DBCM: DUAL-BRANCH CALIBRATION MODULE FOR TEST-TIME ADAPTATION IN TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

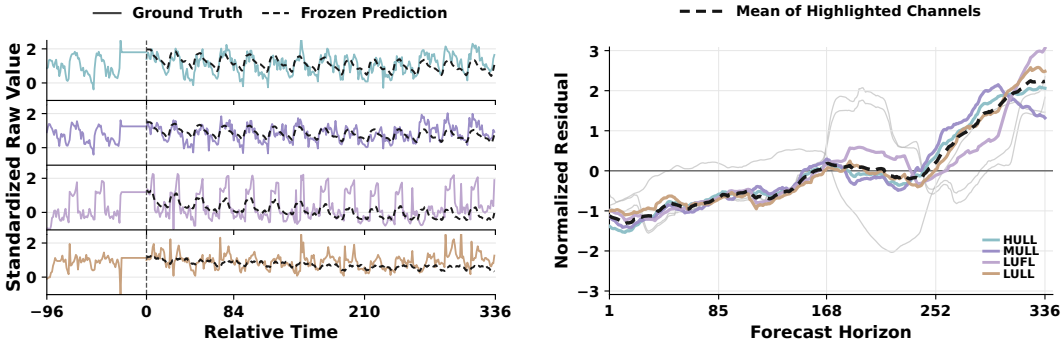
Non-stationarity induces test-time distribution shifts in time series forecasting (TSF), resulting in forecast residuals between predictions and the corresponding ground truth. Recent works on test-time adaptation for TSF address this by introducing lightweight calibration modules around frozen forecasters, updated using observations that become incrementally available at test time. Existing approaches typically employ channel-wise adapters, implicitly assuming that forecast residuals are inherently channel-specific. However, we observe that forecast residuals often exhibit substantial shared structure across channels, suggesting that fully independent channel-wise parameterization may be redundant. Motivated by this observation, we propose the **D**ual-**B**ranch **C**alibration **M**odule (DBCM), which shares adaptation capacity across channels through a shared adapter while retaining limited channel-specific flexibility through a low-rank channel-wise adapter, further enhanced by channel mixing and nonlinear gated modulation. Extensive experiments show that DBCM consistently outperforms prior methods across diverse datasets, horizons, and backbones. Code will be released.

1 INTRODUCTION

Time series forecasting (TSF) aims to predict future values based on historical observations, often involving multiple correlated variables. It serves as a fundamental tool in various real-world applications, including financial markets, energy systems, and weather forecasting (Petropoulos et al., 2022; Verma et al., 2024; Liu et al., 2023a; Chen et al., 2023b; Di Piazza et al., 2021). Despite its broad applicability, TSF is often challenged by non-stationarity, where test-time distribution shifts arise from evolving system dynamics or external factors (Kim et al., 2021; Liu et al., 2022; Fan et al., 2023; Liu et al., 2023b; Petropoulos et al., 2022). Such shifts degrade the performance of pre-trained forecasters, which are typically trained offline under the assumption that training and test data follow similar distributions. To address this challenge, recent studies have adopted test-time adaptation (TTA), which dynamically adjusts a pre-trained model during deployment. In computer vision, TTA commonly relies on entropy minimization over unlabeled test inputs (Wang et al., 2021; Niu et al., 2022). In TSF, however, observations become available after prediction and can serve as direct supervision. Accordingly, TSF-TTA frameworks leverage observed ground truth at test time to update lightweight calibration modules, while keeping the source forecaster frozen (Kim et al., 2025; Medeiros et al., 2025; Grover & Etemad, 2025; Im & Kwon, 2026), thereby improving forecasting performance under distribution shifts.

A key design choice in TSF-TTA is how to model test-time adaptation signals. Since these signals are given by forecast residuals, i.e., the differences between model predictions and the corresponding ground truth, their channel-space structure directly affects adapter design. In practice, most existing methods model these signals with channel-wise adapters, generating independent corrections for each channel (Kim et al., 2025; Medeiros et al., 2025; Grover & Etemad, 2025; Im & Kwon, 2026). This design implicitly assumes that forecast residuals are inherently channel-specific and thus require independent modeling. However, such independent parameterization can be redundant in TSF-TTA: the calibration modules are updated online from a limited number of sequentially revealed observations, and channel-wise adapters spend this limited supervision on repeatedly fitting correction patterns that are shared across channels. Figure 1 shows that although raw channel behaviors are heterogeneous, their forecast residuals reveal common components across channels. Since

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



(a) Raw channel behaviors.

(b) Residual curves after forecast.

Figure 1: **Shared Structure in Forecast Residuals.** Visualization on ETTh1 with iTransformer as the source forecaster. While raw channels are heterogeneous, forecast residuals exhibit structured patterns across channels. This suggests that independent channel-wise correction may be redundant.

TSF-TTA aims to correct these residuals, the shared structure observed in the residuals suggests that the required corrections need not be fully channel-specific. This raises the following research question: *Are residual corrections in TSF-TTA truly channel-specific, or do they exhibit a shared structure that can be modeled more efficiently?*

In this work, we show that forecast residuals often exhibit structured patterns across channels, while channel-specific variations are relatively limited. This indicates that fully independent channel-wise adapters are often over-parameterized for modeling corrections under limited online supervision. Motivated by this insight, we propose the **D**ual-**B**ranch **C**alibration **M**odule (DBCM), which reformulates the correction in TSF-TTA by explicitly decomposing adaptation into shared and channel-specific components. DBCM consists of a shared adapter that captures common correction patterns and a channel-wise low-rank adapter that models residual variations. By constraining the channel-wise branch with low-rank parameterization, our design reduces redundant capacity while preserving flexibility for channel-specific corrections. We further incorporate channel mixing for cross-channel context and nonlinear gated calibration for input-dependent modulation, and validate the effectiveness of the resulting module across diverse settings.

Our contributions are summarized as follows:

- We provide a new empirical perspective on TSF-TTA by analyzing forecast residuals in the channel-space. Our analysis shows that forecast residuals often exhibit substantial shared structure in the channel space, suggesting that fully independent channel-wise parameterization may be redundant.
- We propose the **D**ual-**B**ranch **C**alibration **M**odule (DBCM), which combines adaptation capacity shared across channels with limited, low-rank channel-specific flexibility. This design reduces redundant channel-wise capacity while preserving flexibility, and is further enhanced by channel mixing and nonlinear gated calibration for input-dependent modulation.
- We conduct extensive experiments across diverse datasets, forecasting horizons, and backbone architectures, demonstrating that DBCM consistently outperforms existing TSF-TTA methods while remaining relatively parameter-efficient.

2 Related Work

Test-Time Adaptation for Time Series Forecasting. TSF-TTA has emerged as an effective paradigm for handling non-stationarity. TAFAS (Kim et al., 2025) adapts a pre-trained model during inference using partially observed ground truth with a gated calibration module, while keeping the source forecaster fixed. PETSA (Medeiros et al., 2025) improves efficiency by introducing low-rank adapters to calibrate input and output with dynamic gating. DynaTTA (Grover & Etemad, 2025) proposes a dynamic adaptation framework that estimates distribution shifts in real time and adjusts the adaptation rate accordingly. COSA (Im & Kwon, 2026) introduces a context-aware output-space adapter that directly corrects forecast residuals. While the existing methods rely on

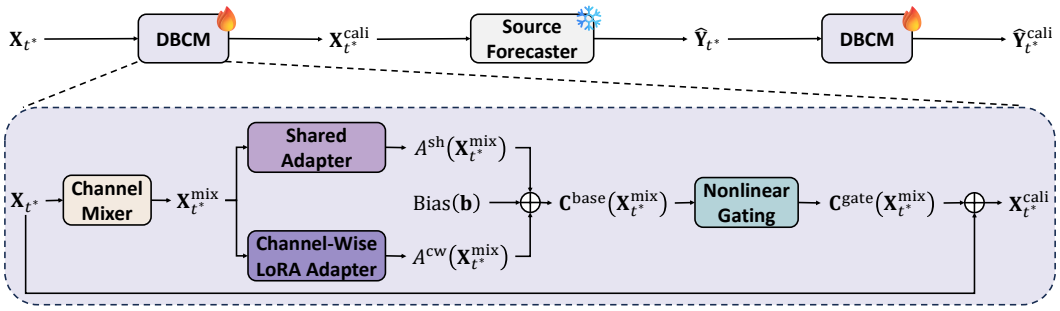


Figure 2: **Overview of the Dual-Branch Calibration Module (DBCM).** Given an input sequence, DBCM first applies channel mixing and processes the mixed sequence with a shared adapter and a channel-wise LoRA adapter. Their outputs are combined with a bias term, modulated by nonlinear gating, and added through a residual path to produce the calibrated sequence. The same DBCM structure is used on the output-side to calibrate the forecast.

channel-wise calibration, we observe that forecast residuals often exhibit a shared low-dimensional structure across channels, indicating that such designs may be over-parameterized. Building on this, we propose DBCM, which decomposes adaptation into shared and channel-specific components, reducing redundant channel-wise capacity while preserving flexibility through a low-rank formulation. Additional related work is discussed in Appendix G.

3 Method

In this section, we present the Dual-Branch Calibration Module (DBCM), which, unlike the fully channel-wise adaptation of TAFAS and PETSA, combines a shared correction branch with a channel-wise low-rank branch.

3.1 Problem Setting

We consider a multivariate time series $\{\mathbf{x}_t\}_{t=1}^{T}$, where $\mathbf{x}_t \in \mathbb{R}^C$ denotes the C-channel observation at time t. Given a lookback window size L and a forecasting horizon H, we define the input and target as $\mathbf{X}_t = \mathbf{x}_{t-L+1:t} \in \mathbb{R}^{L\times C}$ and $\mathbf{Y}_t = \mathbf{x}_{t+1:t+H} \in \mathbb{R}^{H\times C}$, respectively. A source forecaster f_θ produces predictions $\hat{\mathbf{Y}}_t = f_\theta(\mathbf{X}_t)$. At test time, instances arrive sequentially under possible distribution shifts. Following prior methods (Kim et al., 2025; Medeiros et al., 2025), $\mathbf{Y}_t$ is unavailable at prediction time but becomes partially observable as future time steps are revealed, and all methods are evaluated under the same protocol. Following TAFAS (Kim et al., 2025), the reported MSE is computed on the adjusted predictions, identically for all methods. In TSF-TTA, only calibration modules are adapted using the revealed test stream, while the source forecaster f_θ remains frozen to preserve pre-trained knowledge.

3.2 Dual-Branch Calibration Module

Overall Framework. To address distribution shift at test time, we adapt a pre-trained forecaster by applying calibration modules to both the input and output sides. Given an input sequence, the input-side DBCM first calibrates the input, which is then passed through the frozen source forecaster to produce a prediction. The output-side DBCM subsequently calibrates the prediction. Here, the input-side DBCM maps distribution-shifted test inputs closer to the input distribution that the pre-trained forecaster was trained to handle, while the output-side DBCM adjusts the forecaster's prediction toward the continuously evolving test distribution (Kim et al., 2025). Formally, the calibration process is defined as

$$\mathbf{X}_{t^*}^{\text{cali}} = C^{\text{in}}(\mathbf{X}_{t^*}), \quad \hat{\mathbf{Y}}_{t^*} = f_\theta(\mathbf{X}_{t^*}^{\text{cali}}), \quad \hat{\mathbf{Y}}_{t^*}^{\text{cali}} = C^{\text{out}}(\hat{\mathbf{Y}}_{t^*}), \tag{1}$$

where C^{in} and C^{out} denote the input-side and output-side DBCMs, respectively, and t^* denotes the time step at which the update occurs. The overall pipeline is illustrated in Figure 2.

We describe DBCM on the input side, as the output-side module has the same architecture, with the temporal dimension adjusted from L to H. The module consists of a linear channel mixer followed by two correction branches—a shared adapter and a channel-wise low-rank adapter—and a nonlinear gating mechanism.

Linear Channel Mixer. To capture cross-channel interactions (Chen et al., 2023a) before the subsequent calibration steps, a channel mixing transformation is first applied:

$$\mathbf{X}_{t^*}^{\text{mix}} = \mathbf{X}_{t^*} + \mathbf{X}_{t^*}\mathbf{W}^{\text{mix}}, \tag{2}$$

where $\mathbf{W}^{\text{mix}} \in \mathbb{R}^{C\times C}$ is a learnable parameter. The mixed representation is then passed to the shared and channel-wise branches, providing both branches with cross-channel context. In effect, the channel mixer helps reveal shared latent structure in the channel-space, allowing the subsequent adapters to better exploit common correction patterns. This interpretation is consistent with our isolated analysis in Appendix A.1, where the channel mixer makes forecast residuals more concentrated in a low-rank channel subspace while reducing their overall magnitude.

Shared Adapter. As illustrated in Figure 1 and analyzed in Section 4, forecast residuals exhibit shared structure across channels, suggesting that fully independent adaptation parameters for every channel may be redundant. We therefore share adaptation parameters across channels by applying a single temporal transformation to all channels:

$$A^{\text{sh}}(\mathbf{X}_{t^*}^{\text{mix}}) = \mathbf{W}^{\text{sh}}\mathbf{X}_{t^*}^{\text{mix}}, \tag{3}$$

where $\mathbf{W}^{\text{sh}} \in \mathbb{R}^{L\times L}$ is a learnable parameter applied uniformly to all channels. Intuitively, when channels are driven by a few common components, a shared temporal transformation corrects these components jointly across channels, whereas fully channel-wise adapters must learn them separately for each channel. By applying the same temporal transformation to every channel, the shared adapter is designed to capture common correction patterns with substantially fewer parameters than fully channel-wise adapters, and its effectiveness is empirically validated in Section 5.4.

Channel-Wise LoRA Adapter. Although the shared adapter captures common correction patterns across channels, channel-specific residual variations may remain. To model them without using a fully parameterized channel-wise adapter, we introduce a channel-wise low-rank branch. For each channel c, an independent low-rank temporal transformation is applied to the channel-mixed sequence:

$$A^{\text{cw}}(\mathbf{X}_{t^*}^{\text{mix}}) = \text{Concat}\Big(\big\{(\mathbf{U}_c\mathbf{V}_c)\,\mathbf{x}_{t^*,c}^{\text{mix}}\big\}_{c=1}^{C}\Big), \tag{4}$$

where $\mathbf{x}_{t^*,c}^{\text{mix}} \in \mathbb{R}^L$ denotes the c-th channel of $\mathbf{X}_{t^*}^{\text{mix}}$, and $\mathbf{U}_c \in \mathbb{R}^{L\times r}$ and $\mathbf{V}_c \in \mathbb{R}^{r\times L}$ define a channel-specific low-rank parameterization with rank $r \ll L$. $\text{Concat}(\cdot)$ concatenates the corrections along the channel dimension. We constrain this branch to be low rank, limiting its expressive capacity to complement the shared adapter while preserving flexibility for channel-specific corrections.

Nonlinear Gated Calibration. The *base correction* is computed by combining the outputs from the shared and channel-wise adapters:

$$\mathbf{C}^{\text{base}}(\mathbf{X}_{t^*}^{\text{mix}}) = A^{\text{sh}}(\mathbf{X}_{t^*}^{\text{mix}}) + A^{\text{cw}}(\mathbf{X}_{t^*}^{\text{mix}}) + \mathbf{b}, \tag{5}$$

where $\mathbf{b} \in \mathbb{R}^C$ is a learnable channel-wise bias parameter, which is broadcast along the temporal dimension to $\mathbb{R}^{L\times C}$.

While the base correction captures linear correction patterns, we introduce an input-dependent nonlinear gate to model variations that cannot be captured by a purely linear correction:

$$\mathbf{G}(\mathbf{X}_{t^*}^{\text{mix}}) = \sigma\big(\mathbf{W}^{\text{gate}}\mathbf{X}_{t^*}^{\text{mix}} + \mathbf{b}^{\text{gate}}\big), \tag{6}$$

where $\sigma(\cdot)$ denotes the sigmoid function, $\mathbf{W}^{\text{gate}} \in \mathbb{R}^{L\times L}$ and $\mathbf{b}^{\text{gate}} \in \mathbb{R}^L$ are learnable parameters, with $\mathbf{b}^{\text{gate}}$ broadcast to $\mathbb{R}^{L\times C}$ along the channel dimension. Using this gate, we construct the *gated correction*

$$\mathbf{C}^{\text{gate}}(\mathbf{X}_{t^*}^{\text{mix}}) = \mathbf{C}^{\text{base}}(\mathbf{X}_{t^*}^{\text{mix}}) + \mathbf{C}^{\text{base}}(\mathbf{X}_{t^*}^{\text{mix}}) \odot \mathbf{G}(\mathbf{X}_{t^*}^{\text{mix}}), \tag{7}$$

where $\odot$ denotes a Hadamard product.

The calibrated input is then given by

$$\mathbf{X}_{t^*}^{\text{cali}} = \mathbf{X}_{t^*} + \tanh(\mathbf{g}) \odot \mathbf{C}^{\text{gate}}(\mathbf{X}_{t^*}^{\text{mix}}), \tag{8}$$

where $\mathbf{g} \in \mathbb{R}^C$ is a learnable channel-wise gating parameter, and $\tanh(\mathbf{g})$ is broadcast to $\mathbb{R}^{L\times C}$ along the temporal dimension. Our two-level gating design separates fine-grained correction modulation from coarse channel-wise scaling used in prior TSF-TTA methods. Following prior TSF-TTA approaches (Kim et al., 2025; Medeiros et al., 2025; Grover & Etemad, 2025; Im & Kwon,

2026), the outer gate $\tanh(\mathbf{g})$ controls the overall channel-wise contribution of the correction. PETSA (Medeiros et al., 2025) further demonstrates that input-dependent dynamic gating is effective for adapting calibration strength under test-time distribution shifts. However, such gating primarily acts as a scaling mechanism for the calibrated correction. In contrast, our inner nonlinear gate modulates the correction values themselves, enabling input-dependent adjustment of the temporal correction pattern before the outer channel-wise gate is applied. Following the intuition of gated linear units (Dauphin et al., 2017), this formulation preserves the linear correction path through $\mathbf{C}^{\text{base}}$, while $\mathbf{C}^{\text{base}} \odot \mathbf{G}$ introduces input-dependent nonlinear modulation through the sigmoid gate.

The output-side module follows the same design, with the temporal dimension adjusted from the input length L to the prediction horizon H. The complete TSF-TTA algorithm is provided in Appendix B.

4 Analysis

In this section, we analyze TSF-TTA signals in the channel-space to provide empirical evidence for the design of DBCM. Specifically, we characterize whether forecast residuals, which serve as the supervision signal for test-time adaptation, are concentrated along shared channel directions, and examine whether adapter corrections align with these residual patterns. Additional analyses, including the effects of the channel-wise LoRA adapter, are provided in Appendix A.

Channel-Space Analysis Protocol. For each forecasting window $\mathbf{X}_t \in \mathbb{R}^{L\times C}$, we consider a channel-wise signal associated with the window, such as a forecast residual or an adapter correction. The forecast residual is $\mathbf{R}_t = \mathbf{Y}_t - f_\theta(\mathbf{X}_t)$, and the adapter correction is defined as the change in the prediction induced by the calibration modules,

$$\mathbf{\Delta}_t = \hat{\mathbf{Y}}_t^{\text{cali}} - f_\theta(\mathbf{X}_t) = C^{\text{out}}\big(f_\theta(C^{\text{in}}(\mathbf{X}_t))\big) - f_\theta(\mathbf{X}_t), \tag{9}$$

which includes both the effect of the input-side module propagated through the frozen forecaster and the correction directly applied by the output-side module. We stack these signals over the B evaluated forecasting instances and reshape them into a channel-space matrix $\mathbf{S} \in \mathbb{R}^{C\times(BH)}$, where each row corresponds to a channel. We then analyze the channel-space spectrum using the centered Gram matrix $\mathbf{\Sigma} = \tilde{\mathbf{S}}\tilde{\mathbf{S}}^\top$, where $\tilde{\mathbf{S}}$ is obtained by subtracting the mean of each row (channel) of $\mathbf{S}$. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_C$ denote the eigenvalues of $\mathbf{\Sigma}$. The top-k explained variance ratio is defined as

$$\text{EVR@}k = \frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{C}\lambda_i}, \tag{10}$$

and Rank@90 is the smallest k such that $\text{EVR@}k \geq 0.9$. Since the number of channels differs across datasets, we report channel-normalized metrics: $\text{EVR@30\%}$ denotes $\text{EVR@}k$ with $k = \lceil 0.3C \rceil$, and $\text{Rank@90}/C$ normalizes Rank@90 by the number of channels, both reported as percentages. A high $\text{EVR@30\%}$ and a small $\text{Rank@90}/C$ indicate that the signal is concentrated along a few shared channel directions. To compare forecast residuals with adapter corrections, let $\mathbf{U}^R, \mathbf{U}^\Delta \in \mathbb{R}^{C\times k}$ denote orthonormal bases of the top-k principal channel directions of the forecast residuals and adapter corrections, i.e., the leading eigenvectors of their centered Gram matrices. We measure their subspace alignment by the projection similarity

$$\text{Align}(\mathbf{U}^R, \mathbf{U}^\Delta) = \frac{1}{k}\left\|(\mathbf{U}^R)^\top \mathbf{U}^\Delta\right\|_F^2 = \frac{1}{k}\,\text{tr}\big(\mathbf{P}^R\mathbf{P}^\Delta\big), \tag{11}$$

where $\mathbf{P}^R$ and $\mathbf{P}^\Delta$ are the orthogonal projections onto the two subspaces, and we use $k = 2$. The score equals the mean squared cosine of the principal angles between the subspaces, is invariant to the sign and rotation of the basis vectors, and ranges from 0 to 1, with an expected value of k/C for randomly drawn subspaces.

Forecast Residual Structure. Since calibration modules are optimized against the forecast residual $\mathbf{R}_t$, its channel-space spectrum indicates whether residual variation is concentrated in a few shared channel directions or dispersed across channels. This distinction is relevant to adapter design: shared residual components need not be modeled separately for each channel, whereas residual variation not explained by those components motivates channel-specific flexibility. As shown in Table 1a, the residual spectra are consistent across backbone architectures, with $\text{EVR@30\%}$ values around 93% and $\text{Rank@90}/C$ values around 42%. These values are far from those of isotropic

Table 1: **Channel-Space Analysis.** Channel-space concentration of forecast residuals across backbones and datasets, and the subspace alignment between forecast residuals and adapter corrections.

(a) Residual spectra by backbone.

Backbone	EVR@30%	Rank@90/C
DLinear	93.4	41.5
FreTS	93.0	41.5
MICN	93.2	41.5
OLS	93.3	41.5
PatchTST	92.9	41.5
iTransformer	92.8	42.1
Average	93.1	41.6

(b) Residual spectra by dataset.

Dataset	EVR@30%	Rank@90/C
ETTh1	94.7	42.9
ETTh2	93.1	42.9
ETTm1	95.0	42.9
ETTm2	91.8	43.5
Exchange	85.7	50.0
Weather	98.2	27.4
Average	93.1	41.6

(c) Subspace alignment.

Adapter	**Alignment**
Oracle	1.000
Random	0.246
TAFAS	0.671
TAFAS-Shared	0.723
PETSA	0.666
DBCM (Ours)	**0.774**

residuals evenly spread over all channel directions, which would yield an EVR@30% of 33–43% and a Rank@90/C of 90–100%, suggesting that the dominant correction patterns are largely shared across channels. Fully independent channel-wise adapters may redundantly model these shared residual patterns with separate parameters for each channel. Rather than prescribing a specific architecture, this observation serves as empirical motivation for sharing adaptation capacity across channels, whose effectiveness we validate through the ablation in Section 5.4.

Table 1b further shows that the degree of concentration varies across datasets. Weather exhibits the strongest concentration, with an EVR@30% of 98.2% and a Rank@90/C of 27.4%, whereas Exchange Rate exhibits the weakest concentration, with an EVR@30% of 85.7% and a Rank@90/C of 50.0%. Exchange Rate still remains structured, but its lower EVR@30% and higher Rank@90/C indicate that a non-negligible portion of residual variation lies outside the dominant shared directions. In such cases, a purely shared adapter may be too restrictive, while a full channel-wise adapter introduces unnecessary capacity. DBCM addresses this trade-off with two branches: a shared branch for patterns common across channels and a channel-wise LoRA branch for residual variations not captured by the shared branch. The strength of such structure depends on both the data-generating process and the forecaster (West, 1996), and it can arise when many channels share a few factors that are incompletely modeled (Zheng & Sun, 2024; Salinas et al., 2019).

Residual–Correction Alignment. We further analyze how adapter corrections obtained during TSF-TTA align with forecast residuals. Table 1c reports the alignment between their principal channel directions. Oracle denotes the ideal adapter correction, defined as the forecast residual itself, and therefore achieves an alignment score of 1 by construction, while Random denotes the expected score of randomly drawn subspaces. Across methods, adapter corrections consistently align with forecast residuals, yielding alignment scores far above the random baseline. This suggests that the adapted corrections effectively reflect the residual structure. Notably, TAFAS-Shared, a TAFAS variant that uses a single shared adapter instead of channel-wise adapters, achieves higher alignment than TAFAS (0.723 vs. 0.671). This further supports the residual-spectrum analysis: the dominant shared component of the correction can be largely captured without fully independent channel-wise parameterization. Among the compared methods, DBCM achieves the strongest alignment, indicating that its corrections are most closely aligned with the leading residual subspace. This result is consistent with decomposing correction into a shared branch and a low-rank channel-wise branch. Appendix A.5 provides additional dataset-wise alignment scores and analyses.

5 Experiments

We evaluate the effectiveness of DBCM under test-time distribution shifts on multiple benchmark datasets. We consider diverse backbone architectures and forecasting horizons to verify the general applicability of our method. Additional experimental results, including comparisons with DynaTTA (Grover & Etemad, 2025) and COSA (Im & Kwon, 2026), are provided in Appendix C.

5.1 Experimental Setup

Datasets. We conduct experiments on six standard multivariate time series forecasting benchmarks: ETTh1, ETTh2, ETTm1, ETTm2, Exchange Rate, and Weather. These datasets are widely used in prior long-term forecasting studies and exhibit diverse temporal dynamics (Zhou et al.,

Table 2: **Prediction Performance Comparison Across Methods.** Test MSE is reported for six models on six datasets with four prediction horizons. Bold entries indicate the best performance among methods, while underlined entries denote the second-best. Avg. Imp. denotes the relative MSE reduction (%) over the frozen baseline, averaged over all dataset–horizon settings. Full results with standard deviations are provided in Table C.14.

		iTransformer				**DLinear**				**FreTS**			
		Baseline	TAFAS	PETSA	DBCM	Baseline	TAFAS	PETSA	DBCM	Baseline	TAFAS	PETSA	DBCM
ETTh1	96	.4349	.4308	.4284	**.4228**	.4510	.4425	.4400	**.4318**	.4360	.4295	.4296	**.4242**
	192	.4937	.4894	.4886	**.4805**	.5042	.4946	.4943	**.4835**	.4886	.4819	.4842	**.4764**
	336	.5406	.5382	.5392	**.5322**	.5508	.5454	.5455	**.5373**	.5466	.5394	.5462	**.5354**
	720	.6914	.6520	.6535	**.6298**	.7019	.6713	.6617	**.6487**	.7145	.6839	.6795	**.6565**
ETTh2	96	.2430	.2406	.2403	**.2388**	.2294	.2274	.2280	**.2268**	.2376	**.2347**	.2359	**.2347**
	192	.2973	.2931	.2918	**.2914**	.2835	**.2793**	.2840	.2794	.2838	.2776	.2805	**.2761**
	336	.3448	.3328	.3334	**.3301**	.3250	.3174	.3185	**.3152**	.3184	.3051	.3117	**.3048**
	720	.4258	.3999	.3987	**.3939**	.4149	.3913	.3878	**.3820**	.3951	.3690	.3812	**.3653**
ETTm1	96	.3763	.3606	.3601	**.3415**	.3705	.3496	.3524	**.3420**	.3618	.3524	.3525	**.3420**
	192	.4371	.4184	.4145	**.4038**	.4420	.4159	.4175	**.4056**	.4304	.4177	.4168	**.4060**
	336	.5029	.4778	.4739	**.4628**	.5172	.4796	.4795	**.4626**	.4928	.4752	.4719	**.4618**
	720	.5713	.5484	.5472	**.5276**	.5927	.5482	.5526	**.5321**	.5609	.5399	.5404	**.5274**
ETTm2	96	.1555	.1542	.1543	**.1536**	.1589	.1577	.1579	**.1573**	.1563	.1547	.1549	**.1536**
	192	.1938	.1926	.1928	**.1923**	.1929	.1913	.1913	**.1907**	.1914	.1896	.1898	**.1880**
	336	.2345	.2441	.2325	**.2312**	.2325	.2290	.2293	**.2289**	.2310	.2275	.2305	**.2259**
	720	.3101	.3012	.3033	**.2990**	.3061	.2967	.2962	**.2956**	.3026	**.2924**	.3003	.2930
Exchange	96	.0916	.0883	.0879	**.0858**	.0811	**.0789**	.0793	**.0789**	.0827	**.0784**	.0815	.0794
	192	.1914	.1834	.1867	**.1770**	.1708	.1643	.1641	**.1626**	.1787	.1667	.1765	**.1659**
	336	.3375	.3146	.3243	**.2844**	.3205	.2883	.2931	**.2642**	.3265	.2922	.3198	**.2659**
	720	.9433	.8388	.8894	**.8182**	.8383	.8309	.8109	**.7066**	.8251	.7606	.8141	**.6700**
Weather	96	.1719	.1642	.1661	**.1630**	.1936	.1788	.1769	**.1732**	.1809	.1693	.1713	**.1677**
	192	.2233	.2104	.2129	**.2096**	.2383	.2236	.2255	**.2194**	.2266	.2109	.2179	**.2101**
	336	.2800	.2610	.2646	**.2608**	.2913	.2714	.2755	**.2697**	.2806	.2609	.2675	**.2602**
	720	.3576	.3404	.3498	**.3360**	.3643	.3490	.3501	**.3391**	.3557	.3364	.3536	**.3338**
Average		.3687	.3531	.3556	**.3444**	.3655	.3509	.3505	**.3389**	.3585	.3436	.3503	**.3343**
Avg. Rank		3.96	2.50	2.54	**1.00**	3.96	2.33	2.65	**1.06**	4.00	2.02	2.88	**1.10**
Avg. Imp.		–	3.55	3.19	**5.75**	–	3.93	3.83	**6.22**	–	4.10	2.15	**5.83**

		PatchTST				**OLS**				**MICN**			
		Baseline	TAFAS	PETSA	DBCM	Baseline	TAFAS	PETSA	DBCM	Baseline	TAFAS	PETSA	DBCM
ETTh1	96	.4294	.4232	.4255	**.4194**	.4511	.4420	.4436	**.4321**	.4573	.4471	.4453	**.4342**
	192	.4904	.4805	.4849	**.4757**	.5044	.4923	.5040	**.4836**	.5145	.4996	.5063	**.4900**
	336	.5605	.5490	.5600	**.5410**	.5510	.5415	.5505	**.5383**	.5715	.5596	.5708	**.5553**
	720	.6904	.6643	.6878	**.6417**	.6996	.6654	.6728	**.6514**	.7224	.6916	.7201	**.6743**
ETTh2	96	.2309	.2298	.2308	**.2291**	.2305	.2283	.2286	**.2279**	.2346	.2326	.2333	**.2321**
	192	.2871	.2802	.2867	**.2785**	.2838	.2817	.2822	**.2802**	.2934	.2868	.2898	**.2848**
	336	.3311	.3163	.3303	**.3157**	.3257	.3186	.3251	**.3157**	.3380	.3254	.3309	**.3250**
	720	.4067	.3886	.3958	**.3769**	.4164	.3902	.3883	**.3831**	.4328	.4126	.4191	**.4090**
ETTm1	96	.3869	.3723	.3829	**.3469**	.3708	.3520	.3574	**.3420**	.4063	.3723	.3756	**.3576**
	192	.4387	.4220	.4328	**.4057**	.4437	.4165	.4230	**.4054**	.4479	.4315	.4308	**.4155**
	336	.4957	.4780	.4939	**.4593**	.5176	.4786	.4853	**.4623**	.5154	.4885	.4889	**.4676**
	720	.5666	.5390	.5615	**.5255**	.5918	.5478	.5519	**.5317**	.5749	.5404	.5689	**.5206**
ETTm2	96	.1554	.1542	.1552	**.1534**	.1602	.1593	.1592	**.1585**	.1592	.1580	.1582	**.1574**
	192	.1937	.1924	.1935	**.1919**	.1935	.1921	**.1919**	**.1919**	.1955	.1927	.1923	**.1914**
	336	.2321	.2292	.2317	**.2282**	.2331	**.2298**	.2301	**.2298**	.2421	.2364	**.2355**	.2357
	720	.3033	**.2924**	.3005	.2933	.3066	.2982	.2977	**.2964**	.3176	.3040	.3049	**.3023**
Exchange	96	.0846	.0811	.0839	**.0806**	.0814	.0795	.0808	**.0793**	.0829	.0804	.0818	**.0800**
	192	.1805	.1697	.1787	**.1683**	.1723	.1640	.1700	**.1631**	.1886	.1771	.1853	**.1742**
	336	.5375	.4585	.4755	**.4087**	.3226	.2851	.2876	**.2613**	.3529	.3116	.3315	**.2829**
	720	.8814	.8812	.8814	**.6609**	.8360	.6485	.8114	**.5439**	1.4040	**1.1221**	1.3355	1.2209
Weather	96	.1738	.1672	.1730	**.1657**	.1955	.1808	.1819	**.1763**	.1720	.1699	.1726	**.1670**
	192	.2207	.2103	.2183	**.2073**	.2403	.2219	.2272	**.2196**	.2222	.2155	.2237	**.2148**
	336	.2760	.2598	.2721	**.2586**	.2920	.2706	.2772	**.2697**	.2747	.2626	.2701	**.2624**
	720	.3512	.3644	.3410	**.3315**	.3646	.3443	.3626	**.3398**	.3491	.3371	.3546	**.3298**
Average		.3710	.3585	.3657	**.3402**	.3660	.3429	.3538	**.3326**	.3946	.3690	.3844	**.3660**
Avg. Rank		3.94	2.04	2.98	**1.04**	4.00	2.15	2.81	**1.04**	3.88	2.12	2.92	**1.08**
Avg. Imp.		–	3.29	1.24	**6.43**	–	5.09	3.12	**7.07**	–	4.50	2.14	**5.90**

2021; Wu et al., 2021), making them suitable for evaluating adaptation performance under distribution shift. We further consider Electricity in an additional experiment to evaluate scalability. Given its substantially larger number of channels, it serves as a challenging testbed for assessing whether our method remains robust in high-dimensional multivariate settings. Detailed dataset information is provided in Appendix D.

Backbone Models. We evaluate our method on a diverse set of backbone models. These models cover a wide range of architectural paradigms for time series forecasting, including linear models (DLinear (Zeng et al., 2023), OLS (Toner & Darlow, 2024)), transformer-based models (iTransformer (Liu et al., 2024), PatchTST (Nie et al., 2023)), frequency-based methods (FreTS (Yi et al., 2023)), and convolutional architectures (MICN (Wang et al., 2023)). This diversity allows us to assess the general applicability of our approach across different modeling assumptions. In addition, we further validate our method on recent TSFMs, including Moirai (Woo et al., 2024), Chronos (Ansari et al., 2024), and Chronos-2 (Ansari et al., 2025). These models are pre-trained on large-scale datasets and represent a different regime from standard supervised forecasting models. Evaluating on these models demonstrates that our approach remains effective even in large-scale pre-trained settings.

Implementation Details. Following TimesNet (Wu et al., 2023), we use a fixed lookback window $L = 96$ and prediction horizons $H \in \{96, 192, 336, 720\}$. All datasets are split chronologically, with train/validation/test ratios of 0.6/0.2/0.2 for the ETT datasets and 0.7/0.1/0.2 for Exchange Rate, Weather, and Electricity. We report mean squared error (MSE) as the primary evaluation metric. In all experiments, we set the rank of the channel-wise LoRA adapter to $r = 8$ unless otherwise specified. In the ablation studies, we use iTransformer (Liu et al., 2024) as the backbone model. To ensure robustness, results are averaged over three runs with different random seeds.

5.2 Main Results

We compare our method with state-of-the-art TSF-TTA approaches, TAFAS (Kim et al., 2025) and PETSA (Medeiros et al., 2025). As shown in Table 2, our method consistently improves performance across all evaluated backbones. Averaged over all 144 settings, DBCM reduces MSE by 6.20% relative to the frozen forecaster, compared with 4.07% for TAFAS and 2.61% for PETSA, corresponding to about 1.5× the gain of the strongest prior method. This gap widens at longer horizons (4.24% vs. 2.83% at $H = 96$ and 8.87% vs. 5.46% at $H = 720$ for DBCM and TAFAS, respectively). The improvement is particularly pronounced on the Exchange Rate dataset, which exhibits the strongest non-stationarity according to the ADF statistics in Appendix D. For example, under DLinear with $H = 720$, the error decreases from 0.8383 to 0.7066 (15.7% improvement), whereas TAFAS and PETSA achieve only 0.9% and 3.3%, respectively. Similarly, under PatchTST at $H = 720$, our method reduces the error from 0.8814 to 0.6609 (25.0%), while TAFAS and PETSA show negligible gains of 0.02% and 0.00%, respectively. DBCM surpasses channel-wise adaptation methods, indicating that modeling shared and residual components with lightweight nonlinear gating is sufficient for effective adaptation.

5.3 Scalability Across Model and Data Scales

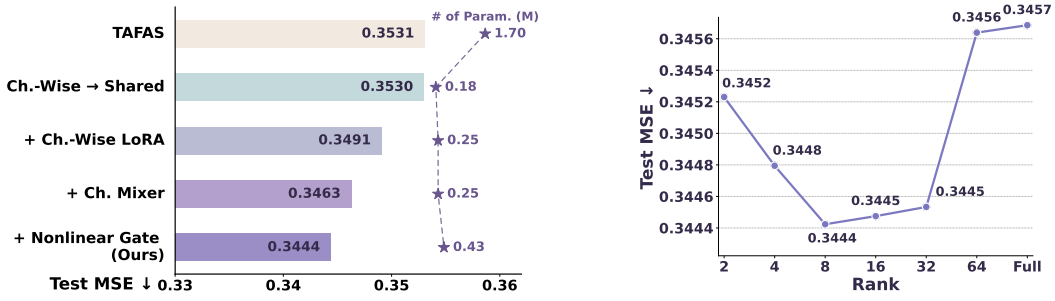
We further evaluate DBCM on the high-dimensional Electricity dataset and on three TSFMs, Moirai, Chronos, and Chronos-2. Table 3 summarizes the results, and the full results are provided in Appendix C.1. On Electricity, DBCM achieves the best performance across all backbones and prediction horizons, with larger gains at longer horizons: for iTransformer at $H = 720$, it improves the baseline by 5.45%, compared with 2.33% for TAFAS and 0.83% for PETSA. On TSFMs, DBCM also achieves the best performance on all datasets, indicating that the proposed calibration is not tied to a specific foundation model family and can complement large pre-trained forecasters under test-time distribution shifts.

Table 3: **Scalability.** Average test MSE on Electricity (over four horizons) and on TSFMs (over the four ETT datasets at $H = 96$).

	Baseline	TAFAS	PETSA	DBCM
Electricity				
iTransformer	.1828	.1770	.1798	**.1744**
DLinear	.2149	.2093	.2118	**.2063**
FreTS	.1978	.1936	.1956	**.1924**
TSFMs				
Moirai	.5039	.3470	.3507	**.3238**
Chronos	.6320	.3349	.3504	**.3176**
Chronos-2	.3943	.3214	.3235	**.3089**

5.4 Ablation Study

Ablation on Components. We conduct an ablation study to evaluate each component. Using iTransformer as the source forecaster, we report test MSE averaged over all six datasets, four forecasting horizons, and three random seeds. Starting from TAFAS, we first replace its channel-wise



(a) Component ablation.

(b) Effect of LoRA rank.

Figure 3: **Ablation Study.** (a) Component ablation with test MSE (across datasets and horizons) and parameter counts under incremental component addition. (b) Effect of LoRA rank on test MSE.

linear adapter with a single shared adapter, and then successively add the channel-wise LoRA adapter, the linear channel mixer, and the nonlinear gate, where the final variant corresponds to DBCM. As shown in Figure 3a, replacing the channel-wise adapter in TAFAS with a shared adapter achieves comparable performance with substantially fewer parameters, supporting the redundancy of fully independent channel-wise parameterization suggested by the channel-space analysis in Section 4, where residuals are concentrated in shared channel directions and TAFAS-Shared aligns with the residual subspace better than TAFAS. Adding a channel-wise low-rank adapter further improves performance with only a modest parameter increase, complementing the restrictiveness of purely shared adaptation by capturing the residual variation outside the dominant shared directions observed in Section 4. The linear channel mixer brings additional gains by providing cross-channel context to the subsequent correction branches, and the nonlinear gating mechanism further refines the correction through input-dependent modulation. Overall, our final design outperforms TAFAS while using fewer parameters. Since increasing the LoRA rank beyond 8 does not improve performance (Figure 3b) and the final design uses fewer parameters than TAFAS, these gains are not explained by added capacity alone. We further show in Appendix C.9 that input- and output-side DBCM each improve over the source forecaster, and applying both yields the best performance.

Ablation on LoRA Rank. We further analyze the effect of the LoRA rank in Figure 3b. Performance improves as the rank increases from 2 to 8, indicating that a minimal level of channel-specific flexibility is required to complement the shared correction branch. However, increasing the rank beyond 8 yields no further improvement, and excessively large settings, such as 64 or full-rank adaptation, even degrade performance.

6 Conclusion

In this work, we examine the channel-space structure of forecast residuals, which serve as the direct supervision signal for TSF-TTA. Our analysis shows that these residuals are concentrated along a small number of shared channel directions, and that adapter corrections are well aligned with this residual structure. Since this structure suggests that fully independent channel-wise parameterization is redundant, we introduce DBCM, which shares adaptation capacity across channels while retaining limited channel-specific flexibility through a lightweight low-rank branch. More broadly, our results suggest that sharing adaptation capacity across channels while retaining low-rank channel-specific flexibility provides an effective inductive bias for TSF-TTA, rather than assuming fully independent channel-wise corrections. Since the amount of residual variation can differ across datasets, an important direction for future work is to adaptively determine the channel-wise branch capacity based on these differences.

Limitations. While our analysis focuses on the output space, including forecast residuals and output-side corrections, it does not explicitly characterize input-side dynamics, a limitation shared with prior TSF-TTA methods. The input-side DBCM is nevertheless trained with the same objective on the final calibrated prediction, and using it alone already improves forecasting performance (Appendix C.9). A direct analysis of input-level adaptation remains future work.

AI USE STATEMENT

In this work, we did not use generative AI tools for any of the tasks with required disclosure. In particular, we have not used generative AI tools for research ideation or experimental design, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for grammar checking and for editing the paper to improve readability. We have reviewed all AI-assisted work. All AI-assisted edits to the text were manually reviewed and revised by the authors to ensure that they accurately reflect our work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work proposes a test-time adaptation method for time series forecasting and uses only publicly available benchmark datasets that contain no personally identifiable information. We are not aware of any ethical concerns associated with this work.

REPRODUCIBILITY STATEMENT

We will release our code upon publication. The complete algorithm of DBCM is provided in Appendix B, the dataset descriptions and preprocessing details are given in Appendix D, and the training setup is described in Appendix E. All results are averaged over three runs with different random seeds.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *TMLR*, 2024.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting. Technical report, Amazon Web Services, 2025.

Mouxiang Chen, Lefei Shen, Han Fu, Zhuo Li, Jianling Sun, and Chenghao Liu. Calibration of time-series forecasting: Detecting and adapting context-driven distribution shift. In *KDD*, 2024.

Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O. Arik, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting. *TMLR*, 2023a.

Weisi Chen, Walayat Hussain, Francesco Cauteruccio, and Xu Zhang. Deep learning for financial time series prediction: A state-of-the-art review of standalone and hybrid models. *CMES*, 2023b.

Tao Dai, Beiliang Wu, Peiyuan Liu, Naiqi Li, Xue Yuerong, Shu-Tao Xia, and Zexuan Zhu. DDN: Dual-domain dynamic normalization for non-stationary time series forecasting. In *NeurIPS*, 2024.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *ICML*, 2024.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *ICML*, 2017.

A. Di Piazza, M.C. Di Piazza, G. La Tona, and M. Luna. An artificial neural network-based forecasting model of energy-related time series for electrical grid management. *MATCOM*, 2021.

Graham Elliott, Thomas J. Rothenberg, and James H. Stock. Efficient tests for an autoregressive unit root. *Econometrica*, 1996.

Wei Fan, Pengyang Wang, Dongkun Wang, Dongjie Wang, Yuanchun Zhou, and Yanjie Fu. Dish-ts: A general paradigm for alleviating distribution shift in time series forecasting. In *AAAI*, 2023.

Shivam Grover and Ali Etemad. Shift-aware test time adaptation and benchmarking for time-series forecasting. In *ICML*, 2025.

Seungha Hong, Sukang Chae, Suyeon Kim, Sanghwan Jang, and Hwanjo Yu. Dynamic multi-period experts for online time series forecasting. In *WWW*, 2026.

Jeonghwan Im and Hyuk-Yoon Kwon. COSA: Context-aware output-space adapter for test-time adaptation in time series forecasting. In *ICLR*, 2026.

HyunGi Kim, Siwon Kim, Jisoo Mok, and Sungroh Yoon. Battling the non-stationarity in time series forecasting via test-time adaptation. In *AAAI*, 2025.

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

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Seunghan Lee, Taeyoung Park, and Kibok Lee. Dataset-driven channel masks in transformers for multivariate time series. In *ICASSP*, 2026.

Thomas L Lee, William Toner, Rajkarn Singh, Artjom Joosen, and Martin Asenov. Lightweight online adaption for time series foundation model forecasts. In *ICML*, 2025.

Zhe Li, Xuanxuan Li, Zhongwen Rao, Lujia Pan, and Zenglin Xu. Mts-mixers: Multivariate time series forecasting via factorized temporal and channel mixing. In *IJCNN*, 2025.

Hangchen Liu, Zheng Dong, Renhe Jiang, Jiewen Deng, Jinliang Deng, Quanjun Chen, and Xuan Song. Spatio-temporal adaptive embedding makes vanilla transformer sota for traffic forecasting. In *CIKM*, 2023a.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *NeurIPS*, 2022.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *ICLR*, 2024.

Zhiding Liu, Mingyue Cheng, Zhi Li, Zhenya Huang, Qi Liu, Yanhu Xie, and Enhong Chen. Adaptive normalization for non-stationary time series forecasting: A temporal slice perspective. In *NeurIPS*, 2023b.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Heitor R Medeiros, Hossein Sharifi-Noghabi, Gabriel L Oliveira, and Saghar Irandoust. Accurate parameter-efficient test-time adaptation for time series forecasting. In *ICML Workshop*, 2025.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *ICLR*, 2023.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *ICML*, 2022.

Fotios Petropoulos, Daniele Apiletti, Vassilios Assimakopoulos, Mohamed Zied Babai, Devon K Barrow, Souhaib Ben Taieb, Christoph Bergmeir, Ricardo J Bessa, Jakub Bijak, John E Boylan, et al. Forecasting: theory and practice. *IJF*, 2022.

Quang Pham, Chenghao Liu, Doyen Sahoo, and Steven CH Hoi. Learning fast and slow for online time series forecasting. In *ICLR*, 2023.

David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *NeurIPS*, 2019.

William Toner and Luke Darlow. An analysis of linear time series forecasting models. In *ICML*, 2024.

Yogesh Verma, Markus Heinonen, and Vikas Garg. Climode: Climate and weather forecasting with physics-informed neural odes. In *ICLR*, 2024.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021.

Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *ICLR*, 2023.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *CVPR*, 2022.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. In *NeurIPS*, 2024.

Qingsong Wen, Weiqi Chen, Liang Sun, Zhang Zhang, Liang Wang, Rong Jin, Tieniu Tan, et al. Onenet: Enhancing time series forecasting models under concept drift by online ensembling. In *NeurIPS*, 2023.

Kenneth D West. Asymptotic inference about predictive ability. *Econometrica*, 1996.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *ICML*, 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*, 2021.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *ICLR*, 2023.

Kun Yi, Qi Zhang, Wei Fan, Shoujin Wang, Pengyang Wang, Hui He, Ning An, Defu Lian, Longbing Cao, and Zhendong Niu. Frequency-domain mlps are more effective learners in time series forecasting. In *NeurIPS*, 2023.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *AAAI*, 2023.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *ICLR*, 2023.

Lifan Zhao and Yanyan Shen. Proactive model adaptation against concept drift for online time series forecasting. In *KDD*, 2025.

Vincent Zhihao Zheng and Lijun Sun. Multivariate probabilistic time series forecasting with correlated errors. In *NeurIPS*, 2024.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, 2021.

Pengfei Zhou, Yunlong Liu, Junli Liang, Qi Song, and Xiangyang Li. Crosslinear: Plug-and-play cross-correlation embedding for time series forecasting with exogenous variables. In *KDD*, 2025.

A Additional Analysis

A.1 Effect of Channel Mixing on Residual Structure.

We further analyze whether the channel mixer indeed makes the subsequent correction problem more structured. Using iTransformer as the source forecaster, Figure A.1 compares the forecast residuals obtained under three settings on the same test window of ETTm2: the original prediction without channel mixing, the prediction obtained with the learned channel mixer adapted under the TSF-TTA setting, and the prediction obtained with an oracle channel mixer optimized for that window to minimize the magnitude of the forecast residual. The oracle result serves as an upper-bound reference, showing that applying channel mixing before correction can substantially reduce the residual magnitude while making the resulting residual more concentrated along a few channel directions. More importantly, the learned channel mixer follows the same trend in practice: it reduces MSE from 0.4361 to 0.0991 and increases EVR@2 from 0.731 to 0.769, while the oracle mixer further reaches 0.0475 MSE and 0.815 EVR@2. This suggests that the channel mixer acts as a lightweight preconditioning step, reorganizing channel information before correction so that the subsequent adapters face a smaller residual target that is more concentrated along the channel dimension.

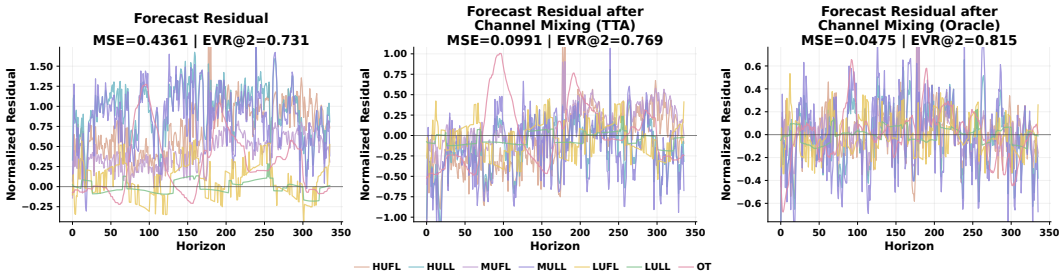


Figure A.1: **Effect of Channel Mixing on Residual Structure.** Visualization of forecast residuals for the same test window, from left to right: the original forecast residual, the residual after applying the learned channel mixer adapted during TSF-TTA, and the residual after applying an oracle channel mixer optimized for that window. MSE and EVR@2 are reported in each panel.

A.2 Effect of the Channel-Wise LoRA Adapter.

We refer to the full model using both the shared branch and the channel-wise LoRA branch as DBCM, and to its variant without the channel-wise LoRA branch as the shared-only variant. Figure A.2 shows the channel-wise relative MSE improvement over the frozen iTransformer for the shared-only variant and DBCM. The Overall column reports the relative MSE improvement computed over all channels, where DBCM consistently outperforms the shared-only variant across datasets. Across datasets, DBCM consistently exhibits more favorable channel-wise improvements than the shared-only variant. In most channels, adding the channel-wise LoRA branch increases the improvement over the source forecaster. More importantly, for channels where the shared-only variant yields negative relative improvement, DBCM substantially mitigates the degradation and even turns negative improvement into positive improvement in some cases. This indicates that a purely shared correction can be insufficient or occasionally misaligned for channels whose residual variations deviate from the shared structure, and that the channel-wise branch complements it. The channel-wise LoRA branch provides lightweight channel-specific flexibility, allowing DBCM to compensate for these remaining variations without relying on fully independent channel-wise adapters.

A.3 Qualitative Analysis.

Figure A.3 presents qualitative forecasting examples under test-time adaptation with iTransformer as the source forecaster. Across datasets, DBCM produces forecasts that are closer to the ground truth than competing methods, although the observed improvements vary across temporal patterns. For ETTh1 and ETTm1, DBCM better captures local peaks and follows the overall shape of the ground truth. In ETTh2, DBCM more accurately matches the scale of the target series, reducing the amplitude mismatch observed in other methods. For the Exchange Rate dataset, where the

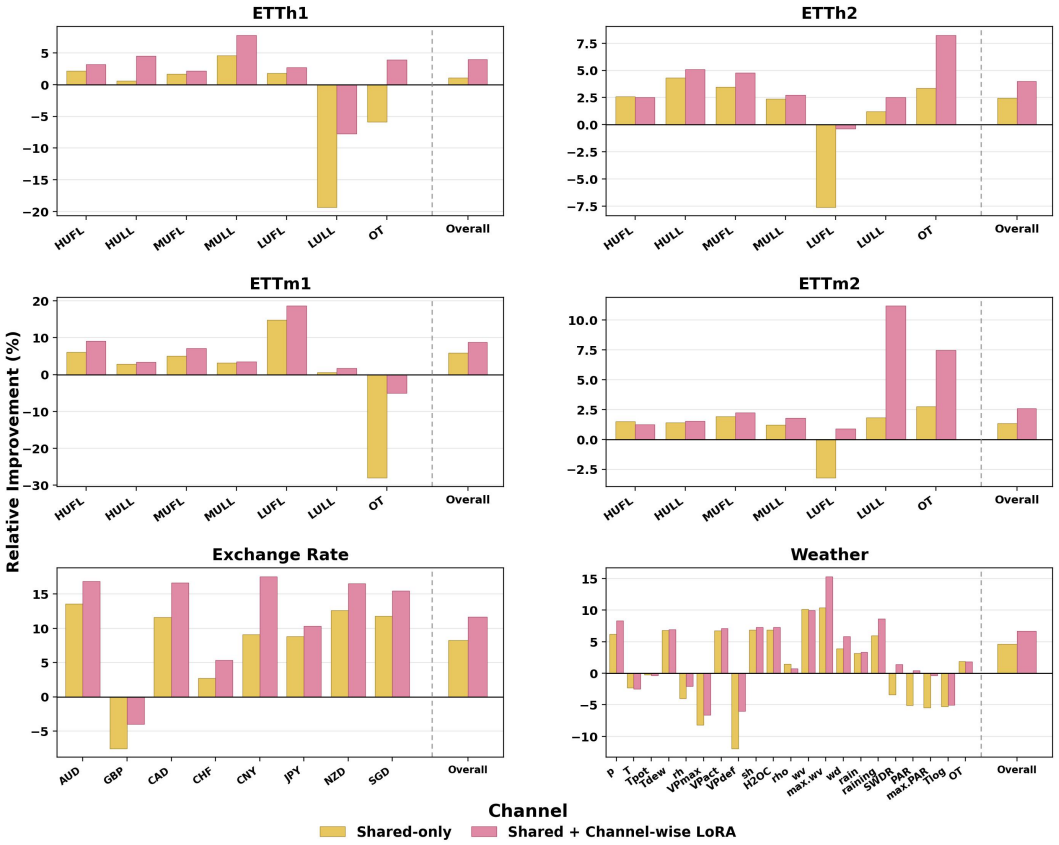


Figure A.2: **Channel-Wise Relative Improvement.** Channel-wise relative MSE improvement over the iTransformer, comparing the shared-only DBCM variant with the full DBCM using both shared and channel-wise LoRA branches. The overall column reports the relative MSE improvement over all channels.

baseline and competing methods struggle to capture the trend, DBCM produces predictions that more closely align with the ground truth. These examples show that DBCM improves qualitative forecasting behavior across different types of test-time shifts, including local peaks, scale changes, and trend shifts.

A.4 Raw Data Drift Spectra.

We examine raw data-side drift through channel-wise descriptors, defined as summary statistics that characterize changes across time-series windows. We consider two reference choices to examine whether the channel-space structure of raw drift depends on how the drift is measured. The source reference compares each test-window descriptor with the mean descriptor computed over the training windows. The previous-window reference instead compares each descriptor with that from the immediately preceding test window.

We split the descriptors into two families to distinguish simple statistical shift from changes in temporal structure. Low-order descriptors capture changes in basic window-level statistics, including mean, variance, and slope. In contrast, frequency/temporal descriptors capture how the temporal dynamics change, including shifts in spectral energy, dominant periodicity, and autocorrelation. This separation allows us to examine how simple statistical shifts and richer temporal changes differently shape the channel-space structure of raw drift. Detailed definitions are provided in Appendix F.

As shown in Table A.1, source-referenced low-order drift is relatively concentrated in the channel space, achieving $\mathrm{EVR@2} = 0.854$ and $\mathrm{Rank@90} = 2.89$. This suggests that simple window-level statistical shifts can share common channel directions. However, this concentration weakens once frequency and temporal descriptors are included. For the combined descriptor matrix, $\mathrm{EVR@2}$ drops to 0.652 under the source reference and 0.596 under the previous-window reference, while

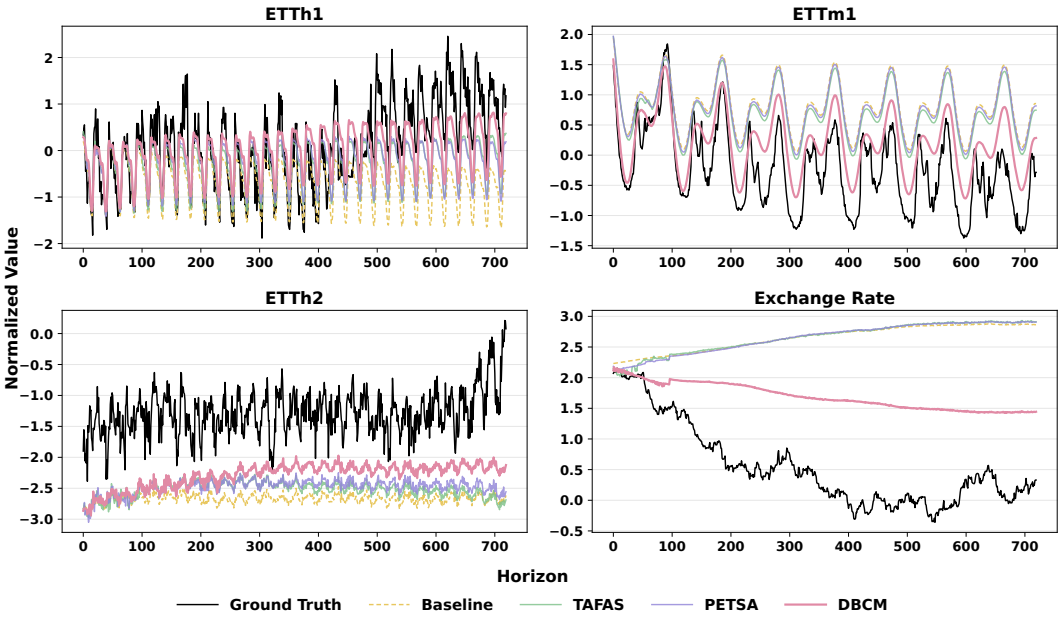


Figure A.3: **Qualitative Analysis.** Visualization of forecasting examples on ETTh1, ETTm1, ETTh2, and Exchange Rate using iTransformer as the source forecaster. The figure compares the frozen source forecaster and TSF-TTA methods, including TAFAS, PETSA, and DBCM.

Rank@90 increases to 5.17 and 5.67, respectively. Thus, the compactness observed in low-order drift does not extend to raw drift as a whole. In other words, raw drift becomes substantially more heterogeneous once it is characterized beyond simple mean, variance, and trend shifts. Since TSF-TTA is not optimized to model raw input drift directly, raw data-side drift provides only indirect evidence for adapter design. We therefore turn to forecast residuals, which serve as the direct supervision signal for TSF-TTA.

Table A.1: **Raw Data Drift Spectra.** EVR@k: top-k explained variance ratio; Rank@90: minimum rank explaining 90% variance.

Signal	EVR@2	Rank@90
Source, low-order desc.	0.854	2.89
Source, freq./temporal desc.	0.728	4.11
Source, all desc.	0.652	5.17
Previous, all desc.	0.596	5.67

A.5 Dataset-Wise Residual–Correction Alignment.

Table A.2 compares how well the corrections generated by adapters align with the channel-space subspaces of forecast residuals. We measure the projection similarity between the top-2 subspaces of residuals and corrections for each dataset. DBCM achieves the highest average alignment score of 0.774, outperforming TAFAS, TAFAS-Shared, and PETSA, and attains the highest alignment on all six datasets. The improvement is especially pronounced on Weather, which has the largest number of channels, where the corrections of DBCM remain closely aligned with the leading residual subspace. Moreover, TAFAS-Shared attains higher alignment than TAFAS on average and on five of the six datasets, further supporting that residual corrections are not fully channel-independent and can be modeled through shared channel-space structure.

A.6 Residual Spectrum on High-Dimensional Data.

Since parameter sharing becomes more relevant as the number of channels grows, we further examine whether the channel-space concentration of forecast residuals persists on the high-dimensional Electricity dataset with 321 channels. Table A.3 compares Electricity with Weather, the dataset with the most channels in Table 1, using the same centered channel-space metrics. On Electricity, the

Table A.2: **Dataset-Wise Residual–Correction Alignment.** Projection similarity between the top-2 channel subspaces of forecast residuals and adapter corrections.

Dataset	TAFAS	TAFAS-Shared	PETSA	DBCM
ETTh1	0.772	0.812	0.649	0.874
ETTh2	0.648	0.676	0.679	0.684
ETTm1	0.876	0.893	0.824	0.897
ETTm2	0.707	0.858	0.787	0.875
Exchange	0.457	0.554	0.533	0.603
Weather	0.566	0.543	0.521	0.712
Average	0.671	0.723	0.666	0.774

top 30% of channel directions explain 83.2% of the residual variance, whereas isotropic residuals would yield only 30.2%, and 90% of the variance is explained by 45.3% of the channel directions, compared with 90.0% for isotropic residuals. Although the concentration is weaker than on Weather, the residuals remain far from isotropic, indicating that the shared channel-space structure persists in high-dimensional settings.

Table A.3: **Residual Spectrum on High-Dimensional Data.** Centered channel-space metrics of forecast residuals, reported in %.

Dataset	C	EVR@30%	Rank@90/C
Weather	21	98.2	27.4
Electricity	321	83.2	45.3

B ALGORITHM OF DBCM

Algorithm 1: PyTorch-style pseudocode for DBCM

```
# L, H: lookback window length and forecasting horizon
# forecaster: frozen source forecaster
# C_in, C_out: input- and output-side DBCM

forecaster.requires_grad_(False)
C_in, C_out = Calibration(L), Calibration(H)

test_x, test_y, period = [], [], None

# test_gt: H future steps, used only once they are revealed
for test_input, test_gt in test_loader:
    # start a new test mini-batch
    if period is None:
        period = PAAS(test_input)

    test_x.append(test_input)
    test_y.append(test_gt)
    # wait until period + 1 windows have arrived
    if len(test_x) < period + 1:
        continue

    test_x = torch.stack(test_x)
    test_y = torch.stack(test_y)

    # (1) update with partially observed ground truth
    pred_cali = C_out(forecaster(C_in(test_x)))
    loss_p = compute_partial_loss(pred_cali, test_y, period)
    optimizer.zero_grad()
    loss_p.backward()
    optimizer.step()

    # (2) update with fully observed ground truth
```

```
    if full_gt_available:
        pred_full = C_out(forecaster(C_in(full_input)))
        loss_f = MSE(pred_full, full_gt)
        optimizer.zero_grad()
        loss_f.backward()
        optimizer.step()

    # (3) adjust predictions for steps not yet observed
    with torch.no_grad():
        pred_adj = C_out(forecaster(C_in(test_x)))
        for i in range(period):
            pred_cali[i, period-i:] = pred_adj[i, period-i:]

    test_x, test_y, period = [], [], None

class Calibration(nn.Module):
    # Z: (T, C), where T = L (input) or H (output)
    def forward(self, Z):
        # linear channel mixer
        Z_mix = Z + Z @ W_mix

        # shared adapter + channel-wise LoRA adapter
        A_sh = W_sh @ Z_mix
        A_cw = stack([(U[c] @ V[c]) @ Z_mix[:, c]
                      for c in range(C)], dim=-1)
        C_base = A_sh + A_cw + b

        # nonlinear gated modulation
        G = sigmoid(W_gate @ Z_mix + b_gate)
        C_gate = C_base + C_base * G

        # residual path with channel-wise outer gate
        return Z + tanh(g) * C_gate

# test_input: (L, C)
def PAAS(test_input):
    x = test_input - test_input.mean(dim=0)
    amplitude = abs(fft(x))
    # dominant frequency of the highest-energy channel
    ch = (amplitude ** 2).sum(dim=0).argmax()
    freq = amplitude[:, ch].argmax()
    return x.shape[0] // freq

def compute_partial_loss(pred_cali, test_y, period):
    B, H = pred_cali.shape[:2]
    # index of the latest window in the batch
    p = min(period, B - 1)
    loss, valid_count = 0.0, 0
    for i in range(p):
        # steps of window i already revealed
        obs_len = min(p - i, H)
        pred_i = pred_cali[i, :obs_len]
        gt_i = test_y[i, :obs_len]
        loss += squared_error(pred_i, gt_i).sum()
        valid_count += pred_i.numel()
    return loss / valid_count
```

C Additional Experiments

C.1 Full Results on Electricity and TSFMs

Table C.1 reports the results on the Electricity dataset, which contains 321 channels, substantially more than the six main benchmarks. DBCM achieves the best performance across all backbones

and prediction horizons, and the improvement becomes more pronounced at longer horizons. For iTransformer at $H = 720$, TAFAS and PETSA improve the baseline by only 2.33% and 0.83%, respectively, whereas DBCM achieves a larger gain of 5.45%. A similar trend is observed at $H = 336$, where TAFAS and PETSA yield improvements of 2.61% and 0.74%, whereas DBCM achieves 3.34%. These results indicate that existing methods become less effective as the forecasting horizon increases, while DBCM continues to deliver stable improvements in high-dimensional multivariate settings.

Table C.2 reports the per-dataset results on Moirai (Woo et al., 2024), Chronos (Ansari et al., 2024), and Chronos-2 (Ansari et al., 2025) at $H = 96$. DBCM achieves the best performance on all four ETT datasets for all three TSFMs, outperforming both TAFAS and PETSA. The gains are particularly large on ETTm1, where the frozen TSFMs exhibit large errors: DBCM reduces the MSE of Moirai from 1.0543 to 0.4010 and that of Chronos from 1.5167 to 0.3862, compared with 0.4720 and 0.4252 for TAFAS and 0.4708 and 0.4505 for PETSA. These results indicate that the proposed adaptation mechanism is not tied to a specific foundation model family, but can complement diverse pre-trained TSFMs by calibrating their predictions under test-time distribution shifts.

Table C.1: **Prediction Performance Comparison on the Electricity Dataset.** Test MSE is reported for iTransformer, DLinear and FreTS across four prediction horizons.

	iTransformer				**DLinear**				**FreTS**			
	Baseline	TAFAS	PETSA	DBCM	Baseline	TAFAS	PETSA	DBCM	Baseline	TAFAS	PETSA	DBCM
96	.1452	.1426	.1446	**.1415**	.1974	.1897	.1924	**.1844**	.1745	.1682	.1704	**.1663**
192	.1692	.1588	.1612	**.1584**	.1971	.1907	.1930	**.1886**	.1814	.1767	.1782	**.1766**
336	.1765	.1719	.1752	**.1706**	.2117	.2059	.2089	**.2051**	.1973	.1931	.1957	**.1923**
720	.2402	.2346	.2382	**.2271**	.2534	.2508	.2530	**.2471**	.2380	.2363	.2379	**.2345**
Avg.	.1828	.1770	.1798	**.1744**	.2149	.2093	.2118	**.2063**	.1978	.1936	.1956	**.1924**

Table C.2: **TSF-TTA on TSFMs.** Test MSE of TSF-TTA methods evaluated on Moirai, Chronos, and Chronos-2 for horizon $H = 96$.

	ETTh1	ETTh2	ETTm1	ETTm2
Moirai	.4917	.2650	1.0543	.2044
+ TAFAS	.4740	.2557	.4720	.1862
+ PETSA	.4816	.2628	.4708	.1875
+ DBCM	**.4629**	**.2524**	**.4010**	**.1788**
Chronos	.5316	.2553	1.5167	.2242
+ TAFAS	.4811	.2467	.4252	.1866
+ PETSA	.5147	.2504	.4505	.1861
+ DBCM	**.4602**	**.2461**	**.3862**	**.1779**
Chronos-2	.4904	.2485	.6544	.1840
+ TAFAS	.4643	.2408	.4109	.1698
+ PETSA	.4687	.2419	.4148	.1686
+ DBCM	**.4540**	**.2385**	**.3772**	**.1657**

C.2 Comparison with DynaTTA

We report the complete prediction performance comparison with DynaTTA (Grover & Etemad, 2025). DynaTTA involves 15 hyperparameters; for comparison, we follow the configuration used in COSA (Im & Kwon, 2026). We also note that the official DynaTTA implementation has reproducibility issues. As shown in Table C.3, DynaTTA exhibits unstable adaptation behavior across datasets and backbone models, often increasing the forecasting error compared to the frozen baseline. In contrast, DBCM consistently achieves the best performance across all evaluated settings. These results suggest that dynamically adjusting adaptation behavior through many hyperparameters can be difficult to stabilize, while DBCM provides more reliable test-time adaptation across diverse forecasting architectures and distribution shifts.

C.3 Comparison with COSA

DBCM follows the TSF-TTA protocol of TAFAS (Kim et al., 2025) and PETSA (Medeiros et al., 2025). Under this protocol, the adapter is updated only with observations available at the current

Table C.3: **Prediction Performance Comparison with DynaTTA.** Test MSE is reported for three backbone models on six datasets with four prediction horizons. Bold entries indicate the best performance among methods, while underlined entries denote the second-best.

		iTransformer			DLinear			FreTS		
		Baseline	DynaTTA	DBCM	Baseline	DynaTTA	DBCM	Baseline	DynaTTA	DBCM
ETTh1	96	.4349	.5253	**.4228**	.4510	.4632	**.4318**	.4360	.5199	**.4242**
	192	.4937	.5608	**.4805**	.5042	.5161	**.4835**	.4886	.5505	**.4764**
	336	.5406	.5979	**.5322**	.5508	.5687	**.5373**	.5466	.6277	**.5354**
	720	.6914	.7576	**.6298**	.7019	.7080	**.6487**	.7145	.9315	**.6565**
ETTh2	96	.2430	.3498	**.2388**	.2294	.2305	**.2268**	.2376	.2523	**.2347**
	192	.2973	.3669	**.2914**	.2835	.2871	**.2794**	.2838	.3111	**.2761**
	336	.3448	.4136	**.3301**	.3250	.3370	**.3152**	.3184	.3607	**.3048**
	720	.4258	.4900	**.3939**	.4149	.4361	**.3820**	.3951	.4225	**.3653**
ETTm1	96	.3763	.8006	**.3415**	.3705	.3910	**.3420**	.3618	.4564	**.3420**
	192	.4371	.8724	**.4038**	.4420	.4995	**.4056**	.4304	.5729	**.4060**
	336	.5029	.9139	**.4628**	.5172	.5983	**.4626**	.4928	.6636	**.4618**
	720	.5713	.7554	**.5276**	.5927	.6570	**.5321**	.5609	.7543	**.5274**
ETTm2	96	.1555	.1903	**.1536**	.1589	.1632	**.1573**	.1563	.2023	**.1536**
	192	.1938	.2237	**.1923**	.1929	.2067	**.1907**	.1914	.2117	**.1880**
	336	.2345	.3038	**.2312**	.2325	.2970	**.2289**	.2310	.2986	**.2259**
	720	.3101	.4069	**.2990**	.3061	.3827	**.2956**	.3026	.4319	**.2930**
Exchange	96	.0916	.1097	**.0858**	.0811	.0842	**.0789**	.0827	.1472	**.0794**
	192	.1914	.2310	**.1770**	.1708	.1846	**.1626**	.1787	.2558	**.1659**
	336	.3375	.3100	**.2844**	.3205	.2965	**.2642**	.3265	.3885	**.2659**
	720	.9433	.9968	**.8182**	.8383	.8381	**.7066**	.8251	.8242	**.6700**
Weather	96	.1719	.2287	**.1630**	.1936	.1839	**.1732**	.1809	.1903	**.1677**
	192	.2233	.3659	**.2096**	.2383	.2963	**.2194**	.2266	.3167	**.2101**
	336	.2800	.6478	**.2608**	.2913	.5204	**.2697**	.2806	.3872	**.2602**
	720	.3576	.5251	**.3360**	.3643	.5951	**.3391**	.3557	.5414	**.3338**
Average		.3687	.4977	**.3444**	.3655	.4059	**.3389**	.3585	.4425	**.3343**
Avg. Rank		2.04	2.96	**1.00**	2.12	2.88	**1.00**	2.04	2.96	**1.00**

test time. Specifically, adaptation uses partially observed ground truth for the current prediction and, when available, fully observed ground truth from past buffered predictions. COSA (Im & Kwon, 2026), in contrast, adopts a different adaptation protocol: it first evaluates a test batch and then directly updates the adapter using the ground truth of the same evaluated batch, regardless of whether those target values have been observed at the corresponding test time. This protocol can expose the adapter to future target values that would not be available in a streaming forecasting scenario, leading to label leakage.

To directly compare DBCM with COSA under the same update strategy, we additionally apply the COSA protocol to DBCM and report the results in Table C.4. Although this protocol is not used in our main experiments due to the aforementioned leakage issue, this comparison isolates the effect of the adaptation module from differences in the update protocol. COSA provides two variants: COSA-F updates the adapter at a fixed interval, using 48 time steps as the default, while COSA-P uses PAAS from TAFAS (Kim et al., 2025) to determine the update interval from the periodicity of the input sequence. Following TAFAS, DBCM also uses PAAS by default to construct test mini-batches for adaptation. Under COSA's protocol, DBCM achieves the best average performance across all three evaluated backbones. These results indicate that, even when evaluated under the same update strategy as COSA, DBCM provides more effective correction on average, suggesting that its dual-branch correction structure is beneficial beyond differences in the test-time adaptation protocol.

C.4 Comparison with DynaME

TSF-TTA and online learning for time series forecasting may appear similar, as both exploit sequentially arriving observations after deployment. However, they differ in their adaptation objectives and deployment constraints. Online learning broadly aims to continuously update the forecasting system itself from streaming data, which may involve modifying the backbone, prediction head, memory, or ensemble weights. In contrast, TSF-TTA assumes a pre-trained source forecaster whose learned temporal knowledge should be preserved, and restricts test-time updates to lightweight aux-

Table C.4: **Prediction Performance Comparison with COSA.** Test MSE obtained by applying the update protocol used in COSA. Bold entries indicate the best performance among methods, while underlined entries denote the second-best.

		iTransformer				DLinear				FreTS			
		Baseline	COSA-F	COSA-P	DBCM	Baseline	COSA-F	COSA-P	DBCM	Baseline	COSA-F	COSA-P	DBCM
ETTh1	96	.4349	.4325	.4326	**.4272**	.4510	.4437	.4433	**.4354**	.4360	.4324	.4326	**.4261**
	192	.4937	.4920	.4913	**.4790**	.5042	.4977	.4997	**.4867**	.4886	.4875	.4877	**.4769**
	336	.5406	.5388	.5322	**.5177**	.5508	.5458	.5420	**.5274**	.5466	.5433	.5383	**.5213**
	720	.6914	.6452	.6229	**.5738**	.7019	.6635	.6425	**.5853**	.7145	.6768	.6573	**.6054**
ETTh2	96	.2430	.2418	.2408	**.2391**	.2294	.2285	.2281	**.2273**	.2376	.2371	.2364	**.2359**
	192	.2973	.2955	**.2953**	.3019	.2835	**.2826**	**.2826**	.2891	.2838	.2829	**.2806**	.2851
	336	.3448	.3429	.3422	**.3335**	.3250	.3266	.3267	**.3178**	.3184	.3172	.3173	**.3066**
	720	.4258	.4053	.4169	**.3818**	.4149	.3945	.4059	**.3726**	.3951	.3745	.3839	**.3542**
ETTm1	96	.3763	.3535	.3589	**.3404**	.3705	.3486	.3487	**.3428**	.3618	.3523	.3532	**.3428**
	192	.4371	.4184	.4211	**.4058**	.4420	.4178	.4165	**.4085**	.4304	.4199	.4193	**.4094**
	336	.5029	.4850	.4863	**.4676**	.5172	.4877	.4853	**.4718**	.4928	.4831	.4819	**.4655**
	720	.5713	.5562	.5601	**.5299**	.5927	.5581	.5632	**.5370**	.5609	.5505	.5532	**.5285**
ETTm2	96	.1555	.1550	.1551	**.1539**	.1589	.1589	.1588	**.1578**	.1563	.1556	.1556	**.1544**
	192	**.1938**	.1948	.1941	.1958	**.1929**	.1941	**.1929**	.1936	.1914	.1928	**.1913**	.1914
	336	.2345	.2357	.2353	**.2344**	**.2325**	.2350	.2327	.2365	.2310	.2321	.2306	**.2292**
	720	.3101	**.3005**	.3098	.3296	.3061	**.2965**	.3042	.3480	.3026	**.2922**	.3000	.3315
Exchange	96	.0916	.0896	.0905	**.0852**	.0811	.0800	.0806	**.0786**	.0827	.0810	.0820	**.0788**
	192	.1914	.1813	.1881	**.1730**	.1708	.1640	.1689	**.1605**	.1787	.1705	.1760	**.1644**
	336	.3375	.3141	.3298	**.2997**	.3205	.2977	.3129	**.2809**	.3265	.3031	.3185	**.2801**
	720	.9433	.7529	.8907	**.5979**	.8383	.6512	.7904	**.6170**	.8251	.6430	.7788	**.6026**
Weather	96	.1719	**.1653**	.1676	.1682	.1936	.1860	.1883	**.1827**	.1809	.1763	.1768	**.1708**
	192	.2233	.2158	.2174	**.2143**	.2383	.2283	.2296	**.2255**	.2266	.2196	.2202	**.2126**
	336	.2800	.2720	.2718	**.2704**	.2913	.2808	.2805	**.2762**	.2806	.2721	.2718	**.2671**
	720	.3576	**.3323**	.3426	.3437	.3643	.3391	.3480	**.3382**	.3557	**.3327**	.3412	.3423
Average		.3687	.3507	.3581	**.3360**	.3655	.3461	.3530	**.3374**	.3585	.3429	.3494	**.3326**
Avg. Rank		3.71	2.25	2.50	**1.54**	3.58	2.46	2.50	**1.46**	3.81	2.35	2.44	**1.40**

iliary modules that calibrate predictions under distribution shifts using delayed or partially observed feedback. DynaME (Hong et al., 2026) is one of the state-of-the-art methods in the online learning literature for time series forecasting. We compare our DBCM with DynaME using iTransformer as the source forecaster. In DBCM, we freeze the source forecaster, whereas DynaME updates the forecaster itself.

For a fair comparison, we follow the dataset splits used throughout this paper. As shown in Table C.5, DBCM achieves the lowest average MSE and the best average rank, reducing the average MSE from 0.3687 to 0.3444 over the baseline and from 0.3614 to 0.3444 over DynaME. Notably, DBCM consistently reduces the forecasting error over the baseline across all 24 dataset–horizon settings, whereas DynaME occasionally degrades the baseline performance, e.g., on ETTh1, ETTm2, and Exchange Rate. This suggests that directly updating the forecaster in an online manner can be sensitive to the target stream and may suffer from negative adaptation under certain shifts. In contrast, DBCM provides more stable improvements by preserving the source forecaster and adapting only the calibration modules. The advantage of DBCM is particularly clear at the longest horizon, where it achieves the best performance on all six datasets, indicating that structured calibration is especially effective under severe long-range distribution shifts.

C.5 Comparison with Normalization Methods

Normalization methods are widely used to mitigate non-stationarity in time series forecasting. RevIN (Kim et al., 2021) removes instance-specific statistics from the input sequence and restores them after forecasting, while DDN (Dai et al., 2024) performs dynamic normalization in both the time and frequency domains to better handle non-stationary temporal patterns. Since these methods are designed to alleviate distributional discrepancies in the source forecaster, we further examine whether TSF-TTA provides additional benefits when the source forecaster is equipped with such normalization modules. To this end, we compare TAFAS, PETSA, and DBCM under three settings: without an additional normalization module, with RevIN, and with DDN.

Figure C.1 reports the test MSE across prediction horizons, averaged over six source forecasters and six datasets. Across all three normalization settings, applying TSF-TTA consistently reduces the test MSE compared with the corresponding source forecaster without test-time adaptation. This demonstrates that TSF-TTA remains effective even when RevIN or DDN is employed, suggesting that test-time adaptation is complementary to normalization-based non-stationarity handling rather than merely replacing it. Among the compared methods, DBCM achieves the lowest test MSE

Table C.5: **Prediction Performance Comparison with DynaME.** Test MSE is reported for iTransformer on six datasets with four prediction horizons. Bold entries indicate the best performance among methods, while underlined entries denote the second-best.

		iTransformer		
		Baseline	DynaME	DBCM
ETTh1	96	.4349	.4370	**.4228**
	192	.4937	.4933	**.4805**
	336	.5406	.5509	**.5322**
	720	.6914	.7008	**.6298**
ETTh2	96	.2430	**.2379**	.2388
	192	.2973	.2934	**.2914**
	336	.3448	.3347	**.3301**
	720	.4258	.4289	**.3939**
ETTm1	96	.3763	**.3274**	.3415
	192	.4371	**.3941**	.4038
	336	.5029	**.4591**	.4628
	720	.5713	.5563	**.5276**
ETTm2	96	.1555	.1538	**.1536**
	192	.1938	.1944	**.1923**
	336	.2345	.2350	**.2312**
	720	.3101	.3093	**.2990**
Exchange	96	.0916	.0966	**.0858**
	192	.1914	.2004	**.1770**
	336	.3375	.3654	**.2844**
	720	.9433	.9265	**.8182**
Weather	96	.1719	**.1622**	.1630
	192	.2233	.2113	**.2096**
	336	.2800	.2657	**.2608**
	720	.3576	.3391	**.3360**
Average		.3687	.3614	**.3444**
Avg. Rank		2.63	2.17	**1.21**

across all prediction horizons and normalization settings. The absolute MSE reduction generally becomes more pronounced at longer horizons, particularly at $H = 720$, indicating that the proposed dual-branch calibration mechanism effectively corrects residual prediction errors that remain after normalization.

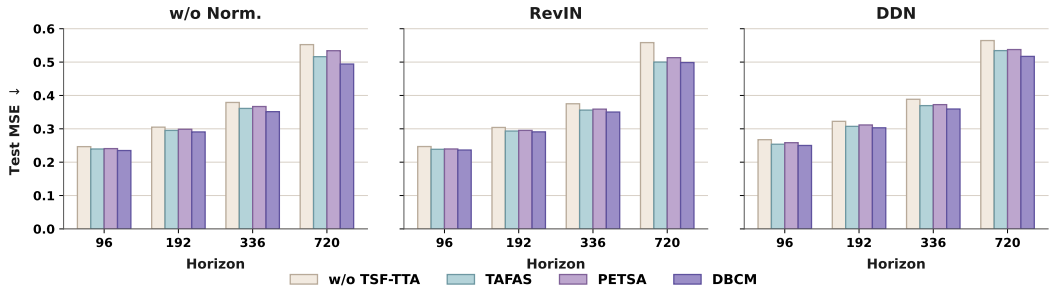


Figure C.1: **Prediction Performance under Different Normalization Settings.** Test MSE of the source forecaster without TSF-TTA and with TAFAS, PETSA, or DBCM across prediction horizons. The three panels correspond to settings without an additional normalization module (w/o Norm.), with RevIN, and with DDN, respectively. For each horizon, results are averaged over six source forecasters and six datasets. Lower values indicate better performance.

C.6 Evaluation on TTFBench

Standard TSF benchmarks often exhibit only mild distribution shifts in their original test splits, which may understate the effectiveness of test-time adaptation under severe deployment-time non-

stationarity. To provide a more challenging stress test, we additionally evaluate on TTFBench, a synthetic-shift benchmark for TSF-TTA. In this setting, the train and validation splits are kept unchanged, and the source forecaster is trained and selected using these original splits. Controlled distribution shifts are then injected only into the test split, so that the trained model is evaluated on test streams with harsher non-stationarity than those in the standard benchmark setting. The injected shifts include piecewise polynomial trends, sinusoidal seasonal components, discrete regime changes, and Gaussian noise with varying intensities. For reproducibility, we use the TTFBench dataset generated with seed 0.

Table C.6 reports the forecasting performance on TTFBench. Overall, DBCM consistently improves robustness under synthesized test-time shifts. For DLinear and FreTS, DBCM achieves both the lowest average MSE and the best average rank, reducing the average MSE from 0.4068 to 0.3825 and from 0.3999 to 0.3782, respectively. For iTransformer, DBCM obtains the best average rank of 1.42, although its average MSE is slightly higher than that of TAFAS. This gap is mainly driven by the Exchange Rate results at the longest prediction horizon, where TAFAS achieves a substantially lower MSE. In contrast, the average-rank result indicates that DBCM achieves more consistently strong performance across all dataset–horizon pairs. These results demonstrate that DBCM provides stable and effective test-time adaptation under harsher synthesized distribution shifts.

Table C.6: **Prediction Performance Comparison in TTFBench.** Test MSE evaluated on TTFBench is reported for six models on six datasets with four prediction horizons. Bold entries indicate the best performance among methods, while underlined entries denote the second-best.

		iTransformer				**DLinear**				**FreTS**			
		Baseline	TAFAS	PETSA	DBCM	Baseline	TAFAS	PETSA	DBCM	Baseline	TAFAS	PETSA	DBCM
ETTh1	96	.4385	.4380	.4372	**.4365**	.4540	.4509	.4519	**.4486**	.4402	.4378	.4381	**.4364**
	192	.5053	.5042	.5048	**.5033**	.5146	.5114	.5129	**.5095**	.5019	.5001	.5002	**.4990**
	336	.5689	.5686	.5690	**.5678**	.5783	.5771	.5768	**.5749**	.5753	.5738	.5738	**.5734**
	720	.7714	**.7589**	.7711	.7644	.7765	**.7606**	.7758	.7694	.7898	**.7771**	.7893	.7831
ETTh2	96	.2476	.2463	.2459	**.2458**	.2348	**.2342**	.2346	.2343	.2445	.2430	**.2418**	**.2418**
	192	.3132	.3114	.3123	**.3110**	.3007	.3000	.3008	**.2999**	.3006	.2971	**.2965**	.2977
	336	.3905	**.3818**	.3890	.3824	.3737	.3706	.3740	**.3691**	.3674	.3627	.3673	**.3616**
	720	.5599	.5508	.5549	**.5365**	.5492	.5409	.5448	**.5249**	.5355	.5285	.5319	**.5255**
ETTm1	96	.3762	.3493	.3507	**.3476**	.3708	.3545	.3567	**.3495**	.3623	.3566	.3565	**.3477**
	192	.4376	.4191	.4181	**.4130**	.4429	.4239	.4258	**.4186**	.4317	.4237	.4219	**.4155**
	336	.5050	.4828	.4815	**.4771**	.5195	.4917	.4921	**.4849**	.4959	.4841	.4811	**.4766**
	720	.5800	.5656	.5630	**.5614**	.6010	.5706	.5697	**.5668**	.5709	.5579	.5540	**.5493**
ETTm2	96	.1558	.1547	.1548	**.1543**	.1590	.1585	.1585	**.1582**	.1566	.1557	.1552	**.1546**
	192	.1945	**.1938**	.1941	.1940	.1935	.1927	.1929	**.1925**	.1922	.1913	.1914	**.1907**
	336	.2363	.2354	**.2351**	.2357	.2342	.2327	.2327	**.2325**	.2330	.2310	.2307	**.2304**
	720	.3161	.3105	**.3084**	.3086	.3121	.3069	.3055	**.3046**	.3093	.3035	**.3031**	.3033
Exchange	96	.1056	.1006	**.0993**	.1001	.0922	.0901	**.0888**	.0902	.0918	.0883	**.0871**	.0881
	192	.2314	**.2209**	.2226	.2219	.2057	.2009	.1998	**.1974**	.2123	.2050	.2028	**.1986**
	336	.4071	.4013	.4055	**.3592**	.3969	.3894	.3845	**.3313**	.4003	.3920	.3898	**.3368**
	720	1.1764	**.9046**	1.1078	1.1206	1.0824	.8670	1.0187	**.8603**	1.0594	1.0301	.9844	**.8298**
Weather	96	.2261	.2152	.2178	**.2144**	.2493	.2346	.2343	**.2239**	.2360	.2241	.2256	**.2193**
	192	.2894	.2767	.2793	**.2720**	.3056	.2883	.2929	**.2811**	.2944	.2794	.2838	**.2738**
	336	.3534	.3332	.3334	**.3281**	.3649	.3454	.3472	**.3379**	.3541	.3353	.3371	**.3297**
	720	.4457	.4235	.4257	**.4159**	.4520	.4300	.4339	**.4210**	.4428	.4234	.4285	**.4151**
Average		.4097	**.3895**	.3992	.3946	.4068	.3885	.3961	**.3825**	.3999	.3917	.3905	**.3782**
Avg. Rank		3.96	2.08	2.54	**1.42**	3.92	2.21	2.71	**1.17**	4.00	2.48	2.29	**1.23**

C.7 Computational Efficiency

Table C.7 summarizes the computational efficiency of different TSF-TTA methods using iTransformer as the backbone forecaster model. Overall, the three methods show broadly comparable computational costs in terms of memory usage, throughput, adaptation time, and inference time, suggesting that test-time adaptation adds only modest practical overhead. These results are averaged over all six benchmark datasets and four forecasting horizons. Although DBCM requires a slightly longer adaptation time, the difference is only 0.76 ms compared with PETSA and 2.30 ms compared with TAFAS, while its inference time remains comparable and its throughput is the highest. Moreover, since DBCM is updated only at the adaptation points determined by PAAS (Kim et al., 2025) rather than at every time step, this millisecond-level overhead is unlikely to become a practical bottleneck, given that the benchmark sampling intervals are several minutes or longer. The more meaningful differences arise in the trade-off between parameter efficiency and predictive performance.

PETSA is the most parameter-efficient method, with the smallest parameter ratio, but its predictive performance is limited, resulting in the highest average MSE among the compared methods. Compared with PETSA, TAFAS achieves better predictive performance, but at the cost of a substantially larger parameter ratio. In contrast, our method achieves the best predictive performance while maintaining a substantially smaller parameter ratio than TAFAS. Since the remaining efficiency metrics stay largely comparable across methods, these results indicate that our method provides a more favorable performance–efficiency trade-off, especially when parameter efficiency is taken into account.

Table C.7: **Computational Efficiency Comparison.** Summary of adapter parameter ratio, memory usage, throughput, adaptation time, inference time, and average MSE across TSF-TTA methods. Param. Ratio denotes the number of additional adapter parameters relative to the number of parameters of the source forecaster.

Method	Param. Ratio (%)	Peak mem (MB)	Samples/sec	Adapt. time (ms)	Infer. time (ms)	Average MSE
TAFAS	16.00	512.04	12022.58	12.93	8.30	0.3531
PETSA	1.14	491.04	11814.63	14.47	8.89	0.3556
DBCM	5.63	492.78	12339.85	15.23	8.77	0.3444

C.8 Computational Efficiency in Large-Scale Settings

Since the channel-wise LoRA branch and the channel mixer scale with the number of channels, we further evaluate the computational efficiency on the high-dimensional Electricity dataset and on Chronos (Ansari et al., 2024). Tables C.8 and C.9 report the results, where the parameter ratio is defined as in Table C.7. The Electricity results are averaged over the four forecasting horizons with iTransformer as the source forecaster, and the Chronos results are averaged over the four ETT datasets with three random seeds.

On Electricity, which contains 321 channels, DBCM achieves the lowest adaptation and inference latency and the highest throughput, while also attaining the lowest MSE. Its parameter ratio is substantially smaller than that of TAFAS and only slightly higher than that of PETSA, while its peak memory usage remains similar to that of TAFAS. On Chronos, the adapter parameters account for less than 0.1% of the forecaster parameters for all methods, and the computational cost is dominated by the frozen foundation model. Accordingly, DBCM exhibits memory usage and latency comparable to those of TAFAS, lower latency than PETSA, and the highest throughput, while achieving the lowest MSE.

Regarding the scalability with respect to the number of channels C, the channel-wise LoRA branch scales linearly with C, as in PETSA, while the shared adapter and the nonlinear gate are shared across channels. In contrast, TAFAS assigns a full linear adapter to every channel. Overall, the number of trainable parameters of DBCM scales as $\mathcal{O}(L^2 + H^2 + Cr(L + H) + C^2)$, where the C^2 term stems from the channel mixer, which remains substantially smaller than the $\mathcal{O}(C(L^2 + H^2))$ parameters of TAFAS whenever C is smaller than $L^2 + H^2$, as in all evaluated settings. Although this term increases the parameter count as C grows, it does not noticeably increase the other computational costs: on Electricity with 321 channels, DBCM still achieves the lowest latency and the highest throughput with memory usage comparable to TAFAS (Table C.8). Together with Table C.7, these results indicate that DBCM does not incur a consistent latency disadvantage across settings.

Table C.8: **Computational Efficiency on Electricity.** Results are averaged over the four forecasting horizons for the iTransformer forecaster.

Method	Param. Ratio (%)	Peak mem (MB)	Samples/sec	Adapt. time (ms)	Infer. time (ms)	Average MSE
TAFAS	74.98	2109.64	2594.97	28.64	10.05	0.1770
PETSA	25.04	1672.08	2567.59	26.77	10.28	0.1799
DBCM	28.80	2143.44	2710.95	26.09	9.34	0.1744

Table C.9: **Computational Efficiency on Chronos.** Results are averaged over ETTh1, ETTh2, ETTm1, and ETTm2 with three random seeds.

Method	Param. Ratio (%)	Peak mem (MB)	Samples/sec	Adapt. time (ms)	Infer. time (ms)	Average MSE
TAFAS	0.0635	4604.39	669.64	163.85	90.68	0.3349
PETSA	0.0126	4603.04	664.87	172.88	93.09	0.3504
DBCM	0.0286	4604.26	670.41	164.92	90.91	0.3176

C.9 Ablation of Input- and Output-Side DBCM

We investigate the effect of applying DBCM to the input and output sides of the forecaster. Following the perspective of TAFAS (Kim et al., 2025), the input-side DBCM aims to calibrate distribution-shifted test inputs toward the input distribution that the pre-trained forecaster was trained to handle, whereas the output-side DBCM directly adjusts the forecaster's predictions toward the continuously evolving test distribution. To assess their individual and combined contributions, we compare three variants: applying DBCM only to the input side, applying DBCM only to the output side, and applying DBCM to both sides.

Table C.10 shows that both input-side and output-side DBCM consistently improve over the frozen baseline across backbone models. The output-side DBCM generally provides stronger gains than the input-side variant, suggesting that directly calibrating forecast outputs is particularly effective for correcting test-time residual errors. However, the input-side DBCM is also beneficial, especially for DLinear, where it achieves a slightly better average MSE than the output-side variant. This indicates that input-side calibration can still play an important role by reducing the mismatch before the forecaster processes the test window.

Most importantly, combining input- and output-side DBCM achieves the best average performance for all three backbone models. These results suggest that the two adaptation sides are complementary: input-side DBCM mitigates distribution shift before forecasting, while output-side DBCM further corrects residual prediction errors after forecasting. Therefore, applying DBCM to both sides provides a more effective calibration strategy than using either side alone.

Table C.10: **Prediction Performance Comparison by Adaptation Side.** Test MSE is reported for three backbone models on six datasets with four prediction horizons. Bold entries indicate the best performance among methods, while underlined entries denote the second-best.

		iTransformer				**DLinear**				**FreTS**			
		Baseline	Input	Output	Both	Baseline	Input	Output	Both	Baseline	Input	Output	Both
ETTh1	96	.4349	.4326	**.4223**	.4228	.4510	.4379	**.4288**	.4318	.4360	.4315	**.4242**	**.4242**
	192	.4937	.4890	**.4805**	**.4805**	.5042	.4888	**.4828**	.4835	.4886	.4850	**.4760**	.4764
	336	.5406	**.5320**	.5326	.5322	.5508	.5405	**.5372**	.5373	.5466	.5405	**.5353**	.5354
	720	.6914	.6886	.6321	**.6298**	.7019	.6716	**.6484**	.6487	.7145	.7055	**.6565**	**.6565**
ETTh2	96	.2430	.2407	**.2378**	.2388	.2294	**.2260**	.2276	.2268	.2376	**.2346**	.2355	.2347
	192	.2973	.2938	**.2908**	.2914	.2835	**.2789**	.2801	.2794	.2838	.2768	.2762	**.2761**
	336	.3448	.3318	**.3299**	.3301	.3250	**.3123**	.3161	.3152	.3184	.3104	.3050	**.3048**
	720	.4258	.4179	**.3937**	.3939	.4149	.3897	.3826	**.3820**	.3951	.3846	.3654	**.3653**
ETTm1	96	.3763	.3542	.3462	**.3415**	.3705	.3428	.3422	**.3420**	.3618	.3526	.3432	**.3420**
	192	.4371	.4246	.4077	**.4038**	.4420	.4080	**.4044**	.4056	.4304	.4237	**.4059**	.4060
	336	.5029	.4906	.4640	**.4628**	.5172	.4664	**.4617**	.4626	.4928	.4908	.4625	**.4618**
	720	.5713	.5641	.5285	**.5276**	.5927	**.5288**	.5317	.5321	.5609	.5598	.5277	**.5274**
ETTm2	96	.1555	.1546	.1539	**.1536**	.1589	.1577	.1575	**.1573**	.1563	.1547	.1542	**.1536**
	192	.1938	.1930	**.1915**	.1923	.1929	.1901	**.1899**	.1907	.1914	.1905	.1883	**.1880**
	336	.2345	.2339	**.2309**	.2312	.2325	**.2274**	.2290	.2289	.2310	.2310	.2260	**.2259**
	720	.3101	.3088	**.2989**	.2990	.3061	**.2923**	.2962	.2956	.3026	.3030	**.2924**	.2930
Exchange	96	.0916	**.0854**	.0868	.0858	.0811	.0792	**.0787**	.0789	.0827	.0798	**.0789**	.0794
	192	.1914	.1799	**.1758**	.1770	.1708	.1633	.1628	**.1626**	.1787	.1664	.1664	**.1659**
	336	.3375	.3027	**.2826**	.2844	.3205	.2680	.2701	**.2642**	.3265	.2907	.2723	**.2659**
	720	.9433	.8283	.9424	**.8182**	.8383	.8016	.8378	**.7066**	.8251	.7853	.8247	**.6700**
Weather	96	.1719	.1697	**.1620**	.1630	.1936	.1745	.1797	**.1732**	.1809	.1699	.1716	**.1677**
	192	.2233	.2170	.2108	**.2096**	.2383	.2213	.2211	**.2194**	.2266	.2121	.2130	**.2101**
	336	.2800	.2733	.2611	**.2608**	.2913	.2703	.2702	**.2697**	.2806	.2646	.2612	**.2602**
	720	.3576	.3495	**.3348**	.3360	.3643	**.3305**	.3400	.3391	.3557	.3362	.3339	**.3338**
Average		.3687	.3565	.3499	**.3444**	.3655	.3445	.3449	**.3389**	.3585	.3492	.3415	**.3343**

C.10 Ablation on Channel Mixing

Table C.11 compares four channel mixing strategies for aggregating cross-channel information during adaptation. The linear channel mixer, used as the default in DBCM, applies a linear transformation along the channel dimension. MLP extends this design with a two-layer feed-forward network across channels at each time step, using a hidden dimension of 64 and GELU activation to enable learned nonlinear interactions. Attention represents each channel by its temporal window and computes input-dependent mixing weights between channels, with the projection dimension set to 64. Uniform performs averaging across channels and broadcasts the shared signal back to each channel. Overall, the performance differences across channel mixing strategies are small, indicating that DBCM is not highly sensitive to this design choice. While the MLP mixer obtains marginally lower MSE than Linear on ETTh2, ETTm2, and Exchange Rate, these gains are marginal and do not consistently transfer across all datasets. The attention-based mixer does not provide clear advantages over Linear, suggesting that input-dependent pairwise channel weighting may introduce unnecessary flexibility for the adaptation signals considered here. Uniform mixing remains competitive in several settings, including Electricity, but is generally less effective than Linear, indicating that simple channel averaging may be too restrictive to consistently capture useful cross-channel structure. These results support the use of a simple learnable linear mixer in DBCM, as it provides comparable performance without the additional complexity of nonlinear alternatives.

Table C.11: **Ablation on Channel Mixing.** Test MSE across datasets under different channel mixing strategies. Bold entries indicate the best performance among the compared variants.

	ETTh1	ETTh2	ETTm1	ETTm2	Exchange	Weather	Electricity
Linear	**.5163**	.3135	**.4339**	.2190	.3413	**.2424**	**.1744**
MLP	.5182	**.3124**	.4381	**.2182**	**.3410**	**.2424**	.1756
Attention	.5278	.3184	.4458	.2215	.3767	.2449	.1753
Uniform	.5212	.3146	.4431	.2213	.3726	.2453	.1746

C.11 Ablation on the Nonlinear Gating Function

Table C.12 compares different nonlinear gating functions used in DBCM. Overall, Sigmoid achieves the lowest MSE across all datasets, indicating that it provides the most stable gating behavior for the proposed calibration branch. ReLU, SiLU, and Tanh all lead to higher errors than Sigmoid, suggesting that alternative nonlinearities do not provide a consistent advantage in this setting. In particular, SiLU does not improve performance despite its smooth nonlinear form, while ReLU may be less suitable for modulating correction values due to its unbounded positive activation. Tanh achieves the best result on Electricity by a very small margin and performs better than ReLU or SiLU on some datasets, but it is generally less effective than Sigmoid across the remaining datasets. Based on these results, we use Sigmoid as the default nonlinear gating function in DBCM.

Table C.12: **Ablation on Nonlinear Gating Function.** Test MSE across datasets under different nonlinear gating functions. Bold entries indicate the best performance among the compared variants.

	ETTh1	ETTh2	ETTm1	ETTm2	Exchange	Weather	Electricity
Sigmoid	**.5163**	**.3135**	**.4339**	**.2190**	**.3413**	**.2424**	.1744
ReLU	.5278	.3148	.4435	.2204	.3692	.2458	.1756
SiLU	.5323	.3154	.4450	.2207	.3718	.2454	.1748
Tanh	.5200	.3150	.4400	.2205	.3631	.2462	**.1743**

C.12 Ablation on Bias Parameterization

Table C.13 evaluates the effect of different bias parameterizations. Overall, the performance differences across variants are small, indicating that DBCM is not highly sensitive to this design choice. We therefore use the channel-wise bias parameterization as the default, as it provides a simple per-channel offset while achieving performance comparable to the other variants. The full, time-wise, scalar, and no-bias variants yield similar results across datasets. These results suggest that the main performance gains of DBCM do not depend on a specific bias parameterization.

Table C.13: **Ablation on Bias Parameterization.** Test MSE under different additive bias parameterizations. Here, L and C denote the temporal length and number of channels, respectively. Bold entries indicate the best performance among the compared variants.

Bias parameterization	ETTh1	ETTh2	ETTm1	ETTm2	Exchange	Weather	Electricity
Full ($\mathbb{R}^{L\times C}$)	.5166	.3136	.4340	**.2190**	**.3412**	**.2422**	**.1744**
Channel-wise ($\mathbb{R}^{C}$)	**.5163**	**.3135**	**.4339**	**.2190**	.3413	.2424	**.1744**
Time-wise ($\mathbb{R}^{L}$)	.5168	.3136	.4344	**.2190**	.3413	.2423	**.1744**
Scalar ($\mathbb{R}$)	.5167	.3136	.4344	**.2190**	.3414	.2424	.1745
None	.5168	.3136	.4345	**.2190**	.3414	.2424	**.1744**

D Detailed Dataset Information

We provide detailed information on the benchmark datasets used in our experiments, including their domains, temporal resolutions, and brief descriptions. Table D.1 presents an overview of all datasets. Table D.2 summarizes the statistical characteristics of the benchmark datasets, including the number of variables, temporal resolution, and stationarity measured by the Augmented Dickey–Fuller (ADF) statistic (Elliott et al., 1996). The ADF statistic serves as an indicator of non-stationarity, where values closer to zero imply stronger non-stationarity (Kim et al., 2025). Although it does not directly measure the discrepancy between training and test distributions, it provides a proxy for how strongly the temporal dynamics evolve over time. As shown in the table, Exchange exhibits the highest ADF value (-1.90), indicating the strongest non-stationarity among the datasets. In contrast, datasets such as Weather (-26.68) show highly negative ADF values, suggesting relatively stable temporal dynamics. Overall, these datasets span a wide range of non-stationarity.

E Training Setup

We follow the experimental setup of TAFAS (Kim et al., 2025). For backbone pre-training, we perform a hyperparameter search over learning rates $\{1\text{e}{-3}, 1\text{e}{-4}, 1\text{e}{-5}\}$ and weight decay values $\{0.0, 1\text{e}{-3}, 1\text{e}{-4}\}$. Each configuration is trained with three random seeds, and we select the hyperparameter setting that achieves the lowest validation MSE. All backbone models are pre-trained for 30 epochs with a batch size of 64 using the Adam optimizer (Kingma & Ba, 2015) and cosine learning rate scheduling (Loshchilov & Hutter, 2017). For all TSF-TTA methods, we follow the tuning procedure of TAFAS (Kim et al., 2025), including the same data split for hyperparameter selection, and search the learning rate over $\{5\text{e}{-3}, 3\text{e}{-3}, 1\text{e}{-3}, 5\text{e}{-4}, 1\text{e}{-4}\}$, the weight decay over $\{0.0, 1\text{e}{-4}\}$, and the initial value of the gating parameter over $\{0.01, 0.05, 0.1, 0.3\}$, so that all methods receive the same tuning budget and selection procedure. This tuning procedure is applied consistently across all experimental settings. For reproducibility, the selected hyperparameter configurations for DBCM will be provided in our official implementation. All experiments are conducted on a single RTX 4090 GPU, except those involving Moirai (Woo et al., 2024), Chronos (Ansari et al., 2024), and Chronos-2 (Ansari et al., 2025), which are conducted on a single RTX 6000 ADA GPU.

F Drift Descriptors

We provide detailed definitions of the channel-wise drift descriptors used to characterize raw data-side drift. Following the notation in the main text, $\mathbf{x}_t \in \mathbb{R}^C$ denotes the multivariate observation at time t. For channel-wise descriptor computation, we write $x_t^{(c)} = (\mathbf{x}_t)_c$ for the scalar value of channel c at time t. All descriptors are computed on channel-wise standardized series:

$$\tilde{x}_t^{(c)} = \frac{x_t^{(c)} - \mu_c^{\text{train}}}{\sigma_c^{\text{train}}}, \tag{F.1}$$

where μ_c^{train} and σ_c^{train} denote the mean and standard deviation of channel c computed over the training split.

We use sliding lookback windows of length $L = 96$ with stride 1. For a descriptor window ending at time τ, let

$$\mathbf{w}_{\tau,c} = [\tilde{x}_{\tau-L+1}^{(c)}, \ldots, \tilde{x}_{\tau}^{(c)}] \in \mathbb{R}^L. \tag{F.2}$$

Table C.14: **Prediction Performance Comparison Across Methods.** Test MSE is reported for six models on six datasets with four prediction horizons. Parenthesized values indicate standard deviation across seeds. Bold entries indicate the best performance among methods, while underlined entries denote the second-best.

		iTransformer			PatchTST			DLinear			OLS			FreTS			MICN		
		Baseline	TAFAS	PETSA	DBCM	Baseline	TAFAS	PETSA	DBCM	Baseline	TAFAS	PETSA	DBCM	Baseline	TAFAS	PETSA	DBCM	Baseline	TAFAS	PETSA	DBCM
ETTh1	96	.4349	.4308 (.0000)	.4284 (.0004)	**.4228 (.0002)**	.4294	.4232 (.0000)	.4255 (.0001)	**.4194 (.0001)**	.4510	.4425 (.0000)	.4400 (.0004)	**.4318 (.0004)**	.4511	.4420 (.0000)	.4436 (.0003)	**.4321 (.0001)**	.4360	.4295 (.0000)	.4296 (.0003)	**.4242 (.0001)**	.4573	.4471 (.0000)	.4453 (.0001)	**.4342 (.0006)**
	192	.4937	.4894 (.0001)	.4886 (.0003)	**.4805 (.0001)**	.4904	.4805 (.0001)	.4849 (.0001)	**.4757 (.0001)**	.5042	.4946 (.0000)	.4943 (.0001)	**.4835 (.0001)**	.5044	.4923 (.0000)	.5040 (.0001)	**.4836 (.0001)**	.4886	.4819 (.0000)	.4842 (.0002)	**.4764 (.0001)**	.5145	.4996 (.0000)	.5063 (.0003)	**.4900 (.0003)**
	336	.5406	.5382 (.0001)	.5392 (.0002)	**.5322 (.0001)**	.5605	.5490 (.0000)	.5600 (.0001)	**.5410 (.0003)**	.5508	.5454 (.0000)	.5455 (.0003)	**.5373 (.0001)**	.5510	.5415 (.0000)	.5505 (.0000)	**.5383 (.0001)**	.5466	.5394 (.0000)	.5462 (.0001)	**.5354 (.0002)**	.5715	.5596 (.0000)	.5708 (.0001)	**.5553 (.0002)**
	720	.6914	.6520 (.0004)	.6535 (.0015)	**.6298 (.0004)**	.6904	.6643 (.0001)	.6878 (.0002)	**.6417 (.0006)**	.7019	.6713 (.0000)	.6617 (.0003)	**.6487 (.0003)**	.6996	.6654 (.0000)	.6728 (.0004)	**.6514 (.0002)**	.7145	.6839 (.0000)	.6795 (.0007)	**.6565 (.0002)**	.7224	.6916 (.0001)	.7201 (.0002)	**.6743 (.0004)**
ETTh2	96	.2430	.2406 (.0000)	.2403 (.0003)	**.2388 (.0001)**	.2309	.2298 (.0000)	.2308 (.0000)	**.2291 (.0001)**	.2294	.2274 (.0000)	.2280 (.0001)	**.2268 (.0001)**	.2305	.2283 (.0000)	.2286 (.0000)	**.2279 (.0001)**	.2376	**.2347 (.0000)**	.2359 (.0001)	**.2347 (.0005)**	.2346	.2326 (.0000)	.2333 (.0001)	**.2321 (.0001)**
	192	.2973	.2931 (.0000)	.2918 (.0001)	**.2914 (.0001)**	.2871	.2802 (.0000)	.2867 (.0000)	**.2785 (.0001)**	.2835	**.2793 (.0000)**	.2840 (.0007)	.2794 (.0001)	.2838	.2817 (.0000)	.2822 (.0001)	**.2802 (.0001)**	.2838	.2776 (.0001)	.2805 (.0001)	**.2761 (.0005)**	.2934	.2858 (.0000)	.2898 (.0001)	**.2848 (.0002)**
	336	.3448	.3328 (.0001)	.3334 (.0001)	**.3301 (.0002)**	.3311	.3163 (.0000)	.3303 (.0001)	**.3157 (.0002)**	.3250	.3174 (.0000)	.3185 (.0007)	**.3152 (.0002)**	.3257	.3186 (.0000)	.3251 (.0001)	**.3157 (.0002)**	.3184	.3085 (.0000)	.3117 (.0002)	**.3048 (.0001)**	.3380	.3254 (.0000)	.3309 (.0001)	**.3250 (.0003)**
	720	.4258	.3999 (.0001)	.3987 (.0002)	**.3939 (.0003)**	.4067	.3886 (.0001)	.3958 (.0002)	**.3769 (.0003)**	.4149	.3913 (.0000)	.3878 (.0009)	**.3820 (.0004)**	.4164	.3902 (.0000)	.3883 (.0001)	**.3831 (.0001)**	.3951	.3690 (.0000)	.3812 (.0003)	**.3653 (.0002)**	.4328	.4126 (.0001)	.4191 (.0003)	**.4090 (.0002)**
ETTm1	96	.3763	.3606 (.0001)	.3601 (.0007)	**.3415 (.0004)**	.3869	.3723 (.0000)	.3829 (.0001)	**.3469 (.0003)**	.3705	.3496 (.0000)	.3524 (.0002)	**.3420 (.0000)**	.3708	.3520 (.0000)	.3574 (.0004)	**.3420 (.0002)**	.3618	.3524 (.0000)	.3525 (.0005)	**.3420 (.0002)**	.4063	.3723 (.0007)	.3756 (.0002)	**.3576 (.0002)**
	192	.4371	.4184 (.0001)	.4145 (.0005)	**.4038 (.0001)**	.4387	.4220 (.0000)	.4328 (.0001)	**.4087 (.0003)**	.4420	.4159 (.0000)	.4175 (.0006)	**.4056 (.0001)**	.4437	.4165 (.0000)	.4230 (.0002)	**.4054 (.0001)**	.4304	.4177 (.0000)	.4168 (.0002)	**.4060 (.0002)**	.4479	.4315 (.0000)	.4308 (.0002)	**.4155 (.0004)**
	336	.5029	.4778 (.0002)	.4739 (.0003)	**.4628 (.0001)**	.4957	.4780 (.0001)	.4939 (.0001)	**.4593 (.0002)**	.5172	.4796 (.0000)	.4795 (.0005)	**.4626 (.0001)**	.5176	.4786 (.0000)	.4853 (.0005)	**.4623 (.0001)**	.4928	.4752 (.0000)	.4719 (.0002)	**.4618 (.0001)**	.5154	.4885 (.0001)	.4889 (.0002)	**.4676 (.0001)**
	720	.5713	.5484 (.0000)	.5472 (.0003)	**.5276 (.0001)**	.5666	.5390 (.0001)	.5615 (.0000)	**.5255 (.0002)**	.5927	.5482 (.0000)	.5526 (.0002)	**.5321 (.0002)**	.5918	.5478 (.0000)	.5519 (.0003)	**.5317 (.0002)**	.5609	.5399 (.0000)	.5404 (.0001)	**.5274 (.0001)**	.5749	.5404 (.0001)	.5689 (.0005)	**.5206 (.0003)**
ETTm2	96	.1555	.1542 (.0000)	.1543 (.0000)	**.1536 (.0001)**	.1554	.1542 (.0000)	.1552 (.0000)	**.1534 (.0001)**	.1589	.1577 (.0000)	.1579 (.0001)	**.1573 (.0001)**	.1602	.1593 (.0000)	.1592 (.0000)	**.1585 (.0001)**	.1563	.1547 (.0000)	.1549 (.0000)	**.1536 (.0001)**	.1592	.1580 (.0000)	.1582 (.0001)	**.1574 (.0001)**
	192	.1938	.1926 (.0001)	.1928 (.0001)	**.1923 (.0001)**	.1937	.1924 (.0000)	.1935 (.0000)	**.1919 (.0001)**	.1929	.1913 (.0000)	.1913 (.0001)	**.1907 (.0007)**	.1935	.1921 (.0000)	**.1919 (.0001)**	**.1919 (.0001)**	.1914	.1896 (.0000)	.1898 (.0001)	**.1880 (.0005)**	.1955	.1927 (.0000)	.1923 (.0001)	**.1914 (.0001)**
	336	.2345	.2441 (.0146)	.2325 (.0003)	**.2312 (.0001)**	.2321	.2292 (.0000)	.2317 (.0000)	**.2282 (.0001)**	.2325	.2290 (.0000)	.2293 (.0001)	**.2289 (.0001)**	.2331	**.2298 (.0000)**	.2301 (.0001)	**.2298 (.0001)**	.2310	.2275 (.0000)	.2305 (.0000)	**.2259 (.0001)**	.2421	.2364 (.0004)	**.2355 (.0001)**	.2357 (.0003)
	720	.3101	.3012 (.0001)	.3033 (.0018)	**.2990 (.0001)**	.3033	**.2924 (.0000)**	.3005 (.0001)	.2933 (.0002)	.3061	.2967 (.0000)	.2962 (.0001)	**.2956 (.0001)**	.3066	.2982 (.0000)	.2977 (.0001)	**.2964 (.0001)**	.3026	**.2924 (.0000)**	.3003 (.0001)	.2930 (.0001)	.3176	.3040 (.0001)	.3049 (.0003)	**.3023 (.0004)**
Exchange	96	.0916	.0883 (.0000)	.0879 (.0002)	**.0858 (.0002)**	.0846	.0811 (.0000)	.0839 (.0001)	**.0806 (.0003)**	.0811	**.0789 (.0000)**	.0793 (.0001)	**.0789 (.0002)**	.0814	.0795 (.0000)	.0808 (.0000)	**.0793 (.0004)**	.0827	**.0784 (.0000)**	.0815 (.0001)	.0794 (.0002)	.0829	.0804 (.0000)	.0818 (.0001)	**.0800 (.0001)**
	192	.1914	.1834 (.0001)	.1867 (.0003)	**.1770 (.0004)**	.1805	.1697 (.0000)	.1787 (.0001)	**.1683 (.0006)**	.1708	.1643 (.0000)	.1641 (.0003)	**.1626 (.0007)**	.1723	.1640 (.0000)	.1700 (.0001)	**.1631 (.0017)**	.1787	.1667 (.0000)	.1765 (.0000)	**.1659 (.0005)**	.1886	.1777 (.0000)	.1853 (.0001)	**.1742 (.0011)**
	336	.3375	.3146 (.0004)	.3243 (.0001)	**.2844 (.0051)**	.3375	.2585 (.0000)	.4755 (.0003)	**.4087 (.0312)**	.3205	.2883 (.0000)	.2931 (.0003)	**.2642 (.0011)**	.3226	.2851 (.0000)	.2876 (.0008)	**.2613 (.0013)**	.3265	.2922 (.0000)	.3198 (.0004)	**.2659 (.0012)**	.3529	.3116 (.0000)	.3315 (.0006)	**.2829 (.0021)**
	720	.9433	.8588 (.0000)	.8894 (.0024)	**.8182 (.0290)**	.8814	.8812 (.0000)	.8814 (.0000)	**.6609 (.0837)**	.8383	.8309 (.0000)	.8109 (.0010)	**.7066 (.0793)**	.8360	.8485 (.0000)	.8114 (.0009)	**.5439 (.0064)**	.8251	.7606 (.0000)	.8141 (.0001)	**.6700 (.0115)**	1.4040	**1.1231 (.0000)**	1.3355 (.0011)	1.2289 (.1460)
Weather	96	.1719	.1642 (.0001)	.1661 (.0007)	**.1630 (.0003)**	.1738	.1672 (.0000)	.1730 (.0001)	**.1657 (.0002)**	.1936	.1788 (.0000)	.1769 (.0005)	**.1732 (.0002)**	.1955	.1808 (.0000)	.1819 (.0015)	**.1763 (.0004)**	.1809	.1693 (.0000)	.1713 (.0006)	**.1677 (.0001)**	.1720	.1699 (.0002)	.1726 (.0002)	**.1670 (.0004)**
	192	.2223	.2104 (.0003)	.2129 (.0001)	**.2096 (.0001)**	.2207	.2103 (.0000)	.2183 (.0001)	**.2073 (.0002)**	.2383	.2236 (.0000)	.2255 (.0003)	**.2194 (.0003)**	.2403	.2219 (.0000)	.2272 (.0001)	**.2196 (.0003)**	.2266	.2109 (.0000)	.2179 (.0002)	**.2101 (.0001)**	.2222	.2155 (.0001)	.2237 (.0009)	**.2148 (.0001)**
	336	.2800	.2610 (.0000)	.2646 (.0001)	**.2608 (.0001)**	.2760	.2598 (.0001)	.2721 (.0001)	**.2586 (.0002)**	.2913	.2714 (.0000)	.2755 (.0003)	**.2697 (.0001)**	.2920	.2706 (.0000)	.2772 (.0001)	**.2697 (.0002)**	.2806	.2609 (.0000)	.2675 (.0003)	**.2602 (.0001)**	.2747	.2626 (.0002)	.2701 (.0008)	**.2624 (.0006)**
	720	.3576	.3404 (.0000)	.3498 (.0001)	**.3360 (.0006)**	.3512	.3644 (.0001)	.3410 (.0005)	**.3315 (.0001)**	.3643	.3490 (.0000)	.3501 (.0006)	**.3391 (.0004)**	.3646	.3433 (.0000)	.3626 (.0001)	**.3398 (.0003)**	.3557	.3364 (.0000)	.3536 (.0001)	**.3338 (.0002)**	.3491	.3371 (.0001)	.3546 (.0028)	**.3298 (.0003)**
Average		.3687	.3531	.3556	**.3444**	.3710	.3585	.3657	**.3402**	.3655	.3509	.3505	**.3389**	.3660	.3429	.3538	**.3326**	.3585	.3436	.3503	**.3343**	.3946	.3690	.3844	**.3660**

Table D.1: **Benchmark Dataset Characteristics.** We summarize the domain, sampling frequency, number of channels, and a brief description of each dataset.

Dataset	Domain	Frequency	# Channels	Description
Electricity	Electricity	1 hour	321	Hourly electricity consumption from 2012 to 2014.
Weather	Weather	10 minutes	21	Meteorological indicators recorded every 10 minutes in 2020.
ETTh	Electricity	1 hour	7	Hourly transformer load and oil-temperature data.
ETTm	Electricity	15 minutes	7	15-minute transformer load and oil-temperature data.
Exchange	Finance	1 day	8	Daily exchange rates from eight countries.

Table D.2: **Benchmark Dataset Statistics.** We report the number of time steps and ADF test statistic for each dataset.

Dataset	Time Steps	ADF
ETTh1	17,420	-5.91
ETTm1	69,680	-14.98
ETTh2	17,420	-4.13
ETTm2	69,680	-5.66
Exchange	7,588	-1.90
Weather	52,696	-26.68

We compute six descriptor types, grouped into low-order and frequency/temporal descriptor families. The low-order descriptors consist of mean, log-variance, and slope, while the frequency/temporal descriptors consist of dominant frequency, spectral-band energy, and autocorrelation.

For the low-order descriptors, we define

$$\mu_{\tau,c} = \frac{1}{L}\sum_{i=1}^{L} w_{\tau,c}[i], \tag{F.3}$$

$$v_{\tau,c} = \frac{1}{L}\sum_{i=1}^{L} \left(w_{\tau,c}[i] - \mu_{\tau,c}\right)^2, \tag{F.4}$$

and use the log-variance descriptor

$$\tilde{v}_{\tau,c} = \log\left(v_{\tau,c} + \epsilon\right), \tag{F.5}$$

where $\epsilon > 0$ is a small constant for numerical stability.

For the slope descriptor, we use a centered normalized time axis

$$a_i = -\frac{1}{2} + \frac{i-1}{L-1}, \qquad i = 1, \ldots, L, \tag{F.6}$$

and compute

$$s_{\tau,c} = \frac{\sum_{i=1}^{L} a_i\, w_{\tau,c}[i]}{\sum_{i=1}^{L} a_i^2}. \tag{F.7}$$

The frequency and temporal descriptors are computed from the centered signal

$$\mathbf{u}_{\tau,c} = \mathbf{w}_{\tau,c} - \mu_{\tau,c}\mathbf{1}_L. \tag{F.8}$$

Let

$$P_{\tau,c}[f] = \left|\mathrm{FFT}(\mathbf{u}_{\tau,c})[f]\right|^2, \qquad f = 0, \ldots, \left\lfloor \frac{L}{2} \right\rfloor, \tag{F.9}$$

denote the one-sided Fourier power spectrum.

We define the dominant-frequency descriptor as the normalized index of the strongest positive-frequency bin:

$$f^{\star}_{\tau,c} = \underset{1 \le f \le \lfloor L/2 \rfloor}{\arg\max}\; P_{\tau,c}[f], \qquad \phi_{\tau,c} = \frac{f^{\star}_{\tau,c}}{\lfloor L/2 \rfloor}. \tag{F.10}$$

To capture spectral energy, we partition the positive-frequency bins into N_{band} disjoint bands $\mathcal{B}_1, \ldots, \mathcal{B}_{N_{\text{band}}}$ and compute log band energies

$$e_{\tau,c}^{(b)} = \log\left(\sum_{f \in \mathcal{B}_b} P_{\tau,c}[f] + \epsilon\right), \qquad b = 1, \ldots, N_{\text{band}}. \tag{F.11}$$

In the default implementation, $N_{\text{band}} = 3$.

For autocorrelation, we use a lag set $\Lambda = \{\ell_1, \ldots, \ell_{N_{\text{lag}}}\}$ and compute

$$\rho_{\tau,c}^{(\ell)} = \frac{\sum_{i=1}^{L-\ell} u_{\tau,c}[i] u_{\tau,c}[i+\ell]}{\sum_{i=1}^{L} u_{\tau,c}[i]^2 + \epsilon}. \qquad \ell \in \Lambda. \tag{F.12}$$

By default, Λ is chosen as $\{1, p_{\text{data}}, 2p_{\text{data}}\}$ after removing invalid lags, where p_{data} denotes the dataset period length.

The concatenated descriptor vector for channel c at time τ is

$$\mathbf{d}_{\tau,c} = [\mu_{\tau,c}, \tilde{v}_{\tau,c}, s_{\tau,c}, \phi_{\tau,c}, e_{\tau,c}^{(1)}, \ldots, e_{\tau,c}^{(N_{\text{band}})}, \rho_{\tau,c}^{(\ell_1)}, \ldots, \rho_{\tau,c}^{(\ell_{N_{\text{lag}}})}] \in \mathbb{R}^K, \tag{F.13}$$

where

$$K = 4 + N_{\text{band}} + N_{\text{lag}}. \tag{F.14}$$

Under the default setting $N_{\text{band}} = 3$ and $N_{\text{lag}} = 3$, we have $K = 10$.

We consider two drift references. Under the source reference, each test-window descriptor is compared against the mean descriptor over the training windows:

$$\bar{\mathbf{d}}_c^{\text{src}} = \frac{1}{|\mathcal{I}_{\text{train}}|} \sum_{\tau \in \mathcal{I}_{\text{train}}} \mathbf{d}_{\tau,c}, \tag{F.15}$$

where $\mathcal{I}_{\text{train}}$ denotes the set of descriptor-window endpoints in the training split. The source-referenced drift is then defined as

$$\Delta_{\tau,c}^{\text{src}} = \mathbf{d}_{\tau,c} - \bar{\mathbf{d}}_c^{\text{src}}. \tag{F.16}$$

Under the previous-window reference, each test-window descriptor is compared against the descriptor of the immediately preceding test window:

$$\Delta_{\tau,c}^{\text{prev}} = \mathbf{d}_{\tau,c} - \mathbf{d}_{\tau-1,c}. \tag{F.17}$$

For each descriptor family j, the resulting drift tensor has shape $[N_{\text{win}}, C, K_j]$, where N_{win} is the number of descriptor windows, C is the number of channels, and K_j is the feature dimension of that descriptor family. We reshape each tensor into a channel-space matrix

$$\mathbf{S}^{(j)} \in \mathbb{R}^{C \times (N_{\text{win}} K_j)} \tag{F.18}$$

for channel-space spectral analysis. For the combined analysis, we first normalize each descriptor-family matrix and then concatenate them along the feature dimension.

G Additional Related Work

Time Series Forecasting. Recent time series forecasting models have introduced diverse architectures for capturing complex temporal dependencies. Transformer-based forecasters, such as PatchTST (Nie et al., 2023) and iTransformer (Liu et al., 2024), model long-range temporal and inter-channel relationships, while alternative approaches explore different inductive biases. DLinear (Zeng et al., 2023) demonstrates the effectiveness of simple linear mappings, FreTS (Yi et al., 2023) leverages frequency-domain representations, and MICN (Wang et al., 2023) adopts multi-scale convolutions. More recently, time series foundation models (TSFMs), including Chronos (Ansari et al., 2024; 2025) and TimesFM (Das et al., 2024), have been proposed to improve generalization through pre-training on large-scale time series corpora. Despite strong performance, they implicitly assume stationarity, whereas real-world time series exhibit evolving and non-stationary distributions, motivating test-time adaptation methods (Kim et al., 2025).

Test-Time Adaptation. Test-time adaptation (TTA) aims to improve model performance under distribution shift by adapting a pre-trained model using test-time data. Representative methods such as Tent (Wang et al., 2021) adapt models by minimizing prediction entropy, and subsequent works improve stability and efficiency under continual distribution shifts (Wang et al., 2022; Niu et al., 2022). These methods, however, are primarily designed for classification, where entropy over class probabilities provides an unsupervised adaptation objective. Extending TTA to time series forecasting requires different assumptions. Forecasting is a regression task, so entropy minimization objectives are not directly applicable. Moreover, future values become observable after prediction, and the sequential structure of time series enables partially observed ground truth before the full horizon is available. At the same time, test windows are temporally dependent rather than IID (Independent and Identically Distributed). Therefore, TSF-TTA should exploit delayed and partially observed targets while adapting a frozen forecaster in a way that respects the streaming, non-IID nature of time series.

Cross-Channel Dependency in Multivariate Time Series Forecasting. Cross-channel dependency is a key factor in multivariate time series forecasting. Recent works have explored various mechanisms to capture such dependencies during representation learning. Crossformer (Zhang & Yan, 2023) introduces cross-dimension attention to explicitly model inter-channel interactions. TSMixer (Chen et al., 2023a) demonstrates that simple MLP-based alternating time and feature mixing can effectively capture cross-channel structure without relying on attention. MTS-Mixer (Li et al., 2025) further extends this line of work by proposing a factorized architecture that separately performs temporal and channel mixing to efficiently model multivariate dependencies. iTransformer (Liu et al., 2024) revisits the Transformer by treating channels as tokens, enabling direct modeling of cross-channel relationships via self-attention. TimeXer (Wang et al., 2024) enhances this paradigm with more expressive mixing mechanisms across time and channels. PCD (Lee et al., 2026) further argues that channel dependence can be partial and dataset-specific, and incorporates channel masks constructed from channel similarity and learnable domain parameters into Transformer attention matrices. Finally, CrossLinear (Zhou et al., 2025) explores linear mappings with cross-channel interactions, showing that even simple linear structures can capture inter-variable dependencies. In contrast, we consider cross-channel structure in the context of test-time adaptation, using channel mixing to provide cross-channel context before calibration.

Cross-Channel Structure of Forecast Residuals. A general theoretical characterization of when forecast residuals are low-rank in the channel space requires restrictive assumptions, since residuals depend on both the data-generating process and the fitted forecaster (West, 1996). Nevertheless, multivariate forecast errors can exhibit contemporaneous, temporal, and cross-lag dependencies (Zheng & Sun, 2024), and high-dimensional cross-channel dependence can sometimes be represented compactly through low-rank structures (Salinas et al., 2019). Such structure may arise when many channels share a small number of factors that are incompletely modeled by the forecaster, such as temperature shocks affecting regional electricity demand. Conversely, cross-channel correlation in the raw signals does not by itself imply concentrated residuals: if the forecaster captures the shared factors well, the remaining residuals are dominated by channel-specific components. Consistently, raw data-side drift is less concentrated in the channel space than forecast residuals once it is characterized beyond simple statistical shifts (Appendix A.4). These observations support the plausibility of our empirical hypothesis, but do not imply that residual low-rankness is a universal property. Accordingly, we treat the low-rank structure of forecast residuals as an empirical hypothesis whose strength varies across datasets and forecasters (Section 4). This variation motivates the dual-branch design of DBCM, which combines parameters shared across channels with a channel-wise LoRA branch that provides limited channel-specific flexibility.

Online Learning for Time Series Forecasting. Handling distribution shift in time series has been widely studied in online learning. FSNet (Pham et al., 2023) introduces a dual-timescale framework that balances fast adaptation and long-term stability. OneNet (Wen et al., 2023) adopts an online ensembling strategy to cope with concept drift. SOLID (Chen et al., 2024) explores calibration-based approaches to detect and correct context-driven distribution shifts. More recently, PROCEED (Zhao & Shen, 2025) proposes a proactive adaptation mechanism that anticipates distribution shift, ELF (Lee et al., 2025) focuses on lightweight online adaptation for time series foundation model forecasts, and DynaME (Hong et al., 2026) dynamically selects multi-period experts for recurring patterns and a general expert for emergent shifts. These approaches handle distribution shift

by continuously updating the forecasting system itself, and do not explicitly model the structure of adaptation signals. In contrast, our scope is restricted to the TSF-TTA setting, in which the pre-trained source forecaster remains frozen and only external calibration modules are updated. Within this setting, we show that adaptation signals exhibit compact shared structure across channels and propose a structured calibration framework that decomposes them into shared and residual components.