Domain Generalization in Time Series Forecasting

SONGGAOJUN DENG, OLIVIER SPRANGERS, and MING LI, AIRLab, University of Amsterdam, Amsterdam, The Netherlands
SEBASTIAN SCHELTER and MAARTEN DE RIJKE, University of Amsterdam, Amsterdam, The Netherlands

Domain generalization aims to design models that can effectively generalize to unseen target domains by learning from observed source domains. Domain generalization poses a significant challenge for time series data, due to varying data distributions and temporal dependencies. Existing approaches to domain generalization are not designed for time series data, which often results in suboptimal or unstable performance when confronted with diverse temporal patterns and complex data characteristics. We propose a novel approach to tackle the problem of domain generalization in time series forecasting. We focus on a scenario where time series domains share certain common attributes and exhibit no abrupt distribution shifts. Our method revolves around the incorporation of a key regularization term into an existing time series forecasting model: *domain discrepancy regularization*. In this way, we aim to enforce consistent performance across different domains that exhibit distinct patterns. We calibrate the regularization term by investigating the performance within individual domains and propose the *domain discrepancy regularization with domain difficulty awareness*. We demonstrate the effectiveness of our method on multiple datasets, including synthetic and real-world time series datasets from diverse domains such as retail, transportation, and finance. Our method is compared against traditional methods, deep learning models, and domain generalization approaches to provide comprehensive insights into its performance. In these experiments, our method showcases superior performance, surpassing both the base model and competing domain generalization models across all datasets. Furthermore, our method is highly general and can be applied to various time series models.

CCS Concepts: • **Applied computing** → **Forecasting**; • **Mathematics of computing** → **Time series analysis**; • **Computing methodologies** → **Regularization;**

Additional Key Words and Phrases: Time series forecasting, domain generalization, regularization

ACM Reference Format:
Songgaojun Deng, Olivier Sprangers, Ming Li, Sebastian Schelter, and Maarten de Rijke. 2024. Domain Generalization in Time Series Forecasting. *ACM Trans. Knowl. Discov. Data.* 18, 5, Article 113 (February 2024), 24 pages. https://doi.org/10.1145/3643035

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, and 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).
Authors' addresses: S. Deng, O. Sprangers, and M. Li, AIRLab, University of Amsterdam, Science Park 904, 1098 XH, Amsterdam, The Netherlands; e-mails: s.deng@uva.nl, o.r.sprangers@uva.nl, m.li@uva.nl; S. Schelter and M. de Rijke, University of Amsterdam, Science Park 904, 1098 XH, Amsterdam, The Netherlands; e-mails: s.schelter@uva.nl, m.derijke@uva.nl.

ACM 1556-4681/2024/02-ART113
https://doi.org/10.1145/3643035

1 INTRODUCTION

Time series data are ubiquitous, and forecasting such data plays a crucial role in many applications such as financial forecasting [27, 31], meteorology prediction [19, 60], healthcare analysis [10], and demand estimation [11]. The goal of time series forecasting is to predict future values based on historical observations. A major challenge in time series forecasting is the presence of domain shifts, where the underlying data distribution may vary due to data collection from various sources, locations, or conditions. Traditional forecasting models trained on data from a specific context often struggle to generalize well to unseen data. This issue arises in scenarios such as demand forecasting for new products, where accurate predictions are essential for effective planning and decision-making.

Domain generalization seeks to address this challenge by developing models that have consistent performance across different domains [63]. Significant progress has been made in domain generalization, yet mainly in computer vision [22, 65] and natural language processing [7]. Many existing methods encounter challenges when applied to time series forecasting, primarily due to two factors: (1) They require categorical label information in their problem settings, making them better suited for classification tasks [7, 22, 65], and (2) the inherent complexity and dynamic nature of time series data introduce significant stochasticity, complicating generalization efforts [15]. Existing domain generalization approaches fall short in effectively addressing the underlying temporal dependencies and distribution shifts in time series data.

In this article, we propose an approach to tackle the problem of domain generalization in time series forecasting. Our method focuses on addressing the challenges posed by the diverse patterns and complex characteristics of time series data from different domains. We base our approach on carefully considered assumptions concerning the presence of common patterns across domains and restrictions on data shifts within each domain. We introduce two regularization terms that enhance a model's ability to generalize effectively in time series forecasting: a basic version that ensures consistent performance across various domains, named *Domain discrepancy regularization* (Section 3.2.1), and an extension that takes into account the difficulty of individual domains, named *Domain discrepancy regularization with domain difficulty awareness* (Section 3.2.2). These regularization terms control the model's learning and fitting process across diverse domains, ensuring consistent forecast performance and generalizability to unseen domains.

In summary, the main contributions of this work are:

— We introduce a novel domain generalization problem in the context of time series forecasting. We formalize the time series forecasting task under specific assumptions, laying the groundwork for further exploration (Section 3.1).
— We propose a novel regularization term that improves a forecasting model's generalization capabilities by regulating cross-domain performance differences weighted by domain discrepancies (Section 3.2.1).
— We present an extended version of the regularization term by incorporating a notion of domain difficulty awareness, where we assign less penalty to challenging domains, allowing the model to learn more complex patterns (Section 3.2.2).

We conduct extensive experiments on diverse synthetic and real-world time series datasets to demonstrate the effectiveness of our method (Section 4). Our method achieves higher accuracy in domain generalization tasks than existing approaches and has low training overhead, making it applicable to real-world scenarios on top of an existing forecasting model.

2 RELATED WORK

We survey work in time series forecasting and domain generalization on time series data.

2.1 Time Series Forecasting

Time series forecasting is a regression task [33, 48], distinct from classification tasks [68, 69] within the realm of predictive modeling. Time series forecasting finds applications in various domains, such as finance [27, 31], retail [11], and healthcare [10]. Traditional time series forecasting models, such as **autoregressive (AR)** [55], **autoregressive integrated moving average model (ARIMA)** [8], and exponential smoothing [18], use statistical methods to analyze historical data and make future predictions. The **k-nearest neighbors (kNN)**, traditionally developed for classification tasks [68–71], has also been explored to address time series forecasting challenges [40, 67]. For better modeling nonlinear relationships and capturing more complex temporal dependencies, machine learning (e.g., regression [55] and tree-based models [30]) and deep learning models have received increased attention. **Recurrent neural networks (RNNs)** and their variants, such as **long short-term memory (LSTM)** [25] and **gated recurrent units (GRUs)** [12], have demonstrated strong performance in capturing long-term dependencies and nonlinear patterns in time series data. **Temporal convolutional networks (TCNs)** [6] and WaveNet [44] have gained popularity for their fast training, which allows for easy parallelization. Attention-based models [4], such as transformer [58] and TransformerConv [36], have achieved state-of-the-art performance in many prediction tasks for sequential data. Recently, many researchers have explored graph neural networks [64] in spatial-temporal forecasting [23, 60] and learning dynamic relationships among potential factors of the predictions [14, 27].

Time series forecasting can be classified into point estimate and probabilistic forecasting [21, 38], which provide a single predicted value and a probability distribution, respectively. Probabilistic time series forecasting plays a critical role in decision-making due to its ability to quantify uncertainties [21]. In this work, we focus on probabilistic forecasting.

2.2 Domain Generalization on Time Series Data

Domain generalization (DG) [63] refers to the problem of learning a model that can generalize well to unseen target domains that differ from the training domains. DG may help to reduce labeling efforts, handle distribution shifts, and facilitate transferability to new tasks [63]. Existing approaches to DG can be classified into three main categories: (i) data manipulation, (ii) learning strategy, and (iii) representation learning. Data manipulation techniques typically enhance generalization through randomization [56], augmentation of input data [52, 59], or generation of diverse samples [46]. Learning strategy-based methods mainly focus on meta-learning [7, 34, 37] and gradient-based techniques [53]. Representation learning involves learning domain-invariant representations through techniques such as kernel method, adversarial training, feature alignment, or invariant risk minimization [2, 35]. Most existing research has focused on particular types of data such as images, texts, or observation data for reinforcement learning.

Recently, DG has been explored in time series *classification*. These studies highlight the use of class information to learn domain invariance using distribution matching [72], data augmentation [72], contrastive learning [26, 47], and adversarial learning [39]. An empirical framework for DG has been explored in the context of time series classification within the clinical domain [66]. Additionally, researchers have introduced various time series benchmarks covering a diverse range of data modalities, such as videos and brain recordings [16]. Yet, these benchmarks are highly biased towards classification tasks. There have also been notable contributions to temporal domain generalization [5, 43]: the ability of a model to generalize well across different time periods. There are a few studies in domain adaptation for time series forecasting [29] that assume that data from the targeted domain is partially seen during training. Due to the complexity and stochasticity of time series, there is limited research on DG for time series *forecasting*. This is where we contribute: methods that enable domain generalization in time series forecasting.

Table 1. Important Notations and Descriptions

Notation	Description
T	the number of timesteps in the history window
h	the forecasting horizon
N	the number of samples in a time series dataset
θ	the learnable parameters of a model
K	the total number of domains
M	the total number of training domains
$\mathbf{y}_{1:T}$	time series variables until time T
$\mathbf{a}_{1:T}$	exogenous attributes until time T
D^k, D^{k_1}, D^{k_2}	time series datasets for domain k, k_1, k_2, respectively
$\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}}$	datasets for training and test domains, respectively
$\mathcal{H}(\cdot)$	the high-level representation of time series from a domain
$d_{\mathcal{H}}(,)$	the distribution divergence between two domains
$\mathcal{L}_{\text{fcst}}(\cdot)$	the forecasting loss in a domain
$d_{\mathcal{L}_{\text{fcst}}}(,)$	the difference in time series forecasting performance between two domains
$Mean(\cdot), Std(\cdot)$	the mean and standard deviation of a set of values
$MMD(,)$	maximum mean discrepancy
$\omega(,)$	the scaling factor that modulates the penalty for regularizing two domains

3 METHODOLOGY

We formalize the domain generalization problem for time series forecasting and introduce our proposed method. The necessary mathematical notations are in Table 1.

3.1 Problem Formulation

We first introduce the problem of time series forecasting and extend it to domain generalization.

Time series forecasting. We denote time series variables as $\mathbf{y}_{1:T} = \{y_1, y_2, \ldots, y_T\}$, where y_t represents the value at time t (e.g., sales in retail), and T is the number of timesteps in the history window.[1] Usually, we assume the timestep t to be constant (e.g., a day or an hour). The goal of time series forecasting is to estimate the future values $\mathbf{y}_{T+1:T+h} = \{y_{T+1}, y_{T+2}, \ldots, y_{T+h}\}$, where h is the forecasting horizon. We focus on multi-step prediction with $h > 1$, because it offers more valuable insights by providing longer prediction horizons, which are more relevant and informative in real-world scenarios. A time series dataset with N samples can be denoted by $D = \{(\mathbf{y}_{i,1:T}, \mathbf{a}_{i,1:T}), \mathbf{y}_{i,T+1:T+h}\}_{i=1}^{N}$, where the variable $\mathbf{a}$ represents possible exogenous attributes (e.g., day of the week, categorical features). For simplicity, we write $D = \{X, Y\}$, where $X = \{\mathbf{y}_{i,1:T}, \mathbf{a}_{i,1:T}\}_{i=1}^{N}$, $Y = \{\mathbf{y}_{i,T+1:T+h}\}_{i=1}^{N}$. We are interested in modeling the conditional distribution:

$$\begin{aligned} P(Y \mid X; \theta) &= P(\mathbf{y}_{i,T+1:T+h} \mid \mathbf{y}_{i,1:T}, \mathbf{a}_{i,1:T}; \theta) \\ &= \prod_{t=T+1}^{T+h} P(y_{i,t} \mid \mathbf{y}_{i,1:t-1}, \mathbf{a}_{i,1:t-1}; \theta), \end{aligned} \tag{1}$$

where θ is the learnable parameters of a model. For probabilistic forecasting, we have $\theta = (\mu_\theta, \sigma_\theta)$ where $\mu_\theta, \sigma_\theta$ denotes the location and scale parameters of a distribution (e.g., a normal distribution) parameterized by θ.

[1]In this article, y_t denotes a scalar value, but it can also be extended to a vector.

Domain generalization for time series forecasting. We extend the concept of time series forecasting to the scenario where the model needs to generalize well across multiple domains. We consider $\mathcal{D} = \{D^1, D^2, \ldots, D^K\}$ as the set of K domains, where each domain D^k consists of a time series dataset $D^k = \{X^k, Y^k\}$.

In the domain generalization problem, we assume to have access to M training domains $\mathcal{D}_{\text{train}} = \{D^k\}_{k=1}^M$ where $M < K$. The domain generalization task is to learn a time series forecaster $F : X \longrightarrow Y$ that generalizes well to unseen test domains $\mathcal{D}_{\text{test}} = \{D^k\}_{k=M+1}^K$ that cannot be accessed during training.

Data assumptions. Given the inherent complexity of real-world time series data, we limit the scope of domain generalization for time series forecasting. We employ the following assumptions regarding the characteristics of time series data within and across domains. These assumptions are widely used or have similar implications in general studies on domain generalization [2, 15, 39, 63].

Assumption 1 (Common underlying patterns). *There exist common underlying patterns among different domains, despite their individual idiosyncrasies. The dissimilarity between the joint distributions for two domains falls within a range defined by a lower bound ϵ_l and an upper bound ϵ_u. This range captures the extent of variation allowed between the shared patterns across different domains. ϵ_l should also be sufficiently large to prevent identical data across domains. Mathematically it can be expressed as: $\epsilon_l \leq |P^{k_1}(X, Y) - P^{k_2}(X, Y)| \leq \epsilon_u, \forall 1 \leq k_1 \neq k_2 \leq K, 0 < \epsilon_l \leq \epsilon_u$.*

Assumption 1 imposes constraints on the common patterns/invariance that can be leveraged to improve domain generalization. In practice, we can leverage prior knowledge or domain expertise to define the domains of interest that align with the assumed common patterns. For instance, retail, meteorology, and environmental factors-related data often show recurring seasonal patterns. To quantitatively measure this assumption, metrics such as the Pearson correlation coefficient or Dynamic Time Warping [42] can be employed for comparing time series across different domains.

Assumption 2 (No abrupt distribution shifts). *There are no sudden or abrupt distribution shifts within each domain of the time series data, while gradual changes may be present. Sudden changes imply large uncertainty in the unseen time series domain and raise concerns about the efficacy of developing generalization methods. Suppose $\Delta D^k(t) = |P^k(X_t, Y_t) - P^k(X_{t-1}, Y_{t-1})|$ denotes the distribution shift indicator for domain k at time t. We expect that $|\Delta D^k(t)|$ remains within the bounds of a potential threshold ϵ_s for all timesteps, for each domain, i.e., $\forall 1 < t \leq T, 1 \leq k \leq K$.*

Assumption 2 focuses on scenarios where the within-domain data distributions maintain relative stability over time. This assumption is well-suited to certain fields such as meteorology, transportation, and environmental monitoring where data tend to exhibit gradual changes rather than abrupt shifts. However, there are fields where abrupt distribution shifts are common, such as financial markets, natural disaster data, and social media activity, which may not conform to this assumption. Nevertheless, in such cases, it is still possible to adapt to this assumption by limiting the length of the time sequence within each domain, thereby ensuring that within-domain distributions remain acceptably stable. To measure this, distribution shift detection methods can be applied, such as comparing statistical properties of various time series data and using visualization techniques.[2]

[2]Assumption 2 shares similarities with scenarios addressed in related works [15, 39]. These works acknowledge the existence of multiple latent distributions within a time series and aim to first identify time series segments/domains that exhibit substantial distribution shifts. In contrast, our work operates under the assumption that these segments already exist and do not require explicit detection.

These data assumptions offer a practical starting point for devising effective domain generalization methods in time series forecasting.[3] Nonetheless, we acknowledge the difficulties associated with assessing the degree to which real-world datasets conform to our assumptions, e.g., defining precise thresholds and comparing. These challenges arise because time series data are inherently complex and exhibit diverse characteristics across different domains. We believe this requires further breaking down problems in various application areas, and we leave the theoretical in-depth studies for future work.

3.2 Proposed Method

Next, we present our proposed method, domain generalization in time series forecasting using **c**ross-domain r**e**gularizations with **d**ifficulty **a**wa**r**eness (**Cedar**). It consists of two novel regularization terms: (1) *domain discrepancy regularization*, which ensures consistent performance across various domains with distinct patterns, and (2) *domain discrepancy regularization with domain difficulty awareness*, an extension of the domain discrepancy regularization by considering the performance within each domain, accounting for the difficulty in training different domains.

3.2.1 Domain Discrepancy Regularization. In domain generalization, where information about the target domains is unknown, our objective is to learn a generalized model that exhibits consistent and stable performance across diverse temporal patterns in different domains. To achieve this, we propose a *domain discrepancy regularization* to prevent severe overfitting in all seen source domains, ensuring the robustness and generalization capability of the model when applied to new domains. It builds on the insight that dissimilar domains should not exhibit significant variations in forecasting performance. This regularization term is straightforward, as it calculates the difference of the forecasting performance between domain pairs, weighted by the discrepancy of the respective domain pair. We express the regularization term $\mathcal{R}_{DD}$ as follows:

$$\mathcal{R}_{DD} = \sum_{k_1, k_2}^{M} d_{\mathcal{H}}\left(D^{k_1}, D^{k_2}\right) \cdot d_{\mathcal{L}_{\text{fcst}}}\left(D^{k_1}, D^{k_2}\right). \tag{2}$$

We use $d_{\mathcal{H}}(,)$ to represent the distribution divergence between two domains. We use **maximum mean discrepancy (MMD)** as the difference metric in our experiments due to its easy implementation, widespread popularity, and kernel-based theoretical foundation supporting its use to capture complex relationships. Other distance metrics can also be applied (e.g., Euclidean distance and KL divergence). MMD has been used in distribution matching regularization in domain adaptation [15, 61] and generalization [72]. The definition of $d_{\mathcal{H}}(,)$ is:

$$d_{\mathcal{H}}\left(D^{k_1}, D^{k_2}\right) = MMD\left(\mathcal{H}(D^{k_1}), \mathcal{H}(D^{k_2})\right), \tag{3}$$

where $\mathcal{H}(D^k)$ represents the high-level representation of time series from domain k, which is learned from a forecasting model (e.g., hidden states of an RNN or convolutional vectors of a CNN) and evaluated on a batch of samples. Such a high-level representation captures temporal dependencies inherent in time sequence data. We compute the mean value of these representations across all samples in a batch that belongs to the specific domain.

The second term $d_{\mathcal{L}_{\text{fcst}}}(,)$ quantifies the difference in time series forecasting performance between two domains, computed by the Euclidean distance of the average loss values for batches of

[3]Note that this work differs from temporal domain generalization [5], which deals with time series data that exhibits changing patterns over time, necessitating adaptation of the model for future time segments. This work focuses on developing a single global model intended for universal applicability across all domains.

samples in each respective domain. Formally, we express the calculation as:

$$d_{\mathcal{L}_{\text{fcst}}}\big(D^{k_1}, D^{k_2}\big) = \big|Mean\big(\mathcal{L}_{\text{fcst}}(D^{k_1})\big) - Mean\big(\mathcal{L}_{\text{fcst}}(D^{k_2})\big)\big|^p. \tag{4}$$

Here, $\mathcal{L}_{\text{fcst}}(D^k)$ denotes the forecasting loss (e.g., Gaussian negative log-likelihood or L2 loss) of the batch samples in domain k; $|\cdot|$ denotes the absolute operation, and the parameter p can be set to 1 or 2. When $p = 2$, it implies a higher penalty for large differences in mean losses between domains. We can use matrix operations to efficiently calculate the multiplication of the two terms across all domain pairs.

By minimizing the regularization term (Equation (2)), we aim to prevent significant disparities in forecasting performance between different domains, which in turn can hinder the model's ability to effectively generalize to new patterns. Our approach differs from existing methods primarily tailored for classification tasks [1, 61, 72]. Unlike these approaches, which concentrate on aligning distributions within the same class across different domains to capture domain invariance, our method operates without class/label information. Class/label information provides a more straightforward way to group similar samples in different domains. Our focus is on promoting consistency in the model's predictive capacities across diverse domains while penalizing substantial differences in forecasting performance.

3.2.2 Domain Discrepancy Regularization with Domain Difficulty Awareness. In the regularization term $\mathcal{R}_{DD}$ (Equation (2)), we use the mean value of loss for each domain to gauge its performance in the current model, and the difference in mean loss is used to evaluate the performance disparity. However, time series data often exhibits irregular patterns or contains abnormal points and outliers, which may lead to inaccuracies in assessing the true performance difference based solely on mean loss, e.g., large stock price fluctuations can occur in financial markets, leading to inaccuracies in assessing the true performance of time segments based solely on their mean values.

Starting from this motivation, we adjust the penalty to account for the difficulty of the domains. If a domain exhibits higher variance in its loss values, then it implies greater challenges in training. Thus, we consider applying a milder penalty to that domain to give the model more flexibility in learning from that domain's data. By reducing the penalty for difficult domains, we allow the model to concentrate more on adapting to the complex aspects of those domains, which may lead to improved generalization performance.

To incorporate domain difficulty into the pairwise regularization term $\mathcal{R}_{DD}$ (Equation (2)), we propose a simple approach based on the variance that takes into account both domains in a pair:

$$\mathcal{R}_{DDD} = \sum_{k_1, k_2}^{M} d_{\mathcal{H}}\big(D^{k_1}, D^{k_2}\big) \cdot d_{\mathcal{L}_{\text{fcst}}}\big(D^{k_1}, D^{k_2}\big) \cdot \omega(D^{k_1}, D^{k_2}). \tag{5}$$

Here, $\omega(D^{k_1}, D^{k_2})$ is a scaling factor that modulates the penalty based on the difficulty of each domain in a pair, defined as follows:

$$\omega(D^{k_1}, D^{k_2}) = \frac{1}{Std\big(\mathcal{L}_{\text{fcst}}(D^{k_1})\big) + Std\big(\mathcal{L}_{\text{fcst}}(D^{k_2})\big) + \varepsilon}, \tag{6}$$

where $Std\big(\mathcal{L}_{\text{fcst}}(D^k)\big)$ denotes the standard deviation of the losses of batch samples in domain k. If any domain in the pair exhibits poor performance (with a large standard deviation of loss values), then we consider the domain discrepancy regularization to be less reliable. We introduce $\varepsilon = 1$ to prevent very small standard deviations from causing very large values or undefined values (e.g., 0) of ω, and the maximum value of ω is 1. There are other methods that can be used to quantify the difficulty of a domain, such as expert knowledge, exogenous feature analysis, or more advanced domain-specific metrics. We leave this to future work.

Table 2. Summary of Synthetic Datasets

Dataset	**Attributes**			**#Domains**	**#Points**
	Period	**Trend**	**Noise**		
NT-P		✓	✓	30	500
PT-N	✓	✓		30	500
PN-T	✓		✓	30	500
T-PN		✓		30	500

(✓) indicates common attributes across domains.

Training and optimization. Given a time series forecaster F and the proposed regularization terms for domain generalization, we train the forecaster F by minimizing the following total loss:

$$\mathcal{L} = \mathcal{L}_{\text{fcst}} + \gamma \cdot [\mathcal{R}_{DD} \text{ or } \mathcal{R}_{DDD}], \tag{7}$$

where $\mathcal{L}_{\text{fcst}}$ is the empirical loss of the base model F, which uses **empirical risk minimization (ERM)** [57] to minimize the average errors on training domains. It places equal emphasis on enhancing sample-level performance across all training domains. γ is a hyperparameter for balancing the contributions of cross-domain regularization and the empirical loss. The selection of the two regularization terms can be guided by prior knowledge or experiments that provide valuable insights into the difficulty of training each domain, e.g., a large variance in performance of the base model on individual domains would indicate greater challenges in training, and in such cases, $\mathcal{R}_{DDD}$ can be a better choice.

4 EXPERIMENTS

4.1 Experimental Settings

Datasets. We evaluate the performance of our proposed method on both synthetic and real-world time series datasets.

Synthetic data. To assess the efficacy of **Cedar** in achieving generalization across diverse scenarios, we construct multiple synthetic time series datasets that manifest distinct patterns of invariance across domains; see Table 2 for a summary. These datasets satisfy the data assumptions from Section 3.1. We assume that there are no abrupt distribution shifts within each domain of time series, and all domains share a common characteristic, i.e., periodicity. Periodicity is a prevalent pattern in various real-world time series data, such as sales data in retail and temperature fluctuations. We use the sinusoidal function to generate periodic signals and apply Gaussian noise to those periodic signals. We use no trend (i.e., horizontal trend) when it is a common attribute. We use the previous 270 timestamps for training, the subsequent 90 timestamps for validation, and the remaining timestamps for testing. No exogenous attributes are used.

Real-world data. We also assess the domain generalization ability of **Cedar** using real-world time series data, focusing on retail, transportation, and finance. The dataset summary is in Table 3.

— *Retail.* We use Favorita [41], which is a Kaggle dataset that contains five years (2013–2017) of daily sales for store-product combinations taken from a retail chain. We construct two datasets based on Favorita, namely, **Favorita-cat** and **Favorita-store**. In Favorita-cat, we focus on the category-level sales in the same store, treating each category as a separate domain. Favorita-store comprises the time series data for a single category (Grocery I) across multiple stores. We use data from the year 2015. The training data span from 2015-03-01 to 2015-06-30. The validation samples are from 2015-07-01 and 2015-8-15. The remaining data are used for testing. The sale variables are log-transformed.

Table 3. Summary of Real-world Datasets

Dataset	Domains	#Points	Granularity	(T, h)
Favorita-cat	26 categories	306	day	(60,14)
Favorita-store	45 stores	306	day	(60,14)
US-traffic	19 stations	244	day	(28,7)
Stock-volume	12 stocks	516	day	(60,14)

(T, h) denotes the sizes of the historical and prediction windows.

— *Transportation.* We use the U.S. Traffic Volume Data, which is openly available on the official website of U.S. Department of Transportation.[4] The traffic volume data are collected by state highway agencies. We use data in California from 2022-04 to 2022-11. The traffic volume is aggregated on a daily basis, considering both directions of travel, and the final values are divided by 1,000 to reduce the range of values. We use traffic volume data from 2022-04-01 to 2022-07-15 for training, data from 2017-07-16 to 2017-08-20 for validation, and the rest for testing.

— *Finance.* We use Stock Exchange Data, which contains daily price data for indexes tracking stock exchanges collected from Yahoo! Finance.[5] We use the stock trading volume as the target variable, representing the number of shares of security traded between its daily open and close. The training data span from 2020-01-01 to 2020-08-30, the validation data extends up to 2020-11-30, and the remaining data up to 2021-05-31 are for testing. Since some stocks have large averages, we used a simple method for normalization, i.e., dividing all trading volumes by 1e7.

Data assumption validation. We assess the alignment of real-world datasets with our assumptions primarily through prior knowledge and visualization methods. For instance, retail and transportation datasets demonstrate consistent seasonal patterns, and stock volume data accounts for overall market influence (Assumption 1). To mitigate abrupt distribution shifts (Assumption 2), we restrict the time sequence length for each domain and use category-level sales for retail, daily traffic volume, and daily stock volume data.

Exogenous attributes. We add numerical covariates consisting of time indicators (e.g., day of the week) to real-world datasets. The time indicators are represented by two Fourier terms [28] to represent the periodic nature of the time [54].

Data Visualization. We visualize some datasets in Figure 1 to illustrate the disparity in patterns across domains. For the synthetic dataset PN-T, time series from different domains exhibit varying trends while sharing the same period and Gaussian noise applied to the periodic signals. For real-world datasets, we can observe that the data have diverse cross-domain and within-domain patterns but roughly follow a pattern similar to the synthetic dataset.

Methods used for comparison. To evaluate the domain generalization performance of **Cedar**, we consider several state-of-the-art and popular domain generalization methods that are modality-agnostic (i.e., can be applied to time series data.) and are compatible with forecasting tasks as follows:

— **DANN** [17] is an adversarial learning method that learns features that are not capable of discriminating between training and test domains.

[4]https://www.fhwa.dot.gov/policyinformation/tables/tmasdata/
[5]https://www.kaggle.com/datasets/mattiuzc/stock-exchange-data?select=indexProcessed.csv

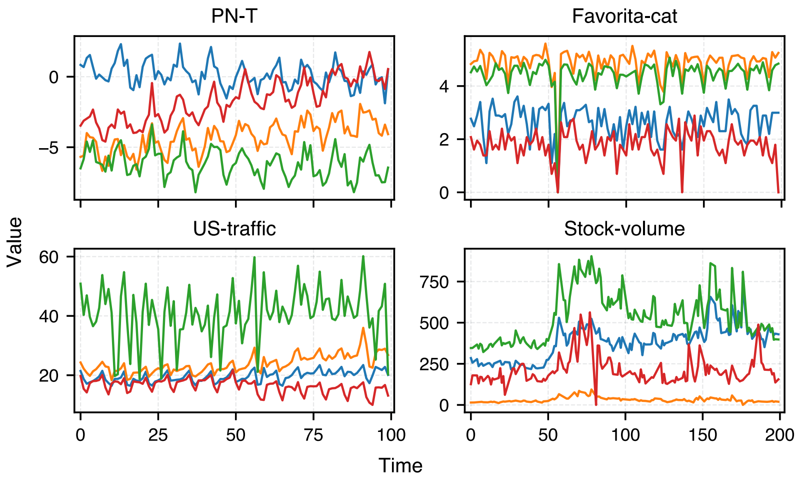


Fig. 1. Data visualization on partial domain data in PN-T, Favorita-cat, US-traffic, and Stock-volume datasets. Each time sequence represents a domain.

— **GroupDRO** [49] is the Group Distributionally Robust Optimization approach that aims to improve the robustness of models in the presence of domain shifts or changes in the data distribution.
— **MLDG** [34] is a meta-learning algorithm for domain generalization. The method trains for domain generalization by meta-optimization on simulated train/test splits with domain-shift.
— **IDGM** [53] is a gradient matching approach that learns invariance by maximizing the inner product between gradients from different domains.
— **wERM** is a weighted empirical risk minimization method adapted from the time series prediction model under distribution shift using differentiable forgetting [9]. **wERM-exp** and **wERM-mix** are variants using two forgetting mechanisms, corresponding to exponential decay and a mixture of various functional forms of decay, respectively.[6]
— **MMD** [61, 62] is a distribution matching method that matches the distributions between representations of data of two domains.

Cedar and the above domain generalization methods can be applied to different base models that forecast time series. Specifically, we consider two popular and representative time series models as the base model[7]:

— **DeepAR** [50] is an RNN-based probabilistic forecasting model.
— **WaveNet** [44] is a CNN-based forecasting model.

Apart from the base model with domain generalization methods, we also consider the following two types of methods as our baselines:

— Traditional time series models: **Seasonal Naive (SN)** and **Exponential Smoothing (ES)**.[8]

[6]This approach is not applicable when the prediction model is a neural network. We used the proposed objective function but adapted the bi-level optimization used in the original experiments to grid search for the optimal hyperparameter.
[7]The MMD calculation is performed on the last hidden state of all RNN layers. For IDGM, we use the first-order version proposed in the original paper.
[8]For SN, we take the observation from the last period as our forecast. We only compare ES on real-world datasets and two synthetic datasets (PT-N and PN-T), since they exhibit the same seasonality (e.g., weekly) across all domains.

— Latest deep learning models: The variational recurrent autoencoder **VRNN** [13] that learns latent variables that capture temporal dependencies and an adaptive time series model, **AdaRNN** [15], which can be fit to our domain generalization for time series forecasting.[9]

Evaluation metrics. We employ both the point accuracy metric and range accuracy metric to evaluate the probabilistic forecasting performance, following prior studies [50, 54]. For point accuracy, we use the **normalized root mean squared error (NRMSE)** and **symmetric mean absolute percentage error (sMAPE)** [3]:

$$NRMSE = \sqrt{\frac{1}{h}\sum_{t=T+1}^{T+h}(y_t - \hat{y}_t)^2} \Big/ \left(\frac{1}{h}\sum_{t=T+1}^{T+h}|\hat{y}_t|\right), \tag{8}$$

$$sMAPE = \frac{1}{h}\sum_{t=T+1}^{T+h}\frac{2|y_t - \hat{y}_t|}{|y_t| + |\hat{y}_t|}. \tag{9}$$

For range accuracy, we use the normalized quantile loss function [50]:

$$Q(q) = \sum_{i}^{N}\sum_{t=T+1}^{T+h} 2\left|(y_{i,t} - \hat{y}^q_{i,t}) \cdot (\mathbb{1}_{y_{i,t} \leq \hat{y}^q_{i,t}} - q)\right| \Big/ \sum_{i}^{N}\sum_{t=T+1}^{T+h}|y_{i,t}|, \tag{10}$$

where q is the quantile value, $\hat{y}^q_t$ is the prediction for quantile q. $\mathbb{1}$ is the indicator function. We report the scores when $q = 0.5$, denoted by Q(0.5), and the mean quantile performance over the nine quantiles in the range $q = \{0.1, 0.2, \ldots, 0.9\}$, denoted by Q(mean). The evaluation scores are computed across all training/test domains.

Implementation details. We implemented, trained, and evaluated all methods using PyTorch [45] 1.7.1 with CUDA 10.2 on TITAN Xp. For all datasets, we use the scaling mechanism following prior studies [50, 54]. All parameters are initialized with Glorot initialization [20] and trained using the Adam [32] optimizer; the dropout rate is 0.3. The learning rate is searched from $\{0.0005, 0.001, 0.005\}$. The batch size is 64 for all models and datasets. The hidden state size is set to be consistent across layers for all models searched from $\{16, 32, 64\}$. We adopt specific configurations for the number of hidden layers and the kernel size of the convolution operation based on prior work [54]. For RNN-based models (DeepAR and VRNN), the number of hidden layers is 3. For CNN-based models (e.g., WaveNet), the number of hidden layers is 5 and the kernel size is 9. For the GroupDRO, MLDG baseline models, we adopt the hyperparameter selection suggestions from previous work [24]. For wERM, we use the hyperparameter initialization method introduced in the original code [9]. For models that employ **maximum mean discrepancy (MMD)** as the discrepancy measure, we utilize the squared linear MMD [51] due to its efficiency and effectiveness. The coefficient multiplied to MMD is searched from $\{10^i\}_{i=-7}^{0}$. For AdaRNN, the number of RNN layers is 2 following the original paper's suggestion. The MMD coefficient has to be tuned with other parameters, and we set its range to $\{0.001, 0.0001\}$ to control the search space.

For **Cedar**, we grid-search the hyperparameter γ applied to $\mathcal{L}_{DD}$ or $\mathcal{L}_{DDD}$ from the range $\{10^i\}_{i=-7}^{0}$ and p in Equation (4) from $\{1, 2\}$. We do not tune γ in conjunction with other hyperparameters, such as the learning rate. We directly utilize the optimal settings of the other hyperparameters learned from the base model. It allows us to focus on the impact of γ on the model's performance, without introducing additional variations from tuning other hyperparameters.

[9]We did not report the performance of recent approaches and **CCDG** [47] and **Diversify** [39], because they are developed for classification tasks and their adaptation to a forecasting task is not possible without significant alterations to their formulations.

Model selection. We use 80% of domains for training and 20% for testing and adopt the training-domain validation approach [24].[10] We partition the available domains into training and test domains. Within the training domain, we further split the data into training and validation subsets in chronological order. Subsequently, we train models using the training subsets and choose the model that has the lowest loss on validation subsets. We run each experiment for 150 epochs with an early stopping criterion of 10 epochs. After we select the best model parameters based on the predefined criterion, we perform the evaluation on 5 random seeds for all models and report the average results. For each seed, we shuffle the training and test domains, ensuring that the model's generalization performance is assessed under varying conditions.

4.2 Results on Synthetic Data

Table 4 presents the results of the normalized quantile loss metrics Q(0.5) and Q(mean) on the four synthetic datasets. Table 5 shows the results of point accuracy metrics NRMSE and sRMSE. We apply **Cedar** to DeepAR and WaveNet and compare both with traditional time series models and deep learning models.

Among the traditional methods (SN and ES), ES performs well on datasets with fixed seasonality (PT-N and PN-T) but it falls behind some of the other methods. For deep learning baselines, the latent variable model and AdaRNN show poor performance and exhibit instability in different cases. This might be due to the complexity of the model structures, making it difficult to learn diverse time series patterns. The original paper of AdaRNN reports that the model's performance significantly drops when the number of time series periods/domains increases beyond 10, which explains its performance in our experiments. We also notice that its training time becomes extremely long when the number of domains is large (e.g., 30).

For domain generalization methods, we notice that widely used domain generalization methods such as GroupDRO and MLDG do not yield good performance in time series forecasting tasks. This can be attributed to their limited ability to account for certain inherent characteristics within time series data. Baselines DANN, IDGM, and MMD consistently deliver better results. The methods designed for distribution shifts, i.e., wERM-exp and wERM-mix, also perform well. **Cedar** and the variant model **Cedar**(-D) achieve favorable and stable performance compared to others when applied to both DeepAR and WaveNet. We see the performance of **Cedar** based on DeepAR sometimes surpasses that of WaveNet and vice versa. The effectiveness of the regularization is influenced by the temporal representation learned by the base model. The MMD method can be regarded as a naive variant of our approach, as it simply makes all domain representations similar (removing $d_{\mathcal{L}_{\text{fcst}}}$ in Equation (2)). Its results are inferior to ours, indicating the efficacy of leveraging cross-domain forecasting performance to enhance domain generalization. For the two proposed regularization terms, we observe that in most cases (except on PN-T and T-PN), **Cedar** consistently outperforms **Cedar**(-D) or performs on par with **Cedar**(-D). Hence, for the patterns observed in synthetic datasets, considering the prediction performance within individual domains is effective when applying cross-domain regularization.

4.3 Results on Real-world Data

Tables 6 and 7 list the results on the four real-world datasets. The traditional models achieve very good performance on Favorita-cat and US-traffic datasets, which can be attributed to the clear seasonality (weekly) and lower fluctuation in the category-level retail sales and daily traffic volume

[10]We did not allocate a specific percentage of domains as validation domains due to the inherent discrepancies between domains. It is worth noting that a model's performance on a subset of validation domains does not necessarily ensure similar performance on test domains.

Table 4. Forecasting Results of *Range Accuracy Metrics* on Synthetic Datasets

	NT-P		**PT-N**		**PN-T**		**T-PN**	
	Q(0.5)	Q(mean)	Q(0.5)	Q(mean)	Q(0.5)	Q(mean)	Q(0.5)	Q(mean)
SN	–	–	0.8673 (0.115)	0.8673 (0.115)	0.0949 (0.015)	0.0949 (0.015)	–	–
ES	1.1569 (0.034)	0.8907 (0.026)	0.4068 (0.074)	0.3230 (0.058)	0.0419 (0.006)	0.0335 (0.005)	0.5758 (0.046)	0.4482 (0.036)
VRNN	0.9683 (0.039)	0.7730 (0.027)	0.5707 (0.079)	0.4545 (0.071)	0.0521 (0.006)	0.0415 (0.005)	1.0595 (0.129)	0.8567 (0.111)
AdaRNN	0.6524 (0.012)	0.5200 (0.009)	0.8004 (0.124)	0.6186 (0.100)	0.5281 (0.036)	0.3715 (0.014)	7.3983 (13.40)	4.9784 (8.836)
DeepAR	0.6273 (0.027)	0.4879 (0.022)	0.4245 (0.072)	0.3412 (0.062)	0.0401 (0.007)	0.0314 (0.005)	0.5085 (0.109)	0.3985 (0.085)
-DANN	0.6234 (0.025)	0.4861 (0.021)	0.5173 (0.096)	0.4621 (0.104)	0.0473 (0.018)	0.0379 (0.014)	1.5007 (1.015)	6.9220 (7.889)
-GroupDRO	0.7717 (0.011)	0.5982 (0.008)	3.3673 (5.127)	2.5735 (3.410)	0.7112 (0.851)	0.5195 (0.529)	5.0898 (5.672)	3.6056 (3.770)
-MLDG	0.7873 (0.117)	0.6062 (0.082)	0.6204 (0.078)	0.5251 (0.074)	0.0788 (0.010)	0.0832 (0.015)	0.6288 (0.119)	0.5759 (0.131)
-IDGM	0.6201 (0.015)	0.4814 (0.012)	0.4154 (0.085)	0.3307 (0.068)	0.0388 (0.005)	0.0306 (0.004)	0.5494 (0.087)	0.5878 (0.232)
-wERM-exp	0.6229 (0.014)	0.4832 (0.011)	0.4037 (0.081)	0.3242 (0.068)	0.0394 (0.003)	0.0306 (0.003)	0.5086 (0.107)	0.3980 (0.084)
-wERM-mix	0.6229 (0.014)	0.4832 (0.011)	0.4054 (0.078)	0.3214 (0.063)	0.0396 (0.003)	0.0307 (0.003)	0.5034 (0.105)	0.3979 (0.082)
-MMD	0.6203 (0.031)	0.4831 (0.027)	0.4298 (0.090)	0.3408 (0.070)	0.0389 (0.005)	0.0305 (0.004)	0.5101 (0.096)	0.4027 (0.076)
-Cedar(-D)	***0.6143 (0.013)**	***0.4769 (0.011)**	*0.4005 (0.067)	*0.3174 (0.054)	*0.0379 (0.004)$^\diamond$	*0.0297 (0.003)$^\diamond$	***0.4942 (0.111)**$^\diamond$	***0.3872 (0.089)**$^\diamond$
-Cedar	***0.6143 (0.013)**	***0.4769 (0.011)**	***0.3843 (0.071)**$^\diamond$	***0.3055 (0.059)**$^\diamond$	*0.0389 (0.005)	*0.0307 (0.004)	0.5020 (0.109)	0.3942 (0.085)
WaveNet	0.8671 (0.044)	0.6641 (0.042)	0.4538 (0.096)	0.3577 (0.076)	0.0390 (0.006)	0.0302 (0.004)	0.5114 (0.109)	0.4136 (0.091)
-DANN	0.6863 (0.078)	0.5417 (0.071)	0.5108 (0.166)	0.4562 (0.199)	0.0378 (0.005)	0.0294 (0.004)	0.5341 (0.128)	0.4461 (0.110)
-GroupDRO	0.7288 (0.054)	0.5742 (0.050)	0.6443 (0.189)	0.5223 (0.132)	0.0395 (0.006)	0.0329 (0.004)	1.0071 (0.705)	0.8691 (0.447)
-MLDG	0.7397 (0.092)	0.5792 (0.077)	0.5056 (0.093)	0.4476 (0.112)	0.0711 (0.008)	0.0689 (0.009)	0.6055 (0.075)	0.4907 (0.062)
-IDGM	0.8738 (0.043)	0.6800 (0.040)	0.4154 (0.085)	0.3307 (0.068)	0.0388 (0.005)	0.0306 (0.004)	0.6463 (0.263)	0.5533 (0.243)
-wERM-exp	0.8303 (0.066)	0.7612 (0.067)	0.4044 (0.086)	0.3337 (0.069)	0.0385 (0.005)	0.0298 (0.004)	0.5118 (0.118)	0.4208 (0.096)
-wERM-mix	0.8228 (0.045)	0.7532 (0.047)	0.4035 (0.077)	0.3325 (0.062)	0.0391 (0.005)	0.0303 (0.004)	0.5218 (0.116)	0.4323 (0.097)
-MMD	0.6855 (0.070)	0.5414 (0.063)	0.4020 (0.086)	0.3245 (0.071)	0.0377 (0.005)	0.0291 (0.004)	0.5094 (0.106)	0.4109 (0.089)
-Cedar(-D)	*0.6829 (0.070)	*0.5378 (0.063)	*0.4042 (0.088)	*0.3283 (0.071)	***0.0373 (0.005)**	***0.0289 (0.004)**	0.5162 (0.104)	0.4191 (0.083)
-Cedar	*0.6817 (0.070)$^\diamond$	*0.5372 (0.063)$^\diamond$	*0.4014 (0.087)	*0.3254 (0.066)	***0.0373 (0.005)**	***0.0289 (0.004)**	0.5018 (0.114)	0.4058 (0.091)

Bolded values are the best scores for a column. Underlining indicates the method achieves the best performance when the base model is DeepAR or WaveNet. * indicates that our methods are significantly better than the base model, and $\diamond$ indicates the method is significantly better than the second-best baseline given the base model (paired t-test based on Equation (10) when the $\sum_i^N$ and denominator term are removed, p-value < 0.05).

Table 5. Forecasting Results of *Point Accuracy Metrics* on Synthetic Datasets

	NT-P		**PT-N**		**PN-T**		**T-PN**	
	NRMSE	sMAPE	NRMSE	sMAPE	NRMSE	sMAPE	NRMSE	sMAPE
SN	–	–	1.0735 (0.135)	1.0212 (0.168)	0.1126 (0.018)	0.1672 (0.066)	–	–
ES	1.4183 (0.043)	1.5571 (0.043)	0.6576 (0.094)	0.5774 (0.117)	0.0523 (0.008)	0.0837 (0.042)	0.7285 (0.077)	0.9815 (0.098)
VRNN	1.1891 (0.041)	1.1822 (0.027)	0.7670 (0.111)	0.8734 (0.190)	0.0641 (0.007)	0.0981 (0.048)	1.3524 (0.151)	1.7646 (0.152)
AdaRNN	0.8191 (0.016)	0.9300 (0.022)	1.0814 (0.234)	1.0116 (0.137)	0.5910 (0.035)	0.7507 (0.062)	7.9452 (14.14)	1.1955 (0.368)
DeepAR	0.7938 (0.036)	0.8475 (0.025)	0.6598 (0.076)	0.6425 (0.157)	0.0501 (0.009)	0.0783 (0.045)	0.6706 (0.146)	0.8484 (0.129)
-DANN	0.7902 (0.034)	0.8427 (0.028)	0.7191 (0.102)	0.7987 (0.237)	0.0589 (0.022)	0.0855 (0.037)	1.9563 (1.381)	1.3360 (0.214)
-GroupDRO	0.9638 (0.016)	1.0822 (0.029)	3.8473 (5.660)	1.3250 (0.367)	0.8060 (0.930)	0.8303 (0.843)	6.1757 (6.330)	1.6515 (0.235)
-MLDG	0.9735 (0.124)	1.1813 (0.341)	0.8228 (0.111)	0.9016 (0.167)	0.0956 (0.013)	0.1290 (0.041)	0.8133 (0.163)	1.0119 (0.140)
-IDGM	0.7825 (0.019)	0.8414 (0.013)	0.6360 (0.099)	0.6494 (0.161)	0.0484 (0.007)	0.0755 (0.038)	0.7155 (0.121)	0.9272 (0.124)
-wERM-exp	0.7840 (0.018)	0.8438 (0.014)	0.6252 (0.094)	0.6093 (0.157)	0.0492 (0.004)	0.0752 (0.036)	0.6673 (0.146)	0.8666 (0.141)
-wERM-mix	0.7840 (0.018)	0.8438 (0.014)	0.6234 (0.095)	0.6089 (0.148)	0.0493 (0.004)	0.0753 (0.036)	0.6607 (0.142)	0.8628 (0.140)
-MMD	0.7864 (0.045)	0.8423 (0.030)	0.6655 (0.096)	0.6300 (0.163)	0.0486 (0.006)	0.0769 (0.042)	0.6760 (0.133)	0.8599 (0.131)
-**Cedar**(-D)	0.7883 (0.051)	0.8416 (0.033)	0.6203 (0.083)	0.6158 (0.138)	0.0474 (0.006)	0.0766 (0.042)	**0.6538 (0.151)**	**0.8454 (0.152)**
-**Cedar**	**0.7758 (0.017)**	**0.8349 (0.014)**	**0.6026 (0.082)**	**0.5971 (0.143)**	0.0487 (0.007)	0.0770 (0.042)	0.6632 (0.148)	0.8495 (0.153)
WaveNet	1.0791 (0.061)	1.4159 (0.034)	0.6771 (0.115)	0.8221 (0.241)	0.0489 (0.007)	0.0765 (0.042)	0.6735 (0.149)	0.8647 (0.144)
-DANN	0.8873 (0.125)	0.9027 (0.077)	0.8171 (0.267)	0.6500 (0.162)	0.0472 (0.007)	0.0760 (0.040)	0.7034 (0.170)	0.8765 (0.159)
-GroupDRO	0.9387 (0.091)	0.9731 (0.061)	0.8725 (0.198)	0.8382 (0.149)	0.0493 (0.007)	0.0778 (0.040)	1.3287 (0.936)	1.1952 (0.306)
-MLDG	0.9446 (0.137)	0.9751 (0.085)	0.7070 (0.111)	0.7762 (0.202)	0.0877 (0.010)	0.1139 (0.043)	0.7693 (0.104)	0.9982 (0.142)
-IDGM	1.0892 (0.065)	1.4362 (0.038)	0.6360 (0.099)	0.6494 (0.161)	0.0484 (0.007)	0.0755 (0.038)	0.8502 (0.330)	0.9446 (0.210)
-wERM-exp	1.0686 (0.103)	1.0074 (0.058)	0.6269 (0.102)	0.6483 (0.170)	0.0481 (0.006)	0.0761 (0.040)	0.6729 (0.154)	0.8689 (0.155)
-wERM-mix	1.0396 (0.066)	1.0167 (0.041)	0.6260 (0.093)	0.6392 (0.153)	0.0488 (0.006)	0.0783 (0.042)	0.6852 (0.157)	0.8808 (0.154)
-MMD	0.8852 (0.110)	0.9106 (0.074)	0.6208 (0.103)	0.6400 (0.168)	0.0471 (0.007)	0.0751 (0.041)	0.6711 (0.149)	0.8726 (0.144)
-**Cedar**(-D)	0.8801 (0.110)	0.9106 (0.079)	0.6260 (0.102)	0.6415 (0.168)	**0.0466 (0.007)**	0.0752 (0.041)	0.6794 (0.143)	0.8813 (0.148)
-**Cedar**	0.8791 (0.110)	0.9086 (0.079)	0.6226 (0.102)	0.6400 (0.169)	**0.0466 (0.007)**	**0.0748 (0.040)**	0.6675 (0.155)	0.8599 (0.149)

Bolded values are the best scores for a column. Underlining indicates the method achieves the best performance when the base model is DeepAR or WaveNet. No pairwise significance tests are performed, because the point accuracy metrics are calculated at the group level.

Table 6. Forecasting Results of *Range Accuracy Metrics* on Real-world Datasets

	Favorita-cat		**Favorita-store**		**US-traffic**		**Stock-volume**	
	Q(0.5)	Q(mean)	Q(0.5)	Q(mean)	Q(0.5)	Q(mean)	Q(0.5)	Q(mean)
SN	0.1107 (0.025)	0.1107 (0.025)	0.0211 (0.001)	0.0211 (0.001)	**0.0588 (0.031)**	0.0588 (0.031)	0.2729 (0.071)	0.2729 (0.071)
ES	0.0885 (0.018)	**0.0725 (0.014)**	0.0190 (0.002)	0.0153 (0.001)	0.0627 (0.030)	**0.0528 (0.025)**	0.2371 (0.071)	0.2007 (0.061)
VRNN	0.9683 (0.047)	0.5867 (0.054)	0.8452 (0.120)	0.4669 (0.054)	0.9838 (0.028)	0.6629 (0.062)	0.9706 (0.042)	0.6342 (0.061)
AdaRNN	0.5554 (0.025)	0.3612 (0.009)	0.5108 (0.025)	0.3334 (0.006)	0.4910 (0.011)	0.3280 (0.004)	0.5879 (0.081)	0.4034 (0.026)
DeepAR	0.1461 (0.079)	0.1145 (0.055)	0.0209 (0.001)	0.0164 (0.001)	0.0894 (0.025)	0.0789 (0.020)	0.2415 (0.079)	0.1934 (0.069)
-DANN	0.1003 (0.033)	0.0837 (0.028)	0.0202 (0.001)	0.0165 (0.001)	0.0805 (0.030)	0.0752 (0.017)	0.2324 (0.052)	0.1787 (0.042)
-GroupDRO	0.1549 (0.026)	0.1444 (0.026)	0.0247 (0.003)	0.0263 (0.005)	0.0868 (0.037)	0.0857 (0.019)	0.2270 (0.060)	0.1800 (0.050)
-MLDG	0.1506 (0.043)	0.1339 (0.025)	0.1071 (0.060)	0.1116 (0.039)	0.1091 (0.056)	0.1033 (0.030)	0.2659 (0.098)	0.2081 (0.071)
-IDGM	0.1145 (0.033)	0.0943 (0.029)	0.0244 (0.006)	0.0201 (0.006)	0.0864 (0.023)	0.0706 (0.016)	0.2269 (0.065)	0.1804 (0.053)
-wERM-exp	0.1088 (0.029)	0.0889 (0.024)	0.0204 (0.002)	0.0163 (0.002)	0.1075 (0.065)	0.0912 (0.056)	0.2300 (0.088)	0.1990 (0.080)
-wERM-mix	0.1073 (0.029)	0.0875 (0.024)	0.0209 (0.001)	0.0164 (0.001)	0.0964 (0.024)	0.0792 (0.020)	0.2307 (0.089)	0.2002 (0.081)
-MMD	0.0907 (0.031)	0.0754 (0.026)	0.0201 (0.002)	0.0163 (0.002)	0.0795 (0.029)	0.0722 (0.019)	0.2061 (0.049)	0.1645 (0.038)
-**Cedar**(-D)	*0.0911 (0.030)	*0.0754 (0.026)	*0.0200 (0.002)	*0.0161 (0.002)	*0.0783 (0.030)	*0.0717 (0.020)	**0.2023 (0.047)**	**0.1610 (0.038)**
-**Cedar**	*0.0908 (0.030)	*0.0751 (0.026)	*0.0202 (0.002)	*0.0160 (0.001)$^\diamond$	*0.0757 (0.031)	*0.0727 (0.019)	0.2098 (0.055)	0.1661 (0.044)
WaveNet	0.1206 (0.034)	0.0981 (0.027)	0.0210 (0.001)	0.0165 (0.001)	0.0948 (0.029)	0.0781 (0.022)	0.2278 (0.048)	0.1826 (0.049)
-DANN	0.0882 (0.026)	0.0733 (0.022)	0.0193 (0.002)	0.0152 (0.002)	0.0845 (0.028)	0.0771 (0.017)	0.2672 (0.052)	0.2128 (0.036)
-GroupDRO	0.8020 (0.820)	0.5865 (0.566)	0.0268 (0.004)	0.0286 (0.008)	0.1094 (0.037)	0.1166 (0.060)	0.4056 (0.306)	0.3087 (0.211)
-MLDG	0.3297 (0.199)	0.2572 (0.125)	0.0738 (0.028)	0.1157 (0.017)	0.0992 (0.030)	0.0959 (0.014)	0.3027 (0.059)	0.2436 (0.056)
-IDGM	0.1419 (0.022)	0.1163 (0.016)	0.0248 (0.004)	0.0200 (0.003)	0.0952 (0.022)	0.0785 (0.018)	0.3348 (0.135)	0.2619 (0.119)
-wERM-exp	0.1238 (0.029)	0.1068 (0.026)	0.0229 (0.003)	0.0185 (0.002)	0.0959 (0.022)	0.0841 (0.021)	0.2286 (0.072)	0.2032 (0.071)
-wERM-mix	0.1408 (0.064)	0.1221 (0.057)	0.0226 (0.003)	0.0183 (0.003)	0.0848 (0.038)	0.0723 (0.033)	0.2254 (0.056)	0.2033 (0.055)
-MMD	0.0876 (0.028)	0.0733 (0.024)	0.0175 (0.003)	0.0141 (0.002)	0.0841 (0.031)	0.0743 (0.020)	0.2301 (0.033)	0.1820 (0.032)
-**Cedar**(-D)	***0.0863 (0.026)**	*0.0727 (0.023)	***0.0159 (0.001)**$^\diamond$	***0.0128 (0.001)**$^\diamond$	*0.0845 (0.030)	0.0791 (0.020)	0.2074 (0.034)	0.1699 (0.032)
-**Cedar**	*0.0884 (0.028)	*0.0740 (0.024)	*0.0166 (0.002)	*0.0133 (0.001)	*0.0735 (0.031)$^\diamond$	*0.0688 (0.019)$^\diamond$	0.2322 (0.055)	0.1825 (0.046)

Bolded values are the best scores for a column. Underlining indicates the method achieves the best performance when the base model is DeepAR or WaveNet. * indicates that our methods are significantly better than the base model, and ⋄ indicates the method is significantly better than the second-best baseline given the base model (paired t-test based on Equation (10) when the $\sum_i^N$ and denominator term are removed, p-value < 0.05).

Table 7. Forecasting Results of *Point Accuracy Metrics* on Real-world Datasets

	Favorita-cat		**Favorita-store**		**US-traffic**		**Stock-volume**	
	NRMSE	sMAPE	NRMSE	sMAPE	NRMSE	sMAPE	NRMSE	sMAPE
SN	0.1709 (0.041)	0.2554 (0.071)	0.0280 (0.002)	0.0210 (0.001)	0.1394 (0.058)	**0.0776 (0.037)**	0.6551 (0.192)	0.2437 (0.052)
ES	0.1319 (0.028)	0.2022 (0.065)	0.0258 (0.002)	0.0190 (0.002)	0.1268 (0.047)	0.0808 (0.034)	0.5610 (0.209)	**0.2274 (0.048)**
VRNN	1.1002 (0.044)	1.8591 (0.170)	0.8484 (0.118)	1.5062 (0.356)	1.2127 (0.115)	1.9354 (0.105)	1.7084 (0.120)	1.7996 (0.240)
AdaRNN	0.6498 (0.038)	0.7362 (0.041)	0.5169 (0.025)	0.6903 (0.044)	0.6174 (0.045)	0.6451 (0.023)	1.0712 (0.153)	1.0103 (0.130)
DeepAR	0.2104 (0.116)	0.2501 (0.084)	0.0271 (0.002)	0.0208 (0.001)	0.1446 (0.027)	0.0977 (0.024)	0.5052 (0.147)	0.6536 (0.272)
-DANN	0.1450 (0.055)	0.1984 (0.072)	0.0262 (0.002)	0.0201 (0.001)	0.1278 (0.043)	0.0889 (0.030)	0.4967 (0.121)	0.3644 (0.181)
-GroupDRO	0.2114 (0.043)	0.2427 (0.065)	0.0314 (0.004)	0.0246 (0.003)	0.1351 (0.045)	0.1020 (0.037)	0.4927 (0.123)	0.4470 (0.162)
-MLDG	0.2024 (0.064)	0.2396 (0.063)	0.1276 (0.069)	0.1087 (0.063)	0.1659 (0.057)	0.1181 (0.061)	0.5319 (0.163)	0.6102 (0.303)
-IDGM	0.1596 (0.053)	0.2205 (0.061)	0.0319 (0.007)	0.0244 (0.006)	0.1391 (0.028)	0.0942 (0.024)	0.4840 (0.117)	0.5924 (0.241)
-wERM-ex	0.1548 (0.044)	0.2203 (0.083)	0.0272 (0.003)	0.0203 (0.002)	0.1827 (0.107)	0.1015 (0.044)	0.5210 (0.146)	0.5613 (0.232)
-wERM-mix	0.1517 (0.044)	0.2202 (0.087)	0.0272 (0.002)	0.0208 (0.001)	0.1547 (0.035)	0.1038 (0.021)	0.5255 (0.148)	0.5805 (0.249)
-MMD	0.1326 (0.051)	0.1926 (0.071)	0.0258 (0.003)	0.0201 (0.002)	0.1295 (0.042)	0.0864 (0.030)	0.4679 (0.123)	0.3260 (0.131)
-**Cedar**(-D)	0.1331 (0.051)	0.1917 (0.070)	0.0260 (0.003)	0.0199 (0.002)	0.1266 (0.044)	0.0863 (0.029)	**0.4622 (0.123)**	0.2992 (0.141)
-Cedar	0.1328 (0.051)	**0.1910 (0.069)**	0.0261 (0.002)	0.0202 (0.002)	**0.1252 (0.043)**	0.0844 (0.029)	0.4706 (0.130)	0.3268 (0.152)
WaveNet	0.1696 (0.055)	0.2178 (0.060)	0.0272 (0.001)	0.0209 (0.001)	0.1412 (0.038)	0.1094 (0.022)	0.4937 (0.103)	0.5318 (0.210)
-DANN	0.1301 (0.042)	0.1964 (0.074)	0.0250 (0.002)	0.0192 (0.002)	0.1393 (0.045)	0.0937 (0.028)	0.5673 (0.104)	0.5304 (0.189)
-GroupDRO	0.8848 (0.883)	0.5471 (0.377)	0.0335 (0.004)	0.0266 (0.004)	0.1757 (0.054)	0.1112 (0.031)	0.7155 (0.389)	0.7722 (0.230)
-MLDG	0.3839 (0.227)	0.4484 (0.314)	0.0839 (0.027)	0.0753 (0.030)	0.1501 (0.038)	0.1167 (0.036)	0.6035 (0.074)	0.8271 (0.345)
-IDGM	0.1955 (0.043)	0.2348 (0.061)	0.0325 (0.005)	0.0247 (0.004)	0.1533 (0.032)	0.1050 (0.020)	0.6039 (0.168)	0.7062 (0.334)
-wERM-exp	0.1762 (0.049)	0.2289 (0.069)	0.0293 (0.003)	0.0228 (0.003)	0.1532 (0.022)	0.1065 (0.022)	0.4924 (0.115)	0.6304 (0.260)
-wERM-mix	0.2076 (0.110)	0.2549 (0.073)	0.0297 (0.004)	0.0225 (0.003)	0.1341 (0.047)	0.0932 (0.035)	0.5114 (0.121)	0.5479 (0.196)
-MMD	0.1311 (0.045)	0.1933 (0.067)	0.0236 (0.003)	0.0175 (0.003)	0.1372 (0.044)	0.0916 (0.030)	0.5079 (0.095)	0.5589 (0.240)
-**Cedar**(-D)	**0.1293 (0.041)**	0.1911 (0.069)	**0.0217 (0.001)**	**0.0159 (0.001)**	0.1383 (0.039)	0.0897 (0.032)	0.4815 (0.109)	0.6342 (0.364)
-Cedar	0.1313 (0.046)	0.1943 (0.068)	0.0223 (0.002)	0.0166 (0.002)	0.1255 (0.044)	0.0819 (0.030)	0.4971 (0.114)	0.6066 (0.279)

Bolded values are the best scores for a column. Underlining indicates the method achieves the best performance when the base model is DeepAR or WaveNet. No pairwise significance tests are performed, because the point accuracy metrics are calculated at the group level.

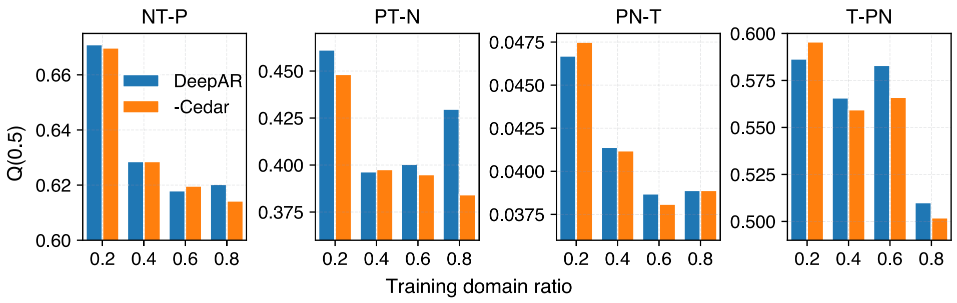


Fig. 2. Prediction performance on synthetic datasets with varying ratios of training domains.

data. **Cedar** consistently outperforms the base models (DeepAR and WaveNet), demonstrating its effectiveness in real-world scenarios compared to the naive empirical risk minimization method. When compared to other domain generalization methods, **Cedar** exhibits the best overall performance. GroupDRO and MLDG show unfavorable results on both the point accuracy and range accuracy metrics. MMD, IDGM, and DANN show promising performance in certain scenarios, but they do not consistently achieve satisfactory results across all scenarios. **Cedar** achieves superior results on US-traffic, while on other datasets, **Cedar**(-D) performs exceptionally well or exhibits close performance. It indicates that considering the difficulty of each domain by examining the within-domain performance becomes less effective. This may be due to the existence of highly regular intra-domain patterns in all domains. It also suggests that there might be other methods that are potentially better for quantifying the difficulties of domains than just using loss variance.

4.4 Sensitivity Analysis

Ratio of training domains (M/K). We investigate the impact of different training domain ratios on our approach's performance. Figure 2 shows the Q(0.5) results for the base model DeepAR and the model with our proposed method **Cedar**, as we vary the training domain ratio from 0.2 to 0.8. The performance of other metrics shows a similar pattern. As the number of training domains increases, models tend to achieve better performance on test domains. This could be attributed to the fact that more training domains allow models to digest a greater variety of data, which enhances their ability to generalize to unseen domains. With fewer test domains, the model might have already seen similar data during training, leading to better generalization performance.

Cedar generally performs better or shows competitive results in most cases. However, when the number of training domains is smaller (0.2 and 0.4), we observe that the base model performs more stable across datasets. This might be because a reduced number of training domains introduces greater challenges for generalizing to unseen domains. In such scenarios, considering the base model in the first place could be a safer option.

Values of γ. In Figure 3, we illustrate the impact of different values of γ on the regularization term $\mathcal{R}_{DDD}$ in **Cedar** by plotting the corresponding validation loss for all datasets. We observe that the optimal value of γ varies across datasets (typically achieving better results when less than 1), depending on the scale of the loss and the magnitude of differences between domain losses. In datasets with substantial differences in losses between domains, a smaller γ is preferred to avoid excessive regularization term optimization at the expense of forecasting performance. Choosing an appropriate γ is crucial to strike a balance between domain generalization and accurate forecasting performance at the sample level.

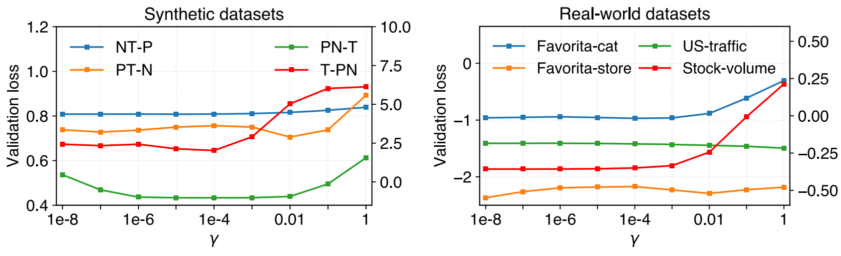


Fig. 3. Validation loss on all datasets with varying γ values.

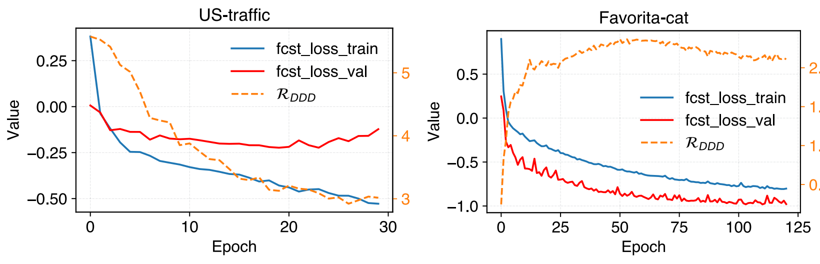


Fig. 4. Example forecast loss and regularization curves.

4.5 Convergence Analysis

We analyze the convergence of **Cedar** applied to DeepAR by recording the forecast losses on training and validation sets, along with the regularization term. In Figure 4, we plot the curves ($\mathcal{R}_{DDD}$ is multiplied with the hyperparameter γ) for the US-traffic and Favorita-cat datasets. We see that **Cedar** can easily be trained with smooth convergence, witness the decrease in training and validation loss. For the regularization term, we notice that it continues to decrease on the US-traffic dataset. It indicates that the regularization is effectively guiding the model to reduce the performance discrepancies between domains, promoting better generalization across different domains. It also implies that the domains have enough similarity or overlap in their data distributions, making it easier for the model to generalize across domains. On Favorita-cat, we observe an increasing pattern of the regularization term, although it still outperforms other methods. This suggests that the regularization term is effectively guiding the model to adapt to the differences between domains to some extent, even if it does not entirely eliminate the performance discrepancy (possibly due to the large difference between domains). The increasing pattern may also occur when the variance of within-domain losses becomes smaller through training.

4.6 Domain Performance Analysis

Cedar regulates the performance across domains during training, preventing overfitting in each domain and ensuring good generalization on unseen domains. To assess the within-domain performance distributions, we analyze the Stock-volume dataset, which has the smallest number of domains for better visualization. The results are presented in a boxplot (Figure 5), where the y-axis represents the loss value and the x-axis corresponds to different domains. **Cedar** achieves more even distributions for some training domains (e.g., 6–9), resulting in a more uniform loss distribution across domains, i.e., less underfitting and overfitting. Moreover, for the test domains, **Cedar** demonstrates notable performance improvements across all domains.

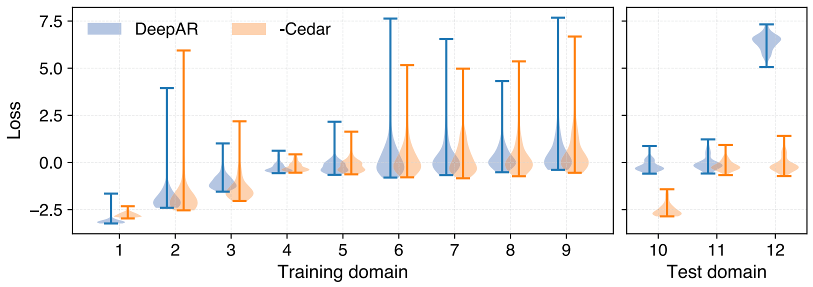


Fig. 5. Forecast performance on training domains and test domains between the base model and **Cedar** on Stock-volume.

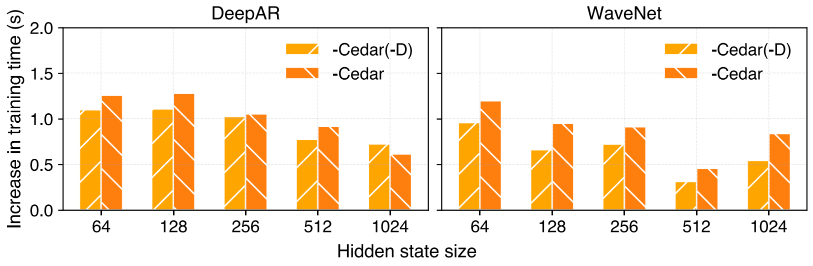


Fig. 6. Training time analysis on the NT-P dataset with varying model sizes.

4.7 Computational Analysis

We conduct two experiments to evaluate the computational efficiency of **Cedar**, taking into account variations in both base forecasting model size and dataset size.

In the first experiment, we analyze the training time of **Cedar** in comparison with the base models DeepAR and WaveNet while varying the base model size. These experiments are performed using the synthetic dataset NT-P with default settings, e.g., a batch size of 64 and a training domain ratio of 0.8. Figure 6 displays the increase in training time (per epoch) of **Cedar** relative to the base model. We use the average training time of three epochs as the final measurement. We observe that the additional overhead incurred by **Cedar** is very small, ranging only within a few seconds. Furthermore, this overhead exhibited a slight decreasing trend as the model size increases (i.e., the hidden state size increases). Our generalization approach causes nearly constant time overhead, demonstrating its effectiveness for larger models. These findings indicate that **Cedar** is well-suited for real-world applications with more complex data, which typically require large models for accurate forecasting.

In the second experiment, we investigate how **Cedar** performs with larger time series datasets. We extend the synthetic datasets from NT-P, increasing the length of each domain's time series from 500 data points to 1k, 10k, and 100k data points. The training domain ratio (e.g., 0.8) and the ratios of training, validation, and testing timestamps remain the same as in our main experiments. To accommodate the larger dataset, we adjust the batch size to 1,024 and increase the hidden states of the base model to 1,024. The results in Figure 7 show the percentage increase in training time (per epoch) for **Cedar** compared to the base model. We also use the average training time of three epochs as the final measurement. Notably, when the base model is DeepAR, the percentage increase is consistently less than 9%. For WaveNet, there is some randomness, but the largest increase percentage remained under 10%. In some cases, **Cedar** adds almost no additional time

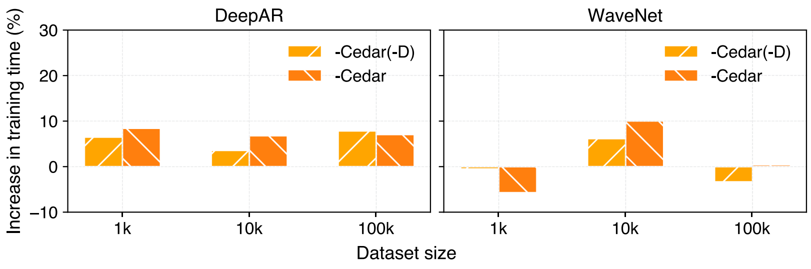
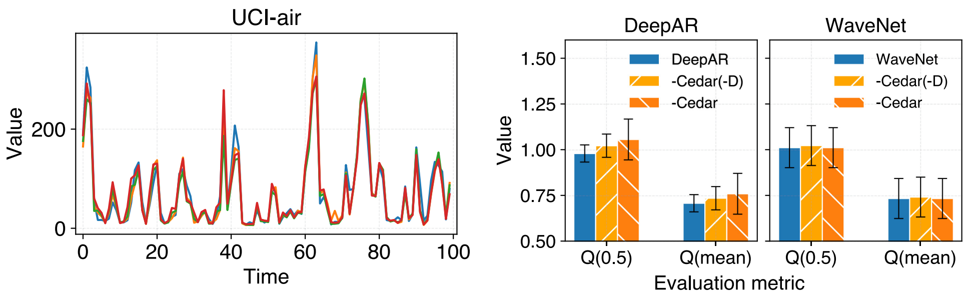


Fig. 7. Training time analysis on larger NT-P datasets.



(a) Data visualization

(b) Comparison results

Fig. 8. (a) Visualization of partial domain data on the UCI-air dataset, where each sequence represents a domain and (b) comparison results of our method with the base model.

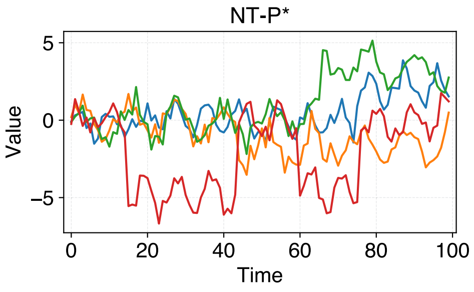
to the base model. These results highlight the efficiency of **Cedar** when dealing with larger datasets.

4.8 Beyond Our Assumptions

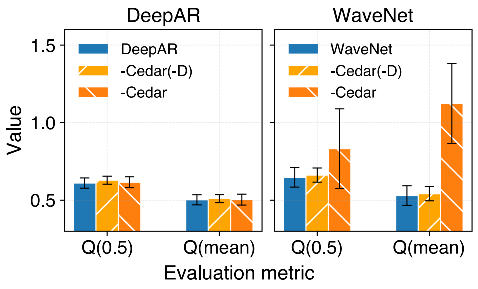
Cedar may not work effectively in scenarios that violate our data assumptions. To illustrate this, we present two negative examples where our method could potentially encounter challenges.

Violating Assumption 1. We conduct an experiment using the **UCI_air** dataset. This dataset is built from the Beijing Multi-site Air-quality Dataset, which contains hourly air pollutant data from 12 nationally controlled air-quality monitoring sites from 2013 to 2017.[11] Each monitoring site is considered as a domain, and we use PM2.5 data from 2016-01-01 to 2016-09-10 for training, data from 2016-09-11 to 2016-12-01 for validation, and the rest of the data up to 2017-02-28 for testing. Due to a large number of missing values, we convert the data to daily values by averaging over each hour of the day. The historical window size is 28, and the prediction window is 7. In Figure 8(a), we observe a high degree of similarity and overlap across different domains, which violates Assumption 1, where ϵ_l is close to 0. The presence of these highly similar patterns also raises concerns about the suitability of applying a domain generalization model in this particular context. We show the comparison results of our method with the two base models in Figure 8(b). The base models perform better. This finding emphasizes the importance of considering the shared patterns and differences between domains to design effective domain generalization models (Assumption 1).

[11] https://www.kaggle.com/datasets/sid321axn/beijing-multisite-airquality-data-set



(a) Data visualization



(b) Comparison results

Fig. 9. (a) Visualization of partial domain data on the NT-P* dataset, where each sequence represents a domain and (b) comparison results of our method with the base model.

Violating Assumption 2. Assumption 2 assumes that the time series data should not exhibit abrupt distribution shifts within each domain. To test this assumption, we apply a straightforward method to manipulate the synthetic dataset NT-P by introducing random mean shifts within different segments while keeping other properties unchanged. A portion of the manipulated data **NT-P*** is presented in Figure 9(a), and the comparison results are shown in Figure 9(b). Our observations reveal that when the base model is DeepAR, the performance differences between **Cedar** and the base model are relatively minor, with the base model performing slightly better. However, for the base model WaveNet, **Cedar** struggles to produce favorable results. When considering domain difficulties in the regularization, the results become much less stable. This outcome can be attributed to the fact that the factors considered for training, which address domain difficulties, do not effectively apply to unseen domains that may exhibit substantial disparities. When time series data involve abrupt changes or significant distribution shifts (i.e., scenarios that contradict Assumption 2), some researchers [15, 39] propose characterizing temporal distributions to segment time series based on distribution differences, thus establishing distinct domains. Their characterization method presents a potential solution to address the limitations of our current approach. We leave further analysis of this potential enhancement for future research.

5 CONCLUSION

We have presented **Cedar**, a novel approach to address the problem of domain generalization in time series forecasting. By incorporating predefined assumptions about cross- and within-domain patterns, we introduce two novel regularization methods to improve forecasting performance across different time series domains. Through comprehensive experiments conducted on both synthetic and real-world time series datasets, we have systematically compared our method against several state-of-the-art approaches, demonstrating its effectiveness. **Cedar** has advantages such as applicability to various forecasting models and potential use in non-time series regression problems. It can be used to enhance decision-making processes and optimize resource allocation by improving overall forecasting accuracy across diverse domains, thereby offering practical and beneficial applications in various fields.

Cedar has demonstrated its effectiveness based on two important assumptions about the data. However, when these assumptions are violated, its performance drops (Section 4.8). How can we overcome this limitation and achieve generalization in such cases? Another important line for future work is to develop adaptive regularization techniques that can dynamically adjust to different time series data characteristics and domain shifts.

REPRODUCIBILITY

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

ACKNOWLEDGMENTS

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Faruk Ahmed, Yoshua Bengio, Harm van Seijen, and Aaron Courville. 2021. Systematic generalisation with group invariant predictions. In *International Conference on Learning Representations*.

[2] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. *arXiv preprint arXiv:1907.02893* (2019).

[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 preprint arXiv:1409.0473* (2014).

[5] Guangji Bai, Chen Ling, and Liang Zhao. 2022. Temporal domain generalization with drift-aware dynamic neural networks. *arXiv preprint arXiv:2205.10664* (2022).

[6] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[7] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. 2018. MetaReg: Towards domain generalization using meta-regularization. *Adv. Neural Inf. Process. Syst.* 31 (2018).

[8] Michael A. Benjamin, Robert A. Rigby, and D. Mikis Stasinopoulos. 2003. Generalized autoregressive moving average models. *J. Am. Stat. Assoc.* 98, 461 (2003), 214–223.

[9] Stefanos Bennett and Jase Clarkson. 2022. Time series prediction under distribution shift using differentiable forgetting. *arXiv preprint arXiv:2207.11486* (2022).

[10] James Lopez Bernal, Steven Cummins, and Antonio Gasparrini. 2017. Interrupted time series regression for the evaluation of public health interventions: A tutorial. *Int. J. Epidemiol.* 46, 1 (2017), 348–355.

[11] Real Carbonneau, Kevin Laframboise, and Rustam Vahidov. 2008. Application of machine learning techniques for supply chain demand forecasting. *Eur. J. Oper. Res.* 184, 3 (2008), 1140–1154.

[12] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[13] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. *Adv. Neural Inf. Process. Syst.* 28 (2015).

[14] Weizhen Dang, Haibo Wang, Shirui Pan, Pei Zhang, Chuan Zhou, Xin Chen, and Jilong Wang. 2022. Predicting human mobility via graph convolutional dual-attentive networks. In *15th ACM International Conference on Web Search and Data Mining*. 192–200.

[15] 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 *30th ACM International Conference on Information & Knowledge Management.* 402–411.

[16] 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 preprint arXiv:2203.09978* (2022).

[17] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.* 17, 1 (2016), 2096–2030.

[18] Everette S. Gardner Jr. 2006. Exponential smoothing: The state of the art-Part II. *Int. J. Forecast.* 22, 4 (2006), 637–666.

[19] Michael Ghil, Myles R. Allen, Michael D. Dettinger, Kayo Ide, Dmitri A. Kondrashov, Michael E. Mann, Andrew W. Robertson, Amira Saunders, Yudong Tian, Ferenc Varadi, and Pascal Yiou. 2002. Advanced spectral methods for climatic time series. *Rev. Geophys.* 40, 1 (2002), 3–1.

[20] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS'10).*

[21] Tilmann Gneiting and Matthias Katzfuss. 2014. Probabilistic forecasting. *Ann. Rev. Stat. Applic.* 1 (2014), 125–151.

[22] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. 2019. DLOW: Domain flow for adaptation and generalization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2477–2486.

[23] Xiaochuan Gou and Xiangliang Zhang. 2023. Telecommunication traffic forecasting via multi-task learning. In *16th ACM International Conference on Web Search and Data Mining*. 859–867.

[24] Ishaan Gulrajani and David Lopez-Paz. 2020. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434* (2020).

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

[26] Yeping Hu, Xiaogang Jia, Masayoshi Tomizuka, and Wei Zhan. 2022. Causal-based time series domain generalization for vehicle intention prediction. In *International Conference on Robotics and Automation (ICRA'22)*. IEEE, 7806–7813.

[27] Thanh Trung Huynh, Minh Hieu Nguyen, Thanh Tam Nguyen, Phi Le Nguyen, Matthias Weidlich, Quoc Viet Hung Nguyen, and Karl Aberer. 2023. Efficient integration of multi-order dynamics and internal dynamics in stock movement prediction. In *16th ACM International Conference on Web Search and Data Mining*. 850–858.

[28] Rob J. Hyndman and George Athanasopoulos. 2018. *Forecasting: Principles and Practice*. OTexts.

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

[30] 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. *Adv. Neural Inf. Process. Syst.* 30.

[31] Kyoung-jae Kim. 2003. Financial time series forecasting using support vector machines. *Neurocomputing* 55, 1-2 (2003), 307–319.

[32] Durk Kingma and J. Ba Adam. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, Vol. 5.

[33] Colin Lewis-Beck and Michael Lewis-Beck. 2015. *Applied Regression: An Introduction*. Vol. 22. Sage Publications.

[34] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. 2018. Learning to generalize: Meta-learning for domain generalization. In *AAAI Conference on Artificial Intelligence*, Vol. 32.

[35] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C. Kot. 2018. Domain generalization with adversarial feature learning. In *IEEE Conference on Computer Vision and Pattern Recognition*. 5400–5409.

[36] 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. *Adv. Neural Inf. Process. Syst.* 32 (2019).

[37] Yiying Li, Yongxin Yang, Wei Zhou, and Timothy Hospedales. 2019. Feature-critic networks for heterogeneous domain generalization. In *International Conference on Machine Learning*. PMLR, 3915–3924.

[38] Bryan Lim and Stefan Zohren. 2021. Time-series forecasting with deep learning: A survey. *Philos. Trans. Roy. Societ. A* 379, 2194 (2021), 20200209.

[39] Wang Lu, Jindong Wang, Xinwei Sun, Yiqiang Chen, and Xing Xie. 2022. Out-of-distribution representation learning for time series classification. In *11th International Conference on Learning Representations*.

[40] Francisco Martínez, María Pilar Frías, María Dolores Pérez, and Antonio Jesús Rivera. 2019. A methodology for applying k-nearest neighbor to time series forecasting. *Artif. Intell. Rev.* 52, 3 (2019), 2019–2037.

[41] Augusto Steves Mendoza Calero. 2018. *Corporación Favorita Grocery Sales Forecasting Kaggle Competition*. Master's thesis. Universidad Internacional de Andalucía.

[42] Meinard Müller. 2007. Dynamic time warping. In *Information Retrieval for Music and Motion*. Springer, 69–84.

[43] 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. *Adv. Neural Inf. Process. Syst.* 34 (2021), 19198–19209.

[44] 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 preprint arXiv:1609.03499* (2016).

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* 32 (2019).

[46] Fengchun Qiao, Long Zhao, and Xi Peng. 2020. Learning to learn single domain generalization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12556–12565.

[47] 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 Trans. Instrum. Measur.* 71 (2022), 1–12.

[48] Thomas P. Ryan. 2008. *Modern Regression Methods*. Vol. 655. John Wiley & Sons.

[49] 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 preprint arXiv:1911.08731* (2019).

[50] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecast.* 36, 3 (2020), 1181–1191.

[51] Uri Shalit, Fredrik D. Johansson, and David Sontag. 2017. Estimating individual treatment effect: Generalization bounds and algorithms. In *International Conference on Machine Learning*. JMLR. org, 3076–3085.

[52] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. 2018. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745* (2018).

[53] Yuge Shi, Jeffrey Seely, Philip H. S. Torr, N. Siddharth, Awni Hannun, Nicolas Usunier, and Gabriel Synnaeve. 2021. Gradient matching for domain generalization. *arXiv preprint arXiv:2104.09937* (2021).

[54] Olivier Sprangers, Sebastian Schelter, and Maarten de Rijke. 2023. Parameter-efficient deep probabilistic forecasting. *Int. J. Forecast.* 39, 1 (2023), 332–345.

[55] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *J. Roy. Stat. Societ.: Series B (Methodol.)* 58, 1 (1996), 267–288.

[56] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'17)*. IEEE, 23–30.

[57] Vladimir Vapnik. 1999. *The Nature of Statistical Learning Theory*. Springer Science & Business Media.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Adv. Neural Inf. Process. Syst.* 30 (2017).

[59] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C. Duchi, Vittorio Murino, and Silvio Savarese. 2018. Generalizing to unseen domains via adversarial data augmentation. *Adv. Neural Inf. Process. Syst.* 31 (2018).

[60] Chunyang Wang, Yanmin Zhu, Tianzi Zang, Haobing Liu, and Jiadi Yu. 2021. Modeling inter-station relationships with attentive temporal graph convolutional network for air quality prediction. In *14th ACM International Conference on Web Search and Data Mining*. 616–634.

[61] Jindong Wang, Yiqiang Chen, Wenjie Feng, Han Yu, Meiyu Huang, and Qiang Yang. 2020. Transfer learning with dynamic distribution adaptation. *ACM Trans. Intell. Syst. Technol.* 11, 1 (2020), 1–25.

[62] Jindong Wang, Wenjie Feng, Yiqiang Chen, Han Yu, Meiyu Huang, and Philip S. Yu. 2018. Visual domain adaptation with manifold embedded distribution alignment. In *26th ACM International Conference on Multimedia*. 402–410.

[63] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. 2023. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering* 35, 8 (2023), 8052–8072.

[64] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2020. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 32, 1 (2020), 4–24.

[65] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. 2019. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *IEEE/CVF International Conference on Computer Vision*. 2100–2110.

[66] Haoran Zhang, Natalie Dullerud, Laleh Seyyed-Kalantari, Quaid Morris, Shalmali Joshi, and Marzyeh Ghassemi. 2021. An empirical framework for domain generalization in clinical settings. In *Conference on Health, Inference, and Learning*. 279–290.

[67] Ningning Zhang, Aijing Lin, and Pengjian Shang. 2017. Multidimensional k-nearest neighbor model based on EEMD for financial time series forecasting. *Phys. A: Stat. Mechan. Applic.* 477 (2017), 161–173.

[68] Shichao Zhang and Jiaye Li. 2023. KNN classification with One-step computation. *IEEE Transactions on Knowledge and Data Engineering* 35, 3 (2023), 2711–2723.

[69] Shichao Zhang, Jiaye Li, and Yangding Li. 2023. Reachable distance function for KNN classification. *IEEE Transactions on Knowledge and Data Engineering* 35, 7 (2023), 7382–7396.

[70] Shichao Zhang, Jiaye Li, Wenzhen Zhang, and Yongsong Qin. 2022. Hyper-class representation of data. *Neurocomputing* 503 (2022), 200–218.

[71] Shichao Zhang, Xuelong Li, Ming Zong, Xiaofeng Zhu, and Ruili Wang. 2017. Efficient kNN classification with different numbers of nearest neighbors. *IEEE Trans. Neural Netw. Learn. Syst.* 29, 5 (2017), 1774–1785.

[72] Wenyu Zhang, Mohamed Ragab, and Chuan-Sheng Foo. 2022. Domain generalization via selective consistency regularization for time series classification. In *26th International Conference on Pattern Recognition (ICPR'22)*. IEEE, 2149–2156.

Received 30 October 2023; revised 4 January 2024; accepted 8 January 2024