OATS: Online Data Augmentation for Time Series Foundation Models

Junwei Deng
University of Illinois Urbana-Champaign

Chang Xu*
Microsoft Research

Jiaqi W. Ma
University of Illinois Urbana-Champaign

Ming Jin
Griffith University

Chenghao Liu
Datadog AI Research

Xu Zhang
Microsoft Research

Li Zhao
Microsoft Research

Jiang Bian
Microsoft Research

Abstract

Time Series Foundation Models (TSFMs) are a powerful paradigm for time series analysis and are often enhanced by synthetic data augmentation to improve the training data quality. Existing augmentation methods, however, typically rely on heuristics and static paradigms. Motivated by dynamic data optimization, which shows that the contribution of samples varies across training stages, we propose OATS (Online Data Augmentation for Time Series Foundation Models), a principled strategy that generates synthetic data tailored to different training steps. OATS leverages valuable training samples as principled guiding signals and dynamically generates high-quality synthetic data conditioned on them. We further design a diffusion-based framework to produce realistic time series. Experiments on TSFMs demonstrate that OATS consistently outperforms regular training and yields substantial performance gains over static data augmentation baselines across six validation datasets and two TSFM architectures. The code is available at the link https://github.com/microsoft/TimeCraft.

1 Introduction

Time series modeling plays a critical role across a wide range of domains, including finance [Kim et al., 2019, Li et al., 2024], healthcare [Guo et al., 2023], climate science [Liang et al., 2023], and industrial monitoring [Zamanzadeh Darban et al., 2024]. Recent developments in time series foundation models (TSFMs) have further advanced this field by leveraging large-scale datasets collected from multiple third-party sources [Yao et al., 2024, Ansari et al., 2024], enabling cross-domain learning and zero-shot generalization. Nevertheless, the success of TSFMs relies on the availability of high-quality data. Typical challenges in time series datasets include missing values [Junninen et al., 2004], heterogeneous sampling rates [Woo et al., 2024], imbalanced domain distributions [Yao et al., 2024], data duplication [Lin et al., 2023]. These issues make it more difficult to curate reliable large-scale time series datasets than in domains such as natural language processing [Gao et al., 2020, Raffel et al., 2020]. As the community moves toward large-scale foundation models, synthetic data augmentation has emerged as a critical and widely adopted method for enhancing training data with synthetic samples. Leveraging the high controllability of time series patterns and operational simplicity, synthetic data augmentation can not only address data scarcity but also enriches domain diversity and improves the robustness of TSFMs [Liu et al., 2025].

*Corresponding author. Email: chanx@microsoft.com

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Various data augmentation methods have been proposed in TSFM studies to enrich training datasets with realistic and diverse synthetic data. These approaches can be broadly categorized into two groups. The first involves directly introducing *manually designed patterns* to synthesize data. Such patterns include sinusoidal waves [Goswami et al., 2024], decomposed time series [Dooley et al., 2023, Lan et al., 2025], and handcrafted kernel banks [Shi et al., 2024, Ansari et al., 2024, Xie et al., 2025]. The second group focuses on applying *basic transformations to existing time series data* to generate synthetic variants. Examples of such techniques include smoothing, jittering [Um et al., 2017, Moroshan et al., 2025], and TSMixup [Ansari et al., 2024, 2025, Moroshan et al., 2025].

While these approaches have shown empirical effectiveness, they rely on *handcrafted heuristics* that are often *agnostic to the model training process*, which result in two critical challenges in designing high-performance data augmentation strategies. First, the quality of time series data for TSFMs is difficult to quantify in a principled way. The heuristics that work for one time series model may fail for another [Liu et al., 2025, Kuvshinova et al., 2024]. Second, recent studies show that the value of the same data sample varies across the training process [Wang et al., 2024a], which challenges the capacity of the static data augmentation paradigm that generates synthetic data once and uniformly incorporated along the whole training process.

In this study, we address these challenges by leveraging recent advancements in *data attribution* [Koh and Liang, 2017, Deng et al., 2025] and *online data optimization* [Wang et al., 2024c]. For the first challenge, data attribution aims to quantify the influence of individual training data points on model outputs; instead of studying heuristic rules like which pattern of time series data is helpful for the training, data attribution allows us to define and assess the quality of time series data by its influence on a utility function, e.g., validation loss, in a principle way. For the second challenge, we go beyond the static data augmentation paradigm in existing TSFM literature and conduct online data augmentation, which incorporates the training process information and dynamically generates high-quality data for each training step. For example, our analysis of a TSFM training run (Section 4.4) reveals that the contribution of each training sub-dataset evolves dynamically across training steps and does not align with its proportion in the corpus, suggesting that a static augmentation strategy is unlikely to remain optimal throughout training.

Concretely, we propose `OATS` (Online Data Augmentation for Time Series Foundation Models), a strategy to dynamically generate high-quality synthetic data in a principled manner. `OATS` generates high-quality synthetic data by using training samples with high data attribution scores [Koh and Liang, 2017] as guiding signals. `OATS` consists of two core components: **Time-series influence scores (TSIS)** integrate data attribution with time series–specific knowledge to dynamically assess the quality of each training sample in a principled manner to create a generation guiding signal. **High-quality guided data augmentation** leverages the guiding signal to condition a diffusion model trained on a small subset of the TSFM training data for synthetic data generation.

We summarize our contributions in this paper as follows:

- **An Online Data Augmentation Paradigm for TSFM**. We propose a new paradigm for TSFM training that generates synthetic data tailored to each training step. This approach expands the potential of the traditional static data augmentation paradigm by taking the training process information into consideration.
- **A Novel Online Data Augmentation Strategy:** `OATS`. `OATS` leverages valuable training samples identified by data attribution scores as guiding signals and employs a diffusion model to generate synthetic data conditioned on these signals for each training step. Overall, `OATS` offers a principled approach to dynamically generating high-quality synthetic data.
- **Comprehensive Empirical Evaluation**. We evaluate `OATS` on six forecasting datasets across two TSFM architectures. Through systematic ablations, we isolate the contributions of TSIS guidance, the generation strategy, and the validation set design, demonstrating that each component contributes to the overall improvement over baseline augmentation methods.

2 Related Works

Time Series Data Augmentation. Traditional data augmentation for time series mostly relies on static, manually designed heuristics. Techniques such as injecting sinusoidal wave [Goswami et al., 2024], transformations of decomposed time series [Dooley et al., 2023], jittering [Um et al., 2017], and TSMixup [Ansari et al., 2024] aim to enrich datasets before training. More advanced,

AutoTCL [Zheng et al., 2024] leverages parameterized contrastive learning to decompose time series into information and noise components, applying transformations to the former. However, these paradigms are executed offline, remaining agnostic to the dynamic optimization state of the TSFMs during training. Notably, although Taga et al. [2025] proposes an online algorithm combining RHO-based selection with traditional transformations, its simplistic generation process may lead to performance degradation. Moreover, the RHO-based approach incurs substantial computational overhead as it requires training a separate reference model.

Dynamic Data Optimization and Online Selection. Recent advances highlight that the contribution of training data changes significantly across optimization stages. In the large language models, DoReMi [Xie et al., 2023] demonstrates the necessity of dynamic data mixing. Down to the instance level, GREATS [Wang et al., 2024c] formulates online batch selection as an optimization problem, leveraging Taylor expansions to evaluate data influence efficiently. Despite these strides, existing online methods mainly focus on sample pruning or reweighting coupled with naive static augmentations. OATS pioneers the integration of rigorous online influence evaluation with a high-fidelity conditional generation, effectively closing the loop between real-time data valuation and dynamic synthetic data generation.

Time Series Generation. Time series generation has been explored using a variety of model architectures. GAN-based approaches train adversarial objectives to generate realistic sequences [Yoon et al., 2019]. VAE-based methods design specialized decoders tailored for time series [Desai et al., 2021]. More recently, diffusion-based methods have been developed, leveraging different denoising architectures. Beyond unconditional synthesis, conditional generation has been studied in settings such as metadata-conditioned generation [Narasimhan et al., 2024] and prompt-based generation [Huang et al., 2025].

3 Method

In this section, we introduce the modules of Online Data Augmentation for Time Series Foundation Models (OATS). OATS consists of two modules, i.e., *Time-series Influence Scores (TSIS)* (Section 3.2) quantitatively estimate the quality of time series samples and to identify valuable data as guiding signal; *High-quality guided Data Generation* (Section 3.3) generates synthetic samples conditioned on the guiding signal. A diagram of the whole algorithm is presented in Figure 1, and an algorithm block is shown in Section 3.4.

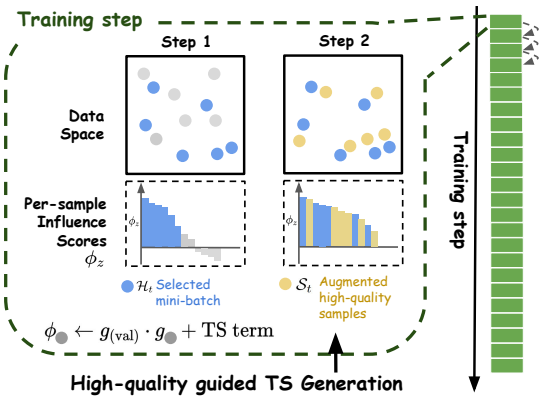


Figure 1: Architecture of OATS. OATS employs two modules: ① Time-Series Influence Scores (TSIS) create generation guiding signals as high-quality data samples, and ② guide time series synthetic data generation for augmentation.

3.1 Problem Setup

Set-up of online data augmentation in TSFM. Suppose we have a training dataset $\mathcal{D}_{tr}$ of size N. We also have a validation dataset $\mathcal{D}_{val} = \{z_v | v = [N_v]\}$, $z_v \in \mathcal{Z}$ and $\mathcal{D}_{val}$ is not overlapped with $\mathcal{D}_{tr}$.

A TSFM parameterized by $w \in \mathcal{W}$ is being trained on $\mathcal{D}_{tr}$ to minimize the loss function ℓ via an iterative optimization algorithm, e.g., stochastic gradient descent [Ruder, 2016] for T steps. The intermediate checkpoints of each step are represented as $\{w_t | t = [T]\}$. At iteration t, a mini-batch of data $\mathcal{B}_t = \{z_1, z_2, \ldots, z_B\}$ is sampled. Online data augmentation then generates synthetic samples $\mathcal{S}_t$, which are combined with all or part of $\mathcal{B}_t$ to update the model.

3.2 Construct Guiding Signal via Time Series Influence Scores

Generating synthetic data is widely used in the training process, yet a core question remains unsettled: *What should be generated?* A consistent and universally accepted answer has yet to emerge for the definition of the quality criteria for time series synthetic data in data augmentation. In OATS, we employ *data attribution* [Koh and Liang, 2017, Pruthi et al., 2020] as a principled method to estimate the influence of a data point on the model output, which has shown significant usefulness in dataset optimization in different areas. The data attribution scores to validation loss are taken as the quantitative quality indicator.

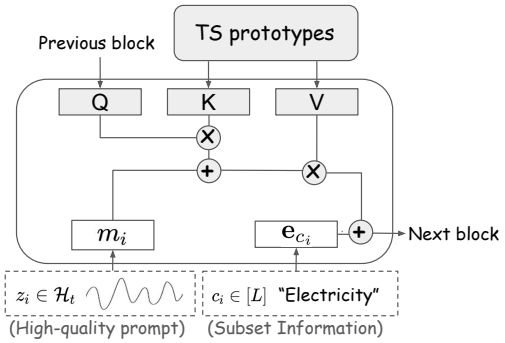


Figure 2: The architecture of the denoising diffusion model to generate synthetic data conditioned on constructed generation signals.

We design a time-series influence score (TSIS)[2] function with respect to a validation dataset $\mathcal{D}_{val}$ as $\mathcal{F}_{\mathcal{D}_{val}}$ for each sample to be $\mathcal{F}_{\mathcal{D}_{val}} : \mathcal{D}_{tr} \times \mathcal{W} \to \mathbb{R}$, which estimates the data attribution score of a data sample to be trained on w_t with respect to the performance on $\mathcal{D}_{val}$. A larger score indicates that training on z for the next training step $(t+1)$ leads to better performance on $\mathcal{D}_{val}$. The TSIS function can be defined as:

$$\mathcal{F}_{\mathcal{D}_{val}}(z_i, w_t) = \underbrace{g_{(\text{val});\text{t}} \cdot g_{z_i;t}}_{\text{Influence Score}} - \underbrace{\mathbf{I}_{\text{SNR}(z_i)<k} \cdot \infty}_{\text{TS-specific quality}}, \tag{1}$$

where $g_{(\text{val});\text{t}} = \nabla_w \sum_{v_i \in \mathcal{D}_{val}} \ell(v_i, w_t)/|\mathcal{D}_{val}|$ and $g_{z_i;t} = \nabla_w \ell(z_i, w_t)$. The data attribution score is a first-order Taylor approximation of the validation loss change, where a careful derivation is stated in Lemma 1. We additionally consider TS-specific quality indicators, which we use signal-to-noise ratio (SNR) in a pre-selection process, where k is a threshold of minimal SNR and $\mathbf{I}_{\cdot}$ is the indicator function. High-quality data $\mathcal{H}_t$ is identified as the samples with top-k TSIS scores in each mini-batch $\mathcal{B}_t$ of training step t, which will be used as guiding signals for synthetic data generation. k is a hyperparameter while we follow a previously-adopted ratio (50%) in Wang et al. [2024c].

Lemma 1 (First-order Taylor approximated influence score)**.** *The influence score term of TSIS is a first-order Taylor approximation of the difference of a utility function before and after a training step. Typically, we define the utility function to be the validation loss, which can be represented as*

$$\begin{aligned}
\ell(\mathcal{D}_{val}, w_t) &= \frac{1}{|\mathcal{D}_{val}|} \sum_{v_i \in \mathcal{D}_{val}} \ell(w_t, v_i) \\
\ell(\mathcal{D}_{val}, w_t) - \ell(\mathcal{D}_{val}, w_{t+1}) &= (w_t - w_{t+1})^\top \nabla_w \ell(\mathcal{D}_{val}, w_t) + \mathcal{O}(\|w_{t+1} - w_t\|^2) \\
&\simeq (w_t - w_{t+1})^\top \nabla_w \ell(\mathcal{D}_{val}, w_t) \\
&= \eta_t \nabla_w \ell(z_i, w_t)^\top \nabla_w \ell(\mathcal{D}_{val}, w_t),
\end{aligned}$$

where the model is updated through gradient descent[3] *on a data sample z_i with learning rate η_t, i.e.,*

$$w_{t+1} = w_t - \eta_t \nabla_w \ell(z_i, w_t).$$

The first-order approximation error depends on the magnitude of parameter updates and may vary across learning rates and optimizers. To empirically validate TSIS, we compare its rankings against single-step leave-one-out (LOO) influence on 100 randomly sampled training examples. At training step 1,000, TSIS achieves Spearman correlations of 0.926 ± 0.029 with AdamW and 0.966 ± 0.008 with SGD under the actual scheduled learning rate, averaged over five random seeds. These results support the reliability of TSIS for online data ranking. Detailed results are provided in Appendix D.6.

[2]We will use "influence scores" and "data attribution scores" interchangeably.

[3]Here we use SGD on a single data sample, which has been widely accepted as a reasonable approximation for other optimizers like Adam in data attribution studies[Wang et al., 2024c,b].

3.3 High-quality Data Guided Data Generation

Once the guiding signal is constructed, a natural question arises: *how can the synthetic data be generated?* We design a high-quality data guided generation model $\mathcal{G}$ for online synthetic data generation. We aim to model the conditional distribution $p(\hat{z}|\mathcal{H}_t)$, which defines the probability of generating a sample $\hat{z}$ given the guiding signal $\mathcal{H}_t \subseteq \mathcal{B}_t$ for each training step t. The conditional generation model $\mathcal{G}$ serves as a parameterized approximation of this distribution, enabling practical sampling via $\hat{z} \sim \mathcal{G}(\mathcal{H}_t)$. We design the architecture of $\mathcal{G}$ to be a diffusion model utilizing a time series semantic prototype module and take $\mathcal{H}_t$ as a generation condition. To incorporate the condition into the intermediate layers of the noise prediction network, we take Huang et al. [2025] as our backbone model.

Our generation model $\mathcal{G}$ introduces a set of latent arrays P as time series prototypes and extracts prototype weights $\mathcal{M} = \{m_i|z_i \in \mathcal{H}_t\}$, where we leverage the same design of weight extractor and time series prototypes initialization in Huang et al. [2025]. We additionally include a class conditional guidance based on the subset information to enhance the condition, denoted as $\mathcal{C}_{\text{high}} = \{c_i|z_i \in \mathcal{H}_t\}$, where each $c_i \in [L]$ indicates which disjoint subset $\mathcal{D}_l$ the sample z_i belongs to.

The generation (reverse diffusion) for each prompt can be expressed as

$$p(z^{(t-1)}|z^{(t)}, \mathcal{D}_{\text{high}}) = \mathcal{N}(z^{(t-1)}|\mu(z^{(t)}, t, \underbrace{m_i, c_i}_{\text{High-quality data guidance}}), \sigma_t), \tag{2}$$

where $z^{(t)}$ is the intermediate sample at diffusion timestep t, $\mathcal{N}$ is Gaussian distribution, σ_t is the noise covariance at diffusion step t. The architecture of the parameterized conditional noise predictor incorporates the prototype weights m_i and class condition c_i. We present the architecture of the high-quality guided data augmentation in Figure 2 in an intermediate u^{th} U-Net layer. The formula can be represented as

$$h^{(u)} = \text{FF}(\text{Softmax}(\frac{QK^T}{\sqrt{d}} + m_i)V||\mathbf{e}_{c_i}), \tag{3}$$

where $Q = h^{(u-1)}W_Q^{(u)}$, $K = PW_K^{(u)}$ and $V = PW_V^{(u)}$ are the query, key, and value for the attention. $W_Q \in \mathbb{R}^{d\times d}$, $W_K \in \mathbb{R}^{d\times d}$, $W_V \in \mathbb{R}^{d\times d}$ are learnable parameters, $P \in \mathbb{R}^{N_p\times d}$ is the prototype latent arrays, $\mathbf{e}_{c_i} \in \mathbb{R}^{d_c}$ is the embedding of class c_i, d and d_c is the hidden dimension of prototype embedding and class embedding, and FF is a feed-forward network. The final U-Net layer's output h^{final} is followed by another feed-forward network to produce the predicted noise.

3.4 Algorithm Overview

We summarize Online Data Augmentation for Time Series Foundation Models (OATS) and present in Algorithm 1. To further improve efficacy and efficiency, we propose an explore–exploit mechanism that leverages cached influence scores to reduce computational overhead in Appendix A and discuss the computational cost in Appendix B.

Algorithm 1 Online Data Augmentation for Time Series Foundation Models (OATS)

Require: Training dataset $\mathcal{D}_{tr}$, validation dataset $\mathcal{D}_{val}$, training batch size b, conditional augmentation algorithm $\mathcal{G}$, training loss ℓ, TSIS function $\mathcal{F}_{\mathcal{D}_{val}}$.
Initialize model w_0
Initialize TSIS per subset $\Phi_{\mathcal{D}_l}, l \in [L]$ to be subset proportion.
for $t = 1$ to T **do**
  Sample $\mathcal{B}_t$ ▷ Step 1
  Calculate $\mathcal{F}_{\mathcal{D}_{val}}(z_i, w_t)$ for $z_i \in \mathcal{B}_t$
  Update $\Phi_{\mathcal{D}_l}$ according to Equation 4.
  Select a subset $\mathcal{H}_t \subseteq \mathcal{B}_t$ of samples with top-$b/2$ of value $\mathcal{F}_{\mathcal{D}_{val}}(z_i, w_t)$.
  Generate $b/2$ samples as $\mathcal{S}_t$ using $\mathcal{G}$ guided by $\mathcal{H}_t$. ▷ Step 2
  Update w_t on mini-batch data $\mathcal{H}_t \cup \mathcal{S}_t$ and get w_{t+1}, continue to next step $t + 1$.
end for

4 Experiments

In this section, we present the empirical evaluation of OATS. We first introduce the experiment setup in Section 4.1. We then evaluate the performance of OATS on typical TSFM architectures and datasets in Section 4.2. We also include several ablation studies in Section 4.3.

4.1 Experiment Setup

Model. We conduct experiments on two typical TSFM architectures, i.e., encoder-only and decoder-only Transformer. We follow the same training settings and model definitions, which incorporate patch embedding, rotary positional embedding, and a mixture of distributions to better adapt to time series forecasting while preserving extensibility, in Yao et al. [2024] to match the popular TSFM forecasting models, Moirai [Woo et al., 2024] and Chronos [Ansari et al., 2024]. We will call these two models "Encoder-only TSFM" and "Decoder-only TSFM" in our following experiments.

Datasets. Following Yao et al. [2024], we train "Encoder-only TSFM" and "Decoder-only TSFM" on the LOTSA dataset in the pretraining stage. These models are then evaluated on the out-of-distribution LSF dataset [Wu et al., 2023], using various prediction lengths and a preprocessing pipeline as in Yao et al. [2024]. We evaluate 6 out-of-distribution datasets in LSF [Wu et al., 2023] (ETTm1, ETTm2, ETTh1, ETTh2, Weather, Electricity). We also take a very small number of samples (32 in all our experiments) from these evaluation datasets as a validation set used by TSIS (held out from the test set). The training dataset of the high-quality prompt-guided diffusion model is a very small subset sampled from the training dataset of TSFM. In our experiments, we sample 5% of the training data.

Baselines. We examine two popular data augmentation methods for TSFM. "TSMixup" or Time Series Mixup is proposed in Ansari et al. [2024] which creates new data samples from k existing ones through weighted summation. The generation process can be represented as $\hat{z} = \sum_{i=1}^{k} \lambda_i z_i$, where $\hat{z}$ is the generated sample and λ_i is the weight for the ith sample. "Jitter" [Um et al., 2017] is another widely adopted data augmentation method that involves small noise on existing samples to increase the robustness and diversity. The generation process can be represented as $\hat{z} = z + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma)$ and σ is the noise variance. We also include the result of the regular training process without additional data augmentation.

Evaluation Metrics. To be consistent with Yao et al. [2024], we primarily report the normalized mean absolute percentage error (MAPE) and negative log-likelihood (NLL) for Encoder-only TSFM and Decoder-only TSFM, as these metrics avoid distortions caused by high-amplitude samples. We specifically report the metrics for prediction length to be 192 and "overall" (average performance of using prediction lengths).

4.2 Performance of OATS

In Table 1 and Table 2, we present the NLL and MAPE on the test datasets on Encoder-only TSFM and Decoder-only TSFM, respectively. OATS outperforms baselines as well as regular training in almost all cases and both metrics. Especially, for datasets ETTm1, ETTm2, ETTh2, Weather and Electricity, OATS reaches the best result. TSMixup and Jitter get mixed performance compared to regular training, as presented by the light green and red backgrounds. While OATS performs more consistently better than regular training. In Figure 3, we present the test loss (NLL) curve on evaluation datasets on Encoder-only TSFM and different test datasets. OATS achieves a faster reduction in NLL than other baseline data augmentation methods, as well as the regular training, and achieves better overall performance. Additional results are provided in Appendix D.

We further evaluate the scalability of OATS on a 100M-parameter encoder-only TSFM, following the same training protocol as the original 10M model. OATS outperforms regular training in both NLL and MAPE across all three evaluated datasets, for both prediction length 192 and the average over prediction lengths. Detailed results are provided in Appendix D.7.

4.3 Ablation Studies

In this section, we perform a variety of ablation studies to present the effect of several core components: validation set $\mathcal{D}_{val}$ (Section 4.3.1, TSIS guidance (Section 4.3.2), and generation strategy (Section 4.3.3).

Table 1: Performance comparison of various data augmentation methods on encoder-only TSFM with prediction length be 192 / average over all prediction lengths) and $\epsilon = 1$. **Bold** means the best result. Light green background means that the performance is better than regular training process, while light red background means that the performance is worse than regular training process. The error bar shows the standard error of mean over 5 independent runs.

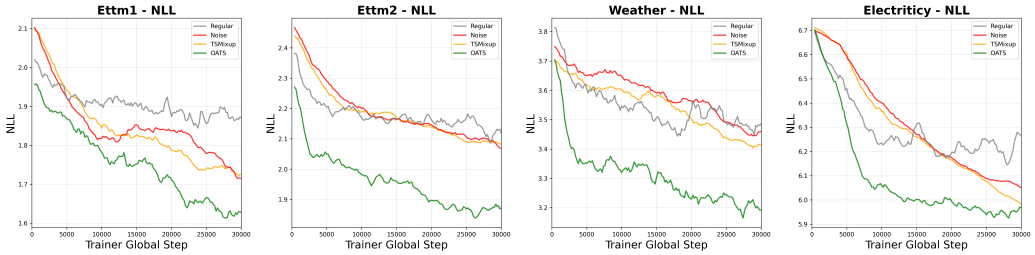
Dataset	Pred. length	OATS		TSMixup		Jitter		Regular	
		NLL	MAPE	NLL	MAPE	NLL	MAPE	NLL	MAPE
ETTm1	192	**1.627 ± 0.042**	**0.672 ± 0.043**	1.725 ± 0.031	0.759 ± 0.038	1.715 ± 0.047	0.691 ± 0.044	1.870 ± 0.019	0.844 ± 0.056
	Overall	**1.614 ± 0.020**	**0.623 ± 0.024**	1.715 ± 0.018	0.700 ± 0.019	1.731 ± 0.025	0.650 ± 0.019	1.854 ± 0.016	0.783 ± 0.037
ETTm2	192	**1.872 ± 0.014**	**0.208 ± 0.005**	2.083 ± 0.016	0.271 ± 0.006	2.068 ± 0.020	0.256 ± 0.007	2.118 ± 0.028	0.267 ± 0.009
	Overall	**1.863 ± 0.013**	**0.222 ± 0.004**	2.097 ± 0.011	0.292 ± 0.003	2.072 ± 0.009	0.273 ± 0.005	2.128 ± 0.014	0.290 ± 0.008
ETTh1	192	**1.794 ± 0.035**	0.681 ± 0.025	1.883 ± 0.030	0.634 ± 0.025	1.959 ± 0.028	**0.629 ± 0.035**	1.897 ± 0.040	0.758 ± 0.059
	Overall	**1.814 ± 0.020**	0.709 ± 0.021	1.860 ± 0.027	**0.634 ± 0.011**	1.913 ± 0.026	0.644 ± 0.021	1.915 ± 0.019	0.812 ± 0.031
ETTh2	192	**1.857 ± 0.063**	**0.267 ± 0.007**	2.123 ± 0.029	0.294 ± 0.011	2.127 ± 0.022	0.296 ± 0.013	2.084 ± 0.047	0.282 ± 0.015
	Overall	**1.876 ± 0.025**	**0.263 ± 0.005**	2.113 ± 0.020	0.294 ± 0.005	2.122 ± 0.022	0.298 ± 0.005	2.086 ± 0.021	0.283 ± 0.008
Weather	192	**3.193 ± 0.043**	**1.778 ± 0.127**	3.413 ± 0.041	2.016 ± 0.040	3.460 ± 0.043	2.619 ± 0.280	3.486 ± 0.051	2.331 ± 0.285
	Overall	**3.216 ± 0.036**	**1.659 ± 0.069**	3.428 ± 0.023	1.948 ± 0.033	3.499 ± 0.020	2.591 ± 0.145	3.526 ± 0.026	2.267 ± 0.125
Electricity	192	**5.967 ± 0.021**	**0.515 ± 0.071**	5.983 ± 0.010	0.553 ± 0.034	6.050 ± 0.015	0.659 ± 0.062	6.266 ± 0.019	0.750 ± 0.033
	Overall	**5.945 ± 0.011**	**0.507 ± 0.045**	5.987 ± 0.007	0.555 ± 0.013	6.044 ± 0.010	0.642 ± 0.025	6.260 ± 0.015	0.767 ± 0.021

Table 2: Performance comparison of various data augmentation methods on decoder-only TSFM with prediction length be 192 / average over all prediction lengths) and $\epsilon = 1$. **Bold** means the best result. Light green background means that the performance is better than regular training process, while light red background means that the performance is worse than regular training process. The error bar shows the standard error of mean over 5 independent runs.

Dataset	Pred. length	OATS		TSMixup		Jitter		Regular	
		NLL	MAPE	NLL	MAPE	NLL	MAPE	NLL	MAPE
ETTm1	192	**1.654 ± 0.018**	**0.659 ± 0.027**	1.691 ± 0.013	0.664 ± 0.027	1.767 ± 0.041	0.675 ± 0.031	1.737 ± 0.034	0.677 ± 0.022
	Overall	**1.740 ± 0.015**	**0.635 ± 0.014**	1.785 ± 0.025	0.649 ± 0.016	1.842 ± 0.034	0.650 ± 0.018	1.812 ± 0.022	0.653 ± 0.015
ETTm2	192	**1.726 ± 0.039**	**0.202 ± 0.003**	1.789 ± 0.014	0.206 ± 0.002	1.857 ± 0.044	0.222 ± 0.007	1.841 ± 0.009	0.217 ± 0.001
	Overall	**1.765 ± 0.030**	**0.228 ± 0.002**	1.824 ± 0.015	0.229 ± 0.002	1.880 ± 0.034	0.244 ± 0.008	1.874 ± 0.008	0.242 ± 0.004
ETTh1	192	**1.759 ± 0.046**	0.534 ± 0.022	1.933 ± 0.069	0.536 ± 0.017	1.808 ± 0.069	**0.504 ± 0.021**	1.852 ± 0.046	0.537 ± 0.026
	Overall	**1.824 ± 0.020**	0.562 ± 0.022	2.038 ± 0.031	0.568 ± 0.014	1.878 ± 0.030	**0.538 ± 0.016**	1.943 ± 0.025	0.563 ± 0.013
ETTh2	192	**1.817 ± 0.045**	**0.261 ± 0.011**	1.977 ± 0.036	0.289 ± 0.009	1.923 ± 0.036	0.275 ± 0.014	1.890 ± 0.050	0.284 ± 0.008
	Overall	**1.936 ± 0.026**	**0.274 ± 0.007**	2.087 ± 0.030	0.307 ± 0.005	1.994 ± 0.022	0.295 ± 0.008	2.027 ± 0.023	0.309 ± 0.006
Weather	192	**2.914 ± 0.081**	**2.885 ± 0.650**	2.974 ± 0.059	3.935 ± 0.334	3.025 ± 0.065	5.423 ± 0.822	2.914 ± 0.026	3.059 ± 0.622
	Overall	**3.168 ± 0.058**	**2.376 ± 0.323**	3.265 ± 0.039	3.643 ± 0.186	3.276 ± 0.037	4.624 ± 0.411	3.245 ± 0.041	2.946 ± 0.574
Electricity	192	**6.041 ± 0.017**	**0.528 ± 0.030**	6.074 ± 0.033	0.563 ± 0.010	6.090 ± 0.022	0.627 ± 0.028	6.049 ± 0.026	0.588 ± 0.020
	Overall	**6.040 ± 0.011**	**0.526 ± 0.014**	6.079 ± 0.018	0.564 ± 0.014	6.107 ± 0.019	0.631 ± 0.018	6.057 ± 0.015	0.579 ± 0.011

4.3.1 Effect of Validation Set

We evaluate OATS under two practical scenarios: a few-shot setting where the validation set is drawn from the target domain, and an in-domain setting where it is drawn from held-out training data. Table 3 shows that both settings outperform regular training. The few-shot setting provides stronger guidance as expected.



(a) ETTm1 (b) ETTm2 (c) Weather (d) Electricity

Figure 3: Test loss (NLL) of OATS, TSMixup, Jitter and Regular training for each training step.

Table 3: Effect of validation set source. "Few-shot" uses a small number of samples from the target domain as the validation set. "In-domain" uses held-out samples from the training data as the validation set. Cells are colored green if better than Regular and red if worse.

Dataset	Pred. length	OATS (few-shot)		OATS (in-domain)		Dataset	Pred. length	OATS (few-shot)		OATS (in-domain)	
		NLL	MAPE	NLL	MAPE			NLL	MAPE	NLL	MAPE
ETTm1	192	**1.627**	**0.672**	1.781	0.761	ETTh2	192	**1.857**	**0.267**	1.915	0.322
	Overall	**1.614**	**0.623**	1.728	0.670		Overall	**1.876**	**0.263**	1.910	0.312
ETTm2	192	1.872	0.208	**1.830**	**0.206**	Weather	192	**3.193**	**1.778**	3.304	1.821
	Overall	1.863	0.222	**1.838**	**0.221**		Overall	**3.216**	**1.659**	3.319	1.545
ETTh1	192	**1.794**	**0.681**	1.827	0.720	Electricity	192	**5.967**	**0.515**	6.032	0.545
	Overall	**1.814**	**0.709**	1.840	0.777		Overall	**5.945**	**0.507**	6.013	0.604

4.3.2 Effect of TSIS Guidance

To isolate the contribution of TSIS, we compare OATS with a variant that uses the same conditional diffusion model but with randomly selected prompts instead of TSIS-selected ones. As shown in Table 4, TSIS-guided generation outperforms random-guided generation in NLL across all three datasets, and in MAPE on ETTh1 and ETTm1. On Electricity MAPE, the two are comparable, suggesting that the benefit of TSIS guidance is more pronounced on the likelihood metric.

We further examine whether the generated samples retain favorable influence characteristics of their guiding samples. Across three training checkpoints, generated samples consistently achieve higher average TSIS than unselected training samples, suggesting that influence-guided generation can produce synthetic samples with favorable influence scores. Detailed results are provided in Appendix D.10.

4.3.3 Effect of Generation Strategy

We compare a number of augmentation strategies to understand how each design choice contributes to OATS's performance. Table 4 shows a clear progression: selection only improves over regular training but remains limited by data diversity; unconditional improves by introducing synthetic variety while suffering from low-quality samples; and TSIS-guided conditional generation achieves the best results by combining targeted selection with high-fidelity generation. In addition, we also included AutoTCL [Zheng et al., 2024] as a generation strategy baseline. We additionally compare OATS with [Taga et al., 2025], an online data selection and augmentation method in Appendix D.8.

We further evaluate the fidelity of generated samples. Compared with class-only conditional and unconditional generation, OATS achieves lower distributional discrepancies across all five evaluated subsets and all three metrics, demonstrating the effectiveness of high-quality sample guidance. Detailed results are provided in Appendix D.9.

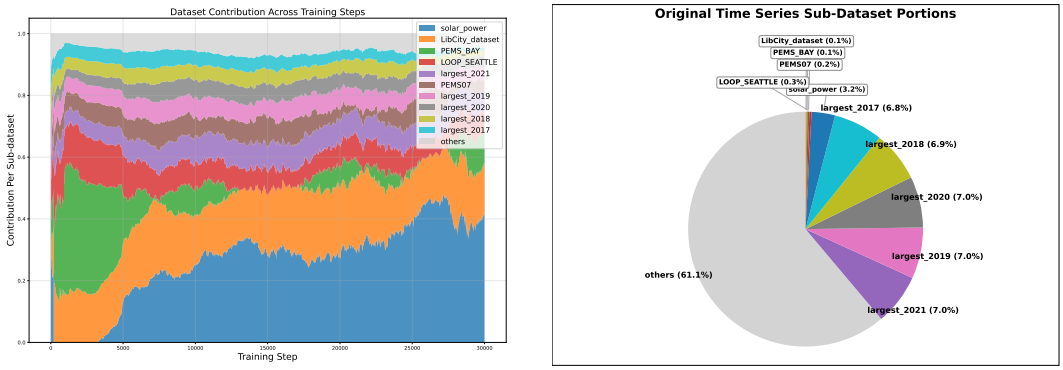
4.4 Case studies

In this subsection, we carry out two case studies to further provide intuitive evidence of our motivation and some insights.

Contribution of each sub-dataset. In the first case study, we present the per-sub-dataset TSIS, i.e., $\Phi_{\mathcal{D}_l}$, throughout the training process (Figure 4a), along with the original proportion of each sub-dataset in the training corpus (Figure 4b). The results reveal two key insights: (1) the contribution of each sub-dataset evolves during training, which intuitively motivates the use of online data augmentation; and (2) the contribution of a training sub-dataset does not necessarily align with its size. For instance, a relatively small sub-dataset (e.g., `solar_power`) can contribute substantially to reducing test loss, whereas some large sub-datasets (e.g., those grouped under `others`, which account for 60% of the total data in Figure 4b) provide only a limited contribution (around 10% in Figure 4a). Such findings are difficult to uncover through heuristic design or empirical intuition alone, but could be revealed through principled influence analysis.

Table 4: Ablation study on the effect of generation strategy and TSIS guidance. "Selection only" trains on TSIS-selected samples (with duplication if needed to fill the batch) without generation. "Unconditional" uses a single unconditional diffusion model. "Random-guided" uses the same conditional diffusion model but with randomly selected prompts instead of TSIS-selected ones. AutoTCL [Zheng et al., 2024] is a baseline generation method. `OATS` uses the conditional diffusion model with TSIS-selected prompts. Cells are colored green if better than Regular and red if worse.

Dataset	Pred. length	`OATS`		Random-guided		Selection only		Unconditional		AutoTCL	
		NLL	MAPE	NLL	MAPE	NLL	MAPE	NLL	MAPE	NLL	MAPE
ETTh1	192	**1.794**	0.681	1.831	0.767	1.874	0.793	1.811	**0.617**	1.811	0.933
	Overall	**1.814**	0.709	1.853	0.824	1.899	0.836	1.827	**0.632**	1.825	1.024
ETTm1	192	**1.627**	**0.672**	1.766	0.859	1.804	0.815	1.782	0.811	1.821	0.844
	Overall	**1.614**	**0.623**	1.729	0.778	1.742	0.703	1.723	0.732	1.753	0.739
Electricity	192	**5.967**	0.515	6.221	**0.470**	6.147	0.632	6.151	0.538	6.207	0.513
	Overall	**5.945**	0.507	6.174	0.509	6.156	0.648	6.114	0.576	6.114	**0.501**



(a) Sub-dataset TSIS ($\Phi_{\mathcal{D}_l}$) along the training.

(b) Sub-dataset portion in training dataset.

Figure 4: Comparison between sub-dataset data contribution and data portion.

Selected high-quality prompts and generated samples. In the second case study, we present the high-quality data samples prompts selected according to TSIS, and the corresponding guided generation in Figure 5. The results show that the generation model of the high-quality guided data augmentation could generate realistic data samples similar to the prompt with various patterns.

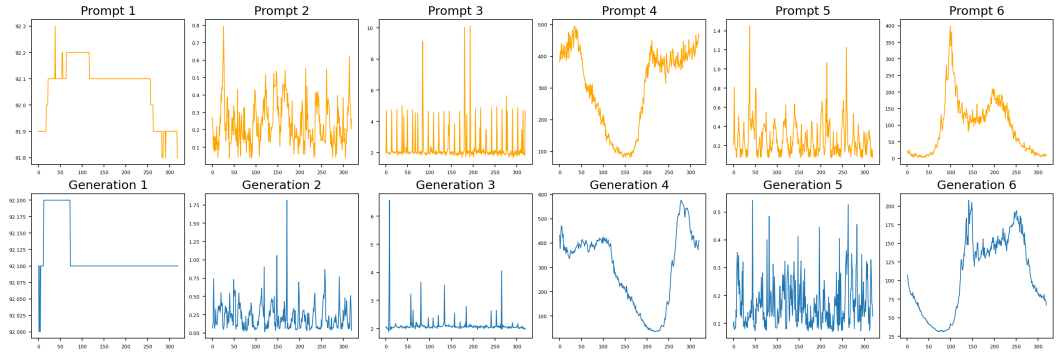


Figure 5: (**First row**) High-quality data samples prompt. (**Second row**) The corresponding guided generation.

5 Conclusion

In this paper, we introduce OATS, a principled strategy for generating synthetic data tailored to different training steps in TSFMs. The core idea behind OATS is to leverage valuable training samples as principled guiding signals and dynamically generate high-quality synthetic data conditioned on them. Empirical evaluations across diverse model architectures, datasets, and prediction lengths demonstrate that OATS substantially outperforms both standard data augmentation methods and regular training in terms of TSFM performance. By expanding the scope of online TSFM data augmentation, OATS enables researchers to systematically optimize TSFM training datasets in a principled way.

Limitation. OATS introduces additional computational overhead from TSIS gradient evaluation and diffusion-based sampling compared to regular training, though this can be mitigated in practice (see Appendix A). The method also depends on a small validation set to define data quality, and its current evaluation is limited to forecasting; generalization to other time series tasks such as classification and anomaly detection remains future work.

Acknowledgments and Disclosure of Funding

Funding. This work was conducted during Junwei Deng's research internship at Microsoft Research and was supported by Microsoft Research.

Competing Interests. Junwei Deng was a research intern at Microsoft Research during the conduct of this work. The authors declare no other competing interests.

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.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting, 2025. URL https://arxiv.org/abs/2510.15821.

Junwei Deng, Yuzheng Hu, Pingbang Hu, Ting-Wei Li, Shixuan Liu, Jiachen T. Wang, Dan Ley, Qirun Dai, Benhao Huang, Jin Huang, Cathy Jiao, Hoang Anh Just, Yijun Pan, Jingyan Shen, Yiwen Tu, Weiyi Wang, Xinhe Wang, Shichang Zhang, Shiyuan Zhang, Ruoxi Jia, Himabindu Lakkaraju, Hao Peng, Weijing Tang, Chenyan Xiong, Jieyu Zhao, Hanghang Tong, Han Zhao, and Jiaqi W. Ma. A survey of data attribution: Methods, applications, and evaluation in the era of generative ai. 2025. doi: 10.2139/ssrn.5451054. Available at SSRN: https://ssrn.com/abstract=5451054 or http://dx.doi.org/10.2139/ssrn.5451054.

Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095*, 2021.

Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha Naidu, and Colin White. Forecastpfn: synthetically-trained zero-shot forecasting. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

Lin Lawrence Guo, Ethan Steinberg, Scott Lanyon Fleming, Jose Posada, Joshua Lemmon, Stephen R Pfohl, Nigam Shah, Jason Fries, and Lillian Sung. Ehr foundation models improve robustness in the presence of temporal distribution shift. *Scientific Reports*, 13(1):3767, 2023.

Yu-Hao Huang, Chang Xu, Yueying Wu, Wu-Jun Li, and Jiang Bian. Timedp: Learning to generate multi-domain time series with domain prompts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 17520–17527, 2025.

Heikki Junninen, Harri Niska, Kari Tuppurainen, Juhani Ruuskanen, and Mikko Kolehmainen. Methods for imputation of missing values in air quality data sets. *Atmospheric environment*, 38 (18):2895–2907, 2004.

Raehyun Kim, Chan Ho So, Minbyul Jeong, Sanghoon Lee, Jinkyu Kim, and Jaewoo Kang. Hats: A hierarchical graph attention network for stock movement prediction. *arXiv preprint arXiv:1908.07999*, 2019.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

Kseniia Kuvshinova, Olga Tsymboi, Alina Kostromina, Dmitry Simakov, and Elizaveta Kovtun. Towards foundation time series model: To synthesize or not to synthesize? *arXiv preprint arXiv:2403.02534*, 2024.

Tian Lan, Hao Duong Le, Jinbo Li, Wenjun He, Meng Wang, Chenghao Liu, and Chen Zhang. Towards foundation models for zero-shot time series anomaly detection: Leveraging synthetic data and relative context discrepancy, 2025. URL `https://arxiv.org/abs/2509.21190`.

Junjie Li, Yang Liu, Weiqing Liu, Shikai Fang, Lewen Wang, Chang Xu, and Jiang Bian. Mars: a financial market simulation engine powered by generative foundation model. *arXiv preprint arXiv:2409.07486*, 2024.

Yuxuan Liang, Yutong Xia, Songyu Ke, Yiwei Wang, Qingsong Wen, Junbo Zhang, Yu Zheng, and Roger Zimmermann. Airformer: Predicting nationwide air quality in china with transformers. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 14329–14337, 2023.

Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. 2024 ieee. In *CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 5392–5399, 2023.

Xu Liu, Taha Aksu, Juncheng Liu, Qingsong Wen, Yuxuan Liang, Caiming Xiong, Silvio Savarese, Doyen Sahoo, Junnan Li, and Chenghao Liu. Empowering time series analysis with synthetic data: A survey and outlook in the era of foundation models. *arXiv preprint arXiv:2503.11411*, 2025.

Vladyslav Moroshan, Julien Siems, Arber Zela, Timur Carstensen, and Frank Hutter. Tempopfn: Synthetic pre-training of linear rnns for zero-shot time series forecasting, 2025. URL `https://arxiv.org/abs/2510.25502`.

Sai Shankar Narasimhan, Shubhankar Agarwal, Oguzhan Akcin, Sujay Sanghavi, and Sandeep Chinchali. Time weaver: A conditional time series generation model. *arXiv preprint arXiv:2403.02682*, 2024.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33: 19920–19930, 2020.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts. *arXiv preprint arXiv:2409.16040*, 2024.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *CoRR*, abs/2010.02502, 2020. URL https://arxiv.org/abs/2010.02502.

Ege Onur Taga, Halil Alperen Gozeten, Kutay Tire, Rahul Dalvi, Reinhard Heckel, and Samet Oymak. Filter, augment, forecast: Online data selection for robust time series forecasting. In *1st ICML Workshop on Foundation Models for Structured Data*, 2025. URL https://openreview.net/forum?id=YVsXgr8MUQ.

Terry T Um, Franz MJ Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM international conference on multimodal interaction*, pages 216–220, 2017.

Jiachen T Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data shapley in one training run. *arXiv preprint arXiv:2406.11011*, 2024a.

Jiachen T Wang, Dawn Song, James Zou, Prateek Mittal, and Ruoxi Jia. Capturing the temporal dependence of training data influence. *arXiv preprint arXiv:2412.09538*, 2024b.

Jiachen Tianhao Wang, Tong Wu, Dawn Song, Prateek Mittal, and Ruoxi Jia. Greats: Online selection of high-quality data for llm training in every iteration. *Advances in Neural Information Processing Systems*, 37:131197–131223, 2024c.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. 2024.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ju_Uqw384Oq.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023.

Shifeng Xie, Vasilii Feofanov, Marius Alonso, Ambroise Odonnat, Jianfeng Zhang, Themis Palpanas, and Ievgen Redko. Cauker: classification time series foundation models can be pretrained on synthetic data only, 2025. URL https://arxiv.org/abs/2508.02879.

Qingren Yao, Chao-Han Huck Yang, Renhe Jiang, Yuxuan Liang, Ming Jin, and Shirui Pan. Towards neural scaling laws for time series foundation models. *arXiv preprint arXiv:2410.12360*, 2024.

Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32, 2019.

Zahra Zamanzadeh Darban, Geoffrey I Webb, Shirui Pan, Charu Aggarwal, and Mahsa Salehi. Deep learning for time series anomaly detection: A survey. *ACM Computing Surveys*, 57(1):1–42, 2024.

Xu Zheng, Tianchun Wang, Wei Cheng, Aitian Ma, Haifeng Chen, Mo Sha, and Dongsheng Luo. Parametric augmentation for time series contrastive learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=EIPLdFy3vp.

A Explore & Exploit Mechanism

Ideally, $\mathcal{H}_t$ should be identified from the whole training dataset $\mathcal{D}_{tr}$ in Section 3.2, while the process of calculating TSIS for all training samples will be time-consuming. We are inspired by the multi-armed bandit problem and design an explore-exploit mechanism that could reuse the TSIS calculated for samples of previous training steps as well as explore new samples.

The mechanism divides training steps into *explore* steps and *exploit* steps. In *explore* steps, we calculate TSIS for a batch of data samples $\mathcal{B}_t$ sampled by strategy π_r. We assume the locality of TSIS among z in the same subset. Based on this assumption, we maintain a dynamic cache $\Phi_{\mathcal{D}_l} \in \mathbb{R}, l \in [L]$ for each subset and update it through exponentially moving average. The partition of $\mathcal{D}_{tr} = \bigcup_{l=1}^{L} \mathcal{D}_l$ can naturally be the sub-datasets of a large collection or some clustering results. The update of $\Phi_{\mathcal{D}_l}$ on training step t can be represented as

$$\Phi_{\mathcal{D}_l} = (1-\beta)\Phi_{\mathcal{D}_l} + \beta \sum_{z_i \in \mathcal{D}_l \cap \mathcal{B}_t} \mathcal{F}_{\mathcal{D}_{val}}(z_i, w_t)/|\mathcal{D}_l \cap \mathcal{B}_t| \tag{4}$$

where $\mathcal{B}_t$ is a sample of size $|\mathcal{B}_t|$ from π_r, and π_r is a uniform sampling strategy $\mathcal{U}(\mathcal{D}_{tr})$, $\beta \in [0,1]$ is the hyperparameter controls the decay factor of the exponentially moving average. The design of π_r for exploring steps reflects the spirit to visit new data points and update the TS data. It randomly samples from the full training dataset $\mathcal{D}_{tr}$. Additionally, we refer to the influence score part of the TSIS in this subsection when we use $\mathcal{F}_{\mathcal{D}_{val}}$.

For the *exploit* step, we design a sample algorithm π_o utilizing the $\Phi_{\mathcal{D}_l}$ cached in *explore* step. The design of π_o reflects the latest estimation of each subset's quality.

$$\pi_o = \mathcal{U}(\mathcal{D}_l) \quad \text{with probability } \frac{|\mathcal{D}_l| \cdot \max(0, \Phi_{\mathcal{D}_l})}{\sum_k |\mathcal{D}_k| \cdot \max(0, \Phi_{\mathcal{D}_k})} \tag{5}$$

The choice of *explore* or *exploit* step is handled by ϵ-greedy, a popular strategy for explore and exploit. We design a strategy ψ that is defined to be "explore" with probability ϵ and "exploit" with probability $(1-\epsilon)$, where ϵ is a hyperparameter controlling the balance between explore and exploit. Notably, the overhead of the *exploit* step is small enough to be ignored since TSIS is not calculated in such steps. A more careful complexity analysis is introduced in Appendix B.

A.1 Algorithm with Explore-exploit

We summarize Online Data Augmentation for Time Series Foundation Models (`OATS`) and present in Algorithm 2.

Algorithm 2 Online Data Augmentation for Time Series Foundation Models (`OATS`)

Require: Training dataset $\mathcal{D}_{tr} = \bigcup_{l=1}^{L} \mathcal{D}_l$, validation dataset $\mathcal{D}_{val}$, batch size b, augmentation algorithm $\mathcal{G}$, loss ℓ, strategy ψ, sampling algorithm π_r/π_o, TSIS function $\mathcal{F}_{\mathcal{D}_{val}}$.

Initialize model w_0
Initialize $\Phi_{\mathcal{D}_l}, l \in [L]$ to be subset proportion.
for $t = 1$ to T **do**
  if $\psi(t) ==$ "explore" **then**
    Sample $\mathcal{B}_t \sim \pi_r^b$ *// Step 1*
    Calculate $\mathcal{F}_{\mathcal{D}_{val}}(z_i, w_t)$ for $z_i \in \mathcal{B}_t$
    Update $\Phi_{\mathcal{D}_l}$ according to Eq. 4.
    Select subset $\mathcal{H}_t \subseteq \mathcal{B}_t$ with top-$\lfloor b/2 \rfloor$ of $\mathcal{F}_{\mathcal{D}_{val}}$.
    Generate $\mathcal{S}_t$ of size $\lfloor b/2 \rfloor$ via $\mathcal{G}$ guided by $\mathcal{H}_t$. *// Step 2*
  end if
  if $\psi(t) ==$ "exploit" **then**
    Sample $\mathcal{H}_t \sim \pi_o^{\lfloor b/2 \rfloor}$ *// Step 1*
    Generate $\mathcal{S}_t$ of size $\lfloor b/2 \rfloor$ via $\mathcal{G}$ guided by $\mathcal{H}_t$. *// Step 2*
  end if
  Update w_t on mini-batch $\mathcal{H}_t \cup \mathcal{S}_t$ to get w_{t+1}.
end for

A.2 Performance of OATS under Different Explore-Exploit Levels

The key experiment goal of this subsection is to examine the sensitivity and behavior of OATS on different levels of explore-exploit mechanisms through adjusting the value of ϵ. ϵ is a hyperparameter designed to control the possibility of carrying out *explore* (ϵ) or *exploit* step ($1-\epsilon$). *Explore* step will examine new data by calculating the TSIS ($\mathcal{F}_{\mathcal{D}_{val}}$), which is time-consuming. *Exploit* step will leverage cached TSIS ($\Phi_{\mathcal{D}_l}$) calculated in previous steps to directly sample a high-quality batch, which is light-weighted.

In Figure 6 we present the performance of OATS on different levels of explore-exploit mechanisms. We choose to set $\epsilon = [0.3, 0.5, 0.7, 1.0]$ and report the performance on two test datasets (ETTh1 and Electricity) and four prediction lengths (96, 192, 336, 720).

The best performance in each setting is often not achieved by $\epsilon = 1$ (all explore step), the most computationally heavy setting that calculates TSIS in every step. This shows that leveraging cached TSIS is helpful to select higher-quality batch (using π_o) to guide the data augmentation and demonstrates the potential of using the mechanism to reduce computational overhead. Another observation also shows that OATS consistently performs better than the regular training process regardless of the setting of ϵ. We included a detailed complexity and time cost analysis in Appendix B.

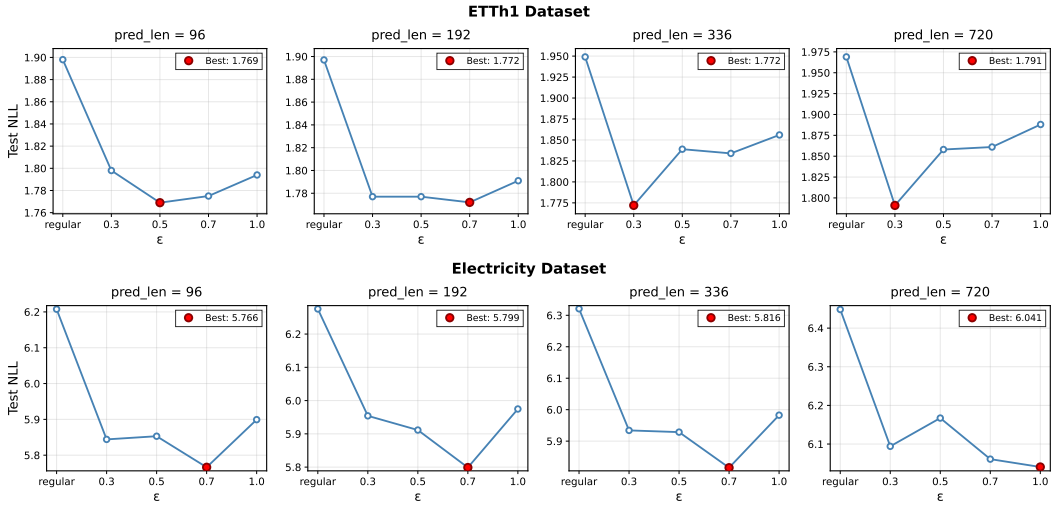


Figure 6: Performance of OATS on different explore-exploit ratio ϵ. (**First Row**) Test NLL on ETTh1. (**Second Row**) Test NLL on Electricity. The best performance is labeled by a red circle.

B Complexity Analysis

OATS involves computational overhead to the training process, which can be spitted into two sources: 1) calculation of TSIS and 2) synthetic data generation. **In summary**, the overhead of TSIS calculation is up to $\epsilon\times$ of the regular training step. The good performance shown in Figure 6 with $\epsilon = 0.3$ shows that this overhead could be relatively small. The overhead of synthetic data generation model is a fixed number, which gets lower as the regular training step has higher time cost, e.g., a larger TSFM with more parameters.

Calculation of TSIS For TSIS calculation, the overhead of Equation 1 mainly comes from the gradient calculation of validation data samples in "explore" steps. In our experiment, we use a small number of validation data samples (32) that matches the training batch size. This helps maintain the same computation complexity between TSIS and the regular gradient descent training step as $\mathcal{O}(b)$, where b is the training batch size. Additionally, we also adopted an efficient implementation called "Ghost Inner Product" [Wang et al., 2024a] to avoid per-sample gradient calculation while get the dot product result.

Lemma 2 (Ghost Inner Product)**.** *In the influence function calculation in TSIS, which includes sample-level gradient dot products, i.e.,* $\nabla_w \ell(z^{(1)}, w_t) \cdot \nabla_w \ell(z^{(2)}, w_t)$*. We may only perform the*

backpropagation once and calculate the gradient dot product. Ghost Inner Product perform the product layer by layer in a model and here we present the method for a simple linear layer.

$$\frac{\partial \ell^{(1)}}{\partial w} \cdot \frac{\partial \ell^{(2)}}{\partial w} = (a^{(1)} \otimes \frac{\partial \ell^{(1)}}{\partial s^{(1)}}) \cdot (a^{(2)} \otimes \frac{\partial \ell^{(2)}}{\partial s^{(2)}}) = ((a^{(1)})^\top a^{(2)})((\frac{\partial \ell^{(1)}}{\partial s^{(1)}})^\top \frac{\partial \ell^{(2)}}{\partial s^{(2)}}), \tag{6}$$

where $a^{(1)}$ and $a^{(2)}$ is the linear layer's input, $s^{(1)}$ and $s^{(2)}$ is the pre-activation output, and $\otimes$ indicates the outer product.

In TSIS, $\ell^{(1)} = \ell(z_i, w_t)$ in each training step t and $z_i \in \mathcal{B}_t$ and $\ell^{(2)} = \ell(\mathcal{D}_{val}, w_t)$. "Ghost Inner Product" enable the implementation of influence score in TSIS to carry out **only one additional backpropagation** on $\ell(\mathcal{B}_t, w_t) + \ell(\mathcal{D}_{val}, w_t)$ to get all terms in Equation 6 compared to regular training.

Furthermore, the overhead of "exploit" steps is small enough to be ignored since TSIS in Equation 1 is not used in those steps, **which means that the computational overhead of TSIS will be reduced by (1-ϵ) and memory footprint will be on par with the regular training**. In Figure 6, $\epsilon = 0.3$ also substantially outperform regular training process, which shows good potential of reducing overhead. Following table 5 is a time cost record for training batch size 32. It is worth noting that the actual implementation may affect the time cost.

Table 5: Per-training step time cost under different settings of Explore-exploit ratio ϵ.

ϵ	Regular	0.3	0.5	0.7	1.0
Avg. time cost per step (s) on A40 GPU	0.74	0.95	1.09	1.20	1.32

Synthetic data generation. For synthetic data generation through diffusion model sampling, it is hard to compare the complexity with the regular gradient descent training step. Empirically, we find that some accelerated sampling strategies like DDIM [Song et al., 2020] could perform well enough with very small sample steps. Moreover, the geneartion overhead will be a fixed term and the relative proportion will get lower as the TSFMs get larger, e.g., with more parameters. The overhead runtime of current synthetic data generation is 0.022s per synthetic sample, which is $0.022 \times 32 \times 0.5 = 0.35$s additional to the Table 5.

Empirical results at same training time. We also show the performance of `OATS` and baselines' performance under the **same training time as regular training**. `OATS` still outperform other baselines. It is also worth noting that with a limited training time, `OATS` may be more far away from converging.

Table 6: Performance (NLL) of `OATS` and baselines' performance under the same time cost.

Dataset	`OATS`(ϵ=0.3)	`OATS`(ϵ=0.5)	`OATS`(ϵ=0.7)	`OATS`(ϵ=1)	TSMixup	Jitter	Regular
Electricity	**6.00**	6.11	6.02	6.04	6.15	6.14	6.26
ETTh1	1.82	**1.81**	1.85	1.83	1.88	1.94	1.89

C Experiment Settings

TSFM Models. We conduct the experiment on the TSFM models with the same configuration as the encoder-only model with a 10M parameter size in Yao et al. [2024]. The model is a modified version of Moirai [Woo et al., 2024], introduced by Yao et al. [2024], which incorporates patch embedding, rotary positional embedding, and a mixture of distributions to better adapt to time series forecasting while preserving extensibility.

TSFM Training Process. We adopt a similar training setting as in Yao et al. [2024] for the experiment. We utilize the AdamW optimizer with a batch size of 32 and a maximum learning rate of 10^{-3} with a linear warm-up of 10^4 training steps, followed by cosine decay for the remaining 2×10^4 steps.

Datasets for TSFM. We pretrain the TSFM on the modified (balanced domain sample, quality filtering) LOTSA-100M dataset in the pretraining stage provided by Yao et al. [2024]. We take the native sub-dataset division as subsets in our experiment. These models are then evaluated on the out-of-distribution LSF dataset [Wu et al., 2023], using various prediction lengths (96, 192, 336) and the same preprocessing pipeline as in Yao et al. [2024]. The detailed information of evaluation datasets is stated in Table 7. The datasets are under Apache license 2.0 license.

Table 7: Evaluation datasets and properties.

Dataset	Domain	Frequency	# Prediction Length
ETTh1	Energy	H	96/192/336/720
ETTh2	Energy	H	96/192/336/720
ETTm1	Energy	15min	96/192/336/720
ETTm2	Energy	15min	96/192/336/720
Electricity	Energy	H	96/192/336/720
Weather	Climate	H	96/192/336/720

Generation Model. We leverage the architecture in Huang et al. [2025] as the backbone model. The denoising network of the diffusion model employed adopts a U-Net architecture comprising 4 up/down sampling blocks with 8 attention heads and the dimension of each head is 64. An additional class embedding with 64 dimension is added to the output of each block.

Generation Model Training Process. We train a Latent Diffusion Model for time series generation using 200 diffusion time steps with a linear noise schedule ranging from 0.0005 to 0.1. The model employs L1 loss as the training objective and a dropout probability of 0.5 to enable classifier-free guidance. We set the base learning rate to 0.001.

Datasets for Generation Model. We train the generation model for high-quality guided data augmentation described in Section 3 by a sampled dataset from the training dataset of TSFM. We sample 5% of the dataset in 20 selected subsets (Table 8) in LOTSA-100M as the training set of the diffusion model. The generation length of the diffusion model is set to 320, and the diffusion step is set to 200. We use DDIM for the sampling process with 20 steps.

Table 8: Training datasets of generation model.

Dataset	Domain	Frequency
CMIP6	Climate	6H
ERA5	Climate	H
CloudOpsTSF	CloudOTS	5T
Azure VM Traces 2017	CloudOTS	5T
Loop Seattle	Transport	5T
PEMS07	Transport	5T
PEMS Bay	Transport	5T
Q-Traffic	Transport	15T
Largest 2017	Transport	5T
Largest 2018	Transport	5T
Largest 2019	Transport	5T
Largest 2020	Transport	5T
Largest 2021	Transport	5T
Australian Electricity	Energy	30T
Buildings900K	Energy	H
Solar Power	Energy	4S
Favorita Sales	Sales	D
Wiki-Rolling	Web	D
LibCity	Transport	5T
OthersLOTSA	Energy	H

Hyperparameters. For experiments in Table 1 and Table 2, we set hyperparameters of TSMixup baseline to have $k = 2$ and $\lambda_0 \sim \mathcal{U}(0.1, 0.9)$, $\lambda_1 = 1 - \lambda_0$. For Jitter, we use $\sigma = 0.03$. For OATS, we fixed $\epsilon = 1$ in Table 1 and Table 2. Experiments with $\epsilon < 1$ will be included in Figure 6.

Computation resources. We run the experiment on A100 GPU and the full GPU hours is around 900 hours.

D Additional Results

In Figure 7, we present more examples between high-quality data sample prompts and the corresponding generated samples.

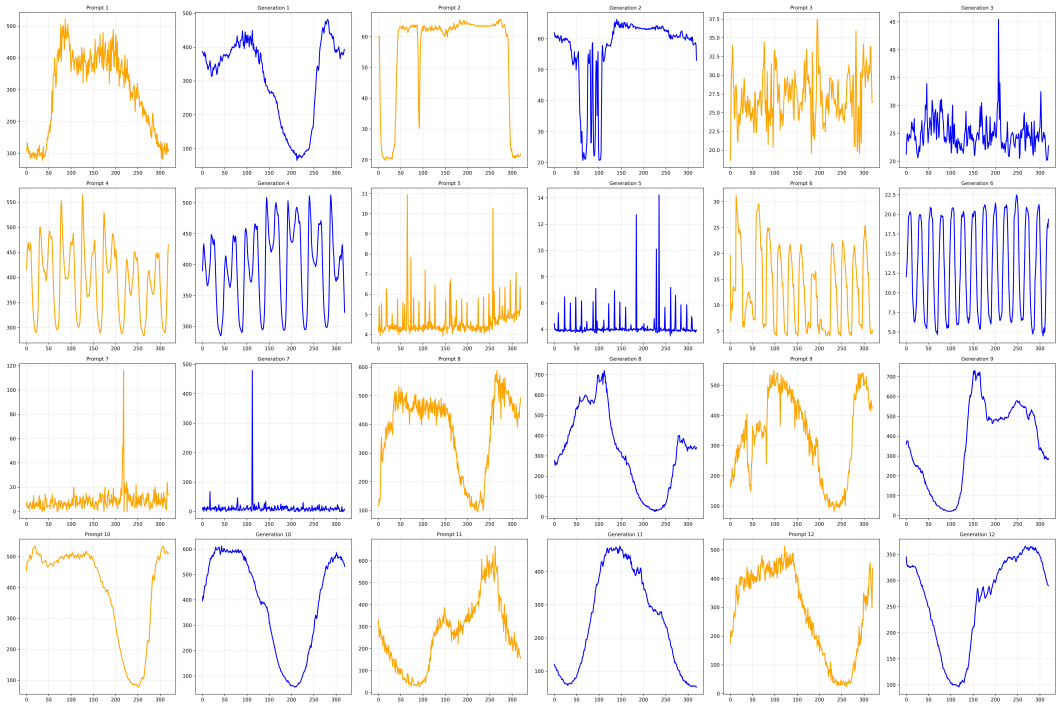


Figure 7: More examples of high-quality data samples prompts (yellow) and the corresponding guided generation (blue).

D.1 Improvement of class embedding on conditional generation.

Figures 8 and 9 demonstrate the improvement brought by class embeddings. In Figure 8, the samples generated without class embeddings (the following five) exhibit broken patterns that fail to align with the prompts (first in each row). In contrast, Figure 9 illustrates that class embeddings help the model achieve better structural alignment with the prompts.

D.2 Sensitivity of initial partition

In Table 9, we follow the native partition structure (sub-datasets) provided in the LOTSA dataset and evaluate different settings of initial data granularity. "Native Subdataset" refers to the default setting used in the paper, where we maintain the original partition structure. "Combine each two" represents a coarser granularity setting where we merge two native subdatasets into a single partition. Additionally, we include a baseline with random partitioning, where data samples are randomly assigned to partitions while maintaining the same total number of partitions as the native subdataset configuration.

We observe two key findings that align with our initial assumptions. First, we assume that the data exhibits locality within partitions based on semantic similarity (e.g., time series domains);

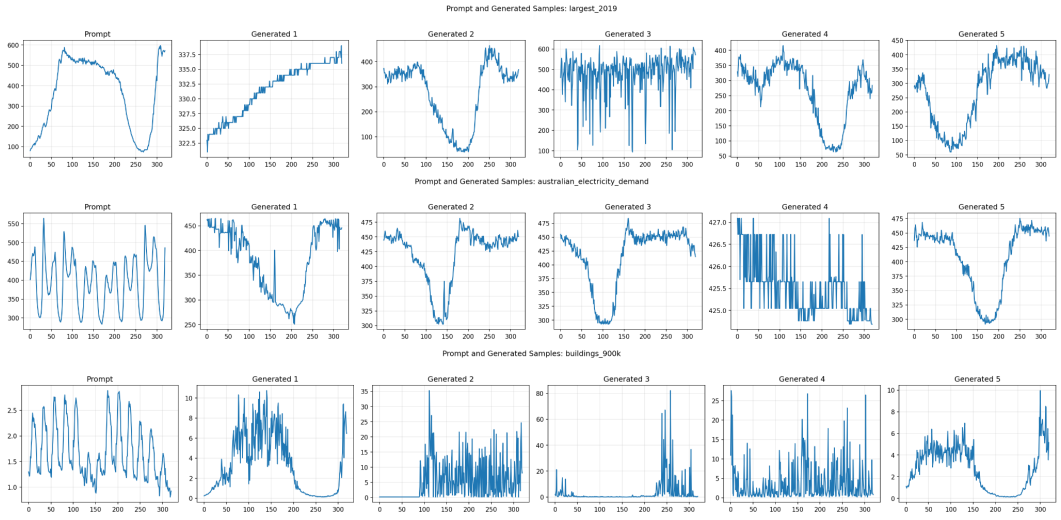


Figure 8: Examples of data generation of diffusion model **without** class embedding. For each row, the first sample is prompt and the following five are guided generations.

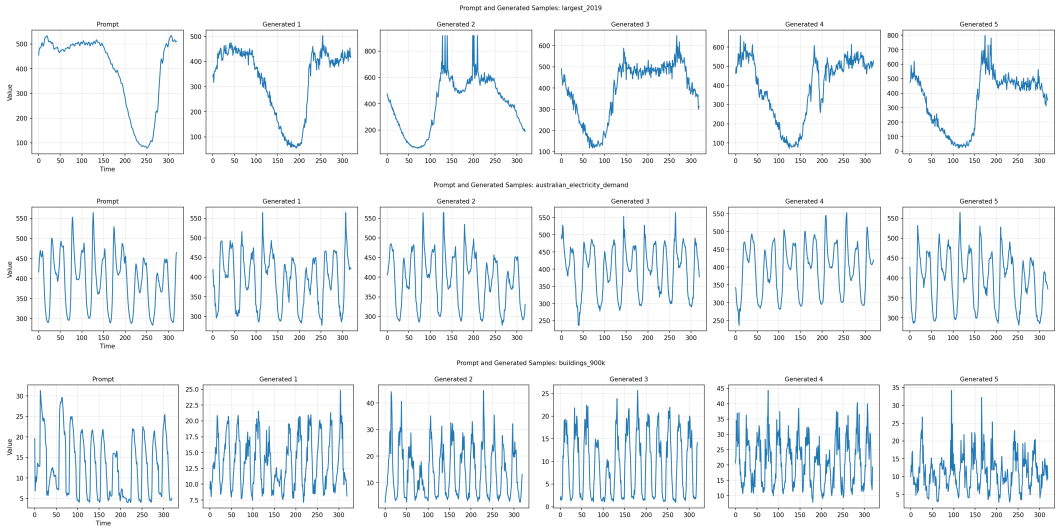


Figure 9: Examples of data generation of diffusion model **with** class embedding. For each row, the first sample is prompt, and the following five are guided generations.

consequently, random partitioning yields the poorest performance due to a lack of semantic coherence. Second, coarser granularity weakens the locality within each partition, thereby degrading performance relative to the "Native Subdataset" configuration.

D.3 Sensitivity to decay factor β

The performance in Table 10 is not very sensitive to the β setting as long as they are set to a reasonable range. In our experiment throughout the paper, we choose $\beta = 0.01$.

D.4 Sensitivity to SNR filter bar

We also conducted an experiment to evaluate the impact of the SNR threshold on performance in Table 11. This filtering mechanism is designed to eliminate noisy data points based on domain knowledge, thus complementing the influence score-based filtering. We observe that an overly strict threshold (e.g., 5dB) leads to the exclusion of excessive data, which degrades performance. Optimal

Table 9: Performance comparison of different granularity partitions.

Dataset	Pred. length	Combine each two		Native Subdataset		Random Partition	
		NLL	MAPE	NLL	MAPE	NLL	MAPE
ETTh1	96	1.837	0.782	**1.760**	**0.694**	1.965	0.752
	192	1.860	0.740	**1.766**	**0.682**	1.969	0.723
	336	1.885	0.834	**1.825**	**0.746**	1.998	0.832
	720	1.911	0.991	**1.853**	**0.947**	2.009	0.934

Table 10: Performance comparison of different beta values.

Dataset	Pred. length	β=0.1		β=0.01		β=0.001		β=0.0001	
		NLL	MAPE	NLL	MAPE	NLL	MAPE	NLL	MAPE
ETTh1	96	1.744	0.814	1.760	**0.694**	**1.688**	0.758	1.762	0.739
	192	1.729	0.727	1.766	**0.682**	**1.717**	0.714	1.754	0.704
	336	1.774	0.835	1.825	**0.746**	**1.753**	0.783	1.781	0.787
	720	1.807	1.061	1.853	**0.947**	**1.783**	0.962	1.793	0.953

results are achieved at relatively small values of k. While more sophisticated hyperparameter tuning strategies may exist, we leave such exploration for future work and adopt $k = 3$dB for the remainder of this paper.

Table 11: Performance comparison of different SNR values.

Dataset	Pred. length	k=1dB		k=3dB		k=5dB	
		NLL	MAPE	NLL	MAPE	NLL	MAPE
ETTm1	96	1.630	0.647	**1.557**	**0.549**	1.672	0.633
	192	1.709	0.764	**1.627**	**0.672**	1.753	0.800
	336	1.751	0.734	**1.658**	**0.641**	1.769	0.746
	720	1.785	0.762	**1.690**	**0.646**	1.775	0.733

D.5 Sensitivity to validation set selection

Table 12 presents the results for various validation set sizes. Throughout this paper, we use a fixed validation size of 32.

Table 12: Performance comparison of different batch sizes.

Dataset	Pred. length	Val size=8		Val size=32		Val size=128	
		NLL	MAPE	NLL	MAPE	NLL	MAPE
ETTm1	96	1.633	0.515	1.557	0.549	**1.506**	**0.505**
	192	1.730	0.728	1.627	0.672	**1.569**	**0.584**
	336	1.787	0.758	1.658	0.641	**1.644**	**0.587**
	720	1.825	0.807	1.690	**0.646**	**1.710**	0.672

D.6 Validation of TSIS Rankings

We empirically validate the fidelity of TSIS by comparing its rankings against single-step LOO influence, measured by the actual validation loss reduction after a single parameter update. We randomly sample 100 training examples and evaluate the correlation between their TSIS scores and actual influence.

We first evaluate three training checkpoints (steps 0, 500, and 3000). As shown in Table 13, TSIS achieves Spearman correlations ranging from 0.649 to 0.844, indicating reasonable agreement.

Table 13: Correlation between TSIS and single-step LOO influence at different training steps.

Training Step	Pearson	Spearman
0	0.7597	0.8436
500	0.6591	0.6488
3000	0.6994	0.7242

We further investigate the effects of learning rate and optimizer choice at step 1,000. Table 14 reports Spearman correlations averaged over five random seeds. Under the maximum learning rate of our run, TSIS achieves moderate correlations with both AdamW and SGD. Using the actual scheduled learning rate substantially improves the correlations to 0.926 and 0.966, respectively. The relatively small differences between AdamW and SGD suggest that the approximation error associated with the learning rate, rather than optimizer mismatch alone, accounts for much of the observed discrepancy.

Table 14: Spearman correlation between TSIS and single-step LOO influence under different learning rates and optimizers.

Learning Rate	AdamW	SGD
Maximum	0.614 ± 0.071	0.677 ± 0.043
Scheduled	0.926 ± 0.029	0.966 ± 0.008

D.7 Scalability to 100M Parameters

We evaluate the performance of OATS on a 100M-parameter encoder-only TSFM, following the same training protocol as the 10M model. The validation set consists exclusively of in-domain samples drawn from the training corpus, while evaluation is conducted on out-of-domain forecasting datasets.

As shown in Table 15, OATS consistently outperforms regular training in both NLL and MAPE across the three evaluated datasets, for prediction length 192 and averaged over all prediction lengths. These results demonstrate that the performance gains of OATS persist when scaling the TSFM from 10M to 100M parameters.

Table 15: Performance comparison between OATS and regular training on a 100M-parameter encoder-only TSFM using in-domain validation samples. Lower is better.

		NLL ↓		MAPE ↓	
Dataset	Pred. Length	OATS	Regular	OATS	Regular
ETTh1	192	**1.7082**	1.8793	**0.7806**	0.7994
ETTh1	Overall	**1.7216**	1.8777	**0.8438**	0.8573
ETTm1	192	**1.7011**	1.9287	**0.5895**	0.7992
ETTm1	Overall	**1.6919**	1.8684	**0.5678**	0.7570
Electricity	192	**5.9636**	6.1681	**0.5313**	0.6966
Electricity	Overall	**5.9220**	6.1757	**0.5504**	0.7410

D.8 Comparison with FAF

We further compare OATS with Taga et al. [2025], a closely related online data selection and augmentation method. We evaluate both methods on three forecasting datasets (ETTh1, ETTm1, and Electricity) with prediction length 192 and averaged over all prediction lengths.

As shown in Table 16, OATS consistently outperforms FAF in both NLL and MAPE across all six reported settings. These results further support the effectiveness of combining online influence-based selection with conditional synthetic data generation.

Table 16: Performance comparison between OATS and FAF. Lower values indicate better performance.

		OATS		FAF	
Dataset	Pred. Length	NLL	MAPE	NLL	MAPE
ETTh1	192	**1.794**	**0.681**	1.835	0.742
ETTh1	Overall	**1.814**	**0.709**	1.840	0.792
ETTm1	192	**1.627**	**0.672**	1.772	0.771
ETTm1	Overall	**1.614**	**0.623**	1.739	0.709
Electricity	192	**5.967**	**0.515**	6.434	0.753
Electricity	Overall	**5.945**	**0.507**	6.421	0.785

D.9 Quantitative Evaluation of Generation Fidelity

We quantitatively evaluate the fidelity of synthetic time series generated by OATS. Specifically, we employ three distributional metrics: MDD, Flat-KL, and MMD-RBF. Lower values indicate better distributional fidelity.

We select 100 reference samples from five subsets and compare three generation strategies: (1) OATS, which uses both high-quality sample guidance and class conditioning; (2) class-only conditional generation, which uses only subset information; and (3) unconditional generation, which does not use either condition.

As shown in Table 17, OATS achieves the lowest distributional discrepancy across all five subsets and all three metrics. These results provide quantitative evidence that high-quality sample guidance improves the fidelity of synthetic data beyond class conditioning alone.

Table 17: Quantitative evaluation of generation fidelity across five subsets. We compare OATS (O), class-only conditional generation (C), and unconditional generation (U). Lower values indicate better fidelity.

	MDD ↓			Flat-KL ↓			MMD-RBF ↓		
Dataset	O	C	U	O	C	U	O	C	U
Loop Seattle	**0.1316**	0.2435	0.2663	**0.0379**	0.8135	2.8674	**0.0294**	0.1806	0.5802
PEMS07	**0.2215**	0.2902	0.3599	**0.0842**	1.7195	3.1269	**0.0448**	0.2235	0.5908
Australian Electricity	**0.1948**	0.2033	0.3475	**0.0357**	0.0730	2.2330	**0.0343**	0.0420	0.6866
Buildings900K	**0.0152**	0.0993	0.1132	**0.0674**	0.7556	0.7705	**0.0016**	0.1805	0.2758
OthersLOTSA	**0.0431**	0.0712	0.0963	**0.1732**	3.6798	2.9223	**0.0158**	0.0397	0.0609
Overall	**0.1213**	0.1815	0.2366	**0.0797**	1.4083	2.3968	**0.0252**	0.1373	0.4389

D.10 Influence Scores of Generated Samples

We investigate whether synthetic samples generated by OATS preserve the favorable influence characteristics of their guiding samples. Specifically, we evaluate the TSIS of three groups: (1) high-quality training samples selected by TSIS, (2) unselected training samples, and (3) synthetic samples generated under the guidance of the selected samples. We evaluate these groups at three training checkpoints (steps 0, 1500, and 3000), with 16 samples in each group.

As shown in Table 18, generated samples consistently achieve higher average TSIS than unselected samples across all three checkpoints. At steps 0 and 3000, the generated samples even achieve higher average TSIS than the selected guiding samples. At step 1500, their average TSIS remains positive and higher than that of unselected samples.

Table 18: Average TSIS of selected training samples, unselected training samples, and generated samples at different training steps. Each group contains 16 samples. Higher values indicate more favorable influence scores.

Training Step	Selected	Unselected	Generated
0	4.206×10^{-11}	-8.743×10^{-12}	5.881×10^{-11}
1500	5.944×10^{-10}	-7.651×10^{-11}	2.117×10^{-10}
3000	7.414×10^{-11}	-5.070×10^{-11}	1.244×10^{-10}

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contribution and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are stated in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The full derivation is stated in Lemma 1.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have carefully included the experiment details in Appendix C and Section 4.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is provided in the url stated in paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We have carefully included the experiment details in Appendix C and Section 4.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide the standard error of mean in Section 4.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have carefully stated the compute resources in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Every author of this submission has reviewed the code of ethics guidelines and confirms compliance.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: We don't see immediate societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper has no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have carefully stated the licenses in Appendix C.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not include crowdsourcing and research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not include crowdsourcing and research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.