# DAD4TS: Data-Augmentation-Oriented Diffusion Model for Time-Series Forecasting with Small-Scale Data

Masahiro Suzuki
Sony Group Corporation
Tokyo, Japan
Masahiro.D.Suzuki@sony.com

Bohui Xia
Sony Group Corporation
Tokyo, Japan
Bohui.Xia@sony.com

Hiroto Yamamoto
Sony Group Corporation
Tokyo, Japan
Hiroto.Yamamoto@sony.com

Masanori Miyahara
Sony Group Corporation
Tokyo, Japan
Masanori.Miyahara@sony.com

## Abstract

Small-scale data is a critical problem in time-series forecasting tasks. Data augmentation is an effective strategy for this task, but it has a limitation in generating meaningful data. To address this limitation, we propose DAD4TS, a diffusion-model-based data augmentation method with reinforcement learning, designed for time-series forecasting with small-scale data. In DAD4TS, a data generator is simultaneously trained with a time-series model and controlled by a reinforcement learning model to efficiently generate samples that improve the forecast accuracy of the time-series model. To support small-scale data, we use mathematical methods instead of conventional VAE methods to train the diffusion model by projecting the time-series data into the geometric space. We validated the effectiveness of DAD4TS with seven comparative methods through qualitative and quantitative experiments on six real-world datasets and eight time-series models. As a result, DAD4TS was validated on five datasets.

## CCS Concepts

• **Computing methodologies** → **Machine learning**; **Feature selection**.

## Keywords

time series forecasting, data augmentation, diffusion models, small-scale data

**ACM Reference Format:**
Masahiro Suzuki, Bohui Xia, Hiroto Yamamoto, and Masanori Miyahara. 2018. DAD4TS: Data-Augmentation-Oriented Diffusion Model for Time-Series Forecasting with Small-Scale Data. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*Conference acronym 'XX, Woodstock, NY*
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

**Table 1: Average RMSE over multiple prediction lengths with input length fixed to 12. Results are averaged over forecast lengths $\in \{3, 6, 9, 12\}$ for all datasets, except ILI where $\in \{2, 4, 8, 12\}$. "Gaussian" refers to the case where time-series data—generated by adding Gaussian noise to the residuals from an STL decomposition of the original data and then reconstructing it—is used as training data alongside the original data. Δ(%) shows the relative change from baseline to "Gaussian". (blue = degradation).**

| Dataset | Methods | RNN RMSE↓ | LSTM RMSE↓ | Trans. RMSE↓ | iTrans. RMSE↓ | Patch. RMSE↓ | Times. RMSE↓ | S2IP RMSE↓ | OLin. RMSE↓ | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|
| Employees | baseline | 1.32 | 1.38 | 1.30 | 0.153 | 0.144 | 0.192 | 0.184 | 0.201 | |
| | +Gaussian | 1.30 | 1.42 | 1.41 | 0.161 | 0.148 | 0.207 | 0.182 | 0.209 | |
| | Δ(%) | -1.52% | +2.72% | +8.67% | +5.30% | +2.85% | +7.77% | -1.09% | +3.85% | +3.57% |
| Forest | baseline | 1.15 | 1.27 | 1.08 | 1.02 | 1.10 | 1.08 | 1.02 | 1.03 | |
| | +Gaussian | 1.18 | 1.28 | 1.14 | 1.02 | 1.07 | 1.10 | 1.05 | 1.05 | |
| | Δ(%) | +2.30% | +0.988% | +5.95% | +0.220% | -2.77% | +1.60% | +2.30% | +1.26% | +1.48% |
| German | baseline | 1.20 | 1.44 | 1.34 | 1.34 | 1.47 | 1.37 | 1.48 | 1.39 | |
| | +Gaussian | 1.25 | 1.47 | 1.37 | 1.41 | 1.49 | 1.42 | 1.43 | 1.53 | |
| | Δ(%) | +4.44% | +2.07% | +2.23% | +5.22% | +1.21% | +3.71% | -3.45% | +9.91% | +3.17% |
| ILI | baseline | 1.83 | 1.81 | 1.98 | 0.567 | 0.580 | 0.584 | 0.601 | 0.614 | |
| | +Gaussian | 1.84 | 1.87 | 1.99 | 0.572 | 0.549 | 0.567 | 0.555 | 0.595 | |
| | Δ(%) | +0.273% | +3.31% | +0.253% | +0.793% | -5.35% | -2.91% | -7.57% | -3.05% | -1.78% |
| Inventories | baseline | 0.748 | 0.719 | 0.859 | 0.552 | 0.549 | 0.567 | 0.525 | 0.561 | |
| | +Gaussian | 0.746 | 0.731 | 0.771 | 0.531 | 0.555 | 0.555 | 0.522 | 0.499 | |
| | Δ(%) | -0.267% | +1.70% | -10.2% | -3.85% | +1.19% | -2.03% | -0.524% | -11.2% | -3.14% |
| Consumption | baseline | 4.72 | 4.58 | 4.51 | 0.342 | 0.398 | 0.394 | 0.688 | 0.530 | |
| | +Gaussian | 4.71 | 4.52 | 4.52 | 0.380 | 0.479 | 0.388 | 0.469 | 0.498 | |
| | Δ(%) | -0.0530% | -1.42% | +0.222% | +11.1% | +20.4% | -1.59% | -31.9% | -6.00% | -1.14% |

## 1 Introduction

Time-series forecasting (TSF) is one of the important tasks to solve a wide range of real-world problems, such as power management [26], traffic forecasting [25], and weather forecasting [10]. On the other hand, there are many cases in which sufficient training data cannot be collected [15] in actual operation, and it is an important challenge to build effective forecasting models with small-scale data [41]. In addressing such cases, data augmentation is frequently employed as a strategy when training data is small scale [42], and various methods have been proposed to date, not limited to time-series domain [4, 9, 20, 32, 35, 46, 49]. Existing research in this field has extensively examined generation-based data augmentation [7–9, 15, 35, 46–49], and it has been demonstrated that, when data augmentation methods are designed appropriately, such augmentation can indeed improve predictive performance [7, 8, 12, 15, 35, 36, 47, 48]. In this way, design approaches that take downstream tasks into account have been gaining attention in recent years.

However, despite such progress, many previous studies have focused on how closely the generated data resembles the original

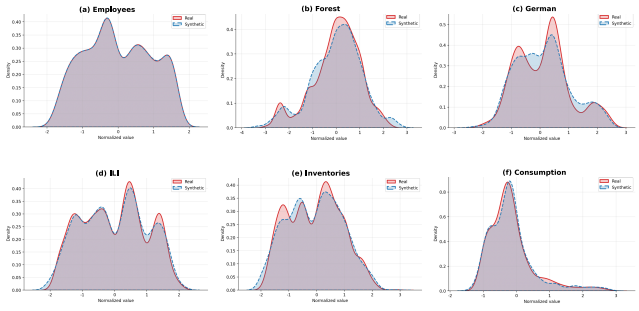


**Figure 1: Distribution plots for each dataset of real and synthetic data.**

data, or have generated data to improve the accuracy of TSF models on the assumption that large amounts of data are available. However, when considering downstream tasks, generating and using data that resemble real data may also result in the generation and use of data that do not positively contribute to learning, which could adversely affect downstream tasks. In fact, as shown in Table 1 based on preliminary experiments, it was confirmed in several cases that generating and using data resembling real data (Figure 1) negatively affected downstream tasks. Furthermore, several previous studies have confirmed that the performance of generative models declines when the amount of data available for training is limited [9, 12, 15]. Consequently, when training data is small scale, there are limitations to applying many of the previously reported studies to downstream tasks.

On the other hand, many methods are static, with the data generation phase completed in a single step, and there are few methods that dynamically continue to generate samples optimized for TSF models. While the utility of synthetic samples depends on the current state of the forecasting model, even existing online augmentation frameworks provide only limited mechanisms for explicitly determining, based on forecasting performance, which samples should be generated and retained, and for integrating such decisions into the updates of both the generator and the forecaster.

In this paper, we introduce our method that Data-Augmentation-Oriented Diffusion Model for Time-Series Forecasting with Small-Scale Data (DAD4TS), a data augmentation method that generates samples to improve the accuracy of TSF models. The contributions of our method are as follows:

- **Optimization of Data Augmentation for Small-Scale Time-Series Data.** We propose a data augmentation framework for TSF models in small-scale data settings. The primary objective of our framework is to directly maximize downstream forecasting performance, and we demonstrate that it significantly improves time-series forecast accuracy under limited data settings compared to existing data augmentation methods.
- **Value-Based Sampling of Generated Data.** Instead of indiscriminately using all generated samples for training, our framework introduces a Selector that continuously chooses informative generated samples based on a forecast-oriented reward. The Selector is trained to evaluate the utility of each generated sample by using improvements in the forecasting performance of the TSF models on a validation set as the reward signal.
- **Joint training strategy.** Rather than treating the TSF models, the data generation model, and the Selector as independent components, we jointly train and update all three modules in an online manner. Data generation is not performed as a one-shot preprocessing step; instead, the Selector continuously determines which samples should be generated and retained based on their contribution to forecasting performance. These decisions are integrated into the updates of both the generator and the forecaster, resulting in a dynamic data augmentation process that continually adapts to the target forecasting task.
- **TSF models-agnostic integration.** DAD4TS can be integrated in a manner independent of the architecture of TSF models and functions as a general-purpose extension module for time-series forecasting.

## 2 Related Work

### 2.1 Generation of Time-Series Data

Existing methods for time-series data generation include GAN and VAE based approaches [9, 11, 44, 46], as well as diffusion-model-based approaches [7, 8, 15, 34, 35, 47, 49, 51]. Similar to the image and speech domains, diffusion models have recently achieved remarkable success in time-series data generation. Building on the success of image generation using diffusion models, Suzuki et al. [35] visualized time-series data, used diffusion models to sample the data, and combined real and synthetic data to improve the accuracy of forecasting model. Yuan et al. [48] identify data points where the forecasting model is prone to overfitting, utilise these points as anchors, and dynamically perform data augmentation to prevent the model from overfitting to specific patterns, thereby improving forecast accuracy. Deng et al. [8] propose data augmentation for time-series foundation models (TSFMs), performing data augmentation that scales efficiently whilst calculating the balance of various time-series data, thereby improving the performance of TSFMs. Yu et al. [47] focus on temporal continuity—a factor previously overlooked in time-series data generation—and adopt a two-stage approach comprising a pre-training phase to preserve temporal continuity and a guidance augmentation phase to enable diffusion models to generate augmented data. Gonen et al. [15] propose an approach that involves training a diffusion model on a large volume of time-series data, followed by fine-tuning on a small dataset, thereby enabling efficient data generation for small-scale time-series data.

However, in many recent diffusion model approaches, data generation is performed only once, and data cannot be continuously generated toward the optimization of downstream tasks. In contrast, we aim to continuously perform data generation in conjunction with the training of TSF models, sequentially producing samples that contribute to improving forecasting accuracy. Moreover, many prior studies are based on DDPM [17], which requires a large number of sampling steps. Continuous data generation under such settings would incur prohibitive computational costs. Therefore,

we use an approach that enables efficient data generation with a small number of sampling steps.

## 2.2 Data selection methods

The quality of training data used for machine learning and deep learning models has been extensively discussed in prior studies [13, 16, 21, 22, 45]. Since learning models indiscriminately learn from both informative data and noisy data, performance degradation can occur across various tasks regardless of how well the model itself is designed. To address this issue, Just et al. [22] proposed a method for estimating data value that is independent of the downstream learning algorithm. Their approach evaluates data by comparing feature–label pairs between training and validation datasets. In contrast, Yoon et al. [45] introduce a selection model that determines which data samples should be used for training. The selection model is trained such that a forecasting model trained on the selected data achieves higher validation performance than a model trained on the full dataset, thereby identifying valuable data samples. The DVRL method proposed by Yoon et al. [45] has been shown to be effective and highly generalizable across image, tabular, and language datasets.

As our study aims to continuously generate samples that contribute to improving the predictive performance of TSF models, it is necessary to control the diffusion model to enable effective data generation. In this context, the DVRL framework has the potential to serve as an effective mechanism. Accordingly, we apply the DVRL method to the time-series data generation task in our work.

# 3 DAD4TS

This section describes the DAD4TS. The proposed framework consists of two main components. The first is a Rectified Flow–based diffusion model, which enables data generation with a small number of sampling steps during the generation process. The second is Data Valuation using Deep Reinforcement Learning for Generated Samples (DVRL-G), a method that evaluates the value of samples generated by the generative model and determines whether they should be used for training the time-series forecasting model. Each of the following subsections details these components. Figure 2 shows an overview of DAD4TS.

## 3.1 Diffusion Model

This subsection describes the diffusion model developed in this study. Conventionally, diffusion models first map input data into a latent space using a VAE [23], and then learn a diffusion process on the latent representations to generate data [5, 33]. However, when the amount of available training data is limited, VAEs based on deep learning may not be able to learn sufficiently, making it difficult to capture the essential features of the data.

To address this issue, our approach maps time-series data into a geometric space without relying on deep learning–based methods. Specifically, the following steps are applied to the time-series data.

(1) For a univariate time-series $\{d_t\}_{t=1}^{T}$, where $d_t \in \mathbb{R}$ for each $t$. We split the data into training, validation, and test sets; $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, and $\mathcal{D}_{\text{test}}$, respectively. The split is in a ratio of 6:2:2. All splits are standardized to zero mean and unit variance using the mean and standard deviation estimated from $\mathcal{D}_{\text{train}}$.
(2) Prepare a dataset $\mathcal{D}_{\text{train}}^{\text{win}}$ from $\mathcal{D}_{\text{train}}$ with a sliding window size of 1 and a batch size of B. The same settings are used for time-series forecasting.
(3) Extract samples from $\mathcal{D}_{\text{train}}^{\text{win}}$ and calculate Gram matrices for each extracted sample to capture time-series features.
(4) Given a mini-batch from $\mathcal{D}_{\text{train}}^{\text{win}}$, we fit a PCA [29] model per mini-batch on its representations and transform them into 2D embeddings.
(5) For each two-dimensional vector obtained from each mini-batch in Step 4, we compute its outer product and set all off-diagonal elements to zero. This processing takes into account the correspondence of coordinates in the geometric space.

The matrix computed through the above procedures constitutes the geometric space used by the diffusion model and is provided as input to the model. In addition, the time variable $t \in [0, 1]$ and a conditioning variable $c$ are simultaneously fed into the diffusion model. The conditioning variable (c) consists of embeddings extracted from TSFMs—specifically, Chronos-Tiny and Chronos-Base based on Chronos-Bolt [1], as well as Tirex [2]—which encode representations of the input time-series data. Among these embeddings, those selected by a Top-1 mixture-of-experts (MoE) setting are used as the conditioning input.

The diffusion model is implemented using a U-Net architecture, in which the conditioning variable $c$ and the time variable $t$ are integrated with the PCA-based representations via cross-attention.

*3.1.1 Training.* In training the diffusion model $\mathcal{M}_\psi$, we first learn representations in the geometric space. Subsequently, we compute the Mean Squared Error (MSE) loss between the predictions $\hat{\mathcal{Y}}_s$ of the TSF model $\mathcal{F}_\theta$ for the selected samples $\mathcal{X}_s$, which are chosen by the Selector described later in Section 3.2, and the corresponding selected samples $\mathcal{Y}_s$, and update $\mathcal{M}_\psi$ accordingly. Through this two-stage learning method, the diffusion model learns the geometric features of the time-series data and how the $\mathcal{F}_\theta$ model interprets the time-series data.

In the initial training stage, the MSE loss is computed between the input data $x$ and the initial noise $x_t$, which has the same dimensionality as $x$ at time $t \in [0, 1]$, and learning is performed based on the resulting MSE loss. This formulation is derived from Eq. (6), which is introduced later.

$$v = \mathcal{M}_\psi(x_t, t, c), \text{ with } x_t = tx + (1-t)x_0 \tag{1}$$

$$Loss = \frac{1}{n}\sum_{i=1}^{n}((x - x_0) - v)^2 \tag{2}$$

$v$ is the expected velocity output by the diffusion model. Furthermore, to incorporate condition c into the training process, we applied Classifier-Free Guidance (CFG) [18]. In this study, 10% of the training is conducted without conditions, enabling the model to learn to emphasise features when conditions are present. For the condition-free state, we provide the diffusion model with an empty condition 0. In other words, the following equation is given:

$$v = \mathcal{M}_\psi(x_t, t, \emptyset) \tag{3}$$

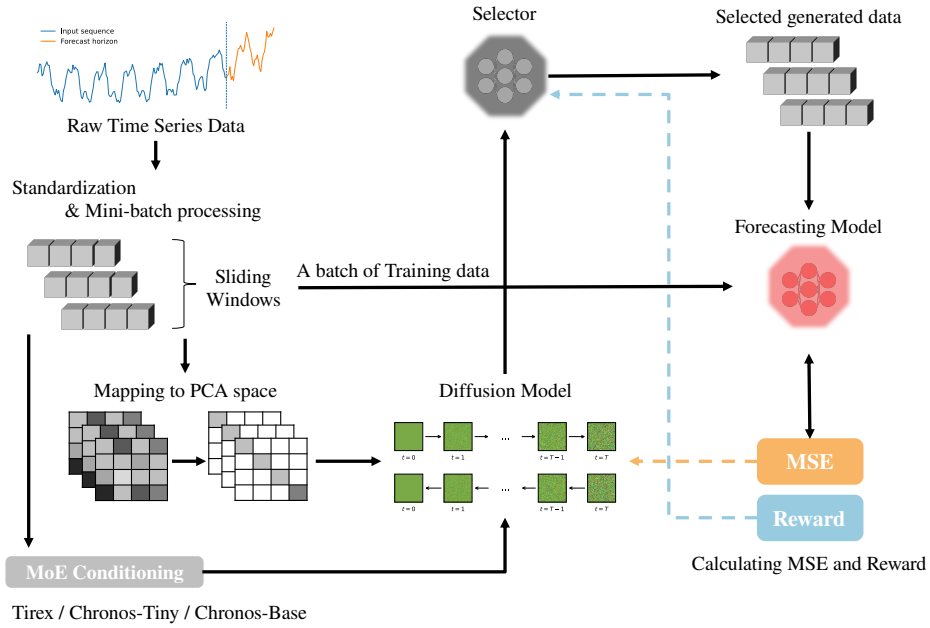


**Figure 2: Overview of DAD4TS. After dividing the time-series data into batches, it is mapped onto a PCA space and fed into the diffusion model for training. The Selector then determines whether the data generated by the diffusion model is suitable for use; the generated data deemed suitable, together with the real data, is fed into the forecasting model for training. Finally, the forecasting model and the diffusion model are updated using MSE loss, and the Selector is updated using the calculated reward.**

We optimize the diffusion model parameters using AdamW with a learning rate of $3 \times 10^{-3}$ and a weight decay of 0.1.

*3.1.2 Rectified Flow.* When applying diffusion models to time-series data generation, approaches based on DDPM [17] are commonly employed [24, 35, 49]. While the computational cost is not a major concern when data generation is performed only once, it becomes non-negligible when repeated generation is required, in which case the number of sampling steps emerges as a critical computational bottleneck. To address this issue, we adopt Rectified Flow [27], an approach that enables data generation with diffusion models using a small number of sampling steps.

Rectified Flow is formulated as an Ordinary Differential Equation (ODE) model that transports a distribution $\pi_0$ to $\pi_1$ by following a path that is as close as possible to a straight line between noise and clean data. In this study, the number of sampling steps was set to 20.

Given $X_0 \sim \pi_0$ and $X_1 \sim \pi_1$, the Rectified Flow induced by the pair $(X_0, X_1)$ is formulated as the ODE over time $t \in [0, 1]$. Let $Z_t$ be the values sampled from the distribution $\pi_t$. The dynamics are given by

$$dZ_t = v(Z_t, t)dt \tag{4}$$

which describes the ODE that transforms $Z_0$ into $Z_1$.

In addition, unlike conventional diffusion models, Rectified Flow generates data by solving a problem that estimates the expected velocity $v^X$ of a continuously differentiable stochastic process $\{X_t : t \in [0, 1]\}$, where the process is encouraged to follow, as closely as possible, straight-line trajectories from $X_0$ to $X_1$. Specifically, this requires computing

$$v^X(x, t) = \mathbb{E}\left[\dot{X}_t \mid X_t = x\right], \ \forall x \in \mathrm{supp}(X_t) \tag{5}$$

$$\min_v \int_0^1 \mathbb{E}\left[\|(X_1 - X_0) - v(X_t, t)\|^2\right] dt, \ \text{with} \ X_t = tX_1 + (1-t)X_0. \tag{6}$$

**Algorithm 1** Sampling Process

**Input:** Time $t_0 \in [0, 1]$, $t_1 \in [0, 1]$ $(t_0 < t_1)$, initial noise $x_0$, sampling step $\mathcal{T}$, conditioning variable $c$
**Models:** Diffusion model $\mathcal{M}_\psi$ **Initialize** $\psi$
**Output:** Sampling data $\hat{X}$

$h \leftarrow t_1 - t_0$
$v_0 \leftarrow \mathcal{M}_\psi(x_0, t_0, c)$
$x_\epsilon \leftarrow x_0 + hv_0$
$v_\epsilon \leftarrow \mathcal{M}_\psi(x_\epsilon, t_1, c)$
$x \leftarrow x_0 + \frac{h}{2}(v_0 + v_\epsilon)$
**for** $k = 1$ to $\mathcal{T}$ **do**
  $\tilde{h} \leftarrow t_{k+1} - t_k$
  $v_{k+1} \leftarrow x + \frac{3}{2}\tilde{h}v_\epsilon - \frac{1}{2}\tilde{h}v_0$
  $\tilde{v}_{k+1} \leftarrow x + \frac{1}{2}\tilde{h}(v_\epsilon + \mathcal{M}_\psi(v_{k+1}, t_{k+1}, c))$
  $v_0 \leftarrow v_\epsilon$
  $v_\epsilon \leftarrow \mathcal{M}_\psi(v_{k+1}, t_{k+1}, c)$
  $x \leftarrow \tilde{v}_{k+1}$
**end for**
$\hat{X} \leftarrow x$

*3.1.3 Sampling.* In this study, the Heun method and the Adams-Bashforth and Adams–Moulton methods were applied to ODEs as samplers. The Heun method is a technique that produces higher quality results than the Euler method, which has traditionally been widely used as a sampler. The Adams–Bashforth and Moulton methods are linear multistep methods, which are commonly employed to obtain numerical approximations of ODE solutions. Unlike single-step methods such as the Euler method, the Adams–Bashforth method computes the numerical approximation at step $k$ by retaining and referencing information from previous steps up to $k-1$. The Adams-Moulton method is used to correct the predictions derived using the Adams-Bashforth method. However, Adams methods cannot be applied at the first sampling step. Therefore, the Heun method is employed to compute the approximate solution at the initial sampling step. In this study, second-order Adams methods are adopted. The overall sampling procedure is shown in Algorithm 1.

Since CFG is applied in this study, both conditional and unconditional data are generated during the sampling process. Based on $\hat{X}$ computed in Algorithm 1, the sampling procedure that incorporates both cases is given by

$$\tilde{X} = (1 + w)\hat{X}_c - w\hat{X}_\emptyset, \quad \text{with } w = 1. \tag{7}$$

After that, $\tilde{X}$ undergoes a process that reverses the transformation performed in Section 3.1: first, Singular Value Decomposition (SVD) is applied to convert it into a two-dimensional vector; then, the inverse PCA transformation is performed to convert it into a Gram matrix; finally, the square roots of the values on the diagonal are taken to convert it back into the original time-series data.

## 3.2 DVRL-G

*3.2.1 Selector.* In the DVRL-G framework, we first introduce a Selector $\mathcal{S}_\phi$ that selects generated data used for training the TSF model $\mathcal{F}_\theta$. The Selector $\mathcal{S}_\phi$ is implemented as a Transformer-based model. As inputs to $\mathcal{S}_\phi$, we use the data $\tilde{X}$ generated by the diffusion model, which are split into an input segment $x_b$ and a forecast segment $y_b$ according to the input length and forecast length required by $\mathcal{F}_\theta$. In addition, we provide the MSE loss between the ground-truth $y_b$ and the forecast $\hat{y}_b$ obtained by feeding $x_b$ into $\mathcal{F}_\theta$. These inputs enable the Selector to capture both the generated data and how $\mathcal{F}_\theta$ interprets each generated sample, allowing it to assess the value of individual generated data points. Based on these inputs, $\mathcal{S}_\phi$ outputs a selection vector in the form of weights $W_{vec}$, which represent the value of each sample. This process is formulated as follows:

$$W_{vec} = \mathcal{S}_\phi(x_b, y_b, loss_b) \tag{8}$$

Next, we prepare a Sampler that maps the vector output by $\mathcal{S}_\phi$ to the range [0, 1]. This Sampler follows a Bernoulli distribution. The generated data selected by this Sampler is used to train $\mathcal{F}_\theta$. Table 2 summarizes the Selector's hyperparameter.

*3.2.2 Training.* For learning $\mathcal{S}_\phi$, we use the datasets $\mathcal{D}_{\text{train}}^{\text{win}}$ and $\mathcal{D}_{val}$ prepared in Section 3.1. In addition, we construct $\mathcal{D}_{val}^{1/2}$, which consists of only the latter half of the validation data. The rationale for constructing $\mathcal{D}_{val}^{1/2}$ is as follows. In a small-scale time-series setting, the Selector may over-adapt to limited validation data, causing the selection policy for generated data to depend on specific temporal patterns. To mitigate this risk, we divide the validation data into two disjoint subsets: $\mathcal{D}_{val}^{1/2}$, which is used for updating the Selector policy, the remaining validation data, which is not directly accessed by the Selector. While the Selector is updated based on the performance on $\mathcal{D}_{val}^{1/2}$, the best weights of the TSF model used for reward computation in updating the Selector are selected according to the predictive performance over the entire validation set $\mathcal{D}_{val}$. This design suppresses over-adaptation of the Selector to only the data that it directly optimizes.

**Table 2: Selector architecture Information.**

| *Hyperparameter* | |
|---|---|
| Encoder | Transformer Encoder |
| $d_{\text{model}}$ | 64 |
| # of Attention heads | 4 |
| # of Encoder layers | 2 |
| # of FFN dim. | 128 |
| Dropout | 0.1 |
| Activation | GELU |
| Pooling | Adaptive average |
| Classifier | Linear($2d_{\text{model}} \to 64$)→ReLU→Linear($64 \to 1$) |
| Output | $\sigma(\text{logit}/\tau)$, $\tau = 0.7$ |

Regarding the training procedure, first, starting from the $\mathcal{F}_\theta$, we train it for one epoch on $\mathcal{D}_{\text{train}}^{\text{win}}$ and compute the MSE loss $L_{base}$ on $\mathcal{D}_{val}^{1/2}$. Subsequently, $\mathcal{F}_\theta$ is trained using the generated data selected by the Selector together with a batch of training samples drawn from $\mathcal{D}_{\text{train}}^{\text{win}}$. After each batch update of $\mathcal{F}_\theta$, we compute the MSE loss $L_{val^{1/2}}$ on $\mathcal{D}_{val}^{1/2}$. The reward $\mathcal{R}$ is defined as $L_{base} - L_{val^{1/2}}$, and $\mathcal{S}_\phi$ is trained and updated so as to maximize $\mathcal{R}$.

To incorporate the influence of the selected samples into $\mathcal{R}$, we use the MSE loss $L_f$ obtained by feeding each generated sample into $\mathcal{F}_\theta$, as well as the log-probability under a Bernoulli distribution. When computing $L_f$, the loss is calculated for all generated samples regardless of whether they are selected by the Sampler. The loss $L_f$ reflects how $\mathcal{F}_\theta$ perceives the generated data (i.e., whether the data are easy or difficult to predict), while the log-probability indicates whether selected samples are valuable for training $\mathcal{F}_\theta$ and whether unselected samples are not valuable. Algorithm 2 shows the sequence of steps in the DVRL-G process. The entire training process of DAD4TS is DVRL-G itself. We optimize the Selector parameters with Adam, using a learning rate of $1 \times 10^{-3}$.

**Algorithm 2: DVRL-G Process**

**Input:** The datasets $\mathcal{D}_{train}$, $\mathcal{D}_{val}$ and $\mathcal{D}_{val}^{1/2}$, moving average of reward $\mathcal{R}_{ema} = 0$, MSE loss function $\mathcal{L}$

**Models:** A Selector $\mathcal{S}_\phi$, TSF model $\mathcal{F}_\theta$ and diffusion model $\mathcal{M}_\psi$
**Initialize** $\phi, \theta$ and $\psi$

**Optimizers:** The optimizers $O_\phi$, $O_\theta$ and $O_\psi$

**Hyperparameter:** Batch size $B$, number of epoch $N_{epoch}$, and number of mini-batches in $\mathcal{D}_{val}^{1/2}$ $Len_{\mathcal{D}_{val}^{1/2}}$

**Output:** Best parameters $\phi, \theta$ and $\psi$

1: **for** each mini-batch $(\mathbf{x}_{train}, \mathbf{y}_{train}) \subset \mathcal{D}_{train}$ **do**
2: Compute prediction $\hat{\mathbf{y}}_{train} \leftarrow \mathcal{F}_\theta(\mathbf{x}_{train})$
3: Compute loss $\mathcal{L}_{train}(\hat{\mathbf{y}}_{train}, \mathbf{y}_{train})$

Update $\theta$ using optimizer $O_\theta$ with $\mathcal{L}_{train}$
**end for**
**for** each mini-batch $(\mathbf{x}_{val^{1/2}}, \mathbf{y}_{val^{1/2}}) \subset \mathcal{D}_{val^{1/2}}$ **do**
Compute prediction $\hat{\mathbf{y}}_{val^{1/2}} \leftarrow \mathcal{F}_\theta(\mathbf{x}_{val^{1/2}})$
Compute loss $\mathcal{L}_{base}(\hat{\mathbf{y}}_{val^{1/2}}, \mathbf{y}_{val^{1/2}})$
**end for**

Initialize parameters $\theta$ of $\mathcal{F}_\theta$

**for** epoch = 1 to $N_{epoch}$ **do**
**for** each mini-batch $(\mathbf{x}_{train}, \mathbf{y}_{train}) \subset \mathcal{D}_{train}$ **do**
**Train Diffusion Model** $\mathcal{M}_\psi$ ▷ Section 3.1 and Section 3.1.1
Update $\psi$ using optimizer $O_\psi$ with MSE Loss
**end for**
Initialize buffers $\mathcal{X}_f \leftarrow [\,]$, $\mathcal{Y}_f \leftarrow [\,]$
**for** each mini-batch $(\mathbf{x}_{train}, \mathbf{y}_{train}) \subset \mathcal{D}_{train}$ **do**
$x_{gen}, y_{gen} \leftarrow$ **Data Generation** ▷ Section 3.1.3
add $x_{gen}$ and $y_{gen}$ to $\mathcal{X}_f, \mathcal{Y}_f$
**end for**
Compute prediction $\hat{\mathcal{Y}}_f \leftarrow \mathcal{F}_\theta(\mathcal{X}_f)$
Compute loss $\mathcal{L}_e(\hat{\mathcal{Y}}_f, \mathcal{Y}_f)$
**for** each mini-batch $(\mathbf{x}_{train}, \mathbf{y}_{train}) \subset \mathcal{D}_{train}$ **do**
$p \leftarrow \mathcal{S}_\phi(\mathcal{X}_f, \mathcal{Y}_f, L_e)$ ▷ Section 3.2.1
Sample mask $m \sim$ Bernoulli$(p)$ ▷ Section 3.2.1
$(\mathcal{X}_s, \mathcal{Y}_s) \leftarrow m \odot (\mathcal{X}_f, \mathcal{Y}_f)$
Compute prediction $\hat{\mathcal{Y}}_s \leftarrow \mathcal{F}_\theta(\mathcal{X}_s)$
Compute loss $\mathcal{L}_s(\hat{\mathcal{Y}}_s, \mathcal{Y}_s)$
Update $\psi$ using optimizer $O_\psi$ with $\mathcal{L}_s$ ▷ Section 3.1.1
Compute prediction $\hat{\mathbf{y}}_{train} \leftarrow \mathcal{F}_\theta(\mathbf{x}_{train})$
Compute loss $\mathcal{L}_{train}(\hat{\mathbf{y}}_{train}, \mathbf{y}_{train})$
$\tilde{\mathcal{L}}_{train} \leftarrow \mathcal{L}_{train} + \mathcal{L}_s$
Update $\theta$ using optimizer $O_\theta$ with $\tilde{\mathcal{L}}_{train}$
Initialize buffers $\mathcal{R}_c \leftarrow [\,]$
**for** each mini-batch $(\mathbf{x}_{val^{1/2}}, \mathbf{y}_{val^{1/2}}) \subset \mathcal{D}_{val}^{1/2}$ **do**
Compute prediction $\hat{\mathbf{y}}_{val^{1/2}} \leftarrow \mathcal{F}_\theta(\mathbf{x}_{val^{1/2}})$
Compute loss $\mathcal{L}_{val^{1/2}}(\hat{\mathbf{y}}_{val^{1/2}}, \mathbf{y}_{val^{1/2}})$
Compute reward $\mathcal{R} \leftarrow \mathcal{L}_{base} - \mathcal{L}_{val^{1/2}}$
add $\mathcal{R}$ to $\mathcal{R}_c$
**end for**
$\mathcal{R}_c \leftarrow \frac{\mathcal{R}_c}{Len_{\mathcal{D}_{val}^{1/2}} \odot (\bar{m}+\epsilon)}$
$\mathcal{R}_c \leftarrow \mathcal{R}_c - \mathcal{R}_{ema}$
$\mathcal{R}_{ema} \leftarrow 0.9\mathcal{R}_{ema} + 0.1\mathcal{R}_c$
Compute per sample quality $\mathcal{Q} \leftarrow$ Standardize$(\mathcal{R}_c \odot -(L_e))$
Compute log-probability $\log \Pi_\phi(m|p) \leftarrow m \odot \log(p + \epsilon) + (1 - m) \odot \log(1 - p + \epsilon)$
Compute final reward $\mathcal{R}_\phi \leftarrow -(\mathcal{Q} \odot \log \Pi_\phi(m|p))$
Update $\phi$ using optimizer $O_\phi$ with $\mathcal{R}_\phi$
**end for**
**end for**

## 4 Evaluation Experiment

In this section, we describe the experimental environment.

**Table 3: Dataset statistics and experimental configuration.**

| Dataset | Records | Frequency | Window | Batch size | $L_{\text{forecast}}$ |
|---|---|---|---|---|---|
| Employees | 427 | Monthly | 3 | 4 | {3,6,9,12} |
| Forest | 517 | unknown | - | 4 | {3,6,9,12} |
| German | 221 | Every three months | 3 | 4 | {3,6,9,12} |
| ILI | 966 | Weekly | 2–4 | 4 | {2,4,8,12} |
| Inventories | 402 | Monthly | 3 | 4 | {3,6,9,12} |
| Consump. | 96 | Yearly | 3 | 4 | {3,6,9,12} |

**Table 4: Shared training configuration for all downstream TSF models.**

| Hyperparameter | Value |
|---|---|
| Optimizer | AdamW |
| Learning rate | $3 \times 10^{-4}$ |
| Weight decay | 0.1 |
| Max epochs | 100 |
| Early stopping patience | 10 epochs |
| Early stopping metric | Validation MSE |
| Batch size | 4 |
| Random seed | 2025 |

### 4.1 Dataset

Wang et al. [41] implicitly defined a dataset of approximately 100 records as small-scale, and Gonen et al. [15], who proposed a data generation method for small-scale time-series data, utilised datasets of 1,000 records or fewer for time-series forecasting in few-shot tasks; consequently, we prepared a dataset of approximately 100–1,000 records. Furthermore, regarding the selection criteria for the data used, as recent models might have employed methods tuned specifically to popular benchmark datasets, we selected the widely used **ILI** dataset [50] alongside the relatively newly proposed benchmark dataset [14], specifically the univariate dataset **Inventories to Sales Ratio** [39], **German House Prices** [3], **Personal Consumption Expenditures** [37], **Employees Health Care** [38], and **Forest Fires** [6] from the relatively recently proposed benchmark dataset [14].

*4.1.1 Dataset Details.* Table 3 summarizes the datasets used in our experiments. Each dataset is split chronologically into training, validation, and test sets with a ratio of **6 : 2 : 2**. The training split is z-score normalized (zero mean, unit variance), and the same mean and standard deviation are applied to the validation and test splits. To avoid information leakage at split boundaries, the validation and test splits are extended backward by $L_{\text{input}}$ time steps so that the first sliding window of each split can look back into the preceding partition.

The input length is fixed at $L_{\text{input}} = 12$ for all datasets. Prediction lengths are chosen to reflect the seasonal periodicity of each dataset: for the five univariate datasets we use $L_{\text{forecast}} \in \{3, 6, 9, 12\}$, while for ILI (weekly data with an approximate 52-week annual cycle) we use $L_{\text{forecast}} \in \{2, 4, 8, 12\}$. The seasonal period parameter (`window`) is set per-dataset based on domain knowledge and is used by models that require explicit periodicity information (e.g. $S^2$IP-LLM, OLinear).

**Table 5: Average RMSE and DTW over multiple forecast lengths with input length fixed to 12. Results are averaged over forecast lengths $\in \{3, 6, 9, 12\}$ for all datasets, except ILI where $\in \{2, 4, 8, 12\}$. R represents RMSE, and D represents DTW. '-' indicates that the experiment could not be conducted in our experimental environment.**

| Dataset | Methods | Imp.(%) | | RNN | | LSTM | | Trans. | | iTrans. | | Patch. | | Times. | | S2IP. | | OLin. | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Rate ↑ | Mean ↓ | R ↓ | D ↓ | R ↓ | D ↓ | R ↓ | D ↓ | R ↓ | D ↓ | R ↓ | D ↓ | R ↓ | D ↓ | R ↓ | D ↓ | R ↓ | D ↓ |
| Employees | baseline | 0.00 | 0.00 | 1.32 | 10.1 | 1.38 | 10.5 | 1.30 | 9.67 | 0.153 | 1.12 | 0.144 | 0.953 | 0.192 | 1.49 | 0.184 | 1.31 | 0.201 | 1.47 |
| | TimeGAN | 48.4 | -2.60 | 1.30 | 9.89 | 1.45 | 10.9 | 1.43 | 10.8 | 0.150 | 1.05 | 0.142 | 0.948 | 0.169 | 1.23 | 0.182 | 1.34 | 0.148 | 0.942 |
| | TimeVAE | 48.4 | 0.269 | 1.34 | 10.4 | 1.33 | 10.0 | 1.28 | 9.55 | 0.169 | 1.25 | 0.147 | 0.986 | 0.175 | 1.32 | 0.202 | 1.48 | 0.184 | 1.30 |
| | SMFG | 59.4 | -13.6 | 1.36 | 10.6 | 1.55 | 11.8 | 1.54 | 11.7 | **0.115** | **0.734** | **0.117** | **0.725** | 0.132 | **0.839** | **0.114** | **0.716** | **0.131** | **0.855** |
| | Diffusion-TS | 64.1 | -9.20 | 1.38 | 10.8 | 1.41 | 10.8 | 1.69 | 12.8 | 0.130 | 0.881 | 0.126 | 0.793 | **0.130** | 0.860 | 0.130 | 0.856 | 0.154 | 1.17 |
| | ReAugment | 54.7 | -5.29 | 1.31 | 9.88 | 1.41 | 10.7 | 1.37 | 10.1 | 0.150 | 1.00 | 0.146 | 0.963 | 0.151 | 1.07 | 0.159 | 1.03 | 0.165 | 1.06 |
| | DiffAT | 54.7 | 0.0758 | 1.19 | 8.85 | 1.43 | 10.6 | 1.44 | 10.8 | 0.184 | 1.34 | 0.140 | 0.936 | 0.158 | 1.12 | 0.177 | 1.21 | 0.195 | 1.39 |
| | AutoDA | 81.2 | -28.1 | 0.588 | 4.00 | 0.453 | 3.03 | 0.799 | 4.61 | 0.149 | 0.985 | 0.136 | 0.846 | 0.151 | 1.03 | 0.148 | 0.987 | 0.162 | 1.06 |
| | DAD4TS | **89.1** | **-34.5** | **0.454** | **3.29** | **0.376** | **2.94** | **0.540** | **3.98** | 0.144 | 0.986 | 0.143 | 0.933 | 0.144 | 0.989 | 0.153 | 1.06 | 0.142 | 0.941 |
| Forest | baseline | 0.00 | 0.00 | 1.15 | 7.39 | 1.27 | 8.42 | **1.08** | 7.10 | 1.02 | 6.12 | 1.10 | 6.41 | **1.08** | 6.31 | 1.02 | **6.18** | **1.03** | **6.29** |
| | TimeGAN | 37.5 | 1.62 | 1.16 | 7.17 | 1.21 | 7.81 | 1.15 | 7.54 | 1.04 | 6.24 | **1.06** | **6.22** | 1.09 | 6.33 | 1.08 | 6.59 | 1.05 | 6.73 |
| | TimeVAE | 35.9 | 1.11 | 1.14 | 7.24 | 1.24 | 8.25 | 1.14 | 7.41 | 1.02 | 6.14 | 1.12 | 6.41 | 1.12 | 6.34 | 1.03 | 6.32 | 1.05 | 6.33 |
| | SMFG | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| | Diffusion-TS | **45.3** | -1.27 | 1.08 | 6.98 | 1.11 | 7.36 | 1.13 | 7.49 | 1.02 | 6.22 | 1.10 | 6.46 | 1.09 | **6.18** | **1.02** | 6.20 | 1.05 | 6.66 |
| | ReAugment | 39.1 | 0.790 | 1.21 | 7.53 | 1.15 | 7.71 | 1.09 | **7.03** | 1.06 | 6.27 | 1.09 | 6.34 | 1.12 | 6.35 | 1.04 | 6.18 | 1.06 | 6.62 |
| | DiffAT | 37.5 | 1.78 | 1.20 | 7.48 | 1.27 | 8.46 | 1.10 | 7.29 | **1.00** | **6.04** | 1.10 | 6.53 | 1.10 | 6.44 | 1.05 | 6.28 | 1.07 | 6.77 |
| | AutoDA | 32.8 | 1.24 | 1.20 | 7.51 | 1.24 | 8.22 | 1.10 | 7.28 | 1.05 | 6.20 | 1.09 | 6.26 | 1.11 | 6.22 | 1.05 | 6.27 | 1.06 | 6.66 |
| | DAD4TS | 45.3 | **-2.33** | **1.05** | **6.66** | **1.06** | **6.83** | 1.10 | 7.32 | 1.03 | 6.24 | 1.10 | 6.52 | 1.09 | 6.40 | 1.04 | 6.21 | 1.05 | 6.46 |
| German | baseline | 0.00 | 0.00 | 1.20 | 8.42 | 1.44 | 10.2 | 1.34 | 9.46 | 1.34 | 9.77 | 1.47 | 10.8 | 1.37 | 9.90 | 1.48 | 11.3 | **1.39** | 10.2 |
| | TimeGAN | 46.9 | 2.74 | 1.19 | 8.56 | 1.45 | 10.00 | 1.30 | **8.98** | 1.35 | 9.56 | 1.47 | 10.7 | 1.35 | 9.73 | **1.33** | 9.67 | 1.59 | 11.7 |
| | TimeVAE | 53.1 | 1.20 | 1.18 | 8.36 | 1.41 | 9.74 | 1.33 | 9.48 | 1.37 | 10.00 | 1.44 | 10.6 | 1.35 | **9.63** | 1.49 | 10.2 | 1.52 | 11.6 |
| | SMFG | 45.3 | 1.44 | 1.14 | 8.02 | 1.52 | 11.0 | 1.32 | 9.21 | 1.41 | 10.2 | 1.41 | 10.4 | 1.46 | 10.8 | 1.40 | 10.2 | 1.43 | 10.5 |
| | Diffusion-TS | 39.1 | 3.62 | 1.18 | 8.07 | 1.42 | 10.00 | **1.28** | 9.26 | 1.42 | 10.4 | 1.49 | 10.8 | 1.39 | 10.2 | 1.43 | 10.0 | 1.51 | 11.2 |
| | ReAugment | 60.9 | 2.14 | 1.18 | 8.43 | 1.42 | 9.78 | 1.32 | 9.11 | 1.33 | **9.54** | 1.53 | 11.0 | 1.41 | 10.0 | 1.33 | **9.39** | 1.51 | 11.3 |
| | DiffAT | 62.5 | -0.937 | 1.18 | 8.23 | 1.46 | 10.2 | 1.33 | 9.11 | 1.35 | 9.93 | 1.40 | 10.5 | 1.38 | 10.0 | 1.38 | 10.0 | 1.41 | 10.3 |
| | AutoDA | 46.9 | 0.305 | 1.24 | 8.88 | 1.42 | 9.71 | 1.34 | 9.72 | 1.39 | 10.1 | 1.51 | 11.3 | 1.38 | 9.89 | 1.42 | 10.2 | 1.47 | 11.1 |
| | DAD4TS | **71.9** | **-3.42** | **1.09** | **7.69** | **1.36** | **9.55** | 1.34 | 9.88 | **1.32** | 9.73 | **1.36** | **9.88** | **1.32** | 9.70 | 1.38 | 10.4 | 1.43 | **10.2** |
| ILI | baseline | 0.00 | 0.00 | 1.83 | 12.5 | 1.81 | 12.1 | 1.98 | 13.3 | 0.567 | 3.66 | 0.580 | 3.72 | 0.584 | 3.70 | 0.601 | 3.92 | 0.614 | 4.12 |
| | TimeGAN | 37.5 | 0.865 | 1.83 | 12.5 | 1.84 | 12.5 | 2.06 | 14.0 | 0.560 | 3.57 | 0.568 | 3.66 | 0.597 | 3.81 | 0.573 | **3.61** | 0.601 | 3.99 |
| | TimeVAE | 46.9 | -0.568 | 1.70 | 11.7 | 1.83 | 12.0 | 1.91 | 12.5 | 0.588 | 3.83 | 0.592 | 3.84 | 0.629 | 4.07 | 0.587 | 3.74 | 0.590 | 3.85 |
| | SMFG | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| | Diffusion-TS | 50.0 | -0.833 | 1.76 | 12.3 | 1.72 | 11.8 | 2.03 | 13.5 | 0.570 | 3.66 | 0.594 | 3.88 | 0.607 | 3.96 | 0.582 | 3.76 | 0.589 | 4.01 |
| | ReAugment | 28.1 | 1.25 | 1.80 | 12.2 | 1.81 | 12.1 | 1.99 | 12.7 | 0.584 | 3.80 | 0.597 | 3.91 | 0.607 | 3.91 | 0.599 | 3.79 | 0.591 | 3.91 |
| | DiffAT | 43.8 | 1.46 | 1.90 | 13.2 | 1.86 | 12.3 | 2.02 | 13.6 | 0.579 | 3.77 | 0.594 | 3.79 | 0.615 | 3.91 | 0.597 | 3.91 | 0.563 | 3.73 |
| | AutoDA | 76.6 | -11.3 | 1.44 | 8.86 | 1.27 | **7.03** | 1.48 | 9.84 | 0.550 | **3.45** | 0.544 | 3.43 | 0.554 | 3.50 | **0.573** | 3.64 | 0.576 | 3.76 |
| | DAD4TS | **96.9** | **-20.4** | **1.13** | **7.38** | **1.07** | 7.13 | **1.12** | **7.41** | **0.539** | 3.45 | **0.535** | **3.40** | **0.534** | **3.26** | 0.575 | 3.66 | **0.536** | **3.43** |
| Inventories | baseline | 0.00 | 0.00 | 0.748 | 5.35 | 0.719 | 5.20 | 0.859 | 6.24 | 0.552 | 3.78 | **0.548** | **3.64** | **0.567** | 3.83 | **0.524** | **3.47** | 0.561 | 3.82 |
| | TimeGAN | 43.8 | 1.14 | 0.735 | 5.25 | 0.784 | 5.75 | 0.818 | 6.26 | **0.548** | **3.73** | 0.572 | 3.80 | 0.570 | 3.89 | 0.569 | 3.83 | **0.527** | 3.60 |
| | TimeVAE | 40.6 | -0.186 | 0.741 | 5.46 | 0.730 | 5.50 | 0.730 | 5.50 | 0.571 | 3.98 | 0.579 | 3.92 | 0.584 | 3.93 | 0.557 | 3.75 | 0.566 | 3.99 |
| | SMFG | 35.9 | 3.62 | 0.755 | 5.45 | 0.756 | 5.21 | 0.897 | 6.28 | 0.551 | 3.74 | 0.584 | 3.91 | 0.584 | 3.98 | 0.532 | 3.68 | 0.577 | 4.00 |
| | Diffusion-TS | 26.6 | 6.29 | 0.816 | 6.18 | 0.839 | 6.48 | 0.908 | 6.68 | 0.568 | 3.93 | 0.564 | 3.81 | 0.594 | 4.07 | 0.553 | 3.86 | 0.558 | 3.85 |
| | ReAugment | 26.6 | 4.60 | 0.735 | 5.22 | 0.746 | 5.38 | 0.865 | 6.50 | 0.592 | 4.11 | 0.585 | 3.97 | 0.598 | 4.09 | 0.573 | 3.92 | 0.575 | 4.01 |
| | DiffAT | 45.3 | -1.48 | 0.730 | 5.24 | 0.753 | 5.39 | 0.725 | 5.44 | 0.559 | 3.80 | 0.569 | 3.86 | 0.580 | 4.02 | 0.538 | 3.59 | 0.529 | **3.54** |
| | AutoDA | 60.9 | -4.90 | 0.738 | 5.72 | 0.637 | 4.60 | 0.687 | 5.31 | 0.554 | 3.76 | 0.558 | 3.77 | 0.568 | **3.77** | 0.531 | 3.50 | 0.550 | 3.71 |
| | DAD4TS | **62.5** | **-12.8** | **0.469** | **3.23** | **0.547** | **4.20** | **0.508** | **3.57** | 0.552 | 3.77 | 0.566 | 3.81 | 0.586 | 3.97 | 0.551 | 3.78 | 0.560 | 3.92 |
| Consump. | baseline | 0.00 | 0.00 | 4.71 | 36.6 | 4.58 | 35.7 | 4.51 | 34.9 | 0.342 | 1.46 | 0.398 | 1.95 | 0.394 | 1.51 | 0.688 | 3.95 | 0.529 | 2.95 |
| | TimeGAN | 37.5 | 15.2 | 5.03 | 40.5 | 5.99 | 47.7 | 4.74 | 38.1 | 0.379 | 1.55 | 0.463 | 2.44 | 0.397 | 1.74 | 0.736 | 3.75 | 0.552 | 3.50 |
| | TimeVAE | 51.6 | 7.52 | 4.72 | 36.5 | 4.51 | **35.1** | 4.37 | 34.0 | 0.475 | 2.02 | 0.462 | 2.41 | 0.435 | 1.70 | 0.709 | 4.02 | **0.342** | **1.62** |
| | SMFG | 40.6 | 10.4 | 4.75 | 37.0 | 5.11 | 41.6 | 4.36 | 33.5 | 0.328 | 1.39 | 0.575 | 3.29 | 0.390 | 1.68 | 0.711 | 4.23 | 0.638 | 5.31 |
| | Diffusion-TS | 46.9 | -1.62 | 5.03 | 39.2 | 4.82 | 38.0 | 4.86 | 38.2 | 0.336 | 1.42 | 0.387 | 1.95 | 0.344 | 1.55 | 0.401 | 1.76 | 0.480 | 2.81 |
| | ReAugment | 54.7 | 3.61 | 4.64 | **35.8** | 4.56 | 35.3 | 4.47 | 34.5 | 0.453 | 1.94 | **0.370** | **1.81** | 0.411 | 1.68 | 0.644 | 3.16 | 0.503 | 2.90 |
| | DiffAT | 42.2 | 2.09 | 4.80 | 37.5 | 4.59 | 35.8 | 4.48 | 34.5 | 0.429 | 1.77 | 0.406 | 2.09 | 0.348 | 1.54 | 0.569 | 3.06 | 0.452 | 2.18 |
| | AutoDA | **64.1** | **-5.42** | **4.32** | 36.1 | **4.22** | 35.2 | **3.88** | **31.3** | **0.312** | **1.31** | 0.469 | 2.42 | 0.370 | 1.52 | **0.391** | **1.68** | 0.570 | 3.40 |
| | DAD4TS | 43.8 | 2.82 | 5.09 | 40.8 | 5.71 | 43.2 | 5.41 | 42.4 | 0.344 | 1.40 | 0.435 | 2.22 | **0.323** | **1.39** | 0.429 | 2.26 | 0.554 | 3.11 |

## 4.2 TSF Models

In previous studies, evaluations often relied on only a few TSF models [9, 12, 15, 36, 46, 48, 49], which carried the risk of producing biased results. Therefore, in this paper, we conducted an evaluation using eight TSF models, ranging from conventional to state-of-the-art models. The time-series forecasting models include RNN, LSTM [19], Transformer [40], iTransformer [28], PatchTST [30], TimesNet [43], $S^2$IP-LLM [31], and OLinear [50].

Table 4 lists the hyperparameters shared across all eight downstream TSF models. All TSF models are trained using AdamW with weight decay 0.1. Early stopping monitors the validation MSE loss with a patience of 10 epochs.

## 4.3 Comparison Methods

To evaluate the proposed method, we selected TimeGAN [46], TimeVAE [9], SMFG [35], Diffusion-TS [49], ReAugment [48], DiffAT [47], and AutoDA-Timeseries [12] from previous studies as comparison methods. These methods selected include data augmentation methods that improve the accuracy of downstream tasks despite limited training data [35], methods that generate data closely resembling the distribution of the training data [9, 46, 49], methods that use reinforcement learning to generate data for downstream tasks [48], and methods that perform sequential data augmentation [12]. By comparing DAD4TS with these methods, we evaluate its ability to optimize data augmentation for small-scale time-series data.

**Table 6: Comparison table of average runtimes for DAD4TS and the baseline. The averages for all forecast lengths and all TSF models are recorded.**

| Dataset | Methods | Runtime (Sec.) |
|---|---|---|
| German | baseline | 18.3 |
| | DAD4TS | 638.0 |
| Consump. | baseline | 4.82 |
| | DAD4TS | 141.9 |

## 4.4 Implementation Details

All experiments are conducted on an Intel Xeon Platinum 8259CL CPU (4 vCPUs), 16 GB RAM, and one NVIDIA Tesla T4 GPU with 16 GB VRAM (CUDA 12.6). The codebase is implemented in Python 3.12 using PyTorch 2.6.0 and TensorFlow 2.19.0. A fixed random seed of **2025** is used throughout all experiments to ensure reproducibility.

# 5 Experimental Results

Table 5 reports the comprehensive experimental results of DAD4TS. The best results are highlighted in red, and the second-best results are highlighted in blue with underlines. The baseline indicates the accuracy of the predictive model when trained using only real data. Imp. Rate (%) indicates the percentage of cases in which the forecasting accuracy is improved across all settings, and Imp. Mean (%) denotes the average improvement in forecasting accuracy. The RMSE and DTW values listed in Table 5 are the results for the test dataset prepared in Section 4.1.1.

## 5.1 Model Analysis

*Distribution of generated data.* Figure 3 shows the data distributions of the samples generated by three representative baseline methods and by the proposed method. From Figure 3, we can see that the baseline methods overall generate data whose distribution is similar to that of the real data, whereas the proposed method generates data whose distribution is slightly different from the real data. This is presumably because, in DAD4TS, training and sampling with the diffusion model are performed on the geometric space constructed by PCA, which leads to generation that emphasizes features with particularly large variance in the time-series data. We also observe that the distributions of the data generated by the diffusion model differ to some extent across TSF models, and that the samples selected by the Selector as more valuable training data, as well as their quantities, also vary. These observations suggest that the Selector assesses the generated data as valuable for training in a manner tailored to each TSF model. In addition, the results confirm that not all generated data are used and only a subset is utilized; thus, the Selector can subsample generated data that would act as noise during training while selecting and training on data suited to the TSF model, which is considered to have improved the prediction accuracy of the TSF model as a result.

*Runtime.* We compare the runtime of the baseline method with that of the DAD4TS method. We used two datasets for this comparison. Table 6 shows the results. Table 6 indicates that the runtime of DAD4TS is significantly longer than that of the baseline. This is because DAD4TS trains and infers the diffusion model at every epoch, rather than generating data only once as in conventional methods, resulting in higher computational costs.

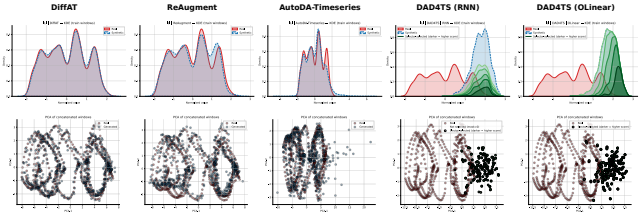


**Figure 3: Distribution of generated data for each representative method. Red indicates real data, and blue indicates generated data. Green indicates generated data selected by the Selector as suitable for training. The darker the color, the higher the data is rated as valuable for learning. In this figure, the ILI dataset is used, and the input and forecast lengths are fixed at 12.**

**Table 7: Analysis of information retention and reconstruction error during the generation process. Averaged over the eight forecasters and forecast lengths $\in \{3, 6, 9, 12\}$.**

| | | Information retention ↑ | | | Reconstruction | | |
|---|---|---|---|---|---|---|---|
| Dataset | Method | $\rho_{\text{out}}$ | $\rho_{\text{pca}}$ | $\rho_{\text{diag}}$ | $E_{\text{rec}} \downarrow$ | $r_{\text{shape}}$ | $\delta_{\text{diff}}$ |
| German | DAD4TS | 1.000 | 0.995 | 0.080 | 3.92 | 0.08 | 0.22 |
| Consump. | DAD4TS | 1.000 | 1.000 | 0.066 | 6.97 | 0.04 | 0.42 |

*Reconstruction Error Analysis.* In this study, various transformations are applied to the original time-series data. Among these transformations, we analyze the extent to which information is lost from the original time-series data at each transformation step. The results are presented in Table 7. The metric $\rho_{\text{out}}$ indicates that no information is lost, since it only involves computing the outer product of the time-series data. The metric $\rho_{\text{pca}}$ represents the proportion of information in the Gram matrix retained by the two-dimensional vectors obtained through PCA. These results suggest that, up to this stage, almost no information from the original time-series data is lost.

In contrast, $\rho_{\text{diag}}$ indicates the amount of information retained when only the diagonal components are extracted after computing the outer product of the two-dimensional vectors obtained by PCA. Although a substantial amount of information is lost at this step, each index value of the two-dimensional vectors obtained through PCA represents a coordinate in the PCA space. Therefore, because the calculation results between corresponding coordinates are preserved, the characteristics of the original data in the PCA space are still retained.

Next, we analyze the reconstruction error. $E_{\text{rec}}$ denotes the NRMSE between the original time-series data and the generated time-series data. As shown in Table 7, the generated time-series data are substantially different from the original time-series data. This observation is also supported by $r_{\text{shape}}$, which represents the correlation between the original and generated time-series data.

**Table 8: Ablation study: Average RMSE and DTW over multiple forecast lengths with input length fixed to 12. Results are averaged over forecast lengths $\in \{3, 6, 9, 12\}$ for all datasets, except ILI where $\in \{2, 4, 8, 12\}$. R represents RMSE, and D represents DTW. DAD4TS Once indicates a setting in which data generation is performed only once.**

| Dataset | Methods | Imp.(%) | | RNN | | LSTM | | Trans. | | iTrans. | | Patch. | | Times. | | S2IP. | | OLin. | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Rate ↑ | Mean ↓ | R ↓ | D ↓ | R ↓ | D ↓ | R ↓ | D ↓ | R ↓ | D ↓ | R ↓ | D ↓ | R ↓ | D ↓ | R ↓ | D ↓ | R ↓ | D ↓ |
| Employees | baseline | 0.00 | 0.00 | 1.32 | 10.1 | 1.38 | 10.5 | 1.30 | 9.67 | 0.153 | 1.12 | 0.144 | 0.953 | 0.192 | 1.49 | 0.184 | 1.31 | 0.201 | 1.47 |
| | DAD4TS Once | **92.2** | -30.4 | 0.532 | 3.97 | 0.643 | 4.91 | 0.633 | 4.89 | **0.141** | 1.00 | **0.139** | **0.910** | 0.147 | 1.01 | **0.152** | **1.03** | 0.154 | 1.04 |
| | DAD4TS w/o Selector | 59.4 | -3.24 | 1.35 | 10.3 | 1.35 | 10.2 | 1.27 | 9.60 | 0.156 | 1.12 | 0.143 | 0.954 | 0.154 | 1.10 | 0.210 | 1.51 | 0.166 | 1.22 |
| | DAD4TS w/o TSFMs | 81.2 | -33.3 | **0.449** | **3.16** | 0.405 | 3.12 | 0.585 | 4.54 | 0.145 | 1.01 | 0.143 | 0.943 | **0.138** | **0.922** | 0.155 | 1.09 | **0.138** | **0.900** |
| | DAD4TS | 89.1 | **-34.5** | 0.454 | 3.29 | **0.376** | **2.94** | **0.540** | **3.98** | 0.144 | **0.986** | 0.143 | 0.933 | 0.144 | 0.989 | 0.153 | 1.06 | 0.142 | 0.941 |
| Forest | baseline | 0.00 | 0.00 | 1.15 | 7.39 | 1.27 | 8.42 | 1.08 | 7.10 | 1.02 | **6.12** | 1.10 | 6.41 | 1.08 | 6.31 | **1.02** | **6.18** | **1.03** | **6.29** |
| | DAD4TS Once | 43.8 | -2.23 | 1.08 | 6.65 | **1.05** | **6.73** | 1.07 | **7.01** | 1.03 | 6.25 | 1.10 | 6.46 | 1.09 | 6.25 | 1.05 | 6.33 | 1.04 | 6.64 |
| | DAD4TS w/o Selector | 39.1 | 0.234 | 1.15 | 7.38 | 1.27 | 8.46 | 1.09 | 7.14 | **1.01** | 6.14 | **1.05** | **6.29** | **1.08** | **6.15** | 1.05 | 6.46 | 1.04 | 6.43 |
| | DAD4TS w/o TSFMs | 40.6 | **-2.35** | **1.04** | **6.49** | 1.06 | 6.84 | **1.07** | 7.12 | 1.04 | 6.34 | 1.11 | 6.62 | 1.10 | 6.41 | 1.03 | 6.28 | 1.06 | 6.39 |
| | DAD4TS | **45.3** | -2.33 | 1.05 | 6.66 | 1.06 | 6.83 | 1.10 | 7.32 | 1.03 | 6.24 | 1.10 | 6.52 | 1.09 | 6.40 | 1.04 | 6.21 | 1.05 | 6.46 |
| German | baseline | 0.00 | 0.00 | 1.20 | 8.42 | 1.44 | 10.2 | 1.34 | 9.46 | 1.34 | 9.77 | 1.47 | 10.8 | 1.37 | 9.90 | 1.48 | 11.3 | **1.39** | 10.2 |
| | DAD4TS Once | 62.5 | -1.32 | **1.05** | **7.18** | 1.40 | 10.1 | 1.30 | 9.46 | 1.38 | 10.1 | 1.44 | 10.5 | 1.37 | 10.0 | 1.35 | 9.86 | 1.40 | **10.1** |
| | DAD4TS w/o Selector | 40.6 | 3.92 | 1.22 | 8.23 | 1.50 | 10.3 | 1.43 | 10.7 | 1.40 | 10.2 | 1.43 | 10.4 | 1.37 | 9.86 | **1.33** | **9.17** | 1.54 | 11.7 |
| | DAD4TS w/o TSFMs | 60.9 | -3.21 | 1.08 | 7.54 | 1.38 | 9.83 | **1.27** | **9.27** | 1.35 | 10.1 | **1.33** | **9.70** | 1.37 | 10.1 | 1.39 | 9.98 | 1.46 | 11.2 |
| | DAD4TS | **71.9** | **-3.42** | 1.09 | 7.69 | **1.36** | **9.55** | 1.34 | 9.88 | **1.32** | **9.73** | 1.36 | 9.88 | **1.32** | **9.70** | 1.38 | 10.4 | 1.43 | 10.2 |
| ILI | baseline | 0.00 | 0.00 | 1.83 | 12.5 | 1.81 | 12.1 | 1.98 | 13.3 | 0.567 | 3.66 | 0.580 | 3.72 | 0.584 | 3.70 | 0.601 | 3.92 | 0.614 | 4.12 |
| | DAD4TS Once | 81.2 | -19.0 | **1.10** | 7.41 | 1.07 | **6.75** | 1.19 | 7.51 | 0.556 | 3.51 | 0.531 | 3.34 | 0.560 | 3.48 | **0.532** | **3.37** | 0.566 | 3.69 |
| | DAD4TS w/o Selector | 60.9 | -2.39 | 1.73 | 11.9 | 1.86 | 12.9 | 2.00 | 13.3 | 0.551 | 3.54 | 0.546 | 3.44 | 0.552 | 3.54 | 0.583 | 3.78 | 0.556 | 3.55 |
| | DAD4TS w/o TSFMs | **96.9** | **-20.5** | 1.11 | 7.42 | **1.04** | 6.96 | 1.16 | 7.70 | 0.547 | 3.50 | **0.516** | **3.21** | 0.542 | 3.44 | 0.563 | 3.54 | **0.531** | 3.44 |
| | DAD4TS | 96.9 | -20.4 | 1.13 | **7.38** | 1.07 | 7.13 | **1.12** | **7.41** | **0.539** | **3.45** | 0.535 | 3.40 | **0.534** | **3.26** | 0.575 | 3.66 | 0.536 | **3.43** |
| Inventories | baseline | 0.00 | 0.00 | 0.748 | 5.35 | 0.719 | 5.20 | 0.859 | 6.24 | 0.552 | 3.78 | **0.548** | **3.64** | 0.567 | 3.83 | 0.524 | **3.47** | 0.561 | 3.82 |
| | DAD4TS Once | 59.4 | -11.6 | **0.461** | **3.16** | **0.514** | **3.76** | 0.535 | 3.77 | 0.554 | 3.80 | 0.583 | 3.92 | 0.587 | 4.06 | 0.582 | 3.93 | 0.548 | 3.91 |
| | DAD4TS w/o Selector | 50.0 | 3.05 | 0.819 | 5.91 | 0.935 | 7.02 | 0.711 | 5.25 | **0.538** | **3.64** | 0.556 | 3.75 | **0.558** | **3.75** | 0.538 | 3.58 | **0.541** | **3.70** |
| | DAD4TS w/o TSFMs | **64.1** | **-13.0** | 0.477 | 3.28 | 0.539 | 4.04 | 0.550 | 3.94 | 0.554 | 3.84 | 0.571 | 3.84 | 0.574 | 3.81 | **0.516** | 3.49 | 0.547 | 3.80 |
| | DAD4TS | 62.5 | -12.8 | 0.469 | 3.23 | 0.547 | 4.20 | **0.508** | **3.57** | 0.552 | 3.77 | 0.566 | 3.81 | 0.586 | 3.97 | 0.551 | 3.78 | 0.560 | 3.92 |
| Consump. | baseline | 0.00 | 0.00 | **4.71** | 36.6 | **4.58** | **35.7** | 4.51 | 34.9 | 0.342 | 1.46 | 0.398 | 1.95 | 0.394 | 1.51 | 0.688 | 3.95 | 0.529 | 2.95 |
| | DAD4TS Once | 37.5 | 6.54 | 5.11 | 41.2 | 5.71 | 43.3 | 5.16 | 41.1 | 0.402 | 1.66 | **0.344** | **1.69** | 0.326 | 1.54 | 0.575 | 2.91 | **0.406** | **1.94** |
| | DAD4TS w/o Selector | 40.6 | 8.55 | 5.53 | 42.9 | 5.99 | 47.2 | **4.40** | **33.9** | 0.381 | 1.70 | 0.351 | 1.71 | 0.385 | 1.90 | 0.626 | 2.90 | 0.525 | 3.06 |
| | DAD4TS w/o TSFMs | **54.2** | 3.68 | 4.74 | **36.1** | 5.67 | 40.3 | 5.37 | 40.3 | **0.321** | **1.19** | 0.415 | 2.20 | **0.314** | **1.28** | 0.572 | 2.88 | 0.417 | 2.26 |
| | DAD4TS | 43.8 | **2.82** | 5.09 | 40.8 | 5.71 | 43.2 | 5.41 | 42.4 | 0.344 | 1.40 | 0.435 | 2.22 | 0.323 | 1.39 | **0.429** | **2.26** | 0.554 | 3.11 |

Furthermore, in the data generation flow employed in this study, we computed the NRMSE between the time-series data reconstructed through inverse transformation without using the diffusion model and the time-series data generated using the diffusion model. The resulting value was small, as indicated by $\delta_{\text{diff}}$. This suggests that the reconstruction error is strongly influenced by the data processing steps, whereas the diffusion model itself is capable of generating data while capturing the characteristics of the time-series data in the geometric space.

However, procedures such as dimensionality reduction by PCA and SVD are not specifically designed for time-series data. Therefore, structural information inherent to time-series data may have been lost. In future work, it will be necessary to investigate data processing methods that are more suitable for time-series data.

*Ablation Study.* Compared to many conventional methods, DAD-4TS sequentially generates synthetic data in tandem with the training of the TSF model. It also utilizes a Selector to determine whether the generated data is valuable for training the TSF model. In addition, DAD4TS uses TSFMs to condition the diffusion model during synthetic data generation. To investigate how each of these components affects the performance of DAD4TS, we compared DAD4TS with versions where specific conditions were restricted. Specifically, we prepared three ablated versions of DAD4TS: one in which data generation was performed only once, one in which the Selector was removed, and one in which the TSFM-based conditioning was removed from the diffusion model. The results are shown in Table 8. Table 8 confirms that the Selector is necessary for the sequential generation of synthetic data in sync with the training of the time-series forecasting model. This demonstrates that the Selector functions effectively even when the amount of trainable data is small. Furthermore, it confirms that in environments where the Selector is available, sequentially generating synthetic data in sync with the forecasting model—rather than performing a one-time static generation—enables the improvement of accuracy in downstream tasks. However, no significant difference was observed when comparing cases where TSFMs were used with those where they were not. This is thought to be because, in this study, TSFMs were applied to data they had not been trained on, and thus were unable to effectively capture the characteristics of data.

## 6 Conclusion

In this paper, we propose DAD4TS, a data augmentation framework for time-series forecasting in the small-scale data regime. This framework employs a 'joint training strategy' that simultaneously trains and updates, in an online manner, 'time-series forecasting models', a 'time-series data generation model' implemented based on diffusion models, and a Selector that continuously selects useful generated samples based on prediction-oriented rewards. We have demonstrated that, compared to existing data augmentation methods, this approach significantly improves the accuracy of time-series forecasting even under conditions of small-scale data. This framework imposes no restrictions on the time-series forecasting models. In future work, to address the limitations of the proposed method's applicability—restricted to univariate time-series forecasting tasks—we are going to aim to extend it to downstream tasks beyond time-series forecasting as well as to multivariate data settings.

## References


[1] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Syndar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. 2024. Chronos: Learning the Language of Time Series. *Transactions on Machine Learning Research* (2024). https://openreview.net/forum?id=gerNCVqqtR

[2] Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. 2025. TiRex: Zero-Shot Forecasting Across Long and Short Horizons with Enhanced In-Context Learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=v7UqniC9pF

[3] Bank for International Settlements. Manufacturers: Real Residential Property Prices for Germany [QDER628BIS]. 2026. https://fred.stlouisfed.org/series/QDER628BIS Retrieved February 20, 2026.

[4] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[5] Zheng Chen, Zichen Zou, Kewei Zhang, Xiongfei Su, Xin Yuan, Yong Guo, and Yulun Zhang. 2025. DOVE: Efficient One-Step Diffusion Model for Real-World Video Super-Resolution. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=DkJImu7t3A

[6] Paulo Cortez and Anbal Morais. 2007. Forest Fires. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5D88D.

[7] Bowen Deng, Chang Xu, Hao Li, Yu-hao Huang, Min Hou, and Jiang Bian. 2025. TarDiff: Target-Oriented Diffusion Guidance for Synthetic Electronic Health Record Time Series Generation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2* (Toronto ON, Canada) *(KDD '25)*. Association for Computing Machinery, New York, NY, USA, 474–485. doi:10.1145/3711896.3737147

[8] Junwei Deng, Chang Xu, Jiaqi W. Ma, and Jiang Bian. 2025. OATS: Online Data Augmentation for Time Series Foundation Models. In *Recent Advances in Time Series Foundation Models Have We Reached the 'BERT Moment'?* https://openreview.net/forum?id=kxdRkZqJLp

[9] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. 2022. TimeVAE: A Variational Auto-Encoder for Multivariate Time Series Generation. https://openreview.net/forum?id=VDdDvnwFoyM

[10] Tripti Dimri, Shamshad Ahmad, and Mohammad Sharif. 2020. Time series analysis of climate variables using seasonal ARIMA approach. *Journal of Earth System Science* 129, 1 (2020), 149.

[11] Chris Donahue, Julian McAuley, and Miller Puckette. 2019. Adversarial Audio Synthesis. In *ICLR*.

[12] Zijun Dou, Zhenhe Yao, Zhe Xie, Xidao Wen, Tong Xiao, and Dan Pei. 2026. AutoDA-Timeseries: Automated Data Augmentation for Time Series. In *The Fourteenth International Conference on Learning Representations*. https://openreview.net/forum?id=vTLmHAkoIW

[13] Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: discovering the long tail via influence estimation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 242, 11 pages.

[14] Denizalp Goktas, Amy Greenwald, Gerardo Riano-Briceno, Alexandra Magnusson, Alif Abdullah, and Beatriz de Lucio. 2025. TempusBench: An Evaluation Framework for Time-Series Forecasting. In *Recent Advances in Time Series Foundation Models Have We Reached the 'BERT Moment'?* https://openreview.net/forum?id=3fMa060Ag5

[15] Tal Gonen, Itai Pemper, Ilan Naiman, Nimrod Berman, and Omri Azencot. 2025. Time Series Generation Under Data Scarcity: A Unified Generative Modeling Approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=p324ryBKTc

[16] Kaveen Hiniduma, Suren Byna, and Jean Luca Bez. 2025. Data Readiness for AI: A 360-Degree Survey. *ACM Comput. Surv.* 57, 9, Article 219 (April 2025), 39 pages. doi:10.1145/3722214

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

[18] Jonathan Ho and Tim Salimans. 2021. Classifier-Free Diffusion Guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*. https://openreview.net/forum?id=qw8AKxfYbI

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

[20] Masato Ishii and Atsushi Sato. 2019. Training Deep Neural Networks with Adversarially Augmented Features for Small-scale Training Datasets. In *2019 International Joint Conference on Neural Networks (IJCNN)*. 1–8. doi:10.1109/IJCNN.2019.8852250

[21] Kevin Jiang, Weixin Liang, James Y Zou, and Yongchan Kwon. 2023. Opendataval: a unified benchmark for data valuation. *Advances in Neural Information Processing Systems* 36 (2023), 28624–28647.

[22] Hoang Anh Just, Feiyang Kang, Tianhao Wang, Yi Zeng, Myeongseob Ko, Ming Jin, and Ruoxi Jia. 2023. LAVA: Data Valuation without Pre-Specified Learning Algorithms. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=JJuP86nBl4q

[23] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[24] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. 2020. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761* (2020).

[25] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SJiHXGWAZ

[26] Hengbo Liu, Ziqing Ma, Linxiao Yang, Tian Zhou, Rui Xia, Yi Wang, Qingsong Wen, and Liang Sun. 2023. SADI: A Self-Adaptive Decomposed Interpretable Framework for Electric Load Forecasting Under Extreme Events. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1–5. doi:10.1109/ICASSP49357.2023.10096002

[27] Xingchao Liu, Chengyue Gong, and qiang liu. 2023. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=XVjTT1nw5z

[28] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2023. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625* (2023).

[29] Andrzej Maćkiewicz and Waldemar Ratajczak. 1993. Principal components analysis (PCA). *Computers Geosciences* 19, 3 (1993), 303–342. doi:10.1016/0098-3004(93)90090-R

[30] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730* (2022).

[31] Zijie Pan, Yushan Jiang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. 2024. $S^2$IP-LLM: Semantic space informed prompt learning with LLM for time series forecasting. In *Forty-first International Conference on Machine Learning*.

[32] Parsa Rahimi, Damien Teney, and Sébastien Marcel. 2025. AugGen: Synthetic Augmentation using Diffusion Models Can Improve Recognition. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=LuKlBH8DAT

[33] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.

[34] Kai Shu, Le Wu, Yuchang Zhao, Aiping Liu, Ruobing Qian, and Xun Chen. 2024. Data augmentation for seizure prediction with generative diffusion model. *IEEE Transactions on Cognitive and Developmental Systems* 17, 3 (2024), 577–591.

[35] Masahiro Suzuki, Megumi Kodaka, and Yusuke Fukazawa. 2024. Synthetic Mobility Feature Generation for Mental Health Prediction using Diffusion Models. In *2024 IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. 102–109. doi:10.1109/WI-IAT62293.2024.00022

[36] Hongming Tan, Ting Chen, Ruochong Jin, and Wai Kin Victor Chan. 2026. Data augmentation in time series forecasting through inverted framework. *Pattern Recognition Letters* 201 (2026), 152–159. doi:10.1016/j.patrec.2026.01.019

[37] U.S. Bureau of Economic Analysis. Personal Consumption Expenditures: Chain-type Price Index [DPCERG3A086NBEA]. 2026. https://fred.stlouisfed.org/series/DPCERG3A086NBEA Retrieved February 20, 2026.

[38] U.S. Bureau of Labor Statistics. All Employees, Health Care [CES6562000101]. 2026. https://fred.stlouisfed.org/series/CES6562000101 Retrieved February 20, 2026.

[39] U.S. Census Bureau. Manufacturers: Inventories to Sales Ratio [MNFCTRIRSA]. 2026. https://fred.stlouisfed.org/series/MNFCTRIRSA Retrieved February 20, 2026.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[41] Qiyao Wang, Ahmed Farahat, Chetan Gupta, and Shuai Zheng. 2021. Deep time series models for scarce data. *Neurocomputing* 456 (2021), 504–518. doi:10.1016/j.neucom.2020.12.132

[42] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. 2020. Time series data augmentation for deep learning: A survey. *arXiv preprint arXiv:2002.12478* (2020).

[43] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2022. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186* (2022).

[44] Tianlin Xu, Li Kevin Wenliang, Michael Munn, and Beatrice Acciaio. 2020. Cot-gan: Generating sequential data via causal optimal transport. *Advances in neural*

*information processing systems* 33 (2020), 8798–8809.

[45] Jinsung Yoon, Sercan O. Arik, and Tomas Pfister. 2020. Data valuation using reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 1005, 10 pages.

[46] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. 2019. *Time-series generative adversarial networks.* Curran Associates Inc., Red Hook, NY, USA.

[47] Yang Yu, Ruizhe Ma, Wenbo Gu, and Zongmin Ma. 2025. DiffAT: Effective data augmentation with diffusion models for time series forecasting. *Engineering Applications of Artificial Intelligence* 161 (2025), 112091. doi:10.1016/j.engappai.2025.112091

[48] Haochen Yuan, Yutong Wang, Yihong Chen, Yunbo Wang, and Xiaokang Yang. 2025. ReAugment: Model Zoo-Guided RL for Few-Shot Time Series Augmentation and Forecasting. arXiv:2409.06282 [cs.LG] https://arxiv.org/abs/2409.06282

[49] Xinyu Yuan and Yan Qiao. 2024. Diffusion-TS: Interpretable Diffusion for General Time Series Generation. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=4h1apFjO99

[50] Wenzhen Yue, Yong Liu, Hao Wang, Haoxuan Li, Xianghua Ying, Ruohao Guo, Bowei Xing, and Ji Shi. 2025. OLinear: A Linear Model for Time Series Forecasting in Orthogonally Transformed Domain. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=DAyKP1tvwI

[51] Yuren Zhang, Zhongnan Pu, and Lei Jing. 2025. A time-series data augmentation model through diffusion and transformer integration. *arXiv preprint arXiv:2505.03790* (2025).