VARIATIONAL INFERENCE FOR DYNAMIC MIXTURES IN PROBABILISTIC TIME-SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Probabilistic time-series forecasting requires modeling complex, dynamic, and often multimodal conditional distributions while maintaining computational efficiency. We propose a lightweight finite mixture framework that uses a mixture sieve model and an analytically derived optimal posterior for reconstruction, thereby reducing the variational inference problem for conditional distribution estimation to a multi-class classification task. The framework defines the model output in the fractionally differenced domain, which alleviates distribution shift in the target variable and yields a dynamic mixture representation with time-varying component locations on the original scale. The proposed objective provides a quantization-level distributional approximation guarantee and has bounded logit-level gradients. Experiments on real-world benchmarks demonstrate that our method consistently outperforms existing probabilistic forecasting approaches while generating plausible future trajectories and competitive point estimates.

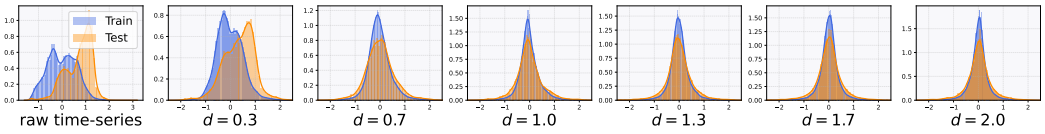
1 INTRODUCTION

Time-series forecasting has long supported decision-making across a wide range of real-world applications (Kaneko & Yada, 2016; Sezer et al., 2020; Fildes et al., 2022; Hong et al., 2024b). In such applications, the endogenous target variable is strongly influenced by multiple exogenous variables, whose interactions induce substantial volatility in future outcomes (Lago et al., 2021; Nayak et al., 2024). Despite the uncertainty observed in real-world data, many existing forecasting methods remain primarily focused on minimizing the mean squared error (MSE), thereby restricting their predictions to estimates of the conditional mean (Wu et al., 2021; Zhou et al., 2021; Liu et al., 2022; 2024; Wang et al., 2024a; 2026). These forecasts do not explicitly represent predictive uncertainty, limiting their ability to describe the range of possible future outcomes.

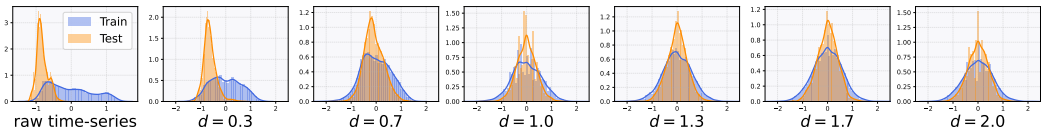
Probabilistic forecasting models are designed to quantify this uncertainty but face an inherent trade-off between distributional expressiveness and computational efficiency. Traditional likelihood-based methods (Salinas et al., 2020), multi-quantile approaches (Wen et al., 2017; Lim et al., 2021), and approaches based on the continuous ranked probability score (CRPS) (Gasthaus et al., 2019; Hong et al., 2024a) use structured output parameterizations for stable and efficient estimation, but their restricted expressiveness can introduce model bias into forecasts. Diffusion-based methods (Rasul et al., 2021; Shen & Kwok, 2023; Li et al., 2024; Yuan & Qiao, 2024) model more flexible predictive distributions but require costly iterative denoising during inference, whereas methods based on kernel density estimation (Li et al., 2026) can be sensitive to hyperparameters such as bandwidth. Motivated by this trade-off, we propose a lightweight yet highly expressive finite mixture framework that incorporates temporal mixture transitions through fractional differencing.

First, the proposed model represents the predictive density using mixture components whose locations are fixed on a predefined grid. This corresponds to expanding the density using basis functions centered on the grid, yielding a mixture sieve model (Genovese & Wasserman, 2000; Fox et al., 2016). With this mixture sieve formulation, substituting the analytically derived optimal posterior for reconstruction into the variational lower bound yields a categorical cross-entropy loss for learning the mixing weights. Consequently, we reduce the variational inference problem to a standard multi-class classification task, with class probabilities parameterized by a neural network. The resulting objective provides a principled probabilistic foundation for classification, rather than treating it merely as a heuristic proxy task, while supporting stable training and efficient inference.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) NP dataset



(b) ETTh1 dataset

Figure 1: **Effect of fractional differencing: alleviating distribution shift**. In each figure, the leftmost panel shows histograms of the standardized series from the training and test datasets, while the other panels show histograms of the fractionally differenced series for $d \in \{0.3, 0.7, 1.0, 1.3, 1.7, 2.0\}$. For clarity, the transformation $\text{sign}(x) \cdot \log(1 + |x|)$ is applied. For other datasets, see Appendix A.10.

Furthermore, our model integrates fractional differencing by defining its output space in the fractionally differenced domain. This output formulation alleviates distribution shift in the target variable and yields a dynamic mixture representation with time-varying component locations on the original scale. Although the transformation alone provides no intrinsic CRPS gain for an unrestricted forecast class, it helps our mixture model adapt to location shifts that can otherwise cause excess CRPS. The empirical results suggest that this formulation is particularly effective for forecasting series with complex and volatile dynamics. Previous deep forecasting studies have mainly used fractional differencing in two-stage approaches, where ARFIMA is fitted separately, and a neural network then models the remaining nonlinear structure (Bukhari et al., 2020; Saâdaoui & Rabbouch, 2024).

The theoretical analysis establishes that minimizing the classification-based objective aligns the model-induced forecast distribution with the target conditional distribution at the level of quantized regions. The objective has bounded logit-level gradients, which can improve optimization stability. Experiments with multiple neural network architectures on real-world benchmarks show that our method achieves strong probabilistic forecasting performance, provides competitive point estimates, and integrates readily with different backbones.

2 Proposed Method

This section introduces our method, and the overall procedure is illustrated in Figure 5 in the Appendix. Let $\{(\mathbf{z}_t, \mathbf{x}_t)\}_{t=1}^{T}$ be a multivariate time series of length T, where $(\mathbf{z}_t, \mathbf{x}_t) \in \mathbb{R}^D$, $\mathbf{x}_t \in \mathbb{R}$ denotes the endogenous target variable, and $\mathbf{z}_t \in \mathbb{R}^{D-1}$ denotes the exogenous covariates at time step t.

2.1 Fractionally Differenced Output Space

The target variable is defined as the fractionally differenced series $(1-B)^d \mathbf{x}_{t+1}$, where B is the backshift operator, $d \geq 0$ is the differencing order, and $(1-B)^d = \sum_{k=0}^{\infty} (-B)^k \binom{d}{k}$ (Hosking, 1981; Baillie, 1996). This expansion is approximated by truncation at a fixed lag C (Wilkins, 2004; Jensen & Nielsen, 2014). For notational convenience, the contribution of past observations is denoted by

$$\mathbf{h}_t \coloneqq \sum_{k=1}^{C} (-1)^{k+1} \binom{d}{k} \mathbf{x}_{t+1-k}, \qquad \Delta_{t+1}^{(d)} \coloneqq \mathbf{x}_{t+1} - \mathbf{h}_t,$$

where $\Delta_{t+1}^{(d)}$ denotes the truncated fractionally differenced series. As special cases, $d = 1$ and $d = 2$ yield the first- and second-order differences $\Delta_{t+1}^{(1)} = \mathbf{x}_{t+1} - \mathbf{x}_t$ and $\Delta_{t+1}^{(2)} = \mathbf{x}_{t+1} - 2\mathbf{x}_t + \mathbf{x}_{t-1}$ respectively, whereas $d = 0$ corresponds to no differencing, i.e., the raw time series.

Figure 1 illustrates histograms of the raw time-series values and their fractionally differenced counterparts. Compared to the raw time series ($d = 0$), applying fractional differencing ($d > 0$)

substantially alleviates the distribution shift between the training and test datasets, which is clearly present in the raw series, particularly in terms of mean and variance (i.e., weak stationarity), and value ranges (see Table 5 in Appendix A.9 for quantitative results using distribution distances). This supports defining the model output space in the fractionally differenced domain and directly estimating the corresponding conditional distribution.

2.2 Variational Inference for Dynamic Sieve Mixtures

Given the history of length L ending at time step t (i.e., the lookback window), the forecasting task is to estimate the conditional distribution of $\Delta_{t+1}^{(d)}$:

$$p(\Delta_{t+1}^{(d)} \mid \mathbf{x}_{t-L+1:t}, \mathbf{z}_{t-L+1:t}) \tag{1}$$

for $t < T$, where $\mathbf{x}_{t:t'}$ denotes the subsequence $(\mathbf{x}_t, \mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t'})$ for $t' > t$, with analogous notation for $\mathbf{z}$. Note that $C \leq L$, ensuring that $\mathbf{h}_t$ is fully determined by the lookback window. For multi-step horizons, forecasts are obtained recursively in an autoregressive fashion (Cortes et al., 2025). For notational simplicity, $(\mathbf{x}_{t-L+1:t}, \mathbf{z}_{t-L+1:t})$ will be denoted by $\mathbf{x}_{t-L+1:t}$ hereafter, as the inclusion of $\mathbf{z}_{t-L+1:t}$ in the conditioning set is straightforward.

Mixture sieve model At each time step $t < T$, a latent variable modeling approach is adopted to represent the conditional distribution of $\Delta_{t+1}^{(d)}$ in (1). Let $\theta \in \mathbb{R}$ denote the latent variable shared across all time steps. It is assumed that θ has finite support with M points (Fox et al., 2016), defined as $\Theta_M \coloneqq \{\theta_1, \theta_2, \cdots, \theta_M\}$, where $\theta_1 < \theta_2 < \cdots < \theta_M$.

Then, our mixture sieve model for (1) is defined as

$$p\big(\Delta_{t+1}^{(d)} \mid \mathbf{x}_{t-L+1:t}; \phi\big) \coloneqq \sum_{m=1}^{M} \pi_m\big(\mathbf{x}_{t-L+1:t}; \phi\big)\, p(\Delta_{t+1}^{(d)} \mid \theta_m, \beta^2), \tag{2}$$

where $p(\Delta_{t+1}^{(d)} \mid \theta_m, \beta^2)$ denotes the *known* decoder distribution with the fixed location parameter θ_m and the scale parameter $\beta > 0$, and the *unknown* prior distribution over the latent variable is given by the categorical distribution $\mathrm{Cat}\big(\Theta_M;\, \pi(\mathbf{x}_{t-L+1:t}; \phi)\big)$. Here, $\sum_{m=1}^{M} \pi_m(\cdot; \phi) = 1$, where $\pi_m(\cdot; \phi) \geq 0$, the subscript m refers to the mth element of the output vector $\pi(\cdot; \phi)$, and ϕ is a trainable neural network parameter. The fixed-location mixture representation is a reasonable design choice in the fractionally differenced domain because fractional differencing alleviates distribution shift in the target variable, as shown empirically in Figure 1.

Variational inference Rather than being estimated directly via MLE, the prior distribution in (2) is learned by optimizing a variational lower bound on the log-likelihood (Jordan et al., 1999). With a posterior $q(\theta \mid \Delta_{t+1}^{(d)})$, which is defined on Θ_M, the lower bound of the log-likelihood is

$$\begin{aligned} \log p(\Delta_{t+1}^{(d)} \mid \mathbf{x}_{t-L+1:t}; \phi) \quad \geq \quad & \underbrace{\sum_{m=1}^{M} q(\theta_m \mid \Delta_{t+1}^{(d)}) \log p(\Delta_{t+1}^{(d)} \mid \theta_m, \beta^2)}_{\text{reconstruction term}} \\ + \quad & \sum_{m=1}^{M} q(\theta_m \mid \Delta_{t+1}^{(d)}) \log \frac{\pi_m(\mathbf{x}_{t-L+1:t}; \phi)}{q(\theta_m \mid \Delta_{t+1}^{(d)})} \end{aligned} \tag{3}$$

by Jensen's inequality (see Appendix A.1 for a detailed derivation).

Proposition 2.1 (Optimal posterior for reconstruction)**.** *If, for every m, the decoder distribution $p(\Delta_{t+1}^{(d)} \mid \theta_m, \beta^2)$ is unimodal and has a density that strictly decreases as a function of $|\Delta_{t+1}^{(d)} - \theta_m|$ within its support, then the reconstruction term in (3) is maximized by the following posterior distribution:*

$$q(\theta_m \mid \Delta_{t+1}^{(d)}) = \begin{cases} 1, & \text{if } m = m(\Delta_{t+1}^{(d)}) \\ 0, & \text{otherwise} \end{cases}, \quad \text{where} \quad m(\Delta_{t+1}^{(d)}) \coloneqq \operatorname*{arg\,min}_{m=1,\cdots,M} \big|\Delta_{t+1}^{(d)} - \theta_m\big|.$$

Proposition 2.1 shows that the optimal posterior can be derived analytically without learnable parameters (see Appendix A.2 for the proof). Specifically, this posterior is the latent distribution over Θ_M that maximizes the reconstruction term in (3). The result further shows that the optimal posterior need not condition on the lookback window $\mathbf{x}_{t-L+1:t}$ and that the target variable $\Delta_{t+1}^{(d)}$ can be discretized through $m(\Delta_{t+1}^{(d)})$. Several distributions satisfy the decoder conditions in Proposition 2.1, including Gaussian and Laplace distributions. Also, under a winner-takes-all interpretation (Cortes et al., 2025), Θ_M represents M hypotheses for the differenced value at the next time step, and $\theta_{m(\Delta_{t+1}^{(d)})}$ can be viewed as the winner among them.

Objective function Under the optimal posterior in Proposition 2.1, the reconstruction term is constant with respect to the model parameters. Therefore, the objective derived from (3) is given by

$$\min_{\phi} \; \mathcal{L}(\Delta_{t+1}^{(d)}, \mathbf{x}_{t-L+1:t}; \phi) \;\coloneqq\; -\sum_{m=1}^{M} \mathbb{I}(m(\Delta_{t+1}^{(d)}) = m) \; \log \pi_m(\mathbf{x}_{t-L+1:t}; \phi), \tag{4}$$

where $\mathbb{I}(\cdot)$ is the indicator function and the entropy of the optimal posterior is zero. As shown in (4), our training objective reduces to a categorical cross-entropy loss for learning only the mixing weights of the mixture sieve model in (2). Unlike conventional VAEs (Kingma & Welling, 2014), which typically assume a fixed prior, our framework treats the prior as unknown and learns it through ϕ. The training procedure is outlined in Algorithm 1.

Algorithm 1 Training algorithm

Require: $\mathbf{x}_{t+1}, \mathbf{x}_{t-L+1:t}, \Theta_M, \pi(\cdot; \phi)$

1: $\mathbf{h}_t \leftarrow \sum_{k=1}^{C} (-1)^{k+1} \binom{d}{k} \mathbf{x}_{t+1-k}$
2: $\Delta_{t+1}^{(d)} \leftarrow \mathbf{x}_{t+1} - \mathbf{h}_t$
3: $m(\Delta_{t+1}^{(d)}) \leftarrow \underset{m=1,\cdots,M}{\arg\min} \left|\Delta_{t+1}^{(d)} - \theta_m\right|$
4: Compute $\mathcal{L}(\Delta_{t+1}^{(d)}, \mathbf{x}_{t-L+1:t}; \phi)$.
5: Update ϕ using gradient descent.

Inference Using the inverse relation $\mathbf{x}_{t+1} = \Delta_{t+1}^{(d)} + \mathbf{h}_t$ in (2) yields the predictive distribution of $\mathbf{x}_{t+1}$:

$$p\big(\mathbf{x}_{t+1} \mid \mathbf{x}_{t-L+1:t}; \phi\big) = \sum_{m=1}^{M} \pi_m\big(\mathbf{x}_{t-L+1:t}; \phi\big)\, p\big(\mathbf{x}_{t+1} \mid \theta_m + \mathbf{h}_t, \beta^2\big), \tag{5}$$

which highlights an important role of fractional differencing. Although the mixture sieve model in (2) uses fixed locations θ_m in the fractionally differenced domain, the inverse transformation shifts each location to $\theta_m + \mathbf{h}_t$ on the original scale. Since $\mathbf{h}_t$ is determined by past observations in the lookback window and varies over time, the resulting component locations also vary over time. Fractional differencing is therefore not an independent preprocessing step but an integral part of our classification-based framework: the proposed objective learns mixture weights over fixed locations in the differenced domain, while the inverse transformation induces a dynamic mixture predictive model with time-varying component locations on the original scale.

Then, probabilistic forecasts are generated as follows:

$$\theta_m \sim \mathrm{Cat}\big(\Theta_M;\, \pi(\mathbf{x}_{t-L+1:t}; \phi)\big) \implies \Delta_{t+1}^{(d)} \sim p(\Delta_{t+1}^{(d)} \mid \theta_m, \beta^2) \implies \mathbf{x}_{t+1} \leftarrow \Delta_{t+1}^{(d)} + \mathbf{h}_t,$$

which corresponds to ancestral sampling from the predictive mixture model in (5). By recursively applying this procedure K times, probabilistic forecasts over multiple horizons are obtained in the form of K predicted trajectories. Our framework employs a Gaussian decoder that satisfies the conditions in Proposition 2.1, enables efficient trajectory sampling, and performs well empirically. Moreover, under the Gaussian decoder, the point estimate admits a closed-form solution given by a weighted sum of the latent variables. The main experiments use the average of the generated trajectories for point estimation, while the closed-form expression and its experimental results are reported in Appendix A.9.

Latent support Under weak stationarity, $\Delta_1^{(d)}, \Delta_2^{(d)}, \cdots, \Delta_{T'}^{(d)}$ are treated as T' realizations of a random variable $\Delta^{(d)}$ (T': the length of the training set), and the empirical cumulative distribution function $\hat{F}$ of $\Delta^{(d)}$ is estimated after standardization. Then, Θ_M is defined as the midpoints of the intervals defined by consecutive quantiles $\hat{F}^{-1}(\tau)$ for $\tau \in \mathcal{T}$, where $\mathcal{T}$ denotes $M+1$ equally spaced quantile levels in $(0, 1)$. Although the nearest-center assignment, i.e., discretization via $m(\cdot)$, does not exactly align with the original quantile boundaries when the intervals have unequal widths (as is typical in practice), the resulting midpoints still provide representative location parameters for the corresponding interval.

2.3 Theoretical Rationale and Properties

This section provides a rationale for integrating fractional differencing into our model and establishes the cell-level distributional approximation and bounded-gradient properties of the objective.

Proposition 2.2. *For unrestricted forecast classes, defining the output in the fractionally differenced domain does not change the minimum achievable expected test CRPS, even under distribution shift.*

Given the lookback window, the fractionally differenced and original targets differ only by a known translation. Because CRPS is translation-invariant, defining the predictive distribution in the fractionally differenced domain alone provides no intrinsic CRPS gain for an unrestricted forecast class (see Appendix A.3 for the proof). Our mixture model, however, can benefit from defining the target in the fractionally differenced domain: the inverse transformation induces time-varying locations $\theta_m + \mathbf{h}_t$ on the original scale, which help the model adapt to location shifts. This flexibility matters because a mixture class with component locations fixed on the original scale can incur positive expected excess test CRPS in such settings (see Proposition A.1 for the formal statement).

Next, a quantization-based theoretical analysis of our distributional learning framework is established using the one-dimensional Voronoi cells induced by Θ_M. For each $m \in \{1, \dots, M\}$, the corresponding Voronoi cell is defined as $\mathcal{B}_m := \left\{\Delta^{(d)} \in \mathbb{R} : |\Delta^{(d)} - \theta_m| \leq |\Delta^{(d)} - \theta_l|, \forall l \neq m\right\}$.

Proposition 2.3. *The total variation distance between the model-induced and target distributions over the corresponding cells is upper-bounded by two terms: the cell-probability estimation error and the cross-cell decoder leakage. The former is controlled by the proposed objective, while the latter decreases as β becomes small relative to the quantization resolution of Θ_M.*

Proposition 2.3 establishes a cell-level distributional approximation guarantee for the proposed classification-based objective. At the population level, our objective controls the cell-probability estimation error in total variation distance, while the remaining error arises from decoder leakage, which decreases as β becomes small relative to the spacing between adjacent quantization points. The formal bound and proof are provided in Appendix A.4.

Proposition 2.4. *The gradient of the proposed objective in (4) w.r.t. the logits is bounded.*

Lastly, Proposition 2.4 shows that the gradients of our objective with respect to the logits are bounded by the probability-simplex constraint. This boundedness can improve optimization stability (Imani & White, 2018). In contrast, the gradient of MSE grows linearly with the magnitude of the prediction residual. See Appendix A.5 for the proof.

3 Experiments

To evaluate the effectiveness of our method, two tasks are considered for an endogenous target variable with exogenous covariates: multi-horizon probabilistic forecasting and point estimation. Our method is denoted by `CLS-PF` (CLaSsification-based Probabilistic Forecaster), or `CLS` for short. The name refers to the classification-based objective used to train the model.

3.1 Experimental Setup

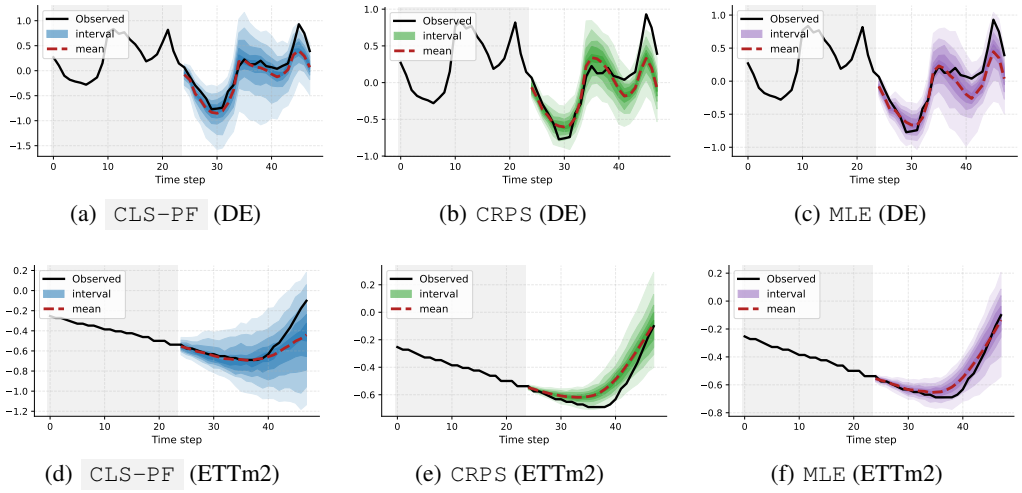
Datasets Following Wang et al. (2024c); Zhou et al. (2025), five electricity price forecasting (EPF) datasets (BE, DE, FR, NP, PJM) (Lago et al., 2021) are used, along with five real-world forecasting datasets (ETT (Zhou et al., 2021), Weather[1]). The ETT datasets are chronologically split in a 6:2:2 ratio, while the remaining datasets follow a 7:1:2 ratio (Wang et al., 2024b). See Appendix A.6 for detailed dataset descriptions.

Implementation All experiments are repeated with five random seeds. Our method has four hyperparameters: M, β, d, and C. Unless otherwise specified, $M = 10$ and $\beta = 0.1$ are used without tuning. Only the fractional differencing order d is selected from $\{0.3, 0.7, 1.0, 1.3, 1.7, 2.0\}$ based on performance on the validation dataset. The truncation lag is fixed at $C = 150$ in all experiments and does not exceed the input window length, i.e., $C \leq L$. All implementation details for our method are provided in Appendix A.7.

[1] https://www.bgc-jena.mpg.de/wetter/

Table 1: **Forecasting performance** averaged over three neural network architectures. The best results are highlighted in **bold**, while the second-best results are underlined. Lower is better.

	Probabilistic forecasting												*Point estimation*											
	CRPS						QICE						RMSE						MAE					
dataset	CLS	PPM	WTA	DDIM	CRPS	MLE	CLS	PPM	WTA	DDIM	CRPS	MLE	CLS	PPM	WTA	DDIM	CRPS	MLE	CLS	PPM	WTA	DDIM	CRPS	MLE
BE	**.287**	.346	.387	.346	.308	.336	.021	.025	.021	.015	.027	**.011**	.623	.761	.708	.666	**.616**	.652	**.249**	.303	.321	.297	.260	.282
DE	**.419**	.549	.472	.651	.445	.453	**.009**	.078	.040	.037	.034	.022	**.557**	.890	.598	.753	.579	.602	**.346**	.440	.376	.504	.360	.373
FR	**.245**	.344	.280	.312	.257	.260	**.008**	.022	.029	.026	.020	.009	**.639**	.994	.732	.681	**.639**	.648	**.194**	.295	.226	.249	.208	.209
NP	**.220**	.323	.288	.363	.245	.234	**.013**	.096	.039	.048	.028	.015	**.456**	.915	.520	.569	.482	.480	**.234**	.344	.288	.349	.256	.253
PJM	.299	.343	.314	.343	.302	**.297**	**.010**	.084	.037	.025	.032	.020	.261	.393	.281	.289	.264	**.259**	.161	.180	.169	.185	.163	**.160**
ETTh1	**.077**	.081	.081	.090	.084	.082	**.019**	.032	.041	.036	.041	.029	**.167**	.174	.172	.200	.179	.175	**.128**	.133	.131	.152	.136	.133
ETTh2	**.174**	.192	.186	.218	.184	.180	**.014**	.037	.031	.025	.038	.026	**.275**	.296	.298	.381	.297	.290	**.201**	.221	.219	.259	.213	.210
ETTm1	**.046**	.050	.048	.069	.048	.047	**.012**	.051	.024	.049	.031	**.012**	**.100**	.110	.106	.145	.106	.103	**.073**	.078	.077	.108	.077	.074
ETTm2	.125	.145	.143	.207	**.118**	.119	.034	.098	.038	.027	.029	**.019**	.168	.190	.198	.267	**.161**	.167	**.104**	.117	.129	.179	.106	.108
Weather	**.159**	.209	.252	.228	.173	.294	**.019**	.097	.078	.058	.049	.043	**.019**	.021	.026	.025	**.019**	.028	**.012**	.014	.018	.016	**.012**	.019



(a) `CLS-PF` (DE) (b) `CRPS` (DE) (c) `MLE` (DE)

(d) `CLS-PF` (ETTm2) (e) `CRPS` (ETTm2) (f) `MLE` (ETTm2)

Figure 2: **Qualitative results** for the DE and ETTm2 datasets. Visualization of probabilistic forecasting for each method, where the shaded regions represent the density estimated from the generated 100 trajectories, and the red dashed line denotes the mean prediction.

Baselines The baseline comparison includes the following probabilistic forecasting methods, distinguished by their training objectives: maximum likelihood estimation (`MLE`; Salinas et al., 2020), the continuous ranked probability score (`CRPS`; Gasthaus et al., 2019; Hong et al., 2024a), a diffusion model with accelerated DDIM sampling (`DDIM`; Rasul et al., 2021; Shen & Kwok, 2023; Li et al., 2024; Yuan & Qiao, 2024; Song et al., 2021), the winner-takes-all loss (`WTA`; Cortes et al., 2025), and parametric prior mapping with kernel density estimation (`PPM`; Li et al., 2026). For all methods, including ours, hyperparameters are selected through an exhaustive grid search on the validation set, with the hyperparameter search spaces reported in Appendix Table 4. Implementation details for all baselines are provided in Appendix A.7.

Evaluation For probabilistic forecasting, the performance is evaluated using CRPS (Matheson & Winkler, 1976), QICE (Han et al., 2022), and Total Variation (TV) (Cortes et al., 2025). CRPS measures the agreement between the empirical cumulative distribution function constructed from the generated trajectories and the observed data, while QICE (quantile interval calibration error) measures deviations between the empirical coverage of each quantile interval and its optimal proportion. Total Variation assesses the smoothness of the predicted trajectories. For all methods, including ours, 100 future trajectories are generated at inference time, and point estimates are obtained by averaging these trajectories. The RMSE and MAE are adopted for the point-estimation evaluation metrics. Especially, CRPS was used as a model selection criterion, and details of all evaluation metrics are provided in Appendix A.8.

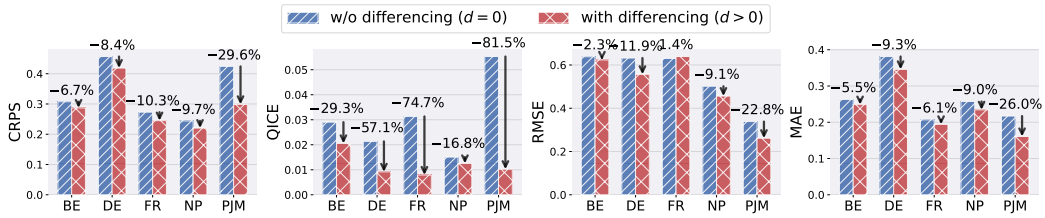


Figure 3: **Effect of fractional differencing in our model**. Forecasting performance comparing no differencing ($d = 0$) and fractional differencing ($d > 0$), averaged across three neural architectures. Lower values are better; more negative relative percentage changes indicate larger improvements.

Following previous works (Olivares et al., 2023; Wang et al., 2024c; Zhou et al., 2025), the lookback and horizon subsequence lengths are set to 168 and 24, respectively. Our experiments focus on evaluating how effectively different methods quantify predictive uncertainty for the endogenous target, rather than comparing their multivariate forecasting capabilities. Accordingly, all evaluation metrics are reported only for the endogenous target variable.

Backbone architectures To ensure a fair comparison, each baseline method is evaluated using the same three backbone architectures: TSMixer (Chen et al., 2023), TimeXer (Wang et al., 2024c), and CrossLinear (Zhou et al., 2025). These architectures are designed to capture cross-variable dependencies or incorporate exogenous inputs and represent MLP-, Transformer-, and CNN-based models, respectively. Detailed configurations are provided in Table 3 in the Appendix. For normalization, global normalization is applied by independently standardizing each variable using statistics computed from the training set.

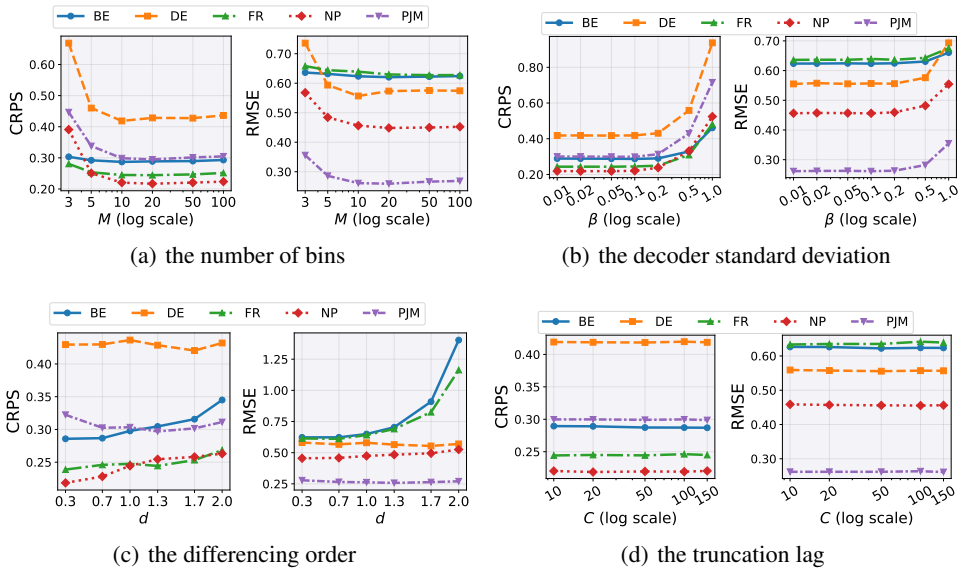
3.2 Main Results

Forecasting performance Table 1 reports the probabilistic and point forecasting results. Our method shows strong probabilistic forecasting performance, achieving the best CRPS on 8 of the 10 datasets and ranking first or second on 9 datasets. Its point forecasts are also competitive, with particularly strong performance in terms of MAE. These results indicate that our method produces reliable one-step predictions, despite the potential accumulation of errors during iterative multi-step forecasting. The observed performance may be partly attributed to the bounded-gradient property established in Proposition 2.4. Results for each neural architecture, including standard errors, are provided in Appendix A.9. The paired Wilcoxon signed-rank tests reported in Table 7 of the Appendix show that the CRPS improvements in Table 1 are statistically significant at the 0.05 significance level.

Qualitative comparison Figure 2 compares forecasts from our method, `CRPS`, and `MLE` on the DE and ETTm2 datasets. Our method outperforms these two strong baselines on DE but does not achieve the best performance on ETTm2. Additional qualitative results for the other datasets and baselines are provided in Appendix A.11. The DE example shows frequent fluctuations and changes in direction, whereas the ETTm2 example follows a smoother trend. Around time steps 35–40 in DE, the predictive region of our method covers the observed movement, and its mean forecast closely aligns with the observations. In ETTm2, however, the baselines follow the smooth upward turn more closely, while our mean forecast underestimates the rise. These examples suggest that our method is particularly effective for forecasting series with complex and volatile dynamics.

Ablation study: effect of fractional differencing Figure 3 compares forecasting performance with and without fractional differencing. Relative to the raw time series ($d = 0$), fractional differencing ($d > 0$) improves performance for all but one dataset-metric combination across the five EPF datasets. These results are consistent with the analysis in Section 2.3 and suggest that defining the output in the fractionally differenced domain makes the finite mixture representation more effective. Additional ablation results for the remaining datasets are provided in Appendix A.9.

To further assess the effect of fractional differencing, the baselines are trained in the same fractionally differenced domain as our model, with their hyperparameters tuned accordingly (see Table 8 in



(a) the number of bins

(b) the decoder standard deviation

(c) the differencing order

(d) the truncation lag

Figure 4: **Sensitivity analyses** of CRPS and RMSE on five EPF datasets, with results averaged across three neural network architectures.

the Appendix for the detailed results). Across the five EPF datasets, our method still outperforms these baseline variants in the CRPS metric. Compared with training on the raw time series ($d = 0$), fractional differencing ($d > 0$) yields an average CRPS reduction of 12.94% for our method, substantially larger than the 5.64% reduction achieved by PPM, which shows the second-largest improvement. These results show that fractional differencing can improve baseline performance as a preprocessing step but is particularly effective in our model, supporting its role as an integral part of the predictive model rather than a separate preprocessing component.

3.3 Model Analysis

This section presents sensitivity analyses for the key hyperparameters of our method. Corresponding results for the remaining datasets are provided in Appendix A.9.1.

Sensitivity analysis 1: Number of bins Figure 4(a) evaluates $M \in \{3, 5, 10, 20, 50, 100\}$. Both CRPS and RMSE remain stable for $M \geq 10$, indicating that $M = 10$ provides sufficiently fine discretization for conditional density estimation. A similar pattern is observed across the remaining datasets. Although increasing M further increases the discretization resolution, its potential benefits may be offset by two trade-offs. First, each class contains approximately $1/M$ of the training observations, increasing the variance of the estimated class probabilities. Second, more closely spaced support points increase component overlap and probability leakage between neighboring Voronoi cells. Thus, increasing M does not necessarily yield further performance improvements.

Sensitivity analysis 2: Decoder standard deviation As shown in Figure 4(b), over $\beta \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0\}$, both CRPS and RMSE remain relatively stable for $\beta \leq 0.2$ but increase at larger values of β. A similar pattern is observed across the remaining datasets. The degradation is consistent with the cross-cell decoder leakage in Proposition 2.3, as a larger β allows more probability mass to diffuse across Voronoi cell boundaries.

Sensitivity analysis 3: Differencing order Across $d \in \{0.3, 0.7, 1.0, 1.3, 1.7, 2.0\}$, Figure 4(c) shows that CRPS generally increases slightly with d, indicating a modest decline in probabilistic forecasting performance. RMSE remains relatively stable on most datasets but increases at larger values of d on the BE and FR datasets. Together with the results for the remaining datasets, these findings show that forecasting performance is generally stable for $d \in \{0.3, 0.7, 1.0, 1.3\}$, whereas larger differencing orders can degrade performance on some datasets.

Sensitivity analysis 4: Truncation lag Figure 4(d) evaluates truncation lag C in $\{10, 20, 50, 100, 150\}$. Forecasting performance remains highly stable across the evaluated metrics and datasets, with a similar pattern observed for the remaining datasets. These results indicate that our mixture sieve model remains effective with the time-varying component locations induced by the fractionally differenced output, even when short truncation lags are used.

Due to space limitations, two additional analyses are briefly summarized; full results are provided in Appendix A.9. **(1) Total variation.** Table 9 shows that our method is competitive in terms of the temporal smoothness of its generated trajectories. **(2) Inference runtime.** Figure 7 shows that our method achieves a competitive runtime for trajectory generation during inference.

4 Related Works

Probabilistic time-series forecasting Probabilistic forecasting (PF) aims to estimate the conditional distribution of future variables rather than produce a single point estimate. Early deep learning approaches (Rangapuram et al., 2018; Salinas et al., 2020) assumed predefined parametric output distributions and were trained via maximum likelihood estimation, limiting the predictive distribution to the chosen parametric family. To improve flexibility, Chung et al. (2015); Nguyen & Quanz (2021); Tang & Matteson (2021) adopted generative modeling frameworks based on variational inference. Diffusion-based approaches (Rasul et al., 2021; Shen & Kwok, 2023; Li et al., 2024; Yuan & Qiao, 2024) further improve expressiveness by modeling complex distributions through score-function estimation. A complementary line of research addresses the misalignment between training objectives and evaluation metrics. Scoring rule-based methods directly optimize proper scoring rules (Gasthaus et al., 2019; Pacchiardi et al., 2024; Hong et al., 2024a). More recently, Cortes et al. (2025) adopted a winner-takes-all objective that supervises only the best trajectory among multiple candidates. Li et al. (2026) combined a negative log-likelihood objective based on kernel density estimation with reparameterized sampling.

Endogenous forecasting with exogenous variables In many applications, the endogenous target variable exhibits high volatility because it is affected by multiple exogenous variables, resulting in substantial predictive uncertainty. Examples include the effects of climate change on crop prices (Nayak et al., 2024) and renewable energy penetration on electricity prices (Lago et al., 2021). TSMixer (Chen et al., 2023) models temporal and cross-variable dependencies through MLP-based mixing operations along the temporal and feature dimensions. More recent methods explicitly model directed interactions from exogenous inputs to the endogenous target. TimeXer (Wang et al., 2024c) combines specialized embedding layers with self- and cross-attention mechanisms to effectively integrate information across variables, whereas CrossLinear (Zhou et al., 2025) employs a CNN-based architecture to capture time-invariant and direct variable dependencies.

5 Conclusion and Limitations

We present a probabilistic time-series forecasting framework that couples fractional differencing with a mixture sieve model formulated via variational inference. Fractional differencing is an integral part of the predictive model by alleviating distribution shift in the target variable and yielding a dynamic mixture representation with time-varying component locations. Extensive experiments on benchmark datasets show that our method outperforms the baselines in probabilistic forecasting, remains competitive in point estimation, and is robust to hyperparameter choices.

Despite its empirical strengths, our framework has several limitations that suggest promising directions for future research. First, while our method performs well across diverse datasets, some aspects require further theoretical characterization. Specifically, a formal theoretical analysis is needed to characterize the effects of truncating the fractional differencing expansion over finite historical windows and to explain the strong empirical performance observed even in settings where the differencing order exceeds unity ($d > 1$). Second, the fractional differencing order d is currently treated as a tunable hyperparameter; end-to-end learning or dynamic adaptation of d during training could streamline model selection and eliminate manual tuning. Lastly, extending the current univariate formulation to jointly forecast multiple correlated targets by explicitly modeling their dependencies remains an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation and to formalize mathematical claims. We have not used generative AI tools for any other tasks requiring disclosure under the ICLR 2027 AI Policy for Authors. Additionally, we used generative AI tools for grammatical correction and stylistic refinement of the manuscript. We reviewed and verified all AI-assisted translations and edits. We take responsibility for the final content of this work, including any text produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the information required to reproduce our results throughout the paper and supplementary material. The proposed method and experimental setup are described in Sections 2 and 3. Detailed descriptions and citations of the datasets used in our experiments are provided in Appendix A.6. Implementation details, hyperparameters, and evaluation metrics are provided in Appendix A.7 and A.8. Additional results and analyses are included in Appendix A.9, A.10, and A.11. The anonymized source code and configuration files are included in the supplementary material.

REFERENCES

Richard T Baillie. Long memory processes and fractional integration in econometrics. *Journal of econometrics*, 73(1):5–59, 1996.

Ayaz Hussain Bukhari, Muhammad Asif Zahoor Raja, Muhammad Sulaiman, Saeed Islam, Muhammad Shoaib, and Poom Kumam. Fractional neuro-sequential arfima-lstm for financial market forecasting. *Ieee Access*, 8:71326–71338, 2020.

Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. TSMixer: An all-MLP architecture for time series forecasting. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=wbpxTuXgm0.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *NIPS*, 2015.

Adrien Cortes, Remi Rehm, and Victor Letzelter. Winner-takes-all for multivariate probabilistic time series forecasting. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=4QcFfTu6UT.

Robert Fildes, Shaohui Ma, and Stephan Kolassa. Retail forecasting: Research and practice. *International Journal of Forecasting*, 38(4):1283–1318, 2022.

Jeremy T Fox, Kyoo il Kim, and Chenyu Yang. A simple nonparametric approach to estimating the distribution of random coefficients in structural models. *Journal of Econometrics*, 195(2):236–254, 2016.

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*, pp. 1901–1910. PMLR, 2019.

Christopher R Genovese and Larry Wasserman. Rates of convergence for the gaussian mixture sieve. *The Annals of Statistics*, 28(4):1105–1127, 2000.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 69(2): 243–268, 2007.

Yannig Goude, Raphael Nedellec, and Nicolas Kong. Local short and middle term electricity load forecasting with semi-parametric additive models. *IEEE transactions on smart grid*, 5(1):440–446, 2013.

Xizewen Han, Huangjie Zheng, and Mingyuan Zhou. Card: Classification and regression diffusion models. *Advances in Neural Information Processing Systems*, 35:18100–18115, 2022.

Sungchul Hong, Seunghwan An, and Jong-June Jeon. Cryptocurrency price forecasting using variational autoencoder with versatile quantile modeling. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 4530–4537, 2024a.

Sungchul Hong, Yunjin Choi, and Jong-June Jeon. Interpretable water level forecaster with spatiotemporal causal attention mechanisms. *International Journal of Forecasting*, 2024b.

J. R. M. Hosking. Fractional differencing. *Biometrika*, 68(1):165–176, 1981. ISSN 00063444. URL http://www.jstor.org/stable/2335817.

Ehsan Imani and Martha White. Improving regression performance with distributional losses. In *International conference on machine learning*, pp. 2157–2166. PMLR, 2018.

Andreas Noack Jensen and Morten Ørregaard Nielsen. A fast fractional difference algorithm. *Journal of Time Series Analysis*, 35(5):428–436, 2014.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Yuta Kaneko and Katsutoshi Yada. A deep learning approach for the prediction of retail store sales. In *2016 IEEE 16th International conference on data mining workshops (ICDMW)*, pp. 531–537. IEEE, 2016.

Nektaria V Karakatsani and Derek W Bunn. Fundamental and behavioural drivers of electricity price volatility. *Studies in Nonlinear Dynamics & Econometrics*, 14(4), 2010.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International conference on learning representations*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *The third International Conference on Learning Representations*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Jesus Lago, Grzegorz Marcjasz, Bart De Schutter, and Rafał Weron. Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark. *Applied Energy*, 293:116983, 2021.

Jinglin Li, Jun Tan, QI Fang, and Ning Gui. Parametric prior mapping framework for non-stationary probabilistic time series forecasting. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=r4GbYGbWM5.

Yuxin Li, Wenchao Chen, Xinyue Hu, Bo Chen, Baolin Sun, and Mingyuan Zhou. Transformer-modulated diffusion models for probabilistic multivariate time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X. Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=0EXmFzUn5I.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=JePfAI8fah.

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

Adam Misiorek and Rafał Weron. Forecasting spot electricity prices with time series models. In *Proceedings of the European Electricity Market EEM-05 Conference*, pp. 133–141, 2005.

GH Harish Nayak, Md Wasi Alam, KN Singh, G Avinash, Rajeev Ranjan Kumar, Mrinmoy Ray, and Chandan Kumar Deb. Exogenous variable driven deep learning models for improved price forecasting of top crops in india. *Scientific Reports*, 14(1):17203, 2024.

Nam H. Nguyen and Brian Quanz. Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting. In *AAAI Conference on Artificial Intelligence*, 2021.

Kin G Olivares, Cristian Challu, Grzegorz Marcjasz, Rafał Weron, and Artur Dubrawski. Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with nbeatsx. *International Journal of Forecasting*, 39(2):884–900, 2023.

Lorenzo Pacchiardi, Rilwan A Adewoyin, Peter Dueben, and Ritabrata Dutta. Probabilistic forecasting with generative networks via scoring rule minimization. *Journal of Machine Learning Research*, 25(45):1–64, 2024.

Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31, 2018.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International conference on machine learning*, pp. 8857–8868. PMLR, 2021.

Foued Saâdaoui and Hana Rabbouch. Financial forecasting improvement with lstm-arfima hybrid models and non-gaussian distributions. *Technological Forecasting and Social Change*, 206:123539, 2024.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3): 1181–1191, 2020.

Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

Lifeng Shen and James Kwok. Non-autoregressive conditional diffusion models for time series prediction. In *International Conference on Machine Learning*, pp. 31016–31029. PMLR, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

Binh Tang and David S. Matteson. Probabilistic transformer for time series analysis. In *Neural Information Processing Systems*, 2021.

Hao Wang, Licheng Pan, Yuan Lu, Zhixuan Chu, Xiaoxi Li, Shuting He, Zi Chan, Qingsong Wen, Haoxuan Li, and Zhouchen Lin. Distdf: Time-series forecasting needs joint-distribution wasserstein alignment. In *International Conference on Learning Representations*, volume 2026, pp. 131512–131535, 2026.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and JUN ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=7oLshfEIC2.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Mingsheng Long, and Jianmin Wang. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*, 2024b.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. *Advances in Neural Information Processing Systems*, 37:469–498, 2024c.

Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

Nigel Wilkins. Indirect estimation of long memory volatility models. In *Econometric Society 2004 Far Eastern Meetings*, number 459. Econometric Society, 2004.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

Xinyu Yuan and Yan Qiao. Diffusion-TS: Interpretable diffusion for general time series generation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4h1apFjO99.

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. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Pengfei Zhou, Yunlong Liu, Junli Liang, Qi Song, and Xiangyang Li. Crosslinear: Plug-and-play cross-correlation embedding for time series forecasting with exogenous variables. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 4120–4131, 2025.

A APPENDIX

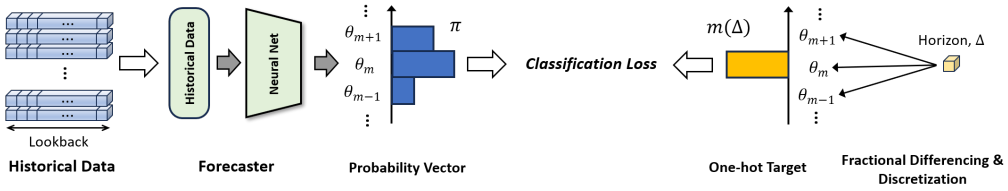


Figure 5: Overall process of our proposed method, CLS-PF.

A.1 LOWER-BOUND DERIVATION IN (3)

The lower bound in (3) can be derived as follows: by Jensen's inequality,

$$\begin{aligned}
&\log p(\Delta_{t+1}^{(d)} \mid \mathbf{x}_{t-L+1:t}; \phi) \\
= \;& \log \sum_{m=1}^{M} p(\Delta_{t+1}^{(d)} \mid \theta_m, \beta^2) \pi_m(\mathbf{x}_{t-L+1:t}; \phi) \\
= \;& \log \sum_{m=1}^{M} p(\Delta_{t+1}^{(d)} \mid \theta_m, \beta^2) \pi_m(\mathbf{x}_{t-L+1:t}; \phi) \frac{q(\theta_m \mid \Delta_{t+1}^{(d)})}{q(\theta_m \mid \Delta_{t+1}^{(d)})} \\
\geq \;& \sum_{m=1}^{M} q(\theta_m \mid \Delta_{t+1}^{(d)}) \log \left(p(\Delta_{t+1}^{(d)} \mid \theta_m, \beta^2) \frac{\pi_m(\mathbf{x}_{t-L+1:t}; \phi)}{q(\theta_m \mid \Delta_{t+1}^{(d)})} \right) \\
= \;& \sum_{m=1}^{M} q(\theta_m \mid \Delta_{t+1}^{(d)}) \log p(\Delta_{t+1}^{(d)} \mid \theta_m, \beta^2) + \sum_{m=1}^{M} q(\theta_m \mid \Delta_{t+1}^{(d)}) \log \frac{\pi_m(\mathbf{x}_{t-L+1:t}; \phi)}{q(\theta_m \mid \Delta_{t+1}^{(d)})}.
\end{aligned}$$

A.2 PROOF OF PROPOSITION 2.1

Proof. Given arbitrary $\Delta_{t+1}^{(d)} \in \mathbb{R}$, let $m^* := m(\Delta_{t+1}^{(d)})$ for notational simplicity. Since the decoder has a unique maximum at the location parameter, for $l \neq m^*$,

$$\log p(\Delta_{t+1}^{(d)} \mid \theta_{m^*}, \beta^2) > \log p(\Delta_{t+1}^{(d)} \mid \theta_l, \beta^2).$$

Consider an arbitrary posterior distribution $\tilde{q}$ such that $\tilde{q}(\theta_i \mid \Delta_{t+1}^{(d)}) = \alpha_i$ for all $i = 1, 2, \cdots, M$, where $\alpha_i \geq 0$ for all i, $\sum_{i=1}^{M} \alpha_i = 1$, and $\alpha_{m^*} < 1$.

Then, we have

$$\begin{aligned}
&\sum_{i=1}^{M} \tilde{q}(\theta_i \mid \Delta_{t+1}^{(d)}) \log p(\Delta_{t+1}^{(d)} \mid \theta_i, \beta^2) = \sum_{i=1}^{M} \alpha_i \log p(\Delta_{t+1}^{(d)} \mid \theta_i, \beta^2) \\
= \;& \log p(\Delta_{t+1}^{(d)} \mid \theta_{m^*}, \beta^2) + \sum_{i \neq m^*} \alpha_i \underbrace{\left(\log p(\Delta_{t+1}^{(d)} \mid \theta_i, \beta^2) - \log p(\Delta_{t+1}^{(d)} \mid \theta_{m^*}, \beta^2) \right)}_{<0} \\
< \;& \log p(\Delta_{t+1}^{(d)} \mid \theta_{m^*}, \beta^2) \\
= \;& \sum_{i=1}^{M} q(\theta_i \mid \Delta_{t+1}^{(d)}) \log p(\Delta_{t+1}^{(d)} \mid \theta_i, \beta^2),
\end{aligned}$$

where

$$q(\theta_m \mid \Delta_{t+1}^{(d)}) = \begin{cases} 1, & \text{if } m = m(\Delta_{t+1}^{(d)}) \\ 0, & \text{otherwise} \end{cases}.$$

Note that this proof holds regardless of whether the conditioning set of $\tilde{q}$ includes the history $(\mathbf{x}_{t-L+1:t}, \mathbf{z}_{t-L+1:t})$. □

A.3 CRPS AND THE FRACTIONALLY DIFFERENCED OUTPUT SPACE

Proof of Proposition 2.2. Suppose that $C \leq L$, then $\mathbf{h}_t$ is determined by the observed lookback window $\mathbf{x}_{t-L+1:t}$. Let $\widehat{G}_t$ be a predictive CDF for $\Delta_{t+1}^{(d)}$ conditional on this lookback window, fitted using the training data, and define $\widehat{F}_t(x) = \widehat{G}_t(x - \mathbf{h}_t)$. The change of variable $u = x - \mathbf{h}_t$ gives

$$\begin{aligned} \mathrm{CRPS}(\widehat{F}_t, \mathbf{x}_{t+1}) &= \int_{\mathbb{R}} \Big(\widehat{G}_t(x - \mathbf{h}_t) - \mathbb{I}(\mathbf{x}_{t+1} \leq x)\Big)^2 \, dx \\ &= \int_{\mathbb{R}} \Big(\widehat{G}_t(u) - \mathbb{I}(\Delta_{t+1}^{(d)} \leq u)\Big)^2 \, du \\ &= \mathrm{CRPS}(\widehat{G}_t, \Delta_{t+1}^{(d)}), \end{aligned}$$

where $\Delta_{t+1}^{(d)} = \mathbf{x}_{t+1} - \mathbf{h}_t$. This equality holds for every realization and therefore also in expectation under the test distribution. Since $\mathbf{h}_t$ is known given the lookback window, the inverse mapping is $\widehat{G}_t(u) = \widehat{F}_t(u + \mathbf{h}_t)$. Thus, unrestricted conditional forecasts on the original and fractionally differenced domains are in one-to-one correspondence, with each pair having identical expected test CRPS. Consequently, the minimum achievable expected test CRPS is the same on both domains, even when the distribution shift occurs. □

Fixed-location mixtures under distribution shift All distributions below are conditional on the observed lookback window $\mathbf{x}_{t-L+1:t}$. Consider a Gaussian mixture forecasting model with fixed component locations $\{\mu_1, \ldots, \mu_M\}$ and common variance $\beta^2 > 0$. Its predictive CDF $\widehat{F}_t$ varies with t through the mixture weights, while the component locations remain fixed.

To examine the effect of component location shifts, assume that the true test conditional CDF F_t^* is a finite Gaussian mixture. This idealized setting isolates a limitation of fixed-location mixtures.

Proposition A.1. *If F_t^* contains a Gaussian component with positive weight and a location outside $\{\mu_1, \ldots, \mu_M\}$ on a set of lookback windows with positive probability under the test distribution, every forecast from the fixed-location model incurs strictly positive expected excess test CRPS.*

Proof. Let $\widehat{F}_t$ be any forecast from the fixed-location mixture model. For each lookback window satisfying the condition, identifiability of finite Gaussian mixtures implies $\widehat{F}_t \neq F_t^*$, because F_t^* contains a positive-weight component at a location unavailable to the model. Since this condition holds with positive probability, the CRPS regret identity (Gneiting et al., 2007) gives

$$\mathbb{E}\Big[\mathrm{CRPS}(\widehat{F}_t, \mathbf{x}_{t+1}) - \mathrm{CRPS}(F_t^*, \mathbf{x}_{t+1})\Big] = \mathbb{E}\left[\int_{\mathbb{R}} \Big(\widehat{F}_t(x) - F_t^*(x)\Big)^2 \, dx\right] > 0,$$

where the expectations are taken under the test distribution. □

In our model, the inverse transformation maps the fixed component locations θ_m in the fractionally differenced domain to $\theta_m + \mathbf{h}_t$ on the original scale. Therefore, the effective component locations vary with the observed lookback window and time step t, potentially mitigating the limitation described above when $\mathbf{h}_t$ captures the location changes.

A.4 FORMAL STATEMENT AND PROOF OF PROPOSITION 2.3

Proposition A.2. *For notational simplicity, let the lookback window at time step t be denoted by $\mathbf{x} \coloneqq \mathbf{x}_{t-L+1:t}$. For each $m \in \{1, \ldots, M\}$, define the target probability that the outcome at the next time step $t+1$ falls into Voronoi cell $\mathcal{B}_m$ as*

$$\pi_m^*(\mathbf{x}) \coloneqq \Pr\big(\Delta_{t+1}^{(d)} \in \mathcal{B}_m \mid \mathbf{x}\big).$$

For each component k and cell m, define

$$Q_{k \to m} \coloneqq \int_{\mathcal{B}_m} p(\Delta \mid \theta_k, \beta^2) \, d\Delta,$$

which is the probability that a sample generated from component k falls into cell $\mathcal{B}_m$.

Let $\widehat{P}_{\mathcal{B}}(\mathbf{x})$ and $P^*_{\mathcal{B}}(\mathbf{x})$ denote the model-induced and target distributions over the Voronoi cells, respectively. Since the predictive distribution is

$$p(\Delta^{(d)}_{t+1} \mid \mathbf{x}; \phi) = \sum_{k=1}^{M} \pi_k(\mathbf{x}; \phi)\, p(\Delta^{(d)}_{t+1} \mid \theta_k, \beta^2),$$

their probabilities for cell $\mathcal{B}_m$ are given by

$$\widehat{P}_{\mathcal{B},m}(\mathbf{x}) \coloneqq \Pr\Big(\widehat{\Delta}^{(d)}_{t+1} \in \mathcal{B}_m \mid \mathbf{x}\Big) = \sum_{k=1}^{M} \pi_k(\mathbf{x}; \phi) Q_{k\to m}, \quad \textit{and} \quad P^*_{\mathcal{B},m}(\mathbf{x}) \coloneqq \pi^*_m(\mathbf{x}).$$

Moreover, let

$$\delta_M \coloneqq \min_{1\le m<M} |\theta_{m+1} - \theta_m|$$

denote the minimum spacing between adjacent quantization points.

Then, under the Gaussian decoder,

$$\begin{aligned}
&\mathrm{TV}\Big(\widehat{P}_{\mathcal{B}}(\mathbf{x}), P^*_{\mathcal{B}}(\mathbf{x})\Big) \\
\le\ &\mathrm{TV}\big(\pi(\mathbf{x};\phi), \pi^*(\mathbf{x})\big) + \frac{1}{2}\sum_{m=1}^{M} \max\left\{ \sum_{k\neq m} \pi_k(\mathbf{x};\phi) Q_{k\to m},\ \pi_m(\mathbf{x};\phi)(1 - Q_{m\to m}) \right\} \\
\le\ &\frac{1}{\sqrt{2}} \left(\mathbb{E}_{\Delta^{(d)}_{t+1}|\mathbf{x}} \Big[\mathcal{L}(\Delta^{(d)}_{t+1}, \mathbf{x}; \phi) \Big] - H(\pi^*(\mathbf{x})) \right)^{1/2} + 2\Phi\left(-\frac{\delta_M}{2\beta} \right),
\end{aligned}$$

where $\Phi(\cdot)$ denotes the CDF of the standard normal distribution.

Proof. By the definition of the predictive distribution, for each $m \in \{1, \dots, M\}$,

$$\begin{aligned}
\widehat{P}_{\mathcal{B},m}(\mathbf{x}) &= \int_{\mathcal{B}_m} p(\Delta^{(d)}_{t+1} \mid \mathbf{x}; \phi)\, d\Delta^{(d)}_{t+1} \\
&= \sum_{k=1}^{M} \pi_k(\mathbf{x}; \phi) \int_{\mathcal{B}_m} p(\Delta^{(d)}_{t+1} \mid \theta_k, \beta^2)\, d\Delta^{(d)}_{t+1} \\
&= \sum_{k=1}^{M} \pi_k(\mathbf{x}; \phi) Q_{k\to m}.
\end{aligned}$$

It follows that

$$\begin{aligned}
\widehat{P}_{\mathcal{B},m}(\mathbf{x}) - P^*_{\mathcal{B},m}(\mathbf{x}) &= \sum_{k=1}^{M} \pi_k(\mathbf{x}; \phi) Q_{k\to m} - \pi^*_m(\mathbf{x}) \\
&= \pi_m(\mathbf{x}; \phi) - \pi^*_m(\mathbf{x}) + \sum_{k\neq m} \pi_k(\mathbf{x}; \phi) Q_{k\to m} - \pi_m(\mathbf{x}; \phi)(1 - Q_{m\to m}).
\end{aligned}$$

Therefore, by the triangle inequality,

$$\begin{aligned}
\left| \widehat{P}_{\mathcal{B},m}(\mathbf{x}) - P^*_{\mathcal{B},m}(\mathbf{x}) \right| &\le |\pi_m(\mathbf{x}; \phi) - \pi^*_m(\mathbf{x})| \\
&\quad + \left| \sum_{k\neq m} \pi_k(\mathbf{x}; \phi) Q_{k\to m} - \pi_m(\mathbf{x}; \phi)(1 - Q_{m\to m}) \right|.
\end{aligned}$$

Since both $\sum_{k\neq m} \pi_k(\mathbf{x}; \phi) Q_{k\to m}$ and $\pi_m(\mathbf{x}; \phi)(1 - Q_{m\to m})$ are nonnegative, the inequality $|a - b| \le \max\{a, b\}$ for $a, b \ge 0$ gives

$$\begin{aligned}
\left| \widehat{P}_{\mathcal{B},m}(\mathbf{x}) - P^*_{\mathcal{B},m}(\mathbf{x}) \right| &\le |\pi_m(\mathbf{x}; \phi) - \pi^*_m(\mathbf{x})| \\
&\quad + \max\left\{ \sum_{k\neq m} \pi_k(\mathbf{x}; \phi) Q_{k\to m},\ \pi_m(\mathbf{x}; \phi)(1 - Q_{m\to m}) \right\}.
\end{aligned}$$

Summing over m and dividing by two yields:

$$\begin{aligned}\mathrm{TV}\Big(P^*_{\mathcal{B}}(\mathbf{x}), \widehat{P}_{\mathcal{B}}(\mathbf{x})\Big) &\leq \mathrm{TV}\big(\pi(\mathbf{x};\phi), \pi^*(\mathbf{x})\big) \\ &+ \frac{1}{2}\sum_{m=1}^{M} \max\left\{\sum_{k\neq m} \pi_k(\mathbf{x};\phi)Q_{k\to m},\ \pi_m(\mathbf{x};\phi)(1-Q_{m\to m})\right\}.\end{aligned}$$

For each cell $\mathcal{B}_m$, the two quantities inside the maximum represent, respectively, the probability mass entering $\mathcal{B}_m$ from other components and the mass generated by component m that falls outside $\mathcal{B}_m$. Thus, the final term captures the cell-probability error induced by cross-cell decoder leakage.

Moreover, by the definition of the Voronoi cell $\mathcal{B}_m$,

$$\begin{aligned}\mathbb{E}_{\Delta^{(d)}_{t+1}|\mathbf{x}}\Big[\mathcal{L}(\Delta^{(d)}_{t+1}, \mathbf{x};\phi)\Big] &= \mathbb{E}_{\Delta^{(d)}_{t+1}|\mathbf{x}}\Big[-\sum_{m=1}^{M} \mathbb{I}(m(\Delta^{(d)}_{t+1}) = m)\ \log \pi_m(\mathbf{x};\phi)\Big] \\ &= -\sum_{m=1}^{M} \mathbb{E}_{\Delta^{(d)}_{t+1}|\mathbf{x}}\Big[\mathbb{I}(m(\Delta^{(d)}_{t+1}) = m)\Big]\ \log \pi_m(\mathbf{x};\phi) \\ &= -\sum_{m=1}^{M} \pi^*_m(\mathbf{x})\ \log \pi_m(\mathbf{x};\phi).\end{aligned}$$

Then, Pinsker's inequality gives

$$\begin{aligned}\mathrm{TV}\big(\pi(\mathbf{x};\phi), \pi^*(\mathbf{x})\big) &\leq \sqrt{\frac{1}{2}\,\mathrm{KL}\left(\pi^*(\mathbf{x})\|\pi(\mathbf{x};\phi)\right)} \\ &= \frac{1}{\sqrt{2}}\Bigg(\underbrace{-\sum_{m=1}^{M} \pi^*_m(\mathbf{x})\ \log \pi_m(\mathbf{x};\phi)}_{\text{our objective at the population level}} - H(\pi^*(\mathbf{x}))\Bigg)^{1/2} \\ &= \frac{1}{\sqrt{2}}\left(\mathbb{E}_{\Delta^{(d)}_{t+1}|\mathbf{x}}\Big[\mathcal{L}(\Delta^{(d)}_{t+1}, \mathbf{x};\phi)\Big] - H(\pi^*(\mathbf{x}))\right)^{1/2},\end{aligned}$$

where $H(\cdot)$ denotes the entropy. Thus, minimizing the population cross-entropy loss controls the cell-probability estimation error in total variation distance.

Let $\delta_M \coloneqq \min_{1\leq m<M} |\theta_{m+1} - \theta_m|$ denote the minimum spacing between adjacent quantization points. Since a sample generated from component m must deviate from θ_m by at least $\delta_M/2$ to leave its corresponding Voronoi cell, for the Gaussian decoder,

$$1 - Q_{m\to m} \leq 2\Phi\left(-\frac{\delta_M}{2\beta}\right).$$

Since the Voronoi cells form a partition, $\sum_{m=1}^{M} Q_{k\to m} = 1$ for all k. Therefore,

$$\begin{aligned}&\frac{1}{2}\sum_{m=1}^{M} \max\left\{\sum_{k\neq m} \pi_k(\mathbf{x};\phi)Q_{k\to m},\ \pi_m(\mathbf{x};\phi)(1-Q_{m\to m})\right\} \\ \leq\ &\frac{1}{2}\sum_{m=1}^{M} \left\{\sum_{k\neq m} \pi_k(\mathbf{x};\phi)Q_{k\to m} + \pi_m(\mathbf{x};\phi)(1-Q_{m\to m})\right\} \\ =\ &\frac{1}{2}\sum_{k=1}^{M} \pi_k(\mathbf{x};\phi)(1-Q_{k\to k}) + \frac{1}{2}\sum_{m=1}^{M} \pi_m(\mathbf{x};\phi)(1-Q_{m\to m}) \\ =\ &\sum_{m=1}^{M} \pi_m(\mathbf{x};\phi)(1-Q_{m\to m}) \\ \leq\ &2\Phi\left(-\frac{\delta_M}{2\beta}\right).\end{aligned}$$

It shows that, under the Gaussian decoder, cross-cell leakage is controlled by the ratio β/δ_M. Thus, choosing β sufficiently small relative to δ_M tightens the cell-level approximation bound. Note that the proof remains valid regardless of whether the lookback window includes covariates $\mathbf{z}_{t-L+1:t}$. □

A.5 Proof of Proposition 2.4

Proof. Let $\pi(\cdot;\phi)$ be computed via the `softmax` operation with logit functions $f_m(\cdot;\phi)$, i.e.,

$$0 \le \pi_m(\mathbf{x}_{t-L+1:t};\phi) = \frac{\exp\left(f_m(\mathbf{x}_{t-L+1:t};\phi)\right)}{\sum_{l=1}^{M}\exp\left(f_l(\mathbf{x}_{t-L+1:t};\phi)\right)} \le 1$$

for all m. For notational simplicity, let $m^* = m(\Delta_{t+1}^{(d)})$.

If $m = m^*$, then

$$\begin{aligned}
\frac{\partial}{\partial f_m}\mathcal{L}(\Delta_{t+1}^{(d)}, \mathbf{x}_{t-L+1:t};\phi) &= -\frac{1}{\pi_{m^*}(\mathbf{x}_{t-L+1:t};\phi)}\left(\frac{\partial}{\partial f_m}\pi_{m^*}(\mathbf{x}_{t-L+1:t};\phi)\right) \\
&= -\frac{1}{\pi_{m^*}(\mathbf{x}_{t-L+1:t};\phi)}\left(\pi_{m^*}(\mathbf{x}_{t-L+1:t};\phi) - \pi_{m^*}^2(\mathbf{x}_{t-L+1:t};\phi)\right) \\
&= \pi_{m^*}(\mathbf{x}_{t-L+1:t};\phi) - 1 = \pi_m(\mathbf{x}_{t-L+1:t};\phi) - 1.
\end{aligned}$$

If $m \neq m^*$, then

$$\begin{aligned}
\frac{\partial}{\partial f_m}\mathcal{L}(\Delta_{t+1}^{(d)}, \mathbf{x}_{t-L+1:t};\phi) &= -\frac{1}{\pi_{m^*}(\mathbf{x}_{t-L+1:t};\phi)}\left(\frac{\partial}{\partial f_m}\pi_{m^*}(\mathbf{x}_{t-L+1:t};\phi)\right) \\
&= -\frac{1}{\pi_{m^*}(\mathbf{x}_{t-L+1:t};\phi)}\left(-\pi_{m^*}(\mathbf{x}_{t-L+1:t};\phi)\cdot\pi_m(\mathbf{x}_{t-L+1:t};\phi)\right) \\
&= \pi_m(\mathbf{x}_{t-L+1:t};\phi).
\end{aligned}$$

Therefore, for all m, the gradient of the proposed objective in (4) with respect to the logit functions f_m is bounded:

$$\left|\frac{\partial}{\partial f_m}\mathcal{L}(\Delta_{t+1}^{(d)}, \mathbf{x}_{t-L+1:t};\phi)\right| \le 1, \quad \forall m.$$

Note that the proof remains valid regardless of whether the lookback window includes covariates $\mathbf{z}_{t-L+1:t}$. □

The property established in Proposition 2.4 can be beneficial in settings where heavy-tailed observations produce large regression residuals. Under our cross-entropy loss, an extreme target may be assigned to a different class, but the logit-level gradient remains bounded and does not grow with the magnitude of the observation. In contrast, the gradient of an MSE objective grows with the prediction residual magnitude. Our objective can therefore provide more stable optimization in the presence of extreme observations. Nevertheless, we do not claim specialized extreme-tail estimation, as a fixed finite set of support points may represent very rare tail behavior only coarsely.

A.6 Dataset Details

Five electricity price forecasting (EPF) datasets (Lago et al., 2021) from major power markets are used: (1) **BE**: The Belgian day-ahead electricity market, recording hourly day-ahead prices from 2011-01-09 to 2016-12-31; (2) **DE**: The German electricity market, containing hourly day-ahead prices, zonal load forecasts for the TSO Amprion zone, and wind and solar generation forecasts, spanning from 2012-01-09 to 2017-12-31; (3) **FR**: The French day-ahead electricity market, recording hourly day-ahead prices from 2011-01-09 to 2016-12-31; (4) **NP** (Nord Pool): The European power market of the Nordic countries, containing hourly day-ahead electricity prices from 2013-01-01 to 2018-12-24; (5) **PJM**: The Pennsylvania–New Jersey–Maryland market in the United States, providing zonal hourly day-ahead electricity prices for the Commonwealth Edison (COMED) zone, spanning from 2013-01-01 to 2018-12-24. As noted by Lago et al. (2021), electricity prices across

Table 2: **Dataset descriptions and statistics**. '#Exo' denotes the number of exogenous variables, and 'dataset size' is organized in (Train, Validation, Test).

dataset	#Exo	exogenous descriptions	endogenous descriptions	frequency	dataset size
BE	2	Generation, System Load	Belgium's Electricity Price	1 Hour	(36523, 5242, 10460)
DE	2	Wind Power, Amprion Zonal Load	German's Electricity Price	1 Hour	(36523, 5242, 10460)
FR	2	Generation, System Load	France's Electricity Price	1 Hour	(36523, 5242, 10460)
NP	2	Grid Load, Wind Power	Nord Pool Electricity Price	1 Hour	(36523, 5242, 10460)
PJM	2	System Load, Zonal COMED Load	Pennsylvania-New Jersey-Maryland Electricity Price	1 Hour	(36523, 5242, 10460)
ETTh1	6	Power Load Features	Oil Temperature	1 Hour	(8472, 2880, 2857)
ETTh2	6	Power Load Features	Oil Temperature	1 Hour	(8472, 2880, 2857)
ETTm1	6	Power Load Features	Oil Temperature	15 Minutes	(34392, 11520, 11497)
ETTm2	6	Power Load Features	Oil Temperature	15 Minutes	(34392, 11520, 11497)
Weather	20	Meteorological Indicators	CO2 Consumption	10 Minutes	(36719, 5270, 10516)

these markets exhibit "very distinct dynamics," differing in the occurrence and frequency of negative prices, zero prices, and price spikes. These five EPF datasets are tailored to day-ahead forecasting over a relatively short 24-hour horizon.

The following time-series forecasting benchmarks are also used: **ETT** (Zhou et al., 2021) contains a time series of oil temperature and power load collected from electricity transformers between July 2016 and July 2018. Among its variants, ETTm1 and ETTm2 are recorded every 15 minutes, while ETTh1 and ETTh2 are recorded hourly; **Weather**[2] includes 21 meteorological indicators collected every 10 minutes in 2020 from the Weather Station of the Max Planck Biogeochemistry Institute. See Table 2 for descriptions and statistics.

A.7 Implementation Details

- All experiments are conducted on NVIDIA GeForce RTX 5090 GPUs using implementations in `PyTorch`.
- All reported results are averaged over five random seeds.
- When defining Θ_M, $\mathcal{T}$ is constructed as the $M+1$ equally spaced quantile levels in $[0.01, 0.99]$ to exclude extreme values.
- Training is conducted using the Adam optimizer (Kingma & Ba, 2015), with the initial learning rate selected from $\{0.01, 0.001, 0.0001\}$. All methods are trained for 100 epochs with a batch size of 512.
- The `ReduceLROnPlateau` scheduler from `PyTorch` is employed with a patience of 3 and a reduction factor of 0.1. Also, early stopping is applied with a patience of 10 epochs. For all methods, the stopping criterion is the validation loss.
- For model selection, the CRPS metric evaluated on the validation set is used.
- During evaluation, all methods are given observed exogenous inputs over the forecast horizon (Goude et al., 2013). This setting isolates uncertainty quantification performance for the endogenous target from errors in forecasting exogenous inputs.
- Table 3 lists the hyperparameter configurations for the three neural network architectures. These configurations are applied uniformly across all methods and experiments. If necessary, the patch length is set to 24 without overlapping. A parameter-free Reversible Instance Normalization (Kim et al., 2021) is applied as a local normalization.

Detailed baseline implementations The hyperparameter search spaces for our method and all baseline methods are provided in Table 4. We allocate comparable optimization budgets across all methods to ensure a fair comparison.

- **Parametric Prior Mapping** (`PPM`)[3]: A Gaussian kernel for kernel density estimation (KDE) is used with the bandwidth selected from $\{0.1, 0.2, 0.3, 0.4\}$, and 100 samples are generated to approximate the estimated distribution. The KDE-based negative log-likelihood term is weighted by $\alpha \in \{0.1, 0.2\}$ and combined with an auxiliary MSE term to form the final training loss.

[2] https://www.bgc-jena.mpg.de/wetter/
[3] https://github.com/ljl8336/PPM

Table 3: Configurations for each neural network architecture.

model	hyperparameter	value
CrossLinear (Zhou et al., 2025)	hidden dimension	{32}
	factor*	{2}
	α	{1}
	β	{1}
TimeXer (Wang et al., 2024c)	hidden dimension	{32}
	#layers	{2}
	#heads	{2}
	dropout	{0.1}
TSMixer (Chen et al., 2023)	hidden dimension	{32}
	#layers	{2}
	dropout	{0.1}

*The factor controls the dimension of the hidden representation.

- **Winner-takes-all loss** (`WTA`)[4]: The relaxed version of the winner-takes-all loss function introduced in TimeMCL (Cortes et al., 2025) is adopted, with a relaxation constant over $\{0.01, 0.05, 0.1, 0.2\}$. The mean squared error is used as the loss function for each head, which is parameterized by the outputs of CrossLinear[5], TimeXer[6], or TSMixer[7]. The weight parameter of the confidence loss is set to $\beta \in \{0.5, 1.0\}$. For a fair comparison with `WTA`, we follow the procedure originally proposed in Cortes et al. (2025): before computing the metrics, the K hypotheses are resampled with replacement, using their normalized predicted scores as sampling weights.

- **Diffusion loss** (`DDIM`): The loss function is implemented following the formulation used in TimeGrad (Rasul et al., 2021)[8]. The squared L_2-norm is used for the loss function. A linear variance schedule from $\beta_1 \in \{0.0001, 0.0005\}$ to $\beta_N \in \{0.05, 0.1\}$ with $N = 100$ is employed. For accelerated diffusion during inference, DDIM (Song et al., 2021)[9] is applied with 20 sampling steps. The noise prediction network is constructed based on the residual block architecture introduced in Rasul et al. (2021) with $\{2, 8\}$ residual blocks and 8 residual channels. The dilation cycle length is set to 2, and the noise index embedding dimension is set to 32. The hidden states are parameterized by the outputs of CrossLinear, TimeXer, or TSMixer, meaning that the LSTM-based RNN in the original model is replaced by these neural network architectures.

- **Continuous Ranked Probability Score** (`CRPS`)[10]: The conditional quantile function is modeled with a linear isotonic spline, which results in a closed-form scoring rule-based objective function (Gasthaus et al., 2019). The parameters of the linear isotonic spline are parameterized by the outputs of CrossLinear, TimeXer, or TSMixer. The number of knot points is set to $\{10, 20, 50\}$.

- **Maximum Likelihood Estimation** (`MLE`): The output distribution is a Gaussian distribution with trainable mean and standard deviation parameters. The mean and standard deviation are parameterized by the outputs of CrossLinear, TimeXer, or TSMixer. The standard deviation is parameterized with the `softplus` function, following the original implementation of DeepAR (Salinas et al., 2020).

[4] https://github.com/Victorletzelter/timeMCL
[5] https://github.com/mumiao2000/CrossLinear
[6] https://github.com/thuml/TimeXer
[7] https://github.com/google-research/google-research/blob/master/tsmixer
[8] https://github.com/zalandoresearch/pytorch-ts/blob/master/pts/model/time_grad
[9] https://github.com/ermongroup/ddim
[10] https://github.com/Optim-Lab/CryptoVAE

Table 4: Hyperparameter search space for each method.

method	hyperparameter	search space
Common	(initial) learning rate	{0.01, 0.001, 0.0001}
CLS-PF	d	{0.3, 0.7, 1.0, 1.3, 1.7, 2.0}
PPM	α h (bandwidth)	{0.1, 0.2} {0.1, 0.2, 0.3, 0.4}
WTA	β ϵ	{0.5, 1.0} {0.01, 0.05, 0.1, 0.2}
DDIM	β_1 β_N #blocks	{0.0001, 0.0005} {0.05, 0.1} {2, 8}
CRPS	#knots	{10, 20, 50}
MLE	-	-

A.8 Evaluation Metrics

Let $\left\{(\hat{\mathbf{x}}_1^{(k)}, \hat{\mathbf{x}}_2^{(k)}, \cdots, \hat{\mathbf{x}}_T^{(k)})\right\}_{k=1}^K$ be generated K trajectories of the endogenous target variable, and $(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T)$ be observed time series.

- **CRPS** (Continuous Ranked Probability Score) (Matheson & Winkler, 1976) is defined as

$$\text{CRPS} \coloneqq \frac{1}{|\mathcal{A}|} \sum_{\alpha \in \mathcal{A}} \frac{\sum_{t=1}^T 2(\mathbf{x}_t - Q_t(\alpha))\big(\alpha - \mathbb{I}\big(\mathbf{x}_t < Q_t(\alpha)\big)\big)}{\sum_{t=1}^T |\mathbf{x}_t|},$$

  where $Q_t(\alpha)$ is the α-level quantile of $\left\{\hat{\mathbf{x}}_t^{(k)}\right\}_{k=1}^K$ at timestep t, and $\mathcal{A} = \{0.05, 0.1, 0.15, \cdots, 0.95\}$. Note that the normalized version of CRPS is employed.

- **QICE** (Quantile Interval Calibration Error) (Han et al., 2022) is defined as

$$\text{QICE} \coloneqq \frac{1}{N} \sum_{n=1}^N \left| R_n - \frac{1}{N} \right|,$$

  where $R_n \coloneqq \frac{1}{T} \sum_{t=1}^T \mathbb{I}\big(Q_t(n_{\text{low}}) < \mathbf{x}_t \leq Q_t(n_{\text{high}})\big)$, and $Q_t(n_{\text{low}})$ and $Q_t(n_{\text{high}})$ denote the lower and upper boundaries of the nth quantile interval at timestep t, which are calculated from $\left\{\hat{\mathbf{x}}_t^{(k)}\right\}_{k=1}^K$, respectively. $N = 10$ equispaced quantile intervals is used.

- **TV** (Total Variation) (Cortes et al., 2025) is defined as

$$\text{TV} \coloneqq \frac{1}{K} \sum_{k=1}^K \left(\frac{1}{T-1} \sum_{t=2}^T \left| \hat{\mathbf{x}}_t^{(k)} - \hat{\mathbf{x}}_{t-1}^{(k)} \right| \right).$$

- **MSE** (Mean Squared Error) and **MAE** (Mean Absolute Error) are defined as

$$\text{MSE} \coloneqq \frac{1}{T} \sum_{t=1}^T \left(\mathbf{x}_t - \bar{\mathbf{x}}_t\right)^2 \quad \text{and} \quad \text{MAE} \coloneqq \frac{1}{T} \sum_{t=1}^T \left|\mathbf{x}_t - \bar{\mathbf{x}}_t\right|$$

  where $\bar{\mathbf{x}}_t$ denotes the closed-form conditional expectation for our proposed method, while for the baseline probabilistic forecasters it is computed by averaging multiple trajectories as $\bar{\mathbf{x}}_t = \frac{1}{K} \sum_{k=1}^K \hat{\mathbf{x}}_t^{(k)}$ at timestep t, respectively. Point estimation forecasters directly use the model output as $\bar{\mathbf{x}}_t$.

A.9 Detailed Experimental Results

Effect of fractional differencing on distribution shift The effect of fractional differencing is quantified using the maximum mean discrepancy (MMD) with an RBF kernel and a bandwidth fixed

Table 5: **Alleviating distribution shifts**. Distribution distances (MMD and Wasserstein-1 distance) between the training and test sets. Due to computational complexity, up to $5,000$ observations are randomly sampled to estimate these distances.

metric	dataset	$d=0$	$d=0.3$	$d=0.7$	$d=1.0$	$d=1.3$	$d=1.7$	$d=2.0$
MMD	BE	0.115	0.008	0.001	0.000	0.000	0.000	0.000
	DE	0.004	0.002	0.001	0.001	0.001	0.001	0.001
	FR	0.057	0.008	0.003	0.003	0.003	0.003	0.003
	NP	0.255	0.103	0.006	0.005	0.005	0.005	0.005
	PJM	0.004	0.000	0.000	0.000	0.000	0.000	0.000
	ETTh1	0.444	0.309	0.032	0.021	0.017	0.016	0.016
	ETTh2	0.224	0.092	0.002	0.001	0.001	0.001	0.001
	ETTm1	0.479	0.398	0.042	0.021	0.020	0.020	0.020
	ETTm2	0.256	0.166	0.015	0.001	0.000	0.000	0.000
	Weather	0.001	0.000	0.000	0.000	0.000	0.000	0.000
W1	BE	0.513	0.137	0.068	0.064	0.054	0.058	0.064
	DE	0.151	0.093	0.064	0.058	0.057	0.060	0.058
	FR	0.327	0.155	0.131	0.135	0.144	0.157	0.163
	NP	1.238	0.630	0.197	0.177	0.184	0.192	0.194
	PJM	0.141	0.068	0.073	0.071	0.069	0.068	0.070
	ETTh1	1.282	0.950	0.327	0.289	0.269	0.256	0.254
	ETTh2	0.987	0.620	0.090	0.052	0.061	0.075	0.088
	ETTm1	1.319	1.141	0.360	0.279	0.269	0.264	0.264
	ETTm2	1.036	0.842	0.217	0.048	0.040	0.046	0.047
	Weather	0.065	0.062	0.036	0.022	0.024	0.020	0.016

Table 6: **Point estimation performance** averaged over three neural network architectures. For our method, point estimates are obtained using the closed-form of (6). The best results are highlighted in **bold**, while the second-best results are underlined. Lower is better.

	RMSE						MAE					
dataset	CLS	PPM	WTA	DDIM	CRPS	MLE	CLS	PPM	WTA	DDIM	CRPS	MLE
BE	.623	.761	.708	.666	**.616**	.652	**.249**	.303	.321	.297	.260	.282
DE	**.566**	.890	.598	.753	.579	.602	**.348**	.440	.376	.504	.360	.373
FR	**.637**	.994	.732	.681	.639	.648	**.194**	.295	.226	.249	.208	.209
NP	**.458**	.915	.520	.569	.482	.480	**.235**	.344	.288	.349	.256	.253
PJM	.263	.393	.281	.289	.264	**.259**	.162	.180	.169	.185	.163	**.160**
ETTh1	**.172**	.174	**.172**	.200	.179	.175	**.130**	.133	.131	.152	.136	.133
ETTh2	**.274**	.296	.298	.381	.297	.290	**.198**	.221	.219	.259	.213	.210
ETTm1	**.101**	.110	.106	.145	.106	.103	**.073**	.078	.077	.108	.077	.074
ETTm2	.199	.190	.198	.267	**.161**	.167	.121	.117	.129	.179	**.106**	.108
Weather	**.019**	.021	.026	.025	**.019**	.028	**.012**	.014	.018	.016	**.012**	.019

at 1, together with the Wasserstein-1 (W1) distance between the training and test distributions. As shown in Table 5, fractional differencing ($d > 0$) consistently reduces both distances relative to the original time series ($d = 0$) across all datasets, providing empirical evidence that it alleviates distribution shift in the target variable.

Inference: point estimation in closed-form Under the Gaussian decoder, the point estimate can be computed in closed form:

$$\begin{aligned}\mathbb{E}\big[\mathbf{x}_{t+1} \mid \mathbf{x}_{t-L+1:t}; \phi\big] &= \mathbb{E}\big[\Delta^{(d)}_{t+1} \mid \mathbf{x}_{t-L+1:t}; \phi\big] + \sum_{k=1}^{C} (-1)^{k+1} \binom{d}{k} \mathbf{x}_{t+1-k} \\ &= \sum_{m=1}^{M} \theta_m \, \pi_m\big(\mathbf{x}_{t-L+1:t}; \phi\big) + \sum_{k=1}^{C} (-1)^{k+1} \binom{d}{k} \mathbf{x}_{t+1-k}. \qquad (6)\end{aligned}$$

This conditional expectation also holds for any decoder whose location parameter equals its mean. Table 6 compares the point estimation performance of our closed-form conditional expectation with that of baseline approaches. Our method achieves the best performance on 7 and 8 datasets in terms

Table 7: Holm-adjusted p-values for paired comparisons between CLS−PF and each baseline across all datasets, with results averaged across neural network architectures. The Holm correction is applied separately for each evaluation metric. Asterisks indicate statistical significance at $\alpha = 0.05$.

baseline	Holm-adjusted p-value	
	CRPS	RMSE
PPM	0.0049*	0.0049*
WTA	0.0049*	0.0049*
DDIM	0.0049*	0.0049*
CRPS	0.0137*	0.0967
MLE	0.0186*	0.0098*

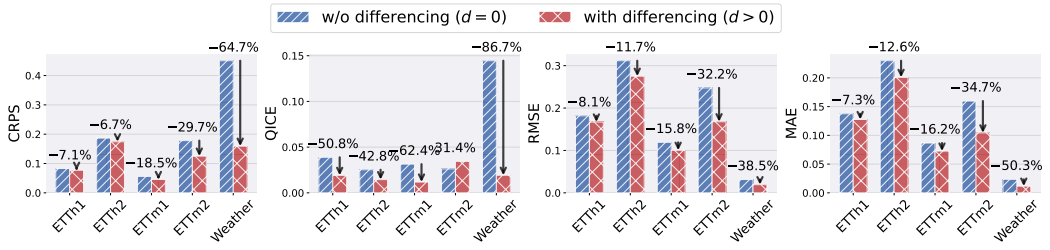


Figure 6: **Effect of fractional differencing in our model**. Results are averaged across three neural architectures. Lower values are better; more negative relative percentage changes indicate larger improvements.

of RMSE and MAE, respectively. These results demonstrate that our method provides competitive point forecasts without generating and averaging 100 trajectories to estimate the predictive mean, as required by sampling-based alternatives.

Statistical significance To assess whether the performance gains in Table 1 are consistent across datasets, we conduct one-sided paired Wilcoxon signed-rank tests comparing CLS−PF with each baseline after averaging the results across neural network architectures. For each comparison, the alternative hypothesis is that CLS−PF achieves a lower error than the corresponding baseline. As shown in Table 7, all Holm-adjusted p-values for both CRPS and RMSE are below the significance level of 0.05, except for the RMSE comparison with the CRPS baseline. These results provide statistical evidence that CLS−PF achieves lower CRPS than every baseline while maintaining competitive RMSE performance.

Ablation study: effect of fractional differencing Figure 6 compares forecasting performance with and without fractional differencing on the remaining five datasets from the ETT and Weather benchmarks. Relative to the raw time series ($d = 0$), fractional differencing ($d > 0$) improves performance for all but one dataset-metric combination. Together with the results in the main text, these findings suggest that defining the output in the fractionally differenced domain makes the finite mixture representation more effective.

Table 8 compares the original baseline methods with variants trained in the same fractionally differenced domain across all datasets and metrics. For each dataset-backbone combination, the baseline variants use the same differencing order d as our method, while the remaining hyperparameters listed in Table 4 are selected through a full grid search on the validation set. Fractional differencing improves baseline performance in many settings, although DDIM benefits primarily on the ETT and Weather datasets. Our model shows improvements on every dataset.

Relative to training on the raw time series ($d = 0$), the average CRPS changes with fractional differencing ($d > 0$) are -19.14%, -1.52%, $+5.19\%$, $+44.55\%$, -5.05%, and $+16.43\%$ for CLS−PF, PPM, WTA, DDIM, CRPS, and MLE, respectively, where negative values indicate improvements. Excluding the Weather dataset, the corresponding changes are -14.08%, -5.77%, -5.29%, $+48.24\%$, -4.50%, and -4.58% for CLS−PF, PPM, WTA, DDIM, CRPS, and MLE, respectively. Therefore, these results indicate that our method achieves the largest average CRPS reduction. These results

Table 8: **Comparison with baseline variants**. Results are averaged across three neural architectures. Parentheses report relative percentage changes from the corresponding original baseline; negative values shown in red indicate improvements. Best and second-best results are shown in **bold** and underlined, respectively.

	CRPS					
dataset	CLS	PPM	WTA	DDIM	CRPS	MLE
BE	**.287**	.398	.365	.447	.300	.333
	(−6.7)	(+15.0)	(−5.7)	(+29.3)	(−2.7)	(−0.8)
DE	**.419**	.474	.449	1.515	.427	.430
	(−8.4)	(−13.8)	(−4.8)	(+132.6)	(−4.0)	(−5.1)
FR	**.245**	.287	.270	.636	.253	.257
	(−10.3)	(−16.6)	(−3.5)	(+104.1)	(−1.6)	(−1.2)
NP	**.220**	.288	.274	.392	.234	.225
	(−9.7)	(−10.6)	(−4.6)	(+8.0)	(−4.5)	(−3.8)
PJM	**.299**	.336	.320	.975	.302	.303
	(−29.6)	(−2.2)	(+1.7)	(+183.9)	(−0.2)	(+2.1)
ETTh1	**.077**	.085	**.077**	.082	.082	**.077**
	(−7.1)	(+4.3)	(−5.2)	(−9.0)	(−1.6)	(−5.8)
ETTh2	.174	.182	.179	.252	.168	**.165**
	(−6.7)	(−5.4)	(−3.3)	(+15.7)	(−8.7)	(−8.6)
ETTm1	.046	.048	.047	.056	.046	**.045**
	(−18.5)	(−4.8)	(−2.0)	(−19.6)	(−2.9)	(−2.7)
ETTm2	.125	.119	.114	.185	**.101**	**.101**
	(−29.7)	(−17.8)	(−20.2)	(−10.8)	(−14.3)	(−15.3)
Weather	.159	.286	.503	.253	**.156**	.897
	(−64.7)	(+36.7)	(+99.5)	(+11.3)	(−10.0)	(+205.5)

Table 9: **Total variation** metric averaged over three neural architectures. The best results are highlighted in **bold**, while the second-best results are underlined. Lower is better.

dataset	CLS	PPM	WTA	DDIM	CRPS	MLE
BE	**0.185**	0.270	0.354	0.372	0.203	0.277
DE	0.232	0.238	0.235	0.436	**0.220**	0.236
FR	**0.164**	0.230	0.193	0.263	0.179	0.197
NP	**0.145**	0.180	0.173	0.271	0.153	0.156
PJM	**0.119**	0.133	0.137	0.199	0.129	0.128
ETTh1	0.050	0.050	**0.043**	0.086	**0.043**	0.055
ETTh2	0.081	0.124	0.096	0.149	**0.078**	0.087
ETTm1	0.026	**0.017**	0.020	0.036	0.020	0.025
ETTm2	0.039	**0.021**	0.026	0.084	0.022	0.029
Weather	**0.002**	**0.002**	0.009	0.008	0.003	0.035

suggest that fractional differencing can improve baseline performance as a preprocessing step, but it is particularly effective in our model, supporting its role as an integral part of our predictive model rather than a separate preprocessing component.

The particularly strong performance of the proposed objective on the five EPF datasets, characterized by highly volatile electricity prices (Misiorek & Weron, 2005; Karakatsani & Bunn, 2010), suggests that it captures uncertainty in the temporal changes represented by fractional differencing more effectively than the baseline objectives in volatile settings. These findings indicate that the performance gains cannot be explained by fractional differencing alone and that the proposed objective also contributes to distribution learning.

Total variation Table 9 evaluates the temporal smoothness of the generated trajectories using the TV metric, defined as the average absolute change between consecutive forecast values. Lower TV indicates fewer abrupt fluctuations over the forecast horizon. Our method achieves competitive results overall, suggesting that it generates trajectories without excessive temporal fluctuations.

Inference runtime To assess the computational efficiency of each method, we measure the runtime, in seconds, required to generate 100 trajectories for the five EPF datasets. Figure 7 reports the runtime distributions aggregated across these datasets and neural network architectures. The red dashed lines

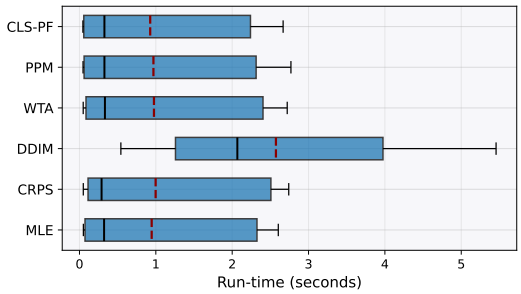
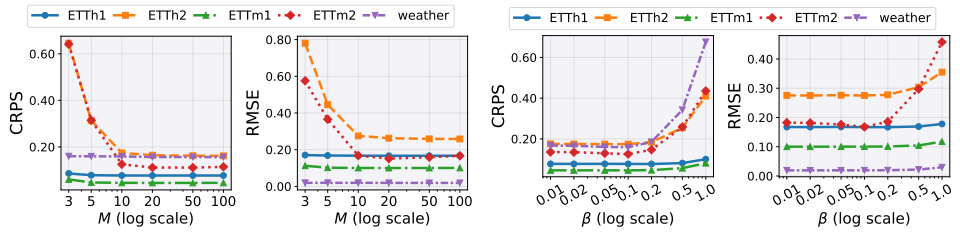
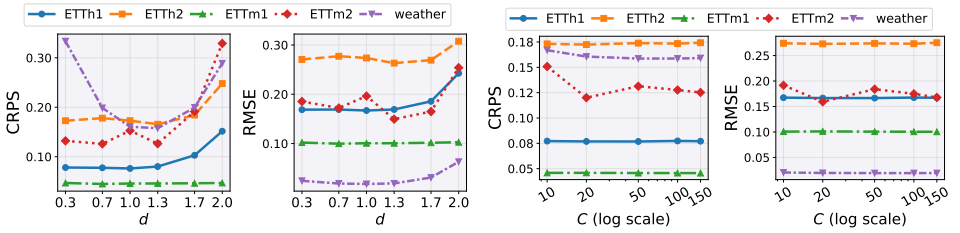


Figure 7: **Inference runtime**. Boxplots of the time(seconds) required to generate 100 trajectories, aggregated across the five EPF datasets and neural network architectures. Red dashed lines indicate mean runtimes.



(a) the number of bins (b) the decoder standard deviation



(c) the differencing order (d) the truncation lag

Figure 8: **Sensitivity analyses** of CRPS and RMSE on the remaining five datasets from the ETT and Weather benchmarks, with results averaged across three neural network architectures.

indicate the mean runtimes. Most methods require approximately one second on average, whereas `DDIM` takes about 2.5 times longer. Our method generates trajectories in parallel through broadcasted computations. In contrast, although `DDIM` employs an accelerated diffusion sampler, it still requires multiple sequential denoising steps at inference time, resulting in higher computational cost.

A.9.1 Sensitivity Analysis

Figure 8 presents the corresponding sensitivity analyses on the remaining five datasets from the ETT and Weather benchmarks. Overall, the results are consistent with those reported in the main text.

Sensitivity analysis 1: Number of bins As shown in Figure 8(a), both CRPS and RMSE remain stable for $M \geq 10$, indicating that $M = 10$ provides sufficient discretization resolution. In particular, performance on ETTh1, ETTm1, and Weather remains stable across the entire range $M \in \{3, 5, 10, 20, 50, 100\}$, indicating robustness to the choice of M.

Sensitivity analysis 2: Decoder standard deviation Figure 8(b) shows that both CRPS and RMSE remain relatively stable for $\beta \leq 0.2$ but increase at larger values of β, where $\beta \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0\}$. Together with the main text results, this finding indicates that performance is broadly robust to β up to 0.2 but can deteriorate beyond this range.

Sensitivity analysis 3: Differencing order Across $d \in \{0.3, 0.7, 1.0, 1.3, 1.7, 2.0\}$, CRPS generally increases slightly with d, as shown in Figure 8(c). On the Weather dataset, however, CRPS reaches its lowest values at $d \in \{1.0, 1.3\}$ before increasing at larger values. RMSE remains relatively stable on most datasets but tends to increase at larger differencing orders. Together with the main text results, these findings indicate that forecasting performance is generally stable for $d \in \{0.3, 0.7, 1.0, 1.3\}$, whereas larger differencing orders can degrade performance on some datasets.

Sensitivity analysis 4: Truncation lag Figure 8(d) evaluates truncation lag C in $\{10, 20, 50, 100, 150\}$. Forecasting performance remains highly stable across truncation lags and evaluation metrics, except on the ETTm2 dataset. Taken together with the main text results, these findings indicate that the benefits of time-varying mixture component locations obtained by fractional differencing do not require a long truncation window.

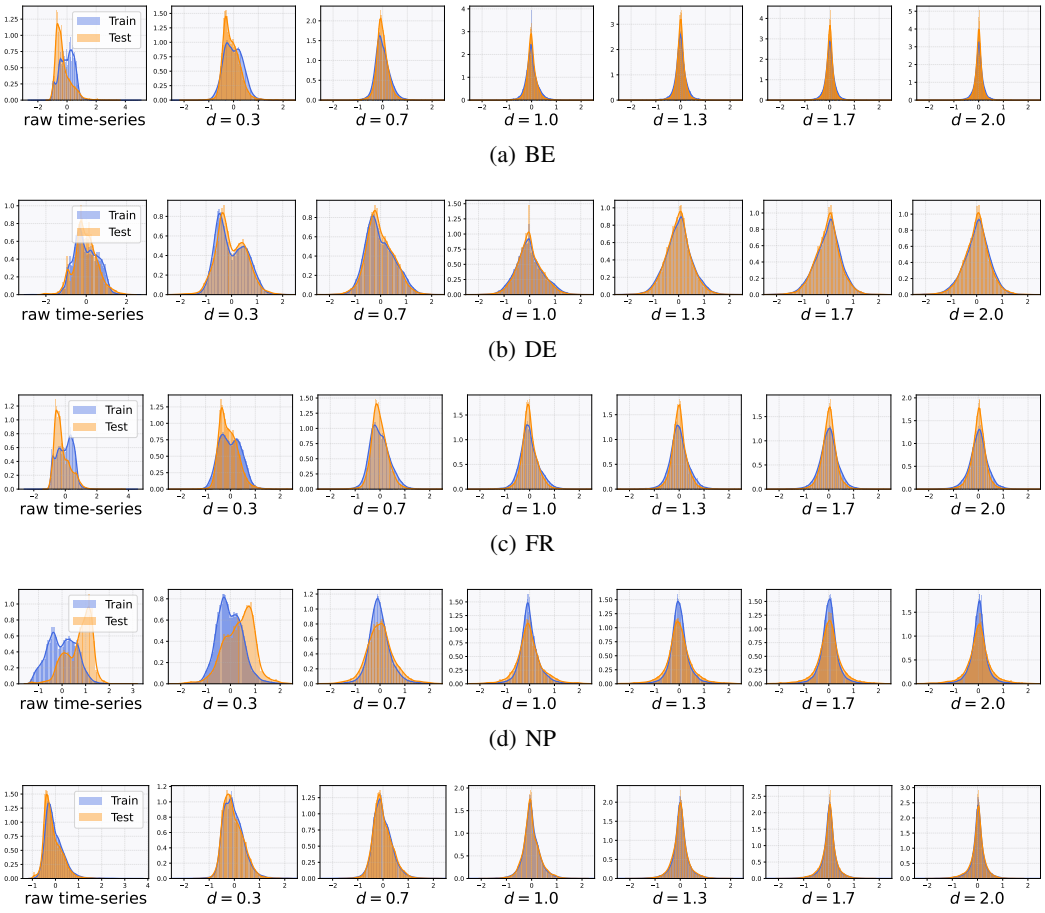
Table 10: **Probabilistic forecasting performance** for each neural network architecture. The best results are highlighted in **bold**, while the second-best results are underlined. Lower is better. Standard errors are shown in parentheses.

	CRPS						QICE					
dataset	CLS	PPM	WTA	DDIM	CRPS	MLE	CLS	PPM	WTA	DDIM	CRPS	MLE
	CrossLinear (Zhou et al., 2025)											
BE	**.299**(.001)	.368(.005)	.371(.009)	.348(.001)	.339(.004)	.358(.008)	.022(.000)	.027(.002)	.027(.003)	.014(.001)	.027(.001)	**.012**(.001)
DE	**.429**(.001)	.550(.007)	.481(.003)	.651(.028)	.473(.002)	.481(.003)	**.008**(.000)	.086(.002)	.028(.001)	.027(.007)	.037(.001)	.028(.001)
FR	**.253**(.002)	.373(.010)	.297(.010)	.324(.002)	.270(.002)	.268(.004)	.011(.000)	.012(.002)	.018(.001)	.028(.001)	.018(.000)	**.009**(.001)
NP	**.224**(.001)	.319(.011)	.287(.007)	.366(.008)	.251(.004)	.234(.004)	**.011**(.000)	.090(.004)	.031(.001)	.054(.002)	.027(.001)	.012(.001)
PJM	.312(.001)	.341(.001)	.322(.003)	.348(.001)	.307(.002)	**.305**(.001)	**.009**(.001)	.097(.001)	.030(.001)	.027(.001)	.029(.001)	.021(.000)
ETTh1	**.078**(.000)	.081(.001)	.082(.001)	.093(.002)	.083(.001)	.083(.001)	**.017**(.000)	.033(.002)	.035(.002)	.051(.003)	.036(.002)	.030(.001)
ETTh2	**.180**(.001)	.182(.002)	.191(.003)	.245(.011)	.193(.003)	.183(.003)	.022(.001)	.018(.003)	.051(.002)	.026(.002)	.035(.002)	**.016**(.002)
ETTm1	**.045**(.000)	.050(.000)	.050(.001)	.076(.003)	.048(.000)	.046(.000)	.011(.000)	.065(.003)	.019(.000)	.058(.003)	.025(.000)	**.007**(.001)
ETTm2	.135(.001)	.134(.002)	.142(.005)	.241(.014)	**.107**(.001)	**.107**(.002)	.045(.000)	.096(.003)	.030(.002)	.030(.002)	.025(.000)	**.010**(.001)
Weather	**.162**(.000)	.245(.007)	.253(.012)	.257(.003)	.205(.001)	.239(.007)	**.020**(.000)	.120(.005)	.074(.002)	.083(.004)	.059(.001)	.044(.002)
	TimeXer (Wang et al., 2024c)											
BE	**.274**(.001)	.346(.005)	.384(.005)	.345(.001)	.283(.001)	.332(.011)	.016(.001)	.014(.002)	.025(.002)	.014(.001)	.020(.001)	**.008**(.001)
DE	**.405**(.001)	.515(.008)	.490(.006)	.703(.021)	.431(.005)	.447(.009)	**.011**(.001)	.061(.002)	.032(.001)	.054(.002)	.025(.001)	.015(.001)
FR	**.237**(.001)	.333(.009)	.272(.004)	.309(.001)	.243(.001)	.256(.001)	**.006**(.001)	.023(.001)	.025(.001)	.025(.001)	.017(.001)	.008(.001)
NP	**.221**(.001)	.342(.010)	.308(.003)	.390(.030)	.240(.007)	.238(.002)	**.012**(.001)	.110(.001)	.033(.000)	.043(.006)	.023(.003)	**.012**(.003)
PJM	.289(.002)	.333(.003)	.312(.004)	.342(.001)	**.285**(.001)	.287(.002)	**.011**(.000)	.079(.002)	.033(.001)	.024(.001)	.027(.000)	.016(.002)
ETTh1	**.077**(.000)	.081(.000)	.080(.001)	.085(.002)	.081(.001)	.081(.001)	.019(.001)	.022(.003)	.020(.001)	.017(.002)	.030(.003)	**.016**(.001)
ETTh2	**.179**(.001)	.217(.006)	.186(.005)	.204(.007)	.180(.004)	**.179**(.005)	**.011**(.001)	.032(.003)	.022(.003)	.026(.005)	.021(.002)	.022(.003)
ETTm1	**.045**(.000)	.051(.000)	.048(.000)	.062(.001)	.047(.000)	.047(.000)	**.008**(.001)	.044(.002)	.024(.001)	.041(.003)	.026(.002)	.014(.001)
ETTm2	**.124**(.002)	.196(.009)	.177(.016)	.217(.023)	.151(.010)	.150(.011)	.031(.001)	.132(.002)	.042(.011)	**.024**(.003)	.030(.004)	.029(.009)
Weather	**.159**(.000)	.208(.005)	.245(.004)	.237(.006)	.160(.001)	.383(.023)	**.019**(.000)	.091(.002)	.075(.001)	.035(.004)	.038(.002)	.058(.007)
	TSMixer (Chen et al., 2023)											
BE	**.288**(.001)	.323(.002)	.406(.010)	.345(.001)	.303(.004)	.317(.003)	.024(.001)	.033(.001)	**.010**(.001)	.017(.000)	.033(.001)	.013(.002)
DE	**.423**(.002)	.583(.001)	.445(.005)	.600(.032)	.430(.004)	.431(.004)	**.009**(.000)	.086(.001)	.060(.002)	.030(.008)	.040(.002)	.024(.001)
FR	**.245**(.002)	.326(.014)	.270(.002)	.303(.002)	.258(.002)	.255(.002)	**.007**(.000)	.032(.004)	.045(.001)	.026(.002)	.024(.002)	.011(.001)
NP	**.216**(.001)	.307(.001)	.268(.002)	.332(.004)	.245(.009)	.230(.004)	**.015**(.001)	.089(.002)	.053(.002)	.048(.002)	.034(.000)	.020(.001)
PJM	**.295**(.003)	.356(.001)	.308(.001)	.340(.002)	.314(.006)	.299(.003)	**.011**(.001)	.077(.003)	.048(.001)	.024(.000)	.040(.000)	.023(.002)
ETTh1	**.077**(.000)	.081(.000)	.083(.001)	.092(.002)	.086(.001)	.081(.000)	**.021**(.001)	.042(.001)	.068(.002)	.041(.002)	.059(.001)	.040(.001)
ETTh2	**.164**(.001)	.177(.002)	.180(.003)	.205(.012)	.179(.001)	.179(.002)	**.010**(.000)	.061(.003)	.019(.003)	.023(.005)	.057(.001)	.041(.002)
ETTm1	.047(.000)	.049(.000)	.047(.000)	.071(.004)	.048(.000)	**.046**(.000)	.016(.001)	.043(.001)	.030(.001)	.049(.005)	.040(.001)	**.015**(.002)
ETTm2	.116(.001)	.104(.002)	.110(.002)	.163(.017)	**.096**(.002)	.099(.002)	.028(.001)	.065(.003)	.043(.002)	.028(.005)	.031(.002)	**.017**(.001)
Weather	**.156**(.003)	.174(.001)	.258(.007)	.189(.005)	**.156**(.003)	.260(.011)	**.019**(.000)	.080(.002)	.085(.002)	.057(.005)	.050(.002)	.026(.003)

Table 11: **Point estimation performance** for each neural network architecture. The best results are highlighted in **bold**, while the second-best results are underlined. Lower is better. Standard errors are shown in parentheses.

	RMSE						MAE					
dataset	CLS	PPM	WTA	DDIM	CRPS	MLE	CLS	PPM	WTA	DDIM	CRPS	MLE
	CrossLinear (Zhou et al., 2025)											
BE	.644(.002)	.695(.010)	.687(.014)	.666(.001)	**.642**(.008)	.680(.009)	**.260**(.001)	.303(.004)	.305(.006)	.298(.001)	.282(.003)	.302(.006)
DE	**.585**(.002)	1.292(.354)	.616(.008)	.759(.021)	.597(.003)	.614(.003)	**.361**(.001)	.471(.036)	.389(.003)	.510(.020)	.379(.002)	.391(.002)
FR	**.643**(.004)	1.340(.192)	.851(.056)	.707(.003)	.718(.007)	.726(.028)	**.201**(.001)	.333(.017)	.243(.009)	.259(.001)	.225(.002)	.224(.005)
NP	**.470**(.002)	1.216(.325)	.522(.010)	.564(.004)	.516(.008)	.499(.008)	**.240**(.001)	.373(.037)	.291(.007)	.347(.004)	.269(.004)	.259(.004)
PJM	.272(.002)	.276(.004)	.292(.003)	.293(.002)	.272(.002)	**.266**(.002)	.169(.001)	.167(.001)	.175(.002)	.187(.001)	.167(.001)	**.165**(.001)
ETTh1	**.168**(.001)	.173(.001)	.174(.001)	.206(.002)	.182(.003)	.176(.003)	**.129**(.001)	.132(.001)	.134(.001)	.154(.002)	.138(.002)	.135(.002)
ETTh2	**.283**(.002)	.290(.003)	.296(.005)	.371(.014)	.321(.005)	.306(.008)	**.207**(.001)	.215(.002)	.222(.005)	.286(.012)	.231(.003)	.219(.003)
ETTm1	**.100**(.000)	.107(.001)	.110(.001)	.157(.006)	.108(.001)	.103(.000)	**.072**(.000)	.077(.000)	.079(.001)	.118(.005)	.078(.001)	.074(.000)
ETTm2	.185(.002)	.174(.004)	.205(.005)	.290(.011)	**.153**(.001)	.156(.003)	.114(.002)	.111(.001)	.131(.003)	.199(.009)	**.100**(.000)	.101(.002)
Weather	**.020**(.000)	.023(.001)	.027(.001)	.026(.000)	.021(.000)	.025(.000)	**.012**(.000)	.015(.000)	.019(.001)	.018(.000)	.014(.000)	.017(.000)
	TimeXer (Wang et al., 2024c)											
BE	.616(.001)	.777(.065)	.710(.011)	.666(.002)	**.595**(.001)	.653(.012)	**.239**(.001)	.328(.014)	.325(.005)	.296(.001)	.242(.001)	.279(.008)
DE	**.540**(.001)	.675(.020)	.631(.008)	.798(.029)	.567(.005)	.610(.009)	**.334**(.001)	.407(.009)	.395(.004)	.535(.018)	.352(.003)	.370(.007)
FR	**.600**(.002)	.844(.027)	.650(.008)	.678(.003)	.605(.011)	.613(.003)	**.187**(.001)	.292(.008)	.219(.003)	.246(.001)	.195(.001)	.201(.001)
NP	**.447**(.002)	.975(.318)	.563(.004)	.606(.024)	.459(.006)	.482(.004)	**.233**(.001)	.369(.046)	.315(.003)	.374(.020)	.250(.004)	.255(.002)
PJM	.258(.002)	.610(.325)	.284(.005)	.286(.001)	**.250**(.001)	.251(.001)	.156(.001)	.192(.020)	.169(.002)	.184(.001)	**.153**(.000)	.155(.001)
ETTh1	**.168**(.001)	.177(.001)	.174(.001)	.191(.003)	.176(.001)	.178(.002)	**.127**(.001)	.134(.001)	.132(.001)	.145(.003)	.132(.001)	.134(.002)
ETTh2	**.282**(.002)	.330(.009)	.309(.008)	.313(.012)	.292(.004)	.286(.008)	**.206**(.002)	.249(.006)	.221(.005)	.237(.011)	.211(.004)	.209(.006)
ETTm1	**.099**(.000)	.117(.001)	.106(.000)	.129(.001)	.104(.001)	.105(.000)	**.071**(.000)	.082(.000)	.077(.001)	.095(.001)	.075(.001)	.075(.000)
ETTm2	**.164**(.004)	.258(.010)	.238(.016)	.259(.022)	.195(.006)	.207(.012)	**.102**(.001)	.150(.005)	.158(.013)	.181(.017)	.130(.003)	.132(.008)
Weather	.019(.000)	.021(.000)	.025(.000)	.026(.001)	**.018**(.000)	.032(.001)	**.012**(.000)	.014(.000)	.017(.000)	.017(.000)	**.012**(.000)	.023(.001)
	TSMixer (Chen et al., 2023)											
BE	**.610**(.001)	.810(.121)	.726(.011)	.665(.001)	**.610**(.005)	.624(.003)	**.247**(.001)	.279(.003)	.332(.010)	.297(.001)	.256(.003)	.266(.002)
DE	**.545**(.006)	.703(.003)	.546(.005)	.703(.029)	.572(.007)	.580(.006)	**.342**(.002)	.443(.001)	.343(.003)	.468(.021)	.351(.004)	.357(.003)
FR	.674(.008)	.797(.049)	.694(.022)	.659(.005)	**.594**(.004)	.604(.005)	**.194**(.002)	.259(.011)	.218(.003)	.242(.002)	.203(.001)	.202(.001)
NP	**.451**(.002)	.554(.035)	.476(.008)	.538(.011)	.470(.012)	.459(.004)	**.231**(.001)	.290(.005)	.258(.002)	.325(.007)	.250(.007)	.245(.003)
PJM	**.254**(.003)	.291(.001)	.268(.003)	.287(.002)	.269(.005)	.259(.003)	**.158**(.002)	.181(.001)	.162(.001)	.183(.001)	.168(.003)	.161(.002)
ETTh1	**.166**(.000)	.173(.001)	.168(.001)	.204(.004)	.179(.002)	.171(.001)	**.127**(.000)	.132(.001)	.129(.001)	.156(.003)	.137(.001)	.131(.001)
ETTh2	**.260**(.002)	.269(.002)	.288(.005)	.459(.142)	.279(.002)	.278(.002)	**.190**(.001)	.198(.001)	.213(.003)	.256(.021)	.199(.001)	.203(.002)
ETTm1	.103(.000)	.106(.000)	**.102**(.000)	.148(.007)	.107(.000)	**.102**(.001)	.075(.000)	.076(.000)	**.074**(.000)	.110(.006)	.077(.000)	**.074**(.001)
ETTm2	.154(.002)	.138(.002)	.152(.002)	.251(.034)	**.134**(.002)	.137(.002)	.097(.001)	.090(.001)	.099(.001)	.158(.014)	**.088**(.001)	.090(.002)
Weather	.019(.000)	.019(.000)	.026(.000)	.023(.001)	**.018**(.000)	.025(.001)	**.011**(.000)	.012(.000)	.018(.000)	.014(.000)	.012(.000)	.018(.001)

A.10 Visualization of Fractionally Differenced Time Series Histograms



(a) BE

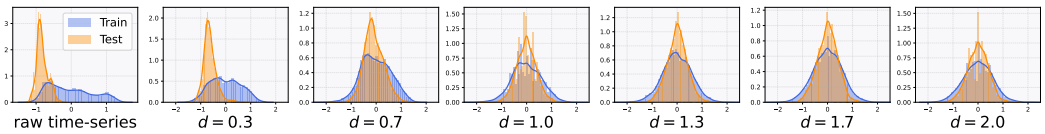
(b) DE

(c) FR

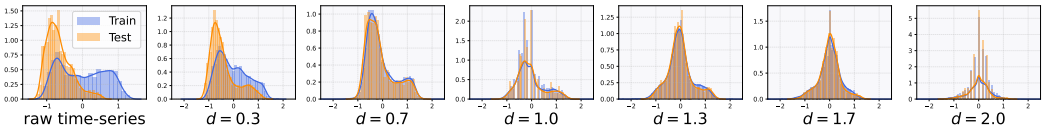
(d) NP

(e) PJM

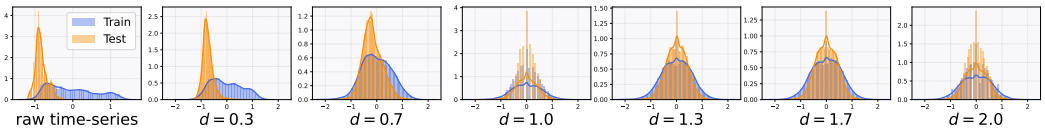
Figure 9: **Effect of fractional differencing: alleviating distribution shift**. In each figure, the leftmost panel shows histograms of the standardized series from the training and test datasets, while the other panels show histograms of the fractionally differenced series for $d \in \{0.3, 0.7, 1.0, 1.3, 1.7, 2.0\}$. For clarity, the transformation $\text{sign}(x) \cdot \log(1 + |x|)$ is applied.



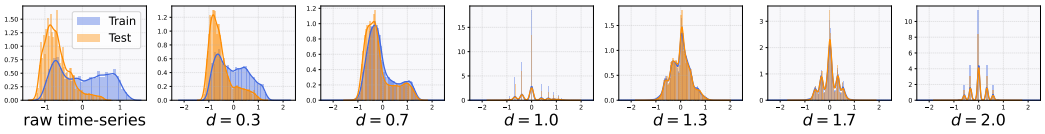
(a) ETTh1



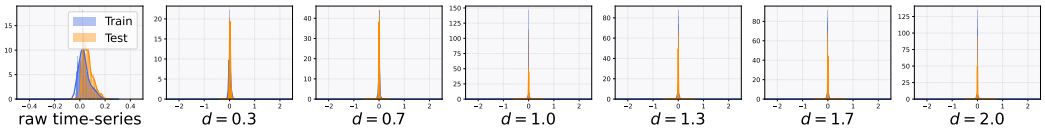
(b) ETTh2



(c) ETTm1



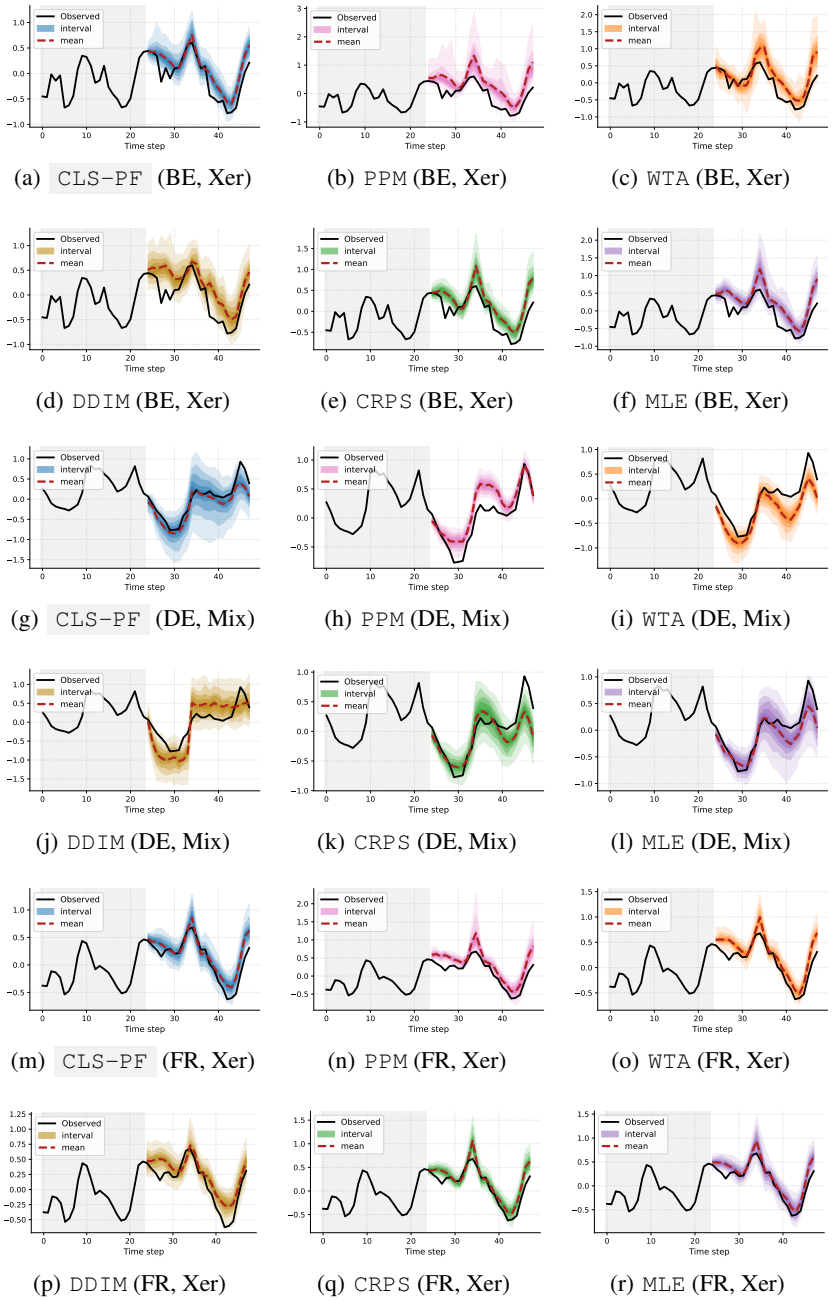
(d) ETTm2



(e) Weather

Figure 10: **Effect of fractional differencing: alleviating distribution shift**. In each figure, the leftmost panel shows histograms of the standardized series from the training and test datasets, while the other panels show histograms of the fractionally differenced series for $d \in \{0.3, 0.7, 1.0, 1.3, 1.7, 2.0\}$. For clarity, the transformation $\text{sign}(x) \cdot \log(1 + |x|)$ is applied.

A.11 QUALITATIVE RESULTS



(a) CLS-PF (BE, Xer) (b) PPM (BE, Xer) (c) WTA (BE, Xer)

(d) DDIM (BE, Xer) (e) CRPS (BE, Xer) (f) MLE (BE, Xer)

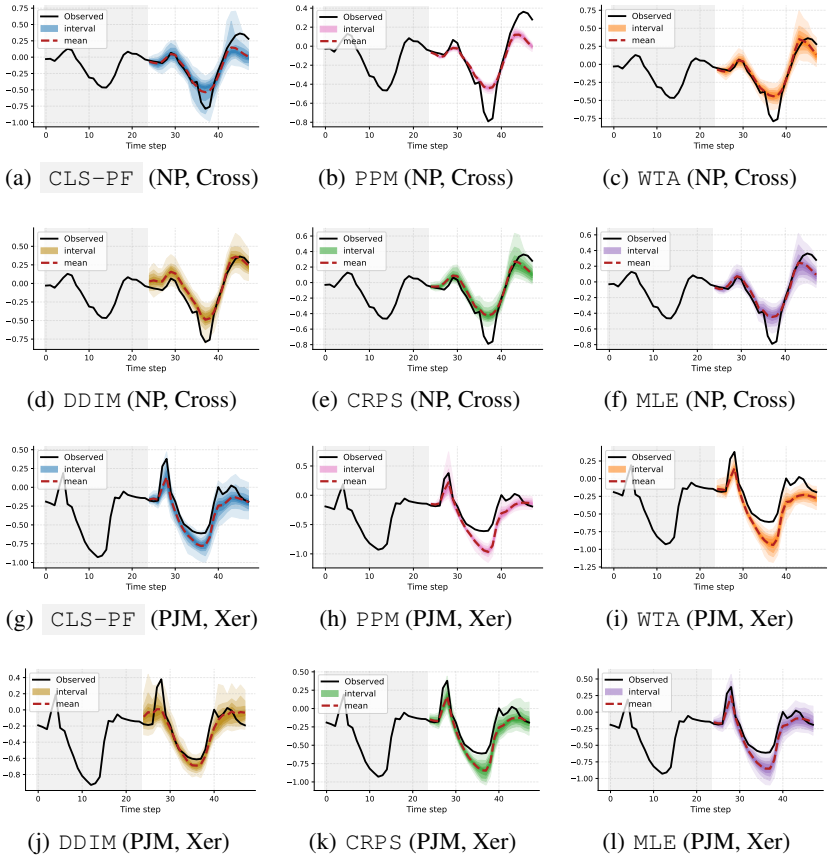
(g) CLS-PF (DE, Mix) (h) PPM (DE, Mix) (i) WTA (DE, Mix)

(j) DDIM (DE, Mix) (k) CRPS (DE, Mix) (l) MLE (DE, Mix)

(m) CLS-PF (FR, Xer) (n) PPM (FR, Xer) (o) WTA (FR, Xer)

(p) DDIM (FR, Xer) (q) CRPS (FR, Xer) (r) MLE (FR, Xer)

Figure 11: **Qualitative results** with various backbone architectures ('Xer' denotes TimeXer and 'Mix' denotes TSMixer) on BE, DE, and FR datasets. Visualization of probabilistic forecasting, where the shaded regions represent the density estimated from the generated 100 trajectories, and the red dashed line denotes the mean prediction.



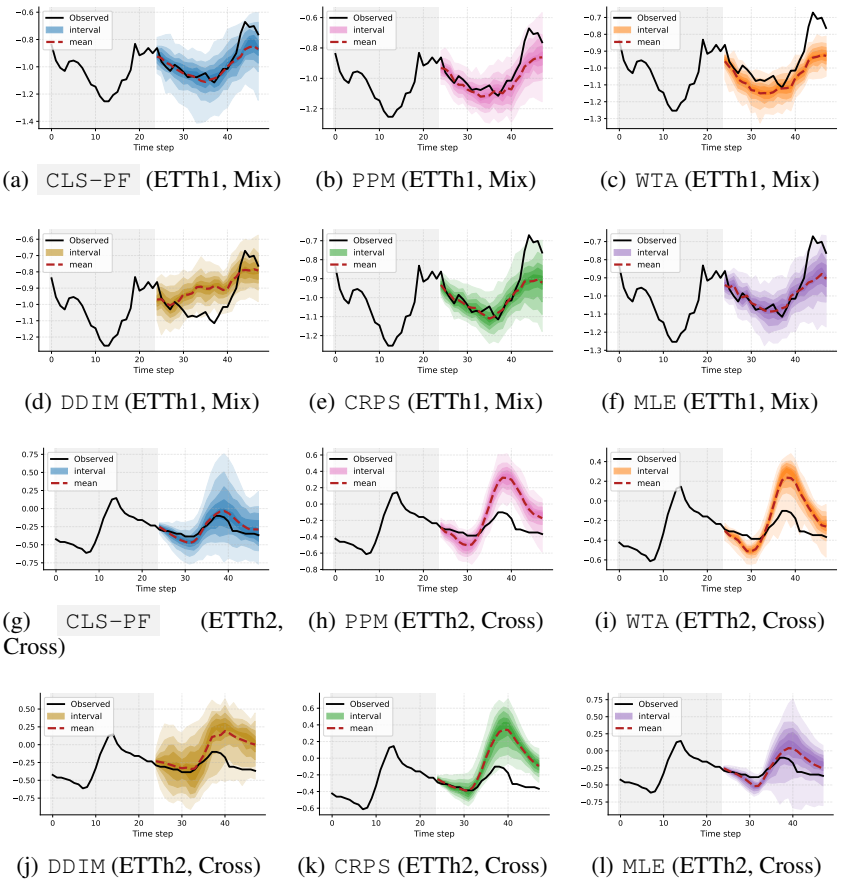
(a) CLS-PF (NP, Cross) (b) PPM (NP, Cross) (c) WTA (NP, Cross)

(d) DDIM (NP, Cross) (e) CRPS (NP, Cross) (f) MLE (NP, Cross)

(g) CLS-PF (PJM, Xer) (h) PPM (PJM, Xer) (i) WTA (PJM, Xer)

(j) DDIM (PJM, Xer) (k) CRPS (PJM, Xer) (l) MLE (PJM, Xer)

Figure 12: **Qualitative results** with various backbone architectures ('Cross' denotes CrossLinear and 'Xer' denotes TimeXer) on NP and PJM datasets. Visualization of probabilistic forecasting, where the shaded regions represent the density estimated from the generated 100 trajectories, and the red dashed line denotes the mean prediction.



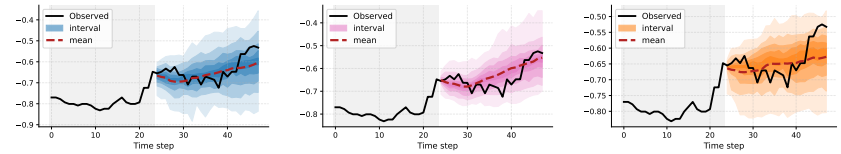
(a) CLS-PF (ETTh1, Mix) (b) PPM (ETTh1, Mix) (c) WTA (ETTh1, Mix)

(d) DDIM (ETTh1, Mix) (e) CRPS (ETTh1, Mix) (f) MLE (ETTh1, Mix)

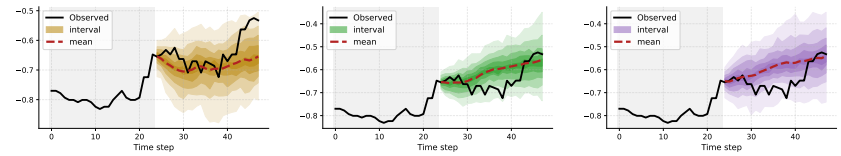
(g) CLS-PF (ETTh2, Cross) (h) PPM (ETTh2, Cross) (i) WTA (ETTh2, Cross)

(j) DDIM (ETTh2, Cross) (k) CRPS (ETTh2, Cross) (l) MLE (ETTh2, Cross)

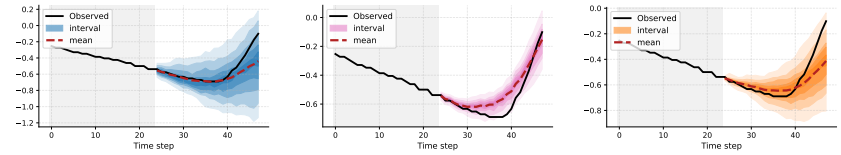
Figure 13: **Qualitative results** with various backbone architectures ('Cross' denotes CrossLinear and 'Mix' denotes TSMixer) on ETTh1 and ETTh2 datasets. Visualization of probabilistic forecasting, where the shaded regions represent the density estimated from the generated 100 trajectories, and the red dashed line denotes the mean prediction.



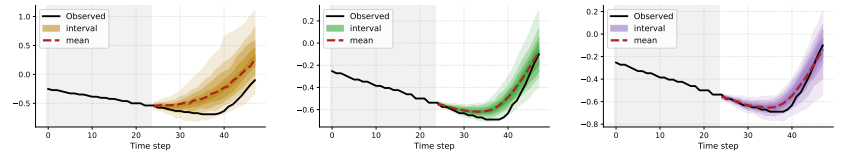
(a) CLS-PF (ETTm1, Xer) (b) PPM(ETTm1, Xer) (c) WTA(ETTm1, Xer)



(d) DDIM(ETTm1, Xer) (e) CRPS(ETTm1, Xer) (f) MLE(ETTm1, Xer)

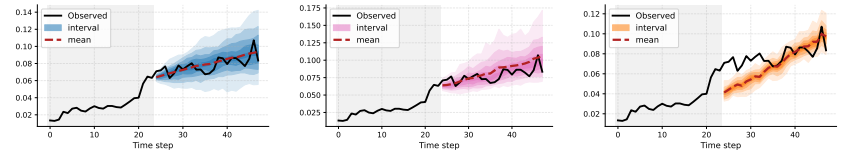


(g) CLS-PF (ETTm2, Mix) (h) PPM (ETTm2, Mix) (i) WTA (ETTm2, Mix)

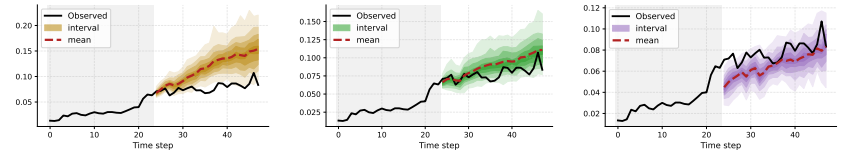


(j) DDIM (ETTm2, Mix) (k) CRPS (ETTm2, Mix) (l) MLE (ETTm2, Mix)

Figure 14: **Qualitative results** with various backbone architectures ('Xer' denotes TimeXer and 'Mix' denotes TSMixer) on ETTm1 and ETTm2 datasets. Visualization of probabilistic forecasting, where the shaded regions represent the density estimated from the generated 100 trajectories, and the red dashed line denotes the mean prediction.



(a) CLS-PF (Weather, Mix) (b) PPM (Weather, Mix) (c) WTA (Weather, Mix)



(d) DDIM (Weather, Mix) (e) CRPS (Weather, Mix) (f) MLE (Weather, Mix)

Figure 15: **Qualitative results** with various backbone architectures ('Mix' denotes TSMixer) on the Weather dataset. Visualization of probabilistic forecasting, where the shaded regions represent the density estimated from the generated 100 trajectories, and the red dashed line denotes the mean prediction.