Hierarchically Regularized Deep Forecasting

Biswajit Paria[†][*] **Rajat Sen**[‡] **Amr Ahmed**[‡] **Abhimanyu Das**[‡]
[†]Carnegie Mellon University [‡]Google Research
bparia@cs.cmu.edu, {senrajat, amra, abhidas}@google.com

Abstract

Hierarchical forecasting is a key problem in many practical multivariate forecasting applications - the goal is to simultaneously predict a large number of correlated time series that are arranged in a pre-specified aggregation hierarchy. The challenge is to exploit the hierarchical correlations to simultaneously obtain good prediction accuracy for time series at different levels of the hierarchy. In this paper, we propose a new approach for hierarchical forecasting based on decomposing the time series along a global set of basis time series and modeling hierarchical constraints using the coefficients of the basis decomposition for each time series. Unlike past methods, our approach is scalable at inference-time (forecasting for a specific time series only needs access to its own data) while (approximately) preserving coherence among the time series forecasts. We experiment on several publicly available datasets and demonstrate significantly improved overall performance on forecasts at different levels of the hierarchy, compared to existing state-of-the-art hierarchical reconciliation methods.

1 Introduction

Multivariate time series forecasting is a key problem in many domains such as retail demand forecasting (Böse et al., 2017), financial predictions (Zhou et al., 2020), power grid optimization (Hyndman and Fan, 2009), road traffic modeling (Li et al., 2017), and online ads optimization (Cui et al., 2011). In many of these setting, the problem involves simultaneously forecasting a large number of possibly correlated time series for various downstream applications. In the retail domain, the time series might capture sales of items in a product inventory, and in power grids, the time series might correspond to energy consumption in a household. Often, these time series are arranged in a natural multi-level hierarchy - for example in retail forecasting, items could be grouped into subcategories and categories, and arranged in a product taxonomy. For power consumption forecasting, individual households are grouped into neighborhoods, counties, and cities. The hierarchical structure among the time series can usually be represented as a tree, with the leaf nodes corresponding to time series at the finest granularity, and the edges representing parent-child relationships. Figure 1a illustrates a typical hierarchy in the retail forecasting domain for time series of product sales.

In such settings, it is often required to obtain good forecasts, not just for the leaf level time-series (fine grained forecasts), but also for the aggregated time-series corresponding to higher level nodes (coarse gained forecasts). Furthermore, for interpretability and business decision making purposes, it is often desirable to obtain predictions that are roughly *coherent* or *consistent* (Thomson et al., 2019) with respect to the hierarchy tree. This means that the predictions for each parent time-series is equal to the sum of the predictions for its children time-series. Incorporating coherence constraints in the predictions captures the natural inductive bias in most typical hierarchical datasets, where the ground truth parent and children time series indeed adhere to additive constraints. For example, total sales of some product category is equal to the sum of sales of all items in that category.

[*]Part of this work was done while BP was an intern at Google.

Preprint. Under review.

The standard approach to addressing hierarchical forecasting is to either use a bottom-up approach, or reconciliation based approaches (Ben Taieb and Koo, 2019; Hyndman et al., 2016). The bottom-up approach involves training a model(s) to obtain predictions at the leaf nodes, and then aggregate up along the hierarchy tree to obtain predictions for higher-level nodes. Reconciliation methods make use of a trained model(s) to obtain predictions for all nodes of the tree, and then *reconcile* (or modify) them in a post-processing step to obtain coherent predictions. Both of these approaches suffer from shortcomings in term of either aggregating noise as one moves to higher level predictions (for example, bottom-up aggregation), or not jointly optimizing the forecasting predictions along with the coherence constraints (for example, reconciliation).

At the same time, there have been several recent advances on using Deep Neural Network models for multivariate forecasting, including Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) architectures, that have been shown to outperform classical time-series models (McKenzie, 1984; Hyndman et al., 2008) such as autoregressive and exponential smoothing models, for large datasets (Salinas et al., 2020; Sen et al., 2019; Wang et al., 2019). However, most of these approaches do not explicitly address the question of how to model the hierarchical relationships in the dataset. Deep forecasting models based on Graph Neural Networks (GNN) (Bai et al., 2020; Cao et al., 2020) do offer a general framework for learning on graph-structured data. However it is well known (Bojchevski et al., 2020) that GNNs are hard to scale for learning on graphs with a very large number of nodes, which in real-world settings such as retail forecasting, could involve hundreds of thousands of time series. More importantly, a desirable practical feature for multi-variate forecasting models is to be able to allow for prediction of a single (or a small subset of) time-series, without requiring to feed in the historical data for all the time series in the hierarchy. This is not possible for GNN-based forecasting models, or for models that use a reconciliation-based post-processing step.

In this paper, we propose a principled methodology to address all these issues. We present a deep neural network based multi-variate forecasting model that explicitly incorporates hierarchical constraints to provide accurate and approximately coherent predictions for all time series in the hierarchy. Our model is relatively easy to scale to a large number of time series, and at inference time, does not need historical data about other time-series in order to obtain predictions for a single time-series. The main idea behind our model is to decompose the multivariate data along a small set of *global* basis time series shared across all the time series in the hierarchy, and enforce hierarchical constraints on the coefficients of the basis decomposition of each time series. We also present some theoretical justification for our model in a simplified setting to formally show how modeling hierarchical additivity constraints can lead in better prediction accuracy.

Our notion of global basis time series decomposition differs from that of a few prior works (Wang et al., 2019; Sen et al., 2019) that explore related ideas. Wang et al. (2019) propose DeepFactors, which decomposes the time series as the output of a linear combination of RNNs (also known as global factors), which do not truly form a global basis on the time series space (since a single *factor* in their model will produce a different output vector depending on the time series passed as input). On the other hand, the DeepGLO (Sen et al., 2019) model uses an approach to alternatingly learn a regularizer network for low-rank matrix factorization on the basis vectors, and use this to explicitly decompose the input time series into a set of global basis vectors. It explicity stores the basis vectors for a fixed training period. This can be sub-optimal due to the lack of end-to-end optimization. More importantly, storing the basis time series as explicit vectors spanning the training window, instead of using a functional form might lead to poor generalization, and necessitates recomputing the basis vectors in the case of continuous prediction over a rolling window. In contrast, our approach captures the basis time series in a functional form by representing each of them as an RNN. Also, in constrast to DeepFactors, our approach constrains the output of the RNNs to produce global time series basis vectors. This lets us design a natural regularization strategy to capture the hierarchical relationship between the coefficients of the basis decomposition of each time series.

We experimentally evaluate our approach on multiple publicly available hierarchical forecasting datasets. We compare our approach to state-of-the-art (non-hierarchical) deep forecasting models, GNN-based models and reconciliation models, and show that our approach can obtain consistently more accurate predictions at all levels of the hierarchy tree.

2 Related Work

Time series forecasting has been studied for decades with the earliest works dating back to Box-Jenkins methodology (Box and Jenkins, 1968). Other traditional models include linear autoregressive models such as ARIMA (McKenzie, 1984), state space models, and exponential smoothing (Hyndman et al., 2008). While such models are effective in modelling small datasets robustly, they are unable to learn from large amounts of data due their small number of parameters.

Deep neural networks can scale to large amounts of data and have been effectively used for sequence modelling in various domains including multivariate time series data. While standard deep architectures such as dense neural networks are commonly used in practice, recurrent neural networks (RNNs) are more suitable for sequence data, and have been recently used for multi-variate time series data. DeepAR (Salinas et al., 2020) is an auto-regressive model based on RNNs and produces probabilistic forecasts. NBeats (Oreshkin et al., 2019) proposes a novel recurrent architecture that is interpretable while having the capacity to scale and learn from large number of time series. DeepState (Rangapuram et al., 2018) introduces deep state space models. A number of works have extended latent factor models to time series - Deep Factors (Wang et al., 2019) is a factor model based probabilistic forecasting method, TRMF (Yu et al., 2016) and DeepGLO (Sen et al., 2019) are based on temporally regularized matrix factorizations and decompose the data into a global basis. Convolutional architectures have also recently gained popularity as a replacement of RNNs for sequential data (Bai et al., 2018; Lai et al., 2018). Furthermore, temporal convolutions have been shown to effectively model time series data (Borovykh et al., 2017; Sen et al., 2019).

Graph Neural Networks (GNNs) (Hamilton et al., 2017) have also been successfully used to model multi-variate forecasting (Yu et al., 2017; Bai et al., 2020; Cao et al., 2020), where the time series are related according to a given graph structure. DCRNN (Li et al., 2017) introduce recurrent neural networks with diffusion convolutions on a provided correlation graph. There have been several other works in a similar vein (Cao et al., 2020; Wu et al., 2020).

While GNN based methods provide a general purpose framework to model correlated time series, most methods specifically designed for tree-structured hierarchical forecasting are based on a post-processing reconciliation step. That is, given base forecasts from any method of choice, a reconciliation function is learnt on an held out validation set, producing hierarchically coherent forecasts. A number of reconciliation techniques have been proposed in the literature. MinT (Wickramasuriya et al., 2015) assumes that the base forecasts are unbiased and reduces the problem of coherent hierarchical forecasting to a trace minimization problem. ERM (Ben Taieb and Koo, 2019) does not assume unbiasedness of the base classifiers and reduces it to a sparsity regularized empirical risk minimization problem. Other works in this area include Hyndman et al. (2016, 2011); Taieb et al. (2017); Wickramasuriya et al. (2020). Yanchenko et al. (2021) take a fully Bayesian approach by modelling the hierarchy using conditional distributions.

3 Problem Setting

We are given a set of N coherent time series of length T, arranged in a pre-defined hierarchy consisting of N nodes. At time step t, the time series data can be represented as a vector $\boldsymbol{y}_t \in \mathbb{R}^N$ denoting the time series values of all N nodes. We compactly denote the set of time series for all T steps as a matrix $\boldsymbol{Y} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_T]^{\boldsymbol{T}} \in \mathbb{R}^{T\times N}$. Also define $\boldsymbol{y}^{(i)}$ as the ith column of the matrix $\boldsymbol{Y}$ denoting all time steps of the i th time series, and $\boldsymbol{y}_t^{(i)}$ as the t th value of the i th time series. Time series forecasts can often be improved by using features as input to the model along with historical time series. The features often evolve with time, for example, categorical features such as *type of holiday*, or continuous features such as *time of the day*. We denote the matrix of such features by $\boldsymbol{X} \in \mathbb{R}^{T\times D}$, where the t th row denotes the D-dimensional feature vector at the t time step. For simplicity, we assume that the features are *global*, meaning that they are shared across all time series. Our proposed model can also utilize time series specific *local* features if available.

The focus of this paper is to forecast F future time steps conditioned on the past H time steps. When predicting future time series, we condition on the past time series, past features, and also the future features. When forecasting for time step t, we denote the predictions by $\widehat{\boldsymbol{y}}_t \in \mathbb{R}^N$. We compactly denote the H-step history of $\boldsymbol{Y}$ by $\boldsymbol{Y}_H = [\boldsymbol{y}_{t-H}, \cdots, \boldsymbol{y}_{t-1}]^T \in \mathbb{R}^{H\times N}$, the H-step history of $\boldsymbol{y}^{(i)}$ by $\boldsymbol{y}_H^{(i)} \in \mathbb{R}^H$, and the set of past + future features as $\boldsymbol{X}_H = [\boldsymbol{x}_{t-H}, \cdots, \boldsymbol{x}_{t+F-1}] \in \mathbb{R}^{(H+F)\times D}$.

Hierarchically Coherent Time Series. We assume that the time series data are coherent, that is, they satisfy the *sum constraints* over the hierarchy. The time series at each node of the hierarchy is the equal to the sum of the time series of its children, or equivalently, equal to the sum of the leaf time series of the sub-tree rooted at that node. Figure 1a shows an example of a sub-tree rooted at a node.

As a result of aggregation, the data can have widely varying scales with the values at higher level nodes being magnitudes higher than the leaf level nodes. It is well known that neural network training is more efficient if the data are similarly scaled. Hence, in this paper, we work with rescaled time series data. The time series at each node is downscaled by the number of leaves in the sub-tree rooted at the node, so that now they satisfy *mean constraints* rather than sum constraints described above. Denote by $\mathcal{L}(p)$, the set of leaf nodes of the sub-tree rooted at p. Hierarchically coherent data satisfy the following *data mean property*,

$$\boldsymbol{y}^{(p)} = \frac{1}{|\mathcal{L}(p)|} \sum_{i \in \mathcal{L}(p)} \boldsymbol{y}^{(i)} \quad \textit{(Data Mean Property).} \tag{1}$$

4 Hierarchically Regularized Deep Forecasting - HiReD

We first motivate our modeling decisions through a general assumption on multi-variate time-series datasets. Then we present various practical components of our model and justify that these components can together capture inductive biases in hierarchical forecasting datasets, for instance the mean aggregation constraint in Eq. (1).

Global Basis Assumption. We assume that all the time series in a dataset is a linear combination of a small set of basis time series. That is, $\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{\theta} + \boldsymbol{w}$, where $\boldsymbol{B} \in \mathbb{R}^{T \times K}$ denotes the set of basis vectors, $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_N] \in \mathbb{R}^{K \times N}$ denotes the set of weight vectors used in the linear combination for each time series, and $\boldsymbol{w} \in \mathbb{R}^{T \times N}$ denotes the noise matrix. Each row of $\boldsymbol{B}$ can be thought of as an evolving *global state* from which all the individual time series are derived. A classical example of such a basis set can be a small subset of Fourier or Wavelet basis (Strang, 1993; van den Oord et al., 2016) that is relevant to the dataset.

This assumption has also been invoked in recent deep learning literature (Sen et al., 2019; Wang et al., 2019). The basis recovered in the implementation of (Wang et al., 2019) (corresponding to the output vector of the *factors*) is allowed to vary unconstrained for each input time-series and therefore is not a true global basis. Sen et al. (2019) does explicitly recovers an approximate basis in the training set through low-rank matrix factorization regularized by a deep global predictive model alternatingly trained on the basis vectors. Thus the method does not lend itself to end-to-end optimization and the global predictive model can be data starved due to the low dimensionality of the recovered global basis. In the next section we present a method to implicitly represent a *data-dependent basis* in terms of regularized component models, that pertains itself to end-to-end training.

Implicit Data-Dependent Basis. For simplicity of exposition, here we consider the case of forecasting $F = 1$ step into the future. It is straightforward to generalize to forecasting for multiple time steps, as we also do in our experiments. Our aim is to obtain an implicit representation of a global basis that can be maintained as the weights of a deep network architecture like a set of seq-to-seq models when initialized in a data dependent manner. Moreover due to practical considerations, once trained on the whole dataset, we would like to be able to recover the basis given just a single time-series from the dataset. This would let us infer the future of each dimension without loading the whole dataset into memory. A classical example of such a scenario can be (i) given the dataset infer a set of Fourier basis that can represent the whole dataset at training time (ii) during inference given any time-series we would be able to get the coefficients of that time-series w.r.t to the chosen basis. In practice, though, we would like the choice of basis to be learnt from the data, rather than fix a specific form such as Fourier or Wavelet basis.

Guided by the above requirements, we model the global basis as a function of any given time-series in the dataset as follows,

$$\widehat{\boldsymbol{y}}_t^{(i)} = \underbrace{\mathcal{B}(\boldsymbol{X}_H, \boldsymbol{y}_H^{(i)};\, \Theta)}_{\widehat{\boldsymbol{B}}_t^{(i)}}\theta_i, \quad \forall \text{ nodes } i. \tag{2}$$

where θ_i is a learnable embedding for the ith time-series, Θ denotes global parameters shared across all time series, and $\widehat{\boldsymbol{B}}_t^{(i)}$ denotes an estimate of the global basis at time t recovered from time series i. Note that in the above equation, $\mathcal{B}$ is ideally a model that can implicitly recover the global basis given the history of any single time-series from the dataset, i.e., we would like the estimates $\widehat{\boldsymbol{B}}_t^{(i)}$ to be the same for all i. However, if we train such a model just using a forecasting-accuracy driven loss such as $\sum_{t,i} |\widehat{\boldsymbol{y}}_t^{(i)} - \boldsymbol{y}_t^{(i)}|$, the model would be free to have a different basis output for each dataset and thus would fail to learn a truly global basis. Therefore we propose a *basis regularization* that encourages the output of $\mathcal{B}$ (at any given time step) to be close to each other for all time-series in the dataset. In practice, the basis regularization can be implemented as,

$$B_{\mathrm{reg}}(\widehat{\boldsymbol{B}}_t) = \sum_{i=1}^{N} \left\| \widehat{\boldsymbol{B}}_t^{(i)} - \widehat{\boldsymbol{B}}_t^{(\pi(i))} \right\|_2^2, \tag{3}$$

for any random permutation π of $[1, \cdots, N]$. Note that the above loss can be minimized efficiently by mini-batch SGD. Now that we have introduced the part of our model that learns a data dependent global basis, we will focus on approximately capturing the mean aggregation constraints.

Approximate Coherence through Embedding Regularization. We would like to encourage our model to satisfy the mean aggregation property in Eq. (1) directly during training. As discussed in Section 3, for any coherent dataset, it holds that, the time series values of any node p is equal to the mean of the time series values of the leaf nodes of the sub-tree rooted at p. Applying the constraint under the global basis assumption, we get

$$\widehat{\boldsymbol{y}}_t^{(p)} = \frac{1}{|\mathcal{L}(p)|} \sum_{i \in \mathcal{L}(p)} \widehat{\boldsymbol{y}}_t^{(i)}, \quad \text{or,} \quad \widehat{\boldsymbol{B}}_t \left(\theta_p - \frac{1}{|\mathcal{L}(p)|} \sum_{i \in \mathcal{L}(p)} \theta_i \right) = \mathbf{0}.$$

The above vector equality must hold for any real $\widehat{\boldsymbol{B}}_t$ which implies that, for any node p, it holds,

$$\theta_p = \frac{1}{|\mathcal{L}(p)|} \sum_{i \in \mathcal{L}(p)} \theta_i \quad (\textit{Embedding Mean Property}). \tag{4}$$

It follows that the *embedding mean property* is a necessary and sufficient condition for the forecasts to be coherent. Therefore, we propose the following *embedding regularization*,

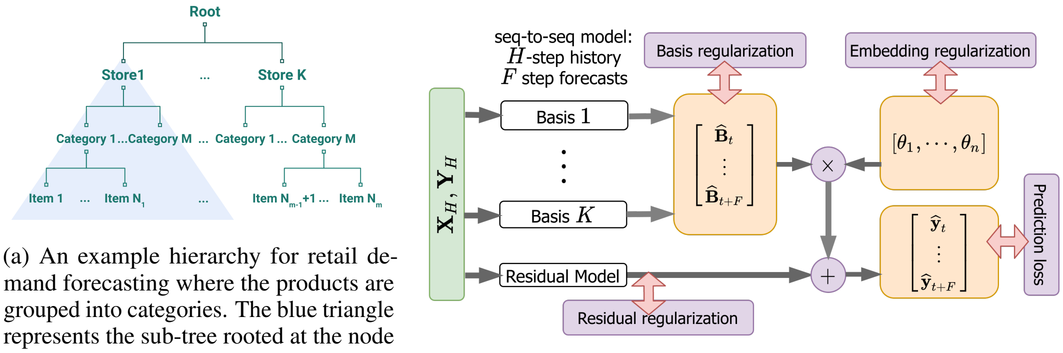
$$E_{\mathrm{reg}}(\boldsymbol{\theta}) = \sum_{p=1}^{N} \sum_{i \in \mathcal{L}(p)} \|\theta_p - \theta_i\|_2^2. \tag{5}$$

This regularization directly encourages the mean aggregation property during training due to the fact that, for fixed θ_i, the regularizer is minimized when θ_p satisfies the mean embedding property. Now we are at a position to present our final composite training loss.

Training Loss and Model Architecture. The function $\mathcal{B}$ can be modelled using any differentiable learning model such as recurrent neural networks (Hochreiter and Schmidhuber, 1997), transformers (Vaswani et al., 2017) or temporal convolution networks (Borovykh et al., 2017). For our experiments, we use the Sequence-to-Sequence (seq-to-seq) architecture (Sutskever et al., 2014), but we emphasize that the main ideas in our model is agnostic to the specific type of neural network architecture used. We minimize the mean absolute error (MAE) in predictions along with the two regularization terms introduced above. For regularization parameters λ_B and λ_E, the final loss function can be written as,

$$\ell(\Theta, \boldsymbol{\theta}) = \sum_t \left(\underbrace{\sum_i |\boldsymbol{y}_t^{(i)} - \widehat{\boldsymbol{y}}_t^{(i)}|}_{\text{Prediction loss}} + \underbrace{\lambda_B B_{\mathrm{reg}}(\widehat{\boldsymbol{B}}_t)}_{\text{Basis regularization}} \right) + \underbrace{\lambda_E E_{\mathrm{reg}}(\boldsymbol{\theta})}_{\text{Embedding regularization}} . \tag{6}$$

We model the basis generating function $\mathcal{B}$ using K independent seq-to-seq models, which take as input, the past history $(\boldsymbol{X}_H, \boldsymbol{Y}_H)$ and predict the basis $\widehat{\boldsymbol{B}}_t$ for the next F time steps. The basis predictions are further multiplied by $\boldsymbol{\theta}$ as described above, to produce the F-step forecasts. In order to account for any model mis-specifications, we also include a separate seq-to-seq component to model any residuals. The output of this *residual model* is added to the initial forecasts to yield the final forecast. We also regularize the outputs of the residual model to have small ℓ_2 norm, to prevent overfitting. The full architecture along with the various components are visualized in Figure 1b.



(a) An example hierarchy for retail demand forecasting where the products are grouped into categories. The blue triangle represents the sub-tree rooted at the node *Store1* with leaves denoted by *Item i*.

(b) The full model architecture.

Figure 1

5 Theoretical Justification for Hierarchical Modeling

In this section, we theoretically analyze the benefits of modeling hierarchical constraints explicitly in a forecasting problem, and show how it can result in provably improved accuracy, even at the leaf level. Since analyzing our actual deep non-linear model for an arbitrary hierarchical set of time series is complex, we make some simplifications to the problem and model. We will consider a 2-level hierarchy of time-series, consisting of a single root node (indexed by 0) with L children (denoted by $\mathcal{L}(0)$). We will assume that the ground-truth leaf time series are indeed generated using the global basis assumption in our model: $\boldsymbol{y} = \boldsymbol{B\theta} + \boldsymbol{w}$, where $\boldsymbol{B} \in \mathbb{R}^{T \times K}$ corresponds to K (unknown) basis vectors and $\boldsymbol{w}$ is noise sampled i.i.d as $w \sim \mathcal{N}(0, \sigma^2)$ for the leaf nodes.

We will also assume that instead of learning the K basis vectors $\boldsymbol{B}$ from scratch, the K basis vectors are assumed to come from a much larger dictionary $\bar{\boldsymbol{B}} \in \mathbb{R}^{T \times D}$ of D ($\gg K$) vectors that is fixed and known to the model. While the original problem learns the basis and the coefficients $\boldsymbol{\theta}$ simultaneously, in this case the goal is to select the basis from among a larger dictionary, and learn the coefficients $\boldsymbol{\theta}$.

We analyze this problem, and show that under the reasonable assumption of the parent embedding θ_0 being close to all the children embeddings θ_n, using the hierarchical constraints can result in a mean-square error at the leaf nodes that is a multiplicative factor L smaller than the optimal mean-square error of any model that does not use the hierarchical constraints.

Our proposed HiRed model, when applied in this setting would result in the following (hierarchically) regularized regression problem:

$$\min_{\boldsymbol{\theta}} \frac{1}{NT} \|\boldsymbol{y} - \boldsymbol{B\theta}\|_2^2 + \lambda \sum_{n \in \mathcal{L}(0)} \|\theta_0 - \theta_n\|_2^2. \tag{7}$$

For simplicity of analysis, we instead consider a three-stage version, described in Algorithm 1 and Algorithm 2: we first recover the support of the basis using Basis Pursuit (Chen et al., 2001). We then estimate the parameters of the root node, which is then plugged-in to solve for the parameters of the children node. We also define the baseline (unregularized) optimization problem for the leaf nodes that does not use any hierarchical information, as

$$\tilde{\theta}_n = \underset{\theta_n}{\operatorname{argmin}} \frac{1}{T} \|y_n - \boldsymbol{B}\theta_n\|_2^2 \quad \forall n \in \mathcal{L}(0). \tag{8}$$

The basis support recovery follows from standard analysis (Wainwright, 2009) detailed in Lemma 1 in the Appendix. We focus on the performance of Algorithm 2 here. The following theorem bounds the error of the unregularized ($\widetilde{\theta}_n$) and the hierarchically-regularized ($\widehat{\theta}_n$, see Algorithm 2) optimization solutions. A proof of the theorem can be found in Appendix A.2.

Theorem 1. *Suppose the rows of $\boldsymbol{B}$ are norm bounded as $\|\boldsymbol{B}_i\|_2 \le r$, and $\|\theta_n - \theta_0\|_2 \le \beta$. Define $\Sigma = \boldsymbol{B}^T\boldsymbol{B}/T$ as the empirical covariance matrix. For $\lambda_E = \frac{\sigma^2 K}{T\beta^2}$, $\widetilde{\theta}_n$ and $\widehat{\theta}_n$ can be bounded as,*

$$\mathbb{E}\|\widetilde{\theta}_n - \theta_n\|_\Sigma^2 \le \frac{\sigma^2 K}{T}, \quad \mathbb{E}\|\widehat{\theta}_n - \theta_n\|_\Sigma^2 \le 3\frac{\sigma^2 K}{T}\frac{1}{1 + \frac{\sigma^2 K}{Tr^2\beta^2}} + 6\frac{\sigma^2 K}{TL}. \tag{9}$$

Algorithm 1: Basis Recovery

```
Input: Observed $\boldsymbol{y}$, basis dict $\bar{\boldsymbol{B}}$, regularization parameter $\lambda_L$
Output: Estimated basis $\boldsymbol{B}$
$\widehat{\alpha}_0 \leftarrow \underset{\alpha \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2T}\|y_0 - \bar{\boldsymbol{B}}\alpha\|_2^2 + \lambda_L\|\alpha\|_1$
Estimate support $\widehat{S} = \{i \mid |\widehat{\alpha}_0| > 0\}$
Estimate true basis $\boldsymbol{B} \leftarrow \bar{\boldsymbol{B}}_{\widehat{S}}$
```

Algorithm 2: Parameter Recovery

```
Input: Observed time series $\boldsymbol{y}$, estimated basis $\boldsymbol{B}$, regularization parameter $\lambda_E$
Output: Estimated parameters $\boldsymbol{\theta}$
$\widehat{\theta}_0 \leftarrow \operatorname{argmin}_{\theta_0} \frac{1}{T}\|y_0 - \boldsymbol{B}\theta_0\|_2^2$
for $n \in \mathcal{L}(0)$ do
    $\widehat{\theta}_n \leftarrow \operatorname{argmin}_{\theta_n} \frac{1}{T}\|y_n - \boldsymbol{B}\theta_n\|_2^2 + \lambda_E\|\widehat{\theta}_0 - \theta_n\|_2^2$.
end
```

The gains due to the regularization can be understood by considering the case of a small β. Note that a small β implies that the children time-series have structural similarities which is common in hierarchical datasets. As $\beta \to 0$, the hierarchically regularized estimator approaches an error of $\mathcal{O}(\frac{\sigma^2 K}{TL})$ which is L times smaller when compared to the unregularized estimator. In fact, if $1/\beta^2 = \omega(T)$, then the numerator $1 + \frac{\sigma^2 K}{Tr^2\beta^2}$ in Eq. (9) is $\omega(1)$ resulting in $\mathbb{E}\|\widehat{\theta}_n - \theta_n\|_\Sigma^2 = o(\frac{\sigma^2 K}{T})$. This also shows that with larger T (more data), when $T = \omega(1/\beta^2)$, the errors bounds are of similar order and thus the gains due to regularization are minimal in this regime.

6 Experiments

We implemented our proposed model in Tensorflow (Abadi et al., 2016) and compared against multiple baselines on popular hierarchical time-series datasets.

Datasets. We experimented with three hierarchical forecasting datasets - Two retail forecasting datasets, M5 (M5, 2020) and Favorita (Favorita, 2017); and the Tourism (Tourism, 2019) dataset consisting of tourist count data. The history length and forecast horizon (H, F) were set to (28, 7), (28, 7) and (24, 4), for Favorita, M5 and Tourism respectively. More information can be found in Appendix B.2. We divide each of the datasets into training, validation and test sets, with details on the splits provided in Appendix B.3.

Baselines. We compare our proposed approach HiReD with the following baseline models: (i) *RNN* - we use a vanilla LSTM shared across all the time series, (ii) *DeepGLO* (Sen et al., 2019), (iii) *DCRNN* (Li et al., 2017), a GNN based approach where we feed the hierarchy tree as the input graph, (iv) *Deep Factors (DF)* (Wang et al., 2019), and (v) *DF+E_{reg}* - a version of DF where we add our hierarchical regularizer using E_{reg}[2]. For a fair comparison, we also make sure that all models have approximately the same number of parameters. We use code publicly released by the authors for DeepGLO[3] and DCRNN[4]. We implemented our own version of DeepFactors for a fair comparision, since the official implementation makes rolling probabilistic forecasts, whereas we make F-step forecasts directly using seq-to-seq models.

Additionally, we also compare with the recent *ERM* (Ben Taieb and Koo, 2019) reconciliation method applied to the base forecasts from the RNN model. It has been shown in (Ben Taieb and Koo, 2019) to outperform many previous reconciliation techniques such as MinT (Wickramasuriya et al., 2019).

Metrics. We compare the accuracy of the various approaches with respect to a number of metrics: (i) *mean absolute percentage error (MAPE)*, (ii) *weighted absolute percentage error (WAPE)*, and (iii) *symmetric mean absolute percentage error (SMAPE)*. A description of these metrics can be found in Appendix B.1. We report the metrics on the test data, for each level of the hierarchy. We also report the metrics on the full hierarchy denoted by *All* in the result Tables 1 and 2. Lastly, as a measure of the aggregate performance across all the levels, we also report the mean of the metrics in all the levels of the hierarchy denoted by *Mean*. Note, that our training metric is essentially a normalized version of the *All WAPE* metric.

[2] Note that the original DF model does not have hierarchical forecasting - we add this baseline to highlight that simply adding a hierarchical regularizer to an existing non-hierarchical model is usually not sufficient.

[3] https://github.com/rajatsen91/deepglo

[4] https://github.com/liyaguang/DCRNN/

Table 1: We provide MAPE/WAPE/SMAPE test metrics for the Favorita dataset averaged over 10 runs, for all the models. Level 0 corresponds to the root node, and Level 3 to the leaf node metrics. *All* denotes the average metric computed over all the time-series in the data, and *Mean* is obtained by first averaging the metrics across time-series at each level, and then taking the average of these.

	Level 0	Level 1	Level 2	Level 3	Mean	All
HIRED	0.062 / 0.062 / 0.063	**0.170** / 0.107 / **0.186**	**0.221 / 0.130 / 0.270**	**0.332 / 0.203 / 0.320**	**0.196 / 0.126 / 0.210**	**0.323 / 0.199 / 0.315**
RNN	0.068 / 0.067 / 0.068	0.189 / 0.114 / 0.197	0.246 / 0.134 / 0.290	**0.333 / 0.203** / 0.339	0.209 / 0.130 / 0.223	0.325 / **0.200** / 0.334
DF+E_{reg}	0.061 / 0.062 / 0.062	0.178 / 0.112 / 0.196	0.228 / 0.134 / 0.275	0.341 / 0.207 / **0.320**	0.202 / 0.129 / 0.213	0.331 / 0.203 / **0.316**
DF	0.063 / 0.064 / 0.064	0.179 / 0.110 / 0.194	0.235 / 0.135 / 0.291	0.349 / 0.213 / 0.343	0.206 / 0.130 / 0.223	0.339 / 0.209 / 0.338
DeepGLO	0.094 / 0.098 / 0.088	0.194 / 0.126 / 0.197	0.241 / 0.156 / 0.338	0.339 / 0.226 / 0.404	0.217 / 0.151 / 0.256	0.330 / 0.222 / 0.397
DCRNN	0.079 / 0.080 / 0.080	0.248 / 0.120 / 0.212	0.254 / 0.134 / 0.328	0.399 / 0.204 / 0.389	0.245 / 0.134 / 0.252	0.387 / **0.200** / 0.383
RNN+ERM	**0.057 / 0.056 / 0.058**	0.176 / **0.103 / 0.185**	0.242 / **0.129** / 0.283	0.480 / 0.220 / 0.348	0.239 / **0.127** / 0.219	0.459 / 0.215 / 0.342

Table 2: We provide MAPE/WAPE/SMAPE test metrics for the M5 dataset averaged over 10 runs, for all the models. Definitions for the metrics corresponding to Levels 0-3, *All* and *Mean* are the same as in Table 1.

	Level 0	Level 1	Level 2	Level 3	Mean	All
HIRED	**0.049 / 0.051 / 0.051**	**0.057 / 0.058 / 0.059**	**0.078 / 0.072 / 0.082**	**0.445 / 0.268 / 0.496**	**0.158 / 0.112 / 0.172**	**0.444 / 0.268 / 0.494**
RNN	0.057 / 0.059 / 0.059	0.077 / 0.083 / 0.083	0.093 / 0.085 / 0.098	0.457 / 0.282 / 0.517	0.171 / 0.127 / 0.189	0.456 / 0.281 / 0.516
DF+E_{reg}	0.051 / 0.053 / 0.053	**0.057** / 0.060 / **0.059**	0.080 / 0.076 / 0.084	**0.445** / 0.271 / 0.499	**0.158** / 0.115 / 0.174	**0.443** / 0.270 / 0.497
DF	0.054 / 0.055 / 0.056	**0.058** / 0.061 / **0.060**	0.081 / 0.076 / 0.085	**0.445** / 0.272 / 0.501	**0.159** / 0.116 / 0.176	**0.443** / 0.271 / 0.500
DeepGLO	0.077 / 0.077 / 0.081	0.086 / 0.087 / 0.092	0.106 / 0.099 / 0.113	0.446 / 0.278 / 0.538	0.178 / 0.135 / 0.206	0.445 / 0.277 / 0.536
DCRNN	0.078 / 0.078 / 0.079	0.091 / 0.096 / 0.092	0.171 / 0.165 / 0.193	0.469 / 0.282 / 0.512	0.202 / 0.156 / 0.219	0.467 / 0.282 / 0.511
RNN+ERM	**0.050 / 0.052 / 0.052**	0.068 / 0.066 / 0.071	0.096 / 0.084 / 0.104	0.464 / 0.286 / 0.520	0.169 / 0.122 / 0.187	0.462 / 0.286 / 0.518

Training. We train our model using minibatch SGD with the Adam optimizer (Kingma and Ba, 2014). We use learning rate decay and early stopping based on the *All WAPE* metric on the validation set. We perform 10 independent runs and report the average metrics on the test set. All our experiments were performed using a single Titan Xp GPU with 12GB of memory. Further training details and model hyperparameters can be found in Appendix B.3.

6.1 Results

Tables 1, 2, and 3 show the averaged test metrics for Favorita, M5, and Tourism datasets respectively. We present only the *Mean* and *All* metrics for the Tourism dataset due to lack of space. Complete results with the variances for all the three datasets can be be found in Appendix B.4.

We find that for all three datasets, our proposed model either yields the smallest error or close to the smallest error across most metrics and most levels (except Level 0 - the root). We find that ERM is the best performing model at the root node level for all datasets. In general, we observe that ERM yields an improvement over the base RNN predictions for the higher levels which are closer to the root node (Levels 0 and 1), while, worsening at the lower levels. In the case of non-reconciliation based methods, no method performs consistently better than our proposed model for all datasets. DeepFactors and DeepGLO perform well for some metrics especially on the Tourism dataset, while yielding sub-optimal results on the other two datasets. We outperform methods that do not explicitly account for the hierarchy like DeepGLO, Deep Factors and RNN on the higher level time-series, in most datasets. We also observe that simply adding a hierarchical regularizer to Deep Factors (DF+E_{reg}) does not lead to large gains especially in the Tourism and M5 datasets (this might be due to the fact that Deep Factors does not use a truly global basis). DCRNN, in spite of using the hierarchy as a graph also does not perform as well as our approach, especially in Tourism and M5 Datasets - possibly due to the fact that a GNN graph is not the most effective way to model the hierarchical data. Overall, we find that our proposed method consistently works better or at par with the other baselines at all hierarchical levels.

Ablation study. Next, we perform an ablation study of our proposed model to understand the effects of its various components, the results of which are presented in Table 5. We compare our proposed

Table 3: We provide MAPE/WAPE/SMAPE test metrics for the Tourism dataset averaged over 10 runs. The results for the *All* and *Mean* metrics shown here. The individual results for all five levels can be found in the Appendix.

	Mean	All
HIRED	**0.407 / 0.197 / 0.332**	**0.816 / 0.315** / 0.674
RNN	**0.404** / 0.211 / **0.333**	0.828 / 0.327 / **0.661**
DF+E_{reg}	0.414 / 0.203 / 0.338	**0.815** / 0.317 / 0.679
DF	0.415 / 0.204 / 0.334	0.819 / 0.317 / **0.662**
DeepGLO	0.457 / **0.199** / 0.346	0.929 / 0.321 / 0.744
DCRNN	0.553 / 0.281 / 0.392	1.101 / 0.391 / 0.729
RNN+ERM	0.780 / 0.251 / 0.417	1.773 / 0.429 / 0.856

Table 4: Comparison of the coherency metric at each level for Favorita and M5 datasets, for the HiReD and RNN models.

	Favorita			M5		
	L0	L1	L3	L0	L1	L3
HIRED	**0.035**	**0.039**	**0.040**	**0.024**	**0.029**	**0.032**
RNN	0.043	0.044	0.042	0.042	0.057	0.047

Table 5: We present an ablation study for each of the components in the HiReD model in terms of the MAPE/WAPE/SMAPE test metrics for the Favorita dataset.

	Level 0	Level 1	Level 2	Level 3	Mean	All
HIRED	**0.062 / 0.062 / 0.063**	**0.170 / 0.107 / 0.186**	**0.221 / 0.130 / 0.270**	**0.332 / 0.203 / 0.320**	**0.196 / 0.126 / 0.210**	**0.323 / 0.199 / 0.315**
Base + E_{reg}	**0.062 / 0.063 / 0.064**	0.172 / 0.109 / 0.191	0.224 / 0.134 / 0.279	0.343 / 0.207 / 0.333	0.200 / 0.128 / 0.217	0.332 / 0.203 / 0.328
Base + B_{reg}	0.068 / 0.069 / 0.070	0.179 / 0.113 / 0.192	0.229 / 0.138 / 0.275	0.344 / 0.213 / 0.324	0.205 / 0.133 / 0.215	0.334 / 0.208 / 0.319
Base	**0.063 / 0.062** / 0.065	0.179 / 0.112 / 0.194	0.229 / 0.136 / 0.279	0.342 / 0.212 / 0.329	0.203 / 0.131 / 0.217	0.332 / 0.208 / 0.324

model (with both the regularizers activated), to the *Base* model consisting of only the prediction loss (See Eq. (6)) and two other models where only one of the basis and embedding regularizers is activated at a time. We find that, using the regularizers one at a time yields only a similar or slightly better performance compared to the Base model. Therefore, either regularizer is not sufficient by itself to yield significant improvements, showing the importance of both regularizers in our model.

Coherence. We also compare the coherence of our predictions to that of the RNN model. Specifically, for each node we measure the deviation of our forecast from the most coherent forecast. In our setting, we define the most coherent forecast at node p as $\boldsymbol{c}^{(p)} = 1/\mathcal{L}(p) \sum_{i \in \mathcal{L}(p)} \widehat{\boldsymbol{y}}^{(i)}$, the mean of the leaf node predictions of the corresponding sub-tree. In Table 4, we report the *All WAPE* metric between the predictions from our model $\widehat{\boldsymbol{y}}$ and the most coherent forecasts $\boldsymbol{c}$, for each of the hierarchical levels. We do not report the leaf level metrics, since the coherence metric equals zero for leaf level predictions. We find that our proposed model consistently produces more coherent predictions compared to RNN.

7 Conclusion

In this paper, we proposed a method for hierarchical time series forecasting, which was motivated by the idea of approximate reconciliation. We started with the global basis assumption, and imposing mean constraints on the model predictions, we showed that satisfying mean constraints on the predictions is equivalent to satisfying mean constraints on the embeddings. Our training loss, in addition to the prediction loss, also includes the basis and embedding regularizations required to enforce the constraints on the predictions. Our model is fully differentiable and is trainable via SGD, while also being scalable with respect to the number of nodes in the hierarchy. We empirically evaluated our method on three benchmark datasets and showed that our model consistently improved over state of the art baselines for most levels of the hierarchically. We perform an ablation study to justify the important model components and also show empirically that our forecasts are more coherent than the RNN baseline.

In this work, we aimed at maximizing the overall performance without emphasizing performance at individual hierarchical levels. For future work, we plan to treat this is as a multi-objective problem with the goal of optimizing each of the individual levels. We also plan to extend our current model to probabilistic forecasting. Lastly, we do not anticipate any direct negative societal impacts of our work. Time series forecasting is a benign topic, that can be used to make well informed decisions resulting in lesser wastage of resources, time, and energy.

References

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

L. Bai, L. Yao, C. Li, X. Wang, and C. Wang. Adaptive graph convolutional recurrent network for traffic forecasting. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17804–17815. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/ce1aad92b939420fc17005e5461e6f48-Paper.pdf.

S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

S. Ben Taieb and B. 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.

A. Bojchevski, J. Klicpera, B. Perozzi, A. Kapoor, M. Blais, B. Rózemberczki, M. Lukasik, and S. Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2464–2473, 2020.

A. Borovykh, S. Bohte, and C. W. Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

J.-H. Böse, V. Flunkert, J. Gasthaus, T. Januschowski, D. Lange, D. Salinas, S. Schelter, M. Seeger, and Y. Wang. Probabilistic demand forecasting at scale. *Proceedings of the VLDB Endowment*, 10 (12):1694–1705, 2017.

G. E. Box and G. M. Jenkins. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 17(2):91–109, 1968.

D. Cao, Y. Wang, J. Duan, C. Zhang, X. Zhu, C. Huang, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang. Spectral temporal graph neural network for multivariate time-series forecasting. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17766–17778. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/cdf6581cb7aca4b7e19ef136c6e601a5-Paper.pdf.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

Y. Cui, R. Zhang, W. Li, and J. Mao. Bid landscape forecasting in online ad exchange marketplace. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 265–273, 2011.

Favorita. Favorita forecasting dataset. https://www.kaggle.com/c/favorita-grocery-sales-forecast, 2017.

W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. pages 1024–1034, 2017.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

D. Hsu, S. M. Kakade, and T. Zhang. Random design analysis of ridge regression. In *Conference on learning theory*, pages 9–1. JMLR Workshop and Conference Proceedings, 2012.

R. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

R. J. Hyndman and S. Fan. Density forecasting for long-term peak electricity demand. *IEEE Transactions on Power Systems*, 25(2):1142–1153, 2009.

R. J. Hyndman, R. A. Ahmed, G. Athanasopoulos, and H. L. Shang. Optimal combination forecasts for hierarchical time series. *Computational statistics & data analysis*, 55(9):2579–2589, 2011.

R. J. Hyndman, A. J. Lee, and E. Wang. Fast computation of reconciled forecasts for hierarchical and grouped time series. *Computational statistics & data analysis*, 97:16–32, 2016.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

G. Lai, W.-C. Chang, Y. Yang, and H. Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104, 2018.

Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

M5. M5 forecasting dataset. https://www.kaggle.com/c/m5-forecasting-accuracy/, 2020.

E. McKenzie. General exponential smoothing and the equivalent arma process. *Journal of Forecasting*, 3(3):333–344, 1984.

B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31: 7785–7794, 2018.

D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

R. Sen, H.-F. Yu, and I. Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *arXiv preprint arXiv:1905.03806*, 2019.

G. Strang. Wavelet transforms versus fourier transforms. *Bulletin of the American Mathematical Society*, 28(2):288–305, 1993.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems (NIPS 2014)*, 2014.

S. B. Taieb, J. W. Taylor, and R. J. Hyndman. Coherent probabilistic forecasts for hierarchical time series. In *International Conference on Machine Learning*, pages 3348–3357. PMLR, 2017.

M. E. Thomson, A. C. Pollock, D. Önkal, and M. S. Gönül. Combining forecasts: Performance and coherence. *International Journal of Forecasting*, 35(2):474–484, 2019.

Tourism. Tourism forecasting dataset. https://robjhyndman.com/publications/mint/, 2019.

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, pages 125–125, 2016.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017.

M. J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1-constrained quadratic programming (lasso). *IEEE transactions on information theory*, 55(5): 2183–2202, 2009.

Y. Wang, A. Smola, D. Maddix, J. Gasthaus, D. Foster, and T. Januschowski. Deep factors for forecasting. In *International Conference on Machine Learning*, pages 6607–6617. PMLR, 2019.

S. L. Wickramasuriya, G. Athanasopoulos, R. J. Hyndman, et al. Forecasting hierarchical and grouped time series through trace minimization. *Department of Econometrics and Business Statistics, Monash University*, 105, 2015.

S. L. Wickramasuriya, G. Athanasopoulos, and R. 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.

S. L. Wickramasuriya, B. A. Turlach, and R. J. Hyndman. Optimal non-negative forecast reconciliation. *Statistics and Computing*, 30(5):1167–1182, 2020.

Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 753–763, 2020.

A. K. Yanchenko, D. D. Deng, J. Li, A. J. Cron, and M. West. Hierarchical dynamic modeling for individualized bayesian forecasting. *arXiv preprint arXiv:2101.03408*, 2021.

B. Yu, H. Yin, and Z. Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

H.-F. Yu, N. Rao, and I. S. Dhillon. Temporal regularized matrix factorization for high-dimensional time series prediction. In *NIPS*, pages 847–855, 2016.

D. Zhou, L. Zheng, Y. Zhu, J. Li, and J. He. Domain adaptive multi-modality neural attention network for financial forecasting. In *Proceedings of The Web Conference 2020*, pages 2230–2240, 2020.

A Theory

A.1 Support Recovery

Lemma 1. *Suppose $\boldsymbol{B}$ satisfies the lower eigenvalue condition (Assumption 1 in Appendix A.3) with parameter $C_{\min}$ and the mutual incoherence condition (Assumption 2 in Appendix A.3) with parameter γ. Also assume that the columns of the basis pool $\bar{\boldsymbol{B}}$ are normalized so that $\max_{j \in S^c} \|\bar{\boldsymbol{B}}^{(j)}\| \leq \sqrt{T}$, and the true parameter θ_0 of the root satisfies*

$$\|\theta_0\|_\infty \geq \lambda_L \left[|\!|\!| \Sigma^{-1} |\!|\!|_\infty + \frac{4\sigma}{\sqrt{LC_{\min}}} \right], \tag{10}$$

where $|\!|\!|A|\!|\!|_\infty = \max_i \sum_j |A_{ij}|$ denotes matrix operator ℓ_∞ norm, and $\Sigma = \boldsymbol{B}^T\boldsymbol{B}/T$ denotes the empirical covariance matrix. Then for $\lambda_L \geq \frac{2}{\gamma}\sqrt{\frac{2\sigma^2 \log d}{LT}}$, with probability least $1 - 4\exp(-c_1 T \lambda^2)$ (for some constant c_1), the support $\widehat{S} = \{i \mid |\widehat{\alpha}_0| > 0\}$ recovered from the Lasso solution (see Algorithm 1) is equal to the true support S.

Proof. We are given a pool of basis vectors $\bar{\boldsymbol{B}}$ from which the observed data is generated using a subset of K columns which we have denoted by $\boldsymbol{B}$ in the text. We denote the correct subset of columns by S and recover them from the observed data using basis pursuit - also known as the support recovery problem in the literature. Given the observed data and the pool of basis vectors $\bar{\boldsymbol{B}}$, we recover the support from the following regression problem corresponding to the root node time series.

$$y_0 = \bar{\boldsymbol{B}}\alpha + w_0, \quad w_0 \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}/L), \tag{11}$$

where α is K-sparse with the non-zero indices at S, and the non-zero values equal θ_0 - the true parameters of the root node. Here we have used the fact that the root node has a $1/L$ times smaller variance due to aggregation. The true support S can be recovered from the observed data y_0, by solving the sparse regression problem (Lasso) given in Algorithm 2. A number of standard Lasso assumptions are needed to ensure that the support is identifiable, and that the non-zero parameters are large enough to be estimated. Assuming that $\bar{\boldsymbol{B}}_S$ $(= \boldsymbol{B})$ and α satisfy all the assumptions of Theorem 2, the theorem ensures that the true support S is recovered with high probability. □

A.2 Proof of Theorem 1 - Error Bounds for Regularized Estimators

For this proof, we assume that the support S is recovered and the true basis functions $\boldsymbol{B}$ are known with high probability (see Section A.1). We divide the proof into multiple steps.

Step I: By Corollary 1, the OLS estimate $\widehat{\theta}_0$ (see Algorithm 2) of parameters of the root node and the OLS estimate $\widetilde{\theta}_n$ (see Eq. (8)) can be bounded as,

$$\mathbb{E}[\|\widehat{\theta}_0 - \theta_0\|_\Sigma^2] \leq \frac{\sigma^2 K}{TL}, \quad \mathbb{E}[\|\widetilde{\theta}_n - \theta_n\|_\Sigma^2] \leq \frac{\sigma^2 K}{T} \quad \forall n \in \mathcal{L}(0). \tag{12}$$

Step II: Next, using change of variables, we notice that the ridge regression loss for the child nodes (see Algorithm 2) is equivalent to the following.

$$\widehat{\psi}_n = \underset{\psi_n}{\operatorname{argmin}} \frac{1}{T}\|y_n - \boldsymbol{B}\widehat{\theta}_0 - \boldsymbol{B}\psi_n\|_2^2 + \lambda\|\psi_n\|_2^2 \quad \forall n \in \mathcal{L}(0), \tag{13}$$

where $\psi_n = \theta_n - \widehat{\theta}_0$. The final estimate for the child parameters can be written as a sum of the ψ_n estimate and the root node estimate, $\widehat{\theta}_n = \widehat{\psi}_n + \widehat{\theta}_0$. We also consider a related problem that will help us in computing the errors bounds.

$$\widetilde{\psi}_n = \underset{\psi_n}{\operatorname{argmin}} \frac{1}{T}\|y_n - \boldsymbol{B}\theta_0 - \boldsymbol{B}\psi_n\|_2^2 + \lambda\|\psi_n\|_2^2 \quad \forall n \in \mathcal{L}(0). \tag{14}$$

Here we have replaced $\widehat{\theta}_0$ with the true value θ_0. Note that this regression problem cannot be solved in practice since we do not have access to the true value of θ_0. We will only use it to assist in the analysis. Now we will bound the difference between the estimates $\widehat{\psi}_n$ and $\widetilde{\psi}_n$. The closed form solution for ridge regression is well known in the literature.

$$
\begin{aligned}
\widehat{\psi}_n &= T^{-1}(\Sigma + \lambda \boldsymbol{I})^{-1}\boldsymbol{B}^T(y_n - \boldsymbol{B}\widehat{\theta}_0) \\
\widetilde{\psi}_n &= T^{-1}(\Sigma + \lambda \boldsymbol{I})^{-1}\boldsymbol{B}^T(y_n - \boldsymbol{B}\theta_0),
\end{aligned}
$$

where $\Sigma = \boldsymbol{B}^T\boldsymbol{B}/T$, as defined earlier. The norm of the difference of the estimates can be bounded as

$$
\begin{aligned}
\widehat{\psi}_n - \widetilde{\psi}_n &= T^{-1}(\Sigma + \lambda \boldsymbol{I})^{-1}\boldsymbol{B}^T\boldsymbol{B}(\widetilde{\theta}_0 - \widehat{\theta}_0) \\
\implies \|\widehat{\psi}_n - \widetilde{\psi}_n\|_\Sigma^2 &= (\widetilde{\theta}_0 - \widehat{\theta}_0)^T\Sigma(\Sigma + \lambda \boldsymbol{I})^{-1}\Sigma(\Sigma + \lambda \boldsymbol{I})^{-1}\Sigma(\widetilde{\theta}_0 - \widehat{\theta}_0) \\
&= (\widetilde{\theta}_0 - \widehat{\theta}_0)^T\Sigma(\Sigma + \lambda \boldsymbol{I})^{-1}\Sigma(\Sigma + \lambda \boldsymbol{I})^{-1}\Sigma(\widetilde{\theta}_0 - \widehat{\theta}_0) \\
&= (\widetilde{\theta}_0 - \widehat{\theta}_0)^T V D\left[\frac{\lambda_i^3}{(\lambda_i + \lambda)^2}\right] V^T(\widetilde{\theta}_0 - \widehat{\theta}_0).
\end{aligned}
$$

Here we have used an eigen-decomposition of the symmetric sample covariance matrix as $\Sigma = VD[\lambda_i]V^T$. We use the notation $D[\lambda_i]$ to denote a diagonal matrix with values λ_i on the diagonal. The above can be further upper bounded using the fact that $\lambda_i \leq \lambda_i + \lambda$.

$$
\|\widehat{\psi}_n - \widetilde{\psi}_n\|_\Sigma^2 \leq (\widetilde{\theta}_0 - \widehat{\theta}_0)^T V D[\lambda_i] V^T(\widetilde{\theta}_0 - \widehat{\theta}_0) = \|\widetilde{\theta}_0 - \widehat{\theta}_0\|_\Sigma^2. \tag{15}
$$

Step III: Now we will bound the error on $\widetilde{\psi}_n$ and finally use it in the next step with triangle inequality to prove our result. Note that $y_n - \boldsymbol{B}\theta_0 = \boldsymbol{B}(\theta_n - \theta_0) + w_n$. Therefore, we can see from Eq. (14) that $\widetilde{\psi}_n$ is an estimate for $\theta_n - \theta_0$. Using the fact that $\|\theta_n - \theta_0\|_2 \leq \beta$ and corollary 1, $\widetilde{\psi}_n$ can be bounded as,

$$
\mathbb{E}[\|\widetilde{\psi} - (\theta_n - \theta_0)\|_\Sigma^2] \leq \frac{r^2\beta^2\sigma^2 K}{Tr^2\beta^2 + \sigma^2 K}. \tag{16}
$$

Finally using triangle inequality, we bound the error of our estimate $\widehat{\theta}_n$.

$$
\begin{aligned}
\|\widehat{\theta}_n - \theta_n\|_\Sigma^2 &= \|\widehat{\psi}_n + \widehat{\theta}_0 - \theta_n\|_\Sigma^2 \quad \text{(Using the decomposition from Step II).} \\
&\leq 3\left(\|\widehat{\psi}_n - \widetilde{\psi}_n\|_\Sigma^2 + \|\widetilde{\psi}_n - (\theta_n - \theta_0)\|_\Sigma^2 + \|\widehat{\theta}_0 - \theta_0\|_\Sigma^2\right) \\
&\quad \text{(Using triangle and Cauchy-Schwartz inequality)} \\
&\leq 3\left(\|\widetilde{\psi}_n - (\theta_n - \theta_0)\|_\Sigma^2 + 2\|\widehat{\theta}_0 - \theta_0\|_\Sigma^2\right) \quad \text{(Using Eq. (15)).}
\end{aligned}
$$

Taking the expectation of the both sides, and using Eq. (12) and (16), we get the desired result.

$$
\mathbb{E}\|\widehat{\theta}_n - \theta_n\|_\Sigma^2 \leq 3\frac{r^2\beta^2\sigma^2 K}{Tr^2\beta^2 + \sigma^2 K} + 6\frac{\sigma^2 K}{TL}.
$$

A.3 Review of Sparse Linear Regression

We consider the following sparse recovery problem. We are given data $(\boldsymbol{X}, y) \in \mathbb{R}^{n\times d} \times \mathbb{R}^n$ following the observation model $y = \boldsymbol{X}\theta^* + w$, where $w \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$, and θ^* is supported in the indices indexed by a set S (S-sparse). We estimate θ^* using the following Lagrangian Lasso program,

$$
\widehat{\theta} \in \underset{\theta\in\mathbb{R}^n}{\operatorname{argmin}} \left\{\frac{1}{2n}\|y - \boldsymbol{X}\theta\|_2^2 + \lambda_n\|\theta\|_1\right\} \tag{17}
$$

We consider the fixed design setting, where the matrix $\boldsymbol{X}$ is fixed and not sampled randomly. Following (Wainwright, 2009), we make the following assumptions required for recovery of the true support S of θ^*.

Assumption 1 (Lower eigenvalue)**.** *The smallest eigenvalue of the sample covariance sub-matrix indexed by S is bounded below:*

$$
\Lambda_{\min}\left(\frac{\boldsymbol{X}_S^T\boldsymbol{X}_S}{n}\right) \geq C_{\min} > 0 \tag{18}
$$

Assumption 2 (Mutual incoherence)**.** *There exists some* $\gamma \in (0, 1]$ *such that*

$$\left|\!\left|\!\left| \boldsymbol{X}_{S^c}^T \boldsymbol{X}_S (\boldsymbol{X}_S^T \boldsymbol{X}_S)^{-1} \right|\!\right|\!\right|_\infty \leq 1 - \gamma, \tag{19}$$

where $|\!|\!| A |\!|\!|_\infty = \max_i \sum_j |A_{ij}|$ *denotes matrix operator* ℓ_∞ *norm.*

Theorem 2 (Support Recovery, Wainwright (2009))**.** *Suppose the design matrix satisfies assumptions 1 and 2. Also assume that the design matrix has its* n*-dimensional columns normalized so that* $\max_{j \in S^c} \|X_j\|_2 \leq \sqrt{n}$*. Then for* λ_n *satisfying,*

$$\lambda_n \geq \frac{2}{\gamma} \sqrt{\frac{2\sigma^2 \log d}{n}}, \tag{20}$$

the Lasso solution $\widehat{\theta}$ *satisfies the following properties with a probability of at least* $1 - 4\exp(-c_1 n \lambda_n^2)$*:*

1. *The Lasso has a unique optimal solution* $\widehat{\theta}$ *with its support contained within the true support* $S(\widehat{\theta}) \subseteq S(\theta^*)$ *and satisfies the* ℓ_∞ *bound*

$$\|\widehat{\theta}_S - \theta_S^*\|_\infty \leq \underbrace{\lambda_n \left[\left|\!\left|\!\left| \left(\frac{\boldsymbol{X}_S^T \boldsymbol{X}_S}{n} \right)^{-1} \right|\!\right|\!\right|_\infty + \frac{4\sigma}{\sqrt{C_{\min}}} \right]}_{g(\lambda_n)}, \tag{21}$$

2. *If in addition, the minimum value of the regression vector* θ^* *is lower bounded by* $g(\lambda_n)$*, then it recovers the exact support.*

A.4 Review of Ridge Regression

In this section we review the relevant background from (Hsu et al., 2012) on fixed design ridge regression. As usual, we assume data $(\boldsymbol{X}, y) \in \mathbb{R}^{n \times d} \times \mathbb{R}^n$ following the observation model $y = \boldsymbol{X}\theta^* + w$, where $w \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$. Define the *ridge estimator* $\widehat{\theta}$ as the minimizer of the ℓ_2 regularized mean squared error,

$$\widehat{\theta} \in \underset{\theta \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \frac{1}{n} \|y - \boldsymbol{X}\theta\|_2^2 + \lambda \|\theta\|_2^2 \right\} \tag{22}$$

We denote the sample covariance matrix by $\Sigma = \boldsymbol{X}^T \boldsymbol{X}/n$. Then for any parameter θ, the expected ℓ_2 prediction error is given by, $\|\theta - \theta^*\|_\Sigma^2 = \|\boldsymbol{X}(\theta - \theta^*)\|_2^2/n$. We also assume the standard ridge regression setting of bounded $\|\theta^*\|_2 \leq B$. We have the following proposition from Hsu et al. (2012) on expected error bounds for ridge regression.

Proposition 1 (Hsu et al. (2012))**.** *For any regularization parameter* $\lambda > 0$*, the expected prediction loss can be upper bounded as*

$$\mathbb{E}[\|\widehat{\theta} - \theta^*\|_\Sigma^2] \leq \sum_j \frac{\lambda_j}{(\lambda_j/\lambda + 1)^2} \theta_j^{*2} + \frac{\sigma^2}{n} \sum_j \left(\frac{\lambda_j}{\lambda_j + \lambda} \right)^2, \tag{23}$$

where λ_i *denote the eigenvalues of the empirical covariance matrix* Σ*.*

Using the fact that $\lambda_j \leq \textbf{tr}\,(\Sigma)$, and $x/(x + c)$ is increasing in x for $x \geq 0$, the above bound can be simplified as,

$$\begin{aligned} \mathbb{E}[\|\widehat{\theta} - \theta^*\|_\Sigma^2] &\leq \frac{\textbf{tr}\,(\Sigma)}{(\textbf{tr}\,(\Sigma)/\lambda + 1)^2} \|\theta^*\|^2 + \frac{\sigma^2 d}{n} \left(\frac{\textbf{tr}\,(\Sigma)}{\textbf{tr}\,(\Sigma) + \lambda} \right)^2 \\ &\leq \frac{\textbf{tr}\,(\Sigma)\, B^2 \lambda^2 + \textbf{tr}\,(\Sigma)^2\, \sigma^2 d/n}{(\textbf{tr}\,(\Sigma) + \lambda)^2} \end{aligned}$$

Assuming that the covariate vectors $\boldsymbol{X}_i$ are norm bounded as $\|\boldsymbol{X}_i\|_2 \leq r$, and using the fact that $\textbf{tr}\,(\Sigma) \leq r^2$, gives us the following corollary.

Corollary 1. *When choosing* $\lambda = \frac{\sigma^2 d}{n B^2}$*, the prediction loss can be upper bounded as,*

$$\mathbb{E}[\|\widehat{\theta} - \theta^*\|_\Sigma^2] \leq \frac{r^2 B^2 \sigma^2 d}{n r^2 B^2 + \sigma^2 d}. \tag{24}$$

The usual ordinary least squares bound of $\frac{\sigma^2 d}{n}$ *can be derived when considering the limit* $B \to \infty$*, corresponding to* $\lambda = 0$*.*

B Further Experimental Details

B.1 Accuracy Metrics

In this section we define the evaluation metrics used in this paper. Denote the true values by $\boldsymbol{y}$ and the predicted values by $\widehat{\boldsymbol{y}}$, both n-dimensional vectors.

1. Mean absolute percent error MAPE $= \frac{1}{n}\sum_i \frac{|\widehat{\boldsymbol{y}}_i - \boldsymbol{y}_i|}{|\boldsymbol{y}_i|}$, where the mean is taken over non-zero values of $\boldsymbol{y}_i$ only.
2. Symmetric mean absolute percent error SMAPE $= \frac{2}{n}\sum_i \frac{|\widehat{\boldsymbol{y}}_i - \boldsymbol{y}_i|}{|\boldsymbol{y}_i| + |\widehat{\boldsymbol{y}}_i|}$.
3. Weighted absolute percentage error WAPE $= \frac{\sum_i |\widehat{\boldsymbol{y}}_i - \boldsymbol{y}_i|}{\sum_i |\boldsymbol{y}_i|}$.

B.2 Dataset Details

We use three publicly available benchmark datasets for our experiments.

1. The M5 dataset[5] consists of time series data of product sales from 10 Walmart stores in three US states. The data consists of two different hierarchies: the product hierarchy and store location hierarchy. For simplicity, in our experiments we use only the product hierarchy consisting of 3k nodes and 1.8k time steps. The validation scores are computed using the predictions from time steps 1843 to 1877, and test scores on steps 1878 to 1913.
2. The Favorita dataset[6] is a similar dataset, consisting of time series data from Corporación Favorita, a South-American grocery store chain. As above, we use the product hierarchy, consisting of 4.5k nodes and 1.7k time steps. The validation scores are computed using the predictions from time steps 1618 to 1652, and test scores on steps 1653 to 1687.
3. The Australian Tourism dataset[7] consists of monthly domestic tourist count data in Australia across 7 states which are sub-divided into regions, sub-regions, and visit-type. The data consists of around 500 nodes and 230 time steps. The validation scores are computed using the predictions from time steps 122 to 156, and test scores on steps 157 to 192.

For the three datasets, all the time-series (corresponding to both leaf and higher-level nodes) of the hierarchy that we used are present in the training data.

B.3 Training Details

Table 6 presents all the hyperparameters used in our proposed model. All models were trained via SGD using the Adam optimizer, and the training data was standardized to mean zero and unit variance. The datasets were split into train, val, and test, the sizes of which are given in the Table 6. We used early stopping using the *All WAPE* score on the validation set, with a patience of 10 for all models. We tuned most of the parameters manually with respect to the *All WAPE* metric on the validation set, with the exception of the regularization parameters, for which we performed model based hyper-parameter optimization.

All our experiments were implemented in Tensorflow 2, and run on a Titan Xp GPU with 12GB of memory. The computing server we used, had 256GB of memory, and 32 CPU cores, however, our code did not seem to use more than 10GB of memory and 4 CPU cores.

B.4 More Results

Tables 7, 8, and 9 show the test metrics averaged over 10 independent runs on the three datasets along with the variances.

[5] https://www.kaggle.com/c/m5-forecasting-accuracy/

[6] https://www.kaggle.com/c/favorita-grocery-sales-forecasting/

[7] https://robjhyndman.com/publications/mint/

Table 6: Various model hyperparameters

	Favorita	M5	Tourism
LSTM hidden dim	20	20	10
Embedding dim K	8	8	4
Basis regularization λ_B	6.6e-9	5e-6	1e-10
Embedding regularization λ_E	4.9e-4	1e-6	3e-9
Residual LSTM regularization	1e-10	1e-10	0.1
History length H and forecast horizon F	(28, 7)	(28, 7)	(24, 4)
No. of rolling val/test windows	5	5	3
Initial learning rate	0.01	0.01	0.1
Decay rate and decay interval	(0.5, 6)	(0.5, 6)	(0.5, 6)
Early stopping patience	10	10	10
Training epochs	40	40	40
Batch size	512	512	512

Table 7: MAPE/WAPE/SMAPE test metrics for the Favorita dataset. The metrics are averaged over 10 runs, and the standard deviations are provided in parenthesis. We bold the smallest mean in each column and anything that comes within one standard deviation.

	Level 0	Level 1	Level 2	Level 3	Mean	All
HIRED	0.062 / 0.062 / 0.063 (0.003) / (0.003) / (0.004)	**0.170** / 0.107 / **0.186** (0.002) / (0.002) / (0.003)	**0.221** / **0.130** / **0.270** (0.002) / (0.002) / (0.006)	**0.332** / **0.203** / **0.320** (0.003) / (0.001) / (0.008)	**0.196** / **0.126** / **0.210** (0.002) / (0.002) / (0.003)	**0.323** / **0.199** / **0.315** (0.003) / (0.001) / (0.007)
RNN	0.068 / 0.067 / 0.068 (0.004) / (0.004) / (0.003)	0.189 / 0.114 / 0.197 (0.005) / (0.003) / (0.004)	0.246 / 0.134 / 0.290 (0.006) / (0.002) / (0.005)	**0.333** / **0.203** / 0.339 (0.004) / (0.001) / (0.005)	0.209 / 0.130 / 0.223 (0.004) / (0.002) / (0.004)	0.325 / **0.200** / 0.334 (0.004) / (0.001) / (0.005)
DF+E_{reg}	0.061 / 0.062 / 0.062 (0.004) / (0.004) / (0.004)	0.178 / 0.112 / 0.196 (0.003) / (0.003) / (0.004)	0.228 / 0.134 / 0.275 (0.003) / (0.003) / (0.011)	0.341 / 0.207 / **0.320** (0.004) / (0.001) / (0.011)	0.202 / 0.129 / 0.213 (0.003) / (0.003) / (0.007)	0.331 / 0.203 / **0.316** (0.003) / (0.001) / (0.011)
DF	0.063 / 0.064 / 0.064 (0.003) / (0.003) / (0.004)	0.179 / 0.110 / 0.194 (0.003) / (0.002) / (0.003)	0.235 / 0.135 / 0.291 (0.003) / (0.002) / (0.007)	0.349 / 0.213 / 0.343 (0.006) / (0.001) / (0.007)	0.206 / 0.130 / 0.223 (0.003) / (0.002) / (0.004)	0.339 / 0.209 / 0.338 (0.005) / (0.001) / (0.007)
DeepGLO	0.094 / 0.098 / 0.088 (0.001) / (0.001) / (0.001)	0.194 / 0.126 / 0.197 (0.001) / (0.001) / (0.001)	0.241 / 0.156 / 0.338 (0.001) / (0.001) / (0.001)	0.339 / 0.226 / 0.404 (0.002) / (0.001) / (0.001)	0.217 / 0.151 / 0.256 (0.001) / (0.001) / (0.001)	0.330 / 0.222 / 0.397 (0.002) / (0.001) / (0.001)
DCRNN	0.079 / 0.080 / 0.080 (0.004) / (0.004) / (0.005)	0.248 / 0.120 / 0.212 (0.010) / (0.001) / (0.002)	0.254 / 0.134 / 0.328 (0.002) / (0.000) / (0.000)	0.399 / 0.204 / 0.389 (0.003) / (0.000) / (0.000)	0.245 / 0.134 / 0.252 (0.002) / (0.001) / (0.001)	0.387 / **0.200** / 0.383 (0.003) / (0.000) / (0.000)
RNN+ERM	**0.057** / **0.056** / **0.058** (0.002) / (0.002) / (0.002)	0.176 / **0.103** / **0.185** (0.004) / (0.001) / (0.003)	0.242 / **0.129** / 0.283 (0.004) / (0.001) / (0.005)	0.480 / 0.220 / 0.348 (0.020) / (0.001) / (0.005)	0.239 / **0.127** / 0.219 (0.005) / (0.001) / (0.003)	0.459 / 0.215 / 0.342 (0.018) / (0.001) / (0.005)

Table 8: MAPE/WAPE/SMAPE test values for the M5 dataset.

	Level 0	Level 1	Level 2	Level 3	Mean	All
HIRED	**0.049** / **0.051** / **0.051** (0.001) / (0.001) / (0.001)	**0.057** / **0.058** / **0.059** (0.001) / (0.001) / (0.001)	**0.078** / **0.072** / **0.082** (0.002) / (0.001) / (0.002)	**0.445** / **0.268** / **0.496** (0.001) / (0.000) / (0.002)	**0.158** / **0.112** / **0.172** (0.000) / (0.001) / (0.001)	**0.444** / **0.268** / **0.494** (0.001) / (0.000) / (0.002)
RNN	0.057 / 0.059 / 0.059 (0.003) / (0.002) / (0.003)	0.077 / 0.083 / 0.083 (0.008) / (0.013) / (0.011)	0.093 / 0.085 / 0.098 (0.004) / (0.002) / (0.004)	0.457 / 0.282 / 0.517 (0.005) / (0.006) / (0.007)	0.171 / 0.127 / 0.189 (0.004) / (0.005) / (0.005)	0.456 / 0.281 / 0.516 (0.005) / (0.006) / (0.007)
DF+E_{reg}	0.051 / 0.053 / 0.053 (0.002) / (0.002) / (0.002)	**0.057** / 0.060 / **0.059** (0.001) / (0.001) / (0.002)	0.080 / 0.076 / 0.084 (0.001) / (0.001) / (0.002)	**0.445** / 0.271 / 0.499 (0.001) / (0.000) / (0.003)	**0.158** / 0.115 / 0.174 (0.001) / (0.001) / (0.001)	**0.443** / 0.270 / 0.497 (0.001) / (0.000) / (0.003)
DF	0.054 / 0.055 / 0.056 (0.001) / (0.001) / (0.001)	**0.058** / 0.061 / **0.060** (0.001) / (0.001) / (0.001)	0.081 / 0.076 / 0.085 (0.002) / (0.001) / (0.002)	**0.445** / 0.272 / 0.501 (0.001) / (0.000) / (0.002)	**0.159** / 0.116 / 0.176 (0.001) / (0.001) / (0.001)	**0.443** / 0.271 / 0.500 (0.001) / (0.000) / (0.002)
DeepGLO	0.077 / 0.077 / 0.081 (0.0003) / (0.0003) / (0.0004)	0.086 / 0.087 / 0.092 (0.0004) / (0.0003) / (0.0004)	0.106 / 0.099 / 0.113 (0.0003) / (0.0003) / (0.0003)	0.446 / 0.278 / 0.538 (0.0001) / (0.0001) / (0.0001)	0.178 / 0.135 / 0.206 (0.0003) / (0.0003) / (0.0003)	0.445 / 0.277 / 0.536 (0.0001) / (0.0001) / (0.0001)
DCRNN	0.078 / 0.078 / 0.079 (0.006) / (0.006) / (0.007)	0.091 / 0.096 / 0.092 (0.003) / (0.005) / (0.004)	0.171 / 0.165 / 0.193 (0.005) / (0.003) / (0.007)	0.469 / 0.282 / 0.512 (0.001) / (0.000) / (0.000)	0.202 / 0.156 / 0.219 (0.002) / (0.002) / (0.003)	0.467 / 0.282 / 0.511 (0.001) / (0.000) / (0.000)
RNN+ERM	**0.050** / **0.052** / **0.052** (0.001) / (0.001) / (0.001)	0.068 / 0.066 / 0.071 (0.001) / (0.001) / (0.002)	0.096 / 0.084 / 0.104 (0.002) / (0.001) / (0.002)	0.464 / 0.286 / 0.520 (0.005) / (0.002) / (0.004)	0.169 / 0.122 / 0.187 (0.002) / (0.001) / (0.001)	0.462 / 0.286 / 0.518 (0.005) / (0.002) / (0.004)

Table 9: MAPE/WAPE/SMAPE test values for the Tourism dataset.

	Level 0	Level 1	Level 2	Level 3	Level 4	Mean	All
HIRED	0.084 / 0.084 / 0.086 (0.004) / (0.004) / (0.004)	0.172 / 0.129 / 0.161 (0.002) / (0.001) / (0.001)	0.245 / **0.179** / 0.236 (0.002) / (0.001) / (0.000)	0.556 / 0.237 / **0.374** (0.013) / (0.000) / (0.002)	0.978 / **0.354** / 0.802 (0.008) / (0.001) / (0.014)	**0.407** / **0.197** / **0.332** (0.004) / (0.001) / (0.002)	**0.816** / **0.315** / 0.674 (0.008) / (0.000) / (0.010)
RNN	0.101 / 0.110 / 0.106 (0.001) / (0.001) / (0.001)	**0.162** / 0.148 / 0.164 (0.001) / (0.001) / (0.002)	0.230 / 0.188 / 0.231 (0.002) / (0.001) / (0.001)	0.520 / 0.240 / 0.385 (0.008) / (0.000) / (0.006)	1.009 / 0.369 / **0.782** (0.015) / (0.001) / (0.012)	**0.404** / 0.211 / **0.333** (0.004) / (0.001) / (0.002)	0.828 / 0.327 / **0.661** (0.012) / (0.000) / (0.008)
DF+E_{reg}	0.091 / 0.094 / 0.094 (0.002) / (0.002) / (0.002)	0.186 / 0.138 / 0.171 (0.005) / (0.001) / (0.003)	0.252 / 0.186 / 0.240 (0.002) / (0.001) / (0.002)	0.570 / 0.241 / 0.379 (0.012) / (0.001) / (0.002)	**0.973** / 0.356 / 0.807 (0.004) / (0.000) / (0.012)	0.414 / 0.203 / 0.338 (0.004) / (0.001) / (0.003)	**0.815** / 0.317 / 0.679 (0.005) / (0.000) / (0.008)
DF	0.093 / 0.097 / 0.096 (0.002) / (0.003) / (0.002)	0.183 / 0.141 / 0.170 (0.004) / (0.002) / (0.002)	0.252 / 0.187 / 0.240 (0.004) / (0.001) / (0.002)	0.571 / 0.241 / 0.380 (0.016) / (0.001) / (0.002)	0.977 / **0.355** / **0.783** (0.009) / (0.000) / (0.014)	0.415 / 0.204 / 0.334 (0.006) / (0.001) / (0.003)	0.819 / 0.317 / **0.662** (0.010) / (0.000) / (0.010)
DeepGLO	**0.080** / 0.089 / **0.079** (0.0001) / (0.0002) / (0.0002)	0.174 / **0.126** / **0.158** (0.0002) / (0.0001) / (0.0001)	0.239 / **0.179** / **0.218** (0.0003) / (0.0001) / (0.0001)	0.684 / **0.234** / **0.372** (0.001) / (0.0001) / (0.0001)	1.104 / 0.364 / 0.900 (0.0011) / (0.0001) / (0.0002)	0.457 / **0.199** / 0.346 (0.0007) / (0.0001) / (0.0001)	0.929 / 0.321 / 0.744 (0.001) / (0.0001) / (0.0001)
DCRNN	0.162 / 0.187 / 0.171 (0.005) / (0.003) / (0.003)	0.265 / 0.231 / 0.248 (0.004) / (0.002) / (0.003)	0.304 / 0.258 / 0.279 (0.002) / (0.001) / (0.002)	0.696 / 0.293 / 0.398 (0.010) / (0.001) / (0.001)	1.339 / 0.434 / 0.865 (0.013) / (0.000) / (0.000)	0.553 / 0.281 / 0.392 (0.004) / (0.000) / (0.001)	1.101 / 0.391 / 0.729 (0.010) / (0.000) / (0.000)
RNN+ERM	**0.082** / **0.078** / **0.079** (0.005) / (0.005) / (0.005)	0.236 / 0.155 / 0.206 (0.010) / (0.003) / (0.006)	0.358 / 0.225 / 0.291 (0.012) / (0.004) / (0.006)	1.015 / 0.307 / 0.498 (0.067) / (0.006) / (0.008)	2.209 / 0.488 / 1.013 (0.074) / (0.009) / (0.010)	0.780 / 0.251 / 0.417 (0.032) / (0.005) / (0.006)	1.773 / 0.429 / 0.856 (0.064) / (0.008) / (0.009)