FRT: Flow-based Reconcile Transformer for Hierarchical Time Series

Shiyu Wang*†‡
kwuking@gmail.com
Ant Group
Hangzhou, China

Wei Lu
xiaobo.lw@antgroup.com
Ant Group
Hangzhou, China

Jiawei Li*
jiawei.li@it.uu.se
Uppsala University
Uppsala, Sweden

Xiaoming Shi
sxm728@hotmail.com
Ant Group
Hangzhou, China

Xinyue Zhong
lesley.xinyuezhong@gmail.com
Ant Group
Hangzhou, China

Zhou Ye
yezhou199032@gmail.com
Alibaba
Hangzhou, China

Ming Jin†
mingjinedu@gmail.com
Griffith University
Brisbane, Australia

Qingsong Wen†
qingsongedu@gmail.com
Squirrel Ai Learning
Bellevue, United States

Abstract

Many real-world applications contain data in the form of multivariate *time series* (TS) with the hierarchical structure, where classic methods forecasting each TS independently are inadequate for coherency (i.e., satisfying the hierarchical aggregation constraints). Furthermore, the discrepancies between statistical properties of different levels can be huge, exacerbated by non-Gaussian distributions and non-linear correlations. In this paper, we propose a novel end-to-end hierarchical TS forecasting model, i.e., a Flow-based Reconcile Transformer (FRT). FRT employs a conditional normalizing flow-based autoregressive transformer, to represent complex data distribution, while simultaneously reconciling the forecasts to ensure coherency. Go beyond other state-of-the-art methods, FRT accomplishes forecasting and reconciliation simultaneously, while avoiding any post-processing steps. Moreover, FRT is a deep model that does not rely on any strong assumptions such as unbiased estimates or Gaussian distribution. Our experiments are conducted on four real-world hierarchical datasets from different industrial domains (three public ones and a dataset from the application servers of our company's data center) and the results demonstrate the efficacy of our proposed method. Our method has been implemented extensively within the production environments of a prominent global payment company. It has emerged as a cornerstone for workload forecasting within their data center and plays a critical role in the optimization of cloud computing resource allocation across the entire cluster. This successful deployment of the application demonstrates that our approach achieves precise hierarchical workload prediction, which is of great significance for efficient resource scheduling, enhancing resource utilization, and reducing server resource use and energy consumption in data centers.

CCS Concepts

• **Information systems** → **Temporal data**; • **Mathematics of computing** → **Time series analysis**; • **Computing methodologies** → **Neural networks**.

Keywords

hierarchical time series; transformer; neural network

ACM Reference Format:
Shiyu Wang, Wei Lu, Jiawei Li, Xiaoming Shi, Xinyue Zhong, Zhou Ye, Ming Jin, and Qingsong Wen. 2025. FRT: Flow-based Reconcile Transformer for Hierarchical Time Series. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3737224

1 Introduction

Multivariate *time series* (TS) forecasting with the hierarchical structure, such as department sales of multiple stores at different locations and traffic flow in hierarchical regions, is widespread in real-world applications [1, 8, 18, 31, 36, 48, 55], which involves simultaneously forecasting multiple hierarchically related TS via aggregation operations. These TS not only interact with each other in the hierarchy but also imply coherency, i.e., TS at upper levels are the aggregation/summation of those at lower levels. For instance, in the hierarchical structure of Australian domestic tourism demand [4], the data contains 4 levels from top to bottom, including 1 country, 7 states, 27 zones, and 82 regions, among which different levels of forecasts have distinct business goals. Specifically, bottom-level forecasts are often about specific demands to help with regional government decisions, while upper-level forecasts look at

*Both authors contributed equally to this research.
†The corresponding author.
‡The project lead.

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3737224

the macro perspective to assist national strategies. Besides the general statistical disparities between levels in real-world hierarchies, the entanglement of their interactions and correlations presents a great challenge to the prediction model. Coherency constraints [49, 60] also add more complication to the prediction model.

The straightforward methods to utilize hierarchical structure include *bottom-up* forecasting and *top-down* forecasting, which, as their names suggest, make individual predictions at the bottom or upper levels and then aggregate according to the hierarchical structure. Although these methods naturally satisfy coherency, they are unable to adapt to varying statistical properties of all levels. Specifically, when forecasting from a single level of the aggregation structure, these methods either aggregated or disaggregated to obtain forecasts of all the rest levels, ignoring the fact that the statistical characteristics at different levels can be drastically distinct, e.g., the TS of upper levels tend to be more stationary, while those at the bottom levels are often more fluctuant.

To address these issues, the reconciliation method [15] is proposed, taking a two-stage approach: 1) independently forecast all TS (generating incoherent base forecasts); 2) reconcile the base forecast by applying forecast combination using a bottom-up procedure. In other words, the base forecasts are modified for coherency. MinT [56] is a typical example, which reconciles the base forecasts via the optimal combination with minimum variance among all unbiased revised forecasts.

Most previous works that reconcile forecasts of all TS to ensure coherency, face the following challenges: **(1)** The base forecast is obtained independently without any shared information from other TS. **(2)** State-of-the-art methods rely on strong statistical assumptions, such as unbiased forecasts and Gaussian noises, but the data distributions in real-world are mostly non-Gaussian/non-linear across the hierarchy, which calls for a method that can transform data into Gaussian space where tractable methods can be applied. **(3)** The two-stage approaches reconcile the base forecast without any regard to the learned model parameters and cannot utilize deep parametric models that can share information between the processes of prediction and reconciliation. **(4)** Most methods only focus on generating point estimates, but probabilistic forecasts are often necessary in practice to facilitate the subsequent decision-making processes.

In this paper, we present a novel end-to-end approach, i.e., a Flow-based Reconcile Transformer (FRT) that tackles forecasting and reconciliation simultaneously for hierarchical TS and avoids explicit post-processing steps. The recently popular *autoregressive transformer* [20] and *conditional normalizing flow* (CNF) [27] are combined to obtain coherent probabilistic forecasts.

Specifically, the base forecast via the autoregressive transformer is **firstly** obtained, modeling the multivariate TS of all levels. Using an encoder-decoder transformer structure, which has been successful in recent advances in multivariate TS forecasting [61], the information fusion of all levels in the hierarchy via globally shared parameters is achieved while benefiting from the representation power of the autoregressive transformer model. Transformer models have also shown superior performance in capturing long-range dependency than RNN-based models [58]. **Second**, the base forecasts are reconciled into coherent forecasts via conditional normalizing flow [27] with a bottom-up aggregation matrix. Due to the complex probabilistic distributions in the hierarchical data, normalizing flow (NF), a proven powerful density approximator, becomes our natural choice. By extending NF to the conditional case, base forecasts from all levels can be incorporated as additional conditions for the latent space, leveraging the information available across all levels for reconciliation, while modeling the non-Gaussian distributions and non-linear correlations in the hierarchy to obtain probabilistic forecasts. **Finally**, forecasting and reconciliation are combined simultaneously for end-to-end training, while ensuring continuity and differentiability at all steps. Moreover, our framework accommodates different loss functions besides log-likelihood, and by sampling from the forecasted distribution, sufficient statistics can be obtained via the empirical distribution to facilitate complicated optimization objectives.

Our Contributions. We summarize the contributions of our proposed Flow-based Reconcile Transformer (FRT) as follows:

- A novel reconcile transformer via conditional normalizing flow with bottom-up aggregation matrix to model hierarchical TS, which integrates information of all levels in the hierarchy for coherent probabilistic forecasting without relying on any statistical assumption
- An end-to-end learning framework of hierarchical TS achieving forecasting and reconciliation simultaneously, without requiring any explicit post-processing step
- A large-scale deployment at a world-leading payment company for data center workload forecasting to support its cloud computing resource scheduling, along with extensive experiments on real-world hierarchical datasets from various industrial domains demonstrating the superior performance of FRT.

In the following content, we introduce the involved background and related work in Section I and Section II. We then present our proposed method and describe the procedure of training and inference in Section III. Next, in Section IV, we analyze the non-Gaussian/non-linear properties of real-world hierarchical data and demonstrate the advantages of our approach through experiments and ablation studies. Finally, in Section V, we have provided a detailed introduction to the application practices of FRT in large-scale deployments at world-leading payment companies. It has now become a cornerstone for workload forecasting in the company's data centers.

2 Related Work

2.1 Hierarchical Time Series Forecasting

Existing hierarchical time series forecasting methods mainly follow the two-stage approach: **(i)** Obtain each h-period-ahead base forecasts $\hat{\boldsymbol{y}}_{T+h}$ independently; **(ii)** Reconcile the base forecasts by the reconciliation process (Equation 1) to obtain coherent forecasts $\tilde{\boldsymbol{y}}_{T+h}$. This approach has the following advantages: **(1)** The forecasts are coherent by construction; **(2)** The combination of forecasts from all levels is applied via the projection matrix $\boldsymbol{P}$, where information from all levels of hierarchy is incorporated simultaneously. The main work of the current state-of-the-art method is to improve the reconciliation process, which will be reviewed in this section.

The reconciliation process in hierarchical time series can be represented as

$$\tilde{y}_{T+h} = \boldsymbol{SP}\hat{y_{T+h}} \,. \tag{1}$$

where $\boldsymbol{P} \in \mathbb{R}^{m\times n}$ is a matrix that projects the base forecasts (of dimension n) into the bottom-forecast(of dimension m), which is then summed up by the aggregation matrix $\boldsymbol{S} \in \{0,1\}^{n\times m}$ using the aggregation structure to produce a set of *coherent forecasts* $\tilde{y}_{T+h} \in \mathbb{R}^n$, which satisfy the aggregation constraints.

Wickramasuriya *et al.* [56] proposed **MinT** to optimally combine the base forecasts. Specifically, assuming the base forecasts $\hat{y}_{T+h}$ are unbiased, Mint computes the projection matrix as $\boldsymbol{P} = (S^T W_h^{-1} S)^{-1}(S^T W_h^{-1})$ giving the minimum variance unbiased revised forecasts, i.e., minimizing $tr[SPW_hP^TS^T]$ with constraint $SPS = S$, where W_h is the covariance matrix of the h-period-ahead forecast errors $\hat{\epsilon}_{T+h} = y_{T+h} - \hat{y}_{T+h}$ ($\boldsymbol{S}$ and $\boldsymbol{P}$ are matrices defined in eq. (1)). However, the covariance of errors W_h is hard to obtain for general h and the strong assumption of unbiased base forecasts is normally unrealistic.

The unbiased assumption is relaxed in **Regularized Regression for Hierarchical Forecasting** [2], which also follows the two-stage approach and seeks the revised forecasts with the trade-off between bias and variance of the forecast by solving an empirical risk minimization (**ERM**) problem.

Probabilistic Method for Hierarchical Forecasting [40, 51] also employs the two-stage approach, but in contrast to the above methods, it considers forecasting probability distributions rather than just the means (point forecasts). This probabilistic method starts by generating independent forecasts of the conditional marginal distributions, followed by samplings from the above distributions as the base forecasts, which are then reconciled using Equation 1. In this approach, existing reconciliation methods for point forecasts can be extended to a probabilistic setting. However the reconciliation is only applied to the samples rather than the forecast distribution, which creates another level and uncertainty with unsalable computation complexity.

One typical work in end-to-end modeling of hierarchical time series is **Hier-E2E** [44], where base forecasts are obtained using DeepVAR [46] with diagonal Gaussian distribution, followed by reconciliation using a closed-form formulation of an optimization problem, i.e., minimizing the errors of the base forecast $\hat{y}$ and reconciling forecast $\tilde{y}$ subject to coherent constraints of hierarchical structure. The reconciliation process is as follows:

$$\tilde{y_t} = M\hat{y}_t$$
$$M := I - A^T(AA^T)^{-1}A \,,$$

where M is a fixed matrix and A is a structure matrix from the upper half of the aggregated matrix. In contrast to the previous work, the model no longer has a relationship with the predicted value y. The matrix M is time-invariant and can be computed offline before the training. This reconciliation approach is essentially a fine-tuning of the base forecasts under the coherence constraint, and the learned model parameters are not used to revise the base forecast in the reconciliation stage.

Considering the pros and cons of the above models, we combine the advantages and propose a novel end-to-end model that can not only obtain coherent probabilistic forecasts but can also achieve the reconciliation in the forecast distributions (rather than just on samples), ensuring that the reconciliation is related the predicted value y via integrating information from all levels, to improve the overall performance.

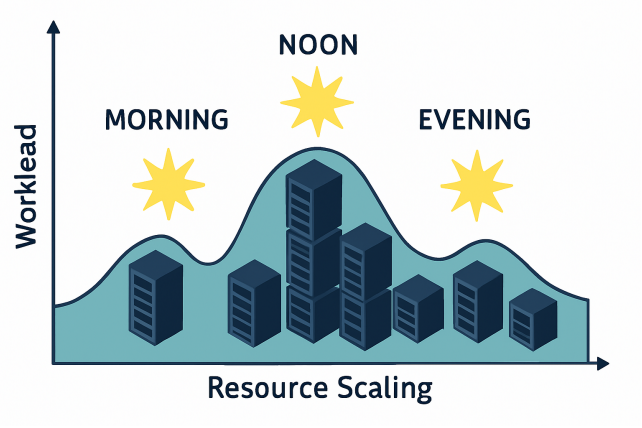

Figure 1: An illustration of resource scaling based on workload forecasting

2.2 Normalizing Flow

Normalizing flows (NF) [41], which learn a distribution by transforming the data to samples from a tractable distribution where both sampling and density estimation can be efficient and exact, have been proven to be powerful density approximators. The change of variables formula (Equation 2 below) empowers the computation of exact likelihood [25], which is in contrast to other powerful density estimator such as Variational Autoencoders or Generative Adversarial Networks. Impressive estimation results, especially in the field of nonlinear high-dimensional data generation, have led to the great popularity of flow-based deep models [22, 53].

NF are invertible neural networks that typically transform isotropic Gaussians to characterize a more complex data distribution. They map from $\mathbb{R}^D$ to $\mathbb{R}^D$ such that densities p_Y on the input space $Y \in \mathbb{R}^D$ are transformed into some tractable distribution p_Z (e.g., an isotropic Gaussian) on space $Z \in \mathbb{R}^D$. This mapping function, $f : Y \to Z$ and inverse mapping function, $f^{-1} : Z \to Y$ are composed of a sequence of bijections or invertible functions, and we can express the target distribution densities $p_Y(\boldsymbol{y})$ by

$$p_Y(\boldsymbol{y}) = p_Z(\boldsymbol{z})|det(\frac{\partial f(\boldsymbol{y})}{\partial \boldsymbol{y}})| \,, \tag{2}$$

where $\partial f(y)/\partial y$ is the Jacobian of f at y. NF have the property that the inverse $y = f^{-1}(\boldsymbol{z})$ is easy to evaluate and computing the Jacobian determinant takes $O(D)$ time.

For mapping functions f, the bijection introduced by RealNVP architecture (the coupling layer) [12] satisfies the above properties, leaving part of its inputs unchanged, while transforming the other part via functions of the un-transformed variables (with superscript denoting the coordinate indices)

$$\begin{cases} y^{1:d} = z^{1:d} \\ y^{d+1:D} = z^{d+1:D} \odot exp(s(z)^{1:d} + t(z^{1:d})) \end{cases} , \tag{3}$$

where $\odot$ is an element wise product, $s()$ is a scaling and $t()$ is a translation function from $\mathbb{R}^D \mapsto \mathbb{R}^{D-d}$, using neural networks. To model a nonlinear density map $f(x)$, a number of coupling layers, mapping $\mathcal{Y} \mapsto \mathcal{Y}_\infty \mapsto ... \mapsto \mathcal{Y}_{\mathcal{K}-\infty} \mapsto \mathcal{Y}_\mathcal{K} \mapsto \mathcal{Z}$, are composed together with unchanged dimensions.

Conditional Normalizing Flow. Inspired by the conditional extension of NF [27], with the conditional distribution $p_Y(\boldsymbol{y}|\boldsymbol{h})$, we realize that the scaling and translation function approximators do not need to be invertible [27], which means we can make the transformation dependant on condition $\boldsymbol{h} \in \mathbb{R}^H$. Implementing $p_Y(\boldsymbol{y}|\boldsymbol{h})$ on $\boldsymbol{h}$ is straight-forward: we concatenate h to both the inputs of the scaling and translation function approximators of the coupling layers, i.e., $s(concat(\boldsymbol{z}^{1:d}, \boldsymbol{h}))$ and $t(concat(\boldsymbol{z}^{1:d}, \boldsymbol{h}))$, which are modified to map $\mathbb{R}^{d+H} \mapsto \mathbb{R}^{D-d}$.

2.3 Workload Forecasting

The rapid expansion of cloud computing has significantly changed the landscape of IT services, leading to an urgent need for efficient resource management within cloud systems [35, 59]. As businesses increasingly rely on cloud services, the ability to accurately predict future workloads has become a cornerstone of effective resource scheduling [33, 52]. This capability not only improves operational efficiency but also ensures that resources are optimally allocated to meet demand, thereby improving service quality [10, 11, 32].

In microservices architectures, one of the primary metrics for assessing service capacity is Requests Per Second (RPS), which serves as a vital indicator of workload intensity. Accurately forecasting RPS is essential for anticipating future demands and planning resource scaling accordingly. This need has prompted extensive research into various workload forecasting methodologies.

Major cloud service providers have recognized the significance of workload forecasting and have launched their own resource scaling service frameworks. For instance, Google's Autopilot [45] leverages the Autoregressive Integrated Moving Average (ARIMA) model [3], while Microsoft's FIRM [43] employs historical statistical methods to anticipate workload changes. Amazon's AWS Autoscaling incorporates advanced techniques such as DeepAR [47] and DeepVar [46], a deep learning-based approach, to enhance forecast accuracy. These diverse methodologies reflect the growing complexity of cloud environments and the increasing importance of tailored forecasting solutions for specific workloads [6, 26].

2.4 Resource Scaling

Resource scaling is a critical aspect of cloud computing that addresses the dynamic needs of applications and services (in the Figure 1). As organizations increasingly rely on cloud infrastructure, the ability to adjust resources in real-time becomes essential for maintaining performance and cost-efficiency. Autoscaling, a key feature of cloud environments, allows for the automatic adjustment of computing resources, such as virtual machines (VMs) or pods (containers hosting microservices), based on pre-defined criteria like wordload levels and utilization rates. This capability not only enhances operational efficiency by reducing the need for manual intervention during workload fluctuations but also ensures that resources are allocated optimally. By automatically scaling resources up or down, businesses can better handle peak loads while minimizing costs during periods of low demand, thus achieving a balance between performance and expenditure.

3 Methodology

In this section, we detail our proposed Flow-based Reconcile Transformer (FRT) for coherent probabilistic forecasting.

3.1 Model

A schematic overview of our hierarchical end-to-end architecture can be found in Figure 2.

3.1.1 ***All Levels Base Forecasts*** . First, we generate *all-levels base forecasts* with the transformer. The encoder-decoder transformer architecture has been highly successful in advancing the research on multivariate TS forecasting, enabled by its multi-head self-attention mechanism to capture both long- and short-term dependencies, and can be further extended to have autoregressive properties by using *causal masking*. These advantages motivate the use of *multivariate autoregressive transformer* as our building block to better capture patterns (e.g., trends and cycles) inside each TS. Note that the global parameters of the transformer are shared across different TS to exploit common patterns over the entire history.

We denote the entities of a hierarchical TS by $y_{t,i} \in \mathbb{R}$ for $i \in \{1, 2, ..., n\}$, where t is the time index. We consider TS with $t \in [1, T]$, sampled from the complete history of our data, where for training we split the sequence by some context window $[1, t_0)$ and prediction window $[t_0, T]$. We use $x_{t,i}$ to denote time-varying covariate vectors associated with each univariate TS i at time step t.

In the encoder-decoder transformer architecture, the encoder embeds $\boldsymbol{y}_{1:t_0-1}$ and the decoder outputs the base forecast of all levels as a condition for the density estimations over $\boldsymbol{y}_{t_0:T}$ via a masked attention module:

$$\begin{aligned} \boldsymbol{c}_{t_0} &= TransformerEncoder(concat(\boldsymbol{y}_{1:t_0-1}, \boldsymbol{x}_{1:t_0-1}); \theta) \\ \hat{\boldsymbol{y}}_t &= TransformerDecoder(concat(\boldsymbol{y}_{t_0:t-1}, \boldsymbol{x}_{t_0:t-1}), \boldsymbol{c}_{t_0}; \phi) \ , \end{aligned} \tag{4}$$

where θ and ϕ are parameters of the transformer's encoder and decoder, respectively. These parameters are shared globally, achieving information fusion across all levels in the hierarchy to generate the *base forecasts* $\hat{\boldsymbol{y}}_t$.

During training, care has to be taken to prevent using information in the future. Specifically, to ensure the autoregressive property of the model, we employ a mask that reflects the causal direction of the progressing time, i.e. to mask out future time points.

Note that, in our case, base forecasts $\hat{\boldsymbol{y}}_t$ do not directly correspond to un-reconciled forecasts for the base series, but rather represent predictions of unobserved latent states, which can be used for subsequent density estimations. The transformer allows the model to access any part of the historic TS regardless of temporal distance. It thus is potentially able to generate better condition $\boldsymbol{h}_t \in \mathbb{R}^H$ for the subsequent density estimations.

3.1.2 ***Coherent Probabilistic Forecasts*** . Next, we describe how to obtain the *coherent probabilistic forecasts*, given the base forecasts described above from the autoregressive transformer.

To estimate the probability density of data (to obtain a probabilistic forecast), one straightforward method is to use parameterized Gaussian distribution, but as mentioned above, the real-world hierarchical data are mostly non-Gaussian/non-linear. Equipped with

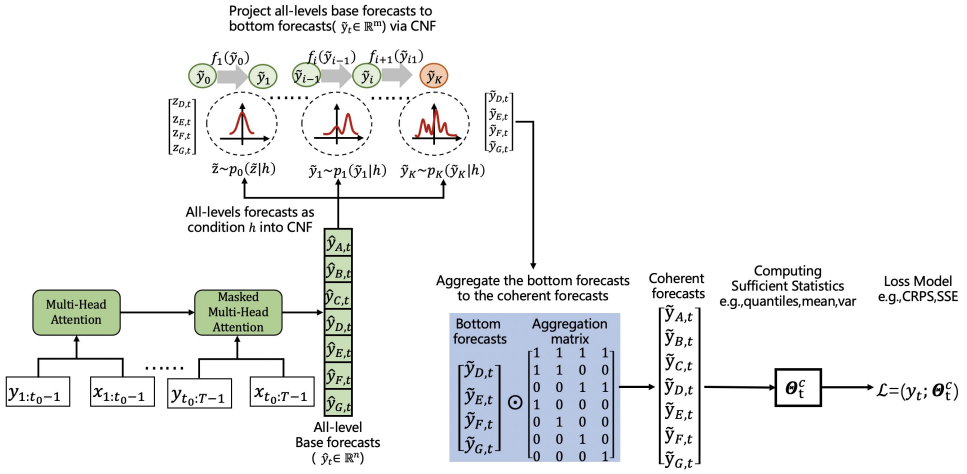

Figure 2: Model Architecture. Sufficient statistics can be computed from the samples via the empirical distribution to facilitate more complicated optimization objectives.

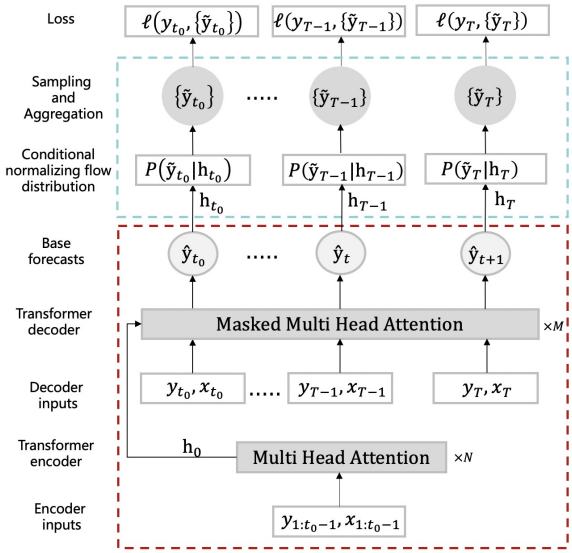

Figure 3: Training Stage. The red dashed line represents the multivariate autoregressive transformer; Reconciliation via conditional NF with bottom-up aggregation matrix is highlighted.

the powerful density approximator, *normalizing flow* (NF), we can tackle this challenge, capturing the nonlinear relationships among all levels in the hierarchy.

According to the definition of Equation 1, the reconciliation takes a projection matrix $\boldsymbol{P} \in \mathbb{R}^{m \times n}$, projecting from the base forecasts (of dimension n) into the bottom-forecasts (of dimension m). In our approach, this projection is replaced using *conditional normalizing flow* (CNF), i.e., the conditional joint distribution $p_Y(\tilde{\boldsymbol{y}}_t|\boldsymbol{h}_t)$, where $\boldsymbol{h}_t$ is the condition (base forecasts $\hat{\boldsymbol{y}}_t$ in our case) and the current $\tilde{\boldsymbol{y}}_t$ is the reconciled bottom-forecasts (of dimension m).

In the Real-NVP architecture [12], we extend the method by concatenating condition $\boldsymbol{h}_t$ to both the inputs of the scaling and translation function approximators of the coupling layers as follows:

$$\begin{cases} \boldsymbol{y}^{1:d} = \boldsymbol{z}^{1:d} \\ \boldsymbol{y}^{d+1:D} = \boldsymbol{z}^{d+1:D} \odot exp(s(\boldsymbol{z}^{1:d}, \boldsymbol{h}) + t(\boldsymbol{z}^{1:d}, \boldsymbol{h})) \end{cases}, \tag{5}$$

where $\boldsymbol{z}$ is a noise vector sampled from an isotropic Gaussian, functions s (scale) and t (translation) are usually deep neural networks, which, as mentioned above, do not need to be invertible.

To obtain an expressive distribution representation, we can stack K layers of conditional flow modules (Real-NVP), generating the conditional distribution of the future sequences of all TS in the hierarchy, given the past time $t \in [1, t_0)$ and the covariates in $t \in [1, T]$. Specifically, it can be written as a product of factors (as an autoregressive model):

$$p_Y(\tilde{\boldsymbol{y}}_{t_0:T}|\boldsymbol{y}_{1:t_0-1}, \boldsymbol{x}_{1:T}; \theta, \phi, \psi) = \prod_{t=t_0}^{T} p_Y(\tilde{\boldsymbol{y}}_t|\boldsymbol{h}_t; \theta, \phi, \psi) , \tag{6}$$

where θ and ϕ are parameters of the transformer and ψ is the parameter of conditional NF.

Then with the power of reparameterization trick [23], we can generate directly a set of Monte Carlo samples from the above conditional joint distribution $p_Y(\tilde{\boldsymbol{y}}_t|\boldsymbol{h}_t)$ as the reconciled bottom forecasts, e.g., $[\tilde{y}_{D,t}, \tilde{y}_{E,t}, \tilde{y}_{F,t}, \tilde{y}_{G,t}]$ in Figure 2. According to Equation 4 ($\tilde{y}_t = \boldsymbol{S}\boldsymbol{P}\hat{y}_t$), the bottom forecasts are multiplied by the aggregation matrix S to obtain the coherent probabilistic forecasts $\tilde{\boldsymbol{y}}_t$ of all levels.

Our approach unifies prediction and reconciliation with deep parametric models, facilitating information sharing in the process described above via global parameters. Our reconciliation method not only ensures hierarchical coherence constraints but also dynamically revises the base forecast by integrating information of all levels, to improve the overall performance. Unlike the current probabilistic method for hierarchical forecasting, we apply the projection matrix P on the conditional joint distribution of CNF, rather than samples of distribution. In other words, our model conducts reconciliation in the forecast distribution rather than just sample points, which makes the estimation exact and more efficient.

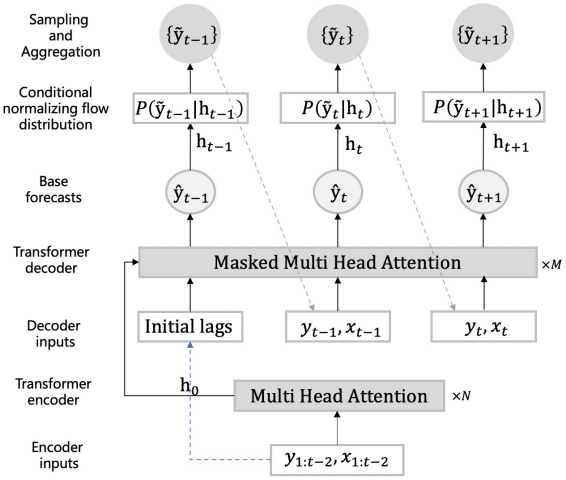

Figure 4: Inferencing Stage. In the multi-step inference, the dashed line indicates that the prediction output from the previous step feeds the inference of the next step via masked attention.

Moreover, our framework can accommodate different loss functions besides log-likelihood, and by sampling from the forecast distribution, we can also obtain sufficient statistics via the empirical distribution to facilitate more complicated optimization objectives.

3.2 Training

The overall training process is shown in Figure 3. During training, our loss function is directly computed on the coherent forecast samples. Specifically, given $\mathcal{D}$, defined as a batch of TS $Y := \{y_1, y_2, ..., y_T\}$, and the associated covariates $X := \{x_1, x_2, ..., x_T\}$, we can maximize the likelihood given by Equation 6 via Adam optimizer [21], i.e.

$$\begin{aligned}\mathcal{L} &= \frac{1}{|\mathcal{D}|T} \prod_{x_{1:T}, y_{1:T} \in \mathcal{D}} \prod_{t=1}^{T} p_Y(y_t | y_{1:t-1}; x_{1:t}, \theta, \phi, \psi) \\ &= \frac{1}{|\mathcal{D}|T} \prod_{x_{1:T}, y_{1:T} \in \mathcal{D}} \prod_{t=1}^{T} p_Y(y_t | h_t, \theta, \phi, \psi)\end{aligned}, \quad (7)$$

where the globally shared parameters (θ, ϕ) and ψ are from the transformer and the conditional NF modules, respectively.

Please note that we can easily obtain θ, ϕ, and ψ for other loss functions such as quantile loss, CRPS (continuously ranked probability score) or any other metrics preferred in the forecasting community, as long as we can compute the sufficient statistics from the Monte Carlo samples $\{\tilde{y}_t\}$ via the empirical distribution function.

3.3 Inference

The overall inference process is shown in Figure 4. During the inference, we can predict them using the autoregressive transformer in a step-by-step fashion over the time horizon. Specifically, we first generate the base forecasts $\hat{y}_t$ using an autoregressive transformer for one time step using the covariate vector $x_{1:t-1}$ and the observed value $y_{1:t-1}$. Then we can incorporate base forecasts $\hat{\boldsymbol{y}}_t$ from all levels into conditional NF as additional condition $\boldsymbol{h}_t$ in the latent space to model conditional joint distribution $p_Y(\tilde{\boldsymbol{y}}_t|\boldsymbol{h}_t)$. Lastly, we directly obtain a set of Monte Carlo samples from the distribution $p_Y(\tilde{\boldsymbol{y}}_t|\boldsymbol{h}_t)$ as the reconciled bottom forecasts, which are multiplied by the aggregation matrix S to obtain the coherent probabilistic forecasts $\tilde{\boldsymbol{y}}_t$.

Note that we can repeat the above procedure for the h-period-ahead forecasting to obtain a set of coherent forecasts $\{\tilde{y_T}, \tilde{y_{T+1}}, ..., \tilde{y_{T+h}}\}$.

4 Experiments

In this section, we conduct extensive empirical evaluations on four real-world hierarchical datasets from different industrial domains, including three public datasets and one dataset collected from the application servers of our company.

4.1 Datasets and Baselines

The three public datasets include Tourism [4], Tourism-L [56], and Traffic [7]. The new dataset is the service-workload TS data collected from our company's application servers, where our method is deployed for data traffic forecasting. We conduct the performance comparison against state-of-the-art reconciliation algorithms including MinT [56], ERM [2], and Hier-E2E [44], along with other classical baselines, including bottom-up (NaiveBU) approach that generates univariate point forecasts for the bottom-level time series independently followed by the aggregation according to the coherent constraints to obtain point forecasts for the aggregated series. We extensively compare the well-acknowledged and advanced time series forecasting models, which can be divided into three categories as a whole (correspond to the three rows in Table 1), namely classical statistical methods (including ARIMA, ETS [17]), deep point prediction methods (including LSTM [14], TFT [29], N-BEATS [39], N-HiTS [5], Transformer, Reformer [24], LogTrans [28], Informer [61], Autoformer [58], Fedformer [62], TimesNet [57], PatchTST [37], iTransformer [30], TimeMixer [54]),and deep probability prediction methods (including DeepAR [47], DeepVAR [46], Hier-E2E [44]). The details about datasets and implementation are provided in the Appendix A, B.

4.2 Evaluation Metrics

Considering that our method generates probabilistic forecasts instead of point forecasts, it is necessary to evaluate the corresponding probabilistic accuracy with proper metrics, but commonly-used *Mean Absolute Error* (MAE) or *Mean Absolute Percent Error* (MAPE) cannot be directly used for this purpose. On the other hand, *Continuous Probability Ranked Score* (CRPS)[34] generalizes the MAE for probabilistic measurement, making it one of the most widely used accuracy metrics for probabilistic forecasts (including Hier-E2E[44], DeepVAR[46], PROFHIT[19], SHARQ[13], DPMN[38], and TDProb[9], etc.), which is adopted in our evaluation stage, to measure the compatibility of a cumulative distribution function $\hat{F}_{t,i}^{-1}$ for TS i against the ground-truth observation $y_{t,i}$, to estimate the accuracy of our forecast distributions. CRPS can be defined as

$$CRPS(\hat{F}_t, y_t) := \sum_i \int_0^1 QS_q(\hat{F}_{t,i}^{-1}(q), y_{t,i})\, dq\,, \quad (8)$$

where QS_q is the quantile score for the q-th quantiles:

$$QS_q = 2(\mathbb{1}\{y_{t,i} \leq \hat{F}^{-1}_{t,i}(q)\} - q)(\hat{F}^{-1}_{t,i}(q) - y)$$

We use the discrete version in our experiment, with the integral in Equation 8 replaced by the weighted sum over the quantile set, and we use the quantiles ranging from 0.05 to 0.95 in steps of 0.05.

4.3 Experiment Results

Main Results. As illustrated in the table 1, our Flow-based Reconciliation Transformer (FRT) achieves significant performance enhancements over more than **25 advanced baseline models across four datasets, notably achieving an improvement exceeding 40% on the Server-Workload dataset. Furthermore, confidence levels have been calculated, surpassing 95% across all datasets.** This thoroughly underscores the superiority of our approach, particularly when juxtaposed with traditional statistical methods, whereby we realize a leapfrog advancement.

We can also observe that the performance of traditional classical statistical methods is far inferior to the deep learning models based on Transformer or RNN. With the help of neural networks, we not only achieve the best results but also avoid strong assumptions, such as unbiased estimates or Gaussian distribution, of traditional statistical models.

Additional Results. We compared our techniques to those discussed in five recently published papers that do not provide code including PROFHIT[19], SHARQ[13], DPMN[38], TDProb[9], HIRED[42] in Table 2. According to the results on the Tourism-L dataset reported in the paper PROFHIT, SHARQ, DPMN, TDProb, HIRED, and the results of DPMN and TDProb on the Traffic dataset, our method achieves significant improvements compared with the latest methods mentioned above.

4.4 Ablation Study

To further demonstrate the effectiveness of our model, we conduct ablation studies using the variant of our model and the related models. The results of each aggregation level are shown in Figure 5. In **FRT-NaiveBU**, we use the FRT without reconciliation to generate the bottom levels forecasts, then use the above NaiveBU method to ensure coherency, which aggregates according to the hierarchical structure. This method serves as a comparison to prove the efficacy of our proposed flow-based reconciliation.

DeepVAR-lowrank-copula is an RNN-based model using the low-rank Gaussian coupla [46] and **Autoregressive Transformer** is an encoder-decoder transformer-based architecture [20], enabled by its multi-head self-attention mechanism to capture both long- and short-term dependencies in TS data. **Hier-E2E** is the end-to-end model combing DeepVAR (Gaussian distribution) and projection matrix [44].

Transformer-based models outperform RNN-based ones, especially in upper-level TS. The transformer allows access to any part of the historic data regardless of temporal distance and is thus capable of generating better conditioning for NF head, as evidenced by our experiments.

The non-Gaussian data distribution and nonlinear correlations attest to the need for more expressive density estimators, i.e., CNF,

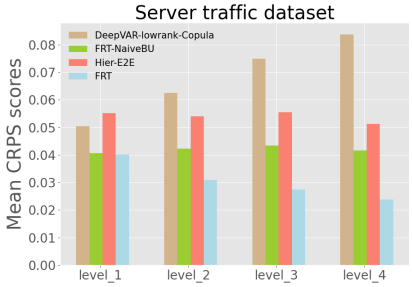

Figure 5: Mean CRPS scores (lower is better) of all aggregation levels in the Tourism, Traffic datasets.

which makes our approach significantly outperform other methods, especially in the bottom-level TS.

In summary, (1) As shown in Figure 5, **we can observe the improvement achieved by using only the Transformer component (*FRT-NavieBU*) for prediction, demonstrating the necessity to adopt Transformer architecture**. (2) Furthermore, we achieve significant improvements in low-level TS in *FRT* with CNF, **which confirms the analysis of the non-Gaussian/nonlinear characteristics of the bottom-level TS in our motivation**.

4.5 Model Efficiency

We analyze model efficiency in terms of both model size and running time. We measured both items for the main methods on the Tourism dataset and presented the results in Table 3. By combining the performance comparison in Table 1 with the efficiency comparison in Table 3, our approach outperforms the other methods while maintaining comparable efficiency. Moving forward, we will continue to explore more model architectures to enhance the model's efficiency. More model efficiency analysis is provided in Appendix C.2.

5 Deployed Application

The method proposed in this paper has been successfully deployed in a world-leading payment technology company for workload forecasting across different services in its data centers, supporting the scheduling and scaling of its cloud computing resources. This corporation depends on an extensive array of application servers to underpin its complex online financial activities, which are distributed across various Internet Data Centers (IDCs) that constitute the computational backbone. Enhancing the efficiency of resource utilization necessitates the preemptive scheduling of application services, which in turn demands precise predictions of workload to inform resource management strategies.

In our company's IDC, the deployment of application services in each IDC follows a hierarchical structure. Specifically, as shown in the Figure 6, each service has four levels in the deployment and workload traffic distribution within the IDC. The first level is the overall workload traffic. The second level involves deployment to **different cities** (e.g., Shanghai, Hangzhou, Shenzhen, etc.). The third deployment level is the zone (our company divides each city's IDC into different virtual data centers, which we refer to as **zones**, allowing for more granular deployment). The final level is the specific deployment **pods** (containers hosting services)) for

Table 1: CRPS values (lower is better) averaged over 5 runs. We conducted the ablation study on FRT-NaiveBU. SOTA methods based on ARIMA and ETS produce consistent results over multiple runs, and thus, we have not presented their standard deviation.

Method	Traffic	Tourism	Tourism-L	Service-Workload
ARIMA-NaiveBU	0.0808	0.1138	0.1741	0.3834
ETS-NaiveBU	0.0665	0.1008	0.1690	0.3899
ARIMA-MinT-shr	0.0770	0.1171	0.1609	0.2713
ARIMA-MinT-ols	0.1116	0.1195	0.1729	0.2588
ETS-MinT-shr	0.0963	0.1013	0.1627	0.3472
ETS-MinT-ols	0.1110	0.1002	0.1668	0.2652
ARIMA-ERM	0.0466	0.5887	0.5635	0.2320
ETS-ERM	0.1027	2.3755	0.5080	0.2501
LSTM-NaiveBU	0.1029 ± 0.0017	0.1251 ± 0.0029	0.2035 ± 0.0022	0.2578 ± 0.0016
TFT-NaiveBU	0.0954 ± 0.0011	0.1093 ± 0.0031	0.1857 ± 0.0017	0.1580 ± 0.0007
N-BEATS-NaiveBU	0.0703 ± 0.0021	0.1062 ± 0.0011	0.1912 ± 0.0019	0.1213 ± 0.0027
N-HITS-NaiveBU	0.0591 ± 0.0023	0.0962 ± 0.0015	0.1579 ± 0.0021	0.1105 ± 0.0015
Transformer-NaiveBU	0.0621 ± 0.0029	0.1102 ± 0.0013	0.1877 ± 0.0031	0.1121 ± 0.0008
Reformer-NaiveBU	0.0633 ± 0.0024	0.1015 ± 0.0015	0.1734 ± 0.0010	0.1138 ± 0.0009
LogTrans-NaiveBU	0.0614 ± 0.0016	0.1073 ± 0.0020	0.1652 ± 0.0013	0.1146 ± 0.0022
Informer-NaiveBU	0.0571 ± 0.0018	0.0911 ± 0.0011	0.1601 ± 0.0012	0.1101 ± 0.0034
Autoformer-NaiveBU	0.0384 ± 0.0029	0.0803 ± 0.0016	0.1342 ± 0.0011	0.0723 ± 0.0028
Fedformer-NaiveBU	0.0352 ± 0.0020	0.0755 ± 0.0017	0.1307 ± 0.0011	0.0671 ± 0.0021
TimesNet-NaiveBU	0.0378 ± 0.0007	0.0702 ± 0.0010	0.1392 ± 0.0007	0.0707 ± 0.0012
PatchTST-NaiveBU	0.0405 ± 0.0019	0.0775 ± 0.0007	0.1271 ± 0.0021	0.0655 ± 0.0020
iTransformer-NaiveBU	0.0342 ± 0.0003	0.0711 ± 0.0019	0.1248 ± 0.0009	0.0559 ± 0.0011
TimeMixer-NaiveBU	0.0337 ± 0.0005	0.0708 ± 0.0001	0.1286 ± 0.0013	0.0541 ± 0.0012
DeepAR-NaiveBU	0.0574 ± 0.0026	0.1023 ± 0.0019	0.1816 ± 0.0088	0.1136 ± 0.0073
DeepVAR-lowrank-Copula	0.0583 ± 0.0071	0.0991 ± 0.0083	0.1781 ± 0.0093	0.1125 ± 0.0041
Hier-E2E	0.0376 ± 0.0060	0.0834 ± 0.0052	0.1520 ± 0.0032	0.0530 ± 0.0012
FRT(***Ours***)	**0.0217 ± 0.0055**	**0.0611 ± 0.0077**	**0.1135 ± 0.0059**	**0.0314 ± 0.0067**
FRT-NaiveBU	0.0377	0.0815	0.1471	0.0417
Confidence	**95%**	**99%**	**99%**	**99%**

Table 2: CRPS values (lower is better) compared to new baselines. The results are cited from the original papers. The '-' symbol indicates that the original paper does not report results on the dataset.

Method	Tourism-L	Traffic
PROFHIT	0.12	-
SHARQ	0.17	-
DPMN	0.1249 ± 0.0020	0.0704 ± 0.0014
TDProb	0.137	0.0575 ± 0.0006
HIRED	0.186	-
FRT(Ours)	**0.1135 ± 0.0059**	**0.0217 ± 0.0055**

Table 3: Model efficiency comparison including model size (parameter) and running time (seconds).

Method	Parameter (MB)	Running Time (s)
Hier-E2E	2.152	0.49
DeepVAR	1.985	0.47
TFT-BU	2.587	0.62
Informer-BU	1.903	0.51
Autoformer-BU	1.848	0.48
Fedformer-BU	2.901	0.98
FRT(Ours)	1.910	0.45

each service. When the workload traffic for each application service arrives, it is distributed through the load balancing system

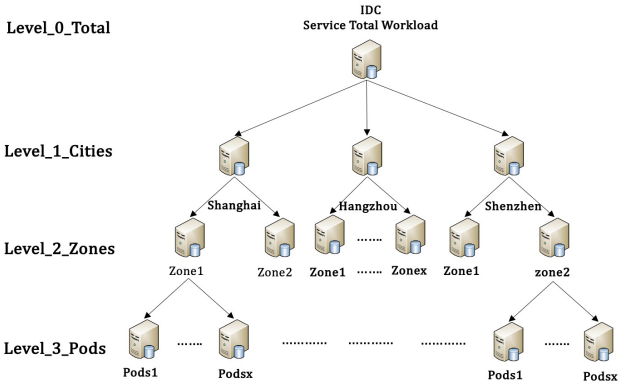

Figure 6: A deployment example of our company's different service in IDC.

to different cities, then to different zones within those cities, and finally reaches each deployment unit. This hierarchical deployment allows for mutual disaster recovery between different cities and zones, enabling a response to unexpected situations.

Additionally, the Figure 7 presented herein illustrates an analysis of real service-workload data obtained from our company's IDC. We employed workload TS data from three distinct service categories—namely, web services, database services, and AI inference services and utilized Seaborn to visualize the relationships within varying levels of the service-workload dataset.

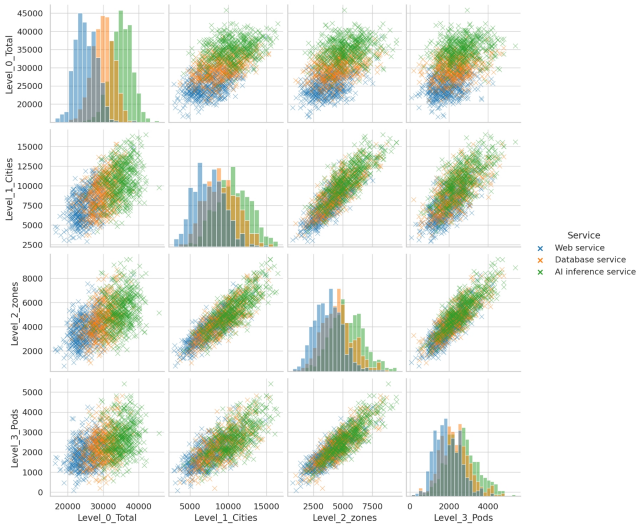

Figure 7: The diagonal plots are each univariate time series' distributions (histograms). Plots outside of the diagonal are scatterplots analyzing the correlation between the two time series. The different colors represent the different types of services.

The above figure demonstrates not only the non-Gaussian distribution of data but also the nonlinear relationship between data at different levels.

As shown in Table 1, our approach demonstrates a remarkable improvement over the current SOTA techniques in HTS forecasting. **Our method achieves a relative improvement of up to 40% compared to the baseline models in online deployment**. Figure 8 presents an example of using FRT for workload prediction in an IDC to achieve cloud computing resource scaling. By accurately predicting workloads across different services within the hierarchical structure of the IDC, we achieved precise resource scaling, enabling efficient configuration of cloud computing resources and significantly increasing their utilization rate. After activating our FRT prediction technology, **the number of pods (containers hosting microservices) used by the microservices significantly decreased from over 1000 to below 300 (a reduction in resource consumption of over 70%), greatly enhancing resource efficiency**. More details of deployment are provided in Appendix D.

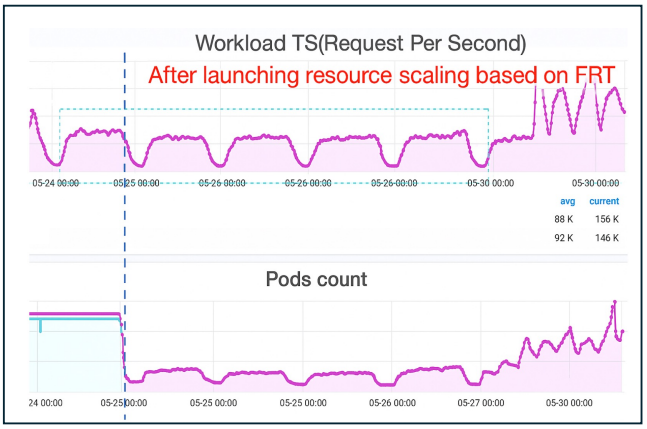

Figure 8: Blue vertical dotted line indicates the launch of resource scaling based on workload forecasting. Green frame highlights the comparison of the metrics under the same workload before and after enabling our method.

6 Conclusion

In this paper, we propose a novel end-to-end approach (FRT) that conducts forecasting and reconciliation simultaneously for HTS. FRT generates coherent probabilistic forecasts without explicit post-processing steps, by combining *autoregressive transformer* and *conditioned normalizing flow*. We conducted extensive evaluations on real-world datasets, demonstrating the competitiveness of our method under various conditions against other state-of-the-art methods. We have also successfully deployed FRT in real-world applications as a continuous and robust forecasting service for predicting server traffic, thereby promoting energy efficiency. Note that the FRT has been extensively deployed within our company, acting as the pivotal model for workload forecasting across IDC, thereby bolstering the predictive service capabilities of the entire data center infrastructure. Owing to the precision of our model's predictive capabilities, we have facilitated the efficient allocation of data center resources, culminating in substantial resource conservation.

References

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

[2] Souhaib Ben Taieb and Bonsoo Koo. 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.

[3] George EP Box and Gwilym M Jenkins. 1968. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 17, 2 (1968), 91–109.

[4] Robyn Bushell, Gary M Prosser, Herbert William Faulkner, and Jafar Jafari. 2001. Tourism research in Australia. *Journal of travel research* 39, 3 (2001), 323–326.

[5] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza, Max Mergenthaler-Canseco, and Artur Dubrawski. 2022. N-hits: Neural hierarchical interpolation for time series forecasting. *arXiv preprint arXiv:2201.12886* (2022).

[6] Zhixuan Chu, Hui Ding, Guang Zeng, Shiyu Wang, and Yiming Li. 2024. Causal Interventional Prediction System for Robust and Explainable Effect Forecasting. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 4431–4438.

[7] Marco Cuturi. 2011. Fast Global Alignment Kernels.. In *ICML*. 929–936.

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

[9] Abhimanyu Das, Weihao Kong, Biswajit Paria, and Rajat Sen. 2022. A Top-Down Approach to Hierarchically Coherent Probabilistic Forecasting. *arXiv preprint arXiv:2204.10414* (2022).

[10] Fan Deng, Jie Lu, Shi-Yu Wang, Jie Pan, and Li-Yong Zhang. 2019. A distributed PDP model based on spectral clustering for improving evaluation performance. *World Wide Web* 22 (2019), 1555–1576.

[11] Fan Deng, Shiyu Wang, Liyong Zhang, Xiaoqian Wei, and Jingping Yu. 2018. Establishment of attribute bitmaps for efficient XACML policy evaluation. *Knowledge-Based Systems* 143 (2018), 93–101.

[12] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016).

[13] Xing Han, Sambarta Dasgupta, and Joydeep Ghosh. 2021. Simultaneously Reconciled Quantile Forecasting of Hierarchically Related Time Series. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 190–198.

[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

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

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

[17] Rob J Hyndman and Shu Fan. 2009. Density forecasting for long-term peak electricity demand. *IEEE Transactions on Power Systems* 25, 2 (2009), 1142–1153.

[18] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. 2024. Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations (ICLR)*.

[19] Harshavardhan Kamarthi, Lingkai Kong, Alexander Rodríguez, Chao Zhang, and B Aditya Prakash. 2022. PROFHIT: Probabilistic Robust Forecasting for Hierarchical Time-series. *arXiv preprint arXiv:2206.07940* (2022).

[20] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*. PMLR, 5156–5165.

[21] Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA.

[22] Diederik P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039* (2018).

[23] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[24] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451* (2020).

[25] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. 2020. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[26] Yaxuan Kong, Yiyuan Yang, Shiyu Wang, Chenghao Liu, Yuxuan Liang, Ming Jin, Stefan Zohren, Dan Pei, Yan Liu, and Qingsong Wen. 2025. Position: Empowering Time Series Reasoning with Multimodal LLMs. *arXiv preprint arXiv:2502.01477* (2025).

[27] Iryna Korshunova, Yarin Gal, Joni Dambre, and Arthur Gretton. 2018. Conditional bruno: A deep recurrent process for exchangeable labelled data. In *Bayesian Deep Learning NeurIPS Workshop*. 72371B.

[28] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems* 32 (2019).

[29] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. 2021. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting* 37, 4 (2021), 1748–1764.

[30] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2023. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. *arXiv preprint arXiv:2310.06625* (2023).

[31] Zitao Liu, Yan Yan, and Milos Hauskrecht. 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.

[32] Yang Luo, Shiyu Wang, Zhemeng Yu, Wei Lu, Xiaofeng Gao, Lintao Ma, and Guihai Chen. 2024. Adaptive two-stage cloud resource scaling via hierarchical multi-indicator forecasting and bayesian decision-making. *arXiv preprint arXiv:2408.01000* (2024).

[33] Mohammad Masdari and Afsane Khoshnevis. 2020. A survey and classification of the workload forecasting methods in cloud computing. *Cluster Computing* 23, 4 (2020), 2399–2424.

[34] James E Matheson and Robert L Winkler. 1976. Scoring rules for continuous probability distributions. *Management science* 22, 10 (1976), 1087–1096.

[35] Janmenjoy Nayak, Bighnaraj Naik, AK Jena, Rabindra K Barik, and Himansu Das. 2018. Nature inspired optimizations in cloud computing: applications and challenges. *Cloud computing for optimization: Foundations, applications, and challenges* (2018), 1–26.

[36] Juntong Ni, Zewen Liu, Shiyu Wang, Ming Jin, and Wei Jin. 2025. TimeDistill: Efficient Long-Term Time Series Forecasting with MLP via Cross-Architecture Distillation. arXiv:2502.15016 [cs.LG] https://arxiv.org/abs/2502.15016

[37] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations*.

[38] Kin G Olivares, O Nganba Meetei, Ruijun Ma, Rohan Reddy, Mengfei Cao, and Lee Dicker. 2021. Probabilistic hierarchical forecasting with deep poisson mixtures. *arXiv preprint arXiv:2110.13179* (2021).

[39] Boris N Oreshkin, Grzegorz Dudek, Paweł Pełka, and Ekaterina Turkina. 2021. N-BEATS neural network for mid-term electricity load forecasting. *Applied Energy* 293 (2021), 116918.

[40] Anastasios Panagiotelis, Puwasala Gamakumara, George Athanasopoulos, Rob J Hyndman, et al. 2020. Probabilistic forecast reconciliation: Properties, evaluation and score optimisation. *Monash econometrics and business statistics working paper series* 26 (2020), 20.

[41] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. 2019. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762* (2019).

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

[43] Haoran Qiu, Subho S Banerjee, Saurabh Jha, Zbigniew T Kalbarczyk, and Ravishankar K Iyer. 2020. {FIRM}: An Intelligent Fine-grained Resource Management Framework for {SLO-Oriented} Microservices. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 805–825.

[44] Syama Sundar Rangapuram, Lucien D Werner, Konstantinos Benidis, Pedro Mercado, Jan Gasthaus, and Tim Januschowski. 2021. End-to-End Learning of Coherent Probabilistic Forecasts for Hierarchical Time Series. In *International Conference on Machine Learning*. PMLR, 8832–8843.

[45] Krzysztof Rzadca, Pawel Findeisen, Jacek Swiderski, Przemyslaw Zych, Przemyslaw Broniek, Jarek Kusmierek, Pawel Nowak, Beata Strack, Piotr Witusowski, Steven Hand, et al. 2020. Autopilot: workload autoscaling at Google. In *Proceedings of the Fifteenth European Conference on Computer Systems*. 1–16.

[46] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. 2019. High-dimensional multivariate forecasting with low-rank gaussian copula processes. *arXiv preprint arXiv:1910.03002* (2019).

[47] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.

[48] Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. 2024. Time-MoE: Billion-Scale Time Series Foundation Models with Mixture of Experts. arXiv:2409.16040 https://arxiv.org/abs/2409.16040

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

[50] Souhaib Ben Taieb, James W Taylor, and Rob J Hyndman. 2021. Hierarchical probabilistic forecasting of electricity demand with smart meter data. *J. Amer. Statist. Assoc.* 116, 533 (2021), 27–43.

[51] Shiyu Wang. 2024. Neuralreconciler for hierarchical time series forecasting. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 731–739.

[52] Shiyu Wang, Yinbo Sun, Xiaoming Shi, Shiyi Zhu, Lin-Tao Ma, James Zhang, Yifei Zheng, and Jian Liu. 2023. Full scaling automation for sustainable development

of green data centers. *arXiv preprint arXiv:2305.00706* (2023).
[53] Shiyu Wang, Yinbo Sun, Yan Wang, Fan Zhou, Lin-Tao Ma, James Zhang, and YangFei Zheng. 2023. Flow-Based End-to-End Model for Hierarchical Time Series Forecasting via Trainable Attentive-Reconciliation. In *International Conference on Database Systems for Advanced Applications*. Springer, 167–176.
[54] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and JUN ZHOU. 2024. TimeMixer: Decomposable Multiscale Mixing for Time Series Forecasting. In *International Conference on Learning Representations (ICLR)*.
[55] Shiyu Wang, Fan Zhou, Yinbo Sun, Lintao Ma, James Zhang, and Yangfei Zheng. 2022. End-to-end modeling of hierarchical time series using autoregressive transformer and conditional normalizing flow-based reconciliation. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 1087–1094.
[56] Shanika L Wickramasuriya, George Athanasopoulos, and Rob J Hyndman. 2019. Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. *J. Amer. Statist. Assoc.* 114, 526 (2019), 804–819.
[57] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2022. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186* (2022).
[58] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems* 34 (2021), 22419–22430.
[59] Jianwei Yin, Xingjian Lu, Hanwei Chen, Xinkui Zhao, and Neal N Xiong. 2014. System resource utilization analysis and prediction for cloud based applications under bursty workloads. *Information Sciences* 279 (2014), 338–357.
[60] Fan Zhou, Chen Pan, Lintao Ma, Yu Liu, Shiyu Wang, James Zhang, Xinxin Zhu, Xuanwei Hu, Yunhua Hu, Yangfei Zheng, et al. 2023. Sloth: Structured learning and task-based optimization for time series forecasting on hierarchies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11417–11425.
[61] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of AAAI*.
[62] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*. PMLR, 27268–27286.

A Datasets Details

Three real-world hierarchical datasets and a dataset of the workload TS data from application services of our company are listed below for our experiments:

- **Traffic** [7] provides the occupancy rate (between 0 and 1) of 963 car lanes of San Francisco bay area freeways. We aggregate sub-hourly data to obtain daily observations for one year and generate a 207-series hierarchy using the same aggregation strategy as in [2], which is divided into 4 levels. Bottom-level contains 200 series, aggregated-levels contain 7 series in total, and the prediction length is 1.
- **Tourism** [1, 4] 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, and the prediction length is 8. This dataset is frequently referenced in hierarchical forecasting studies [16, 50].
- **Tourism-L** [56] is a larger, more detailed version of Tourism, which contains 555 total series in a grouped structure and 228 observations; This dataset has two hierarchies, i.e., based on geography and based on purpose-of-travel, respectively, sharing a common root, which is divided into 4 or 5 levels. Bottom-level contains 76 or 304 series, aggregated-level contains 175 series, and the prediction length is 12.
- **Service-Workload**: The workload TS dataset is the request per second (RPS) from our company's online microservices, which is a large-scale dataset containing 1,589 time series. Each point is aggregated and sampled every 10 minutes. The dataset is compiled from the past three years of historical data.

B Implementation Details

B.1 Module Architecture

We adopt multi-head attention as the encoder and masked multi-head attention as decoder in encoder-decoder transformer architecture, where uses the H = 8 heads and n = 3 encoding and m = 3 decoding layers and a dropout rate of 0.1. In conditioned NF modules, we employ Real-NVP with 3 layers of flow modules to map data sampled from Gaussian distribution to target distribution. In the sampling stage, we generate the 500 Monte Carlo samples from the predicted distribution.

B.2 Hyper-parameter Setting

We conduct experiments on four real-world datasets, where the train/test set by 0.8 at the timeline, where the preceding segments are used for training and the following ones are for testing. We also split 10% samples from the train set as the validation set to avoid overfitting. Batch size of 128 is used for training for 100 epochs in total, starting with a learning rate of 0.001 and reduced by a factor of 10 at every 20 iterations. The complete code of our approach will be released upon the internal approval.

B.3 Experiment Environment

All experiments were conducted on a Linux server operating with the Ubuntu 16.04 distribution. The server was equipped with an Intel(R) Xeon(R) Silver 4214 processor, which boasts a base clock speed of 2.20GHz. Additionally, the system was configured with 64GB of RAM, providing ample memory resources to facilitate the processing requirements of the experiments. For AI inference tasks, the server was outfitted with 8 Nvidia A-100 GPU, which is renowned for its advanced capabilities in handling intensive computational workloads. This hardware configuration was carefully selected to ensure optimal performance and efficiency during the execution of experimental protocols.

C More Analysis

C.1 Hyperparameter sensitivity

We conducted a sensitivity analysis on three key sets of hyperparameters—**encoder layers**, **decoder layers**, and **Real-NVP layers**—using the service-workload dataset. The results of this analysis are summarized in Table 4.

From the table, it is evident that the best performance, as measured by the Continuous Ranked Probability Score (CRPS), is achieved when each of the three sets of hyperparameters is set to 3. Specifically, the CRPS values for encoder layers, decoder layers, and Real-NVP layers are 0.0321, 0.0329, and 0.0311, respectively, at this configuration. Further increases in the number of layers do not significantly enhance performance and instead result in additional computational overhead.

Therefore, based on these results, we selected this set of hyperparameters (3 encoder layers, 3 decoder layers, and 3 Real-NVP layers) as the optimal configuration for our workload forecasting model.

Table 4: Hyperparameter Sensitivity Analysis Results

Encoder Layers	CRPS	Decoder Layers	CRPS	Real-NVP Layers	CRPS
1	0.0351	1	0.0412	1	0.0390
2	0.0347	2	0.0344	2	0.0332
3	**0.0321**	3	**0.0329**	3	**0.0311**
4	0.0337	4	0.0329	4	0.0318

Table 5: Efficiency Analysis of Different Models

Method	GPU Memory (MB)	Running Time (s)	Parameter (MB)	Performance (CRPS)
Hier-E2E	1447	2.34	121	0.0530
TFT-BU	2047	1.89	141	0.1580
Informer-BU	2982	2.42	192	0.1101
Autoformer-BU	2871	1.92	174	0.0723
FRT (Ours)	**1426**	**1.69**	**112**	**0.0314**

C.2 More Efficiency Analysis

To evaluate the effectiveness and efficiency of our proposed method, we conducted a comparison in our actual production environment. The comparison included our FRT model, Google's TFT, Amazon's Hier-E2E, and transformer-based models such as Informer and Autoformer. These tests were performed on traffic time series forecasting for over 1,589 sets of web services.

As shown in Table 5, our method (FRT) demonstrates superior performance in practical application deployments while maintaining commendable efficiency. Specifically, FRT achieves the best performance in terms of CRPS (0.0314), while also requiring the least GPU memory (1,426 MB), the shortest running time (1.69 seconds), and fewer parameters (112 MB). This combination of high effectiveness and efficiency makes FRT particularly advantageous for large-scale industrial applications.

D Deployment Details

Our company's cloud computing cluster management system is built upon the proprietary **Sigma framework**, which functions similarly to Kubernetes by providing unified container management. This system is designed to handle the complexities of managing and scheduling resources across a large-scale cloud environment. As an online resource scheduling platform, it integrates our proposed workload forecasting method to predict workload fluctuations for individual microservices. These predictions enable the system to allocate resources optimally, ensuring both high efficiency and cost-effectiveness.

To meet the high demands of forecasting workloads for a vast array of microservices, our solution is deployed across a robust infrastructure consisting of **100 Linux servers**, each running **Ubuntu 16.04**. These servers are powered by **Intel(R) Xeon(R) Silver 4214 2.20GHz CPUs** and equipped with **64GB of memory**. This configuration is capable of supporting over **3000 QPS (queries per second)** of prediction request traffic during peak periods, such as the Double Eleven shopping festival. This demonstrates the scalability and performance of our system under extreme traffic conditions.

Each of our online prediction servers is further equipped with an **NVIDIA A-10 GPU** to accelerate inference services. These GPUs ensure that prediction requests are processed with low latency, meeting the stringent requirements of real-time resource scheduling. In addition, our system is designed to handle the dynamic nature of workload patterns, which necessitates regular updates to the forecasting models. To address this, we schedule **weekly model update training** sessions using **NVIDIA A-100 80GB GPUs**, which are specifically chosen for their high performance in large-scale machine learning tasks. These updates are crucial for maintaining the accuracy and reliability of the forecasting models over time.

Over the past three years, our proposed workload forecasting method has undergone continuous iteration and optimization. It has been seamlessly integrated into our company's resource scheduling system and has demonstrated exceptional stability in production environments. The system has successfully passed multiple stress tests, including several Double Eleven promotions, which are known for their extreme traffic spikes and high demands on system performance. These real-world tests have effectively validated the robustness, scalability, and reliability of our solution.

Furthermore, the integration of our forecasting method into the Sigma framework has not only improved resource utilization but also reduced operational costs by minimizing over-provisioning and under-provisioning of resources. This has had a significant impact on the overall efficiency of our cloud computing infrastructure, enabling the system to adapt dynamically to changing workloads while maintaining service-level agreements (SLAs).

In summary, our deployment strategy combines cutting-edge hardware, efficient resource management frameworks, and a scalable forecasting solution to deliver a highly reliable and performant system. This ensures that our cloud infrastructure remains robust and efficient, even during peak operational periods, and highlights the practical value of our proposed method in real-world applications.