Shift-Aware Test-Time Adaptation and Benchmarking for Time-Series Forecasting

Shivam Grover [1] **Ali Etemad** [1]

Abstract

Test time adaptation (TTA) has shown promise in addressing distribution shifts in different areas, but remains significantly underexplored in time-series forecasting (TSF), where temporal dependencies and the evolving nature of the signals present unique challenges. We present DynaTTA, a dynamic TTA framework for TSF that estimates distribution shifts in real time by tracking prediction errors and embedding drift. This estimate allows us to employ two key mechanisms, a dynamic adaptation rate that is adjusted based on the severity of the shift, and shift-conditioned gating that controls the influence of the learned adaptations as required. These mechanisms enable meaningful and appropriate adaptations in the presence of distribution shifts, while retaining the prior knowledge of the source model. DynaTTA is modular and can be used with any existing pretrained model for TSF, without requiring retraining. We also propose TTFBench, a first-of-its-kind benchmark for evaluating TTA for TSF, comprising thousands of time-series with varying types and intensities of shifts. Through extensive experiments with various backbones and datasets including TTFBench, we show that DynaTTA consistently improves performance. The code and data are available at https://github.com/shivam-grover/DynaTTA.

1. Introduction

Time-series forecasting (TSF) plays a critical role in a wide array of real-world applications, such as forecasting weather (He et al., 2021; Lin et al., 2022), energy consumption (Zhou et al., 2021; Bu & Cho, 2020; Wang et al., 2022), traffic (Bai et al., 2020; Cirstea et al., 2022), and financial markets (Cheng et al., 2022). While deep learning has advanced TSF by enabling models to capture long-range dependencies and complex dynamics, a core challenge persists: real-world time-series are often non-stationary, with shifting statistical properties that can degrade model generalization during inference (Kim et al., 2025). Test-time adaptation (TTA) has emerged as a promising approach to mitigate the effects of distribution shifts at inference by allowing models to update themselves during test-time without retraining from scratch. While extensively studied in classification tasks within computer vision (Wang et al., 2020; Chen et al., 2022) and language (Karmanov et al., 2024; Ma et al., 2023), TTA for TSF, simply referred to as TSF-TTA, requires fundamentally different considerations due to the sequential and regression-based nature of the task (Kim et al., 2025). Unlike classification settings, which typically assume each test sample as independent from others, TSF involves a continuous stream of data which follows a strong temporal structure. Furthermore, as new data arrives, the ground truth values for previous forecasts are gradually revealed, enabling potential feedback for adaptation. Through the following, we identify two distinct challenges in the area of TSF-TTA and subsequently propose two solutions to tackle them.

Problem 1: Dynamic nature of time-series. Existing methods for TSF-TTA (Kim et al., 2025; Christou et al., 2024) exhibit two key limitations: (1) they employ fixed adaptation rates that do not adjust based on the severity or nature of the distribution shift, leading to under or over-adaptation, and (2) while certain works (Kim et al., 2025) employ gating mechanisms to regulate how much adapted information is incorporated into predictions, these gates are not explicitly conditioned on the magnitude or direction of the distribution shift. As a result, the model cannot adjust its reliance on the original (pre-trained) model versus the adapted model. For instance, in scenarios where the distribution shift encountered at inference time is cyclic in nature, *i.e.* it temporarily deviates but eventually returns to the source domain, it would be desirable to reduce the impact of adaptation or even disable adaptation. Conversely, when facing novel or distant distributions, it would be more desirable to allow the adaptation module to take a more involved role. To address this, we propose **Dyna**mic **T**est-**T**ime **A**daptation (DynaTTA), a novel TTA approach for TSF that estimates

[1]Queen's University, Canada. Correspondence to: Shivam Grover <shivam.grover@queensu.ca>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

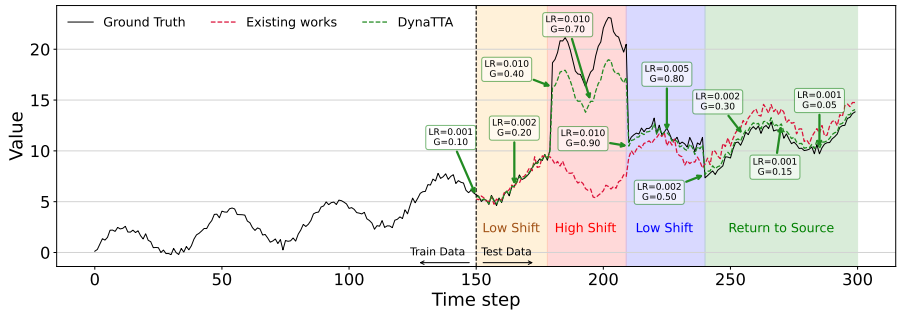


Figure 1. Comparison between DynaTTA, which dynamically adjusts adaptation rate (**LR**) and gating parameters (**G**) based on estimated distribution shift, and TAFAS (Kim et al., 2025) with fixed adaptation rate and non-dynamic gating. DynaTTA adapts more accurately because of its dynamic nature.

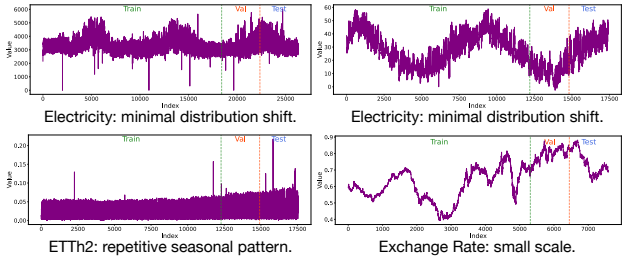


Figure 2. Train/validation/test splits for standard TSF datasets. Most datasets exhibit minimal distribution shift across splits, underscoring the need for stronger benchmarks.

the severity of distribution shift at test-time and uses it to dynamically control *when to adapt and how much to rely on the adapted model*. An illustrative example of these limitations and how dynamic adaptation can address them is shown in Figure 1.

Problem 2: Lack of suitable benchmarks for TSF-TTA. Standard TSF datasets often contain minimal or repetitive test-time shifts, limiting the evaluation of adaptation methods (Figure 2). To address this critical gap, we introduce a comprehensive benchmarking framework, **T**est-time adaptation **T**ime-series **F**orecasting **Bench**mark (TTFBench), explicitly designed for evaluating TSF-TTA methods under meaningful, varied, and controlled distribution shifts, built by injecting controlled perturbations (trends, seasonality, regime shifts, and localized noise) into existing datasets. DynaTTA, evaluated extensively on both standard and TTFBench datasets, consistently outperforms existing methods.

Our contributions are summarized as follows: ① We propose DynaTTA, a dynamic TTA framework that estimates distribution shifts by tracking historical MSE and embedding drift, and uses this shift estimate to adjust both the adaptation rate and adaptation gating, enabling aggressive updates for significant shifts, conservative updates for minor shifts, and disabling adaptation when the distribution aligns with the source. ② We introduce TTFBench, the first comprehensive benchmark suite for TSF-TTA. Our benchmark addresses the lack of coherent distribution shifts in existing datasets by injecting diverse perturbations into the test splits of standard forecasting datasets. ③ Our method consistently outperforms existing TTA methods across diverse TSF benchmarks (including TTFBench) and architectures, significantly reducing test-time errors. To contribute to the area of TSF, we make our solution and benchmark publicly available at: https://github.com/shivam-grover/DynaTTA

2. DynaTTA

Problem Statement. Given a multivariate time-series $\mathbf{X}_{1:T} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T] \in \mathbb{R}^{C \times T}$ with C channels over T time-steps, and a model f_θ trained on $\mathcal{D}_{\text{train}}$, the task of TSF is to predict H future steps using a context window of length L_{in} as $\hat{\mathbf{Y}}_t = f_\theta(\mathbf{X}^{\text{train}}_{t-L_{\text{in}}:t-1}) \in \mathbb{R}^{C \times H}$, for $t > L_{\text{in}}$. At test time, we encounter a target sequence $\mathbf{X} \sim \mathcal{D}_{\text{test}}(t)$, where the distribution $\mathcal{D}_{\text{test}}(t)$ may differ from $\mathcal{D}_{\text{train}}$, and may evolve over time, *i.e.*, $\mathcal{D}_{\text{test}}(t) \neq \mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}(t) \neq \mathcal{D}_{\text{test}}(t')$ for $t \neq t'$. In the TTA setting, we assume access to neither labels nor training data during inference. The primary goal is to dynamically update the model parameters θ at each test step t, using only the test stream $\{\mathbf{X}_s\}_{s<t}$. Furthermore, the adaptation procedure must be dynamic with respect to the intensity of distribution shift while assuming no direct access to the source data.

Estimating Distribution Shifts. We estimate shift using three metrics: (1) **MSE Z-score** which tracks recent performance degradation by tracking the model's test-time mean squared error (MSE) in a rolling buffer. A spike in z-score signals deviation from the training distribution; (2) **Short-term embedding drift** obtained using a real-time adaptation buffer (RTAB) that stores recent input embeddings and their (partial/full) MSEs; (3) long-term embedding drift obtained using a stable reference distribution buffer (RDB) that retains the top-K lowest-error embeddings over time and serves as a proxy for the source distribution, enabling detection of long-term divergence.

Shift-Conditioned Gating. In TTA, it is not only important

to decide how to adapt, but also when to adapt and how much to do so. While adaptation may be beneficial in the presence of distribution shift, it can be harmful when the current data remains close to the training distribution. To manage this trade-off, we introduce a gating mechanism that controls the strength of adaptation based on the degree of shift detected in the input stream. We introduce $\mathbf{m}_t \in \mathbb{R}^d$ for time-step t, as the set of shift metrics containing the MSE z-score, short-term RTAB embedding distance, and long-term RDB embedding distance, which were measured earlier. We initialize learnable parameters $\phi_{\text{base}} \in \mathbb{R}^C$ with all elements set to zero. We then pass $\mathbf{m}_t$ through a small multilayer perceptron (MLP) f_{gate} and its output (which represents the current estimated distribution shift) is used to obtain $\phi_{\text{dynamic}} = \phi_{\text{base}} + f_{\text{gate}}(\mathbf{m}_t)$. We use ϕ_{dynamic} to regulate the contribution of the adaptation module in calibrating the input as $\mathbf{X}_{\text{cal}} = \mathbf{X} + \tanh\left(\phi_{\text{dynamic}}\right) \circ (\mathbf{W} * \mathbf{X} + \mathbf{b})$ where $\mathbf{W}$ and $\mathbf{b}$ are learnable weights and bias for temporal calibration, $\circ$ denotes element-wise multiplication, and $*$ represents a variable-wise temporal transformation.

Dynamic Adaptation Rate Adjustment. A fixed adaptation rate across all inputs can lead to two primary issues. If the adaptation rate is too high, the model would unnecessarily adapt aggressively on inputs that are only mildly shifted. On the other hand, if the adaptation rate is too low, the model may fail to adapt quickly enough when faced with abrupt or significant distribution shifts. Because the intensity of shift can vary throughout the test stream, the adaptation rate should also vary accordingly. We first normalize each metric $\mathbf{m}_t^{(i)}$ using z-score, and then sum them together to obtain a shift score S_t. We then compute a multiplier through a scaled sigmoid as: $\lambda_t = 1 + \left(\frac{\alpha_{\max}}{\alpha_{\min}} - 1\right) \cdot \frac{1}{1+e^{-\kappa S_t}}$ where $\alpha_{\max}$ and $\alpha_{\min}$ are the predetermined maximum and minimum adaptation rates permitted, and κ is a scaling parameter that controls the sensitivity. Then we obtain the target adaptation rate as $\alpha_{\text{target}} = \alpha_{\min} \cdot \lambda_t$, and we update the rate via

$$\alpha_{t+1} = \alpha_t + \eta\left(\alpha_{\text{target}} - \alpha_t\right). \tag{1}$$

where η is the exponential smoothing coefficient.

Warm-Up and Momentum-Based Adaptation Rate Adjustment. In the early stages of adaptation when only few samples have been observed, our memory buffers provide insufficient data to compute a reliable metrics. To avoid unreliable early adaptation, we apply a warm-up factor $\gamma_t = \min\left(1, \frac{n_t}{\alpha_{\text{warm}} \cdot H}\right)$ where α_{warm} is a tunable warm-up factor and H is the prediction length. After that, we update the target adaptation rate as $\alpha_{\text{target}} = \alpha_{\text{base}} \cdot \left[1 + \gamma_t(\lambda_t - 1)\right]$.

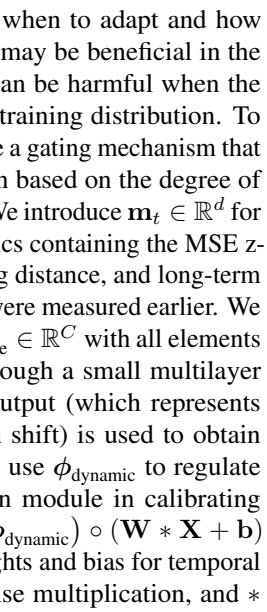


Figure 3. Adaptation rate adjustment with and without warmup.

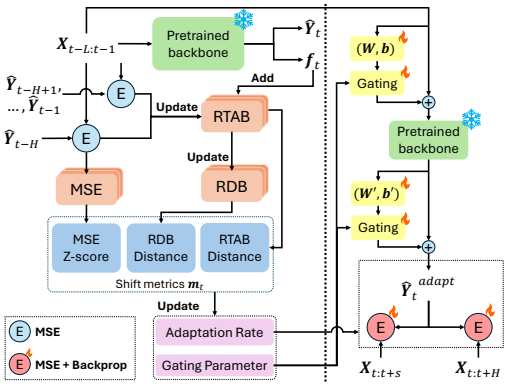


Figure 4. An overview of DynaTTA.

Overall Flow. Figure 4 shows a holistic overview of the full DynaTTA framework. At each time-step, we update the three memory buffers (RTAB, RDB, and MSE buffer), and use these to compute $\mathbf{m}_t$ which is then used to adjust the adaptation rate and the gating parameter. Following (Kim et al., 2025), we keep the backbone frozen at all times, and perform adaptation using two lightweight adapters: one for normalization of the input to the frozen model, and the other for denormalization of its output. Each adapter contains a set of learnable parameters ($\mathbf{W}$, $\mathbf{b}$). Our adjustable adaptation rates are used to apply relevant updates to these parameters, and the gating parameters are used to control their influence in the overall output.

3. TTFBench

To robustly evaluate TSF-TTA methods, we develop a new benchmark TTFBench that applies diverse perturbations into the test split of the standard datasets. TTFBench's dataset generation has three key stages: (1) profiling the original time-series signals where we characterize each channel by attributes such as flatness, trend, seasonality, outliers, regime shifts, and drift, (2) sampling shift parameters, based on polynomial trends, sinusoidal seasonalities, regime changes, and Gaussian noise, based on both global and local (channel-wise) statistics and profile analysis, and (3) generating perturbed test splits by dynamically injecting different perturbations with varying intensities. To preserve cross-channel correlations, a global signal $g(t)$ is added per channel, signed according to its average Pearson correlation with other channels. Finally, we generate a diverse test suite for each dataset by performing the sampling and synthetic generation process across $N = 1000$ perturbation variants. This sampling strategy ensures a broad coverage of diverse combinations of shifts and their intensities, while maintaining internal consistency within each variant. Additional details regarding TTFBench is provided in Appendix A.3.

4. Experiments

Datasets. We evaluate on five standard multivariate TSF datasets: **ETTh1**, **ETTh2**, **ETTm1** (Zhou et al., 2021),

Table 1. Forecasting results on standard time-series forecasting benchmarks.

Dataset	H	iTransformer			PatchTST			DLinear			FreTS			MICN		
		Base	TAFAS	Ours	Base	TAFAS	Ours	Base	TAFAS	Ours	Base	TAFAS	Ours	Base	TAFAS	Ours
ETTh1	96	0.444	0.438	**0.429**	0.436	0.429	**0.410**	0.451	0.442	**0.432**	0.441	0.437	**0.419**	0.455	0.446	**0.437**
	192	0.503	0.492	**0.485**	0.492	0.481	**0.469**	0.504	0.493	**0.469**	0.498	0.491	**0.486**	0.513	0.501	**0.491**
	336	0.562	0.554	**0.550**	0.539	0.529	**0.516**	0.551	0.541	**0.540**	0.563	0.555	**0.549**	0.574	0.561	**0.552**
	720	0.786	**0.704**	0.751	0.713	**0.690**	0.695	0.700	0.669	**0.658**	0.715	0.684	**0.659**	0.735	0.702	**0.702**
ETTh2	96	0.241	0.239	**0.234**	0.233	0.232	**0.225**	0.229	0.227	**0.219**	0.234	0.232	**0.225**	0.234	0.231	**0.226**
	192	0.291	0.287	**0.269**	0.282	0.277	**0.269**	0.283	0.281	**0.275**	0.286	0.282	**0.275**	0.285	0.282	**0.276**
	336	0.334	0.326	**0.319**	0.329	0.320	**0.312**	0.324	0.322	**0.314**	0.326	0.320	**0.316**	0.331	0.325	**0.320**
	720	0.416	0.392	**0.389**	0.416	**0.395**	**0.395**	0.414	0.392	**0.389**	0.420	0.395	**0.387**	0.415	0.397	**0.389**
ETTm1	96	0.389	0.369	**0.360**	0.389	0.379	**0.369**	0.371	0.356	**0.352**	0.367	0.355	**0.349**	0.398	0.376	**0.370**
	192	0.448	0.431	**0.419**	0.440	0.434	**0.420**	0.443	0.419	**0.415**	0.429	0.419	**0.407**	0.448	0.429	**0.423**
	336	0.520	0.497	**0.485**	0.500	0.487	**0.478**	0.518	0.481	**0.477**	0.494	0.478	**0.469**	0.524	0.493	**0.488**
	720	0.592	0.569	**0.557**	0.562	0.546	**0.541**	0.593	0.550	**0.541**	0.560	0.538	**0.529**	0.602	0.568	**0.559**
Weather	96	0.181	0.173	**0.162**	0.176	0.172	**0.163**	0.195	0.183	**0.179**	0.181	0.173	**0.167**	0.178	0.176	**0.168**
	192	0.227	0.218	**0.209**	0.221	0.214	**0.206**	0.240	0.223	**0.214**	0.225	0.212	**0.206**	0.224	0.223	**0.219**
	336	0.284	0.265	**0.259**	0.276	0.268	**0.256**	0.292	0.277	**0.268**	0.280	0.267	**0.254**	0.280	0.274	**0.266**
	720	0.360	0.343	**0.334**	0.352	0.334	**0.325**	0.364	0.345	**0.339**	0.357	0.339	**0.328**	0.350	0.352	**0.345**
Exchange	96	0.089	0.084	**0.075**	0.084	0.083	**0.078**	0.080	0.081	**0.079**	0.084	0.081	**0.079**	0.082	0.080	**0.078**
	192	0.175	0.168	**0.163**	0.179	0.172	**0.167**	0.171	0.163	**0.158**	0.175	0.166	**0.162**	0.183	0.172	**0.168**
	336	0.330	0.282	**0.279**	0.352	**0.289**	0.292	0.321	0.282	**0.280**	0.326	0.285	**0.280**	0.343	0.297	**0.289**
	720	0.844	0.775	**0.758**	0.845	0.845	**0.839**	0.837	0.712	**0.701**	0.840	0.775	**0.769**	1.278	0.950	**0.941**

Weather (for Biogeochemistry), and **Exchange** (Lai et al., 2018). We also evaluate on their perturbed variants generated via **TTFBench**, which introduces controlled test-time distribution shifts. Training data remains unmodified to ensure fair comparisons.

Pretrained Backbones. We test DynaTTA across five diverse forecasting models: (1) **PatchTST** (Nie et al., 2023), (2) **iTransformer** (Liu et al., 2023), (3) **DLinear** (Zeng et al., 2023), (4) **FreTS** (Yi et al., 2023), and (5) **MICN** (Wang et al., 2023b). DynaTTA is model-agnostic and compatible even with backbones lacking internal embeddings (e.g., DLinear), where it operates in a reduced mode using only prediction errors (MSE) for estimating shift intensity.

Baselines. We compare our proposed method, DynaTTA, against two key baselines for each backbone: the unadapted pretrained model (**Base**) and **TAFAS** (Kim et al., 2025) as the only other work on TSF-TTA.

5. Results.

Forecasting Results We evaluate DynaTTA with various backbones on two settings: (1) standard TSF benchmark datasets, and (2) TTFBench. On standard benchmarks (see Table 1), DynaTTA consistently outperforms both baselines across all backbones and datasets, with up to 7.21% gains over iTransformer, and 6.1% gains over TAFAS. On TTFBench (see Table 4), we observe a notable increase in MSE for both the baselines and DynaTTA, due to the greater difficulty posed by diverse test samples. Despite this, DynaTTA maintains a consistent performance improvement over both baselines across all backbones with up to 8.41% gains over PatchTST, and 4.39% gains over TAFAS.

Ablation Study. We perform ablations on: ETTh1, ETTh2, and ETTm1, for two prediction horizons: 96 and 720 using iTransformer as the backbone. The results (MSE values) are shown in Table 2. The table is split into two parts:

Table 2. Ablation results.

Method	ETTh1	ETTh2	ETTm1
w/o SCG	0.436	0.245	0.368
w/o DAR	0.433	0.238	0.366
w/o WU	0.513	0.267	0.371
w/o MSE-score	0.441	0.251	0.368
w/o RTAB	0.431	0.236	0.363
w/o RBD	0.447	0.257	0.369
DynaTTA	0.429	0.234	0.360

Table 4. Forecasting results on TTFBench.

Dataset	H	iTransformer			PatchTST			DLinear			FreTS			MICN		
		Base	TAFAS	Ours	Base	TAFAS	Ours	Base	TAFAS	Ours	Base	TAFAS	Ours	Base	TAFAS	Ours
ETTh1	96	0.4687	0.4673	**0.4644**	1.249	0.992	**0.947**	0.486	0.485	**0.478**	0.463	0.462	**0.452**	1.182	1.042	**1.029**
	192	0.541	0.539	**0.528**	1.320	1.020	**0.994**	0.566	0.567	**0.557**	0.547	0.549	**0.541**	1.290	1.250	**1.100**
	336	0.655	0.677	**0.645**	1.337	1.012	**0.1004**	0.669	0.654	**0.652**	0.658	0.683	**0.642**	1.348	1.346	**1.339**
	720	**0.972**	1.035	0.989	1.461	1.235	**1.105**	0.992	1.002	**0.989**	**1.001**	1.075	1.034	1.496	1.501	**1.494**
ETTh2	96	0.272	0.273	**0.264**	0.366	0.364	**0.359**	0.253	0.250	**0.243**	0.258	0.259	**0.238**	0.413	0.408	**0.397**
	192	0.338	0.340	**0.328**	0.392	0.387	**0.378**	0.329	0.325	**0.315**	0.328	0.33	**0.326**	0.459	0.462	**0.457**
	336	0.407	0.408	**0.397**	0.495	0.497	**0.485**	**0.395**	0.405	0.398	0.400	0.378	**0.356**	0.628	0.624	**0.619**
	720	0.577	0.621	**0.557**	0.515	0.498	**0.478**	0.565	0.564	**0.557**	**0.534**	0.604	0.576	0.722	0.720	**0.719**
ETTm1	96	0.412	0.410	**0.396**	0.476	0.467	**0.447**	0.415	0.420	**0.410**	0.452	0.448	**0.436**	0.461	0.451	**0.448**
	192	0.489	0.488	**0.476**	0.488	0.475	**0.469**	0.491	0.489	**0.489**	0.512	0.506	**0.497**	0.512	**0.498**	0.501
	336	0.599	0.601	**0.587**	0.598	0.594	**0.590**	0.589	0.582	**0.575**	0.561	0.556	**0.549**	0.597	0.578	**0.569**
	720	0.672	0.656	**0.648**	0.623	0.612	**0.609**	0.678	0.672	**0.656**	0.623	0.609	**0.601**	0.682	0.612	**0.605**
Weather	96	0.241	0.233	**0.221**	0.340	0.333	**0.331**	0.263	0.252	**0.247**	0.256	0.254	**0.247**	0.245	0.242	**0.239**
	192	0.310	0.293	**0.278**	0.398	0.387	**0.376**	0.396	0.384	**0.382**	0.289	0.278	**0.273**	0.278	0.276	**0.270**
	336	0.399	0.375	**0.368**	0.460	0.446	**0.434**	0.387	0.367	**0.358**	0.366	0.354	**0.346**	0.310	0.307	**0.299**
	720	0.487	0.475	**0.457**	0.559	0.543	**0.539**	0.493	0.483	**0.478**	0.417	0.409	**0.401**	0.435	0.437	**0.434**
Exchange	96	0.095	0.095	**0.087**	0.239	0.232	**0.224**	0.149	0.144	**0.136**	0.126	0.123	**0.114**	**0.216**	0.217	**0.216**
	192	0.286	0.281	**0.275**	0.395	0.389	**0.378**	0.297	0.289	**0.283**	0.274	0.270	**0.259**	0.399	0.401	**0.398**
	336	0.476	0.471	**0.454**	0.691	**0.688**	**0.688**	0.536	0.537	**0.529**	0.518	0.52	**0.515**	0.637	0.640	**0.631**
	720	1.266	1.264	**1.249**	1.408	1.404	**1.397**	1.338	1.334	**1.325**	1.246	1.245	**1.239**	1.418	1.413	**1.411**

(1) the first three rows evaluate the effect of each adaptation mechanism: (1) shift-conditioned gating (SCG), (2) dynamic adaptation rate adjustment (DARA), and (3) warm-up (WU). And the next three rows assess the contribution of each shift metric: (1) MSE z-score, (2) RTAB embedding distance, and (3) RDB embedding distance. The last row simply shows the result of DynaTTA, with all components enabled. It can be observed that removing each component of DynaTTA results in a drop in performance of the model. Based on the results, we can also conclude that warm-up based adaptation rate adjustment and RDB are the most important components in DynaTTA.

Sensitivity to random seed. To evaluate the impact of random seeds, we perform 10 separate trials with different seed values, and present the standard deviations in Table 3. We observe that DynaTTA has negligible impact on the sensitivity of backbone to random seed values.

Table 3. Sensitivity of DynaTTA to random seeds.

Method	ETTh1	ETTh2	ETTm1
iTransformer	0.0004	0.0007	0.0003
PatchTST	0.0006	0.0004	0.0001
DLinear	0.0000	0.0001	0.0000
FreTS	0.0002	0.0001	0.0005
MICN	0.0006	0.0005	0.0002

6. Conclusion

Summary. We propose DynaTTA, a dynamic test-time adaptation framework for time-series forecasting that adjusts both adaptation rate and gating based on estimated distribution shifts. We also propose TTFBench, a first-of-its-kind benchmark for evaluating TSF-TTA frameworks under diverse distribution shifts. Through extensive experiments on TSF, across five architectures and multiple datasets, we show that DynaTTA consistently outperformed all baselines and sees gains up to 8.41%.

Broader Impact. Time-series data is ubiquitous, impacting our lives daily, in the form of weather patterns, financial market trends, etc. DynaTTA enables TSF models to generalize to previously unseen distributions, which is a common challenge in real-world scenarios. Since no retraining is required for our dynamic adaptation approach, DynaTTA helps pretrained models stay relevant and up to date, preventing model obsolescence and reducing retraining costs.

References

Abid, A. and Zou, J. Y. Learning a warping distance from unlabeled time series using sequence autoencoders. *Advances in Neural Information Processing Systems*, 31, 2018.

Bai, L., Yao, L., Li, C., Wang, X., and Wang, C. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems*, 33:17804–17815, 2020.

Bartler, A., Bühler, A., Wiewel, F., Döbler, M., and Yang, B. Mt3: Meta test-time training for self-supervised test-time adaption. In *International Conference on Artificial Intelligence and Statistics*, pp. 3080–3090. PMLR, 2022.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Bu, S.-J. and Cho, S.-B. Time series forecasting with multi-headed attention-based deep learning for residential energy consumption. *Energies*, 13(18):4722, 2020.

Chatfield, C. *The analysis of Time Series: An introduction*. Chapman & Hall, 1996.

Chen, D., Wang, D., Darrell, T., and Ebrahimi, S. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 295–305, 2022.

Cheng, D., Yang, F., Xiang, S., and Liu, J. Financial time series forecasting with multi-modality graph neural network. *Pattern Recognition*, 121:108218, 2022.

Christou, P., Chen, S., Chen, X., and Dube, P. Test time learning for time series forecasting. *arXiv preprint arXiv:2409.14012*, 2024.

Cirstea, R.-G., Yang, B., Guo, C., Kieu, T., and Pan, S. Towards spatio-temporal aware traffic time series forecasting. In *IEEE International Conference on Data Engineering*, pp. 2900–2913, 2022.

Dubey, A., Ramanathan, V., Pentland, A., and Mahajan, D. Adaptive methods for real-world domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14340–14349, 2021.

Ekambaram, V., Jati, A., Nguyen, N., Sinthong, P., and Kalagnanam, J. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 459–469, 2023.

Findley, D. F., Lytras, D. P., and McElroy, T. S. Detecting seasonality in seasonally adjusted monthly time series. *Statistics*, 3, 2017.

Fleuret, F. et al. Test time adaptation through perturbation robustness. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

for Biogeochemistry, M. P. I. Jena climate and weather data. https://www.bgc-jena.mpg.de/wetter/.

Galstyan, A. and Cohen, P. R. Empirical comparison of "hard" and "soft" label propagation for relational classification. In *International Conference on Inductive Logic Programming*, pp. 98–111. Springer, 2007.

Gandelsman, Y., Sun, Y., Chen, X., and Efros, A. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022.

Gao, J., Zhang, J., Liu, X., Darrell, T., Shelhamer, E., and Wang, D. Back to the source: Diffusion-driven adaptation to test-time corruption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11786–11796, 2023.

Gardner Jr, E. S. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1):1–28, 1985.

Gong, P., Ragab, M., Wu, M., Chen, Z., Su, Y., and Li, X. Augmented contrastive clustering with uncertainty-aware prototyping for time series test time adaptation. In *31st SIGKDD Conference on Knowledge Discovery and Data Mining - Research Track*, 2025.

Gong, T., Jeong, J., Kim, T., Kim, Y., Shin, J., and Lee, S.-J. Note: Robust continual test-time adaptation against temporal correlation. *Advances in Neural Information Processing Systems*, 35:27253–27266, 2022.

Hakim, G. A. V., Osowiechi, D., Noori, M., Cheraghalikhani, M., Bahri, A., Ben Ayed, I., and Desrosiers, C. Clust3: Information invariant test-time training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6136–6145, 2023.

Harvey, A. C. Forecasting, structural time series models and the kalman filter. 1990.

He, S., Li, X., DelSole, T., Ravikumar, P., and Banerjee, A. Sub-seasonal climate forecasting via machine learning: Challenges, analysis, and advances. In *AAAI Conference on Artificial Intelligence*, volume 35, pp. 169–177, 2021.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Hess, A., Iyer, H., and Malm, W. Linear trend analysis: a comparison of methods. *Atmospheric Environment*, 35 (30):5211–5222, 2001.

Hoaglin, D. C. and Iglewicz, B. Fine-tuning some resistant rules for outlier labeling. *Journal of the American statistical Association*, 82(400):1147–1149, 1987.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Jung, S., Lee, J., Kim, N., Shaban, A., Boots, B., and Choo, J. Cafa: Class-aware feature alignment for test-time adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19060–19071, 2023.

Kaku, A., Mohan, S., Parnandi, A., Schambra, H., and Fernandez-Granda, C. Be like water: Robustness to extraneous variables via adaptive feature normalization. *arXiv preprint arXiv:2002.04019*, 2020.

Karmanov, A., Guan, D., Lu, S., El Saddik, A., and Xing, E. Efficient test-time adaptation of vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14162–14171, 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.

Kim, T.-H. and White, H. On more robust estimation of skewness and kurtosis. *Finance Research Letters*, 1(1): 56–73, 2004.

Kovács, S., Bühlmann, P., Li, H., and Munk, A. Seeded binary segmentation: a general methodology for fast and optimal changepoint detection. *Biometrika*, 110(1):249–256, 2023.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long-and short-term temporal patterns with deep neural networks. In *International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, 2018.

Lee, D.-H. et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, pp. 896. Atlanta, 2013.

Lee, J., Jung, D., Lee, S., Park, J., Shin, J., Hwang, U., and Yoon, S. Entropy is not enough for test-time adaptation: From the perspective of disentangled factors. In *International Conference on Learning Representations*, 2024.

Li, H., Lu, H., and Chen, Y.-C. Bi-tta: Bidirectional test-time adapter for remote physiological measurement. In *European Conference on Computer Vision*, pp. 356–374. Springer, 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.

Li, W. and Law, K. E. Deep learning models for time series forecasting: a review. *IEEE Access*, 2024.

Li, Y., Wang, N., Shi, J., Liu, J., and Hou, X. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

Li, Y., Hao, M., Di, Z., Gundavarapu, N. B., and Wang, X. Test-time personalization with a transformer for human pose estimation. *Advances in Neural Information Processing Systems*, 34:2583–2597, 2021.

Li, Y., Xu, X., Su, Y., and Jia, K. On the robustness of open-world test-time training: Self-training with dynamic prototype expansion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11836–11846, 2023.

Lin, H., Gao, Z., Xu, Y., Wu, L., Li, L., and Li, S. Z. Conditional local convolution for spatio-temporal meteorological forecasting. In *AAAI Conference on Artificial Intelligence*, volume 36, pp. 7470–7478, 2022.

Lin, W., Mirza, M. J., Kozinski, M., Possegger, H., Kuehne, H., and Bischof, H. Video test-time adaptation for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22952–22961, 2023.

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. In *The Twelfth International Conference on Learning Representations*, 2023.

Ma, X., Zhang, J., Guo, S., and Xu, W. Swapprompt: Test-time prompt adaptation for vision-language models. *Advances in Neural Information Processing Systems*, 36: 65252–65264, 2023.

Mirza, M. J., Shin, I., Lin, W., Schriebl, A., Sun, K., Choe, J., Kozinski, M., Possegger, H., Kweon, I. S., Yoon, K.-J., et al. Mate: Masked autoencoders are online 3d test-time learners. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16709–16718, 2023a.

Mirza, M. J., Soneira, P. J., Lin, W., Kozinski, M., Possegger, H., and Bischof, H. Actmad: Activation matching to align distributions for test-time-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24152–24161, 2023b.

Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B., and Snoek, J. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*, 2020.

Nie, W., Guo, B., Huang, Y., Xiao, C., Vahdat, A., and Anandkumar, A. Diffusion models for adversarial purification. In *International Conference on Machine Learning*, pp. 16805–16827. PMLR, 2022.

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

Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., and Tan, M. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pp. 16888–16905. PMLR, 2022.

Osowiechi, D., Hakim, G. A. V., Noori, M., Cheraghalikhani, M., Bahri, A., Yazdanpanah, M., Ben Ayed, I., and Desrosiers, C. Nc-ttt: A noise constrastive approach for test-time training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6078–6086, 2024.

Ouyang, Z., Ravier, P., and Jabloun, M. Stl decomposition of time series can benefit forecasting done by statistical methods but not by machine learning ones. *Engineering Proceedings*, 5(1):42, 2021.

Pandey, P., Raman, M., Varambally, S., and Ap, P. Generalization on unseen domains via inference-time label-preserving target projections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12924–12933, 2021.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1406–1415, 2019.

Prabhudesai, M., Ke, T.-W., Li, A., Pathak, D., and Fragkiadaki, K. Diffusion-tta: Test-time adaptation of discriminative models via generative feedback. *Advances in Neural Information Processing Systems*, 36:17567–17583, 2023.

Ragab, M., Eldele, E., Tan, W. L., Foo, C.-S., Chen, Z., Wu, M., Kwoh, C.-K., and Li, X. Adatime: A benchmarking suite for domain adaptation on time series data. *ACM Transactions on Knowledge Discovery from Data*, 17(8): 1–18, 2023.

Sain, A., Bhunia, A. K., Potlapalli, V., Chowdhury, P. N., Xiang, T., and Song, Y.-Z. Sketch3t: Test-time training for zero-shot sbir. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7462–7471, 2022.

Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., and Bethge, M. Improving robustness against common corruptions by covariate shift adaptation. *Advances in neural information processing systems*, 33: 11539–11551, 2020.

Sinha, S., Gehler, P., Locatello, F., and Schiele, B. Test: Test-time self-training under distribution shift. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2759–2769, 2023.

Su, Y., Xu, X., and Jia, K. Towards real-world test-time adaptation: Tri-net self-training with balanced normalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 15126–15135, 2024.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., and Hardt, M. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

Tomar, D., Vray, G., Bozorgtabar, B., and Thiran, J.-P. Tesla: Test-time self-learning with automatic adversarial augmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20341–20350, 2023.

Varsavsky, T., Orbes-Arteaga, M., Sudre, C. H., Graham, M. S., Nachev, P., and Cardoso, M. J. Test-time unsupervised domain adaptation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part I 23*, pp. 428–436. Springer, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 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*, 2023a.

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 *International Conference on Learning Representations*, 2023b.

Wang, J., Wang, H., Deng, J., Wu, W., and Zhang, D. Efficientclip: Efficient cross-modal pre-training by ensemble confident learning and language modeling. *arXiv preprint arXiv:2109.04699*, 2021.

Wang, W., Zhong, Z., Wang, W., Chen, X., Ling, C., Wang, B., and Sebe, N. Dynamically instance-guided adaptation: A backward-free approach for test-time domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24090–24099, 2023c.

Wang, Z., Xu, X., Trajcevski, G., Zhang, K., Zhong, T., and Zhou, F. Pref: Probabilistic electricity forecasting via copula-augmented state space model. In *AAAI Conference on Artificial Intelligence*, volume 36, pp. 12200–12207, 2022.

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.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

Xiao, Z. and Snoek, C. G. Beyond model adaptation at test time: A survey. *arXiv preprint arXiv:2411.03687*, 2024.

Xiao, Z., Zhen, X., Liao, S., and Snoek, C. G. Energy-based test sample adaptation for domain generalization. *arXiv preprint arXiv:2302.11215*, 2023.

Yang, T., Zhou, S., Wang, Y., Lu, Y., and Zheng, N. Test-time batch normalization. *arXiv preprint arXiv:2205.10210*, 2022.

Yaro, A. S., Maly, F., and Prazak, P. Outlier detection in time-series receive signal strength observation using z-score method with s n scale estimator for indoor localization. *Applied Sciences*, 13(6):3900, 2023.

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 *AAAI Conference on Artificial Intelligence*, volume 37, pp. 11121–11128, 2023.

Zhang, T., Wang, J., Guo, H., Dai, T., Chen, B., and Xia, S.-T. Boostadapter: Improving test-time adaptation via regional bootstrapping. *arXiv e-prints*, pp. arXiv–2410, 2024.

Zhang, Y. and Yan, J. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh 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 *AAAI Conference on Artificial Intelligence*, volume 35, pp. 11106–11115, 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 *International conference on machine learning*, pp. 27268–27286. PMLR, 2022.

Zhou, Y., Ren, J., Li, F., Zabih, R., and Lim, S. N. Test-time distribution normalization for contrastively learned visual-language models. *Advances in Neural Information Processing Systems*, 36:47105–47123, 2023.

A. Appendix

In this Appendix, we first provide a detailed discussion of the related literature on time-series forecasting (TSF) and test-time adaptation (TTA) in Section A.1. We then present visualizations and analysis of the dynamic adaptation behavior of our proposed method, DynaTTA, including adaptation rate schedules and gating mechanisms across multiple datasets. Finally, we provide a detailed description of the construction methodology for TTFBench in Section A.3.

A.1. Related Work

Time-series forecasting. TSF is a well-researched area that plays an important role in a variety of real-world applications, such as forecasting weather (He et al., 2021; Lin et al., 2022), energy consumption (Zhou et al., 2021; Bu & Cho, 2020; Wang et al., 2022), traffic (Bai et al., 2020; Cirstea et al., 2022), and financial markets (Cheng et al., 2022). Traditional approaches, such as autoregressive models (Box et al., 2015), structural time-series models (Harvey, 1990), exponential smoothing (Gardner Jr, 1985), etc., focus on parametric approaches that are domain-specific. More recently, deep learning has become a prominent paradigm in TSF (Li & Law, 2024), overcoming limitations of traditional parametric models by learning temporal dependencies directly from data. A wide range of architectures have been proposed for TSF, including Transformers (Zhou et al., 2021; Wu et al., 2021; Li et al., 2019; Liu et al., 2022; Zhang & Yan, 2023), multi-layer perceptrons (MLP) (Yi et al., 2023; Wang et al., 2023a; Ekambaram et al., 2023), and linear solutions (Zeng et al., 2023), each offering unique trade-offs in modeling capacity and scalability. Transformer-based architectures, originally introduced for natural language processing tasks (Vaswani et al., 2017), have gained significant traction in TSF due to their ability to model long-range dependencies through self-attention mechanisms. However, they struggle with long input sequences, and hence from high computational cost and inefficiency. Informer (Zhou et al., 2021) introduced a sparse attention mechanism and a self-attention distillation strategy to improve scalability and efficiency in long-sequence forecasting. Autoformer (Wu et al., 2021) incorporated a series decomposition block and an autocorrelation mechanism, allowing the model to learn trend and seasonal components explicitly. FEDformer (Zhou et al., 2022) and Crossformer (Zhang & Yan, 2023) integrated frequency-domain operations and cross-dimension dependencies to improve prediction accuracy. Despite these advances, many of these models rely on full-sequence encoding, which may be inefficient or brittle under noisy or multivariate conditions. PatchTST (Nie et al., 2023) considers TSF as a patch-level sequence modeling task using non-overlapping segments to efficiently capture long-range dependencies while avoiding the overhead of full-sequence attention. iTransformer (Liu et al., 2023) revisits the standard Transformer design by inverting the attention dimensions and applying self-attention across time points within each individual variate rather than across variates at each timestamp. This helps in improving scalability to longer input sequences without architectural modification. Recently, more works have been considering whether transformer-based TSF models are effective: (Zeng et al., 2023) DLinear demonstrated that a simple linear model, built on the premise of decomposing time-series into trend and seasonal components, can outperform or match the performance of more elaborate Transformer variants on several benchmarks. On the other hand, MLP-based models like (Wang et al., 2023a; Ekambaram et al., 2023) have re-emerged as strong alternatives, especially when equipped with architectural modifications (such as channel-mixing) that better capture temporal and multivariate dependencies.

Test-time adaptation. TTA is a learning paradigm designed to improve the performance of machine learning models under distribution shifts that occur during testing. Unlike domain adaptation, which requires access to target data during training, or domain generalization, which assumes no knowledge of the target distribution at test time, TTA adapts a model using only unlabeled test data during inference (Xiao & Snoek, 2024). This setting is practical for deployment scenarios where retraining is infeasible due to privacy constraints, data availability, or storage limitations. TTA techniques are categorized based on what component of the system they adapt, as follows: **(1) model parameters adaptation**, which is performed using auxiliary self-supervision (Sun et al., 2020; Varsavsky et al., 2020; Gandelsman et al., 2022; Prabhudesai et al., 2023; Osowiechi et al., 2024; Mirza et al., 2023a; Sain et al., 2022; Hakim et al., 2023; Li et al., 2021; Mirza et al., 2023b; Li et al., 2023; Bartler et al., 2022), entropy minimization (Wang et al., 2020; Lee et al., 2024; Abid & Zou, 2018; Niu et al., 2022; Gong et al., 2022), pseudo-labeling (Galstyan & Cohen, 2007; Lee et al., 2013; Wang et al., 2021; Sinha et al., 2023; Tomar et al., 2023), or feature alignment (Fleuret et al., 2021; Jung et al., 2023; Li et al., 2024; Lin et al., 2023); **(2) inference adaptation** (Zhang et al., 2024; Dubey et al., 2021; Ioffe & Szegedy, 2015; Zhou et al., 2023; Wang et al., 2023c) where model parameters are estimated for each test batch or sample in a single forward pass; **(3) normalization adaptation**, (Su et al., 2024; Yang et al., 2022; Li et al., 2016; Nado et al., 2020; Kaku et al., 2020; Schneider et al., 2020) which recalibrates batch normalization statistics at test time to better match the test data distribution; and **(4) sample adaptation** (Gao et al., 2023; Nie et al., 2022; Pandey et al., 2021; Xiao et al., 2023), which transforms target data into the source domain using generative models such as diffusion or energy-based methods.

Most TTA research has focused on classification tasks, primarily because methods like entropy minimization, pseudo-labeling, and feature alignment are naturally well-suited to classification objectives. These techniques rely on prediction confidence, label distribution assumptions, or feature alignment strategies that are well-defined and easily evaluated in classification settings, and can be extended to time-series classification methods. For example, building upon the concept of entropy minimization, (Gong et al., 2025) incorporates uncertainty-aware prototypes and entropy comparison for enhancing reliability of psuedo-labels for time-series classification. However, these methods are not directly applicable to sequential data like TSF, due to it's regression-based nature, temporal dependencies, evolving dynamics, and the absence of discrete decision boundaries. The need to model temporal dependencies, account for different types and intensities of drift, and operate without discrete label boundaries, renders many classification-oriented TTA methods impractical in such settings. One notable early effort to bridge this gap is the recent work TAFAS (Kim et al., 2025) which introduces a model-agnostic TTA framework specifically for TSF, using partially-observed ground truth and a gated calibration module to maintain semantic alignment. While promising, such methods remain relatively non-dynamic to the ever-evolving shifts in distribution. However, TAFAS uses a fixed adaptation rate and a gating mechanism that does not adapt its behavior based on the intensity of distribution shift, limiting its responsiveness to varying test-time conditions. We build upon the the calibration module and adaptation cycle of TAFAS and introduce novel mechanisms that dynamically modulate the adaptation rate and gating in response to changing shift intensities. Finally, unlike the well-established TTA benchmarks for image classification (e.g., CIFAR-C (Hendrycks & Dietterich, 2019), ImageNet-C (Hendrycks & Dietterich, 2019), DomainNet (Peng et al., 2019)) and time-series classification (e.g., ADATIME (Ragab et al., 2023)), TSF lacks standardized benchmarks for evaluating TTA methods, making robust and consistent evaluation an open challenge. To address this, we introduce TTFBench, a benchmark explicitly designed for evaluating TSF-TTA methods under meaningful, varied, and controlled distribution shifts.

A.2. Dynamic adaptation rate and gating

DynaTTA adjusts the adaptation rate and the overall gating values based on the estimated distribution shifts metrics ($\mathbf{m}_t$). In this section, we go into detail of the methodology pertaining to DynaTTA.

A.2.1. Estimating Distribution Shifts.

In the absence of access to source data at test time, we estimate the intensity of the distribution shift and the proximity of incoming test data to the original training distribution by tracking the performance metric, particularly MSE of the original model's predictions on the test data. Since the model is originally trained to minimize prediction error on the source distribution (training data), a low test-time MSE strongly suggests that the input data remains close to what the model has already learned, requiring little to no adaptation. In contrast, a rising MSE could be a signal that the model's learned patterns are no longer sufficient, indicating that the input distribution could be drifting away from the training distribution. For a given input window $\mathbf{X}_{t-L_{\text{in}}:t-1}$, its corresponding prediction is $\hat{\mathbf{Y}}_t = f_\theta(\mathbf{X}_{t-L_{\text{in}}:t-1}) = \left[\hat{x}^{(c)}_{t+l-1}\right]_{c,l} \in \mathbb{R}^{C\times H}$ where $\hat{x}^{(c)}_{t+l-1}$ is the prediction for channel c at time $t+l-1$ and $0 < l < H$. Accordingly, the MSE at time t is computed as:

$$\text{MSE}_t^{(H)} = \frac{1}{CH}\sum_{c=1}^{C}\sum_{l=1}^{H}\left(\hat{x}^{(c)}_{t+l-1} - x^{(c)}_{t+l-1}\right)^2. \tag{2}$$

To track this signal over time, we maintain a memory buffer of recent MSE values and compute the z-score of the latest error $z_t = \frac{\text{MSE}_t^{(L)} - \mu_t}{\sigma_t + \epsilon}$, where μ_t and σ_t are the mean and standard deviation of MSEs in the buffer, and ϵ is a small constant for numerical stability. A high z_t indicates a significant degradation in model performance, which we interpret as a strong shift away from the source distribution. However, MSE alone captures only output-level deviations and may not fully reflect deeper changes in the data's underlying temporal patterns. It also becomes available after a certain delay (when the full ground truth is revealed). To supplement this, we also analyze embedding drift, *i.e.*, how much the model's internal representation of the current input deviates from embeddings associated with previously low-error predictions. To do this, we maintain two complementary memory buffers: (1) the Real-Time Adaptation Buffer (RTAB) which stores the most recent embeddings $\mathbf{f}_i$ alongside their corresponding MSE values E_i, computed using the frozen source model once the ground truth becomes available. This buffer captures short-term variations and provides a real-time view of the model performance on the stream of input data; and (2) the Reference Distribution Buffer (RDB) which retains the past embeddings with the lowest MSEs over time, serving as a more stable and long-term representation of the source distribution. Note that for calculating the embeddings for both RTAB and RDB, we directly use the original model without any adaptation applied to it.

At time t, let $\mathcal{B}_t$ be the set of RTAB indices containing stored embeddings $\mathbf{f}_i \in \mathbb{R}^d$ (where d is the dimensionality of the embedding space). When ground truth for a past input becomes available, we update the MSE for its corresponding stored embedding. However, because the full ground truth for future steps is revealed gradually over time, we make use of the MSE of the partially observed ground truth as an intermediate approximation until the full prediction window is observed. For embedding $\mathbf{f}_i \in \mathbb{R}^d$ with $l < H$ observed future steps, the partial MSE E_i is defined as:

$$E_i = \text{MSE}_t^{(l)} = \frac{1}{C \cdot l} \sum_{c=1}^{C} \sum_{j=1}^{l} \left(\hat{x}_{t+j-1}^{(c)} - x_{t+j-1}^{(c)} \right)^2. \tag{3}$$

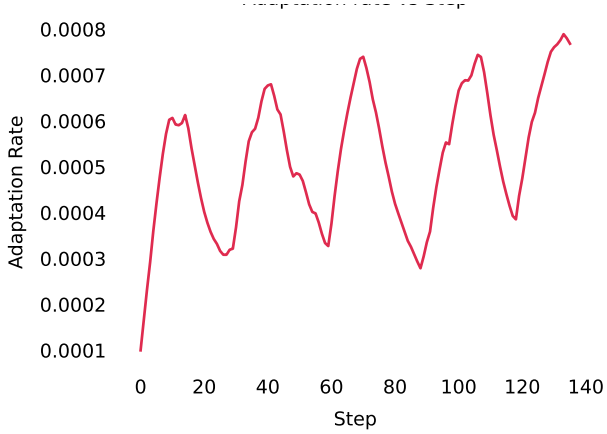
This partial error is used provisionally to update the buffer. Once the full ground truth at time t becomes available, we replace its corresponding partial MSE with $\text{MSE}_t^{(H)}$. This allows us to update the RTAB more promptly and maintain a responsive estimation of distribution shifts. At each step, we update the partial MSE for the most recent embeddings and calculate a weighted average embedding, where each embedding is assigned a weight inversely proportional to its MSE, giving more influence to embeddings that result in more accurate predictions. To account for the uncertainty in partial estimates, we apply discounted weighting for embeddings whose MSEs are still partial. Let $\alpha_i \in (0, 1]$, be a confidence factor such that $\alpha_i = \frac{l}{H}$ for partial MSEs, and $\alpha_i = 1$ for full MSEs. We define the weight for each embedding as:

$$w_i^{(t)} = \frac{\alpha_i \cdot \beta}{\sum_{j \in \mathcal{B}_t} \alpha_j \cdot \beta}, \tag{4}$$

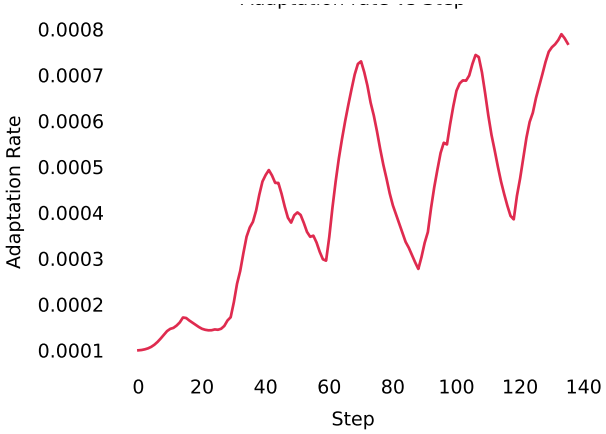
where $\beta = \frac{1}{E_i + \epsilon}$. The resulting weighted average embedding at time t is then $\bar{\mathbf{f}}^{(t)} = \sum_{i \in \mathcal{B}_t} w_i^{(t)} \mathbf{f}_i$. RTAB captures short-term fluctuations in prediction performance, and is inherently sensitive to recent changes in the data stream. To also provide a more stable and reliable representation of the source distribution, we introduce a second memory buffer, the RDB, which is designed to maintain a long-term fixed-size memory of the model's best-performing samples, which are most likely to be close to the source distribution. RDB only stores embeddings for which the full ground truth is available and whose corresponding MSE is among the lowest observed over time. We define a buffer capacity $K \in \mathbb{N}$ as a hyperparameter to limit the number of entries in RDB. Let $\mathcal{R}_t$ denote the set of indices currently stored in RDB at time t, with $|\mathcal{R}_t| \leq K$. At each step, we examine the newly updated full-MSE entries from RTAB. If any new entry $(E_i, \mathbf{f}_i)$ satisfies $E_i < \max_{j \in \mathcal{R}_t} E_j$, it is added to RDB. If the size exceeds K, the entry with the highest E_j is removed. This ensures that RDB always holds the best-performing K samples (in terms of MSE), regardless of whether they are still present in RTAB, ensuring a long-term memory of high-confidence examples. At each step, we compute a weighted average embedding using the same inverse-MSE weighting logic described earlier (without the confidence scaling since there is no partial MSE in RDB).

We have now obtained three distinct but complementary metrics to quantify different aspects of the distribution shift of the test data at each time-step: (1) the MSE z-score, (2) the short-term RTAB embedding distance, and (3) the long-term RDB embedding distance.

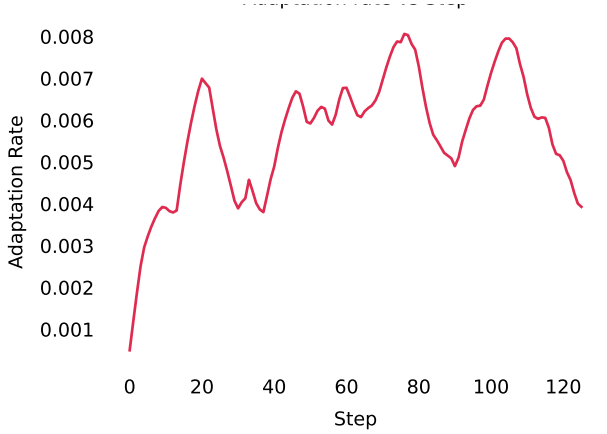
In Figure A1, we show several examples of the dynamic adaptation rate. Each row represents a specific dataset and horizon that the model was adapted on; the left column shows the adaptation with no warmup ($W = 0$) while the right column shows the adaptation with $W = 1$. In Figures A2 and A3, we show how the final gating values for both the input and output adaptation evolve over time for the ETTh2 and ETTm1 datasets respectively at two different prediction horizons.



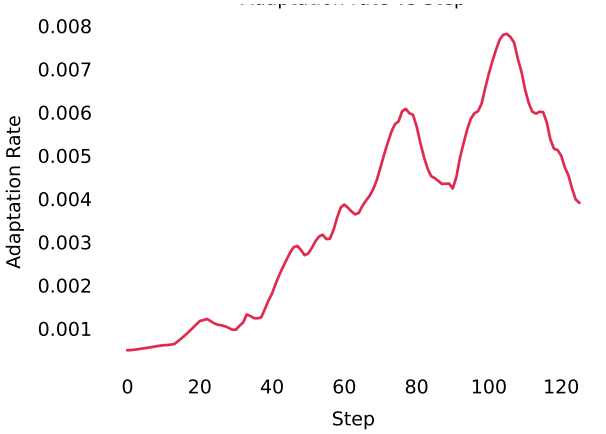
(a) ETTh2 ($H = 96, \alpha_{\text{warm}} = 0$)



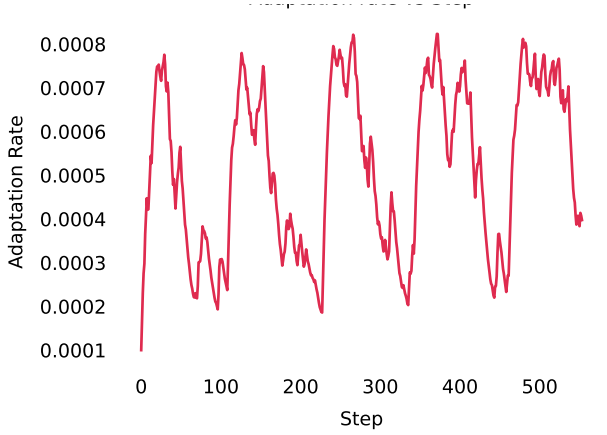
(b) ETTh2 ($H = 96, \alpha_{\text{warm}} = 1$)



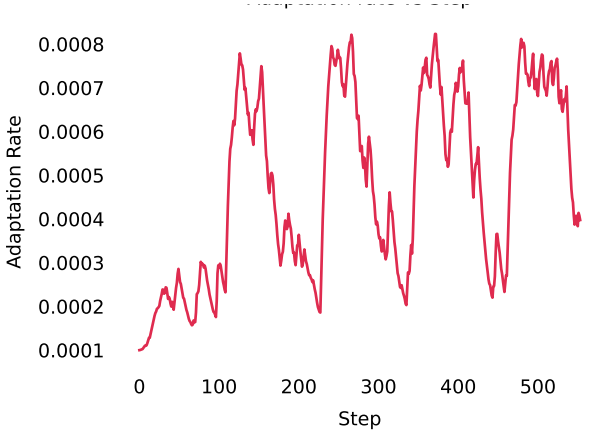
(c) ETTh1 ($H = 336, \alpha_{\text{warm}} = 0$)



(d) ETTh1 ($H = 336, \alpha_{\text{warm}} = 1$)

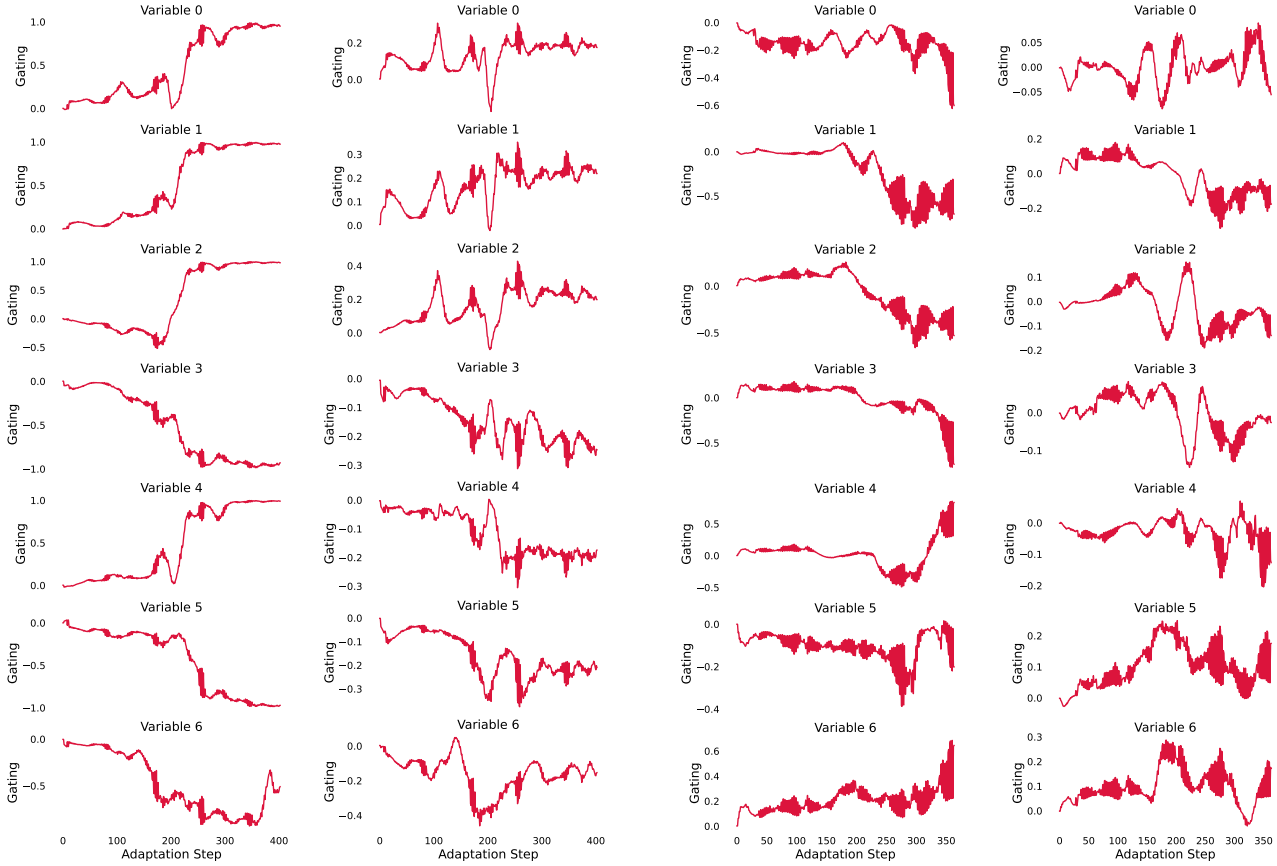


(e) ETTm1 ($H = 96, \alpha_{\text{warm}} = 0$)



(f) ETTm1 ($H = 96, \alpha_{\text{warm}} = 1$)

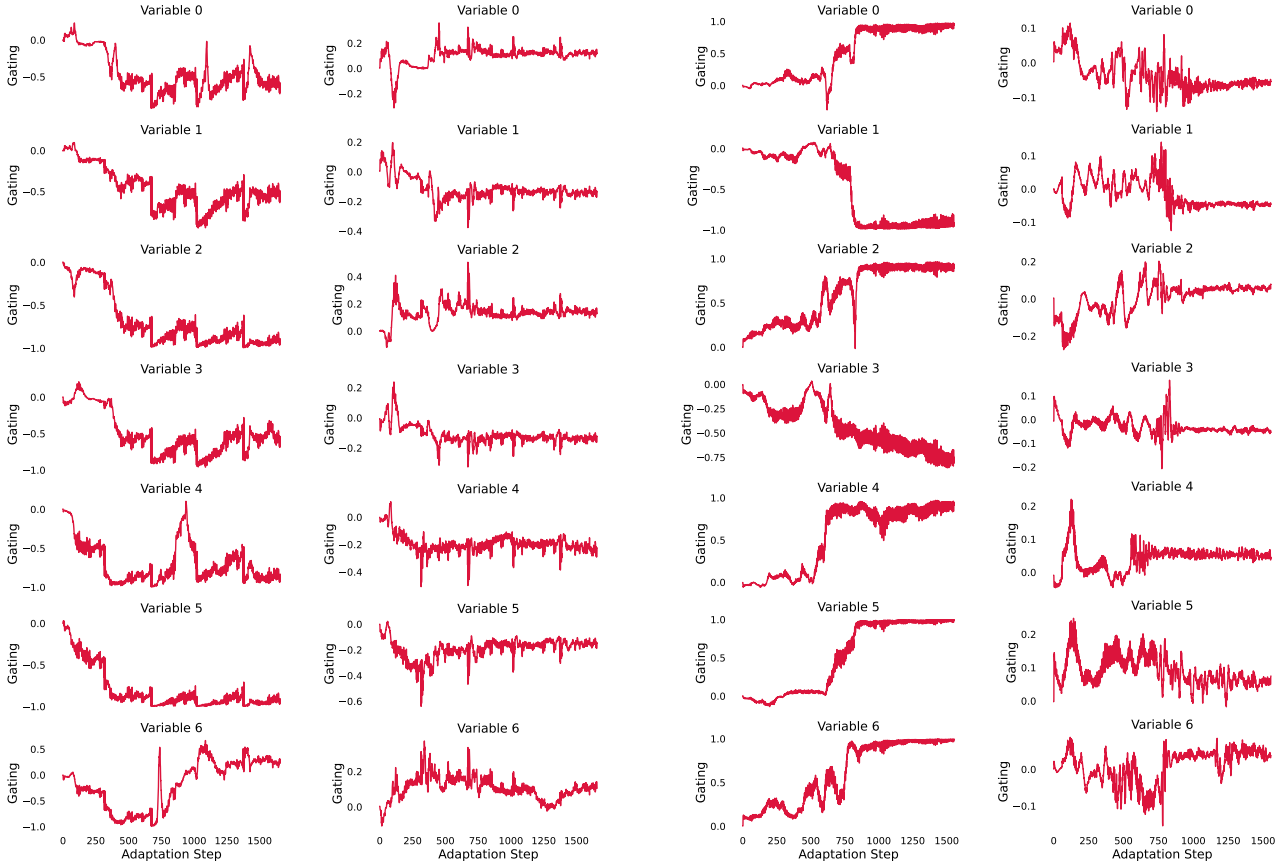
Figure A1. Evolution of adaptation rate with ($\alpha_{\text{warm}} = 1$) and without ($\alpha_{\text{warm}} = 0$) warmup, where α_{warm} is the warm-up factor.



(a) Input adaptation. (b) Output adaptation. (c) Input adaptation. (d) Output adaptation.

$H = 96$ $H = 336$

Figure A2. Evolution of the dynamically adjusted gating values per variable for input and output adaptation across ETTh2 datasets.



(a) Input adaptation. (b) Output adaptation. $H = 96$

(c) Input adaptation. (d) Output adaptation. $H = 720$

Figure A3. Evolution of the dynamically adjusted gating values per variable for input and output adaptation across ETTm1 datasets.

A.3. TTFBench

To systematically generate diverse distribution shifts, we design a synthetic perturbation framework that carefully modulates the structure of multivariate time-series data. This procedure accounts for the intrinsic properties of each signal, leveraging both local channel-specific characteristics and global inter-channel dependencies to produce meaningful distribution shifts. In this section we provide a step-by-step mathematical description of the three steps in TTFBench generation pipeline, namely (1) profiling the original time-series signals, (2) sampling shift parameters based on both global and local (channel-wise) statistics, and (3) generating perturbed test splits.

Let the input multivariate time-series be denoted as $\{x_t^{(c)}\}_{t=1}^{T}$ for channel $c \in \{1, \ldots, C\}$ and time index t. The full sequence is partitioned into training, validation, and test splits, defined by indices T_{train}, T_{val}, and T_{test} respectively, such that $1 < T_{\text{train}} < T_{\text{val}} < T$.

A.3.1. Profiling and Signal Characterization

Each channel c is analyzed individually using its training segment $\{x_t^{(c)}\}_{t=1}^{T_{\text{train}}}$ to extract a profile vector $\mathbf{p}^{(c)}$ encoding its structural characteristics. This vector includes the following scalar attributes: *flatness*, *trend*, *seasonality*, *spikes*, *regime shifts*, *drift*, and a binary indicator of *outlier dominance*. These properties are carefully computed and are normalized to lie within $[0, 1]$ for better control, interpretability, and robustness to outliers.

Flatness. To quantify short-term variability, following (Chatfield, 1996), and (Hoaglin & Iglewicz, 1987), we first compute the standard deviation of first differences $\sigma_{\text{slope}}^{(c)} = \text{std}\left(\Delta x_t^{(c)}\right)$, where $\Delta x_t^{(c)} = x_t^{(c)} - x_{t-1}^{(c)}$. Then we compute the interquartile range (IQR) to standard deviation ratio $r_{\text{IQR}}^{(c)} = \frac{\text{IQR}(x^{(c)})}{\text{std}(x^{(c)})+\epsilon}$. The flatness score is then defined as: $\text{flatness}^{(c)} = \left[1 - \frac{\sigma_{\text{slope}}^{(c)}}{0.2}\right]_+ \cdot \left[\frac{r_{\text{IQR}}^{(c)} - 10}{20}\right]_+$, where $[z]_+ = \max(0, \min(1, z))$ ensures the result is clipped to $[0, 1]$.

Trend. Following (Hess et al., 2001), a linear regression line is fit to the series using least-squares: $x_t^{(c)} \approx \hat{\beta}_0^{(c)} + \hat{\beta}_1^{(c)} t$, and the normalized trend strength is defined as:

$$\text{trend}^{(c)} = \frac{|\hat{\beta}_1^{(c)}|}{\text{std}(x^{(c)}) + \epsilon}, \tag{5}$$

which is capped at 1.0.

Seasonality. The presence of a strong periodic structure is identified using the periodogram (Findley et al., 2017). Let $P(f)$ be the power spectral density of $x^{(c)}$. Define:

$$\text{peak}_P = \max_f P(f), \quad \text{med}_P = \text{median}_f P(f), \quad \text{tot}_P = \sum_f P(f) + \epsilon. \tag{6}$$

Then, the seasonality indicator is defined as:

$$\text{seasonality}^{(c)} = \begin{cases} 1.0, & \text{if peak}_P > 5 \cdot \text{med}_P \text{ and peak}_P > 0.1 \cdot \text{tot}_P, \\ 0.0, & \text{otherwise.} \end{cases} \tag{7}$$

Spikes. Let $\mu^{(c)}$ and $\sigma^{(c)}$ be the mean and standard deviation of the signal. Define the z-score series: $z_t^{(c)} = \left|\frac{x_t^{(c)} - \mu^{(c)}}{\sigma^{(c)} + \epsilon}\right|$. Following (Yaro et al., 2023), we determines the spike score using the proportion of samples with $z_t^{(c)} > 3$ as:

$$\text{spikes}^{(c)} = \min\left(1.0, 10 \cdot \frac{1}{T_{\text{train}}} \sum_{t=1}^{T_{\text{train}}} \mathbb{I}[z_t^{(c)} > 3]\right). \tag{8}$$

Regime Shifts. Using a binary segmentation algorithm (Kovács et al., 2023), we detect breakpoints $\{t_k\}$ in the signal. Let K be the number of detected breakpoints. Then:

$$\text{regime}^{(c)} = \min\left(1.0, \frac{10K}{T_{\text{train}}}\right),$$

where the constant scales the ratio to the unit interval.

Drift. Drift is assessed by STL decomposition (Ouyang et al., 2021) as: $x_t^{(c)} = T_t^{(c)} + S_t^{(c)} + R_t^{(c)}$, where $T_t^{(c)}$ is the trend component. Then:

$$\text{drift}^{(c)} = \min\left(1.0, \frac{\text{std}(T^{(c)})}{\text{std}(x^{(c)}) + \epsilon}\right). \tag{9}$$

Outlier Dominance. We compute the skewness $s^{(c)}$ and kurtosis $k^{(c)}$ of the signal. If $k^{(c)} > 8 \quad \text{or} \quad |s^{(c)}| > 2$, then the channel is flagged as outlier-dominated (Kim & White, 2004) as :

$$\text{outlier\_dominated}^{(c)} = 1.0; \quad \text{else } = 0.0. \tag{10}$$

In Table A1, we show the channel-wise sampled parameter ranges used to generate synthetic perturbations for the ETTh2 dataset. For all channels, we observe moderately varied perturbation parameters, reflecting typical time-series structures with trends, seasonality, and noise, except for LULL which is a mostly flat channel as seen in Figure A5. We show similar tables for ETTh1 (Table A2), ETTm1 (Table A3), weather (Table A4), and exchange rate (Table A5). In Figures A5 and A6, we show the applied perturbations on the ETTh2 and exchange rate datasets respectively. The green and red vertical lines indicate the train/validation and validation/test boundaries, respectively. As it can be seen, we only apply the perturbations to the test portion of the dataset. In Fig A5, for ETTh2, we observe that the channel LULL, which is mostly flat in the original time-series, our framework applies no seasonal/trend based perturbations to it.

	Description	**HUFL**	**HULL**	**MUFL**	**MULL**	**LUFL**	**LULL**	**OT**
K	Segments	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)
d	Degree	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 1.00)	(1.00, 3.00)
coeff	Poly Coeff	(-0.32, 0.32)	(-0.29, 0.29)	(-0.22, 0.22)	(-0.34, 0.34)	(-0.18, 0.18)	(0.00, 0.00)	(-0.11, 0.11)
M	Sinusoids	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(0.00, 0.00)	(1.00, 3.00)
R	Regime Shifts	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 0.00)	(0.00, 4.00)
Δ	Regime Range	(-1.38, 1.38)	(-1.29, 1.29)	(-0.81, 0.81)	(-1.40, 1.40)	(-1.18, 1.18)	(0.00, 0.00)	(-0.55, 0.55)
σ	Noise Std	(0.20, 0.59)	(0.16, 0.48)	(0.12, 0.36)	(0.16, 0.49)	(0.08, 0.22)	(0.50, 0.50)	(0.10, 0.31)
γ	Std Fraction	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(0.00, 0.00)	(1.00, 1.00)
α	Scale Factor	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(1.00, 1.00)	(0.80, 1.2)
β	Shift Offset	(-0.11, 0.11)	(-0.16, 0.16)	(-0.05, 0.05)	(-0.13, 0.13)	(-0.19, 0.19)	(0.00, 0.00)	(-0.09, 0.09)

Table A1. Ranges of parameters for ETTh2

Notation	Description	HUFL	HULL	MUFL	MULL	LUFL	LULL	OT
K	Segments	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)
d	Degree	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)
coeff	Poly Coeff	(-0.36, 0.36)	(-0.40, 0.40)	(-0.36, 0.36)	(-0.41, 0.41)	(-0.45, 0.45)	(-0.28, 0.28)	(-0.12, 0.12)
M	Sinusoids	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)
R	Regime Shifts	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)
Δ	Regime Range	(-1.96, 1.96)	(-1.51, 1.51)	(-2.04, 2.04)	(-1.77, 1.77)	(-3.09, 3.09)	(-1.46, 1.46)	(-0.51, 0.51)
σ	Noise Std	(0.19, 0.57)	(0.18, 0.53)	(0.20, 0.58)	(0.18, 0.52)	(0.28, 0.84)	(0.17, 0.52)	(0.10, 0.29)
γ	Std Fraction	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)
α	Scale Factor	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)
β	Shift Offset	(-0.01, 0.01)	(0.07, -0.07)	(-0.01, 0.01)	(0.05, -0.05)	(0.06, -0.06)	(0.06, -0.06)	(-0.17, 0.17)

Table A2. Ranges of parameters for ETTh1

Notation	Description	HUFL	HULL	MUFL	MULL	LUFL	LULL	OT
K	Segments	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)
d	Degree	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 1.00)	(1.00, 3.00)
coeff	Poly Coeff	(-0.21, 0.21)	(-0.18, 0.18)	(-0.13, 0.13)	(-0.19, 0.19)	(-0.13, 0.13)	(0.00, 0.00)	(-0.04, 0.04)
M	Sinusoids	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(0.00, 0.00)	(1.00, 3.00)
R	Regime Shifts	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 0.00)	(0.00, 4.00)
Δ	Regime Range	(-0.70, 0.70)	(-0.66, 0.66)	(-0.54, 0.54)	(-0.77, 0.77)	(-0.50, 0.50)	(0.00, 0.00)	(-0.18, 0.18)
σ	Noise Std	(0.12, 0.37)	(0.11, 0.34)	(0.08, 0.26)	(0.11, 0.34)	(0.07, 0.20)	(0.50, 0.50)	(0.12, 0.38)
γ	Std Fraction	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(0.00, 0.00)	(1.00, 1.00)
α	Scale Factor	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(1.00, 1.00)	(0.80, 1.20)
β	Shift Offset	(-0.11, 0.11)	(-0.15, 0.15)	(-0.05, 0.05)	(-0.13, 0.13)	(-0.18, 0.18)	(0.00, 0.00)	(-0.09, 0.09)

Table A3. Ranges of parameters for ETTm1

Notation	Description	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	Tlog (degC)	OT
K	Segments	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)
d	Degree	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)
coeff	Poly Coeff	(-0.01, 0.01)	(-0.04, 0.04)	(-0.04, 0.04)	(-0.04, 0.04)	(-0.07, 0.07)	(-0.04, 0.04)	(-0.03, 0.03)	(-0.14, 0.14)
M	Sinusoids	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)
R	Regime Shifts	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)
Δ	Regime Range	(-0.05, 0.05)	(-0.19, 0.19)	(-0.20, 0.20)	(-0.22, 0.22)	(-0.40, 0.40)	(-0.25, 0.25)	(-0.17, 0.17)	(-0.86, 0.86)
σ	Noise Std	(0.02, 0.07)	(0.07, 0.21)	(0.07, 0.20)	(0.06, 0.17)	(0.13, 0.40)	(0.11, 0.32)	(0.08, 0.23)	(0.24, 0.72)
γ	Std Fraction	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)
α	Scale Factor	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)
β	Shift Offset	(-0.04, 0.04)	(-0.08, 0.08)	(-0.07, 0.07)	(-0.01, 0.01)	(0.13, -0.13)	(-0.08, 0.08)	(-0.08, 0.08)	(0.07, -0.07)

Table A4. Ranges of parameters for weather. Only the first 5 and last 2 channels are shown.

Notation	Description	0	1	2	3	4	5	6	OT
K	Segments	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)	(2.00, 6.00)
d	Degree	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)
coeff	Poly Coeff	(-0.04, 0.04)	(-0.07, 0.07)	(-0.04, 0.04)	(-0.04, 0.04)	(-0.04, 0.04)	(-0.05, 0.05)	(-0.03, 0.03)	(-0.04, 0.04)
M	Sinusoids	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)	(1.00, 3.00)
R	Regime Shifts	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)	(0.00, 4.00)
Δ	Regime Range	(-0.16, 0.16)	(-0.31, 0.31)	(-0.16, 0.16)	(-0.13, 0.13)	(-0.02, 0.02)	(-0.14, 0.14)	(-0.14, 0.14)	(-0.13, 0.13)
σ	Noise Std	(0.03, 0.10)	(0.06, 0.18)	(0.04, 0.11)	(0.03, 0.09)	(0.02, 0.05)	(0.04, 0.11)	(0.03, 0.08)	(0.04, 0.12)
γ	Std Fraction	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)	(1.00, 1.00)
α	Scale Factor	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)	(0.80, 1.20)
β	Shift Offset	(0.20, -0.20)	(-0.12, 0.12)	(0.15, -0.15)	(0.27, -0.27)	(0.13, -0.13)	(0.16, -0.16)	(0.26, -0.26)	(0.22, -0.22)

Table A5. Ranges of parameters for exchange rate.

A.3.2. Synthetic Generation and Parameterization.

To synthesize distribution shifts, we perturb the test split of the data by applying four structural components: (1) a piecewise polynomial trend, (2) multiple seasonal (sinusoidal) components, (3) discrete regime shifts, and (4) additive Gaussian noise. These components are combined into a base perturbation signal, and then adjusted using an offset and a scale to control its amplitude and level shift. We denote the independent perturbation for a given channel c at time-step $t \in [T_{\text{test}}, T]$, where T_{test} marks the start of the test split, as:

$\delta_t^{(c)} = \text{scale}_c \cdot \left[\delta_{\text{poly}}^{(c)}(t) + \delta_{\text{seasonal}}^{(c)}(t) + \delta_{\text{regime}}^{(c)}(t) + \epsilon_t^{(c)}\right] + \text{offset}_c$, where $\delta_{\text{poly}}^{(c)}(t)$ is a piecewise polynomial function; $\delta_{\text{seasonal}}^{(c)}(t)$ is a sum of sinusoids with sampled amplitude, frequency, and phase; $\delta_{\text{regime}}^{(c)}(t)$ consists of step-wise level shifts inserted at random positions; $\epsilon_t^{(c)} \sim \mathcal{N}(0, \sigma_c^2)$ is Gaussian noise scaled to match the residual variability of the original signal; and scale_c and offset_c are the scale and offset parameters for channel c. These are governed by a set of underlying parameters (including polynomial degree and coefficient scale, number and amplitude of sinusoids, frequency and magnitude of regime changes, and noise variance) which control the nature and intensity of each component. For each channel c, we define sampling ranges for these perturbation parameters based on three sources of information: (1) local (channel-specific) statistics computed from the training portion of the time-series for each channel ($\{x_t^{(c)}\}_{t=1}^{T_{\text{train}}}$ where T_{train} marks the end of the train split). These include: the *mean* which is used to scale the offset, the *standard deviation of first differences* (local variation) which is used to determine the range for polynomial coefficients, the *STL residual variance* which guides the standard deviation of injected Gaussian noise, the *dominant frequencies* and their *amplitude ranges* which define the frequency and strength of the injected seasonal sinusoids, and the *number of change points* and the magnitude of level changes between them which control the number and size of regime shifts that can be applied; (2) channel-specific profile vector $\mathbf{p}^{(c)}$ which modulate or suppress certain components (channels characterized as flat do not receive any trend or seasonal components); (3) global statistics aggregated across all channels that are used to parameterize a shared global signal $g(t)$ which helps preserve global consistency across multivariate channels.

A.3.3. Preserving Channel Relationships.

To preserve relationships in multivariate time-series across each channel pair, we incorporate a correlation-aware adjustment when applying the global components by adjusting $g(t)$ per channel based on its average Pearson correlation with other channels in the training data. Specifically, each channel receives a signed version $g^{(c)}(t) = \text{sign}(\bar{\rho}^{(c)}) \cdot g(t)$, where $\bar{\rho}^{(c)} = \frac{1}{C-1}\sum_{j \neq c} \rho(c, j)$ denotes the average correlation of channel c with all others. This ensures that positively and negatively correlated channels respond appropriately to the same global trend. The final perturbed signal is then constructed as: $\tilde{x}_t^{(c)} = x_t^{(c)} + \lambda_{\text{global}}^{(c)} \cdot g^{(c)}(t) + \lambda_{\text{indep}}^{(c)} \cdot \delta_t^{(c)} + \epsilon_t^{(c)}$, where $\lambda_{\text{global}}^{(c)} \in [0, 1]$ and $\lambda_{\text{indep}}^{(c)} \in [0.4, 1.0]$ control the strength of global and independent perturbations, respectively. Channels with low absolute correlation, *i.e.*, $|\bar{\rho}^{(c)}| \approx 0$, receive suppressed global contributions, preventing artificial alignment or distortion of weakly related signals. In Figure A4, we show the correlation between each channel-pair before and after applying perturbations.

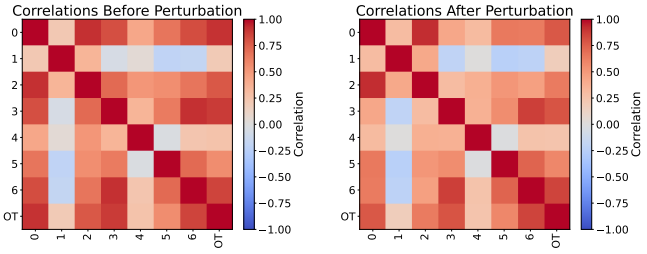
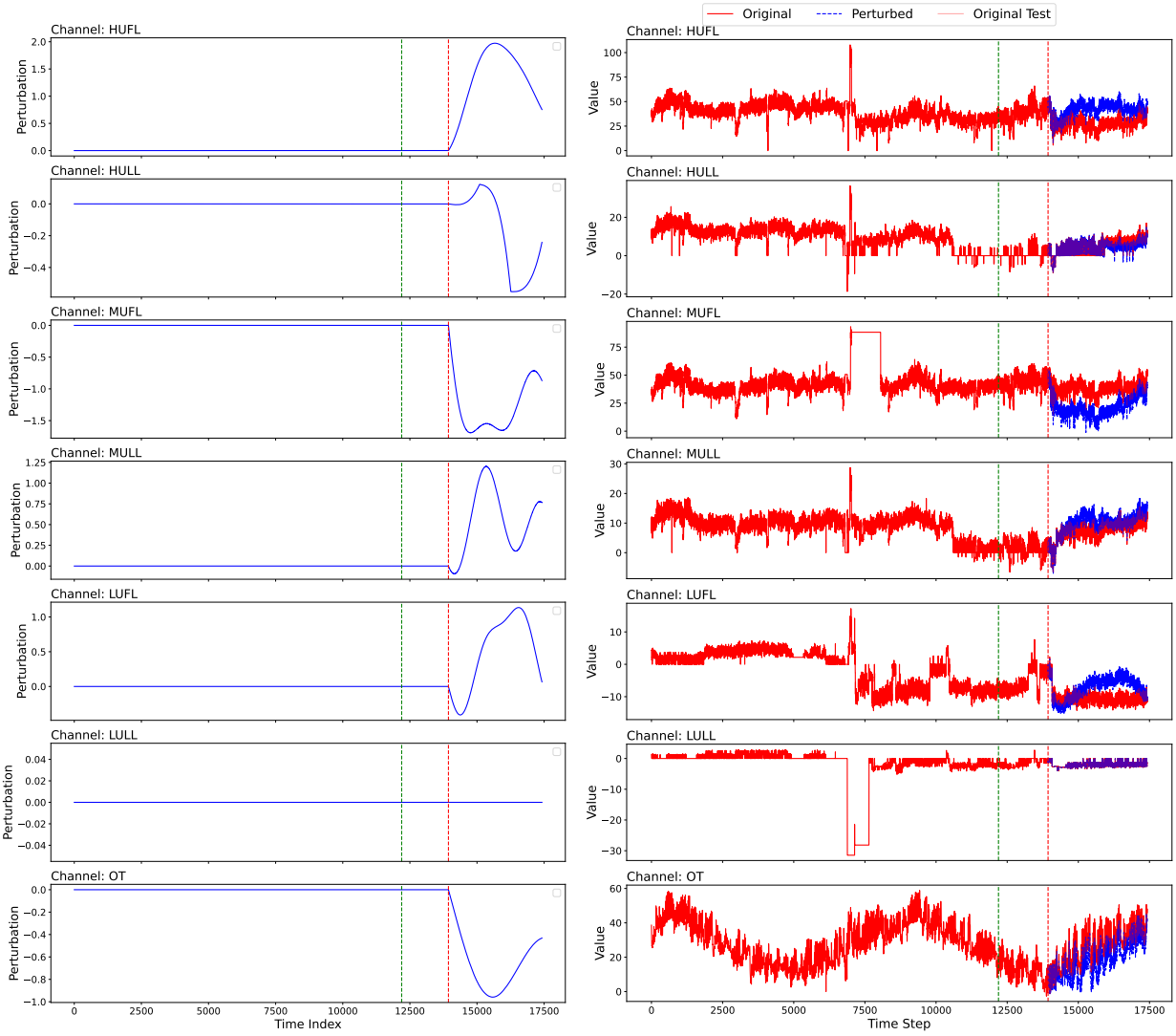


Figure A4. Our approach preserves cross-channel correlations after applying perturbations (e.g. Exchange Rate dataset).

A.3.4. Benchmark Curation.

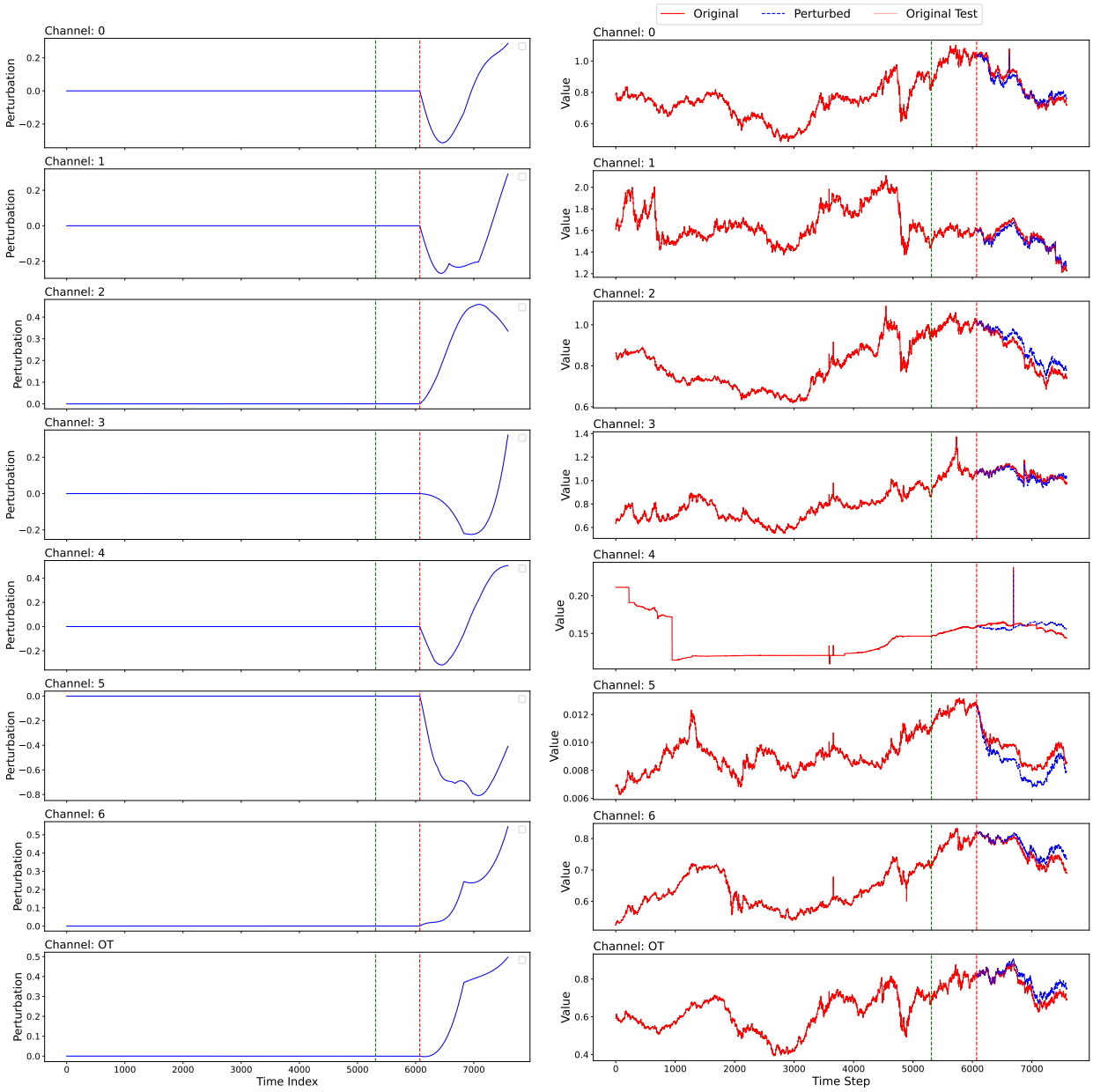
To generate a diverse test suite for a dataset, we repeat the sampling and synthetic generation process across $N = 1000$ perturbation variants. At each iteration, we first sample a random binary decision on whether to include a shared global perturbation $g(t)$. If selected, a set of global parameters is drawn from the aggregated channel statistics, and a common signal is generated. For each channel, we then independently sample individual perturbation parameters, conditioned on local statistics and structural profile $\mathbf{p}^{(c)}$, and use them to generate a unique perturbed signal. These are then combined with the global signal to produce the final perturbed time-series. This sampling strategy ensures a broad coverage of diverse combinations of shifts and their intensities, while maintaining internal consistency within each variant. Additional details regarding TTFBench is provided in Appendix A.3.



(a) Signals to guide the perturbations.

(b) Modified vs. original time-series.

Figure A5. Example visualization of synthetic distribution shifts applied on the multivariate ETTh2 time-series dataset.



(a) Signals to guide the perturbations.

(b) Modified vs. original time-series.

Figure A6. Example visualization of synthetic distribution shifts applied on the multivariate exchange rate time-series dataset.

Following the standard evaluation protocol of (Zhou et al., 2021), we create the training, validation, and test sets for each dataset, chronologically splitting the ETT datasets with a 60:20:20 ratio, and the remaining datasets with a 70:10:20 ratio. For all datasets, we fix the lookback window of $L = 96$ and evaluate for 4 different prediction horizons $H \in \{96, 192, 336, 720\}$ following (Wu et al., 2022). To ensure consistency with prior work, we adopt the experiment setup used in TAFAS (Kim et al., 2025) for pretraining the 5 backbone models as well as for training and evaluating TAFAS as a baseline for TTA. For DynaTTA, we set our base learning rate equal to the optimal learning rates for TAFAS. We set the minimum and maximum possible learning rates during adaptation as 0.0005 and 0.01 respectively. The buffer capacity of RTAB is set as twice the prediction horizon, *i.e.*, 2H where $H \in \{96, 192, 336, 720\}$. The buffer capacity of RDB is fixed to 96. All adaptation procedures are done on top of frozen pretrained models, with no updates made to the backbone parameters. Our code is implemented with PyTorch, and all of our experiments are conducted on a single NVIDIA Quadro RTX 6000 GPU. We release our code and benchmark on https://github.com/shivam-grover/DynaTTA.