Adaptive Latent Decomposition for Domain Generalization in Time Series Forecasting

SONGGAOJUN DENG, Eindhoven University of Technology, Eindhoven, The Netherlands
ZEHAO XIAO, AI for Medical Imaging Laboratory, University of Amsterdam, The Netherlands
MAARTEN DE RIJKE, University of Amsterdam, Amsterdam, The Netherlands

Time series forecasting is essential in many real-world applications, yet developing models that generalize well to unseen and related domains—such as forecasting web traffic on new web sites/platforms or predicting e-commerce demand in new regions—remains a big challenge. Prior work addressing this problem, known as domain generalization, focuses on identifying common patterns but often overlooks the complex characteristics of time series data and fails to use the available information from test samples of unseen domains.

We propose a novel approach, **adaptive latent decomposition (ALD)** for domain generalization in time series forecasting, which consists of a decomposed **variational autoencoder (VAE)** and an adaptive inference mechanism to improve predictive performance in unseen domains. The decomposed VAE involves a learnable kernel-selection mechanism that learns latent variables of decomposed components of time series, i.e., trend-cyclical and seasonal components. These latent variables capture hidden temporal dependencies in time series data, allowing forecasting models to learn general patterns from various seen domains. The adaptive inference mechanism bridges the gap between seen and unseen domains with a sample-wise optimization strategy specifically designed for time series forecasting. ALD builds a latent variable-aware pre-trained model and tailors it for each test sample, improving the generalization on unseen test domains. We validate ALD across six real-world datasets, from online behavior to complex temporal systems, demonstrating its superior generalization performance compared to state-of-the-art methods.

CCS Concepts: • **Applied computing** → **Forecasting**; • **Mathematics of computing** → **Time series analysis**; • **Information systems** → *Web mining*; *Traffic analysis*;

Additional Key Words and Phrases: Time series forecasting, Domain generalization, Test-time adaptation

Associate Editor: Jason Wang

Work was conducted while Songgaojun Deng was at AIRLab, University of Amsterdam.
This research was (partially) funded by Ahold Delhaize, the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, https://hybrid-intelligence-centre.nl, project LESSEN with project number NWA.1389.20.183 of the research program NWA ORC 2020/21, which is (partly) financed by the Dutch Research Council (NWO), project ROBUST with project number KICH3.LTP.20.006, which is (partly) financed by the Dutch Research Council (NWO) and the Dutch Ministry of Economic Affairs and Climate Policy (EZK) under the program LTP KIC 2020-2023, and the UNITE project funded by the European Union under project no. 101201510.
All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.
Authors' Contact Information: Songgaojun Deng (corresponding author), Eindhoven University of Technology, Eindhoven, The Netherlands; e-mail: s.deng@tue.nl; Zehao Xiao, AI for Medical Imaging Laboratory, University of Amsterdam, The Netherlands; e-mail: z.xiao@uva.nl; Maarten de Rijke, University of Amsterdam, Amsterdam, The Netherlands; e-mail: m.derijke@uva.nl.

ACM 1556-472X/2026/8-ART109
https://doi.org/10.1145/3819822

ACM Reference format:
Songgaojun Deng, Zehao Xiao, and Maarten de Rijke. 2026. Adaptive Latent Decomposition for Domain Generalization in Time Series Forecasting. *ACM Trans. Knowl. Discov. Data.* 20, 7, Article 109 (August 2026), 29 pages.
https://doi.org/10.1145/3819822

1 Introduction

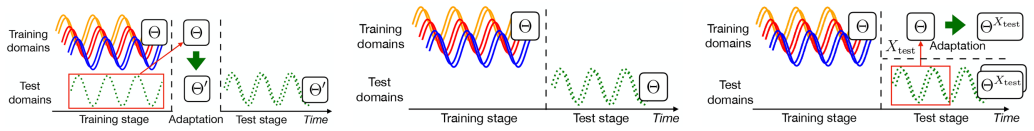
Time series forecasting is a critical task across various practical fields, including web traffic analysis [9], e-commerce [7], financial markets [45], and energy usage [15]. It focuses on estimating future movements based on past observations, making accurate forecasting vital for informed decision-making and resource allocation. Effective forecasting can lead to significant benefits, such as optimized web sites, improved supply chain management, enhanced user experiences, and better public health responses [63].

A major challenge in time series forecasting is domain distribution shift [20, 25], which stems from variations in data sources, platforms, or interaction contexts. This is critical in time-sensitive applications such as allocating server resources for new web sites, forecasting demand for new products, analyzing user behavior across demographics, or predicting disease spread in new regions.

Addressing Distribution Shifts. To address distribution shifts, there are two classes of methods, **domain adaptation (DA)** [24, 54] and **domain generalization (DG)** [25, 79]. As illustrated in Figure 1(a), DA requires target domain data during training, which is not always available in real applications, e.g., in retail demand forecasting for a newly opened store with no historical records. In contrast, we focus on DG, which (Figure 1(b)) does not assume available target data during training. In time series forecasting, prior studies (i) primarily use feature alignment/matching [20, 21, 89] to capture common features for DG across different sources (i.e., distribution shifts over data *sources*), (ii) develop time-sensitive networks [5, 59] or learn invariant representations across inferred pseudo-environments [49] to achieve generalization over time domains (i.e., distribution shifts over *time*) [25]. These methods often treat time series as a single undifferentiated sequence, overlooking their underlying latent factors, which correspond to distinct components such as seasonality, trend, and residual variations that compose a time series. In multi-domain time series, these components often manifest as shared temporal patterns (e.g., common seasonality) and domain-specific elements (e.g., a unique local trend), as shown in Figure 2. These patterns are not easily apparent when data is noisy or the number of domains is large. Consequently, failing to disentangle these factors hinders knowledge transfer between domains and ultimately limits a model's generalization capabilities. Moreover, a key challenge in DG is the distribution gap between training and unseen test domains, which makes existing methods unreliable when applied to new scenarios, especially in time series forecasting, where dynamic patterns in samples further complicate generalization.

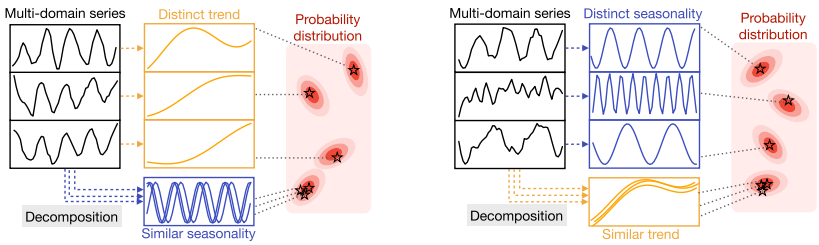
Research Question. Based on the above challenges, this work seeks to answer the following main research question: *How can a time series forecasting model leverage latent temporal factors and test-time information to generalize to unseen domains?*

Research Objectives. To answer this research question, this work pursues two primary objectives. First, we aim to enhance a model's generalization to unseen domains by exploring the potential of *latent variables* that correspond to distinct temporal patterns. Second, to bridge the distribution gap between training and unseen test domains, we seek to integrate information from test samples into the trained model, enabling an adaptive inference capability for unseen scenarios. To achieve our goals, we need to address several challenges: (i) Temporal dependencies are inherently complicated to quantify and explicitly model (i.e., how previous timesteps affect future ones). This difficulty



(a) Domain adaptation (b) Domain generalization (c) Test-time adaptive inference

Fig. 1. Learning frameworks for addressing domain shifts in time series forecasting. (a) *DA* uses both training (source) and limited test (target) domain data (i.e., both input and output sequences) during training. (b) *DG* requires no access to target domain samples during training while lacking test domain information for predictions. (c) Our *test-time adaptive inference* performs optimization on each target-free test sample (i.e., using only the input sequences) during inference, improving generalization by leveraging test-time information. Red boxes indicate data used for adaptation. Θ denotes model parameters. DA, domain adaptation; DG, domain generalization.



(a) Similar-seasonality domains (b) Similar-trend domains

Fig. 2. Motivating examples showing multi-domain time series where decomposition reveals either similar or distinct components (trend vs. seasonality), each associated with different underlying probability distributions.

arises from information loss caused by windowing processes and, more importantly, the complex nature of underlying time series components like trends and seasonality. (ii) Acquiring knowledge from test domains to enhance generalization is hindered by the lack of ground truth data from unseen domain samples, making it difficult to effectively adjust a pre-trained model. This challenge is further compounded when test samples exhibit diverse characteristics.

Proposed Framework. To address the challenges listed above, we propose **adaptive latent decomposition (ALD)**, a framework for DG in time series forecasting. ALD models decomposed temporal dependencies in latent spaces to enhance generalization and further improve forecasting precision on unseen domains with a test-time adaptive inference mechanism. To do so, ALD introduces a decomposed **variational autoencoder (VAE)** that estimates latent representations from the decomposed components of the input time series (i.e., trend-cyclical and seasonal). The latent representations are trained to capture different aspects of underlying temporal dependencies, modeling key and general knowledge of the input time series dynamics from training domains to improve generalization for forecasting. Based on the decomposed VAE, we propose a sample-specific adaptive inference strategy at test time, which performs a single-step test-time optimization with a self-synthetic sample constructed using only the single test input sequence as shown in Figure 1(c). The strategy yields tailored model parameters of each test sample, effectively narrowing the domain gaps between the source-trained model and the unseen test data. The approach incorporates test domain information without requiring resource- or compute-intensive adaptation and re-training, ensuring both effectiveness and efficiency.

Contributions. In summary, we make the following key contributions:

—We propose ALD, a framework for DG in time series forecasting that captures decomposed temporal information and enables adaptive inference on unseen data.
—We introduce an ALD framework with a learnable kernel-selection mechanism that learns latent variables modeling key components of time series data. The latent variables capture the essential underlying data dynamics, improving forecasting performance on unseen domains.
—We develop an efficient test-time adaptive inference strategy that performs sample-specific optimization with a self-synthetic sample generation method, which tailors model parameters for each test sample, yielding adaptive and robust forecasting.

We conduct extensive experiments on six real-world datasets from diverse application domains. The results show that our proposed method outperforms other baselines, achieving state-of-the-art performance in DG for time series forecasting.

2 Related Work

2.1 Time Series Forecasting

Over the past few decades, many efforts have been made to develop time series forecasting models. Classical approaches, like **seasonal naive (SN)**, **autoregressive (AR)** [75], ARIMA [10], and exponential smoothing [28], mainly model linear patterns in time series data. To capture non-linear dependencies, machine learning approaches such as regression [75] and tree-based models [40] have been explored. In recent years, deep learning methods have gained significant attention for their ability to capture more complex patterns, including **recurrent neural networks (RNNs)** [17, 35], temporal convolutional networks [6, 61], attention-based models [4, 60, 77], transformer-based models such as Transformer [47], Informer [93], Autoformer [83], Fedformer [95], and latent representation based model LaST [81] for time series forecasting. A body of work addresses non-stationary time series forecasting by handling temporal distribution shifts, such as RevIN [42], DishTS [23], and FAN [86]. Nowadays, the success of **large language models (LLMs)** [65, 91] has boosted time-series forecasting models based on pre-trained LLMs, leading to state-of-the-art models like GPT4TS [96], TimeLLM [38], and UniTime [53]. Different perspectives have been published about the actual effectiveness of LLMs in time series tasks [74]. Recently, large-scale time series foundation models with LLM backbones, such as MOIRAI [82], Chronos [2], and TimesFM [19], have demonstrated strong performance across diverse tasks. This work explores lightweight models for DG of time series forecasting.

2.2 Out-of-Distribution (OOD) Generalization in Time Series

OOD generalization, which addresses distribution shifts between training and test domains, has been studied extensively in static computer vision tasks [50, 79]. Recently, OOD generalization has gained increasing attention in time series applications [25, 39, 52]. Such efforts can be categorized into three areas.

DA focuses on transferring knowledge learned from source domains to different but related target domains, often assuming *limited or unlabeled target domain data are available during training*. DA can be categorized into supervised [57] and unsupervised [54] methods, depending on whether labels of target domain data are used. In the context of time series, prior work has mainly focused on techniques such as adversarial training [64], pattern alignment [13, 33, 52, 84], and self-supervision methods [39]. These methods heavily rely on target domain data during the training phase. Our work addresses DG.

DG aims to develop models that perform well on unseen domains without access to data from those domains during training. DG in time series can be divided into generalization across *time* domains or *source* domains, depending on whether the shifts occur over time or across data sources.

In *time* DG, research has explored time-sensitive networks [5, 14] and gradient-based methods [59] to adapt evolving behaviors. Other approaches identify domains with different distributions and learn common patterns using distribution matching [21], adversarial learning [56], or invariant learning [49]. For *source* DG, most studies focus on classification tasks, leveraging label information through distribution matching [89], data augmentation [89], and contrastive learning [36, 66]. A recent study proposed Cedar [20], which addresses DG in time series forecasting using domain discrepancy regularization with domain difficulty. In a more general setting than the time series domain, Shi et al. [70] proposed IDGM, which aligns gradients across domains to achieve DG. Invariant learning approaches are also widely used for DG. For example, GroupDRO [68] is a robust optimization method that minimizes the worst-case group loss, and DANN [26] is an adversarial domain alignment framework that learns domain-invariant representations. In contrast, we focus on the time series forecasting task by learning latent variables of decomposed time series components and developing an effective test-time adaptive inference strategy to enhance the generalization abilities of forecasting models in new domains.

Test-time adaptation (TTA) achieves adaptation on target data during inference, with a training stage identical to DG, and can be an effective tool for DG [22]. Unlike traditional DA, which requires access to target-domain data (both input and output sequences) during training, TTA adapts the model at test time using only test input sequences of unseen domains. TTA has recently emerged as a novel learning paradigm [48, 85], offering the advantage of computational efficiency. Existing methods can be categorized based on the aspect of adaptation they target, including model [67, 72, 73, 78], inference [90], normalization [30], sample [27], and prompt [1] adaptations. Most studies focus on applications in classification tasks in computer vision [27, 73, 78], but there has also been notable work in reinforcement learning [51] and natural language processing [8]. Recent works explore TTA in time series forecasting with partially observed ground truth at test time [41] or introducing specialized layers during training [16], which are not suitable for our DG setting. Our approach is a fully TTA strategy that relies solely on self-synthesized samples and does not impact the training process.

3 Problem Formulation

This study focuses on DG in univariate time series forecasting [20, 25]. We provide a detailed introduction as follows. Our key mathematical notation is listed in Table 1.

Time Series Forecasting. A univariate time series dataset $D = \{(X_i, Y_i)\}_{i=1}^{N}$ consists of N samples. $X_i = (\mathbf{x}_i, \mathbf{a}_i)$ represents the input features, including the observed historical time series $\mathbf{x}_i \in \mathbb{R}^{T\times 1}$ and possible external factors $\mathbf{a}_i \in \mathbb{R}^{T\times d_a}$ (e.g., time-related features with dimension d_a). T is the length of the historical window. The output Y_i can be written as $\mathbf{y}_i \in \mathbb{R}^{h\times 1}$ representing the future values to be predicted, where $h \geq 1$ is the forecasting horizon. For simplicity, we omit subscripts in the following discussion. In this work, we focus on probabilistic forecasting [2, 20, 69, 82], which estimates the probability distribution of future time series given the input features $P(Y|X;\Theta)$. Here, $\Theta = (\mu, \sigma)$ represents the location and scale parameters of a Normal distribution. It captures uncertainty and provides more insight into predicting real data.

DG. Let $\mathcal{D} = \{D^1, D^2, \ldots, D^K\}$ represent the set of K domains, where the jth domain consists of a time series dataset D^j. We have access to M training domains $\mathcal{D}_{\text{train}} = \{D^j\}_{j=1}^{M}$ where $M < K$. The goal is to learn a time series forecaster that performs well to unseen test domains $\mathcal{D}_{\text{test}} = \{D^j\}_{j=M+1}^{K}$.

Distribution Shifts. We focus on *covariate shifts* [88]. Covariate shifts occur when marginal probability distributions of X differ across training and test domains, $P_{\text{train}}(X) \neq P_{\text{test}}(X)$, while conditional probability distributions generally remain constant across domains, $P_{\text{train}}(Y|X) = P_{\text{test}}(Y|X)$.

Table 1. Key Notations

Notation	Description
T, h	Historical and forecasting window sizes
N, K, M	Number of samples, total domains, and training domains
Θ	Learnable model parameters
$\mathbf{x}_i \in \mathbb{R}^{T\times 1}$	Input historical time series
$\mathbf{a}_i \in \mathbb{R}^{T\times d_a}$	Exogenous attributes (d_a: dimension)
$\mathbf{y}_i \in \mathbb{R}^{h\times 1}$	Output time series
$\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}}$	Training and test domains
N_k	Number of candidate kernels for decomposition
$\mathbf{x}_s, \mathbf{x}_t$	Seasonal and trend-cyclical components
$\hat{\mathbf{x}}_s, \hat{\mathbf{x}}_t$	Reconstructed components from decomposed VAE
$\mathbf{z}_s, \mathbf{z}_t, \mathbf{z}$	Latents of seasonal, trend-cyclical, and their combination
$p_{\theta_s}, p_{\theta_t}$	Parameters of seasonal/trend VAE decoders
q_{ϕ_s}, q_{ϕ_t}	Parameters of seasonal/trend VAE encoders

Data Assumptions. We consider the *common underlying patterns* assumption [20], which posits that the existence of shared patterns across domains supports the relevance of generalization tasks. This is natural, and we experiment with domains pertinent to specific fields, such as the various country domains of a web site.[1]

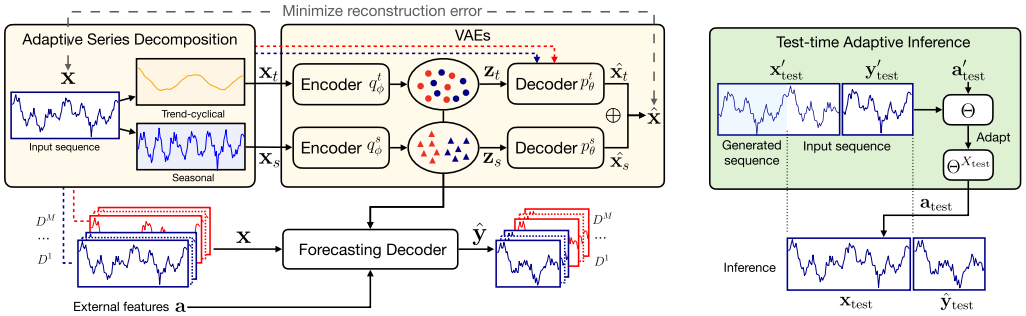
4 Methodology

We propose a novel approach for DG in time series forecasting, named ALD, illustrated in Figure 3. The key novelty of ALD lies in its "*adaptive*" design, which enables automatic kernel selection for series decomposition and facilitates the learning of latent decomposed representations across domains, as well as in its TTA strategy, which further dynamically adjusts the model to unseen scenarios during inference. Our approach consists of two key parts that work complementarily: a *decomposed VAE* and *test-time adaptive inference*.

- The decomposed VAE adaptively separates each input series into trend-cyclical and seasonal components through an automatic kernel selection mechanism and encodes them via a VAE trained across multiple source domains. This process captures latent factors that are general and shared across domains while filtering out domain-specific noise, thereby improving generalization to unseen domains.
- The test-time adaptive inference further refines the model for each test sample using self-synthetic sample generation and adaptation. This helps the latent representations to align with the target distribution without requiring additional labeled data, mitigating distribution shifts at inference time.

Together, they ensure that the forecaster leverages stable, transferable latent representations while remaining flexible to domain-specific variations, leading to robust forecasting performance on unseen domains. Below, we illustrate each part of our approach in detail.

[1]We do not emphasize the *no abrupt distribution shifts* assumption in prior work [20] as it is difficult to quantify in real-world cases. We alleviate this effect using a reversible instance norm [42] and also ensure the recency of training domains' data.



(a) Decomposed VAE (b) Test-time adaptive inference

Fig. 3. The proposed framework, ALD, consists of two parts: (a) Decomposed VAE, which learns decomposed latent variables that capture general underlying temporal dependencies of different aspects of the data, assisting the forecasting decoder in improving generalization to new domains; and (b) Test-time adaptive inference, which uses target-free test samples to optimize the pre-trained model during inference.

4.1 Decomposed VAE

To achieve DG, prior work has focused on capturing common patterns across domains through temporal feature distribution alignment [20, 21, 89] or learning evolving model parameters [5, 14] to dynamically model unseen domain patterns over time. However, relying solely on domain-shared features can lead to over-generalization [94], reducing forecasting accuracy for time series, and parameter generation methods result in over-parameterization, increasing model complexity and thus hindering generalization ability. To address these limitations, we propose a decomposed VAE that captures the underlying factors of complex time series data and uses this information to enhance generalization in unseen domains. The decomposed VAE is a *VAE* framework built upon a *time series decomposition* component.

4.1.1 Adaptive Series Decomposition. Time series decomposition is a powerful technique for disentangling components like trend, seasonality, and noise [32]. However, existing methods [18, 32, 83, 95] typically rely on static parameters (e.g., fixed kernel size), which are ill-suited for datasets where seasonal patterns may vary across domains (see Figure 2(b)). We address this limitation with a novel, adaptive series decomposition block that formulates kernel selection as a differentiable, learnable process conditioned on the input, enabling adaptation to domain-specific temporal structures. By modifying the decomposition from Autoformer [83], our method dynamically selects the optimal moving average kernel size for a given input series $\mathbf{x}$, thereby decoupling it into its trend-cyclical $\mathbf{x}_t$ and seasonal $\mathbf{x}_s$ components.

While classical **adaptive moving averages (AMAs)** (e.g., Kaufman's AMA) exist, they rely on non-differentiable heuristic rules (based on volatility or trend) to adjust the smoothing factor. This makes them incompatible with end-to-end deep learning optimization. Crucially, these methods only adjust the smoothing factor but fail to select the optimal temporal kernel size needed for complex seasonal patterns. Our approach is novel and overcomes these limitations by making the kernel selection a differentiable learning process.

Our method first employs a prediction network, $f_k(\cdot)$ to generate logits for a set of N_k candidate kernels $K = \{k_1, k_2, \ldots, k_{N_k}\}$ by learning the input sequence, i.e., $\mathbf{l} = [l_1, l_2, \ldots, l_{N_k}] = f_k(\mathbf{x})$. To ensure a smooth, differentiable training signal for the prediction network, we convert these logits into a probability distribution $\mathbf{w}$ using a softmax function, where each weight $w_i = \frac{e^{l_i}}{\sum_{j=1}^{N_k} e^{l_j}}$ represents the model's learned importance weight for kernel k_i. The trend-cyclical component is

computed as a weighted sum of the moving average outputs from all candidate kernels:

$$\mathbf{x}_t = \sum_{i=1}^{N_k} w_i \cdot \mathrm{MA}(k_i.\mathbf{x}). \tag{1}$$

Here, $\mathrm{MA}(k_i,)$ denotes a moving average operation with kernel size k_i and appropriate padding to preserve the sequence length. The seasonal component is then obtained as the residual:

$$\mathbf{x}_s = \mathbf{x} - \mathbf{x}_t. \tag{2}$$

This adaptive series decomposition separates the time series into trend-cyclical $\mathbf{x}_t$ and seasonal $\mathbf{x}_s$ components, while dynamically adapting to varying seasonalities, thereby enhancing the model's DG capabilities.

4.1.2 VAE. We model each decomposed component, i.e., trend-cyclical and seasonal, independently using a VAE to obtain latent variables $\mathbf{z}_t$ and $\mathbf{z}_s$, which capture key information of trend-cyclical and seasonal patterns, respectively. The VAE follows an encoder–decoder structure, as detailed below.

Encoder. Two encoders learn to approximate the posterior distributions $q_{\phi_t}(\mathbf{z}_t|\mathbf{x}_t), q_{\phi_s}(\mathbf{z}_s|\mathbf{x}_s)$ of the latent variables $\mathbf{z}_t, \mathbf{z}_s$ given the decomposed trend-cyclical and seasonal components $\mathbf{x}_t, \mathbf{x}_s$:

$$\mathbf{z}_t \sim q_{\phi_t}(\mathbf{z}_t|\mathbf{x}_t), \text{ and } \mathbf{z}_s \sim q_{\phi_s}(\mathbf{z}_s|\mathbf{x}_s). \tag{3}$$

Decoder. Two decoders reconstruct the trend-cyclical and seasonal components, respectively, using latent variables and the domain identifier. The domain identifier embeds domain-aware information during VAE training to enhance accurate sequence reconstruction and thus to help effective latent variable learning:

$$\hat{\mathbf{x}}_t = p_{\theta_t}(\mathbf{x}_t|\mathbf{z}_t, \mathsf{DomID}) \text{ and } \hat{\mathbf{x}}_s = p_{\theta_s}(\mathbf{x}_s|\mathbf{z}_s, \mathsf{DomID}), \tag{4}$$

where $\hat{\mathbf{x}} = \hat{\mathbf{x}}_t + \hat{\mathbf{x}}_s$ is the final reconstructed sequence. DomID is the identifier of a training domain, implemented as a one-hot encoded binary vector. This design allows domain-agnostic testing, as only the encoder is required to generate the latent variables.

4.1.3 Objective Function. We train the decomposed VAE by minimizing the negative evidence lower bound for both trend-cyclical and seasonal components. The total objective combines the reconstruction loss between the original input $\mathbf{x}$ and its reconstructed estimate $\hat{\mathbf{x}}$, together with the **Kullback-Leibler (KL)** divergence regularization for the latent posteriors:

$$\mathcal{L}_{\text{latent}} = \mathbb{E}_{\mathbf{x}}[\|\hat{\mathbf{x}} - \mathbf{x}\|^2] + \beta \left(\mathrm{KL}\left(q_{\phi_t}(\mathbf{z}_t|\mathbf{x}_t) \,\|\, p(\mathbf{z}_t)\right) + \mathrm{KL}\left(q_{\phi_s}(\mathbf{z}_s|\mathbf{x}_s) \,\|\, p(\mathbf{z}_s)\right)\right), \tag{5}$$

where $p(\mathbf{z}_t), p(\mathbf{z}_s)$ are Gaussian distributions $\mathcal{N}(0, \boldsymbol{I})$ and $\boldsymbol{I}$ is the identity matrix.[2] We introduce a weighting hyperparameter β, to regulate the capacity of the latent space in our model [34]. Larger values of β (e.g., $\beta > 1$) restrict the capacity of the latent vector, encouraging each dimension of the latent vector to capture distinct, conditionally independent factors from the input sequence. By reconstructing $\mathbf{x}$ through latent variables $\mathbf{z}_t$ and $\mathbf{z}_s$ with VAEs, the encoders are trained to capture general temporal information of the decomposed patterns across domains for new domain prediction. Our VAE model differs from LaST [81] in that LaST learns to separate seasonal-trend latent representations in a standard single-domain setting and does not aim to learn representations that generalize across domains.

[2]We analyze and demonstrate that an identical prior distribution does not lead to similar learned latent variables for the two components (Section 5.4.2).

Algorithm 1: Test-Time Adaptive Inference

Input: Test domain data $\mathcal{D}_{\text{test}}$ and pre-trained model Θ.

for *each* $X_{test} = (\mathbf{x}_{test}, \mathbf{a}_{test})$ *in* $\mathcal{D}_{test}$ **do**

▷ Generate a self-synthetic sample.

$(\mathbf{x}'_{\text{test}}, \mathbf{a}'_{\text{test}}, \mathbf{y}'_{\text{test}}) \leftarrow \text{Augment}(\mathbf{x}_{\text{test}}, \mathbf{a}_{\text{test}})$

▷ Perform optimization.

$\mathcal{L}_{\text{tto}}(X_{\text{test}}) = -\log p_\Theta\left(\mathbf{y}'_{\text{test}} \mid \mathbf{x}'_{\text{test}}, \mathbf{a}'_{\text{test}}\right)$

$\Theta^{X_{\text{test}}} = \Theta - \nabla_\Theta\left(\mathcal{L}_{\text{tto}}(X_{\text{test}})\right)$

▷ Perform inference.

$\hat{\mathbf{y}}^{\text{test}} = p_{\Theta^{X_{\text{test}}}}\left(\mathbf{y}_{\text{test}} \mid \mathbf{x}_{\text{test}}, \mathbf{a}_{\text{test}}\right)$

Output: All predictions on test domain samples.

4.1.4 Forecasting Decoder. We incorporate the learned decomposed latent variables into a flexible forecasting decoder (i.e., the backbone) to enhance its generalization in time series forecasting. We first construct a gated latent representation by linearly combining the trend-cyclical and seasonal components:

$$\mathbf{z} = \sigma(\alpha) \cdot \mathbf{z}_t + \left(1 - \sigma(\alpha)\right) \cdot \mathbf{z}_s, \tag{6}$$

where α is a learned gating scalar constrained to the range $(0, 1)$ with the Sigmoid function σ. While we use a scalar gate, more flexible options, such as dimension-wise or input-dependent gating, could also be applied. The fused latent vector $\mathbf{z}$ is then concatenated with the original input $\mathbf{x}$ and passed through a linear transformation with residual connection:

$$\bar{\mathbf{x}} = \mathbf{x} + [\mathbf{z} \,\|\, \mathbf{x}]\, \mathbf{W}_x + \mathbf{b}_x, \tag{7}$$

where $\mathbf{W}_x \in \mathbb{R}^{(d_z+T)\times T}$, $\mathbf{b}_x \in \mathbb{R}^T$ are learnable parameters and $\|$ denotes concatenation. We compute $\bar{\mathbf{a}}$ for potential external features using the same method as in Equation (7). The enhanced inputs $\bar{\mathbf{x}}, \bar{\mathbf{a}}$ are fed into a time series forecasting model for downstream calculations. In the probabilistic forecasting setting, we optimize the forecasting decoder by minimizing the negative log-likelihood:

$$\mathcal{L}_{\text{fcst}} = -\mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\log p_\Theta\left(\mathbf{y} \mid \bar{\mathbf{x}}, \bar{\mathbf{a}}\right)\right]. \tag{8}$$

4.1.5 Training. We propose a two-stage training approach to effectively learn latent variables, enabling the forecasting decoder to fully use them for optimizing the forecasting objective.

Stage 1: Learning Decomposed Latent Variables. We minimize $\mathcal{L}_{\text{latent}}$ (Equation (5)), ensuring the latent spaces capture meaningful information through training the decomposed VAE framework.

Stage 2: Forecasting Optimization with Latent Integration. We minimize the forecasting loss $\mathcal{L}_{\text{fcst}}$ (Equation (8)). Decomposed VAE encoders are fine-tuned for the forecasting task. The transformation of latent vectors is jointly trained (Equation (7)) to enable the forecasting model to fully use latent information, thereby enhancing forecasting accuracy.

4.2 Test-Time Adaptive Inference (TTA)

Adaptive Inference through Test-Time Optimization. Our decomposed VAE models general temporal dependencies at both trend and seasonal levels, enhancing robust predictions across diverse domains. Nevertheless, domain shifts still introduce adaptation gaps between the trained model and the data in the test domain [22]. To bridge this gap, we introduce an adaptive inference mechanism that performs a single gradient update on each individual test sample before inference. This updates the model parameters to better align with the test sample, reducing the domain gap and yielding more robust predictions.

Specifically, the pre-trained model parameters Θ are updated for each test sample $X_{\text{test}} = (\mathbf{x}_{\text{test}}, \mathbf{a}_{\text{test}})$ to create sample-specific parameters $\Theta^{X_{\text{test}}}$:

$$\Theta^{X_{\text{test}}} = \Theta - \eta_{\text{tta}} \nabla_{\Theta} \big(\mathcal{L}_{\text{tto}} \left(X_{\text{test}} \right) \big), \tag{9}$$

where $\mathcal{L}_{\text{tto}}$ represents the objective function for test-time optimization. η_{tta} is the test-time adaptation learning rate.

By updating the model parameters for each test sample, the model parameters become adaptive and tailored to the test sample, such as aligning the seasonality and means. This reduces the domain gaps between model parameters and the test sample, yielding more robust predictions in unseen scenarios. The entire process is performed on a per-sample basis, fully adhering to the DG setting of not using external target data. The single optimization step also ensures minimal computational overhead.

Self-Synthetic Test-Time Optimization. As the target sequences of test samples are unavailable at inference, the optimization in Equation (9) requires an unsupervised objective function $\mathcal{L}_{\text{tto}}$. We address this by proposing a novel and flexible self-supervision strategy where a test sample is augmented to generate a pseudo-target for optimization. Our approach operates fully in the DG setting and differs from existing studies that rely on delayed or partially observed test targets [41].

Self-Synthetic Sample Generation. The augmentation process is simple yet effective. The main strategy (illustrated in Figure 3(b)) is to prepend a segment of length h from the beginning of the input sequence $\mathbf{x}_{\text{test}}$ to create an augmented sequence $\mathbf{x}_{\text{test}}^{\text{aug}} \in \mathbb{R}^{(T+h)\times 1}$. We add Gaussian noise to the prepended segment to provide a regularizing signal. We avoided more complex time series augmentations to prevent distortion of trend and seasonal patterns and leave further exploration to future work. This augmented sequence naturally yields the pseudo-input and pseudo-target for our self-supervised task. Specifically, the last h steps of $\mathbf{x}_{\text{test}}^{\text{aug}}$ serve as the pseudo-target $\mathbf{y}'_{\text{test}} \in \mathbb{R}^{h\times 1}$, while the preceding portion is treated as the pseudo-input $\mathbf{x}'_{\text{test}} \in \mathbb{R}^{T\times 1}$. Alternatively, simpler methods, such as zero-padding or mean-padding, can be used.[3] For *any additional features*, we obtain $\mathbf{a}'_{\text{test}}$ by repeating categorical features or extending temporal features by inferring the previous time features based on the sequence rules (e.g., by day or hour).

Objective Function in TTA. The objective function for test-time optimization is defined as:

$$\mathcal{L}_{\text{tto}}(X_{\text{test}}) = -\log p_{\Theta}\big(\mathbf{y}'_{\text{test}} \mid \mathbf{x}'_{\text{test}}, \mathbf{a}'_{\text{test}}\big). \tag{10}$$

Our self-synthetic test-time optimization strategy fully uses the information available in each test sample, facilitating target-free optimization during inference. Moreover, the sample-wise synthetic optimization closely aligns with the actual prediction procedure in time series forecasting, such as streaming inference. The test-time optimized parameters are applied directly during prediction, ensuring consistency and more robust forecasting on test domains. Importantly, while our method prepends a past segment (similar to standard padding), its key contribution is using this synthetic input for sample-wise gradient-based adaptation, allowing the model to adjust latent representations for each test sample. The procedures are summarized in Algorithm 1.

5 Experiments

5.1 Experimental Setup

5.1.1 Datasets. We evaluate the time series DG task in six real-world datasets from diverse fields: *Web-traffic*, *Stock-volume*, *Favorita-cat*, *Favorita-store*, *Power-cons*, and *Exchange* [46]. We use the provided data and code to load and process the data for Stock-volume, Favorita-cat, and

[3] We did not apply techniques to ensure continuity between the prepended segment and $\mathbf{x}_{\text{test}}$. In practice, this does not noticeably affect adaptation loss or performance. Exploration of boundary-smoothing techniques is left for future work.

Table 2. Summary of Real-World Datasets Used in This Article

Dataset	Domains	#Time	Granularity	(T, h)	|a|	ADF	DTW
Web-traffic	9 projects	803	Day	(90, 30)	4	−2.13	218
Stock-volume	12 stocks	516	Day	(60, 14)	2	−3.44	130
Favorita-store	45 stores	306	Day	(60, 14)	2	−4.10	99
Favorita-cat	26 categories	306	Day	(60, 14)	2	−6.86	53
Power-cons	3 zones	244	Hour	(120, 24)	4	−4.82	1,073
Exchange	8 countries	7,587	Day	(120, 30)	4	−1.96	1,528

Favorita-store, which were introduced by prior work [20]. We provide a detailed description of the data processing for the newly introduced dataset below.

— *Web-traffic* contains daily traffic of different Wikipedia projects (e.g., en.wikipedia.org and fr.wikipedia.org).[4] We use data from 1 July 2015 to 15 April 2016 for training, 15 April 2016 to 1 July 2016 for validation, and the remainder until 10 September 2017 for testing. Missing values are filled with 0, and a simple pre-processing step scales the data by dividing all values by 1e7.
— *Power-cons* contains hourly power consumption data from three different distribution networks in Tetouan City, located in northern Morocco.[5] The training data span from 1 May 2017 to 1 August 2017, the validation data extend until 30 September 2017, and the remaining data up to 30 December 2017 are used for testing. No missing values were observed. We scale the values by dividing all data points by 1e4.
— *Exchange* [46] contains daily exchange rates for eight countries: Australia, Britain, Canada, Switzerland, China, Japan, New Zealand, and Singapore, spanning 1990–2016. We use data from 1 January 1990 to 31 December 2001 for training, extend the validation set until 31 December 2003, and assign the remaining data until 10 October 2010 as the test set. No missing values were reported, and no additional data processing was applied.

A dataset summary is provided in Table 2, where #Time denotes the total number of timestamps, (T, h) denotes the historical window T and the predicted sequence length h. $|\mathbf{a}|$ is the dimension of external features. To better quantify the challenges for DG, we include two additional attributes: stationarity and domain similarity. We quantify stationarity using the **augmented Dickey-Fuller (ADF)** test [55, 58], regardless of domain labels, where lower ADF values indicate higher stationarity. Domain similarity is quantified using **dynamic time warping (DTW)** [11] for all domain pairs, where lower DTW values denote higher similarity across domains. Accordingly, we consider datasets with both higher ADF and DTW values, such as Exchange, to be more challenging for DG.

Overall, these datasets cover a diverse set of domains (from 3 to 45), each with distinct temporal characteristics. They exhibit cross-domain variability, including differences in seasonality, trends, and dynamics, as reflected by the ADF and DTW statistics, making them well-suited for evaluating DG.

5.1.2 Evaluation Metrics. We use standard evaluation metrics to assess probabilistic forecasting performance, including both *range* and *point* accuracy [20, 69, 71].

For *range accuracy*, we use the normalized quantile loss including $Q(0.5)$ and $Q(\text{mean})$ [20, 69]:

$$Q(q) = \frac{\sum_{t=T+1}^{T+h} 2\left|(\mathrm{Y}_t - \hat{\mathrm{Y}}_t^q) \cdot (1_{\mathrm{Y}_t \le \hat{\mathrm{Y}}_t^q} - q)\right|}{\sum_{t=T+1}^{T+h} |\mathrm{Y}_t|},$$

[4] https://www.kaggle.com/competitions/web-traffic-time-series-forecasting/data.
[5] https://archive.ics.uci.edu/dataset/849/power+consumption+of+tetouan+city.

where $\hat{\mathbf{Y}}^q \in \mathbb{R}^{N\times h}$ is the prediction for quantile q. 1 is the indicator function. We use results when $q = 0.5$, denoted as $Q(0.5)$, and the mean quantile performance across nine quantiles $q = \{0.1, 0.2, \ldots, 0.9\}$, to which we refer as $Q(\text{mean})$.

For *point accuracy*, we report the **normalized root mean squared error (NRMSE)** and **symmetric mean absolute percentage error (sMAPE)** [3]:

$$sMAPE = \frac{1}{h}\sum_{t=T+1}^{T+h}\frac{2|\mathbf{Y}_t - \hat{\mathbf{Y}}_t|}{|\mathbf{Y}_t| + |\hat{\mathbf{Y}}_t| + \epsilon}, \quad NRMSE = \frac{\sqrt{\frac{1}{h}\sum_{t=T+1}^{T+h}(\mathbf{Y}_t - \hat{\mathbf{Y}}_t)^2}}{\frac{1}{h}\sum_{t=T+1}^{T+h}|\mathbf{Y}_t| + \epsilon},$$

where $\mathbf{Y}, \hat{\mathbf{Y}} \in \mathbb{R}^{N\times h}$ are the matrices of the actual and predicted forecasting sequences for N samples, and t is the forecasting step. The prediction $\hat{\mathbf{Y}}$ is obtained as the *median* forecast, i.e., the prediction at quantile $q = 0.5$, from the probabilistic output. $\epsilon = 10^{-6}$ prevents possible division by zero.

All evaluation scores are computed and averaged on all training/test domains, with lower scores indicating better performance.

5.1.3 Comparison Methods. We introduce all methods compared in our main experiments.

Backbones. Our proposed ALD is a model-agnostic framework. We select three different types of time series forecasting models as the backbone (i.e., the forecasting decoder). In this work, we focus on probabilistic forecasting and evaluate all backbones under a probabilistic forecasting setting. Following prior work [20, 71] and prioritizing both efficiency and performance in probabilistic forecasting, we use the RNN-based DeepAR [69], CNN-based WaveNet [61], and **Multilayer Perceptron (MLP)**-based DLinear [87].

Baselines. We comprehensively compare our method ALD with the following *model-agnostic distribution shift methods* applicable in time series forecasting:

— *Invariant Learning Approaches*. GroupDRO [68], a robust optimization method that minimizes worst-case group loss. DANN [26], an adversarial domain alignment framework.
— *Gradient-Based DG Method*. IDGM [70], which aligns gradients across domains to improve generalization.
— *Cross-Domain Regularization*. Cedar [20], a difficulty-aware cross-domain regularization method for DG in time series forecasting.
— *Temporal Distribution Shift Methods*. RevIN [42], a reversible instance normalization for reducing distribution shift. DishTS [23], an input- and output-space distribution disentangled normalization for handling non-stationarities. FAN [86], a frequency-domain adaptive normalization for non-stationary time series forecasting.[6]

We additionally include the following *time series forecasting models* as baselines:

— *Traditional Model*. SN that repeats the value from the same season in the last period.
— *Long-Term Forecasting Model*. Autoformer [83] that uses auto-correlation to capture periodic patterns and decomposition to model trend and seasonality.
— *VAE-Based Decomposition Model*. LaST [81] that learns to separate seasonal-trend latent factors via mutual information-regularized VAE.
— *LLM-Based Time Series Models*. GPT4TS [96] that fine-tunes later layers of GPT-2 on time series; TimeLLM [38] that uses frozen LLM with trainable prompts and "reprograms" time

[6]We do not apply Cedar and DANN to DLinear because its single-layer linear architecture lacks intermediate hidden features required for domain alignment and classification.

series for forecasting; and UniTime [53] that unifies multiple time series tasks with a shared temporal representation.[7]

— *Pre-Trained Time Series Foundation Models*. MOIRAI [82], a masked encoder-based universal time series forecasting transformer using patch-based tokenization, trained on a large-scale open time series archive. Chronos [2], a pre-trained probabilistic time series models that tokenizes time series values using scaling and quantization into a fixed vocabulary.[8]

5.1.4 Implementation Details. We implemented all models using Pytorch [62] 2.4.0 with CUDA 12.1 on NVIDIA TITAN Xp and RTX A6000. We use a simple data normalization block, reversible instance norm, to ensure stable training across domains with varying scales [42]. We use Glorot initialization [29] to initialize parameters and the Adam [43] optimizer. Batch size is 64, and dropout rate is 0.3. Except for LLM-based time series models, which are inherently for point forecasting, we perform probabilistic forecasting, where the output layers predict the location and scale parameters of a Normal distribution.

Structure of Decomposed VAEs. We used a simple 1D convolutional layer with average pooling as the kernel prediction network $f_k(\cdot)$ for the adaptive series decomposition. For VAE components, we employ an MLP layer followed by two linear readouts to estimate the mean and covariance of the latent distribution. Our experiments show that the MLP is more effective than an RNN-based structure [92]. We employ a decoder network that consists of a single linear layer.

Hyperparameter Settings. To balance efficiency and performance, we set the number of hidden layers for DeepAR and the kernel size for WaveNet according to established guidelines [71]. The hidden size is the same for all hidden layers. We perform a random search with 40 trials, tuning the learning rate from $\{0.0001, 0.0005, 0.001\}$, the dimension of hidden states from the set $\{32, 64\}$, the value of β from $\{1, 5, 10\}$. The moving average kernel candidates for adaptive series decomposition are $\{5, 9, 13, 17\}$, chosen to capture patterns at multiple temporal scales while maintaining computational efficiency. The same search strategy also applies to other baselines. For GPT4TS, we search the number of layers of GPT2-backbone from $\{3, 6\}$. For TimeLLM, we use LLaMA-7B as the LLM backbone. For time series foundation models, we utilize the pre-trained models, Chronos-T5-mini and MOIRAI-base.

The TTA learning rate η_{tta} is generally set to $0.1\times$ the optimal training learning rate determined during hyperparameter tuning. A lower rate of $0.01\times$ was used for the Favorita-cat and Favorita-store datasets, given their high domain similarity. As shown in the dataset summary (Table 2), these datasets have the lowest DTW values, indicating relatively small differences between domains, including between training and test domains. This observation can serve as a practical guideline for selecting TTA learning rates in future studies on datasets with similar characteristics. We avoid tuning the adaptation learning rate on the validation set, because it already serves for hyperparameter selection, which can lead to overfitting and overly small adaptation learning rates.

Model Selection. We strictly follow the standard DG settings from the benchmark paper by Gagnon-Audet et al. [25], where training and test domains are explicitly disjoint. To ensure robustness, we randomly shuffle training/test domain splits with a ratio of 8:2 and report average results, similar to a cross-validation strategy over domains [20]. To select the best-trained model for DG tasks, we use the training-domain validation method [25, 31].

[7]LLM-based models (GPT4TS, TimeLLM, and UniTime) are inherently for point forecasting, we use MSE as the forecasting loss and only report their *point accuracy* results.

[8]We test foundation models without fine-tuning, as we believe it is unnecessary and could risk overfitting given moderate sizes of our datasets. We omit TimesFM [19] from comparisons since we are unable to set up the required environments.

Table 3. Forecasting Performance in $Q(0.5)$ and $Q(\text{Mean})$ on Six Real-World Datasets

	Web-traffic		Stock-volume		Favorita-store		Favorita-cat		Power-cons		Exchange	
	$Q(0.5)$	$Q(\text{mean})$	$Q(0.5)$	$Q(\text{mean})$	$Q(0.5)$	$Q(\text{mean})$	$Q(0.5)$	$Q(\text{mean})$	$Q(0.5)$	$Q(\text{mean})$	$Q(0.5)$	$Q(\text{mean})$
SN	0.1420	0.1420	0.2568	0.2568	0.0211	0.0211	0.1107	0.1107	0.4815	0.4815	0.0193	0.0193
Autoformer	0.1950	0.1579	0.2963	0.2257	0.1548	0.2443	0.3396	0.2988	0.3245	0.2428	0.0509	0.0429
LaST	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
GPT4TS	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
TimeLLM	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
UniTime	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
MOIRAI[a]	0.7098	0.6532	0.2248	0.1847	0.0331	0.0264	0.1481	0.1216	0.3613	0.2868	0.3724	0.3710
Chronos[a]	0.1028	0.0813	0.1967	0.1599	0.0265	0.0206	0.1179	0.0922	0.0718	0.0600	0.0157	0.0125
DeepAR	0.1821	0.1374	0.2251	0.1763	0.0198	0.0160	0.0927	0.0750	0.3142	0.2276	0.0754	0.0699
+ DANN	0.2393	0.1885	0.2546	0.1982	0.0200	0.0162	0.1023	0.0854	0.1286	0.1041	0.0945	0.0837
+ GroupDRO	0.1937	0.1516	0.2482	0.1931	0.0196	0.0156	0.0950	0.0784	0.1343	0.1074	0.0754	0.0699
+ IDGM	0.1665	0.1323	0.2551	0.2043	0.0214	0.0174	0.0966	0.0797	0.2277	0.1861	0.0342	0.0270
+ Cedar	0.1821	0.1374	**0.2089**	**0.1657**	0.0196	0.0158	0.0918	0.0748	0.3142	0.2274	0.0695	0.0638
+ RevIN	0.1717	0.1348	0.2405	0.1899	0.0204	0.0162	**0.0814**	0.0722	0.1275	0.1029	0.0211	0.0174
+ DishTS	0.1505	0.1184	0.2270	0.1801	0.0200	0.0156	0.0872	**0.0720**	0.1290	0.0995	0.1446	0.0205
+ FAN	0.1399	0.1115	0.2183	0.1765	0.0321	0.1010	0.1093	0.1304	0.2103	0.1764	0.0321	0.1010
+ ALD	**0.1285**	**0.1040**	0.2352	0.1878	**0.0158**	**0.0154**	0.0833	0.0822	**0.0963**	**0.0764**	**0.0162**	**0.0131**
WaveNet	0.1388	0.1140	0.2870	0.2239	0.0165	0.0132	0.0874	**0.0728**	0.2757	0.2237	0.0226	0.0182
+ DANN	0.1974	0.1508	0.2481	0.1907	0.0193	0.0153	0.0941	0.0773	0.1594	0.1268	0.0266	0.0214
+ GroupDRO	0.1683	0.1337	0.2384	0.1869	0.0169	0.0134	0.0895	0.0749	0.2047	0.1614	0.0224	0.0182
+ IDGM	0.1886	0.1514	0.2447	0.1903	0.0208	0.0166	0.0940	0.0774	0.2911	0.2523	0.0231	0.0199
+ Cedar	0.3340	0.2743	0.2373	0.1833	0.0160	0.0129	0.0875	0.0740	0.2946	0.2232	0.0234	0.0194
+ RevIN	0.1600	0.1263	0.2397	0.1884	**0.0153**	**0.0125**	0.0848	0.0780	0.1236	0.1091	0.0215	0.0182
+ DishTS	0.1661	0.1403	0.2395	0.1866	0.0157	0.0126	0.0876	0.0735	0.1164	0.0975	0.0222	0.0182
+ FAN	0.1333	0.1062	0.2190	0.1759	0.0293	0.0942	0.1047	0.1084	0.2085	0.1803	0.0211	0.0174
+ ALD	**0.1210**	**0.0977**	**0.2113**	**0.1674**	0.0155	0.0132	**0.0837**	0.0778	**0.0866**	**0.0691**	**0.0159**	**0.0132**
DLinear	0.1703	0.1536	0.2156	0.1731	0.0156	0.0127	0.0875	0.0749	0.1650	0.1344	0.0163	0.0133
+ GroupDRO	0.1701	0.1535	0.2154	0.1731	0.0156	0.0127	0.0876	0.0750	0.1650	0.1344	0.0163	0.0133
+ IDGM	0.1427	0.1336	0.2163	0.1737	0.0152	0.0125	0.0878	0.0790	0.1907	0.1639	0.0156	0.0135
+ RevIN	0.1344	0.3013	0.2078	0.1732	0.0583	0.2963	0.1016	0.1838	0.1506	0.1613	0.0152	0.0145
+ DishTS	0.1425	0.1808	0.2199	0.1777	**0.0146**	**0.0118**	**0.0845**	**0.0725**	0.1904	0.1470	0.0158	0.0133
+ FAN	0.1441	0.1579	0.2414	0.1930	0.0266	0.1037	0.0994	0.1289	0.1742	0.1464	0.0171	0.0149
+ ALD	**0.1153**	**0.1199**	**0.2012**	**0.1629**	**0.0146**	0.0134	0.0856	0.0907	**0.0716**	**0.0599**	**0.0151**	**0.0124**

The best and second results per model are bolded and underlined. Overall best is highlighted in blue. Point forecasting models with no reported results are indicated by n/a.
[a]Denotes pre-trained large models.

5.2 Main Results

Tables 3 and 4 present the range accuracy ($Q(0.5)$ and $Q(\text{mean})$) and point accuracy (sMAPE, NRMSE) results across six real-world datasets, comparing our proposed method, ALD, applied to DeepAR, WaveNet, and DLinear, against other generalization techniques and time series forecasting models. LaST separates latent representations of trend and seasonality, using these representations to contribute individually to parts of the prediction. However, its ability to generalize to unseen domains is limited. This can be due to its mechanism (i.e., learning seasonal-trend representations within one domain) that does not effectively extract common structure shared across domains, limiting its ability to generalize to new domains. Our method ALD consistently improves backbones, especially in challenging scenarios, such as Web-traffic, Power-cons, and Exchange datasets, characterized by high instability and high cross-domain variance. ALD demonstrates superior performance over other methods, particularly when combined with DLinear, where it achieves the best overall results. This highlights that DLinear+ALD is a lightweight yet robust solution for DG in time series forecasting.

Table 4. Forecasting Performance in sMAPE and NRMSE on Six Real-World Datasets

	Web-traffic		Stock-volume		Favorita-store		Favorita-cat		Power-cons		Exchange	
	sMAPE	NRMSE	sMAPE	NRMSE	sMAPE	NRMSE	sMAPE	NRMSE	sMAPE	NRMSE	sMAPE	NRMSE
SN	0.1523	0.2059	0.2429	0.6310	0.0210	0.0280	0.2554	0.1709	0.5032	0.5486	0.0195	0.0256
Autoformer	0.2508	0.2861	0.4344	0.6381	0.1656	0.1845	0.3508	0.4966	0.5042	0.3996	0.0544	0.0792
LaST	0.1872	0.2394	0.5707	0.5250	0.0790	0.0982	0.2426	0.1948	0.4777	0.5000	0.1204	0.1214
GPT4TS	0.1780	0.3799	0.2027	0.5102	0.0185	0.0241	0.1905	0.1253	0.2676	0.3225	0.0142	0.0248
TimeLLM	0.1573	0.2303	0.1977	0.1173	0.0250	0.4888	0.1903	0.0310	0.2207	0.1285	0.0164	0.2685
UniTime	0.1450	0.2254	0.2134	0.5241	0.0292	0.0361	0.1967	0.1386	0.2932	0.3445	0.0166	0.0272
MOIRAI[a]	0.6476	0.8931	0.2183	0.7937	0.0651	0.0782	0.3682	0.2520	0.3774	0.4530	0.3737	0.3706
Chronos[a]	0.1094	0.1620	0.1973	0.4742	0.0264	0.0328	0.2723	0.1794	0.0827	0.1146	0.0141	0.0254
DeepAR	0.2412	0.2506	0.4139	0.4965	0.0197	0.0258	0.1978	0.1338	0.3327	0.3627	0.3431	0.0937
+ DANN	0.2990	0.3086	0.4840	0.5179	0.0200	0.0259	0.1983	0.1471	0.1469	0.1640	0.3686	0.1116
+ GroupDRO	0.1979	0.2642	0.3904	0.5265	0.0196	0.0254	0.1975	0.1390	0.1519	0.1694	0.3431	0.0937
+ IDGM	0.1737	0.2358	0.4812	0.5544	0.0213	0.0277	0.1955	0.1379	0.2522	0.2995	0.0478	0.0448
+ Cedar	0.2411	0.2505	0.2575	**0.4812**	0.0196	0.0251	0.1966	0.1332	0.3326	0.3623	0.3035	0.0863
+ RevIN	0.1705	0.2490	0.2728	0.5113	0.0203	0.0259	**0.1875**	**0.1204**	0.1420	0.1620	0.0190	0.0316
+ DishTS	0.1560	0.2151	**0.2147**	0.4978	0.0200	0.0254	0.1923	0.1299	0.1446	0.1631	0.1446	0.0356
+ FAN	0.1529	0.1962	0.3685	0.4933	0.0319	0.0397	0.2051	0.1519	0.2174	0.2768	0.0319	0.0397
+ ALD	**0.1421**	**0.1876**	0.2715	0.5046	**0.0157**	**0.0219**	0.1889	0.1245	**0.1084**	**0.1362**	**0.0147**	**0.0253**
WaveNet	0.1679	0.1924	0.6101	0.5775	0.0165	0.0224	0.1931	0.1301	0.3265	0.3489	0.0221	0.0333
+ DANN	0.2062	0.2616	0.4389	0.5074	0.0193	0.0253	0.1955	0.1385	0.1849	0.2036	0.0321	0.0362
+ GroupDRO	0.1975	0.2344	0.3004	0.5118	0.0169	0.0226	0.1932	0.1316	0.2497	0.2571	0.0208	0.0335
+ IDGM	0.1920	0.2700	0.3738	0.5134	0.0207	0.0271	0.2011	0.1357	0.3871	0.4088	0.0214	0.0341
+ Cedar	0.3894	0.4501	0.4785	0.4991	0.0160	0.0218	0.1946	0.1300	0.3741	0.3661	0.0227	0.0346
+ RevIN	0.1613	0.2312	0.2685	0.5064	**0.0153**	**0.0210**	0.1917	0.1252	0.1440	0.1609	0.0195	0.0324
+ DishTS	0.1757	0.2355	**0.2153**	0.5243	0.0157	0.0213	0.1911	0.1274	0.1322	0.1530	0.0217	0.0327
+ FAN	0.1418	0.1905	0.3619	0.4833	0.0294	0.0374	0.2038	0.1440	0.2647	0.2772	0.0304	0.0284
+ ALD	**0.1321**	**0.1809**	0.2222	**0.4739**	0.0155	0.0211	**0.1900**	**0.1243**	**0.0997**	**0.1199**	**0.0143**	**0.0250**
DLinear	0.1726	0.2434	0.2981	0.4953	0.0156	0.0211	0.1938	0.1267	0.1691	0.2212	0.0151	0.0255
+ GroupDRO	0.1725	0.2432	0.2985	0.4956	0.0156	0.0212	0.1938	0.1268	0.1691	0.2212	0.0151	0.0255
+ IDGM	0.1490	0.2004	0.4810	0.4780	0.0151	0.0207	**0.1926**	0.1265	0.1970	0.2593	0.0143	0.0247
+ RevIN	0.1441	0.1896	0.2480	0.4855	0.0593	0.1064	0.2120	0.1463	0.1526	0.2128	0.0139	0.0244
+ DishTS	0.1464	0.2052	0.2613	0.4900	0.0146	**0.0202**	0.1940	**0.1241**	0.2060	0.2329	0.0146	0.0252
+ FAN	0.1470	0.2373	0.3437	0.5393	0.0265	0.0334	0.2078	0.1375	0.1892	0.2213	0.0183	0.0254
+ ALD	**0.1252**	**0.1794**	**0.2225**	**0.4673**	**0.0145**	0.0203	0.1959	0.1267	**0.0823**	**0.0997**	**0.0135**	**0.0242**

The best and second results per model are bolded and underlined. Overall best is highlighted in blue.
[a]Denotes pre-trained large models.

For LLM-based methods, which are inherently designed for point forecasting, we report only their *point accuracy* results. As shown in Table 4, GPT4TS and TimeLLM perform well on the Stock-volume, Favorita-cat, and Exchange datasets but fail to outperform ALD. They also incur high time and space complexity. The time series foundation model, Chronos, has strong results on several datasets, likely benefits from much larger training data, whereas ALD strictly follows DG using no target domain data. The pre-trained model MOIRAI, instead, is limited on specific domain distributions without adaptation. This further motivates our work on effective DG and TTA techniques to handle diverse samples in unseen domains.

5.3 Ablation Results

We conduct ablation studies to evaluate different aspects of ALD. Unless stated otherwise, we use DeepAR and DLinear as backbones and focus on Web-traffic and Power-cons, which exhibit moderate stationarity (ADF) and domain similarity (DTW) (see Table 2).

5.3.1 Ablation of Decomposed VAE. To evaluate the effectiveness of our proposed decomposed VAE, we conducted an ablation study on three components: (i) replacing our adaptive series

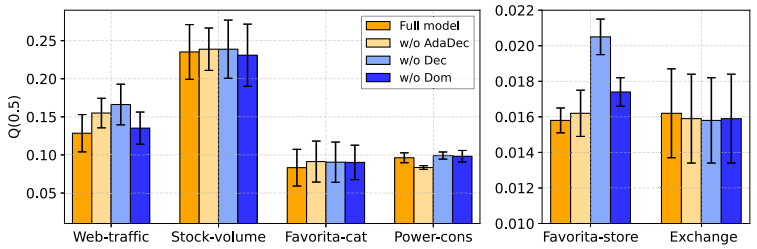


Fig. 4. Ablation study on the decomposed VAE using DeepAR. Datasets are grouped for better visualization.

decomposition with a static moving average method [83] (w/o AdaDec), (ii) removing the series decomposition entirely (w/o Dec), and (iii) excluding the domain identifier from the VAE decoder (w/o Dom). The results, presented in Figure 4, demonstrate that all three components positively contribute to the model's ability to forecast time series from unseen domains. Notably, the series decomposition (w/o Dec) leads to a more significant performance degradation than excluding the conditional domain information (w/o Dom). This suggests that the decomposition mechanism is critical for learning effective and informative latent variables. Additionally, our adaptive series decomposition method outperforms the static decomposition on four of the six datasets, confirming its effectiveness.

One exception is the Exchange dataset, where removing the series decomposition entirely (w/o Dec) achieves slightly better performance. This may be due to its unique domain characteristics; e.g., a high DTW indicates *low domain similarity*, making it difficult to learn seasonal and trend-cyclical components that generalize across domains. A similar observation holds for Power-cons, which also has high DTW and where adding the decomposition provides limited gains. This observation provides insights into dataset properties and potential ALD model adjustments. Nevertheless, the performance differences are marginal, and the full model still outperforms all baseline methods as shown in Tables 3 and 4. Overall, these results confirm the effectiveness of the decomposed VAE.

5.3.2 Ablation of Test-Time Adaptive Inference. We assess the effectiveness of our adaptive inference on different *adapted parameters*, *adaptation strategies*, and *synthetic sample strategies*.

In Table 5, we report the results of three settings of *parameter adaptation*: (i) the base setting, where *all* parameters are fine-tuned; (ii) *decoder only*, where only the forecasting decoder is trainable; and (iii) *none*, without TTA. The results indicate that fine-tuning all parameters yields more stable and improved performance. This explains the model's ability to capture domain shifts at both low and high levels. In addition, the significant gap between adaptation (i and ii) and no adaptation (iii) highlights the effectiveness of incorporating sample-specific knowledge for unseen domain samples.

Table 6 shows the results of different *adaptation strategies*: (i) *sample*-based method (base setting), which updates the model on individual samples; (ii) *online*-based method, which follows the sample-based approach but continuously maintains the updated model; (iii) *domain*-based method, which aggregates samples from the same domain and updates the model based on domain-wise average performance, assuming domain labels are available in the test set; and (iv) *random*-based method, which updates the model using randomly sampled batches of size 64, irrespective of domain labels. In most cases, the sample- or online-based approaches achieve better performance. This can be attributed to the varying characteristics of time series samples, even within a single domain. Thus, updating the model with a batch of samples may introduce additional bias and noise, which might hinder accurate predictions.

Table 5. Ablation Study Results in $Q(0.5)$ and sMAPE on Adapted *Parameters* during Test-Time Adaptive Inference

		Web-traffic		Power-cons	
	Parameters	$Q(0.5)$	sMAPE	$Q(0.5)$	sMAPE
DeepAR	All	**0.1285 0.025**	**0.1338 0.031**	**0.0963 0.007**	**0.1084 0.007**
	Decoder only	0.1360 0.023	0.1412 0.032	0.1244 0.021	0.1368 0.021
	None	0.1503 0.024	0.1553 0.038	0.2940 0.068	0.3138 0.074
DLinear	All	**0.1153 0.008**	**0.1252 0.026**	0.0716 0.002	0.0823 0.002
	Decoder only	0.1255 0.004	0.1331 0.024	**0.0702 0.001**	**0.0813 0.001**
	None	0.1418 0.019	0.1412 0.032	0.1142 0.010	0.1220 0.009

Best performances are bolded.

Table 6. Ablation Study Results in $Q(0.5)$ and sMAPE on *Adaptation* Methods during Test-Time Adaptive Inference

		Web-traffic		Power-cons	
	Adaptation	$Q(0.5)$	sMAPE	$Q(0.5)$	sMAPE
DeepAR	Sample	**0.1285 0.025**	**0.1338 0.031**	0.0963 0.007	0.1084 0.007
	Online	0.1285 0.025	0.1339 0.031	**0.0961 0.007**	**0.1081 0.006**
	Domain	0.1484 0.022	0.1532 0.036	0.2351 0.069	0.2476 0.064
	Random	0.1496 0.022	0.1544 0.037	0.2340 0.069	0.2466 0.064
DLinear	Sample	0.1153 0.008	0.1252 0.026	0.0716 0.002	0.0823 0.002
	Online	**0.1134 0.008**	**0.1241 0.029**	**0.0714 0.002**	**0.0821 0.002**
	Domain	0.1297 0.008	0.1355 0.026	0.0824 0.003	0.0958 0.004
	Random	0.1327 0.010	0.1382 0.025	0.0828 0.003	0.0962 0.004

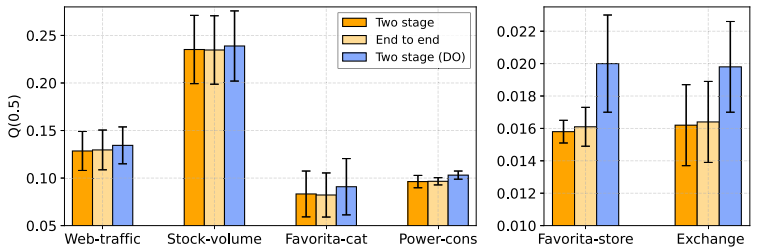
Best performances are bolded.

Table 7. Ablation Study Results in $Q(0.5)$ and sMAPE on Sample *Synthetic Strategies (i.e., Padding)* during Test-Time Adaptive Inference

		Web-traffic		Power-cons	
	Padding	$Q(0.5)$	sMAPE	$Q(0.5)$	sMAPE
DeepAR	Repeat	**0.1285 0.025**	0.1338 0.031	0.0963 0.007	0.1084 0.007
	Zero	0.1292 0.022	**0.1321 0.029**	**0.0872 0.004**	**0.0968 0.004**
	Mean	0.1311 0.023	0.1346 0.029	0.0991 0.005	0.1146 0.005
DLinear	Repeat	**0.1153 0.008**	**0.1252 0.026**	**0.0716 0.002**	**0.0823 0.002**
	Zero	0.1168 0.008	0.1258 0.025	0.0811 0.004	0.0951 0.007
	Mean	0.1172 0.008	0.1272 0.026	0.0823 0.004	0.0967 0.008

Best performances are bolded.

Table 7 presents the results of different *self-synthetic sample generation* methods, i.e., the default padding strategy (*repeat*), *zero* padding, and *mean* padding. The results show that repeat padding yields the best overall performance, though zero and mean padding also produce favorable outcomes. This suggests that using target-free test samples can be highly effective in helping the model capture the characteristics of new domain samples for forecasting. The flexibility to choose between these strategies allows for adaptation to different applications.



(a) Training strategy

Fig. 5. Model analysis of training time. Datasets are grouped for better visualization.

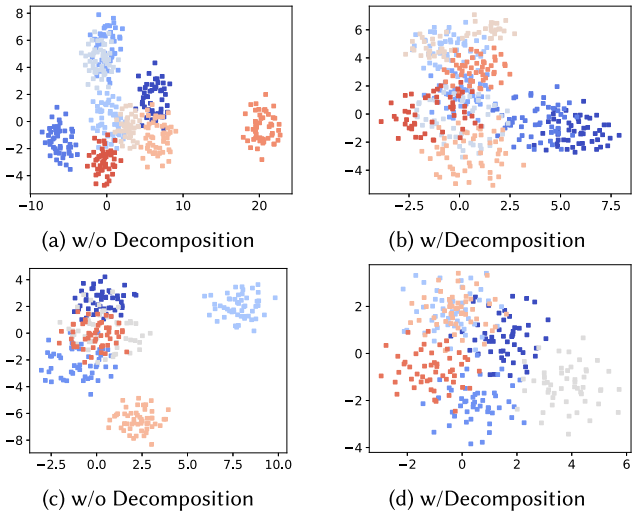
5.3.3 Ablation of Training Strategies. To evaluate our two-stage training strategy, we compare it against two alternatives: (i) a two-stage approach where only the forecasting decoder is trained in the second stage, denoted as Two-stage (DO) and (ii) a single-stage, end-to-end training procedure. The results, obtained using DeepAR and shown in Figure 5, demonstrate that our two-stage strategy outperforms both alternatives. This emphasizes the importance of a well-trained latent space, which has a positive impact on final forecasting results. The end-to-end approach's favorable performance relative to Two-stage (DO) further suggests that fixing latent variables during the decoder optimization stage can impede generalization.

5.4 Decomposed VAE Analysis

5.4.1 Domain Latent Space Analysis. We analyze the learned latent space across domains of ALD on the Favorita-store and Favorita-cat datasets, given their relatively higher number of domains, favorable for visualization. We use DLinear as the backbone. Figure 6 visualizes the latent variables of test domains with t-SNE [76], obtained from Equation (6), as well as the latent vectors from a variant model without time series decomposition. We observe that the latent variables with decomposition (b, d) show noticeable overlap in their distributions across domains, whereas those without decomposition are more separable (a, c). This shows that the decomposed VAE helps capture *general* characteristics across domains, e.g., similar trends or seasonality following Equation (6), thus enhancing DG.

5.4.2 Decomposed Latent Variables Analysis. ALD learns latent variables for the decomposed input sequence, i.e., trend-cyclical and seasonal components, using the standard Gaussian prior $\mathcal{N}(0, \boldsymbol{I})$ in the decomposed VAE. This choice is consistent with prior structured VAE work [34, 37, 44, 92], where separation arises from architectural inductive biases rather than different priors. In our case, the two encoders operate on *different frequency components* of the input (a smoothed low-frequency trend-cyclical part and a high-frequency seasonal part), and the reconstruction loss pushes the latent variables to specialize accordingly. The KL divergence regularization constrains each posterior from memorizing the input (i.e., overfitting the reconstruction), and together with the distinct inputs and reconstruction objectives, this encourages the two latent spaces to diverge [37].

Our experiments also show that the resulting latent variables enhance the generalization accuracy of time series forecasting models. Moreover, the learned latent variables are both meaningful and separable across components, without *posterior collapse* [80]. Figure 7 plots the learned latent variables in four test domains on the Favorita-store dataset. The t-SNE plots reveal distinct clusters, confirming that the latent representations effectively capture the unique characteristics of each component.



(a) w/o Decomposition (b) w/Decomposition

(c) w/o Decomposition (d) w/Decomposition

Fig. 6. t-SNE visualization of latent spaces of the test domains in the Favorita-store (a, b) and Favorita-cat (c, d) datasets. Different colors represent different domains.

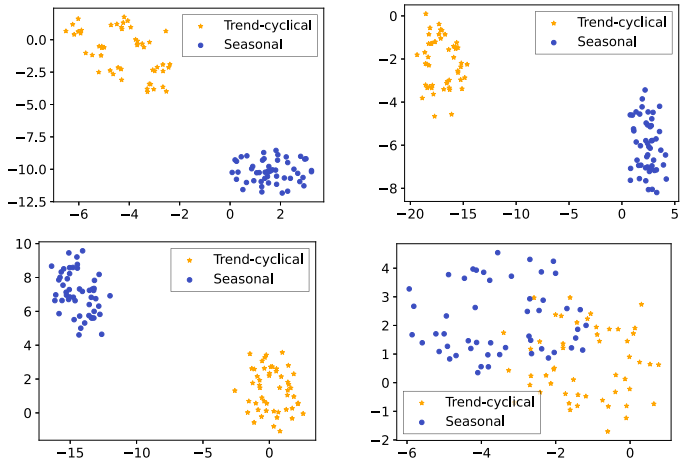


Fig. 7. t-SNE visualization of decomposed latent variables in individual test domains in the Favorita-store dataset.

5.4.3 VAE-Based Decomposition and Spectral Validation. Our decomposed VAE learns latent variables that represent the seasonal and trend-cyclical (hereafter referred to as "trend" for brevity) components of sequence data. We assess whether these reconstructed components preserve their intended frequency characteristics, as this reflects the *quality of the latent space*. Figure 8 visualizes representative reconstructions for training samples: $\hat{\mathbf{x}}_s$ exhibits a smooth periodic structure, while $\hat{\mathbf{x}}_t$ captures slower dynamics with residual cyclical fluctuations.

To quantify frequency separation, we compute *leakage*, defined as the proportion of energy in the "opposite" band (low-frequency energy in $\hat{\mathbf{x}}_s$ and high-frequency energy in $\hat{\mathbf{x}}_t$) using one-sided FFT power spectra. On the Web-traffic dataset with the DLinear backbone, the mean seasonal leakage is 0.0937, indicating that the seasonal branch captures predominantly high-frequency content

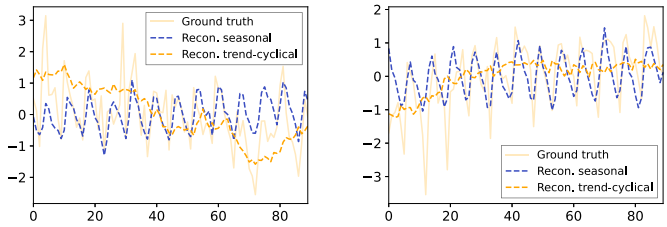


Fig. 8. The original training samples alongside the VAE-reconstructed components of the samples in the Web-traffic dataset.

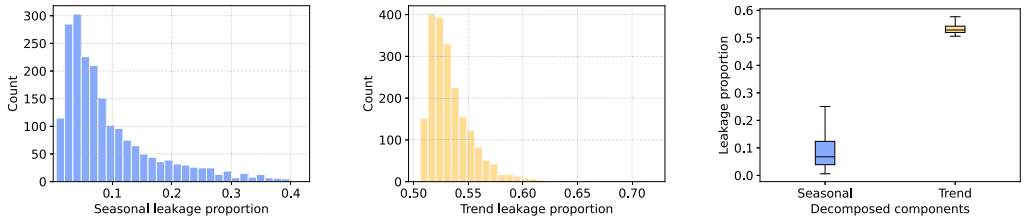


Fig. 9. Seasonal and trend leakage distributions for all training samples in the Web-traffic dataset. The right one is a combined view in a boxplot.

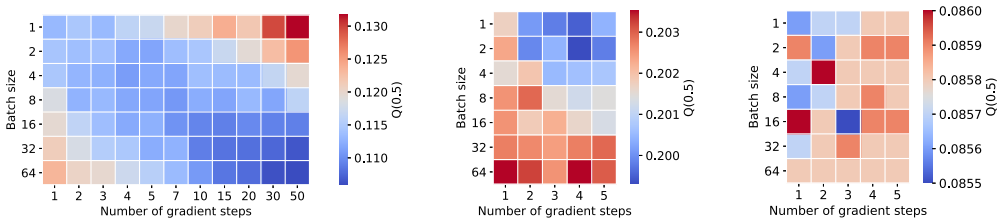
with minimal low-frequency contamination. The mean trend leakage is 0.5337, which is higher but consistent with our design choice for the trend branch to retain both low-frequency trends and slower cyclical variations. We show the leakage portion of seasonal and trend components in Figure 9. This design supports our gating mechanism (Equation (6)) by allowing the forecaster to balance long- and mid/high-frequency information underlying latent variables.

5.5 TTA Analysis

5.5.1 Sensitivity of Test-Time Adaptive Inference. We investigate the impact of *number of gradient steps* and *batch size* on our proposed TTA method's performance. The number of gradient steps and the batch size were both increased from one. Adaptation was performed without access to test domain labels. We conducted experiments on three datasets: Web-traffic, Stock-volume, and Favorita-cat datasets. The first two are more challenging, as detailed in Table 2. For the Web-traffic dataset (Figure 10(a)), a single gradient step with a batch size of one already yields strong performance. However, increasing both the batch size and the number of gradient steps together can further improve generalization. For the Stock-volume dataset (Figure 10(b)), slightly increasing the number of gradient steps can lead to improved performance. In contrast, on the Favorita-cat dataset (Figure 10(c)), larger batch sizes and an increased number of updates do not yield a stable gain in generalization performance. This instability likely arises because larger batches introduce conflicting patterns or noise, which hinders effective adaptation.

TTA in classification tasks is typically beneficial with larger batches because they provide a richer set of categorical information. In our time series regression setting, each sample is more self-contained due to the high variance of the sequence data, so larger batches may introduce conflicting patterns or noise, which hinders effective adaptation.

5.5.2 Standalone Effectiveness of TTA. Our test-time adaptive inference is model-agnostic and can be seamlessly applied to any forecasting backbone. To assess its standalone effectiveness, we apply TTA directly to forecasting backbones and present the results in Table 8. TTA alone substantially



(a) Web-traffic (b) Stock-volume (c) Favorita-cat

Fig. 10. Sensitivity of test-time adaptive inference. Darker blue indicates more accurate predictions.

Table 8. Forecasting Performance in $Q(0.5)$ and sMAPE on Six Real-World Datasets for Backbones with TTA-Only and ALD

	Web-traffic		Stock-volume		Favorita-store		Favorita-cat		Power-cons		Exchange	
	$Q(0.5)$	sMAPE	$Q(0.5)$	sMAPE	$Q(0.5)$	sMAPE	$Q(0.5)$	sMAPE	$Q(0.5)$	sMAPE	$Q(0.5)$	sMAPE
DeepAR	0.1821	0.2412	**0.2251**	0.4139	0.0198	0.0197	0.0927	0.1978	0.3142	0.3327	0.0754	0.3431
+ TTA	0.1284	0.1428	0.2437	0.3924	0.0226	0.0225	0.1031	0.2040	0.0998	0.1130	0.0343	0.0376
+ ALD	**0.1285**	**0.1421**	0.2352	**0.2715**	**0.0158**	**0.0157**	**0.0833**	**0.1889**	**0.0963**	**0.1084**	**0.0162**	**0.0147**
WaveNet	0.1388	0.1679	0.2870	0.6101	0.0165	0.0165	0.0874	0.1931	0.2757	0.3265	0.0226	0.0221
+ TTA	0.1175	0.1342	0.2372	0.2368	0.0224	0.0224	0.1007	0.1996	0.0886	0.1006	0.0224	0.0214
+ ALD	**0.1210**	**0.1321**	**0.2113**	**0.2222**	**0.0155**	**0.0155**	**0.0837**	**0.1900**	**0.0866**	**0.0997**	**0.0159**	**0.0143**
DLinear	0.1703	0.1726	0.2156	0.2981	0.0156	0.0156	0.0875	0.1938	0.1650	0.1691	0.0163	0.0151
+ TTA	0.1177	0.1314	0.2360	0.2239	0.0150	0.0150	0.0873	**0.1937**	0.0889	0.1009	0.0169	0.0157
+ ALD	**0.1153**	**0.1252**	**0.2012**	**0.2225**	**0.0146**	**0.0145**	**0.0856**	0.1959	**0.0716**	**0.0823**	**0.0151**	**0.0135**

The best results per model are bolded.

improves the generalization of base models in unseen domains, especially when their original performance is limited (e.g., DeepAR and WaveNet), and the overall results remain competitive with other generalization methods (Tables 3 and 4). Nevertheless, our full model—combining TTA with latent variable learning on decomposed components—consistently delivers the overall best results, highlighting the strength of our overall design.

5.6 Computational Analysis

5.6.1 Training Efficiency. We evaluate the training efficiency of ALD on the Web-traffic dataset in which we use relatively long input sequences. We measure the average training time per epoch for different DG methods based on DeepAR. We vary the model size (i.e., hidden size), as larger models are often preferred for handling more complex datasets. The runtime for both training and pre-training is shown in Figure 11. The training (Stage 2) of ALD consistently requires less total training time than Cedar, and in some cases, it takes less time than the backbone. Compared to the gradient matching baseline IDGM, which pre-trains the full DeepAR model via empirical risk minimization, the equivalent pre-training of ALD (Stage 1, which trains the lightweight decomposed VAEs) significantly reduces pre-training time. As the hidden size increases, the pre-training time for ALD scales linearly, while it increases more rapidly for IDGM. This indicates that it adds a limited total computational load to the backbone.

5.6.2 Inference Efficiency. In this section, we evaluate the test-time efficiency and inference latency introduced by TTA on the Web-traffic dataset. Figure 12(a) illustrates the effect of TTA

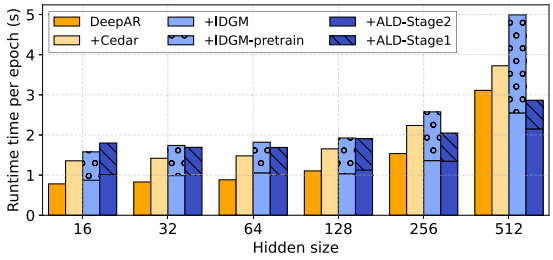
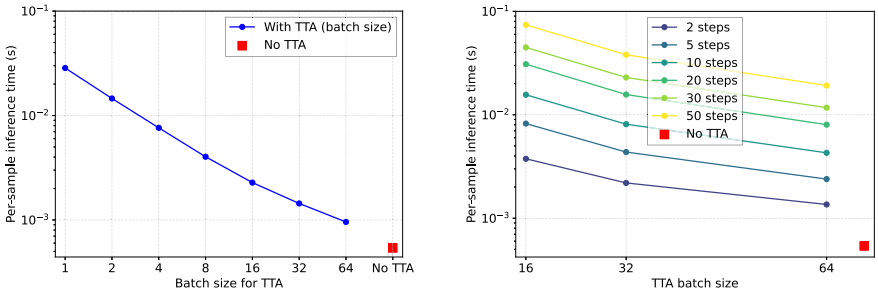


Fig. 11. Training efficiency analysis on the Web-traffic dataset.



(a) Effect of TTA batch size on per-sample inference time.

(b) Effect of TTA gradient update steps on per-sample inference time.

Fig. 12. Inference/test-time analysis on the Web-traffic dataset.

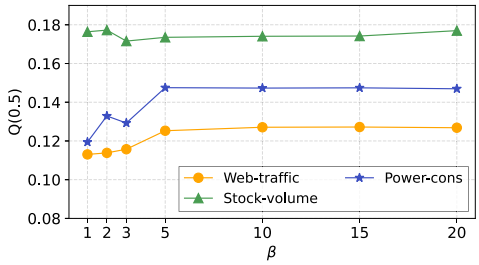


Fig. 13. Analysis of different β values in decomposed VAE on different datasets.

batch size on per-sample inference time when applying a single gradient update. As expected, sample-wise adaptation (batch size = 1) incurs the highest additional latency compared with No TTA (red square). Increasing the batch size for TTA largely reduces the per-sample inference time, approaching the No TTA baseline. Notably, even at small batch sizes, the per-sample inference time remains on the order of $1e^{-2}$ seconds, indicating that TTA does not substantially increase test-time cost; this effect becomes negligible in streaming test scenarios.

Building on our previous analysis (Section 5.5.1), which shows that increasing both the batch size and the number of gradient steps in TTA improve the generalization ability of ALD (Figure 10(a)), we further examine inference latency for larger batch sizes (16, 32, 64) under multiple TTA gradient steps. As shown in Figure 12(b), a batch size of 64 is relatively efficient, and increasing the number of gradient steps moderately raises the per-sample inference time. This highlights a tradeoff between improved generalization and additional inference cost.

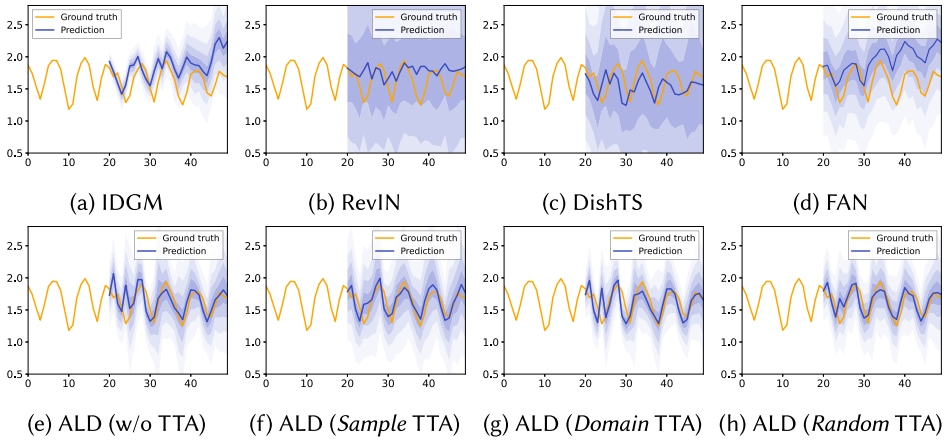


Fig. 14. Probabilistic forecasting visualizations for a test domain sample from the Web-traffic dataset, using DLinear enhanced with different generalization/normalization methods.

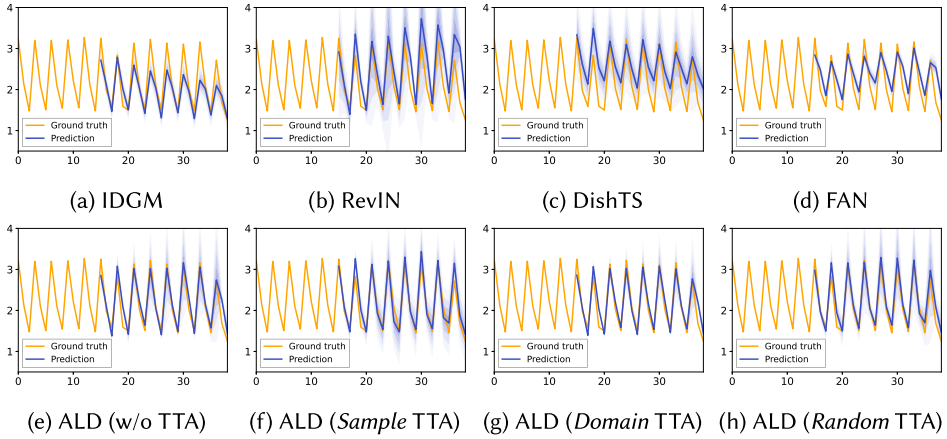


Fig. 15. Probabilistic forecasting visualizations for a test domain sample from the Power-cons dataset, using DLinear enhanced with different generalization/normalization methods.

5.7 Additional Model Analysis

5.7.1 Probabilistic Forecasting Visualization. Figures 14 and 15 show the probabilistic forecasting sequences of various methods based on DLinear on the Web-traffic and Power-cons datasets, respectively. The shaded areas represent different quantiles, ranging from 0.1 to 0.9. ALD demonstrates more accurate trend and seasonality estimates than other strong baselines on unseen data, even without adaptation (Figures 14(e) and 15(e)).

We test different adaptation strategies used in Table 6. Sample-based adaptation (Figures 14(f) and 15(f)) introduces a bit more uncertainty, as shown by the wider shaded areas across the quantiles. This outcome is reasonable, as forecasting for new, unseen samples is inherently challenging. This increased uncertainty may also stem from the noise introduced by self-synthetic samples or from overfitting to individual samples. However, this higher tolerance in probabilistic forecasting can be beneficial in real-world applications. Domain-based adaptation (Figures 14(g)

and 15(g)) and random-based adaptation (Figures 14(h) and 15(h)) also improve prediction accuracy, but their probabilistic forecasts are less precise and reliable compared to the sample-based approach.

5.7.2 Effect of β. In the decomposed VAE, we use β to regulate the learning of latent variables, encouraging each dimension of the latent vector to capture more distinct factors [12, 34]. To assess its impact, we analyze the forecasting performance of ALD using the DLinear model on the Web-traffic, Stock-volume, and Power-con datasets, presented in Figure 13. All other datasets have similar patterns. We observe that a smaller value of β generally yields robust forecasting performance. When β becomes large (e.g., $\beta > 5$), the performance mostly degrades, due to over-regularization hindering input sequence reconstruction, causing the latent vector to fail in capturing essential sequence information.

6 Conclusion

We propose a novel ALD framework for DG in time series forecasting. ALD consists of a decomposed VAE that adaptively mines decomposed latent variables of various series components to improve DG of time series forecasting models. It further enhances unseen domain predictions by leveraging the unique characteristics of individual test samples through an effective test-time adaptive inference strategy. Extensive experiments and analyses on real data validate the effectiveness of ALD. Importantly, ALD also offers the potential to enhance generalizability in spatial and spatiotemporal forecasting.

As for *limitations*, the inherent noise and complexity of real-world time series can impact the quality of learned representations, potentially affecting the model's ability to generalize to new domains. ALD may be less effective for highly irregular series or when trend and seasonal components are not well separated, as the decomposed VAE relies on this structure to learn informative latent variables. For instance, in the Exchange and Power-cons datasets, which exhibit high DTW values indicating low domain similarity, the decomposition provides limited gains (see Figure 4), suggesting that the model may struggle when domain differences are large or component patterns are difficult to disentangle. Additionally, the TTA may converge to local minima due to its reliance on partially true self-synthesized samples.

As for *future work*, we plan to evaluate ALD on chaotic or highly irregular time series, including scenarios with abrupt distribution shifts, to test its effectiveness on more challenging data. We will also explore alternative optimization strategies and advanced synthetic sample generation to improve the robustness of TTA. In addition, since ALD is model-agnostic and independent of the forecasting backbone, we plan to integrate it with sophisticated transformer-based architectures to assess its effectiveness under more computationally heavier models. Finally, we will study channel dependencies in multivariate time series and extend our approach to diverse data modalities to further enhance DG.

Reproducibility

The code and data used to produce the results in this article are available at https://github.com/songgaojundeng/TSDG-ALD.

Acknowledgments

We thank our reviewers for their insightful comments that helped to improve the article.

References

[1] Jameel Abdul Samadh, Mohammad Hanan Gani, Noor Hussein, Muhammad Uzair Khattak, Muhammad Muzammal Naseer, Fahad Shahbaz Khan, and Salman H. Khan. 2023. Align your prompts: Test-time prompting with distribution alignment for zero-shot generalization. In *Advances in Neural Information Processing Systems*, Vol. 36, 80396–80413.

[2] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. 2024. Chronos: Learning the language of time series. arXiv:2403.07815. Retrieved from https://arxiv.org/abs/2403.07815

[3] J. Scott Armstrong. 1978. *Long-Range Forecasting: From Crystal Ball to Computer*. Wiley.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473. Retrieved from https://arxiv.org/abs/1409.0473

[5] Guangji Bai, Chen Ling, and Liang Zhao. 2023. Temporal domain generalization with drift-aware dynamic neural networks. In *Proceedings of the 11th International Conference on Learning Representations*.

[6] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271. Retrieved from https://arxiv.org/abs/1803.01271

[7] Kasun Bandara, Peibei Shi, Christoph Bergmeir, Hansika Hewamalage, Quoc Tran, and Brian Seaman. 2019. Sales demand forecast in e-commerce using a long short-term memory neural network methodology. In *Proceedings of the International Conference on Neural Information Processing*. Springer, 462–474.

[8] Pratyay Banerjee, Tejas Gokhale, and Chitta Baral. 2021. Self-supervised test-time learning for reading comprehension. arXiv:2103.11263. Retrieved from https://arxiv.org/abs/2103.11263

[9] Sabyasachi Basu, Amarnath Mukherjee, and Steve Klivansky. 1996. Time series models for internet traffic. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM '96)*, Vol. 2. IEEE, 611–620.

[10] Michael A. Benjamin, Robert A. Rigby, and D. Mikis Stasinopoulos. 2003. Generalized autoregressive moving average models. *Journal of the American Statistical Association* 98, 461 (2003), 214–223.

[11] Donald J. Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, 359–370.

[12] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. 2018. Understanding disentangling in beta-VAE. arXiv:1804.03599. Retrieved from https://arxiv.org/abs/1804.03599

[13] Ruichu Cai, Jiawei Chen, Zijian Li, Wei Chen, Keli Zhang, Junjian Ye, Zhuozhang Li, Xiaoyan Yang, and Zhenjie Zhang. 2021. Time series domain adaptation via sparse associative structure alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35, 6859–6867.

[14] Zekun Cai, Guangji Bai, Renhe Jiang, Xuan Song, and Liang Zhao. 2024. Continuous temporal domain generalization. arXiv:2405.16075. Retrieved from https://arxiv.org/abs/2405.16075

[15] Jui-Sheng Chou and Duc-Son Tran. 2018. Forecasting energy consumption time series using machine learning techniques based on usage patterns of residential householders. *Energy* 165 (2018), 709–726.

[16] Panayiotis Christou, Shichu Chen, Xupeng Chen, and Parijat Dube. 2024. Test time learning for time series forecasting. arXiv:2409.14012. Retrieved from https://arxiv.org/abs/2409.14012

[17] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555. Retrieved from https://arxiv.org/abs/1412.3555

[18] Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. 1990. STL: A seasonal-trend decomposition. *Journal of Official Statistics* 6, 1 (1990), 3–73.

[19] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. 2023. A decoder-only foundation model for time-series forecasting. arXiv:2310.10688. Retrieved from https://arxiv.org/abs/2310.10688

[20] Songgaojun Deng, Olivier Sprangers, Ming Li, Sebastian Schelter, and Maarten de Rijke. 2024. Domain generalization in time series forecasting. *ACM Transactions on Knowledge Discovery from Data* 18, 5 (2024), 1–24.

[21] Yuntao Du, Jindong Wang, Wenjie Feng, Sinno Pan, Tao Qin, Renjun Xu, and Chongjun Wang. 2021. AdaRNN: Adaptive learning and forecasting of time series. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 402–411.

[22] Abhimanyu Dubey, Vignesh Ramanathan, Alex Pentland, and Dhruv Mahajan. 2021. Adaptive methods for real-world domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14340–14349.

[23] Wei Fan, Pengyang Wang, Dongkun Wang, Dongjie Wang, Yuanchun Zhou, and Yanjie Fu. 2023. Dish-TS: A general paradigm for alleviating distribution shift in time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37, 7522–7529.

[24] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R. Arabnia. 2021. A brief review of domain adaptation. In *Proceedings of the Advances in Data Science and Information Engineering (ICDATA '20 and IKE '20)*, 877–894.

[25] Jean-Christophe Gagnon-Audet, Kartik Ahuja, Mohammad-Javad Darvishi-Bayazi, Pooneh Mousavi, Guillaume Dumas, and Irina Rish. 2022. WOODS: Benchmarks for out-of-distribution generalization in time series. arXiv:2203.09978. Retrieved from https://arxiv.org/abs/2203.09978

[26] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research* 17, 1 (2016), 2096–2030.

[27] Jin Gao, Jialing Zhang, Xihui Liu, Trevor Darrell, Evan Shelhamer, and Dequan Wang. 2023. Back to the source: Diffusion-driven adaptation to test-time corruption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11786–11796.

[28] Everette S. Gardner, Jr. 2006. Exponential smoothing: The state of the art—Part II. *International Journal of Forecasting* 22, 4 (2006), 637–666.

[29] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*.

[30] Taesik Gong, Jongheon Jeong, Taewon Kim, Yewon Kim, Jinwoo Shin, and Sung-Ju Lee. 2022. Note: Robust continual test-time adaptation against temporal correlation. In *Advances in Neural Information Processing Systems*, Vol. 35, 27253–27266.

[31] Ishaan Gulrajani and David Lopez-Paz. 2020. In search of lost domain generalization. arXiv:2007.01434. Retrieved from https://arxiv.org/abs/2007.01434

[32] James D. Hamilton. 2020. *Time Series Analysis*. Princeton University Press.

[33] Huan He, Owen Queen, Teddy Koker, Consuelo Cuevas, Theodoros Tsiligkaridis, and Marinka Zitnik. 2023. Domain adaptation for time series under feature and label shifts. In *Proceedings of the International Conference on Machine Learning*. PMLR, 12746–12774.

[34] Irina Higgins, Loic Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *Proceedings of the International Conference on Learning Representations*.

[35] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[36] Yeping Hu, Xiaogang Jia, Masayoshi Tomizuka, and Wei Zhan. 2022. Causal-based time series domain generalization for vehicle intention prediction. In *Proceedings of the 2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 7806–7813.

[37] Maximilian Ilse, Jakub M. Tomczak, Christos Louizos, and Max Welling. 2020. Diva: Domain invariant variational autoencoders. In *Proceedings of the Medical Imaging with Deep Learning*. PMLR, 322–348.

[38] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. 2024. Time-LLM: Time series forecasting by reprogramming large language models. In *Proceedings of the 12th International Conference on Learning Representations*.

[39] Xiaoyong Jin, Youngsuk Park, Danielle Maddix, Hao Wang, and Yuyang Wang. 2022. Domain adaptation for time series forecasting via attention sharing. In *Proceedings of the International Conference on Machine Learning*. PMLR, 10280–10297.

[40] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, 30.

[41] HyunGi Kim, Siwon Kim, Jisoo Mok, and Sungroh Yoon. 2025. Battling the non-stationarity in time series forecasting via test-time adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence* 39 (2025), 17868–17876.

[42] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *Proceedings of the International Conference on Learning Representations*.

[43] Diederik P. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, Vol. 5.

[44] Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational Bayes. arXiv:1312.6114. Retrieved from https://arxiv.org/abs/1312.6114

[45] Bjoern Krollner, Bruce Vanstone, and Gavin Finnie. 2010. Financial time series forecasting with machine learning techniques: A survey. In *European Symposium on Artificial Neural Networks: Computational Intelligence and Machine Learning*, 25–30.

[46] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 95–104.

[47] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems*, Vol. 32.

[48] Jian Liang, Ran He, and Tieniu Tan. 2024. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision* 133 (2024), 1–34.

[49] Haoxin Liu, Harshavardhan Kamarthi, Lingkai Kong, Zhiyuan Zhao, Chao Zhang, and B. Aditya Prakash. 2024. Time-series forecasting for out-of-distribution generalization using invariant learning. arXiv:2406.09130. Retrieved from https://arxiv.org/abs/2406.09130

[50] Jiashuo Liu, Zheyan Shen, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. 2021. Towards out-of-distribution generalization: A survey. arXiv:2108.13624. Retrieved from https://arxiv.org/abs/2108.13624

[51] Jinxin Liu, Hongyin Zhang, Zifeng Zhuang, Yachen Kang, Donglin Wang, and Bin Wang. 2024. Design from policies: Conservative test-time adaptation for offline policy optimization. In *Advances in Neural Information Processing Systems*, Vol. 36.

[52] Qiao Liu and Hui Xue. 2021. Adversarial spectral kernel matching for unsupervised time series domain adaptation. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, 2744–2750.

[53] Xu Liu, Junfeng Hu, Yuan Li, Shizhe Diao, Yuxuan Liang, Bryan Hooi, and Roger Zimmermann. 2024. UniTime: A language-empowered unified model for domain time series forecasting. In *Proceedings of the ACM on Web Conference 2024*, 4095–4106.

[54] Xiaofeng Liu, Chaehwa Yoo, Fangxu Xing, Hyejin Oh, Georges El Fakhri, Je-Won Kang, and Jonghye Woo. 2022. Deep unsupervised domain adaptation: A review of recent advances and perspectives. *APSIPA Transactions on Signal and Information Processing* 11, 1 (2022), 1–51.

[55] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *Advances in Neural Information Processing Systems*, Vol. 35, 9881–9893.

[56] Wang Lu, Jindong Wang, Xinwei Sun, Yiqiang Chen, and Xing Xie. 2023. Out-of-distribution representation learning for time series classification. In *Proceedings of the 11th International Conference on Learning Representations*.

[57] Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. 2017. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, 5715–5725.

[58] Rizwan Mushtaq. 2011. Augmented dickey fuller test.

[59] Anshul Nasery, Soumyadeep Thakur, Vihari Piratla, Abir De, and Sunita Sarawagi. 2021. Training for the future: A simple gradient interpolation loss to generalize along time. In *Advances in Neural Information Processing Systems*.

[60] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A time series is worth 64 words: Long-term forecasting with transformers. In *Proceedings of the 11th International Conference on Learning Representations*.

[61] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A generative model for raw audio. arXiv:1609.03499. Retrieved from https://arxiv.org/abs/1609.03499

[62] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, Vol. 32.

[63] Fotios Petropoulos, Daniele Apiletti, Vassilios Assimakopoulos, Mohamed Zied Babai, Devon K. Barrow, Souhaib Ben Taieb, Christoph Bergmeir, Ricardo J. Bessa, Jakub Bijak, John E. Boylan, et al. 2022. Forecasting: Theory and practice. *International Journal of Forecasting* 38, 3 (2022), 705–871.

[64] Sanjay Purushotham, Wilka Carvalho, Tanachat Nilanon, and Yan Liu. 2017. Variational recurrent adversarial deep domain adaptation. In *Proceedings of the International Conference on Learning Representations*.

[65] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1, 8 (2019), 9.

[66] Mohamed Ragab, Zhenghua Chen, Wenyu Zhang, Emadeldeen Eldele, Min Wu, Chee-Keong Kwoh, and Xiaoli Li. 2022. Conditional contrastive domain generalization for fault diagnosis. *IEEE Transactions on Instrumentation and Measurement* 71 (2022), 1–12.

[67] Evgenia Rusak, Steffen Schneider, George Pachitariu, Luisa Eck, Peter Gehler, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. 2021. If your data distribution shifts, use self-learning. arXiv:2104.12928. Retrieved from https://arxiv.org/abs/2104.12928

[68] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. 2019. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. arXiv:1911.08731. Retrieved from https://arxiv.org/abs/1911.08731

[69] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.

[70] Yuge Shi, Jeffrey Seely, Philip H. S. Torr, N. Siddharth, Awni Hannun, Nicolas Usunier, and Gabriel Synnaeve. 2021. Gradient matching for domain generalization. arXiv:2104.09937. Retrieved from https://arxiv.org/abs/2104.09937

[71] Olivier Sprangers, Sebastian Schelter, and Maarten de Rijke. 2023. Parameter-efficient deep probabilistic forecasting. *International Journal of Forecasting* 39, 1 (2023), 332–345.

[72] Yongyi Su, Xun Xu, and Kui Jia. 2022. Revisiting realistic test-time training: Sequential inference and adaptation by anchored clustering. In *Advances in Neural Information Processing Systems*, Vol. 35, 17543–17555.

[73] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the International Conference on Machine Learning*. PMLR, 9229–9248.

[74] Mingtian Tan, Mike A. Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. 2024. Are language models actually useful for time series forecasting? arXiv:2406.16964. Retrieved from https://arxiv.org/abs/2406.16964

[75] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58, 1 (1996), 267–288.

[76] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008), 2579–2605.

[77] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, Vol. 30.

[78] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. 2020. Tent: Fully test-time adaptation by entropy minimization. arXiv:2006.10726. Retrieved from https://arxiv.org/abs/2006.10726

[79] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip S. Yu. 2022. Generalizing to unseen domains: A survey on domain generalization. arXiv:2103.03097. Retrieved from https://arxiv.org/abs/2103.03097

[80] Yixin Wang, David Blei, and John P. Cunningham. 2021. Posterior collapse and latent variable non-identifiability. In *Advances in Neural Information Processing Systems*, Vol. 34, 5443–5455.

[81] Zhiyuan Wang, Xovee Xu, Weifeng Zhang, Goce Trajcevski, Ting Zhong, and Fan Zhou. 2022. Learning latent seasonal-trend representations for time series forecasting. In *Advances in Neural Information Processing Systems*, Vol. 35, 38775–38787.

[82] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. 2024. Unified training of universal time series forecasting transformers. arXiv:2402.02592. Retrieved from https://arxiv.org/abs/2402.02592

[83] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, Vol. 34, 22419–22430.

[84] Xiaowei Xiang, Yang Liu, Gaoyun Fang, Jing Liu, and Mengyang Zhao. 2023. Two-stage alignments framework for unsupervised domain adaptation on time series data. *IEEE Signal Processing Letters* 30 (2023), 698–702.

[85] Zehao Xiao and Cees G. M. Snoek. 2024. Beyond model adaptation at test time: A survey. arXiv:2411.03687. Retrieved from https://arxiv.org/abs/2411.03687

[86] Weiwei Ye, Songgaojun Deng, Qiaosha Zou, and Ning Gui. 2024. Frequency adaptive normalization for non-stationary time series forecasting. In *Advances in Neural Information Processing Systems*.

[87] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37, 11121–11128.

[88] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. 2013. Domain adaptation under target and conditional shift. In *Proceedings of the International Conference on Machine Learning*. PMLR, 819–827.

[89] Wenyu Zhang, Mohamed Ragab, and Chuan-Sheng Foo. 2022. Domain generalization via selective consistency regularization for time series classification. In *Proceedings of the 2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, 2149–2156.

[90] Yifan Zhang, Xue Wang, Kexin Jin, Kun Yuan, Zhang Zhang, Liang Wang, Rong Jin, and Tieniu Tan. 2023. AdaNPC: Exploring non-parametric classifier for test-time adaptation. In *Proceedings of the International Conference on Machine Learning*. PMLR, 41647–41676.

[91] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. arXiv:2303.18223. Retrieved from https://arxiv.org/abs/2303.18223

[92] Yixiu Zhao and Scott Linderman. 2023. Revisiting structured variational autoencoders. *Proceedings of the International Conference on Machine Learning*. PMLR, 42046–42057.

[93] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence* 35 (2021), 11106–11115.

[94] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. 2021. Domain generalization with mixstyle. arXiv:2104.02008. Retrieved from https://arxiv.org/abs/2104.02008

[95] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proceedings of the International Conference on Machine Learning*. PMLR, 27268–27286.

[96] Tian Zhou, Peisong Niu, Liang Sun, and Rong Jin. 2023. One fits all: Power general time series analysis by pretrained LM. In *Advances in Neural Information Processing Systems*, Vol. 36, 43322–43355.

Received 25 August 2025; revised 1 April 2026; accepted 22 May 2026