Coherent Probabilistic Aggregate Queries on Long-horizon Forecasts

Prathamesh Deshpande[*] , **Sunita Sarawagi**

Department of Computer Science and Engineering, IIT Bombay

pratham@cse.iitb.ac.in, sunita@iitb.ac.in

Abstract

Long range forecasts are the starting point of many decision support systems that need to draw inference from high-level aggregate patterns on forecasted values. State of the art time-series forecasting methods are either subject to concept drift on long-horizon forecasts, or fail to accurately predict coherent and accurate high-level aggregates.

In this work, we present a novel probabilistic forecasting method that produces forecasts that are coherent in terms of base level and predicted aggregate statistics. We achieve the coherency between predicted base-level and aggregate statistics using a novel inference method based on KL-divergence that can be solved efficiently in closed form. We show that our method improves forecast performance across both base level and unseen aggregates post inference on real datasets ranging three diverse domains. (Project URL)

1 Introduction

Long-term forecasting is critical for decision support in several application domains including finance, governance, and climate studies. Interactive decision making tools need accurate probability distributions not just over the long-term base forecasts but also on dynamically chosen aggregates of the forecasts. An analyst may be interested in a variety of aggregates such as average, sum, or trend of values for various windows of granularity. For example, the base-level forecasts may be retail price of a commodity at a daily-level for the next two quarters, but the end user may also inspect average at weekly or monthly granularity, or change in price (inflation) from week to week. For the aggregates too, the user is interested in the associated uncertainty represented as a distribution that is coherent with the base distribution.

A time-series in our model is characterized by a sequence $(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_n, y_n)$ where $\mathbf{x}_t \in R^d$ denotes a vector of input features and y_t is a real-value at a time t. We are given a prefix or history from such a series up to a present time T which we denote as H_T. Our goal is to predict future values $y_{T+1} \ldots y_{T+R}$ given input features $\mathbf{x} = \mathbf{x}_{T+1} \ldots \mathbf{x}_{T+R}$ and

the history H_T. Unlike in conventional time-series forecasting where the interest is short-term (say R between 1 and 10), in long-term forecasting we focus on large values of R (say between 100 and 1000). We wish to capture the uncertainty of the prediction by outputting a distribution over the predicted values. We use $\hat{P}(y_{T+1} \ldots y_{T+R}|H_T, \mathbf{x})$ to denote the predicted distribution for future time horizon given history H_T.

Our goal is not just to provide accurate probabilistic forecasts for the the base-level values but also for aggregates of these values. For example, the base-level forecasts may be at the daily-level and we may wish to view monthly sales. In theory, we could marginalize the base-level forecast distribution $\hat{P}(y_{T+1} \ldots y_{T+R}|H_T, \mathbf{x})$ to compute coherent distribution of any dynamic aggregate. In practice, we need to ensure that the marginalization can be efficiently computed. Furthermore, the accuracy of the predicted aggregate should be no worse than the accuracy of an independently trained model to directly predict the distribution of the aggregate. For example, a practitioner could train separate models for predicting monthly sales and daily sales for greater accuracy at the monthly-level. However, training separate models for each aggregate has two limitations: (1) independently predicted aggregate distributions may not be coherent with each other or with the base-level forecasts, and (2) the set of aggregate functions may be decided dynamically during data analysis where training a separate model is impractical. Thus, we seek solutions that explicitly marginalize a fixed distribution to ensure coherence of distributions of aggregate quantities.

Existing forecast models can be categorized as either auto-regressive (AR) [Flunkert et al., 2017; Mukherjee et al., 2018] or non-auto regressive (NAR). We will show in this paper that both these models are inadequate for our goal of efficiently and dynamically defining the distribution of any aggregate quantity. In AR models the output distribution captures the full joint dependency among all predicted values by defining that the value at t depends on all values before it. During training we condition on true previous values but during inference we need to perform forward sampling where each predicted value is conditioned on sampled prior values. In long-term forecasting, such forward sampling suffers from drift caused by cascading errors [Deshpande et al., 2021]. A second major limitation of the AR model is that the joint distribution over the forecast variables is expressed via a black-box neural network. Marginalizing the distribution to obtain

---

[*]Contact author

distribution over dynamically chosen aggregates will involve repeated sampling steps, which may incur huge latency during interactive analysis.

This has led to greater interest in the non-auto-regressive (NAR) models which independently predict each variable in the forecast horizon conditioned on the history. NAR models have been shown to work better in practice both for medium-term and long-term forecasting [Wen *et al.*, 2017; Deshpande and Sarawagi, 2019; Zhou *et al.*, 2020]. Another major benefit of the NAR models is that during inference all values can be predicted in parallel, unlike in the AR method that entails sequential sampling. However, a limitation of the non-auto-regressive methods is that the dependency among the output variables is not modeled. The distribution over aggregates computed from independent base predictions may be inaccurate. For example, if an analyst marginalizes the data to inspect the distribution of the difference between adjacent values, the variance from the independent model might be over-estimated. Further, in the raw form a time-series often contains noise that hinders a forecasting model from capturing top-level patterns in the series over long prediction horizons. In applications that involve analysis over aggregates of forecast values, these might give rise to non-smooth or inconsistent aggregates.

In this paper we propose a novel method of preserving higher-level patterns in long-range forecasts of non-auto-regressive forecast models while allowing accurate and efficient computation of the distribution of dynamically chosen aggregates. Our key idea is to train the model to independently output various high-order statistics over the forecast values, along with the base-level forecasts. Aggregated values provide a noise-removed signal of how the series evolves in time. We use such signals to guide inference to capture top-level trends in the data. We formulate that as a task of obtaining a revised consensus distribution that minimizes the KL-distance with each predicted aggregate and base forecasts. The consensus distribution is represented as a Gaussian distribution that can be easily marginalized. The parameters of the consensus can be obtained via an efficient numerical optimization problem. Unlike existing methods [Han *et al.*, 2021], our consensus distribution not only provides coherence and accuracy for aggregate quantities, but also improves base-level forecasts.

1.1 Contributions

- We propose a probabilistic forecasting method that is more accurate than state of the art for long-term forecasting.

- Our method outputs an easy to marginalize joint distribution for computing accurate and coherent aggregate distributions over dynamically defined functions on forecast values.

- We propose an efficient inference algorithm for computing the consensus distribution.

- Empirical evaluation on benchmark datasets show that our method improves both base-level and aggregate forecasts compared to several existing baselines including a recent hierarchical consensus model.

2 Our Model

Our method seeks to retain the advantage of parallel training and inference that modern transformer-based NAR models offer while providing accurate distributions of forecasts at the base-level and dynamically chosen aggregates over them. We train multiple NAR models for forecasting different types of aggregate functions at different levels of granularity. During inference we obtain a consensus distribution, as a joint Gaussian distribution that can be marginalized in closed form for computing distributions over linear aggregates defined dynamically during analysis. The consensus distribution is obtained by minimizing the KL distance with the various base and aggregate-level forecasts by an efficient inference algorithm. We present the details of each of these steps next.

2.1 Aggregate Functions

We choose a set $\mathcal{A}$ of aggregate functions depending on likely use in downstream analysis. This set does not need to include all possible aggregate functions to be used during analysis, since our method generalizes well to new aggregates even when trained with two aggregate functions. Each aggregate function A_i is characterized with a window size K_i and a fixed real vector $\mathbf{a}^i \in \mathbf{R}^{K_i}$ that denotes the weight of aggregated values. For each A_i, we create an aggregated series by aggregating on disjoint windows of size K_i in the original series. The j-th value in the ith aggregated series is the aggregation of the values $y_{(j-1)K_i+1} \cdots y_{jK_i}$ and denoted z_j^i. We will use the notation $w_{i,j}$ to refer to the window of indices $[(j-1)K_i, \ldots, jK_i]$ that the j-th value of the ith function aggregates. We denote i-th aggregate series as $z_1^i, \ldots, z_{T_i}^i$ where $T_i = \frac{T}{K_i}$ and calculate as follows:

$$z_j^i = A_i(\mathbf{y}_{w_{i,j}}) = \sum_{r=1}^{K_i} a_r^i y_{r+(j-1)K_i} \qquad (1)$$

Here are examples of two aggregations – (i) average and (ii) trend. Average is just:

$$z_j^i = \sum_{r=1}^{K_i} \underbrace{\frac{1}{K_i}}_{a_r^i} y_{(j-1)K_i+r} \qquad (2)$$

Trend aggregate of the window captures how the series changes along time. A positive (negative) trend denotes that the value increases (decreases) as we move forward in the window. We compute trend as the slope of a linear fit on the values in each window as

$$z_j^i = \sum_{r=1}^{K_i} \underbrace{\left(\frac{r}{K_i} - \frac{K_i+1}{2K_i}\right)}_{a_r^i} \cdot y_{(j-1)K_i+r} \qquad (3)$$

We show examples of a base-level time series and one computed aggregate (average) in Figure 1. Our framework can work with any aggregate like the above two that can be expressed as a linear weighted sum of its arguments. For notational ease we refer to the base-level also as an aggregate function A_0 with window size $K_i = 1$ and $\mathbf{a}^i = [1]$.

2.2 Forecast Method

We train independent probabilistic forecasting models for the series corresponding to each aggregate $A_i \in \mathcal{A}$ which includes the original time-series as well. Our work is agnostic of the exact neural architecture used for the forecasts. We present details of how to train these models in Section 2.3.

During inference we use the known history H_T and the trained models to get forecasts $\hat{P}(z_j^i | H_T, \mathbf{x}_j)$ for each of the variables $z_1^i, \dots, z_{T_i}^i$ for each aggregate function A_i, which includes the base-level quantities.

Since models for all aggregates and for the original series are trained independently, these predicted distributions are not necessarily coherent. We infer a new consensus distribution $Q(y_{T+1} \dots y_{T+R})$ that minimizes the KL-distance with each of the forecast distributions. We choose a tractable form for Q: a multivariate Gaussian with mean $\boldsymbol{\mu} = [\mu_{T+1} \dots \mu_{T+R}]^T$ and covariance Σ. With this form we can compute the marginal distribution of aggregate variable z_j^i as:

$$Q_j^i(z_j^i | \boldsymbol{\mu}, \Sigma) = \mathcal{N}\left(z_j^i; \boldsymbol{\mu}_{w_{i,j}}^T \mathbf{a}^i, \ \mathbf{a}^{i\,T} \Sigma_{w_{i,j}} \mathbf{a}^i \right) \quad (4)$$

We use the notation $\Sigma_{w_{i,j}}$ to denote the sub-matrix of Σ that spans over the indices in $w_{i,j}$. Likewise for $\boldsymbol{\mu}_{w_{i,j}}$.

Using these we can write our objective of finding the parameters $\boldsymbol{\mu}$ and Σ so as to minimize the KL-distance with each forecast distribution as:

$$\min_{\boldsymbol{\mu}, \Sigma} \sum_{i \in \mathcal{A}} \sum_{j=T_i}^{T_i+R_i} \alpha_i D_{\mathrm{KL}}\left(Q_j^i(z_j^i | \boldsymbol{\mu}, \Sigma) \,\big\|\, \hat{P}(z_j^i | \bullet) \right) \quad (5)$$

Here, the weights α_i are hyper-parameters which denote importance of KL-distance of i-th aggregate.

When the distribution of each forecast variable $\hat{P}(z_j^i | \bullet)$ is represented as a Gaussian with mean $\hat{\mu}(z_j^i)$ and variance $\hat{\sigma}(z_j^i)$, their KL distance can be expressed in closed form[1]. Also, the optimization over the $\boldsymbol{\mu}$ and Σ terms separate out into two independent optimization problems. For $\boldsymbol{\mu}$ the objective is now:

$$\min_{\boldsymbol{\mu}} \sum_{i \in \mathcal{A}} \sum_{j=T_i}^{T_i+R_i} \frac{1}{\hat{\sigma}(z_j^i)^2} (\boldsymbol{\mu}_{w_{i,j}}^T \mathbf{a}^i - \hat{\mu}(z_j^i))^2 \quad (6)$$

This is a convex quadratic objective that can be solved in closed form as

$$\boldsymbol{\mu}^* = \Big[\sum_{i \in \mathcal{A}} \sum_{j=T_i}^{T_i+R_i} \frac{\tilde{\mathbf{a}}_j^i \tilde{\mathbf{a}}_j^{i\,T}}{\hat{\sigma}(z_j^i)^2} \Big]^{-1} \Big[\sum_{i \in \mathcal{A}} \sum_{j=T_i}^{T_i+R_i} \tilde{\mathbf{a}}_j^{i\,T} \hat{\boldsymbol{\mu}} \Big] \quad (7)$$

where we use $\tilde{\mathbf{a}}_j^i \in \mathbf{R}^R$ to denote the padded version of the indices $w_{i,j}$.

For Σ the objective is:

$$\min_{\Sigma} \sum_{i \in \mathcal{A}} \sum_{j=T_i}^{T_i+R_i} \frac{\mathbf{a}^{i\,T} \Sigma_{w_{i,j}} \mathbf{a}^i}{2\hat{\sigma}(z_j^i)^2} - \log(\mathbf{a}^{i\,T} \Sigma_{w_{i,j}} \mathbf{a}^i) \quad (8)$$

---

[1] $D_{\mathrm{KL}}\left(\mathcal{N}(\mu_q, \sigma_q^2) \,\big\|\, \mathcal{N}(\mu_p, \sigma_p^2)\right) = \frac{(\mu_q - \mu_p)^2 + \sigma_q^2}{2\sigma_p^2} - \log \frac{\sigma_q}{\sigma_p} - \frac{1}{2}$

Algorithm 1 KLST Training and Inference Algorithm

1: **Input:** Training data $\{(\mathbf{x}_1, y_1) \dots (\mathbf{x}_n, \mathbf{y}_n)\}$, aggregate functions $\mathcal{A} = \{(\mathbf{a}_i, K_i)\}$, forecast horizon R.
2: **for** A_i in $\mathcal{A}$ **do**
3:   /*Train i-th aggregate model using training data for i-th aggregate */
4:   $\mathcal{D}_{\mathrm{trn}}^i = z_1^i, \dots, z_{n/K_i}^i$.
5:   $\theta^i \leftarrow \mathrm{TRAIN}(\mathcal{D}_{\mathrm{trn}}^i)$ using Eqn 11
6:   Get forecasts $\hat{P}(z_j^i | \theta^i)$ for $j = T_i + 1, \dots, T_i + R_i$.
7: **end for**
8: Get coherent mean forecasts $\boldsymbol{\mu}^*$ using Eqn. 7.
9: Get coherent variance forecasts (V^*, σ^*) using Eqn. 9.
10: /* End of Training */
11: Inference: Given new aggregate A_{new} defined via $\tilde{\mathbf{a}}^{\mathrm{new}}$
12: $\boldsymbol{\mu}^{*\mathrm{new}} = \boldsymbol{\mu}^{*T} \tilde{\mathbf{a}}^{\mathrm{new}}$.
13: $\Sigma^{*\mathrm{new}} = \sum_{j=1}^{R} (\tilde{a}_j^{\mathrm{new}})^2 \sigma_j^{*2} + (\tilde{\mathbf{a}}^{\mathrm{new}})^T V^* V^{*T} (\tilde{\mathbf{a}}^{\mathrm{new}})$

This objective cannot be solved in closed form but we follow a strategy to efficiently approximate it. We consider a simplification of Σ to be a low-rank matrix. Following [Salinas *et al.*, 2019] we express the covariance Σ in a tractable format as follows:

$$\boldsymbol{\Sigma} = \mathrm{Diag}(\sigma^2) + VV^T$$

$$= \begin{pmatrix} \sigma_{T+1}^2 & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & \sigma_{T+R}^2 \end{pmatrix} + \begin{pmatrix} v_{T+1} \\ \vdots \\ v_{T+R} \end{pmatrix} \begin{pmatrix} v_{T+1} \\ \vdots \\ v_{T+R} \end{pmatrix}^T$$

where $v_{T+r} \in \mathbf{R}^k$ is r-th row of V, and k is the chosen rank of the covariance matrix.

Using the above form of Σ, Equation 8 can be rewritten as:

$$\min_{V, \sigma} \sum_{i \in \mathcal{A}} \sum_{j=T_i}^{T_i+R_i} \frac{\tilde{\mathbf{a}}_j^{i\,T}[\mathrm{Diag}(\sigma^2)]\tilde{\mathbf{a}}_j^i + (\tilde{\mathbf{a}}_j^{i\,T} V)^2}{2\hat{\sigma}(z_j^i)^2} \quad (9)$$

$$- \log\left(\tilde{\mathbf{a}}_j^{i\,T}[\mathrm{Diag}(\sigma^2)]\tilde{\mathbf{a}}_j^i + (\tilde{\mathbf{a}}_j^{i\,T} V)^2 \right) \quad (10)$$

With the above representation of the joint distribution $Q(y_T, \dots y_{T+R})$, the number of parameters needed to store the distribution is linear in R, and also we can compute the distribution of any linear aggregate efficiently. Algorithm 1 presents an overview of our method.

2.3 Training Parameters of Forecast Models

As mentioned earlier we train separate models for each aggregate series. For each aggregate A_i the forecast model can be viewed as a function $F(H_T, \mathbf{x}_R | \theta^i)$ of the history of known values $H_T = (\mathbf{x}_1, y_1) \dots, (\mathbf{x}_T, y_T)$, and known input features $\mathbf{x}_{T+1} \dots \mathbf{x}_{T+R}$ of the next R values for which the model needs to forecast. We use θ^i to denote the parameters of the ith forecast model. The exact form of the forecast model $F()$ that we used is described next, followed by a description of the training procedure.

Architecture of Forecast Model

Our forecasting model $F(H_T, \mathbf{x}_R | \theta^i)$ is based on Transformers like Informer [Zhou *et al.*, 2020] but with subtle differences that provide gains even for the baseline predictions.

Our model first applies a convolution on the input $\mathbf{z}$ series and features $\mathbf{x}$. The Transformer's multi-headed attention layers are on a concatenation of the convolution layer output.

$$\mathbf{z}^c = \text{Conv}(\mathbf{z}, w_y, 1) \quad \mathbf{x}^c = \text{Conv}(\mathbf{x}, w_f, 1)$$
$$\mathbf{h} = \text{Transformer}([\text{Concat}(\mathbf{z}_t^c, \mathbf{x}_t^c) + \mathbf{e}_t : t = 1 \ldots T])$$

where $\mathbf{h} \in \mathbb{R}^{T \times d_{\text{model}}}$ is the output of the transformer encoder and $\mathbf{e}_t$ denotes the positional encoding of time index t. In the decoder, we *warm start* to provide more context to the decoder apart from the encoder state $\mathbf{h}$. In warm start, we select the last s values in the input series and use them in the decoder of the transformer.

$$\mathbf{z}^c = \text{CONV}([\mathbf{z}_{T-s+1\ldots T}, \mathbf{0}_R], w_z, 1)$$
$$\mathbf{x}^c = \text{CONV}([\mathbf{x}_{T-s+1\ldots T}, \mathbf{x}_{T+1\ldots T+R}], w_x, 1)$$
$$\mathbf{h}_\mu = \text{TransformerDecoder}_\mu(\text{Concat}(\mathbf{z}^c, \mathbf{x}^c) + \mathbf{e}, \mathbf{h})$$
$$\mathbf{h}_\sigma = \text{TransformerDecoder}_\sigma(\text{Concat}(\mathbf{z}^c, \mathbf{x}^c) + \mathbf{e}, \mathbf{h})$$
$$\mu = \text{Linear}(\mathbf{h}_\mu) \quad \sigma = \text{Softplus}(\text{Linear}(\mathbf{h}_\sigma))$$

where $\mathbf{0}_R$ is a vector of zeros of length R, $\mathbf{h}_\mu \in \mathbb{R}^{R \times d_{\text{model}}}$, $\mathbf{h}_\sigma \in \mathbb{R}^{R \times d_{\text{model}}}$, $\mu \in \mathbb{R}^{R \times 1}$, and $\sigma \in \mathbb{R}^{R \times 1}$.

Training Procedure

We start with base level series $(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_n, y_n)$. First we split the series into training, validation, and test sets of lengths l_{trn}, l_{val}, and l_{test} respectively as follows:

$$\mathcal{D}_{\text{trn}} = \{(\mathbf{x}_t, y_t) | t = 1, \ldots, l_{\text{trn}}\}$$
$$\mathcal{D}_{\text{val}} = \{(\mathbf{x}_t, y_t) | t = l_{\text{trn}} - T + 1, \ldots, l_{\text{trn}} + l_{\text{val}}\}$$
$$\mathcal{D}_{\text{test}} = \{(\mathbf{x}_t, y_t) | t = l_{\text{trn}} + l_{\text{val}} - T + 1, \ldots, n\}$$

Note that when $l_{\text{val}} \geq R$ or $l_{\text{test}} \geq R$, we use rolling-window setting as done in [Deshpande and Sarawagi, 2019; Rangapuram *et al.*, 2018] on validation and test sets.

We create training batches by sampling *chunks* of length $T + R$ from training set $\mathcal{D}_{\text{trn}}$ as

$$\{(\mathbf{x}_t, y_t) | t = p, \ldots p + T + R\}$$
$$\text{where } p \sim [1, \ldots, l_{\text{trn}} - (T + R - 1)]$$

Similarly, we process i-th aggregate series $z_1^i, \ldots, z_{n/K_i}^i$. Validation and test sets across base-level and aggregated series are aligned. To aggregate features, we simply take average value of each feature in the aggregation window for all aggregates. For an aggregate model for i-th aggregate with window size K_i, the forecast horizon is $R_i = R/K_i$. Similarly, the size of the history also changes. We set $T_i = B(T/K_i)$, so the aggregate model can effectively see the history B-times the size of base level model's history. We found setting $B{=}2$ works well in practice. It is also a good option for smaller datasets such as ETTH.

For each aggregate function A_i, we sample training chunks of size $T_i + R_i$ and train the parameters θ^i of the Transformer

model for the following training objective.

$$\max_{\theta^i} \sum_{(\mathbf{x}_j^i, \mathbf{z}_j^i)} \sum_{t=T_i+1}^{T_i+R_i} \log \mathcal{N}(z_t; (\mu_t, \sigma_t) = F(H_T, \mathbf{x}, t | \theta^i)) \tag{11}$$

3 Related Work

Time series forecasting is an extensively researched problem. In deep learning, the problem got renewed interest [Benidis *et al.*, 2020] after an RNN-based auto-regressive model [Flunkert *et al.*, 2017; Mukherjee *et al.*, 2018] established conclusive gains over conventional machine learning and statistical methods. While these early neural models were based on RNNs, more recently Transformer-based models have been found to provide faster and efficient predictions on time-series data [Li *et al.*, 2019; Lim *et al.*, 2020; Bansal *et al.*, 2021; Zhou *et al.*, 2020]. For long-term forecasting auto-regressive models are both slow and subject to cascading errors from previous predicted values for next-step predictions. Another option is non-auto-regressive models (NAR) which predict each future forecast independently in parallel [Wen *et al.*, 2017; Deshpande and Sarawagi, 2019; Zhou *et al.*, 2020]. However, since NAR models do not capture the joint distribution, they fail to provide accurate results for aggregate queries on forecasts. N-beats [Oreshkin *et al.*, 2019] is another recent architecture for time-series forecasting based on layered residual connections, specifically designed for interpretability. Our technique is orthogonal to the underlying time-series forecasting model.

Multi-level Models Our approach of independently predicting models at different aggregation levels and then reconciling is related to the approach in [Taieb *et al.*, 2017; Ben Taieb and Koo, 2019; Wickramasuriya *et al.*, 2019]. Their hierarchies are over items and consider only sum-based aggregates whereas we create hierarchies along time and consider arbitrary linear aggregates. Also, our objective is to minimize the KL distance with the distribution at various levels whereas they focus on error minimization and their parameterization of the reconciliation is different from ours. [Han *et al.*, 2021] is another multi-level model for item hierarchies that train models in a bottom up manner with higher level models regularized to be consistent with lower layers. They do not model inter-item correlation, can only efficiently handle fixed aggregates, and do not revise base-level forecasts. We obtain gains at the base-level too via the hierarchies as we show in our empirical comparison. [Rangapuram *et al.*, 2021] propose an end-to-end method that produces coherent forecasts. However, their method relies on the sampling to obtain covariance of the joint distribution whereas we train the covariance through our inference method alone.

Beyond Base-level Forecasting Loss Our method of learning the slope as an aggregate in the forecast horizon is related to recent efforts at trying to learn the shape of the series. Recently, [LE GUEN and THOME, 2019; Guen and Thome, 2020] learn the shape of the series in forecast horizon using a differentiable approximation of Dynamic Time Warping (DTW) loss function. They also contain additional losses that minimize distortion and increase diversity. SRU
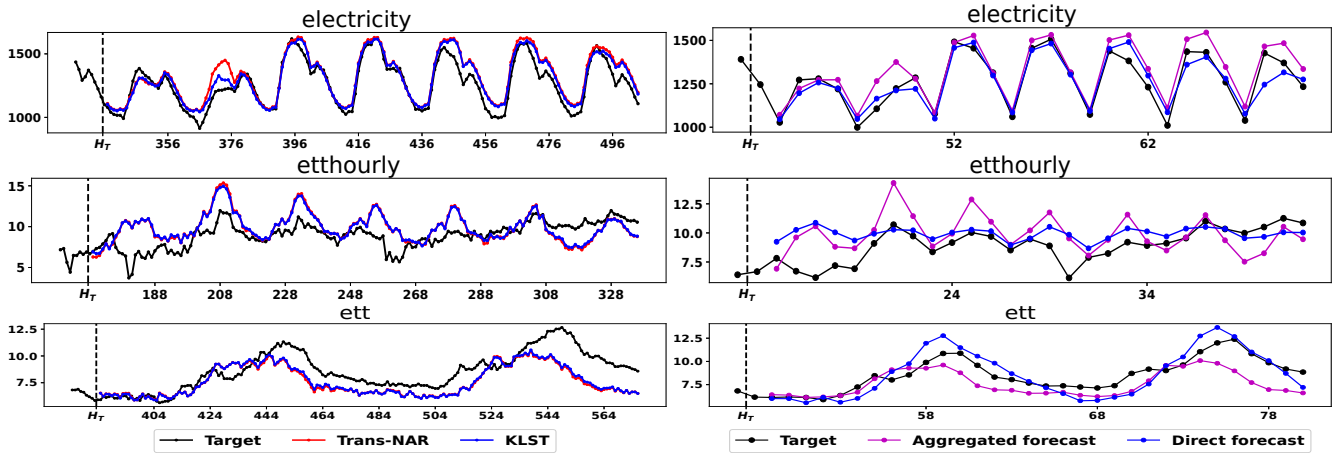
Figure 1: Left Column: Base level forecasts, Right Column: Forecasts of Sum aggregate with $K = 6$.

captures moving statistics of the history through a recurrent layer [Oliva *et al.*, 2017]. However, they use the statistics only to capture dependence among observed variables.

Long-term Forecasting Model Recently [Zhou *et al.*, 2020] proposed a Transformer architecture for long-range time-series forecasting. Their focus is to increase efficiency of the self-attention in Transformers for long series by enforcing sparsity. To the best of our knowledge, this is the best performing of existing methods for long-term forecasting. We compare empirically with this model and present significant gains on multiple datasets.

4 Experiments

We evaluate our method on four real-life datasets and contrast with state-of-the-art methods of long-term forecasting.

4.1 Datasets

Electricity This dataset contains national electricity load at Panama power system[2]. The dataset is collected at an hourly granularity from January 2015 to March 2020.

Solar This dataset is on hourly photo-voltaic production of 137 stations and was used in [Salinas *et al.*, 2019].

ETT (Hourly and 15-minute) This dataset contain a time-series of oil-temperature at an electrical transformer [3] collected from July 2016 to June 2017. The dataset is available at two granularities — 15 minutes (ETT) and one hour (ETTH). Both were used in [Zhou *et al.*, 2020].

4.2 Base Models

Informer [Zhou *et al.*, 2020] is a recent Transformer-based model specifically designed for long-term forecasting. They also use a non-auto-regressive model and predict each future value independently.

Trans-NAR refers to our Transformer architecture described in Section 2.3 but where we predict only a single model on only the base-level forecasts.

---

[2]https://data.mendeley.com/datasets/byx7sztj59/1
[3]https://github.com/zhouhaoyi/ETDataset

Trans-AR Trans-AR is an autoregressive version of the transformer architecture described in Section 2.3. At the t-th step during prediction, it performs convolution on the window $\hat{z}_{t-w_y+1}, \ldots, \hat{z}_{t-1}$ of predicted values. When $t < w_f$, the convolution window overlaps with the history H_T and observed values from H_T is used.

SHARQ SHARQ [Han *et al.*, 2021] is a hierarchical forecasting method in which a regularization loss ensures that the coherency error between aggregate predictions and aggregation of base-level predictions is minimized. Since SHARQ is agnostic to the underlying forecasting model, we use our baseline Trans-NAR as the base model for SHARQ. For j-th value of an aggregate model, we write the regularization term of SHARQ as:

$$\lambda_i(\hat{\mu} - \mathbf{a}^T \boldsymbol{\mu}_{w_{\cdot,j}})^2 + \left((\hat{q}_\tau - \hat{\mu})^2 - \mathbf{a}^T(\hat{q}_\tau - \boldsymbol{\mu}_{w_{\cdot,j}})^2\mathbf{a}\right)^2 \tag{12}$$

where $\hat{\mu}$ and $\hat{q}_\tau$ denote the predicted mean and aggregate model at j-th position. This model requires a fresh model to be trained for each aggregate unlike our method.

KLST is our method as described in Algorithm 1. The choice of aggregate functions and range of K values are based on validation set and chosen from aggregate functions = { Sum, Slope} and K={6,12}. For all datasets, we assign weights α={10,0.5} to Sum and Slope aggregates respectively. Since direct predictions on Sum aggregate are often more accurate, we assign higher weight to it.

4.3 Evaluation Metric

We evaluate probabilistic forecasts using the well-known [Salinas *et al.*, 2019] Continuous Ranked Probability Score (CRPS) metric. If we denote the predicted CDF for a target value y_t by $F_t(\cdot)$, we can write the CRPS by integrating the quantile loss over all possible quantiles [Salinas *et al.*, 2019]:

$$\text{CRPS}(F_t^{-1}, y) = \int_0^1 2\big(\alpha - \mathcal{I}_{[F_t^{-1}(\alpha) < q]}\big)(y_t - q)d\alpha$$

It reduces to Mean Absolute Error (MAE) when the forecast is a point, and is a proper scoring metric of the quality of an entire distribution including its mean and variance.

Dataset Agg	Model	K				
		1	4	8	12	24
ETT Sum	Informer	7.01	7.00	7.00	7.00	6.98
	Trans-AR	3.03	3.29	3.38	3.38	3.43
	Trans-NAR	1.25	1.36	1.39	1.39	1.38
	SHARQ	1.25	1.87	1.78	1.80	1.82
	KLST	**1.17**	**1.14**	**1.17**	**1.19**	**1.22**
ETT Slope	Trans-NAR	1.25	**0.13**	**0.07**	**0.06**	0.05
	KLST	**1.17**	0.30	0.12	**0.06**	**0.04**
ETT Diff	Trans-NAR	1.25	**0.14**	**0.16**	**0.20**	0.29
	KLST	**1.17**	0.33	0.26	0.25	**0.26**
Solar Sum	Informer	41.02	36.31	34.85	17.55	13.14
	Trans-AR	21.13	18.91	18.40	16.37	16.17
	Trans-NAR	13.85	13.25	12.95	12.78	12.43
	SHARQ	13.85	13.36	13.22	14.21	**11.60**
	KLST	**12.95**	**12.73**	**12.54**	**12.43**	12.21
Solar Slope	Trans-NAR	13.85	4.86	3.02	4.10	0.39
	KLST	**12.95**	**4.49**	**2.82**	**3.98**	**0.37**
Solar Diff	Trans-NAR	13.85	5.03	5.98	12.59	5.62
	KLST	**12.95**	**4.63**	**5.53**	**12.23**	**5.35**
ETTH Sum	Informer	4.80	4.77	4.73	4.67	4.57
	Trans-AR	1.96	2.01	1.98	2.01	1.96
	Trans-NAR	1.79	1.92	1.93	1.92	1.89
	SHARQ	1.79	1.91	1.73	1.75	1.78
	KLST	**1.64**	**1.61**	**1.65**	**1.67**	**1.69**
ETTH Slope	Trans-NAR	1.79	**0.26**	0.20	0.14	0.07
	KLST	**1.64**	0.37	**0.18**	**0.11**	**0.06**
ETTH Diff	Trans-NAR	1.79	**0.27**	0.39	0.46	0.50
	KLST	**1.64**	0.40	**0.37**	**0.39**	**0.41**
Elec Sum	Informer	172.3	159.7	155.8	118.1	109.6
	Trans-AR	140.2	137.8	134.0	109.6	104.7
	Trans-NAR	54.1	53.5	52.3	50.8	48.4
	SHARQ	54.1	**49.8**	**47.0**	50.5	46.3
	KLST	**50.2**	50.6	49.6	**48.4**	**46.2**
Elec Slope	Trans-NAR	54.1	8.96	6.25	5.65	2.23
	KLST	**50.2**	**8.26**	**5.76**	**5.18**	**2.14**
Elec Diff	Trans-NAR	54.1	9.50	13.23	18.37	16.13
	KLST	**50.2**	**8.80**	**12.22**	**16.84**	**15.44**

Table 1: Comparison of KLST using CRPS – (i) with all baselines on Long-Term Forecasts (Column with $K=1$). (ii) with Trans-NAR on unseen aggregates (Columns $K=4,8,12,24$).

4.4 Qualitative Results

We present some anecdotes to provide insights on why KLST should enhance the accuracy of base-level forecasts. In Figure 1 (left column) we show that forecasts by KLST (blue) are closer to the ground truth (black) compared to the base-level model Trans-NAR (red). We explain these gains by showing the improved accuracy of the model that independently predicts the sum aggregate for a window of $K = 6$ in the right column of Figure 1. We observe that direct forecast values

Dataset	# Series	Avg. T	R	train-len. /series	test-len. /series
ETT	1	384	192	55776	13824
ETTH	1	168	168	14040	3360
Electricity	1	336	168	36624	9072
Solar	137	336	168	7009	168

Table 2: Summary of the datasets

are much closer to the ground-truth than those from aggregating base-level forecasts. When these direct aggregates are used to revise the base-level forecasts to be consistent with these aggregates (as described in Section 2.2), we obtain better forecasts at the base-level.

4.5 Comparison of KLST with Baselines

Impact of KLST on Long-Term Forecasts In Table 1, we compare KLST with the baselines described in Sec. 4.2. Under Sum aggregate and column $K=1$ (highlighted), we compare the CRPS of base level forecasts for Informer, Trans-AR, Trans-NAR, SHARQ and KLST. For base-level forecasts, KLST performs better than all baselines while Trans-NAR and SHARQ are share the second position. Since SHARQ does not update base level model parameters, base level forecasts of SHARQ are identical to Trans-NAR.

Results on Unseen Aggregates Post Inference An important aspect of time-series analytics is the effectiveness of base-level forecasts on various types of aggregates at various granularities. Training an aggregate model for each aggregate and at each granularity is rather expensive. Our approach is to train a few representative aggregate models using a fixed set of K_i values. Then, forecasts after consensus can be used to compute other aggregate and at other granularities.

In Table 1 (columns with $K=4,8,12,24$), we compare KLST with (i) all baselines for Sum aggregate and (ii) with Trans-NAR for Slope and Diff aggregates. The set of K values we use in Table 1 also contain the K for which no aggregate model is trained. Also, no aggregate model was trained corresponding to the Diff aggregate. For each aggregate and for each K, first we aggregate the forecasts of baselines and KLST and then evaluate the aggregates using CRPS. Since SHARQ does not update the bottom level forecasts, we have trained a separate aggregate model for each K and Sum aggregate using SHARQ regularization loss (Eqn. 12).

For Sum aggregate, KLST outperforms all baselines on ETT and ETTH dataset. SHARQ performs better than KLST on Solar and Electricity for $K=24$ and $K=4,8$ respectively. Since SHARQ requires training a separate model for each aggregate and K, we compare SHARQ only on Sum aggregate.

For Slope and Diff aggregates, KLST outperforms Trans-NAR on Solar and Electricity. Whereas for ETT and ETTH, KLST is better than Trans-NAR in 12 out of 20 cases.

5 Conclusion

In this paper we addressed the problem of long-range forecasting with the goal of providing accurate probabilistic predictions, not just at the base-level but also at dynamically chosen aggregates. We proposed a simple method of independently training forecast models at different aggregation levels, and then designing a consensus distribution to minimize distance with forecasts from each component distribution. We obtained significant gains in accuracy over the base-level and over new aggregate functions at new granularity levels defined during test time. Compared to existing hierarchical methods, we impact both base-level predictions and new aggregates defined dynamically.

References

[Bansal *et al.*, 2021] Parikshit Bansal, Prathamesh Deshpande, and Sunita Sarawagi. Missing value imputation on multidimensional time series. *arXiv preprint arXiv:2103.01600*, 2021.

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

[Benidis *et al.*, 2020] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Bernie Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, et al. Neural forecasting: Introduction and literature overview. *arXiv preprint arXiv:2004.10240*, 2020.

[Deshpande and Sarawagi, 2019] Prathamesh Deshpande and Sunita Sarawagi. Streaming adaptation of deep forecasting models using adaptive recurrent units. In *ACM SIGKDD*, 2019.

[Deshpande *et al.*, 2021] Prathamesh Deshpande, Kamlesh Marathe, Abir De, and Sunita Sarawagi. Long horizon forecasting with temporal point processes. In *ACM WSDM*, 2021.

[Flunkert *et al.*, 2017] Valentin Flunkert, David Salinas, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *CoRR*, abs/1704.04110, 2017.

[Gasthaus *et al.*, 2019] Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. Probabilistic forecasting with spline quantile function rnns. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1901–1910, 2019.

[Guen and Thome, 2020] Vincent Le Guen and Nicolas Thome. Probabilistic time series forecasting with structured shape and temporal diversity. *arXiv preprint arXiv:2010.07349*, 2020.

[Han *et al.*, 2021] Xing Han, Sambarta Dasgupta, and Joydeep Ghosh. Simultaneously reconciled quantile forecasting of hierarchically related time series. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 190–198, 2021.

[LE GUEN and THOME, 2019] Vincent LE GUEN and Nicolas THOME. Shape and time distortion loss for training deep time series forecasting models. In *Advances in Neural Information Processing Systems 32*. 2019.

[Li *et al.*, 2019] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems*, pages 5243–5253, 2019.

[Lim *et al.*, 2020] Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting, 2020.

[Mukherjee *et al.*, 2018] Srayanta Mukherjee, Devashish Shankar, Atin Ghosh, Nilam Tathawadekar, Pramod Kompalli, Sunita Sarawagi, and Krishnendu Chaudhury. AR-MDN: associative and recurrent mixture density networks for eretail demand forecasting. *CoRR*, abs/1803.03800, 2018.

[Oliva *et al.*, 2017] Junier B. Oliva, Barnabás Póczos, and Jeff G. Schneider. The statistical recurrent unit. In *ICML*, pages 2671–2680, 2017.

[Oreshkin *et al.*, 2019] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[Rangapuram *et al.*, 2018] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7796–7805. 2018.

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

[Salinas *et al.*, 2019] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *Advances in Neural Information Processing Systems*, pages 6827–6837, 2019.

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

[Wen *et al.*, 2017] Ruofeng Wen, Kari Torkkola, and Balakrishnan Narayanaswamy. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

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

[Zhou *et al.*, 2020] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. *arXiv preprint arXiv:2012.07436*, 2020.