Filter, Augment, Forecast: Online Data Selection for Robust Time Series Forecasting

Ege Onur Taga[1,*] **Halil Alperen Gozeten**[1] **Kutay Tire**[2] **Rahul Dalvi**[1]

Reinhard Heckel[3,4] **Samet Oymak**[1]

[1]University of Michigan, Ann Arbor [2]University of Texas at Austin

[3]Technical University of Munich [4]Munich Center for Machine Learning

Abstract

Data curation pipelines play a central role in training deep learning architectures, with their impact in time series forecasting still relatively underexplored. In this work, we propose *Filter, Augment, Forecast (FAF)*: an online data curation strategy based on (1) data selection to filter out low-quality (e.g., noisy) examples and (2) augmentation of the remaining high-quality data. We use reference model-based filtering inspired by the reducible holdout loss selection (RHO-LOSS) from the language modeling literature. We identify limitations of RHO-LOSS under domain shifts common in time series and introduce the adaptive RHO method (*AdaRho*), which improves performance by updating the reference model during training. Using random matrix theory, we provide a statistical analysis that characterizes the role of the reference model, sample size, and noise statistics in data selection. *FAF* consistently improves forecasting accuracy across diverse architectures without modifying them, achieving state-of-the-art results.

1 INTRODUCTION

Time-series forecasting remains a formidable challenge, as real-world sequences frequently exhibit low signal-to-noise ratios (Wang and Ventre, 2024), non-stationary dynamics (Han et al., 2024), and data scarce, low-redundancy regimes (Che, 2024). These conditions obscure the patterns a model aims to learn, allowing noise and other irregularities to degrade forecasting performance. Recent work tackles these difficulties in time-series forecasting by pretraining time-series foundation models on billions of points from heterogeneous domains, including energy, finance, healthcare, climate, and beyond, so that a single model can generalize across granularity, noise levels, and time scales (Ansari et al., 2024; Das et al., 2024; Chen et al., 2025; Woo et al., 2024). These ideas build on rapid architectural progress in deep forecasting (Zhou et al., 2022; Liu et al., 2022; Zhang and Yan, 2023), i.e. patch-based decoders (Nie et al., 2023), encoder-only transformers (Liu et al., 2023), and seasonal–trend hybrid models (Wu et al., 2021), and the emergence of large-scale time-series corpora such as the 100-billion-point TimesFM dataset (Das et al., 2024). Yet model capacity alone cannot compensate for low-quality data. Unlike NLP and computer vision, fields that now routinely curate, filter, and augment their training corpora (Gadre et al., 2023; Li et al., 2024; Wang et al., 2022), time-series research still largely leans on automatically collected data through sensors or other means, underscoring the need for systematic dataset curation and data selection.

We tackle the challenge of performant time-series forecasting by introducing *Filter, Augment, Forecast*, an online data curation strategy that performs data selection and augmentation. For selection, we build on RHO-LOSS from the vision and language modeling literature (Mindermann et al., 2022; Lin et al., 2024). RHO-LOSS distinguishes training examples more amenable to learning from those that are not, such as noisy or unpredictable tokens. This is done using a reference model, selecting samples at each batch based on the loss gap between target and reference models.

*Correspondence: Ege Onur Taga egetaga@umich.edu.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

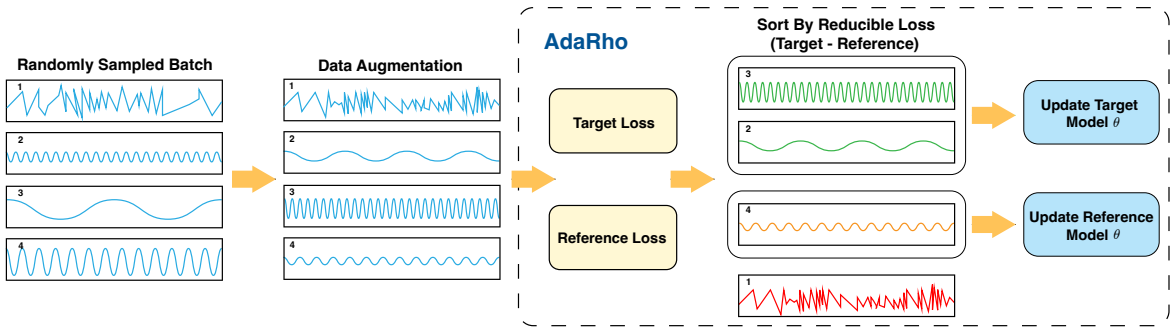


Figure 1: In *FAF*, each batch is randomly sampled, augmented, and ranked by *reducible loss*. Top samples update the target model using the standard learning rate, while the next-best update the reference model more conservatively with a smaller rate.

• Our first contribution is to introduce an adaptive variant of the RHO-LOSS method called *AdaRho* as described under Section 2.1. *AdaRho* addresses a key weakness of the RHO-LOSS algorithm which uses a static reference model and struggles under distribution shifts that are common in time-series (e.g. dynamic/streaming data or heterogeneous domains). We achieve this by updating the reference model through a secondary data selection mechanism.

• As a second contribution, we shed light on the fundamental mechanics of reference-based data selection through a statistical analysis. We study a regression setting with mixtures of two distributions: one with high noise and one with low noise. Our analysis sharply quantifies the roles of the reference model and noise levels in shaping the quality of selected samples and the performance of the eventual model. This reveals how weak reference models can degrade performance, motivating our *AdaRho* method, and how reduced sample size can harm forecasting accuracy, motivating the use of data augmentation.

• As a final contribution, we complement our adaptive data selection method, *AdaRho* (which filters data), with time series data augmentation (Cheema and Sugiyama, 2024; Bandara et al., 2021; Chung, 2020), yielding our final *Filter, Augment, Forecast (FAF)* online data curation method for time-series forecasting. The motivation for incorporating augmentation is to help the model not only learn from high-quality examples but also mitigate reduced sample size.

We demonstrate that *FAF*, as seen in Figure 1, boosts forecast performance and enables state-of-the-art results. Applied to eight state-of-the-art models, *FAF* yields a 6.55% mean MSE reduction across 9 tasks (see Section 4). Our studies reveal that certain training loss statistics predict when and why *FAF* is most effective, highlighting task-specific gains (e.g., when all examples are equally good, data selection offers little benefit).

Related Work. Our work fits within the broader literature on data selection and curation (Wenzek et al., 2020; Gao et al., 2020; Schuhmann et al., 2022), where filtering is a standard part of the pipeline. We focus on sample-level selection, building on prior work on prioritized and adaptive training-point selection (Loshchilov and Hutter, 2015; Mindermann et al., 2022; Schaul et al., 2015; Katharopoulos and Fleuret, 2018). Our approach is most closely related to RHO-LOSS (Mindermann et al., 2022), which we adapt to time series forecasting. Similar RHO-LOSS-style ideas have also been explored in computer vision, language modeling, and reinforcement learning (Mindermann et al., 2022; Lin et al., 2024; Sujit et al., 2023).

In time series forecasting, related data-centric methods study robustness, anomaly filtering, and augmentation (Cheng et al., 2024; Du et al., 2021; Fan et al., 2023; Yoon et al., 2022; Bandara et al., 2021). For example, RobustTSF (Cheng et al., 2024) formalizes several anomaly types and filters them using variance-based criteria, while Yoon et al. (2022) improve robustness through smoothed training. Other work focuses on offline sensor cleaning during data collection (Bachechi et al., 2020; Zhang et al., 2017; Ding et al., 2019). In contrast, we target variation in sample quality within standard time series datasets rather than explicit anomalies, sensor-level corrections. We extend RHO-LOSS to forecasting by allowing the reference model to adapt to distribution shift and by combining filtering with time series-specific augmentation, which helps offset the reduced sample size caused by filtering.

2 METHODOLOGY

Our online training data curation strategy *Filter, Augment, Forecast (FAF)* relies on two components: An online batch selection algorithm that filters out noisy samples, *AdaRho*, and an augmentation strategy that enriches the training data. While *AdaRho* alone yields

strong forecasting performance, it reduces the effective sample size for training. To address this, we combine it with data augmentation, which leads to further improvements in forecasting performance across diverse data domains. *FAF* extends *AdaRho* by coupling adaptive filtering with augmentation that offsets the retained-sample reduction induced by filtering. *FAF* is model-agnostic and can be applied to any time series forecasting model. As demonstrated in the experiments, it improves performance across all models considered. In the following, we detail *AdaRho* and the augmentation strategies we developed.

2.1 *AdaRho*: Adaptive Domain-Aware Batch Selection

We define a multivariate time series dataset as $\mathcal{D} := \{(t, \boldsymbol{y}_t)\}_{t=1}^{T}$, where $\boldsymbol{y}_t := [y_{t,1}, \ldots, y_{t,N}]^\top \in \mathbb{R}^N$ denotes the observation at time t with N channels. For a univariate series, we set $N = 1$. Each observation $y_{t,j}$ may exhibit both temporal dependencies and interactions with other channels. Given a partially observed sequence $\{(t, \boldsymbol{y}_t)\}_{t=1}^{\widetilde{T}}$, where $\widetilde{T} < T$, the forecasting objective is to predict future values $\boldsymbol{y}_{\widetilde{T}+1}, \ldots, \boldsymbol{y}_T$. For deep learning-based forecasting approaches, the time series is divided into training samples through a sliding window approach. Thus, one extracts training points with context length t and forecast horizon h as $(\boldsymbol{y}^i_{1:t},\ y^i_{t+1:t+h})$ for $i = 1, \ldots, n$, given that in total we have n samples through such process. For ease of understanding and for consistency with supervised learning notation, which we use to develop the theory, we refer to $(\boldsymbol{y}^i_{1:t}, \boldsymbol{y}^i_{t+1:t+h})$ as $(\boldsymbol{x}_i, \boldsymbol{y}_i) := (\boldsymbol{y}^i_{1:t}, \boldsymbol{y}^i_{t+1:t+h})$.

Time series data can vary widely across domains in terms of modality, number of channels, forecast horizon $\widetilde{T}$, and temporal dynamics. As a result, identifying informative and non-noisy samples for training is a domain-specific task. A pattern that is typical and meaningful in one domain may indicate sensor failure or measurement error in another.

To address this challenge, an effective online batch selection algorithm for time series forecasting should be domain-aware, recognizing the semantics of recurring patterns; distribution-sensitive, accounting for dataset-specific shifts and irregularities; and adaptive to non-stationary behaviors. Specifically, the algorithm should assign lower priority to noisy or corrupted samples and de-prioritize redundant data, such as easily learnable periodic trends, while emphasizing time-specific or rare patterns. For example, the sudden drop in household electricity usage after peak hours is more informative than repeatedly observed daily cycles. Now, let's construct the algorithmic framework for this task.

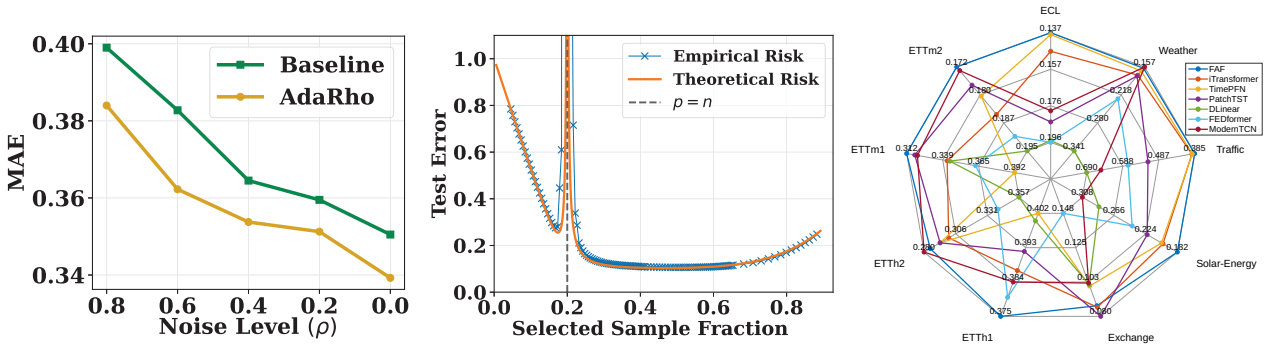
Reducible Loss Filtering Framework. We introduce two models: f_θ, the *target model*, parametrized by θ, whose forecasting performance we aim to improve, and $f_{\theta'}$, the *reference model*, which is parametrized by θ' and pretrained on a subset of the data, and used to estimate sample-level informativeness. Both models are optimized with stochastic gradient descent (SGD), where at each epoch mini-batches are sampled from the training data and the model parameters are updated through a gradient descent step. In the training of the *target model*, we filter out noisy or redundant samples from the minibatches with the framework introduced below.

Given a training set $(\boldsymbol{x}_i,\ \boldsymbol{y}_i)$ for $i = 1, \ldots, n$, we define the *reducible loss* (Mindermann et al., 2022) for each sample as

$$\varepsilon_i := \ell(f_\theta(\boldsymbol{x}_i), \boldsymbol{y}_i) - \ell(f_{\theta'}(\boldsymbol{x}_i), \boldsymbol{y}_i), \tag{1}$$

where ℓ is a forecasting loss function (e.g., the mean squared error). A large reducible loss indicates that the target model performs worse than the reference model on that training example, suggestive of a potential improvement, because the reference model is trained on the same data distribution, but only on a subset of it. Therefore, when a data point is encountered during training, there is a high probability that the reference model has not seen it before. Given that the reference model captures the underlying distribution reasonably well, out-of-distribution samples are likely to yield high loss when evaluated by it. Such out-of-distribution samples can arise from noise introduced during the measurement process or may result from rare external events that cannot be forecasted using the available data. In such cases, we prefer not to include these samples in training. However, when we have a low reference loss of a sample, this signals that such a sample can be learned effectively. If the target loss is high in that case, then that sample can be effectively learned but has not yet been learned (Mindermann et al., 2022). In contrast, if the target loss is also low, it signals that even though it can be effectively learned, it is already learned by the target model, hence there is no point of prioritizing those samples during learning iterations. Overall, high reducible loss signals useful samples that we would like to prioritize during SGD updates. Thus, the key step of the algorithm is to sample instances giving top $k\%$ reducible loss during training from each batch and to update the model's parameters on that subset.

Adaptivity. In standard RHO, the reference model's parameters are fixed and do not adapt to new samples during target model training. When datasets are large, as in web-scraped image or language corpora, a reference model trained on a sizable subset can approximate the data distribution under the i.i.d. assumption,



(a) *AdaRho* across noise levels (b) Theoretical *vs.* empirical risk (c) *FAF vs.* architectures in MSE

Figure 2: **(a)** Performance under increasing noise (averaged over ETTh1/2 and ETTm1/2). *AdaRho* remains robust to multiplicative noise. **(b)** Theoretical vs. empirical risk by selected fraction; Section 3 analyzes risk in terms of sample size, noise, and filtering. **(c)** MSE comparison of *FAF*-enhanced TimePFN with SOTA models. Both input and forecast horizons are set to 96 time steps.

making adaptation unnecessary. However, in many domains, especially time series, datasets are smaller. More importantly, time series data violates the i.i.d. assumption, as training samples are drawn from a time-dependent distribution. As a result, a reference model trained on a subset of the data may fail to represent the distribution accurately. Thus, it becomes essential for the reference model to adapt to the evolving distribution while maintaining decorrelated outputs from the target model. To achieve this, we update the reference model parameters using samples ranked between the top $k\%$ and $(k+r)\%$ in terms of reducible loss, and we apply a significantly lower learning rate (i.e. 5% of the target model's learning rate), as detailed in Algorithm 1. These samples are **informative**, as they come from the data distribution the reference model seeks to capture; **decorrelated**, since the target model trains on the top $k\%$ of samples while the reference model trains on the next $k\%-(k+r)\%$, resulting in largely distinct data; and **less noisy**, because low reducible loss signals noise, so selecting high-reducible-loss samples in the $[k\%, (k+r)\%]$ range helps target cleaner and informative examples.

We refer to the adaptive RHO algorithm as *AdaRho*, with its pseudocode in Algorithm 1 in the Appendix. We use *AdaRho* in all experiments. In Section 3, we provide theoretical justification for adapting RHO. Additionally, our ablation studies compare *AdaRho* to standard RHO and show substantial performance improvements across a range of settings.

2.2 Robustness of *AdaRho* to Multiplicative Noise

To demonstrate the robustness of *AdaRho* under realistic sensor–failure scenarios, we conduct a case study on the four Electric Transformer Temperature (ETT) benchmarks (ETTh1/2-ETTm1/2). We inject synthetic noise by randomly simulating *multiplicative* sensor failures with varying probabilities. Formally, given training pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i) := (\boldsymbol{y}^i_{1:t}, \boldsymbol{y}^i_{t+1:t+h})$ we first subsample a fraction $\rho \in \{0, 0.2, 0.4, 0.6, 0.8\}$ of the instances. For every selected sample we draw a *uniform* subset of 20–40 % of its time steps and channels, and corrupt them with i.i.d. Gaussian noise with $\sigma^2 = 4$:

$$\bar{y}^i_{t',j} = y_{t',j} \cdot z^i_{t',j} \text{ s.t. } z^i_{t',j} \sim \mathcal{N}\big(1, \sigma^2\big) \text{ and } j \in [N]. \quad (2)$$

Setting. For each noise ratio ρ we train two models from scratch on the corrupted training split: (i) the vanilla iTransformer baseline and (ii) its *AdaRho*-applied counterpart. All hyperparameters follow the official iTransformer configuration ($(t, h) = (96, 96)$). We evaluate the *clean* test split and report mean absolute error (MAE). Figure 2a plots the average MAE across the four ETT datasets; the corresponding raw MAE and MSE numbers are provided in Appendix B.

Results. Two trends emerge from Figure 2a. First, performance degrades for both models as ρ increases, but the drop is noticeably milder for *AdaRho*. At the highest noise level ($\rho = 0.8$), *AdaRho* cuts MSE by **7.4**% and MAE by **4.0**% relative to baseline. At $\rho = 0.6$ the gains widen to **9.0**% MSE and **5.3**% MAE, underscoring the benefit of targeted data selection under heavy corruption. Even in the noise-free setting ($\rho = 0$), *AdaRho* improves MSE by **4.0**% and MAE by **3.2**%. Because real-world time-series are never perfectly clean, these results suggest that *AdaRho* delivers accuracy gains across both low- and high-noise regimes.

2.3 Data Augmentation

At each batch, *AdaRho* filters out potentially noisy observations, then trains the target model on the resulting

subset of high-quality samples. A reduced training set, however, can hurt forecasting accuracy. In Section 3, we analyze how noise magnitude and sample–target correlations interact, showing that a thresholding rule retains a higher proportion of clean data. Yet, as Figure 2b illustrates, the benefit of cleaner samples can be offset by reduced sample size: the risk may not fall below the no-threshold baseline because sample size remains a dominant term in the risk bound.

To recover lost diversity, we augment the data while avoiding transformations that violate the causal structure of the series. Building on prior work in time-series augmentation, specifically StiefelGen (Cheema and Sugiyama, 2024), Gaussian smoothing (Rosenblatt, 1956), and jittering, we apply these transformations sequentially at each training batch, sampling with probabilities 0.5, 0.25, and 0.5, respectively. StiefelGen augments multivariate sequences by projecting them onto the Stiefel manifold, thereby preserving underlying temporal dynamics. Jittering introduces small Gaussian perturbations to increase local variability. In our pipeline, the reference model is trained solely on the subset of non-augmented data. We then create augmented batches and use *AdaRho* to filter out both noisy samples and those that may exhibit physically implausible behavior due to augmentation. Importantly, we keep this augmentation pipeline simple: its sole purpose is to counteract the reduced sample size from filtering. Our contribution lies not in devising new augmentations, but in showing how data selection can degrade accuracy if it shrinks the training set, and how lightweight augmentation helps mitigate this effect.

3 THEORETICAL RESULTS

In this section, we provide a theoretical analysis of reference-loss filtering in a regression setting aligned with point forecasting, and quantify trade-offs between data quality, noise, and sample size. Modeling with linear regression and random features, we leverage random matrix theory for a sharp characterization (Tulino and Verdú, 2004; Hastie et al., 2022; Bartlett et al., 2020). Unlike prior work on heterogeneous noise (Akhtiamov et al., 2024; Song et al., 2024; Patil et al., 2024; Jain et al., 2024), our setting presents unique challenges due to statistical dependencies between label noise and features induced by reference-based filtering, resulting in estimation bias. We study the linear regression framework as it represents the high-dimensional statistics setting for analyzing deep learning models: nonlinear architectures are analytically intractable, whereas the linear regime allows sharp characterizations through random matrix theory. Our analysis provides meaningful insights into data selection and the roles of the reference model and noise statistics.

3.1 System Model

Suppose that we observe i.i.d. pairs $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ that follow the linear model $y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta}_\star + \xi_i$ for $i = 1, \dots, n$, where $\boldsymbol{\beta}_\star \in \mathbb{R}^p$ is the latent/optimal model we wish to learn. The features obey $\boldsymbol{x}_i \in \mathbb{R}^p$ with $\boldsymbol{x}_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and, crucially, the noise term ξ_i is distributed as a mixture with two components:

$$\xi_i = \begin{cases} \sigma_1 \, z_i, & \text{with probability } p_1, \\ \sigma_2 \, z_i, & \text{with probability } p_2, \end{cases}, \; p_1 + p_2 = 1. \quad (3)$$

Here $z_i \overset{\text{i.i.d.}}{\sim} \mathcal{D}_z$ is a distribution such that $\mathbb{E}[z_i] = 0$ and $\mathrm{Var}(z_i) = 1$, and $\sigma_2 \geq \sigma_1 > 0$ are the variance levels corresponding to more vs less noisy components respectively. Define $\gamma = p/n$. We estimate $\boldsymbol{\beta}_\star$ by solving the following minimum ℓ_2–norm least-squares problem:

$$\hat{\boldsymbol{\beta}} := \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \left\{ \| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \|_2^2 : \text{with minimal } \|\boldsymbol{\beta}\|_2 \right\}. \quad (4)$$

where $\boldsymbol{X} = [\boldsymbol{x}_1^\top, \dots, \boldsymbol{x}_n^\top]^\top \in \mathbb{R}^{n \times p}$ and $\boldsymbol{y} = [y_1, \dots, y_n]^\top \in \mathbb{R}^n$. We know that the estimate can be equivalently written as $\hat{\boldsymbol{\beta}} = \boldsymbol{X}^\dagger \boldsymbol{y}$. For a test sample $\boldsymbol{x}_0 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ which is independent of the training data, the risk of $\hat{\boldsymbol{\beta}}$ is defined as:

$$\mathcal{R}_{\boldsymbol{X}}(\hat{\boldsymbol{\beta}}; \boldsymbol{\beta}) = \mathbb{E}\left[(\boldsymbol{x}_0^\top \boldsymbol{\beta}_\star - \boldsymbol{x}_0^\top \hat{\boldsymbol{\beta}})^2 \mid \boldsymbol{X} \right]. \quad (5)$$

Filtering with a Reference Model. Suppose that we have a reference parameter $\boldsymbol{\beta}_{\text{ref}} \in \mathbb{R}^p$. For every $i = 1, \dots, n$, we compute the residual for a chosen threshold $\tau > 0$ as:

$$r_i = y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}_{\text{ref}}, \text{ with } \mathcal{I}(\tau) = \{ i \leq n : |r_i| \leq \tau \}. \quad (6)$$

Let the cardinality of filtered indices be $n_f = |\mathcal{I}(\tau)|$ and define the filtered data matrix as $\boldsymbol{X}_f = \left[\boldsymbol{x}_i^\top\right]_{i \in \mathcal{I}(\tau)}^\top \in \mathbb{R}^{n_f \times p}$ and the label vector as $\boldsymbol{y}_f := [y_i]_{i \in \mathcal{I}(\tau)} \in \mathbb{R}^{n_f}$. Similar to (4), we solve the minimum ℓ_2-norm least squares problem on the filtered data $\boldsymbol{X}_f$ with corresponding labels $\boldsymbol{y}_f$ and obtain $\hat{\boldsymbol{\beta}}_f = {\boldsymbol{X}_f}^\dagger \boldsymbol{y}_f$. Following the equation (5), we similarly define the risk $\mathcal{R}_{\boldsymbol{X}_f}(\hat{\boldsymbol{\beta}}_f; \boldsymbol{\beta})$. Below, we provide an analysis of this risk in terms of the key problem variables.

3.2 Derivation of Filtered Model Asymptotics

To derive our results, we consider a standard large dimensional setting, i.e., proportional asymptotics, where the number of samples n and dimension p grow together to infinity while obeying $\lim_{n \to \infty} p/n = \gamma$. Let us first recall from Hastie et al. (2022) that, for the standard linear regression problem $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||$ with

ground-truth model $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}_\star + \boldsymbol{\xi}$ with the label noise $\boldsymbol{\xi}$ having i.i.d. zero mean and σ^2 variance entries, the asymptotic risk converges to:

$$\mathcal{R}_{\boldsymbol{X}}(\hat{\boldsymbol{\beta}}; \boldsymbol{\beta}_\star) \to \begin{cases} \sigma^2 \frac{\gamma}{1-\gamma}, & \gamma < 1, \\ \|\boldsymbol{\beta}_\star\|_2^2 \left(1 - \frac{1}{\gamma}\right) + \frac{\sigma^2}{\gamma - 1}, & \gamma > 1. \end{cases}$$

Our analysis will build on this fact, however, our setting is made challenging by the fact that we have two distinct distributions and a filtering process. To proceed, we will first study the impact of filtering by characterizing the effective noise level and the sample size post-filtering.

Characterizing sample size. We now define a few key terms before stating our asymptotic risk formula. Define the norm of the gap between two models as $\delta := \|\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}\|_2$. Looking at the residual given by (6), we notice that $\boldsymbol{x}_i^\top(\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}})$ is distributed as $\delta\, g_i$ for $g_i \sim \mathcal{N}(0,1)$. Hence, $|r_i| \le \tau$ translates to the event $|\delta\, g_i + \sigma_j z_i| \le \tau$. Accordingly, for any $g, z \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$, we denote:

$$\Pi_{\delta,\sigma_j}(\tau) = \mathbb{P}\left(|\delta g + \sigma_j z| \le \tau\right), \qquad \text{for } j = 1, 2.$$

Equivalently, we have $\Pi_{\delta,\sigma_j}(\tau) = 2\,\Phi(\tau/\sqrt{\delta^2 + \sigma_j^2}) - 1$ for $j = 1, 2$ where $\Phi(\cdot)$ denote the standard normal CDF. We denote the fraction of the data points that are kept after filtering and the resulting parameterization ratio by:

$$\pi(\tau) := p_1 \Pi_{\delta,\sigma_1}(\tau) + p_2 \Pi_{\delta,\sigma_2}(\tau), \quad \gamma_f(\tau) := \frac{\gamma}{\pi(\tau)}. \tag{7}$$

Characterizing noise level and correlation. After filtering, we decompose each noise component $\sigma_j z$ into a part orthogonal to δg and the residual that is aligned with δg. Using the shorthand notation $\zeta_j = \delta g + \sigma_j z$, we define for $j = 1, 2$:

$$\begin{aligned} \sigma_{j,\perp}^2(\tau, \delta) &:= \sigma_j^2 \operatorname{Var}\left(z \,\middle|\, |\zeta_j| \le \tau\right), \\ \sigma_{j,\parallel}^2(\tau, \delta) &:= \sigma_j^2 \frac{\operatorname{Cov}\left(z, \delta g \,\middle|\, |\zeta_j| \le \tau\right)^2}{\operatorname{Var}\left(\delta g \,\middle|\, |\zeta_j| \le \tau\right)}. \end{aligned} \tag{8}$$

Finally, we let the effective variances in the filtered dataset be decomposed as:

$$\begin{aligned} \sigma_\perp^2(\tau, \delta) &:= \frac{p_1 \Pi_{\delta,\sigma_1}(\tau)\, \sigma_{1,\perp}^2(\tau, \delta) + p_2 \Pi_{\delta,\sigma_2}(\tau)\, \sigma_{2,\perp}^2(\tau, \delta)}{\pi(\tau)}, \\ \sigma_\parallel^2(\tau, \delta) &:= \frac{p_1 \Pi_{\delta,\sigma_1}(\tau)\, \sigma_{1,\parallel}^2(\tau, \delta) + p_2 \Pi_{\delta,\sigma_2}(\tau)\, \sigma_{2,\parallel}^2(\tau, \delta)}{\pi(\tau)}. \end{aligned} \tag{9}$$

In the initial regression, z and g are independent, with z pure noise. Post-filtering, however, they become negatively correlated due to absolute-value thresholding, which biases estimation of the latent parameter $\beta_\star$. Our noise decomposition captures this effect to refine the risk estimate. After applying threshold τ to residuals $|r_i|$, the retained proportions of each noise component deviate from the original mixture weights p_1, p_2. Since $\Pi_{\delta,\sigma_1}(\tau) \ge \Pi_{\delta,\sigma_2}(\tau)$ for all τ when $\sigma_2 \ge \sigma_1$, thresholding favors retention of lower-noise samples. Thus, the filtered dataset contains a higher fraction from the σ_1 component, effectively shifting the noise distribution toward a cleaner regime.

Asymptotic Risk Formula Under Filtering. We now present our main theoretical result, which compares the asymptotic risks of the standard and filtered least-squares estimators in the high-dimensional regime. As $n, p \to \infty$ such that $p/n \to \gamma \in (0, 1)$, the unfiltered min-norm least-squares estimator $\hat{\boldsymbol{\beta}}$, almost surely satisfies:

$$\lim_{n \to \infty} \mathcal{R}_{\boldsymbol{X}}(\hat{\boldsymbol{\beta}}; \boldsymbol{\beta}_\star) = \left(p_1\, \sigma_1^2 + p_2\, \sigma_2^2\right) \frac{\gamma}{1 - \gamma}. \tag{10}$$

The above result is obtained by extending Theorem 1 of Hastie et al. (2022) for a mixture model noise. Before stating the risk formula for filtered least squares, we assume $\boldsymbol{\beta}_\star \perp (\boldsymbol{\beta}_{\text{ref}} - \boldsymbol{\beta}_\star)$. Let $\hat{\boldsymbol{\beta}}_f$ be the minimum ℓ_2-norm estimator on the dataset obtained by the filtering with threshold τ given by (6), and define $\gamma_f(\tau) = \gamma/\pi(\tau)$ to be the parameterization ratio after filtering. Consider the decomposition of the effective noise into $\sigma_\perp^2(\tau, \delta)$ and $\sigma_\parallel^2(\tau, \delta)$ given by (9). The aligned part $\sigma_\parallel^2(\tau, \delta)$, which is a signal that lies in the column space of $\boldsymbol{X}$, behaves like the bias component. As a result, the aligned part of the noise contributes to risks by $\sigma_\parallel^2(\tau, \delta)$ and $\frac{\sigma_\parallel^2(\tau,\delta)}{\gamma_f(\tau)}$ in the two different regimes. When $\gamma_f(\tau) > 1$, the min-norm solution shrinks any aligned component of the noise by a factor $1/\gamma_f(\tau)$. On the other hand, the orthogonal term $\sigma_\perp^2(\tau, \delta)$ is treated as i.i.d. Gaussian noise. Therefore, as $n \to \infty$, we state the following result with $\mathcal{R}_{\boldsymbol{X}_f} := \mathcal{R}_{\boldsymbol{X}_f}(\hat{\boldsymbol{\beta}}_f; \boldsymbol{\beta}_\star)$ as:

$$\mathcal{R}_{\boldsymbol{X}_f} \approx \begin{cases} \sigma_\perp^2(\tau, \delta)\, \frac{\gamma_f(\tau)}{1 - \gamma_f(\tau)} + \sigma_\parallel^2(\tau, \delta) & \gamma_f(\tau) < 1, \\ \|\boldsymbol{\beta}_\star\|_2^2 \left(1 - \frac{1}{\gamma_f(\tau)}\right) + \frac{\sigma_\perp^2(\tau,\delta)}{\gamma_f(\tau) - 1} + \frac{\sigma_\parallel^2(\tau,\delta)}{\gamma_f(\tau)} & \gamma_f(\tau) > 1. \end{cases}$$

The above fact utilizes Theorem 1 of Hastie et al. (2022), and generalizes the results to filtering. The formula indicates that starting from the under-parameterized regime $\gamma < 1$, filtering may move the system to either side of the interpolation threshold depending on $\pi(\tau)$. Moreover, we note that as $\tau \to \infty$, we have $\gamma_f(\tau) \to \gamma$ and $\sigma_\perp^2(\tau, \delta) \to p_1\sigma_1^2 + p_2\sigma_2^2$, so the filtered estimator recovers the unfiltered one. Stating the asymptotic risk formula, we assumed that $\boldsymbol{\beta}_{\text{ref}}$ is sufficiently close to $\boldsymbol{\beta}_\star$, which we discuss in Appendix C.

Figure 2b compares the theoretical and empirical risk estimates for $n = 2000$, $p = 400$, $\|\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}\|_2 = 0.25$, and $(\sigma_1, \sigma_2) = (0.5, 2)$. Although filtering increases the

proportion of lower-noise samples, the reduced dataset size can raise the risk, highlighting the trade-off between removing noisy points and retaining sufficient data, and motivating the need for augmentation. Additionally, when the reference model $\boldsymbol{\beta}_{\text{ref}}$ better aligns with $\boldsymbol{\beta}_\star$, the final risk decreases further, supporting the adaptivity of *AdaRho*.

4 EXPERIMENTS

We evaluate *FAF* on a variety of multivariate (MTS) and univariate time-series benchmarks and compare the results with several baselines. We consider several deep-learning models as baselines, and for each model, we report both the native performance (i.e., the performance of the model with standard SGD) and its performance with *FAF* training.

Multivariate Time Series Forecasting Baselines. We evaluate *FAF* with six state-of-the-art multivariate forecasters: PatchTST (Nie et al., 2023), iTransformer (Liu et al., 2023), TimePFN (Taga et al., 2025), Autoformer (Wu et al., 2021), Informer (Zhou et al., 2021), and a convolutional baseline (ModernTCN (Donghao and Xue, 2024)). PatchTST and TimePFN both employ patch embeddings but differ in how they capture inter-channel interactions, whereas iTransformer takes each channel as a single token, making it behave almost like a linear model. Autoformer exploits autocorrelation, Informer uses sparse long-range attention, and TimePFN adds synthetic pretraining.

Univariate Time Series Baselines. We apply *FAF* to two state-of-the-art zero-shot time-series foundation models, Chronos-Bolt-Base (Ansari et al., 2024) and Moirai-Base (Woo et al., 2024), fine-tuning each with and without *FAF*.

Benchmark Datasets & Metrics. We evaluated *FAF* on nine MTS datasets: ETTh1/2, ETTm1/2, Weather, Solar Energy, ECL, Exchange, and Traffic. Solar Energy is from Lai et al. (2018), and the others are from Wu et al. (2021). In the univariate setting, only the target column is used. Although Exchange is not ideal for evaluating forecasting models (Zeng et al., 2023; Rossi, 2013) due to its near random-walk behavior, we include it to assess how *FAF* performs under such conditions. We report MSE and MAE in all experiments, and also weighted quantile loss (WQL) for the univariate baselines that provide probabilistic forecasts, following the setting of Ansari et al. (2024).

Experimental Setup. For multivariate tasks, we use a 96-timestep lookback window and a 96-timestep forecast horizon. Because we already train six baselines and their *FAF*-enhanced counterparts across nine benchmark datasets, reporting results for every forecast horizon would be prohibitive. We therefore present results for a single horizon, while Appendix F, Table 15 confirms that *FAF* also performs well at longer horizons. For univariate tasks, which emphasize shorter horizons, we use 96 input steps but forecast the next 36. Each dataset receives a dedicated reference model trained on 25% of its training data. In the multivariate *FAF* pipelines, we reuse a single reference model, TimePFN, across all target models to streamline comparisons. Appendix F, Table 14 shows that replacing TimePFN with iTransformer yields similar results. In the univariate setting, we instead use Chronos-Bolt as the reference model, fine-tuned on the 25% subset (see Appendix D for details).

The training of the target and reference models are performed using SGD aiming to minimize the training MSE loss. We compute the *reducible loss* for every instance in each batch using the MSE, as MSE is particularly sensitive to outliers. During training, the target model is updated with the top k% of instances ranked by reducible loss, where the hyper-parameter $k \in \{25, 50, 75\}$ is held fixed for a given experiment. The reference model is updated with samples whose reducible loss lies between the k% and $(k+r)$% quantiles. We set $r = \frac{1}{2}k$ except when $k = 75$, in which case we cap r at 20. Although r could be tuned via a grid search, we leave it fixed to keep the hyper-parameter space small. The value of k is selected by choosing the setting that minimizes the validation loss. Details on baselines, training, and hyperparameters are in Appendix D.

Main Results. The results in Table 1 show that *FAF* improves multivariate forecasting across a range of models. It delivers up to a 20% reduction in MSE and an 11.7% reduction in MAE at the model level. Averaged over all datasets and models in MTS forecasting, the dataset-level MSE drops by 6.4% (right-most columns of Table 1). Gains are most pronounced on low-dimensional datasets such as ETTh1 (7 channels) and taper off on high-dimensional datasets like Traffic (862 channels) and ECL (321 channels). A potential reason is statistical: when MSE is averaged over many channels, reducible-loss values become less dispersed, weakening *FAF*'s filtering effect. Dispersion metrics, namely coefficient of variation and quartile coefficient of dispersion, computed during training in Table 4 confirm that reducible losses in `ECL` and `Traffic` are far less dispersed than in the `ETT` datasets. Channel-wise application of *AdaRho* through masking of the loss contributions from the highest reducible loss yielding channels could mitigate this limitation.

Model-wise benefits also vary. iTransformer, ModernTCN, TimePFN, Informer, and Autoformer see substantial gains, whereas PatchTST has modest im-

Table 1: MTS forecasting results with respect to baseline, and its *FAF*-applied version. Input and forecast lengths are 96. The rightmost column reports average relative MSE improvement per dataset; bottom shows average relative improvements in MSE and MAE per architecture.

Dataset	iTransformer				TimePFN				Informer				*avg.impr.*
	Baseline		FAF		Baseline		FAF		Baseline		FAF		
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	
ETTh1	0.387	0.405	**0.381**	**0.402**	0.402	0.417	**0.375**	**0.402**	0.930	0.763	**0.745**	**0.619**	**9.4%**
ETTh2	0.300	0.349	**0.293**	**0.343**	0.293	0.343	**0.285**	**0.335**	2.928	1.349	**1.466**	**0.991**	**18.3%**
ETTm1	0.342	0.376	**0.325**	**0.365**	0.392	0.402	**0.312**	**0.350**	0.623	0.559	**0.583**	**0.490**	**10.6%**
ETTm2	0.185	0.272	**0.172**	**0.252**	0.180	0.262	**0.172**	**0.250**	0.396	0.474	**0.310**	**0.430**	**11.0%**
Solar	0.201	0.233	**0.190**	**0.225**	0.203	0.219	**0.182**	**0.213**	0.190	**0.216**	**0.166**	0.227	**9.5%**
Traffic	0.393	0.268	**0.388**	**0.262**	0.392	0.260	**0.385**	**0.254**	0.735	0.409	**0.694**	**0.392**	**2.9%**
ECL	0.147	0.239	0.147	**0.237**	0.138	0.234	**0.137**	**0.233**	0.327	0.413	**0.311**	**0.399**	**1.9%**
Exchng.	0.086	0.206	**0.085**	**0.204**	0.100	0.223	**0.087**	**0.205**	0.921	0.774	**0.655**	**0.654**	**14.3%**
Weather	0.175	0.215	**0.169**	**0.207**	0.166	0.208	**0.160**	**0.201**	0.455	0.481	**0.318**	**0.383**	**12.4%**
avg.impr.			**3.0%**	**2.7%**			**7.1%**	**4.5%**			**20.0%**	**11.7%**	**10.0%**
	ModernTCN				PatchTST				Autoformer				*avg.impr.*
	Baseline		FAF		Baseline		FAF		Baseline		FAF		
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	
ETTh1	0.386	**0.393**	**0.385**	0.394	0.392	0.404	0.392	0.404	0.440	0.446	**0.414**	**0.437**	**2.1%**
ETTh2	0.295	0.341	**0.290**	**0.339**	0.293	0.343	**0.289**	**0.339**	0.364	0.408	**0.336**	**0.381**	**3.6%**
ETTm1	0.323	0.366	**0.302**	**0.348**	0.318	0.357	**0.315**	**0.356**	0.520	0.490	**0.493**	**0.460**	**4.2%**
ETTm2	0.171	0.255	**0.169**	**0.250**	0.177	0.260	**0.173**	**0.254**	0.233	**0.311**	**0.224**	0.312	**2.4%**
Solar	0.315	**0.335**	**0.314**	0.338	0.222	0.267	0.222	0.267	0.455	**0.480**	**0.438**	0.513	**1.4%**
Traffic	0.735	0.459	**0.708**	**0.439**	0.517	0.334	**0.494**	**0.319**	**0.605**	**0.376**	0.613	0.381	**2.3%**
ECL	0.214	0.298	**0.211**	**0.294**	0.185	0.267	**0.182**	0.269	0.214	0.327	**0.190**	**0.305**	**4.7%**
Exchng.	0.103	0.228	**0.097**	**0.221**	0.080	0.196	0.083	0.200	**0.147**	**0.279**	0.153	0.287	**-0.7%**
Weather	0.162	0.211	**0.158**	**0.206**	0.177	0.218	**0.175**	0.218	0.273	0.344	**0.241**	**0.313**	**5.1%**
avg.impr.			**2.6%**	**1.9%**			**0.9%**	**0.6%**			**4.9%**	**2.1%**	**2.8%**

Table 2: Univariate forecasting results with baseline and FAF-applied version. Input and forecast lengths are 96 and 36. The rightmost column reports dataset averages over both models; the bottom row shows model improvements over all datasets.

Dataset	Chronos						Moirai						*avg.impr.*
	Baseline			FAF			Baseline			FAF			
	MSE	MAE	WQL	MSE	MAE	WQL	MSE	MAE	WQL	MSE	MAE	WQL	
ETTh1	0.043	0.156	0.219	**0.036**	**0.141**	**0.198**	0.037	0.146	0.205	**0.036**	**0.144**	0.205	**9.5%**
ETTh2	0.082	0.215	0.318	**0.080**	**0.211**	**0.300**	0.105	0.240	0.337	**0.094**	**0.229**	**0.323**	**6.7%**
ETTm1	0.015	0.089	0.128	0.015	**0.088**	**0.126**	0.016	0.091	0.129	**0.015**	**0.089**	**0.125**	**3.1%**
ETTm2	0.028	0.111	**0.154**	**0.027**	**0.108**	0.157	0.032	0.113	0.159	**0.030**	**0.110**	0.159	**4.9%**
Solar	0.230	0.272	0.402	**0.213**	**0.253**	**0.360**	0.221	**0.216**	**0.308**	**0.204**	0.226	0.320	**7.5%**
Traffic	0.107	0.176	0.249	**0.100**	**0.166**	**0.245**	0.139	0.222	0.313	**0.120**	**0.196**	**0.280**	**10.1%**
ECL	0.191	0.302	0.420	**0.186**	**0.295**	**0.414**	0.293	0.388	**0.541**	**0.285**	**0.385**	0.543	**2.7%**
Exchng.	0.040	0.151	0.208	**0.036**	**0.145**	0.208	0.039	0.148	0.211	**0.035**	**0.145**	**0.209**	**10.1%**
Weather	0.0005	0.015	0.023	0.0005	0.015	**0.022**	0.0006	0.016	0.023	**0.0005**	**0.015**	**0.022**	**8.3%**
avg.impr.				**5.4%**	**3.8%**	**3.6%**				**8.5%**	**3.0%**	**2.1%**	**6.9%**

provements. PatchTST forecasts each channel independently, so noise averages out during backpropagation, reducing the marginal value of filtering. By contrast, Informer, whose ProbSparse attention increases architectural complexity, accumulates the largest rewards, with MSE and MAE falling by 20% and 11.7%. Overall, *FAF* substantially improves performance. In Figure 2c, we report the *FAF*-applied performance of TimePFN against state-of-the-art architectures, including FedFormer (Zhou et al., 2022) and DLinear (Zeng et al., 2023). Results show that the FAF-applied architecture yields state-of-the-art results.

In the univariate benchmarks, *FAF* delivers significant

Table 3: Ablation on ETTh/m datasets (MSE and MAE) using TimePFN for MTS forecasting (input length and forecast horizon 96). The columns denote the method used. *FAF* yields almost uniformly superior performance, followed by *AdaRho*.

Dataset	FAF		Augmentation		AdaRho		RHO-LOSS		HTL		LRL		Uniform	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.375**	**0.402**	0.398	0.418	0.383	0.406	0.386	0.408	0.429	0.436	0.395	0.410	0.402	0.417
ETTh2	**0.285**	**0.335**	0.296	0.344	0.286	0.336	0.294	0.341	0.311	0.364	0.299	0.346	0.293	0.343
ETTm1	0.312	**0.350**	0.388	0.395	**0.307**	0.353	0.319	0.355	0.356	0.390	0.322	0.346	0.392	0.402
ETTm2	**0.172**	**0.250**	0.185	0.269	0.173	0.253	0.173	0.255	0.201	0.284	0.176	0.254	0.180	0.262

gains in forecast performance. For ETTh1, the MSE falls by 9.5%, and large improvements appear across several datasets. On average, Chronos achieves a 5.4% MSE reduction, while Moirai improves by 8.5%, yielding an overall mean reduction of 6.9% in univariate forecasting. MAE and WQL metrics also show significant improvements. These results underscore *FAF*'s substantial benefit on both time-series foundation models. Ultimately, across eight architectures, *FAF* achieves a 6.55% mean reduction in MSE and a 3.79% mean reduction in MAE, averaged across 9 datasets.

Scalability. Since forward passes require at least 3× less computation than a forward-backward pass (Jouppi et al., 2017), *AdaRho* is highly scalable. Across experiments, *AdaRho* increased per-epoch time by no more than 40%. When combined with data augmentation (i.e., the full *FAF* pipeline), the total per-epoch cost is roughly 2× that of the baseline. Overall training times are even lower than these per-epoch ratios. While further speedups are possible by computing losses for the next batch in parallel, this was unnecessary, as most training runs complete within minutes. Moreover, *FAF* is applied only during training, so inference times remain unaffected. For memory, since *FAF* backpropagates only through a subset of data, usage does not increase and may even decrease. We discuss scalability in Appendix E and present runtime results in Table 13.

Ablations. Table 3 highlights the superior performance of *FAF* in ablation. To isolate individual contributions, we apply *AdaRho* and augmentation separately. For comparison, we include the following baselines for data selection: RHO-Loss (Mindermann et al., 2022) for batch selection, HTL (Loshchilov and Hutter, 2015) for high target-loss-based selection, and LRL, which selects samples with the lowest reference losses. Uniform denotes standard random selection. Further details, including an additional ablation with an alternative reference model and an experiment demonstrating the applicability of *FAF* across different forecast horizons, are provided in Appendix F.

Role of adaptivity. To isolate adaptivity, we train the reference model on ETTh2 and apply it to ETTh1, creating a controlled mismatch. AdaRho updates the reference online to reduce this mismatch, whereas RHO-LOSS keeps it fixed. Table 6 shows that AdaRho achieves lower error than both RHO-LOSS and the baseline on ETTh1, consistent with our theory that shrinking $\delta = \|\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}\|_2$ reduces post-filtering risk through $\sigma_\parallel^2(\tau, \delta)$.

Table 4: Dispersion of reducible losses.

Metric	ETTh1	ETTm1	ECL	Traffic
QCD	0.61	0.70	0.18	0.23
CV	1.30	1.58	0.33	0.45

When AdaRho or FAF helps most. AdaRho and FAF help in complementary regimes. When the dataset is large or filtering is mild, performance is mainly noise-limited, and AdaRho captures most of the gain by reducing $\delta = \|\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}\|_2$. When the dataset is smaller or filtering is aggressive, the penalty $\gamma_f(\tau) = \gamma/\pi(\tau)$ becomes more important, and FAF helps by augmenting the retained data. This is consistent with Figure 2b and the larger FAF–AdaRho gap on ETTh1.

5 CONCLUSION

We introduced *FAF*, a model-agnostic, online data curation strategy for time series forecasting that augments data and filters noisy training samples using *AdaRho*. *FAF* improves forecasting accuracy across eight state-of-the-art models and is supported by a theoretical analysis using random matrix theory, which quantifies trade-offs between data quality, noise, and sample size. Despite its effectiveness, *FAF* has limitations. *FAF* struggles on high-dimensional multivariate datasets where reducible loss dispersion is low. Also, the augmentation space is limited. Incorporating more sophisticated techniques could further boost performance. Furthermore, *FAF* has difficulties in low-data regimes. In future work, we aim to address these challenges by computing reducible loss on a per-channel basis and expanding the augmentation space to increase data diversity.

Acknowledgements

This work is supported by the National Science Foundation grants CCF-2550179, CCF-2403075, CCF-2212426, the Office of Naval Research grant N000142412289, a gift from Open Philanthropy, and an Adobe Data Science Research Award. The computational aspects of the research are generously supported by computational resources provided by the Amazon Research Award on Foundation Model Development.

References

Akhtiamov, D., Bosch, D., Ghane, R., Varma, K. N., and Hassibi, B. (2024). A novel gaussian min-max theorem and its applications. *arXiv preprint arXiv:2402.07356*.

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Pineda Arango, S., Kapoor, S., Zschiegner, J., Maddix, D. C., Mahoney, M. W., Torkkola, K., Gordon Wilson, A., Bohlke-Schneider, M., and Wang, Y. (2024). Chronos: Learning the language of time series. *Transactions on Machine Learning Research*.

Bachechi, C., Rollo, F., and Po, L. (2020). Real-time data cleaning in traffic sensor networks. In *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–8. IEEE.

Bandara, K., Hewamalage, H., Liu, Y.-H., Kang, Y., and Bergmeir, C. (2021). Improving the accuracy of global forecasting models using time series data augmentation. *Pattern Recognition*, 120:108148.

Bartlett, P. L., Long, P. M., Lugosi, G., and Tsigler, A. (2020). Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070.

Che, G. (2024). Generative models for financial time series data: Enhancing signal-to-noise ratio and addressing data scarcity in a-share market. *arXiv preprint arXiv:2501.00063*.

Cheema, P. and Sugiyama, M. (2024). Stiefelgen: A simple, model agnostic approach for time series data augmentation over riemannian manifolds. *arXiv preprint arXiv:2402.19287*.

Chen, S., Long, G., Jiang, J., and Zhang, C. (2025). Federated foundation models on heterogeneous time series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(15):15839–15847.

Cheng, H., Wen, Q., Liu, Y., and Sun, L. (2024). RobustTSF: Towards theory and design of robust time series forecasting with anomalies. In *The Twelfth International Conference on Learning Representations*.

Chung, M. K. (2020). Gaussian kernel smoothing. *arXiv preprint arXiv:2007.09539*.

Cipollini, F. and Gallo, G. M. (2025). Multiplicative error models: 20 years on. *Econometrics and Statistics*, 33:209–229.

Das, A., Kong, W., Sen, R., and Zhou, Y. (2024). A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*.

Ding, X., Wang, H., Su, J., Li, Z., Li, J., and Gao, H. (2019). Cleanits: a data cleaning system for industrial time series. *Proceedings of the VLDB Endowment*, 12(12):1786–1789.

Donghao, L. and Xue, W. (2024). ModernTCN: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*.

Du, Y., Wang, J., Feng, W., Pan, S., Qin, T., Xu, R., and Wang, C. (2021). Adarnn: Adaptive learning and forecasting of time series. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 402–411.

Fan, W., Wang, P., Wang, D., Wang, D., Zhou, Y., and Fu, Y. (2023). Dish-ts: a general paradigm for alleviating distribution shift in time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 7522–7529.

Gadre, S. Y., Ilharco, G., Fang, A., Hayase, J., Smyrnis, G., Nguyen, T., Marten, R., Wortsman, M., Ghosh, D., Zhang, J., Orgad, E., Entezari, R., Daras, G., Pratt, S. M., Ramanujan, V., Bitton, Y., Marathe, K., Mussmann, S., Vencu, R., Cherti, M., Krishna, R., Koh, P. W., Saukh, O., Ratner, A., Song, S., Hajishirzi, H., Farhadi, A., Beaumont, R., Oh, S., Dimakis, A., Jitsev, J., Carmon, Y., Shankar, V., and Schmidt, L. (2023). Datacomp: In search of the next generation of multimodal datasets. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. (2020). The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Godahewa, R., Bergmeir, C., Webb, G. I., Hyndman, R. J., and Montero-Manso, P. (2021). Monash time series forecasting archive.

Han, H., Liu, Z., Barrios Barrios, M., Li, J., Zeng, Z., Sarhan, N., and Awwad, E. M. (2024). Time series forecasting model for non-stationary series pattern extraction using deep learning and garch modeling. *Journal of Cloud Computing*, 13(1):2.

Hastie, T., Montanari, A., Rosset, S., and Tibshirani, R. J. (2022). Surprises in high-dimensional ridgeless least squares interpolation. *Annals of statistics*, 50(2):949.

Jain, A., Montanari, A., and Sasoglu, E. (2024). Scaling laws for learning with real and surrogate data. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Information Processing Systems*, volume 37, pages 110246–110289. Curran Associates, Inc.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. (2017). In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12.

Katharopoulos, A. and Fleuret, F. (2018). Not all samples are created equal: Deep learning with importance sampling. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2525–2534. PMLR.

Khuntia, S. R., Rueda, J. L., and Van der Meijden, M. A. (2018). Long-term electricity load forecasting considering volatility using multiplicative error model. *Energies*, 11(12).

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. (2018). Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 95–104, New York, NY, USA. Association for Computing Machinery.

Li, J., Fang, A., Smyrnis, G., Ivgi, M., Jordan, M., Gadre, S., Bansal, H., Guha, E., Keh, S., Arora, K., Garg, S., Xin, R., Muennighoff, N., Heckel, R., Mercat, J., Chen, M., Gururangan, S., Wortsman, M., Albalak, A., Bitton, Y., Nezhurina, M., Abbas, A., Hsieh, C.-Y., Ghosh, D., Gardner, J., Kilian, M., Zhang, H., Shao, R., Pratt, S., Sanyal, S., Ilharco, G., Daras, G., Marathe, K., Gokaslan, A., Zhang, J., Chandu, K., Nguyen, T., Vasiljevic, I., Kakade, S., Song, S., Sanghavi, S., Faghri, F., Oh, S., Zettlemoyer, L., Lo, K., El-Nouby, A., Pouransari, H., Toshev, A., Wang, S., Groeneveld, D., Soldaini, L., Koh, P. W., Jitsev, J., Kollar, T., Dimakis, A. G., Carmon, Y., Dave, A., Schmidt, L., and Shankar, V. (2024). Datacomp-lm: In search of the next generation of training sets for language models. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Information Processing Systems*, volume 37, pages 14200–14282. Curran Associates, Inc.

Lin, Z., Gou, Z., Gong, Y., Liu, X., yelong shen, Xu, R., Lin, C., Yang, Y., Jiao, J., Duan, N., and Chen, W. (2024). Not all tokens are what you need for pretraining. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., and Dustdar, S. (2022). Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M. (2023). itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*.

Loshchilov, I. and Hutter, F. (2015). Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Mindermann, S., Brauner, J. M., Razzak, M. T., Sharma, M., Kirsch, A., Xu, W., Höltgen, B., Gomez, A. N., Morisot, A., Farquhar, S., et al. (2022). Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pages 15630–15649. PMLR.

Nie, Y., H. Nguyen, N., Sinthong, P., and Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*.

Patil, P., Du, J.-H., and Tibshirani, R. (2024). Optimal ridge regularization for out-of-distribution prediction. In *Forty-first International Conference on Machine Learning*.

Rosenblatt, M. (1956). Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832 – 837.

Rossi, B. (2013). Exchange rate predictability. *Journal of economic literature*, 51(4):1063–1119.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., Schramowski, P., Kundurthy, S., Crowson, K., Schmidt, L., Kaczmarczyk, R., and Jitsev, J. (2022). Laion-5b: An open large-scale dataset for training next generation image-text models. In Koyejo, S., Mohamed, S., Agarwal, A.,

Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 25278–25294. Curran Associates, Inc.

Song, Y., Bhattacharya, S., and Sur, P. (2024). Generalization error of min-norm interpolators in transfer learning. *arXiv preprint arXiv:2406.13944*.

Sujit, S., Nath, S., Braga, P., and Ebrahimi Kahou, S. (2023). Prioritizing samples in reinforcement learning with reducible loss. *Advances in Neural Information Processing Systems*, 36:23237–23258.

Taga, E. O., Ildiz, M. E., and Oymak, S. (2025). Timepfn: Effective multivariate time series forecasting with synthetic data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 20761–20769.

Tian, Y., Huffman, G. J., Adler, R. F., Tang, L., Sapiano, M., Maggioni, V., and Wu, H. (2013). Modeling errors in daily precipitation measurements: Additive or multiplicative? *Geophysical Research Letters*, 40(10):2060–2065.

Tulino, A. M. and Verdú, S. (2004). *Random matrix theory and wireless communications*. Now Publishers Inc.

Um, T. T., Pfister, F. M., Pichler, D., Endo, S., Lang, M., Hirche, S., Fietzek, U., and Kulić, D. (2017). Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM international conference on multimodal interaction*, pages 216–220.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. (2022). Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Wang, Z. and Ventre, C. (2024). A financial time series denoiser based on diffusion models. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 72–80.

Wenzek, G., Lachaux, M.-A., Conneau, A., Chaudhary, V., Guzmán, F., Joulin, A., and Grave, E. (2020). CCNet: Extracting high quality monolingual datasets from web crawl data. In Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., and Sahoo, D. (2024). Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning*.

Wu, H., Xu, J., Wang, J., and Long, M. (2021). Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*.

Yoon, T., Park, Y., Ryu, E. K., and Wang, Y. (2022). Robust probabilistic time series forecasting. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I., editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 1336–1358. PMLR.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. (2023). Are transformers effective for time series forecasting? In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press.

Zhang, A., Song, S., Wang, J., and Yu, P. S. (2017). Time series data cleaning: From anomaly detection to anomaly repairing. *Proceedings of the VLDB Endowment*, 10(10):1046–1057.

Zhang, Y. and Yan, J. (2023). Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pages 11106–11115. AAAI Press.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. (2022). FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proc. 39th International Conference on Machine Learning (ICML 2022)*.

Checklist

1. For all models and algorithms presented, check if you include:

   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes. The setting, assumptions, and algorithm are stated clearly in the main body and in the appendix (see Section 2.1, Section 3, and Algorithm 1).

(b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes. We analyzed the properties of filtering in the theory section in detail and we have investigated the complexity of *FAF* in the Experiments section (Section 4) in the main body and also provided runtime results in the Appendix E, Table 13.

(c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes. We provide an anonymized source code that is scripted and commented with specification of all dependencies provided in the *requirements.txt* file.

2. For any theoretical claim, check if you include:

(a) Statements of the full set of assumptions of all theoretical results. Yes. We provide full set of assumptions in the theory sections (Section 3 and Appendix C).

(b) Complete proofs of all theoretical results. Yes. We provide the sketches and definitions in Section 3 and full proofs in Appendix C.

(c) Clear explanations of any assumptions. Yes. Section 3 and Appendix C elaborates on assumptions.

3. For all figures and tables that present empirical results, check if you include:

(a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes. The code is provided as part of supplementary material and datasets we used are explicitly cited with the instructions to download them in the code. We also provided instructions to run the experiments in the code.

(b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes. We provide the training and hyperparameter details in Section 4 and Appendix D.

(c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes. We run the experiments with fixed seed and used the metrics of MSE, MAE and WQL. In the Appendix D we further elaborated on these metrics. As we conduct experiments on 8 architectures and 9 datasets, and require training of the baseline model and their *FAF*-applied counterparts, we could not report averages of multiple seeds due to prohibitively expensive and time consuming training of such large scale training runs. Yet, the improvements of *FAF* are highly significant on range of models and datasets. We believe that the results clearly demonstrate the benefit of *FAF* in the pipeline.

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes. We detailed the computing infrastructure in Appendix D.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

(a) Citations of the creator If your work uses existing assets. Yes. We used public benchmark datasets and we clearly cited the origins of the datasets in Section 4 and in the Appendix D.

(b) The license information of the assets, if applicable. Not Applicable. We used public benchmarks and cited their origins.

(c) New assets either in the supplemental material or as a URL, if applicable. Yes. We included the codebase in the supplementary material.

(d) Information about consent from data providers/curators. Not Applicable. As stated, we used publicly available benchmark datasets and cited their origins. Explicit consent is not required in our case.

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable. We do not have sensible content.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. Not Applicable. We did not use crowdsourcing.

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable.

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable.

Appendix

The appendix is organized as follows:

1. Appendix A elaborates on the related work.
2. Appendix B discusses the noise robustness of *AdaRho*.
3. Appendix C extends the theoretical discussion of *AdaRho*.
4. Appendix D details the experimental setup, including datasets, implementation specifics, reference models, augmentation methods, and computational aspects.
5. Appendix E presents additional results, both qualitative and quantitative.
6. Appendix F provides further ablation studies, including evaluations with an alternative reference model, longer horizon forecasts, and expanded analyses of the main ablations.

A Extended Related Work

Our work fits within the broader framework of data selection and curation (Wenzek et al., 2020; Gao et al., 2020; Schuhmann et al., 2022), where filtering is a common component of the data pipeline. We specifically focus on sample-level selection, building on prior work (Loshchilov and Hutter, 2015; Mindermann et al., 2022; Schaul et al., 2015; Katharopoulos and Fleuret, 2018), and adapt these ideas to time series forecasting, in particular the RHO-LOSS (Mindermann et al., 2022).

As discussed previously, although there are works such as Bachechi et al. (2020); Zhang et al. (2017); Ding et al. (2019) that incorporate anomalies and/or sensor errors in time series contexts, our work differs from the existing literature in several key ways. First, we do not focus on sensor noise (Bachechi et al., 2020) or explicitly anomalous datasets (Zhang et al., 2017). Instead, motivated by the observation that standard time series datasets often suffer from data quality issues, we propose a novel approach that builds on RHO-LOSS (Mindermann et al., 2022) and extends it to the time series domain by making it adaptive and augmenting it with time series-specific data augmentations. RHO-LOSS-based methods have already been applied to computer vision (Mindermann et al., 2022), language modeling (Lin et al., 2024), and reinforcement learning (Sujit et al., 2023). In that context, our core contribution is the development of a RHO-LOSS-based framework tailored to time series forecasting, which addresses the limitations of existing methods and introduces an adaptive and augmentation-included variant, *FAF*, specifically designed for this domain.

Moreover, unlike prior work, our approach does not rely on explicitly modeling noise or anomalies. It is fully integrated into the training process and requires minimal tuning or assumptions about noise or data quality. While previous work has demonstrated the benefits of data augmentation in time series forecasting (Bandara et al., 2021), our key contribution in *FAF* from that aspect lies in coupling data filtering and augmentation to mitigate the effects of reduced sample size.

B AdaRho's Noise Robustness

In Section 2.2, we presented a mechanistic experiment demonstrating the robustness of *AdaRho* to multiplicative noise. To simulate sensor errors, we corrupted sampled points from the ETT(h/m)(1/2) datasets with randomly generated Gaussian multiplicative noise. We used the official iTransformer hyperparameters from (Liu et al., 2023) for all experiments. For each dataset, we trained a reference model, also an iTransformer with the same hyperparameters, on a randomly selected 25% subset of the data. The training sets were corrupted with Gaussian multiplicative noise, while the validation and test sets remained clean.

Algorithm 1 *AdaRho*: Reducible loss-based sample selection with dual model updates

Input: Target model f_θ, reference model $f_{\theta'}$ with pretrained parameters θ'^0, target batch size n_b, reference update size $n_r > 0$, batch size $n_B > n_b + n_r$, learning rates $\eta \gg \eta'$
Output: Optimized parameters θ

Initialize θ^0, $t = 0$
for $t = 0, 1, \ldots$ **do**
  Sample batch B_t of size n_B
  Compute `TargetLoss[i]` $= \ell(f_\theta(x_i), y_i)$, $\forall i \in B_t$
  Compute `RefLoss[i]` $= \ell(f_{\theta'}(x_i), y_i)$, $\forall i \in B_t$
  Compute `ReducibleLoss[i]` $\leftarrow$ `TargetLoss[i]` $-$ `RefLoss[i]`, $\forall i \in B_t$
  Sort B_t by `ReducibleLoss[i]` in descending order
  $b_t^{\text{target}} \leftarrow$ top-n_b samples from sorted B_t
  $b_t^{\text{ref}} \leftarrow$ samples ranked $n_b{+}1$ to $n_b + n_r$ in sorted B_t
  $g_t \leftarrow$ mini-batch gradient on b_t^{target} w.r.t. θ
  $g'_t \leftarrow$ mini-batch gradient on b_t^{ref} w.r.t. θ'
  $\theta^{t+1} \leftarrow \theta^t - \eta g_t$
  $\theta'^{t+1} \leftarrow \theta'^t - \eta' g'_t$
end for

Table 5: MSE and MAE values for Baseline and AdaRho under different noise levels on ETT(h/m)(1/2), including percentage improvements of AdaRho over Baseline. Results are averaged over 4 datasets.

Noise Level	Baseline		AdaRho		% Improvement	
	MSE	MAE	MSE	MAE	MSE	MAE
0.8	0.388	0.399	0.359	0.383	+7.4%	+4.0%
0.6	0.354	0.382	0.322	0.361	+9.0%	+5.3%
0.4	0.324	0.364	0.310	0.353	+4.3%	+3.0%
0.2	0.317	0.359	0.306	0.351	+3.5%	+2.2%
0.0	0.304	0.350	0.291	0.338	+4.0%	+3.2%

In *AdaRho*, we selected the top 25% of samples with the highest reducible loss to train the target model and samples in the 25–37.5% range to train the reference model.

Multiplicative label noise is a realistic assumption when measurement uncertainty or intrinsic variability scales with the magnitude of the signal, commonly seen in domains such as finance, energy load forecasting, and physical sensors (Cipollini and Gallo, 2025; Khuntia et al., 2018; Tian et al., 2013). We compare the native model (i.e., trained directly with stochastic gradient descent) with the model enhanced by *AdaRho*. Both models were trained using identical hyperparameters, and we report the evaluation metrics corresponding to the epoch with the lowest validation loss. In Figure 3 and Table 5, we present the raw MSE and MAE values from our mechanistic study.

As shown in Table 5, *AdaRho* effectively filters out samples corrupted with multiplicative noise, capturing realistic measurement-based errors. Notably, the largest improvement occurs at a noise level of 0.6, exceeding that of 0.8. This is expected: since only the top 25% of high-loss samples are used to train the target model, and 80% of the data is noisy at the 0.8 noise level, at least 5% of the selected training samples are inevitably corrupted.

This also justifies our use of a dataset-dependent threshold k (i.e., the percentage of top reducible-loss samples used for target model updates) based on validation performance in real-world experiments. Because real-world datasets vary in their noise characteristics, choosing a fixed k across all datasets may not be optimal. Nonetheless, as evidenced by Table 5, even with a fixed $k = 25$, *AdaRho* achieves substantial gains, even under heavy noise where some selected samples are still corrupted.

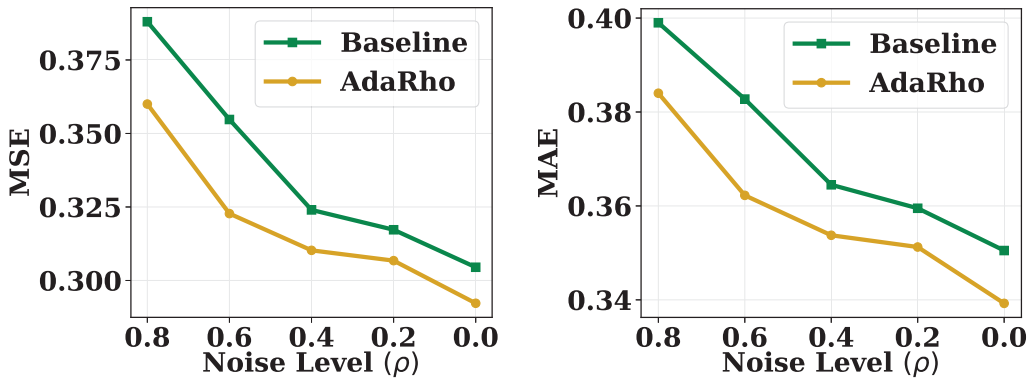


Figure 3: *AdaRho* performance across increasing multiplicative noise levels, measured by MSE (left) and MAE (right).

C Further Theoretical Discussions

Recall from Section 3 that the threshold condition on data point i from noise component j ($j = 1, 2$) is:

$$\left|\boldsymbol{x}_i^\top(\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}) + \sigma_j\, z_i\right| \le \tau.$$

Because $\boldsymbol{x}_i \sim \mathcal{N}(0, \boldsymbol{I}_p)$ and $\delta := \|\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}\|$, we have $\boldsymbol{x}_i^\top(\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}) = \delta\, g_i$ with $g_i \sim \mathcal{N}(0,1)$. Thus, the above thresholding event equivalently becomes $\{|\delta\, g_i + \sigma_j\, z_i| \le \tau\}$. We then define the following two datasets, containing filtered data points from the two noise mixtures:

$$\begin{aligned} D_1 := \Big\{(x_i, y_i) \;:\; & i \in [n],\; y_i = x_i^\top \boldsymbol{\beta}_\star + \sigma_1 z_i, \\ & |x_i^\top(\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}) + \sigma_1 z_i| \le \tau\Big\} \\ D_2 := \Big\{(x_i, y_i) \;:\; & i \in [n],\; y_i = x_i^\top \boldsymbol{\beta}_\star + \sigma_2 z_i, \\ & |x_i^\top(\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}) + \sigma_2 z_i| \le \tau\Big\} \end{aligned}$$

We know that the frequencies of D_1 and D_2 in the filtered dataset are proportional to $p_1\Pi_{\delta,\sigma_1}(\tau)$ and $p_2\Pi_{\delta,\sigma_2}(\tau)$ as $n \to \infty$, respectively. Although z and δg (the projection of $\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}$) are initially independent in the regression model, the variables $\delta\, g$ and z become correlated after the filtering (thresholding). Using the shorthand notation $\zeta_j := \delta\, g + \sigma_j\, z$, and following the discussions in Section 3, the noise can be decomposed into two parts for $j = 1, 2$:

$$\begin{aligned} \sigma_{j,\perp}^2(\tau, \delta) &:= \sigma_j^2\, \mathrm{Var}\big(z \,\big|\, |\zeta_j| \le \tau\big), \\ \sigma_{j,\parallel}^2(\tau, \delta) &:= \sigma_j^2\, \frac{\mathrm{Cov}\big(z, \delta g \,\big|\, |\zeta_j| \le \tau\big)^2}{\mathrm{Var}\big(\delta g \,\big|\, |\zeta_j| \le \tau\big)}. \end{aligned}$$

Finally, we let the frequency-weighted effective variances in the filtered dataset be decomposed as:

$$\begin{aligned} \sigma_\perp^2(\tau, \delta) &= \frac{p_1\Pi_{\delta,\sigma_1}(\tau)\, \sigma_{1,\perp}^2(\tau, \delta) + p_2\Pi_{\delta,\sigma_2}(\tau)\, \sigma_{2,\perp}^2(\tau, \delta)}{\pi(\tau)}, \\ \sigma_\parallel^2(\tau, \delta) &= \frac{p_1\Pi_{\delta,\sigma_1}(\tau)\, \sigma_{1,\parallel}^2(\tau, \delta) + p_2\Pi_{\delta,\sigma_2}(\tau)\, \sigma_{2,\parallel}^2(\tau, \delta)}{\pi(\tau)}. \end{aligned} \tag{11}$$

In the following parts, we will show how we derive the above formulas for $\sigma_\perp^2(\tau, \delta)$ and $\sigma_\parallel^2(\tau, \delta)$. Defining

$\boldsymbol{\beta}_{\text{diff}} := \boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}$, we write the filtered datasets D_1 and D_2 as:

$$D_1 := \big\{(x_i, y_i) : i \in [n],\ \big|x_i^\top \boldsymbol{\beta}_{\text{diff}} + \sigma_1 z_i\big| \le \tau,\\ y_i = x_i^\top \big(\boldsymbol{\beta}_\star + \alpha_1(\tau,\delta)\boldsymbol{\beta}_{\text{diff}}\big) + \sigma_{1,\perp}(\tau,\delta)\, z_i\big\},$$

$$D_2 := \big\{(x_i, y_i) : i \in [n],\ \big|x_i^\top \boldsymbol{\beta}_{\text{diff}} + \sigma_2 z_i\big| \le \tau,\\ y_i = x_i^\top \big(\boldsymbol{\beta}_\star + \alpha_2(\tau,\delta)\boldsymbol{\beta}_{\text{diff}}\big) + \sigma_{2,\perp}(\tau,\delta)\, z_i\big\}.$$

where $\alpha_j(\tau,\delta)$ are given by:

$$\alpha_j(\tau,\delta) = \sqrt{\frac{\sigma_{j,\parallel}^2(\tau,\delta)}{\text{Var}(\delta g \mid |\zeta_j| \le \tau)}}, \quad \text{for } j = 1, 2.$$

This means the post-filtering data consists of two datasets with different parameters and noise levels.

Key Approximation: In the large-sample limit $n \to \infty$, each group's proportion $\frac{p_j \Pi_{\delta,\sigma_j}(\tau)}{\pi(\tau)}$ is fixed, so we approximate the combined filtered data by a single regression with parameter

$$\boldsymbol{\beta}_f = \boldsymbol{\beta}_\star + \left(\sum_{j=1}^{2} \frac{p_j\, \Pi_{\delta,\sigma_j}(\tau)}{\pi(\tau)}\, \alpha_j(\tau,\delta)\right) \boldsymbol{\beta}_{\text{diff}},$$

which is the frequency-weighted average of two parameters. Define $\alpha(\tau,\delta) = \left(\sum_{j=1}^{2} \frac{p_j\, \Pi_{\delta,\sigma_j}(\tau)}{\pi(\tau)}\, \alpha_j(\tau,\delta)\right)$ so that we compactly write $\boldsymbol{\beta}_f = \boldsymbol{\beta}_\star + \alpha(\tau,\delta)\, \boldsymbol{\beta}_{\text{diff}}$. The errors in our analysis at finite n arise from the approximation since we treat the two subsets as if they shared a single parameter. However, our approximation is reasonable in the sense that, in the infinite sample size $n \to \infty$ regime, each shift $\alpha_j(\tau,\delta)\, \boldsymbol{\beta}_{\text{diff}}$ from $\boldsymbol{\beta}_\star$ will be combined in a frequency-weighted manner, and the approximation becomes exact.

We now consider the proportional limit regime $n, p \to \infty$ with $p/n \to \gamma$. As shown in (Hastie et al., 2022), for the standard linear regression problem $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2$ with ground-truth model $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}_\star + \boldsymbol{\xi}$ with the label noise $\boldsymbol{\xi}$ having i.i.d. zero mean and σ^2 variance entries, the asymptotic risk converges to:

$$\mathcal{R}_{\boldsymbol{X}}(\hat{\boldsymbol{\beta}}; \boldsymbol{\beta}_\star) \to \begin{cases} \sigma^2 \frac{\gamma}{1-\gamma}, & \gamma < 1, \\ \|\boldsymbol{\beta}_\star\|_2^2 \left(1 - \frac{1}{\gamma}\right) + \frac{\sigma^2}{\gamma - 1}, & \gamma > 1 \end{cases} \tag{12}$$

We know that $\hat{\boldsymbol{\beta}}_f - \boldsymbol{\beta}_\star = \boldsymbol{X}_f{}^\dagger \boldsymbol{X}_f \boldsymbol{\beta}_f + \boldsymbol{X}_f{}^\dagger \boldsymbol{\xi}_f - \boldsymbol{\beta}_\star = (\boldsymbol{X}_f{}^\dagger \boldsymbol{X}_f - \boldsymbol{I})\, \boldsymbol{\beta}_\star + \alpha(\tau,\delta) \boldsymbol{X}_f{}^\dagger \boldsymbol{X}_f \boldsymbol{\beta}_{\text{diff}} + \boldsymbol{X}_f{}^\dagger \boldsymbol{\xi}_f$, where $\boldsymbol{\xi}_f := [\xi_i]_{i \in \mathcal{I}(\tau)} \in \mathbb{R}^{n_f}$ is the noise vector on the filtered dataset and $\hat{\boldsymbol{\beta}}_f$ is the estimate obtained from the filtered dataset. Recall the definition of risk from Section 3:

$$\begin{aligned} \mathcal{R}_{\boldsymbol{X}}(\hat{\boldsymbol{\beta}}; \boldsymbol{\beta}) &= \mathbb{E}\left[(\boldsymbol{x}_0^\top \boldsymbol{\beta}_\star - \boldsymbol{x}_0^\top \hat{\boldsymbol{\beta}})^2 \mid \boldsymbol{X}\right] \\ &= \mathbb{E}\left[(\boldsymbol{\beta}_\star - \hat{\boldsymbol{\beta}})^\top (\boldsymbol{\beta}_\star - \hat{\boldsymbol{\beta}}) \mid \boldsymbol{X}\right]. \end{aligned} \tag{13}$$

Plugging the above $\hat{\boldsymbol{\beta}}_f - \boldsymbol{\beta}_\star$ into (13) and recalling our assumption $\boldsymbol{\beta}_\star \perp \boldsymbol{\beta}_{\text{diff}}$, we see that the contribution of the term $\alpha(\tau,\delta) \boldsymbol{X}_f{}^\dagger \boldsymbol{X}_f \boldsymbol{\beta}_{\text{diff}}$ to the final risk is decoupled from the rest. As can be seen from the asymptotic risks for standard linear regression in (12), for the overparameterized regime, the bias error term is $\|\boldsymbol{\beta}_\star\|_2^2 \left(1 - \frac{1}{\gamma}\right)$, while for the underparameterized regime there's no bias error term in the final asymptotic risk. Accordingly, the contribution of the term $\alpha(\tau,\delta) \boldsymbol{X}_f{}^\dagger \boldsymbol{X}_f \boldsymbol{\beta}_{\text{diff}}$ is shrunk by $1/\gamma_f(\tau)$ when the filtering results in an overparameterized setting $\gamma_f(\tau) > 1$. On the other hand, if $\gamma_f(\tau) < 1$, there is no shrinkage in its contribution. Consequently, in the under-parametrized region, the contribution of $\alpha(\tau,\delta)\, \boldsymbol{\beta}_{\text{diff}}$ to the risk as a bias term is:

$$\begin{aligned} &\left(\sum_{j=1}^{2} \frac{p_j\, \Pi_{\delta,\sigma_j}(\tau)}{\pi(\tau)}\, \alpha_j(\tau,\delta)\right) \sqrt{\text{Var}\big(\delta g \mid |\zeta_j| \le \tau\big)} \\ &= \frac{p_1 \Pi_{\delta,\sigma_1}(\tau)\, \sigma_{1,\parallel}^2(\tau,\delta) + p_2 \Pi_{\delta,\sigma_2}(\tau)\, \sigma_{2,\parallel}^2(\tau,\delta)}{\pi(\tau)} \\ &= \sigma_\parallel^2(\tau,\delta). \end{aligned} \tag{14}$$

In the over-parameterized region, the contribution of $\alpha(\tau,\delta)\,\boldsymbol{\beta}_{\text{diff}}$ to the risk as a bias term gets shrunk by the $1/\gamma_f(\tau)$, and therefore:

$$\begin{aligned}
&\frac{1}{\gamma_f(\tau)}\Big(\sum_{j=1}^{2}\frac{p_j\,\Pi_{\delta,\sigma_j}(\tau)}{\pi(\tau)}\,\alpha_j(\tau,\delta)\Big)\sqrt{\mathrm{Var}\big(\delta g \mid |\zeta_j|\le\tau\big)}\\
&=\frac{p_1\Pi_{\delta,\sigma_1}(\tau)\,\sigma_{1,\|}^2(\tau,\delta)+p_2\Pi_{\delta,\sigma_2}(\tau)\,\sigma_{2,\|}^2(\tau,\delta)}{\pi(\tau)\,\gamma_f(\tau)}\\
&=\frac{\sigma_{\|}^2(\tau,\delta)}{\gamma_f(\tau)}.
\end{aligned}\tag{15}$$

On the other hand, the orthogonal noise terms $\sigma_{j,\perp}(\tau,\delta)\,z$ for $j=1,2$ behave as i.i.d. noise in the filtered system. Hence, combining this fact with (14) and (15), we state the following result as $n\to\infty$:

$$\mathcal{R}_{\boldsymbol{X}_f}(\hat{\boldsymbol{\beta}}_f;\boldsymbol{\beta}_\star)\;\approx\;\begin{cases}\sigma_\perp^2(\tau,\delta)\,\frac{\gamma_f(\tau)}{1-\gamma_f(\tau)}+\sigma_{\|}^2(\tau,\delta), & \text{if } \gamma_f(\tau)<1,\\[2ex] \|\boldsymbol{\beta}_\star\|_2^2\left(1-\frac{1}{\gamma_f(\tau)}\right)+\frac{\sigma_\perp^2(\tau,\delta)}{\gamma_f(\tau)-1}+\frac{\sigma_{\|}^2(\tau,\delta)}{\gamma_f(\tau)}, & \text{if } \gamma_f(\tau)>1\end{cases}$$

Further Discussion on the Validity of Approximation: As $\boldsymbol{\beta}_{\text{ref}}$ approaches $\boldsymbol{\beta}_\star$, the difference $\boldsymbol{\beta}_{\text{diff}}=\boldsymbol{\beta}_\star-\boldsymbol{\beta}_{\text{ref}}$ shrinks, causing the shifts $\alpha_j(\tau,\delta)\,\boldsymbol{\beta}_{\text{diff}}$ in each subset to become negligible. In this regime, the overall noise scales σ_1,σ_2 dominate, and the single-parameter approximation with $\boldsymbol{\beta}_f\approx\boldsymbol{\beta}_\star$ becomes more accurate. Although our derivation does not strictly prove optimality in finite-sample settings, the frequency-weighted structure provides a reasonable proxy that aligns well with empirical results in the large-sample limit.

Remark on the orthogonality approximation. Throughout the analysis, we assume $\boldsymbol{\beta}_\star\perp(\boldsymbol{\beta}_{\text{ref}}-\boldsymbol{\beta}_\star)$. This assumption is mainly introduced for tractability, since it yields a clean decomposition of the post-filtering error into components parallel and orthogonal to the difference direction. It is also a natural first-order approximation in high dimensions: writing $\boldsymbol{\beta}_{\text{ref}}=\boldsymbol{\beta}_\star+\boldsymbol{\beta}_{\text{diff}}$, if the perturbation direction $\frac{\boldsymbol{\beta}_{\text{diff}}}{\|\boldsymbol{\beta}_{\text{diff}}\|_2}$ is approximately isotropic, then its inner product with $\frac{\beta_\star}{\|\boldsymbol{\beta}_\star\|_2}$ is $O_p(p^{-1/2})$, so the cross term $\boldsymbol{\beta}_\star{}^\top\big(\boldsymbol{\beta}_{\text{ref}}-\boldsymbol{\beta}_\star\big)$ is lower-order relative to the leading quadratic terms.

D Experimental Details

D.1 Datasets

We evaluate *FAF* across standard multivariate time-series forecasting benchmarks. In the univariate setting, we use the target variable as the response and exclude other covariates. The datasets include ECL (Electricity Consumption Load), Exchange, Traffic, Weather, and Solar Energy. In addition, four datasets are from the Electricity Transformer Temperature (ETT) collection introduced by Zhou et al. (2021): ETTh1, ETTh2, ETTm1, and ETTm2. Except for the Solar Energy dataset, originally introduced by Lai et al. (2018), all other benchmarks were evaluated by Autoformer (Wu et al., 2021). Following common practice (Nie et al., 2023; Liu et al., 2023; Taga et al., 2025), we split each dataset chronologically into training, validation, and test sets. Key details for each dataset are summarized below (split sizes assume context length = 96, input length = 96). Overall, our approach adheres to standard time-series forecasting practices.

ECL. The Electricity Consumption Load dataset (Wu et al., 2021) records hourly power usage from 321 customers (321 variables). The dataset includes 18,317 training samples, 2,633 for validation, and 5,261 for testing. Due to its highly multivariate nature, architectures such as iTransformer and TimePFN—which explicitly incorporate multivariate interactions—demonstrate strong performance on this dataset.

Exchange. This dataset consists of daily exchange rates from eight countries (eight variables), with 5,120 training samples, 665 for validation, and 1,422 for testing. Although commonly excluded (e.g., (Nie et al., 2023)) due to the noisy nature of financial data and the strong performance of naive models (Zeng et al., 2023; Rossi, 2013), it serves as a valuable testbed for evaluating *FAF* under high-noise and low-predictability conditions. Our experiments show that *FAF* significantly reduces overfitting to noise in models such as Informer (Zhou et al., 2021) and TimePFN (Taga et al., 2025). The strongest baseline on this dataset was PatchTST, likely due to

its channel-independent design and shared backbone, which spreads gradient updates and mitigates overfitting through averaging.

Solar Energy. Introduced by (Lai et al., 2018), the Solar Energy dataset includes measurements from 137 solar stations sampled every 10 minutes. It contains 36,601 training samples, 5,161 for validation, and 10,417 for testing. Day-night cycles induce sharp spikes and drops aligned with sunrise and sunset. Informer (Zhou et al., 2021) performs surprisingly well, possibly due to its ProbSparse attention mechanism capturing abrupt time-dependent patterns. We observe that *FAF* particularly improves the performance of weaker models by prioritizing harder, spiky patterns over simpler periodic components.

ETT Datasets. The ETT collection (Zhou et al., 2021) includes seven variables measuring transformer temperature-related signals. For the hourly datasets (ETTh1 and ETTh2), we use 8,545 training samples, 2,881 for validation, and 2,881 for testing. For the 15-minute datasets (ETTm1 and ETTm2), the splits are 34,465 for training, 11,521 for validation, and 11,521 for testing. Across these datasets, we observe substantial performance improvements with *FAF* in multivariate forecasting models.

Traffic. This dataset, collected from 862 sensors measuring hourly road occupancy (862 variables), provides 12,185 training samples, 1,757 for validation, and 3,509 for testing (Wu et al., 2021). It is the highest-dimensional dataset in our benchmarks. Similar to ECL, architectures that explicitly model multivariate relationships tend to perform better.

Weather. The Weather dataset (Wu et al., 2021) consists of 21 meteorological variables recorded every 10 minutes. We use 36,792 training samples, 5,271 for validation, and 10,540 for testing. Convolution-based models such as ModernTCN perform well here, possibly due to their ability to capture local dependencies through smoothing and lag-based operations in this noisy and highly time-dependent dataset.

Note that for univariate forecasting tasks, we use only the target variable, effectively modeling the datasets as univariate. In these settings, the input length is set to 96, and the forecast horizon to 36.

D.2 Implementation Details and Hyperparameters

We used the official codebases of the respective models to ensure consistency with the literature. Specifically, in the multivariate setting, we used iTransformer (Liu et al., 2023), PatchTST (Nie et al., 2023), TimePFN (Taga et al., 2025), Informer (Zhou et al., 2021), and ModernTCN (Donghao and Xue, 2024).

In the univariate setting, we relied on the codebases of Chronos (Ansari et al., 2024) and Moirai (Woo et al., 2024), and used the Chronos-bolt-base and Moirai-base models, respectively. We used AdamW (Loshchilov and Hutter, 2017) as the optimizer.

When reporting *FAF* results, we fixed the model hyperparameters and varied the values of k and r, which determine the subset of samples (ranked by highest reducible loss) used to train the target (top $k\%$) and reference (next $r\%$) models. We set $r = \frac{k}{2}$ by default, and for $k = 75$, we set $r = 20$. The value of $k \in \{25, 50, 75\}$ was selected based on the lowest validation MSE. Furthermore, for a fair comparison, we run the native models (i.e., baseline) and their *FAF*-applied counterparts the same number of epochs and chose the lowest validation loss-yielding checkpoint to report the final scores.

D.3 Reference Model

As reference models, we selected Chronos-bolt-base (Ansari et al., 2024) for the univariate setting and TimePFN (Taga et al., 2025) for the multivariate setting. Both models were trained on a randomly subsampled 25% of the training points for each dataset, with separate models trained for each dataset. We fixed these two reference models to streamline comparisons across different models under *FAF*, and to demonstrate that a single reference model can improve the performance of various target models, thereby reducing the computational overhead of training additional reference models.

However, the space of potential reference models is much broader. Simpler models such as DLinear (Zeng et al., 2023) or smaller versions of existing architectures could also serve effectively as reference models.

As discussed in Section 3, the quality of the reference model plays a crucial role in robust data selection. Therefore, we expect the reference model to perform reasonably well on the data itself. Since the reference model is trained

on a subset of the data, models that generalize well under limited data budgets are more likely to yield better performance with *AdaRho*. Conversely, if the reference model performs poorly, *AdaRho*-based filtering should be applied less aggressively (i.e., filter out fewer samples), as its reliability in selecting informative examples diminishes.

TimePFN is a strong choice as a reference model because it is pretrained on large-scale synthetic data and has been shown to perform well under low-data regimes, making its learned parameters competitive. Similarly, pretrained foundational models such as Chronos or Moirai also serve as highly effective reference models for the same reason. This suggests that synthetic pretraining, or pretraining through other means, can improve the generalization ability of various reference model architectures in low-data regimes. Nevertheless, pretraining is not strictly necessary. In the Appendix F, we demonstrate that iTransformer (Liu et al., 2023) can also serve effectively as a reference model without pretraining. In Table 7, we show the performances of the reference model architectures. In the univariate setting, the architecture used is Chronos-bolt-base, whereas for the multivariate setting it is TimePFN. The lowest validation loss yielding epoch is chosen as the reference model.

D.4 Metrics

Mean Squared Error (MSE) and Mean Absolute Error (MAE). To evaluate point forecasts, we use the mean squared error (MSE) and mean absolute error (MAE), i.e., two standard regression metrics. Given a set of forecasts $\hat{x}_{i,t}$ and ground-truth observations $x_{i,t}$ across series i and forecast horizons t, they are defined as

$$\mathrm{MSE} = \frac{1}{N}\sum_{i,t}(x_{i,t} - \hat{x}_{i,t})^2, \qquad \mathrm{MAE} = \frac{1}{N}\sum_{i,t}|x_{i,t} - \hat{x}_{i,t}|,$$

where N is the total number of forecasted points. MSE penalizes large deviations more heavily, making it sensitive to outliers, while MAE measures average absolute deviations and is more robust to extreme errors. Both metrics are widely used in time series forecasting (Nie et al., 2023; Liu et al., 2023; Zhou et al., 2021) to evaluate the accuracy of point forecasts.

Weighted Quantile Loss (WQL). We use WQL to assess probabilistic forecasts based on predicted quantiles for Chronos and Moirai, following Ansari et al. (2024). For a predicted α-quantile q of an observation x, the quantile loss is:

$$\mathrm{QL}_\alpha(q, x) = \begin{cases} \alpha(x - q), & x > q, \\ (1-\alpha)(q - x), & x \le q, \end{cases}$$

which penalizes under- and over-predictions asymmetrically. To evaluate forecasts over multiple series and horizons, we compute a weighted summation

$$\mathrm{WQL}_\alpha = \frac{2}{\sum_{i,t}|x_{i,t}|}\sum_{i,t}\mathrm{QL}_\alpha(q_{i,t}^{(\alpha)}, x_{i,t}),$$

where $q_{i,t}^{(\alpha)}$ denotes the predicted α-quantile for series i at time t. Finally, we average over a grid of quantile levels $\{\alpha_1, \ldots, \alpha_Q\}$, yielding

$$\mathrm{WQL} = \frac{1}{Q}\sum_{j=1}^{Q}\mathrm{WQL}_{\alpha_j}.$$

In practice, we set the quantile levels as $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ following the common practice outlined by Ansari et al. (2024).

D.5 Compute

We conducted all experiments on L40S GPUs with 48GB of memory, using a slurm cluster system. While all experiments fit within a single GPU, we used 4–5 GPUs in parallel to speed up the experiments (each running on a seperate GPU). The total training time varies by dataset size: high-dimensional datasets like Traffic require couple of hours, whereas smaller datasets such as the ETT variants take only minutes. As we report results across 8 models and 9 datasets, along with the development of our method, the total GPU usage exceeded 300 hours for this paper. The experimental codebase is implemented entirely in PyTorch.

E Additional Results

E.1 Quantitative Results

Comparison to architectural variations. To demonstrate how *FAF* compares with various architectures, we compared many architectural variations and *FAF* applied TimePFN in Figure 2c. We present the table form of the results in Table 8. All in all, one sees that compared to various architectural variations, *FAF* yields state-of-the-art results.

We see from the table that *FAF* strengthens the TimePFN architecture, even more so in domains such as Solar or ETTm1 where it lacks the good baseline performance. In the experiments, we used the official codebases with input and forecast horizons set to 96. The table clearly demonstrates that the application of *FAF* on TimePFN yields a state-of-the art architecture uniformly among baselines. This is surprising as different architectures have different strengths, yet the application of *FAF* yields a uniformly SOTA architecture.

Comparison of *FAF* and the Reference Model. In Table 1 and Table 2, we did not include the performance of the reference models when comparing with *FAF*. In Table 9 and Table 10, we present how *FAF* performs relative to *FAF*-applied TimePFN and *FAF*-applied Chronos, where the target models are TimePFN and Chronos, respectively.

As shown in Table 9, the performance of *FAF* significantly surpasses that of the reference model, demonstrating that *AdaRho* not only improves upon the baseline model but also outperforms the reference model itself. Notably, in a few cases—such as ETTh1—the reference model slightly outperforms the baseline. This outcome is expected, as reference models are trained on a randomly sampled 25% subset of the training data. Such subsampling may inadvertently exclude noisy training points that do not contribute to accurate forecasting, thereby improving performance. Nonetheless, even in these cases, the target model trained with *FAF* consistently outperforms the reference model by a substantial margin.

A similar pattern holds for Chronos-Bolt as shown in Table 10: *FAF* consistently outperforms the reference model, even in cases where the reference model is more accurate than the baseline.

The Dispersion of Reducible Loss. As shown in Table 1, in the multivariate benchmarks, the relative improvement of *FAF* is greater in datasets with fewer channels than in those with a larger number of channels. As discussed in Section 4, this is primarily due to statistical reasons: in high-dimensional datasets, losses are averaged across many channels, resulting in lower dispersion and thus less effective filtering. Table 4 demonstrates dispersion metrics, namely, the coefficient of variation and the quartile coefficient of dispersion, computed during training and confirm that reducible losses in `ECL` and `Traffic` are far less dispersed than in the `ETT` datasets.

Quantile coefficient of dispersion (QCD) and the coefficient of variation (CV) are widely used measures of data dispersion. QCD is more robust and is defined as $\text{QCD} = \frac{Q_3 - Q_1}{Q_3 + Q_1}$, where Q_1 and Q_3 are the first and third quartiles, respectively. Because it relies on quartiles, QCD is less sensitive to outliers. In contrast, CV is defined as $\text{CV} = \frac{\sigma}{\mu}$, the ratio of the standard deviation to the mean.

Table 4 shows that when data dispersion is low (e.g., when all data points are equally informative or when noise cancels out due to averaging), filtering provides less performance improvement. This offers a straightforward conceptual guideline for understanding where and when to apply *FAF*.

Scalability. Our primary focus in this work is not computational speed-up, since many time series models already train quickly, on the order of minutes or a few hours, which is significantly faster than typical NLP or vision workloads. Moreover, *FAF* is applied only during training, and inference proceeds identically to the baseline.

That said, the RHO-LOSS approach, as demonstrated by prior work (Mindermann et al., 2022; Lin et al., 2024), can lead to substantial speed-ups. Forward passes are at least 3× cheaper than backward passes (Jouppi et al., 2017), and when filtering is non-adaptive, the reference model's forward pass can be precomputed once at the start of training. If the filtering strategy selects, for instance, only 25% of each batch for training, then backward computation is applied to just those samples. This leads to a runtime reduction of

$$\frac{1}{3}\ (\text{forward pass}) + \frac{1}{4}\ (\text{forward-backward pass}) \approx 0.58,$$

which corresponds to a 42% reduction in training time.

In contrast, our method *AdaRho* requires training the target model on $k\%$ of the data and the reference model on an additional $r\%$ (i.e., between $k\%$ and $(k+r)\%$), leading to backpropagation through $(k+r)\%$ of the batch. For example, with $k = 25$ and $r = 12.5$, only 37.5% of the data undergoes backpropagation. However, since *AdaRho* is adaptive, we must compute forward passes for the reference model at every batch. Under these settings, approximate per-batch cost is

$$\frac{2}{3}\,(\text{forward passes}) + \frac{3}{8}\,(\text{forward-backward passes}) \approx 0.70,$$

implying a 30% speed-up. Even in the worst case (e.g., $k = 75$), the overhead is only about 66%. Thus, depending on the choice of k, *AdaRho* can reduce training time by up to 30% or increase it by at most 66%.

This overhead can be further reduced by precomputing the reference model's forward passes in parallel for the next epoch, effectively hiding this cost during training.

All augmentations used in this work are scalable. Jittering and smoothing both have time complexity $\mathcal{O}(mn)$ for an $m \times n$ multivariate time series, i.e., linear in the number of elements. StiefelGen (Cheema and Sugiyama, 2024) has time complexity $\mathcal{O}(mn^2)$, which serves as an upper bound for our augmentation overhead.

In practice, we observed greater runtime overhead from augmentations than from *AdaRho*, as shown in Table 13. This is partly due to technical constraints, we used only 4 CPU cores per GPU. Since augmentations run on the CPU, using more CPU cores could significantly reduce this cost.

In summary, Table 13 reports the runtime overheads of *AdaRho*, augmentation, and their combination (*FAF*). On a per-epoch basis, *AdaRho* increases runtime by an average of 39.4%, augmentation by 69.8%, and the combined *FAF* pipeline by 129.8%. However, when considering total training time, the average overhead is significantly lower: 20.7% for *AdaRho*, 24.8% for augmentation, and 66.2% for the full *FAF* setup. This discrepancy arises because standard training includes validation loss evaluation at each epoch, which incurs additional runtime. As a result, the relative overhead introduced by our methods is amortized, reducing their overall impact on total training time.

Memory overhead is easier to analyze. The backward pass dominates overall consumption because a forward pass uses at least three times less memory than a combined forward-backward pass. By selecting only a subset of training samples in each batch, we shrink the effective sample size during the backward pass. Concretely, we process only $(k+r)\%$ of each batch: the target model is updated with the top $k\%$ of samples ranked by reducible loss, and the reference model with the next $r\%$. Since $r = \frac{k}{2}$, the dominant memory cost comes from the $k\%$ allocated to the target model, so the maximum backward–pass memory is $k\%$ of the original batch. The forward-pass memory does rise, but because a forward pass is at least three times cheaper than a forward–backward pass, the extra cost is offset. In our experiments, *FAF* introduced no additional memory overhead: Table 11 shows its footprint matches the baseline.

E.2 Qualitative Results

In Figures 5-12, we qualitatively demonstrate how *FAF* impacts forecasting performance across the datasets used in this paper. The examples, drawn from multivariate tasks, compare TimePFN with and without *FAF*. We observe that *FAF* results in better aligned point forecasts.

E.3 Augmentation Methods

We applied StiefelGen (Cheema and Sugiyama, 2024), Gaussian smoothing (Rosenblatt, 1956), and jittering (Um et al., 2017) sequentially with probabilities of 0.5, 0.25, and 0.5. Since this paper centers on filtering rather than augmentation, we limited the augmentation space. In Figure 4, we visualize the augmented sequences alongside the original data. As shown, the augmentations preserve the physical behavior of the underlying time series, demonstrating that they generate physically plausible variations. For visualization purposes, we show only the input time series, although the augmentations are applied to both the input and forecast segments during training.

F Ablation Details

F.1 Main Ablations

In the paper, we reported an ablation study in Table 3, highlighting the clear advantage of *FAF* over several competitive baselines on the ETT datasets. To disentangle the individual contributions of *AdaRho* and the data augmentation strategy, we conducted ablation studies by isolating each component separately (showing results with only that component). For a broader comparison, we also included several established sample selection methods: RHO-Loss (Mindermann et al., 2022), which employs a learnable prioritization mechanism; HTL (Loshchilov and Hutter, 2015), which focuses on samples with high training loss; and LRL, a simple heuristic that selects samples with the lowest loss according to a reference model. Additionally, we reported results under uniform sample shuffling as a baseline. These comparisons offer insight into how *FAF*'s selective strategy interacts with data quality and learning dynamics. In the end, we compared each method basically adhering to the same experimental configurations. Although *AdaRho* is competitive with *FAF* in some benchmarks, we see that *FAF* has much superior performance overall.

F.2 Ablation on Replacing the Reference Model

We conducted the experiments using Chronos as the reference model for univariate settings and TimePFN for multivariate settings. To demonstrate that other models can also serve as reference models, we present a study in Table 14 where both iTransformer and TimePFN are used as reference models, with iTransformer as the target model. The results show that using iTransformer as the reference model yields similarly competitive performance.

F.3 Ablation on the Applicability of FAF on Longer Horizons

To demonstrate the applicability of *FAF* across different forecast horizons, we report results over varying context lengths in Table 15. Specifically, we compare the baseline iTransformer with its *FAF*-enhanced counterpart across forecast horizons $192, 336, 720$, using a fixed input context length of 96. Both the target and reference models are iTransformer, with the reference model trained on a 25% subset of the data. The rightmost column shows the percentage improvement in MSE over the baseline. These results indicate that *FAF* generalizes well across a range of forecast horizons.

Table 6: The reference model is trained on ETTh2 and applied to ETTh1. Both input and forecast lengths are set to 96.

Method	MSE	MAE
Baseline	0.402	0.417
RHO-LOSS	0.395	0.409
AdaRho	**0.383**	**0.405**

Table 7: We report the test performance of the reference models. For the univariate setting, we used Chronos-bolt-base with an input length of 96 and a forecast horizon of 36. For the multivariate setting, TimePFN was used with both input and forecast horizons set to 96.

Datasets	Univariate		Multivariate	
	MSE	MAE	MSE	MAE
ETTh1	0.040	0.148	0.394	0.411
ETTh2	0.086	0.221	0.303	0.348
ETTm1	0.015	0.089	0.354	0.382
ETTm2	0.028	0.110	0.184	0.266
Solar	0.379	0.395	0.209	0.243
Traffic	0.105	0.170	0.406	0.275
ECL	0.199	0.304	0.146	0.243
Exchange	0.037	0.148	0.088	0.209
Weather	0.0006	0.016	0.163	0.207

Table 8: Results of multivariate time-series forecasting comparing *FAF* to leading architectures. Input and forecast lengths are both 96. *FAF* refers to *FAF* applied TimePFN. We see that *FAF* applied architecture yields uniformly SOTA performance among many architectural variations, showing the benefit of online data selection.

Dataset	ECL		Weather		Traffic		Solar-Energy		Exchange		ETTh1		ETTh2		ETTm1		ETTm2	
Models	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
FAF	0.137	0.233	0.160	0.201	0.385	0.254	0.182	0.213	0.087	0.205	0.375	0.402	0.285	0.335	0.312	0.350	0.172	0.250
TimePFN	0.138	0.234	0.166	0.208	0.392	0.260	0.203	0.219	0.100	0.223	0.402	0.417	0.293	0.343	0.392	0.402	0.180	0.262
iTransformer	0.147	0.239	0.175	0.215	0.393	0.268	0.201	0.233	0.086	0.206	0.387	0.405	0.300	0.349	0.342	0.376	0.185	0.272
PatchTST	0.185	0.267	0.177	0.218	0.517	0.334	0.222	0.267	0.080	0.196	0.392	0.404	0.293	0.343	0.318	0.357	0.177	0.260
DLinear	0.195	0.278	0.341	0.412	0.690	0.432	0.286	0.375	0.101	0.237	0.400	0.412	0.357	0.406	0.344	0.371	0.195	0.293
FEDformer	0.196	0.310	0.227	0.313	0.573	0.357	0.242	0.342	0.148	0.289	0.380	0.417	0.340	0.386	0.363	0.408	0.191	0.286
Informer	0.327	0.413	0.455	0.481	0.735	0.409	0.190	0.216	0.921	0.774	0.930	0.763	2.928	1.349	0.623	0.559	0.396	0.474
Autoformer	0.214	0.327	0.273	0.344	0.605	0.376	0.455	0.480	0.147	0.279	0.440	0.446	0.364	0.408	0.520	0.490	0.233	0.311
ModernTCN	0.214	0.298	0.162	0.211	0.735	0.459	0.315	0.335	0.103	0.228	0.386	0.393	0.295	0.341	0.323	0.366	0.171	0.255
# of Variates	321		21		862		137		8		7		7		7		7	

Table 9: Multivariate time-series forecasting performances of the baseline TimePFN, *FAF*-applied TimePFN, and the corresponding reference model. We report the test performances of all models. For all cases, TimePFN was used with both input and forecast horizons set to 96.

Datasets	Baseline		FAF		Reference	
	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.402	0.417	0.375	0.402	0.394	0.411
ETTh2	0.293	0.343	0.285	0.335	0.303	0.348
ETTm1	0.392	0.402	0.312	0.350	0.354	0.382
ETTm2	0.180	0.262	0.172	0.250	0.184	0.266
Solar	0.203	0.219	0.182	0.213	0.209	0.243
Traffic	0.392	0.260	0.385	0.254	0.406	0.275
ECL	0.138	0.234	0.137	0.233	0.146	0.243
Exchange	0.100	0.223	0.087	0.205	0.088	0.209
Weather	0.166	0.208	0.160	0.201	0.163	0.207

Table 10: Univariate time-series forecasting performances of the baseline Chronos-Bolt, *FAF*-applied Chronos-Bolt, and the corresponding reference model. We report the test performance of all models. For all cases, Chronos-Bolt was used with an input length of 96 and a forecast horizon of 36.

Datasets	Baseline		FAF		Reference	
	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.043	0.156	0.036	0.141	0.040	0.148
ETTh2	0.082	0.215	0.080	0.211	0.086	0.221
ETTm1	0.015	0.089	0.015	0.088	0.015	0.089
ETTm2	0.028	0.111	0.027	0.108	0.028	0.110
Solar	0.230	0.272	0.213	0.253	0.379	0.395
Traffic	0.107	0.176	0.100	0.166	0.105	0.170
ECL	0.191	0.302	0.186	0.295	0.199	0.304
Exchange	0.040	0.151	0.036	0.145	0.037	0.148
Weather	0.0005	0.015	0.0005	0.015	0.0006	0.016

Table 11: Maximum memory usage of approaches in megabytes (MB). We see that *FAF* incurs no additional significant memory overhead. The architecute is iTransformer and the setting is multivariate with input and forecast lengths of 96.

Memory Usage	`ETTh1`	`ETTh2`	`ETTm1`	`ETTm2`
Baseline	163.7	154.5	154.4	154.4
FAF	169.4	151.6	151.6	151.6

Table 12: MAE results on four diverse datasets spanning different domains as NN5 Daily, Rideshare, FRED-MD and COVID-Deaths (Godahewa et al., 2021) on the testing portions. Lower is better. We do not report MSE due to the spiky nature of some time series. The lookback window is 336 for NN5, Rideshare, and FRED-MD, and 128 for COVID-Deaths due to its shorter history.

Model	NN5 Daily	Rideshare	FRED-MD	COVID-Deaths
Chronos (Baseline)	3.52	1.18	1,452.97	209.27
Chronos (+FAF)	**3.39**	**1.12**	**1,408.44**	**177.31**
Moirai (Baseline)	3.58	1.34	1,982.48	158.29
Moirai (+FAF)	**3.46**	**1.23**	**1,934.72**	**155.82**

Table 13: Training runtime across datasets. The upper half shows total training time, while the lower half shows per-epoch durations (in seconds). Overheads are computed relative to the Baseline.

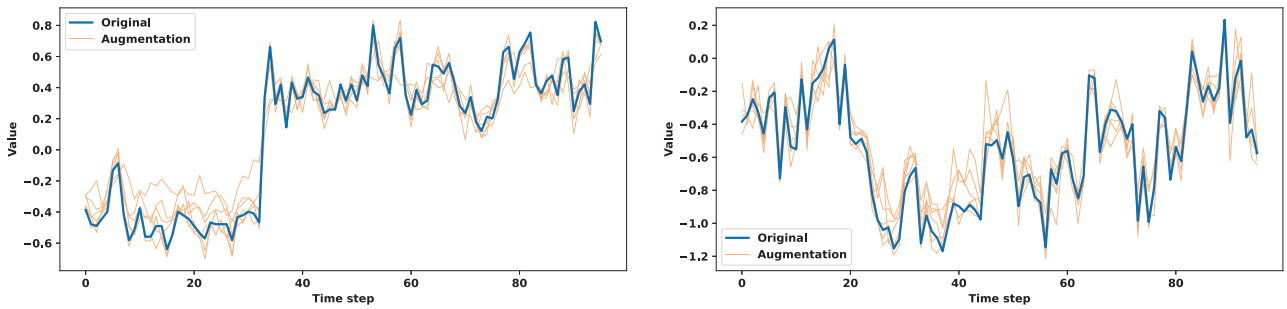
	Total training time (s) ↓			
Dataset	Baseline	AdaRho	Aug.	FAF
ETTh1	103.30	119.20	139.77	173.39
ETTh2	104.21	134.49	134.63	192.34
ETTm1	501.40	469.82	433.83	581.39
ETTm2	327.01	472.68	484.28	642.65
Avg. overhead	–	+20.7%	+24.8%	+66.2%
	Per-epoch training time (s) ↓			
Dataset	Baseline	AdaRho	Aug.	FAF
ETTh1	7.07	9.61	11.98	15.34
ETTh2	7.09	9.48	11.30	15.28
ETTm1	24.08	34.81	42.31	58.80
ETTm2	24.17	34.64	42.23	58.67
Avg. overhead	–	+39.4%	+69.8%	+129.9%

Table 14: We report the test performance of the target model (iTransformer) using two different reference models. *FAF-r-TimePFN* refers to *FAF* with TimePFN as the reference model, while *FAF-r-iTransformer* refers to *FAF* with iTransformer as the reference model. Both the input and forecast lengths are set to 96.

Datasets	Baseline		FAF-r-TimePFN		FAF-r-iTransformer	
	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.387	0.405	0.381	0.402	0.375	0.395
ETTh2	0.300	0.349	0.293	0.343	0.295	0.342
ETTm1	0.342	0.376	0.325	0.365	0.334	0.364
ETTm2	0.185	0.272	0.172	0.252	0.176	0.258
Solar	0.201	0.233	0.190	0.225	0.192	0.217
Traffic	0.393	0.268	0.388	0.262	0.392	0.268
ECL	0.147	0.239	0.147	0.237	0.147	0.237
Exchange	0.086	0.206	0.085	0.204	0.086	0.205
Weather	0.175	0.215	0.169	0.207	0.169	0.208

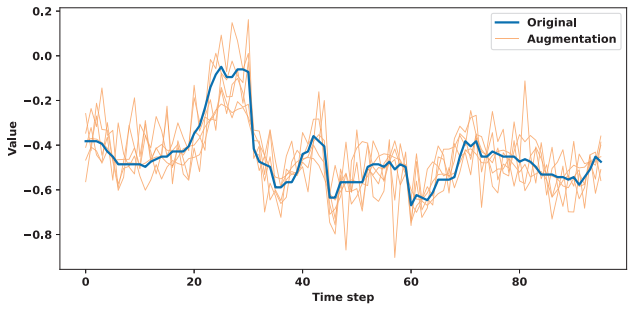
Table 15: We compare baseline iTransformer with its *FAF* applied counterpart across different context lengths. The input context length is 96 and we vary the forecast horizon in $\{192, 336, 720\}$. Both the reference and target models are iTransformer. The reference model iTransformer is trained on 25% subset of the data. The rightmost column reports percentage improvements over Baseline in MSE. We see that *FAF* generalizes accross different forecast horizons.

H:	192				336				720				*avg.impr.*
	Baseline		FAF		Baseline		FAF		Baseline		FAF		
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	
ETTh1	0.441	0.436	**0.434**	**0.430**	0.487	0.458	**0.474**	**0.455**	0.503	0.491	**0.476**	**0.474**	**3.21%**
ETTh2	0.380	0.400	**0.374**	**0.393**	0.428	0.432	**0.418**	**0.429**	0.427	0.445	**0.423**	**0.442**	**1.62%**
ETTm1	**0.377**	**0.391**	**0.377**	0.394	0.426	**0.420**	**0.421**	**0.420**	0.491	0.459	**0.489**	**0.457**	**0.53%**
ETTm2	0.250	0.309	**0.240**	**0.297**	0.311	0.348	**0.299**	**0.335**	0.412	0.407	**0.403**	**0.394**	**3.35%**
avg.impr.			**1.79%**	**1.56%**			**2.51%**	**1.27%**			**2.22%**	**1.94%**	**2.18%**

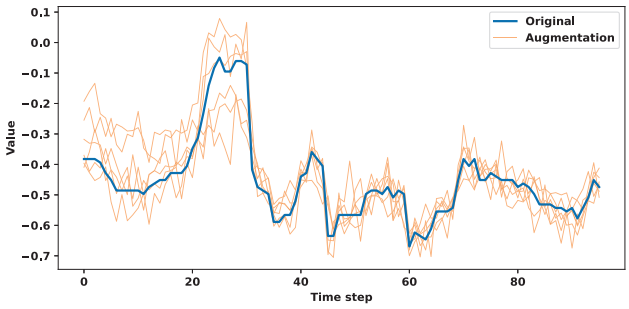


(a) ETTh1

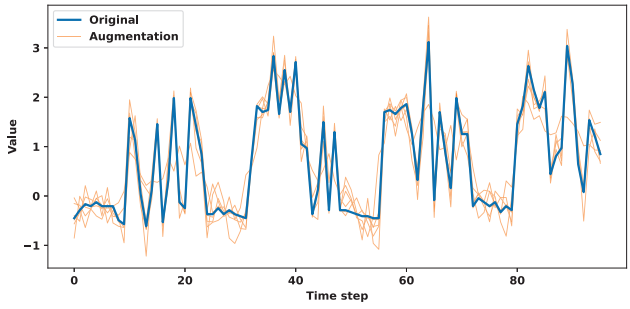
(b) ETTh2



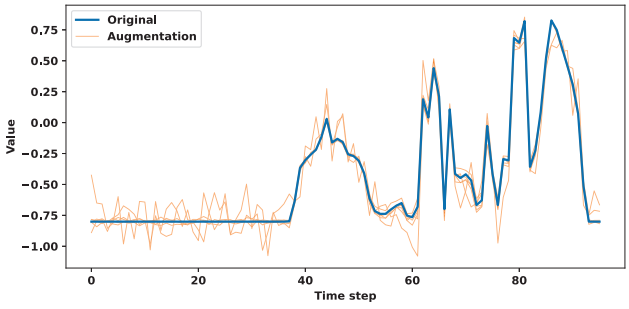
(c) ETTm1



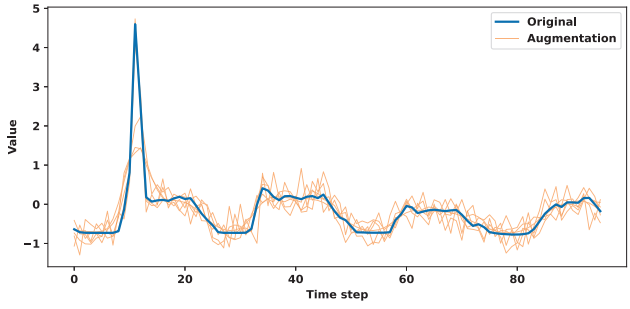
(d) ETTm2



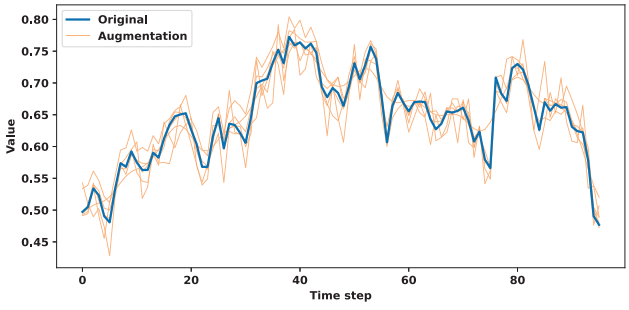
(e) ECL



(f) Solar



(g) Traffic



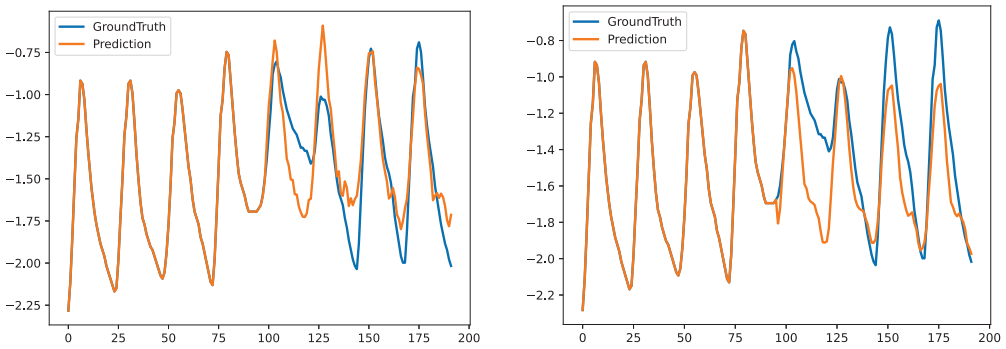
(h) Exchange

Figure 4: We visualize the data augmentations in each dataset. As one can see, the augmentations preserve the space-time structure of the time series data, yielding diverse but temporally consistent examples.

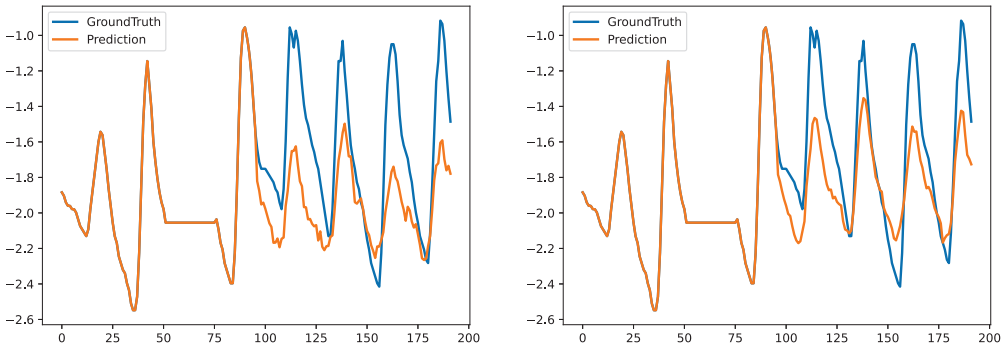
(a) Baseline (b) *FAF*

(c) Baseline (d) *FAF*

Figure 5: Comparison of Baseline vs. FAF-applied results on ETTh1 dataset. The first 96 time steps form the input series, and the remaining steps constitute the forecast. We see that *FAF*-applied training improves forecast by combining *AdaRho* with augmentation.



(a) Baseline (b) *FAF*



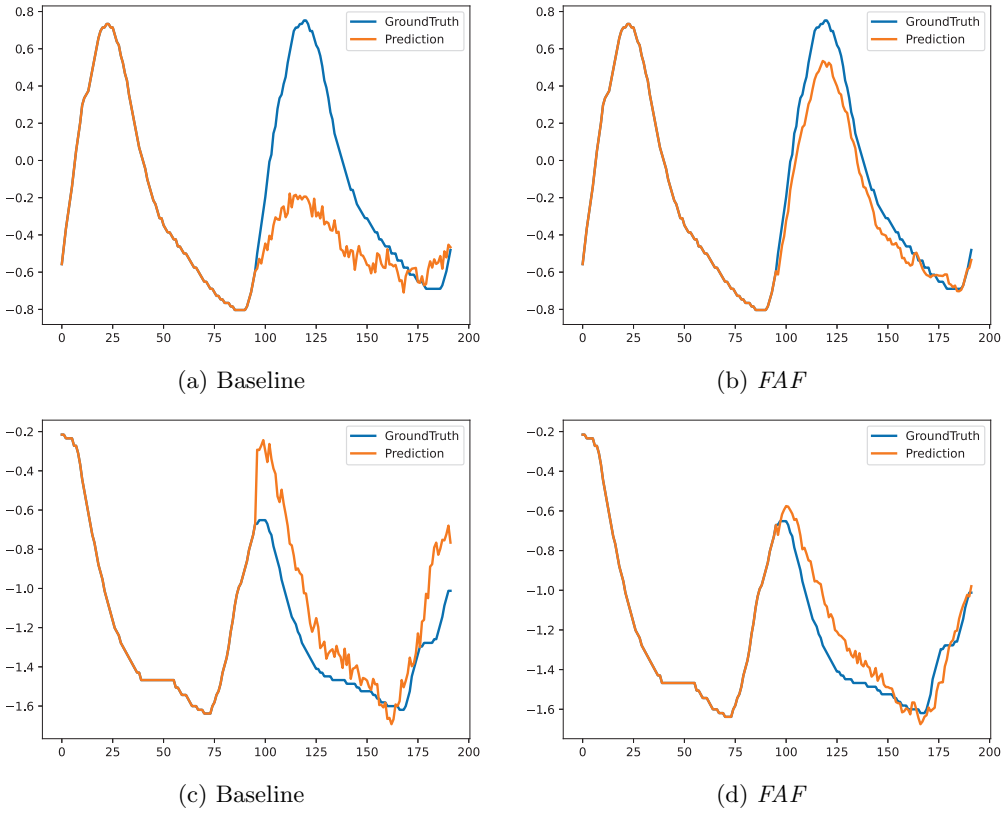
(c) Baseline (d) *FAF*

Figure 6: Comparison of Baseline vs. FAF-applied results on ETTh2 dataset. The first 96 time steps form the input series, and the remaining steps constitute the forecast. We see that *FAF*-applied training improves forecast by combining *AdaRho* with augmentation.

(a) Baseline (b) *FAF*

(c) Baseline (d) *FAF*

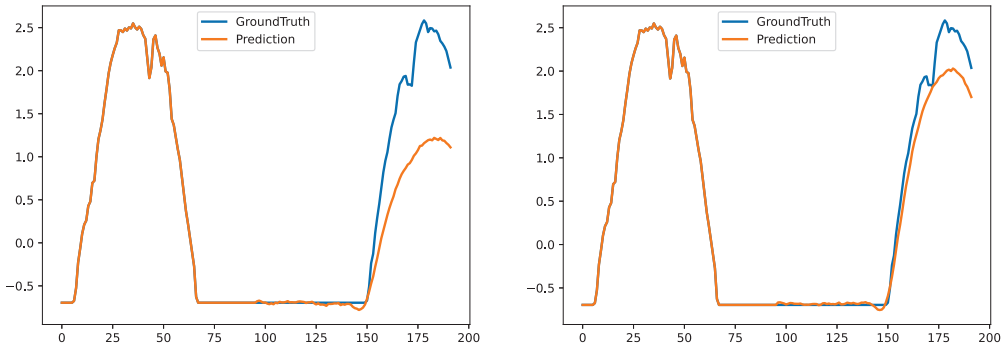
Figure 7: Comparison of Baseline vs. FAF-applied results on ETTm1 dataset. The first 96 time steps form the input series, and the remaining steps constitute the forecast. We see that *FAF*-applied training improves forecast by combining *AdaRho* with augmentation.



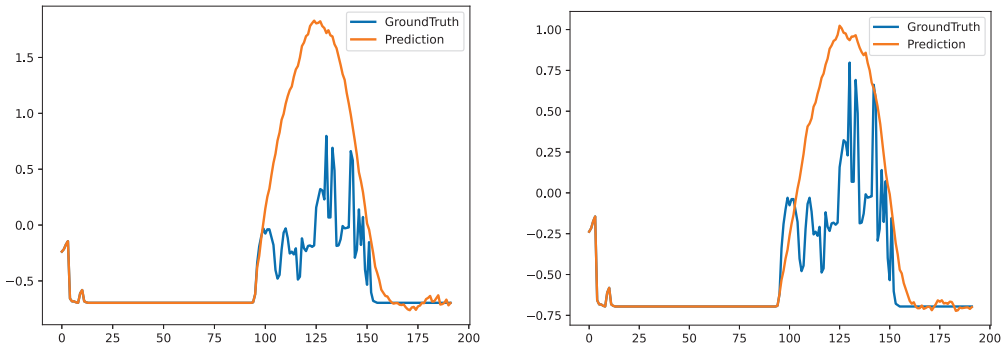
(a) Baseline (b) *FAF*

(c) Baseline (d) *FAF*

Figure 8: Comparison of Baseline vs. FAF-applied results on ETTm2 dataset. The first 96 time steps form the input series, and the remaining steps constitute the forecast. We see that *FAF*-applied training improves forecast by combining *AdaRho* with augmentation.

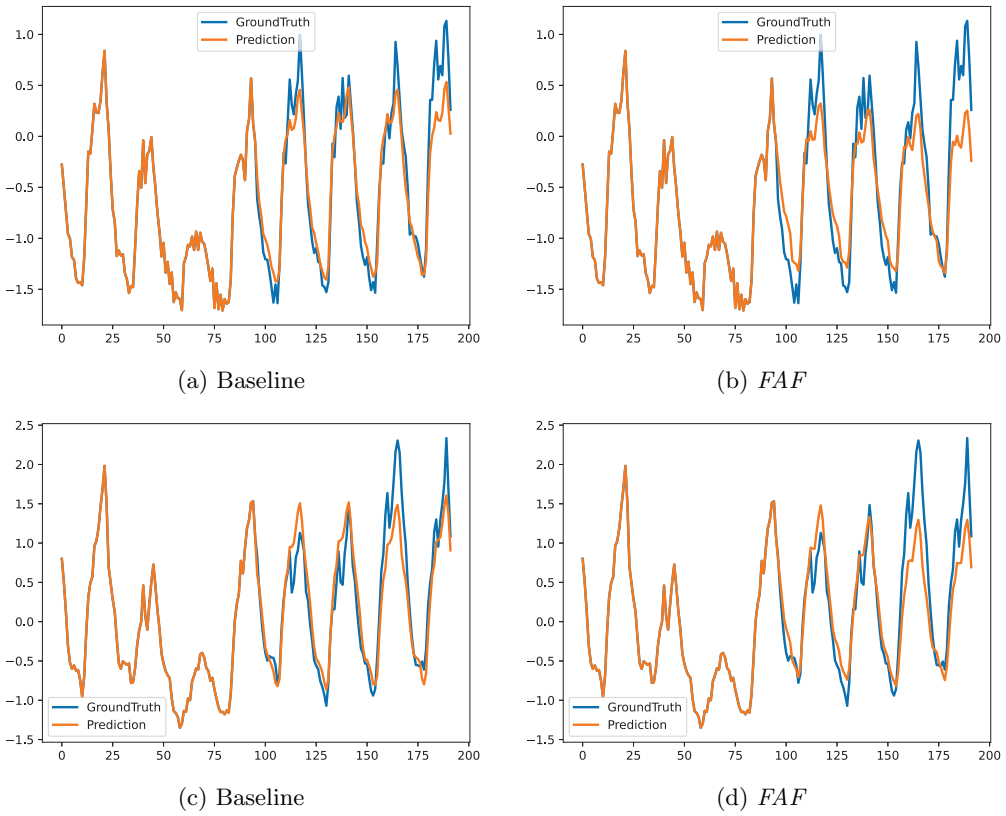


(a) Baseline (b) *FAF*



(c) Baseline (d) *FAF*

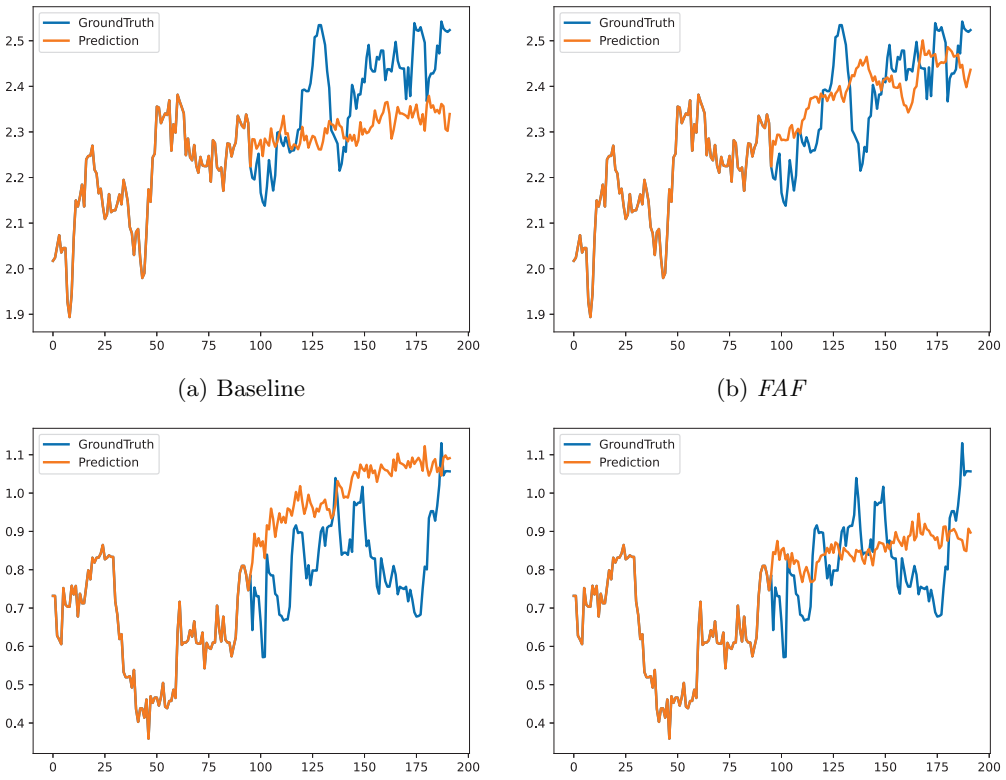
Figure 9: Comparison of Baseline vs. FAF-applied results on Solar dataset. The first 96 time steps form the input series, and the remaining steps constitute the forecast. We see that *FAF*-applied training improves forecast by combining *AdaRho* with augmentation.



(a) Baseline (b) *FAF*

(c) Baseline (d) *FAF*

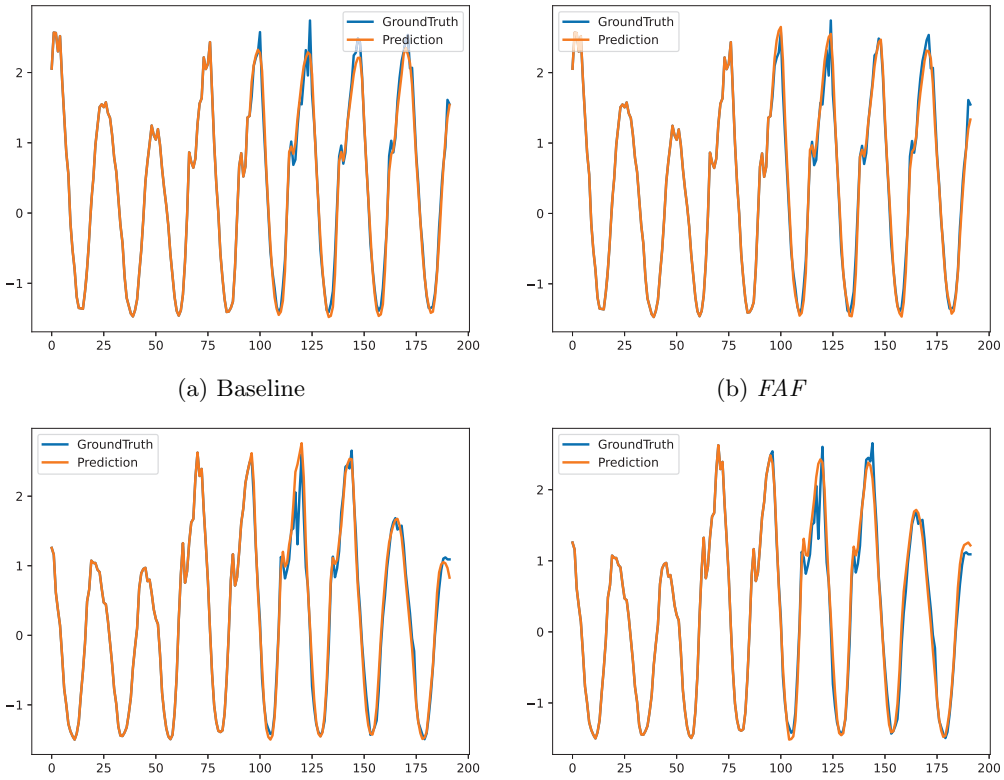
Figure 10: Comparison of Baseline vs. FAF-applied results on ECL dataset. The first 96 time steps form the input series, and the remaining steps constitute the forecast. We see that *FAF*-applied training improves forecast by combining *AdaRho* with augmentation.



(a) Baseline (b) *FAF*

(c) Baseline (d) *FAF*

Figure 11: Comparison of Baseline vs. FAF-applied results on Exchange dataset. The first 96 time steps form the input series, and the remaining steps constitute the forecast. We see that *FAF*-applied training improves forecast by combining *AdaRho* with augmentation.



(a) Baseline (b) *FAF*

(c) Baseline (d) *FAF*

Figure 12: Comparison of Baseline vs. FAF-applied results on Traffic dataset. The first 96 time steps form the input series, and the remaining steps constitute the forecast. We see that *FAF*-applied training improves forecast by combining *AdaRho* with augmentation.