Learning Mixture Structure on Multi-Source Time Series for Probabilistic Forecasting

Tian Guo
RAM Active Investments, Geneva, Switzerland
tig@ram-ai.com

Abstract

In data-driven applications nowadays, it is increasingly desirable to collect data from different sources for enhancing performance. In this paper, we are interested in the problem of probabilistic forecasting with multi-source time series. We propose a neural mixture structure based probability model for learning different predictive relations and their adaptive combinations from multi-source time series. We present the prediction and uncertainty quantification methods which are applicable to different distributions of target variables. Additionally, given the imbalanced and unstable behaviors observed during the direct training of the proposed mixture model, we develop a phased learning method and provide a theoretical analysis. In experimental evaluations, the mixture model trained by the phased learning exhibits competitive performance on both point and probabilistic prediction metrics. Meanwhile, the proposed uncertainty conditioned error suggests the potential of the mixture model's uncertainty score as a reliability indicator of predictions.

1 Introduction

Time series data are prevalent in many applications [1], and in this paper we are interested in the probabilistic forecasting in the setting of multi-source time series. The data source can be distinct locations, entities, records, sensors, etc., from where data is collected. This multi-source data setting can fit into a variety of scenarios, where learning from a single source may be sub-optimal. For instance, in finance, different operations by market participants (e.g., performing transactions, putting limit orders, etc.) are separately recorded, thereby giving rise to distinct sources of features reflecting the underlying characteristics of markets [2, 3]. In environmental monitoring where sensors are usually deployed in multiple sites, each site is a data source, and collectively using the data from different sensor sites would be favorable to predictive analytics [4].

Challenges. Multi-source time series data usually carry different dynamics information and have time-varying relevance to the target variable [5, 6], as illustrated in Fig. 1. Meanwhile, the target variable might follow different distributions depending on applications [7]. For downstream decision making, besides point predictions, it is increasingly preferred to provide additional predictive information, e.g., uncertainty score, quantiles, etc [8, 9, 10]. All the above highlight the need for a flexible and adaptive probabilistic model structure on multi-source data.

As for the probability mixture model, it is typically composed of hierarchical components, and the interactions between these components during the training affect the end performance. For instance, as shown in Sec. 4.2, direct optimizing our mixture model' loss function leads to imbalanced learning behaviors of different data-source-specific components, implying that some data sources' predictive power is inadequately or even biasedly captured. This further hinders exploiting the full capacity of mixture models.

A causal view on dynamical systems workshop at the 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Contributions. Specifically, the contribution is as follows:

(1) We present the neural mixture structure based probability model on multi-source time series. It is equipped with a representation module to encode multi-source data, and a representation sharing mechanism to serve the prediction and weight modules.

(2) We present the adaptive prediction and uncertainty quantification methods, and analyse the implication of different components in the uncertainty score. This inference process is applicable to different target distributions, e.g., normal, and log-normal distributions in this paper.

(3) We demonstrate the imbalanced learning behaviors when directly minimizing the proposed mixture model's loss function, and then develop the phased learning method with theoretical analysis.

(4) Through experiments on real datasets from finance and environmental monitoring scenarios, our mixture model trained by the phased learning exhibits competitive performance on both point and probabilistic prediction metrics. Meanwhile, we put forward uncertainty conditioned error analysis to compare uncertainty scores from different methods.

2 Related work

Probabilistic Forecasting on Time Series. Probabilistic forecasting focuses on the predictive probabilistic characteristics of target variables, and is aimed to provide a variety of predictive quantities for downstream decision making, e.g., uncertainty scores, predictive quantiles, prediction intervals, etc [9, 7, 11].

Recently, many neural network based probabilistic forecasting models have been proposed for time series data. [12, 8] respectively equipped RNNs with the linear state space models and random effect components. [7] used RNNs with autoregressive and covariate inputs for probabilistic forecasts. [13, 14] focused on modeling multi-variate time series. Stochastic sequential models [15, 16, 17, 18, 19, 20] developed stochastic temporal latent states for modeling generative dynamics in time series. Given the setting of multi-source data in this paper, above methods mostly model the input data as a whole and could hardly differentiate various time-varying relations in multi-source data, thereby potentially leading to inferior predictive performance.

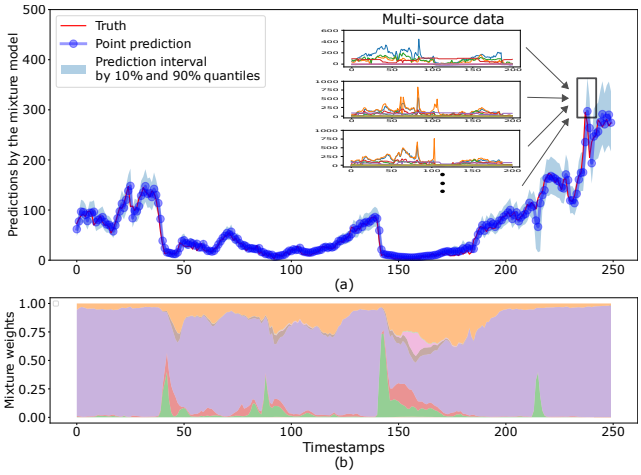

Figure 1: (a) Illustration of multi-source time series based probabilistic forecasting. The prediction at each timestep (e.g., highlighted by the rectangle) is based on the multi-source data in a look-back window. (b) Time-varying mixture weights of data sources in the prediction. Each color corresponds to a data source. At each timestamp, the larger the colored area, the higher the weight and the more relevance to the target the corresponding data source has. The sum of all colored value ranges is 1.0 at each timestamp.

Mixture Models. Due to the flexibility and adaptivity, mixture models have been applied in a variety of areas, such as unsupervised learning [21, 22, 23], natural language processing [24, 25, 26], computer vision [27, 28, 29], etc. Few attempts have been made to explore the mixture structure, its inference, and learning methods for probabilistic forecasting with multi-source data.

As a category of latent variable models, mixture models are normally trained through (stochastic) expectation maximization (EM) style optimizations [30]. [31] provided theoretical convergence analysis for linear mixture models. [32] integrated variance reduction into stochastic EM algorithms. In this paper, we explore the practical treatments for stably and balancedly learning different data source corresponding modules of the mixture model.

3 Problem and Proposed Model

3.1 Problem Statement

Let y_t be the value of the target variable at time t. Assume S number of different data sources, which are indexed by $s = 1, \cdots, S$. The multi-source data used to predict y_t is denoted by $\mathbf{x}_{1:S,<t} = \{\mathbf{x}_{s,<t}\}_{s=1}^{S}$, where each element $\mathbf{x}_{s,<t}$ is a multi-dimensional time series from a limited window of historical data of source s prior to timestamp t. Depending on applications, the latest timestamp in $\mathbf{x}_{s,<t}$ could be $t-1$ for one-step or $t-h$ for h-step ahead prediction. For simplicity, we skip the notations for time series dimensions and history window length.

The task is to model the predictive density function $p(y_t \,|\, \mathbf{x}_{1:S,<t})$ for providing predictive quantities such as point predictions, uncertainty scores and so on [33, 34]. Note that in this paper, we assume that $\mathbf{x}_{1:S,<t}$ implicitly contains the auto-regressive data of the target variable, i.e., $y_{<t}$ when available, either as an individual data source or being included into one.

3.2 Model Structure

The idea of the mixture structure on $p(y_t \,|\, \mathbf{x}_{1:S,<t})$ is to introduce a latent discrete random variable z_t for differentiating various predictive relations between $\mathbf{x}_{1:S,<t}$ and y_t. In our case, z_t is defined on the data source index set, i.e., $z_t \in \{1, \cdots, S\}$, thereby leading to data-source-wise prediction modules. The probability mass function of z_t serves as the weight module.

Meanwhile, instead of using the raw data $\mathbf{x}_{1:S,<t}$ as inputs to the weight and prediction modules, we feed a set of latent representations that are learned from $\mathbf{x}_{1:S,<t}$ by powerful neural networks. Let $\mathbf{h}_{1:S,t}$ denote this set of representation vectors $\{\mathbf{h}_{s,t}\}_{s=1}^{S}$. The factorization of the joint density function including z_t and $\mathbf{h}_{1:S,t}$ as latent variables is expressed as:

$$p(y_t \,|\, \mathbf{x}_{1:S,<t}) = \int \underbrace{p(\,\mathbf{h}_{1:S,t} \,|\, \mathbf{x}_{1:S,<t})}_{\text{representation module}} \cdot \sum_{s=1}^{S} \underbrace{\mathbb{P}(z_t = s \,|\, \mathbf{h}_{1:S,t})}_{\text{weight module}} \cdot \underbrace{p(y_t \,|\, z_t = s, \mathbf{h}_{s,t})}_{\text{prediction module}} \, \mathrm{d}\mathbf{h}_{1:S,t} \tag{1}$$

Representation Module is to encode each source's data into the latent representations that will serve the prediction and weight modules. There are various probabilistic representation learning methods for time series, e.g., state-space models, stochastic recurrent neural networks, etc [35, 36, 20]. Since advancing the representation learning is not the focus of this paper, we choose a simple yet flexible way to formulate the representation module, namely, placing a Dirac delta distribution centered at the hidden state output of a temporal data encoder [37, 16, 38]. This module is formulated as:

$$p(\,\mathbf{h}_{1:S,t} \,|\, \mathbf{x}_{1:S,<t}) = \prod_{s=1}^{S} \delta(\mathbf{h}_{s,t} - \widetilde{\mathbf{h}}_{s,t}) \quad \text{s.t.}\, \widetilde{\mathbf{h}}_{s,t} = \mathrm{NN}_{\eta_s}(\mathbf{x}_{s,<t}), \forall s = 1, \cdots, S \tag{2}$$

, where $\delta(a) = 1$ only when $a = 0$. $\mathrm{NN}_{\eta_s}(\mathbf{x}_{s,<t})$ is the encoder to model the temporal dynamics of source s and η_s denotes the trainable parameter in it. In the experiment, we employ a LSTM as the temporal encoder, while it is flexible to use others.

If it is hypothesized to there exist global latent factors governing the dynamics across data sources, the above formulation is extensible by learning a global representation from all sources and then sharing it across individual representations [39, 8].

Prediction Module is to parameterize the source-wise predictive distribution $p_{\omega_s}(y_t \,|\, z_t = s, \mathbf{h}_{s,t})$ with the learnable parameters ω_s. For instance, for $y_t \in \mathbb{R}$ and the normal distribution is applied, we have $y_t \,|\mathbf{h}_{s,t} \sim \mathcal{N}(\mu_{s,t}, \sigma_{s,t}^2)$, where $\mu_{s,t}, \sigma_{s,t}^2 = g_{\omega_s}(\mathbf{h}_{s,t})$, and $g_{w_s}(\mathbf{h}_{s,t})$ can be any flexible function such as multiple dense layers parameterized by ω_s [7]. For $y_t \in \mathbb{R}^+$, we can choose the log-normal distribution, i.e., $\log y_t \,|\mathbf{h}_{s,t} \sim \mathcal{N}(\mu_{s,t}, \sigma_{s,t}^2)$ [40, 33].

Weight Module is the probability mass function of z_t:

$$\mathbb{P}_\theta(z_t = s \,|\, \mathbf{h}_{1:S,t}) \triangleq \frac{\exp(\, f_s(\mathbf{h}_{s,t})\,)}{\sum_{k=1}^{S} \exp(\, f_k(\mathbf{h}_{k,t})\,)} \tag{3}$$

, where θ_s denotes the parameters of the logit function for data source s. $f_{\theta_s}(\cdot)$ can be any flexible function such as neural networks. The trainable parameters from all logit functions constitute the parameter set of this module, i.e., $\theta = \{\theta_s\}_{s=1}^{S}$.

4 Adaptive Inference and Phased Learning

4.1 Inference

We mainly present the adaptive point prediction and uncertainty quantification. Other predictions like quantiles, intervals, etc., and their derivations are described in Appendix A.2. For simplicity, we express the weight module as $\mathbb{P}_\theta(z_t = s \mid \cdot)$.

Point Prediction. It is the predictive mean of the mixture, which is the weighted sum of each source's predictive mean.

$$\hat{y}_t \triangleq \mathbb{E}[y_t \mid \mathbf{x}_{1:S,<t}] = \sum_{s=1}^{S} \mathbb{P}_\theta(z_t = s \mid \cdot)\mathbb{E}[y_t | z_t = s, \widetilde{\mathbf{h}}_{s,t}] \tag{4}$$

, where $\mathbb{E}[y_t \mid z_t = s, \widetilde{\mathbf{h}}_s]$ denotes the predictive mean by source s. For instance, it is $\mu_{s,t}$ for normal distribution and $\exp(\mu_{s,t} + \frac{1}{2}\sigma^2_{s,t})$ for the log-normal distribution [33].

Uncertainty Score. In forecasting tasks, the predictive variance is commonly used as an uncertainty measure [41, 42]. With a single forward pass, our mixture model provides the uncertainty score $\hat{u}_t$ consisting of two components as follows:

$$\hat{u}_t = \underbrace{\sum_{s=1}^{S} \mathbb{P}_\theta(z_t = s|\cdot)\mathrm{Var}(y_t|z_t = s, \widetilde{\mathbf{h}}_{s,t})}_{\text{Aleatoric Uncertainty}} + \underbrace{\sum_{s=1}^{S} \mathbb{P}_\theta(z_t = s|\cdot)\, \mathbb{E}^2[y_t|z_t = s, \widetilde{\mathbf{h}}_{s,t}] - \mathbb{E}^2\left[y_t \mid \mathbf{x}_{1:S,<t}\right]}_{\text{Mixture Uncertainty}} \tag{5}$$

For instance, the variance term $\mathrm{Var}(y_t|z_t = s, \widetilde{\mathbf{h}}_{s,t})$ is $\sigma^2_{s,t}$ for normal distribution, and $\exp(\sigma^2_{s,t} - 1) \cdot \exp(2\mu_{s,t} + \sigma^2_{s,t})$ for log-normal distribution [43].

In Eq.5, the aleatoric uncertainty quantifies predictive noise inherent in the target, while the mixture part reflects the variation among individual data sources' predictions. The mixture weights serve to adjust the contributions of different sources. This is different from some uncertainty quantification methods using equal-weighted components, e.g., model snapshots, latent states [44, 45, 20].

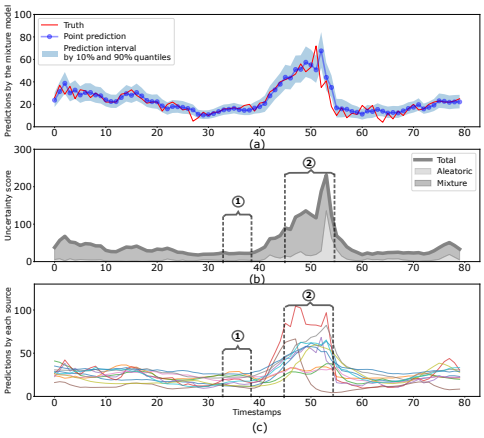

Figure 2: Illustration of the predictive uncertainty by the mixture model on AIR data. (a) True values, point and interval predictions. (b) Uncertainty and its component values. The total uncertainty score shown by the thick line is the sum of values represented by the light and dark areas. (c) Point predictions by data-source-specific prediction modules.

In the following, through the testing examples in Fig.2, we give a qualitative interpretation of these two uncertainty components. In time period ①, the target variable stays relatively stable in Fig.2(a). Accordingly, in Fig.2(b), the aleatoric uncertainty is low. In Fig.2(c), the predictions by individual sources show some consensus and reside in a close value range, and thus the mixture uncertainty is also low in ①. On the other hand, in period ②, when the target is fluctuating in Fig.2(a), besides the aleatoric part, the mixture uncertainty drives up the total uncertainty because of the widened prediction dispersion as shown in Fig.2(c). Interestingly, Fig.2(c) also reveals that when the target is in a volatile period (e.g., ②), just some of the data sources can provide seasonably reasonable predictions, while the rest of the sources output too high or too low predictions and seem to lack enough predictive power.

4.2 Issues with Direct Learning

In this part, we analyze the imbalanced learning behaviors when directly minimizing the loss function of the mixture model.

We first define the training dataset $\mathcal{D} = \left\{y_t, \mathbf{x}_{1:S,<t}\right\}_{t\in\mathcal{T}}$, where $\mathcal{T}$ represents the set of timestamps at which data instances are collected. Let Θ represent the set of trainable parameters, i.e., $\Theta = \left\{\{\eta_s, \omega_s\}_{s=1}^{S}, \theta\right\}$. Then, the mixture loss is expressed as:

$$\mathcal{L}\left(\Theta\,;\mathcal{D}\right) \triangleq -\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} \log \mathbb{E}_{\mathbf{h}_{1:S,t}}\left[\sum_{s=1}^{S} \mathbb{P}_\theta(z_t = s|\mathbf{h}_{s,t})\, p_{\omega_s}(y_t|z_t = s, \mathbf{h}_{s,t})\right] \tag{6}$$

Next, through Fig.3, we illustrate the imbalanced learning status of different data sources when directly minimizing the mixture loss via stochastic gradient descent (SGD) based methods [46]. This imbalanced and unstable behavior might hinder the mixture model from adequately capturing different data sources' predictive relations and their combinations, and thus impairs the overall performance as shown in Sec.5.

Specifically, Fig.3(a) reports the source-wise prediction error which is defined by the rooted mean squared error between training true values and predictive means by each source's prediction module over training epochs. It shows that some error curves converge poorly, e.g., the purple one implies the corresponding source's prediction module hardly captures useful predictive relations. As for the converged curves, the noticeable error dispersion suggests that those prediction modules might be learned to different extents.

As a comparison, Fig.3(b) shows the source-wise prediction error of the model trained by the proposed phased learning in Sec.4.3. The source-wise errors stably and rapidly decline to a close range, and, likely, all prediction modules are reasonably learned. In the zoom-in part, the spread between converged error curves still implies the different predictive powers of data sources. Notably, in Fig.3(b), the purple error curve also converges nicely. It implies the corresponding data source has the predictive power, but it is not properly captured in Fig.3(a). In Sec.5, we will show that the overall prediction performance of the mixture model considerably benefits from this phased learning.

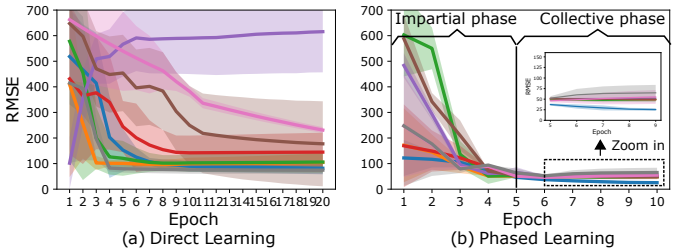

Figure 3: Illustration of imbalanced and unstable behaviours of directly learning the mixture model, compared with the proposed phased learning. Each curve represents the rooted mean squared error (RMSE) between target values and predictive means by a source's prediction module over training epochs. The results are based on five different training runs on AIR data in Sec.5.

Lemma 4.1. *In minimizing the mixture loss via SGD optimization, given a random training sample indexed by t, the vectorized derivative of a source's prediction and representation modules are denoted by:*

$$\mathbf{g}_{\omega_s,t} \triangleq -\frac{\partial \log p_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t})}{\partial \omega_s}\Big|_{\Theta} \tag{7}$$

$$\mathbf{g}_{\eta_s,t} \triangleq -\frac{\partial \log p_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t})}{\partial \widetilde{\mathbf{h}}_{s,t}} \nabla \widetilde{\mathbf{h}}_{s,t}\Big|_{\Theta} \tag{8}$$

$$\mathbf{d}_{\eta_s,t} \triangleq -\sum_{k=1}^{S} \pi_{k,t} \frac{\partial \log \mathbb{P}_\theta(z_t = k|\cdot)}{\partial \widetilde{\mathbf{h}}_{s,t}} \nabla \widetilde{\mathbf{h}}_{s,t}\Big|_{\Theta} \tag{9}$$

where $\nabla \widetilde{\mathbf{h}}_{s,t}$ denotes the Jacobian matrix of source s's representation w.r.t. η_s. Meanwhile, the posterior weight $\pi_{s,t}$ is defined as:

$$\pi_{s,t} \triangleq p(z_t = s \,|\, y_t, \mathbf{x}_{1:S,<t}; \Theta) = \frac{p_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t})\, \mathbb{P}_\theta(z_t = s \,|\, \cdot)}{\sum_{k=1}^{S} p_{\omega_k}(y_t \,|\, z_t = k, \widetilde{\mathbf{h}}_{k,t})\, \mathbb{P}_\theta(z_t = k \,|\, \cdot)} \tag{10}$$

Then, the (stochastic) updates to the prediction and representation parameters of source s, i.e., ω_s and η_s, are:

$$\pi_{s,t}\,\mathbf{g}_{\omega_s,t} \;\; \textit{and} \;\; \pi_{s,t}\,\mathbf{g}_{\eta_s,t} + \mathbf{d}_{\eta_s,t} \tag{11}$$

(Derivation details in the appendix section.)

Lemma 4.1 sheds light on one potential cause of the imbalanced learning, i.e., the intertwined relation between posterior weights and parameter updates, and Sec.4.3 will present another theoretical analysis. Specifically, at the beginning of the training, all representation, prediction, and weight modules are barely trained. Some data sources receive low posterior weights (i.e., $\pi_{s,t}$), if they happen to have low (stochastic) predictive density values and/or low mixture weights, according to Eq.10. Then, based on Eq.11, low posterior weights lead to under-weighted (stochastic) updates and might slow down or even corrupt the learning progress of corresponding source's modules. In turn, this posterior weight bias might persist and result in the imbalanced learning status in Fig.3(a).

4.3 Phased Learning

To relax the above issues, we propose the phased learning method splitting the training process into two phases respectively for unbiasedly learning each data source's predictive power and then the adaptive combinations. (The proofs in this part are in the appendix section.)

Lemma 4.2. *In the iterative learning process, at any point of Θ, there exists the upper bound of the mixture loss:*

$$\mathcal{L}\left(\Theta\,;D_t\right) \leq \overline{\mathcal{L}}\big(\{\eta_s,\omega_s\}_{s=1}^{S}\,;D_t\big) \tag{12}$$

where D_t is a random training sample $D_t \triangleq (y_t, \mathbf{x}_{1:S,<t}) \in \mathcal{D}$.

$$\overline{\mathcal{L}}\big(\{\eta_s,\omega_s\}_{s=1}^{S}\,;D_t\big) \triangleq -\sum_{s=1}^{S}\frac{1}{S}\log p_{\omega_s}(y_t \mid z_t = s, \widetilde{\mathbf{h}}_{s,t}) - \log S a_t^* \tag{13}$$

and a_t^* is a constant value, $a_t^* = \min_s(\{\,\mathbb{P}(z_t = s \mid \cdot)\}\,)$.

First, the **impartial phase** learning is inspired by Lemma 4.2. It minimizes the upper bound Eq.13, thereby leading to the following update scheme:

$$\frac{1}{S}\mathbf{g}_{\omega_s,t} \;\; \text{and} \;\; \frac{1}{S}\mathbf{g}_{\eta_s,t} \tag{14}$$

, where $\mathbf{g}_{\omega_s,t}$ and $\mathbf{g}_{\eta_s,t}$ are defined in Eq.7 and 8. These updates only involve the prediction related parameters, i.e., the representation and prediction modules. Meanwhile, the equal-weighted manner allows to impartially learn the predictive relation of each data source.

Theorem 4.3. *For all stochastic derivatives $\mathbf{g} \in \{\mathbf{g}_{\omega_s}, \mathbf{g}_{\eta_s}, \mathbf{d}_{\eta_s}\}$ corresponding to the data source s, suppose it is bounded $\mathbb{E}\|\mathbf{g}\|^2 \leq G^2$. The trainable parameters related to a data source's prediction are denoted by $\Theta_s = [\omega_s, \eta_s]^\top$ and the corresponding gradient is $\mathcal{G}_s = \mathbb{E}[\mathbf{g}_{\omega_s}, \mathbf{g}_{\eta_s}]^\top$. Over the learning step $i = 0, \cdots, I$, the convergence of $\mathcal{G}_s$ to a stationary point in the direct optimization and the impartial-phase optimization are respectively:*

$$\textit{Direct:}\;\; \frac{1}{I}\sum_{i=0}^{I-1}\mathbb{E}\|\mathcal{G}_{s,i}\|^2 \leq \sqrt{10}G\sqrt{\frac{L\left(\mathcal{L}_{s,0}-\mathcal{L}_{s,I}\right)}{I\pi_s^*}} + \sum_{i=0}^{I-1}\frac{1}{2I}\left(\|\mathbb{E}[\mathbf{g}_{\eta_s,i}] - \mathbb{E}[\mathbf{d}_{\eta_s,i}]\|^2\right) \tag{15}$$

$$\textit{Impartial phase:}\;\; \frac{1}{I}\sum_{i=0}^{I-1}\mathbb{E}\|\mathcal{G}_{s,i}\|^2 \leq 2G\sqrt{\frac{L\left(\mathcal{L}_{s,0}-\mathcal{L}_{s,I}\right)}{I}} \tag{16}$$

, where $\pi_s^* \in (0,1)$ is the minimum of poster weights across data samples and iteration steps. $\mathcal{L}_{s,i}$ represents the negative log-likelihood of the predictions by data source s at step i.

Theorem 4.3 explains the convergence patterns of the direct and impartial-phase learning observed in Fig.3. As shown in Eq.15, there are two more factors that impact the convergence of each data source's prediction related parameters in the direct learning, compared with the impartial-phase

learning. The value of π_s^* is less than 1 and implies the slow value decline of the first term on the right-hand-side of Eq.15. Meanwhile, if the second term on the right-hand-side of Eq.15 dose not vanish completely, it converges to the neighbourhood of a stationary point, i.e., the corresponding prediction modules might be inadequately trained. For instance, the converged error curves in Fig.3(a) are still higher than the counterparts in Fig.3(b).

Second, the **collective phase** will update all modules' parameters by switching to minimize the original mixture loss Eq.6 until the end of the training. The weigh module is now activated to learn the adaptive combinations. The representations and prediction modules are updated as Eq.11. Thanks to the reasonably learned representation and prediction modules in the impartial phase, the posterior weight now in Eq.10 more reliably reflects different data sources' relevance to the target. In this phase, it amounts to fine-tuning the representations and prediction modules by considering the data source relevance reflected in posterior weights. For instance, in Fig.3(b), the zoom-in figure shows that some sources' error curves continue to decline in the collective phase.

In practice, the number of epochs of the impartial phase can be tuned as a hyper parameter. On the experiment datasets, the range of 5 to 15 epochs is enough for the impartial phase. Meanwhile, we found that the total number of epochs needed in the phased learning is comparable to the direct learning, i.e., having two phases does not incur the need for more epochs in total. Compared with the direct learning with a certain number of epochs, the phased learning can mostly present enhanced performance by running the impartial and collective phase with the same budget of epochs.

5 Experiments

In this section, we report the experimental comparison of different models and learning methods. The Appendix contains additional experiment details and results.

5.1 Baselines

DAR denotes DeepAR, which uses auto-regressive recurrent neural networks to model the probabilistic distribution of the target variable [7]. **DF** refers to the Deep Factor model, which is a global-local method based on a global neural network backbone and local probabilistic random effect models [8]. **AF** represents Autoformer [47], a decomposition based Transformer [48] variant with an auto-correlation mechanism.

DAR and DF are probability models, and AF is adapted for probabilistic forecasting based on [49]; hence all baselines can parameterize different distributions for the target variable and be evaluated on point and probabilistic prediction metrics. For baselines, multi-source data are fed as a whole. For the mixture model and baselines, probabilistic forecasts like predictive quantiles and intervals are derived via the same type of methods presented in A.2.

5.2 Datasets

Existing time series forecasting models mostly take as input single (univariate or multivariate) time series, while our mixture model works on multi (multivariate) time series from distinct sources. Thus, we use the following data from two scenarios. For each dataset, data instances are time-ordered, and we use the first 70% of points for training, the next 10% for validation, and the last 20% for testing.

AIR is from the environmental monitoring scenario [50]. There are twelve air-quality monitoring sites, and each site collects an eleven-dimensional time series including hourly air pollutant and meteorological observations, from March 2013 to February 2017. The target variable, which is the air pollutant value reflected in PM2.5 measurements, lies in the non-negative domain, and thus the log-normal distribution is applied. The time series from all sites forms a twelve-source data for forecasting the target variable of each site.

VOL is from the finance area, i.e., two liquid cryptocurrency exchanges, Bitfinex and Bitstamp. From the limit order book (LOB) and transaction records of each exchange, it collects two minute-level multi-dimensional feature time series, from May 2018 to September 2018. The target variable is the intra-day traded volume of bitcoins. It is the raw volume deduced by the intra-day periodicity and lies in the real value domain, and thus the normal distribution is applied. In a cryptocurrency

Table 1: Results of point and probabilistic prediction metrics (mean $\pm$ standard error). The best result is marked by the grey box. Due to the space limitation, the results of the rest of sites in AIR data are in Appendix C.

		RMSE ↓				MAE ↓			
	Data	DAR	DF	AF	**MIX**	DAR	DF	AF	**MIX**
AIR	Site 0	23.650 ±0.645	23.792 ±0.756	68.293 ±0.985	19.022 ±0.054	12.015 ±0.223	13.743 ±0.291	35.196 ±0.307	10.297 ±0.048
	Site 1	20.012 ±0.205	31.467 ±0.570	66.199 ±0.912	18.517 ±0.086	10.596 ±0.109	13.247 ±0.772	39.196 ±0.307	9.929 ±0.027
VOL	Exc. 0	1.206±0.00342	1.157±0.00389	1.892±0.00253	1.149 ±0.00168	0.925±0.00276	0.916±0.00259	1.021±0.00324	0.898 ±0.00588
	Exc. 1	1.382±0.0075	1.271±0.00177	2.010±0.0098	1.265 ±0.00302	0.995±0.00464	0.981±0.00101	1.032±0.00433	0.977 ±0.00197
		NLLm ↓				QLm ↓			
	Data	DAR	DF	AF	**MIX**	DAR	DF	AF	**MIX**
AIR	Site 0	3.922±0.011	4.196±0.013	5.439±0.004	3.879 ±0.011	0.298±0.004	0.327±0.003	0.233±0.001	0.0463 ±0.001
	Site 1	3.860±0.021	3.992±0.029	5.208±0.002	3.696 ±0.003	0.231±0.004	0.202±0.004	0.237±0.001	0.0505 ±0.001
VOL	Exc. 0	1.594±0.00485	1.578±0.00550	1.601±0.00324	1.541 ±0.00258	2.974±0.0413	4.498±0.0694	4.982±0.0983	2.840 ±0.0624
	Exc. 1	1.698±0.00853	1.660±0.00131	1.723±0.00732	1.647 ±0.00220	2.606±0.0958	4.801±0.119	5.012±0.1012	2.353 ±0.0930

exchange, it records different trading related operations, e.g., transaction data stores the information about executed trades, the limited order book (LOB) records the standing orders with buy or sell limits, etc. We formulate a multi-source data consisting of the time series from LOB and transaction features as well as historical volume.

5.3 Setup

Hyper-Parameters. The hyper-parameter search and associated training processes are run on a server with NVIDIA A100 GPUs. For all models, Bayesian optimization is applied to search in the hyper-parameter space [51]. Once the best hyper-parameters is fixed, each model is retrained for five times with different random seeds, and the average performance is reported. The hyper-parameter space is detailed in Appendix.

Point Prediction Metrics. The root-mean-square error (RMSE) and mean absolute error (MAE) are defined on true values and predictive means.

Probabilistic Prediction Metrics. The mean negative log likelihood (NLLm) is the predictive negative log-likelihood of testing instances averaged by the number of instances. The mean quantile loss (QLm) is the mean of quantile losses w.r.t. a set of quantile levels [9]. A quantile loss is defined on true values and predictive quantiles to reflect how well the predictive distribution fits the target variable's probabilistic characterises [8]. Detailed formations of these metrics are in Appendix.

5.4 Results

Comparison of Prediction Performance. In Table 1, on each site or exchange (exc.), the mixture model outperforms baselines on the point and probabilistic prediction metrics. It indicates that the mixture models more accurately characterize the distributional properties of the target variable. Compared with the baselines modeling multi-source data as a whole, the outperformance of the mixture model suggests that capturing adaptive relations in multi-source data can better harness the predictive power.

Comparison of Training Methods. Table 2 shows the results of the mixture models respectively trained by different learning methods. The phased learning noticeably improves the performance of the mixture model on all metrics, while in some cases the mixture model trained by the direct learning under-performs the baselines in Table 1. This observation suggests that the specialized training process is beneficial for exploiting the capacity of mixture models. Meanwhile, the low standard errors of the phased learning imply the training process is relatively more stable, thereby leading to more consistent generalization performance.

Table 2: Performance of the mixture models respectively trained by the direct and phased learning methods (mean $\pm$ standard error). Additional results are in Appendix C.

	Data	RMSE ↓		MAE ↓		NLLm ↓		QLm ↓	
		Direct	Phased	Direct	Phased	Direct	Phased	Direct	Phased
AIR	Site 0	27.173±1.617	19.022 ±0.054	14.905±0.423	10.297 ±0.048	4.103±0.133	3.879 ±0.011	0.0582±0.009	0.0463 ±0.001
	Site 1	41.549±2.013	18.517 ±0.086	17.876±1.102	9.929 ±0.027	3.772±0.006	3.696 ±0.003	0.0589±0.009	0.0505 ±0.001
VOL	Exc. A	1.172±0.00123	1.149 ±0.00168	0.934±0.0356	0.898 ±0.0588	1.596±0.00216	1.541 ±0.00258	3.012±0.0563	2.840 ±0.0624
	Exc. B	1.282±0.00203	1.265 ±0.00302	0.998±0.00123	0.977 ±0.00197	1.656±0.00158	1.647 ±0.00220	2.510±0.0890	2.353 ±0.0930

Comparison of Uncertainty Conditioned Errors. The uncertainty score can an indicator of the prediction error level, namely, high uncertainty scores imply unreliable predictions and thus potentially large prediction errors, and vice versa. Following this intuition, we propose the uncertainty conditioned prediction error to compare the uncertainty scores from different models. Specifically, given the predictions and associated uncertainty scores, we first derive the empirical quantiles of the uncertainty scores. Then, as shown in Fig.4, we calculate the prediction errors (RMSE) by using the predictions with the uncertainty scores falling within each interval of quantile levels.

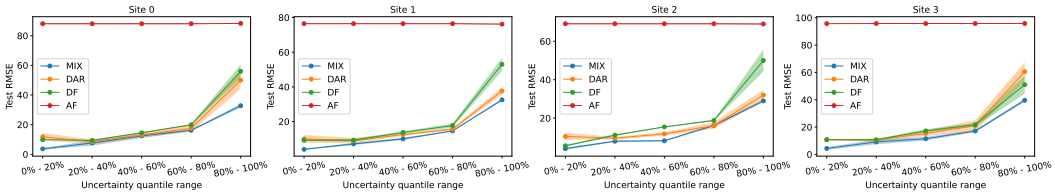

Figure 4: Uncertainty Conditioned Prediction Errors.

In Fig.4, based on the above intuition, the uncertainty conditioned error curve is expected to present a monotonic increasing pattern. The mixture model has monotonic error curves, while the baselines DAR and DF have some irregular errors, i.e., the error of the high uncertainty interval is lower than that of the low uncertainty interval. Compared with the baselines only capturing the aleatoric uncertainty, the mixture model's uncertainty score consists of the weighted aleatoric and mixture components, and it is more effective in reflecting the prediction error extent as a function of the uncertainty. Meanwhile, the mixture model has lower errors on all quantile level intervals. This also explains the overall competitive prediction performance of mixture models in Table 1 from the uncertainty perspective.

6 Conclusion

This paper focuses on the probabilistic forecasting with multi-source time series data . We present a neural mixture structure based probability model on multi-source time series and the associated phased learning method. The experiment results exhibit competitive performance of the proposed model and learning method on both point and probabilistic metrics. It reveals that capturing data-source-wise predictive relations and their adaptive combinations is important for exploiting the predictive power of multi-source data. Meanwhile, for mixture models with hierarchical and interacted components, understanding the learning behaviors of these components and accordingly customizing the learning method help realize the capacity of models.

The limitation of the present work lies in the representation learning of time series data. It will be worth exploring more advanced representation learning methods and studying the effect on the prediction performance. Meanwhile, the transition from the impartial to collect phase is more favored to have a systematic mechanism to avoid tedious hyper-parameter search.

References

[1] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20200209, 2021.

[2] R. Næs and J. A. Skjeltorp, "Order book characteristics and the volume–volatility relation: Empirical evidence from a limit order market," *Journal of Financial Markets*, vol. 9, pp. 408–432, 2006.

[3] D. Stosic, D. Stosic, T. B. Ludermir, and T. Stosic, "Collective behavior of cryptocurrency price changes," *Physica A: Statistical Mechanics and its Applications*, vol. 507, pp. 499–509, 2018.

[4] M. Wu, L. Tan, and N. Xiong, "Data prediction, compression, and recovery in clustered wireless sensor networks for environmental monitoring applications," *Information Sciences*, vol. 329, pp. 800–818, 2016.

[5] J. C. B. Gamboa, "Deep learning for time-series analysis," *arXiv preprint arXiv:1701.01887*, 2017.

[6] X. Zhu, F. Cai, J. Tian, and T. K.-A. Williams, "Spatiotemporal fusion of multisource remote sensing data: Literature survey, taxonomy, principles, applications, and future directions," *Remote Sensing*, vol. 10, no. 4, p. 527, 2018.

[7] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[8] Y. Wang, A. Smola, D. Maddix, J. Gasthaus, D. Foster, and T. Januschowski, "Deep factors for forecasting," in *International conference on machine learning*. PMLR, 2019, pp. 6607–6617.

[9] J. Gasthaus, K. Benidis, Y. Wang, S. S. Rangapuram, D. Salinas, V. Flunkert, and T. Januschowski, "Probabilistic forecasting with spline quantile function rnns," in *The 22nd international conference on artificial intelligence and statistics*. PMLR, 2019, pp. 1901–1910.

[10] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, 2021.

[11] A. Alaa and M. Van Der Schaar, "Frequentist uncertainty in recurrent neural networks via blockwise influence functions," in *International Conference on Machine Learning*. PMLR, 2020, pp. 175–190.

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

[13] K. Rasul, A.-S. Sheikh, I. Schuster, U. Bergmann, and R. Vollgraf, "Multivariate probabilistic time series forecasting via conditioned normalizing flows," *arXiv preprint arXiv:2002.06103*, 2020.

[14] K. Rasul, C. Seward, I. Schuster, and R. Vollgraf, "Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting," *arXiv preprint arXiv:2101.12072*, 2021.

[15] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," *Advances in neural information processing systems*, vol. 28, pp. 2980–2988, 2015.

[16] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther, "Sequential neural models with stochastic layers," *Advances in Neural Information Processing Systems*, vol. 29, pp. 2199–2207, 2016.

[17] A. Goyal, A. Sordoni, M.-A. Côté, N. R. Ke, and Y. Bengio, "Z-forcing: training stochastic recurrent networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6716–6726.

[18] R. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[19] S. Pal, L. Ma, Y. Zhang, and M. Coates, "Rnn with particle flow for probabilistic spatio-temporal forecasting," *arXiv preprint arXiv:2106.06064*, 2021.

[20] C. Qiu, S. Mandt, and M. Rudolph, "History marginalization improves forecasting in variational recurrent neural networks," *Entropy*, vol. 23, no. 12, p. 1563, 2021.

[21] D. Eigen, M. Ranzato, and I. Sutskever, "Learning factored representations in a deep mixture of experts," *arXiv preprint arXiv:1312.4314*, 2013.

[22] Y. Shi, B. Paige, P. Torr *et al.*, "Variational mixture-of-experts autoencoders for multi-modal deep generative models," *Advances in Neural Information Processing Systems*, vol. 32, pp. 15 718–15 729, 2019.

[23] R. Kurle, S. Günnemann, and P. van der Smagt, "Multi-source neural variational inference," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4114–4121.

[24] T. Shen, M. Ott, M. Auli, and M. Ranzato, "Mixture models for diverse machine translation: Tricks of the trade," in *International conference on machine learning*. PMLR, 2019, pp. 5719–5728.

[25] J. Cho, M. Seo, and H. Hajishirzi, "Mixture content selection for diverse sequence generation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3121–3131.

[26] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat *et al.*, "Glam: Efficient scaling of language models with mixture-of-experts," *arXiv preprint arXiv:2112.06905*, 2021.

[27] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[28] A. Varamesh and T. Tuytelaars, "Mixture dense regression for object detection and human pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 086–13 095.

[29] M. J. Vowels, N. C. Camgoz, and R. Bowden, "Vdsm: Unsupervised video disentanglement with state-space modeling and deep mixtures of experts," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8176–8186.

[30] P. Jain and P. Kar, "Non-convex optimization for machine learning," *Foundations and Trends® in Machine Learning*, vol. 10, no. 3-4, pp. 142–336, 2017.

[31] S. Balakrishnan, M. J. Wainwright, and B. Yu, "Statistical guarantees for the em algorithm: From population to sample-based analysis," *The Annals of Statistics*, vol. 45, no. 1, pp. 77–120, 2017.

[32] J. Chen, J. Zhu, Y. W. Teh, and T. Zhang, "Stochastic expectation maximization with variance reduction." in *NeurIPS*, 2018, pp. 7978–7988.

[33] D. J. MacKay, "Information theory, inference & learning algorithms," 2003.

[34] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty years of mixture of experts," *IEEE transactions on neural networks and learning systems*, vol. 23, pp. 1177–1193, 2012.

[35] A. Doerr, C. Daniel, M. Schiegg, N.-T. Duy, S. Schaal, M. Toussaint, and T. Sebastian, "Probabilistic recurrent state-space models," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1280–1289.

[36] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," *Advances in Neural Information Processing Systems*, vol. 32, pp. 4650–4661, 2019.

[37] J. Bayer and C. Osendorfer, "Learning stochastic recurrent networks," in *NIPS 2014 Workshop on Advances in Variational Inference*, 2014.

[38] M. Karl, M. Soelch, J. Bayer, and P. Van der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," *arXiv preprint arXiv:1605.06432*, 2016.

[39] Y. Li and S. Mandt, "Disentangled sequential autoencoder," *arXiv preprint arXiv:1803.02991*, 2018.

[40] A. C. Cohen and B. J. Whitten, "Estimation in the three-parameter lognormal distribution," *Journal of the American Statistical Association*, vol. 75, no. 370, pp. 399–404, 1980.

[41] T. Pearce, A. Brintrup, M. Zaki, and A. Neely, "High-quality prediction intervals for deep learning: A distribution-free, ensembled approach," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4075–4084.

[42] X. Qiu, E. Meyerson, and R. Miikkulainen, "Quantifying point-prediction uncertainty in neural networks via residual estimation with an i/o kernel," in *International Conference on Learning Representations*, 2020.

[43] E. L. Crow and K. Shimizu, *Lognormal distributions*. Marcel Dekker New York, 1987.

[44] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[45] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.

[46] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[47] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with autocorrelation for long-term series forecasting," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[49] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[50] S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. X. Chen, "Cautionary tales on air-quality improvement in beijing," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2205, p. 20170457, 2017.

[51] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *International conference on machine learning*. PMLR, 2013, pp. 115–123.

Appendices

A Model Specification

A.1 Target Distributions

We show the flexibility of specifying the prediction modules of the mixture models through two different target distributions.

For the target variable in normal distribution, i.e., $y_t \in \mathbb{R}$, the prediction module of each data source is expressed as:

$$p_{\omega_s}(y_t \mid z_t = s, \mathbf{h}_s) = \frac{1}{\sqrt{2\pi}\sigma_{s,t}} \exp\left(-\frac{(y_t - \mu_{s,t})^2}{2\sigma_{s,t}^2}\right) \tag{17}$$

$\mu_{s,t}$ and $\sigma_{s,t}^2$ are derived by $g_{w_s}(\mathbf{h}_{s,t})$, where $g_{w_s}(\cdot)$ can be any flexible function such as dense layers parameterized by ω_s [7]. The corresponding predictive mean and variance are $\mathbb{E}[y_t \mid z_t = s, \widetilde{\mathbf{h}}_s] = \mu_{s,t}$ and $\text{Var}(y_t | z_t = s, \widetilde{\mathbf{h}}_{s,t}) = \sigma_{s,t}^2$.

When $y_t \in \mathbb{R}^+$ and is assumed to follow the log-normal distribution, the density function is:

$$p_{\omega_s}(y_t \mid z_t = s, \mathbf{h}_s) = \frac{1}{\sqrt{2\pi} y_t \sigma_{s,t}} \exp\left(-\frac{(\ln y_t - \mu_{s,t})^2}{2\sigma_{s,t}^2}\right) \tag{18}$$

The predictive mean and variance by the data source s are $\mathbb{E}[y_t \mid z_t = s, \widetilde{\mathbf{h}}_s] = \exp(\mu_{s,t} + \frac{1}{2}\sigma_{s,t}^2)$ and $\text{Var}(y_t | z_t = s, \widetilde{\mathbf{h}}_{s,t}) = \exp(\sigma_{s,t}^2 - 1) \cdot \exp(2\mu_{s,t} + \sigma_{s,t}^2)$ [43, 33].

A.2 Inference

In this part, we present several predictive quantities that can be derived from the mixture model.

Point Prediction. It is the predictive mean defined as:

$$\hat{y}_t \triangleq \mathbb{E}[y_t \mid \mathbf{x}_{1:S,<t}] = \int \sum_{s=1}^{S} \mathbb{P}_\theta(z_t = s \mid \cdot) p_{\omega_s}(y_t | z_t = s, \widetilde{\mathbf{h}}_s)\, y_t \, \mathrm{d}y_t \tag{19}$$

$$= \sum_{s=1}^{S} \mathbb{P}_\theta(z_t = s \mid \cdot) \mathbb{E}[y_t | z_t = s, \widetilde{\mathbf{h}}_{s,t}] \tag{20}$$

, where $\mathbb{E}[y_t \mid z_t = s, \widetilde{\mathbf{h}}_s]$ denotes the predictive mean by source s. As aforementioned, it is $\mu_{s,t}$ for the normal distribution and $\exp(\mu_{s,t} + \frac{1}{2}\sigma_{s,t}^2)$ for the log-normal distribution [33].

Uncertainty Score. According to the variance definition, we have:

$$\hat{u}_t \triangleq \text{Var}(\, y_t \mid \mathbf{x}_{1:S,<t}) = \mathbb{E}[y_t^2 \mid \mathbf{x}_{1:S,<t}] - \mathbb{E}^2[y_t \mid \mathbf{x}_{1:S,<t}] \tag{21}$$

Plugging the following into Eq.21 gives rise to Eq.5

$$\mathbb{E}[y_t^2 \mid \mathbf{x}_{1:S,<t}] = \sum_{s=1}^{S} \mathbb{P}_\theta(z_t = s \mid \cdot) \mathbb{E}[y_t^2 | z_t = s, \widetilde{\mathbf{h}}_s] \tag{22}$$

$$\mathbb{E}[y_t^2 | z_t = s, \widetilde{\mathbf{h}}_s] = \text{Var}(y_t | z_t = s, \widetilde{\mathbf{h}}_{s,t}) + \mathbb{E}^2[y_t | z_t = s, \widetilde{\mathbf{h}}_s] \tag{23}$$

, where the variance term $\text{Var}(y_t | z_t = s, \widetilde{\mathbf{h}}_{s,t})$ is $\sigma_{s,t}^2$ for the normal distribution, and $\exp(\sigma_{s,t}^2 - 1) \cdot \exp(2\gamma_{s,t} + \sigma_{s,t}^2)$ for the log-normal distribution [43].

Quantile Prediction. Based on the mixture model, the predictive cumulative probability function (CDF) of the target is expressed as:

$$
\begin{aligned}
P(y_t \,|\, \mathbf{x}_{1:S,<t}) &= \int p(\,\mathbf{h}_{1:S,t} \,|\, \mathbf{x}_{1:S,<t}) \cdot \sum_{s=1}^{S} \mathbb{P}_\theta(z_t = s \,|\, \mathbf{h}_{1:S,t}) \cdot P_{\omega_s}(y_t \,|\, z_t = s, \mathbf{h}_{s,t})\, \mathrm{d}\mathbf{h}_{1:S,t} \\
&= \sum_{s=1}^{S} \mathbb{P}_\theta(z_t = s \,|\, \widetilde{\mathbf{h}}_{1:S,t}) \cdot P_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t})
\end{aligned}
\tag{24}
$$

where $\widetilde{\mathbf{h}}_{s,t} = \mathrm{NN}_{\eta_s}(\mathbf{x}_{s,<t})$, and $P(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t})$ represents the individual data source's CDF. The second equality is due to the delta distribution applied to $p(\,\mathbf{h}_{1:S,t} \,|\, \mathbf{x}_{1:S,<t})$, as is presented in Sec.3.2.

Then, for the quantile level $\alpha \in [0, 1]$, we resort to a numerical method by applying the root-finding algorithm to the function in Eq. 25, while sampling based method can also applied. The solution of the root-finding algorithm is the quantile prediction $\hat{y}_{t,\alpha}$.

$$
Q\big(y\,; \alpha, P(\cdot \,|\, \mathbf{x}_{1:S,<t})\big) \triangleq P(y \,|\, \mathbf{x}_{1:S,<t}) - \alpha \tag{25}
$$

Prediction Interval. For obtaining a prediction interval corresponding to a probability range, we make use of the corresponding quantile predictions of the probability range. For instance, a prediction interval corresponding to the probability range $[0.1, 0.9]$ is constructed by the root solutions of $Q\big(y\,; \alpha = 0.1, P(\cdot \,|\, \mathbf{x}_{1:S,<t})\big) = 0$ and $Q\big(y\,; \alpha = 0.9, P(\cdot \,|\, \mathbf{x}_{1:S,<t})\big) = 0$.

A.3 Learning

The loss function of the mixture model is:

$$
\mathcal{L}\,(\Theta\,; \mathcal{D}) \triangleq \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} -\log \mathbb{E}_{\mathbf{h}_{1:S,t}} \left[\sum_{s=1}^{S} \mathbb{P}_\theta(z_t = s \,|\, \mathbf{h}_{1:S,t})\, p_{\omega_s}(y_t \,|\, z_t = s, \mathbf{h}_{s,t}) \right] \tag{26}
$$

where $\Theta = \big\{\{\eta_s, \omega_s\}_{s=1}^{S}, \theta\big\}$ denotes the set of all trainable parameters.

Since in this paper the delta distribution is applied to $p(\,\mathbf{h}_{1:S,t} \,|\, \mathbf{x}_{1:S,<t})$ and $p(\,\mathbf{r}_{1:S,t} \,|\, \mathbf{x}_{1:S,<t})$, the integral over the hidden states is reduced to single points, i.e., $\widetilde{\mathbf{h}}_{s,t} = \mathrm{NN}_{\eta_s}(\mathbf{x}_{s,<t})$.

B Proofs

Lemma 4.1. *In minimizing the mixture loss via SGD optimization, given a random training sample indexed by t, the vectorized derivative of a source's prediction and representation modules are denoted by:*

$$
\mathbf{g}_{\omega_s,t} \triangleq -\frac{\partial \log p_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t})}{\partial \omega_s}\Big|_{\Theta} \tag{27}
$$

$$
\mathbf{g}_{\eta_s,t} \triangleq -\frac{\partial \log p_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t})}{\partial \widetilde{\mathbf{h}}_{s,t}} \nabla \widetilde{\mathbf{h}}_{s,t}\Big|_{\Theta} \tag{28}
$$

$$
\mathbf{d}_{\eta_s,t} \triangleq -\sum_{k=1}^{S} \pi_{k,t} \frac{\partial \log \mathbb{P}_\theta(z_t = k|\cdot)}{\partial \widetilde{\mathbf{h}}_{s,t}} \nabla \widetilde{\mathbf{h}}_{s,t}\Big|_{\Theta} \tag{29}
$$

where $\nabla \widetilde{\mathbf{h}}_{s,t}$ denotes the Jacobian matrix of source s's representation w.r.t. η_s. Meanwhile, the posterior weight $\pi_{s,t}$ is defined as:

$$
\begin{aligned}
\pi_{s,t} &\triangleq p(z_t = s \,|\, y_t, \mathbf{x}_{1:S,<t}; \Theta) \\
&= \frac{p_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t})\, \mathbb{P}_\theta(z_t = s \,|\, \cdot)}{\sum_{k=1}^{S} p_{\omega_k}(y_t \,|\, z_t = k, \widetilde{\mathbf{h}}_{k,t})\, \mathbb{P}_\theta(z_t = k \,|\, \cdot)}
\end{aligned}
\tag{30}
$$

Then, the (stochastic) updates to the prediction and representation parameters of source s, i.e., ω_s and η_s, are:

$$
\pi_{s,t}\, \mathbf{g}_{\omega_s,t} \;\; \textit{and} \;\; \pi_{s,t}\, \mathbf{g}_{\eta_s,t} + \mathbf{d}_{\eta_s,t} \tag{31}
$$

Proof. Given a random training sample $(y_t, \mathbf{x}_{1:S,<t})$, the mixture loss Eq.26 is expressed:

$$-\log \sum_{k=1}^{S} \mathbb{P}_\theta(z_t = k|\widetilde{\mathbf{h}}_{1:S,t})\, p_{\omega_k}(y_t|z_t = k, \widetilde{\mathbf{h}}_{k,t}) \text{ where } \widetilde{\mathbf{h}}_{k,t} = \text{NN}_{\eta_k}(\mathbf{x}_{k,<t})$$

Take the derivative w.r.t. the prediction module parameter of source s,

$$\begin{aligned}
&\frac{\partial -\log \sum_{k=1}^{S} \mathbb{P}_\theta(z_t = k|\widetilde{\mathbf{h}}_{1:S,t})\, p_{\omega_k}(y_t|z_t = k, \widetilde{\mathbf{h}}_{k,t})}{\partial \omega_s}\Big|_\Theta \\
&= -\frac{\mathbb{P}_\theta(z_t = s|\widetilde{\mathbf{h}}_{1:S,t})}{\sum_{k=1}^{S} \mathbb{P}_\theta(z_t = k|\widetilde{\mathbf{h}}_{1:S,t})\, p_{\omega_k}(y_t|z_t = k, \widetilde{\mathbf{h}}_{k,t})} \frac{\partial p_{\omega_s}(y_t|z_t = s, \widetilde{\mathbf{h}}_{s,t})}{\partial \omega_s}\Big|_\Theta \\
&= -\underbrace{\frac{\mathbb{P}_\theta(z_t = s|\widetilde{\mathbf{h}}_{1:S,t}) p_{\omega_s}(y_t|z_t = s, \widetilde{\mathbf{h}}_{s,t})}{\sum_{k=1}^{S} \mathbb{P}_\theta(z_t = k|\widetilde{\mathbf{h}}_{1:S,t})\, p_{\omega_k}(y_t|z_t = k, \widetilde{\mathbf{h}}_{k,t})}}_{\pi_{s,t}} \frac{\partial \log p_{\omega_s}(y_t|z_t = s, \widetilde{\mathbf{h}}_{s,t})}{\partial \omega_s}\Big|_\Theta
\end{aligned} \tag{32}$$

In Eq.32, the second equality is through the derivative trick of log functions, i.e., $\partial f(\omega)/\partial\omega = f(\omega)\, \partial \log f(\omega)/\partial\omega$.

Likewise, for η_s, we have the following:

$$\begin{aligned}
&\frac{\partial -\log \sum_{k=1}^{S} \mathbb{P}_\theta(z_t = k|\widetilde{\mathbf{h}}_{1:S,t})\, p_{\omega_k}(y_t|z_t = k, \widetilde{\mathbf{h}}_{k,t})}{\partial \eta_s}\Big|_\Theta \\
&= -\frac{1}{\sum_{k=1}^{S} \mathbb{P}_\theta(z_t = k|\widetilde{\mathbf{h}}_{1:S,t})\, p_{\omega_k}(y_t|z_t = k, \widetilde{\mathbf{h}}_{k,t})} \sum_{k=1}^{S} \frac{\partial\, \mathbb{P}_\theta(z_t = k|\widetilde{\mathbf{h}}_{1:S,t}) p_{\omega_k}(y_t|z_t = k, \widetilde{\mathbf{h}}_{k,t})}{\partial \eta_s}\Big|_\Theta \\
&= -\sum_{k=1}^{S} \pi_{k,t} \frac{\partial \log \mathbb{P}_\theta(z_t = k|\widetilde{\mathbf{h}}_{1:S,t}) p_{\omega_k}(y_t|z_t = k, \widetilde{\mathbf{h}}_{k,t})}{\partial \eta_s}\Big|_\Theta \\
&= -\sum_{k=1}^{S} \pi_{k,t} \frac{\partial \log \mathbb{P}_\theta(z_t = k|\widetilde{\mathbf{h}}_{1:S,t})}{\partial \eta_s} - \pi_{s,t} \frac{\partial \log p_{\omega_s}(y_t|z_t = s, \widetilde{\mathbf{h}}_{s,t})}{\partial \eta_s}\Big|_\Theta
\end{aligned} \tag{33}$$

In Eq.33, the second equality is based on the derivative trick of log functions. Because the weight module $\mathbb{P}_\theta(z_t = k|\widetilde{\mathbf{h}}_{1:S,t})$ takes as input the representations of all data sources, there exists the sum of derivatives in the first term in the last equality. Meanwhile, since $\widetilde{\mathbf{h}}_{1:S,t}$ is the output of representation modules parameterized by $\{\eta_s\}_{s=1}^{S}$, the derivative w.r.t. η_s is through $\widetilde{\mathbf{h}}_{s,t}$. Then, the update scheme in Eq. 31 is obtained.

■

Lemma 4.2. *In the iterative minimization process of the mixture loss, at any point of Θ, there exists the following upper bound:*

$$\mathcal{L}(\Theta\,; D_t) \leq \overline{\mathcal{L}}\big(\{\eta_s, \omega_s\}_{s=1}^{S}\,; D_t\big) \tag{34}$$

where D_t is a random training sample $D_t \triangleq (y_t, \mathbf{x}_{1:S,<t}) \in \mathcal{D}$.

$$\overline{\mathcal{L}}\big(\{\eta_s, \omega_s\}_{s=1}^{S}\,; D_t\big) \triangleq -\sum_{s=1}^{S} \frac{1}{S} \log p_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t}) - \log S a_t^* \tag{35}$$

and a_t^* is a constant value, $a_t^* = \min_s(\{\, \mathbb{P}(z_t = s \,|\, \cdot)\,\})$.

Proof. The mixture loss on a random training sample D_t is expressed:

$$\mathcal{L}(\Theta\,; D_t) \triangleq -\log \sum_{s=1}^{S} \mathbb{P}_\theta(z_t = s \,|\, \widetilde{\mathbf{h}}_{1:S,t})\, p_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t}) \tag{36}$$

where $\widetilde{\mathbf{h}}_{s,t} = \mathrm{NN}_{\eta_s}(\mathbf{x}_{s,<t})$.

Given Θ, by defining $a_t^* \triangleq \min_s \left(\{ \mathbb{P}(z_t = s \,|\, \cdot)\}_{s=1}^{S} \right)$, we have

$$\mathcal{L}\left(\Theta\,; D_t\right) \leq -\log \sum_{s=1}^{S} a_t^* \, p_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t}) \tag{37}$$

Eq. 37 holds, because the probability densities are non-negative. The equality holds when all weights are equal.

Then, Eq. 37 is further expressed as follows:

$$= -\log \sum_{s=1}^{S} \frac{a_t^*}{S a_t^*} \, p_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t}) - \log S a_t^* \tag{38}$$

$$\leq -\sum_{s=1}^{S} \frac{1}{S} \log p_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t}) - \log S a_t^* \tag{39}$$

S is the number of data sources. The last inequality is based on Jensen's inequality. ■

Theorem 4.3. *For all stochastic derivatives $\mathbf{g} \in \{\mathbf{g}_{\omega_s}, \mathbf{g}_{\eta_s}, \mathbf{d}_{\eta_s}\}$ corresponding to the data source s, suppose it is bounded $\mathbb{E}\|\mathbf{g}\|^2 \leq G^2$. The trainable parameters related to a data source's prediction are denoted by $\Theta_s = [\omega_s, \eta_s]^\top$ and the corresponding gradient is $\mathcal{G}_s = \mathbb{E}[\mathbf{g}_{\omega_s}, \mathbf{g}_{\eta_s}]^\top$. Over the learning step $i = 0, \cdots, I$, the convergence of $\mathcal{G}_s$ to a stationary point in the direct optimization and the impartial-phase optimization are respectively:*

$$\textit{Direct:}\ \frac{1}{I}\sum_{i=0}^{I-1} \mathbb{E}\|\mathcal{G}_{s,i}\|^2 \leq \sqrt{10} G \sqrt{\frac{L\left(\mathcal{L}_{s,0} - \mathcal{L}_{s,I}\right)}{I \pi_s^*}} + \sum_{i=0}^{I-1} \frac{1}{2I} \left(\|\mathbb{E}[\mathbf{g}_{\eta_s,i}] - \mathbb{E}[\mathbf{d}_{\eta_s,i}]\|^2 \right) \tag{40}$$

$$\textit{Impartial phase:}\ \frac{1}{I}\sum_{i=0}^{I-1} \mathbb{E}\|\mathcal{G}_{s,i}\|^2 \leq 2G \sqrt{\frac{L\left(\mathcal{L}_{s,0} - \mathcal{L}_{s,I}\right)}{I}} \tag{41}$$

, where $\pi_s^* \in (0, 1)$ is the minimum of poster weights across data samples and iteration steps. $\mathcal{L}_{s,i}$ represents the negative log-likelihood of the predictions by data source s at step i.

Proof. Before going to the proof details, we first recall some notations.

As defined in Lemma 4.1., we have the (stochastic) derivatives on a random training sample t as follows:

$$\begin{aligned}
\mathbf{g}_{\omega_s} &\triangleq -\frac{\partial \log p_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t})}{\partial \omega_s}\Big|_{\Theta_i} \\
\mathbf{g}_{\eta_s} &\triangleq -\frac{\partial \log p_{\omega_s}(y_t \,|\, z_t = s, \widetilde{\mathbf{h}}_{s,t})}{\partial \widetilde{\mathbf{h}}_{s,t}} \nabla \widetilde{\mathbf{h}}_{s,t}\Big|_{\Theta_i} \\
\mathbf{d}_{\eta_s} &\triangleq -\sum_{k=1}^{S} \pi_{k,i} \frac{\partial \log \mathbb{P}_\theta(z_t = k|\cdot)}{\partial \widetilde{\mathbf{h}}_{s,t}} \nabla \widetilde{\mathbf{h}}_{s,t}\Big|_{\Theta_i}
\end{aligned}$$

where $\nabla \widetilde{\mathbf{h}}_{s,t}$ denotes the Jacobian matrix of source s's representation module w.r.t. η_s.

Given the parameter $\Theta_{s,i}$ at step i, the predictive performance of data source s is reflected in the negative log-likelihood which is a function of $\Theta_{s,i}$ as:

$$\mathcal{L}_{s,i} \triangleq \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} -\log \mathbb{E}_{\mathbf{h}_{s,t}} \left[p_{\omega_s}(y_t \,|\, z_t = s, \mathbf{h}_{s,t}) \right] \tag{42}$$

Then, the empirical expected gradient of $\Theta_{s,i}$ w.r.t. $\mathcal{L}_{s,i}$ is as follows: (the step index i is omitted for simplicity)

$$\mathcal{G}_s \triangleq \mathbb{E}[\mathbf{g}_{\omega_s}, \mathbf{g}_{\eta_s}]^\top = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}[\mathbf{g}_{\omega_s,t}, \mathbf{g}_{\eta_s,t}]^\top \tag{43}$$

Direct optimization

In the direct optimization, as shown in Lemma 4.1., the stochastic update to the prediction and representation parameters of a data sources s is expressed as: $\widehat{\mathbf{g}}_{s,i} \triangleq [\pi_{s,i}\,\mathbf{g}_{\omega_s,i},\ \pi_{s,i}\,\mathbf{g}_{\eta_s,i} + \mathbf{d}_{\eta_s,i}]^\top$, where $\pi_{s,i}$ is the posterior weight. In the following, we will drop the random training sample subscript t for simplicity when unnecessary.

Following the L-smoothness assumption of $\mathcal{L}_s$ yields:

$$\mathcal{L}_{s,i+1} - \mathcal{L}_{s,i} \leq \langle \mathcal{G}_{s,i}, -\gamma\mathbb{E}[\widehat{\mathbf{g}}_{s,i}]\rangle + \frac{L}{2}\mathbb{E}\left[\langle -\gamma\widehat{\mathbf{g}}_{s,i}, -\gamma\widehat{\mathbf{g}}_{s,i}\rangle\right] \tag{44}$$

$$= -\gamma\left\langle \mathbb{E}\begin{bmatrix}\mathbf{g}_{\omega_{s,i}}\\ \mathbf{g}_{\eta_{s,i}}\end{bmatrix}, \mathbb{E}\begin{bmatrix}\pi_{s,i}\,\mathbf{g}_{\omega_{s,i}}\\ \pi_{s,i}\,\mathbf{g}_{\eta_{s,i}}\end{bmatrix}\right\rangle - \gamma\left\langle \mathbb{E}[\mathbf{g}_{\eta_{s,i}}], \mathbb{E}[\mathbf{d}_{\eta_{s,i}}]\right\rangle + \frac{L}{2}\mathbb{E}\left[\langle -\gamma\widehat{\mathbf{g}}_{s,i}, -\gamma\widehat{\mathbf{g}}_{s,i}\rangle\right] \tag{45}$$

$$\leq -\gamma\,\pi^*_{s,i}\left\|\mathbb{E}\begin{bmatrix}\mathbf{g}_{\omega_{s,i}}\\ \mathbf{g}_{\eta_{s,i}}\end{bmatrix}\right\|^2 \underbrace{-\gamma\left\langle \mathbb{E}[\mathbf{g}_{\eta_{s,i}}], \mathbb{E}[\mathbf{d}_{\eta_{s,i}}]\right\rangle}_{(a)} + \underbrace{\frac{L}{2}\mathbb{E}\left[\langle -\gamma\widehat{\mathbf{g}}_{s,i}, -\gamma\widehat{\mathbf{g}}_{s,i}\rangle\right]}_{(b)} \tag{46}$$

where γ is the learning step size. $\pi^*_{s,i}$ is the minimum of posterior weights across data samples, i.e., $\pi^*_{s,i} = \min_t(\{\pi_{s,t,i}\}_t)$, and it is always less than 1, i.e., $\pi^*_{s,i} \in (0,1)$. $\langle\cdot,\cdot\rangle$ is the inner product of two vectors.

Then, for the (a) and (b) terms, we have the following:

$$(a) = -\frac{\gamma}{2}\left\|\mathbb{E}[\mathbf{g}_{\eta_s,i}]\right\|^2 - \frac{\gamma}{2}\left\|\mathbb{E}[\mathbf{d}_{\eta_s,i}]\right\|^2 + \frac{\gamma}{2}\left\|\mathbb{E}[\mathbf{g}_{\eta_s,i}] - \mathbb{E}[\mathbf{d}_{\eta_s,i}]\right\|^2 \tag{47}$$

$$\leq \frac{\gamma}{2}\left\|\mathbb{E}[\mathbf{g}_{\eta_s,i}] - \mathbb{E}[\mathbf{d}_{\eta_s,i}]\right\|^2 \tag{48}$$

$$(b) = \frac{L\gamma^2}{2}\mathbb{E}\left\|\begin{bmatrix}\pi_{s,i}\,\mathbf{g}_{\omega_{s,i}}\\ \pi_{s,i}\,\mathbf{g}_{\eta_{s,i}} + \mathbf{d}_{\eta_{s,i}}\end{bmatrix}\right\|^2 \tag{49}$$

$$= \frac{L\gamma^2}{2}\mathbb{E}\left[\pi^2_{s,i}\|\mathbf{g}_{\omega_{s,i}}\|^2 + \pi^2_{s,i}\|\mathbf{g}_{\omega_{s,i}}\|^2 + 2\pi_{s,i}\langle\mathbf{g}_{\eta_{s,i}}, \mathbf{d}_{\eta_{s,i}}\rangle + \|\mathbf{d}_{\eta_{s,i}}\|^2\right] \tag{50}$$

$$\leq \frac{L\gamma^2}{2}\mathbb{E}\left[\pi^2_{s,i}\|\mathbf{g}_{\omega_{s,i}}\|^2 + \pi^2_{s,i}\|\mathbf{g}_{\omega_{s,i}}\|^2 + \pi_{s,i}\|\mathbf{g}_{\eta_{s,i}}\|^2 + \pi_{s,i}\|\mathbf{d}_{\eta_{s,i}}\|^2 + \|\mathbf{d}_{\eta_{s,i}}\|^2\right] \tag{51}$$

$$\leq \frac{5}{2}L\gamma^2G^2 \tag{52}$$

The first inequality is because $2\langle\mathbf{a},\mathbf{b}\rangle \leq \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2$ and the last one is due to the gradient bound assumption.

Plugging Eq.48 and 52 into Eq.46 and summing the inequality from $i=0$ to $i=I$ lead to:

$$\mathcal{L}_{s,i+1} - \mathcal{L}_{s,i} \leq -\gamma\sum_{i=0}^{I-1}\pi^*_i\left\|\mathcal{G}_{s,i}\right\|^2 + \sum_{i=0}^{I-1}\frac{\gamma}{2}\left\|\mathbb{E}[\mathbf{g}_{\eta_s,i}] - \mathbb{E}[\mathbf{d}_{\eta_s,i}]\right\|^2 + \frac{5}{2}L\gamma^2G^2I \tag{53}$$

If we further take the minimum of poster weights across iteration steps by setting $\pi^*_s = \min_i(\{\pi^*_{s,i}\}_i)$ or equivalently $\pi^*_s = \min_{t,i}(\{\pi_{s,t,i}\}_{t,i})$, rearranging Eq. 53 gives:

$$\frac{1}{I}\sum_{i=0}^{I-1}\mathbb{E}\|\mathcal{G}_{s,i}\|^2 \leq \frac{(\mathcal{L}_{s,0} - \mathcal{L}_{s,I})}{\gamma I\pi^*_s} + \sum_{i=0}^{I-1}\frac{1}{2}\left(\|\mathbb{E}[\mathbf{g}_{\eta_s,i}] - \mathbb{E}[\mathbf{d}_{\eta_s,i}]\|^2\right) + \frac{5}{2}L\gamma G^2 \tag{54}$$

Plugging in the step size γ that minimizes the RHS of Eq. 54 shows the result of the direct optimization:

$$\frac{1}{I}\sum_{i=0}^{I-1}\mathbb{E}\|\mathcal{G}_{s,i}\|^2 \leq \sqrt{10}G\sqrt{\frac{L\left(\mathcal{L}_{s,0}-\mathcal{L}_{s,I}\right)}{I\pi_s^*}} + \sum_{i=0}^{I-1}\frac{1}{2I}\left(\|\mathbb{E}[\mathbf{g}_{\eta_s,i}]-\mathbb{E}[\mathbf{d}_{\eta_s,i}]\|^2\right) \tag{55}$$

Impartial phase optimization

In the impartial phase, the stochastic update to the prediction and representation parameters of a data source s is expressed as: $\mathring{\mathbf{g}}_{s,i} \triangleq [\frac{1}{S}\mathbf{g}_{\omega_s,i},\ \frac{1}{S}\mathbf{g}_{\eta_s,i}]^\top$.

Following the L-smoothness assumption yields:

$$\mathcal{L}_{s,i+1}-\mathcal{L}_{s,i} \leq \langle \mathcal{G}_{s,i}\,, -\gamma\mathbb{E}[\mathring{\mathbf{g}}_{s,i}]\rangle + \frac{L}{2}\mathbb{E}\left[\langle -\gamma\mathring{\mathbf{g}}_{s,i}\,, -\gamma\mathring{\mathbf{g}}_{s,i}\rangle\right] \tag{56}$$

$$= -\frac{\gamma}{S}\|\mathcal{G}_{s,i}\|^2 + \frac{L\gamma^2}{2S^2}\mathbb{E}\left[\|\mathbf{g}_{\omega_s,i}\|^2 + \|\mathbf{g}_{\eta_s,i}\|^2\right] \tag{57}$$

$$\leq -\frac{\gamma}{S}\|\mathcal{G}_{s,i}\|^2 + \frac{L\gamma^2G^2}{S^2} \tag{58}$$

Then, by summing the inequality Eq. 58 from $i=0$ to $i=I$ and some rearrangements, we obtain

$$\frac{1}{I}\sum_{i=0}^{I-1}\mathbb{E}\|\mathcal{G}_{s,i}\|^2 \leq \frac{S\left(\mathcal{L}_{s,0}-\mathcal{L}_{s,I}\right)}{\gamma I} + \frac{L\gamma G^2}{S} \tag{59}$$

Plugging in the step size γ that minimizes the RHS of Eq. 59 gives the convergence:

$$\frac{1}{I}\sum_{i=0}^{I-1}\mathbb{E}\|\mathcal{G}_{s,i}\|^2 \leq 2G\sqrt{\frac{L\left(\mathcal{L}_{s,0}-\mathcal{L}_{s,I}\right)}{I}} \tag{60}$$

■

C Additional Experiment Details and Results

Hyper-Parameters. The hyper-parameter search and associated training processes are run on a server with NVIDIA A100 GPUs. For all models, Bayesian optimization is applied to search in the hyper-parameter space [51]. Once the best hyper-parameters is fixed, each model is retrained for five times with different random seeds, and the average performance is reported. For recurrent neural networks used in DAR, DF and mixture models, the layer size and number of layers are searched over $\{64, 128, 256, 512, 768, 1024\}$ and $\{1, 2, 3\}$. The learning rate is chosen in the range $\{1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3\}$ following an exponential decay scheme with a rate of 0.85 every 10 epochs. The search spaces of batch sizes, weight decay and dropout rates respectively lie in $[256, 512]$, $\{1e-7, 5e-7, 1e-6, 5e-6, 1e-5, 5e-5, 1e-4, \}$, and $\{0.0, 0.1, 0.2, 0.3\}$

Performance Metrics. Given the point prediction $\hat{y}_t$ and the true value y_t, the point prediction metrics are defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{T}_{\text{test}}|}\sum_{t\in\mathcal{T}_{\text{test}}}(y_t-\hat{y}_t)^2} \ \text{ and } \ \text{MAE} = \frac{1}{|\mathcal{T}_{\text{test}}|}\sum_{t\in\mathcal{T}_{\text{test}}}|y_t-\hat{y}_t| \tag{61}$$

where $\mathcal{T}_{\text{test}}$ represents the set of timesteps in the testing data, and $|\mathcal{T}_{\text{test}}|$ is the number of testing examples.

As for the mean quantile loss (QLm), we define the (normalized) quantile loss of a given quantile level $\alpha \in (0, 1)$ as:

$$\text{QL}(\alpha) = \frac{\sum_{t\in\mathcal{T}_{\text{test}}} 2\left[\alpha(y_t-\hat{y}_{t,\alpha})\,\mathbb{I}_{y_t-\hat{y}_{t,\alpha}>0} + (1-\alpha)(\hat{y}_{t,\alpha}-y_t)\,\mathbb{I}_{y_t-\hat{y}_{t,\alpha}\leq 0}\right]}{\sum_{t\in\mathcal{T}_{\text{test}}}|y_t|} \tag{62}$$

where $\mathbb{I}$ takes value 1 only when the corresponding condition holds [8]. $\hat{y}_{t,\alpha}$ is the quantile prediction obtained through the method presented in Sec. A.2.

Then, the QLm is the mean value of the quantile losses $\text{QL}(\alpha)$ for $\alpha = 0.1, 0.3, 0.5, 0.7, 0.9$.

Results. In this part, we report the performance result on all datasets.

Table 3 and 4 show the point and probabilistic prediction performance on all datasets. Our mixture model **MIX** shows competitive performance over baselines.

Table 3: Results of point prediction metrics (mean and standard error respectively in the first and second line of each row). The best result is marked by the grey box.

	RMSE ↓				MAE ↓			
Dataset	DAR	DF	AF	**MIX**	DAR	DF	AF	**MIX**
Site 0	23.650	23.792	68.293	19.022	12.015	13.743	35.196	10.297
	0.645	0.756	0.958	0.054	0.223	0.291	0.307	0.048
Site 1	20.012	31.467	66.199	18.517	10.596	13.247	36.188	9.929
	0.205	0.570	0.912	0.086	0.109	0.772	0.462	0.027
Site 2	18.060	19.801	69.008	17.081	9.054	11.007	53.221	8.573
	0.210	0.277	0.323	0.087	0.028	0.389	0.862	0.037
Site 3	29.134	40.351	95.691	21.969	11.352	14.858	70.882	11.293
	0.147	0.574	0.823	0.101	0.108	0.229	0.677	0.059
Site 4	22.393	22.594	87.732	19.637	10.883	14.151	66.508	10.638
	0.278	0.292	0.932	0.062	0.013	0.505	0.709	0.084
Site 5	26.133	29.102	93.721	20.578	11.341	11.234	66.004	10.318
	1.392	1.101	1.236	0.128	0.106	0.102	0.979	0.049
Exc. A	1.206	1.157	1.892	1.149	0.925	0.916	1.021	0.898
	0.00342	0.00389	0.00253	0.00168	0.00276	0.00259	0.00324	0.00588
Exc. B	1.382	1.271	2.010	1.265	0.995	0.981	1.032	0.977
	0.0075	0.00177	0.0098	0.00302	0.00464	0.00101	0.00433	0.00197

Table 5 shows the performance of the mixture models trained by the direct and phased learning methods. The phased learning constantly helps enhance the performance. This implies that when the prediction and representation related modules of mixture models are insufficiently learned yet, simultaneously learning the prediction modules with the weight modules may fail to reliably capture the predictive power of different data sources, thereby leading to less accurate weighted combination.

Fig. 5 show the uncertainty conditioned prediction errors of all datasets. The uncertainty quantile level on the x-axis of each sub-figure is derived from the empirical distribution of predictive uncertainties.

Table 4: Results of probabilistic prediction metrics (mean and standard error respectively in the first and second line of each row). The best result is marked by the grey box.

	NLLm ↓				QLm ↓			
Dataset	DAR	DF	AF	**MIX**	DAR	DF	AF	**MIX**
Site 0	3.922	4.196	5.439	3.879	0.298	0.327	0.233	0.0463
	0.011	0.013	0.004	0.011	0.004	0.003	0.001	0.001
Site 1	3.860	3.992	5.208	3.696	0.231	0.202	0.237	0.0505
	0.021	0.029	0.002	0.003	0.004	0.004	0.001	0.001
Site 2	3.745	3.945	5.124	3.679	0.234	0.130	0.241	0.0502
	0.005	0.014	0.001	0.001	0.002	0.008	0.001	0.001
Site 3	4.034	4.150	5.510	3.962	0.277	0.199	0.233	0.0471
	0.016	0.015	0.001	0.001	0.011	0.007	0.001	0.001
Site 4	3.945	4.154	5.466	3.906	0.276	0.191	0.227	0.0453
	0.016	0.007	0.003	0.005	0.006	0.009	0.001	0.003
Site 5	3.917	4.081	5.460	3.866	0.274	0.533	0.235	0.0445
	0.015	0.009	0.001	0.001	0.003	0.103	0.001	0.001
Exc. A	1.594	1.578	1.601	1.541	2.974	4.498	4.982	2.840
	0.00485	0.00550	0.00324	0.00258	0.0413	0.0694	0.0983	0.0624
Exc. B	1.698	1.660	1.723	1.647	2.606	4.801	5.012	2.353
	0.00853	0.00131	0.00732	0.00220	0.0958	0.119	0.1012	0.0930

Table 5: Performance of the mixture models respectively trained by the direct and phased learning methods (mean and standard error respectively in the first and second line of each row).

	RMSE ↓		MAE ↓		NLLm ↓		QLm ↓	
Dataset	Direct	Phased	Direct	Phased	Direct	Phased	Direct	Phased
Site 0	27.173	19.022	14.905	10.297	4.103	3.879	0.0582	0.0463
	1.617	0.054	0.423	0.048	0.133	0.011	0.009	0.001
Site 1	41.549	18.517	17.876	9.929	3.772	3.696	0.0589	0.0505
	2.013	0.086	1.102	0.027	0.006	0.003	0.009	0.001
Site 2	17.279	17.081	8.832	8.573	3.720	3.679	0.0550	0.0502
	0.021	0.087	0.011	0.037	0.013	0.001	0.002	0.001
Site 3	30.509	21.969	14.241	11.293	4.392	3.962	0.0669	0.0471
	1.132	0.101	0.102	0.059	0.029	0.001	0.002	0.001
Site 4	26.592	19.637	17.908	10.638	3.948	3.906	0.0456	0.0453
	1.123	0.062	0.812	0.084	0.040	0.005	0.003	0.003
Site 5	30.233	20.578	19.843	10.318	3.891	3.866	0.0448	0.0445
	1.932	0.128	1.324	0.049	0.013	0.001	0.001	0.001
Exc. A	1.172	1.149	0.934	0.898	1.596	1.541	3.012	2.840
	0.00123	0.00168	0.0356	0.00588	0.00216	0.00258	0.0563	0.0624
Exc. B	1.282	1.265	0.998	0.977	1.656	1.647	2.510	2.353
	0.00203	0.00302	0.00123	0.00197	0.00158	0.00220	0.0890	0.0930

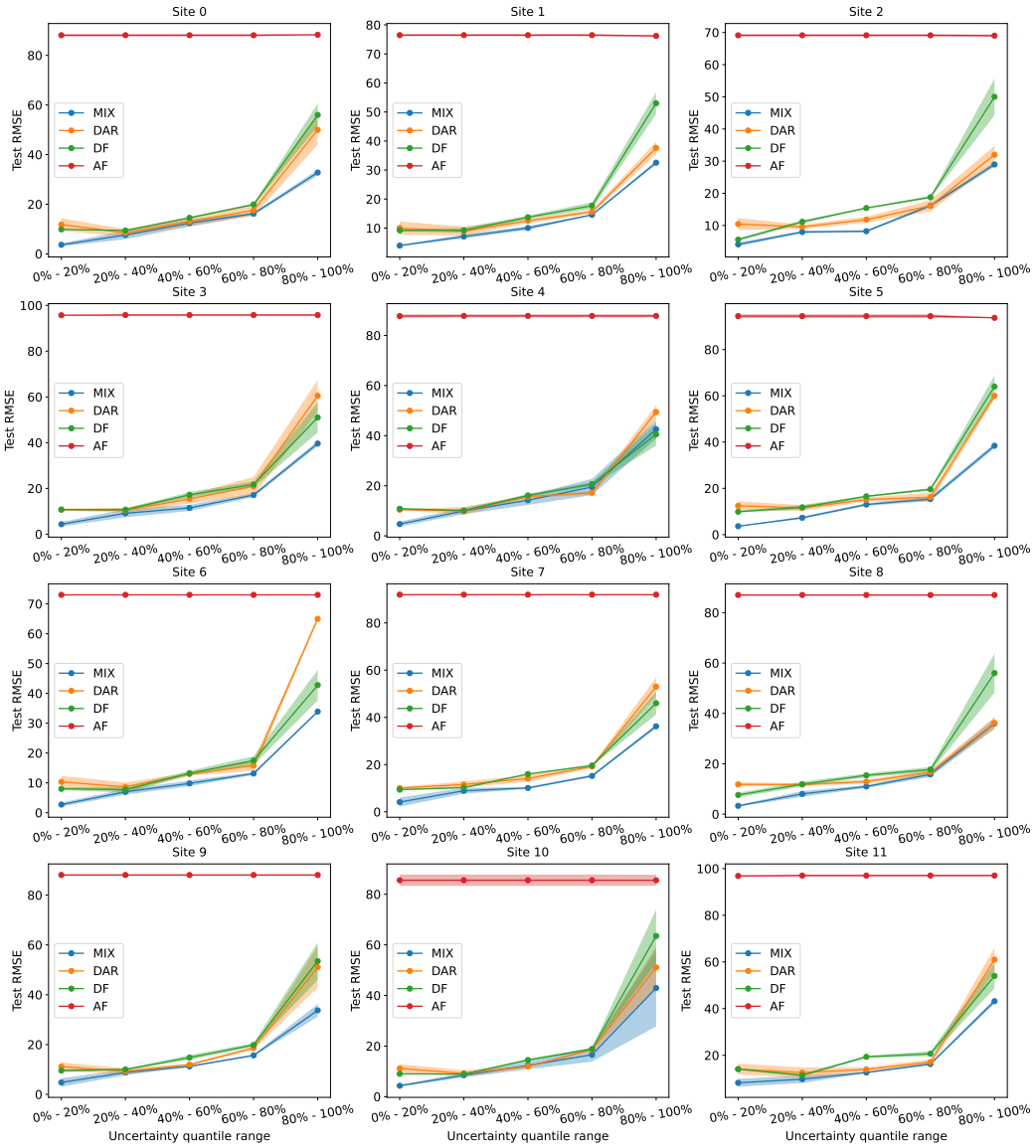

Figure 5: Uncertainty Conditioned Prediction Errors.