Accurate Parameter-Efficient Test-Time Adaptation for Time Series Forecasting

Heitor R. Medeiros [*1 2] **Hossein Sharifi-Noghabi** [1] **Gabriel L. Oliveira** [1] **Saghar Irandoust** [1]

Abstract

Real-world time series often exhibit a non-stationary nature, degrading the performance of pre-trained forecasting models. Test-Time Adaptation (TTA) addresses this by adjusting models during inference, but existing methods typically update the full model, increasing memory and compute costs. We propose PETSA, a parameter-efficient method that adapts forecasters at test time by only updating small calibration modules on the input and output. PETSA uses low-rank adapters and dynamic gating to adjust representations without retraining. To maintain accuracy despite limited adaptation capacity, we introduce a specialized loss combining three components: (1) a robust term, (2) a frequency-domain term to preserve periodicity, and (3) a patch-wise structural term for structural alignment. PETSA improves the adaptability of various forecasting backbones while requiring fewer parameters than baselines. Experimental results on benchmark datasets show that PETSA achieves competitive or better performance across all horizons. Our code is available at: https://github.com/BorealisAI/PETSA.

1. Introduction

Time series forecasting (TSF) plays a critical role in applications such as weather prediction, traffic monitoring, and financial modeling (Wu et al., 2021; Zhou et al., 2021; Kudrat et al., 2025). While deep learning models like Transformers and MLPs have significantly improved TSF performance (Shabani et al., 2022; Wang et al., 2025), they often assume stationarity and struggle when the data distribution shifts over time (Kim et al., 2025). In practice, such shifts, which are normally caused by seasonality, structural breaks, or domain shifts, lead to a significant degradation in accuracy (Kim et al., 2024; 2025).

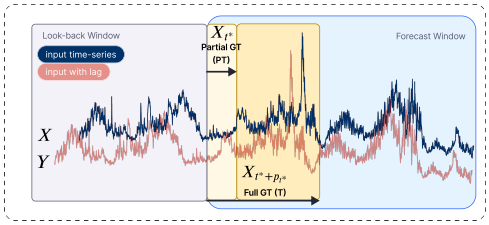

Figure 1. **Illustration of the test-time adaptation setup in PETSA.** The model observes a look-back window and makes predictions over the forecast window. A partial portion of the ground truth (PT) becomes available shortly after prediction (light yellow), which is used to adapt the model online. Full ground truth (T) may also be observed after the forecast window completes (shaded yellow). PETSA uses both partial and delayed T to update lightweight calibration modules during inference. The X is the time-series input, and Y is the same time-series with a lag, which can be partially used as ground truth. The X_{t^*} is the input at time t^* and the partial batch goes until $t^* + p_{t^*}$ timestep.

Test-Time Adaptation (TTA) has emerged as a promising strategy to mitigate these shifts by updating models during inference (Wang et al., 2020; Kim et al., 2025). However, most TTA methods either rely on access to source data (Wang et al., 2020) or update the entire model (Hu et al., 2022), resulting in high computational overhead. Furthermore, limited information at test time makes reliable adaptation challenging (Kim et al., 2024; Kudrat et al., 2025) In this paper, we introduce Parameter-Efficient Time-Series Adaptation (PETSA) framework (Figure 2), tailored for test-time adaptation of time-series forecasters.

Our main contributions can be summarized as follows.

(1) We propose PETSA, a test-time adaptation framework that calibrates input and output features using lightweight low-rank adapters and dynamic gating.

(2) We design a unified PETSA loss combining Huber, frequency, and patch-wise structural terms for robust and structure-aware adaptation.

(3) We benchmark PETSA across six datasets and show that it improves multiple forecasters while maintaining high efficiency.

[*]Work done during an internship at Borealis AI.

[1]Borealis AI, Montreal, Canada [2]Dept. of Systems Engineering, ETS Montreal, Canada. Correspondence to: Heitor R. Medeiros <heitor.rapela-medeiros.1@ens.etsmtl.ca>.

Second Workshop on Test-Time Adaptation: Putting Updates to the Test! at ICML 2025, Vancouver, Canada. 2025. Copyright 2025 by the author(s).

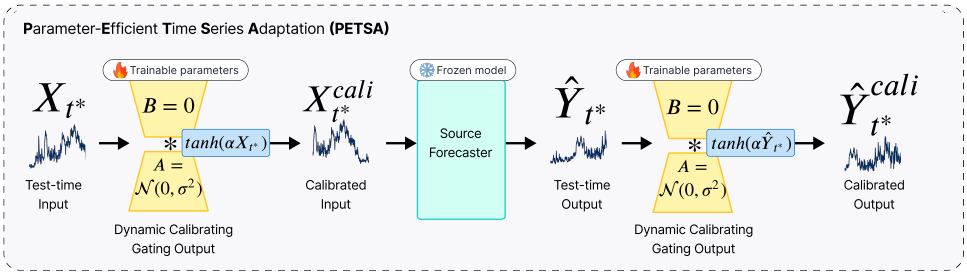

Figure 2. **PETSA.** At test time, the input X_t is first passed through a dynamic input calibration module that applies a gated low-rank transformation. The calibrated input X_t^{cali} is then processed by a frozen pre-trained forecaster. Its output $\hat{Y}_{t*}$ is refined by a similar output calibration module to produce the final prediction $\hat{Y}_{t*}^{\text{cali}}$. Only the calibration modules are updated during test-time adaptation using the PETSA loss with partially and fully observed ground truth available with a delay. Modules with trainable parameters have a fire icon, while frozen ones have an ice icon.

2. Related Works

Time-Series Forecasting (TSF). Recent TSF models span Transformers, linear projections, and MLP-based forecasters. Transformer-based models like iTransformer (Liu et al., 2024b) and PatchTST (Nie et al., 2023) capture long-range dependencies through self-attention, while linear approaches such as DLinear (Zeng et al., 2023) and OLS (Toner & Darlow, 2024) offer competitive performance with lower complexity. MLP-based methods like FreTS (Yi et al., 2023) and MICN (Wang et al., 2023) balance expressiveness and efficiency using global/local mixing. These models highlight the trade-off between accuracy and computational cost in TSF.

Parameter-Efficient Fine-Tuning (PEFT). PEFT techniques adapt large models using a small number of tunable parameters. Popular strategies include LoRA (Hu et al., 2022), DoRA (Liu et al., 2024a), and visual adapters like VPT (Jia et al., 2022) or AdaptFormer (Chen et al., 2022). While PEFT has seen wide use in vision and NLP, recent efforts extend to TSF (Gupta et al., 2024; Ruan et al., 2024; Nie et al., 2024). However, existing methods mainly focus on fine-tuning and do not address test-time adaptation.

Test-Time Adaptation (TTA). TTA enables models to adapt to distribution shifts during inference using unlabeled data (Zhao et al., 2023; Liang et al., 2025). Techniques like TENT (Wang et al., 2020), LAME (Boudiaf et al., 2022), and entropy minimization update model statistics or outputs. In TSF, TAFAS (Kim et al., 2025) introduces a batch-level adaptation scheme using delayed partial labels. PETSA builds on this line by introducing a parameter-efficient, gating-based architecture with specialized losses for robust and structured test-time adaptation.

3. Proposed Method

3.1. Preliminary Definitions

TSF. TSF involves predicting future values of a sequence based on historical observations. Formally, given a historical multivariate time series $X = \{x_{t-L}, x_{t-L+1}, \ldots, x_{t-1}\}$ consisting of L consecutive observations, the goal of TSF is to learn a forecasting model $f_\theta(\cdot)$ that generates accurate predictions of the next T future steps, denoted as: $Y = \{x_t, x_{t+1}, \ldots, x_{t+T-1}\} = f_\theta(X)$.

TTA in TSF. In TSF, TTA mitigates distribution shifts by updating the model using only test inputs. Methods like TAFAS assume that partial ground truth becomes available shortly after prediction, enabling online updates. The adaptation window is defined using the dominant period, estimated via Fast Fourier Transform (FFT). PETSA adopts this setup, using both partial and full labels to update its lightweight gating modules during inference, same as in TAFAS (Kim et al., 2025).

3.2. PETSA

We propose Parameter-Efficient Time-Series Adaptation (PETSA), a lightweight framework, designed to adapt time-series forecasting models at inference without modifying the core model parameters. It introduces input and output calibration modules that leverage low-rank adapters and dynamic gating mechanisms to correct for distribution shifts.

Dynamic Calibration Mechanism. At test time, PETSA calibrates both the input and output of a frozen forecaster using lightweight low-rank adapters and dynamic gating, inspired by Dynamic Tanh (DyT) (Zhu et al., 2025) and TAFAS. The calibrated input ($\hat{X}_{t*}^{\text{cali}}$) and calibrated output ($\hat{Y}_{t*}^{\text{cali}}$) are computed as follows:

Table 1. **MSE across datasets and window sizes**. The input training sequence length is set to 96 for all baselines. Results for ✗-checkpoint, TF - TAFAS, and PETSA - PT. The lower MSE is marked in bold. Additionally, we provided a row-counter (RW), which counts the winner for each row, meaning the best for the window length on the dataset among all models, and a column-counter (CW), with the winner per model, and the total sum of column winners.

	Models	Transformer-based						Linear-based						MLP-based						Counter	
		iTransformer			PatchTST			DLinear			OLS			FreTS			MICN			RW	
	Wind.	✗	TF	PT	✗	TF	PT	✗	TF	PT	✗	TF	PT	✗	TF	PT	✗	TF	PT	TF	PT
ETTh1	96	0.449	0.435	**0.432**	0.433	**0.426**	**0.426**	0.470	0.462	**0.459**	0.451	0.442	**0.440**	0.446	0.440	**0.438**	0.520	**0.493**	**0.493**	2	**6**
	192	0.510	0.503	**0.501**	0.491	0.482	**0.481**	0.521	0.512	**0.511**	0.505	**0.492**	**0.492**	0.502	0.494	**0.492**	0.591	0.560	**0.559**	1	**6**
	336	0.564	0.562	**0.561**	0.555	0.546	**0.543**	0.566	0.560	**0.555**	0.551	0.542	**0.538**	0.554	0.548	**0.547**	0.665	**0.632**	0.643	1	**5**
	720	0.702	0.663	**0.659**	0.706	**0.680**	**0.680**	0.712	0.682	**0.679**	0.700	0.666	**0.650**	0.718	**0.687**	0.688	0.904	0.792	**0.785**	2	**5**
	Avg	0.557	0.541	**0.538**	0.546	0.533	**0.532**	0.567	0.554	**0.551**	0.552	0.535	**0.530**	0.555	0.542	**0.541**	0.670	**0.619**	0.620	1	**5**
ETTm1	96	0.439	0.416	**0.413**	0.451	0.437	**0.436**	0.444	0.417	**0.414**	0.444	0.416	**0.415**	0.433	0.421	**0.416**	0.487	0.458	**0.456**	0	**6**
	192	0.508	0.476	**0.473**	0.504	**0.486**	0.489	0.518	0.480	**0.474**	0.518	0.479	**0.475**	0.501	0.482	**0.475**	0.554	0.511	**0.510**	1	**5**
	336	0.613	0.556	**0.552**	0.558	**0.539**	0.542	0.593	0.549	**0.545**	0.593	0.548	**0.543**	0.570	0.547	**0.543**	0.612	0.579	**0.573**	1	**5**
	720	0.485	0.453	**0.450**	0.479	**0.463**	0.465	0.482	0.449	**0.446**	0.481	0.449	**0.446**	0.468	0.452	**0.448**	0.525	0.486	**0.484**	1	**5**
	Avg	0.257	0.255	**0.254**	0.236	**0.235**	**0.235**	0.232	**0.230**	**0.230**	0.231	**0.228**	**0.228**	0.239	**0.236**	**0.236**	0.256	**0.252**	**0.252**	5	**6**
ETTh2	96	0.344	0.330	**0.328**	0.317	**0.308**	0.309	0.325	0.319	**0.318**	0.326	0.319	**0.318**	0.332	**0.321**	**0.321**	0.359	**0.339**	0.342	3	**4**
	192	0.424	**0.396**	0.397	0.433	**0.402**	**0.402**	0.409	0.387	**0.385**	0.416	0.391	**0.388**	0.412	**0.383**	**0.383**	0.437	0.439	**0.434**	3	**5**
	336	0.332	0.320	**0.319**	0.318	**0.305**	**0.305**	0.313	**0.305**	**0.305**	0.314	0.305	**0.304**	0.317	**0.306**	**0.306**	0.345	**0.334**	0.335	4	**5**
	720	0.168	0.167	**0.166**	**0.160**	**0.160**	**0.160**	0.160	**0.158**	**0.158**	0.160	**0.159**	**0.159**	0.158	**0.157**	**0.157**	**0.175**	**0.175**	0.176	**5**	**5**
	Avg	0.220	0.217	**0.215**	0.207	**0.204**	**0.204**	0.193	**0.191**	**0.191**	0.194	**0.192**	**0.192**	0.192	**0.191**	**0.191**	0.213	0.209	**0.203**	4	**6**
ETTm2	96	0.339	0.330	**0.322**	0.334	**0.327**	0.328	0.306	0.297	**0.296**	0.307	**0.298**	**0.298**	0.301	**0.292**	0.293	0.332	0.322	**0.320**	3	**4**
	192	0.250	0.244	**0.241**	0.237	**0.235**	**0.235**	0.223	**0.219**	**0.219**	0.223	**0.220**	**0.220**	0.221	**0.217**	0.218	0.243	0.238	**0.236**	4	**5**
	336	0.087	**0.085**	0.086	0.086	**0.082**	0.083	0.091	0.089	**0.088**	0.081	0.080	**0.078**	0.083	**0.079**	**0.079**	0.115	0.115	**0.109**	3	**4**
	720	0.181	**0.174**	0.175	0.188	**0.174**	0.179	0.183	0.176	**0.173**	0.173	**0.164**	0.165	0.173	0.164	**0.163**	0.216	**0.198**	**0.198**	**4**	3
	Avg	0.343	**0.313**	0.335	0.338	**0.281**	0.332	0.328	0.294	**0.292**	0.323	0.285	**0.281**	0.324	**0.295**	0.298	0.398	0.304	**0.280**	**3**	**3**
Exchange	96	0.366	0.345	**0.341**	0.372	**0.353**	0.367	0.372	0.359	**0.357**	0.353	0.294	**0.286**	0.354	0.335	**0.327**	0.558	**0.307**	0.357	2	**4**
	192	0.173	**0.166**	**0.166**	0.173	**0.170**	0.171	0.195	0.180	**0.176**	0.196	0.181	**0.178**	0.186	0.175	**0.174**	0.176	0.175	**0.174**	2	**5**
	336	0.223	**0.211**	0.212	0.220	**0.214**	0.216	0.240	0.224	**0.223**	0.241	**0.222**	0.223	0.231	**0.215**	0.218	0.224	**0.217**	0.220	**5**	1
	720	0.281	**0.261**	0.265	0.276	**0.265**	0.268	0.292	**0.271**	**0.271**	0.292	**0.271**	0.273	0.284	**0.264**	0.266	0.281	0.269	**0.268**	**5**	2
	Avg	0.355	**0.339**	0.341	0.355	0.337	**0.336**	0.364	0.350	**0.345**	0.364	**0.344**	0.346	0.360	**0.340**	0.344	0.353	0.347	**0.345**	**3**	**3**
Weather	96	0.173	**0.166**	**0.166**	0.173	**0.170**	0.171	0.195	0.180	**0.176**	0.196	0.181	**0.178**	0.186	0.175	**0.174**	0.176	0.175	**0.174**	2	**5**
	192	0.223	**0.211**	0.212	0.220	**0.214**	0.216	0.240	0.224	**0.223**	0.241	**0.222**	0.223	0.231	**0.215**	0.218	0.224	**0.217**	0.220	**5**	1
	336	0.281	**0.261**	0.265	0.276	**0.265**	0.268	0.292	**0.271**	**0.271**	0.292	**0.271**	0.273	0.284	**0.264**	0.266	0.281	0.269	**0.268**	**5**	2
	720	0.355	**0.339**	0.341	0.355	0.337	**0.336**	0.364	0.350	**0.345**	0.364	**0.344**	0.346	0.360	**0.340**	0.344	0.353	0.347	**0.345**	**3**	**3**
	Avg	0.258	**0.244**	0.246	0.256	**0.247**	0.248	0.273	0.256	**0.254**	0.273	**0.255**	**0.255**	0.265	**0.248**	0.251	0.258	**0.252**	**0.252**	**5**	3
		✗	TF	PT	✗	TF	PT	✗	TF	PT	✗	TF	PT	✗	TF	PT	✗	TF	PT		
Counter	CW		13	**19**		**24**	14		7	**30**		14	**23**		18	**19**		12	**22**		
Sum Col.	TF: 88	PT: **127**																			

$$\hat{X}^{\text{cali}}_{t^*} = X_{t^*} + (\tanh(\alpha \odot X_{t^*}) \cdot W + b)$$
$$\hat{Y}^{\text{cali}}_{t^*} = \hat{Y}_{t^*} + \left(\tanh(\alpha \odot \hat{Y}_{t^*}) \cdot W + b\right), \quad (1)$$

where $X_{t^*} \in \mathbb{R}^{B\times L\times V}$ and $\hat{Y}_{t^*} \in \mathbb{R}^{B\times L\times V}$ are the test-time input and output respectively, $\alpha \in \mathbb{R}^V$ is a learnable gating parameter per variable (we control the initialization with a hyperparameter), applied element-wise, $W = A \cdot B$, with $A \in \mathbb{R}^{L\times r}$, $B \in \mathbb{R}^{r\times L\times V}$ forming the low-rank weight tensor, $b \in \mathbb{R}^{L\times V}$is a learnable bias term (A is initialized with Xavier Norm. and B with zeros). This enables PETSA to efficiently calibrate time-series representations by updating only a small number of parameters at test time.

PETSA Optimization. PETSA uses a combination of different losses, while TAFAS only uses MSE loss. Our PETSA loss combines total and partial losses ($\mathcal{L}_{PETSA} = \mathcal{L}_T + \mathcal{L}_{pt}$) , where $\mathcal{L}_T$ is computed using delayed full ground-truth labels and $\mathcal{L}_{pt}$ uses partially observed labels (Kim et al., 2025). Each loss term incorporates three components: (1) a Huber loss ($\mathcal{L}_{\text{Hub}}$) (Huber, 1992) for robustness to outliers (Shabani et al., 2022), described as follows:

$$\mathcal{L}_{Hub} = \begin{cases} 0.5(\hat{Y}^{cali}_{t^*} - Y_{t^*})^2, \text{if } |\hat{Y}^{cali}_{t^*} - Y_{t^*}| < \delta \\ \delta \cdot (|\hat{Y}^{cali}_{t^*} - Y_{t^*}| - 0.5 \cdot \delta), \text{otherwise.} \end{cases} \quad (2)$$

where δ is a hyperparameter to control the sensitivity to outliers and smoothness of the predictions (in this work, δ is fixed at 0.5), (2) a frequency-domain loss ($\mathcal{L}_{\text{freq}}$) that aligns the FFT spectra of predictions and ground truth to preserve periodic patterns, while reducing estimation bias, as described in FreDF (Wang et al., 2025), described as follows:

$$\mathcal{L}_{freq} = \left\| \mathcal{F}(\hat{Y}^{cali}_{t^*}) - \mathcal{F}(Y_{t^*}) \right\|_1, \quad (3)$$

where $\mathcal{F}(.) = FFT$, and (3) a patch-wise structural loss ($\mathcal{L}_{\text{pw}}$) that captures local correlations, means, and variances to enhance structural alignment (Kudrat et al., 2025), described as follows:

$$\mathcal{L}_{pw} = \sum_{k \in \{\text{corr,mean,var}\}} \mathcal{L}_k(\hat{Y}^{cali}_{t^*}, Y_{t^*}). \quad (4)$$

Finally, the partial (PT) and delayed GT (T) loss are described as follows:

$$\mathcal{L}_{pt} = \mathcal{L}_{Hub_{pt}} + \mathcal{L}_{pw_{pt}} + \beta \mathcal{L}_{freq_{pt}} \quad (5)$$
$$\mathcal{L}_{T} = \mathcal{L}_{Hub_{T}} + \mathcal{L}_{pw_{T}} + \beta \mathcal{L}_{freq_{T}}. \quad (6)$$

To the best of our knowledge, this is the first work to do parameter-efficient TTA for TSF. By updating only a small set of calibration parameters at test time, PETSA enables fast, stable, and memory-efficient adaptation across a wide range of forecasting models and datasets.

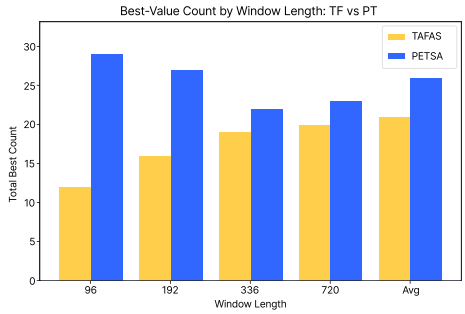

Figure 3. **Total number of best-value wins grouped by window length for TAFAS and PETSA approaches.** PETSA consistently outperforms TAFAS across all horizons.

4. Experiments

4.1. Experimental Protocol

(a) Datasets: We demonstrate the effectiveness of our method, PETSA, using widely used multivariate TSF benchmark datasets: ETTh1, ETTm1, ETTh2, ETTm2, Exchange, and Weather (Wu et al., 2021; Zhou et al., 2021).

(b) Implementation Details: Our framework is built on top of TAFAS (Kim et al., 2025). We used PyTorch for PETSA implementation, and training/adapt the models using one NVIDIA A100.

(c) Baselines: We evaluate our proposed method against a diverse set of baseline models, grouped into three main categories: (1) Transformer-based approaches, including iTransformer (Liu et al., 2024b), PatchTST (Nie et al., 2023); (2) Linear-based models, comprising DLinear (Zeng et al., 2023), OLS (Toner & Darlow, 2024), and (3) MLP-based influential architectures, such as FreTS (Yi et al., 2023), MICN (Wang et al., 2023). Additionally, we provide the methods without and with adaptation using TAFAS (Kim et al., 2025) and PETSA. We provide additional details in the Appendix.

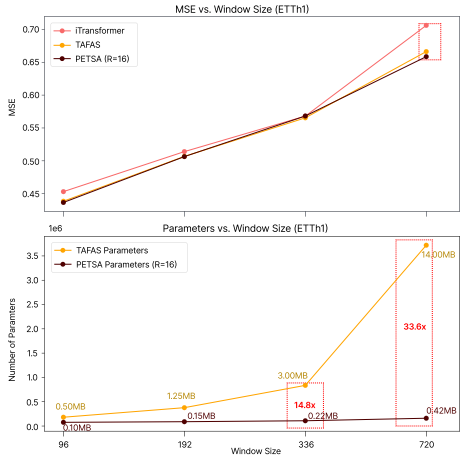

Figure 4. **Comparison of PETSA and TAFAS on ETTh1 for iTransformer.** Top: MSE across different window sizes with no adaptation, TAFAS, and PETSA. Bottom: Number of trainable parameters used for adaptation. PETSA achieves similar or better accuracy while using up to **33.6**× fewer parameters at window size 720. Memory usage is annotated in MB.

4.2. Results

In Table 1, across all datasets and model categories, PETSA achieves the highest number of best-MSE scores (127 wins), outperforming TAFAS (88 wins). Its consistent advantage across transformer-, linear-, and MLP-based architectures demonstrates strong adaptability, where all PETSA models had fewer parameters than TAFAS. Figure 3 shows that PETSA achieves more best-value scores than TAFAS across different window lengths. Even as the forecast window increases, PETSA maintains a strong lead, demonstrating robustness to longer-term uncertainty. In Figure 4, PETSA achieves consistently lower MSE across all window sizes, and for window size 720, it has 33× fewer parameters than TAFAS, highlighting its efficiency due to the low-rank adaptation with dynamic gating, which is input conditioned and more robust to outliers in long-range forecasting as a result of its loss optimization.

5. Conclusion

In this work, we introduced PETSA, a lightweight, parameter-efficient test-time adaptation framework for time-series forecasting that dynamically corrects both inputs and outputs via gated calibration modules. PETSA test-time calibration loss combines a robust component, a frequency-domain term to preserve dominant periodic patterns, and a patch-wise structural term to enforce structural alignment, which are essential to adapt the forecaster during test-time. Across diverse benchmarks, PETSA consistently improves forecasting performance while updating fewer parameters against baselines.

References

Boudiaf, M., Mueller, R., Ben Ayed, I., and Bertinetto, L. Parameter-free online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8344–8353, 2022.

Chen, S., Ge, C., Tong, Z., Wang, J., Song, Y., Wang, J., and Luo, P. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678, 2022.

Gupta, D., Bhatti, A., Parmar, S., Dan, C., Liu, Y., Shen, B., and Lee, S. Low-rank adaptation of time series foundational models for out-of-domain modality forecasting. In *Proceedings of the 26th International Conference on Multimodal Interaction*, pp. 382–386, 2024.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Huber, P. J. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518. Springer, 1992.

Jia, M., Tang, L., Chen, B.-C., Cardie, C., Belongie, S., Hariharan, B., and Lim, S.-N. Visual prompt tuning. In *European conference on computer vision*, pp. 709–727. Springer, 2022.

Kim, D., Park, S., and Choo, J. When model meets new normals: test-time adaptation for unsupervised time-series anomaly detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 13113–13121, 2024.

Kim, H., Kim, S., Mok, J., and Yoon, S. Battling the non-stationarity in time series forecasting via test-time adaptation. *arXiv preprint arXiv:2501.04970*, 2025.

Kudrat, D., Xie, Z., Sun, Y., Jia, T., and Hu, Q. Patch-wise structural loss for time series forecasting. *arXiv preprint arXiv:2503.00877*, 2025.

Liang, J., He, R., and Tan, T. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64, 2025.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024a.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024b.

Nie, T., Mei, Y., Qin, G., Sun, J., and Ma, W. Channel-aware low-rank adaptation in time series forecasting. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 3959–3963, 2024.

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

Ruan, W., Chen, W., Dang, X., Zhou, J., Li, W., Liu, X., and Liang, Y. Low-rank adaptation for spatio-temporal forecasting. *arXiv preprint arXiv:2404.07919*, 2024.

Shabani, A., Abdi, A., Meng, L., and Sylvain, T. Scaleformer: Iterative multi-scale refining transformers for time series forecasting. *arXiv preprint arXiv:2206.04038*, 2022.

Toner, W. and Darlow, L. N. An analysis of linear time series forecasting models. In *International Conference on Machine Learning*, pp. 48404–48427. PMLR, 2024.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Wang, H., Peng, J., Huang, F., Wang, J., Chen, J., and Xiao, Y. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The eleventh international conference on learning representations*, 2023.

Wang, H., Pan, L., Chen, Z., Yang, D., Zhang, S., Yang, Y., Liu, X., Li, H., and Tao, D. Fredf: Learning to forecast in the frequency domain. In *ICLR*, 2025.

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

Yi, K., Zhang, Q., Fan, W., Wang, S., Wang, P., He, H., An, N., Lian, D., Cao, L., and Niu, Z. Frequency-domain mlps are more effective learners in time series forecasting. *Advances in Neural Information Processing Systems*, 36: 76656–76679, 2023.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

Zhao, H., Liu, Y., Alahi, A., and Lin, T. On pitfalls of test-time adaptation. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 42058–42080, 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 *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Zhu, J., Chen, X., He, K., LeCun, Y., and Liu, Z. Transformers without normalization. *arXiv preprint arXiv:2503.10622*, 2025.

Appendix

Here we provide additional information and results for our paper.

Reproducing the results

Our codebase is built on top of TAFAS (Kim et al., 2025), where we followed their experimental setup and hyperparameters for generating the baseline checkpoint models and adapted models. Additionally, for our method, we have hyperparameters to control the frequency loss, the number of low-rank parameters, and the gating initialization, where we provide additional ablations in the next session.

Additional results on Parameter-Efficiency

In Figure 5, PETSA and TAFAS show very similar MSE results across all window sizes on the ETTh1 dataset using the OLS model. Both methods follow the same trend, with PETSA slightly outperforming TAFAS at larger windows. In terms of parameters, PETSA remains highly efficient, using only 0.21 MB at window size 720, while TAFAS requires 3.70 MB. Across all window sizes, PETSA keeps memory usage consistently low while achieving comparable or better performance, highlighting its parameter efficiency.

In Figure 6, PETSA achieves similar MSE to TAFAS across all window sizes on ETTm1 with OLS, with slightly better results at short horizons. For a window of 720, we kept the memory low at 0.11MB, while TAFAS required 3.70. As this dataset is easier than the other one, we had a good trade-off between performance and memory.

We had similar trends for Figures 7 and Figures 8, we are comparable with TAFAS in terms of MSE, and the memory is also lower. However, we can see that ETTh1/ETTh2 requires a bit more memory than ETTm1/ETTm2 to achieve competitive results. This trade-off happens due to the fact that ETTh1/ETTh2 datasets are more challenging than ETTm1/ETTm2; thus, more memory is required to remain performing well in terms of MSE and still being parameter efficient compared to TAFAS.

In Figure 9, PETSA and TAFAS show similar MSE trends across window sizes, with both methods degrading as the horizon increases. However, PETSA requires over 4× less memory than TAFAS at window size 720, making it a much more efficient alternative. Finally, in Figure 10, despite similar performance, PETSA significantly reduces the adaptation cost, using less than half of the memory compared to TAFAS at window size 720.

Ablation on Low-Rank (R) parameter

In this ablation, Figure 11, we conduct a study about the low-rank hyperparameter, which directly impacts the number of additional trainable parameters for the dynamic gating mechanism.

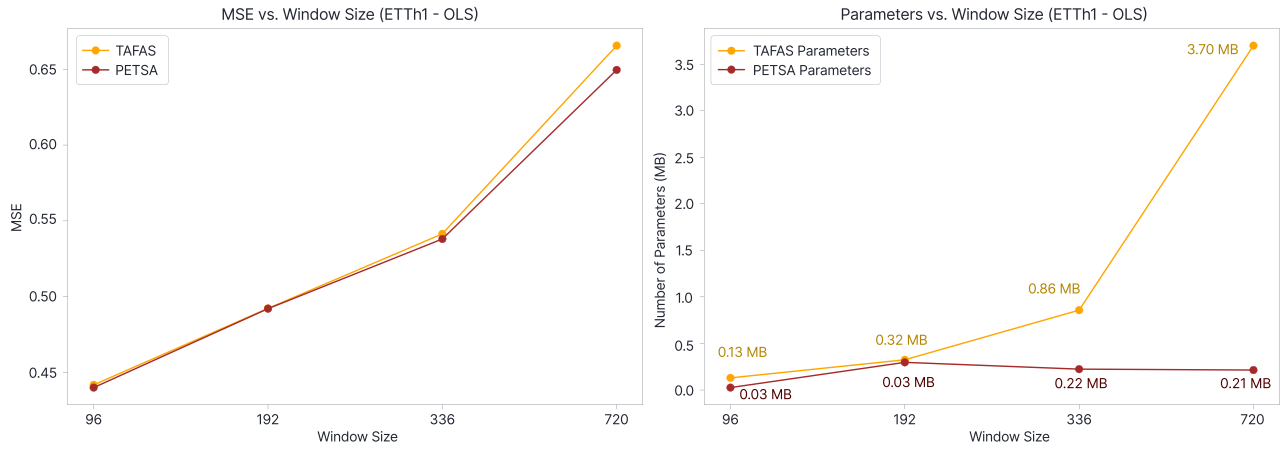

Figure 5. **Comparison of PETSA and TAFAS on ETTh1 for OLS.** Left: MSE across different window sizes with TAFAS, and PETSA. Right: Number of trainable parameters used for adaptation. Memory usage is annotated in MB.

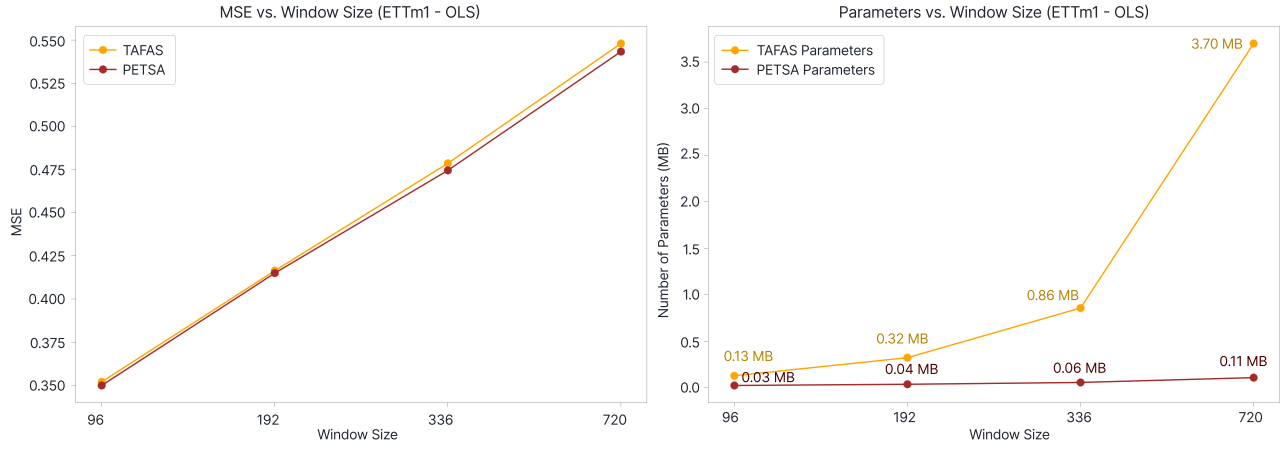

Figure 6. **Comparison of PETSA and TAFAS on ETTm1 for OLS.** Left: MSE across different window sizes with TAFAS, and PETSA. Right: Number of trainable parameters used for adaptation. Memory usage is annotated in MB.

Ablation on Dynamic Gating parameter

In this ablation, Figure 12, we study the initial value for the dynamic gating. This hyperparameter impacts the weights of the low-rank adaptation, providing a learnable way conditioned on the input to adjust its values; higher values make the weights positive due to the tanh; otherwise, lower values make the adapted weight negative, decreasing the value of the final calibrated input.

Ablation on Loss Components

In this ablation, we study the impact of the loss components for PETSA during TTA. In Figure 13, we see that the MSE loss is not sufficient for reaching the best performance values in terms of test MSE, similar to what occurs with only Huber loss. However, the total loss got the best results for ETTh1 OLS with β equal to 0.0, which means that the frequency component harmed the performance for this dataset, and $\beta = 0.0$ means that only the Huber loss and structural patch components are being used. Depending on the model, the frequency loss helps the performance; for instance, the best performance for the FreTS model was when the β was equal to 0.1 (for the majority of the datasets and windows). For some datasets, a higher value can be the best result, so we recommend hyperparameter tuning for optimal performance.

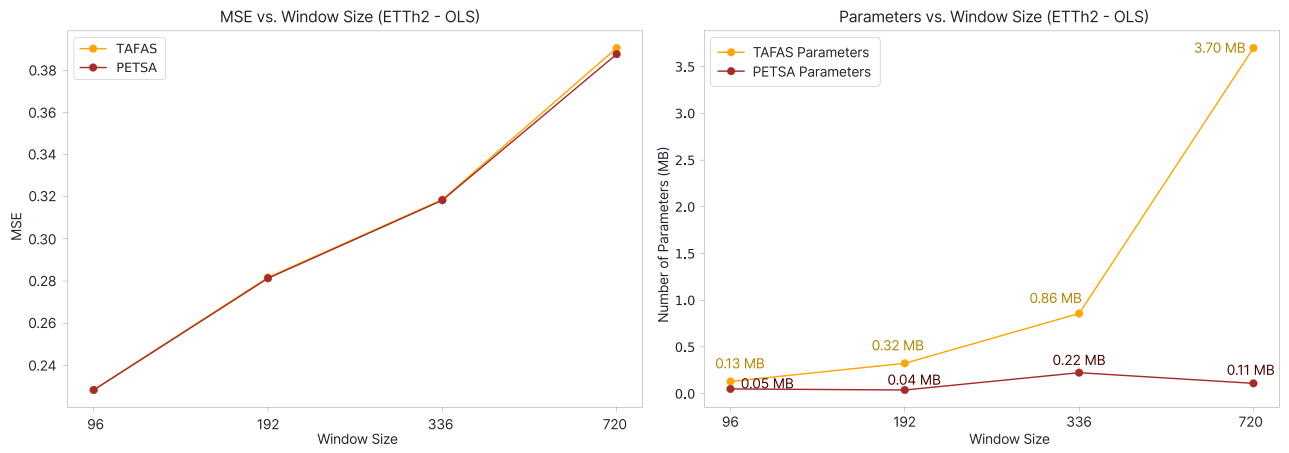

Figure 7. **Comparison of PETSA and TAFAS on ETTh2 for OLS.** Left: MSE across different window sizes with TAFAS, and PETSA. Right: Number of trainable parameters used for adaptation. Memory usage is annotated in MB.

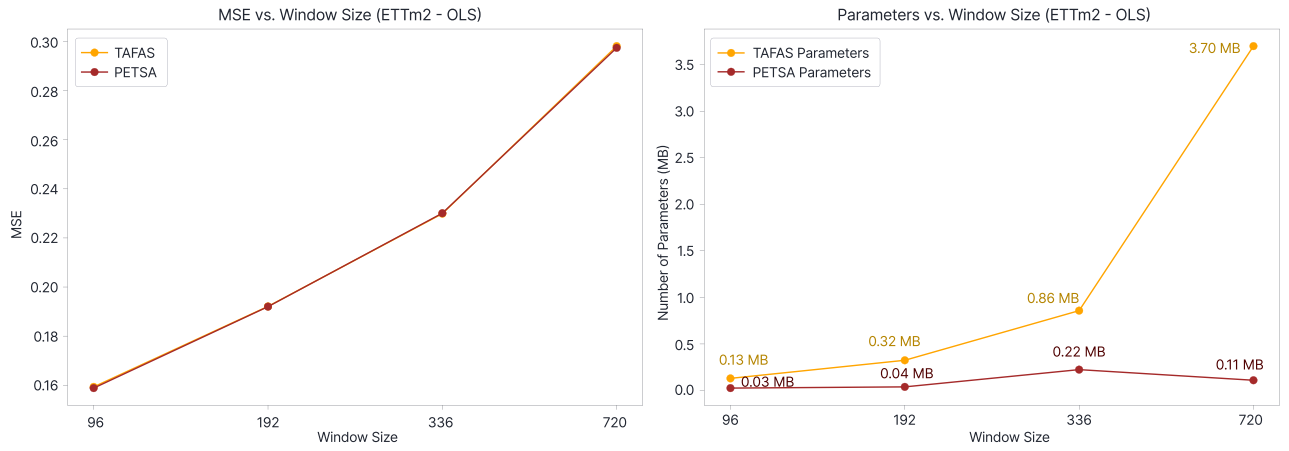

Figure 8. **Comparison of PETSA and TAFAS on ETTm2 for OLS.** Left: MSE across different window sizes with TAFAS, and PETSA. Right: Number of trainable parameters used for adaptation. Memory usage is annotated in MB.

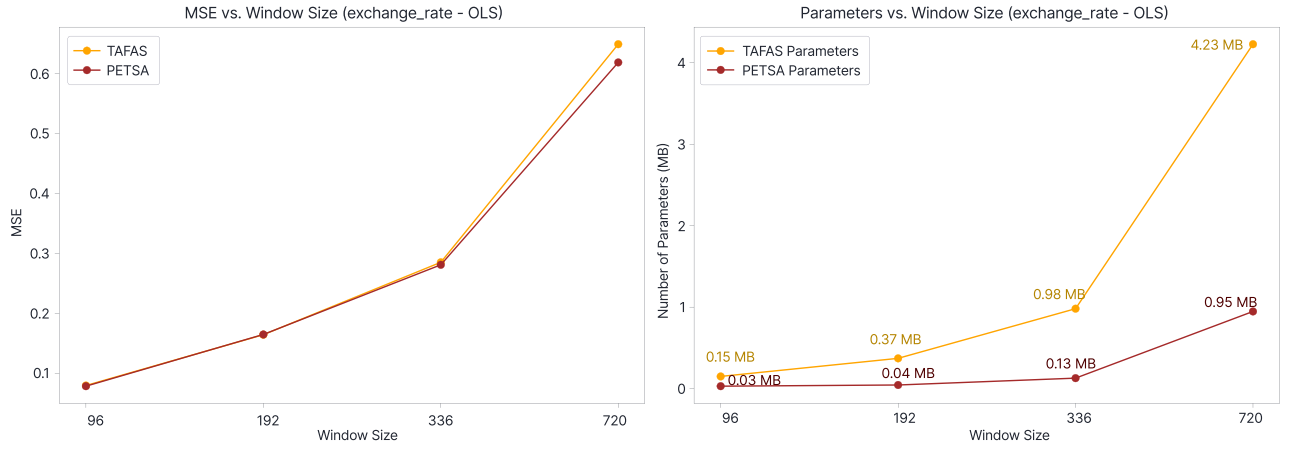

Figure 9. **Comparison of PETSA and TAFAS on Exchange Rate for OLS.** Left: MSE across different window sizes with TAFAS, and PETSA. Right: Number of trainable parameters used for adaptation. Memory usage is annotated in MB.

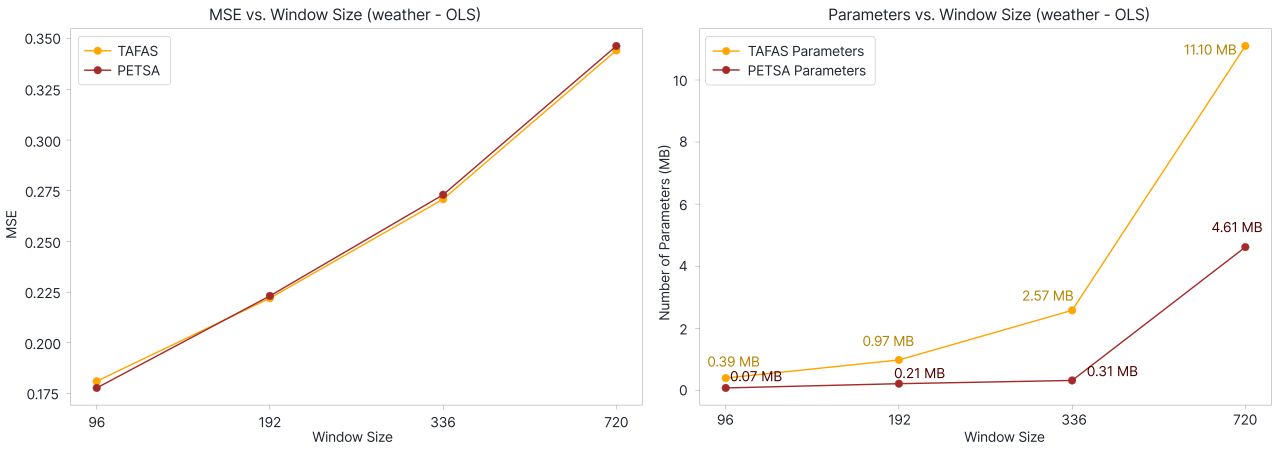

Figure 10. **Comparison of PETSA and TAFAS on Weather for OLS.** Left: MSE across different window sizes with TAFAS, and PETSA. Right: Number of trainable parameters used for adaptation. Memory usage is annotated in MB.

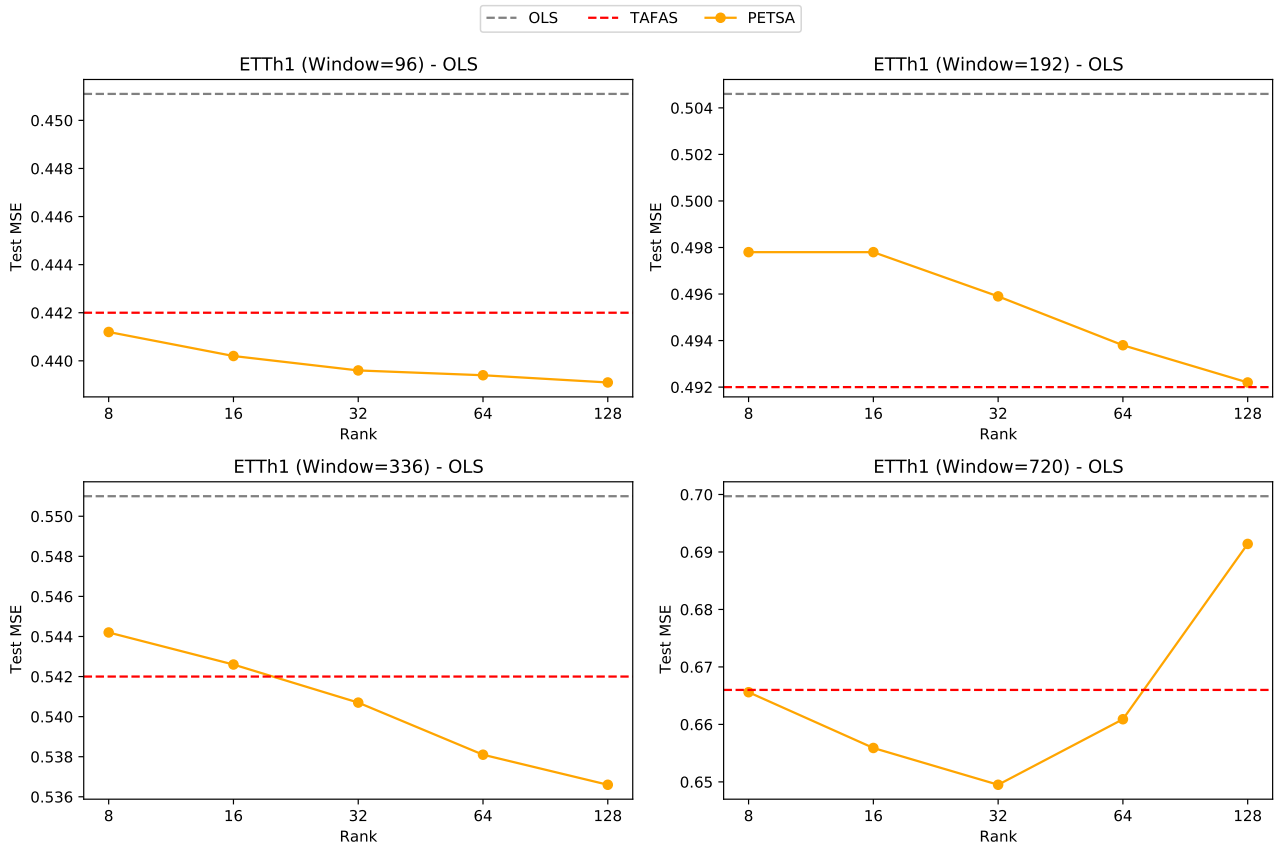

Figure 11. **Comparison of the original model, TAFAS, and PETSA on ETTh1 for OLS.** MSE across different ranks for windows 96, 196, 336, and 720.

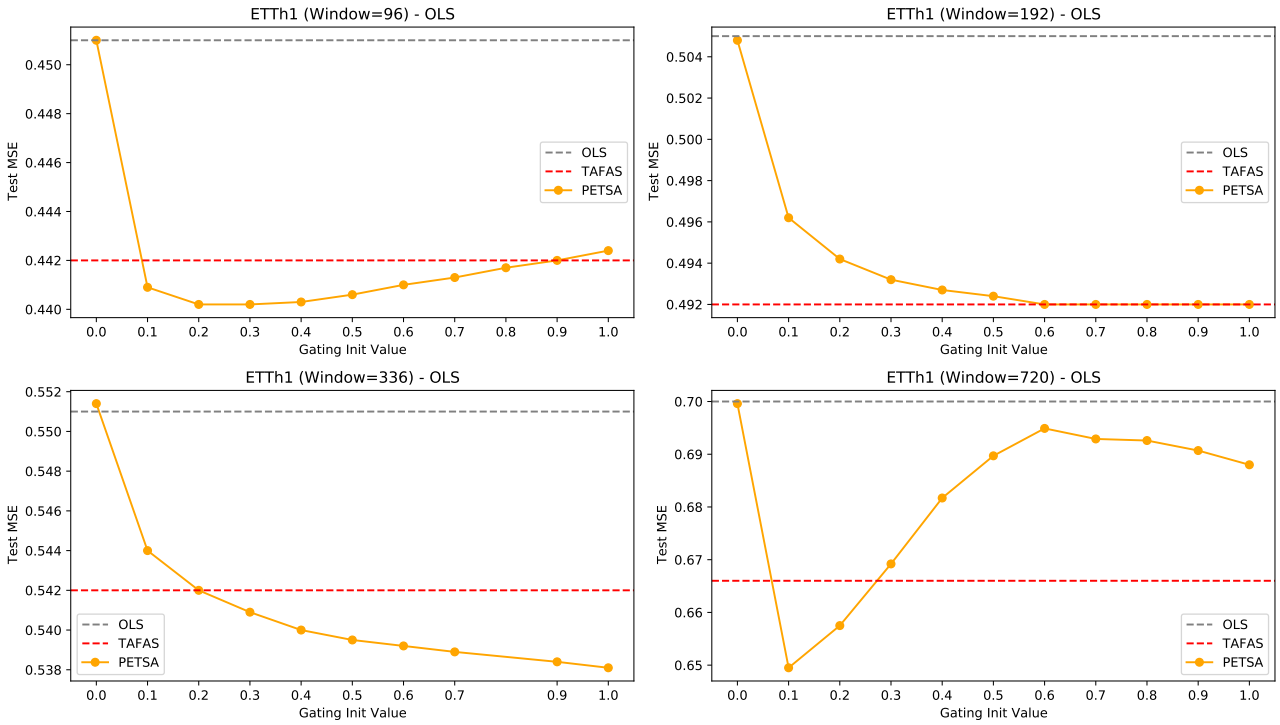

Figure 12. **Comparison of the original model, TAFAS, and PETSA on ETTh1 for OLS.** MSE across different gating initial values for windows 96, 196, 336, and 720.

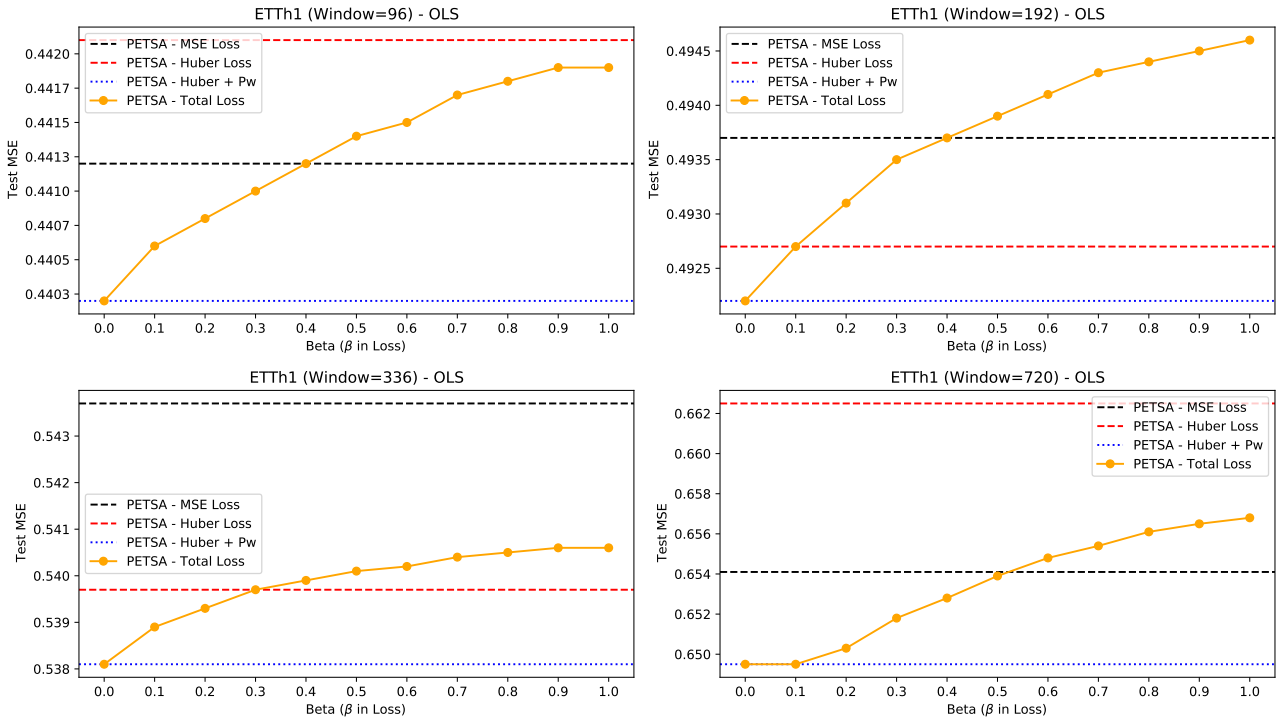

Figure 13. **Comparison of different loss terms in PETSA on ETTh1 for OLS.** MSE across different beta values for windows 96, 196, 336, and 720.