

APT: Affine Prototype-Timestamp For Time Series Forecasting Under Distribution Shift

Yujie Li,^{1,2} Zezhi Shao,¹ Chengqing Yu,^{1,2} Yisong Fu,^{1,2} Tao Sun,¹ Yongjun Xu,^{1,2} Fei Wang^{1,2*}

¹State Key Laboratory of AI Safety, Institute of Computing Technology, Chinese Academy of Sciences

²University of Chinese Academy of Sciences

{liyujie23s, shaozezhi, yuchengqing22b, fuyisong24s, suntao, xyj, wangfei}@ict.ac.cn

Abstract

Time series forecasting under distribution shift remains challenging, as existing deep learning models often rely on local statistical normalization (e.g., mean and variance) that fails to capture global distribution shift. Methods like RevIN and its variants attempt to decouple distribution and pattern but still struggle with missing values, noisy observations, and invalid channel-wise affine transformation. To address these limitations, we propose **Affine Prototype-Timestamp (APT)**, a lightweight and flexible plug-in module that injects global distribution features into the normalization–forecasting pipeline. By leveraging timestamp-conditioned prototype learning, APT dynamically generates affine parameters that modulate both input and output series, enabling the backbone to learn from self-supervised, distribution-aware clustered instances. APT is compatible with arbitrary forecasting backbones and normalization strategies while introducing minimal computational overhead. Extensive experiments across six benchmark datasets and multiple backbone-normalization combinations demonstrate that APT significantly improves forecasting performance under distribution shift.

Code — <https://github.com/blisky-li/APT>

1 Introduction

Time series reflect the artificial or natural regularities of complex dynamic systems across transportation (Li et al. 2024; Shao et al. 2022), health (Ferté et al. 2024) and weather (Yu et al. 2025). However, external factors commonly induce distributional shift and non-stationarity, making forecasting models struggle to effectively capture patterns under changing statistical properties.

Reversible Instance Normalization (RevIN) (Kim et al. 2021) introduces a two-stage paradigm for mitigating temporal distribution shift: instance normalization that removes instance-specific distributions and affine transformation that attempts to restore channel-wise distributional features. This decoupling of time-varying distributions from learnable temporal patterns has laid the foundation for many modern forecasting models, and subsequent advancements—including DishTS (Fan et al. 2023), SAN (Liu et al.

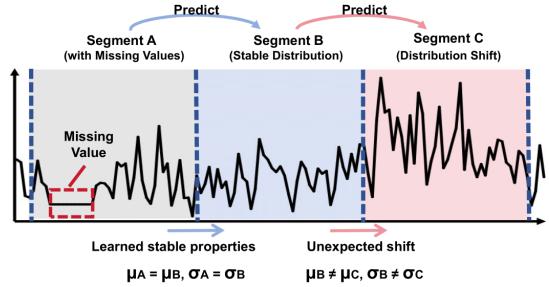


Figure 1: Local statistical normalization fails to handle distribution shift on ECL. The model retains outdated statistics when predicting from flawed Segment A to B but faces unseen shifts in Segment B and C. Shifts across all three segments exceed prior inter and intra shift issues, posing a broader global shift challenge for APT.

2023), SIN (Han, Ye, and Zhan 2024), and FAN (Ye et al. 2024)—have further refined this framework by modeling future distributions adaptively to address intra-instance shift.

Despite their success, RevIN-like methods still face inherent limitations. First, Local statistics such as mean and variance are sensitive to missing values and noise, which are common in time series. Second, their instance-specific normalization fails to capture global shifts, as shown in Figure 1, where features learned from Segments A and B collapse in Segment C under abrupt change.

Moreover, the channel-wise static affine transformation (Bebis et al. 1999) in RevIN yields only limited improvements as shown in Table 1, and is entirely omitted in follow-up works such as DishTS and SAN. This suggests static transformations fail to address inter-channel distribution variance and are ineffective against distribution shift.

To overcome the limitations of local statistical normalization, we propose **Affine Prototype-Timestamps (APT)**, a lightweight and model-agnostic plug-in. APT replaces static affine transformations in standard normalization with dynamically generated parameters conditioned on timestamps, inspired by vision style transfer (Huang and Belongie 2017). This enables the forecasting pipeline to access global temporal semantics, which serve as an underlying schedule guiding the behavior of the time series system, thereby enhancing robustness and adaptability under distribution shift.

*Corresponding authors.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Methods	CATS		Informer+RevIN		iTransformer		SparseTSF		Avg. Δ	
Affine (RevIN)	True	False	True	False	True	False	True	False	True - False	
Metric	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
ECL	0.260	0.165	0.261	0.165	0.307	0.209	0.308	0.212	0.264	0.169
Weather	0.274	0.236	0.274	0.241	0.311	0.307	0.310	0.306	0.284	0.251
ETTh1	0.447	0.456	0.441	0.451	0.582	0.681	0.584	0.679	0.462	0.471
Exchange	0.437	0.352	0.436	0.349	0.677	0.708	0.644	0.727	0.493	0.423
							0.475	0.398	0.499	0.434
							0.492	0.428	↑0.015	↑0.004

Table 1: The affine transformations in RevIN do not provide performance gains, Metric \uparrow means worse performance.

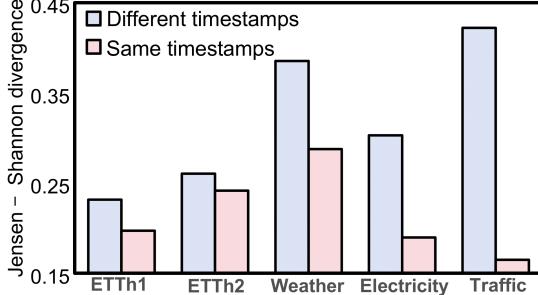


Figure 2: JS divergence of subseries with the same timestamp label in benchmark datasets

Easily accessible timestamps are widely regarded as containing global temporal semantics (Li et al. 2025), and our empirical analysis in Figure 2 confirms subseries sharing similar timestamp labels such as "Time in Day" and "Day in Week" tend to exhibit distributional similarity. APT exploits this regularity by embedding timestamps into a shared latent space to enable conditioned affine transformations.

However, the sparsity of fine-grained timestamps combinations can cause insufficient training or poor generalization to unseen cases. Inspired by few-shot learning (Li et al. 2022), APT employs prototype learning, replacing raw timestamp embeddings with nearest-neighbor prototypes to yield more robust representations. To further promote diversity and balanced prototype usage, we introduce orthogonality and load-balancing losses.

Rather than learning static affine parameters, APT utilizes global temporal semantics in timestamps to adaptively assign distinct affine parameters to different subseries. To achieve this, it first encodes timestamps and obtains robust representations through prototype matching, then converts prototype embeddings into low-dimensional affine parameters via MLPs. Crucially, this process employs self-supervised learning with the backbone and normalization frozen, leveraging orthogonal loss for embedding diversity, load-balancing loss for prototype usage, and extra affine regularization loss to ensure affine parameter convergence.

In summary, the central function of APT is to supply forecasting backbones with learnable and global distribution features suppressed by normalization or disrupted by distribution shift; all components and optimization of APT are explicitly designed to serve this purpose.

Our contributions are as follow:

- We propose APT, a lightweight and model-agnostic plugin for overcoming limitations of local statistical normalization and mitigating distribution shift.
- APT generates dynamic affine parameters via timestamp-conditioned prototype matching and self-supervised learning, delivering learnable distribution features compatible with any forecasting backbone and normalization.
- Extensive experiments with diverse backbones and normalization methods confirm APT’s effectiveness in improving forecasting accuracy under distribution shift.

2 Related Work

2.1 Deep Time Series Forecasting

As the backbone of forecasting, time series models are primarily designed to learn temporal patterns such as seasonality and trends (Wang et al. 2025a; Shao et al. 2025). Deep models like Informer (Zhou et al. 2021) improve long-term forecasting, while DLinear (Zeng et al. 2023) shows that simple MLP can also perform well. Recent developments in forecasting focus on complex temporal dependencies, including intra-channel long-term dependencies, as explored by PatchTST (Nie et al. 2023) and SparseTSF (Lin et al. 2024), as well as inter-channel interactions, as addressed by iTransformer (Liu et al. 2024) and CATS (Kim et al. 2024).

Distribution shift is not typically the focus of forecasting backbone, but it is particularly important in the forecasting pipeline because it significantly increases the difficulty of pattern learning and reduces forecasting performance.

2.2 Normalization for Distribution Shift

Normalization has become a standard component in time series forecasting due to its efficiency in handling distribution shift. RevIN (Kim et al. 2021) proposes a distribution-pattern decoupling scheme by forcing the normalization of histories to a unified domain and de-normalizing predictions, which greatly reduces the complexity of non-stationary forecasting (Fu et al. 2025).

Subsequent works aim to recover suppressed distributional features because the distribution of history and future within a single instance is inconsistent. DishTS (Fan et al. 2023) learns intra- and inter-segment drift. SAN (Liu et al. 2023) focuses on patch-level shift. SIN (Han, Ye, and Zhan 2024) relearns statistics based on local invariance and

global variability criteria, and FAN (Ye et al. 2024) replaces normalization with main frequency extraction and residual forecasting.

2.3 Affine Transformation

Affine transformations were initially used to align parameter spaces between layers after normalization, improving training stability of deep models. In conditional generative networks, they serve to reintroduce task-specific information by modulating distributional features (Karras, Laine, and Aila 2019). CIN (Dumoulin, Shlens, and Kudlur 2017) assigns affine parameters to each style via instance normalization. AdaIN (Huang and Belongie 2017) extracts them from target images for arbitrary style transfer, and FiLM (Perez et al. 2018) extends this idea to feature-level conditioning. Such conditional affine modulation has become standard across vision and language tasks (Ziegler et al. 2019).

However, such conditional normalization techniques are rarely explored in time series forecasting, where normalization still mainly depends on local statistics. Inspired by these advances, APT introduces timestamp-based conditioning to generate dynamic affine parameters, enabling better adaptation to distribution shift.

3 Preliminaries

3.1 Time Series Forecasting

Given the historical multivariate time series $\{x_{t-L:t}^{(i)}\}_{i=1}^C = \{x_{t-L:t}^{(1)}, \dots, x_{t-L:t}^{(C)}\} \in \mathbb{R}^{L \times C}$, where x_t refer to the values at timestamp t , L is the length of historical window and C is the number of channels. The objective of time series forecasting is to predict future series $\{y_{t+1:t+H}^{(i)}\}_{i=1}^C = \{y_{t+1:y+H}^{(1)}, \dots, y_{t+1:y+H}^{(C)}\} \in \mathbb{R}^{H \times C}$ by leveraging the forecasting model \mathcal{M} to learn patterns from historical data, where H is the future horizon:

$$\{y_{t+1:t+H}^{(i)}\}_{i=1}^C = \mathcal{M}(\{x_{t-L:t}^{(i)}\}_{i=1}^C) \quad (1)$$

3.2 Time Series Normalization

Given a normalization method \mathcal{N} , it normalizes historical time series before inputting them into \mathcal{M} to eliminate distribution differences, and then de-normalizes the time series output by \mathcal{M} to restore distribution information:

$$\{y_{t+1:t+H}^{(i)}\}_{i=1}^C = \mathcal{N}^{-1}(\mathcal{M}(\mathcal{N}(\{x_{t-L:t}^{(i)}\}_{i=1}^C))) \quad (2)$$

Since statistics often have additional networks in \mathcal{N} for adaptive learning, the statistics adopted by \mathcal{N} and \mathcal{N}^{-1} often differ from the originals and are not identical to each other. Take mean-variance normalization as an example. μ_t is the mean of historical time series $\{x_{t-L:t}^{(i)}\}_{i=1}^C$, and σ_t is its variance. The subscripts l and h denote the normalization and de-normalization stages respectively, where the statistics are either preserved (RevIN) or relearned (Dish-TS, SAN), depending on the normalization strategy. Equation 2 can be expressed as:

$$y_{t+1:t+H}^{(i)} = \sigma_{h,t}^{(i)} \mathcal{M}\left(\frac{x_{t-L:t}^{(i)} - \mu_{l,t}^{(i)}}{\sigma_{l,t}^{(i)}}\right) + \mu_{h,t}^{(i)} \quad (3)$$

4 Methodology

4.1 Overview

In Figure 3, the mainstream paradigm in time series forecasting follows a normalization–forecasting–denormalization pipeline. In this framework, normalization modules (e.g., RevIN, SAN) serve as plug-in components to either remove or learn distributional information, while forecasting models (e.g., SparseTSF, iTransformer) constitute the backbone for capturing temporal patterns.

Affine Prototype-Timestamp (APT) is a lightweight plugin module designed to break through the limitations of local statistical normalization strategies. Within the forecasting pipeline, the forward transformation of APT is applied after normalization and before model input, and the inverse transformation (de-APT) is applied after model output and before de-normalization.

Instead of learning a single set of affine parameters γ and β , APT leverages global timestamps to adaptively assign distinct affine parameters to different subseries. The goal is to reintroduce learnable distributional features, conditioned on global constraints, after distributional features have been suppressed by normalization or disrupted due to distribution shift. The following sections detail the implementation and training procedure of APT.

4.2 Affine Prototype-Timestamp

To address global distribution shift and reduce reliance on local statistical normalization, we propose APT, a module that learns adaptive affine parameters conditioned on timestamp representations. Timestamps, commonly formatted as %Y-%m-%d %H:%M, are widely available and encode rich temporal semantics, making them a natural proxy for global distributional regularities.

Given the input historical sequence $\{x_{t-L:t}^{(i)}\}_{i=1}^C$ and forecast horizon H , we extract the start timestamp t_{st-L} of the history and the end timestamp t_{st+H} of the forecasting horizon for each sample. To obtain semantically meaningful representations, we discretize timestamps into categorical attributes, such as “Time in Day” (TiD) and “Day in Week” (DiW), which are broadly applicable and empirically correlated with global temporal variation.

Each timestamp t is represented by the sum of its corresponding attribute embeddings \mathbf{T}_t :

$$\mathbf{T}_t = \mathbf{T}_t^{\text{TiD}} + \mathbf{T}_t^{\text{DiW}} \quad (4)$$

However, discrete timestamp combinations can be sparse and fail to generalize across unseen scenarios. Inspired by few-shot learning, we replace raw timestamp embeddings with learnable prototypes that capture shared temporal semantics. As a result, we match each timestamp embedding $\mathbf{T}_t \in \mathbb{R}^D$ against a shared learnable prototype library $\mathbf{P} = \{\mathbf{p}_j \in \mathbb{R}^D\}_{j=1}^N$ via inner product similarity:

$$\mathbf{S}_t = \mathbf{T}_t \mathbf{P}^\top \quad (5)$$

We then select the top- k most similar prototypes per timestamps and compute their softmax similarity scores:

$$\mathbf{w}_{t,[1:k]} = \text{Softmax}(\mathbf{S}_{t,[1:k]}) \quad (6)$$

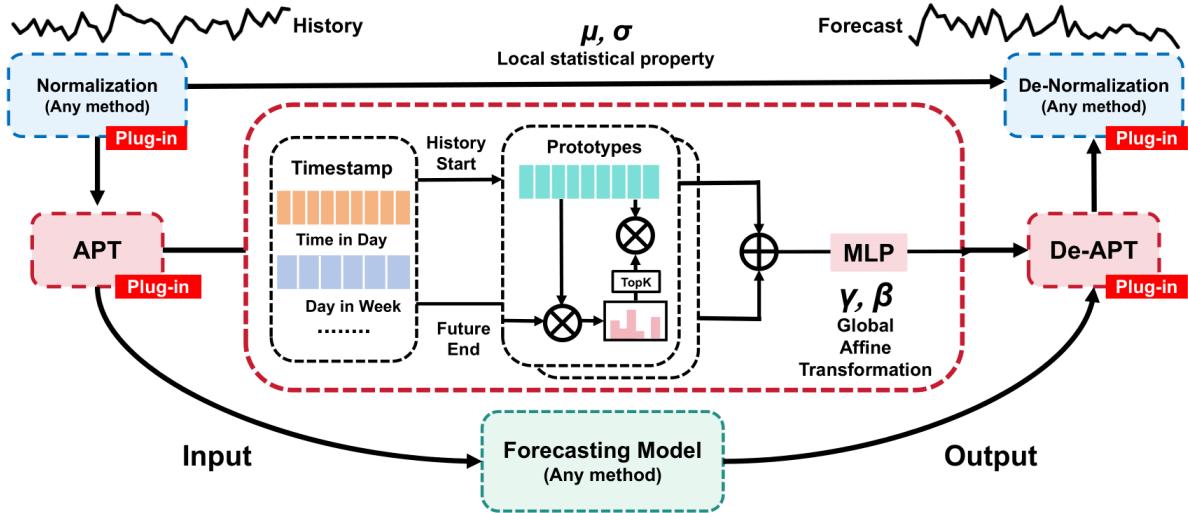


Figure 3: The pipeline of time series forecasting and the schematic of APT.

This yields a sparse weight matrix $\mathbf{W}_t \in \mathbb{R}^N$, where only the top- k entries per row are non-zero. The final timestamp representation is formed via a weighted aggregation over the prototype embeddings:

$$\tilde{\mathbf{T}}_t = \sum_{j=1}^N \mathbf{W}_{t,j} \mathbf{p}_j \quad (7)$$

To encode both past and future temporal context, we aggregate their embeddings: $\tilde{\mathbf{T}}_t = \tilde{\mathbf{T}}_t^{history} + \tilde{\mathbf{T}}_t^{future}$. Optionally, channel identity embeddings $\mathbf{ID} \in \mathbb{R}^{C \times D}$ can be added to incorporate per-channel variation when necessary.

Next, we employ two multi-layer perceptrons to map the aggregated embedding to channel-wise affine parameters $\gamma, \beta \in \mathbb{R}^{|C| \times 1}$, where $\{C\}$ denotes the option of whether to adopt identity embeddings:

$$\gamma_t, \beta_t = \text{MLP}_\gamma(\tilde{\mathbf{T}}_t), \text{MLP}_\beta(\tilde{\mathbf{T}}_t) \quad (8)$$

These learned parameters modulate the normalized time series before and after the forecasting model. Substituting them into the normalization pipeline in Equation 3, the revised inference process becomes:

$$y_{t+1:t+H}^{(i)} = \frac{\sigma_{h,t}^{(i)}}{\gamma_t^{(i)}} (\mathcal{M}(\gamma_t^{(i)} \frac{x_{t-L:t}^{(i)} - \mu_{l,t}^{(i)}}{\sigma_{l,t}^{(i)}} + \beta_t^{(i)}) - \beta_t^{(i)}) + \mu_{h,t}^{(i)} \quad (9)$$

This formulation enables globally informed affine transformations that better adapt to distributional variations across time.

4.3 Training Strategy

Although APT is conceptually simple, learning timestamp-conditioned affine parameters resembles self-supervised clustering, potentially forming a *bi-level* optimization problem (Gould et al. 2016), similar to SAN (Liu et al. 2023).

To stabilize training and ensure effective learning of timestamp semantics, we first freeze the normalization and backbone, and train APT with additional epochs using the following self-supervised losses:

Orthogonal Loss. To ensure diversity among timestamp and prototype embeddings, we encourage orthogonality within $\mathbf{E} = \{\mathbf{T}^{\text{TiD}}, \mathbf{T}^{\text{DiW}}, \mathbf{P}\}$. The loss is defined as:

$$Loss_{orth} = \|\mathbf{E}^\top \mathbf{E} \odot (1 - \mathbf{I})\|_2^2 + \|\mathbf{I} - \mathbf{E}^\top \mathbf{E} \odot \mathbf{I}\|_2^2 \quad (10)$$

Where \odot is the Hadamard product, \mathbf{I} is the identity matrix, and $\|\cdot\|_2$ is the L2 norm. $Loss_{orth}$ penalizes correlation between embeddings and enforces unit-norm behavior, following Barlow Twins (Zbontar et al. 2021).

Load Balancing Loss. Prototype matching may lead to imbalanced usage, where few prototypes dominate. Inspired by MoE imbalance issues (Wang et al. 2024b). To mitigate this, we introduce a load balancing loss that encourages a uniform distribution over prototype usage. This method belongs to batch-wise loss, where we penalize the prototype weights $\mathbf{W} \in \mathbb{R}^{B \times N}$ in each batch in Equation 7:

$$Loss_{balance} = \sum_{j=1}^N \left(\frac{\sum_{i=1}^B \mathbf{W}_{i,j}}{\sum_{i=1}^B \sum_{j=1}^N \mathbf{W}_{i,j}} - \frac{1}{N} \right)^2 \quad (11)$$

Affine Regularization Loss. To prevent trivial or unstable affine parameters, we follow self-supervised representation regularization strategies (Ermolov et al. 2021; Bardes, Ponce, and Lecun 2022). Taking γ as an example, the loss is:

$$Loss_R = \frac{1}{B} \left(1 - \sqrt{\sum_{i=1}^B \gamma_i^2} \right)^2 + \frac{1}{B} \left(\sum_{i=1}^B \gamma_i \right)^2 \quad (12)$$

Training Processing. In the additional epochs of freezing the backbone and normalization strategy, the overall pre-training objective for APT is:

$$Loss_{APT} = Loss_{orth} + Loss_{balance} + Loss_R \quad (13)$$

Once pretrained, APT is jointly optimized with the normalization and forecasting modules under standard regression loss (e.g., MSE). If required by the normalization strategy (e.g., SAN), its loss can be included:

$$Loss_{main} = Loss_{normal} + Loss_{MSE} \quad (14)$$

Methods		CATS				Informer				iTransformer				SparseTSF			
Affine				+APT				+APT				+APT				+APT	
Metrics		MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
ECL	None	0.264	0.168	0.262	0.166	0.426	0.369	0.408	0.332	0.270	0.169	0.265	0.165	0.269	0.173	0.267	0.171
	+RevIN	0.258	0.165	0.259	0.166	0.337	0.252	0.326	0.233	0.259	0.164	0.259	0.162	0.265	0.173	0.264	0.173
	+Dish-TS	0.272	0.173	0.270	0.172	0.379	0.305	0.371	0.297	0.266	0.166	0.261	0.161	0.272	0.173	0.267	0.171
	+SAN	0.277	0.175	0.276	0.175	0.349	0.266	0.324	0.231	0.258	0.159	0.261	0.160	0.271	0.172	0.266	0.165
	+FAN	0.265	0.167	0.263	0.165	0.266	0.165	0.264	0.166	0.274	0.172	0.268	0.169	0.263	0.166	0.261	0.164
ETTh1	None	0.469	0.486	0.436	0.431	0.891	1.255	0.783	1.134	0.499	0.503	0.463	0.457	0.439	0.436	0.437	0.435
	+RevIN	0.443	0.448	0.431	0.427	0.597	0.715	0.562	0.657	0.473	0.485	0.441	0.441	0.429	0.427	0.427	0.424
	+Dish-TS	0.475	0.485	0.452	0.456	0.797	1.062	0.721	0.947	0.498	0.513	0.492	0.495	0.467	0.477	0.456	0.461
	+SAN	0.451	0.472	0.447	0.463	0.542	0.614	0.528	0.599	0.466	0.477	0.453	0.470	0.504	0.564	0.464	0.488
	+FAN	0.478	0.482	0.473	0.478	0.519	0.533	0.506	0.529	0.487	0.497	0.477	0.480	0.484	0.497	0.480	0.488
ETTh2	None	0.504	0.533	0.471	0.486	1.433	2.883	1.194	2.269	0.686	0.863	0.493	0.507	0.492	0.515	0.484	0.498
	+RevIN	0.425	0.392	0.417	0.390	0.565	0.658	0.504	0.531	0.439	0.421	0.422	0.393	0.427	0.399	0.428	0.470
	+Dish-TS	0.476	0.466	0.444	0.419	1.321	3.261	0.795	1.318	0.537	0.587	0.485	0.497	0.631	0.933	0.533	0.612
	+SAN	0.438	0.406	0.433	0.404	0.718	0.939	0.447	0.438	0.438	0.418	0.414	0.410	0.559	0.619	0.485	0.478
	+FAN	0.477	0.487	0.470	0.470	0.565	0.624	0.489	0.499	0.492	0.512	0.483	0.497	0.475	0.467	0.472	0.473
Exchange	None	0.617	0.888	0.415	0.316	0.996	1.640	0.920	1.337	0.655	0.783	0.551	0.599	0.426	0.352	0.418	0.342
	+RevIN	0.435	0.401	0.424	0.381	0.592	0.587	0.519	0.460	0.482	0.468	0.447	0.416	0.481	0.478	0.438	0.386
	+Dish-TS	0.562	0.774	0.493	0.468	0.930	2.105	0.704	1.003	0.509	0.496	0.476	0.395	0.540	0.531	0.502	0.476
	+SAN	0.483	0.521	0.415	0.372	0.457	0.410	0.404	0.338	0.496	0.499	0.457	0.448	0.418	0.350	0.407	0.356
	+FAN	0.492	0.458	0.482	0.457	0.562	0.591	0.533	0.528	0.513	0.502	0.486	0.448	0.473	0.443	0.462	0.419
Traffic	None	0.288	0.554	0.286	0.530	0.425	0.830	0.402	0.780	0.594	0.986	0.428	0.830	0.298	0.443	0.296	0.443
	+RevIN	0.284	0.421	0.280	0.416	0.457	0.874	0.394	0.753	0.289	0.412	0.293	0.415	0.295	0.445	0.295	0.445
	+Dish-TS	0.293	0.438	0.278	0.417	0.445	0.861	0.403	0.761	0.305	0.429	0.309	0.430	0.314	0.461	0.310	0.457
	+SAN	0.292	0.432	0.291	0.437	0.419	0.753	0.394	0.716	0.294	0.427	0.297	0.431	0.407	0.651	0.363	0.569
	+FAN	0.316	0.454	0.301	0.438	0.315	0.457	0.301	0.445	0.340	0.468	0.324	0.454	0.302	0.432	0.298	0.431
1 st count		4	6	26	24	0	1	30	29	7	6	24	25	3	11	28	23

Table 2: Results of the main experiment

5 Experiments

5.1 Experimental Setup

Baselines. As a plug-in designed to mitigate distribution shift, APT is compatible with any time series forecasting backbone and normalization strategy. To validate its effectiveness, we select Informer (Zhou et al. 2021), iTransformer (Liu et al. 2024), SparseTSF (Lin et al. 2024), and CATS (Kim et al. 2024) as the backbone, and RevIN (Kim et al. 2021), Dish-TS (Fan et al. 2023), SAN (Liu et al. 2023), and FAN (Ye et al. 2024) as normalization strategies.

Datasets & Metrics. We select six datasets for main experiments: ECL, ETTh1, ETTh2, Exchange, Traffic, Weather. The historical length L is 336, the main experiments report average results over forecasting lengths $H = \{96, 192, 336, 720\}$, and the average value is reported, while extra experiments use $H = 336$. Metrics include mean squared error (MSE) and mean absolute error (MAE).

Details. We follow the BasicTS benchmark setup (Shao et al. 2024). APT uses “Day in Week” and “Time in Day” as timestamp features, except on Exchange (only “Day of Week”). The number of prototypes depends on sampling

rate: 5 for daily, 30 for hourly, and 40 for 10-minute frequency. All embeddings are 20-dimensional with MLP hidden size 32, yielding 1.5K–5K parameters. APT trains with a learning rate of 5e-5 for 1–5 extra epochs until convergence. The optimizer is shared with other modules, with gradient updates controlled by loss weight λ . More details are in Appendix B and C.

5.2 Main Results

On datasets with strong distribution shifts like ETTh1/2 and Exchange, APT yields 4%–40% performance gains across diverse backbone–normalization combinations. Combined with robust normalization, even the weaker Informer matches or surpasses state-of-the-art backbones.

CATS, iTransformer, and SparseTSF are among the most advanced backbones, while RevIN, FAN and SAN are the most effective normalization strategies. Even on stable datasets like ECL, where performance typically plateaus, APT still yields consistent gains and often achieves the best overall results with negligible risk of degradation. More and combined with data analysis are provided in Appendix C.4.

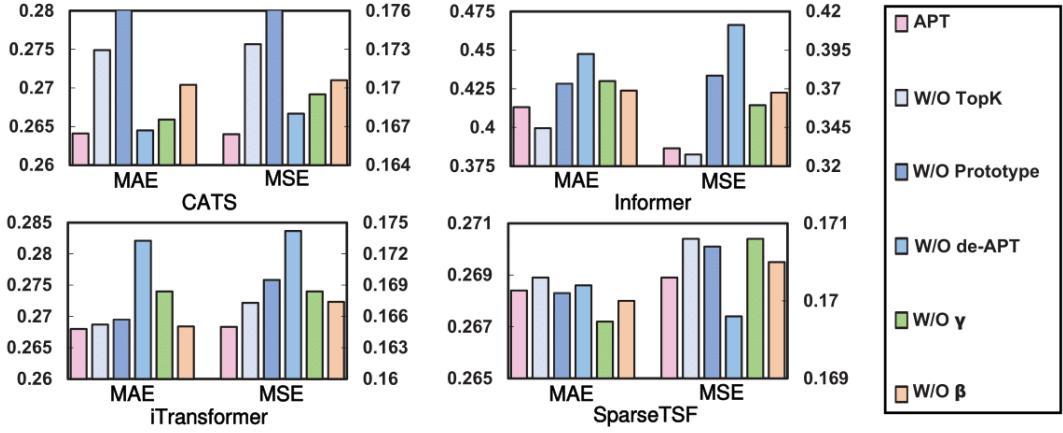


Figure 4: The ablation study results of APT components on the ECL dataset, $L = 336$, $H = 336$.

5.3 Ablation Study

In Figure 4 and Appendix C.7, we conduct ablations from two perspectives: component-wise and training strategy.

Component Ablation: We evaluate APT by removing key components:

- **W/O Top- k :** Remove Top- k from Equation 6;
- **W/O Prototype:** Use raw timestamp embeddings without prototype matching;
- **W/O de-APT:** Remove inverse transformation of APT;
- **W/O γ or β :** Remove scaling or bias in affine transformation;

Removing **de-APT** degrades performance, as it forces the model to reconstruct distributional properties. Top- k is similar to expert activation and mitigates feature over-smoothing by enforcing sparsity during prototype aggregation. In contrast, prototype learning alleviates the challenges of few-shot learning by compressing timestamp semantics, improving robustness and preventing overfitting. Between affine components, γ is more crucial than the bias β under distribution shift, because mean drift will generate abundant supervisory signals to guide the backbone to adapt.

Training Strategy Ablation: APT learns timestamp-aware affine parameters via self-supervised pretraining, while freezing backbone and normalization. Table 3 and Appendix C show the effect of ablating components:

- **W/O λ :** Remove the loss weighting coefficient from APT pre-training, sharing learning rate with the backbone;
- **W/O $Loss_{orth}$ or $Loss_{balance}$ or $Loss_R$:** Remove orthogonality, load balancing, or affine regularization loss;
- **W/O $Loss_{APT}$:** Skip pretraining entirely, optimizing APT only with the main loss.

Recommended learning rates for {CATS, Informer, iTransformer, SparseTSF} are {5e-3, 2e-4, 5e-4, 2e-3}, with APT pretraining fixed at 5e-5. Accordingly, λ is set to {5e-2, 2.5e-1, 1e-1, 2.5e-2}. Without λ , APT shares the backbone’s rate, causing overfitting and unstable timestamp semantics.

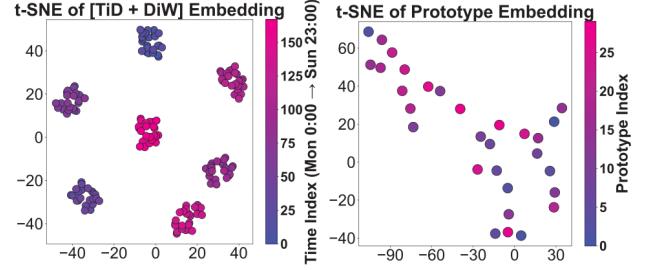


Figure 5: Visualization of APT timestamps and prototype embeddings on ECL dataset and iTransformer

Each pretraining loss targets a distinct goal: $Loss_{orth}$ encourages embedding diversity, $Loss_{balance}$ prevents feature collapse (Hua et al. 2021), and $Loss_R$ stabilizes convergence for regression compatibility. As they govern embeddings, prototype matching, and parameter scaling respectively, these losses are complementary and essential for APT’s stability. Omitting pretraining altogether (W/O $Loss_{APT}$) causes APT to fail, with performance reverting to backbone levels.

5.4 Visualization

t-SNE of Embedding: Figure 5 shows the timestamp and prototype embeddings learned by APT with iTransformer on the ECL dataset. Timestamp embeddings form distinct clusters aligned with “Day in Week,” while “Time in Day” varies orthogonally in Appendix C.9. Prototypes are also well-separated, but their embeddings still preserve certain timestamp-related structure as orthogonal losses serve as soft supervision.

Forecasting cases: Figure 6 presents forecasting cases on the 6th channel of ETTh2 without normalization. While most backbones, except Informer, capture periodic patterns well, they fail to handle distribution shifts over time. APT mitigates this issue by applying timestamp-conditioned affine transformations, independently of the backbone, to better align predictions with data distributions.

Methods		APT		W/O λ		W/O $Loss_{orth}$		W/O $Loss_{balance}$		W/O $Loss_R$		W/O $Loss_{APT}$	
Metrics		MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
ECL	CATS	0.264	0.166	0.269	0.169	0.265	0.167	0.266	0.169	0.265	0.168	0.267	0.170
	Informer	0.413	<u>0.332</u>	0.441	0.379	0.429	0.350	0.412	0.329	0.436	0.377	0.453	0.395
	iTransformer	0.268	0.165	0.269	0.169	0.268	0.165	0.269	0.167	0.269	0.168	0.270	0.168
	SparseTSF	0.268	0.170	0.270	0.171	0.269	0.171	0.269	0.171	0.269	0.171	0.268	0.171
ETTh1	CATS	0.437	0.443	0.451	0.458	0.438	0.446	0.448	0.460	0.439	0.446	0.448	0.459
	Informer	0.797	1.121	0.856	1.249	0.839	1.215	1.009	1.495	0.787	1.121	0.818	1.198
	iTransformer	0.474	0.478	0.472	0.481	0.472	0.479	0.469	0.474	0.487	0.498	0.483	0.491
	SparseTSF	0.438	0.451	0.447	0.458	0.441	0.453	0.448	0.460	0.441	0.454	0.448	0.459
Exchange	CATS	0.402	0.276	0.411	0.286	0.407	0.290	0.406	0.278	0.402	0.276	0.560	0.500
	Informer	0.976	<u>1.490</u>	1.144	2.070	1.009	1.789	<u>1.009</u>	1.506	0.989	1.490	1.082	1.734
	iTransformer	0.502	0.445	0.634	0.6549	0.629	0.669	0.559	0.531	0.544	0.508	0.655	0.785
	SparseTSF	0.431	0.316	0.432	0.321	0.443	0.333	0.431	0.316	0.439	0.330	0.448	0.343
Weather	CATS	0.283	0.242	0.287	0.243	0.300	0.246	0.300	0.247	0.295	<u>0.243</u>	0.301	0.246
	Informer	0.315	0.270	0.315	<u>0.271</u>	0.326	0.284	0.328	0.291	0.410	0.389	0.554	0.816
	iTransformer	0.293	0.251	0.296	<u>0.254</u>	0.293	0.251	<u>0.295</u>	0.254	0.296	<u>0.254</u>	0.300	0.257
	SparseTSF	0.312	0.267	0.314	0.267	0.320	0.270	0.322	0.271	0.320	0.270	0.319	0.269

Table 3: Results of ablation studies on training strategies across four datasets. $L = 336, H = 336$

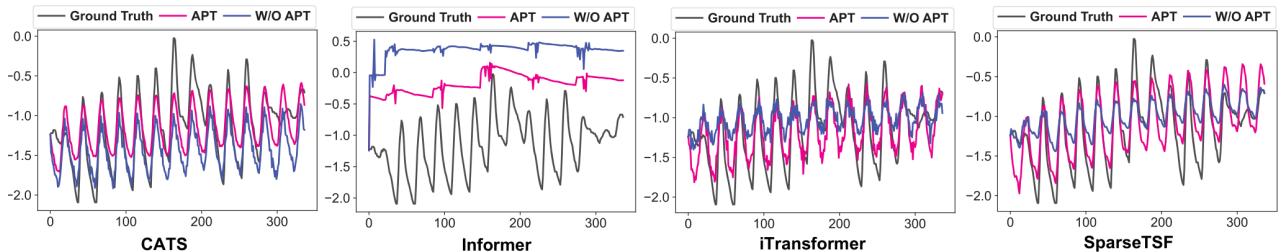


Figure 6: Visualization of forecasting results for different models on the ETTh2 dataset without normalization strategy

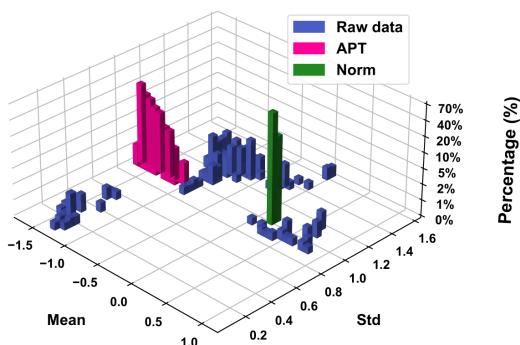


Figure 7: 3D visualization of temporal distribution and ratio of forecasting pipeline at different stages on ETTh1.

Temporal distribution: As shown in Figure 7, raw data exhibit severe distribution shift and its distributional features are disrupted, which is the phenomenon we emphasized earlier and it can cause great damage to backbone’s learning of temporal patterns. Normalization maps series to zero mean and unit variance; while RevIN decouples distribution from patterns to improve robustness, it also suppresses informa-

tive distribution. In contrast, APT uses timestamps to encode global distribution features, forming compact yet discriminative representations adaptable to temporal shift.

Due to space limitations, the complete results and more experiments are provided in Appendix C to better understand APT: data motivation (C.2), other plug-ins (C.3), extra main results (C.4), hyper-Parameter (C.5), parameter count (C.6), extra ablation study (C.7), cross-setting fine-tuning (C.8), visualization (C.9) and discussion (D).

6 Conclusion

In this work, we propose Affine Prototype-Timestamp (APT), a lightweight and model-agnostic plug-in to enhance time series forecasting under distribution shifts. APT employs discretized timestamps and prototype learning to introduce timestamp-conditioned affine transformations, enabling forecasting models to recover global distributional features that may be suppressed by normalization or distorted by distribution shifts.

In future work, we will explore online adaptation for streaming data and incorporate external modalities such as text or images to enhance shift awareness, further strengthening the real-world applicability of deep forecasting models in domains like weather, finance, and energy.

Acknowledgment

This work is supported by the NSFC under Grant Nos. 62372430 and 62502505, the Youth Innovation Promotion Association CAS No.2023112, the Postdoctoral Fellowship Program of CPSF under Grant Number GZC20251078, the China Postdoctoral Science Foundation No.2025M771542 and HUA Innovation fundings.

References

- Bardes, A.; Ponce, J.; and Lecun, Y. 2022. VICReg: Variance-Invariance-Covariance Regularization For Self-Supervised Learning. In *ICLR*.
- Bebis, G.; Georgopoulos, M.; da Vitoria Lobo, N.; and Shah, M. 1999. Learning affine transformations. *Pattern recognition*, 32(10): 1783–1799.
- Dumoulin, V.; Shlens, J.; and Kudlur, M. 2017. A Learned Representation For Artistic Style. In *ICLR*.
- Ermolov, A.; Siarohin, A.; Sangineto, E.; and Sebe, N. 2021. Whitening for self-supervised representation learning. In *ICML*, 3015–3024. PMLR.
- Fan, W.; Wang, P.; Wang, D.; Wang, D.; Zhou, Y.; and Fu, Y. 2023. Dish-ts: a general paradigm for alleviating distribution shift in time series forecasting. In *AAAI*, volume 37, 7522–7529.
- Ferté, T.; Dutartre, D.; Hejblum, B. P.; Griffier, R.; Jouhet, V.; Thiébaut, R.; Legrand, P.; and Hinault, X. 2024. Reservoir Computing for Short High-Dimensional Time Series: an Application to SARS-CoV-2 Hospitalization Forecast. In *ICML*, 13570–13591. PMLR.
- Fu, Y.; Shao, Z.; Yu, C.; Li, Y.; An, Z.; Wang, Q.; Xu, Y.; and Wang, F. 2025. Selective Learning for Deep Time Series Forecasting. *arXiv preprint arXiv:2510.25207*.
- Gould, S.; Fernando, B.; Cherian, A.; Anderson, P.; Cruz, R. S.; and Guo, E. 2016. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*.
- Han, L.; Ye, H.-J.; and Zhan, D.-C. 2024. SIN: selective and interpretable normalization for long-term time series forecasting. In *ICML*.
- Hua, T.; Wang, W.; Xue, Z.; Ren, S.; Wang, Y.; and Zhao, H. 2021. On feature decorrelation in self-supervised learning. In *CVPR*, 9598–9608.
- Huang, X.; and Belongie, S. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 1501–1510.
- Karras, T.; Laine, S.; and Aila, T. 2019. A style-based generator architecture for generative adversarial networks. In *CVPR*, 4401–4410.
- Kim, D.; Park, J.; Lee, J.; and Kim, H. 2024. Are Self-Attentions Effective for Time Series Forecasting? In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.
- Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.-H.; and Choo, J. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *ICLR*.
- Li, T.; Li, Z.; Rockwell, H.; Farimani, A.; and Lee, T. S. 2022. Prototype memory and attention mechanisms for few shot image generation. In *ICLR*, volume 18.
- Li, Y.; Shao, Z.; Xu, Y.; Qiu, Q.; Cao, Z.; and Wang, F. 2024. Dynamic frequency domain graph convolutional network for traffic forecasting. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5245–5249. IEEE.
- Li, Y.; Zezhi, S.; Yu, C.; Qian, T.; Zhang, Z.; Du, Y.; He, S.; Wang, F.; and Xu, Y. 2025. STA-GANN: A Valid and Generalizable Spatio-Temporal Kriging Approach. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, 1726–1736.
- Lin, S.; Lin, W.; Wu, W.; Chen, H.; and Yang, J. 2024. SparseTSF: Modeling Long-term Time Series Forecasting with* 1k* Parameters. In *ICML*.
- Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *ICLR*.
- Liu, Z.; Cheng, M.; Li, Z.; Huang, Z.; Liu, Q.; Xie, Y.; and Chen, E. 2023. Adaptive normalization for non-stationary time series forecasting: A temporal slice perspective. *NeurIPS*, 36: 14273–14292.
- Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *ICLR*.
- Perez, E.; Strub, F.; De Vries, H.; Dumoulin, V.; and Courville, A. 2018. Film: Visual reasoning with a general conditioning layer. In *AAAI*, volume 32.
- Shao, Z.; Li, Y.; Wang, F.; Yu, C.; Fu, Y.; Qian, T.; Xu, B.; Diao, B.; Xu, Y.; and Cheng, X. 2025. Blast: Balanced sampling time series corpus for universal forecasting models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining* V. 2, 2502–2513.
- Shao, Z.; Wang, F.; Xu, Y.; Wei, W.; Yu, C.; Zhang, Z.; Yao, D.; Sun, T.; Jin, G.; Cao, X.; et al. 2024. Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis. *TKDE*, 37(1): 291–305.
- Shao, Z.; Zhang, Z.; Wang, F.; Wei, W.; and Xu, Y. 2022. Spatial-temporal identity: A simple yet effective baseline for multivariate time series forecasting. In *CIKM*, 4454–4458.
- Wang, C.; Qi, Q.; Wang, J.; Sun, H.; Zhuang, Z.; Wu, J.; and Liao, J. 2024a. Rethinking the power of timestamps for robust time series forecasting: A global-local fusion perspective. *NeurIPS*, 37: 22206–22232.
- Wang, F.; Li, Y.; Shao, Z.; Yu, C.; Fu, Y.; An, Z.; Xu, Y.; and Cheng, X. 2025a. ARIES: Relation Assessment and Model Recommendation for Deep Time Series Forecasting. *arXiv preprint arXiv:2509.06060*.
- Wang, H.; Pan, L.; Chen, Z.; Yang, D.; Zhang, S.; Yang, Y.; Liu, X.; Li, H.; and Tao, D. 2025b. FreDF: Learning to Forecast in the Frequency Domain. In *ICLR*.

Wang, L.; Gao, H.; Zhao, C.; Sun, X.; and Dai, D. 2024b. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*.

Ye, W.; Deng, S.; Zou, Q.; and Gui, N. 2024. Frequency Adaptive Normalization For Non-stationary Time Series Forecasting. In *NeurIPS*.

Yu, C.; Wang, F.; Shao, Z.; Qian, T.; Zhang, Z.; Wei, W.; An, Z.; Wang, Q.; and Xu, Y. 2025. GinAR+: A Robust End-To-End Framework for Multivariate Time Series Forecasting with Missing Values. *TKDE*.

Zbontar, J.; Jing, L.; Misra, I.; LeCun, Y.; and Deny, S. 2021. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, 12310–12320. PMLR.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *AAAI*, volume 37, 11121–11128.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, volume 35, 11106–11115.

Ziegler, Z. M.; Melas-Kyriazi, L.; Gehrmann, S.; and Rush, A. M. 2019. Encoder-agnostic adaptation for conditional language generation. *arXiv preprint arXiv:1908.06938*.

A Notation

Notation	Description
x	Time series, especially historical parts
L	Length of historical time series
y	Time series, especially future parts
H	Length of future time series
\mathcal{M}	Deep Time Series Forecasting Model
\mathcal{N}	Time Series Normalization Strategy
t	Subscript, representing a certain moment in time
ts	Timestamp information
\mathbf{T}	Timestamp embedding
\mathbf{P}	Prototype embedding, $\mathbf{p} \in \mathbf{P}$.
\mathbf{S}	Similarity matrix
\mathbf{I}	Identity matrix
\mathbf{E}	General term for T and P
\mathbf{W}	Weights used for prototype weighting
μ	Mean
σ	Variance
γ	Affine parameters for scaling
β	Affine parameters for bias
λ	Learning rate factor for APT typically $(5e-5)/lr$, lr is learning rate of MSE

Table 4: Explanation of notations leveraged in APT

B Baselines

B.1 Deep Time Series Forecasting Model

CATS (Kim et al. 2024) is a time series forecasting model constructed entirely by cross-channel attention. It adopts the Patch strategy (Nie et al. 2023), uses future-related parameters as queries and masks potentially interfering temporal information during forecasting. Inspired by the effectiveness of linear model forecasts (Zeng et al. 2023), CATS strikes a balance between efficiency and modeling complex temporal dependencies.

Informer (Zhou et al. 2021) is one of the pioneers in the field of deep time series forecasting. As a classic work, it reflects on the efficiency issues of Transformers in time series forecasting, proposes a sparse attention mechanism, and achieves effective prediction.

iTransformer (Liu et al. 2024) addresses the limitations of previous work, which treated the time series dimension as a temporal token while ignoring inter-channel correlations, and the efficiency issues arising from increasing forecast lengths. To address these issues, it proposes treating different channels as tokens and applying an attention mechanism between channels, thereby achieving more effective forecasts than previous models.

SparseTSF (Lin et al. 2024) is a state-of-the-art linear forecasting model that further reduces the number of parameters. It divides time series into multiple periods through downsampling, and effectively forecasts time series through cross-period forecasting with parameter sharing and upsampling.

Factors for selecting those backbones: Informer is a pioneering work in long-term time series forecasting. As a classic forecasting model, it incorporates two temporal modeling strategies: Transformer and channel dependency. Even though its performance is limited in other works, we can prove that with the help of advanced normalization and APT, it can still achieve SOTA performance in many cases.

SparseTSF inherits from DLinear and offers better performance. It includes strategies such as MLP, channel independence, and patch. iTransformer is based on Transformer and has a channel interaction strategy that captures spatial information and temporal patterns better than Informer’s channel dependency. Compared to iTransformer, CATS uses a more refined channel interaction method with cross-attention, and also has the patch strategy.

B.2 Time Series Normalization Strategy

RevIN (Kim et al. 2021) is a classic work that alleviates shift in time series forecasting. Time series are mutually coupled with patterns and distributions. Forecasting models mainly learn pattern information related to temporal dependencies, while distributions are considered components unrelated to forecasting. By separating distribution information before inputting it into the model through a normalization strategy and restoring that information after forecasting through inverse normalization, RevIN effectively improves the forecasting capabilities of various models.

DishTS (Fan et al. 2023) believes that time series shift is constantly changing, not only within the time series, but also between the history and the future. To this end, it proposes using additional network learning distribution statistics and forecasting the future statistics required for inverse normalization.

SAN (Liu et al. 2023) thinks that the scenarios explored by Dish-TS are still not comprehensive enough. The distribution of time series continues to shift, and the length of history and future has exceeded the scope of effective description. Therefore, it further compresses the learning of drift information to the patch level and attempts to learn the shift flow within a single sample.

FAN (Ye et al. 2024) actually switches the time series shift from a distribution perspective to time series decomposition. Non-stationary information is considered to be minor components that are difficult to learn effectively. Therefore, FAN uses adaptive filtering to input only information at the main frequency into the model, adopts the residuals from the additional network information, and adds residual forecasting terms after model forecasting.

PS: It should be noted that, with the exception of Informer, other forecasting models originally come with RevIN due to its irreplaceable performance gains, or do not have an affine transformation version. In the APT test, we

first completely separated the normalization strategy from the model itself. When testing the addition of RevIN, we adopt the original version containing affine transformation.

C Experiments

C.1 Experimental details

Our experiments fully adopt the default configuration of the public-source and fair benchmark BasicTS¹, including ADAM as the default optimizer, with each model having its own dedicated parameter configuration file for each dataset. All experiments are conducted on a single NVIDIA GeForce RTX 4090 GPU, with an Intel(R) Xeon(R) Gold 6338 CPU @ 2.00GHz, and each experiment is limited to 4 threads.

C.2 Dataset Information

ETTh1&h2²: ETT is a series of datasets, where "1" and "2" represent different transformers, "h" represents the sampling rate per hour. APT does not adopt its 15-minute sampling rate dataset. The first six variables in each ETT data set are overload data, and the last one is oil temperature.

ECL³: ECL, i.e. the Electricity dataset, records the hourly electricity consumption of 321 clients between 2012 and 2014 (unit: kilowatt-hours).

Exchange³: The ExchangeRate datasets includes daily exchange rates for eight countries: Australia, the United Kingdom, Canada, Switzerland, China, Japan, New Zealand, and Singapore.

Traffic⁴: Traffic is a collection of hourly data provided by the California Department of Transportation that describes road usage measured by various sensors on highways in the San Francisco Bay Area.

Weather⁴: Weather records every 10 minutes throughout 2020 from Autoformer, including 21 meteorological indicators such as air temperature and humidity.

Table 5 presents key information about the datasets, particularly the sampling rates. The primary parameters of APT are configured based on the sampling rate, including the timestamps we adopt, the number of prototypes, and top- k . Furthermore, we show the parameter count of APT under the setting of channel-shared affine parameters, which is often significantly smaller than that of time series forecasting models. In experiments utilizing channel-wise affine parameters, the additional parameter count depends on the dataset's channel count and the embedding dimension (fixed at 20), and the increase remains minimal.

Motivation verification from data perspective: One of the motivations for APT is that local statistical normalization cannot address variations within time series, such as missing values or noise.

Noise is affected by random factors from the sensor's own mechanisms and the external environment. As information coupled with each step of time series, noise is typically difficult to detect directly. We refer to BasicTS's discussion on

model information for time series datasets, analyzing pattern stability and distribution drift, and conclude that ECL, Traffic, and Weather are low-noise datasets with stable patterns and low distribution drift, while ETT and ExchangeRate are the opposite.

Missing values often occur in real-world scenarios due to sensor failures. Even benchmarks often contain a large amount of missing data, and prior data processing methods may vary. Due to the lack of more prior knowledge, we have broadly categorized missing value handling into two cases: zero-filling and previous-value-filling:

- Zero-filling: Directly fill missing signals with 0. Note that active filling in the Weather dataset is -9999, and cases where precipitation is 0 have been excluded.
- Previous-value-filling: When the current value cannot be obtained, the previous value is used. Since the probability of recording two completely identical floating-point numbers in succession is very low, we do not perform further exclusions, but otherwise we would misjudge small fluctuations in the values.

The summary of the missing rate is shown in Table 6. We have calculated the two types of missing rates for different datasets, as well as the maximum and minimum missing rates and their channels. The missing rate for ECL, Traffic, and Weather is one level lower than that of other datasets, and its main missing rate is often affected by only a few channels, with most channels remaining almost free of missing values.

On the contrary, the ETT and Exchange datasets not only have fewer channels but also higher minimum missing rates, which implies that the statistical properties of each instance may be significantly affected. However, the impact of previous value filling on mean calculation is limited, but it has a profound detrimental effect on variance.

In summary, the analysis of noise and missing rates indicates that the temporal information in ECL and similar datasets is more stable and reliable, while ETT and Exchange datasets are unreliable and exhibit severe distribution shift. These analyses will support certain phenomena in subsequent performance analysis and ablation studies of APT.

C.3 Plug-in Supplementary Study

In this section, we added two plug-ins unrelated to distribution drift for supplementary study:

- **FreDF** (Wang et al. 2025b): FreDF is a work that explores the autocorrelation of time series from a frequency domain perspective, using an additional frequency domain loss function to enhance the modeling capabilities of time series correlation.
- **GLAFF** (Wang et al. 2024a): GLAFF is a classic study on the role of timestamp information in time series forecasting. It introduces global information to the output of the forecasting pipeline through additional timestamp modeling. Its motivation is completely different from that of APT. GLAFF focuses on modeling global pattern information and weighting it with the forecasting results, while APT improves the normalization and denormalization processes through affine transformations to mitigate

¹<https://github.com/GestaltCogTeam/BasicTS>

²<https://github.com/zhouhaoyi/ETDataset>

³<https://github.com/laiguokun/multivariate-time-series-data>

⁴<https://github.com/thuml/Autoformer>

Datasets	Domain	Period	Frequency	Channel	Split	Timestamps	Prototype	Top K	Param.
ECL	Energy	2012 - 2014	1 hour	321	7:1:2	TiD&DiW	30	3	2.64K
ETTh1&2	Energy	2016/7/1 0:00 - 2018/6/26 19:45	1 hour	7	6:2:2	TiD&DiW	30	3	2.64K
Exchange	Economy	1990 - 2016	1 day	8	7:1:2	DiW	5	2	1.82K
Traffic	Transportation	2016/7/1 02:00 - 2018/7/2 01:00	1 hour	862	7:1:2	TiD&DiW	30	3	2.64K
Weather	Nature	2020/1/1 0:10 - 2021/1/1 0:00	10 mins	21	7:1:2	TiD&DiW	40	4	5.1K

Table 5: Dataset description and corresponding parameters

Datasets	Overall missing rate	0 fill rate	Previous value fill rate	Min. rate and channel	Max. rate and channel
ECL	4.03%	1.08%	2.94%	0.10%	133
ETTh1	7.38%	0.99%	6.38%	3.65%	2
ETTh2	23.68%	9.79%	13.89%	4.24%	0
Exchange	8.60%	0%	8.60%	4.38%	1
Traffic	1.68%	0.89%	0.78%	0.08%	267
Weather	3.39%	0.01%	3.38%	0.05%	17

Table 6: Detailed report on dataset missing rates

Methods	APT			FreTS	GLAFF		
Metrics	MAE	MSE	MAE	MSE	MAE	MSE	
ETTh1	96	0.410	0.399	0.410	0.388	0.414	0.406
	192	0.430	0.432	0.438	0.434	0.462	0.471
	336	0.448	0.457	0.459	0.460	0.479	0.495
	720	0.477	0.475	0.513	0.516	0.542	0.568
	Avg.	0.441	0.441	0.455	0.450	0.475	0.485
Weather	96	0.208	0.158	0.207	0.159	0.211	0.165
	192	0.248	0.201	0.249	0.204	0.250	0.220
	336	0.284	0.250	0.292	0.260	0.288	0.260
	720	0.355	0.323	0.347	0.334	0.350	0.359
	Avg.	0.269	0.233	0.274	0.239	0.275	0.251

Table 7: Performance comparison between APT and other plug-ins on iTransformer. $H = 336$

time series distribution shift, and the dynamic affine parameters provided by APT for each instance have no regression capability.

As shown in Table 7, APT outperforms both FreDF and GLAFF by 2%–10% on ETTh1 and Weather when applied to iTransformer. Moreover, APT achieves better performance with longer forecasting horizons H , implying its stronger global temporal awareness that support long-term dependency modeling. In general, longer prediction horizons exacerbate distribution shift in time series, and this trend aligns well with both the motivation and design of APT timestamps serve better as a tool to alleviate distribution shift than as a pattern modeling signal.

APT’s consistent advantage over GLAFF further supports this view. In our experiments, iTransformer alone contains 6.6M parameters, while GLAFF, as a plug-in, introduces an

additional 1.8M parameters and requires twice the training time compared to APT. In contrast, APT only introduces 2.64K to 5.1K additional parameters and incurs almost no increase in training time. We attribute this inefficiency to GLAFF’s reliance on complex attention mechanisms to capture global timestamp information. Its inferior performance and higher cost inevitably limit its practical applicability and hinder deeper exploration of timestamp-driven time series modeling.

C.4 Main Experiment

The complete results of the main experiment are shown in Tables 12 and 13.

APT yields the most significant improvements on Informer, achieving 3%–50% gains across most datasets. It consistently enhances Informer’s performance when combined with any normalization strategy. Specifically, on ECL and Traffic, combining Informer + APT with FAN achieves performance comparable to or exceeding that of other backbones. On Weather, APT reaches similar results regardless of the normalization strategy used. We consider that this result may affect the role of channel dependence, an early strategy in multivariate time series forecasting. In datasets with stable time series patterns, channel dependence does not show a significant performance gap compared to other methods such as channel independence and interaction, but this still requires further research.

On ETT and Exchange, APT improves performance by 2%–10% across various backbone-normalization combinations, with virtually no cases of degradation. These datasets are known for exhibiting significant distribution shift and containing missing values, which often impair the effectiveness of statistical normalization methods. By leveraging timestamps, APT introduces global temporal distribution awareness and dynamically generates affine parameters, ef-

Methods		APT		+ Double prototype		+ Identity embedding		+ “Day in Month”	
Metrics		MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
ECL	CATS	0.264	0.166	0.269	0.170	0.270	0.170	0.273	0.191
	Informer	0.413	0.332	0.419	0.346	0.512	0.613	0.443	0.389
	iTransformer	0.268	0.165	0.268	0.166	0.270	0.169	0.270	0.169
	SparseTSF	0.268	0.170	0.268	0.170	0.268	0.170	0.268	0.170
ETTh1	CATS	0.437	0.443	0.436	0.442	0.447	0.461	0.450	0.463
	Informer	0.797	1.121	0.790	1.154	0.792	1.293	0.951	1.967
	iTransformer	0.474	0.478	0.470	0.477	0.474	0.482	0.526	0.599
	SparseTSF	0.438	0.451	0.443	0.455	0.437	0.450	0.442	0.454
Exchange	CATS	0.402	0.276	0.406	0.284	0.397	0.265	0.410	0.294
	Informer	0.976	1.490	0.970	1.380	0.947	1.410	1.120	1.884
	iTransformer	0.502	0.445	0.558	0.540	0.477	0.396	0.652	0.743
	SparseTSF	0.431	0.316	0.442	0.333	0.447	0.326	0.459	0.358
Weather	CATS	0.283	0.242	0.308	0.251	0.293	0.246	0.304	0.259
	Informer	0.315	0.270	0.323	0.264	0.364	0.346	0.333	0.702
	iTransformer	0.293	0.251	0.297	0.256	0.303	0.256	0.297	0.257
	SparseTSF	0.312	0.267	0.316	0.270	0.317	0.269	0.317	0.268

Table 8: Results of hyper-parameter sensitivity study across four datasets. $L = 336$, $H = 336$

fectively adapting to time series distribution shift.

On more stable datasets such as ECL, Traffic, and Weather, the improvements from APT are more modest. Without other normalization strategies, APT brings 2%–10% gains to the backbone; when combined with normalization, the gains are typically limited to 1%–5%, and in some cases, APT may not yield further benefits. Given that models like CATS already achieve strong performance on these datasets—and that additional normalization often brings little improvement—APT performs reasonably within expectations. Moreover, APT is lightweight, adds negligible training overhead, and frequently enables state-of-the-art performance. Thus, incorporating APT can be considered non-intrusive and low-risk.

In summary, APT consistently enhances forecasting performance with minimal overhead. On challenging datasets such as ETT and Exchange, which exhibit significant distribution shift and data quality issues, APT achieves SOTA results with negligible risk of degradation. On more stable datasets like ECL, Traffic, and Weather, where strong backbones and normalization strategies already saturate performance, APT remains non-intrusive and yields moderate gains. Its lightweight design and broad compatibility make APT a reliable and effective enhancement to modern time series forecasting pipelines.

C.5 Hyper-parameter Sensitivity Study

In this section, we discuss the impact of hyperparameters on APT. Given that this section does not have a significant impact compared to ablation studies, we have placed it entirely in the appendix for the comprehensiveness of experiment.

Potential extra Parameter:

- + **Double prototype**: Double the number of prototypes assigned to each dataset in Table 5.
- + **Identity embedding**: Add identity embedding $\mathbf{ID} \in \mathbb{R}^{C \times D}$ to $\tilde{\mathbf{T}}_t$ in Equation 8, which is considered optional

in the main text, to explore whether different affine parameters can be assigned to different channels.

- + **“Day in Month”**: Add “Day in Month” to the timestamp representation in Equation 4. This label is considered an insignificant timestamp used to test the acceptance level of redundancy, as time series rarely generate associations on this timestamp.

The results of hyper-parameter sensitivity study are shown in Table 8. Prototype learning is originally introduced to address the few-shot challenge at each timestamp. Increasing the number of prototypes provides more fine-grained temporal representations but also raises the risk of overfitting. This hyper-parameter is dataset-dependent and a larger prototype library tends to improve performance on ETTh1.

Identity embedding aims to differentiate affine parameters across channels. However, we omit this component to maintain APT’s minimal complexity and avoid redundancy with channel-specific mechanisms already present in some backbones. Empirically, it only yields noticeable gains on Exchange and ETTh1 due to their limited channel counts, and affine parameters may struggle to capture complex spatial correlations.

Among timestamp embeddings, “Day in Month” proves less informative than “Day in Week” or “Time in Hour”. The latter two often reflect external factors influenced by diurnal cycles and human activity, whereas “Day in Month” introduces noise and redundancy, degrading APT’s global distributional awareness and consistently reducing performance across all backbones.

Embedding Size is the most important hyper-parameter for APT intuitively, as it determines the representation space of timestamps and prototypes. We show the impact of different embedding sizes on model performance in Table 10.

This parameter has minimal impact on APT’s performance in most settings, with sensitivity primarily observed on Informer and the Exchange dataset. Larger embedding

Module	Backbone				Normalization				APT
	CATS	Informer	iTransformer	SparseTSF	RevIN	Dish-TS	SAN	FAN	
96	1.401M			6.528M	0.081K		0.369M	85.294K	
192	1.401M		11.329M	6.577M	0.137K	0.014K	0.377M	97.678K	
336	1.401M			6.651M	0.221K	4.718k	0.389M	0.116M	
720	1.402M			6.848M	0.445K		0.422M	0.166M	2.54K

Table 9: Parameter count reported for the backbone and normalization on the ETTh1 dataset. $M = 1e^6$, $K = 1e^3$

Embedding Size	10	20*	30	50	
Metrics	MSE	MSE	MSE	MSE	
ECL	CATS	0.167	0.166	0.166	0.167
	Informer	0.344	0.331	0.360	0.327
	iTransformer	0.167	0.165	0.168	0.166
	SparseTSF	0.170	0.170	0.170	0.170
ETTh1	CATS	0.442	0.443	0.443	0.444
	Informer	1.178	1.195	1.197	1.168
	iTransformer	0.486	0.478	0.480	0.489
	SparseTSF	0.452	0.451	0.453	0.454
Exchange	CATS	0.286	0.285	0.325	0.275
	Informer	1.528	1.510	1.356	1.750
	iTransformer	0.452	0.444	0.460	0.519
	SparseTSF	0.321	0.316	0.340	0.348
Weather	CATS	0.247	0.245	0.244	0.242
	Informer	0.286	0.270	0.280	0.281
	iTransformer	0.253	0.252	0.255	0.253
	SparseTSF	0.266	0.266	0.268	0.268

Table 10: Results of embedding sizes on different datasets and backbones. $L = 336$, $H = 336$

sizes may lead to overfitting and degrade forecasting accuracy. In the main experiments, we recommend a fixed embedding size of 20 across all backbone–normalization combinations for simplicity and consistency.

C.6 Parameter Count Comparison

We report the parameter counts of each backbone and normalization on ETTh1 in Table 9. Among the backbones, SparseTSF is a well-known lightweight linear model. Its patch mechanism results in even fewer parameters than DLinear, with a total below 1K. It explains why APT brings limited improvements when paired with SparseTSF—its minimal capacity may be insufficient to fully utilize the dynamic affine transformations provided by APT.

In contrast, CATS, Informer, and iTransformer each exceed 1M parameters, which is over three orders of magnitude larger than APT. Notably, Informer’s parameter count is independent of input length due to channel dependency.

For normalization strategies, RevIN introduces affine parameters twice the number of input channels, yet as shown in Table 1, these have limited effect on forecasting performance. Dish-TS, despite underperforming, requires more than twice the parameters of APT, while advanced methods

like SAN and FAN are 10–100× larger.

Parameter count directly reflects that APT is a lightweight plugin. In addition to the performance improvements brought about by global distribution awareness, it is compatible with any backbone and normalization strategy, and often does not impose any computational burden.

C.7 Additional Ablation Results

Component Ablation: We evaluate APT by removing key components. For ease of understanding, we have copied the introduction from the main text below:

- **W/O Top- k :** Remove Top- k from Equation 6;
- **W/O Prototype:** Use raw timestamp embeddings without prototype matching;
- **W/O de-APT:** Remove inverse transformation of APT;
- **W/O γ or β :** Remove scaling or bias in affine transformation;;

Compared with the main text, more experimental results are shown in Figure 8. The main conclusions align with those presented in the main text. Removing the inverse APT transformation imposes an additional burden on the prediction head, which must learn to reconstruct the original value space from APT’s transformed representation. In contrast, de-APT preserves structural symmetry across the pipeline and alleviates this issue.

Top- k weighting, prototype learning, and affine parameters are all mechanisms related to managing distributional risk. Due to the limited priors and lack of manual intervention in deep learning, models are susceptible to risks such as feature oversmoothing, feature collapse, or diminished reliance on affine parameters when supervising learning. These factors are highly dataset- and model-dependent, often introducing randomness into the optimization process.

Top- k and prototype learning are conceptually opposite in representation: Top- k enforces discriminative assignments across timestamps, while prototype learning promotes generalization for underrepresented or highly variable timestamp features. In APT, these two components are balanced to mitigate prediction instability.

In some dataset–backbone combinations, removing the affine bias term β occasionally leads to performance gains. Nevertheless, we retain both affine parameters in the main experiments, as such fluctuations are irregular and require extensive manual tuning to validate, which does not align with APT’s pursuit of lightweight and flexibility.

Cross-Backbone \mathcal{M} , \mathcal{N} = None, Output length $H = 336$										
Source →	CATS		Informer		iTransformer		SparseTSF			
Target ↓	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE		
CATS	0.436	0.443	0.439	0.449	0.438	0.444	0.436	0.446		
Informer	0.817	1.094	0.797	1.195	0.805	1.134	0.804	1.158		
iTransformer	0.465	0.470	0.472	0.478	0.473	0.478	0.472	0.478		
SparseTSF	0.438	0.451	0.443	0.455	0.442	0.455	0.438	0.451		
Cross-Normalization \mathcal{N} , \mathcal{M} = iTransformer, Output length $H = 336$										
Source →	None		RevIN		Dish-TS		SAN		FAN	
Target ↓	MAE	MSE								
None	0.473	0.478	0.489	0.496	0.482	0.487	0.485	0.492	0.475	0.478
RevIN	0.454	0.468	0.448	0.457	0.452	0.460	0.459	0.476	0.447	0.458
Dish-TS	0.502	0.506	0.503	0.507	0.508	0.513	0.501	0.506	0.501	0.506
SAN	0.469	0.479	0.461	0.486	0.469	0.491	0.462	0.492	0.469	0.490
FAN	0.477	0.487	0.476	0.486	0.478	0.487	0.477	0.487	0.479	0.489
Cross-Output length $H = 336$, \mathcal{M} = iTransformer, \mathcal{N} = None										
Source →	96		192		336		720			
Target ↓	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE		
96	0.410	0.389	0.411	0.398	0.430	0.416	0.440	0.415		
192	0.442	0.438	0.441	0.436	0.447	0.445	0.469	0.482		
336	0.477	0.491	0.504	0.524	0.473	0.478	0.514	0.531		
720	0.546	0.558	0.568	0.571	0.531	0.538	0.528	0.526		

Table 11: Performance report of APT under one epoch of fine-tuning across Settings

C.8 Cross-Setting Fine-Tuning

Since APT’s parameters are independent of forecasting length, and its distribution shift mitigation is inherently dataset-driven, we further investigate the flexibility and generalization of APT under diverse settings. This experiment evaluates APT’s performance under **cross-backbone \mathcal{M}** , **cross-normalization \mathcal{N}** , and **cross-length H** conditions.

We conduct the study on the ETTh1 dataset by transferring pretrained APT parameters and applying one epoch of fine-tuning for each new setting. This setup follows observations from additional experiments, which indicate that zero-shot performance is limited without fine-tuning.

Cross-Backbone \mathcal{M} : We incorporate APT parameters learned from a source backbone into the target backbone’s forecasting pipeline and perform one epoch of fine-tuning. As shown in Table 11, this consistently improves performance across all settings, often surpassing the results of joint training with APT on the target backbone. These results demonstrate that APT possesses cross-backbone transferability and can directly enhance arbitrary forecasting backbone with pretrained parameters and minimal fine-tuning.

Cross-Normalization \mathcal{N} : For iTransformer, we further test transferring APT parameters across normalization

strategies. By injecting APT parameters learned under a source normalization into the target normalization pipeline and fine-tuning for one epoch, we again observe results that are comparable to the original configuration. While only a single epoch is used, performance can steadily improve with additional training, mirroring the behavior observed in the cross-backbone setting.

Cross-Length H : We also attempt to transfer APT across different output lengths, but find that this setup generally fails to reach optimal performance. Fine-tuning in these cases cannot recover the performance of jointly trained APT. We attribute this to distribution shift introduced by varying output lengths, which result in misaligned feature spaces that undermine the generalizability of a length-specific APT model.

In summary, when the output length is consistent, APT demonstrates strong generalization across backbones and normalization methods. This suggests that APT effectively captures global distributional features intrinsic to the dataset, independent of model architecture or normalization strategy. A single epoch of fine-tuning is sufficient to align module interactions and recover performance, highlighting the flexibility and generalization of APT across diverse forecasting configurations.

C.9 Visualization

t-SNE of Embedding: Figure 9-14 shows the timestamp and prototype embeddings learned by APT with iTransformer on all datasets. In general, timestamp representations tend to cluster around “Day in Week”, except on ETT1/2, where human behavioral patterns may introduce additional coupling. Prototypes, “Day in Week”, and “Time in Day” are inherently discrete. However, due to the soft nature of the orthogonality loss $Loss_{orth}$ and the influence of external factors across datasets, their representations still exhibit structured patterns in the t-SNE projected space.

Forecasting cases: Figure 6 and Figure 15-17 presents forecasting examples on the ETTh1, ETTh2 and Exchange datasets without normalization. In these dataset cases, backbone models often exhibit severe distributional shift. For example, Informer forecastings generally have a clear mean offset from the ground truth, while iTransformer outputs reveal systematic trend bias.

APT effectively mitigates these issues using only two-dimensional dynamic affine parameters per instance. As shown in Figure 6, APT amplifies periodic scales; In Figure 16, it reduces volatility of pulse-like time series; In Figures 15 and 17, it corrects deviations in both mean and trend.

These cases collectively demonstrate APT’s mechanism for handling distribution shift, which adjusts scale and mean to better align predicted outputs with real-world distributions, without altering the underlying temporal patterns. This enables seamless compatibility between APT and a wide range of backbones.

Temporal distribution: As illustrated in Figure 18 and 19, we visualize the mean, variance, and their ratio of the raw time series, RevIN-normalized sequences, and APT-transformed outputs in a 3D distribution space. All datasets exhibit varying degrees of distributional shift. The raw data differ significantly in distribution across datasets, with ETT and Exchange showing the most severe shift, followed by Traffic and Weather, while ECL remains relatively stable, which is consistent with our earlier dataset analysis.

RevIN maps all sequences into a standardized space with zero mean and unit variance. While this prevents the backbone from being overwhelmed by distributional complexity, it also eliminates the possibility of leveraging meaningful distributional cues. In contrast, APT strikes a balance between smoothing and diversity through learnable affine transformations. The transformed sequences form clusters in distributional space—not overly uniform, but structured and distinct—allowing the backbone to perceive informative distributional patterns.

These 3D visualizations consistently validate that APT operates on distributional alignment: by introducing global distributional awareness and controlled feature clustering, it can effectively mitigate distribution shift across a wide range of scenarios.

Affine Transformation: Since our affine parameters are dynamically generated from timestamp information, we can visualize them corresponding to the entire “Time in Day”

and “Day in Week” timestamps on the test set. Figure 20-25 presents the learned weekly affine parameters from APT on all datasets using iTransformer as the backbone. Most datasets are sampled hourly, except for Exchange (daily) and Weather (every 10 minutes). Since these parameters are derived from deep networks, their distributions are inherently difficult to interpret in a fully human-interpretable manner.

Nonetheless, we observe that most datasets exhibit distinct affine patterns across different “Day in Week”, which is consistent with the previous t-SNE visualization of embeddings. Importantly, the clustered affine parameters remain bounded, which is critical for convergence in regression tasks. This constraint is guided by our Affine Regularization Loss $Loss_R$, which softly encourages the parameters to maintain zero mean and unit variance. However, due to the soft nature of self-supervision, the loss primarily enforces boundedness rather than strict normalization.

Interestingly, datasets like ECL, Exchange, and Traffic exhibit axis-symmetric patterns, which we speculate result from the scale parameter γ overfitting to redundant distributional signals, with the bias β compensating to restore balance.

D Discussion

D.1 Pattern, Distribution & Distribution Shift

We aim to establish a shared understanding of the core components of time series data—pattern and distribution. Patterns refer to the structural dynamics of the series, such as combinations of trend, seasonality, and residuals. Forecasting backbones are primarily designed to capture these patterns, as they reflect both short- and long-term dependencies critical to accurate prediction.

Distributions, on the other hand, describe the state space of the series. A classic perspective views distributions through statistical properties such as mean and variance, often under the assumption of Gaussianity. Since distributions are shaped by external conditions, they tend to drift over time, posing a fundamental challenge. When distributional shift are entangled with pattern dynamics, the value ranges and variability observed by the backbone vary significantly. This forces the model to allocate capacity toward fitting distributional noise, often at the cost of learning meaningful patterns.

One of the dominant strategies to address distribution shift is normalization. By mapping sequences into a common statistical space (e.g., zero mean and unit variance), normalization enables the backbone to focus solely on pattern learning. A subsequent denormalization step restores the output to its original scale. This design simplifies the learning problem and enhances performance. Recent works such as Dish-TS, SAN, and FAN extend the simple RevIN framework by focusing on the learnable components of distributions like modeling asymmetric statistics between history and future, or going beyond mean-variance assumptions.

D.2 Local statistical property and global temporal semantics

While statistical normalization with local mean and variance is widely adopted, it has inherent limitations. First, time series often contain noise and missing values, making such statistics unreliable. Second, these statistics are computed over limited contexts: RevIN relies on full-length history, while SAN uses patch-level features, yet neither guarantees alignment with the actual distribution of future samples. Furthermore, relying solely on Gaussian assumptions (mean & variance) may fail to capture more comprehensive distribution features.

To overcome these limitations, we argue for leveraging global, non-statistical information. APT adopts timestamps as external priors to mitigate local errors. Timestamps are easy to obtain and strongly correlated with real-world temporal semantics such as traffic spikes during rush hours, high TV ratings at night, or increased foot traffic in shopping malls on weekends. As a result, they implicitly encode global information that statistical normalization alone cannot capture.

D.3 Reasons for affine transformation

APT utilizes a network to apply dynamic affine parameters to time series with different timestamp combinations. Affine transformations have long been integrated with normalization. For instance, in Transformers to align intermediate representations between layers or blocks. More significantly, AdaIN in style transfer showed that affine parameters can serve as carriers of cross-modal information, which demonstrates that such parameters can effectively encode external priors to assist a primary task.

Inspired by this, APT uses affine modulation to inject timestamp-driven global distributional awareness into the forecasting pipeline. Unlike prior works such as GLAFF or Informer’s temporal embedding, which encode timestamp features as full-sequence tensors with the same shape as the time series and use them either jointly or as replacements for the sequence itself, APT follows a minimal design. For example, GLAFF introduces more parameters than some backbones like CATS, and suffers from the uniqueness of timestamp combinations, which is difficult to extract generalizable patterns, similar to few-shot learning challenges.

APT addresses this by producing only two parameters per instance and applying them to modulate the sequence through affine transformation. This lightweight design avoids redundancy, minimizes overhead, and, with the help of prototype learning, ensures robust and scalable global timestamp representation.

D.4 Theoretical support for APT

Generally speaking, the work related to conditional affine parameters in computer vision and natural language processing does not require theoretical explanations because the ideas are very simple. Introducing external variables such as timestamps is equivalent to introducing conditional information into the modeling of time series. According to the principles of Bayesian inference and information gain,

conditioning on informative variables reduces uncertainty, thereby improving generalization and forecasting accuracy.

While the MSE loss used in point forecasting does not strictly follow a Bayesian formulation, we provide the following conceptual argument to clarify APT’s benefit:

In time series forecasting, the goal is to estimate the conditional probability of the future series Y given the historical sequence X , denoted as $P(Y | X)$. By treating timestamp ts as an informative variable, we shift to estimating:

$$P(Y | X, ts) = \frac{P(ts | Y, X), P(Y | X)}{P(ts | X)}$$

This conditioning provides additional benefit to forecasting. According to the conditional entropy relation:

$$\mathcal{H}(Y | X, ts) \leq \mathcal{H}(Y | X),$$

the inclusion of ts always reduces or maintains the uncertainty in Y , leading to improved forecasting performance. We believe that this conditional information can actually be replaced by any task-related information, such as text or images, which is consistent with the broader idea of multimodal and multivariate time series forecasting.

E Limitation and Future Work

Limitation: APT relies on manually chosen timestamps such as “Time in Day” or “Day in Week” with distributional similarity as shown in Figure 2 to obtain information gain. However, in the appendix experiments, irrelevant labels “Day in Month” not only fail to provide performance gains but may even degrade forecasting accuracy.

APT requires pretraining with additional losses. Moreover, since APT relies on self-supervised clustering and requires parameter alignment with the forecasting task, improper hyperparameter configurations may hinder its ability to enhance the performance of the backbone model, posing challenges for practical deployment.

Future Work: APT introduces a new paradigm in time series forecasting by addressing the limitations of local statistical normalization and mitigating distribution shift. Future directions include integrating online distribution shift detection, incorporating richer global temporal semantics, and exploring more expressive parametric transformations.

Moreover, we hope APT inspires multimodal research in time series. Timestamps are sparse yet semantically rich signals, often requiring large models like GLAFF for effective representation before our work. However, each time series sample typically receives only limited external context. Unlike vision or language data, 1-D time series usually cannot accommodate unique but semantically rich external information, often resulting in overfitting of multimodal tasks of time series. APT leverages dynamic affine transformations to compress such redundant, noisy inputs into compact parametric forms, which serves as an information attenuator that aligns external modalities with the limited expressiveness of time series. We consider that this technology is expected to help better integrate auxiliary signals and enhances performance in cross-modal forecasting tasks.

Methods		CATS				Informer				iTTransformer				SparseTSF					
Affine				+APT				+APT				+APT				+APT			
Metrics		MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE		
ETTh1	Nones	96	0.235	0.140	0.234	0.138	0.383	0.308	0.374	0.281	0.236	0.138	0.240	0.139	0.245	0.150	0.245	0.150	
		192	0.257	0.161	0.249	0.154	0.425	0.361	0.406	0.330	0.262	0.160	0.253	0.153	0.258	0.162	0.254	0.159	
		336	0.265	0.168	0.264	0.166	0.446	0.398	0.413	0.331	0.280	0.174	0.268	0.165	0.269	0.171	0.268	0.170	
		720	0.298	0.203	0.302	0.205	0.449	0.411	0.438	0.384	0.302	0.205	0.301	0.203	0.303	0.207	0.301	0.206	
	ECL	RevIN	96	0.232	0.137	0.233	0.138	0.308	0.211	0.289	0.189	0.230	0.135	0.233	0.137	0.244	0.150	0.243	0.149
		192	0.250	0.154	0.249	0.154	0.318	0.222	0.314	0.217	0.252	0.160	0.250	0.154	0.252	0.159	0.251	0.158	
		336	0.260	0.165	0.260	0.165	0.307	0.209	0.306	0.206	0.264	0.169	0.261	0.164	0.266	0.172	0.265	0.171	
		720	0.291	0.203	0.295	0.207	0.416	0.369	0.393	0.322	0.289	0.194	0.292	0.194	0.297	0.212	0.297	0.212	
ETTh2	Dish-ts	96	0.246	0.145	0.242	0.144	0.348	0.258	0.336	0.249	0.236	0.136	0.236	0.136	0.243	0.146	0.242	0.145	
		192	0.259	0.160	0.259	0.160	0.388	0.309	0.335	0.246	0.262	0.163	0.253	0.153	0.254	0.159	0.254	0.159	
		336	0.276	0.175	0.273	0.172	0.338	0.249	0.379	0.296	0.270	0.168	0.262	0.160	0.271	0.174	0.269	0.172	
		720	0.307	0.211	0.307	0.212	0.440	0.403	0.434	0.395	0.294	0.199	0.294	0.194	0.303	0.207	0.303	0.209	
	SAN	96	0.243	0.141	0.241	0.140	0.290	0.186	0.264	0.160	0.233	0.134	0.236	0.136	0.240	0.139	0.239	0.139	
		192	0.266	0.165	0.267	0.166	0.301	0.199	0.296	0.196	0.248	0.148	0.250	0.151	0.255	0.154	0.253	0.153	
		336	0.281	0.179	0.280	0.178	0.352	0.259	0.313	0.208	0.262	0.161	0.264	0.161	0.270	0.167	0.269	0.166	
		720	0.316	0.217	0.315	0.216	0.454	0.420	0.422	0.362	0.291	0.192	0.296	0.193	0.319	0.226	0.305	0.205	
ETTh2	FAN	96	0.238	0.140	0.234	0.137	0.243	0.144	0.237	0.141	0.248	0.147	0.240	0.141	0.236	0.138	0.233	0.136	
		192	0.251	0.154	0.249	0.152	0.251	0.152	0.252	0.153	0.259	0.159	0.254	0.156	0.250	0.153	0.248	0.152	
		336	0.267	0.168	0.267	0.167	0.266	0.163	0.271	0.171	0.278	0.175	0.271	0.170	0.266	0.167	0.265	0.165	
		720	0.305	0.206	0.300	0.205	0.303	0.200	0.296	0.199	0.311	0.209	0.309	0.210	0.301	0.204	0.300	0.203	
	Nones	96	0.416	0.404	0.400	0.382	0.865	1.119	0.669	0.896	0.450	0.430	0.410	0.389	0.392	0.374	0.390	0.374	
		192	0.451	0.460	0.422	0.420	0.898	1.263	0.736	1.018	0.479	0.481	0.441	0.436	0.424	0.420	0.422	0.420	
		336	0.468	0.490	0.436	0.443	0.819	1.153	0.797	1.195	0.497	0.512	0.473	0.478	0.446	0.457	0.438	0.451	
		720	0.541	0.591	0.484	0.481	0.982	1.487	0.930	1.427	0.568	0.591	0.528	0.526	0.496	0.493	0.497	0.497	
ETTh2	RevIN	96	0.411	0.396	0.400	0.382	0.567	0.640	0.441	0.442	0.420	0.410	0.410	0.399	0.391	0.374	0.390	0.374	
		192	0.432	0.437	0.421	0.420	0.592	0.702	0.587	0.694	0.463	0.480	0.430	0.432	0.414	0.414	0.415	0.415	
		336	0.447	0.456	0.434	0.439	0.582	0.681	0.579	0.679	0.462	0.471	0.448	0.457	0.440	0.445	0.439	0.444	
		720	0.483	0.503	0.469	0.469	0.649	0.838	0.640	0.812	0.546	0.581	0.477	0.475	0.471	0.473	0.463	0.462	
	Dish-ts	96	0.419	0.411	0.411	0.398	0.755	0.962	0.575	0.626	0.446	0.433	0.444	0.436	0.415	0.396	0.413	0.402	
		192	0.469	0.485	0.435	0.437	0.746	0.965	0.747	0.997	0.466	0.467	0.454	0.454	0.442	0.440	0.439	0.443	
		336	0.474	0.486	0.458	0.470	0.878	1.157	0.797	1.125	0.512	0.554	0.508	0.513	0.454	0.465	0.466	0.484	
		720	0.536	0.558	0.505	0.517	0.811	1.164	0.765	1.041	0.567	0.596	0.561	0.577	0.556	0.609	0.504	0.516	
ETTh2	SAN	96	0.414	0.411	0.414	0.414	0.506	0.564	0.481	0.535	0.422	0.415	0.419	0.420	0.498	0.566	0.426	0.434	
		192	0.435	0.451	0.434	0.450	0.507	0.569	0.491	0.536	0.447	0.457	0.447	0.467	0.491	0.563	0.449	0.472	
		336	0.446	0.467	0.446	0.466	0.545	0.625	0.539	0.640	0.460	0.472	0.462	0.492	0.499	0.562	0.458	0.488	
		720	0.510	0.560	0.493	0.523	0.611	0.697	0.602	0.686	0.535	0.564	0.486	0.501	0.528	0.564	0.525	0.560	
	FAN	96	0.422	0.407	0.418	0.401	0.451	0.432	0.439	0.432	0.424	0.406	0.420	0.404	0.426	0.408	0.433	0.414	
		192	0.453	0.453	0.450	0.451	0.494	0.499	0.482	0.497	0.460	0.461	0.452	0.451	0.472	0.468	0.460	0.464	
		336	0.483	0.496	0.476	0.487	0.525	0.550	0.524	0.549	0.492	0.511	0.479	0.489	0.474	0.489	0.478	0.500	
		720	0.554	0.571	0.549	0.574	0.607	0.652	0.577	0.638	0.574	0.611	0.556	0.576	0.567	0.623	0.549	0.573	
ETTh2	Nones	96	0.381	0.331	0.365	0.315	1.569	3.349	0.869	1.251	0.602	0.685	0.379	0.327	0.394	0.349	0.398	0.353	
		192	0.462	0.446	0.427	0.413	1.322	2.577	1.168	2.340	0.684	0.859	0.473	0.466	0.451	0.438	0.451	0.437	
		336	0.501	0.507	0.484	0.477	1.324	2.595	1.281	2.540	0.641	0.764	0.519	0.540	0.500	0.515	0.490	0.497	
		720	0.671	0.850	0.607	0.739	1.518	3.013	1.456	2.945	0.816	1.146	0.602	0.697	0.622	0.760	0.598	0.705	
	RevIN	96	0.368	0.316	0.354	0.306	0.516	0.534	0.476	0.484	0.382	0.334	0.364	0.311	0.370	0.323	0.370	0.321	
		192	0.417	0.384	0.405	0.383	0.556	0.650	0.524	0.574	0.440	0.431	0.413	0.383	0.413	0.385	0.413	0.385	
		336	0.443	0.415	0.435	0.417	0.540	0.604	0.487	0.492	0.451	0.440	0.443	0.424	0.438	0.413	0.443	0.411	
		720	0.472	0.															

Methods		CATS				Informer				iTransformer				SparseTSF				
Affine				+APT				+APT				+APT				+APT		
Metrics		MAE	MSE															
Exchange	Nones	96	0.335	0.233	0.246	0.105	0.896	1.255	0.879	1.215	0.443	0.374	0.283	0.152	0.265	0.124	0.256	0.114
	192	0.435	0.371	0.319	0.181	1.016	1.719	0.961	1.364	0.535	0.493	0.462	0.351	0.327	0.184	0.324	0.186	
	336	0.514	0.466	0.410	0.285	1.058	1.855	0.976	1.510	0.661	0.704	0.502	0.444	0.448	0.341	0.432	0.316	
	720	1.184	2.483	0.686	0.695	1.015	1.732	0.863	1.258	0.981	1.560	0.956	1.450	0.664	0.757	0.662	0.754	
	RevIN	96	0.213	0.090	0.199	0.079	0.456	0.354	0.344	0.196	0.235	0.103	0.213	0.089	0.235	0.105	0.230	0.101
	192	0.299	0.166	0.299	0.170	0.500	0.402	0.480	0.386	0.350	0.221	0.328	0.197	0.344	0.221	0.330	0.206	
	336	0.437	0.352	0.423	0.331	0.677	0.708	0.686	0.711	0.493	0.423	0.459	0.378	0.499	0.434	0.460	0.393	
	720	0.789	0.997	0.774	0.943	0.734	0.885	0.565	0.546	0.852	1.124	0.787	1.001	0.844	1.155	0.732	0.846	
	Dish-ts	96	0.270	0.114	0.256	0.105	0.425	0.308	0.423	0.307	0.293	0.155	0.263	0.111	0.301	0.155	0.284	0.128
	192	0.512	0.428	0.353	0.211	0.646	0.735	0.648	0.726	0.386	0.265	0.362	0.234	0.417	0.325	0.420	0.298	
Traffic	SAN	96	0.242	0.112	0.202	0.078	0.254	0.119	0.212	0.081	0.245	0.118	0.204	0.080	0.219	0.087	0.212	0.083
	192	0.338	0.212	0.296	0.163	0.396	0.300	0.298	0.167	0.396	0.307	0.298	0.167	0.312	0.169	0.302	0.171	
	336	0.500	0.474	0.414	0.314	0.484	0.394	0.420	0.321	0.515	0.473	0.471	0.396	0.419	0.305	0.422	0.324	
	720	0.853	1.287	0.748	0.935	0.696	0.827	0.685	0.782	0.827	1.098	0.857	1.149	0.720	0.839	0.694	0.844	
	FAN	96	0.301	0.152	0.261	0.130	0.330	0.194	0.308	0.183	0.278	0.133	0.241	0.104	0.247	0.109	0.247	0.109
	192	0.381	0.248	0.371	0.238	0.487	0.420	0.462	0.372	0.412	0.285	0.394	0.264	0.372	0.251	0.349	0.213	
	336	0.518	0.452	0.531	0.463	0.631	0.683	0.626	0.686	0.594	0.574	0.548	0.494	0.564	0.553	0.540	0.493	
	720	0.764	0.980	0.765	0.997	0.801	1.068	0.738	0.873	0.768	1.018	0.759	0.931	0.710	0.859	0.713	0.861	
	Nones	96	0.269	0.509	0.271	0.503	0.432	0.832	0.397	0.764	0.550	0.876	0.429	0.624	0.287	0.423	0.287	0.423
	192	0.284	0.549	0.281	0.510	0.392	0.772	0.386	0.759	0.560	0.909	0.416	0.617	0.292	0.435	0.292	0.435	
Weather	RevIN	96	0.263	0.382	0.264	0.386	0.395	0.735	0.360	0.689	0.270	0.375	0.276	0.380	0.285	0.424	0.285	0.424
	192	0.283	0.418	0.274	0.407	0.489	0.941	0.374	0.713	0.281	0.403	0.287	0.401	0.290	0.437	0.290	0.436	
	336	0.287	0.431	0.277	0.414	0.430	0.820	0.427	0.814	0.291	0.420	0.294	0.422	0.296	0.448	0.295	0.449	
	720	0.305	0.454	0.306	0.458	0.513	1.000	0.416	0.794	0.311	0.449	0.315	0.456	0.311	0.470	0.310	0.470	
	Dish-ts	96	0.279	0.411	0.263	0.387	0.385	0.710	0.357	0.683	0.288	0.402	0.293	0.403	0.302	0.440	0.299	0.436
	192	0.281	0.421	0.272	0.408	0.388	0.732	0.373	0.702	0.295	0.413	0.299	0.414	0.308	0.454	0.304	0.448	
	336	0.300	0.450	0.280	0.424	0.472	0.925	0.436	0.835	0.308	0.434	0.313	0.440	0.314	0.462	0.310	0.459	
	720	0.312	0.472	0.297	0.450	0.536	1.077	0.447	0.825	0.330	0.467	0.332	0.465	0.331	0.488	0.329	0.484	
	SAN	96	0.275	0.395	0.281	0.405	0.355	0.623	0.339	0.608	0.278	0.389	0.281	0.390	0.420	0.661	0.292	0.419
	192	0.286	0.421	0.290	0.427	0.403	0.715	0.377	0.673	0.288	0.416	0.291	0.418	0.361	0.557	0.346	0.535	
Weather	FAN	96	0.290	0.414	0.288	0.412	0.290	0.418	0.281	0.413	0.313	0.429	0.304	0.422	0.283	0.395	0.277	0.393
	192	0.306	0.439	0.291	0.421	0.304	0.440	0.292	0.432	0.334	0.453	0.310	0.435	0.293	0.419	0.291	0.417	
	336	0.315	0.455	0.297	0.435	0.317	0.459	0.302	0.448	0.340	0.468	0.324	0.455	0.300	0.433	0.302	0.437	
	720	0.352	0.507	0.330	0.484	0.350	0.510	0.329	0.486	0.375	0.520	0.360	0.504	0.330	0.481	0.324	0.478	
	Nones	96	0.203	0.147	0.195	0.145	0.243	0.184	0.218	0.164	0.224	0.163	0.220	0.163	0.258	0.192	0.253	0.190
	192	0.251	0.195	0.244	0.188	0.342	0.327	0.312	0.255	0.310	0.267	0.266	0.209	0.282	0.227	0.280	0.226	
	336	0.294	0.244	0.287	0.245	0.519	0.576	0.315	0.270	0.314	0.261	0.293	0.252	0.320	0.270	0.313	0.266	
	720	0.374	0.329	0.348	0.315	0.482	0.516	0.401	0.381	0.361	0.333	0.360	0.330	0.367	0.329	0.348	0.359	
	RevIN	96	0.193	0.145	0.192	0.144	0.214	0.175	0.213	0.167	0.206	0.158	0.208	0.158	0.235	0.185	0.235	0.185
	192	0.233	0.184	0.234	0.189	0.283	0.258	0.257	0.214	0.245	0.200	0.248	0.201	0.265	0.224	0.263	0.223	
	336	0.274	0.236	0.273	0.238	0.311	0.307	0.290	0.263	0.284	0.251	0.284	0.250	0.294	0.267	0.293	0.267	
	720	0.331	0.316	0.336	0.325	0.366	0.375	0.335	0.329	0.333	0.323	0.335	0.323	0.339	0.333	0.338	0.333	
	Dish-ts	96	0.216	0.151	0.212	0.150	0.231	0.188	0.228	0.187	0.238	0.190	0.220	0.162	0.234	0.168	0.239	0.169
	192	0.255	0.191	0.250	0.191	0.310	0.277	0.285	0.266	0.255	0.199	0.261	0.206	0.281	0.210	0.274	0.210	
	336	0.290	0.240	0.291	0.242	0.334	0.323	0.300	0.272	0.309	0.264	0.300	0.256	0.320	0.257	0.315	0.257	
	720	0.349	0.317	0.352	0.321	0.388	0.394	0.369	0.381	0.382	0.378	0.358	0.336	0.370	0.325	0.366	0.325	
	SAN	96	0.209	0.148	0.215	0.151	0.228	0.181	0.210	0.161	0.213	0.154	0.211	0.153	0.213	0.150	0.213	0.151
	192	0.258	0.195	0.258	0.196	0.253	0.214	0.257	0.217	0.253	0.198	0.254	0.195	0.258	0.195	0.256	0.194</b	

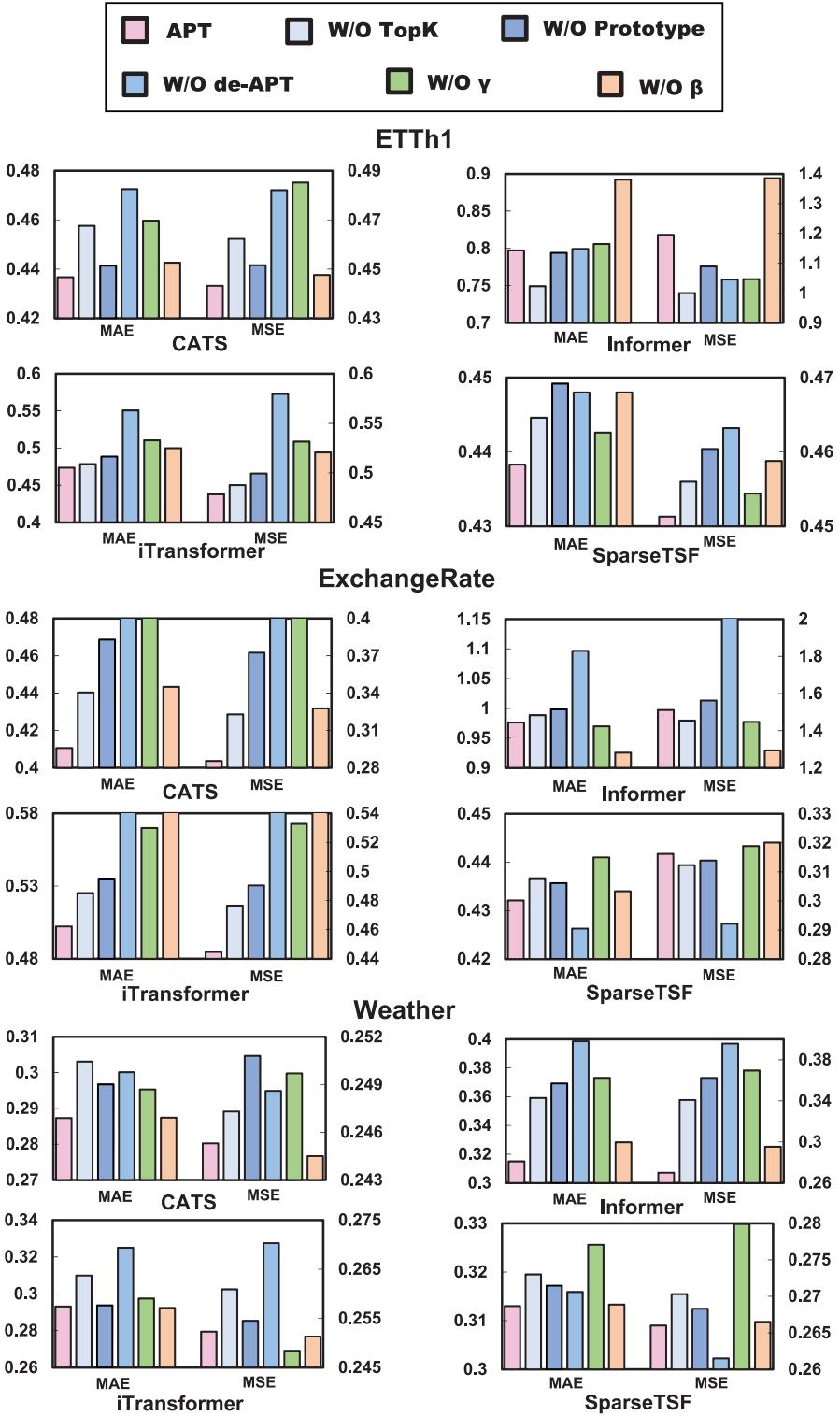


Figure 8: The ablation study results of APT components in ETTh1, ExchangeRate and Weather, $L = 336, H = 336$

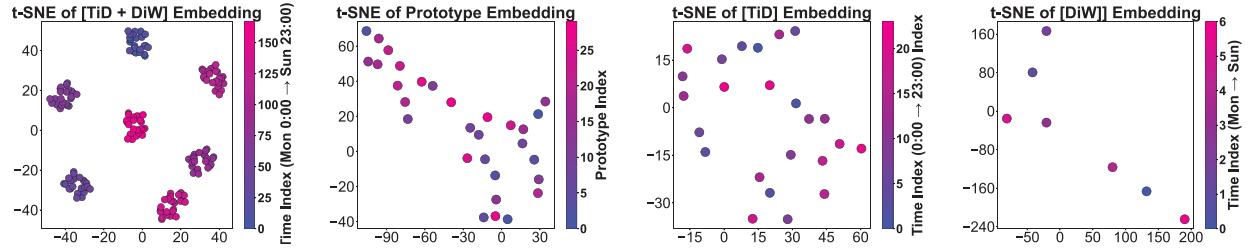


Figure 9: Visualization of APT’s embeddings on ECL dataset and iTransformer

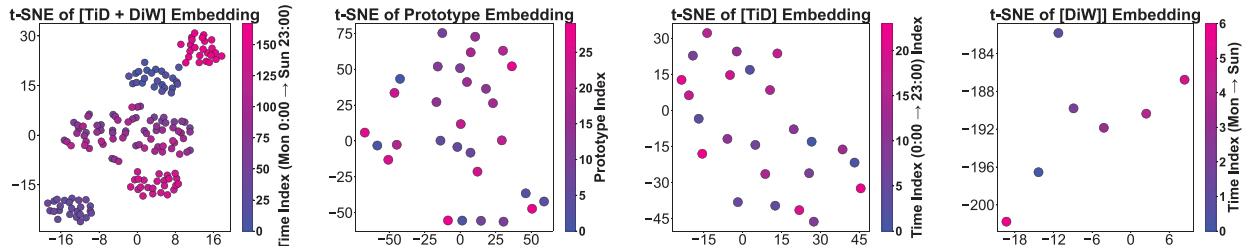


Figure 10: Visualization of APT’s embeddings on ETTh1 dataset and iTransformer

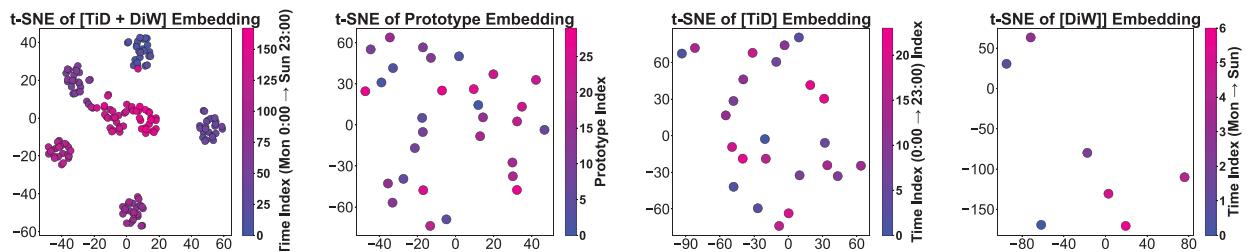


Figure 11: Visualization of APT’s embeddings on ETTh2 dataset and iTransformer

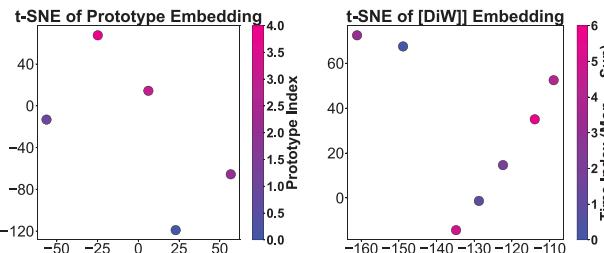


Figure 12: Visualization of APT’s embeddings on Exchange dataset and iTransformer

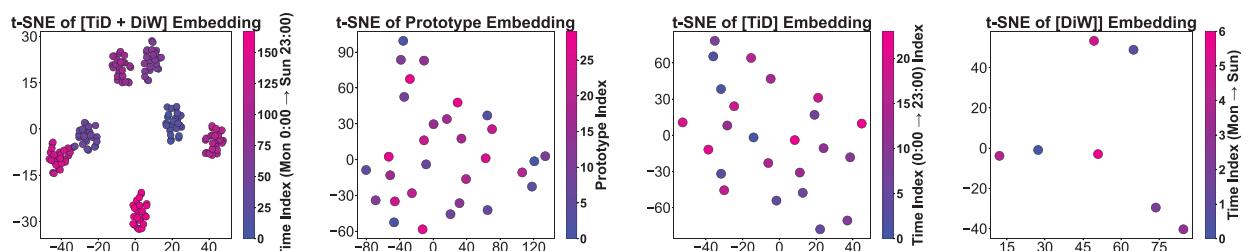


Figure 13: Visualization of APT’s embeddings on Traffic dataset and iTransformer

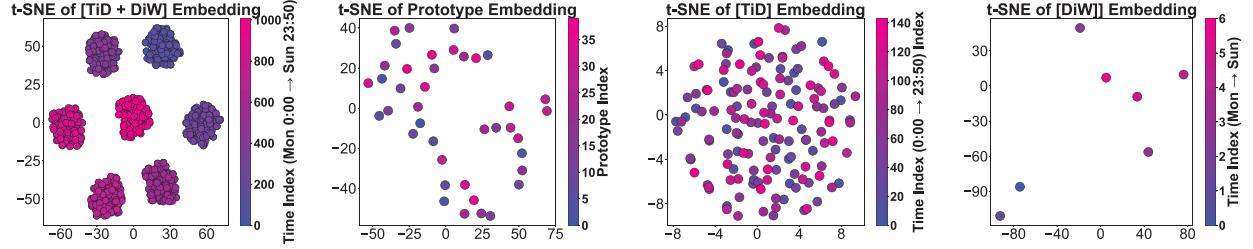


Figure 14: Visualization of APT’s embeddings on Weather dataset and iTransformer

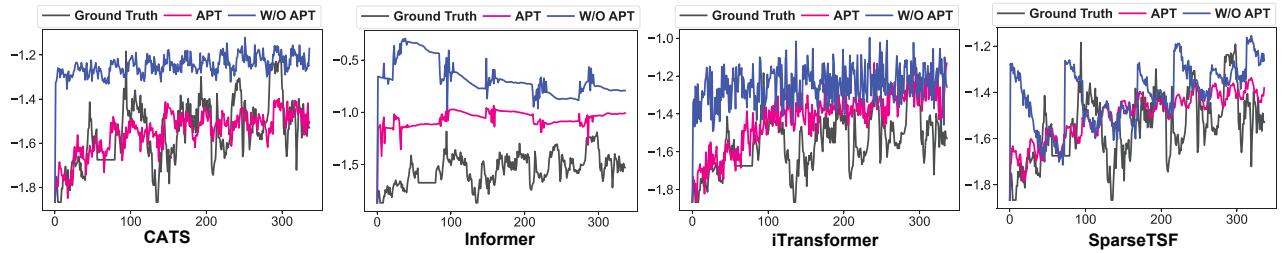


Figure 15: Visualization of forecasting results for different models on the ETTh1 dataset without normalization strategy

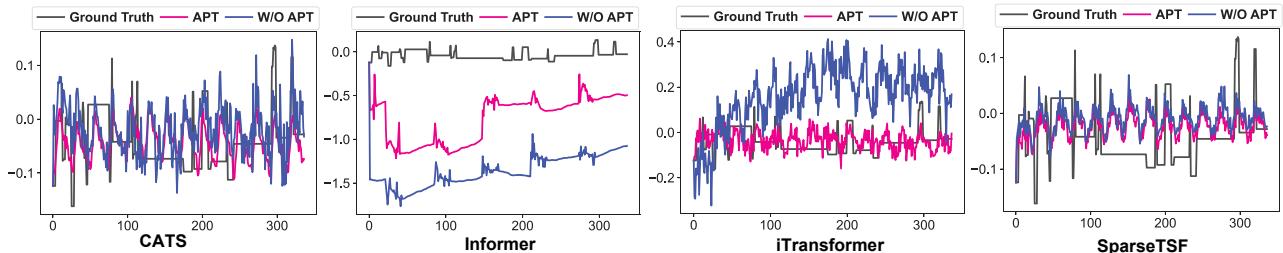


Figure 16: Visualization of forecasting results for another channel on the ETTh2 dataset without normalization strategy for different models

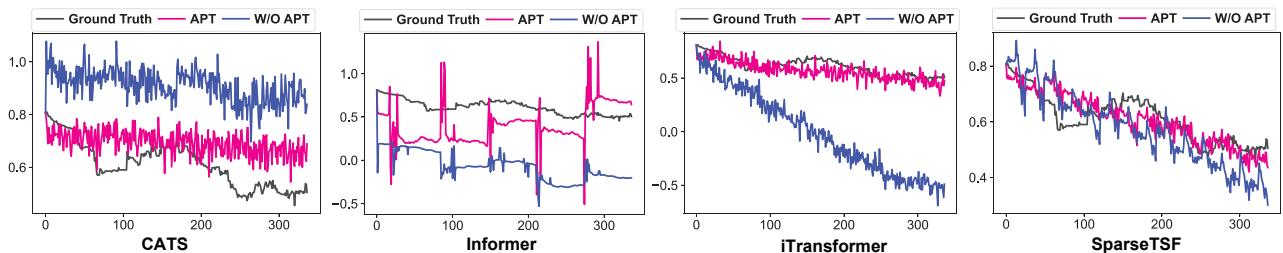


Figure 17: Visualization of forecasting results for different models on the ExchangeRate dataset without normalization strategy

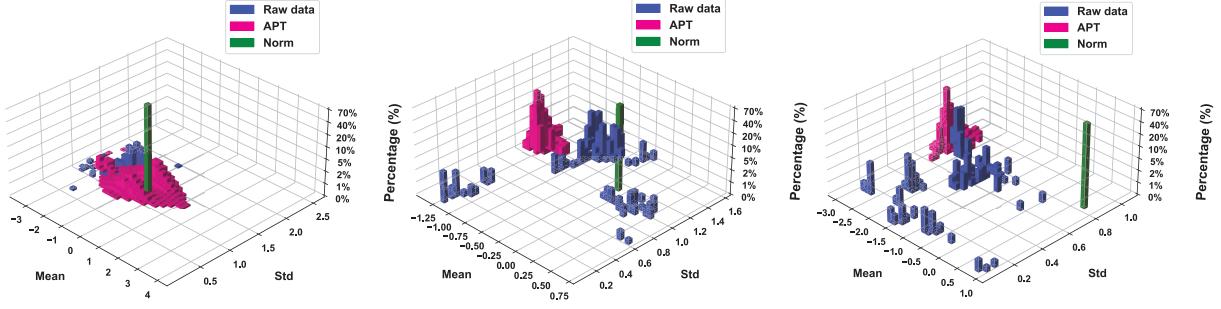


Figure 18: 3D visualization of temporal distribution and ratio of forecasting pipeline at different stages on ELC, ETTh1 & ETTh2.

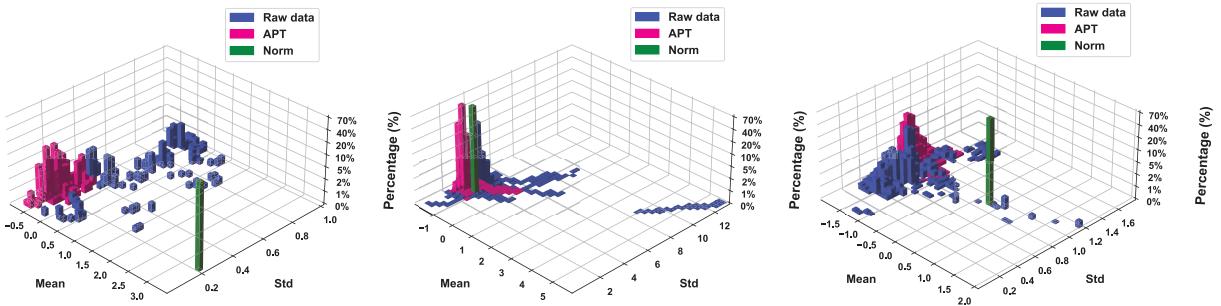


Figure 19: 3D visualization of temporal distribution and ratio of forecasting pipeline at different stages on Exchange, Traffic & Weather.

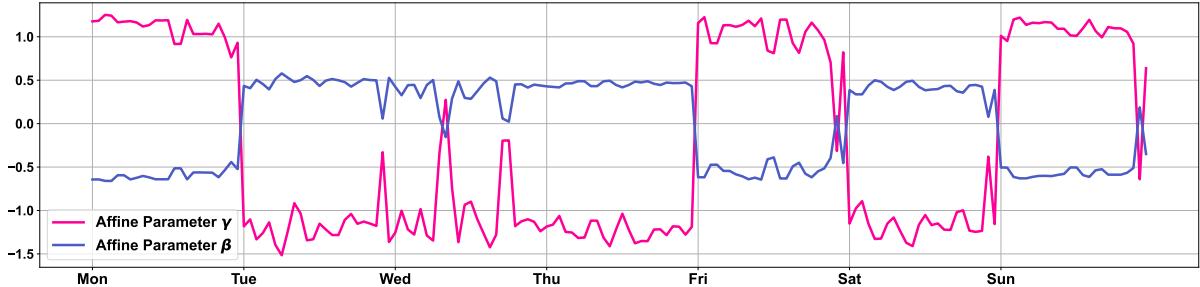


Figure 20: Visualization of affine parameters at different timestamps over a week on the ELC datasets

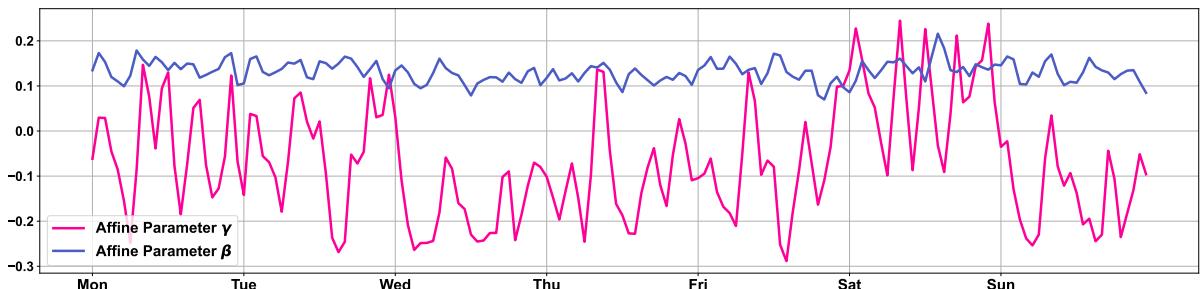


Figure 21: Visualization of affine parameters at different timestamps over a week on the ETTh1 datasets

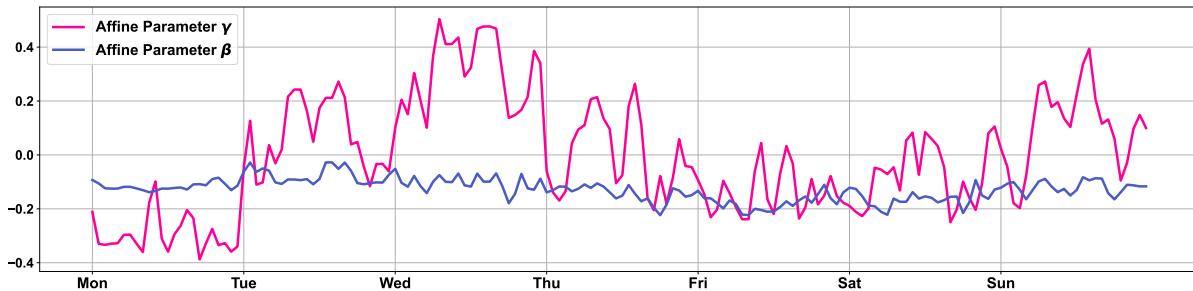


Figure 22: Visualization of affine parameters at different timestamps over a week on the ETTh2 datasets

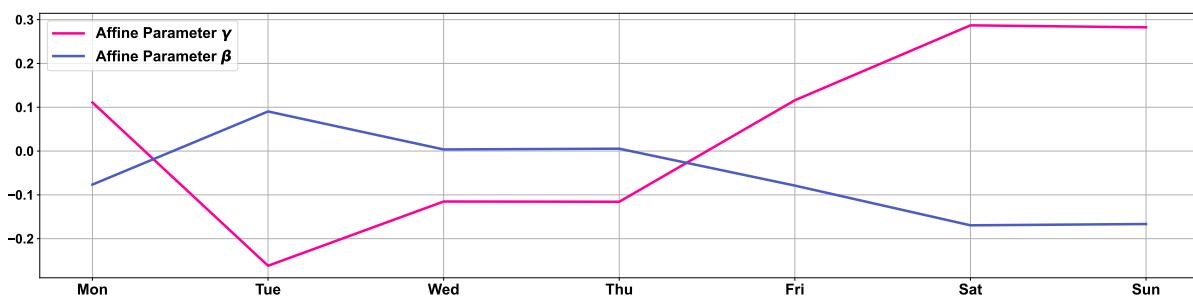


Figure 23: Visualization of affine parameters at different timestamps over a week on the Exchange datasets

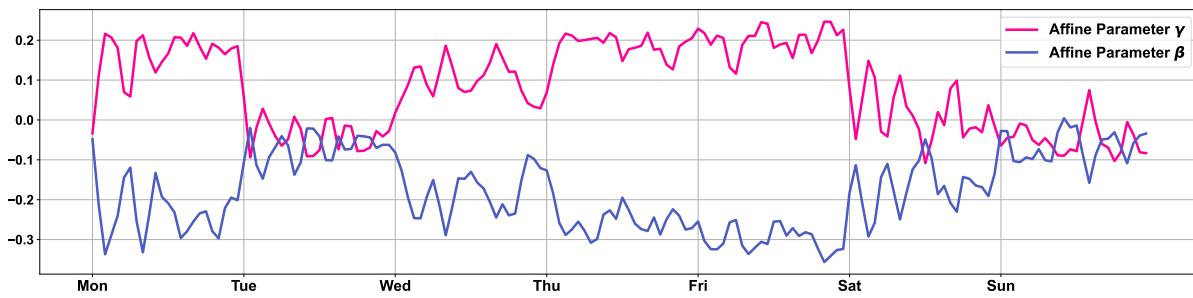


Figure 24: Visualization of affine parameters at different timestamps over a week on the Traffic datasets

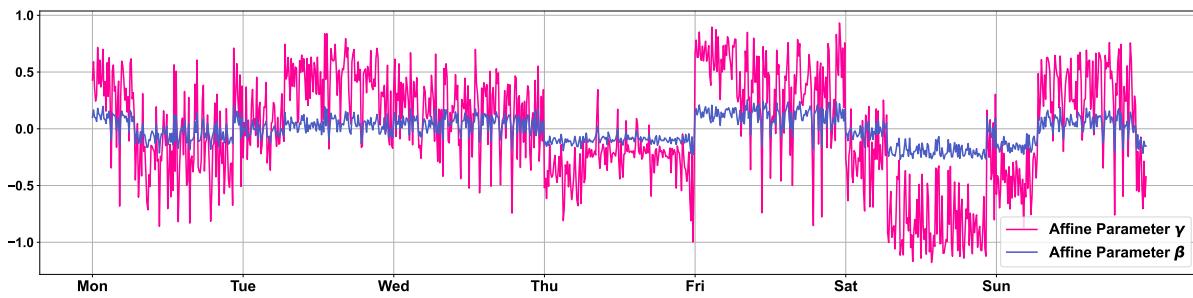


Figure 25: Visualization of affine parameters at different timestamps over a week on the Weather datasets