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*]

[1]State Key Laboratory of AI Safety, Institute of Computing Technology,Chinese Academy of Sciences
[2]University 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

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. 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.

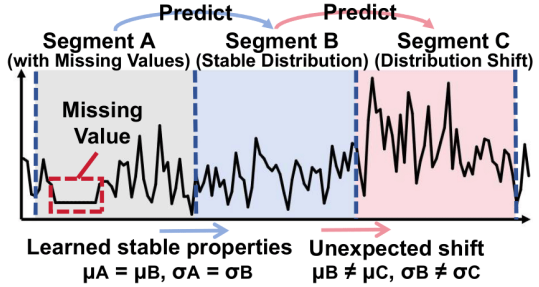

Figure 1: Local statistical normalization fails to handle distribution shift on ECL. The model retains outdated statistics when predicting from flawed Segment A to B but faces unseen shifts in Segment B and C. Shifts across all three segments exceed prior inter and intra shift issues, posing a broader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

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 **A**ffine **P**rototype-**T**imestamps (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.

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	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**	**0.264**	0.170	0.266	0.173	**0.265**	**0.171**	0.000	↓ 0.001
Weather	**0.274**	**0.236**	**0.274**	0.241	0.311	0.307	**0.310**	**0.306**	0.284	0.251	**0.282**	**0.248**	**0.294**	**0.267**	0.297	0.269	0.000	↓ 0.001
ETTh1	0.447	0.456	**0.441**	**0.451**	**0.582**	0.681	0.584	**0.679**	**0.462**	**0.471**	0.464	0.475	0.440	0.445	**0.430**	**0.437**	↑ 0.003	↑ 0.003
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 ↑ means worse performance.

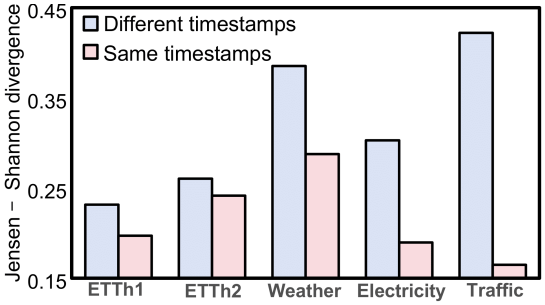

Figure 2: JS divergence of subseries with the same timestamp label in benchmark datasets

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 plug-in 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.

Related Work

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. 2025; 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.

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.

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.

Preliminaries

Time Series Forecasting

Given the historical multivariate time series $\{x^{(i)}_{t-L:t}\}^C_{i=1} = \{x^{(1)}_{t-L:t}, ..., x^{(C)}_{t-L:t}\} \in \mathbb{R}^{L\times C}$, 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 $\{y^{(i)}_{t+1:t+H}\}^C_{i=1} = \{y^{(1)}_{t+1:y+H}, ..., y^{(C)}_{t+1:y+H}\} \in \mathbb{R}^{H\times C}$ by leveraging the forecasting model $\boldsymbol{\mathcal{M}}$ to learn patterns from historical data, where H is the future horizon:

$$\{y^{(i)}_{t+1:t+H}\}^C_{i=1} = \boldsymbol{\mathcal{M}}(\{x^{(i)}_{t-L:t}\}^C_{i=1}) \tag{1}$$

Time Series Normalization

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

$$\{y^{(i)}_{t+1:t+H}\}^C_{i=1} = \boldsymbol{\mathcal{N}}^{-1}(\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{N}}(\{x^{(i)}_{t-L:t}\}^C_{i=1}))) \tag{2}$$

Since statistics often have additional networks in $\boldsymbol{\mathcal{N}}$ for adaptive learning, the statistics adopted by $\boldsymbol{\mathcal{N}}$ and $\boldsymbol{\mathcal{N}}^{-1}$ 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 $\{x^{(i)}_{t-L:t}\}^C_{i=1}$, and σ_t is its variance. The subscripts l and h donate 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:

$$y^{(i)}_{t+1:t+H} = \sigma^{(i)}_{h,t}\boldsymbol{\mathcal{M}}(\frac{x^{(i)}_{t-L:t} - \mu^{(i)}_{l,t}}{\sigma^{(i)}_{l,t}}) + \mu^{(i)}_{h,t} \tag{3}$$

Methodology

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 plug-in 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.

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

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

$$\boldsymbol{T}_t = \boldsymbol{T}^{\mathrm{TiD}}_t + \boldsymbol{T}^{\mathrm{DiW}}_t \tag{4}$$

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 $\boldsymbol{T}_t \in \mathbb{R}^D$ against a shared learnable prototype library $\boldsymbol{P} = \{\boldsymbol{p}_j \in \mathbb{R}^D\}^N_{j=1}$ via inner product similarity:

$$\boldsymbol{S}_t = \boldsymbol{T}_t\boldsymbol{P}^\top \tag{5}$$

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

$$\boldsymbol{w}_{t,[1:k]} = \mathrm{Softmax}(\boldsymbol{S}_{t,[1:k]}) \tag{6}$$

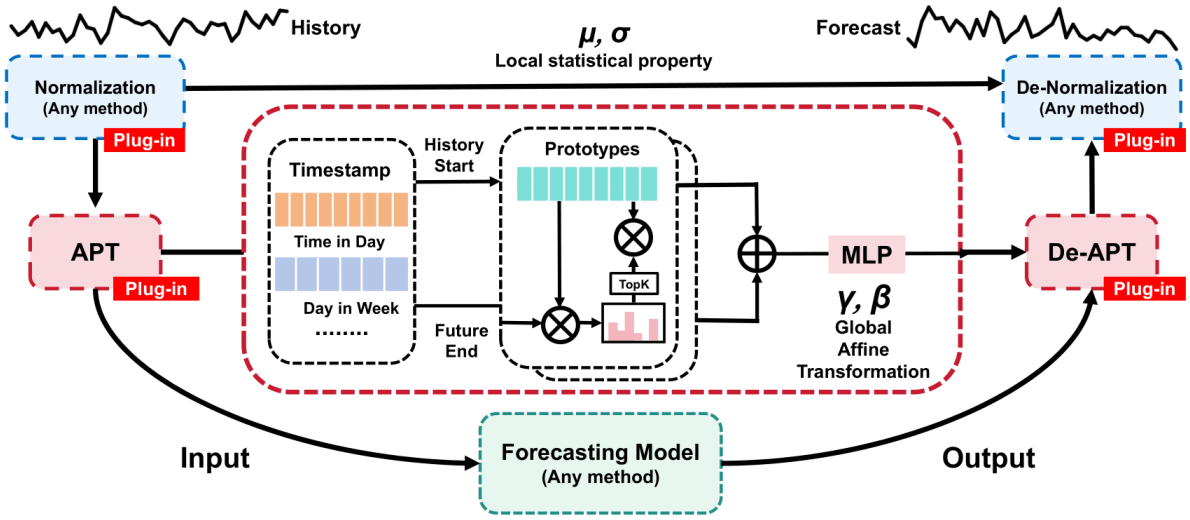

Figure 3: The pipeline of time series forecasting and the schematic of APT.

This yields a sparse weight matrix $\boldsymbol{W}_t \in \mathbb{R}^N$, 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:

$$\widetilde{\boldsymbol{T}}_t = \sum_{j=1}^{N} \boldsymbol{W}_{t,j} \boldsymbol{p}_j \tag{7}$$

To encode both past and future temporal context, we aggregate their embeddings: $\widetilde{\boldsymbol{T}}_t = \widetilde{\boldsymbol{T}}_t^{history} + \widetilde{\boldsymbol{T}}_t^{futher}$. Optionally, channel identity embeddings $\boldsymbol{ID} \in \mathbb{R}^{C \times D}$ 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:

$$\gamma_t, \beta_t = \mathrm{MLP}_\gamma(\widetilde{\boldsymbol{T}}_t), \mathrm{MLP}_\beta(\widetilde{\boldsymbol{T}}_t) \tag{8}$$

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:

$$y^{(i)}_{t+1:t+H} = \frac{\sigma^{(i)}_{h,t}}{\gamma^{(i)}_t}\left(\boldsymbol{\mathcal{M}}\left(\gamma^{(i)}_t \frac{x^{(i)}_{t-L:t} - \mu^{(i)}_{l,t}}{\sigma^{(i)}_{l,t}} + \beta^{(i)}_t\right) - \beta^{(i)}_t\right) + \mu^{(i)}_{h,t} \tag{9}$$

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

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 $\boldsymbol{E} = \{\boldsymbol{T}^{\mathrm{TiD}}, \boldsymbol{T}^{\mathrm{DiW}}, \boldsymbol{P}\}$. The loss is defined as:

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

Where $\odot$ is the Hadamard product, I is the identity matrix, and $||_2$ 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. 2024). 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 $\boldsymbol{W} \in \mathbb{R}^{B \times N}$ in each batch in Equation 7:

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

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:

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

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

$$Loss_{APT} = Loss_{orth} + Loss_{balance} + Loss_R \tag{13}$$

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:

$$Loss_{main} = Loss_{normal} + Loss_{MSE} \tag{14}$$

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**
Weather	None	0.280	0.229	**0.268**	**0.223**	0.396	0.401	**0.311**	**0.267**	0.302	0.256	**0.285**	**0.239**	0.307	**0.254**	**0.299**	0.260
	+RevIN	**0.258**	**0.221**	0.259	0.224	0.294	0.279	**0.274**	**0.243**	**0.267**	**0.233**	0.269	**0.233**	0.283	**0.252**	**0.282**	**0.252**
	+Dish-TS	0.277	**0.225**	**0.276**	0.226	0.316	0.296	**0.295**	**0.276**	0.296	0.258	**0.285**	**0.240**	0.301	**0.240**	**0.299**	**0.240**
	+SAN	**0.277**	**0.225**	0.279	0.227	0.286	0.261	**0.276**	**0.245**	**0.271**	**0.228**	0.276	0.235	0.279	**0.226**	**0.277**	0.227
	+FAN	**0.281**	**0.231**	0.283	0.234	0.285	0.246	**0.280**	**0.245**	0.288	0.236	**0.276**	**0.230**	**0.276**	**0.228**	0.278	0.229
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

Experiments

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.

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.

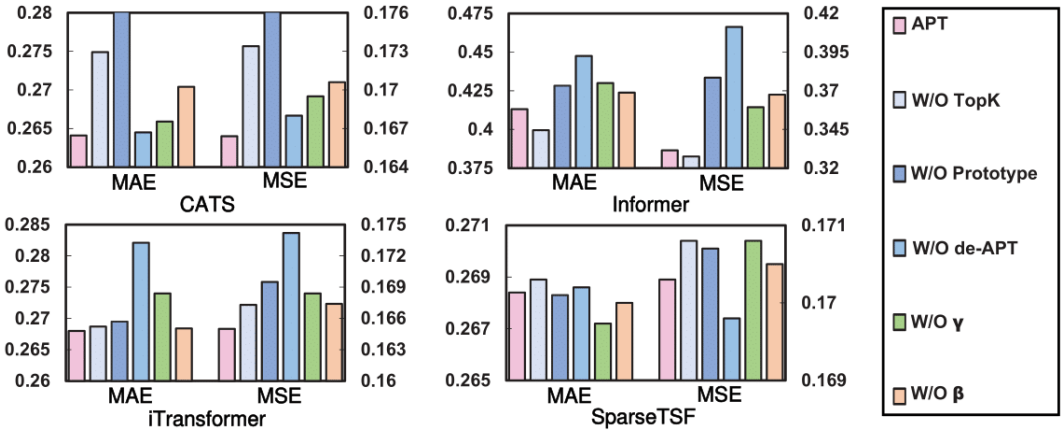

Figure 4: The ablation study results of APT components on the ECL dataset, $L = 336, H = 336$.

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.

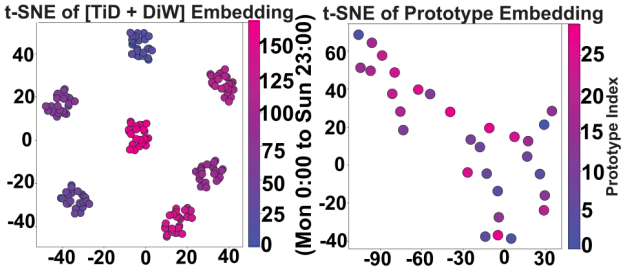

Figure 5: Visualization of APT timestamps and prototype embeddings on ECL dataset and iTransformer

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.

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	0.332	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**	1.490	1.144	2.070	1.009	1.789	1.009	**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	0.243	0.301	0.246
	Informer	**0.315**	**0.270**	**0.315**	0.271	0.326	0.284	0.328	0.291	0.410	0.389	0.554	0.816
	iTransformer	**0.293**	**0.251**	0.296	0.254	**0.293**	**0.251**	0.295	0.254	0.296	0.254	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. $L = 336$, $H = 336$

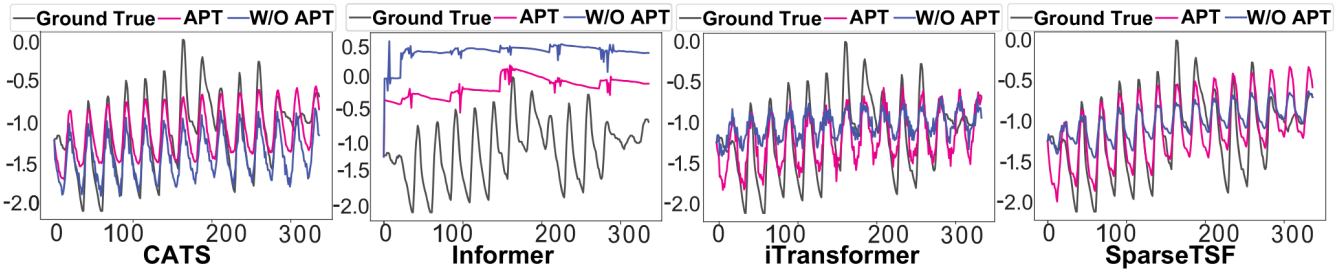

Figure 6: Visualization of forecasting results for different models on the ETTh2 dataset without normalization strategy

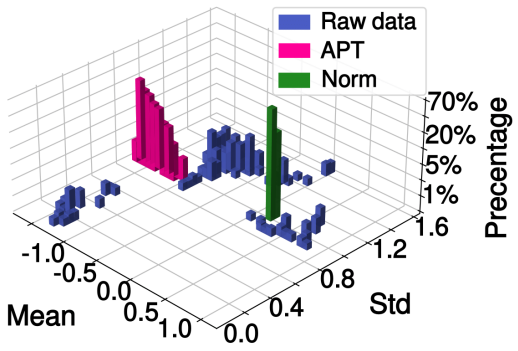

Figure 7: 3D visualization of temporal distribution and ratio of forecasting pipeline at different stages on ETTh1.

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 informative 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), paramenter count (C.6), extra ablation study (C.7), cross-setting finetuning (C.8), visualization (C.9) and discussion (D).

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.

Acknowledgments

This work is supported by the NSFC underGrant 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.; Georgiopoulos, 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 Hinaut, 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, F.; Li, Y.; Shao, Z.; Yu, C.; Fu, Y.; An, Z.; Xu, Y.; and Cheng, X. 2025. ARIES: Relation Assessment and Model Recommendation for Deep Time Series Forecasting. *arXiv preprint arXiv:2509.06060*.

Wang, L.; Gao, H.; Zhao, C.; Sun, X.; and Dai, D. 2024. 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*.