SLOTH: Structured Learning and Task-Based Optimization for Time Series Forecasting on Hierarchies

Fan Zhou, Chen Pan, Lintao Ma, Yu Liu, Shiyu Wang, James Zhang, Xinxin Zhu, Xuanwei Hu, Yunhua Hu, Yangfei Zheng, Lei Lei, Hu Yun

Ant Group

{hanlian.zf, bopu.pc, lintao.mlt, nuoman.ly, weiming.wsy, james.z, qinrui.zxx, xuanwei.hxw, yangfei.zyf, jason.ll, huyun.h}@antgroup.com, wugou.hyh@alibaba-inc.com

Abstract

Multivariate time series forecasting with hierarchical structure is widely used in real-world applications, e.g., sales predictions for the geographical hierarchy formed by cities, states, and countries. The *hierarchical time series* (HTS) forecasting includes two sub-tasks, i.e., *forecasting* and *reconciliation*. In the previous works, hierarchical information is only integrated in the reconciliation step to maintain coherency, but not in forecasting step for accuracy improvement. In this paper, we propose two novel tree-based feature integration mechanisms, i.e., *top-down convolution* and *bottom-up attention* to leverage the information of the hierarchical structure to improve the forecasting performance. Moreover, unlike most previous reconciliation methods which either rely on strong assumptions or focus on coherent constraints only, we utilize deep neural optimization networks, which not only achieve coherency without any assumptions, but also allow more flexible and realistic constraints to achieve task-based targets, e.g., lower under-estimation penalty and meaningful decision-making loss to facilitate the subsequent downstream tasks. Experiments on real-world datasets demonstrate that our tree-based feature integration mechanism achieves superior performances on hierarchical forecasting tasks compared to the state-of-the-art methods, and our neural optimization networks can be applied to real-world tasks effectively without any additional effort under coherence and task-based constraints.

Introduction

Forecasting for multiple time series with complex hierarchical structure has been applied in various real-world problems (Dangerfield and Morris 1992; Athanasopoulos, Ahmed, and Hyndman 2009; Liu, Yan, and Hauskrecht 2018; Jeon, Panagiotelis, and Petropoulos 2018). For instance, the international sales forecasting of transnational corporations needs to consider geographical hierarchies involving the levels of city, state, and country (Han, Dasgupta, and Ghosh 2021). Moreover, retail sales can also be divided into different groups according to the product category to form another hierarchy. The combination of different hierarchies can form a more complicated but realistic virtual topology, e.g., the geographical hierarchies (of different regions) and the commodity hierarchies (of varies categories) are nested to be parts of the supply chain in the retail sector.

The critical challenge in hierarchical forecasting tasks lies in producing accurate prediction results while satisfying aggregation (coherence) constraints (Taieb and Koo 2019). Specifically, the hierarchical structure in these time series implies coherence constraint, i.e., time series at upper levels are the aggregation of those at lower levels. However, independent forecasts from a prediction model (called *base forecasts*) are unlikely to satisfy coherence constraint.

Previous work on hierarchical forecasting mainly focuses on a procedure of two separate stages (Hyndman, Lee, and Wang 2016; Taieb and Koo 2019; Corani et al. 2020; Anderer and Li 2021): In the first stage, *base forecasts* are generated independently for each time series in the hierarchy; in the second stage, these forecasts are adjusted via *reconciliation* to derive coherent results. The base forecasts are obtained by univariate or multivariate time series models, such as traditional forecasting methods (e.g., ARIMA (Taieb and Koo 2019)) and deep learning techniques (e.g. DeepVar (Salinas et al. 2019)). However, both approaches ignore the information of hierarchical structure for prediction. As for reconciliation, traditional statistical methods mostly rely on strong assumption, such as unbiased forecasts and Gaussian noises (e.g., MinT (Wickramasuriya, Athanasopoulos, and Hyndman 2019)), which are often inconsistent with non-Gaussian/non-linear real-world *hierarchical time series* (HTS) data. A notable exception is the approach proposed in (Rangapuram et al. 2021), promising coherence by a closed-form projection to achieve an end-to-end reconciliation without any assumptions. However, this method may introduce huge adjustments on the original forecasts for coherency, making the final forecasts unreasonable. Moreover, even coherent forecasters might still be impractical for some real-world tasks with further practical operational or task-related constraints, such as inventory management and resource scheduling. These reconciliation concerns can be addressed by imposing more realistic constraints to control the scale and optimize the task-based targets for the downstream tasks, which can be achieved by *deep neural optimization layer* (OptNet) (Amos and Kolter 2017).

In this work, we provides an end-to-end framework to generate coherent forecasts of hierarchical time series that incorporates the hierarchical structure in the prediction pro-

cess, while taking task-based constraints and targets into consideration. In detail, our contributions to HTS forecasting and aligned decision-making can be summarised as follows:

- We propose two tree-based mechanisms, including top-down convolution and bottom-up attention, to leverage hierarchical structure information (through feature fusion of all levels in the hierarchy) for performance improvement of the base forecasts. To the best of our knowledge, our approach is the first model that harnesses the power of deep learning to exploit the complicated structure information of HTS.
- We provide a flexible end-to-end learning framework to unify the goals of the forecasting and decision-making by employing a deep differentiable convex optimization layer (OptNet), which not only achieves controllable reconciliation without any assumptions, but also adapts to more practical task-related constraints and targets to solve the real-world problems without any explicit post-processing step.
- Extensive experiments on real-world hierarchical datasets from various industrial domains demonstrate that our proposed approach achieves significant improvements over the state-of-the-art baseline methods, and our approach has been deployed online to cloud resource scheduling project in Ant Group.

Preliminaries

A hierarchical time series can be denoted as a tree structure (see Figure 1) with linear aggregation constraints, expressed by aggregation matrix $\mathbf{S} \in \mathbb{R}^{n \times m}$ (m is the number of bottom-level nodes, and n is the total number of nodes). In the hierarchy, each node represents a time series, to be predicted over a time horizon.

Given a time horizon $t \in \{1, 2, \ldots, T\}$, we use $y_{i,t} \in \mathbb{R}$ to denote the values of the i-th component of a multivariate hierarchical time series, where $i \in \{1, 2, \ldots, n\}$ is the index of the individual univariate time series. Here we assume that the index i abides by the level-order traversal of the hierarchical tree, going from left to right at each level.

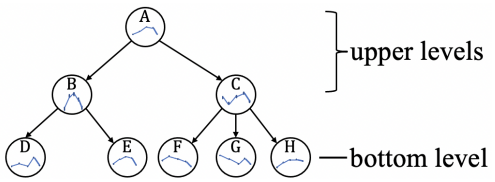

Figure 1: An example of HTS structure for $n = 8$ time series with $m = 5$ bottom-level series and $r = 3$ upper-level series.

In the tree structure, the time series of leaf nodes are called the *bottom-level* series $\mathbf{b}_t \in \mathbb{R}^m$, and those of the remaining nodes are termed *upper-levels* series $\mathbf{u}_t \in \mathbb{R}^r$. Obviously, the total number of nodes $n = r + m$, and $\mathbf{y}_t := [\mathbf{u}_t, \mathbf{b}_t]^\mathsf{T} \in \mathbb{R}^n$ contains observations at time t for all levels, which satisfies

$$\mathbf{y}_t = \mathbf{S}\mathbf{b}_t, \tag{1}$$

where $\mathbf{S} \in \{0, 1\}^{n \times m}$ is an aggregation matrix.

Taking the HTS in Figure 1 as an example, the aggregation matrix is in the form:

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{\text{sum}} \\ \mathbf{I}_5 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & \mathbf{I}_5 & & \end{bmatrix},$$

where $\mathbf{I}_5$ is an identity matrix of size 5. The total number of series in the hierarchy is $n = 3 + 5$. At each time step t,

$$\mathbf{u}_t = [y_{A,t}, y_{B,t}, y_{C,t}] \in \mathbb{R}^3,$$
$$\mathbf{b}_t = [y_{D,t}, y_{E,t}, y_{F,t}, y_{G,t}, y_{H,t}] \in \mathbb{R}^5,$$
$$y_{A,t} = y_{B,t} + y_{C,t} = y_{D,t} + y_{E,t} + y_{F,t} + y_{G,t} + y_{H,t}.$$

The coherence constraint of Eq. (1) can be represented as (Rangapuram et al. 2021)

$$\mathbf{A}\mathbf{y}_t = \mathbf{0}, \tag{2}$$

where $\mathbf{A} := [\mathbf{I}_r | -\mathbf{S}_{\text{sum}}] \in \{0, 1\}^{r \times n}$, $\mathbf{0}$ is an r-vector of zeros, and $\mathbf{I}_r$ is a $r \times r$ identity matrix.

Method

In this section, we introduce our framework (SLOTH), which not only can improve prediction performance by integrating hierarchical structure into forecasting, but also achieves task-based goals with deep neural optimization layer fulfilling coherence constraint. Figure 2 illustrates the architecture of SLOTH, consisting of two main components:

- a *structured hierarchical forecasting module* that produces forecasts over the prediction horizon across all nodes, utilizing both top-down convolution and bottom-up attention to enhance the temporal features for each node.
- a *task-based constrained optimization module* that leverages OptNet to satisfy the coherence constraint and provide a flexible module for real-world tasks with more complex practical constraints and targets.

Structured Hierarchical Forecasting

In this section, we introduce the *structured hierarchical learning module*, which integrates dynamic features from the hierarchical structure to produce better base forecasts.

Temporal Feature Extraction Module This module extracts temporal features of each node as follows:

$$\bar{h}_t^i = \text{UPDATE}(\bar{h}_{t-1}^i, x_t^i; \theta),\ i \in \{1, 2, \ldots, n\}, \tag{3}$$
$$\bar{\mathbf{H}}_t = [\bar{h}_t^1, \bar{h}_t^2, \ldots, \bar{h}_t^n],$$

where x_t^i is the covariates of node i at time t, $\bar{h}_{t-1}^i$ is the hidden feature of the previous time step $t-1$, θ is the model parameters shared across all nodes, and *UPDATE*$(\cdot)$ is the pattern extraction function. Any recurrent-type neural

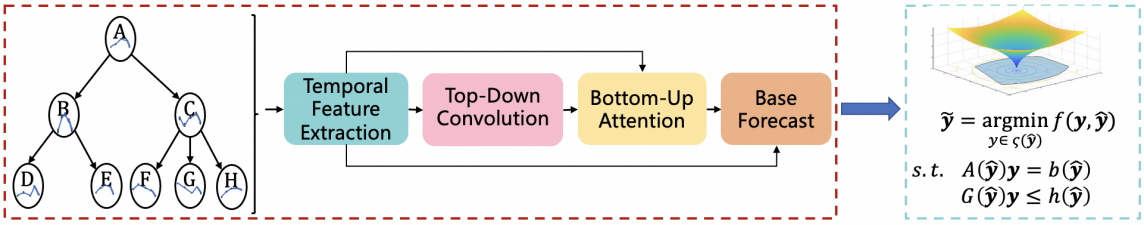

Figure 2: The Architecture of SLOTH: the red dashed box is the hierarchical forecasting component including the temporal feature extraction module, top-down convolution module, bottom-up attention module, and base forecasting module. The base forecasting module generates predictions without reconciliation. The light blue dashed box is the task-based optimization module that generates the reconciliation forecasts and achieves task-based targets for real-world scenarios.

network can be adopted as the temporal feature extraction module, such as the RNN variants (Hochreiter and Schmidhuber 1997; Chung et al. 2014), TCN (Bai, Kolter, and Koltun 2018), WAVENET (Van Den Oord et al. 2016), and NBEATS (Oreshkin et al. 2019). We use GRU (Chung et al. 2014) in our experiments due to its simplicity.

It is worth noting that we process each node independently in this module, which is in contrast to some existing works that utilize multivariate model to extract the relationship between time series (Rangapuram et al. 2021). It is unnecessary for our framework because we leverage the hierarchical structure in feature integration step, which we believe is enough to characterize the relationship between nodes. We reduce the time complexity from O(n^2) to O(n) accordingly.

Top-Down Convolution Module (TD-Conv) This module incorporates structural information to dynamic pattern by integrating temporal features (e.g., trends and seasonality) from nodes at the top level into those at the bottom level to enhance the temporal stability. In other words, clearer seasonality and more smooth evolving (Taieb, Taylor, and Hyndman 2021).

Most of the previous methods indicate that time series of nodes at the upper level are easier to predict than nodes at the bottom level, which implies the dynamic pattern at the top level is more stable. Therefore, the bottom nodes can use their ancestors' features at top levels to improve prediction performance. Similar to Tree Convolution (Mou et al. 2016), our approach introduces a top-down convolution mechanism by extracting effective top-level temporal patterns to denoise the feature of bottom nodes and increase stability.

The top-down convolution mechanism shown in Figure 3 is based on the outputs of the univariate forecasting model. We want to highlight that all nodes share the forecasting model to obtain the temporal features. Please note that ancestor's value are the sum of all children nodes. In other words, ancestors' features are helpful to predict the considered node, which calls for the integration of the features of both levels for better prediction.

In order to reduce the computational complexity, the hidden states are reorganized into matrices (yellow boxes in the middle part of Figure 3) after temporal hidden features are obtained. Each row of the matrices represents the feature concatenation of nodes from considered node to root in the hierarchy. We then apply convolution neural networks (CNNs) to each row of those matrices to aggregate the temporal patterns, since it is well known that CNN is an effective tool to integrate valid information from hidden features.

The computation complexity is very high if the convolution is applied on $\bar{\mathbf{H}}_t$ because it is not sorted as tree structure index. To accelerate the computation, we transform the hierarchical structure to a series of matrix forms to speed up the convolution process as shown in the middle part of Figure 3. Specifically, the nodes' hidden features $\bar{\mathbf{H}}_t$ in the shape of (n, d_h) are reorganized into a matrix form $\bar{\mathbf{H}}'_t = \{\bar{h'}^1_t, ..., \bar{h'}^n_t\}$, where $\bar{h'}^i_t = [\bar{h}^1_t, \ldots, \bar{h}^i_t]$ is the concatenation of temporal features $\bar{h}_t$ of all the ancestors of node i in the shape of (l_i, d_h), where n is number of nodes, l_i is the number of levels of node i, and d_h is the dimension. Please note that $\bar{\mathbf{H}}'_t$ is not an actual matrix but a union of matrices of different dimensions, where the first dimension of each $\bar{h'}_t$ is the level index. Then we apply convolution to $\bar{h'}^i_t$ to integrate temporal features of all ancestors of i as

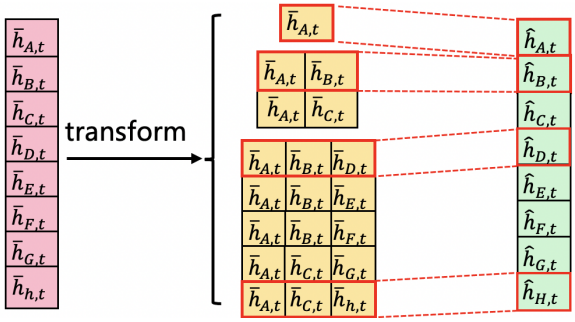

Figure 3: Top-Down Convolution Architecture. Given the temporal features of each node (pink boxes) from the temporal feature extraction module, our approach transforms these hidden features into matrices (yellow boxes) for fast convolution. The right part with red dashed lines demonstrates the integration of ancestors' features with CNN, and the green boxes represent the integration feature.

follows

$$\hat{h}_t^i = \text{Conv}_{l_i}(\bar{h'}_t^i; \Theta_{l_i}) = \sum_{k=1}^{l_i} w_k \bar{h}_t^{l_i-k}, \quad (4)$$
$$\hat{\mathbf{H}}_t = [\hat{h}_t^1, \hat{h}_t^2, \ldots, \hat{h}_t^n],$$

where different levels have different convolution components Conv_{l_i}, while nodes at the same level share the same parameter Θ_{l_i}.

It is important to emphasize that HTS is different from general graph-structured time series, where spatial-temporal information is passed between adjacent nodes, but the relationship between nodes in hierarchical structure only contains value aggregation, which indicates the message passing mechanism of graph time series (DCRNN (Li et al. 2017) and STGCN (Yu, Yin, and Zhu 2017)) is not appropriate for our problem. We compare it with our SLOTH method in Appendix E.

Bottom-Up Attention Module (BU-Attn) This module integrates temporal features of nodes at the bottom levels to their ancestors at top levels to enhance the ability of adapting to dynamic pattern variations including sudden level or trend changes of time series.

TD-Conv carries top-level information downward to improve the predictions at the bottom levels. On the other hand, the information from the bottom-levels should also be useful for the prediction of the top-levels, due to their relationship of value aggregation.

Please note that feature aggregation is different from value aggregation (Eq. (1)). Specifically, the summing matrix ($\mathbf{S}$) is actually a two-level hierarchical structure, rather than a tree structure that is reasonable for value aggregation, but causes structural information loss in feature aggregation. Direct summing operation is inappropriate for feature aggregation as there is no relationship of summation between parents and children in the feature space.

We therefore adopt the attention mechanism (Vaswani et al. 2017; Nguyen et al. 2020) to aggregate temporal features from the bottom levels, due to the following considerations: 1) hierarchical structure has variations (different number of levels and children nodes); 2) child nodes contribute differently to parents with various scales/dynamic patterns. The attention mechanism is appropriate to aggregate the feature as it is flexible for various structures and can learn weighted contributions based on feature similarities.

$$\tilde{h}_t^{\cup_l} = \text{Softmax}\left(\frac{\mathbf{Q}_t^{\cup_l}(\mathbf{K}_t^{\cup_{l+1}})^\mathsf{T}}{\sqrt{d_h}}\right)\mathbf{V}_t^{\cup_{l+1}}, \quad (5)$$
$$\tilde{\mathbf{H}}_t = [\tilde{h}_t^{\cup_1}, \ldots, \tilde{h}_t^{\cup_{n_l}}],$$
$$\mathbf{V}_t^{\cup_l} = \tilde{h}_t^{\cup_l} - \hat{h}_t^{\cup_l} + \bar{h}_t^{\cup_l}. \quad (6)$$

Specifically, as shown in Algorithm 1, attention process starts from the second last to the first level (except for leaf nodes). Parent node takes the original temporal feature $\bar{h}_t$ to generate the query, and the corresponding child node takes the original feature $\bar{h}_t$ to generate the key. Child node feature value $\mathbf{V}_t$ is updated as attention process going upward as in Eq. (6). $\tilde{h}_t^{\cup_l}$ is the feature of all nodes at level l by attention aggregation, which contains the contributions from both children and their ancestors. $\mathbf{V}_t^{\cup_l}$ is attention value of each node at level l, which is used in Eq. (5). Since we not only attempt to strengthen the information of children and the node's own features, but also to weaken the parents' influence in the bottom-up attention process because parent's feature becomes the node's own feature in the next iteration of BU-Attn, therefore, we subtract top-down temporal features $\hat{h}_t^{\cup_l}$ and add original temporal features $\bar{h}_t^{\cup_l}$.

Algorithm 1: Bottom-up Attention Module

Require: $\bar{\mathbf{H}}_t, \hat{\mathbf{H}}_t, \phi_q, \phi_k$
$l \leftarrow n_l - 1, \tilde{h}_t^{\cup_{n_l}} = \hat{h}_t^{\cup_{n_l}}, \mathbf{V}_t = 0$
$\mathbf{V}_t^{n_l} = \hat{h}_t^{\cup_{n_l}}$,
/* $\cup_i$ *means union of nodes at level i*
while $l > 0$ **do**
  $\mathbf{Q}_t^{\cup_l} = F(\bar{h}_t^{\cup_l}; \phi_q)$
  $\mathbf{K}_t^{\cup_{l+1}} = F(\bar{h}_t^{\cup_{l+1}}; \phi_k)$
  Update $\tilde{h}_t^{\cup_l}$ according to Eq. (5)
  Update $\mathbf{V}_t^{\cup_l} = \tilde{h}_t^{\cup_l} - \hat{h}_t^{\cup_l} + \bar{h}_t^{\cup_l}$
  $l \leftarrow l - 1$,
end while
return $\tilde{\mathbf{H}}_t$

It is important to note that computation process at same level can be executed concurrently because it is only related to self temporal hidden feature $\bar{\mathbf{H}}_t$. All the experiments show our bottom-up attention aggregation mechanism carries the bottom-level information containing dynamic patterns to top-level to improve the forecasting performance. More importantly, both TD-Conv and BU-Attn can be independent components to be used in the fitting process (i.e., step t of RNN) for better prediction accuracy, or to be used after the temporal patterns are obtained, trading accuracy for faster computation.

Base Forecast Module This module serves as prediction generation based on dynamic features, and the prediction can either be probabilistic or point estimates. Our framework is agnostic to the forecasting models, such as MLP (Gardner and Dorling 1998), seq2seq (Sutskever, Vinyals, and Le 2014), and attention networks (Vaswani et al. 2017). We employ the MLP to generate the base point estimates for its flexibility and simplicity. In order to avoid the loss of information of temporal features in the cascade of hierarchical learning, we apply residual connection between temporal feature extraction module and base forecast module as follows

$$z_t = \sigma(MLP(\tilde{h}_t)); h_t = (1 - z_t)\tilde{h}_t + z_t\bar{h}_t\,. \quad (7)$$

Then we apply MLP to generate base forecasts as follows

$$\hat{y}_t^i = MLP(h_t^i), \quad \hat{\mathbf{y}}_t = [\hat{y}_t^1, \ldots, \hat{y}_t^n].^1 \quad (8)$$

[1]Note that all nodes share the same generation model.

Task-based Constrained Optimization

In this section, we introduce a task-based optimization module that leverages the deep neural optimization layer to achieve targets in realistic scenarios, while satisfying coherence and task-based constraints.

Optimization with Coherence Constraint in Forecasting Task We formally define HTS forecasting task as a prediction and optimization problem in this section. As shown in Eq. (9), reconciliation on base forecasts can be represented as a constrained optimization problem (Rangapuram et al. 2021), where two categories of constraints are considered, i.e., the equality constraints representing coherency, and the inequality constraints ensuring the reconciliation is restricted, which means the adjustment of the base forecasts is limited in a specific range to reduce the deterioration of forecast performance,

$$\tilde{\mathbf{y}}_t = \underset{\mathbf{y}\in\mathbb{R}^n}{\arg\min} \|\mathbf{y} - \hat{\mathbf{y}}_t\|_2 = \underset{\mathbf{y}\in\mathbb{R}^n}{\arg\min} \frac{1}{2}\mathbf{y}^\top\mathbf{y} - \hat{\mathbf{y}}_t^\top\mathbf{y}$$
$$\text{subject to } \begin{cases} \mathbf{A}\mathbf{y} = \mathbf{0}, \\ \delta_1 abs(\hat{\mathbf{y}}_t) - \varepsilon_1 \le \mathbf{y} - \hat{\mathbf{y}}_t \le \delta_2 abs(\hat{\mathbf{y}}_t) + \varepsilon_2, \end{cases} \tag{9}$$

where $\hat{\mathbf{y}}$ is the base forecasts without reconciliation, and $\delta_i, \varepsilon_i, i = 1, 2$ are some predefined constants.

Recall that the aforementioned end-to-end optimization architecture (Rangapuram et al. 2021) provides a closed-form solution for reconciliation problem. It projects the base forecasts into the solution space effectively by multiplying reconciliation matrix, which only depends on aggregation matrix $\mathbf{S}$ and is thus easy to calculate (for convenience, the details are supplied in Appendix A). However, this procedure only considers aggregation constraints without limiting adjustment scale which may sometime cause the reconciled results $\tilde{\mathbf{y}}$ become unreasonable, e.g., a negative value for small base forecasts in demand forecasting. In addition, loss based on reconciliation projection derails gradient magnitudes and directions, which may cause the model not to converge to the optimum. Moreover, it is not a general solution for real-world scenarios, where more complex task-related constraints and targets are involved.

To keep reconciliation result reasonable and training efficient, we utilize neural network layer OptNet to solve the constrained reconciliation optimization problem, which is essentially a quadratic programming problem. The Lagrangian of formal quadratic programming problem is defined in Eq. (10) (Amos and Kolter 2017), where equality constraints are $\mathbf{A}\mathbf{z} = \mathbf{b}$ and inequality constraints are $\mathbf{G}\mathbf{z} \le \mathbf{h}$:

$$L(\mathbf{z}, \nu, \lambda) = \frac{1}{2}\mathbf{z}^\top\mathbf{Q}\mathbf{z} - \mathbf{q}^\top\mathbf{z} + \nu^\top(\mathbf{A}\mathbf{z} - \mathbf{b}) + \lambda^\top(\mathbf{G}\mathbf{z} - \mathbf{h}). \tag{10}$$

When applied to hierarchical reconciliation problems (where we take the special range constraint $\mathbf{y} \ge \mathbf{0}$ as an example), the Lagrangian can be revised to

$$L(\mathbf{y}, \nu, \lambda) = \frac{1}{2}\mathbf{y}^\top\mathbf{y} - \hat{\mathbf{y}}^\top\mathbf{y} + \nu^\top\mathbf{A}\mathbf{y} + \lambda^\top(-\mathbf{I}\mathbf{y}), \tag{11}$$

where ν, λ are the dual variables of equality and inequality constraints respectively. Then we can derive the differentials of these variables according to the KKT condition, and apply linear differential theory to calculate the Jacobians for backpropagation. The detail is as follows

$$\begin{bmatrix} \mathbf{I} & -\mathbf{I} & \mathbf{A}^\top \\ diag(\lambda)(-\mathbf{I}) & diag(-\mathbf{y}) & \mathbf{0} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} d\mathbf{y} \\ d\lambda \\ d\nu \end{bmatrix} = - \begin{bmatrix} d\mathbf{y} - d\hat{\mathbf{y}} - d\lambda + d\mathbf{A}^\top\nu \\ -diag(\lambda)d\mathbf{y} \\ d\mathbf{A}\mathbf{y} \end{bmatrix}, \tag{12}$$

where function $diag(\cdot)$ means diagonal matrix. We can infer the conditions of the constrained reconciliation optimization problem from the left side, and compute the derivative of the relevant function with respect to model parameters from the right side. In practice, we apply OptNet layer to obtain the solution of argmin differential QPs quickly to solve the linear equation. In this way, our framework achieves end-to-end learning by directly generating reconciliation optimization results, while calculating the derivative and backpropagating the gradient to the optimization model automatically.

Optimization with Task-based Constraints and Target for Real-World Scenarios The coherence constraint is enough in forecasting tasks when the only concern is prediction accuracy, which is, however, most likely unrealistic in real-world tasks with specific limitations and practical targets. Such tasks can be further revised as follows

$$\mathcal{J}(\hat{\mathbf{y}}) = \underset{\mathbf{y}}{\arg\min}\, f(\hat{\mathbf{y}}, \mathbf{y})$$
$$\text{subject to } \begin{cases} \mathbf{A}\mathbf{y} = \mathbf{0}, e_j = 0,\ j = 1, \ldots, n_{\text{eq}}, \\ g_i(\mathbf{y}, \hat{\mathbf{y}}) \le 0, i = 1, \ldots, n_{\text{ineq}}, \end{cases} \tag{13}$$

where f is the task-based quadratic objective, e_j represents task-based equality constraint other than coherence constraint, n_{eq} is the number of equality constraints, and g_i is an inequality constraint, n_{ineq} is the number of task-based inequality constraints. Eq. (13) can be efficiently solved using differential QP layer in an end-to-end fashion (Donti, Amos, and Kolter 2017), where we need to transform our target into a quadratic loss and add equality/inequality constraints. We construct a scheduling experiment on M5 dataset in the following section to validate the superiority of our framework on realistic tasks.

Dataset	levels	nodes	structure	freq
Labour	4	57	1, 8, 16, 32	1M
Tourism	4	89	1, 4, 28, 56	3M
M5	5	114	1, 3, 10, 30 ,70	1D

Table 1: Dataset statistics. Structure column shows the number of nodes at each level from top to bottom, as for 'freq', 1D means one day and 3M means three months.

Experiments

In this section, we conduct extensive evaluations on real-world hierarchical datasets. Firstly, we evaluate the performance of our framework, and compare our approach against the traditional statistical method and end-to-end model (HierE2E). We then add more practical constraints to M5

Model	Tourism		Labour		M5	
Metric	MAPE	w-MAPE	MAPE	w-MAPE	MAPE	w-MAPE
ARIMA-BU	0.2966	0.1212	0.0467	0.0457	0.1134	**0.0638**
ARIMA-MINT-SHR	0.2942	0.1237	0.0506	0.0471	0.1140	0.0675
ARIMA-MINT-OLS	0.3030	0.1254	0.0505	0.0467	0.1400	0.0733
ARIMA-ERM	1.6447	0.6198	0.0495	0.0402	0.1810	0.1163
PERMBU-MINT	0.2947(0.0031)	0.1057(0.0004)	0.0497(0.0003)	0.0453(0.0002)	0.1176(0.0005)	0.0759(0.0007)
DeepAR-Proj	0.3214(0.0202)	0.1171(0.0116)	0.0423(0.0016)	0.0290(0.0013)	0.1546(0.0165)	0.0951(0.0195)
DeepVAR-Proj	0.4214(0.0548)	0.2162(0.0307)	0.0936(0.0206)	0.0884(0.0235)	0.2019(0.0279)	0.1615(0.0311)
NBEATS-Proj	0.3295(0.0231)	0.1359(0.0264)	0.0355(0.0018)	0.0268(0.0043)	0.2256(0.0399)	0.1952(0.0637)
INFORMER-Proj	0.5401(0.0339)	0.5566(0.0261)	0.1537(0.0685)	0.1455(0.0683)	0.3123(0.0735)	0.3098(0.0708)
AUTOFORMER-Proj	0.3983(0.0678)	0.1862(0.0596)	0.0455(0.0037)	0.0367(0.0016)	0.1654(0.0153)	0.1308(0.0560)
FEDFORMER-Proj	0.3741(0.0291)	0.1685(0.0180)	0.0440(0.0038)	0.0334(0.0024)	0.1505(0.0139)	0.1188(0.0044)
DeepAR-BU	0.3065(0.0123)	0.1154(0.0097)	0.0378(0.0014)	0.0278(0.0022)	0.1151(0.0017)	0.0686(0.0012)
DeepVAR-BU	0.4135(0.0562)	0.2195(0.0370)	0.1112(0.0371)	0.1008(0.0352)	0.1851(0.0153)	0.1494(0.0140)
NBEATS-BU	0.2904(0.0308)	0.1259(0.0183)	0.0393(0.0031)	0.0310(0.0046)	0.1740(0.0221)	0.1398(0.0294)
INFROMER-BU	0.5694(0.0065)	0.5707(0.0072)	0.1654(0.0824)	0.1580(0.0840)	0.3128(0.0728)	0.3099(0.0706)
AUTOFORMER-BU	0.3787(0.0578)	0.1868(0.0084)	0.0519(0.0034)	0.0505(0.0011)	0.1506(0.0146)	0.1143(0.0049)
FEDFORMER-BU	0.3408(0.0099)	0.1544(0.0097)	0.0464(0.0041)	0.0369(0.0027)	0.1424(0.0141)	0.1081(0.0034)
SLOTH(Opt)(ours)	0.2613(0.0017)	0.1032(0.0012)	**0.0328(0.0006)**	**0.0183(0.0008)**	**0.1116(0.0018)**	0.0696(0.0017)
SLOTH(Proj)	0.2780(0.0051)	0.1098(0.0008)	0.0370(0.0052)	0.0228(0.0072)	0.1121(0.0014)	0.0704(0.0005)
SLOTH(BU)	**0.2583(0.0015)**	**0.0991(0.0021)**	0.0391(0.0051)	0.0248(0.0065)	0.1127(0.0017)	0.0703(0.0023)

Table 2: MAPE/weighted-MAPE (w-MAPE) values over five independent runs for baselines such as traditional reconciliation methods and end-to-end methods, as well as our approach. Numbers in brackets are the variances over the five runs.

dataset, building a meaningful optimization target to solve an inventory management problem, and again evaluate various approaches for hierarchical tasks under these realistic scenarios. Our framework on a real-world task of cloud resource scheduling in Ant Group is shown in Appendix E.

Real-world Datasets

We take three publicly available datasets with standard hierarchical structures.

- Tourism (Bushell et al. 2001; Athanasopoulos, Ahmed, and Hyndman 2009) includes an 89-series geographical hierarchy with quarterly observations of Australian tourism flows from 1998 to 2006, which is divided into 4 levels. Bottom-level contains 56 series, aggregated-levels contain 33 series, prediction length is 8. This dataset is frequently referenced in hierarchical forecasting studies (Hyndman and Athanasopoulos 2018).
- Labour (Rangapuram et al. 2021) includes monthly Australian employment data from Feb. 1978 to Dec. 2020. By using included category labels, we construct a 57-series hierarchy, which is divided into 4 levels. Specifically, bottom-level contains 57 series, aggregated-levels contain 49 series in total, and prediction length is 8.
- M5 (Han, Dasgupta, and Ghosh 2021) dataset describes daily sales from Jan. 2011 to June 2016 of various products. We construct 5-level hierarchical structure as state, store, category, department, and detail product from the origin dataset, resulting in 70 bottom time series and 44 aggregated-level time series. The prediction length is 8.

Results Analysis

In this section, we validate the overall performance of our method in the prediction task on three public datasets. We report scale-free metrics *MAPE* and scaled metrics *weighted-MAPE* to measure the accuracy. We apply several representative state-of-the-art forecasting baselines (details shown in Appendix B), including DeepAR (Salinas et al. 2020), DeepVAR (Salinas et al. 2019), NBEATS (Oreshkin et al. 2019), and former-based methods (Zhou et al. 2021; Wu et al. 2021; Zhou et al. 2022). We then combine these methods with traditional bottom-up aggregation mechanism and closed-form solution (Rangapuram et al. 2021) (DeepVar-Proj is HierE2E) to generate reconciliation results.

Table 2 reports the results. The top part shows results of traditional statistic methods, the middle part is of deep neural networks methods with closed-formed solution and bottom-up reconciliation, and the bottom part is the results from our approach and the combination of our forecasting mechanism with traditional bottom-up (BU) and closed-form projection methods (Proj).

We can see that traditional statistic methods perform poorly compared with deep neural networks. NBEATS performs best on Tourism dataset. In particular, NBEATS-BU performs best on MAPE while NBEATS-Proj performs best on weighted-MAPE. However, informer-related methods perform poorly. One possible explanation is that it requires much larger training dataset.

One can observe that the models performing best on MAPE do not perform as well on weighted-MAPE, which is caused by different level contributions on the overall performance, e.g., the bottom-level contributes more on MAPE but higher levels contribute more on weighted-MAPE.

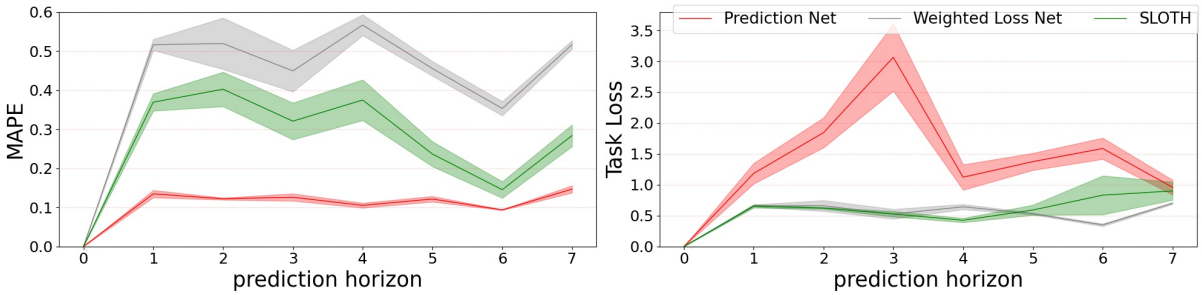

Figure 4: Results of 5 independent runs of M5 scheduling problem for 7-day prediction (lower MAPE and lower task loss are better). As expected, the network using prediction loss achieves the highest accuracy. Our task-net improves the prediction performance by 36.8% compared to weighed loss net, and outperforms the prediction loss net by 53.2%.

Our proposed approach (SLOTH) has superior performance compared to other methods on both MAPE and weighted-MAPE metrics in most scenarios. Specifically, SLOTH achieves best performance among all models on Labour and M5 datasets, and ranks the second on Tourism dataset and the third on w-MAPE for M5. Besides optimization reconciliation, we also combine our forecasting mechanism with the aforementioned bottom-up and closed-form projection methods, and both methods achieve higher accuracy than the baselines. Even SLOTH-BU, with primitive structure, achieves the best performance in Tourism dataset. Please note that smaller variances indicate that our framework shows stabler performances across various scenarios.

In conclusion, our SLOTH mechanism improves the forecasting performance, and optimized reconciliation generates reasonable coherent predictions without much performance loss. We also assess the gains in the performance across all levels and running time in Appendix D, and the ablation study for each component is presented in Appendix E.

M5 Scheduling Task

In this section, we apply our framework to realistic scenarios with meaningful task-based constraints and targets, using a designed scheduling task for product sales based on M5 dataset. Specifically, we define a meaningful task that minimize the cost of scheduling and inventory with more practical conditions:

- Underestimation and overestimation contribute differently to the final cost. Underestimation implies that the store needs to order more commodity to fulfill the demand, which increases scheduling cost. Overestimation implies that the store has to keep the extra commodity, which increases inventory cost.
- Different levels generate different weighted contribution due to aggregation, e.g., scheduling and inventory costs for the top-levels are less than those for the bottom-levels because the company scale is larger at the top-levels.
- Commodities of different types have different inventory and scheduling costs, since the shelf life for food products is shorter than household products.

Settings. We assume scheduling takes place every week. We set prediction length to 7 and context length to 14. The other penalty settings and target are detailed in Appendix F. We then compare the outcome from the following models:

1. Prediction Net: prediction model that takes the prediction metric (MAE) as the loss for optimization, and bottom-up approach for coherency.
2. Weighted Loss Net: prediction model that takes task cost (weighted-MAE) as the loss for optimization, and bottom-up for coherency.
3. SLOTH: our end-to-end approach that takes both cost and constraints, with the task loss for optimization.

As shown in Figure 4, the prediction Net model performs the best on prediction (MAPE), which is the training objective. As for the task cost, which the scheduler really cares in practice, our SLOTH framework outperforms the Prediction Net by a large margin. Specifically, our model improves the task-cost performance by 36.8% compared to the Prediction Net, while at the same time achieving a similar task loss target as the Weighted Loss Net, but with an improvement of 53.2% in the prediction accuracy.

Conclusion

In this paper, we introduced a novel task-based structure-learning framework (SLOTH) for HTS. We proposed two tree-based mechanisms to utilize the hierarchical structure for HTS forecasting. The top-down convolution integrates the temporal feature of the top-level to enhance stability of dynamic patterns, while the bottom-up attention incorporates the features of the bottom-level to improve the coherency of the temporal features. In the reconciliation step, we applied the deep neural optimization layer to produce the controllable coherent result, which also accommodates complicated realistic task-based constraints and targets under coherency without requiring any explicit post-processing step. We unified the goals of forecasting and decision-making and achieved an end-to-end framework. We conducted extensive empirical evaluations on real-world datasets, where the competitiveness of our method under various conditions against other state-of-the-art methods were demonstrated. Furthermore, our ablation studies proved the efficacy of each component we designed. Our method has also been deployed in the production environment in Ant Group for its cloud resources scheduling.

References

Amos, B.; and Kolter, J. Z. 2017. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, 136–145. PMLR.

Anderer, M.; and Li, F. 2021. Forecasting reconciliation with a top-down alignment of independent level forecasts. arXiv:2103.08250.

Athanasopoulos, G.; Ahmed, R. A.; and Hyndman, R. J. 2009. Hierarchical forecasts for Australian domestic tourism. *International Journal of Forecasting*, 25(1): 146–166.

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

Bushell, R.; Prosser, G. M.; Faulkner, H. W.; and Jafari, J. 2001. Tourism research in Australia. *Journal of Travel Research*, 39(3): 323–326.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555.

Corani, G.; Azzimonti, D.; Augusto, J. P.; and Zaffalon, M. 2020. Probabilistic reconciliation of hierarchical forecast via Bayes' rule. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 211–226. Springer.

Dangerfield, B. J.; and Morris, J. S. 1992. Top-down or bottom-up: Aggregate versus disaggregate extrapolations. *International journal of forecasting*, 8(2): 233–241.

Donti, P.; Amos, B.; and Kolter, J. Z. 2017. Task-based end-to-end model learning in stochastic optimization. *Advances in Neural Information Processing Systems*, 30.

Gardner, M. W.; and Dorling, S. 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14-15): 2627–2636.

Han, X.; Dasgupta, S.; and Ghosh, J. 2021. Simultaneously Reconciled Quantile Forecasting of Hierarchically Related Time Series. In *International Conference on Artificial Intelligence and Statistics*, 190–198. PMLR.

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

Hyndman, R. J.; and Athanasopoulos, G. 2018. *Forecasting: Principles and Practice*. OTexts.

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

Jeon, J.; Panagiotelis, A.; and Petropoulos, F. 2018. Reconciliation of probabilistic forecasts with an application to wind power. arXiv:1808.02635.

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

Liu, Z.; Yan, Y.; and Hauskrecht, M. 2018. A flexible forecasting framework for hierarchical time series with seasonal patterns: A case study of web traffic. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 889–892.

Mou, L.; Li, G.; Zhang, L.; Wang, T.; and Jin, Z. 2016. Convolutional neural networks over tree structures for programming language processing. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Nguyen, X.-P.; Joty, S.; Hoi, S. C. H.; and Socher, R. 2020. Tree-structured Attention with Hierarchical Accumulation. arXiv:2002.08046.

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

Rangapuram, S. S.; Werner, L. D.; Benidis, K.; Mercado, P.; Gasthaus, J.; and Januschowski, T. 2021. End-to-end learning of coherent probabilistic forecasts for hierarchical time series. In *International Conference on Machine Learning*, 8832–8843. PMLR.

Salinas, D.; Bohlke-Schneider, M.; Callot, L.; Medico, R.; and Gasthaus, J. 2019. High-dimensional multivariate forecasting with low-rank gaussian copula processes. arXiv:1910.03002.

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

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Taieb, S. B.; and Koo, B. 2019. Regularized regression for hierarchical forecasting without unbiasedness conditions. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1337–1347.

Taieb, S. B.; Taylor, J. W.; and Hyndman, R. J. 2021. Hierarchical probabilistic forecasting of electricity demand with smart meter data. *Journal of the American Statistical Association*, 116(533): 27–43.

Van Den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A. W.; and Kavukcuoglu, K. 2016. WaveNet: A generative model for raw audio. *SSW*, 125: 2.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 5998–6008.

Wickramasuriya, S. L.; Athanasopoulos, G.; and Hyndman, R. J. 2019. Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. *Journal of the American Statistical Association*, 114(526): 804–819.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34: 22419–22430.

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

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of AAAI*.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. *Proceedings of the 39th International Conference on Machine Learning*, 162: 27268–27286.