Graph-based Time Series Clustering for End-to-End Hierarchical Forecasting

Andrea Cini[*1], **Danilo Mandic**[2], **Cesare Alippi**[13]
[1] The Swiss AI Lab IDSIA USI-SUPSI, Università della Svizzera italiana,
[2] Imperial College London, [3] Politecnico di Milano

Abstract

Existing relationships among time series can be exploited as inductive biases in learning effective forecasting models. In hierarchical time series, relationships among subsets of sequences induce hard constraints (hierarchical inductive biases) on the predicted values. In this paper, we propose a graph-based methodology to unify relational and hierarchical inductive biases in the context of deep learning for time series forecasting. In particular, we model both types of relationships as dependencies in a pyramidal graph structure, with each pyramidal layer corresponding to a level of the hierarchy. By exploiting modern – trainable – graph pooling operators we show that the hierarchical structure, if not available as a prior, can be learned directly from data, thus obtaining cluster assignments aligned with the forecasting objective. A differentiable reconciliation stage is incorporated into the processing architecture, allowing hierarchical constraints to act both as an architectural bias as well as a regularization element for predictions. Simulation results on representative datasets show that the proposed method compares favorably against the state of the art.

1 Introduction

In most applications, collections of related time series can be organized and aggregated within a hierarchical structure [41]. One practical example is in forecasting (and monitoring) hourly energy consumption of a single household, as well as at the city, regional, and national levels [57]. Similar arguments can be made for forecasting photovoltaic production [62], financial time series [5], and the influx of tourists [4], to name a few relevant application domains. By exploiting aggregation constraints, forecasts at different levels can be combined to obtain predictions at different resolutions. Similarly, coherency constraints can be used to regularize forecasts obtained for the different levels in the process known as *forecasting reconciliation* (FR), which often results in more accurate predictions [41, 42, 60, 49]. Based on similar ideas, cluster-based aggregate forecasting methods learn to predict aggregates of clustered time series as an intermediate step for obtaining forecasts for the aggregate of all the considered signals [2, 28, 19].

Besides hierarchical constraints, forecasting models for correlated time series can take advantage of relational inductive biases to learn a single model with shared parameters to predict any subset of the time series in the collection [22]. Indeed, the combination of graph deep learning methods [6, 55, 15] and deep learning for time series [10] have been exploited in the context of spatiotemporal forecasting, obtaining state-of-the-art results (e.g., [54, 46, 61, 21]). Moreover, graph pooling operators [34, 11] enable graph neural network (GNN) architectures to learn how to cluster nodes and obtain hierarchical graph representations tailored to the task at hand. However, the application of learnable graph pooling operators and the combination of hierarchical and relational constraints are currently under-explored in graph-based spatiotemporal forecasting.

[*]Corresponding author (mail to andrea.cini@usi.ch), work done while at Imperial.

Temporal Graph Learning Workshop @ NeurIPS 2023, New Orleans.

To fill this void, this paper proposes a novel and comprehensive graph-based framework for hierarchical time series clustering and forecasting. Our approach unifies hierarchical time series processing, graph pooling operators, and graph-based neural forecasting methods. This results in a learning architecture for multi-step ahead forecasting operating at different levels of aggregation. Hierarchical and relational structures are embedded as inductive biases into the processing by exploiting neural message passing [33] and graph pooling [34] operators; a differentiable projection step guarantees coherency of the predictions w.r.t. the hierarchical structure. The proposed methodology, named Hierarchical Graph Predictor (HiGP) can propagate representations and ensure the coherency of the predictions with both pre-defined and learned hierarchical constraints. In the latter case, the forecast combination mechanism is learned in a self-supervised manner, by exploiting the accuracy of the forecasts as a learning signal and the graph topology as a regularization mechanism.

Our main novel contributions are as follows.

- The introduction of a methodology to embed hierarchical constraints as inductive biases in graph-based forecasting architectures [Sec. 3.1].
- A methodological framework, based on graph pooling, to learn a proper hierarchical structure directly from data by clustering the input time series [Sec. 3.2].
- An end-to-end learning architecture incorporating the above components in a time series forecasting model [Sec. 3.1, 3.2, 3.3].

HiGP is extensively validated on relevant benchmarks [Sec. 5]. Besides achieving state-of-the-art forecasting accuracy, we show that our approach can be used as a self-supervised architecture to learn meaningful cluster assignments.

2 Preliminaries

Graph-based spatiotemporal forecasting Consider a set of N time series; $\boldsymbol{x}_t^i \in \mathbb{R}$ indicates the value observed at time step t w.r.t. the i-th time series. The observation vector encompassing all the time series is analogously denoted by $\boldsymbol{X}_t \in \mathbb{R}^{N\times 1}$. Sequences of observations are indicated, e.g., as $\boldsymbol{X}_{t:t+T}$ where the index $t : t+T$ refers to the time interval $[t, t+T)$. Available covariates can be encoded into a matrix $\boldsymbol{U}_t \in \mathbb{R}^{N\times d_u}$. We assume the considered time series to be spatially correlated; i.e., time series are not independent, but are instead characterized by functional dependencies affecting the temporal evolution of the system. Pairwise relationships among time series are encoded within a weighted adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ which is constant over time; the resulting attributed graph at time t is denoted by the triple $\mathcal{G}_t = \langle \boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{A}\rangle$. The multi-step time series forecasting problem can then be modeled as the problem of predicting the H-step-ahead observations $\boldsymbol{X}_{t:t+H}$ given a window of past data $\mathcal{G}_{t-W:t}$ by minimizing some estimate of the forecasting error. We focus on point forecasts, i.e., we do not consider the uncertainty of the predictions.

Spatiotemporal graph neural networks (STGNNs) STGNNs are effective global time series forecasting models for the problem above. As a reference, we consider time-then-space (TTS) architectures [22, 32] with local learnable node embeddings [22] where the spatiotemporal data are processed by a temporal encoder followed by a stack of message-passing layers [33] accounting for "spatial" dependencies such that

$$\boldsymbol{h}_t^{i,0} = \textsc{SeqEnc}\left(\boldsymbol{x}_{t-W:t}^i, \boldsymbol{u}_{t-W:t}^i, \boldsymbol{v}^i\right), \quad \boldsymbol{h}_t^{i,l+1} = \rho^l\Big(\boldsymbol{h}_t^{i,l}, \underset{j\in\mathcal{N}(i)}{\textsc{Aggr}}\left\{\gamma^l\left(\boldsymbol{h}_t^{i,l}, \boldsymbol{h}_t^{j,l}, a_{ji}\right)\right\}\Big), \quad (1)$$

where $\boldsymbol{v}^i \subset \boldsymbol{V} \in \mathbb{R}^{N\times\, d_e}$ is the learnable node embedding associated with the i-th node, ρ^l and γ^l indicate update and message functions at the l-th layer, respectively, which are, e.g., implemented by multi-layer perceptrons (MLPs). $\textsc{SeqEnc}(\cdot)$ denotes a network encoding each input sequence along the temporal dimension, e.g., a recurrent neural network (RNN), $\textsc{Aggr}\{\cdot\}$ is a permutation invariant aggregation function and $\mathcal{N}(i)$ refers to the set of neighbors of the i-th node, each associated to an edge with weight a_{ji}. In the following, the shorthand $\boldsymbol{H}_t^{l+1} = \text{GNN}_l(\boldsymbol{H}_t^l, \boldsymbol{A})$ indicates the message-passing step w.r.t. the full node set and adjacency matrix $\boldsymbol{A}$. Predictions can then be obtained by using any decoder, e.g., an MLP followed by a linear readout for each prediction step.

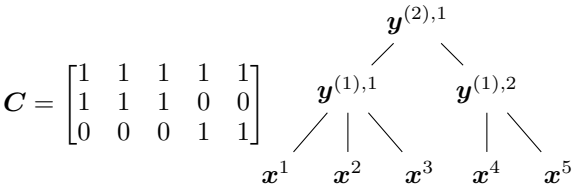


Figure 1: Example of hierarchical time series from [40].

Hierarchical time series In hierarchical settings, the set of raw time series is augmented by considering additional sequences obtained by progressively aggregating those at the level below, thus building a hierarchical structure. In particular, *bottom* observations (time series measurements) are denoted as $\boldsymbol{Y}_t^{(0)} = \boldsymbol{X}_t$, while $\boldsymbol{Y}_t^{(k)} \in \mathbb{R}^{N_k \times 1}$, with $k > 0$, indicates values of N_k time series obtained by aggregating (e.g., summing up) a partition of $\boldsymbol{Y}_t^{(k-1)}$; the full collection of both raw and aggregated observations is denoted by matrix $\boldsymbol{Y}_t \in \mathbb{R}^{M \times 1}$, with $M = \sum_{k=0}^{K} N_k$, obtained by stacking the $\boldsymbol{Y}_t^{(k)}$ matrices vertically in decreasing order w.r.t. index k. In general, the level of the hierarchy is denoted as a superscript between parentheses. The aggregation constraints can be encoded in an aggregation matrix $\boldsymbol{C} \in \{0,1\}^{(M-N)\times N}$ such that the i-th aggregate time series can be obtained as $\boldsymbol{y}_t^i = \sum_{j=1}^{N} c_{ij} \boldsymbol{x}_t^j$, i.e., by summing the bottom-level observations given the hierarchical constraints[2]. Given the above, the following relationships hold:

$$\boldsymbol{Y}_t = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{X}_t, \qquad\qquad \boldsymbol{Q}\boldsymbol{Y}_t = [\, \boldsymbol{I} \mid -\boldsymbol{C} \,] \, \boldsymbol{Y}_t = \boldsymbol{0}, \tag{2}$$

where $\boldsymbol{I}$ indicates an identity matrix of appropriate dimensions and $\mid$ the concatenation operator. Fig. 1 provides an example of a time series hierarchy with the associated aggregation matrix. A forecast $\widehat{\boldsymbol{Y}}_t$ is said to be *coherent* if the equality in Eq. 2 holds, i.e., if $\boldsymbol{Q}\widehat{\boldsymbol{Y}}_t = \boldsymbol{0}$.

3 Graph-based Hierarchical Clustering and Forecasting

This section presents our approach to graph-based hierarchical time series forecasting. We start by discussing how to incorporate the hierarchical structure of the problem into a graph-based neural architecture; then, we show how the hierarchical structure can be directly learned from data. Finally, we introduce an appropriate forecasting reconciliation mechanism to obtain forecasts coherent w.r.t. the learned hierarchy.

3.1 Graph-based Hierarchical Forecasting

Embedding the hierarchical structure into the processing requires defining proper neural operators. In particular, we aim at designing a *pyramidal* processing architecture where each layer corresponds to a level of the time series hierarchy and has its own topology, related to those at the adjacent layers by the hierarchical structure. To obtain such processing, operators have to be specified to control how information is propagated among and within the levels of the hierarchy; we exploit the connection to graph pooling for defining such operators within the *select, reduce, connect* (SRC) framework [34]. In particular, we use SRC building blocks as a high-level formalization of the operators performing clustering, aggregation, and graph rewiring w.r.t. each level of the pyramidal graph in HiGP. The three operators are defined as follows, by indicating as $\boldsymbol{H}_t^{(k)} \in \boldsymbol{R}^{N_k \times d_h}$ a feature matrix corresponding to k-th level of the hierarchy.

Select The selection operator $\textsc{Sel}(\,\cdot\,)$ outputs a mapping from input nodes to super nodes (i.e., clusters) given by the aggregation constraints at each level. The mapping can be encoded in a selection matrix $\textsc{Sel}(\boldsymbol{H}_t^{(k)}, \dots) = \boldsymbol{S}_k \in \{0,1\}^{N_{k-1} \times N_k}$ where s_{ij} is equal to 1 if and only if the i-th time series at level $k-1$ is mapped to the j-th aggregate at the k-th level. If the hierarchy is pre-defined, then the selection mechanism is given; conversely, learning a selection matrix is the key challenge for designing an end-to-end architecture and will be discussed in Sec. 3.2.

Reduce (and Lift) The reduction function $\textsc{Red}(\,\cdot\,)$ is used to aggregate node features and to propagate information from the k-th level to the adjacent upper level in the hierarchy. Reduction

[2]Note the index i does not refer to the level of the hierarchy but to the i-th element of the entire flattened collection $\boldsymbol{Y}_t$.

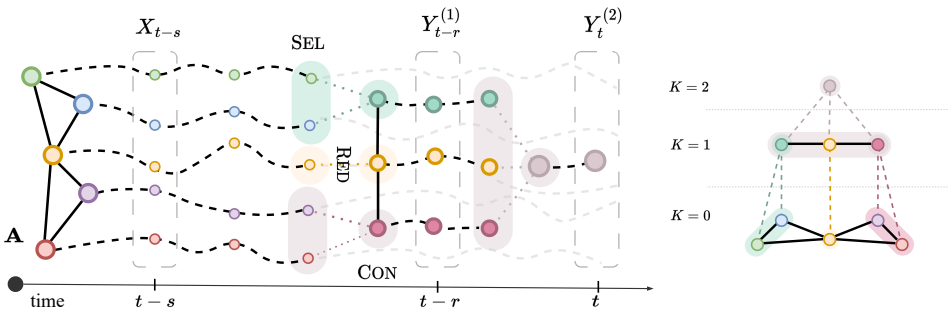


Figure 2: Time series with a hierarchical relational structure. **(Left)** Graphical representation of hierarchical time series with graph-side information; SRC operators allow for modeling relationships among the time series in the hierarchy. **(Right)** Pyramidal graph of both hierarchical and relational dependencies; each pair of levels can be seen as a bipartite graph.

can be obtained by summation, i.e., $\textsc{Red}(\boldsymbol{H}_t^{(k-1)}, \boldsymbol{S}^{(k)}) \doteq \boldsymbol{S}^{(k)^T} \boldsymbol{H}_t^{(k-1)}$, but other choices are possible. In practice, reduction is used in HiGP to propagate information along the pyramidal structure by pooling node representations and allowing for the implementation of an inter-level message-passing mechanism (see Eq. 5). Similarly, we can define a *lift* operator as $\textsc{Lift}(\boldsymbol{H}_t^{(k+1)}, \boldsymbol{S}^{(k)}) \doteq \boldsymbol{S}^{(k)} \boldsymbol{H}_t^{(k+1)}$, i.e., as an upsampling the pooled graph to the original size obtained by mapping each super-node back to the aggregated nodes.

Connect The connect operator $\textsc{Con}(\cdot)$ defines how the topology of the input graph is rewired after each aggregation step. There are several possible choices; we consider as default the rewiring where each pair of super-nodes is connected by an edge with weight obtained by summing the weights of edges between from one subset to the other, i.e., $\textsc{Con}(\boldsymbol{S}^{(k)}, \boldsymbol{A}^{(k-1)}) \doteq \boldsymbol{S}^{(k)^T} \boldsymbol{A}^{(k-1)} \boldsymbol{S}^{(k)}$, where $\boldsymbol{A}^{(k)}$ indicates the adjacency matrix w.r.t. k-th level.

The tree operators listed above can be used to design a neural processing architecture to match the inductive biases coming from the hierarchical structure. Fig. 2 provides a graphical illustration of how the above operators can be used to embed a hierarchical relational structure into the processing of raw spatiotemporal time series. We move from the reference architecture in Eq. 1 to a hierarchical TTS model operating as

$$\boldsymbol{h}_t^{(k),i,0} = \textsc{SeqEnc}^{(k)}\left(\boldsymbol{y}_{t-W:t}^{(k),i}, \boldsymbol{u}_{t-W:t}^{(k),i}, \boldsymbol{v}^{(k),i}\right), \qquad \text{Temporal enc.} \quad (3)$$

$$\boldsymbol{Z}_t^{(k),l} = \text{GNN}_l^{(k)}\left(\boldsymbol{H}_t^{(k),l}, \boldsymbol{A}^{(k)}\right), \qquad \text{Intra-level prop.} \quad (4)$$

$$\boldsymbol{H}_t^{(k),l+1} = \mu_l^{(k)}\left(\boldsymbol{Z}_t^{(k),l}, \underbrace{\boldsymbol{S}^{(k)^T} \boldsymbol{Z}_t^{(k-1),l}}_{\textsc{Red}^{(k)}}, \underbrace{\boldsymbol{S}^{(k)} \boldsymbol{Z}_t^{(k+1),l}}_{\textsc{Lift}^{(k)}}\right). \qquad \text{Inter-level prop.} \quad (5)$$

Equations 3–5 need further consideration to be fully appreciated. In the first place, matrix $\boldsymbol{H}_t^{(k),l}$ indicates representations w.r.t. the t-th time step obtained at the l-th message-passing layer for time series at the k-th level of the hierarchy (note the distinction between layers of message-passing and levels of the hierarchy). Indeed, compared to the model in Eq. 1, the hierarchical constraints add further structure to the processing. As shown in Eq.3, each time series is at first encoded along the temporal dimension by an encoder which can be either shared or different for each aggregation level. Then, representations are processed by a stack of layers propagating information within and among levels. As shown in Eq. 5, the representations are updated at each step by an update function $\mu_l^{(k)}$ (e.g., an MLP) taking as an input 1) the output $\boldsymbol{Z}_t^{(k),l}$ of a message-passing layer w.r.t. the graph topology at the k-th level (Eq. 4), 2) aggregated features from the level $k-1$ and 3) the features corresponding to each node's super-node obtained by lifting $\boldsymbol{H}_t^{(k+1),l}$. Learnable parameters may optionally be shared among the different levels of the hierarchy. Final predictions can be obtained by using an arbitrary readout, i.e., a standard MLP, and by training the model to minimize the forecasting error w.r.t. all the time series as

$$\hat{\boldsymbol{y}}_{t:t+H}^{(k),i} = \text{MLP}^{(k)}\left(\boldsymbol{h}_t^{(k),i,L}\right), \qquad \mathcal{L}\left(\widehat{\boldsymbol{Y}}_{t:t+H}, \boldsymbol{Y}_{t:t+H}\right) \doteq \left\|\widehat{\boldsymbol{Y}}_{t:t+H} - \boldsymbol{Y}_{t:t+H}\right\|_p^p, \quad (6)$$

where $\mathcal{L}(\cdot)$ indicates the loss function and p is equal to, e.g., 1 or 2. Note that, differently from standard spatiotemporal forecasting approaches, here the model is trained to make predictions for each level of the hierarchy. Representation at the different levels can capture patterns at different spatial scales, less apparent at fine-grained resolutions. Indeed, the aggregation and pooling operators increase the receptive field of each filter at each level of the hierarchy.

3.2 End-to-end Clustering and Forecasting

Learning a hierarchy and inherently a cluster-based forecasting architecture translates into learning a (differentiable) parametrization of the selection operator. For this task, we provide a general probabilistic framework, based on modeling cluster assignments as realizations of a parametrized categorical distribution. Then, we briefly discuss the applicability of standard graph pooling methods from the literature at the end of the section.

End-to-end clustering Similarly to popular dense trainable graph pooling operators [12, 64], we parametrize the selection operator with a score matrix $\boldsymbol{\Phi} \in \mathbb{R}^{N_{k-1} \times N_k}$, assigning to each node-cluster pair a score ϕ_{ij}. However, differently from previous works, we interpret such scores as (unnormalized) log-probabilities, such that

$$\boldsymbol{S}^{(k)} \sim P(\boldsymbol{S}^{(k)}_{ij} = 1) = \frac{e^{\phi^{(k)}_{ij}/\tau}}{\sum_j e^{\phi^{(k)}_{ij}/\tau}}, \qquad \boldsymbol{\Phi}^{(k)} = \mathcal{F}_{\boldsymbol{\psi}}\left(\boldsymbol{Y}^{(k-1)}_{t-W:t}, \boldsymbol{A}^{(k-1)}, \boldsymbol{V}^{(k-1)}\right), \tag{7}$$

where τ is a temperature hyperparameter, while $\mathcal{F}_{\boldsymbol{\psi}}(\cdot)$ indicates a generic trainable function with trainable parameters $\boldsymbol{\psi}$. The conditioning on the input window $\boldsymbol{Y}^{(k-1)}_{t-W:t}$ can be dropped to obtain static cluster assignments; furthermore, depending on the dimensionality of the problem, the score matrix might also be parametrized directly as $\boldsymbol{\Phi} = \boldsymbol{\psi}$. Node embeddings and aggregates for the k-th level are then obtained through the reduction operator as $\boldsymbol{V}^{(k)} = {\boldsymbol{S}^{(k)}}^T \boldsymbol{V}^{(k-1)}$ and $\boldsymbol{Y}^{(k)}_t = {\boldsymbol{S}^{(k)}}^T \boldsymbol{Y}^{(k-1)}_t$, respectively. To differentiate through the sampling of $\boldsymbol{S}^{(k)}$ we use the Gumbel softmax reparametrization trick [43, 48] followed by a discretization step to obtain hard cluster assignments via the straight-through gradient estimator [9]. In practice, τ is set to 1 at the beginning of training and is exponentially decayed towards 0 at each training step. The above discretization step avoids soft cluster assignments that could in practice lead to degenerate solutions given the loss in Eq. 6; indeed, uniform soft assignments are likely to minimize the variance of the aggregate time series and thus the prediction error at levels $k > 0$.

Graph-based regularization In order to take the graph structure into account when learning the assignments, we exploit the min-cut regularization introduced in [12], i.e., we add to the loss the term

$$\mathcal{L}^c\left(\boldsymbol{S}^{(k)}_\mu, \boldsymbol{A}^{(k-1)}\right) \doteq -\frac{\mathrm{Tr}\left({\boldsymbol{S}^{(k)}_\mu}^T \widetilde{\boldsymbol{A}}^{(k-1)} \boldsymbol{S}^{(k)}_\mu\right)}{\mathrm{Tr}\left({\boldsymbol{S}^{(k)}_\mu}^T \widetilde{\boldsymbol{D}}^{(k-1)} \boldsymbol{S}^{(k)}_\mu\right)} + \left\| \frac{{\boldsymbol{S}^{(k)}_\mu}^T \boldsymbol{S}^{(k)}_\mu}{\left\| {\boldsymbol{S}^{(k)}_\mu}^T \boldsymbol{S}^{(k)}_\mu \right\|_2} - \frac{\boldsymbol{I}}{\sqrt{N_k}} \right\|_2 \tag{8}$$

where $\boldsymbol{S}^{(k)}_\mu = \mathrm{softmax}\left(\boldsymbol{\Phi}^{(k)}\right)$, $\widetilde{\boldsymbol{D}}^{(k-1)}$ is the degree matrix of $\widetilde{\boldsymbol{A}}^{(k-1)} \doteq \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A}^{(k-1)} \boldsymbol{D}^{-\frac{1}{2}}$, i.e., of the symmetrically normalized adjacency matrix. The first term in the equation is a continuous relaxation of the min-cut problem [26] which rewards clusters that pool together connected components of the graph; the second term helps in preventing degenerate solutions by favoring orthogonal cluster assignments [12].

Training procedure The training objective identified in Eq. 6 entails that the cluster assignments are learned to minimize the forecasting error w.r.t. both the bottom time series and aggregates. As a result, time series are clustered s.t. aggregates at all levels are easier to predict, thus providing a meaningful self-supervised learning signal. Besides the clustering method described here, HiGP is compatible with any graph pooling approach from the literature (see Grattarola et al. [34]). We focus on assessing the proposed methodology in the context of forecasting, leaving to future works the in-depth study of the clusters that can be extracted within the proposed framework.

3.3 Forecast Reconciliation and Combination

As mentioned in Sec. 1, FR allows for obtaining coherent forecast w.r.t. the hierarchical constraints (Eq. 2). We follow Rangapuram et al. [52] and embed a (differentiable) reconciliation step within the architecture as a projection onto the subspace of coherent forecasts. Given the (trainable) selection matrices for each level of the hierarchy, the $\boldsymbol{Q}$ matrix (see Eq. 2) is obtained as

$$\boldsymbol{Q} = \Big[\boldsymbol{I} \Big| -\boldsymbol{C} \Big] = \Big[\boldsymbol{I} \Big| -\big[\textstyle\prod_{k=1}^{K} \boldsymbol{S}^{(k)} \;\big|\; \prod_{k=1}^{K-1} \boldsymbol{S}^{(k-i)} \;\big|\; \cdots \;\big|\; \boldsymbol{S}^{(1)} \big]^T \Big] \tag{9}$$

Then, following [52], given the raw prediction $\widehat{\boldsymbol{Y}}_t$, we can obtain the reconciled (coherent) forecasts $\overline{\boldsymbol{Y}}_t$ by projecting $\widehat{\boldsymbol{Y}}_t$ onto the space of coherent forecasts (i.e., the null space of $\boldsymbol{Q}$) as

$$\boldsymbol{P} \doteq \boldsymbol{I} - \boldsymbol{Q}^T \left(\boldsymbol{Q}\boldsymbol{Q}^T\right)^{-1} \boldsymbol{Q}, \qquad\qquad \overline{\boldsymbol{Y}}_t = \boldsymbol{P}\widehat{\boldsymbol{Y}}_t, \tag{10}$$

where $\boldsymbol{P}$ is obtained by solving the constrained optimization problem $\min_{\boldsymbol{Z}} \|\boldsymbol{Z} - \widehat{\boldsymbol{Y}}_t\|_2$ s.t. $\boldsymbol{Q}\boldsymbol{Z} = \boldsymbol{0}$. Model parameters are then learned by minimizing the loss $\mathcal{L}_f \doteq \mathcal{L}(\widehat{\boldsymbol{Y}}, \boldsymbol{Y}) + \mathcal{L}(\overline{\boldsymbol{Y}}, \boldsymbol{Y}) + \lambda\mathcal{L}(\overline{\boldsymbol{Y}}, \widehat{\boldsymbol{Y}})$ where we omitted the time indices. Note that minimizing the regularization term $\mathcal{L}(\overline{\boldsymbol{Y}}, \widehat{\boldsymbol{Y}})$ is equivalent to minimizing the distance between $\widehat{\boldsymbol{Y}}_{t:t+H}$ and the space of coherent forecasts. Unfortunately, computing the inverse of $\boldsymbol{Q}\boldsymbol{Q}^T$ incurs the cost $\mathcal{O}(M^3)$ in space and $\mathcal{O}(M^2)$ in time, which can be prohibitive for large time series collections. However, the solution is still practical for up to a few thousand nodes (most practical applications), and the regularization term, computed as $\mathcal{L}^{reg}(\widehat{\boldsymbol{Y}}, \lambda) \doteq \lambda\|\boldsymbol{Q}\widehat{\boldsymbol{Y}}\|_2$, can be used in the other cases as the only regularization. As for the clustering method, HiGP is orthogonal to many of the reconciliation approaches present in the literature. Nonetheless, only end-to-end methods will impact the learned cluster assignments. Future works could indeed study alternative reconciliation strategies.

4 Related works

Forecasting hierarchical time series is a widely studied problem [40, 41]. The standard approach consists in obtaining (possibly independent) forecasts for a subset of time series in the hierarchy in the first stage and then, in a separate step, reconciling and combining them to obtain (possibly coherent) predictions for the full hierarchy [41, 8, 60]. In particular, MinT [60] allows for obtaining optimal reconciled forecasts given a set of unbiased H-step-ahead predictions and the covariance matrix of the associated residuals. Analogous reconciliation methods have also been developed for probabilistic forecasts [59, 56, 25]. End-to-end methods have been instead proposed in the context of deep learning for time series forecasting [10] by exploiting the hierarchical structure either as an hard [52, 67] or soft constraint [50, 36]. Notably, Rangapuram et al. [52] incorporate the reconciliation step within the neural architecture as a differentiable convex optimization layer [1]. None of these methods consider relational dependencies among and within the levels of the hierarchical structure.

Graph-based deep architectures have become popular in time series processing [54, 46, 20, 44]. Conversely, differentiable graph pooling operators have been widely studied in GNN models for i.i.d. data [34, 11]. In particular, dense trainable operators [64, 12, 37] learn soft cluster assignment regularized by taking into account the graph structure. Sparse approaches instead produce hard cluster assignments usually learned by exploiting both the graph structure and a learned ranking on the nodes [7, 31]. Finally, non-trainable methods exploit a clustering of the nodes performed independently from the trained model [13, 27]. Pyramidal graph-based architectures have been exploited in reservoir computing [14]. With regards to forecasting architectures, hierarchical representations have been exploited in specific domains such as traffic analytics [65, 35, 39], air quality monitoring [16], financial time series [3], and pandemic forecasting [47]. However, most of the above methods rely on fixed cluster assignments; furthermore, none of them directly address the hierarchical time series forecasting problem by optimizing predictions at each level of the hierarchy and taking into account coherency constraints.

5 Experiments

HiGP is validated over several experimental settings considering time series forecasting benchmarks with no pre-defined hierarchical structure. We then provide a qualitative analysis of the learned time series clusters. Full experimental setup and additional results are provided in the appendix.

Table 1: Performance on benchmark datasets (5 runs). Best result in **bold**, runner-up underlined.

MODELS	Metr-LA		PeMS-Bay		CER		AQI	
	MAE	MRE (%)	MAE	MRE (%)	MAE	MRE (%)	MAE	MRE (%)
RNN	3.543 ± .005	6.134 ± .008	1.773 ± .001	2.839 ± .001	4.57 ± .00	21.65 ± .01	14.00 ± .03	21.84 ± .05
FC-RNN	3.566 ± .018	6.174 ± .031	2.305 ± .006	3.690 ± .009	7.13 ± .02	33.77 ± .11	18.33 ± .11	28.59 ± .18
GConv-TTS	3.071 ± .008	5.317 ± .015	1.584 ± .006	2.536 ± .009	4.12 ± .02	19.50 ± .08	12.30 ± .02	19.20 ± .03
Diff-TTS	3.012 ± .005	5.214 ± .008	1.569 ± .004	2.512 ± .006	4.11 ± .02	19.47 ± .11	12.24 ± .04	19.10 ± .05
Gated-TTS	3.027 ± .008	5.240 ± .013	1.582 ± .006	2.533 ± .009	4.13 ± .01	19.54 ± .06	12.07 ± .02	18.83 ± .03
GUNet-TTS	3.057 ± .016	5.292 ± .028	1.575 ± .006	2.522 ± .010	4.08 ± .02	19.32 ± .10	12.25 ± .03	19.11 ± .05
HiGP-TTS (C)	3.034 ± .008	5.253 ± .013	1.567 ± 0.005	2.508 ± 0.008	4.11 ± .07	19.45 ± .34	12.13 ± .02	18.92 ± .04
HiGP-TTS (D)	3.009 ± .005	5.209 ± .008	**1.566 ± .005**	**2.506 ± .008**	4.12 ± .06	19.49 ± .30	12.10 ± .01	18.88 ± .02
HiGP-TTS (G)	**3.007 ± .009**	**5.205 ± .016**	1.568 ± .008	2.510 ± .013	**4.05 ± .01**	**19.20 ± .03**	**12.02 ± .04**	**18.75 ± .06**

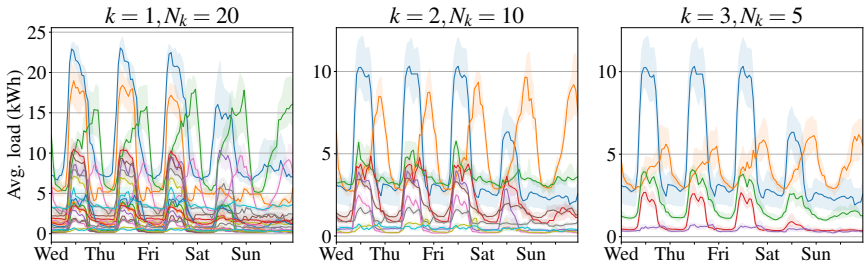
Benchmarks The considered benchmarks comprise of 4 datasets in total and include 2 datasets from the traffic forecasting literature (**METR-LA** and **PEMS-BAY** [46]), one dataset of air quality measurements (**AQI** [66]) and a collection of energy consumption profiles (**CER-E [24]**). Each dataset consists of time series with graph-side information; no prior hierarchical structure is given. We follow the setup of [22], by adopting the training splits and preprocessing protocols of previous works [22]. The performance metrics employed are the *mean absolute error* (MAE) and the *mean relative error* (MRE).

Baselines To carry out meaningful comparisons we select a reference TTS architecture [22, 32] (see Eq. 1) obtained by stacking 2 GNN layers after a node-wise temporal encoder implemented by an RNN. Learnable node embeddings [22] are concatenated to the input representation before both the recurrent encoder and after the message-passing layers. We compare the performance of different spatial propagation schemes from the state-of-the-art and commonly used in spatiotemporal forecasting architectures. In particular, the considered alternatives include the standard graph convolution (**GConv-TTS** [45]), the bidirectional diffusion convolution operator (**Diff-TTS** [46]), a more advanced gated message-passing scheme (**Gated-TTS** [22]), and the hierarchical Graph U-Net (**GUNet-TTS** [31]). We use a standard **GRU** [17] as sequence encoder for all the baselines. Finally, we denote by **FC-RNN** the baseline which considers the input sequences as a single multivariate time series and by **RNN** the global univariate model. The size of the hidden representations is selected for each dataset on the validation set (more details in the appendix), while the other hyperparameters are kept fixed among baselines. The **HiGP-TTS** model is implemented following the above template and Eq. 3–5. Notably, the only architectural difference w.r.t. the above baselines is the addition of a hierarchical propagation step after each message-passing layer. We learn directly (static) selection matrices by parametrizing the associated log-probabilities with tables of learnable parameters. As detailed below, we validate our approach by using different spatial propagation schemes. Furthermore, HiGP is trained end-to-end as to minimize the forecasting error w.r.t. the aggregates corresponding to the learned clusters.

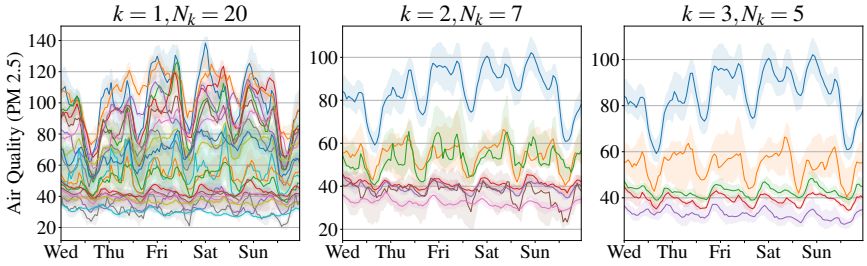
Results Tab. 1 show the results of the extensive empirical evaluation. We report HiGP forecasting accuracy w.r.t. 3 different message-passing schemes; in particular, (C), (D), and (G) indicate respectively the standard graph convolution, the diffusion convolution operator and the gated message-passing operator cited above. HiGP variants are among the best-performing methods in all the considered settings. Notably, adding hierarchical forecasting as secondary does not only act as self-supervision to learn cluster assignments but also provides a positive inductive bias that results – on average – in improved forecasting accuracy w.r.t. the flat architecture. Conversely, the GUNet baseline provides a comparison with a standard hierarchical message-passing architecture. Next, we perform an additional experiment by taking advantage of the popularity of METR-LA and

Table 2: Results on traffic datasets (5 runs). Best results in **bold**, runner-up underlined. MAE@X indicates the MAE w.r.t. the X-minutes-ahead forecast.

MODELS	METR-LA			PEMS-BAY		
	MAE@15	MAE@30	MAE@60	MAE@15	MAE@30	MAE@60
HiGP (T)	**2.68 ± .01**	**3.02 ± .01**	**3.40 ± .01**	1.31 ± .00	1.61 ± .00	**1.87 ± .00**
SGP	2.69 ± .00	3.05 ± .00	3.45 ± .00	**1.30 ± .00**	**1.60 ± .00**	1.88 ± .00
Gated-GN	2.72 ± .01	3.05 ± .01	3.44 ± .01	1.32 ± .00	1.63 ± .01	1.89 ± .01
GWNet	2.72 ± .01	3.10 ± .02	3.54 ± .03	1.31 ± .00	1.64 ± .01	1.94 ± .01
DCRNN	2.82 ± .00	3.23 ± .01	3.74 ± .01	1.36 ± .00	1.71 ± .00	2.08 ± .01



(a) CER-E dataset.



(b) AQI dataset.

Figure 3: Hierarchical cluster assignments learned by HiGP on 2 benchmark datasets. The models have been trained with a 5-level hierarchy and plots show, from left to right, the median for the clusters corresponding to levels from 1 to 3. The shaded areas correspond to 0.6 and 0.4 quantiles.

PEMS-BAY as traffic forecasting benchmarks and compare HiGP against specialized state-of-the-art architectures. We consider the following baselines from the literature: 1) **DCRNN** [46], i.e., a recurrent architecture; 2) **GWNet** [61], i.e., a popular graph convolutional model; 3) **Gated-GN** [53], i.e., a gated message-passing architecture operating on a fully connected graph; 4) **SGP** [21], i.e., a scalable STGNN based on a randomized encoder. In this context, we tuned the HiGP architecture by simply adding residual connections and using a deeper MLP decoder; the tuned architecture is designated as HiGP (T). Simulation results for multistep-ahead forecasting in the traffic datasets, provided in Tab. 2, show that HiGP achieves state-of-the-art accuracy. Appendix D.1 shows the results of an ablation study.

Cluster analysis We then analyze the clusters extracted by HiGP on the CER-E and AQI datasets. Ideally, we would like to cluster customers w.r.t. their consumption patterns in the first case, and to partition air quality monitoring stations w.r.t. the different dynamics and regions of the dataset. HiGP learns the cluster assignments by minimizing the forecasting error at each level of the hierarchy in an end-to-end fashion. This form of self-supervision rewards, then, the formation of clusters that result in aggregates that are easy to predict and regularized by the graph structure. In this context, we configure HiGP to learn 3 hierarchical cluster assignments and show the results of the training procedure in Fig. 3. Results show that the proposed approach is able to extract meaningful clusters with each level of the hierarchy corresponding to progressively smoother dynamics. Appendix D.2 provides a spatial visualization of the clustered nodes for the AQI dataset.

6 Conclusion

We introduced HiGP, unifying relational and hierarchical inductive biases in deep learning architectures for time series forecasting. HiGP has been designed to learn hard cluster assignments end-to-end, by taking the graph structure into account. Performance on relevant benchmarks supports the validity of the approach which, at the same time, has been shown able to learn meaningful hierarchical cluster assignments. Future works might study additional auxiliary objectives to provide additional supervision to the clustering and forecasting procedure, as well as consider alternative reconciliation mechanisms. Finally, future research could also apply HiGP-like methods to settings where the hierarchical constraints are pre-defined.

References

[1] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

[2] Carlos Alzate and Mathieu Sinn. Improved electricity load forecasting via kernel spectral clustering of smart meters. In *2013 IEEE 13th International Conference on Data Mining*, pages 943–948. IEEE, 2013.

[3] Arie Arya, Yao Lei Xu, Ljubisa Stankovic, and Danilo Mandic. Hierarchical graph learning for stock market prediction via a domain-aware graph pooling operator. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[4] George Athanasopoulos, Roman A Ahmed, and Rob J Hyndman. Hierarchical forecasts for australian domestic tourism. *International Journal of Forecasting*, 25(1):146–166, 2009.

[5] George Athanasopoulos, Puwasala Gamakumara, Anastasios Panagiotelis, Rob J Hyndman, and Mohamed Affan. Hierarchical forecasting. *Macroeconomic forecasting in the era of big data: Theory and practice*, pages 689–719, 2020.

[6] Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. A gentle introduction to deep learning for graphs. *Neural Networks*, 129:203–221, 2020.

[7] Davide Bacciu, Alessio Conte, and Francesco Landolfi. Generalizing downsampling from regular data to graphs. In *Thirty-Seventh AAAI Conference on Artificial Intelligence*, 2023.

[8] Souhaib Ben Taieb and Bonsoo Koo. Regularized regression for hierarchical forecasting without unbiasedness conditions. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1337–1347, 2019.

[9] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[10] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, François-Xavier Aubet, Laurent Callot, and Tim Januschowski. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Comput. Surv.*, 55(6), dec 2022. ISSN 0360-0300. doi: 10.1145/3533382. URL `https://doi.org/10.1145/3533382`.

[11] Filippo Maria Bianchi and Veronica Lachi. The expressive power of pooling in graph neural networks. *arXiv preprint arXiv:2304.01575*, 2023.

[12] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *International conference on machine learning*, pages 874–883. PMLR, 2020.

[13] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Hierarchical representation learning in graph neural networks with node decimation pooling. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5):2195–2207, 2020.

[14] Filippo Maria Bianchi, Claudio Gallicchio, and Alessio Micheli. Pyramidal reservoir graph neural network. *Neurocomputing*, 470:389–404, 2022.

[15] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[16] Ling Chen, Jiahui Xu, Binqing Wu, Yuntao Qian, Zhenhong Du, Yansheng Li, and Yongjun Zhang. Group-aware graph neural network for nationwide city air quality forecasting. *arXiv preprint arXiv:2108.12238*, 2021.

[17] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[18] Andrea Cini and Ivan Marisca. Torch Spatiotemporal, 3 2022. URL `https://github.com/TorchSpatiotemporal/tsl`.

[19] Andrea Cini, Slobodan Lukovic, and Cesare Alippi. Cluster-based aggregate load forecasting with deep neural networks. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[20] Andrea Cini, Ivan Marisca, and Cesare Alippi. Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks. In *International Conference on Learning Representations*, 2022.

[21] Andrea Cini, Ivan Marisca, Filippo Maria Bianchi, and Cesare Alippi. Scalable spatiotemporal graph neural networks. *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, 2023.

[22] Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi. Taming Local Effects in Graph-based Spatiotemporal Forecasting. *arXiv preprint arXiv:2302.04071*, 2023.

[23] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *International Conference on Learning Representations (ICLR)*, 2016.

[24] Commission for Energy Regulation. CER Smart Metering Project - Electricity Customer Behaviour Trial, 2009-2010 [dataset]. *Irish Social Science Data Archive. SN: 0012-00*, 2016. URL `https://www.ucd.ie/issda/data/commissionforenergyregulationcer`.

[25] Giorgio Corani, Dario Azzimonti, João PSC Augusto, and Marco Zaffalon. Probabilistic reconciliation of hierarchical forecast via Bayes' rule. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part III*, pages 211–226. Springer, 2021.

[26] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556, 2004.

[27] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11): 1944–1957, 2007.

[28] Fateme Fahiman, Sarah M Erfani, Sutharshan Rajasegarar, Marimuthu Palaniswami, and Christopher Leckie. Improving load forecasting based on deep learning and k-shape clustering. In *2017 international joint conference on neural networks (IJCNN)*, pages 4134–4141. IEEE, 2017.

[29] William Falcon and The PyTorch Lightning team. PyTorch Lightning, 3 2019. URL `https://github.com/PyTorchLightning/pytorch-lightning`.

[30] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[31] Hongyang Gao and Shuiwang Ji. Graph U-Nets. In *International Conference on Machine Learning*, pages 2083–2092. PMLR, 2019.

[32] Jianfei Gao and Bruno Ribeiro. On the equivalence between temporal and static equivariant graph representations. In *International Conference on Machine Learning*, pages 7052–7076. PMLR, 2022.

[33] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[34] Daniele Grattarola, Daniele Zambon, Filippo Bianchi, and Cesare Alippi. Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11, 2022. doi: 10.1109/TNNLS.2022.3190922.

[35] Kan Guo, Yongli Hu, Yanfeng Sun, Sean Qian, Junbin Gao, and Baocai Yin. Hierarchical graph convolution network for traffic forecasting. In *Proceedings of the 35th AAAI conference on artificial intelligence*, volume 35, pages 151–159, 2021.

[36] Xing Han, Sambarta Dasgupta, and Joydeep Ghosh. Simultaneously reconciled quantile forecasting of hierarchically related time series. In *International Conference on Artificial Intelligence and Statistics*, pages 190–198. PMLR, 2021.

[37] Jonas Berg Hansen and Filippo Maria Bianchi. Clustering with total variation graph neural networks. *arXiv preprint arXiv:2211.06218*, 2022.

[38] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[39] Luca Hermes, Barbara Hammer, Andrew Melnik, Riza Velioglu, Markus Vieth, and Malte Schilling. A Graph-based U-Net Model for Predicting Traffic in unseen Cities. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[40] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[41] Rob J Hyndman, Roman A Ahmed, George Athanasopoulos, and Han Lin Shang. Optimal combination forecasts for hierarchical time series. *Computational statistics & data analysis*, 55 (9):2579–2589, 2011.

[42] Rob J Hyndman, Alan J Lee, and Earo Wang. Fast computation of reconciled forecasts for hierarchical and grouped time series. *Computational statistics & data analysis*, 97:16–32, 2016.

[43] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

[44] Ming Jin, Huan Yee Koh, Qingsong Wen, Daniele Zambon, Cesare Alippi, Geoffrey I Webb, Irwin King, and Shirui Pan. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *arXiv preprint arXiv:2307.03759*, 2023.

[45] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[46] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

[47] Yihong Ma, Patrick Gerard, Yijun Tian, Zhichun Guo, and Nitesh V Chawla. Hierarchical spatio-temporal graph neural networks for pandemic forecasting. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1481–1490, 2022.

[48] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

[49] Anastasios Panagiotelis, Puwasala Gamakumara, George Athanasopoulos, and Rob J Hyndman. Probabilistic forecast reconciliation: Properties, evaluation and score optimisation. *European Journal of Operational Research*, 306(2):693–706, 2023.

[50] Biswajit Paria, Rajat Sen, Amr Ahmed, and Abhimanyu Das. Hierarchically regularized deep forecasting. *arXiv preprint arXiv:2106.07630*, 2021.

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[52] Syama Sundar Rangapuram, Lucien D Werner, Konstantinos Benidis, Pedro Mercado, Jan Gasthaus, and Tim Januschowski. End-to-end learning of coherent probabilistic forecasts for hierarchical time series. In *International Conference on Machine Learning*, pages 8832–8843. PMLR, 2021.

[53] Victor Garcia Satorras, Syama Sundar Rangapuram, and Tim Januschowski. Multivariate time series forecasting with latent graph inference. *arXiv preprint arXiv:2203.03423*, 2022.

[54] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, pages 362–373. Springer, 2018.

[55] Ljubiša Stanković, Danilo Mandic, Miloš Daković, Miloš Brajović, Bruno Scalzo, Shengxi Li, Anthony G Constantinides, et al. Data analytics on graphs part III: machine learning on graphs, from graph topology to applications. *Foundations and Trends® in Machine Learning*, 13(4): 332–530, 2020.

[56] Souhaib Ben Taieb, James W Taylor, and Rob J Hyndman. Coherent probabilistic forecasts for hierarchical time series. In *International Conference on Machine Learning*, pages 3348–3357. PMLR, 2017.

[57] Souhaib Ben Taieb, James W Taylor, and Rob J Hyndman. Hierarchical probabilistic forecasting of electricity demand with smart meter data. *Journal of the American Statistical Association*, 116(533):27–43, 2021.

[58] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[59] Shanika L Wickramasuriya. Probabilistic forecast reconciliation under the Gaussian framework. *Journal of Business & Economic Statistics*, pages 1–14, 2023.

[60] Shanika L Wickramasuriya, George Athanasopoulos, and Rob J Hyndman. Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. *Journal of the American Statistical Association*, 114(526):804–819, 2019.

[61] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 1907–1913, 2019.

[62] Dazhi Yang, Hao Quan, Vahid R Disfani, and Licheng Liu. Reconciling solar forecasts: Geographical hierarchy. *Solar Energy*, 146:276–286, 2017.

[63] Xiuwen Yi, Yu Zheng, Junbo Zhang, and Tianrui Li. St-mvl: filling missing values in geo-sensory time series data. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016.

[64] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

[65] Bing Yu, Haoteng Yin, and Zhanxing Zhu. ST-Unet: A spatio-temporal U-network for graph-structured time series modeling. *arXiv preprint arXiv:1903.05631*, 2019.

[66] Yu Zheng, Xiuwen Yi, Ming Li, Ruiyuan Li, Zhangqing Shan, Eric Chang, and Tianrui Li. Forecasting fine-grained air quality based on big data. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2267–2276, 2015.

[67] Fan Zhou, Chen Pan, Lintao Ma, Yu Liu, Shiyu Wang, James Zhang, Xinxin Zhu, Xuanwei Hu, Yunhua Hu, Yangfei Zheng, et al. Sloth: Structured learning and task-based optimization for time series forecasting on hierarchies. *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, 2023.

Appendix

This appendix provides additional details on the setup and datasets used for the experiments presented in the paper as well as additional empirical results.

A Hardware and software platforms

Experimental setup and baselines have been developed with Python [58] by relying on the following open-source libraries:

- numpy [38];
- PyTorch [51];
- PyTorch Lightning [29];
- PyTorch Geometric [30];
- Torch Spatiotemporal [18].

Experiments were run on a server equipped with AMD EPYC 7513 CPUs and NVIDIA RTX A5000 GPUs. The code for the computational experiments will be made publicly available upon publication at an archival venue.

B Datasets

Table 3: Statistics of datasets used in the experiments.

DATASETS	Time steps	Nodes	Edges	Type
METR-LA	34,272	207	1515	Directed
PEMS-BAY	52,128	325	2369	Directed
CER-E	25,728	485	4365	Directed
AQI	8,760	437	2699	Undirected

We use the same spatiotemporal forecasting benchmarks of [22], which consist of the following datasets.

METR-LA The METR-LA dataset [46] consists of measurements from loop detectors in the Los Angeles County Highway.

PEMS-BAY The PEMS-BAY dataset [46], contains traffic speed measurements analogous to those of METR-LA and acquired in the San Francisco Bay Area.

CER-E The CER-E dataset [24] consists of a collection of load profiles (i.e., energy consumption measurements) aggregated into 30-minutes intervals, recorded by 485 smart meters in Irish small and medium-sized enterprises. The dataset has been introduced as a benchmark for graph-based time series processing in [20].

AQI The AQI dataset [66] collects hourly measurements of the PM2.5 pollutant from 437 air quality monitoring stations spread over 43 Chinese cities. Similarly to CER-E, AQI has been introduced as a benchmark for graph-based processing in [20].

All of the above datasets are either openly available (METR-LA, PEMS-BAY, AQI) or obtainable free of charge for research purposes (CER-E[3]). Tab. 3 provides relevant statistics on the considered datasets. For each dataset, we obtain the corresponding adjacency matrix and exogenous variables by following previous works [20, 46, 22]. Following [22], datasets are split into windows of W time steps and the models are trained to predict the subsequent H observations. Window size W and forecasting horizon H are respectively set as $W = 12$ and $H = 12$ for METR-LA and PEMS-BAY, $W = 48, H = 6$ for CER-E, and $W = 24, H = 3$ for AQI. Training, validation, and testing data are respectively obtained with a $70\%/10\%/20\%$ sequential split. Conversely, for AQI, we use the same data splits of [63].

C Baselines and hyperparameters

Reference architectures As discussed in Sec. 5, the main empirical results of the paper (Tab. 1), were obtained by considering, for all the baselines, a template TTS architecture which can be schematically described as follows:

$$\boldsymbol{h}_t^{i,0} = \text{GRU}\left(\boldsymbol{x}_{t-W:t}^i, \boldsymbol{u}_{t-W:t}^i, \boldsymbol{v}^i\right), \tag{11}$$

$$\boldsymbol{H}_t^1 = \text{GNN}_1\left(\boldsymbol{H}_t^0, \boldsymbol{A}\right), \tag{12}$$

$$\boldsymbol{H}_t^2 = \text{GNN}_2\left(\boldsymbol{H}_t^1, \boldsymbol{A}\right), \tag{13}$$

$$\hat{\boldsymbol{x}}_{t+h}^i = \boldsymbol{W}_h \xi\left(\boldsymbol{W}_{fc}\left[\boldsymbol{h}_t^{i,2} | \boldsymbol{v}_i\right] + \boldsymbol{b}_{fc}\right) + \boldsymbol{b}_h, \qquad h = 0, 1, \ldots, H-1, \tag{14}$$

with $\xi(\cdot)$ being the ELU activation function [23], $\boldsymbol{W}_h \in \mathbb{R}^{1 \times d_h}$, $\boldsymbol{W}_h \in \mathbb{R}^{d_h \times d_h}$, $\boldsymbol{b}_h \in \mathbb{R}$, $\boldsymbol{b}_{fc} \in \mathbb{R}^{d_h}$ denoting learnable weights, *GRU* and *GNN* indicating respectively a gated recurrent temporal encoder [17] and a generic message-passing layer (implemented differently for each baseline). For HiGP, the template was modified to account for the hierarchical structure as discussed in Sec. 3.1. Similarly, for the GUNet baselines the template was modified to take into account the pooling and lifting operations. For the tuned version of HiGP we simply added skip connections and used a deeper readout.

[3] `https://www.ucd.ie/issda/data/commissionforenergyregulationcer/`

Table 4: Ablation study on traffic datasets (5 runs). MAE@X indicates the MAE w.r.t. the X-minutes-ahead forecast. *No rel. prop.* indicates the removal of the message-passing layers, while *No hier. prop.* denotes the baseline with no propagation of representation through the hierarchy, i.e., the removal of reduce, lift, and reconciliation operators. Results show that both relational and hierarchical architectural biases improve forecasting accuracy.

MODELS	METR-LA			PEMS-BAY		
	MAE@15	MAE@30	MAE@60	MAE@15	MAE@30	MAE@60
HiGP	**2.68** ± .01	**3.02** ± .01	**3.40** ± .01	**1.31** ± .00	**1.61** ± .00	**1.87** ± .00
No rel. prop.	2.80 ± .01	3.14 ± .01	3.47 ± .02	1.32 ± .00	1.63 ± .00	1.88 ± .01
No hier. prop.	**2.68** ± .01	3.03 ± .02	3.43 ± .02	**1.31** ± .00	1.63 ± .00	1.89 ± .00

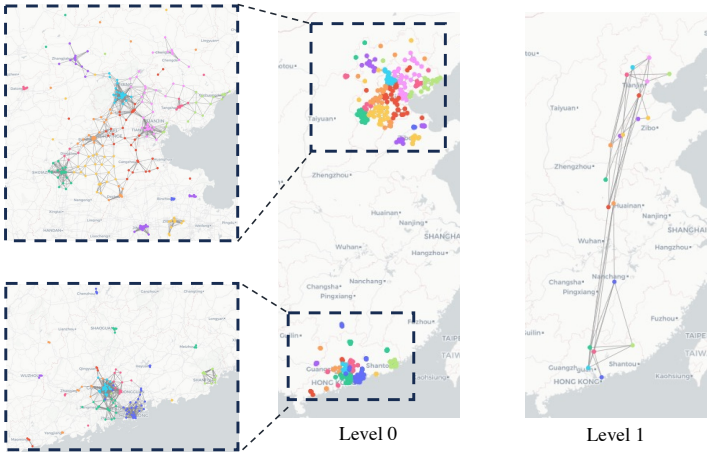


Figure 4: Visualizations of clustered nodes in the AQI dataset.

Hyperparameters and training details We trained each model with early stopping on the validation set and a batch size of 64 samples for a maximum of 200 epochs each of 300 batches maximum. We used the Adam optimizer with an initial learning rate of 0.003 reduced by a factor $\gamma = 0.25$ every 50 epochs. The number of neurons d_h in the layers of each model was set to 64 or 32 based on the validation error on each dataset. For HiGP, the regularization coefficient λ was tuned and set to 0.25 based on the validation error on the METR-LA dataset and simply rescaled for the other datasets to take into account the different magnitude of the input. As discussed in Sec. 5, we used a 3-level hierarchy with 20 super-nodes in the middle level and a single super-node (the total aggregate) at the top level. Intra-level spatial propagation was performed only at the base level. For the Diff-TTS baseline, the order of the diffusion convolution was set to $k = 2$, while the pooling factor for the GUNet was set to $p = 0.1$. For what concerns the experimental results in Tab. 2, for each baseline we used the hyperparameters of the original papers and the open-source implementation provided by [21]. Hyperparameters for HiGP (T) were obtained by tuning the model on the validation set of both datasets separately.

D Additional results

D.1 Ablation study

We carry out an ablation study of the proposed method considering the tuned architecture employed on traffic datasets. In particular, we train 2 variants of the model; the first variant is characterized by the removal of all the message-passing layers, while the second does not perform any propagation of the learned representations through the learned hierarchy. Results (Tab. 4) show that both designs have a significant impact on forecasting accuracy.

D.2 Cluster analysis

Fig. 4 shows a visualization of the learned clusters for the Air Quality dataset to complement the one provided in the paper (Fig. 3b). In particular, the figure shows the geographical location of each sensor, the partitioning of the network into clusters, and a visualization of the pooled graph for the first level of the hierarchy.