DAD4TS: DATA-AUGMENTATION-ORIENTED DIFFUSION MODEL FOR TIME-SERIES FORECASTING WITH SMALL-SCALE DATA

Anonymous authors
Paper under double-blind review

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, designed for time-series forecasting with small-scale data. In DAD4TS, to generate samples that improve the predictive accuracy of a time-series forecasting model, the generator and the time-series forecasting model are trained simultaneously, and the data generator is updated while calculating the usefulness of the generated data. Furthermore, 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 latent space. We validated the effectiveness of DAD4TS with eight comparative methods through qualitative and quantitative experiments on six real-world datasets and six time-series models. As a result, DAD4TS showed improvements across multiple datasets.

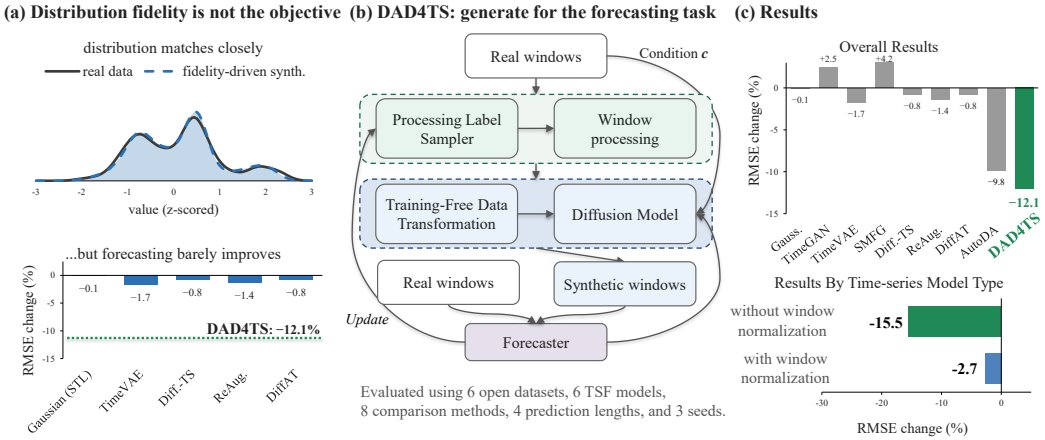


Figure 1: Distribution fidelity is not the right objective for augmentation in low-data time-series forecasting. **(a)** Synthetic windows whose marginal distribution matches the real data almost exactly still leave forecasting error essentially unchanged (top: real vs. TimeVAE on German; bottom: mean RMSE change of the five fidelity-matched baselines). **(b)** In DAD4TS, a generator is used that is trained in parallel with the time-series forecasting model during each epoch; the training objective is the product of the fidelity score and the prediction error of the time-series forecasting model relative to the real data. **(c)** In large-scale experiments, DAD4TS outperformed all comparison methods, and this performance improvement was particularly evident in time-series forecasting models that did not normalize each window internally.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

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 (Liu et al., 2023a), traffic forecasting (Li et al., 2018), and weather forecasting (Dimri et al., 2020). On the other hand, there are many cases in which sufficient training data cannot be collected (Gonen et al., 2025) in actual operation, and it is an important challenge to build effective forecasting models with small-scale data (Wang et al., 2021). In addressing such cases, data augmentation is frequently employed as a strategy when training data is small scale (Wen et al., 2020), and various methods have been proposed to date, not limited to time-series domain (Chawla et al., 2002; Ishii & Sato, 2019; Yoon et al., 2019; Desai et al., 2022; Suzuki et al., 2024; Yuan & Qiao, 2024; Rahimi et al., 2025). Existing research in this field has extensively examined generation-based data augmentation (Yoon et al., 2019; Desai et al., 2022; Suzuki et al., 2024; Yuan & Qiao, 2024; Yuan et al., 2025; Deng et al., 2025a;b; Yu et al., 2025; Gonen et al., 2025), and it has been demonstrated that, when data augmentation methods are designed appropriately, such augmentation can indeed improve predictive performance (Suzuki et al., 2024; Yuan et al., 2025; Deng et al., 2025a;b; Yu et al., 2025; Gonen et al., 2025; Tan et al., 2026; Dou et al., 2026). In this way, design approaches that take forecasting 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 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 forecasting 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 forecasting tasks. In fact, it was confirmed in several cases that generating and using data resembling real data (Figure 1 (a)) negatively affected forecasting tasks. Furthermore, several previous studies have confirmed that the performance of generative models declines when the amount of data available for training is limited (Desai et al., 2022; Gonen et al., 2025; Dou et al., 2026). Consequently, when training data is small scale, there are limitations to applying many of the previously reported studies to forecasting 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.

To address these challenges, 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:

- **Data Generation for Forecasting Tasks.** To generate data that improves the performance of the predictor, the generator is trained in parallel with the predictor at each epoch. Unlike conventional generators, this study does not aim to generate data that closely resembles real data; rather, it focuses on generating data that is difficult for predictive models to learn from.

- **Providing Insights into Time-Series Data Generation for Predictors.** We conduct qualitative and quantitative comparisons with previous research on generating data similar to real-world data to provide insights for future research on data generation. In particular, we analyze the relationship between the type of predictor and the distribution of the generated data.

- **A large, uniform evaluation.** We evaluated the performance of DAD4TS by conducting experiments under conditions involving six real-world datasets, six TSF models, eight comparison methods, and four forecast lengths.

2 Related Work

2.1 Generation of Time-Series Data

Existing methods for time-series data generation include GAN and VAE based approaches (Yoon et al., 2019; Donahue et al., 2019; Xu et al., 2020; Desai et al., 2022), as well as diffusion-model-based approaches (Suzuki et al., 2024; Yuan & Qiao, 2024; Shu et al., 2024; Zhang et al., 2025; Deng et al., 2025a;b; Yu et al., 2025; Gonen et al., 2025). 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. (Suzuki et al., 2024) 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. (Yuan et al., 2025) 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. (Deng et al., 2025b) 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. (Yu et al., 2025) 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. (Gonen et al., 2025) 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 (Ho et al., 2020), 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 (Yoon et al., 2020; Feldman & Zhang, 2020; Just et al., 2023; Jiang et al., 2023; Hiniduma et al., 2025). 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. (Just et al., 2023) 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. (Yoon et al., 2020) 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. (Yoon et al., 2020) 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. However, the amount of data available for this study is limited, making it difficult to train a model for data selection. Furthermore, given that the amount of data is already small, reducing it further should be avoided from a learning perspective. Therefore, in this study, we train the diffusion model to generate more samples that are difficult for the TSF model to predict.

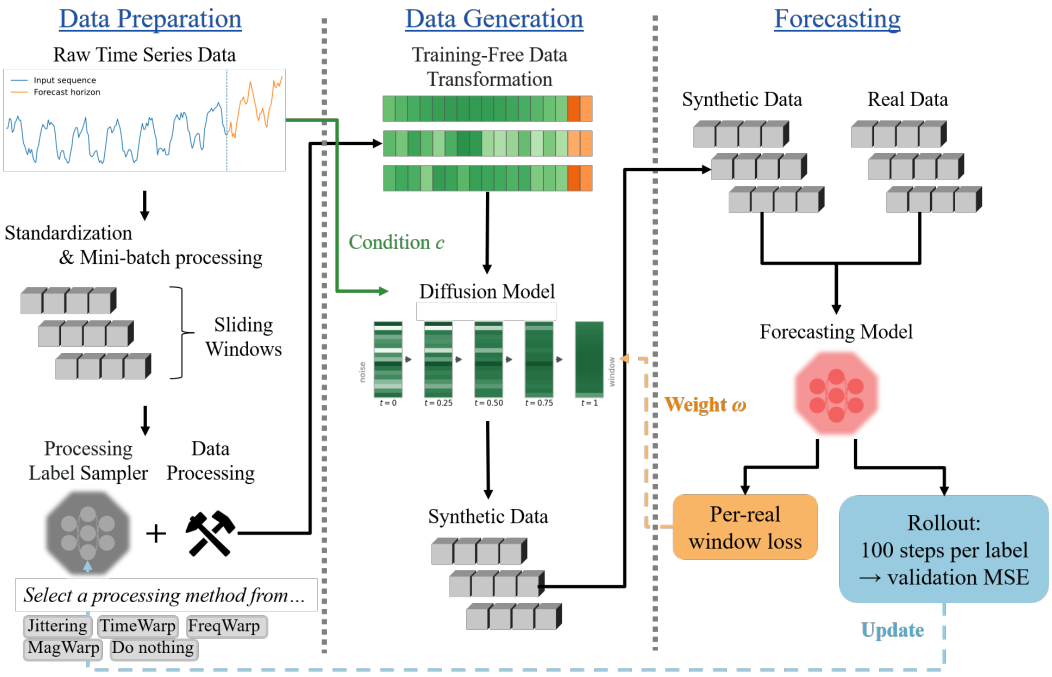


Figure 2: Overview of DAD4TS. After splitting the time-series data into batches, the Thompson sampler is used to select the appropriate data processing method for each data point and perform the processing. The data is then mapped to a latent space and fed into the diffusion model. Next, the generated data and the actual data are fed into the prediction model. Finally, the prediction model is updated using the MSE loss, the Thompson sampler is updated using the reward, and the diffusion model is updated using the MSE loss and the reconstruction error.

3 DAD4TS

DAD4TS simultaneously trains a predictor F_θ and a generator G_ψ, using forecast errors to determine what kind of output the generator should produce. Furthermore, inspired by AutoDA-Timeseries (Dou et al., 2026), it introduces four types of data processing to generate diverse data; to obtain data that is beneficial for the predictor F_θ, it uses the processing label sampler to determine the optimal processing label for each data set.

During each epoch, the following sequence of operations takes place: (1) selection of the processing method via the processing label sampler and data processing; (2) mapping of the data to the latent space; (3) data generation using the diffusion model; (4) training and updating of the TSF model using real and generated data; and (5) updating of the diffusion model and the processing label sampler using the MSE loss of the TSF model.

Figure 2 shows an overview of the DAD4TS.

3.1 Data Processing

The TSF models that normalize all input data remove the level and scale of the data before the model can recognize it. Since such models can still only recognize shape, synthetic data must differ from real data in this respect in order for the model to learn anything. Therefore, we incorporate data processing to alter the shape of the input data. Data processing is performed after the input data has been normalized.

Let $w \in \mathbb{R}^T$ be a window of length $T = L_{\text{input}} + L_{\text{forecast}}$. Furthermore, let $f(\cdot)$ be the standardization formula, $g(\cdot)$ the processing formula, $\tilde{f}(\cdot)$ the formula for eliminating the level and scale deviations caused by processing, and $f^{-1}(\cdot)$ the inverse standardization formula; this series of operations is

expressed by the following formula,

$$o(w) \;=\; f^{-1}\Big(\tilde{f}\big(g(f(w))\big)\Big). \tag{1}$$

In the processing formula $g(\cdot)$, one of five processing methods is assigned. One of these is the identity mapping. Each of the four processing steps is inspired by AutoDA-Timeseries (Dou et al., 2026) and involves Time Warp, Mag Warp, Freq Warp, and Jittering.

3.2 Training-Free Data Transformation

Conventionally, diffusion models first map input data into a latent space using a VAE (Kingma & Welling, 2013), and then learn a diffusion process on the latent representations to generate data (Rombach et al., 2022; Chen et al., 2025). 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 latent space without relying on deep learning–based methods.

We map it to a latent $z \in \mathbb{R}^{T+2}$ by

$$\underbrace{m = \tfrac{1}{T}\textstyle\sum_t w_t, \quad s = \mathrm{std}(w), \quad \hat{w} = (w-m)/s}_{\text{per-window affine normalization}} \qquad \underbrace{q = (\hat{w} - \bar{\mu})\, U^\top}_{\text{orthonormal basis}} \qquad \underbrace{z = [\, q,\ \tilde{m},\ \widetilde{\log s}\,] \oslash \sigma}_{\text{per-coordinate standardisation}} \tag{2}$$

where $U \in \mathbb{R}^{T\times T}$ is orthonormal, $\bar{\mu}$ is the training mean of $\hat{w}$, $\tilde{\cdot}$ denotes standardisation of the statistic channel, and σ is the per-coordinate training standard deviation. U is the eigenbasis of the covariance of $\hat{w}$, estimated once on the training windows, so all mini-batches and epochs share one coordinate system.

3.3 Diffusion Model

The generator G_ψ operates in the latent space of Eq. (2), i.e. on vectors of dimension $T+2$.

3.3.1 Rectified Flow

When applying diffusion models to time-series generation, approaches based on DDPM (Ho et al., 2020) are commonly employed (Kong et al., 2020; Suzuki et al., 2024; Yuan & Qiao, 2024). While the computational cost is not a major concern when generation is performed once, it becomes non-negligible when generation is repeated every epoch, in which case the number of sampling steps is the bottleneck. We therefore adopt Rectified Flow (Liu et al., 2023b), which transports a source distribution π_0 to a target π_1 along paths that are as close to straight as possible, and so admits a small number of sampling steps. In this study, the number of sampling steps was set to 20.

Given $z_0 \sim \pi_0$ and $z_1 \sim \pi_1$, 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 π_t. The dynamics are given by

$$dZ_t = v(z_t, t)dt \tag{3}$$

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^* of a continuously differentiable stochastic process $\{z_t : t \in [0,1]\}$, where the process is encouraged to follow, as closely as possible, straight-line trajectories from z_0 to z_1. Specifically, this requires computing

$$v^*(z,t) = \mathbb{E}[\dot{z}_t \mid z_t = z]\,,\ \forall x \in \mathrm{supp}(z_t) \tag{4}$$

$$\min_v \int_0^1 \mathbb{E}\left[\|(z_1 - z_0) - v(z_t,t)\|^2\right] dt, \ \text{ with } z_t = tz_1 + (1-t)z_0. \tag{5}$$

Furthermore, in this study, we design the velocity field to be represented by a closed-form expression

$$v^*(z,t) = a(t)\, z, \qquad a(t) = \frac{2t-1}{t^2+(1-t)^2} \tag{6}$$

and a residual $r_\psi(z,t,c,k)$; c is the conditioning vector and k is the processing label. This is because, when $z_1 \sim \mathcal{N}(0,I)$ and $z_0 \sim \mathcal{N}(0,I)$, the closed-form expression can be expressed in terms of t and z, allowing the network to focus solely on the residual that it needs to learn. As for $r_\psi(z,t,c,k)$, we initialize the output layer at the beginning. The conditioning vector c consists of embeddings extracted from TSFMs—specifically, Chronos-Tiny and Chronos-Base based on Chronos-Bolt (Ansari et al., 2024), as well as Tirex (Auer et al., 2025)—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.

3.3.2 TRAINING

At each step we draw $t \in [0,1]$, form $z_t = tz_1 + (1-t)z_0$, and minimise

$$\mathcal{L}_{\text{flow}} = \frac{1}{n}\sum_{i=1}^{n} \omega_i \big\| (z_1 - z_0) - \big(v^*(z_t,t) + r_\psi(z_t,t,c_i,k_i)\big) \big\|^2, \tag{7}$$

where z_1 is the latent of the *transformed* window $o_{k_i}(w_i)$ of Eq. (1), c_i is the conditioning vector of the *untransformed* window w_i, and k_i is a processing label that has been converted to a vector. Therefore, the generator is trained to generate post-processing time-series data based on the characteristics of the pre-processing real data. Furthermore, to train the diffusion model to generate certain samples with greater weight, the current loss of the predictor for window w_i is divided by its batch mean to calculate the weight ω_i, which is then multiplied by the loss. This ensures that resources are allocated to windows that the predictor is still mispredicting.

Choosing the label. At the end of each epoch, a pool consisting of up to 64 real windows is selected, and for each label, one synthetic window is generated for each entry in the pool using the same initial noise. As a result, the only difference between pools is the label. For each label, we then train a copy of the current TSF model on that pool's real windows together with its own synthetic windows, with the number of steps 100, the learning rate 3×10^{-3} and the batch order held identical across labels, and evaluate the copy on the validation dataset. Let the error be denoted by MSE_k; the observed value for label k is the logarithmic ratio relative to the identity label $k = 0$,

$$d'_k \;=\; \log \frac{\mathrm{MSE}_0 + \varepsilon}{\mathrm{MSE}_k + \varepsilon}, \tag{8}$$

where ε is 10^{-8}. This takes a positive value when the processing is effective. However, when conditions such as the dataset or the TSF model change, variations occur in the values of d'_k, and if these values are used as is, it becomes difficult to apply d'_k uniformly under all conditions. Therefore, we adjust the scale of d'_k using the Median Absolute Deviation (MAD), which is a robust measure against outliers. Specifically, we calculate the MAD using previously observed data and divide d'_k by $1.4826 \times$ MAD. Furthermore, to limit the influence of a single diverging rollout, the result is then winsorized: values beyond the 5 and 95 percentiles of that history are replaced by the boundary values. We write $\hat{d}_k$ for the observation after these two steps.

Each label carries a Normal-Inverse-Gamma posterior, $\hat{d}_k \sim \mathcal{N}(\mu_k, \sigma_k^2)$ with $\mu_k \mid \sigma_k^2 \sim \mathcal{N}(0, \frac{\sigma_k^2}{\kappa_0})$ and $\sigma_k^2 \sim \mathrm{InvGamma}(\alpha_0, \beta_0)$, which for the weighted count, sum and sum of squares (W_k, S_{1k}, S_{2k}) of its observations has

$$\kappa_k = \kappa_0 + W_k, \quad \mu_k = \frac{S_{1k}}{\kappa_k}, \quad \alpha_k = \alpha_0 + \frac{W_k}{2}, \quad \beta_k = \beta_0 + \frac{S_{2k} - \kappa_k \mu_k^2}{2}, \tag{9}$$

with $\kappa_0 = 1$, $\alpha_0 = 2$ and $\beta_0 = 10^{-2}$. Only the three sums (W_k, S_{1k}, S_{2k}) are carried from one epoch to the next: the prior is fixed, and Eq. (9) is re-evaluated from it whenever the posterior is needed. Because the generator and the TSF model both move, the sums of every label are discounted before that epoch's observation is added,

$$(W_k,\, S_{1k},\, S_{2k}) \;\leftarrow\; \gamma\,(W_k,\, S_{1k},\, S_{2k}) \;+\; \big(1,\, \hat{d}_k,\, \hat{d}_k^2\big), \qquad \gamma = 0.9, \tag{10}$$

so a label never accumulates more than about $\frac{1}{(1-\gamma)} = 10$ epochs of evidence. The prior does not appear in this update: it is never discounted and keeps the weight of a single observation throughout. At the start of the next epoch each training window draws its own label: for every k we sample

$$\tilde{\mu}_k \;=\; \mu_k + \sqrt{\frac{\beta_k}{\alpha_k \kappa_k}}\, T, \qquad T \sim t(2\alpha_k), \tag{11}$$

from the marginal posterior of μ_k, and take the label with the largest draw, or the identity mapping if no draw is positive. Here $t(\nu)$ denotes the Student distribution on ν degrees of freedom and T is a single draw from it. The first two epochs draw labels uniformly.

Algorithm 1 Sampling Process

Input: Time $t_0 \in [0,1]$, $t_1 \in [0,1]$ $(t_0 < t_1)$, initial noise z_0, sampling step $\mathcal{T}$, conditioning variable c, and processing label k
Models: Diffusion model $\mathcal{M}_\psi$ **Initialize** ψ
Output: Sampling data $\hat{Z}$

$h \leftarrow t_1 - t_0$
$v_0 \leftarrow \mathcal{M}_\psi(z_0, t_0, c, k)$
$z_\epsilon \leftarrow z_0 + h v_0$
$v_\epsilon \leftarrow \mathcal{M}_\psi(z_\epsilon, t_1, c, k)$
$z \leftarrow z_0 + \frac{h}{2}(v_0 + v_\epsilon)$
for $s = 1$ to $\mathcal{T}$ **do**
    $\tilde{h} \leftarrow t_{s+1} - t_s$
    $v_{s+1} \leftarrow z + \frac{3}{2}\tilde{h} v_\epsilon - \frac{1}{2}\tilde{h} v_0$
    $\tilde{v}_{s+1} \leftarrow z + \frac{1}{2}\tilde{h}(v_\epsilon + \mathcal{M}_\psi(v_{s+1}, t_{s+1}, c, k))$
    $v_0 \leftarrow v_\epsilon$
    $v_\epsilon \leftarrow \mathcal{M}_\psi(v_{s+1}, t_{s+1}, c, k)$
    $z \leftarrow \tilde{v}_{s+1}$
end for
$\hat{Z} \leftarrow z$

3.3.3 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 s by retaining and referencing information from previous steps up to $s-1$. 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

$$\tilde{Z} = (1+u)\hat{Z}_c - u\hat{Z}_\emptyset, \;\; \text{with}\, u = 1. \tag{12}$$

4 RESULTS

We compared DAD4TS with eight data augmentation techniques across six datasets, six TSF models, four forecast horizons, and three seeds. SMFG is the only exception, as it cannot be applied to two of the datasets. Appendix A describes the experimental design. The percentages indicate the improvement in RMSE. Table 1 shows the main comparison results. DAD4TS obtains the largest improvement, -12.05% against training without augmentation, followed by AutoDA at -9.83%.

Table 1: RMSE change relative to training without augmentation (%), lower is better. Green bold marks the best method in a column and blue the runner-up. *: paired t-test $p < 0.05$ against the no-augmentation baseline over the runs of that column.

Method	Employees	Forest	German	ILI	Invent.	Consum.	**All**
DAD4TS (ours)	-66.44^*	-1.06	-9.94^*	-22.69^*	-18.24^*	$+4.19^*$	-12.05^*
AutoDA	-52.40^*	$+0.57$	-1.65	-15.98^*	-11.80^*	-3.56^*	-9.83^*
TimeVAE	-6.33^*	-0.87	-1.23	$+1.34$	-7.06^*	-0.95^*	-1.72^*
ReAugment	-0.76	-0.95	-1.97^*	-0.29	-7.22^*	-0.34	-1.41^*
DiffAT	-2.92^*	-1.75^*	$+0.30$	$+0.33$	-4.23^*	$+0.20$	-0.76^*
Diffusion-TS	-3.48^*	-2.97^*	-3.52^*	-3.52^*	-3.85	$+4.75^*$	-0.78
Gaussian (STL)	-0.86	$+0.98$	-0.59	$+1.23$	-8.57^*	$+1.75^*$	-0.08
TimeGAN	$+2.72^*$	-0.58	$+0.29$	$+6.10^*$	$+5.52$	$+2.43$	$+2.45^*$
SMFG	-0.79	–	$+3.27^*$	–	-2.80	$+8.12^*$	$+4.17^*$

Table 2: The same comparison as Table 1, split by whether the forecaster normalizes each input window before it sees it.

Method	Employees	Forest	German	ILI	Invent.	Consum.	**All**
TSF models without window normalization (RNN, LSTM, Transformer)							
DAD4TS (ours)	-74.97^*	-2.50	-18.52^*	-28.69^*	-26.57^*	$+5.72^*$	-15.49^*
AutoDA	-57.72^*	$+0.02$	-2.77^*	-20.13^*	-19.19^*	-2.60	-12.74^*
TimeVAE	-7.30^*	-1.71	-1.45	$+1.04$	-12.54^*	-1.24^*	-2.55^*
ReAugment	-0.29	-2.42^*	-2.68^*	-1.02	-12.89^*	-0.19	-1.90^*
DiffAT	-3.54^*	-3.31^*	$+0.08$	$+0.17$	-6.43^*	$+0.21$	-1.17^*
Diffusion-TS	-3.80^*	-5.98^*	-5.63^*	-5.45^*	-6.01	$+5.01^*$	-1.09
Gaussian (STL)	-1.75^*	$+1.25$	-0.96	$+1.70^*$	-10.95^*	$+2.75^*$	$+0.35$
TimeGAN	$+2.70$	-0.98	$+0.48$	$+7.86^*$	$+9.57^*$	$+4.15^*$	$+3.96^*$
SMFG	$+1.55$	–	$+6.94^*$	–	-4.62	$+8.83^*$	$+5.90^*$
TSF models with window normalization (iTransformer, PatchTST, TimesNet)							
DAD4TS (ours)	$+4.70^*$	$+0.68^*$	-1.68	-4.12^*	-4.31^*	-15.16^*	-2.70^*
AutoDA	-8.06^*	$+1.23^*$	-0.57	-3.15^*	$+0.55$	-15.69^*	-1.94^*
TimeVAE	$+1.76$	$+0.14$	-1.02	$+2.27^*$	$+2.12^*$	$+2.65$	$+0.56$
ReAugment	-4.69^*	$+0.81^*$	-1.29	$+1.96^*$	$+2.26^*$	-2.22	-0.06
DiffAT	$+2.20$	$+0.13$	$+0.51$	$+0.81$	-0.56	$+0.02$	$+0.34$
Diffusion-TS	-0.81	$+0.65$	-1.49	$+2.45^*$	-0.25	$+1.46$	$+0.05$
Gaussian (STL)	$+6.54^*$	$+0.65^*$	-0.24	-0.20	-4.58^*	-10.99^*	-1.26^*
TimeGAN	$+2.88$	-0.09	$+0.11$	$+0.66$	-1.27	-19.30^*	-1.65^*
SMFG	-20.29^*	–	-0.27	–	$+0.23$	-0.89	-1.49

Per dataset, DAD4TS improves five of the six, and the improvement is significant on four of them. These results are also influenced by whether or not the TSF model normalizes windows internally; we discuss this point in Section 4.1.

4.1 Results by TSF Model Type

The improvement is not spread evenly over the TSF models. Three of the six models in this comparison (iTransformer, PatchTST and TimesNet) normalize every input window before processing it and undo the normalization on the output; the other three consume the window as it is. Table 2 repeats the comparison of Table 1 for the two families. Pooled over the runs, DAD4TS improves the three models without internal normalization by 15.49% and the three with it by 2.70%. AutoDA-Timeseries shows the same split (12.74% and 1.94%). In this way, the difference in the effect of the generated data is thought to stem from the fact that TSF models with internal normalization have higher initial prediction accuracy than TSF models without internal normalization. On the other hand, among the datasets where prediction accuracy improved for TSF models with internal normalization, there were also datasets where prediction accuracy decreased for TSF models without internal normalization (and vice versa). This indicates that there are favorable and unfavorable

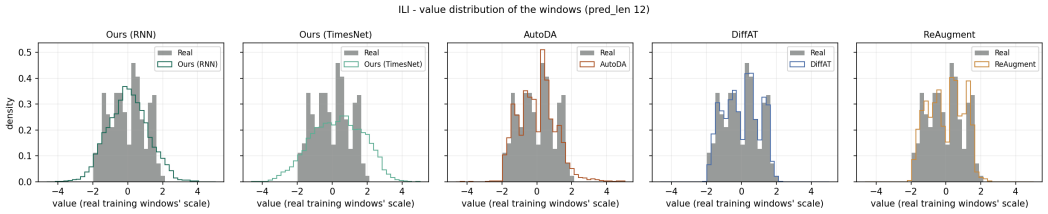


Figure 3: Employees at the longest horizon: the values of the synthetic windows (coloured outline) against the values of the real training windows (grey), pooled over every position of every window and put on the scale of the real windows. The two panels on the left are DAD4TS as it was trained with two different forecasters; the three on the right are comparison methods whose synthetic set does not depend on the forecaster. The comparison method falls within the distribution of the real data, but DAD4TS does not, and its shape differs depending on the two prediction models. The other five datasets are in Appendix D.

combinations of TSF model types and datasets, suggesting that it is necessary to evaluate various patterns based on the specific dataset being used to identify the optimal combination.

4.2 Analysis of Generated Samples

For each dataset, we compare the generated data with the real data in two ways: one is by examining the distribution of values (Figure 3), and the other is by analyzing the time series data unfolded along the time axis. For DAD4TS, we visualize synthetic samples generated under two forecasters, RNN and TimesNet. This is because the generator is trained alongside the prediction model. The three baselines generate one synthetic dataset per cell, regardless of the prediction model.

Table 1 and Figure 3 suggest that the degree of approximation to real data and downstream performance improvements point in opposite directions. In the ILI dataset, DiffAT and ReAugment, which generate data with distributions closer to real data - changed the RMSE by 0.33% and -0.29%, respectively. AutoDA-Timeseries, which generates data that deviates slightly from the real data, further improves the RMSE by -15.98%. On the other hand, DAD4TS, which generates data that deviates the most from the real data compared to these comparison methods, improves the RMSE by -22.69%; this suggests that, in the task of improving the prediction accuracy of TSF models, generated data that closely matches the real data does not contribute significantly to the TSF model.

Furthermore, the results from the RNN and TimesNet in Figure 3 confirm that the distribution of data generated by the TSF model differs somewhat. This indicates that the features of the generated data that improve prediction accuracy differ depending on the TSF model, highlighting a significant difference from methods—such as the comparison method—that generate data without aligning it with the TSF model's training. However, since this study involved a small dataset, future verification will need to be conducted using a much larger dataset.

5 Conclusion

In this study, we introduced DAD4TS, a data augmentation method that generates samples to improve the accuracy of the TSF model when only a small-scale time-series data is available. Through our experiments, we demonstrated that differences in the effectiveness of generated data arise from the characteristics of the TSF model and its inherent high inference capability. We also showed that by training the generator to improve the TSF model's predictive accuracy—rather than aiming to align with the distribution of real data—we can generate data that supports the TSF model. In future work, we plan to evaluate DAD4TS in scenarios where large-scale data is available, and to implement and evaluate DAD4TS for downstream tasks other than prediction tasks, such as AutoDA-Timeseries.

AI USE STATEMENT

In this work, we used generative AI tools for implement methods and assist with translation (tasks with required disclosure). We have not used generative AI tools for [other tasks with required disclosure], and [the rest of the required disclosure tasks] are not applicable to this work. Additionally, we used generative AI tools for create or modify scientific figures or images and create or edit software code(tasks with recommended disclosure). We have reviewed all AI-assisted work. Elaborate. "The code generated by the LLM was verified and tested by a single author. In addition, the experimental environment and procedures were controlled by the author, and the LLM created and edited the code within that controlled environment." We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

An overview of the proposed method, DAD4TS, is shown in Figure 2. It illustrates the entire workflow from data input to prediction. Furthermore, each method within DAD4TS is described in Section 3. The experimental environment is described in Appendix A. The code will be shared with the reviewers in a ZIP file during the peer review period and will be made publicly available on the Internet once the paper is accepted. This is in accordance with the policies of my organization.

REFERENCES

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. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=gerNCVqqtR.

Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. 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*, 2025. URL https://openreview.net/forum?id=v7UqniC9pF.

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

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

Zheng Chen, Zichen Zou, Kewei Zhang, Xiongfei Su, Xin Yuan, Yong Guo, and Yulun Zhang. DOVE: Efficient one-step diffusion model for real-world video super-resolution. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=DkJImu7t3A.

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

Bowen Deng, Chang Xu, Hao Li, Yu-hao Huang, Min Hou, and Jiang Bian. 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*, KDD '25, pp. 474–485, New York, NY, USA, 2025a. Association for Computing Machinery. ISBN 9798400714542. doi: 10.1145/3711896.3737147. URL https://doi.org/10.1145/3711896.3737147.

Junwei Deng, Chang Xu, Jiaqi W. Ma, and Jiang Bian. OATS: Online data augmentation for time series foundation models. In *Recent Advances in Time Series Foundation Models Have We Reached the 'BERT Moment'?*, 2025b. URL https://openreview.net/forum?id=kxdRkZqJLp.

Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. TimeVAE: A variational auto-encoder for multivariate time series generation, 2022. URL https://openreview.net/forum?id=VDdDvnwFoyM.

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

Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. In *ICLR*, 2019.

Zijun Dou, Zhenhe Yao, Zhe Xie, Xidao Wen, Tong Xiao, and Dan Pei. AutoDA-timeseries: Automated data augmentation for time series. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vTLmHAkoIW.

Vitaly Feldman and Chiyuan Zhang. 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*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Denizalp Goktas, Amy Greenwald, Gerardo Riano-Briceno, Alexandra Magnusson, Alif Abdullah, and Beatriz de Lucio. Tempusbench: An evaluation framework for time-series forecasting. In *Recent Advances in Time Series Foundation Models Have We Reached the 'BERT Moment'?*, 2025. URL https://openreview.net/forum?id=3fMa060Ag5.

Tal Gonen, Itai Pemper, Ilan Naiman, Nimrod Berman, and Omri Azencot. Time series generation under data scarcity: A unified generative modeling approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=p324ryBKTc.

Kaveen Hiniduma, Suren Byna, and Jean Luca Bez. Data readiness for ai: A 360-degree survey. *ACM Comput. Surv.*, 57(9), April 2025. ISSN 0360-0300. doi: 10.1145/3722214. URL https://doi.org/10.1145/3722214.

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

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

Masato Ishii and Atsushi Sato. Training deep neural networks with adversarially augmented features for small-scale training datasets. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2019. doi: 10.1109/IJCNN.2019.8852250.

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

Hoang Anh Just, Feiyang Kang, Tianhao Wang, Yi Zeng, Myeongseob Ko, Ming Jin, and Ruoxi Jia. LAVA: Data valuation without pre-specified learning algorithms. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JJuP86nBl4q.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJiHXGWAZ.

Hengbo Liu, Ziqing Ma, Linxiao Yang, Tian Zhou, Rui Xia, Yi Wang, Qingsong Wen, and Liang Sun. 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)*, pp. 1–5, 2023a. doi: 10.1109/ICASSP49357.2023.10096002.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=XVjTT1nw5z.

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

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

Parsa Rahimi, Damien Teney, and Sébastien Marcel. Auggen: Synthetic augmentation using diffusion models can improve recognition. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=LuKlBH8DAT.

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

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

Masahiro Suzuki, Megumi Kodaka, and Yusuke Fukazawa. 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)*, pp. 102–109, 2024. doi: 10.1109/WI-IAT62293.2024.00022.

Hongming Tan, Ting Chen, Ruochong Jin, and Wai Kin Victor Chan. Data augmentation in time series forecasting through inverted framework. *Pattern Recognition Letters*, 201:152–159, 2026. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2026.01.019. URL https://www.sciencedirect.com/science/article/pii/S0167865526000279.

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

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

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

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

Qiyao Wang, Ahmed Farahat, Chetan Gupta, and Shuai Zheng. Deep time series models for scarce data. *Neurocomputing*, 456:504–518, 2021. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2020.12.132. URL https://www.sciencedirect.com/science/article/pii/S0925231221001922.

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

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

Tianlin Xu, Li Kevin Wenliang, Michael Munn, and Beatrice Acciaio. Cot-gan: Generating sequential data via causal optimal transport. *Advances in neural information processing systems*, 33: 8798–8809, 2020.

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

Jinsung Yoon, Sercan O. Arik, and Tomas Pfister. Data valuation using reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Yang Yu, Ruizhe Ma, Wenbo Gu, and Zongmin Ma. Diffat: Effective data augmentation with diffusion models for time series forecasting. *Engineering Applications of Artificial Intelligence*, 161:112091, 2025. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2025.112091. URL https://www.sciencedirect.com/science/article/pii/S0952197625020998.

Haochen Yuan, Yutong Wang, Yihong Chen, Yunbo Wang, and Xiaokang Yang. Reaugment: Model zoo-guided rl for few-shot time series augmentation and forecasting, 2025. URL https://arxiv.org/abs/2409.06282.

Xinyu Yuan and Yan Qiao. Diffusion-TS: Interpretable diffusion for general time series generation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4h1apFjO99.

Wenzhen Yue, Yong Liu, Hao Wang, Haoxuan Li, Xianghua Ying, Ruohao Guo, Bowei Xing, and Ji Shi. OLinear: A linear model for time series forecasting in orthogonally transformed domain. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=DAyKP1tvwI.

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

A Experimental setup

This appendix describes the experimental environment used for the comparison of Section 4. Every method is run with the same data splits, the same forecasters, and the same optimiser settings, so that the differences between them are attributable to the augmentation alone.

A.1 Dataset

Wang et al. (Wang et al., 2021) implicitly defined a dataset of approximately 100 records as small-scale, and Gonen et al. (Gonen et al., 2025), 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 (Yue et al., 2025) alongside the relatively newly proposed benchmark dataset (Goktas et al., 2025), specifically the univariate dataset **Inventories to Sales Ratio** (U.S. Census Bureau. Manufacturers: Inventories to Sales Ratio [MNFCTRIRSA], 2026), **German House Prices** (Bank for International Settlements. Manufacturers: Real Residential Property Prices for Germany [QDER628BIS], 2026), **Personal Consumption Expenditures** (U.S. Bureau of Economic Analysis. Personal Consumption Expenditures: Chain-type Price Index [DPCERG3A086NBEA], 2026), **Employees Health Care** (U.S. Bureau of Labor Statistics. All Employees, Health Care [CES6562000101], 2026), and **Forest Fires** (Cortez & Morais, 2007) from the relatively recently proposed benchmark dataset (Goktas et al., 2025).

Table 3: Dataset statistics and experimental configuration.

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

A.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_{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. Forecast 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 $L_{\text{forecast}} \in \{2, 4, 8, 12\}$.

A.2 TSF Models

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

Hyperparameter	Value
Optimizer	AdamW
Learning rate	3×10^{-4}
Weight decay	0.1
Max epochs	100
Early stopping patience	10 epochs
Early stopping metric	Validation MSE
Batch size	4

In previous studies, evaluations often relied on only a few TSF models (Yoon et al., 2019; Desai et al., 2022; Yuan & Qiao, 2024; Yuan et al., 2025; Gonen et al., 2025; Tan et al., 2026; Dou et al., 2026), which carried the risk of producing biased results. Therefore, in this paper, we conducted an evaluation using six TSF models, ranging from conventional to state-of-the-art models. The TSF models include RNN, LSTM (Hochreiter & Schmidhuber, 1997), Transformer (Vaswani et al., 2017), iTransformer (Liu et al., 2023c), PatchTST (Nie et al., 2022), and TimesNet (Wu et al., 2022). Table 4 lists the hyper parameters shared across all six TSF models.

These TSF models are trained using real and synthetic data, and validation data extracted from the real data is used to evaluate the training results.

A.3 Comparison Methods

To evaluate the proposed method, we selected TimeGAN (Yoon et al., 2019), TimeVAE (Desai et al., 2022), SMFG (Suzuki et al., 2024), Diffusion-TS (Yuan & Qiao, 2024), ReAugment (Yuan et al., 2025), DiffAT (Yu et al., 2025), and AutoDA-Timeseries (Dou et al., 2026) from previous studies as comparison methods. We additionally include Gaussian (STL), a reference that is not learned at all: the training series is decomposed into trend, seasonal and residual components by STL, Gaussian noise with the standard deviation of the residual is added to the reconstruction, and the augmented windows are taken from the result. These methods selected include data augmentation methods that improve the accuracy of downstream tasks despite limited training data (Suzuki et al., 2024), methods that generate data closely resembling the distribution of the training data (Yoon et al., 2019; Desai et al., 2022; Yuan & Qiao, 2024), methods that use reinforcement learning to generate data

Table 5: Employees: RMSE change relative to training without augmentation (%) for every forecaster and every horizon, lower is better. **Mean** pools the six forecasters. Within one horizon, green bold marks the best method in that column and blue the runner-up.

Method	h	RNN	LSTM	Trans.	iTrans.	PatchTST	TimesNet	**Mean**
DAD4TS	3	−75.02	−78.58	−73.90	−6.91	−0.15	−5.92	−71.29
	6	−76.99	−79.02	−66.74	−3.75	−2.82	+0.69	−67.47
	9	−78.38	−71.83	−69.92	−1.02	+2.91	+8.03	−64.30
	12	−75.83	−78.40	−73.74	+12.78	+1.24	+23.47	−63.62
AutoDA	3	−35.46	−34.78	−18.67	+2.50	−2.05	−11.06	−28.08
	6	−75.49	−76.67	−78.30	+4.38	−15.04	−12.72	−70.29
	9	−44.22	−64.47	−60.91	−8.47	−12.67	−20.10	−51.10
	12	−62.44	−74.33	−61.24	+3.28	−11.07	−6.33	−57.56
TimeVAE	3	−7.29	−14.94	−0.30	+9.49	+8.22	−4.85	−6.86
	6	−6.99	−12.19	−8.06	+1.64	+2.50	−13.49	−8.60
	9	−9.58	−9.49	−1.97	+1.86	+1.02	−4.53	−6.47
	12	−2.92	−8.43	−4.84	+7.75	+0.70	+10.30	−3.75
ReAug.	3	−1.22	−1.77	+5.75	−0.45	+5.18	−5.21	+0.79
	6	+0.09	+4.11	−2.61	+7.54	+4.17	−15.11	+0.32
	9	−1.76	−1.29	+2.33	−10.16	−4.15	−18.20	−1.69
	12	−2.24	−2.20	−1.71	−3.37	+1.22	−4.56	−2.10
Diff.-TS	3	−4.33	−11.56	+7.31	−9.83	+4.52	−7.89	−3.08
	6	−1.94	−8.75	−1.48	−7.66	+0.65	−3.85	−4.06
	9	−4.73	−7.77	+3.19	−13.14	+1.66	+4.33	−3.21
	12	−4.14	−8.86	−0.47	+5.92	+0.75	+3.15	−3.54
DiffAT	3	−1.33	−2.76	+5.03	−2.68	−0.08	−16.99	−0.20
	6	−0.13	+0.34	−0.34	+14.15	−0.88	−13.12	−0.12
	9	−7.04	+1.18	+4.49	+15.62	−8.81	+4.31	−0.19
	12	−12.14	−12.96	−13.72	+7.88	+2.06	+7.79	−10.25
Gauss.	3	−3.23	−7.15	+3.61	+14.32	+3.37	+0.81	−1.79
	6	+2.05	−4.00	−3.05	+13.95	−0.74	−1.77	−1.18
	9	−2.86	−3.46	+0.70	+15.11	+2.41	+6.29	−0.79
	12	+1.92	−1.92	−3.62	+10.01	+1.12	+10.75	+0.11
TimeGAN	3	−5.08	+10.52	+4.55	+5.65	+11.97	+2.02	+3.56
	6	+18.19	−0.66	−0.68	+19.16	+28.22	−7.89	+6.26
	9	+2.02	−3.47	+1.30	+2.51	+3.65	−4.74	−0.04
	12	+9.03	−3.89	+0.69	−4.57	−4.30	+4.04	+1.54
SMFG	3	+0.47	+2.66	+3.58	−13.83	+9.98	−9.36	+1.79
	6	+6.48	+3.69	−0.88	−25.19	−8.51	−33.31	+0.67
	9	−0.78	−2.49	+4.20	−29.31	−3.39	−34.30	−2.69
	12	+4.11	−3.15	+1.49	−30.57	−3.85	−29.67	−2.37

for downstream tasks (Yuan et al., 2025), and methods that perform sequential data augmentation (Dou et al., 2026). By comparing DAD4TS with these methods, we evaluate its ability to optimize data augmentation for small-scale time-series data.

B PER-DATASET RESULTS

Tables 5–10 give every run behind Table 1: one table per dataset, with a row for each method and horizon and a column for each forecaster.

Table 6: Forest: RMSE change relative to training without augmentation (%) for every forecaster and every horizon, lower is better. **Mean** pools the six forecasters. Within one horizon, green bold marks the best method in that column and blue the runner-up.

Method	h	RNN	LSTM	Trans.	iTrans.	PatchTST	TimesNet	**Mean**
DAD4TS	3	−5.32	−11.65	−4.58	−1.24	−1.56	−1.08	−4.65
	6	+1.51	−0.26	+1.33	+2.28	+0.87	+2.17	+1.25
	9	−0.42	−2.25	−10.62	+1.64	+0.16	+2.02	−1.90
	12	−0.12	−0.84	+3.12	−0.04	+1.06	+1.34	+0.73
AutoDA	3	−2.27	+1.22	+7.85	+1.33	−0.68	+1.96	+1.50
	6	−0.09	−0.00	−1.86	+1.30	+0.79	+1.69	+0.24
	9	+0.28	−1.15	−4.46	+0.80	−0.12	+1.47	−0.67
	12	−0.31	−0.59	+3.02	+0.96	+3.32	+1.77	+1.29
TimeVAE	3	−5.61	+1.46	+3.33	+1.47	−0.18	−0.09	−0.04
	6	+0.37	−0.09	−3.13	+0.25	+0.20	+0.39	−0.35
	9	−0.63	−0.59	−5.35	+0.65	−1.81	+0.06	−1.39
	12	−0.25	−0.20	−9.22	−0.68	+1.07	+0.57	−1.52
ReAug.	3	−2.31	−3.96	−5.23	+1.94	+1.40	+1.60	−1.36
	6	+1.55	+0.13	+0.15	+0.82	+0.12	+0.31	+0.53
	9	−0.62	−0.78	−12.21	+1.73	−1.36	+0.98	−2.34
	12	+0.84	+0.85	−7.51	+0.54	+1.30	+0.71	−0.59
Diff.-TS	3	−13.68	−6.25	−0.98	−0.22	−0.88	−0.16	−4.18
	6	−0.36	−7.36	−3.91	−0.19	+1.24	+0.37	−1.93
	9	−4.34	−7.25	−14.89	+1.72	+0.05	+0.99	−4.48
	12	−2.93	−6.96	−1.59	+1.32	+2.12	+0.84	−1.40
DiffAT	3	−6.00	−13.53	−4.87	−1.56	−0.96	+0.00	−4.99
	6	+0.50	+0.01	−3.72	+0.99	+0.95	+2.01	+0.07
	9	+0.43	+0.65	−10.74	+1.34	−2.23	+0.12	−1.94
	12	+0.08	−0.00	−3.45	−0.19	−0.03	+1.06	−0.45
Gauss.	3	−4.21	−6.71	+8.59	+0.50	−0.84	−0.29	−0.78
	6	+0.30	−0.73	+15.38	+0.73	+1.49	+1.69	+3.07
	9	−0.83	−0.38	−0.47	+1.95	−1.54	+0.86	−0.12
	12	−0.11	−0.71	+7.80	−0.38	+2.52	+0.97	+1.67
TimeGAN	3	−1.24	+0.29	−3.46	−0.86	−0.53	+0.37	−0.91
	6	−1.39	+0.90	−1.01	−0.52	−0.10	+1.17	−0.17
	9	+1.59	+1.02	−10.77	+0.82	−1.93	+0.99	−1.57
	12	−0.13	−0.80	+3.20	−0.33	+0.12	−0.21	+0.32

Table 7: German: RMSE change relative to training without augmentation (%) for every forecaster and every horizon, lower is better. **Mean** pools the six forecasters. Within one horizon, green bold marks the best method in that column and blue the runner-up.

Method	h	RNN	LSTM	Trans.	iTrans.	PatchTST	TimesNet	**Mean**
DAD4TS	3	−17.17	−28.66	−7.78	−11.74	−5.63	−15.43	−15.43
	6	−18.74	−30.49	−19.80	−2.23	−9.47	−0.41	−14.11
	9	−7.73	−21.71	−24.55	+4.31	+0.43	−0.13	−8.07
	12	−19.04	−19.33	−6.44	+0.53	+0.29	+0.91	−6.35
AutoDA	3	−13.23	−9.42	−7.12	−7.40	−1.05	−6.16	−7.62
	6	−1.24	+1.76	−4.58	−2.52	−2.50	+12.72	+0.61
	9	+2.39	−4.06	+3.79	−3.95	−2.79	−3.41	−1.44
	12	−0.37	−8.00	−0.61	−0.23	+2.10	+2.32	−0.64
TimeVAE	3	−4.97	−7.36	−0.43	−5.51	−3.35	−7.28	−4.91
	6	−0.59	−0.56	+2.73	−4.19	−2.31	+12.47	+1.31
	9	−0.83	−3.79	−5.21	+0.80	+4.20	−7.05	−2.02
	12	+1.11	−3.31	+3.56	−3.08	+0.07	−1.31	−0.60
ReAug.	3	−4.11	−14.67	−6.47	+3.21	−0.37	−6.82	−5.85
	6	−5.81	+0.04	+4.13	−0.94	−1.62	+11.32	+1.29
	9	+2.08	−2.35	−10.38	−0.94	+0.11	−4.12	−2.75
	12	−1.98	+0.73	+1.56	−3.02	−4.35	−2.54	−1.72
Diff.-TS	3	−2.17	−15.45	−7.76	−1.13	+3.60	−0.31	−5.00
	6	−2.26	−6.11	−6.69	+0.34	−5.44	+3.70	−2.93
	9	−4.46	−5.54	−5.68	+0.59	−4.72	−2.19	−3.62
	12	−10.31	+1.66	−4.76	−4.50	+0.39	−2.72	−3.16
DiffAT	3	+2.52	−9.07	+5.16	−7.32	−1.93	−4.76	−2.80
	6	−0.61	−0.83	+0.60	−1.92	−1.58	+13.98	+1.59
	9	+1.87	+1.35	−4.05	+4.52	+2.93	−3.18	+0.51
	12	+4.75	−0.51	+0.83	+0.60	+0.13	−0.98	+0.66
Gauss.	3	+17.37	−3.06	+1.13	+0.72	+1.93	−4.27	+1.74
	6	+9.89	−0.23	−3.96	+3.41	−7.08	−2.62	−0.31
	9	+5.98	−3.18	−10.70	+3.39	+2.38	−2.65	−0.99
	12	+0.84	−6.26	−4.27	+1.15	−2.19	+1.33	−1.48
TimeGAN	3	+0.04	+13.99	−6.24	−3.68	+3.43	−9.20	+0.44
	6	−2.48	+4.62	−8.80	+11.65	−9.72	−2.92	−1.39
	9	+0.72	−0.57	−12.05	+5.27	−2.39	+2.25	−1.18
	12	+2.10	+16.50	−4.26	+4.25	−1.54	−0.92	+2.54
SMFG	3	−0.43	+7.00	+3.98	+0.00	+11.68	−0.28	+3.77
	6	+1.18	+3.47	−6.73	+2.59	+0.04	+3.91	+0.67
	9	+5.21	+32.75	−4.28	+1.05	−2.89	−2.34	+4.60
	12	−0.05	+26.95	+3.25	−3.74	−0.12	−2.28	+3.61

Table 8: ILI: RMSE change relative to training without augmentation (%) for every forecaster and every horizon, lower is better. **Mean** pools the six forecasters. Within one horizon, green bold marks the best method in that column and blue the runner-up.

Method	h	RNN	LSTM	Trans.	iTrans.	PatchTST	TimesNet	**Mean**
DAD4TS	2	−19.95	−41.39	−38.30	−2.14	+2.94	+3.04	−27.20
	4	−15.17	−44.12	−31.70	−4.53	−1.04	−3.45	−24.58
	8	−19.00	−31.65	−31.60	−4.84	−7.45	−7.86	−21.94
	12	−18.33	−27.04	−29.59	−3.89	−7.23	−2.74	−19.25
AutoDA	2	−14.08	−8.40	−23.32	−3.92	+1.03	−2.38	−13.33
	4	+9.23	−4.77	−1.48	−1.43	−1.44	−1.69	+0.38
	8	−14.40	−42.96	−40.80	−0.20	−5.47	−0.24	−24.61
	12	−24.29	−40.06	−21.24	−5.40	−5.70	−5.35	−21.81
TimeVAE	2	−6.39	+4.47	−1.42	−2.00	+5.35	+0.60	−0.72
	4	+3.75	+1.54	−3.01	−0.16	+0.67	+1.24	+0.63
	8	+5.41	+5.69	+3.88	+1.58	−0.12	−0.33	+3.75
	12	−1.98	+3.31	−3.15	+3.54	+10.98	+0.73	+0.97
ReAug.	2	−4.26	+8.58	−0.79	+0.13	+4.42	+1.23	+1.16
	4	−4.24	−6.21	−4.08	+4.36	+4.30	+4.19	−2.88
	8	+1.51	+2.71	+0.30	+4.00	−1.35	+1.77	+1.48
	12	−6.11	+8.70	−6.97	+2.81	−1.23	+2.01	−0.85
Diff.-TS	2	−7.07	−4.19	−6.28	−6.45	−0.06	−2.67	−5.38
	4	−9.93	−9.11	−3.67	−0.42	+0.40	−2.72	−6.06
	8	−5.03	−5.26	−7.75	+1.66	−1.75	+1.01	−4.35
	12	+2.63	+0.09	−11.31	+10.30	+8.30	+6.17	+0.17
DiffAT	2	−1.13	−0.02	+0.05	−4.57	+3.27	−1.65	−0.47
	4	+1.02	−0.31	+0.09	+0.22	+0.15	−2.17	+0.07
	8	+2.73	+1.16	+3.76	+2.11	−0.30	−0.58	+1.99
	12	−4.85	+0.01	−0.27	+1.77	+2.53	+3.69	−0.46
Gauss.	2	+2.09	+2.11	+0.86	+3.27	+9.55	+7.73	+2.56
	4	+1.97	+2.07	+3.05	+5.48	+7.36	+2.25	+2.94
	8	+1.54	+5.39	+1.94	−1.87	−6.21	−1.83	+1.27
	12	−2.84	+2.12	+0.96	−0.35	−7.81	−1.07	−0.83
TimeGAN	2	+10.67	+14.24	+4.56	−1.58	+3.54	+0.74	+7.83
	4	+14.23	+18.12	+5.30	+9.69	+7.25	+4.20	+11.20
	8	+10.42	+4.17	+2.22	−0.03	−5.73	−3.14	+3.28
	12	+10.68	+2.72	+2.57	+1.22	−1.64	+0.95	+3.84

Table 9: Inventories: RMSE change relative to training without augmentation (%) for every forecaster and every horizon, lower is better. **Mean** pools the six forecasters. Within one horizon, green bold marks the best method in that column and blue the runner-up.

Method	h	RNN	LSTM	Trans.	iTrans.	PatchTST	TimesNet	**Mean**
DAD4TS	3	−34.82	−21.87	−32.03	−2.72	−0.94	−1.57	−20.07
	6	−28.13	−22.74	−38.35	−2.62	−4.67	−6.75	−20.62
	9	−16.02	−23.38	−29.37	−2.21	−2.78	−6.44	−15.65
	12	−21.42	−21.84	−32.48	−2.90	−3.47	−10.40	−17.75
AutoDA	3	−38.24	−17.58	−41.97	−0.13	+1.25	+1.53	−21.64
	6	−11.48	−28.43	−21.13	+1.59	−0.63	−4.47	−13.22
	9	−6.04	−14.22	−27.96	+6.10	+5.14	+0.79	−8.50
	12	−11.63	−10.31	−16.80	+1.28	−0.77	−4.09	−8.47
TimeVAE	3	−20.66	−4.41	−26.25	+3.15	+1.04	−1.04	−11.50
	6	−13.25	−21.15	−8.07	+1.17	−1.75	−2.49	−9.17
	9	−2.26	−8.94	−19.24	+7.01	+3.97	+1.97	−4.69
	12	−5.86	−16.37	−8.73	+6.13	+1.45	+1.62	−5.22
ReAug.	3	−11.90	−0.61	−9.67	+2.02	+2.42	+1.25	−4.46
	6	−13.48	−17.66	−20.45	+1.97	−3.68	+1.35	−10.91
	9	−5.79	−3.33	−14.47	+7.22	+3.45	+3.87	−3.05
	12	−24.98	−13.64	−9.19	+5.26	−2.98	+3.46	−9.45
Diff.-TS	3	−11.00	−3.10	−21.54	−2.45	+0.37	−3.16	−8.98
	6	−10.93	+12.63	−21.72	+0.50	−3.76	+0.41	−4.87
	9	−4.60	+14.97	−23.96	+0.98	+3.37	+0.29	−2.43
	12	−22.04	+30.13	−13.71	−0.40	+0.31	−1.42	−1.68
DiffAT	3	−8.40	+6.03	−6.95	−0.55	+0.93	−3.37	−2.89
	6	−1.09	−19.62	−14.01	−0.76	−6.34	−5.70	−8.87
	9	+0.11	−16.80	−10.98	+3.47	+0.96	−3.07	−5.53
	12	−8.39	+10.64	−6.94	+1.89	+4.42	−1.68	−0.51
Gauss.	3	+3.04	+12.93	−15.15	−3.06	−1.27	−2.25	−1.70
	6	−2.03	−17.15	−25.56	−4.46	−8.54	−6.15	−11.90
	9	+3.24	−14.48	−24.80	−3.08	−3.29	−6.50	−9.20
	12	−9.22	−11.69	−15.17	−3.64	−3.39	−6.88	−9.18
TimeGAN	3	+23.16	+71.48	+3.64	+2.06	−2.51	+4.67	+18.84
	6	+12.65	+7.40	−5.51	+3.84	−2.41	−5.72	+2.33
	9	+16.02	+31.93	−7.13	+0.62	−0.65	−4.82	+7.48
	12	+6.95	−1.85	−5.49	+0.44	−0.18	−6.23	−0.67
SMFG	3	−1.59	+0.06	−11.58	+2.51	+1.57	−2.45	−3.05
	6	−0.69	−15.56	−10.43	+0.46	−2.62	−3.22	−6.26
	9	+5.85	−2.86	−3.70	+2.85	+3.98	+3.35	+1.12
	12	−2.52	−5.61	−6.66	−0.28	+2.10	−5.12	−3.45

Table 10: Consumption: RMSE change relative to training without augmentation (%) for every forecaster and every horizon, lower is better. **Mean** pools the six forecasters. Within one horizon, green bold marks the best method in that column and blue the runner-up.

Method	h	RNN	LSTM	Trans.	iTrans.	PatchTST	TimesNet	**Mean**
DAD4TS	3	−10.12	+2.59	+1.20	−10.96	−8.27	−20.38	−3.06
	6	−7.23	+21.62	+3.01	−10.45	−8.93	−34.56	+3.74
	9	−2.06	+14.85	+10.05	−5.03	−12.38	−14.93	+6.13
	12	+5.81	+17.30	+8.39	+4.10	−5.91	−34.47	+8.41
AutoDA	3	−11.13	−7.96	−8.20	+1.24	−9.52	−12.39	−9.00
	6	−12.57	−10.01	−12.01	−9.49	−20.74	−29.66	−12.24
	9	+1.16	−0.96	−1.76	−7.31	−18.43	−28.17	−1.79
	12	+5.69	+16.30	+2.26	−2.01	+3.22	−33.13	+6.33
TimeVAE	3	−0.88	−3.62	−2.20	+11.22	−11.26	−12.04	−2.34
	6	−2.89	−0.44	−2.51	+0.11	+13.47	+10.79	−1.19
	9	−0.36	−0.66	−0.49	+1.92	−1.34	+15.14	−0.04
	12	−2.02	+0.71	−0.26	+5.86	+6.04	−7.88	−0.52
ReAug.	3	+1.57	+2.66	+3.20	+8.26	−10.90	−12.04	+1.95
	6	−0.75	+0.73	+0.77	−3.72	−4.64	−4.68	−0.12
	9	−1.38	−2.30	+0.42	+2.45	−11.16	+1.27	−1.21
	12	−3.05	−0.43	−1.73	+5.77	+11.40	−7.09	−1.50
Diff.-TS	3	+0.21	+2.16	+0.61	+21.10	−9.56	−6.89	+1.02
	6	−0.08	+6.17	+0.87	−0.45	+1.24	−4.56	+2.01
	9	+0.60	+29.95	+5.29	−0.92	−10.58	+17.62	+11.35
	12	−1.32	+12.42	+1.21	+4.44	+15.75	−6.84	+3.91
DiffAT	3	−0.29	+0.53	+1.61	+9.93	−12.05	−12.04	+0.23
	6	−1.11	−0.68	+0.02	−4.76	−0.16	+9.18	−0.39
	9	+0.60	+0.45	+5.52	+2.79	+0.13	+1.36	+2.05
	12	−2.45	−0.37	−0.56	+5.74	+6.94	−7.12	−1.03
Gauss.	3	+4.74	+2.66	+2.61	−3.88	−2.68	−12.01	+2.68
	6	+3.21	+4.24	+2.68	−4.27	+0.70	−24.40	+2.31
	9	+1.34	+4.02	+2.83	−0.31	−10.55	−20.48	+1.71
	12	+1.21	+4.26	−0.17	−5.76	+2.60	−27.98	+0.58
TimeGAN	3	+1.79	+2.58	+0.87	−25.34	−7.71	−28.90	+0.16
	6	+0.65	+4.27	+1.30	−11.85	−16.37	−33.97	+0.26
	9	+3.05	−1.36	+5.56	−8.31	−26.77	−33.84	+0.48
	12	+0.49	+24.14	+3.96	+0.70	−2.63	−24.41	+7.81
SMFG	3	+0.77	+2.22	+0.35	−3.29	+1.40	−1.05	+0.97
	6	−1.15	+24.98	−0.42	−3.13	−4.77	+21.18	+7.64
	9	−1.58	+43.84	−0.25	+1.36	−10.64	+16.83	+13.39
	12	−0.03	+35.38	−2.54	−4.46	+9.09	−25.34	+9.23

Table 11: Removing the foundation-model conditioning. The ablation differs only in that the diffusion model receives no conditioning vector, so a generated window is no longer tied to the real window it was drawn for, while the processing label is still read by its own embedding. Entries are RMSE changes against training without augmentation (%), lower is better. A positive difference means the conditioning helped. Results with seed=2025.

	Employees	Forest	German	ILI	Invent.	Consum.	**All**
DAD4TS (full)	-66.29	-2.44	-7.98	-22.71	-14.53	$+4.06$	-11.52
without TSFM conditioning	-52.70	-1.17	-8.87	-24.21	-14.14	$+10.21$	-8.44
difference (pp)	$+13.59$	$+1.27$	-0.89	-1.50	$+0.40$	$+6.14$	$+3.08$
	RNN	LSTM	Trans.	iTrans.	PatchTST	TimesNet	**All**
DAD4TS (full)	-17.70	-13.99	-11.57	-3.50	-1.52	-5.84	-11.52
without TSFM conditioning	-12.90	-6.73	-10.83	-1.64	-2.98	-6.63	-8.44
difference (pp)	$+4.80$	$+7.26$	$+0.74$	$+1.86$	-1.46	-0.78	$+3.08$

C REMOVING THE FOUNDATION-MODEL CONDITIONING

Table 11 gives the comparison. Pooled over the grid the conditioning is worth 3.08 points: -11.52% with it against -8.44% without.

The conditioning vector c_i is what ties a generated window to one particular real window — its level, its scale and its shape — and the flow's target is that window after a known transformation. A forecaster that normalizes each input window removes the level and the scale before it sees the data, so the part of that link which carries most of the variation never reaches it, and cutting the link costs nothing. The forecasters that read the window as it is lose exactly that. Therefore, we suggest that by introducing c_i—which links the features of the actual window—when training the diffusion model, we are able to generate data that does not deviate significantly from the actual window.

D GENERATED SAMPLES

For every dataset, the synthetic windows folded back onto the training region, and the value distribution of the same sets.

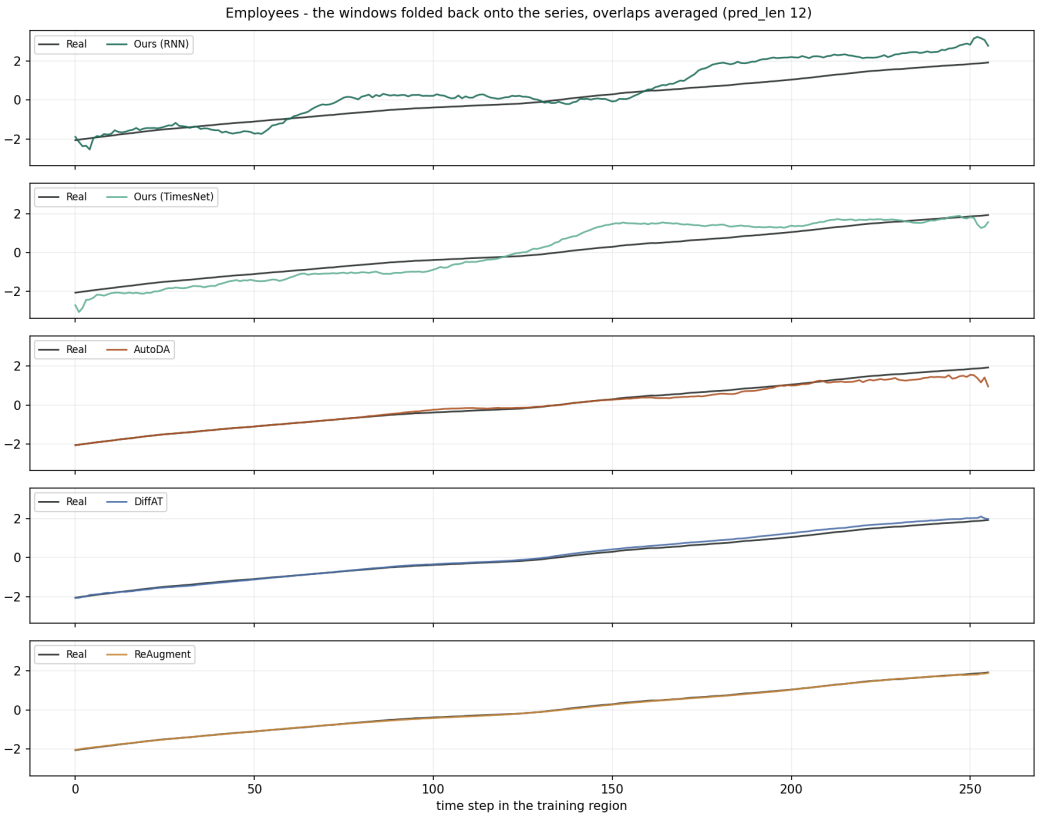


Figure 4: Employees: the synthetic windows folded back onto the training region, overlaps averaged, on the scale of the real training windows. The real series is the dark line in every panel.

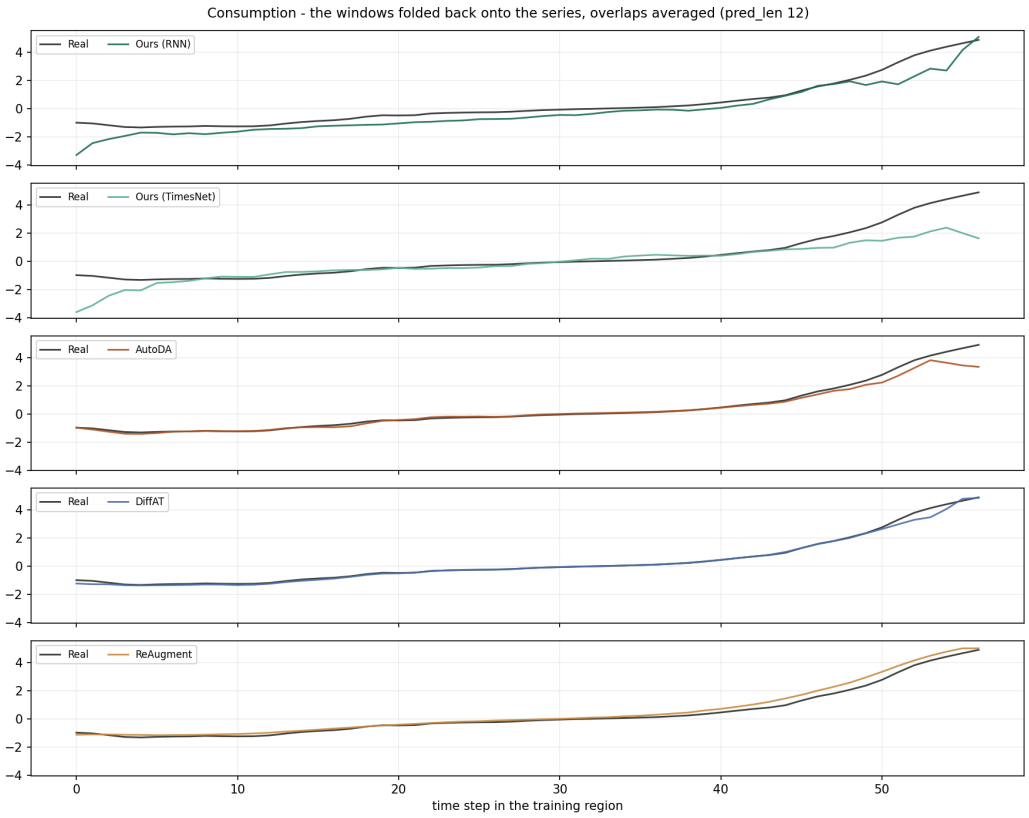


Figure 5: Consumption: the synthetic windows folded back onto the training region, overlaps averaged, on the scale of the real training windows. The real series is the dark line in every panel.

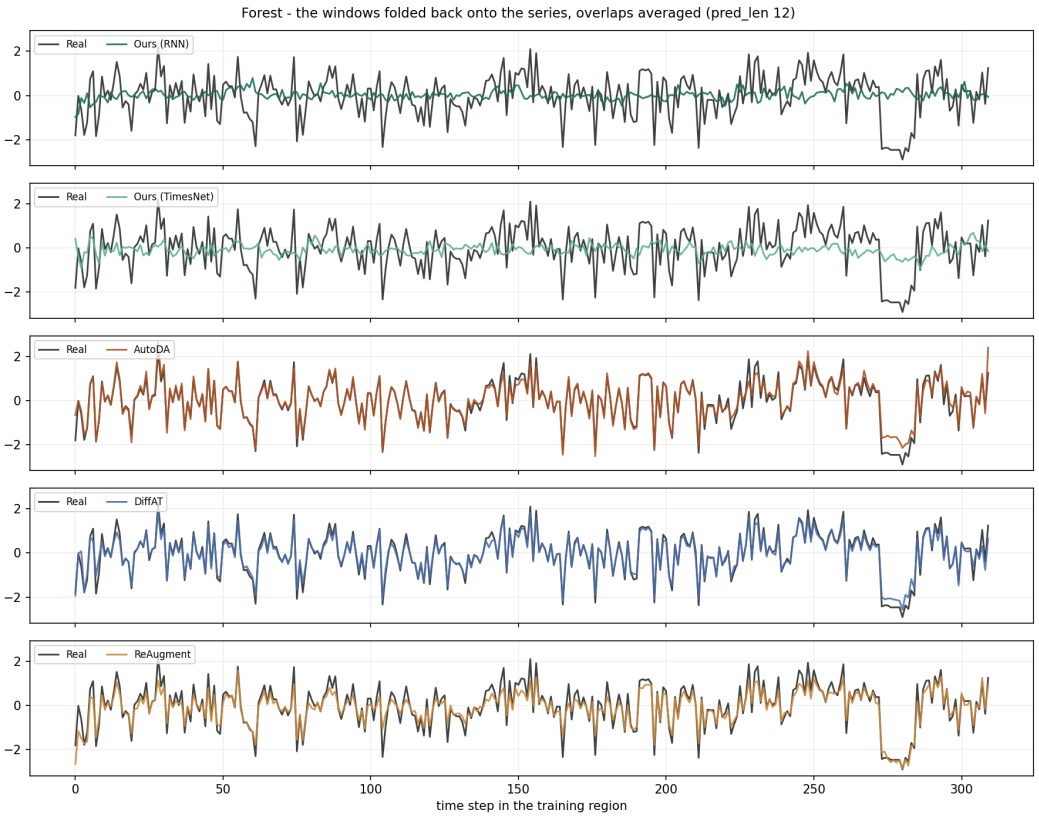


Figure 6: Forest: the synthetic windows folded back onto the training region, overlaps averaged, on the scale of the real training windows. The real series is the dark line in every panel.

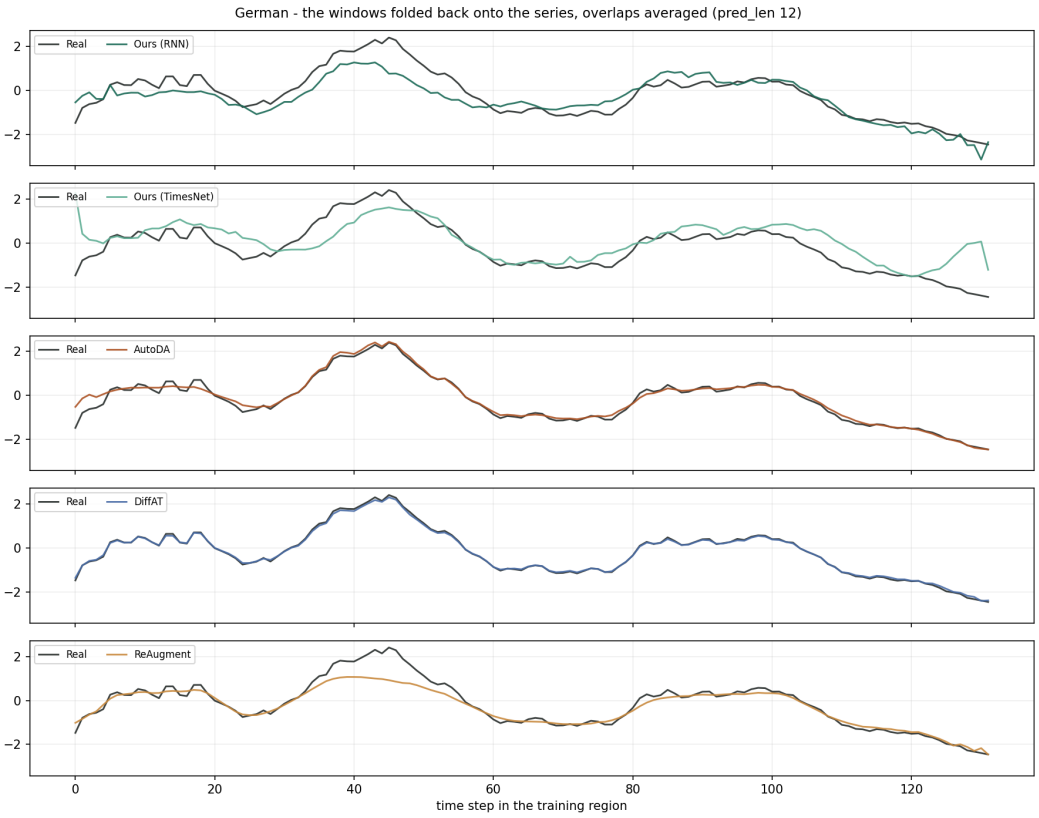


Figure 7: German: the synthetic windows folded back onto the training region, overlaps averaged, on the scale of the real training windows. The real series is the dark line in every panel.

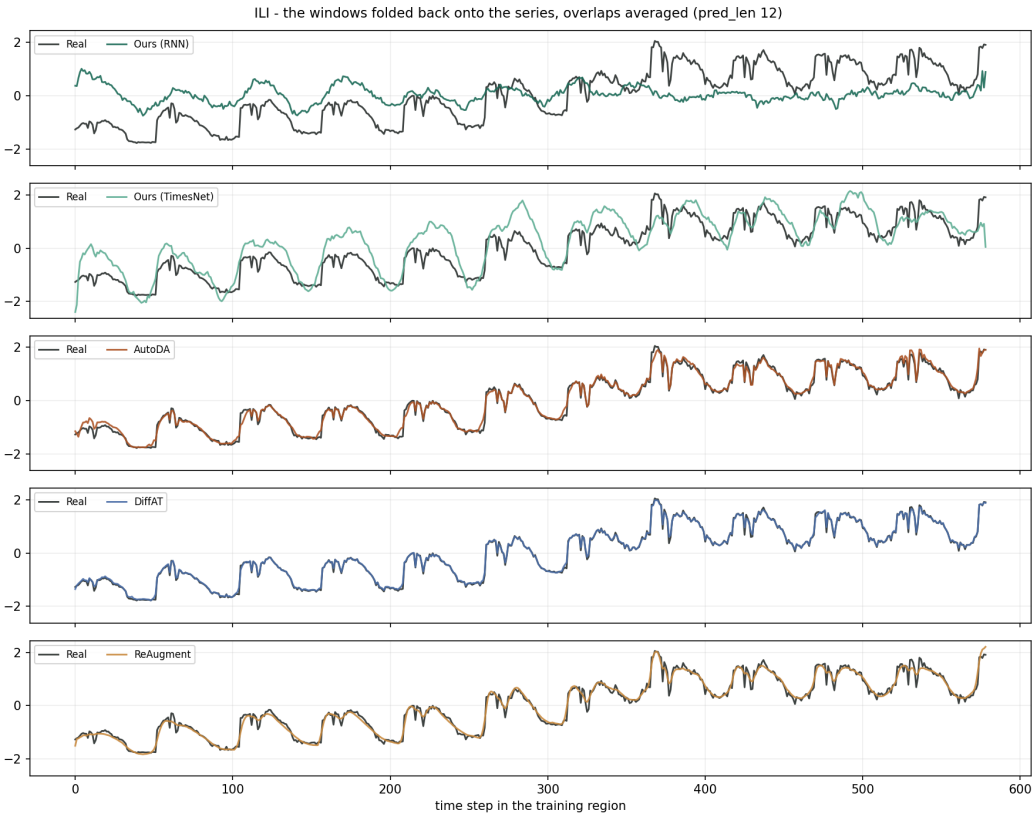


Figure 8: ILI: the synthetic windows folded back onto the training region, overlaps averaged, on the scale of the real training windows. The real series is the dark line in every panel.

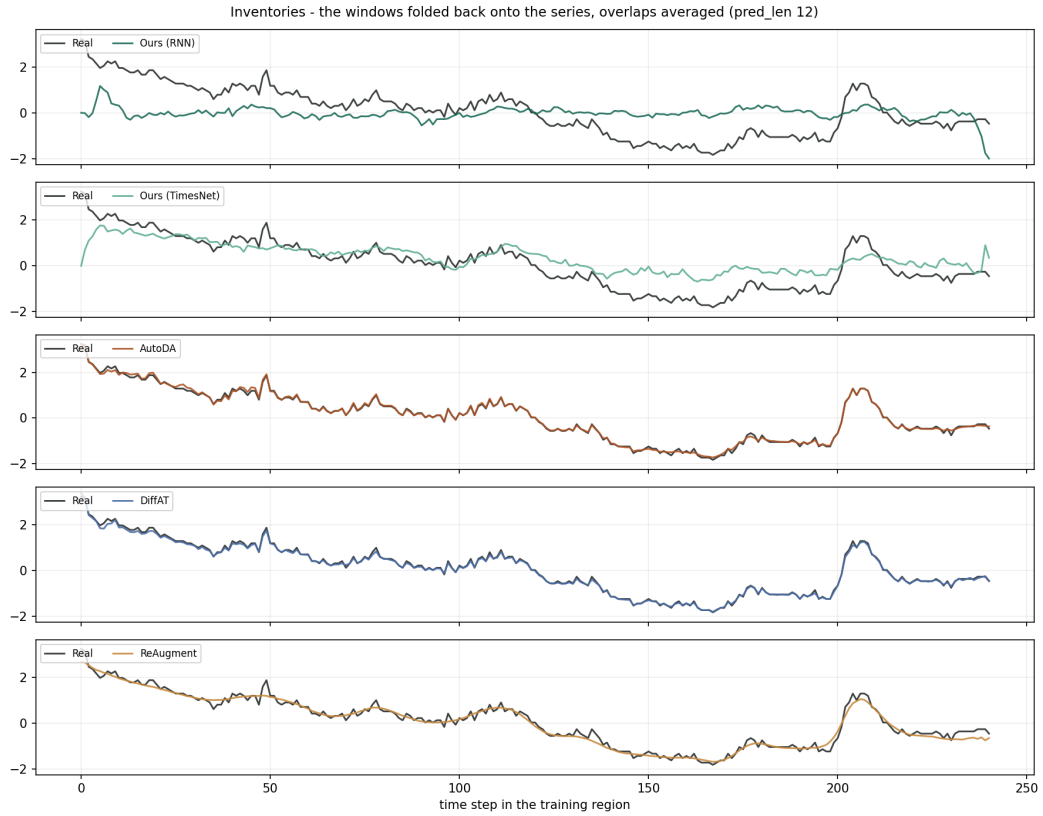


Figure 9: Inventories: the synthetic windows folded back onto the training region, overlaps averaged, on the scale of the real training windows. The real series is the dark line in every panel.

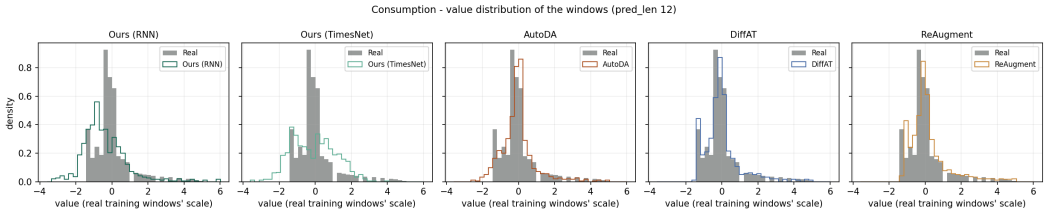


Figure 10: Consumption: the value distribution of the synthetic windows against the real training windows, as in Figure 3.

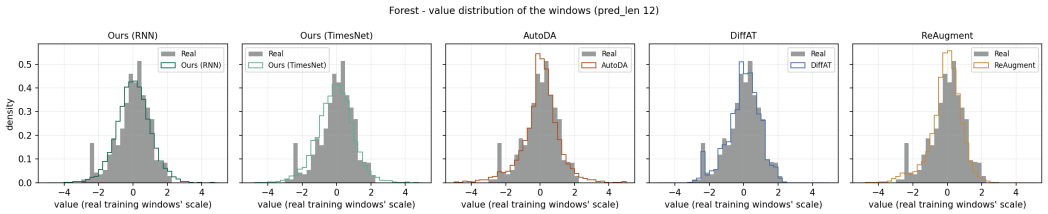


Figure 11: Forest: the value distribution of the synthetic windows against the real training windows, as in Figure 3.

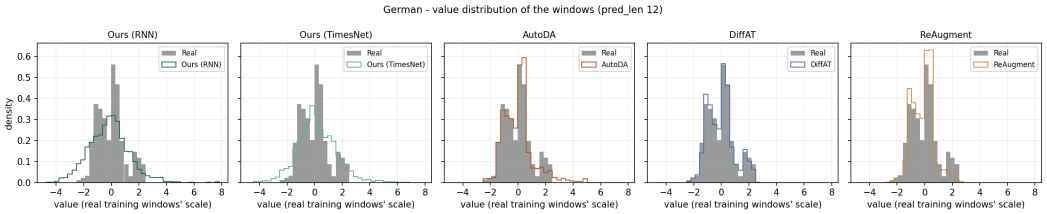


Figure 12: German: the value distribution of the synthetic windows against the real training windows, as in Figure 3.

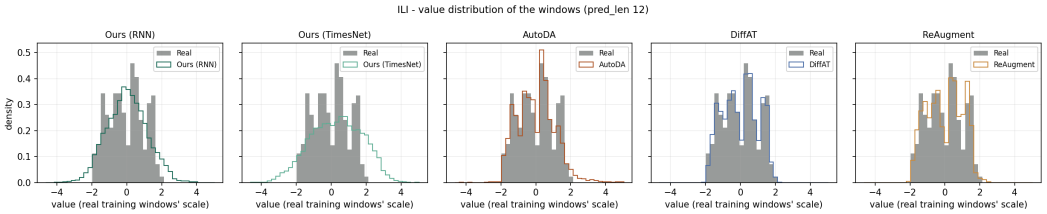


Figure 13: ILI: the value distribution of the synthetic windows against the real training windows, as in Figure 3.

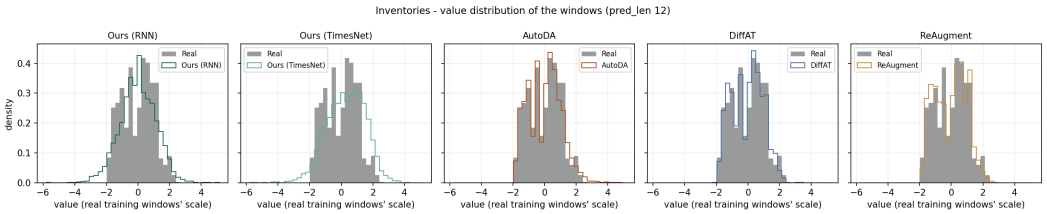


Figure 14: Inventories: the value distribution of the synthetic windows against the real training windows, as in Figure 3.