Reducing Smoothness with Expressive Memory Enhanced Hierarchical Graph Neural Networks

Anonymous Authors[1]

Abstract

Graphical forecasting models learn the structure of time series data via projecting onto a graph, with recent techniques capturing spatial-temporal associations between variables via edge weights. Hierarchical variants offer a distinct advantage by analysing the time series across multiple resolutions, making them particularly effective in tasks like global weather forecasting, where low-resolution variable interactions are significant. A critical challenge in hierarchical models is information loss during forward or backward passes through the hierarchy. We propose the Hierarchical Graph Flow (HiGFlow) network, which introduces a memory buffer variable of dynamic size to store previously seen information across variable resolutions. We theoretically show two key results: HiGFlow reduces smoothness when mapping onto new feature spaces in the hierarchy and non-strictly enhances the utility of message-passing by improving Weisfeiler-Lehman (WL) expressivity. Empirical results demonstrate that HiGFlow outperforms state-of-the-art baselines, including transformer models, by at least an average of 6.1% in MAE and 6.2% in RMSE. Code is available at `https://anonymous.4open.science/r/HiGFlow-6726/`.

1. Introduction

Graphical machine learning, which leverages the inherent topology of graph structures, has numerous applications in the physical sciences (Karniadakis et al., 2021; Wu et al., 2022; Cuomo et al., 2022; Xiong et al., 2021). Global weather forecasting, for instance, involves predicting a signal distributed across Earth's surface using time-varying simulated or observational data (Lam et al., 2023; Bauer et al., 2015). In this domain, Graph Neural Networks effectively model teleconnections at fixed spatial-temporal resolutions (Yi et al., 2024a). However, interactions between climate variables occur at a large scale across multiple spatial-temporal resolutions, leading to suboptimal modelling of underlying global dynamics.

Real-world graphs are, furthermore, often locally dense with significant bottlenecks, making it challenging for message-passing techniques to propagate information (Di Giovanni et al., 2023; Alon & Yahav, 2021) effectively. Hierarchical methods mitigate these issues by summarising global trends through the aggregation of related variables into a single descriptor (Wang et al., 2018; Zhang et al., 2017). In a graphical context, a clustering algorithm maps node groupings to super nodes, forming a coarse graph that captures non-local relationships in the time series (Oskarsson et al., 2024; Yang et al., 2021). However, current models struggle to preserve low-resolution information from top hierarchical levels due to challenges in mapping between feature-spaces of differing dimensionalities, reducing their descriptive power. This issue arises from two key factors: (1) the backward pass through the hierarchy fails to retain relational information between nodes and super nodes, leading to discordant low-resolution information when passing back down the hierarchy, and (2) reliance on unlearnable mappings to project clusters onto super nodes, causing to overfitting to non-coarse signals that *provably* increases feature space smoothness.

We propose the Hierarchical Graph Flow (HiGFlow) network, a novel hierarchical graph framework which addresses these issues. HiGFlow incorporates a memory buffer variable that remains consistent across all hierarchical resolutions. As a result, HiGFlow can propagate information from previously observed resolutions, functioning similarly to the hidden state in an LSTM (Hochreiter, 1997) for long temporal rollouts. Intuitively, this allows for greater information retention, increasing the effectiveness of stacking hierarchical layers. HiGFlow furthermore incorporates node embeddings into learnable mappings between nodes and super nodes, provably enhancing the use of contextual information when mapping across spaces of differing dimensionalities.

We study theoretically the effects of HiGFlow's increased accessibility to multi-resolution information. Specifically, we

[1]**AUTHORERR: Missing \icmlaffiliation.** . Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

prove three theoretical claims; **(i)** linear transition functions smooth the resulting feature space, thereby diminishing the effectiveness of subsequent operations (Rusch et al., 2023); **(ii)** models using highly non-linear transition functions, *e.g.* HiGFlow, limit smoothness, and **(iii)** when using a memory-buffer variable, adding hierarchical depth non-strictly increases expressivity under the Weisfeiler-Lehman (WL) test (Xu et al., 2018), making HiGFlow more expressive than a 1-WL hash function.

Our experimental results demonstrate that HiGFlow outperforms state-of-the-art forecasting models, including both graph-based models (Yi et al., 2024a) and transformer-based approaches (Nie et al., 2023; Liu et al., 2024) on all datasets. In addition, we show via sensitivity tests that increased non-linearity in mappings between node and super-node preserves model performance in a practical setting. Our contributions are as follows:

- We introduce a novel information retention mechanism that captures multi-resolution spatial-temporal relationships in a memory buffer, enhancing the effectiveness of stacking hierarchical levels.
- We prove that neural networks decrease smoothness of feature-spaces in the hierarchy, while linear mappings strictly increase it, and further empirically demonstrate this finding.
- We prove that HiGFlow, via the memory-buffer, non-strictly increases WL expressivity when stacking hierarchical layers, surpassing a 1-WL hash function. Empirical results show HiGFlow outperforming state-of-the-art models.

2. Preliminaries

We outline within this section essential concepts and definitions required for the rest of this work. Our framework is theoretically grounded in message-passing on a time-series graphical embedding.

2.1. Forecasting

We define a *time series* at time t as signal $\mathbf{x}(t|T) \in \mathbb{R}^{N\times T}$; a total of N spatial variables across a sliding window of T time steps, starting at t. The signal then exists over the interval $[t, t+T_{in}]$. The forecasting task takes an input signal $\mathbf{x}(t|T_{in})$ with buffer size T_{in}, and predicts the immediate future signal over T_{out} time steps $\mathbf{x}(t+T_{in}|T_{out})$.

In our work, we denote the T dimensional row-vector for variable i as $\mathbf{x}_i(t|T)$, and to ease notation, we additionally use $\mathbf{x}(t)$ to mean $\mathbf{x}(t|0) \in \mathbb{R}^{N\times 1}$.

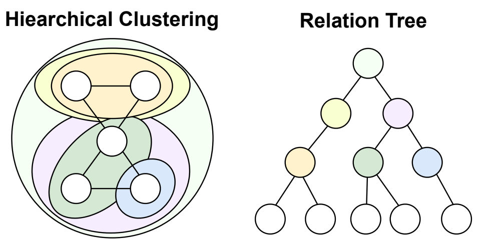

Figure 1. Hierarchical clustering on a graph and the corresponding relationships between regions within the graph.

2.2. Graph Embeddings

We enrich the descriptiveness of the feature space by embedding our input signal onto a hidden dimension of size D with weight vector $\mathbf{w}_e \in \mathbb{R}^{1\times D}$ and activation function σ:

$$\mathbf{h}(t|T_{in}) = \sigma(\mathbf{w}_e \cdot \mathbf{x}(t|T_{in})), \quad (1)$$

where we now have $\mathbf{h}(t|T_{in}) \in \mathbb{R}^{(N\times T_{in})\times D}$.

As shown in (Cao et al., 2020) and later (Yi et al., 2024a), we learn the implicit spatial-temporal relationships within $\mathbf{h}(t|T_{in})$. That is, we consider each $(i,t) \in [1,N]\times[1,T_{in}]$ to be a node within the vertex set of graph G, and find the weight of an edge between any two nodes via multi-headed self-attention (Vaswani, 2017); G describes spatial-temporal relationships. We enforce a hard constraint on variable relationships by truncating edge weights below a fixed threshold $\tau \in (0,1]$.

2.3. Message-Passing Graph Neural Networks

Message Passing Neural Networks (MPNNs) aggregate feature-vectors based on causal relationships described by the graph topology (Kipf & Welling, 2016; Xu et al., 2018; Hamilton et al., 2017; Veličković et al., 2018). Subsequently, a new representation of the feature vectors is found.

The general message-passing equation is defined using the aggregation function AGG and update function UP, which together map input $\mathbf{x}(t)$ to output $\mathbf{y}(t)$:

$$\mathbf{y}_i(t) = \text{UP}\left(\mathbf{h}_i(t), \text{AGG}\left(\{\{\mathbf{h}_j(t) \mid j \in \mathcal{N}(i)\}\}\right)\right), \quad (2)$$

where $\{\{\}\}$ denotes a multi-set, and $\mathcal{N}(i)$ represents the set of nodes in the 1-hop neighbourhood of node i.

3. Hierarchical Graph Flow Network

We introduce the Hierarchical Graph Flow (HiGFlow) network that, by iteratively coarsening an initial graph, views spatial-temporal interactions at multiple resolution scales. The framework incorporates learnable embedding and lifting maps that, according to Theorem 3.5, do not increase the smoothness of the resulting feature spaces. In contrast, we prove previous methods (Cini et al., 2023; Guo et al., 2021) increase smoothness; see Theorem 3.4. With the introduction of a persistent memory buffer variable, we construct a prediction with information at a mixture of resolutions.

3.1. Framework Overview

Figure 2 illustrates our overall approach. We model the topology of $\mathbf{x}(t|T_{in})$, coarsening its graph embedding via clique assignment to find intra-series dependencies across K progressively lower resolutions. See Figure 1 for an illustration.

Definition 3.1 (Abstract and Predecessor Graphs)**.** A graph $G_i = (V(G_i), E(G_i), W(G_i))$, consisting of a vertex set $V(G_i)$, an edge set $E(G_i)$, and edge weights $W(G_i)$, is called the *predecessor* of an *abstract* graph G_{i+1} if G_{i+1} is obtained by clustering on $V(G_i)$. In particular, if there exists a subset $S \subseteq V(G_i)$ that maps to a unique clique $m \in V_{i+1}$.

We define a transition function as a mapping between the feature spaces of a predecessor and its abstraction, in either direction:

Definition 3.2 (Transition Function)**.** A *transition function* $F : \mathcal{X}(G_n) \to \mathcal{X}(G_m)$ maps the feature space of G_n to that of G_m. When $n = i + 1$ and $m = i$, F is referred to as a *lifting map*. Conversely, when $n = i$ and $m = i + 1$, F is called an *embedding map*. For $j \in V(G_n)$ and subgraph $C_{j,m}$ of G_m, the following holds:

$$\mathbf{x}_{j,m}(t) = F(C_{j,n}). \tag{3}$$

In other hierarchical forecasting settings, such as those in (Cini et al., 2023; Guo et al., 2021), the feature space over abstract graphs is derived using a straightforward statistical mapping. In the following theorem, we demonstrate that when the transition function is linear, the Dirichlet energy decreases, resulting in a loss of information due to increased smoothness in the feature space.

Definition 3.3 (Dirichlet Energy)**.** The *Dirichlet energy* measures the similarity between feature-vectors within G_i:

$$\mathcal{D}(G_i) = \sum_{(u,v)\in E_i} \left\| \frac{\mathbf{x}_{u,i}}{d_{u,i}} - \frac{\mathbf{x}_{v,i}}{d_{v,i}} \right\|_2, \tag{4}$$

where $d_{u,i} = deg(u)$ is the degree of u in G_i.

Theorem 3.4. *The statistical transition function $M(C_{j,n}) = \sum_{u\in C_{j,n}} \mathbf{x}_{u,n}$ contracts the total Dirichlet energy when mapping from $\mathcal{X}(G_n)$ to $\mathcal{X}(G_m)$. In particular, for any $j \in V(G_n)$, then it is the case that the strong condition holds:*

$$\mathcal{D}(C_{j,m}) \leqslant \sum_{u\in V(C_{j,n})} \mathcal{D}(C_{u,n}). \tag{5}$$

We prove Theorem 3.4. in Appendix **A**. An increase in feature space smoothness is analogous to information loss, reducing the utility of subsequent operations (Rusch et al., 2023). Our framework introduces **learnable** embedding and lifting maps to transition between predecessor and abstract graphs, consequently enhancing the expressivity of message-passing on the graph. We define $\mathbf{h}{:}, i(t)$ and $\mathbf{u}{:}, i(t)$ as the features at depth i, derived through the embedding and lifting mappings, respectively. During the backward pass, we store information from all resolutions in a depth-persistent memory buffer, $\mathbf{y}_{:,i}(t)$. This mechanism enables information reuse and functions similarly to the gated behaviour of a GRU (Chung et al., 2014).

3.2. Embedding Nodes to Super Nodes

Embedding G_i onto its abstraction involves three steps: **(1)** partitioning the nodes of G_i into a collection of cliques $\mathcal{C}$ which disjointly cover $V(G_i)$, and gathering their corresponding feature-vectors from $\mathbf{h}_{:,i}(t)$, **(2)** aggregating and then mapping the gathered feature-vectors onto a super-node for each cluster, and **(3)** learning a new topological structure using self-attention based on the feature space of the abstract graph.

Clustering. We construct the collection $\mathcal{C}$ via spatial-temporal variable similarity inferred through the self-attention mechanism, emphasising plausible feature relationships rather than the feature vectors directly. We implement Graclus clustering (Dhillon et al., 2007), a greedy algorithm that selects nodes based on the highest edge weight, ensuring each cluster is simply connected.

Dimension Reduction. We map information contained within C_j via aggregating feature vectors onto a single global representation. To ensure diversity in the feature-space, we post-process the aggregation via a neural network $\mathbf{M}_{\phi_i}$ with parameters ϕ_i. Specifically, $\mathbf{M}_{\phi_i}$ enables the feature space $\mathcal{X}(G_{i+1})$ to become learnable. Our embedding is written as:

$$\bar{\mathbf{h}}_{j,i+1}(t) = \sum_{q\in C_j} \mathbf{h}_{q,i}(t), \tag{6}$$

$$\mathbf{h}_{:,i+1}(t) = \mathbf{M}_{\phi}\left(\bar{\mathbf{h}}_{:,i+1}(t) + \mathbf{W}_e^T \cdot \mathbf{v}\right). \tag{7}$$

Here, multiplication of node indices $\mathbf{v} = (1, \ldots, N_{i+1})^T$ by the linear map $\mathbf{W}_e \in \mathbb{R}^{N_{i+1}\times N_{i+1}}$ enables a topology-free linear embedding of the nodes of G_{i+1}, which assists in establishing a vague notion of locality within $\bar{\mathbf{h}}_{j,i+1}(t)$. Our implementation of $\mathbf{M}_{\phi_\mathrm{i}}$ transforms features only along the spatial dimension, significantly reducing the number of learnable parameters.

Theorem 3.5 shows that a highly non-linear transition function can come arbitrarily close to not smoothing the feature space, *e.g.*, when $\mathbf{M}_{\phi_i}$ is a neural network, considering the

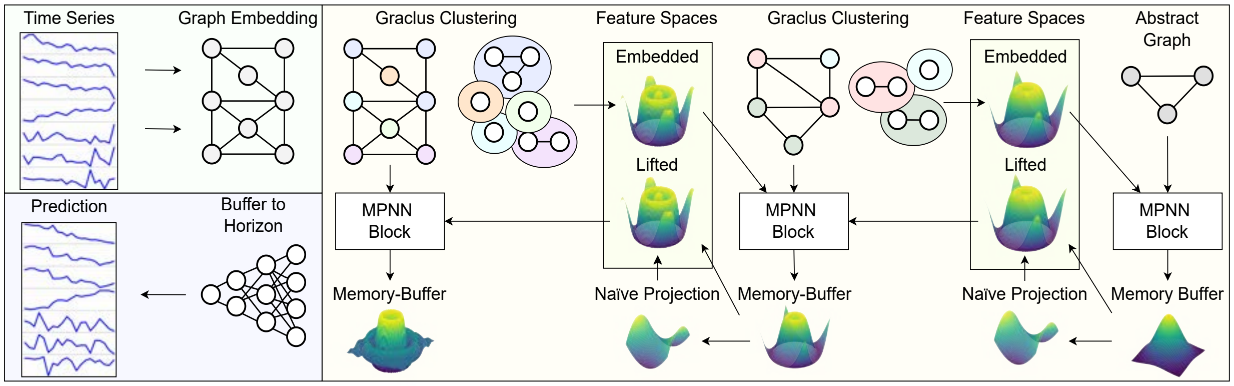

Figure 2. The framework begins by embedding a time series onto a graph, where edges represent relative intra-series associations. Nodes are greedily clustered based on edge weights. Information at any resolution is handled by embedding cluster feature vectors into a lower-resolution space or lifting lower-resolution data upward. A memory-buffer variable, incorporating the time-series topology, iteratively builds the prediction. Finally, a neural network maps the memory buffer to the prediction horizon.

universality theorem (Hornik et al., 1989). We defer the proof and additional details, to Appendix **A**.

Theorem 3.5. *Consider the truncated power-series expansion $p_B(\bar{\mathbf{x}}) = \sum_{b=1}^{B} \omega_b \cdot \bar{\mathbf{x}}^{(b)}$ of the function $f_\theta(\bar{\mathbf{x}})$, which contains B non-zero power-series coefficients ω_b. For statistical transition function $M(C) = \sum_{\mathbf{x}\in C} \mathbf{x}$, the non-linear transition function $F_\theta = f_\theta \circ M$ satisfies:*

$$\left| \mathcal{D}(C_{j,i+1}) - \sum_{u\in C_{j,i+1}} \mathcal{D}(C_{u,i}) \right| \mapsto 0 \text{ as } B \mapsto \infty. \quad (8)$$

Graph Topology. We account for changing feature relationships introduced by the mappings in Equations 6 and 7, such as strong correlations between nodes not captured by the edge weights or clique topology of the predecessor. Specifically, we recompute the graph topology using a new self-attention module.

3.3. Lifting Super Nodes to Nodes

Lifting reconstructs $\mathcal{X}(G_i)$ using the coarse upstream signal from low hierarchical levels. Consequently, the lifting operation is essential for mixing up and downstream signals $\mathbf{u}_{:,i}(t)$ and $\mathbf{h}_{:,i}(t)$ at hierarchical level i. Additionally, we introduce the *memory-buffer* $\mathbf{y}_{:,i}(t)$, a buffer that stores information from all previous abstract graph feature spaces. Unlike $\mathbf{u}_{:,i}(t)$, $\mathbf{y}_{:,i}(t)$ is a constrained with respect to the topology of all G_j, for $j < i$. Note that Theorem 3.5 also applies to lifting maps, as its proof in Appendix **A** does not require the unique assignment of a node to a cluster.

Naïve Encoding. To reconstruct the predecessor feature space from lower-resolution sources, a learnable encoding mitigates biases introduced during projection to a higher-dimensional space. The lifting map $\mathbf{L}_\gamma$, parameterised by γ, projects $\mathbf{y}_{:,i-1}(t) \in \mathbb{R}^{N_{i-1}}$ onto $\mathbf{e}_{:,i}(t) \in \mathbb{R}^{N_i}$:

$$\mathbf{e}_{:,i}(t) = \mathbf{L}_\gamma(\mathbf{y}_{:,i-1}(t)). \quad (9)$$

Inverse Cluster Map. We define the lifting transition function with parameters ξ as $\mathbf{U}_\xi$. This function reconstructs the feature space $\mathcal{X}(G_i)$ using signals from lower levels in the hierarchy, in contrast to the embedding map, which processes information from higher levels. Collectively, $\mathbf{U}_\xi$ enables the integration of mixed-resolution contextual information within the hierarchical framework. The lifting operation is expressed as:

$$\mathbf{u}_{:,i}(t) = \mathbf{U}_\xi(\mathbf{y}_{:,i-1}(t) + \mathbf{W}_l^T \cdot \mathbf{v}') + \mathbf{e}_{:,i}(t). \quad (10)$$

Here, $\mathbf{v}' = (1, \dots, N_i)^T$ and $\mathbf{W}_l \in \mathbb{R}^{N_i\times N_i}$ a linear embedding.

3.4. Memory-Buffer for Information Retention

We compute $\mathbf{y}_{:,i}(t)$ using an MPNN applied to G_i, where our choice of feature vectors integrates information across all topological resolutions.

Message Passing. We exploit the expressivity of MPNNs to propagate hidden states to higher resolutions by combining $\mathbf{h}_{:,i}(t)$ and $\mathbf{u}_{:,i}(t)$. The updated output state $\mathbf{y}_{:,i}(t)$ is defined as:

$$\mathbf{y}_{:,i}(t) = \text{MP}(\mathbf{h}_{:,i}(t)||\mathbf{u}_{:,i}(t)), \quad (11)$$

where MP represents the ubiquitous 1-hop message-passing framework from Equation 2, and $||$ denotes concatenation. The framework's final output is $\mathbf{y}_{:,1}(t)$. Theorem 3.6 demonstrates the effectiveness of including a memory buffer:

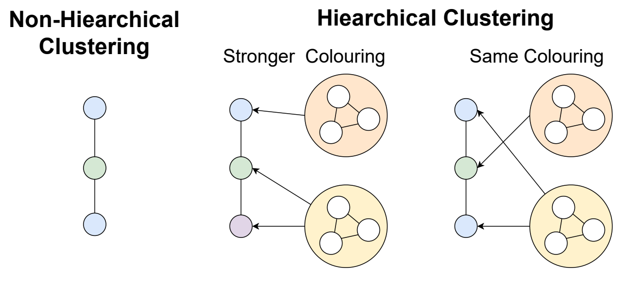

Figure 3. The benefit of a hierarchical framework when using a memory buffer. Hash function mappings of nodes with a cluster colour as an auxiliary argument are conditionally unique.

Theorem 3.6. *For a hierarchical graph framework with a memory buffer across N levels, the memory-enabled hash function $g_N(x_1, \ldots, x_N) = h_N(x_1, g_{N-1}(x_2, \ldots, x_N))$, where h_N is an injective function, induces a graph colouring $\mathcal{X}_{N:1}(G)$ satisfying:*

$$\left|\mathcal{X}^t_{N:1}(G)\right| \geqslant \left|\mathcal{X}^t_{N-1:1}(G)\right| \geqslant \cdots \geqslant \left|\mathcal{X}^t_{1:1}(G)\right|$$

In this context, $\mathcal{X}^t_{1:1}(G)$ is the colouring of G under some hash-function h bounded by the 1-WL isomorphism test. Theorem 3.6 demonstrates that storing information from all resolutions in a memory-buffer non-strictly increases the expressiveness of any hash function, an effect that stacks with hierarchical depth. We provide a proof of Theorem 3.6 in Appendix **B**.

4. Experiments

We evaluate the HiGFlow network against state-of-the-art forecasting models on eight real-world datasets and conduct sensitivity and ablation tests. Appendix **C** contains further sensitivity tests.

4.1. Experimental Set Up

Datasets. We select datasets from a diverse range of real-world forecasting tasks, including Electricity, ECG, Solar, and Traffic. Additionally, we use the ETTm1, ETTm2, ETTh1, and ETTh2 datasets from Wu et al. (2022). We split all datasets are split into 60%, 20%, and 20% for training, validation, and testing, respectively, with z-score normalisation.

Baselines. We compare our HiGFLow network with state-of-the-art Fourier transform orientated forecasting models, including FourierGNN (Yi et al., 2024a), FreTS, NLinear and DLinear (Yi et al., 2024b). Additionally, we evaluate against recent graphical forecasting models, such as StemGNN (Cao et al., 2020), and classic methods; LSTM (Hochreiter, 1997), and GRU (Chung et al., 2014). Given the well-established forecasting capabilities of transformer models, our analysis incorporates recent transformer-based architectures, such as Informer (Zhou et al., 2021) and Autoformer (Wu et al., 2022). We further include state-of-the-art transformer baselines from Liu et al. (2024); iTransformer, based on the Transformer model (Vaswani, 2017), and iFlashformer, an improvement on Flashformer (Hua et al., 2022).

Experiment Settings. We run our experiments using a single A100 NVIDIA GPU. We train all models using the Mean Squared Error Loss function and RMSProp optimiser, with all baseline model training parameters fixed to those used in Cao et al. (2020). For HiGFlow, we tuned the learning rate on the ETTh1 dataset, changing it from 10^{-4} to $5 \cdot 10^{-4}$. We report the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). As in (Cao et al., 2020; Yi et al., 2024a), we set the default input and prediction buffer sizes to 3 and 12 respectively. When implementing HiGFlow, we choose StemGNN (Cao et al., 2020) blocks as the message-passing mapping of Equation 11.

4.2. Evaluation

Comparison with Baselines. We evaluate the HiGFlow network against state-of-the-art baselines. Table 1 summarises the experimental results. Compared to non-transformer methods, HiGFlow achieves the lowest MAE and RMSE across all datasets, reducing MAE by 4–18%, with an average of 7%, while decreasing RMSE within 2–26% by a mean of 5%. Fourier methods, *e.g.* FreTS (Yi et al., 2024b), and Fourier-based GNN methods, such as FourierGNN (Yi et al., 2024a) and StemGNN (Cao et al., 2020), excel in capturing spatiotemporal relationships. However, they are restricted to viewing spatial-temporal relationships at a single resolution. Therefore, while the single-depth model HiGFlow(1), which incorporates elements from these methods, often achieves comparable accuracy, the multi-depth HiGFlow(*) model outperforms these models by obtaining lower error metrics.

Table 1 also compares HiGFlow with transformer baselines. While these models do not explicitly capture joint spatial-temporal correlations, they are excellent at extracting relationships along the temporal dimension. HiGFlow outperforms transformers such as Autoformer (Wu et al., 2021) and Informer (Zhou et al., 2021) across all datasets, achieving an average of at least 15% and 9% improvement in MAE and RMSE. However, compared to state-of-the-art forecasting models like iTransformer and iFlashformer (Liu et al., 2024), HiGFlow shows at worst only marginal gains on the ETT dataset series. In contrast, across the first four datasets, HiGFlow achieves more substantial improvements; on average 12% and 8% in MAE and 11% and 7%

Table 1. Comparison of MAE and RMSE across different datasets and models.

Models	ECG		Solar		Traffic		Electricity		ETTh1		ETTh2		ETTm1		ETTm2	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
LSTM	0.745	1.051	0.075	0.165	0.330	0.637	0.341	0.492	0.717	1.001	0.333	0.439	0.253	0.385	0.151	0.202
GRU	0.733	1.039	0.074	0.162	0.333	0.632	0.342	0.485	0.656	0.927	0.380	0.452	0.236	0.370	0.136	0.186
StemGNN	0.638	0.895	0.065	0.226	0.273	0.540	0.326	0.475	0.396	0.582	0.311	0.391	0.210	0.347	0.115	0.165
FourierGNN	0.644	0.909	0.051	0.128	0.278	0.551	0.266	0.398	0.375	0.567	0.165	0.232	0.202	0.341	0.104	0.157
FreTS	0.642	0.899	0.053	0.136	0.320	0.613	0.334	0.501	0.379	0.575	0.169	0.235	0.202	0.342	0.106	0.158
DLinear	0.712	0.976	0.115	0.219	0.465	0.774	0.381	0.549	0.415	0.605	0.173	0.238	0.203	0.345	0.104	0.158
NLinear	0.752	1.040	0.068	0.158	0.319	0.613	0.410	0.602	0.436	0.644	0.165	0.232	0.206	0.354	0.104	0.158
Autoformer	0.715	0.987	0.073	0.161	0.345	0.589	0.277	0.395	0.383	0.567	0.176	0.245	0.202	0.338	0.103	0.154
Informer	0.623	0.891	0.065	0.154	0.312	0.608	0.265	0.374	0.398	0.581	0.190	0.256	0.247	0.380	0.134	0.183
iFlashformer	0.717	1.010	0.063	0.149	0.283	0.557	0.288	0.415	0.349	0.532	0.156	0.219	0.215	0.359	0.115	0.162
iTransformer	0.641	0.908	0.049	0.129	0.298	0.600	0.303	0.467	0.364	0.567	0.158	0.224	0.202	0.344	0.105	0.159
HiGFlow(1)	0.648	0.910	0.055	0.133	0.305	0.593	0.347	0.524	0.391	0.596	0.173	0.239	0.204	0.347	0.108	0.160
HiGFlow(*)	**0.621**	**0.889**	**0.046**	**0.120**	**0.265**	**0.534**	**0.217**	**0.359**	**0.349**	**0.532**	**0.156**	**0.219**	**0.190**	**0.338**	**0.102**	**0.153**

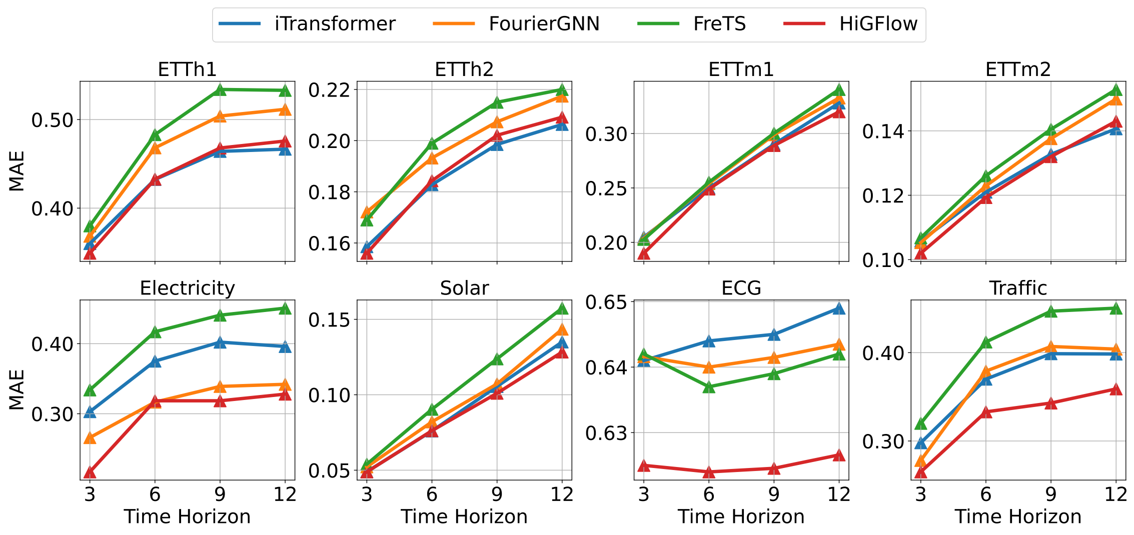

Figure 4. Model performance in terms of MAE while varying prediction buffer size.

in RMSE for iTransformer and iFlashformer, respectively. This underscores an area of key benefit for HiGFlow, and a relative limitation of transformers. These small dataset have a limited degree of spatial correlation, making dependence on temporal variations more pronounced. Therefore comparative performance between state-of-the-art transformer models and HiGFlow is higher than other other datasets with more variables.

Increasing Prediction Buffer Size. Figure 4 illustrates the effect of varying the prediction buffer size from 3 to 12 on MAE for HiGFlow, FourierGNN (Yi et al., 2024a), FreTS (Yi et al., 2024b) and iTransformer (Liu et al., 2024). HiGFlow generally outperforms across all datasets. Exceptions include comparable performance with FourierGNN on Electricity when the buffer size is 6, iTransformer achieving lower error metrics when the time buffer exceeds 6 on ETTh1 and ETTh2, and when the buffer is 12 on ETTm2. Non-transformer baseline models are Fourier-based, and therefore excel in extrapolation over long lead times, however, in some datasets, such as Traffic, Solar, and ETTh1, HiGFlow's relative gain increases with prediction buffer size. The average percentage gain across all datasets from a lead time of 6 onwards is 4.85%, 5.46% and 5.56%. Similar to Table 1, HiGFlow and the transformer baseline often show comparable performance on the ETT dataset, but HiGFlow exhibits a clear advantage on the remaining datasets.

Varying Hierarchical Depth. We analyse the joint relationship between hierarchical depth, the non-linearity of transition functions in embedding and lifting networks, and the performance of HiGFlow. We report MAE, which also doubles as a proxy for feature space smoothness (Rusch et al., 2023), and further demonstrates Theorems 3.4 and 3.5 in a real-world setting. Specifically, we compare the relative performance of four variations: HiGFlow with strictly linear transition functions, **HF-1**, which uses matrix multiplication (as in Cini et al. (2023)) to map clusters to supernodes; a depth-two network with one hidden layer, **HF-2**; a depth-three network with two hidden layers, **HF-3**; and the

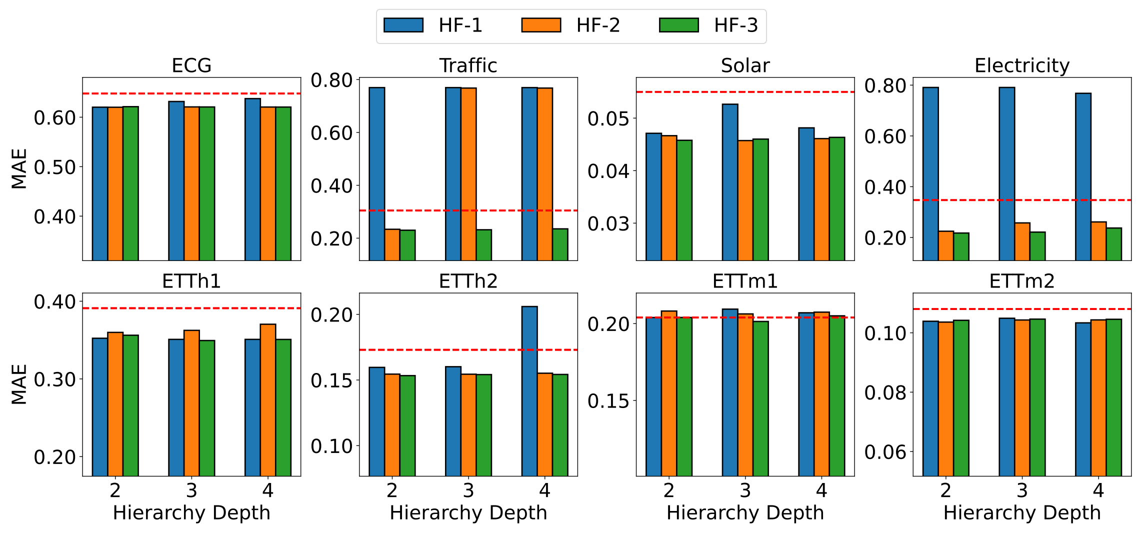

Figure 5. Mean Absolute Error (MAE) as a function of hierarchy depth, showing the impact of varying the non-linearity of the embedding and lifting networks in HiGFlow.

single-depth HiGFlow model in Table 1, called **HiGFlow(1)**. Recall that, by the universality theorem (Hornik et al., 1989), both HF-2 and HF-3 are considered universal approximators. Figure 5 presents these experiments.

In a setting where hierarchical depth is two, we observe that HF-1 generally achieves comparable performance to HF-2 and HF-3 across most datasets. Exceptions include Traffic and Electricity, where MAE increases substantially beyond HiGFlow(1), demonstrating the impact of information loss from a single round of transition function mappings on large datasets with complex spatiotemporal relationships. However, with greater hierarchical depth, HF-1 shows substantial increases in MAE on several datasets; ECG, ETTh2, ETTm1, and Solar. For ETTh2 and ETTm1, this greater than HiGFlow(1). An analogous statement can be made for HF-2 on Traffic and ETTh1, though performance on all other datasets tends to be more stable than HF-1.

Our results show that non-linearity is essential for achieving hierarchical depth, which aligns with our theoretical analysis. However, a linear transition function proves sufficient when the feature space distribution has limited support, as observed in the ETTh1 and ETTm2 datasets. In contrast, datasets like Traffic and Electricity demonstrate the necessity of non-linearity, where substantial increases in error metrics emerge even at a depth of two.

Parameters and Runtime Analysis. We evaluate the computational overhead of HiGFlow by analysing its training runtime and memory cost relative to the number of parameters. Table 2 summarises the results, showing the impact of varying HiGFlow's depth on memory usage and runtime across the ETTh1 and ECG datasets. Notably, HiGFlow exhibits a lower parameter count relative to all other baselines. Compared to FourierGNN (Yi et al., 2024a), which leverages temporal embeddings, HiGFlow achieves a reduction of 7.2% and 6.5% on ETTh1 and ECG, respectively. However, this efficiency diminishes as dataset size increases, with reductions of 7.2%, 7.1%, 6.9%, and 6.5% on ETTh1 compared to 6.5%, 6.3%, 5.9%, and 5.7% on ECG.

During the forward pass, certain operations in HiGFlow, such as finding the strongly connected components of a graph, are not easily vectorisable. This limitation, combined with existing overheads, results in longer training time, at least 1.5 times higher than other baselines. Additionally, this computational cost grows significantly with increased depth, scaling by factors of 2.3, 3.4, and 4.1, respectively.

Ablation Test. We conduct an ablation study to evaluate the empirical utility of embeddings in the Lifter and Embedding networks (Sections 3.3 and 3.2) on the ETTh1, ECG and Traffic datasets. Table 3 presents results comparing the HiGFlow network with configurations excluding node embeddings in the embedding network (**No/Embed-Domain**) or lifting network (**No/Lift-Domain**). Additional baselines include the absence of both lifting network domain and image shifts (**No/Lift-Embed**), domain embeddings in either network (**No/Domain-Embed**), or embeddings altogether (**No/Embed**). The results indicate that incorporating domain shifts in the embedding network consistently improves performance, as seen in the performance of No/Lift-Embed relative to No/Embed. However, to benefit from node embeddings in the domain of the lifting network, the naïve-encoding (image-shift) must also included, as demonstrated by comparing HiGFlow and No/Lift-Embed with No/Lift-Domain. Even then, performance relative to only embed-

Table 2. Training Time and Parameters for Various Models

	ETTh1		ECG	
	Parameters	Training (s/epoch)	Parameters	Training (s/epoch)
StemGNN	$194,294$	4.10 ± 0.25	$258,799$	1.75 ± 0.19
FreTS	$460,419$	9.33 ± 0.67	$460,419$	0.66 ± 0.16
FourierGNN	$182,307$	2.52 ± 0.36	$182,307$	0.84 ± 0.01
iTransformer	$6,313,987$	3.50 ± 0.63	$6,313,987$	2.56 ± 0.47
HiGFlow(1)	$30,291$	3.25 ± 0.29	$30,291$	2.74 ± 0.29
HiGFlow(2)	$85,531$	7.44 ± 0.38	$160,142$	8.81 ± 0.32
HiGFlow(3)	$139,483$	11.69 ± 0.25	$265,178$	13.04 ± 0.32
HiGFlow(4)	$192,596$	16.21 ± 0.52	$355,821$	16.03 ± 0.31

Table 3. Ablation study on ETTh1, ECG, and Traffic datasets. Results are reported for MAE and RMSE.

Dataset	Metric	No/Embed-Domain	No/Lift-Domain	No/Lift-Embed	No/Domain-Embed	No/Embed	HiGFlow
ETTh1	MAE	0.353	0.352	**0.345**	0.353	0.351	0.349
	RMSE	0.539	0.538	0.535	0.543	0.538	**0.532**
ECG	MAE	0.623	0.623	0.624	0.624	0.624	**0.621**
	RMSE	0.892	0.892	0.892	0.892	0.892	**0.889**
Traffic	MAE	0.264	0.268	**0.253**	0.264	0.273	0.265
	RMSE	0.537	0.538	**0.520**	0.536	0.545	0.534

ding network domain shifts may degrade in some instances, such as with the Traffic dataset.

5. Related Work

We review recent advancements in time series forecasting and hierarchical modelling.

5.1. Time Series Forecasting

Time-series to Graph Embeddings. Gao & Ribeiro (2022) prove theoretically that message-passing neural networks trained on GRU-embedded feature spaces are equivalent in expressivity to the Weisfeiler-Lehman (WL) test. Cao et al. (2020) propose a parameter-efficient method for embedding time-series data into graphs by mapping variables onto nodes, effectively capturing implicit temporal structures. In contrast, Yi et al. (2024a) consider both spatial and temporal interdependencies by constructing a hypervariate graph, where each variable at any time step is represented as a unique node and subsequently applying graph convolutions in Fourier space.

Fourier Methods for Forecasting. Fourier methods effectively transform temporal dimensions into scalar frequency values, serving as a form of attention mechanism (Yi et al., 2024b). Traditionally, these methods truncate high-frequency components beyond a specific threshold (Li et al., 2020). Recent studies, however, demonstrate that retaining high-frequency components improves the forecast's high-correlation lead time, resulting in more accurate long-term predictions (Lippe et al., 2023; Zhou et al., 2022).

5.2. Hierarchical Graph Methods

Pooling. Graph pooling methods aim to coarsen a graph, abstracting cliques of nodes onto super node representations (Lee et al., 2019; Zhang et al., 2019). Hierarchical pooling (Ying et al., 2018) offers a straightforward method to reduce granularity, allowing for tailored preservation of high-frequency components (Bianchi & Lachi, 2023).

Hierarchical Forecasting. Early methods (Wang et al., 2018) exploit relational structures by progressively building predictions from the lowest resolution level. Recent works focus on global or regional forecasting, such as (Oskarsson et al., 2024), employ non-homogeneous MPNNs to map between predefined mesh graphs. However, these techniques rely on apriori knowledge of graph structure.

6. Conclusion

We investigated the problem of information retention in hierarchical forecasting models, and proposed the HiGFlow network, which stores mixed-resolution information in a memory buffer persistent across all hierarchical levels. HiGFlow thereby enables consideration of spatiotemporal relationships at multiple resolutions. We prove three key theorems, demonstrating that HiGFlow reduces the smoothness of embedded and lifted spaces, and furthermore non-strictly increases WL expressivity as hierarchical layers are stacked. Our empirical evaluation corroborates this in practice and further highlights HiGFlow's ability to achieve lower MAE and RMSE relative to existing state-of-the-art graphical forecasting and transformer baselines.

References

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. *International Conference on Learning Representations*, 2021.

Bauer, P., Thorpe, A., and Brunet, G. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015.

Bianchi, F. M. and Lachi, V. The expressive power of pooling in graph neural networks. *Advances in neural information processing systems*, 36:71603–71618, 2023.

Cao, D., Wang, Y., Duan, J., Zhang, C., Zhu, X., Huang, C., Tong, Y., Xu, B., Bai, J., Tong, J., et al. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in Neural Information Processing Systems*, 33:17766–17778, 2020.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Advances in Neural Information Processing Systems*, 2014.

Cini, A., Mandic, D., and Alippi, C. Graph-based time series clustering for end-to-end hierarchical forecasting. *International Conference on Learning Representations*, 2023.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

Dhillon, I. S., Guan, Y., and Kulis, B. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11):1944–1957, 2007.

Di Giovanni, F., Giusti, L., Barbero, F., Luise, G., Lio, P., and Bronstein, M. M. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *International Conference on Machine Learning*, pp. 7865–7885. PMLR, 2023.

Gao, J. and Ribeiro, B. On the equivalence between temporal and static equivariant graph representations. In *International Conference on Machine Learning*, pp. 7052–7076. PMLR, 2022.

Guo, K., Hu, Y., Sun, Y., Qian, S., Gao, J., and Yin, B. Hierarchical graph convolution network for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 151–159, 2021.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 2017.

Hochreiter, S. Long short-term memory. *Neural Computation MIT-Press*, 1997.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Hua, W., Dai, Z., Liu, H., and Le, Q. Transformer quality in linear time. In *International conference on machine learning*, pp. 9099–9117. PMLR, 2022.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2016.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., Merose, A., Hoyer, S., Holland, G., Vinyals, O., Stott, J., Pritzel, A., Mohamed, S., and Battaglia, P. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023. doi: 10.1126/science.adi2336.

Lee, J., Lee, I., and Kang, J. Self-attention graph pooling. In *International conference on Machine Learning*, pp. 3734–3743. pmlr, 2019.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Lippe, P., Veeling, B., Perdikaris, P., Turner, R., and Brandstetter, J. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, 36:67398–67433, 2023.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M. itransformer: Inverted transformers are effective for time series forecasting. *International Conference on Learning Representations*, 2024.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. *International Conference on Learning Representations*, 2023.

Oskarsson, J., Landelius, T., Deisenroth, M. P., and Lindsten, F. Probabilistic weather forecasting with hierarchical graph neural networks. In *The Thirty-eighth Annual*

Conference on Neural Information Processing Systems, 2024. URL https://openreview.net/forum?id=wTIzpqX121.

Rusch, T. K., Bronstein, M. M., and Mishra, S. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wang, J., Wang, Z., Li, J., and Wu, J. Multilevel wavelet decomposition network for interpretable time series analysis. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2437–2446, 2018.

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

Wu, L., Cui, P., Pei, J., Zhao, L., and Guo, X. Graph neural networks: foundation, frontiers and applications. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4840–4841, 2022.

Xiong, J., Xiong, Z., Chen, K., Jiang, H., and Zheng, M. Graph neural networks for automated de novo drug design. *Drug discovery today*, 26(6):1382–1393, 2021.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *International Conference on Learning Representations*, 2018.

Yang, J., Zhao, P., Rong, Y., Yan, C., Li, C., Ma, H., and Huang, J. Hierarchical graph capsule network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10603–10611, 2021.

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

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

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

Zhang, L., Aggarwal, C., and Qi, G.-J. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2141–2149, 2017.

Zhang, Z., Bu, J., Ester, M., Zhang, J., Yao, C., Yu, Z., and Wang, C. Hierarchical graph pooling with structure learning. *Association for the Advancement of Artificial Intelligence*, 2019.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pp. 27268–27286. PMLR, 2022.