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] **Samet Oymak** [1]

Abstract

While significant effort has been devoted to developing deep learning architectures for time series forecasting, the role of data in the training pipeline remains relatively overlooked. 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. We provide a theoretical analysis using random matrix theory, highlighting the impact of reference models and noise on data selection. FAF improves forecasting accuracy across diverse architectures without altering them, achieving a 5.6% median MSE and 3.2% median MAE reduction on nine datasets.

1. Introduction

Time-series forecasting remains a formidable challenge, as real-world sequences frequently exhibit low signal-to-noise ratios (Wang & 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.

[1]Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, USA [2]Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, USA [3]Department of Computer Engineering, Technical University of Munich, Munich, Germany. Correspondence to: Ege Onur Taga <egetaga@umich.edu>.

Proceedings of the 1st ICML Workshop on Foundation Models for Structured Data, Vancouver, Canada. 2025.

Recent work tackles these difficulties in time-series forecasting by pretraining time-series foundation models on billions of points from heterogeneous domains—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., 2024a). These ideas build on rapid architectural progress in deep forecasting (Zhou et al., 2022; Li et al., 2019; Liu et al., 2022; Zhang & Yan, 2023)—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.

Our paper tackles the challenge of performant time-series forecasting by introducing *Filter, Augment, Forecast (FAF)*, an online data curation strategy that performs adaptive data selection and data augmentation. For data selection, we build on RHO-LOSS from the vision and language modeling literature (Mindermann et al., 2022). RHO-LOSS distinguishes training examples that are more amenable to learning from those that are not, such as noisy or unpredictable tokens. This is done using a reference model, selecting samples based on the loss gap between the reference and target models. More recent work, Rho-1 (Lin et al., 2024), adds token-level filtering to yield substantial speed-ups and performance gains in language model training.

- Our main contribution is an adaptive variant of RHO-LOSS, called *AdaRho*, described in Algorithm 1. *AdaRho* addresses a key weakness of RHO-LOSS—its static reference model, which struggles under distribution shifts common in time series (e.g., streaming data or heterogeneous domains). We resolve this by strategically updating the reference model via a secondary data selection mechanism.
- As a second contribution, we present a statistical analysis

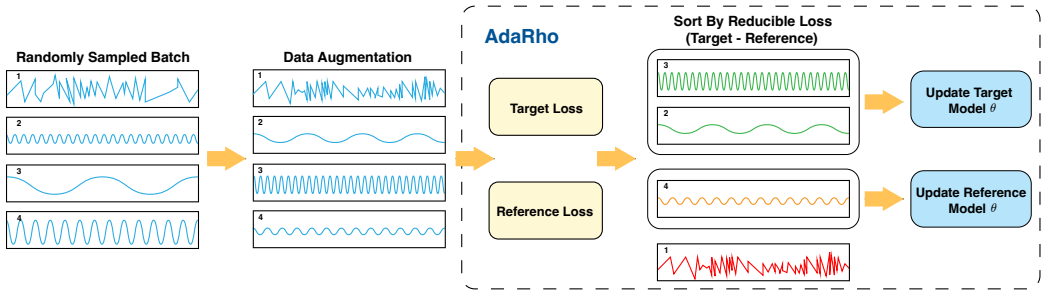


Figure 1: In *FAF*, each batch is randomly sampled, augmented, and ranked by *reducible loss*. Top samples update the target model with a high learning rate, while the next-best update the reference model more conservatively.

of reference-based data selection in regression with mixed noise. It quantifies how reference quality and noise impact selection and performance, highlighting the risks of poor references and motivating *AdaRho*.

• As a final contribution, we combine our adaptive filtering method *AdaRho* with time series data augmentation (Cheema & Sugiyama, 2024; Chung, 2020), yielding the *Filter, Augment, Forecast (FAF)* framework. Augmentation helps the model learn from high-quality examples while compensating for reduced sample size.

We show that *FAF* (Figure 1) consistently improves forecast accuracy, achieving a 5.6% median MSE reduction across nine tasks and eight state-of-the-art models (Section 3). Our analysis links its effectiveness to training loss statistics, highlighting when data selection is most beneficial. Related work is detailed in Appendix A.

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* is model-agnostic and can be applied to any time series forecasting model. As demonstrated in the experiments, it consistently 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. In the case of 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 differ widely across domains in modality, channel count, forecast horizon $\widetilde{T}$, and dynamics. What is informative in one domain may signal noise or sensor failure in another, making sample selection highly domain-specific. An effective online batch selection algorithm must therefore be domain-aware, distribution-sensitive, and adaptive to non-stationarity. It should downweight noisy or redundant samples—such as easily predictable periodic trends—and prioritize informative, rare, or time-specific patterns. For instance, a post–peak-hour drop in electricity use carries more signal than repetitive daily cycles. We now present a framework that meets these criteria.

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 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. However, we know that the reference model is a reasonable forecaster. As a result, out-of-distribution samples are likely to yield high loss when evaluated by the reference model. Such out-of-distribution samples can arise from noise introduced during the measurement process or may result from rare external events (e.g. black swan events (Taleb, 2008)) 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 on them.

Adaptivity. In standard RHO, the reference model's parameters remain fixed during target model training. While this works for large, i.i.d. datasets—like web-scale corpora—it breaks down in domains such as time series, where data are smaller and non-i.i.d. due to temporal dependencies. In such cases, a static reference model may poorly approximate the evolving distribution. To address this, *AdaRho* updates the reference model during training using samples with moderately high reducible loss (ranked between the top $k\%$ and $(k + r)\%$), applying a much lower learning rate. This ensures the updated model captures informative and decorrelated samples while avoiding overfitting to noise. We adopt *AdaRho* for all experiments and provide theoretical and empirical evidence of its benefits over standard RHO.

We study the noise robustness of *AdaRho* in Appendix B. Appendix D theoretically analyzes the filtering framework in linear regression, highlighting how noise, thresholds, and sample size interact. Now, let's discuss the experiments.

3. Experiments

We evaluate *FAF* on a variety of multivariate 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 trained with standard SGD) and its performance with *FAF* training.

Datasets and Baselines. We evaluate *FAF* on nine MTS datasets: ETTh1/2, ETTm1/2, Weather, Solar Energy (Lai et al., 2018), ECL, Exchange, and Traffic (Wu et al., 2021). In univariate settings, only the target column is used. While Exchange is less ideal for absolute accuracy (Zeng et al., 2023; Rossi, 2013), we include it to assess *FAF*'s relative gains (see Appendix F). For multivariate forecasting, we use PatchTST (Nie et al., 2023), TimePFN (Taga et al., 2025), iTransformer (Liu et al., 2023), Autoformer (Wu et al., 2021), Informer (Zhou et al., 2021), and ModernTCN (donghao & wang xue, 2024). For univariate, we fine-tune Chronos-Bolt-Base (Ansari et al., 2024) and Moirai-Base (Woo et al., 2024b), with and without *FAF*.

Experimental Setup. For multivariate tasks, we use a 96-timestep lookback window and a 96-timestep forecast horizon. For univariate tasks, with a focus on shorter horizon, we still feed in 96 time steps but forecast only the next 36. Each dataset receives its own reference model trained on 25% of the training data. In the multivariate *FAF* pipelines we reuse a single reference model, TimePFN, across all target models to streamline comparison. Appendix H shows that swapping in iTransformer yields comparable results. In the univariate setting, we instead use Chronos-bolt (Ansari et al., 2024) as the reference model, fine-tuned on the 25% subset (see Appendix F for reference model 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 $k \in \{0.25, 0.5, 0.75\}$ 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 $k = 0.75$, in which case we cap r at 0.20. 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 provided in Appendix F.

Main Results. The results in Table 1 show that *FAF* markedly improves multivariate forecasting across a wide 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 a statistical effect: when MSE is averaged over many channels, reducible-loss values become less dispersed, weakening *AdaRho* 's filtering effect. Dispersion metrics (coefficient of variation and quartile coefficient of dispersion) computed during TimePFN training, shown in Table 11, confirm that the reducible losses

Table 1: MTS forecasting results with respect to baseline, and its *FAF*-applied version. Input and forecast lengths are 96. **Bold** values indicate improvements with *FAF*. 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%**
Exchange	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%**
Exchange	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 (MSE) with baseline and its *FAF*-applied version. Input and forecast lengths are 96 and 36. The rightmost column reports average model-based MSE improvements; the bottom row shows per-dataset average improvements across models. MAE scores are provided in Appendix G.

MSE		ETTh1	ETTh2	ETTm1	ETTm2	Solar	Traffic	ECL	Exchange	Weather	avg. impr.
Chronos	Baseline	0.043	0.082	**0.015**	0.028	0.286	0.107	0.191	0.040	0.0005	**+6.4%**
	FAF	**0.036**	**0.080**	**0.015**	**0.027**	**0.241**	**0.100**	**0.186**	**0.036**	0.0005	
Moirai	Baseline	0.044	0.113	0.015	0.032	0.238	0.170	0.293	0.039	0.0006	**+10.8%**
	FAF	**0.036**	**0.093**	**0.014**	**0.030**	**0.232**	**0.142**	**0.285**	**0.035**	**0.0005**	
avg. impr.		**+17.2%**	**+10.0%**	**+3.3%**	**+4.9%**	**+9.1%**	**+11.5%**	**+2.7%**	**+10.1%**	**+8.3%**	**+8.6%**

in ECL and Traffic are far less spread out than in the ETT datasets. In fact, the higher univariate improvements (Table 2) support the same phenomena. Channel-wise application of *AdaRho* through masking of the loss contributions from the highest reducible loss yielding channels could mitigate this limitation and is left for future work. Model-wise benefits also vary. iTransformer, ModernTCN, TimePFN, Informer, and Autoformer see substantial gains, whereas PatchTST has modest improvements. PatchTST forecasts each channel independently, so noise averages out during backpropagation, reducing the marginal value of filtering. By contrast, Informer—whose ProbSparse attention increasing the architectural complexity—accumulates the largest rewards, with MSE and MAE falling by 20% and 11.7%, respectively. Overall, *FAF* increases the architectural performance of the models substantially as seen in Table 7. We demonstrate the *FAF* applied performance of TimePFN against state of the art architectures, including additionally FedFormer (Zhou et al., 2022) and DLinear (Zeng et al., 2023). The results demonstrate that the FAF applied architecture yields uniformly state-of-the-art results. We share the detailed results in Appendix G.

In the univariate benchmarks, *FAF* delivers significant gains in forecast performance. For ETTh1, the MSE falls by 17%, and similarly large improvements appear across several other datasets. On average, Chronos achieves a 6.4% MSE reduction, while Moirai improves by 10.8%, yielding an overall mean reduction of 8.6% in univariate forecasting. These results underscore *FAF*'s substantial benefit on both time-series foundation models. Ultimately, across eight architectures, *FAF* achieves a 5.6% median reduction in MSE and a 3.2% median reduction in MAE, averaged across 9 datasets.

We provide an ablation study in Appendix H.

Acknowledgments. This work is supported by NSF grants CCF2403075, CCF-2212426, and CCF-2046816; the Office of Naval Research under grant N000142412289; a gift from Google Research; and computational resources from Google Research and an Amazon Research Award on Foundation Model Development.

References

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

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. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=gerNCVqqtR.

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

Bandara, K., Hewamalage, H., Liu, Y.-H., Kang, Y., and Bergmeir, C. Improving the accuracy of global forecasting models using time series data augmentation. *Pattern Recognition*, 120:108148, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2021.108148. URL https://www.sciencedirect.com/science/article/pii/S0031320321003356.

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

Che, G. 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*, 2024.

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

Chen, S., Long, G., Jiang, J., and Zhang, C. Federated foundation models on heterogeneous time series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(15):15839–15847, Apr. 2025. doi: 10.1609/aaai.v39i15.33739. URL https://ojs.aaai.org/index.php/AAAI/article/view/33739.

Cheng, H., Wen, Q., Liu, Y., and Sun, L. RobustTSF: Towards theory and design of robust time series forecasting with anomalies. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ltZ9ianMth.

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

Cipollini, F. and Gallo, G. M. Multiplicative error models: 20 years on. *Econometrics and Statistics*, 2022.

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

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

donghao, L. and wang xue. ModernTCN: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vpJMJerXHU.

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

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

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. Datacomp: In search of the next generation of multimodal datasets. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=dVaWCDMBof.

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. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

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

Hastie, T., Montanari, A., Rosset, S., and Tibshirani, R. J. Surprises in high-dimensional ridgeless least squares interpolation, 2020. URL https://arxiv.org/abs/1903.08560.

Jain, A., Montanari, A., and Sasoglu, E. 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. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 110246–110289. Curran Associates, Inc., 2024.

Katharopoulos, A. and Fleuret, F. Not all samples are created equal: Deep learning with importance sampling. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2525–2534. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/katharopoulos18a.html.

Khuntia, S. R., Rueda, J. L., and Van der Meijden, M. A. Long-term electricity load forecasting considering volatility using multiplicative error model. *Energies*, 11(12), 2018. ISSN 1996-1073. doi: 10.3390/en11123308. URL https://www.mdpi.com/1996-1073/11/12/3308.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. 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, pp. 95–104, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356572. doi: 10.1145/3209978.3210006. URL https://doi.org/10.1145/3209978.3210006.

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. 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. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 14200–14282. Curran Associates, Inc., 2024.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/6775a0635c302542da2c32aa19d86be0-Paper.pdf.

Lin, Z., Gou, Z., Gong, Y., Liu, X., yelong shen, Xu, R., Lin, C., Yang, Y., Jiao, J., Duan, N., and Chen, W. Not all tokens are what you need for pretraining. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=0NMzBwqaAJ.

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

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

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

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

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. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pp. 15630–15649. PMLR, 2022.

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

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

Rosenblat, M. Remarks on some nonparametric estimates of a density function. *Ann. Math. Stat*, 27:832–837, 1956.

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

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

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. 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. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 25278–25294. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/a1859debfb3b59d094f3504d5ebb6c25-Paper-Datasets_and_Benchmarks.pdf.

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

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

Taleb, N. N. *The impact of the highly improbable*. Penguin Books Limited, 2008.

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

Tulino, A. M., Verdú, S., et al. Random matrix theory and wireless communications. *Foundations and Trends® in Communications and Information Theory*, 1(1):1–182, 2004.

Um, T. T., Pfister, F. M., Pichler, D., Endo, S., Lang, M., Hirche, S., Fietzek, U., and Kulić, D. 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*, pp. 216–220, 2017.

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

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

Wenzek, G., Lachaux, M.-A., Conneau, A., Chaudhary, V., Guzmán, F., Joulin, A., and Grave, E. 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. (eds.), *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 4003–4012, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.494/.

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

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., and Sahoo, D. Unified training of universal time series forecasting transformers, 2024b. URL https://arxiv.org/abs/2402.02592.

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

Yoon, T., Park, Y., Ryu, E. K., and Wang, Y. Robust probabilistic time series forecasting. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I. (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 1336–1358. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/yoon22a.html.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. 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, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i9.26317. URL https://doi.org/10.1609/aaai.v37i9.26317.

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

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

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. 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, pp. 11106–11115. AAAI Press, 2021.

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

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

Input: Target model f_θ to be optimized, reference model $f_{\theta'}$ with its pretrained parameters θ'^0, target batch size n_b, reference update sample size $n_r > 0$, batch size $n_B > n_b + n_r$, learning rates $\eta \gg \eta'$
Initialize θ^0, and $t = 0$
for $t = 0, 1, \ldots$ **do**
  Randomly sample a batch B_t of size n_B
  $\forall i \in B_t$, 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]`
  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'
  $\theta^{t+1} \leftarrow \theta^t - \eta g_t$
  $\theta'^{t+1} \leftarrow \theta'^t - \eta' g'_t$

Outline

The appendix is organized as follows:

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

A. 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 & Hutter, 2015; Mindermann et al., 2022; Schaul et al., 2015; Katharopoulos & Fleuret, 2018), and adapt these ideas to time series forecasting, in particular the RHO-LOSS (Mindermann et al., 2022). In time series forecasting, data-centric methods encompass robustness, anomaly filtering, and data augmentation (Cheng et al., 2024; Du et al., 2021; Fan et al., 2023; Yoon et al., 2022). For instance, RobustTSF (Cheng et al., 2024) links noisy-label learning with forecasting by formalizing three types of anomalies and filtering them using variance-based criteria, while (Yoon et al., 2022) improves robustness to input perturbations through smoothed training of probabilistic forecasters. Other work focuses on offline sensor data cleaning during the data collection phase, targeting anomalies directly (Bachechi et al., 2020; Zhang et al., 2017; Ding et al., 2019). In contrast, our approach does not explicitly target anomalous datasets or sensor-level corrections, but instead aims to identify and leverage sample quality within standard time series datasets during training. Moreover, 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

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, ETTh2, ETTm1, ETTm2). We inject synthetic noise by randomly simulating *multiplicative* sensor noise with varying probabilities. Multiplicative 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 & Gallo, 2022; Khuntia et al., 2018; Tian et al., 2013). 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 $\sigma^2 = 4$:

$$\bar{y}^i_{t',j} = y_{t',j} \cdot z^i_{t',j} \quad \text{where} \quad z^i_{t',j} \sim \mathcal{N}(1, \sigma^2) \quad \text{and} \quad 1 \le j \le N \tag{2}$$

B.1. 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*-augmented counterpart. All hyperparameters follow the official iTransformer configuration ($(t, h) = (96, 96)$). For each dataset, we trained a reference model, also an iTransformer with the same hyperparameters, on a randomly selected 25% subset of the data. We evaluate the *clean* test split and report mean absolute error (MAE).

B.2. Results

Two trends emerge from Table 3. 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 the baseline. At $\rho = 0.6$ the gains widen to **9.0**% MSE and **5.3**% MAE, underscoring the benefit of our targeted data selection under heavy corruption. Even in the noise-free setting ($\rho = 0$), *AdaRho* still improves MSE by **4.0**% and MAE by **3.2**%. Because real-world time-series measurements are never perfectly clean, these results suggest that *AdaRho* can deliver significant accuracy gains across both low- and high-noise regimes.

As shown in Table 3, *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 3, even with a fixed $k = 25$, *AdaRho* achieves substantial gains, even under heavy noise where some selected samples are still corrupted.

Table 3: 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%

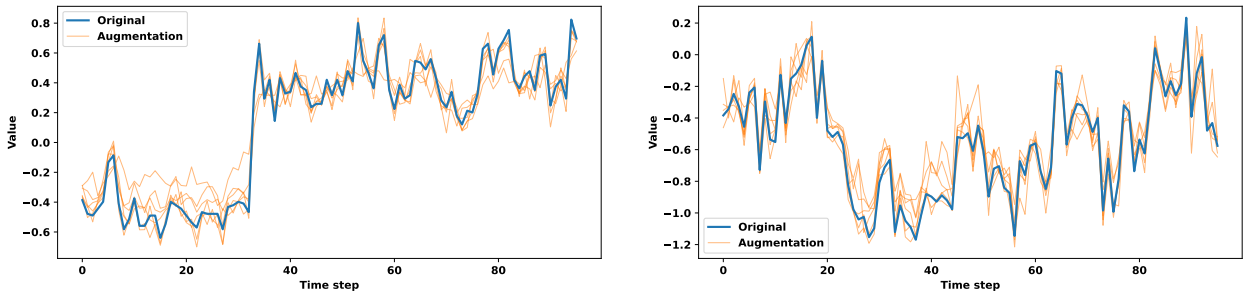
C. Augmentation Methods

At each batch, *AdaRho* first filters out potentially noisy observations, then trains the target model on the smaller set of high-quality samples. A reduced training set, however, can hurt forecasting accuracy. In Section D, we analyze how noise magnitude and sample–target correlations interact, showing that a simple thresholding rule indeed retains a higher proportion of clean data. Yet, as Figure 3 illustrates, fewer samples can offset this benefit: the risk need not fall below the no-threshold baseline because sample size remains a dominant term in the risk bound. To recover the lost diversity, we augment the data, avoiding any transformation that violate the space-time nature of the series. Building on existing time-series data-augmentation literature, specifically StiefelGen (Cheema & Sugiyama, 2024), Gaussian smoothing (Rosenblat, 1956) and jittering (Um et al., 2017), we apply the transformations in Table 4 sequentially, sampling each according to the stated probability at every training batch. StiefelGen augments multivariate sequences by projecting them onto the Stiefel manifold, thus preserving underlying temporal physics and performing well in low-data regimes. Jittering injects small Gaussian perturbations into each sample to increase local variability.

Table 4: Augmentation Space

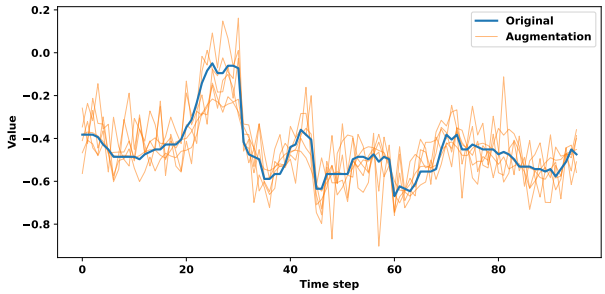
Augmentation	Probability
StiefelGen (Cheema & Sugiyama, 2024)	0.50
Smoothing (Rosenblat, 1956)	0.25
Jittering (Um et al., 2017)	0.50

In our pipeline, the reference model is trained solely on the subset of the non-augmented data. We then create augmented batches and use *AdaRho* to filter out noisy samples as well as samples possibly exhibiting physically implausible behavior due to augmentation. Below, 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.

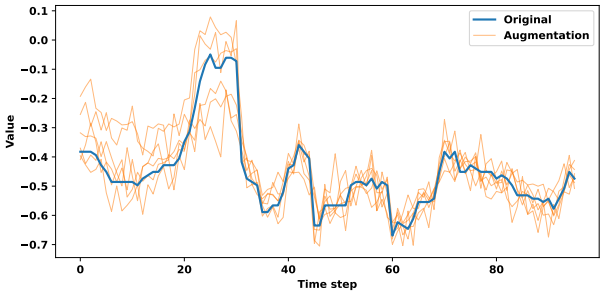


(a) ETTh1

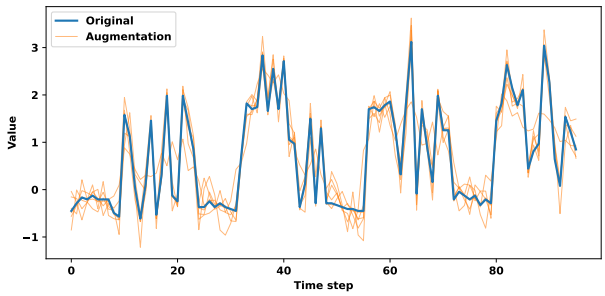
(b) ETTh2



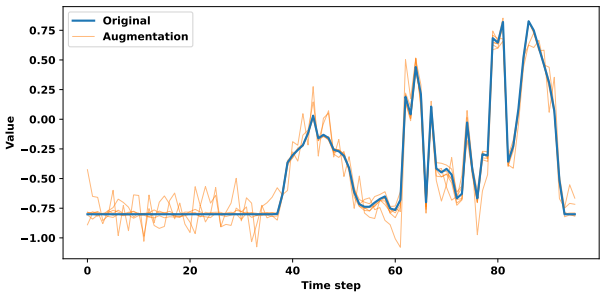
(c) ETTm1



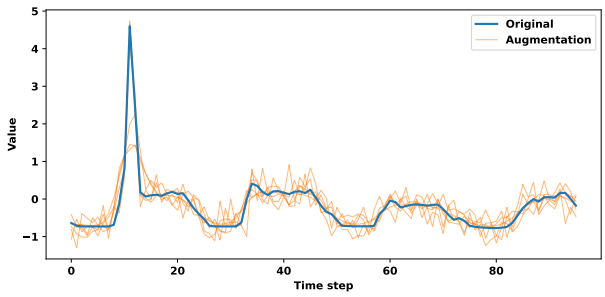
(d) ETTm2



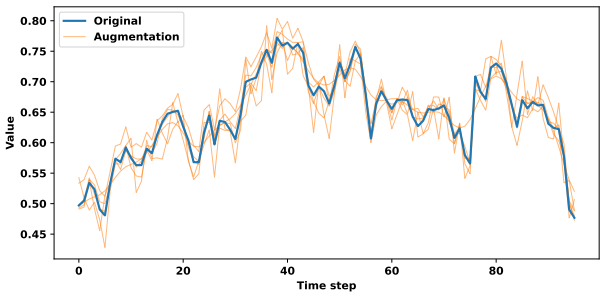
(e) ECL



(f) Solar



(g) Traffic



(h) Exchange

Figure 2: Data augmentation in the context time series for each dataset.

D. Theoretical Results

We provide a theoretical analysis of reference-loss filtering in a regression setting aligned with point forecasting, quantifying trade-offs between data quality, noise, and sample size. By modeling with linear regression and random features, we leverage random matrix theory for a sharp characterization (Tulino et al., 2004; Hastie et al., 2020; 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 introduces unique challenges due to statistical dependencies between label noise and features induced by reference-based filtering, resulting in estimation bias. Nonetheless, our analysis yields empirically meaningful insights into data selection and the roles of the reference model and noise statistics.

D.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 $i = 1, \ldots, n$, 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:

$$\xi_i = \begin{cases} \sigma_1 \, z_i, & \text{with probability } p_1, \\ \sigma_2 \, z_i, & \text{with probability } p_2, \end{cases} \qquad \text{with } p_1 + p_2 = 1 \tag{3}$$

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:

$$\hat{\boldsymbol{\beta}} := \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \left\{ \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 : \boldsymbol{\beta} \text{ has minimal } \|\boldsymbol{\beta}\|_2 \right\} \tag{4}$$

where $\boldsymbol{X} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_n^\top]\top \in \mathbb{R}^{n \times p}$ and $\boldsymbol{y} = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n$. 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:

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

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

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

for a chosen threshold $\tau > 0$. 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 $\mathcal{R}_{X_f}(\hat{\boldsymbol{\beta}}_f; \boldsymbol{\beta})$. Below we provide an analysis of of this risk in terms of key problem variables.

D.2. Derivation of Filtered Model Asymptotic Performance

To derive our results, we consider a standard large dimensional setting – so-called 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., 2020) that, for the standard linear regression problem $\hat{\boldsymbol{\beta}} = \arg\min_\beta \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|$ 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:

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

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

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

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 $j = 1,2$:

$$\sigma^2_{j,\perp}(\tau,\delta) := \sigma_j^2 \operatorname{Var}\left(z \,\middle|\, |\zeta_j| \le \tau\right) \text{ and } \sigma^2_{j,\|}(\tau,\delta) := \sigma_j^2 \frac{\operatorname{Cov}\left(z, \delta g \,\middle|\, |\zeta_j| \le \tau\right)^2}{\operatorname{Var}\left(\delta g \,\middle|\, |\zeta_j| \le \tau\right)}$$

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

$$\sigma^2_{\perp}(\tau,\delta) := \frac{p_1 \Pi_{\delta,\sigma_1}(\tau)\, \sigma^2_{1,\perp}(\tau,\delta) + p_2 \Pi_{\delta,\sigma_2}(\tau)\, \sigma^2_{2,\perp}(\tau,\delta)}{\pi(\tau)}, \; \sigma^2_{\|}(\tau,\delta) := \frac{p_1 \Pi_{\delta,\sigma_1}(\tau)\, \sigma^2_{1,\|}(\tau,\delta) + p_2 \Pi_{\delta,\sigma_2}(\tau)\, \sigma^2_{2,\|}(\tau,\delta)}{\pi(\tau)} \tag{7}$$

In the initial regression, z, g are independent, and z is treated as pure noise. However, post-filtering, z and g variables become negatively correlated due to the filtering (absolute value thresholding). This negative correlation results in a biased estimation of the latent parameter $\beta_\star$. Our decomposition of noise captures this fact to provide a refined estimate of the risk. After applying the threshold τ to the residuals $|r_i|$, the proportion of retained points that come from each noise component changes from the original mixture weights p_1, p_2. Specifically, since $\Pi_{\delta,\sigma_1}(\tau) \ge \Pi_{\delta,\sigma_2}(\tau)$ for all τ as $\sigma_2 \ge \sigma_1$, thresholding retains a larger fraction of the lower-noise samples. Consequently, the post-filtered dataset exhibits a higher proportion of data from the σ_1 component compared to the original mixture, thus effectively shifting the noise distribution toward the 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:

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

The above result is obtained by extending Theorem 1 of (Hastie et al., 2020) 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 $\sigma^2_{\perp}(\tau,\delta)$ and $\sigma^2_{\|}(\tau,\delta)$ given by (7). The aligned part $\sigma^2_{\|}(\tau,\delta)$, 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 $\sigma^2_{\|}(\tau,\delta)$ and $\frac{\sigma^2_{\|}(\tau,\delta)}{\gamma_f(\tau)}$ 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 $\sigma^2_{\|}(\tau,\delta)$ is treated as i.i.d. Gaussian noise. Therefore, as $n \to \infty$, we state the following result:

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

The above fact also utilizes from Theorem 1 of (Hastie et al., 2020), 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 also 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. While stating the above asymptotic risk formula, we assumed that $\boldsymbol{\beta}_{\text{ref}}$ is sufficiently close to $\boldsymbol{\beta}_\star$, which we further discuss in Appendix E.

In Figure 3, we compare the theoretical and empirical risk estimates for $n = 2000$, $p = 400$, $\|\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}\|_2 = 0.25$, and $(\sigma_1, \sigma_2) = (0.5, 2)$. While filtering increases the proportion of lower-noise samples, the reduced dataset size can lead to higher risk, highlighting the trade-off between removing noisy points and retaining sufficient training data. Moreover, having a reference model $\boldsymbol{\beta}_{\text{ref}}$ that more closely aligns with $\boldsymbol{\beta}_\star$ further reduces the final risk, justifying the adaptive nature of *AdaRho*.

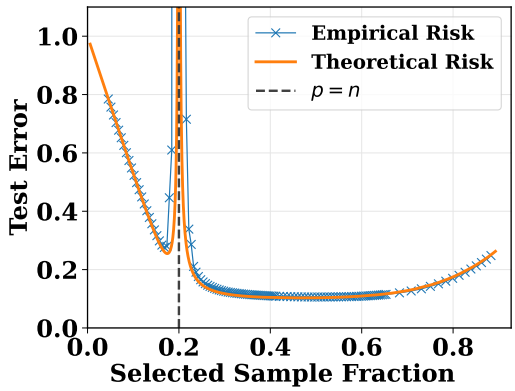


Figure 3: Theoretical *vs.* empirical risk

E. Further Theoretical Discussions

Recall from Section D 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, othe 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:

$$D_1 := \left\{(x_i,\, y_i) : i \in \{1,\dots,n\},\, y_i = x_i^\top\boldsymbol{\beta}_\star + \sigma_1 z_i,\, \left|\, x_i^\top(\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}) + \sigma_1 z_i\right| \le \tau\right\},$$
$$D_2 := \left\{(x_i,\, y_i) : i \in \{1,\dots,n\},\, y_i = x_i^\top\boldsymbol{\beta}_\star + \sigma_2 z_i,\, \left|\, x_i^\top(\boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}) + \sigma_2 z_i\right| \le \tau\right\}.$$

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 D, the noise can be decomposed into two parts for $j = 1, 2$:

$$\sigma_{j,\perp}^2(\tau,\delta) := \sigma_j^2\,\mathrm{Var}\left(z \,\middle|\, |\zeta_j| \le \tau\right) \text{ and } \sigma_{j,\|}^2(\tau,\delta) := \sigma_j^2\,\frac{\mathrm{Cov}\left(z, \delta g \,\middle|\, |\zeta_j| \le \tau\right)^2}{\mathrm{Var}\left(\delta g \,\middle|\, |\zeta_j| \le \tau\right)}$$

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

$$\sigma_\perp^2(\tau,\delta) := \frac{p_1\Pi_{\delta,\sigma_1}(\tau)\,\sigma_{1,\perp}^2(\tau,\delta) + p_2\Pi_{\delta,\sigma_2}(\tau)\,\sigma_{2,\perp}^2(\tau,\delta)}{\pi(\tau)},\ \sigma_\|^2(\tau,\delta) := \frac{p_1\Pi_{\delta,\sigma_1}(\tau)\,\sigma_{1,\|}^2(\tau,\delta) + p_2\Pi_{\delta,\sigma_2}(\tau)\,\sigma_{2,\|}^2(\tau,\delta)}{\pi(\tau)} \tag{8}$$

In the following parts, we will show how we derive the above formulas for $\sigma_\perp^2(\tau,\delta)$ and $\sigma_\|^2(\tau,\delta)$. Defining $\boldsymbol{\beta}_{\text{diff}} := \boldsymbol{\beta}_\star - \boldsymbol{\beta}_{\text{ref}}$, we write the filtered datasets D_1 and D_2 as:

$$D_1 := \left\{(x_i,\, y_i) : i \in \{1,\dots,n\},\, y = \boldsymbol{x}^\top(\boldsymbol{\beta}_\star + \alpha_1(\tau,\delta)\boldsymbol{\beta}_{\text{diff}}) + \sigma_{1,\perp}(\tau,\delta)\, z_1 : \left|\boldsymbol{x}_i^\top\boldsymbol{\beta}_{\text{diff}} + \sigma_1\, z_i\right| \le \tau\right\}$$
$$D_2 := \left\{(x_i,\, y_i) : i \in \{1,\dots,n\},\, y = \boldsymbol{x}^\top(\boldsymbol{\beta}_\star + \alpha_2(\tau,\delta)\boldsymbol{\beta}_{\text{diff}}) + \sigma_{2,\perp}(\tau,\delta)\, z_2 : \left|\boldsymbol{x}_i^\top\boldsymbol{\beta}_{\text{diff}} + \sigma_2\, z_i\right| \le \tau\right\}$$

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

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

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

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

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 $\boldsymbol{\beta}_f = \boldsymbol{\beta}_\star + \alpha(\tau,\delta)\boldsymbol{\beta}_{\mathrm{diff}}$. 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 $\alpha_j(\tau,\delta)\boldsymbol{\beta}_{\mathrm{diff}}$ 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., 2020), for the standard linear regression problem $\hat{\boldsymbol{\beta}} = \arg\min_\beta \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2$ 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:

$$\mathcal{R}_X(\hat{\boldsymbol{\beta}};\boldsymbol{\beta}_\star) \to \begin{cases} \sigma^2 \frac{\gamma}{1-\gamma}, & \gamma < 1, \\ \|\boldsymbol{\beta}_\star\|_2^2 \left(1 - \frac{1}{\gamma}\right) + \frac{\sigma^2}{\gamma - 1}, & \gamma > 1 \end{cases} \tag{9}$$

We know that $\hat{\boldsymbol{\beta}}_f - \boldsymbol{\beta}_\star = \boldsymbol{X}_f{}^\dagger \boldsymbol{X}_f \boldsymbol{\beta}_f + \boldsymbol{X}_f{}^\dagger \boldsymbol{\xi}_f - \boldsymbol{\beta}_\star = (\boldsymbol{X}_f{}^\dagger \boldsymbol{X}_f - \boldsymbol{I})\boldsymbol{\beta}_\star + \alpha(\tau,\delta)\boldsymbol{X}_f{}^\dagger \boldsymbol{X}_f \boldsymbol{\beta}_{\mathrm{diff}} + \boldsymbol{X}_f{}^\dagger \boldsymbol{\xi}_f$, 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 D:

$$\mathcal{R}_X(\hat{\boldsymbol{\beta}};\boldsymbol{\beta}) = \mathbb{E}\left[(\boldsymbol{x}_0^\top \boldsymbol{\beta}_\star - \boldsymbol{x}_0^\top \hat{\boldsymbol{\beta}})^2 \mid \boldsymbol{X} \right] = \mathbb{E}\left[(\boldsymbol{\beta}_\star - \hat{\boldsymbol{\beta}})^\top (\boldsymbol{\beta}_\star - \hat{\boldsymbol{\beta}}) \mid \boldsymbol{X} \right] \tag{10}$$

Plugging the above $\hat{\boldsymbol{\beta}}_f - \boldsymbol{\beta}_\star$ into (10) and recalling our assumption $\boldsymbol{\beta}_\star \perp \boldsymbol{\beta}_{\mathrm{diff}}$, we see that the contribution of the term $\alpha(\tau,\delta)\boldsymbol{X}_f{}^\dagger \boldsymbol{X}_f \boldsymbol{\beta}_{\mathrm{diff}}$ to the final risk is decoupled from the rest. As can be seen from the asymptotic risks for standard linear regression in (9), 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 $\alpha(\tau,\delta)\boldsymbol{X}_f{}^\dagger \boldsymbol{X}_f \boldsymbol{\beta}_{\mathrm{diff}}$ 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 $\alpha(\tau,\delta)\boldsymbol{\beta}_{\mathrm{diff}}$ to the risk as a bias term is:

$$\left(\sum_{j=1}^{2} \frac{p_j\, \Pi_{\delta,\sigma_j}(\tau)}{\pi(\tau)}\, \alpha_j(\tau,\delta) \right) \sqrt{\mathrm{Var}(\delta g \mid |\zeta_j| \le \tau)} = \frac{p_1 \Pi_{\delta,\sigma_1}(\tau)\, \sigma^2_{1,\|}(\tau,\delta) + p_2 \Pi_{\delta,\sigma_2}(\tau)\, \sigma^2_{2,\|}(\tau,\delta)}{\pi(\tau)} = \sigma^2_{\|}(\tau,\delta). \tag{11}$$

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

$$\frac{1}{\gamma_f(\tau)} \left(\sum_{j=1s}^{2} \frac{p_j\, \Pi_{\delta,\sigma_j}(\tau)}{\pi(\tau)}\, \alpha_j(\tau,\delta) \right) \sqrt{\mathrm{Var}(\delta g \mid |\zeta_j| \le \tau)} = \frac{p_1 \Pi_{\delta,\sigma_1}(\tau)\, \sigma^2_{1,\|}(\tau,\delta) + p_2 \Pi_{\delta,\sigma_2}(\tau)\, \sigma^2_{2,\|}(\tau,\delta)}{\pi(\tau)\gamma_f(\tau)} = \frac{\sigma^2_{\|}(\tau,\delta)}{\gamma_f(\tau)}. \tag{12}$$

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 (11) and (12), we state the following result as $n \to \infty$:

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

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.

F. Experimental Details

F.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.

F.2. Implementation Details

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 & wang xue, 2024). We report their results using their original hyperparameters (e.g., number of layers, embedding dimensions, etc.).

In the univariate setting, we relied on the codebases of Chronos (Ansari et al., 2024) and Moirai (Woo et al., 2024b), and

used the Chronos-bolt-base and Moirai-base models, respectively. We used AdamW (Loshchilov & 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 $k = 0.75$, we set $r = 0.25$. The value of $k \in \{0.25, 0.5, 0.75\}$ was selected based on the lowest validation MSE.

We provide the code in the supplementary material.

F.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 D, 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 H, we demonstrate that iTransformer (Liu et al., 2023) can also serve effectively as a reference model without pretraining. In below, 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 default hyperparameter settings of the codebases are applied to the reference models and lowest validation loss yielding epoch is chosen as the reference model.

Table 5: 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

F.4. 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 training. The total training time varies by dataset size: high-dimensional datasets like Traffic require 8–9 hours, whereas smaller datasets such as the ETT variants take only tens of 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. The experimental codebase is implemented entirely in PyTorch.

Note that in this work, our primary focus was not on computational speed-up. However, by using *AdaRho*, when we train the target model on $k\%$ of the training data and the reference model on an additional $r\%$ (i.e., from $k\%$ to $(k + r)\%$), we only backpropagate through $(k + r)\%$ of the data. For moderate values of k and r, for example, $k = 25$ and $r = 12.5$, this means backpropagation is performed on just 37.5% of the data. Since forward passes are significantly cheaper than backward passes, this leads to more than a $2\times$ reduction in overall computational cost. As we vary $k \in 25, 50, 75$, the benefits of reduced backpropagation diminish at higher k values. However, the computational cost does not increase since we still backpropagate on only a subset of the training data. Overall, *AdaRho* does not increase computational complexity, aside from the cost of training the reference model on a data subset—an overhead that is mitigated by (i) training on a smaller subset and (ii) reusing the same reference model across multiple target models.

Moreover, the data augmentations we applied do not significantly increase computational complexity. Throughout our experiments, we did not observe any computational slowdown due to these augmentations. Additionally, since data loading is handled by the CPU, using CPUs with multiple cores can further speed up the process. Therefore, overall, *FAF* does not introduce any noticeable computational overhead.

G. Additional Results

G.1. Quantitative Results

Univariate Results. We first present the complete univariate forecasting results, covering both Chronos and Moirai, and reporting both MSE and MAE metrics. The MSE scores are already shown in Table 2. As seen below (in Table 6), both architectures benefit significantly from *FAF* in terms of both MSE and MAE. The rightmost column reports the average MSE scores for each architecture.

Table 6: Univariate forecasting results (MSE and MAE) for Chronos-Bolt and Moirai, with and without *FAF*.

Dataset	Chronos-Bolt				Moirai				*avg. impr.*
	Baseline		FAF		Baseline		FAF		
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	
ETTh1	0.043	0.156	**0.036**	**0.141**	0.044	0.163	**0.036**	**0.146**	**+17.2%**
ETTh2	0.082	0.215	**0.080**	**0.211**	0.113	0.254	**0.093**	**0.230**	**+10.0%**
ETTm1	0.015	0.089	0.015	**0.088**	0.015	0.091	**0.014**	**0.088**	**+3.3%**
ETTm2	0.028	0.111	**0.027**	**0.108**	0.032	0.113	**0.030**	**0.110**	**+4.9%**
Solar	0.286	0.291	**0.241**	**0.274**	0.238	0.246	**0.232**	**0.245**	**+9.1%**
Traffic	0.107	0.176	**0.100**	**0.166**	0.170	0.250	**0.142**	**0.221**	**+11.5%**
ECL	0.191	0.302	**0.186**	**0.295**	0.293	0.388	**0.285**	**0.385**	**+2.7%**
Exchange	0.040	0.151	**0.036**	**0.145**	0.039	0.148	**0.035**	**0.145**	**+10.1%**
Weather	0.0005	0.015	0.0005	0.015	0.0006	0.016	**0.0005**	**0.015**	**+8.4%**
avg. impr.			**+6.4%**	**+3.7%**			**+10.8%**	**+5.2%**	**+8.6%**

Comparison to architectural variations. To demonstrate how *FAF* compares with various architectures, we compared many architectural variations and *FAF* applied TimePFN in Table 7. All in all, one sees that compared to various architectural variations, *FAF* yields state-of-the-art results.

Table 7: Results of multivariate time-series forecasting comparing *FAF* to leading architectures. Input and forecast lengths are both 96. *FAF* refers to *FAF* applied TimePFN.

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	

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 *FAF* yields highly promising results.

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

Table 8: 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

As shown in Table 8, 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.

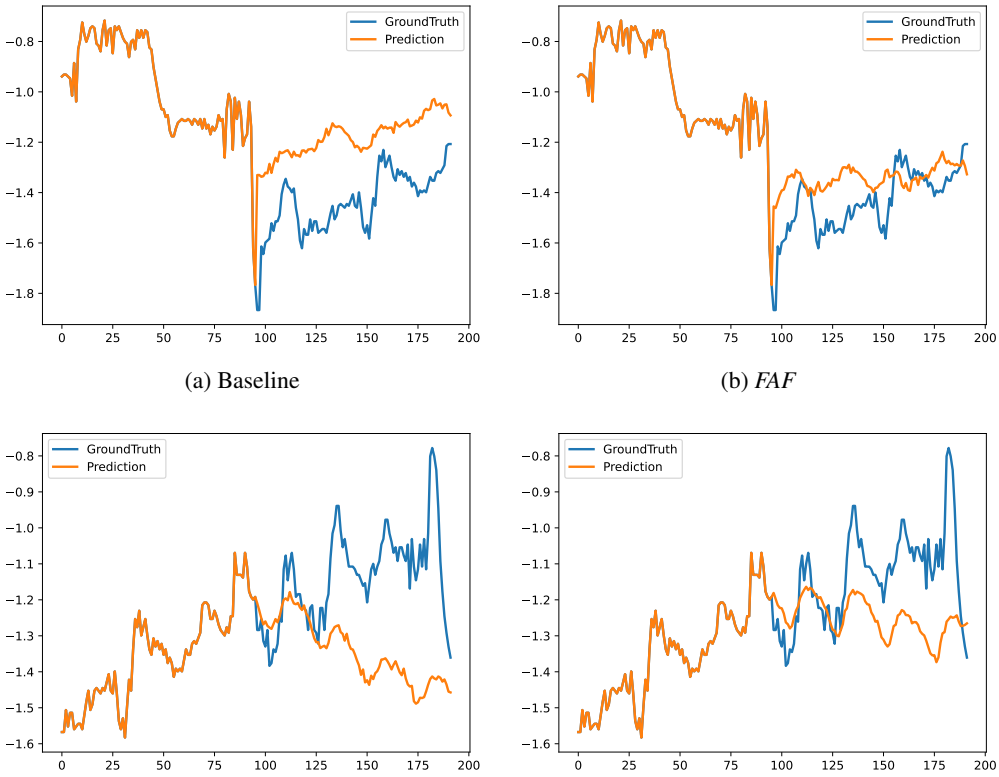
Table 9: Univariate time-series forecasting performances of the baseline Chronos-Bolt, *FAF*-applied Chronos-Bolt, and the corresponding reference model. We report the test performance of all models. For all cases, Chronos-Bolt was used with an input length of 96 and a forecast horizon of 36.

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	0.286	0.291	0.241	0.274	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

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

G.2. Qualitative Results

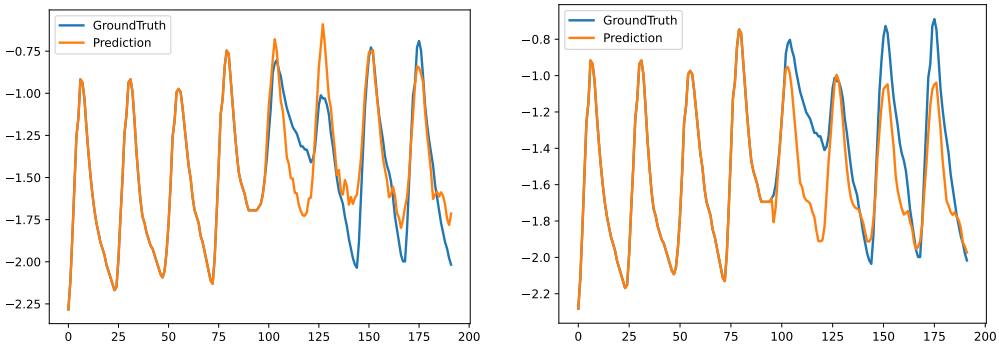
Below, we qualitatively demonstrate how FAF impacts forecasting performance. The examples, drawn from multivariate tasks, compare TimePFN with and without FAF.



(a) Baseline (b) *FAF*

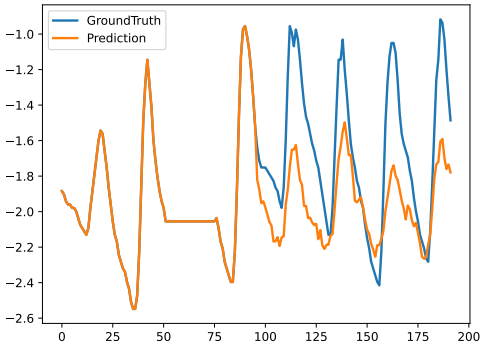
(c) Baseline (d) *FAF*

Figure 4: Comparison of Baseline vs. FAF-applied results on ETTh1 dataset.

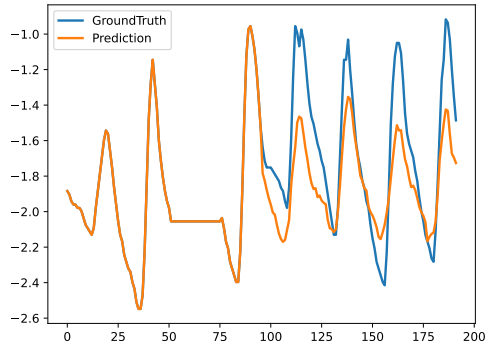


(a) Baseline

(b) *FAF*

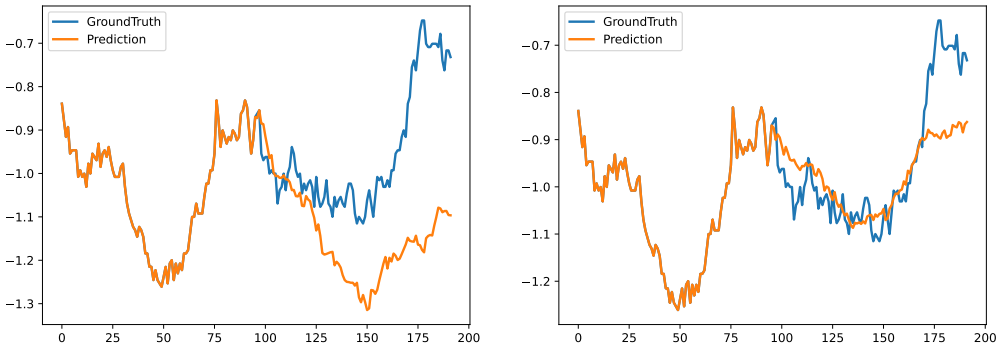


(c) Baseline

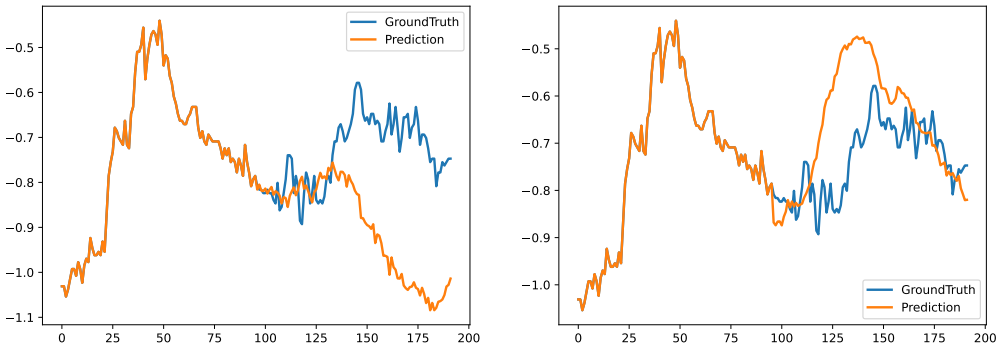


(d) *FAF*

Figure 5: Comparison of Baseline vs. FAF-applied results on ETTh2 dataset.

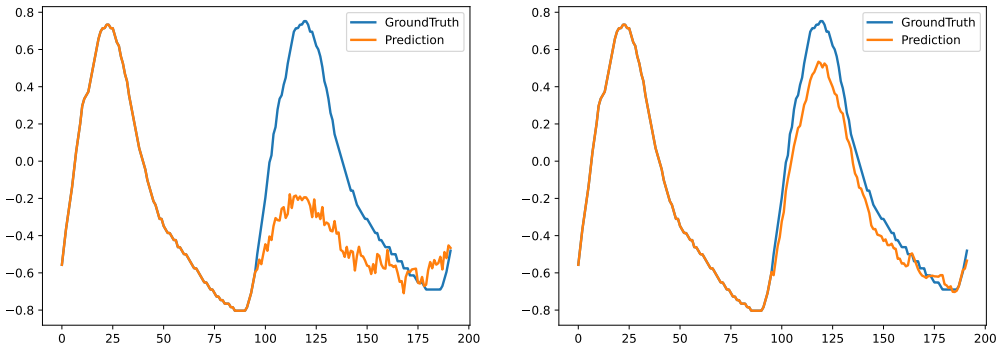


(a) Baseline (b) *FAF*

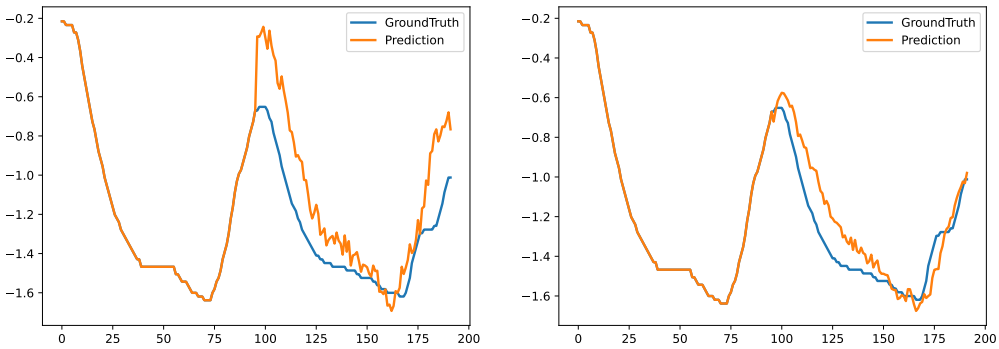


(c) Baseline (d) *FAF*

Figure 6: Comparison of Baseline vs. FAF-applied results on ETTm1 dataset.

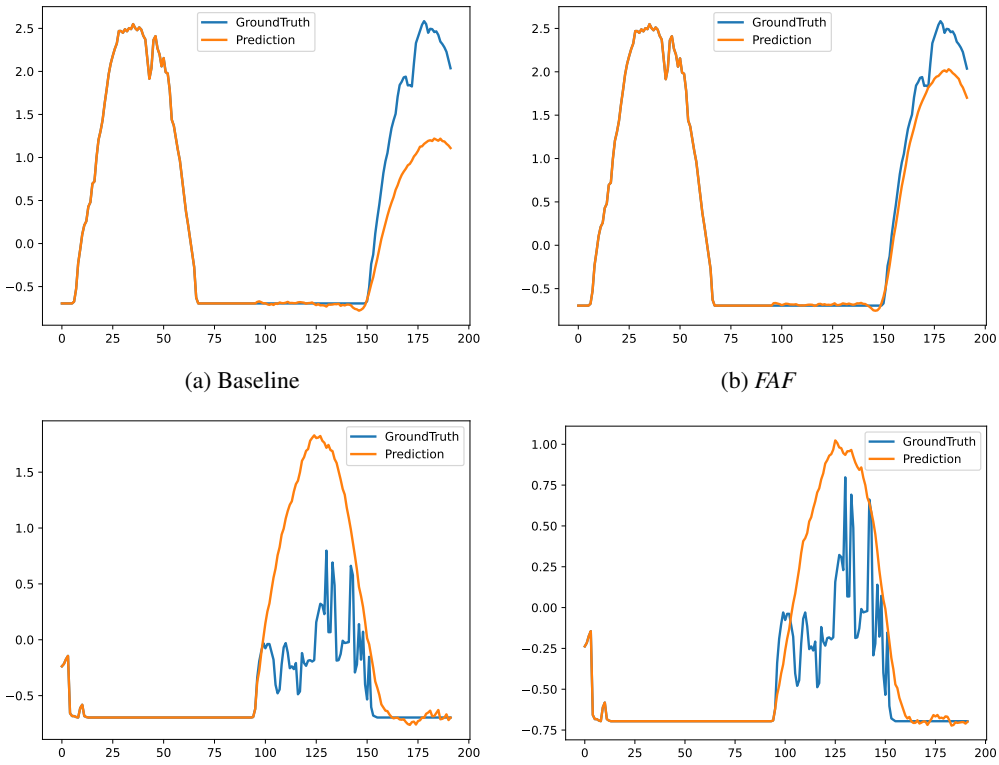


(a) Baseline (b) *FAF*



(c) Baseline (d) *FAF*

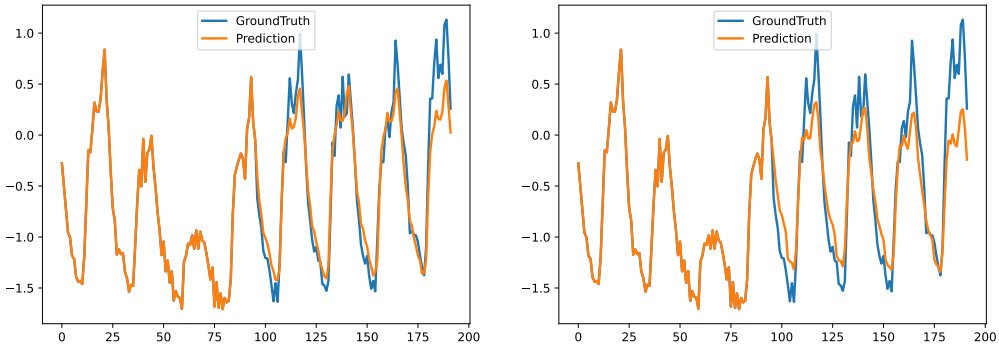
Figure 7: Comparison of Baseline vs. FAF-applied results on ETTm2 dataset.



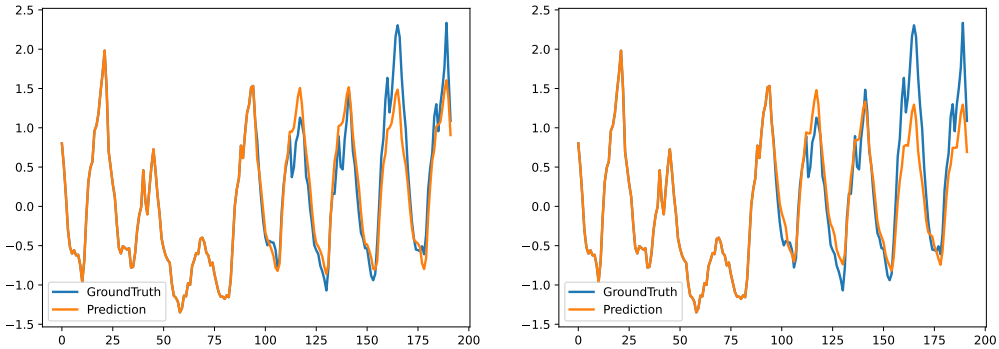
(a) Baseline (b) *FAF*

(c) Baseline (d) *FAF*

Figure 8: Comparison of Baseline vs. FAF-applied results on Solar dataset.

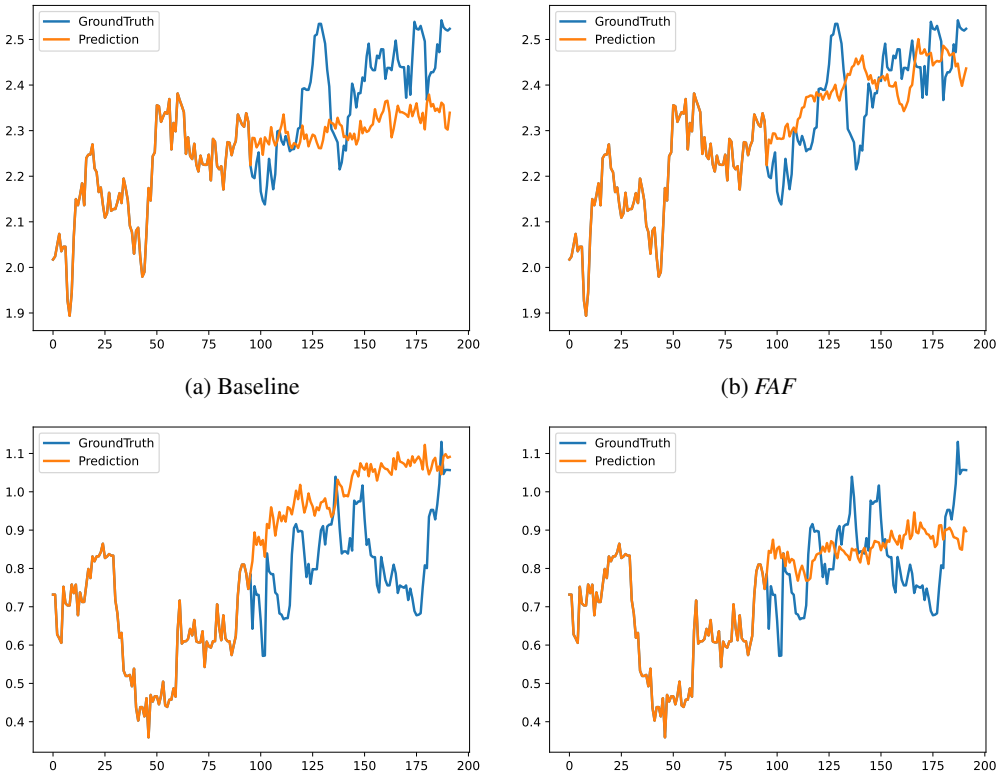


(a) Baseline (b) *FAF*



(c) Baseline (d) *FAF*

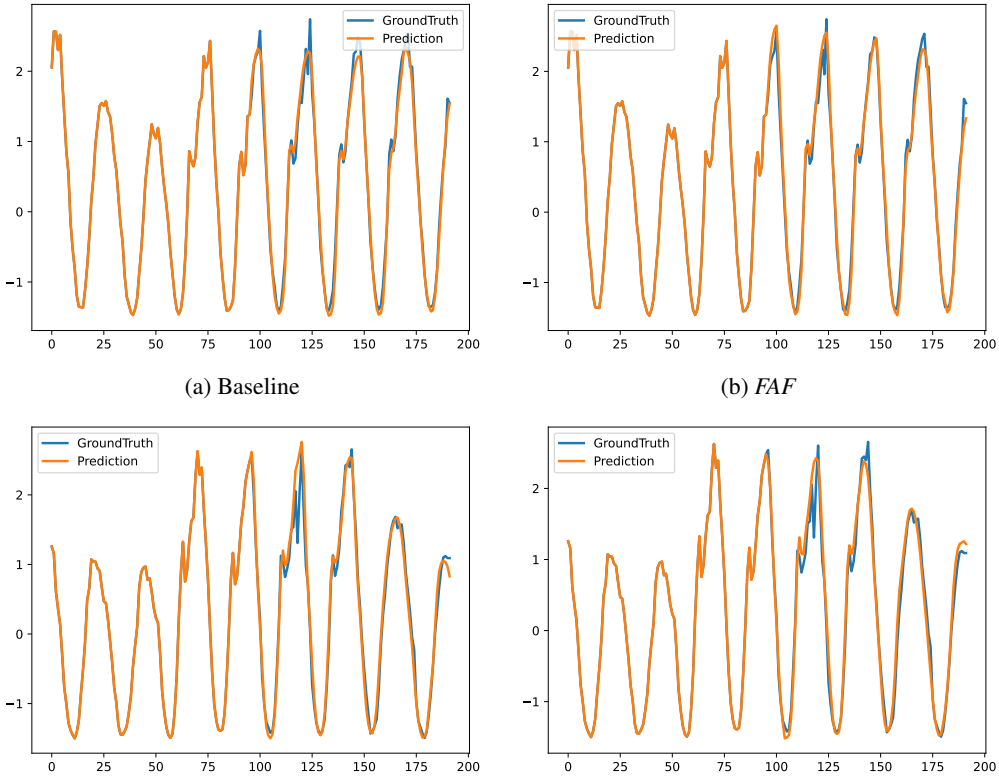
Figure 9: Comparison of Baseline vs. FAF-applied results on ECL dataset.



(a) Baseline (b) *FAF*

(c) Baseline (d) *FAF*

Figure 10: Comparison of Baseline vs. FAF-applied results on Exhange dataset.



(a) Baseline (b) *FAF*

(c) Baseline (d) *FAF*

Figure 11: Comparison of Baseline vs. FAF-applied results on Traffic dataset.

Table 10: Ablation on ETTh/m datasets (MSE and MAE) using TimePFN for MTS forecasting (input length and forecast horizon 96). *FAF* yields almost uniformly superior performance, followed by *AdaRho*.

Dataset	FAF		Augmentation		AdaRho		RHO		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

Table 11: Dispersion of reduc. loss

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

H. Ablations

We report an ablation study in Table 10, 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 removing each component separately. 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 & 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.

As part of our ablation studies, we report the quantile coefficient of dispersion (QCD) and the coefficient of variation (CV), both common measures of data dispersion. QCD, defined as $\text{QCD} = \frac{Q_3 - Q_1}{Q_3 + Q_1}$, uses the first and third quartiles and is more robust to outliers. CV is given by $\text{CV} = \frac{\sigma}{\mu}$, the ratio of standard deviation to mean. We find that when dispersion is low (e.g., all data points are equally informative or noise cancels via averaging), filtering yields smaller performance gains. This provides a simple guideline for when to apply *FAF*.

H.1. 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 an ablation study below 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.

Table 12: 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