenges, we explore zero-shot learning – a strategy that empowers models to generalize to unseen network scenarios without retraining. Recent breakthroughs in foundation models have demonstrated how broad pretraining on diverse data can enable such generalization [5]. While this paradigm has reshaped natural language and vision domains, it remains largely untapped for network monitoring.

In this work, we take a step in that direction by proposing TSGFM, a spatiotemporal GNN architecture pretrained on diverse MTS datasets to support zero-shot forecasting in network environments. Rather than claiming full foundation model capabilities, we position TSGFM as an early-stage effort toward that vision, demonstrating that cross-domain pretraining can enable significant improvements in network telemetry analysis, even in zero-shot conditions. TSGFM captures complex relationships by modeling inter-dependencies among different time series (i.e., spatial correlations) and internal temporal dynamics within each time series (i.e., temporal correlations). To effectively represent these correlations, TSGFM employs an attention-based graph structure, modeling each time series as a node connected through spatial edges in a full-mesh topology, and temporal edges that link nodes across consecutive timesteps. To assess TSGFM performance, we conduct an extensive evaluation using five publicly available datasets spanning multiple diverse domains. We compare TSGFM with a broad selection of state-of-the-art deep learning models, including more traditional neural architectures as well as advanced graph-based architectures. We perform zero-shot testing and cross-validation, training models on a collection of datasets and testing them on an unseen dataset. TSGFM achieves superior performance in six out of seven zero-shot testing scenarios, outperforming all competing models by at least 18% in zero-shot network monitoring, and maintaining a simple and interpretable graph representation.

In addition, we benchmark TSGFM against recently proposed Time Series Foundation Models (TSFMs), including TimeGPT [6] and TimesFM [7], which have shown strong performance in a wide range of forecasting tasks. Despite the lightweight nature of TSGFM, it achieves competitive results with TimeGPT and consistently outperforms TimesFM across network monitoring datasets. TSGFM delivers this performance while requiring orders of magnitude less pretraining data and avoiding the complexity of large-scale architectures; in fact, TSGFM operates with approximately 220K parameters, making it around three orders of magnitude smaller than TimesFM, TimeGPT, and other TSFM models. This makes TSGFM substantially more efficient and deployable in practical settings. Unlike TSFMs, which often demand extensive computational infrastructure for both training and inference, TSGFM offers a scalable and resource-friendly alternative – particularly advantageous for real-time network monitoring scenarios where computational efficiency is critical.

This paper elaborates on our initial design of TSGFM [8] (poster), where we introduced the concept of a graph foundation model for time series analysis and provided preliminary zero-shot evaluation results across multiple domains. Different from [8], here we extend the evaluation with deeper experimental analyses, benchmarking TSGFM not only against traditional AI/ML and GNN baselines but also against recent TSFMs. In addition, we conduct an in-depth study of TSGFM's learned spatial and temporal attention patterns, showing that dense spatial graphs are often unnecessary and that pruning can be applied without loss of accuracy.

The remainder of the paper is organized as follows: Section II reviews the related work. Section III describes the main concepts and architecture of TSGFM. Section IV reports the results from our extensive evaluation and delves into potential refinements to the model, based on an analysis of its self-attention mechanism. Section V presents concluding remarks.

II. Related work

Foundation models are models pretrained with large amounts of data, which allows them to learn a wide range of distributions and to be used for inference in different, unseen scenarios. This capability makes them specially suited for scenarios in which few data samples are available for training, or when limited computational resources are available to train these models from scratch. There is a recent surge in papers targeting the development of foundation models for time series data [9], capable of generating accurate predictions for diverse datasets not seen during training. The underlying concept of these models is to rely on highly expressive, large-scale architectures that are trained on millions or billions of time series data points, coming from very diverse domains and having high heterogeneity in terms of temporal behaviors and characteristics. TimeGPT [6], PromptCast [10], LLMTime [11], TimesFM [7], Lag-Llama [12], and FAE [13] are all examples of novel foundation models for time series forecasting, which target a zero-shot learning application. The main limitation of foundation models for time series forecasting, particularly in the context of network monitoring, lies in their underlying hardware requirements. Due to their extensive pretraining and large-scale architectures, these models often demand significant computational resources, including high-memory GPUs and large-scale distributed systems, making them impractical for deployment in resource-constrained environments.

AI/ML models have been extensively applied to MTS forecasting, with Recurrent Neural Networks (RNNs) standing out for their ability to model temporal dependencies. For example, the authors of [14] use RNNs to forecast weather temperatures and validate their approach on an hourly weather dataset from five Chinese cities. However, RNNs face well-known limitations such as vanishing and exploding gradients, which hinder their performance on long sequences [15]. To mitigate these issues, more advanced variants like Gated Recurrent Units (GRUs) have been proposed. In [16], a GRU-based architecture is used to handle MTS forecasting in the presence of missing values, showing improved robustness.

More recently, Transformer architectures have gained traction for time series modeling due to their ability to capture long-range dependencies through self-attention mechanisms. For instance, [17] introduces a transformer-based model that encodes the value of each time series at each timestamp as individual tokens, enabling the model to learn spatial and temporal correlations jointly, without requiring an explicit graph structure. While this approach provides modeling flexibility, it suffers from a major drawback: quadratic memory and computation costs with respect to the number of time series, making it impractical for large-scale telemetry datasets.

MTS data can naturally be represented as a graph, making Graph Neural Networks (GNNs) a compelling choice for modeling such data [18]. While temporal dependencies are typically the dominant factor in forecasting tasks, spatial relationships – such as correlations between different sensors or metrics – can also carry valuable predictive signals. In an MTS graph representation, each time series is modeled as a node, and edges encode potential dependencies between them. This structure enables GNNs to jointly learn from both the temporal evolution of individual signals and their interdependencies. Moreover, incorporating attention mechanisms further enhances GNNs by allowing the model to dynamically weigh edge strengths, leading to more flexible and accurate spatiotemporal modeling. In [19], the authors propose a MTS forecasting model that considers intra and inter time series correlations, represented by node-to-node dependencies in a fully connected space-time graph, similar to [17]. To address computational complexity, they introduce the Fourier Graph Operator, which reduces dimensionality in the Fourier space while preserving graph information. They compare their

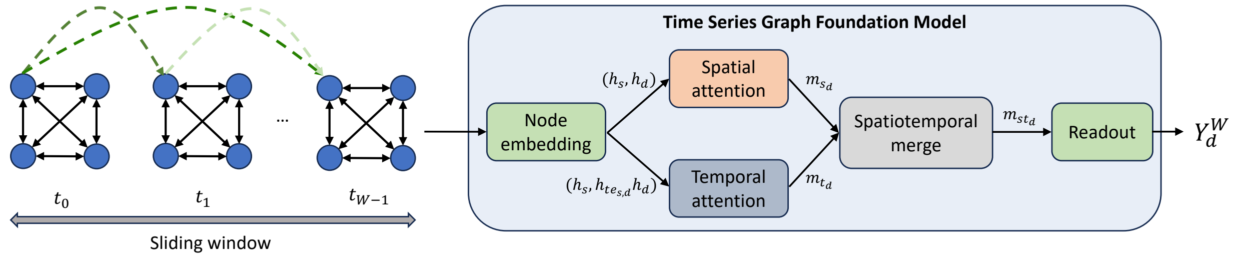


Fig. 1: TSGFM – Time Series Graph Foundation Model's architecture and workflow for MTS forecasting. Sliding windowing for context, node embeddings for time series, spatial and temporal attention, spatiotemporal merge, and readout stages.

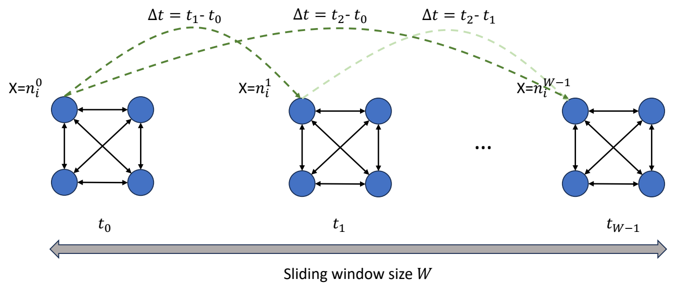


Fig. 2: TSGFM's spatiotemporal graph representation.

proposal against baseline models representative of the state-of-the-art using seven datasets containing different types of data (including transport traffic, energy, web traffic) and prove that their proposed method outperforms the baselines.

III. Time Series Graph Foundation Model

TSGFM is a GNN-based model that incorporates spatial and temporal information from correlated time series to predict future values of each time series. To address the challenge of spatiotemporal time series forecasting, we propose a *step toward a foundation model* – designed to capture complex dependencies across multiple datasets through pretraining and generalization. TSGFM leverages graph-based representations to encode both spatial and temporal correlations between time series. Figure 1 shows an overview TSGFM's architecture and workflow for MTS forecasting. Sliding windowing for context, node embeddings for time series, spatial and temporal attention, spatiotemporal merge, and readout stages are described next.

A. Spatiotemporal Graph Data Representation

Our representation is designed to flexibly capture spatial and temporal dependencies without relying on prior assumptions. We represent each time series as a node within a graph, enabling the model to exploit potential relationships between time series in the same dataset. We opted for fully-connected graphs so that the GNN can discover spatial correlations automatically, with the spatial attention mechanism assigning dynamic weights to each pair of nodes. This avoids imposing a fixed topology and allows the model to focus on the most relevant dependencies.

When handling multiple datasets, we generate N fully-connected graphs for each time interval t. To capture temporal correlations, we introduce temporal edges that connect each node n at timestep t to itself at future timesteps t', where $t' = t + \Delta t$ and $\Delta t \in 1, 2, \ldots$. These edges of the form $(n^t, n^{t'})$ allow the model to learn how a past observation influences future states in the same time series. Inspired by [20], we add a feature Δt to every temporal edge, representing the relative temporal distance between the two nodes. This design leverages temporal attention to highlight which past timesteps are most relevant for predicting future dynamics, effectively modeling the temporal evolution of each series. An overview of this representation is shown in Figure 2.

We implement a sliding window of size W in the temporal dimension, defining the context of the model. For each timestep t, we generate N independent graphs – one per dataset – where each node contains the value of a specific time series at that time. The sliding window produces a total of $N \times W$ graphs, capturing both the spatial structure at each time step and the temporal progression across the windows.

B. Underlying spatiotemporal GNN model

We implement a Message Passing Neural Network (MPNN) to harness the proposed data representation. We design a message passing composed of two phases executed in parallel: (i) a temporal message passing, and (ii) a spatial message passing. Let a node represent a time series at a specific timestep within the sliding window. We begin by projecting the node features into a hidden representation by using a Multilayer Perceptron (MLP), resulting in an initial hidden state h_n for each node n. This is followed by T message passing iterations, where both spatial and temporal information are propagated. In the temporal message passing phase, we generate an embedding for the temporal edges, denoted as h_{te}, using the Δt attribute, which represents the relative temporal distance between the source and destination nodes. For each temporal edge $te_{s,d}$ connecting the source node s to a destination node d, we assign an attention coefficient $\alpha_{te_{s,d}}$, computed as:

$$\alpha_{te_{s,d}} = MLP(h_s, h_{te_{s,d}}, h_d) \quad (1)$$

$$m_{t_d} = \sum s \in \mathcal{N}_t(d) \alpha_{te_{s,d}} \cdot h_s \quad (2)$$

TABLE I: Overview of the five publicly available datasets employed to evaluate TSGFM.

Dataset	Description	# TS	# samples
Spain	Electrical consumption, generation, pricing, and weather data from Spain	20	35K (hourly)
ETTh1	Operational data from two electricity substations, including transformer oil levels and temperature readings	7	17.5K (hourly)
Exchange	Daily exchange rates of eight foreign currencies from 1990 to 2016	8	7.6K (daily)
TELCO	Different time series with a five-minute granularity, representing typical data monitored in a mobile ISP	12	61K (5')
Abilene	Traffic matrix data collected from the Abilene backbone network	132	48K (5')

where h_s and h_d are the embeddings of the source and destination nodes, respectively, and $h_{te_{s,d}}$ is the temporal edge embedding derived from the relative time difference Δt between s and d. The attention coefficients $\alpha_{te_{s,d}}$ are then used to weight the incoming messages received by each node during the aggregation: for each destination node d, we aggregate the incoming messages from its incoming temporal neighbors, $\mathcal{N}_t(d)$, by performing a weighted sum of the source node embedding. Similarly, we model spatial dependencies by assigning attention coefficients to spatial edges between nodes representing different time series at the same timestep. For each spatial edge $se_{s,d}$ we compute:

$$\alpha_{se_{s,d}} = MLP(h_s, h_d) \tag{3}$$

$$m_{s_d} = \sum s \in \mathcal{N}_t(d) \alpha_{te_{s,d}} \cdot h_s \tag{4}$$

Where h_s and h_d are the embeddings of the source and destination nodes, respectively. Rather than updating the node embedding separately with the spatial and temporal messages, m_{s_d} and m_{t_d}, respectively, we first merge these messages in m_{st_d}, and update the node embedding using an *update* function $h'_d = update(h_d, m_{st_d})$. Finally, after T iterations, we use the hidden states of the nodes from the final timestep in the sliding window (i.e., $t = W-1$) to produce the forecasting output for each time series at the next timestep using a *readout* function $Y_d^W = readout(h'_d)$. In total, the model has 220K trainable parameters, with the largest share in the GRU-based update (∼99K), followed by the spatial (∼33K) and temporal (∼37K) attention modules, while the embeddings and readout MLP remain comparatively lightweight.

IV. Evaluation

In this section, we assess and benchmark the performance of TSGFM as a foundation model for MTS forecasting across different domains, further studying its internal workings through the analysis of its attention mechanisms. To foster transparency and reproducibility, we release the complete implementation of TSGFM as open source, as well as all the tested datasets presented in this section (https://github.com/BNN-UPC/Papers/wiki/TSGFM).

TABLE II: Dataset permutations used for training and testing.

Permutation	Training	Testing
A	Exchange, **Abilene**, **TELCO**, ETTh1	Spain
B	**TELCO**, Exchange, Spain, ETTh1	**Abilene**
C	Exchange, Spain, **Abilene**, ETTh1	**TELCO**
D	Spain, **Abilene**, **TELCO**, ETTh1	Exchange
E	Exchange, Spain, **Abilene**, **TELCO**	ETTh1

A. Experimental Setup

*1) **Datasets**:* we evaluate TSGFM on five publicly available datasets from diverse domains: two from the energy sector [21], [22], one from stock-market finance [23], and two from networking [24], [25]. Table I summarizes their main characteristics. The datasets differ widely in the number of series, sample size, and frequency. Both networking datasets span about six months at 5-minute resolution: Abilene contains nearly 48,000 traffic matrices with 132 Origin-Destination flows derived from NetFlow and routing data, reported in units of 100 bytes/5 minutes and showing clear diurnal and weekly patterns; TELCO comprises twelve ISP time series covering data usage, calls, and SMS, capturing realistic seasonal and operational behaviors. In contrast, Exchange contains eight daily financial time series spanning several decades, and the energy datasets include between two and four years of multiple hourly time series. This diversity in scale, frequency, and domain makes the benchmark a rigorous test of TSGFM's generalization capabilities.

To evaluate the zero-shot performance and robustness of TSGFM to adapt to new domains, we generate all possible permutations of the datasets using a 4-to-1 split, where four datasets are used for training and one for testing in each permutation. Table II describes the five permutations. The training datasets are further split into 90% for training and 10% for validation.

*2) **Baseline models**:* we conduct a comprehensive comparison of TSGFM with a range of representative and state-of-the-art methods to evaluate its performance in MTS forecasting. The baseline models include simpler and more complex architectures to study how different complexities and design decisions influence the final results.

First, we compare against basic spatiotemporal architectures, such as a simple spatiotemporal GNN. This model initializes the hidden state of each node using a RNN, which encodes a sliding window of length W, and updates the node states iteratively through message passing with their neighbors.

We also evaluate against GAT-AD [26], a Graph Attention Network (GAT)-based model that incorporates dynamic attention mechanisms to capture spatiotemporal dependencies. Furthermore, we include FourierGNN [19], a GNN-based architecture that combines frequency domain transformations to efficiently capture spatial and temporal relationships.

For non-graph baselines, we include RNNs, which are well-suited for processing sequential data. These models maintain a hidden state that captures temporal correlations across timesteps, making them ideal for modeling temporal data. In

TABLE III: Zero-shot forecasting performance on all five dataset permutations.

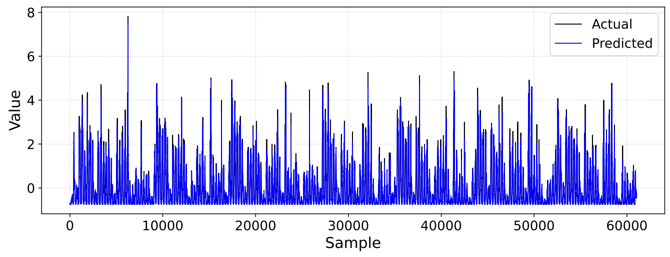
Model \ Perm.	A		B		C		D		E		Avg.	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
RNN	1.23	1.13	1.62	1.04	0.99	1.00	1.25	1.29	**0.97**	**0.94**	1.52	1.05
FreTS	**0.91**	0.86	1.25	1.01	1.11	1.01	**0.75**	**0.86**	0.99	0.96	1.20	**1.00**
BasicSTGNN	1.32	1.08	2.16	1.12	1.02	0.99	1.75	1.57	0.98	0.95	1.96	1.11
FourierGNN	1.03	0.96	3.29	2.82	**0.98**	**0.98**	**0.75**	**0.86**	1.00	0.98	2.83	2.45
GAT-AD	0.98	**0.82**	1.32	1.22	1.40	1.17	4.75	5.29	2.00	2.12	1.34	1.22
TSGFM	1.00	1.00	**1.00**	**1.00**	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**	**1.00**

our case, RNNs process sliding window sequences of size W, where each sequence consists of W consecutive samples of each time series. Furthermore, we compare with FreTS [27], which leverages frequency-domain transformations and simple MLP architectures, demonstrating that lower-complexity models can also be effective for MTS forecasting.

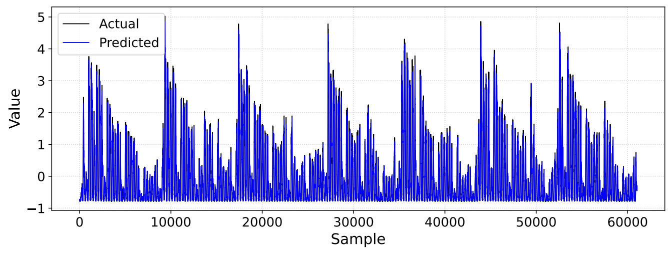
The selection of baselines ensures a comprehensive evaluation of TSGFM against a variety of architectures, ranging from classical models to modern frequency-enhanced and graph-based approaches.

*3) **TSFM benchmark:*** to comprehensively evaluate TSGFM's zero-shot forecasting capabilities, we benchmark it against recently proposed TSFMs, which are specifically designed for generalization across datasets without retraining. We include two prominent TSFMs in our evaluation: TimeGPT [6] is the first foundation model explicitly developed for time series forecasting. It is based on a customized encoder–decoder Transformer architecture trained from scratch on over 100 billion time points spanning diverse domains such as finance, healthcare, meteorology, and energy. Unlike models adapted from large language models (LLMs), TimeGPT is purpose-built to minimize forecasting error. Its strength lies in its ability to generalize to unseen datasets through zero-shot inference. TimesFM [7] adopts a decoder-only Transformer architecture inspired by LLMs, but is trained exclusively on a mix of real-world and synthetic time series data. Key architectural innovations include input patching (analogous to tokenization), patch masking for robust training, and output chunking to enable efficient long-horizon forecasting. TimesFM has demonstrated strong zero-shot performance across several benchmarks [7], but generally benefiting from fine-tuning when tested in new data.

*4) **Experimental settings:*** all models use the same training, validation, and testing datasets, the same split sizes, and the same data normalization strategy. Specifically, we apply min-max normalization to each dataset, scaling values to $[0, 1]$ using the minimum and maximum computed on the training split of that dataset. In all executions, we use a fixed seed $s = 42$ for the sake of reproducibility. All the experiments are conducted in Python using Pytorch 2.4 and a single NVIDIA RTX 3090 GPU. All the models are trained using the Mean Absolute Error (MAE) loss function, a learning rate of $\eta = 0.001$, and we set $T = 1$ for TSGFM. We set up early stopping with a patience of 4 based on the training loss (i.e., the training stops after 4 epochs with no improvement).



(a) Total amount of prepaid mobile data transfers (TS11).



(b) Number of prepaid mobile data transfers (TS12).

Fig. 3: TSGFM predictions and ground truth values for two testing time series in scenario C (TELCO).

In terms of context duration, we use a fixed context length of $W = 32$, which defines the size of the input sliding window. For the networking datasets, where measurements are recorded at 5-minute intervals, this corresponds to a historical window of 2 hours and 30 minutes. This choice strikes a balance between capturing sufficient temporal dynamics and maintaining computational efficiency, and is consistent across all models to ensure a fair comparison.

B. *TSGFM vs. Baselines*

Table III shows the testing performance obtained by TSGFM and the baseline models for each permutation scenario, in terms of MAE and RMSE. All metrics are normalized with respect to TSGFM, allowing for a direct comparison. The final column reports the average performance of each model, weighted by the number of samples and time series of each dataset (i.e., by the number of single-value forecasts).

TSGFM consistently outperforms all competing models when considering the overall weighted performance. Although FourierGNN achieves competitive results in several permutations, its poor performance on the Abilene networking dataset – used in permutation B and the largest dataset in the evaluation – significantly impacts its overall average performance, making it the worst-performing model in aggregate. Across permutations A, C, and E, TSGFM and FourierGNN demonstrate nearly identical performance, indicating that our model is equally effective at capturing spatiotemporal dependencies. However, in permutation B, TSGFM clearly outperforms FourierGNN, highlighting its robustness on larger and more complex datasets. While FourierGNN slightly outperforms TSGFM in permutation D, the overall results suggest that TSGFM offers a more consistent and reliable performance, achieving a good forecasting accuracy without the added

TABLE IV: Dataset permutations used for training and testing (network scenarios).

Permutation	Training	Testing
NET-A	Exchange, Spain, ETTh1	TELCO
NET-B	Exchange, Spain, ETTh1	Abilene

TABLE V: Testing performance of forecasting models on network dataset permutations.

Model \ Perm.	NET-A		NET-B		Avg.	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
RNN	1.14	1.03	1.69	1.03	1.63	1.03
FreTS	1.10	1.00	1.19	1.02	1.18	1.02
BasicSTGNN	1.47	1.10	3.31	1.29	3.117	1.27
FourierGNN	1.67	1.86	4.83	5.01	4.51	4.69
GAT-AD	1.65	1.24	2.35	1.29	2.28	1.28
TSGFM	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

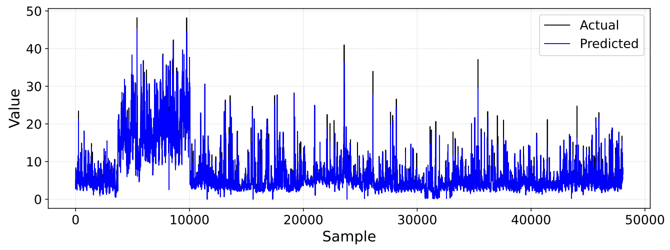
complexity of a hyper-graph representation. More generally, a model's effectiveness should be evaluated based on its consistency across diverse setups, rather than just its peak performance in select scenarios. While FourierGNN performs well in some cases, its severe underperformance on the Abilene dataset highlights its lack of robustness. A model that excels in certain conditions but fails drastically in others cannot be considered reliable, as its effectiveness is highly dependent on the specific dataset or setup. For completeness, Figure 3 provides a visual comparison of the predicted and actual values for two time series from the TELCO dataset – mobile networks' MTS data – in scenario C, as generated by TSGFM. Predictions highlight TSGFM's ability to capture temporal patterns, showcasing how closely the model forecasting aligns with real-world network behavior with different temporal patterns in this case.

To further investigate the feasibility of applying zero-shot models trained on non-network data to the networking domain, we introduce two new evaluation permutations: NET-A and NET-B, as detailed in Table IV. These permutations are adapted from B and C (cf. Table II), but are specifically designed to evaluate the extent to which learned temporal and structural patterns from non-network datasets can be effectively transferred to network-specific scenarios. The goal is to assess how well TSGFM generalizes to network monitoring tasks without direct exposure to network traffic data during pretraining, putting the emphasis on the general lack of labeled data in the networking domain.

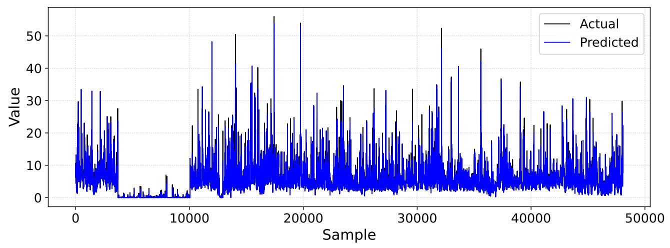
Table V reports the results on the two new permutations. TSGFM consistently outperforms all baselines by at least 18% compared to FreTS and by larger margins over the rest, highlighting its ability to extract correlations and transfer knowledge from diverse data sources to networking scenarios. This strong performance stems from two factors: (i) its capacity to transfer temporal and structural patterns from heterogeneous pretraining datasets, and (ii) its attention-based GNN design, which is well-suited to the non-stationary, bursty nature of

TABLE VI: Testing performance of TSGFM and TSFMs on network dataset permutations.

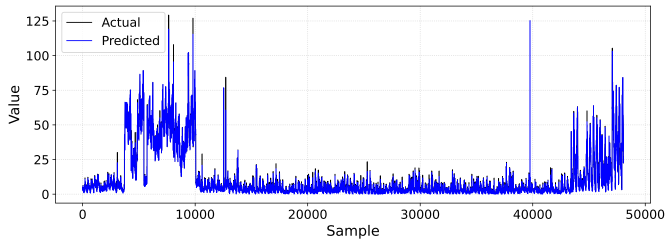
Model \ Perm.	NET-A		NET-B		Avg.	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
TimeGPT - GW	2.57	1.83	1.67	1.64	1.76	1.66
TimeGPT - SW	1.19	1.07	**0.90**	**0.90**	**0.93**	**0.92**
TimesFM - SW	6.32	3.99	2.34	2.26	2.73	2.43
TSGFM	**1.00**	**1.00**	1.00	1.00	1.00	1.00



(a) Network traffic from Kansas City to Houston (TS6).



(b) Network traffic from Atlanta to Denver (TS37).



(c) Network traffic from Seattle to Houston (TS84).

Fig. 4: TSGFM predictions and actual time series for three samples in the NET-B (Abilene) testing dataset.

network traffic. Together, these characteristics enable TSGFM to generalize effectively to network-specific forecasting tasks, reinforcing its potential as a robust and adaptable model for network monitoring. For completeness, Figure 4 illustrates three time series from NET-B (Abilene) along with TSGFM's predictions.

C. *TSGFM vs. Time Series Foundation Models*

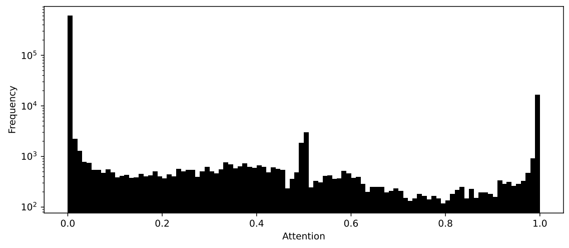
Despite their sophistication and large-scale pretraining, TSGFM achieves competitive results against TimeGPT and consistently outperforms TimesFM in network monitoring. Table VI summarizes performance on the network-specific permutations NET-A and NET-B, with values normalized to TSGFM for direct comparison. We evaluate the baselines on all datasets, while TSFMs were restricted to TELCO and

Abilene to assess their zero-shot capabilities in networking scenarios.

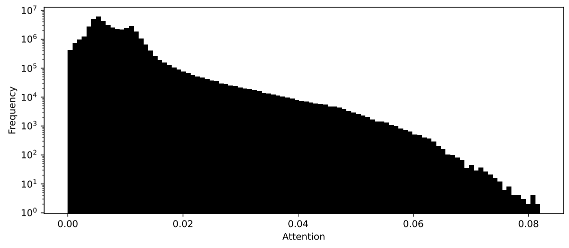
TimeGPT is tested under two different input strategies: *growing window* (GW), where the input window grows from the beginning of the time series until each prediction point, and *sliding window* (SW), where a fixed context length of $W = 32$ is used, identical to TSGFM and the other baselines. The GW configuration represents the standard usage of TimeGPT, where the model leverages an increasingly long historical context to generate each forecast. This setting aligns with the original design of TimeGPT, which assumes that more context leads to better predictive performance. However, our results reveal an interesting deviation from this expectation: the SW version of TimeGPT, with a fixed context length, consistently outperforms the default growing window setup across all metrics. This observation suggests that, for network monitoring tasks, longer historical windows do not necessarily translate into better performance. On the contrary, shorter, more recent contexts appear to be more informative for capturing the underlying dynamics of network traffic, which often exhibits short-term correlations and dynamic changes.

The sliding window version of TimeGPT achieves slightly better average performance than TSGFM (7% lower MAE and 8% lower RMSE on average), with robust results on NET-B. However, the gap is small and within the variability margin commonly observed in zero-shot settings. Notably, this marginal improvement comes at the cost of significantly higher model complexity and massive pretraining on over 100 billion timepoints. TimesFM, on the other hand, performs consistently worse than both TimeGPT and TSGFM across all metrics, with average errors more than 2.4× larger than those of TSGFM. The most likely explanation for this performance gap lies in the model's inherent design: although TimesFM is a foundation model capable of zero-shot forecasting, the authors explicitly demonstrate that it benefits significantly from fine-tuning on downstream tasks. In its out-of-the-box form, without additional adaptation, TimesFM struggles to generalize effectively to the network monitoring domain. This highlights a critical limitation of certain TSFMs – namely, that zero-shot capabilities do not always guarantee robust performance in specialized or high-variability settings unless accompanied by domain-specific adaptation.

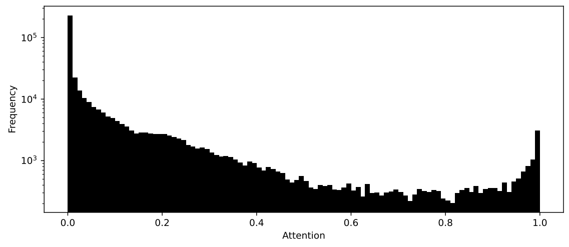
Finally, it is important to emphasize the scale discrepancy among the models. TSGFM operates with only 220K parameters, compared to the 500M+ parameters of TimesFM and the undisclosed but comparably large size of TimeGPT – in the same order of magnitude as TimesFM. This makes TSGFM approximately three orders of magnitude smaller than current TSFMs, while achieving similar or better forecasting accuracy in network-related tasks. This compact design enables low-latency inference and makes TSGFM viable for deployment in resource-constrained environments – a significant advantage in real-world network monitoring applications, where memory and computation budgets are often limited.



(a) Permutation A - Spain dataset



(b) Permutation B - Abilene dataset



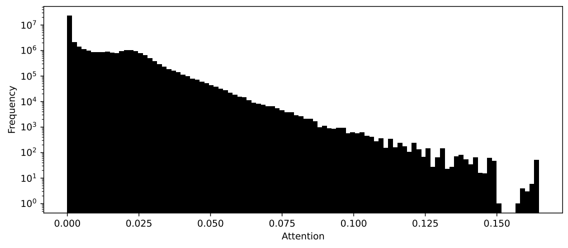
(c) Permutation C - TELCO dataset

Fig. 5: Histograms of spatial attention coefficients for the testing datasets in permutations A, B, and C using TSGFM.

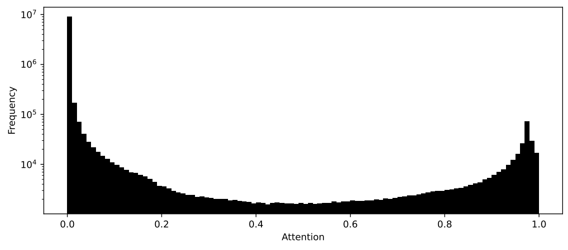
D. Analysis of the Self-Attention Mechanism

The analysis of the attention coefficients generated by the spatial attention mechanisms in TSGFM offers valuable insights into the importance – and potential redundancy – of fully connected spatial relationships. Figures 5a, 5b, and 5c show histograms of the spatial attention coefficients (referred to as *alphas*, cf. Section III-B) obtained during testing for permutations A, B, and C. In permutations A and C, the distributions exhibit a distinctly bimodal pattern, with attention values clustering near 0 and 1, suggesting that the model relies on a small set of key neighbors while disregarding the rest. This implies that the fully connected graph could be pruned to retain only the most relevant edges, simplifying the model without sacrificing accuracy.

In contrast, the spatial attention distribution in permutation B resembles a right-skewed distribution with no dominant attention peaks. The majority of coefficients lie below 0.1, indicating a broader spread of weakly influential neighbors. This suggests that, in this particular scenario, spatial dependencies are more diffuse and less informative, and no specific set of neighbors clearly dominates the spatial message passing. Overall, the variability in spatial attention patterns across



(a) Histogram of spatial attention coefficients.



(b) Histogram of temporal attention coefficients.

Fig. 6: Spatial and temporal attention coefficient distributions generated by TSGFM on the NET-B testing dataset.

TABLE VII: Testing performance of TSGFM under different spatial pruning levels (permutation NET-B, Abilene dataset).

Model	MAE	RMSE
TSGFM ($K = 0.2$)	1.11	1.00
TSGFM ($K = 0.4$)	1.04	1.00
TSGFM ($K = 0.6$)	0.92	0.99
TSGFM ($K = 0.8$)	0.99	1.00
TSGFM (no pruning)	1.00	1.00

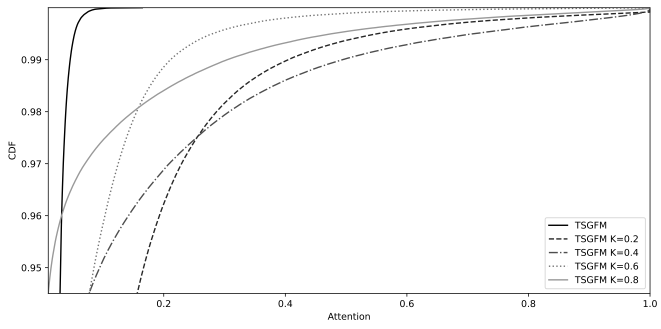


Fig. 7: CDF of spatial attention coefficients for TSGFM in the NET-B testing dataset.

datasets underscores a critical observation: a full-mesh spatial attention mechanism is not always necessary. In many cases, a sparser, more structured adjacency scheme may be sufficient, or even preferable, as it reduces computational complexity and memory usage without degrading performance.

We further examine this behavior in the NET-B permutation, based on network monitoring data. Figure 6a presents the spatial attention coefficients assigned by TSGFM during testing, while Figure 6b shows the corresponding temporal attention distribution. As previously observed in Figure 5b, the spatial attention values in NET-B remain relatively low, with only a few coefficients exceeding 0.16. This reinforces the notion that only a small subset of spatial neighbors meaningfully contribute to node updates. Notably, NET-B (Abilene) contains the largest number of time series among all datasets, and thus the densest spatial graph. Yet even here, the model effectively relies on a reduced number of meaningful connections, highlighting the potential to replace the full-mesh structure with a sparser graph, leading to reduced dimensionality and computational load.

Similarly, the histogram of temporal attention coefficients in Figure 6b reveals that most temporal edges are assigned low attention weights. This indicates that many past time steps contribute minimally to the final prediction, and could potentially be omitted without significant performance degradation. These findings suggest that both spatial and temporal dimensions contain redundant connections that can be safely pruned, opening the door to more efficient and scalable variants of TSGFM.

E. Ablation Study: TSGFM Spatial Pruning

Based on the analysis in Section IV-D, where we observed that a significant number of spatial edges may be redundant, we now perform an ablation study to quantify the impact of systematically pruning these edges on forecasting accuracy. Specifically, we explore a spatial edge pruning strategy aimed at reducing computational overhead while preserving predictive performance. The pruning approach is based on computing pairwise Euclidean distances between the hidden states of source and destination nodes in each spatial graph, retaining only the top K nearest neighbors for message passing. Here, K represents a fraction of the total number of spatial edges, and we vary this parameter to assess how different pruning levels affect model performance.

To evaluate the effectiveness of this edge pruning strategy, we tested TSGFM under varying pruning levels with $K \in [0.2, 0.4, 0.6, 0.8]$. Table VII summarizes the results for the NET-B scenario. The findings reveal that a substantial portion of the spatial edges – up to 40% – can be removed without a noticeable degradation in forecasting performance, highlighting the redundancy in densely connected spatial graphs. Beyond that limit, pruning starts impacting performance, suggesting that relevant information is being removed.

Figure 7 shows the cumulative distribution function (CDF) of the spatial attention coefficients in the NET-B scenario for TSGFM under different pruning levels. The curves shift progressively to the right as the pruning becomes more aggressive (i.e., as K decreases). This skew indicates that fewer spatial edges receive near-zero attention values, meaning that the model has fewer irrelevant spatial edges it can choose to ignore. Despite this, Table VII shows that the model maintains competitive forecasting performance even under high pruning levels such as $K = 0.2$. This suggests that a large number of spatial edges in the fully connected graph are not essential for accurate forecasting, as long as the model has a mechanism that can effectively prioritize the most important spatial adjacencies among the reduced set of available ones. Notably, the pruning strategy used is fully independent of the attention mechanism itself – based solely on pairwise Euclidean distances between node embeddings – ensuring that the evaluation is unbiased and does not artificially favor the attention patterns learned by the model.

V. Conclusions and Future Work

We introduced TSGFM, a GNN-based model for zero-shot multivariate time series forecasting. Its performance was evaluated across five publicly available datasets spanning finance, energy, and network monitoring, and compared against a diverse set of baselines, including MLPs, RNNs, and GNNs. Results show that TSGFM consistently matches or outperforms all baselines across multiple evaluation permutations. Notably, it achieves the best average performance overall and remains competitive with the strongest baseline, FourierGNN, while being significantly more stable on larger datasets such as Abilene. To assess network generalization, we introduced two network-specific scenarios, training only on non-network data. In both cases, TSGFM clearly outperforms all baselines, demonstrating strong transferability to network monitoring.

We also benchmarked TSGFM against two leading Time Series Foundation Models: TimeGPT and TimesFM. Despite being orders of magnitude smaller, TSGFM achieves performance on par with TimeGPT and clearly outperforms TimesFM, at least when no fine-tuning is considered – i.e., a pure zero-shot setting. These results demonstrate that massive pretraining and large-scale architectures are not strictly necessary for strong zero-shot forecasting, particularly in specialized domains like network monitoring.

Our analysis of spatial and temporal attention patterns further reveals that many graph edges contribute minimally to prediction performance. This motivates future work on adaptive graph construction, where edges are learned dynamically rather than fixed a priori. We plan to explore trainable edge mechanisms [28] that reduce redundancy, improve efficiency, and tailor the graph topology to each dataset's structure.

While this work provides strong initial evidence of TSGFM's versatility and robustness, future efforts will expand the evaluation to additional datasets, include more baselines, and investigate its deployment in real-world systems.

Acknowledgment

This work has been supported by the FWF Austrian Science Fund, Project reference I-6653, as well as by the Agencia Estatal de Investigacion, Project reference PCI2023-145974-2, as part of the EU CHIST-ERA-2022-SPiDDS-02 project *GRAPHS4SEC*. This work is also supported by the Catalan Institution for Research and Advanced Studies (ICREA Academia).

References

[1] A. D'Alconzo, I. Drago, A. Morichetta, M. Mellia, and P. Casas, "A survey on big data for network traffic monitoring and analysis," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 3, pp. 800–813, 2019.

[2] P.-W. Tsai, C.-W. Tsai, C.-W. Hsu, and C.-S. Yang, "Network monitoring in software-defined networking: A review," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3958–3969, 2018.

[3] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[4] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Phi. Trans, of the Royal Soc. A*, vol. 379, no. 2194, 2021.

[5] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint 2108.07258*, 2021.

[6] A. Garza, C. Challu, and M. Mergenthaler-Canseco, "TimeGPT-1," *arXiv preprint 2310.03589*, 2023.

[7] A. Das, W. Kong, R. Sen, and Y. Zhou, "A Decoder-only Foundation Model for Time-series Forecasting," *arXiv preprint 2310.10688*, 2024.

[8] H. Latif-Martínez, P. Casas, J. Suárez-Varela, A. Cabellos-Aparicio, and P. Barlet-Ros, "Tsgfm - towards a graph foundation model for time series analysis in network monitoring," in *2025 9th Network Traffic Measurement and Analysis Conference (TMA)*, 2025, pp. 1–4.

[9] Y. Liang, H. Wen, Y. Nie, Y. Jiang, M. Jin, D. Song, S. Pan, and Q. Wen, "Foundation models for time series analysis: A tutorial and survey," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 6555–6565.

[10] H. Xue and F. D. Salim, "PromptCast: A New Prompt-Based Learning Paradigm for Time Series Forecasting," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–14, 2023.

[11] N. Gruver, M. Finzi, S. Qiu, and A. G. Wilson, "Large Language Models Are Zero-Shot Time Series Forecasters," *arXiv preprint 2310.07820*, 2023.

[12] K. Rasul, A. Ashok, A. R. Williams, H. Ghonia, R. Bhagwatkar, A. Khorasani, M. J. D. Bayazi, G. Adamopoulos, R. Riachi, N. Hassen, M. Biloš, S. Garg, A. Schneider, N. Chapados, A. Drouin, V. Zantedeschi, Y. Nevmyvaka, and I. Rish, "Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting," 2024.

[13] G. G. González, P. Casas, E. Martínez, and A. Fernández, "On the quest for foundation generative-ai models for anomaly detection in time-series data," in *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2024, pp. 252–260.

[14] E. A. Nketiah, L. Chenlong, J. Yingchuan, and S. A. Aram, "Recurrent neural network modeling of multivariate time series and its application in temperature forecasting," *Plos one*, vol. 18, no. 5, p. e0285713, 2023.

[15] R. Pascanu, "On the difficulty of training recurrent neural networks," *arXiv preprint 1211.5063*, 2013.

[16] Q. Tan, M. Ye, B. Yang, S. Liu, A. J. Ma, T. C.-F. Yip, G. L.-H. Wong, and P. Yuen, "Data-gru: Dual-attention time-aware gated recurrent unit for irregular multivariate time series," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 01, 2020, pp. 930–937.

[17] J. Grigsby, Z. Wang, N. Nguyen, and Y. Qi, "Long-range transformers for dynamic spatiotemporal forecasting," *arXiv preprint 2109.12218*, 2021.

[18] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[19] K. Yi, Q. Zhang, W. Fan, H. He, L. Hu, P. Wang, N. An, L. Cao, and Z. Niu, "Fouriergnn: Rethinking multivariate time series forecasting from a pure graph perspective," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[20] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," *arXiv preprint 2002.07962*, 2020.

[21] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[22] N. J. Hana, "Energy consumption, generation, prices and weather dataset," 2021, accessed: 2025-03-28. [Online]. Available: https://www.kaggle.com/datasets/nicholasjhana/energy-consumption-generation-prices-and-weather

[23] G. Lai, "Exchange rate multivariate time series dataset," 2017, accessed: 2025-03-28. [Online]. Available: https://github.com/laiguokun/multivariate-time-series-data/blob/master/exchange_rate/exchange_rate.txt.gz

[24] Y. Zhang, "Abilene traffic matrix dataset," 2004, accessed: 2025-03-28. [Online]. Available: http://cs.utexas.edu/~yzhang/research/AbileneTM/

[25] G. García González, S. Martínez Tagliafico, A. Fernández, G. Gómez, J. Acuña, and P. Casas, "Telco," 2023. [Online]. Available: https://dx.doi.org/10.21227/skpg-0539

[26] H. Latif-Martínez, J. Suárez-Varela, A. Cabellos-Aparicio, and P. Barlet-Ros, "Gat-ad: Graph attention networks for contextual anomaly detection in network monitoring," *Com. & Ind. Eng.*, vol. 200, 2025.

[27] K. Yi, Q. Zhang, W. Fan, S. Wang, P. Wang, H. He, N. An, D. Lian, L. Cao, and Z. Niu, "Frequency-domain mlps are more effective learners in time series forecasting," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[28] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," *arXiv preprint 1906.00121*, 2019.