# CAUSAL TIME SERIES GENERATION VIA DIFFUSION MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Time series generation (TSG) synthesizes realistic sequences and has achieved remarkable success. Among TSG, conditional models generate sequences given observed covariates, however, such models learn observational correlations without considering unobserved confounding. In this work, we propose a causal perspective on conditional TSG and introduce *causal time series generation* as a new TSG task family, formalized within Pearl's causal ladder, extending beyond *observational* generation to include *interventional* and *counterfactual* settings. To instantiate these tasks, we develop **CaTSG**, a unified diffusion-based framework whose sampling is guided by backdoor-adjusted score functions targeting interventional and counterfactual distributions under an assumed SCM, while preserving observational fidelity. Extensive experiments on synthetic and real-world datasets show that **CaTSG** achieves superior fidelity, and in a fixed-checkpoint evaluation on a known-SCM synthetic system with a directly constructed interventional target, **CaTSG** attains the best score on every evaluated metric under multiple environment marginals. Overall, we formalize the causal TSG family and present **CaTSG** as a first instantiation, providing empirical evidence on synthetic and real-world settings and opening a direction for interventional and counterfactual generation. Our source code is available at `https://anonymous.4open.science/r/catsg-paper`.

## 1 INTRODUCTION

Time Series Generation (TSG) aims to synthesize realistic sequences that preserve temporal dependencies and inter-dimensional correlations, and is crucial for many applications, e.g., data augmentation (Ramponi et al., 2018), privacy preservation (Yoon et al., 2020) and simulation for downstream tasks (Ang et al., 2023). Conditional TSG generates sequences under given conditions, enabling controlled synthesis (Coletta et al., 2023). In most implementations, conditional TSG aligns generated data with the observational conditional $P(X \mid C)$ using correlation-fitting objectives, e.g., adversarial training in conditional Generative Adversarial Networks (GANs) (Yoon et al., 2019) and conditional diffusion with denoising score-matching objectives (Narasimhan et al., 2024). Yet, these approaches remain fundamentally *correlation-driven*: they learn statistical associations between conditions and outcomes without modeling the underlying data-generating mechanisms.

However, real-world time series systems are rarely governed by observed variables alone. Consider a transportation example (Figure 1a): a dataset contains observational pairs $(X, C)$, where $X$ is school-neighborhood traffic volume and $C$ is temperature. In reality, unobserved variables $E$ usually simultaneously affect both $X$ and $C$ (Figure 1b). For instance, summer holidays can be one such $E$, often coinciding with higher temperatures while also reducing commuting demand, thereby acting as a latent common cause. Given such a dataset, a typical conditional TSG model $p_\theta(X \mid C)$ (Figure 1c) cannot account for $E$ and may thus unknowingly capture *spurious correlations*, e.g., learning that "higher temperature causes lower traffic". Appendix E.3 provides empirical support for this motivation: on Harmonic-VM the conditional mean of the target given one context variable changes sign across environment splits while the relation with the other remains stable, with analogous split-dependent conditional shifts on both real-world datasets. This limits their applicability in real-world decision-making, where the goal may not be merely to replicate what happened, but to answer counterfactual queries such as: *"What would happen if we changed the condition within the same environment (i.e., with the latent factors held fixed)?"*



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

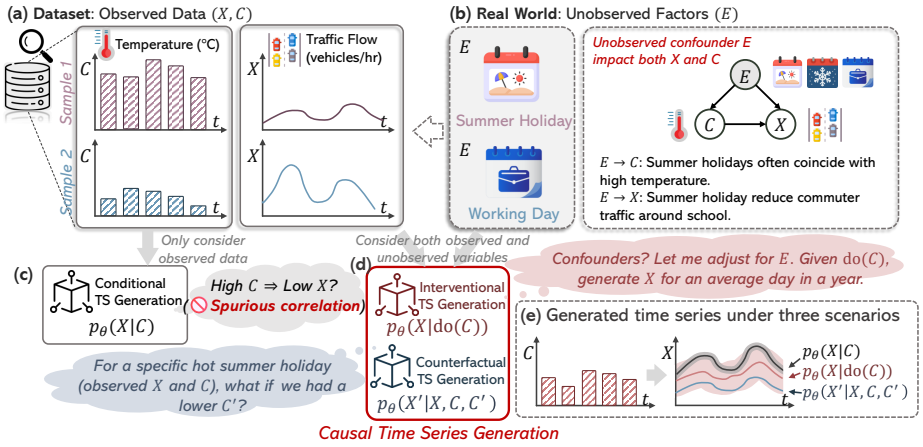



Figure 1: Motivation for causal time series generation. (a) Observed $X$ and $C$ may share (b) latent factors $E$, causing (c) conditional TSG to absorb spurious associations. (d) We extend it with Int-TSG and CF-TSG. (e) The three settings yield distinct outputs under the same condition.

Moving beyond purely observational modeling, we adopt a *causal view* that makes the mechanisms linking conditions and outcomes explicit. Pearl's causal ladder (Pearl, 2009) offers a precise vocabulary for this extension, organizing causal tasks into three levels: association, intervention, and counterfactual. Within this framing, classical conditional TSG learns the observational conditional $P(X \mid C)$, and we refer to this setting as observational TSG (Obs-TSG), which occupies the association level. We then formulate *causal time series generation* as an extension with two higher-level tasks: interventional (Int-TSG) and counterfactual TSG (CF-TSG), shown in Figure 1d. Specifically, Int-TSG generates samples from $P(X \mid \mathrm{do}(C))$, where $\mathrm{do}(\cdot)$ denotes an intervention operation that sets $C$ independent of its causes, thereby adjusting for latent confounding under the assumed SCM to simulate outcomes. CF-TSG generates samples from $P(X' \mid X, C, C')$, producing alternative trajectories by replacing $C$ with $C'$ while preserving learned factual environment information and resampling residual uncertainty. Conceptually, the outcomes under these tasks differ (Figure 1e): Obs-TSG (gray) may reflect spurious correlations with $C$; Int-TSG (red) shows outcomes adjusted for the latent-factor influences represented by the environment bank; CF-TSG (blue) depicts an alternative trajectory conditioned on learned factual environment information.

These tasks raise three challenges: (1) *ground-truth scarcity*, because real-world interventions and counterfactuals are rarely observed (Xia et al., 2025b; Matthay & Glymour, 2022); (2) *latent confounding*, because the relevant common causes are typically unobserved; and (3) *evaluation difficulty*, because time series quality depends on subtle distributional and temporal structure rather than straightforward visual inspection (Narasimhan et al., 2024).

To address these challenges, motivated by the growing adoption of diffusion models in time series practice (Ho et al., 2020; Yang et al., 2024), we introduce a diffusion-based framework **CaTSG** (**Ca**usality-grounded **T**ime **S**eries **G**eneration). **CaTSG** employs backdoor-adjusted guidance to enable interventional sampling without ground-truth labels, together with a learnable latent environment bank and an EnvInfer module that learns a finite representation of the confounding structure (*Challenges 1 and 2*). For *Challenge 3*, we complement real-world evaluations with SCM-grounded synthetic datasets that expose observable counterfactuals, enabling quantitative assessment of CF-TSG. Our contributions are summarized:

- **Causal Expansion of Conditional TSG Paradigm.** We formalize *causal time series generation* as an extension of conditional TSG along Pearl's ladder with interventional (Int-TSG) and counterfactual (CF-TSG), enabling richer generation for real-world decision making.
- **A Unified Causality-Guided Diffusion Framework.** We derive *causal score functions* for interventional and counterfactual generation and integrate them into diffusion sampling through backdoor adjustment, abduction-action-prediction, and a learned finite environment representation.
- **Comprehensive Empirical Evaluation.** On point estimates across four datasets, **CaTSG** leads 14 of 16 dataset-metric comparisons with gains of 3.3%-91.8% over the best baseline where it leads, and 12 of those leads have non-overlapping intervals. On a known-SCM system, it performs best against the directly constructed interventional target across all evaluated metrics and environment marginals. For CF-TSG, it leads its ablations on synthetic data and is further assessed through a Traffic case study and quantitative Air Quality diagnostics.

Table 1: Three levels of TSG following the causal ladder (Pearl, 2009). Color indicates semantic roles of variables: blue for generated outcomes $X$, red for observed condition $C$, green for counterfactual condition $C'$, yellow for counterfactual outcomes $X'$, and gray for latent or unobserved confounders (e.g., summer holiday).

| Causal Level | Description | Example in Transportation | Time Series Generation Task |
|---|---|---|---|
| *Level 1:* Association | **Seeing**: Observe statistical correlations | High temperature ⇒ Low traffic flow (! May capture **spurious correlations**, because both may be caused by summer holiday) | **Obs-TSG**: $P(X \mid C)$ Generate time series $X$ under condition $C$ |
| *Level 2:* Intervention | **Doing**: Generate under controlled change of variables | Generate traffic flow when we set temperature = high, regardless of summer or not | **Int-TSG**: $P(X \mid \text{do}(C))$ Generate time series $X$ given $C$, averaging over unobserved common causes |
| *Level 3:* Counterfactual | **Imagining**: Ask "what if" on a real sample | Observed traffic flow on a hot summer holiday ⇒ what would the counterfactual traffic flow have been if the same holiday had been cool instead? | **CF-TSG**: $P(X' \mid X, C, C')$ Given observed $(X, C)$, generate what $X'$ would look like under $C'$ |

## 2 Background

**Conditional Time Series Generation.** Conditional TSG synthesizes realistic time series matching specified auxiliary information. Early conditional GANs (Esteban et al., 2017; Smith & Smith, 2020) conditioned on discrete labels but suffered from instability and mode collapse (Chen, 2021). Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) have since become the prevalent choice (Yang et al., 2024), injecting side information into the reverse process for stable generation (Tashiro et al., 2021), with extensions to heterogeneous metadata (Narasimhan et al., 2024), structured state-space backbones (Alcaraz & Strodthoff, 2023), score-based transformers (Yuan & Qiao, 2024), and constrained objectives (Coletta et al., 2023). Recent work further pursues cross-domain adaptability via prompt-based control (Huang et al., 2025), dynamics-condition disentanglement (Bao et al., 2024), and textual guidance (Li et al., 2025).

**Diffusion Models & Score Function.** Diffusion probabilistic models (Ho et al., 2020; Song & Ermon, 2019) define a generative process by reversing a fixed forward noising process. Diffusion models can also be interpreted as score-based generative models (Song & Ermon, 2019), The score function estimates the gradient of the log-density of noisy samples: $s_t(x_t) = \nabla_{x_t} \log p(x_t)$. Under the Gaussian forward process, the data score satisfies $\nabla_{x_t} \log q(x_t \mid x_0) = -\frac{1}{\sigma_t}\epsilon, \quad \sigma_t = \sqrt{1-\bar{\alpha}_t}$. Accordingly, learning to predict the noise is equivalent to estimating the score up to a time-dependent scaling factor. In the conditional setting, the score becomes

$$s_t(x_t, c) = \nabla_{x_t} \log p(x_t \mid c) \;\approx\; -\frac{1}{\sigma_t}\varepsilon_\theta(x_t, t, c), \tag{1}$$

which links the denoising network directly to the conditional score function at each timestep. Due to the space limitation, we put preliminaries with more detail in Appendix B.

**Causal Modeling for Generation.** Causal views have entered deep learning across representation learning (Yang et al., 2021), vision-language grounding (Wang et al., 2025), and anomaly detection (Xia et al., 2025c). For generation (Komanduri et al., 2023), vision models such as CausalGAN (Kocaoglu et al., 2018), CausalDiffAE (Komanduri et al., 2024), and Counterfactual Generative Networks (Sauer & Geiger, 2021) enable counterfactual synthesis via interventions on disentangled factors, with extensions to time-varying treatment effects (Wu et al., 2024) and decoupled classifier-free guidance against spurious shifts (Xia et al., 2025a). Beyond vision, SCMs intervene on latent factors in text generation (Hu & Li, 2021); Neural Causal Models (Xia et al., 2021; 2023a) further extend identifiability to NN-based SCMs. Yet causal generative modeling remains largely unexplored for time series, despite well-developed causal analysis via Granger causality (Granger, 1969) and temporal SCMs (Peters et al., 2013; Assaad et al., 2022) (see Appendix H.7).

## 3 A Causal View for Conditional TSG

Traditional approaches to conditional TSG focus solely on observational correlations. To move beyond, we adopt a causal perspective and situate this generative task within Pearl's causal ladder (Pearl, 2009), thereby motivating a structured view across different causal levels.

**Causal Ladder for TSG.** Pearl's Causal Ladder organizes reasoning into three levels: association, intervention, and counterfactual. We adapt this hierarchy to the context of TSG, summarized in Table 1. *Level 1 (association)* captures statistical correlations between observed contextual variables $C$ (e.g., temperature) and the outcome $X$ (e.g., traffic); *Level 2 (intervention)* models the outcome

under controlled changes to $C$ while adjusting for potential confounding from unobserved common causes; and *Level 3 (counterfactual)* simulates alternate outcomes $X'$ had $C$ been different, given the observed sample $X$ and $C$. This structured perspective enables us to unify and extend TSG tasks along the causal hierarchy.

**General Problem Statement.** Let $\mathcal{D} = \{(\mathbf{x}, \mathbf{c})\}_{i=1}^{N}$ denote a dataset consisting of $N$ samples of multivariate time series $\mathbf{x} \in \mathbb{R}^{T \times D}$ and paired contextual sequences $\mathbf{c} \in \mathbb{R}^{T \times D_c}$. Each $\mathbf{x} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T]$ represents a length-$T$ time series over $D$ dimensions, and $\mathbf{c}$ contains auxiliary features in the same time span with dimension $D_c$. **Our goal** is to learn a generative model $\mathcal{F}_\theta$ such that the samples generated from $\mathcal{F}_\theta(\mathcal{I})$ match a target distribution of the form $P(\mathbf{x}^\star \mid \mathcal{I})$, where the form of the conditioning variable $\mathcal{I}$ determines the underlying generation task. Specifically, we consider:

- **Observational TSG (Obs-TSG):** $P(\mathbf{x} \mid \mathbf{c})$, the standard conditional time series generation task;
- **Interventional TSG (Int-TSG):** $P(\mathbf{x} \mid \mathrm{do}(\mathbf{c}))$, which simulates outcomes under an intervention on $\mathbf{c}$, where the $\mathrm{do}(\cdot)$ operator sets $\mathbf{c}$ independent of its causes;
- **Counterfactual TSG (CF-TSG):** $P(\mathbf{x}' \mid \mathbf{x}, \mathbf{c}, \mathbf{c}')$, which simulates an alternative outcome under a counterfactual context $\mathbf{c}'$ given an observed realization $(\mathbf{x}, \mathbf{c})$.

This causal view extends conditional TSG from a purely observational setting to a structured hierarchy of generative tasks. We refer to the latter two settings as *causal time series generation*.

# 4 METHODOLOGY

To instantiate $\mathcal{F}_\theta$ for the causal generative tasks, we first introduce a unified Structural Causal Model (SCM) (Pearl, 2000) as the theoretical foundation (Figure 2). We then derive the interventional and counterfactual objectives using two classical tools (Pearl et al., 2016): *backdoor adjustment* for Int-TSG and *abduction-action-prediction* (AAP) procedure for CF-TSG. Given that diffusion models (Ho et al., 2020) have become a powerful technique for time series modeling with strong performance and stable training (Yang et al., 2024), we adopt it as the backbone. Guided by the causal principles, within the diffusion framework, we derive a *causal score function* that guides the sampling process, and introduce **`CaTSG`** (Figure 3), a generative model that instantiates causal objectives in practice.

## 4.1 SCM & CAUSAL TREATMENTS

Beyond the observational target of Obs-TSG (Figure 2a), we consider an SCM that captures the causation among observed sequences $X$, contextual variables $C$, and unobserved environments $E$ (Figure 2b), which supports the formulation of interventional and counterfactual targets. In our formulation, $C$ and $X$ are sequence-level variables representing entire trajectories and $\mathrm{do}(C)$ corresponds to setting the entire contextual trajectory. This is a deliberate scope choice, so the formulation does not cover time-point-level or lag-specific interventions, and Appendix H.7 states what an extension to time-indexed structural equations would require. More details on these causal concepts and derivations are provided in Appendix B.1.

**Backdoor Adjustment for Int-TSG** (Figure 2c). To estimate the interventional distribution $P(X \mid \mathrm{do}(C))$, we apply the *backdoor adjustment* (Pearl et al., 2016), which controls for confounding by conditioning on the appropriate set of variables that block all backdoor paths from $C$ to $X$, i.e., $C \leftarrow E \rightarrow X$. This decomposition relies on the following assumptions: (A1) the SCM (Figure 2b) is correctly specified; (A2) $E$ captures all common causes of $C$ and $X$. Under A1-A2, and given a sufficient adjustment representation of $E$, the interventional target admits the backdoor decomposition below. **`CaTSG`** operationalizes this decomposition using a learned finite environment representation, which approximates $E$ rather than recovering it. Appendix H.3 states the scope of this decomposition, and Appendix H.4 enumerates the consequences when A1-A2 fail. The interventional distribution can thus be approximated via (full derivation in Eq. 9):

$$P(X \mid \mathrm{do}(C)) = \sum\nolimits_e P(X \mid C, E = e)\, P(E = e) \tag{2}$$

**Abduction-Action-Prediction (AAP) for CF-TSG** (Figure 2d). To estimate the counterfactual distribution $P(X' \mid X, C, C')$, which asks "what would the outcome $X'$ have been if $C$ had been $C'$ instead," we adopt the three-step procedure (Pearl et al., 2016): ***Abduction:*** Infer the posterior over latent environment $E$ from the observed sample $(X, C)$, i.e., estimate $P(E \mid X, C)$. ***Action:*** Replace the observed $C$ with a counterfactual value $C'$ by simulating an intervention, i.e., modify the SCM by

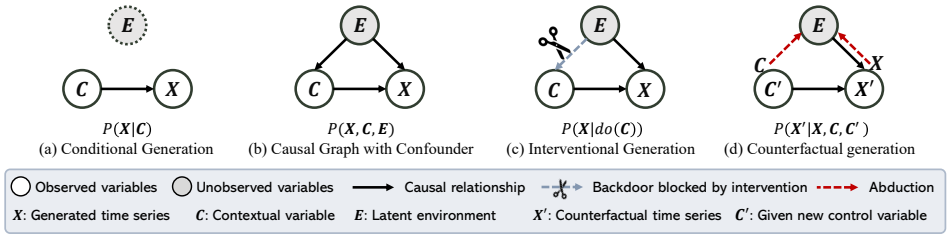



Figure 2: Structural Causal Models (SCMs) for different conditional time series generation paradigms.

applying $\mathrm{do}(C = C')$. ***Prediction:*** Generate the counterfactual outcome $X'$ based on the modified SCM and the inferred $E$, yielding $P(X' \mid \mathrm{do}(C'), E)$. We adopt a distributional counterfactual notion that preserves the inferred latent environment while resampling residual generative uncertainty. For example, in the transportation scenario, suppose we observe low traffic flow $X$ on a hot day $C$. We first infer that the latent environment $E$ likely corresponds to a summer holiday based on our observation $X$ and $C$ (*abduction*); then replace $C$ with a counterfactual condition $C'$, such as 10°C (*action*); finally, we predict the counterfactual traffic $X'$ as if $C = C'$, while keeping $E$ (summer holiday) fixed (*prediction*).

### 4.2 Causal Score Functions

Building on these causal treatments, we next instantiate these formulations within a generative model. Diffusion models (Ho et al., 2020) have recently demonstrated strong performance in time series generation (Yang et al., 2024; Lin et al., 2024), where the denoising network learns to approximate the score function, i.e., the gradient of the log-density of noisy samples $s_t(x_t) = \nabla_{x_t} \log p(x_t)$. Extending this principle, we replace the standard conditional score $s_t(x_t, c) = \nabla_{x_t} \log p(x_t \mid c)$ in Eq. 1 with the following two causal counterparts:

$$\begin{aligned} s_t^{\mathrm{int}}(x_t; \mathrm{do}(c)) &\triangleq \nabla_{x_t} \log p(x_t \mid \mathrm{do}(c)), \\ s_t^{\mathrm{cf}}(x_t'; x, c, c') &\triangleq \nabla_{x_t'} \log p(x_t' \mid x, c, c'), \end{aligned} \tag{3}$$

where $s_t^{\mathrm{int}}$ and $s_t^{\mathrm{cf}}$ are the interventional and counterfactual score functions. Under our SCM (Figure 2b), these can be derived using backdoor adjustment (Eq. 2) and AAP:

$$s_t^{\mathrm{int}} = \nabla_{x_t} \log p(x_t \mid \mathrm{do}(c)) \approx -\frac{1}{\sigma_t} \mathbb{E}_{e \sim r_{e,t}(x_t, c)} \left[ \varepsilon_\theta(x_t, t, c, e) \right] \quad (\omega = 0) \tag{4}$$

$$s_t^{\mathrm{cf}} = \nabla_{x_t'} \log p(x_t' \mid x, c, c') \approx \mathbb{E}_{e \sim p(e \mid x, c)} \left[ \nabla_{x_t'} \log p_t(x_t' \mid c', e) \right] \tag{5}$$

where $\varepsilon_\theta(\cdot)$ is the denoising network parameterized by $\theta$, $\sigma_t$ as in Eq. 1, and $\omega \geq 0$ is the guidance scale. These expressions describe the unguided mixture the SCM motivates, at $\omega = 0$, while the reported generation-quality results use a null-context guidance adjustment with $\omega = 0.2$. Here $r_e(x, c) \propto p(x \mid c, e)p(e)$ is the clean-space responsibility under Eq. 2, and $r_{e,t}$ is its noised-space analogue at level $t$, with $r_{e,0} = r_e$. The underlying mixture identity, the $r_{e,t}$-weighted expectation of the true component scores, is exact under the assumed SCM. Eq. 4 states its $\varepsilon_\theta$-parameterized form, hence the $\approx$. Eq. 5 further replaces the state-dependent AAP responsibilities $p(e \mid x_t', x, c, c')$ by fixed factual weights. **`CaTSG`** operationalizes both with learned quantities: a shared denoiser for the environment-conditioned predictions, and EnvInfer for $q_\phi(e \mid x_t, c)$ as a proxy for $r_{e,t}$ and $q_\phi(e \mid x_0, c)$ as the factual weights. Appendix G.1 compares the generated distribution against a directly constructed interventional target. Full derivations are given in Appendix C.2.

**Practical Discretized Implementation**. In practice, we discretize the continuous latent environment $e$ into a finite set of embeddings $\{\mathbf{e}_1, \ldots, \mathbf{e}_K\}$ with $\mathbf{e}_k \in \mathbb{R}^H$, and estimate their mixture weights as $\{w_1, \ldots, w_K\}$ with $w_k \in \mathbb{R}$. Denoting the environment-conditioned noise prediction $\varepsilon_\theta(\mathbf{x}_t, t, \mathbf{c}, \mathbf{e}_k)$ and all-null-context noise prediction $\varepsilon_\theta(\mathbf{x}_t, t)$ from the denoising network $\varepsilon_\theta(\cdot)$ as $\boldsymbol{\epsilon}^{\mathrm{env}}$ and $\boldsymbol{\epsilon}^{\mathrm{base}}$, it leads to the following *unified approximation* of the guided noise predictions:

$$\hat{\boldsymbol{\epsilon}}_t^{\mathrm{int}} \text{ or } \hat{\boldsymbol{\epsilon}}_t^{\mathrm{cf}} \;\approx\; (1 + \omega) \sum\nolimits_{k=1}^{K} w_k \, \boldsymbol{\epsilon}_k^{\mathrm{env}} - \omega \, \boldsymbol{\epsilon}^{\mathrm{base}}, \tag{6}$$

The corresponding guided score estimate is $\hat{s}_t = -\hat{\boldsymbol{\epsilon}}_t / \sigma_t$, while the sampler uses $\hat{\boldsymbol{\epsilon}}_t$ directly.

### 4.3 A Unified Causality-Guided Diffusion Framework

A conditional generator $\mathcal{F}_\theta$ can target different distributions (Figure 3a). Obs-TSG models $P(X \mid C)$ with $\mathcal{F}_\theta^{\mathcal{O}}$, whereas Int-TSG and CF-TSG require a causal generative model $\mathcal{F}_\theta^{\mathcal{C}}$. Building on the

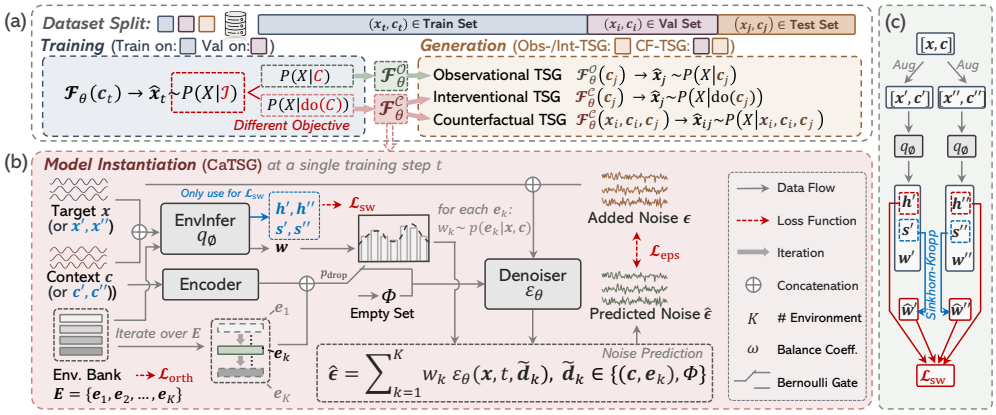



Figure 3: Overview of **CaTSG**. The panels show, in order, the generation objectives, their **CaTSG** instantiation, and environment-assignment training. (a) Observational, interventional, and counterfactual TSG objectives. (b) **CaTSG** with EnvInfer, the Environment Bank, and the Denoiser. (c) Swapped prediction loss $\mathcal{L}_{\text{sw}}$ for environment assignment.

causal guidance of Eq. 6, we instantiate these objectives with **CaTSG**, a unified diffusion framework in which omitting the environment component recovers standard Obs-TSG.

**Framework Overview** (Figure 3b). **CaTSG** comprises EnvInfer $q_\phi$, a learnable environment bank $\mathbf{E}$, and a Denoiser $\varepsilon_\theta$. EnvInfer maps $(\mathbf{x}, \mathbf{c})$ to soft assignments over $\mathbf{E}$, the bank is regularized by $\mathcal{L}_{\text{orth}}$, and the Denoiser predicts environment-conditioned and all-null-context noise under a Classifier-Free Guidance (CFG) scheme (Ho & Salimans, 2022). Architectural details are provided in Appendix C.4.

**Training Objective.** We jointly optimize the Denoiser $\varepsilon_\theta$, EnvInfer $q_\phi$, and the environment bank $\mathbf{E}$. For the Denoiser we use CFG-style training: independently for every temporal token in every environment branch, the fused context is replaced by a learned null context with probability $p_{\text{drop}} = 0.2$. Let $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ be the forward-process noise and $w_k$ the environment weights from EnvInfer. Each training example evaluates all $K$ environment-conditioned branches; their predictions are mixed using mini-batch-balanced weights $\tilde{\mathbf{w}} = \mathrm{Sinkhorn}_{\epsilon=0.05,\, n=3}(\log \mathrm{clamp}(\mathbf{w}, 10^{-12}))$, giving

$$\mathcal{L}_{\text{eps}} = \mathbb{E}_{\mathbf{x}_0, t, \boldsymbol{\epsilon}, \tilde{\mathbf{d}}} \left[ \left\| \boldsymbol{\epsilon} - \sum\nolimits_{k=1}^{K} \tilde{w}_k \varepsilon_\theta(\mathbf{x}_t, t, \tilde{\mathbf{d}}_k) \right\|_2^2 \right], \tag{7}$$

where $\tilde{\mathbf{d}}_k$ denotes the token-wise masked fused context for branch $k$; the objective supervises the $\tilde{\mathbf{w}}$-weighted aggregate of the $K$ branch predictions. EnvInfer learns soft assignments over the $K$ environment embeddings with a SwAV-style swapped prediction objective (Caron et al., 2020), in which the clean sample and its two augmentations ($V = 3$ views) predict one another's balanced Sinkhorn-Knopp assignments $\hat{\mathbf{w}}^{(u)}$ (Cuturi, 2013) (Figure 3c),

$$\mathcal{L}_{\text{sw}} = \big(V(V-1)\big)^{-1} \sum\nolimits_{u \neq v} \ell(\mathbf{w}^{(v)}, \hat{\mathbf{w}}^{(u)}), \tag{8}$$

where $\ell(\cdot, \cdot)$ denotes cross-entropy. Writing the environment bank as $\mathbf{E} \in \mathbb{R}^{K \times H}$ and its row-normalized embeddings as $\bar{\mathbf{e}}_k$, we regularize it with $\mathcal{L}_{\text{orth}} = \sum_{k \neq \ell} (\bar{\mathbf{e}}_k^\top \bar{\mathbf{e}}_\ell)^2$, yielding $\mathcal{L} = \mathcal{L}_{\text{eps}} + \alpha \mathcal{L}_{\text{sw}} + \beta \mathcal{L}_{\text{orth}}$ with balance coefficients $\alpha$ and $\beta$. Algorithm 1 and Appendix D detail training, and Appendix C.4 the assignment computation.

**Generation Procedures.** At each reverse step, the Denoiser returns an all-null-context prediction $\boldsymbol{\epsilon}^{\text{base}}$ and environment-conditioned predictions $\boldsymbol{\epsilon}_k^{\text{env}}$, aggregated with the EnvInfer weights to instantiate the guided noise prediction of Eq. 6: $\hat{\boldsymbol{\epsilon}} = (1 + \omega) \sum_{k=1}^{K} w_k \, \boldsymbol{\epsilon}_k^{\text{env}} - \omega \, \boldsymbol{\epsilon}^{\text{base}}$. This estimate is supplied to the DPM-Solver update. Int-TSG and CF-TSG share this backbone and differ in how the environment weights are obtained and how the context is set (Algorithms 2 and 3).

- **Interventional generation.** Given a context $\mathbf{c}$, **CaTSG** targets $P(X \mid \mathrm{do}(C = \mathbf{c}))$. At each reverse step, EnvInfer provides $q_\phi(\mathbf{e}_k \mid \mathbf{x}_t, \mathbf{c})$ as a learned proxy for the target responsibility $r_{\mathbf{e}_k}(\mathbf{x}_t, \mathbf{c})$, which weights the interventional score mixture.
- **Counterfactual generation.** Given a factual sample $(\mathbf{x}_0, \mathbf{c})$ and a counterfactual context $\mathbf{c}'$, **CaTSG** abducts the environment weights once from $q_\phi(\mathbf{e}_k \mid \mathbf{x}_0, \mathbf{c})$, replaces $\mathbf{c}$ by $\mathbf{c}'$, and resamples residual generative uncertainty under the inferred environment.

# 5 EXPERIMENTS

In this section, we empirically validate **CaTSG** by addressing the following Research Questions (**RQ**s): **RQ1.** Does incorporating environment-level interventions improve generation fidelity under environment shifts? **RQ2.** Can **CaTSG** generate counterfactual time series that are distributionally faithful? **RQ3.** Do the learned environment representations exhibit structure across environment shifts? **RQ4.** How does each component of **CaTSG** contribute to overall performance?

## 5.1 EXPERIMENTAL SETTINGS

**Datasets & Baselines.** We conduct experiments on two synthetic datasets and two real-world datasets. The synthetic datasets, i.e., **Harmonic-VM** and **Harmonic-VP**, are generated from a simulated damped harmonic oscillator governed by second-order differential equations (see Appendix E.1), which provide ground-truth counterfactuals for direct evaluation. For real-world datasets, we use **Air Quality** (Chen, 2019) and **Traffic** (Hogue, 2019) datasets, which do not contain ground-truth counterfactuals (Appendix E.2). As baselines, we consider five representative conditional time series generative models: TimeGAN (Yoon et al., 2019), WaveGAN (Donahue et al., 2019), Pulse2Pulse (Thambawita et al., 2021), and TimeWeaver (Narasimhan et al., 2024) in its two forms, TimeWeaver-CSDI and TimeWeaver-SSSD. Further descriptions are provided in Appendix F.1.

**Evaluation Metrics.** Following previous TSG works, we report marginal distribution distance (MDD), Kullback–Leibler divergence (KL) and maximum mean discrepancy (MMD), together with the Joint Frechet Time Series Distance (J-FTSD) (Narasimhan et al., 2024), which measures the joint distribution of $c$ and $x$ and is designed for conditional generation. Details are given in Appendix F.2.

**Implementation Details.** We implement **CaTSG** in PyTorch 1.13.0 with CUDA 11.7. For denoiser, we adopt a conventional U-Net with an encoder-bottleneck-decoder structure and skip connections. For swapped prediction, we set $\tau = 0.1$. For sampling, we use the second-order single-step DPM-Solver (Lu et al., 2022) with 20 steps. Parameter counts, memory, training and sampling throughput against all baselines, and the step-count trade-off that motivates the 20-step setting are reported in Appendix G.3. Final per-dataset settings are reported in Appendix F.3.

## 5.2 EFFECT OF ENVIRONMENT-AWARE GENERATION

To address **RQ1**, we create "environments"-based splits for all datasets using selected variables as proxies for latent contexts. Baselines model observational TSG, i.e., generating samples from $P(X \mid C)$. In contrast, **CaTSG** incorporates latent environments and targets the interventional distribution $P(X \mid \mathrm{do}(C))$. Note that this comparison targets the environment-shift regime rather than observational fit under near-iid conditions, and we do not claim a necessary advantage over standard Obs-TSG models when $E$ is stable and observed (Appendix H.5). Table 2 reports results across synthetic (Harmonic-VM/VP) and real-world (Air Quality, Traffic) datasets. All Table 2 entries report the mean and standard deviation over five independent training runs per method. Confidence interval analysis is presented in Appendix G.4. According to the table, across four datasets and four metrics (16 comparisons), **CaTSG** gains the best point estimate in 14/16 cases, 12 of them with non-overlapping 95% intervals. Across the 16 dataset–metric cells of Table 2, we average cell-wise

Table 2: Results on synthetic and real-world datasets. TW-CSDI, TW-SSSD: TimeWeaver variants based on CSDI and SSSD.

| Dataset | Metric | TimeGAN | WaveGAN | Pulse2Pulse | TW-CSDI | TW-SSSD | CaTSG (ours) |
|---|---|---|---|---|---|---|---|
| **Harmonic-VM** | MDD | 0.525 ± 0.087 | 0.492 ± 0.001 | 0.339 ± 0.092 | 0.267 ± 0.001 | 0.297 ± 0.001 | **0.098** ± 0.001 |
| | KL | 11.926 ± 0.256 | 4.518 ± 0.076 | 1.717 ± 0.146 | 0.208 ± 0.005 | 0.230 ± 0.003 | **0.017** ± 0.000 |
| | MMD | 0.554 ± 0.458 | 0.234 ± 0.000 | 1.357 ± 1.924 | 0.080 ± 0.000 | 0.160 ± 0.002 | **0.012** ± 0.000 |
| | J-FTSD | 10.713 ± 0.042 | 7.079 ± 0.973 | 4.885 ± 0.773 | 2.782 ± 0.246 | 3.672 ± 0.152 | **0.548** ± 0.085 |
| **Harmonic-VP** | MDD | 0.332 ± 0.011 | 0.308 ± 0.000 | 0.288 ± 0.029 | 0.077 ± 0.000 | 0.083 ± 0.000 | **0.053** ± 0.001 |
| | KL | 17.337 ± 0.675 | 10.043 ± 0.002 | 12.018 ± 4.605 | 0.208 ± 0.006 | 0.275 ± 0.002 | **0.066** ± 0.003 |
| | MMD | 0.516 ± 0.035 | 0.711 ± 0.000 | 0.474 ± 0.078 | 0.106 ± 0.001 | 0.084 ± 0.000 | **0.034** ± 0.001 |
| | J-FTSD | 28.933 ± 0.912 | 38.522 ± 6.362 | 28.163 ± 10.497 | 13.114 ± 2.464 | 16.142 ± 1.428 | **12.677** ± 2.431 |
| **Air Quality** | MDD | 0.199 ± 0.037 | 0.254 ± 0.000 | 0.081 ± 0.000 | 0.100 ± 0.000 | 0.084 ± 0.000 | **0.069** ± 0.000 |
| | KL | 14.990 ± 3.219 | 16.513 ± 0.000 | 1.938 ± 0.001 | 0.502 ± 0.004 | 0.357 ± 0.005 | **0.301** ± 0.001 |
| | MMD | 0.224 ± 0.113 | 1.005 ± 0.000 | 0.116 ± 0.000 | 0.226 ± 0.000 | **0.059** ± 0.001 | 0.070 ± 0.000 |
| | J-FTSD | 12.749 ± 1.132 | 10.302 ± 0.628 | 16.555 ± 4.784 | 3.757 ± 0.634 | 17.191 ± 1.251 | **2.929** ± 0.497 |
| **Traffic** | MDD | 0.443 ± 0.017 | 0.375 ± 0.000 | 0.231 ± 0.000 | 0.094 ± 0.000 | 0.091 ± 0.000 | **0.071** ± 0.000 |
| | KL | 19.339 ± 0.110 | 12.101 ± 0.175 | 4.228 ± 0.001 | **0.155** ± 0.001 | 0.157 ± 0.001 | 0.172 ± 0.045 |
| | MMD | 0.404 ± 0.002 | 0.547 ± 0.000 | 0.153 ± 0.000 | 0.033 ± 0.000 | 0.038 ± 0.000 | **0.006** ± 0.000 |
| | J-FTSD | 10.516 ± 0.326 | 23.649 ± 4.027 | 21.076 ± 0.776 | 6.564 ± 0.525 | 7.880 ± 0.726 | **0.723** ± 0.055 |

relative error reductions against the best baseline in each named family. This gives 61.8% for the best TimeWeaver variant versus the best GAN baseline, and 78.0% and 43.8% for **CaTSG** versus the best GAN and TimeWeaver baselines, respectively. In these environment-shift comparisons, **CaTSG** further improves fidelity to observational samples in the test split. The gains are most pronounced on the synthetic datasets, where shift patterns and counterfactual ground truth are controlled. As a controlled validation on Harmonic-VM, we directly construct the interventional target $P(X \mid \mathrm{do}(C))$ from the structural equations rather than estimating it from held-out data. Appendix G.1 evaluates all methods against this target. There **CaTSG** uses the unguided $\omega = 0$ of Eq. 4, versus $\omega = 0.2$ for its Table 2 samples, and leads all 16 metric–environment-marginal combinations.

### 5.3 Counterfactual Generation Ability

For **RQ2**, since ground-truth counterfactuals are available only for synthetic data, we report quantitative results for CF-TSG on two Harmonic datasets (Table 3). For real-world datasets, we present visualizations (Figure 4). Since real-world data carry no counterfactual ground truth, Appendix G.5 instead reports quantitative distributional diagnostics for real-world counterfactual generation. The five conditional TSG baselines of Table 2 do not natively support our factual-pair-conditioned $C \to C'$ task, so we compare **CaTSG** against the three ablations that keep its factual-abduction pathway (settings in Table 7). As shown in Table 3, **CaTSG** consistently achieves the best distributional performance on both datasets in MDD and KL, supporting the contribution of learned environment weights to counterfactual generation.

Table 3: Counterfactual generation results.

| Method | Harmonic-VM | | Harmonic-VP | |
|---|---|---|---|---|
| | MDD | KL | MDD | KL |
| **CaTSG** | **0.374** $\pm$ 0.002 | **0.210** $\pm$ 0.004 | **0.256** $\pm$ 0.001 | **0.660** $\pm$ 0.100 |
| **RandEnv** | <u>0.469</u> $\pm$ 0.002 | <u>0.691</u> $\pm$ 0.014 | 0.272 $\pm$ 0.001 | 2.243 $\pm$ 0.045 |
| **w/o SW** | 0.474 $\pm$ 0.004 | 0.816 $\pm$ 0.027 | 0.275 $\pm$ 0.001 | <u>2.144</u> $\pm$ 0.066 |
| **FrozenEnv** | 0.462 $\pm$ 0.001 | 0.760 $\pm$ 0.016 | <u>0.271</u> $\pm$ 0.002 | 2.490 $\pm$ 0.056 |

For qualitative analysis on real-world data, we provide visualization results (Figure 4). On Traffic dataset, we select test samples whose factual temperature exceeds 22°C (example $\mathbf{x}$ in Figure 4a). Note that temperature is *not* included in $\mathbf{c}$ but defines the environment split and selects the factual samples above (see Table 6). We first obtain learned factual mixture weights $\mathbf{w}$ from $(\mathbf{x}, \mathbf{c})$ and keep them fixed. We then *action* on $\mathbf{c}$ to obtain $\mathbf{c}'$ by setting the snow indicator to 1 to serve as a counterfactual assignment incompatible with the factual temperature (i.e., $> 22°\mathrm{C}$). Next, we *predict* (i.e., generate) the resulting traffic volumes $\hat{\mathbf{x}}'$ (examples in Figure 4b). The generations preserve the temporal structure of the factual ones while exhibiting a reduced amplitude. To summarize the generated outcomes across samples, Figure 4c compares the global distributions of $\mathbf{x}$ and $\hat{\mathbf{x}}'$ using histograms and empirical cumulative distribution functions (ECDFs). Relative to $\mathbf{x}$, the ECDF of $\hat{\mathbf{x}}'$ lies to the left and is steeper around the median. Together with the narrower histogram, this pattern indicates a lower location and reduced dispersion (smaller amplitude) in the counterfactual "snow in summer" setting.

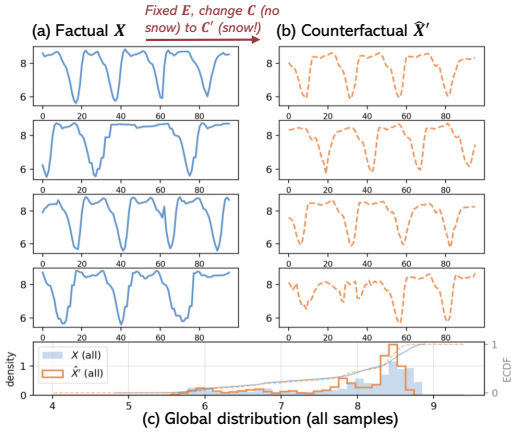



Figure 4: (a) Factual samples $X$ and (b) counterfactual generations $\hat{X}'$ on Traffic. (c) Global distribution over all samples: histogram and ECDF of $X$ (blue) and $\hat{X}'$ (orange).

### 5.4 Interpretation Analysis

To answer **RQ3**, we present environment interpretation analysis. Note that $\mathbf{E}$ is latent, and its size $K$ is a dataset-specific hyperparameter (not tied to the number of splits or any variables), enabling **CaTSG** to capture richer context structure and more complex scenarios. All main experiments use $K = 4$; the following visualization-only analysis uses $K = 8$. We visualize the learned environments and mixture weights for Harmonic-VM with $K = 8$. In Figure 5a, bars show the mean soft mixture weight per environment, and grey markers denote hard-assignment frequency. Their close agreement indicates consistent assignment patterns, and the moderate shifts across splits reflect the intended split-specific context. In Figure 5b, t-SNE reveals compact clusters around the environment embeddings

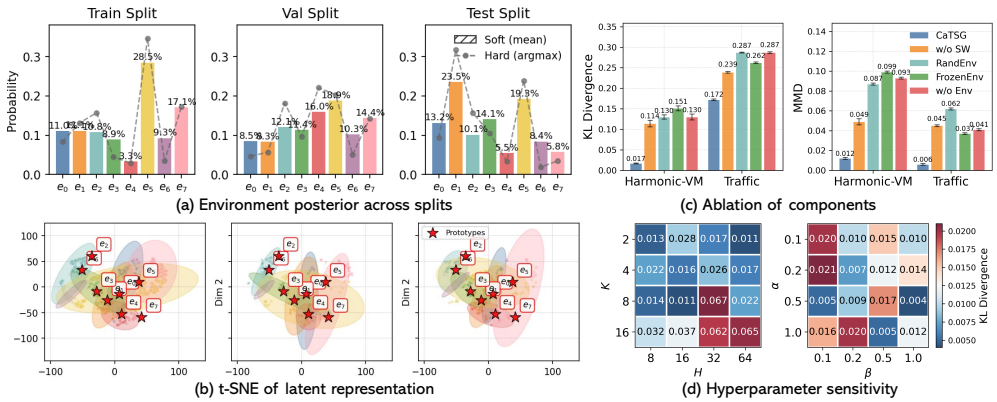



Figure 5: (a) EnvInfer mixture weights $\mathbf{w}$ across splits. (b) t-SNE of representations on Harmonic-VM. Shaded ellipses are one standard-deviation region. (c) Ablation of components. (d) Hyperparameter sensitivity on Harmonic-VM.

(denoted as stars). The limited overlap and cross-split shift indicate structured learned representations that vary across splits.

### 5.5 Ablation Study

For **RQ4**, we conduct a series of ablation studies on core components and hyperparameter sensitivity.

Table 4: #Steps v.s. performance (Harmonic-VM).

| Steps | Runtime (s) | MDD | KL |
|---|---|---|---|
| **5** | 5.49 $\pm$ 0.45 | 0.231 $\pm$ 0.002 | 0.244 $\pm$ 0.006 |
| **10** | 9.96 $\pm$ 1.39 | 0.122 $\pm$ 0.002 | 0.035 $\pm$ 0.001 |
| **20** | 19.30 $\pm$ 2.85 | 0.098 $\pm$ 0.001 | 0.017 $\pm$ 0.000 |
| **50** | 54.63 $\pm$ 2.22 | 0.088 $\pm$ 0.001 | 0.014 $\pm$ 0.000 |
| **100** | 107.01 $\pm$ 4.09 | 0.087 $\pm$ 0.001 | 0.013 $\pm$ 0.000 |

**Effects of Core Components.** To assess the effects of these variants and training choices, we compare RandEnv, w/o SW, FrozenEnv, and w/o Env, whose settings are given in Table 7. Removing the environment bank reduces the model to a standard conditional diffusion model generating $P(X \mid C)$. Note that w/o Env therefore also serves as a same-backbone conditional-diffusion reference, since it keeps the denoiser and the sampler and differs only in the environment component. Figure 5c reports these ablations on Harmonic-VM and Traffic.

**Hyperparameter Sensitivity.** Figure 5d shows the impact of number of environment embeddings $K$, hidden dimension $H$, and loss coefficients $\alpha$ and $\beta$ on KL divergence on Harmonic-VM. Overall, the KL varies only modestly, indicating that **CaTSG** is broadly robust to hyperparameter choices. Two mild trends emerge: (1) small–to–moderate $K \in \{2, 4, 8\}$ with compact hidden size $H \in \{8, 16\}$ give the lowest KL and (2) for loss weights, a shallow "good" basin appears for moderate/large values ($\alpha, \beta \in [0.5, 1.0]$). The dependence on $K$ and $H$ is not monotonic, so we select both per dataset by validation rather than by a fixed rule. Appendix F.3 gives the heuristic we follow when more latent confounders are expected.

**Speed & Quality Trade-off.** We ablate the number of DPM-Solver steps on Harmonic-VM (Table 4). Increasing the step count consistently improves MDD and KL, while runtime scales approximately linearly. A 20-step sampler offers a clear Pareto point: it is $5.5\times$ faster than 100 steps yet remains within $0.011$ MDD and $0.004$ KL of the best scores. We therefore adopt 20 steps as the default, and report the full results in Table 11 (Appendix G.3).

## 6 Conclusion and Future Work

In this work, we present a causal view on conditional TSG by organizing observational, interventional, and counterfactual tasks within Pearl's causal hierarchy. Building on a structural causal model, we develop **CaTSG**, a unified diffusion-based framework that supports observational, interventional, and counterfactual generation via causal score functions under stated assumptions. **CaTSG** operates under a fixed sequence-level SCM and approximates latent confounding via a finite environment bank; relaxing these assumptions, supporting multi-modal conditions, and developing systematic real-world counterfactual evaluation are natural next steps. These challenges also mark rich opportunities for future research toward more flexible and generalizable causal generative models and evaluation frameworks.

### AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on experimental design, interpretation of results and consistency checking. We did not use generative AI to generate experimental data or write mathematical proofs. Additionally, we used generative AI for language editing and assistance with code. We reviewed all AI-assisted work: numerical claims were verified against the underlying experimental outputs, and AI-assisted code and mathematical content were checked by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

We release the **CaTSG** implementation, the synthetic data generators, data preprocessing, evaluation metrics, and configurations. The model and generation procedures are given in Section 4 and Algorithms 1–3. Appendices E.1 and E.2 describe the datasets, and Appendices F.2 and F.3 give the evaluation metrics, hyperparameters, compute resources, and random seeds.

## REFERENCES

Juan Lopez Alcaraz and Nils Strodthoff. Diffusion-based time series imputation and forecasting with structured state space models. *Transactions on Machine Learning Research*, 2023.

Yihao Ang, Qiang Huang, Yifan Bao, Anthony KH Tung, and Zhiyong Huang. Tsgbench: Time series generation benchmark. *Proceedings of the VLDB Endowment*, 17(3):305–318, 2023.

Charles K Assaad, Emilie Devijver, and Eric Gaussier. Survey and evaluation of causal discovery methods for time series. *Journal of Artificial Intelligence Research*, 73:767–819, 2022.

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

Yifan Bao, Yihao Ang, Qiang Huang, Anthony KH Tung, and Zhiyong Huang. Towards controllable time series generation. *arXiv preprint arXiv:2403.03698*, 2024.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

Haiyang Chen. Challenges and corresponding solutions of generative adversarial networks (gans): a survey study. In *Journal of Physics: Conference Series*, volume 1827, pp. 012066. IOP Publishing, 2021.

Song Chen. Beijing multi-site air-quality data, 2019. URL `https://doi.org/10.24432/C5RK5G`.

Andrea Coletta, Sriram Gopalakrishnan, Daniel Borrajo, and Svitlana Vyetrenko. On the constrained time-series generation problem. *Advances in Neural Information Processing Systems*, 36:61048–61059, 2023.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. In *International Conference on Learning Representations*, 2019.

Michael Eichler and Vanessa Didelez. On granger causality and the effect of interventions in time series. *Lifetime data analysis*, 16(1):3–32, 2010.

Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pp. 424–438, 1969.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

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

John Hogue. Metro interstate traffic volume data set, 2019. URL https://doi.org/10.24432/C5X60B.

Zhiting Hu and Li Erran Li. A causal lens for controllable text generation. *Advances in Neural Information Processing Systems*, 34:24941–24955, 2021.

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, pp. 17520–17527, 2025.

Murat Kocaoglu, Christopher Snyder, Alexandros G Dimakis, and Sriram Vishwanath. Causalgan: Learning causal implicit generative models with adversarial training. In *International Conference on Learning Representations*, 2018.

Aneesh Komanduri, Xintao Wu, Yongkai Wu, and Feng Chen. From identifiable causal representations to controllable counterfactual generation: A survey on causal generative modeling. *arXiv preprint arXiv:2310.11011*, 2023.

Aneesh Komanduri, Chen Zhao, Feng Chen, and Xintao Wu. Causal diffusion autoencoders: Toward counterfactual generation via diffusion probabilistic models. *arXiv preprint arXiv:2404.17735*, 2024.

Robert L. Lehrman. *Physics the Easy Way*. Barron's Educational Series, Hauppauge, NY, 3rd edition, 1998. OCLC: 1036803976.

Hao Li, Yu-Hao Huang, Chang Xu, Viktor Schlegel, Ren-He Jiang, Riza Batista-Navarro, Goran Nenadic, and Jiang Bian. Bridge: Bootstrapping text to control time-series generation via multi-agent iterative optimization and diffusion modelling. *ICML*, 2025.

Lequan Lin, Zhengkun Li, Ruikun Li, Xuliang Li, and Junbin Gao. Diffusion models for time-series applications: a survey. *Frontiers of Information Technology & Electronic Engineering*, 25(1): 19–41, 2024.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

Ellicott C Matthay and M Maria Glymour. Causal inference challenges and new directions for epidemiologic research on the health effects of social policies. *Current Epidemiology Reports*, 9 (1):22–37, 2022.

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *ICML*, pp. 1791–1799, 2014.

Sai Shankar Narasimhan, Shubhankar Agarwal, Oguzhan Akcin, Sujay Sanghavi, and Sandeep P Chinchali. Time weaver: A conditional time series generation model. In *ICML*, pp. 37293–37320, 2024.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000. ISBN 9780521773621.

Judea Pearl. *Causality*. Causality: Models, Reasoning, and Inference. Cambridge University Press, 2009. ISBN 9780521895606. URL https://books.google.com/books?id=f4nuexsNVZIC.

Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Causal inference on time series using restricted structural equation models. *Advances in neural information processing systems*, 26, 2013.

Giorgia Ramponi, Pavlos Protopapas, Marco Brambilla, and Ryan Janssen. T-cgan: Conditional generative adversarial network for data augmentation in noisy time series with irregular sampling. *arXiv preprint arXiv:1811.08295*, 2018.

Axel Sauer and Andreas Geiger. Counterfactual generative networks. In *International Conference on Learning Representations*, 2021.

Kaleb E Smith and Anthony O Smith. Conditional gan for timeseries generation. *arXiv preprint arXiv:2006.16477*, 2020.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, pp. 2256–2265, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in neural information processing systems*, 34:24804–24816, 2021.

Vajira Thambawita, Jonas L Isaksen, Steven A Hicks, Jonas Ghouse, Gustav Ahlberg, Allan Linneberg, Niels Grarup, Christina Ellervik, Morten Salling Olesen, Torben Hansen, et al. Deepfake electrocardiograms using generative adversarial networks are the beginning of the end for privacy issues in medicine. *Scientific reports*, 11(1):21896, 2021.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Haoyu Wang, Le Wang, Sanping Zhou, Jingyi Tian, Zheng Qin, Yabing Wang, Gang Hua, and Wei Tang. Towards precise embodied dialogue localization via causality guided diffusion. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 13350–13360, 2025.

Haomin Wen, Youfang Lin, Yutong Xia, Huaiyu Wan, Qingsong Wen, Roger Zimmermann, and Yuxuan Liang. Diffstg: Probabilistic spatio-temporal graph forecasting with denoising diffusion models. In *ACM SIGSPATIAL*, pp. 1–12, 2023.

Shenghao Wu, Wenbin Zhou, Minshuo Chen, and Shixiang Zhu. Counterfactual generative models for time-varying treatments. In *ACM SIGKDD*, pp. 3402–3413, 2024.

Kevin Xia, Kai-Zhan Lee, Yoshua Bengio, and Elias Bareinboim. The causal-neural connection: Expressiveness, learnability, and inference. *Advances in Neural Information Processing Systems*, 34:10823–10836, 2021.

Kevin Muyuan Xia, Yushu Pan, and Elias Bareinboim. Neural causal models for counterfactual identification and estimation. In *The Eleventh International Conference on Learning Representations*, 2023a.

Tian Xia, Fabio De Sousa Ribeiro, Rajat R Rasal, Avinash Kori, Raghav Mehta, and Ben Glocker. Decoupled classifier-free guidance for counterfactual diffusion models. *arXiv preprint arXiv:2506.14399*, 2025a.

Yutong Xia, Yuxuan Liang, Haomin Wen, Xu Liu, Kun Wang, Zhengyang Zhou, and Roger Zimmermann. Deciphering spatio-temporal graph forecasting: A causal lens and treatment. *Advances in Neural Information Processing Systems*, 36:37068–37088, 2023b.

Yutong Xia, Ao Qu, Yunhan Zheng, Yihong Tang, Dingyi Zhuang, Yuxuan Liang, Shenhao Wang, Cathy Wu, Lijun Sun, Roger Zimmermann, et al. Reimagining urban science: Scaling causal inference with large language models. *arXiv preprint arXiv:2504.12345*, 2025b.

Yutong Xia, Yingying Zhang, Yuxuan Liang, Lunting Fan, Qingsong Wen, and Roger Zimmermann. Capulse: Detecting anomalies by tuning in to the causal rhythms of time series. *arXiv preprint arXiv:2508.04630*, 2025c.

Mengyue Yang, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jun Wang. Causalvae: Disentangled representation learning via neural structural causal models. In *CVPR*, pp. 9593–9602, 2021.

Yiyuan Yang, Ming Jin, Haomin Wen, Chaoli Zhang, Yuxuan Liang, Lintao Ma, Yi Wang, Chenghao Liu, Bin Yang, Zenglin Xu, et al. A survey on diffusion models for time series and spatio-temporal data. *arXiv preprint arXiv:2404.18886*, 2024.

Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *NeurIPS*, 32, 2019.

Jinsung Yoon, Lydia N Drumright, and Mihaela Van Der Schaar. Anonymization through data synthesis using generative adversarial networks (ads-gan). *IEEE journal of biomedical and health informatics*, 24(8):2378–2388, 2020.

Xinyu Yuan and Yan Qiao. Diffusion-ts: Interpretable diffusion for general time series generation. In *ICLR*, 2024.

# A NOTATION

Table 5: **Notation summary**. Random variables are uppercase math italic (e.g., $\mathcal{X}$), vectors are lowercase bold (e.g., $\mathbf{x}$), matrices are uppercase bold (e.g., $\mathbf{X}$), and scalars are non-bold (e.g., $x$).

| Symbol | Description |
|---|---|
| $X, C, E$ | Variables: target time series, contextual variable, latent environment |
| $\mathbf{x}, \mathbf{c}, \mathbf{e}$ | Sample of time series, context, or environment |
| $\mathrm{do}(\cdot)$ | do-calculus, an intervention operation that sets a variable independent of its causes |
| $\boldsymbol{\epsilon}$ | Noise vector sampled from Gaussian $\mathcal{N}(0, I)$ |
| $\hat{\boldsymbol{\epsilon}}$ | Predicted noise (from denoising model) |
| $p_\theta(\cdot), \mathcal{F}_\theta(\cdot)$ | Learnable generative model (e.g., diffusion model) |
| $q_\phi(\cdot), \varepsilon_\theta(\cdot)$ | EnvInfer and Denoiser module |
| $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_K\}$ | A learnable latent environment bank of size $K$ |
| $\mathbf{w} = \{w_1, \ldots, w_K\}$ | Predicted environment assignment probability. |
| $s_t(\cdot)$ | Score function |
| $\mathcal{L}_{\text{eps}}$, $\mathcal{L}_{\text{sw}}$, $\mathcal{L}_{\text{orth}}$ | Noise prediction loss, swapped prediction loss and orthogonal loss |
| $\mathbf{s}$, $\hat{\mathbf{w}}$ | Predicted logits and balanced soft assignments by applying Sinkhorn–Knopp to the batch of logits |
| $V$ | Number of augmentations for each samples for swapped prediction loss $\mathcal{L}_{\text{sw}}$ |
| $\alpha$, $\beta$ | Balance coefficients for $\mathcal{L}_{\text{sw}}$ and $\mathcal{L}_{\text{orth}}$ |
| $\sigma_t^2$ | Noise variance at diffusion step $t$ |
| $H$, $N$ | Hidden dimension, the number of samples |
| $t \in \{1, \ldots, T\}$ | Diffusion timestep |
| $\omega$ | Weight for controlling conditional-vs-unconditional trade-off |

# B MORE DETAILS ON PRELIMINARIES

Due to space constraints in the main text, we provide additional preliminaries in this appendix. Specifically, we elaborate on two key components: **(B.1) causal concepts**, which is related to the motivation and theoretical foundations of our proposed task, and **(B.2) diffusion models**, which is related to the technical background of the backbone architecture of our proposed model. These preliminaries are covered in this section, while the full derivation of the proposed CaTSG model is deferred to Appendix C.2.

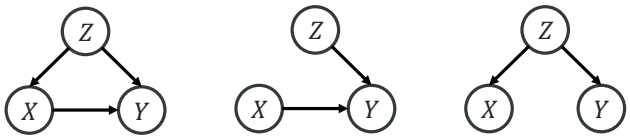
## B.1 CAUSAL CONCEPTS

We now present the causal foundations relevant to our work. We begin by introducing Structural Causal Models (SCMs) (Pearl, 2000; 2009), followed by a brief review of the rules of do-calculus (Pearl et al., 2016), which enable formal reasoning under interventions. We then derive the backdoor adjustment formula (Figure 2c) and introduce the three-step process of counterfactual generation (Figure 2d).

**Structural Causal Model.** A Structural Causal Model (SCM) (Pearl, 2009) is defined as a directed graph where nodes represent variables and directed edges denote direct causal dependencies between them. This graph-based representation allows us to reason about how interventions propagate through a system. Example SCMs are shown in Figure 6.

**Three rules of Do-calculus.** The *do-calculus* (Pearl et al., 2016) is a formal system that enables the identification of interventional distributions, such as $P(Y \mid \mathrm{do}(X))$, from purely observational data, given a SCM. Its core goal is to transform expressions involving interventions (i.e., $\mathrm{do}(\cdot)$) into expressions involving only observed variables. Do-calculus consists of three transformation rules that operate on probabilistic expressions involving the *do*-operator, leveraging the conditional independencies encoded in the causal graph.

Before introducing the rules, we first define the relevant graph transformations used to represent interventions and their structural consequences as shown in Figure 6. Let $\mathcal{G}$ be a directed acyclic graph (DAG) with nodes $X$, $Y$, and $Z$ (confounder) (Figure 6a). The *interventional graph* $\mathcal{G}_{\mathrm{do}(X)}$ is




(a) $\mathcal{G}$ (b) $\mathcal{G}_{\text{do}(X)}$ (c) $\mathcal{G}_{\text{null}(X)}$

Figure 6: Illustration of (a) original DAG $\mathcal{G}$, (b) interventional graph $\mathcal{G}_{\text{do}(X)}$, and (c) nullified graph $\mathcal{G}_{\text{null}(X)}$

defined by removing all incoming edges to $X$ (Figure 6b), while the *nullified graph* $\mathcal{G}_{\text{null}(X)}$ removes all outgoing edges from $X$ (Figure 6c).

Based on these graph structures, the following three rules govern valid manipulation of probabilities under interventions (Pearl et al., 2016):

- **Rule 1 (Insertion/deletion of observations):**

$$P(y \mid do(x), z) = P(y \mid do(x)) \quad \text{if } (y \perp\!\!\!\perp z \mid x)_{G_{do(x)}}$$

- **Rule 2 (Action/observation exchange):**

$$P(y \mid do(x), do(z)) = P(y \mid do(x), z) \quad \text{if } (y \perp\!\!\!\perp z \mid x)_{G_{do(x),null(z)}}$$

- **Rule 3 (Insertion/deletion of actions):**

$$P(y \mid do(x), do(z)) = P(y \mid do(x)) \quad \text{if } (y \perp\!\!\!\perp z \mid x)_{G_{do(x),do(z)}}$$

Here, $(y \perp\!\!\!\perp z \mid x)_{\mathcal{G}}$ denotes conditional independence of $y$ and $z$ given $x$ in the graph $\mathcal{G}$. For instance, Rule 1 states that if $x$ d-separates $y$ from $z$ in $\mathcal{G}_{\text{do}(x)}$, then $z$ can be removed from the conditional set.

**Backdoor Adjustment.** We now apply these rules to derive the backdoor adjustment in Figure 2c, where we conduct the $do$-operation on variable $C$ and adjust for the confounder $E$, which influences both $C$ and $X$. Using $do$-calculus, we derive:

$$\begin{aligned} P(X \mid \text{do}(C)) &= \sum_e P(X \mid \text{do}(C), E = e)\, P(E = e \mid \text{do}(C)) \quad \textit{(Law of total probability)} \\ &= \sum_e P(X \mid \text{do}(C), E = e)\, P(E = e) \quad \textit{(Rule 3)} \\ &= \sum_e P(X \mid C, E = e)\, P(E = e) \quad \textit{(Rule 2)} \end{aligned} \tag{9}$$

**Three-Step Counterfactuals Computing.** Counterfactual reasoning aims to answer questions of the form: *"What would have happened, had a certain condition been different?"* In the language of SCMs, a counterfactual query typically takes the form $P(Y_{x'} \mid X = x, Y = y)$, which quantifies the outcome $Y$ under a hypothetical intervention $do(X = x')$, given that we observed $X = x$ and $Y = y$ in the real world.

To compute such queries, taking our case as an example (Figure 2d), SCMs follow a principled three-step procedure (Pearl, 2000):

- **Abduction**: Infer the latent background variables $E$ from the observed variables, i.e., estimate $P(E \mid X, C)$.
- **Action**: Modify the model by applying an intervention $\text{do}(C = C')$, which removes all incoming edges to $C$ and sets it to a new value $C'$.
- **Prediction**: Use the modified model and the inferred $E$ to compute the counterfactual outcome, i.e., evaluate $P(X' \mid E, C')$.

The resulting counterfactual distribution is given by:

$$P(X' \mid X, C, C') = \sum_e P(X' \mid \text{do}(C'), E = e)\, P(E = e \mid X, C) \tag{10}$$

### B.2 Diffusion Models and Conditional Diffusion Models

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) are a class of probabilistic generative models that learn to generate data through a two-phase procedure: *a forward diffusion process* that progressively adds noise to the data, followed by *a reverse denoising process* that learns to recover the original data distribution. In recent years, diffusion models have become a powerful paradigm for time series modeling, achieving strong performance across a wide range of tasks (Tashiro et al., 2021; Wen et al., 2023; Yuan & Qiao, 2024; Yang et al., 2024; Lin et al., 2024). We begin by introducing the unconditional setting, detailing both the forward and reverse processes. We then introduce the extension of this framework to the conditional setting, where external factors are incorporated to guide the generation process.

#### B.2.1 Diffusion Models

To simplify the presentation, we begin by considering the standard generation setting, without restricting ourselves to time series inputs.

**Forward Process.** Diffusion probabilistic models (Sohl-Dickstein et al., 2015) define a generative process by reversing a Markovian forward noising process. Let $x_0 \in \mathbb{R}^d$ denote a clean data sample drawn from an unknown data distribution $q(x_0)$. The forward process gradually adds Gaussian noise over $T$ discrete time steps to destroy the data:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \quad q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}), \tag{11}$$

where $\beta_t \in (0,1)$ is a pre-defined variance schedule. Thanks to the Gaussian property, we can marginalize the entire forward process in closed form:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I}), \tag{12}$$

where $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$.

**Reverse Process.** The generative model learns to reverse this process by estimating the posterior $p_\theta(x_{t-1}|x_t)$ with a neural network. The model is defined as a chain of conditional distributions:

$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \tag{13}$$

with Gaussian transitions:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)). \tag{14}$$

**Training Objective.** To enable efficient training, we first leverage the closed-form expression of the forward process in Eq 12, which allows us to directly sample a noisy input $x_t$ at any timestep $t$ from the clean data $x_0$ via:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}). \tag{15}$$

Instead of directly maximizing the log-likelihood $\log p_\theta(x_0)$, the model is trained by optimizing a variational lower bound (ELBO). Ho et al. (2020) show that this is equivalent to minimizing a simplified denoising objective that predicts the added noise $\epsilon$:

$$\mathcal{L} = \mathbb{E}_{x_0,\epsilon,t}\left[\|\epsilon - \varepsilon_\theta(x_t, t)\|_2^2\right]. \tag{16}$$

This objective guides the denoising network $\varepsilon_\theta$ to recover the noise component from the perturbed observation $x_t$, thereby implicitly learning the score function of the data distribution.

**Sampling Procedure.** During inference, the model generates samples by iteratively denoising:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t, t)\right) + \tilde{\sigma}_t z, \quad z \sim \mathcal{N}(0, \mathbf{I}), \tag{17}$$

where $x_{t-1}$ refers to the noisy sample at time step $t-1$, $\varepsilon_\theta(x_t, t)$ denotes the predicted noise by the denoising model $\varepsilon_\theta$, $\alpha_t$ and $\bar{\alpha}_t$ denote the noise schedule parameters, $\tilde{\sigma}_t^2 = \beta_t$ or some learnable variance, $z$ denotes the standard Gaussian noise.

**Score Function Interpretation.** Diffusion models can also be interpreted as a special class of *score-based generative models* (Song & Ermon, 2019), which aim to learn the gradient of the log-density of the data at each timestep. This gradient, known as the *score function*, indicates the direction in which the noisy input $x_t$ should be adjusted to increase its likelihood under the marginal distribution $p_t(x_t)$. The score function is formally defined as:

$$s_t(x_t) := \nabla_{x_t} \log p(x_t). \tag{18}$$

Ho et al. (Ho et al., 2020) show that, under the Gaussian forward process (Eq. 15), learning to approximate this score is equivalent to learning to predict the noise $\epsilon$ added during forward diffusion. This leads to the following approximate relationship:

$$\nabla_{x_t} \log p(x_t) \approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \cdot \varepsilon_\theta(x_t, t). \tag{19}$$

This result provides a principled link between score-based modeling and the denoising objective in DDPMs: the trained network $\varepsilon_\theta$ can be interpreted as predicting the (scaled) score function at each timestep.

### B.2.2 Conditional Diffusion Models

**Conditional Setup.** To enable guided generation under external control or domain-specific information (e.g., class labels, time-of-day, etc.), diffusion models can be extended to the conditional setting. Here we show the case where the generative process is conditioned on an auxiliary variable $c$, and both the reverse transition and the denoising network are modified accordingly. Specifically, the reverse process in Eq. 14 becomes:

$$p_\theta(x_{t-1} \mid x_t, c) = \mathcal{N}(\mu_\theta(x_t, t, c), \Sigma_\theta(x_t, t, c)), \tag{20}$$

where the network $\varepsilon_\theta(x_t, t, c)$ now takes $c$ as an additional input to predict the noise added during the forward process. This conditional formulation allows the model to generate samples consistent with the specified condition $c$, providing a foundation for controllable or guided generation tasks. Accordingly, instead of learning the unconditional score function $s_t(x_t)$ in Eq.18, we aim to estimate the conditional score $s_t(x_t, c)$:

$$\begin{aligned} s_t(x_t, c) &:= \nabla_{x_t} \log p(x_t \mid c) && \textit{(Definition)} \\ &\approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \cdot \varepsilon_\theta(x_t, t, c) && \textit{(Similar to Eq.19)} \end{aligned} \tag{21}$$

**Classifier-Free Guidance.** One classical perspective decomposes the conditional score $\nabla_{x_t} \log p(x_t \mid c)$ in Eq. 21 via Bayes' rule:

$$\nabla_{x_t} \log p(x_t \mid c) = \nabla_{x_t} \log \left( \frac{p(x_t) p(c \mid x_t)}{p(c)} \right) = \nabla_{x_t} \log p(x_t) + \nabla_{x_t} \log p(c \mid x_t) \tag{22}$$

where the first term $\nabla_{x_t} \log p(x_t)$ corresponds to the gradient of the unconditional model (Eq 19), and the second term $\nabla_{x_t} \log p(c \mid x_t)$ introduces an additional signal from the classifier $p(c \mid x_t)$. This decomposition reveals the principle behind classifier guidance: to improve conditional alignment during generation, one may simply augment the unconditional score with a gradient term derived from a classifier.

Ho & Salimans (2022) propose *Classifier-Free Guidance* (CFG), which can be interpreted as a reparameterization of classifier guidance by manipulating the gradient expression of the conditional log-probability. Starting from Bayes' rule, we can write the second term in Eq 22 as:

$$\nabla_{x_t} \log p(c \mid x_t) = \nabla_{x_t} \log p(x_t \mid c) - \nabla_{x_t} \log p(x_t) \tag{23}$$

Plugging this back into the classifier-guidance formulation, with classifier-guidance scale $\gamma \geq 1$, gives:

$$
\begin{aligned}
s_t^{\mathrm{CFG}}(x_t, c) &= \nabla_{x_t} \log p(x_t) + \gamma \nabla_{x_t} \log p(c \mid x_t) && \textit{(Classifier guidance)} \\
&= \nabla_{x_t} \log p(x_t) + \gamma \left(\nabla_{x_t} \log p(x_t \mid c) - \nabla_{x_t} \log p(x_t)\right) && \textit{(Plugging Eq. 23)} \\
&= \gamma \nabla_{x_t} \log p(x_t \mid c) - (\gamma - 1) \nabla_{x_t} \log p(x_t) && \textit{(Algebraic rearrangement)} \\
&= \gamma \cdot s_t(x_t, c) - (\gamma - 1) \cdot s_t(x_t) && \textit{(Score-based rewriting)} \\
&\approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \left[\gamma \cdot \varepsilon_\theta(x_t, t, c) - (\gamma - 1) \cdot \varepsilon_\theta(x_t, t)\right] && \textit{(According to Eq. 19 and 21)} \\
&= -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \left[(1+\omega) \cdot \varepsilon_\theta(x_t, t, c) - \omega \cdot \varepsilon_\theta(x_t, t)\right], \quad \omega \equiv \gamma - 1 && \textit{(Reparameterization)}
\end{aligned}
$$

where $\varepsilon_\theta(x_t, t, c)$ denotes the conditional noise prediction network, $\varepsilon_\theta(x_t, t)$ the unconditional one, and $\omega \geq 0$ is the reparameterized guidance scale. Therefore, the CFG-adjusted score field can be written as follows:

$$
s_t^{\mathrm{CFG}}(x_t, c) \approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \left[(1+\omega) \cdot \varepsilon_\theta(x_t, t, c) - \omega \cdot \varepsilon_\theta(x_t, t)\right] \tag{24}
$$

At $\omega = 0$, this recovers the unguided conditional score estimate; $\omega > 0$ gives a guidance-adjusted field. In practice, CFG reformulates conditional generation as a single denoising network training task. This is achieved by randomly dropping the condition $c$ to an empty set $c \leftarrow \varnothing$ during training. This technique allows the network to learn to predict both conditional and unconditional noise simultaneously. Consequently, at inference time, guidance is performed by interpolating between these two predictions to guide the generation process. **CaTSG** departs from this standard setup: rather than dropping the whole condition, it replaces each temporal token of the fused context of each environment branch by a learned null context independently with probability $p_{\text{drop}}$ (Section 4.3). Its $\boldsymbol{\epsilon}^{\text{base}}$ is therefore an all-null-context prediction of that same network, and we do not claim it is a separately calibrated unconditional model.

# C More Details on Model Design

## C.1 Motivation for a Diffusion Backbone

Our choice of a diffusion backbone is driven by the goals of Int-TSG and CF-TSG. First, under the score-based view, diffusion models learn and approximate scores $s(x, c) = \nabla_x \log p(x \mid c)$ (Eq. 1). Given the backdoor decomposition of the interventional target $p(x \mid \mathrm{do}(c))$ instead of $p(x \mid c)$, our objective naturally extends to the interventional score $s(x, \mathrm{do}(c)) = \nabla_x \log p(x \mid \mathrm{do}(c))$, which can be derived from the backdoor-adjusted formulation (Eq. 2). The derivation will be introduced in following sections. Second, diffusion models have become a powerful paradigm for time series modeling, achieving strong performance across imputation, generation, and forecasting tasks (Tashiro et al., 2021; Wen et al., 2023; Yuan & Qiao, 2024; Yang et al., 2024; Lin et al., 2024), offering stable training and good coverage compared to likelihood-free adversarial alternatives. These properties make diffusion a pragmatic and principled backbone for our causal TSG agenda.

## C.2 Derivation of backdoor-adjusted score function

**Goal.** For the interventional and counterfactual generation tasks, rather than sampling from the observational distribution $p(x \mid c)$, we aim to perform guided sampling from the interventional distribution $p(x \mid \mathrm{do}(c))$ and the counterfactual distribution $p(x' \mid x, c, c')$. Accordingly, instead of the conditional score $s_t(x_t, c) = \nabla_{x_t} \log p(x_t \mid c)$ in Eq. 21, we target

$$
s_t^{\mathrm{int}}(x_t; \mathrm{do}(c)) \triangleq \nabla_{x_t} \log p(x_t \mid \mathrm{do}(c)) \quad \text{and} \quad s_t^{\mathrm{cf}}(x_t'; x, c, c') \triangleq \nabla_{x_t'} \log p(x_t' \mid x, c, c').
$$

For notational clarity, we refer to these causal score functions as $s_t^{\mathrm{int}}$ and $s_t^{\mathrm{cf}}$, where $s_t^{\mathrm{int}}$ denotes the interventional score function associated with the distribution $p(x_t \mid \mathrm{do}(c))$, and $s_t^{\mathrm{cf}}$ denotes the counterfactual score function associated with the distribution $p(x_t' \mid x, c, c')$. In the following, we first introduce the underlying causal assumptions, and then detail the derivation of both score functions.

**Causal assumption.** We posit an SCM with a latent confounder $e$ that influences both $x$ and $c$ (Figure 2b). Under this SCM, we apply the backdoor adjustment (Eq. 2) to block the confounding path $c \leftarrow e \rightarrow x$ (Figure 2c) and using Abduction-Action-Prediction (Eq. 10) to do the counterfactual generation (Figure 2d).

**Proposition C.1** (Backdoor decomposition under latent adjustment)**.** *Under assumptions (A1) and (A2) stated in Section 4.1, and given a sufficient adjustment representation of $E$, the interventional distribution admits the backdoor decomposition (Pearl et al., 2016)* $P(X \mid \mathrm{do}(C)) = \sum_e P(X \mid C, E{=}e)\, P(E{=}e)$.

Proposition C.1 is an identity for the conditional mechanisms specified by the SCM. Identifying its right-hand side from observational data additionally requires access to a valid adjustment representation and common support: the queried $c$ must lie in the support of $C \mid E{=}e$ for $P(E)$-almost every $e$. Appendix G.1 instead constructs the target from known structural equations. Fixing $c$ while drawing $E$ independently may produce pairs outside observational support, so that evaluation tests structural extrapolation rather than observational identification.

Based on this SCM and the backdoor adjustment, we derive backdoor-adjusted forms of the score functions for $s_t^{\mathrm{int}}$ and $s_t^{\mathrm{cf}}$ below.

**Interventional score function.** We first derive the interventional score function $s_t^{\mathrm{int}} = \nabla_{x_t} \log p(x_t \mid \mathrm{do}(c))$. Following the standard backdoor formulation in Eq. 2, the interventional score function $s_t^{\mathrm{int}}$ can be expressed by marginalizing over $e$:

$$\nabla_x \log p(x \mid \mathrm{do}(c)) = \nabla_x \log \left( \sum_e p(x \mid c, e)\, p(e) \right). \tag{25}$$

The summation over latent confounder $e$ is inside the logarithm, preventing direct application of the gradient operator. This is a well-known difficulty in score-based inference under marginalization. To address this, we adopt a classical result often referred to as the *log-derivative of a marginal* or *softmax trick* (Mnih & Gregor, 2014), given by:

$$\nabla_x \log \sum_i a_i(x) = \sum_i \frac{a_i(x)}{\sum_j a_j(x)} \nabla_x \log a_i(x). \tag{26}$$

Applying this identity to our target objective with $a_i(x) = p(x \mid c, e_i)\, p(e_i)$, Eq. 25 can be written as:

$$\nabla_x \log \left( \sum_e p(x \mid c, e)\, p(e) \right) = \sum_e \underbrace{\frac{p(x \mid c, e)\, p(e)}{\sum_{e'} p(x \mid c, e')\, p(e')}}_{\text{mixture responsibility } r_e(x,c)} \cdot \nabla_x \log p(x \mid c, e). \tag{27}$$

That is, the clean-space interventional score is the responsibility-weighted expectation of environment-specific scores:

$$\nabla_x \log p(x \mid \mathrm{do}(c)) = \mathbb{E}_{e \sim r_e(x,c)} \left[ \nabla_x \log p(x \mid c, e) \right], \tag{28}$$

where the target mixture responsibility is

$$r_e(x, c) = \frac{p(x \mid c, e)\, p(e)}{\sum_{e'} p(x \mid c, e')\, p(e')} = p(e \mid x, \mathrm{do}(c)). \tag{29}$$

At diffusion time $t$, the analogous noised-space responsibility is

$$r_{e,t}(x_t, c) = \frac{p_t(x_t \mid c, e) p(e)}{\sum_{e'} p_t(x_t \mid c, e') p(e')}, \qquad r_{e,0}(x_0, c) = r_e(x_0, c). \tag{30}$$

Note that $r_e(x, c)$ is the responsibility of $e$ under the interventional mixture of Eq. 2, which differs from the observational posterior $p(e \mid x, c) \propto p(x \mid c, e)\, p(c \mid e)\, p(e)$ by the factor $p(c \mid e)$ whenever $E$ influences $C$. The counterfactual derivation below instead uses the observational posterior, since abduction infers $E$ from the factual sample. EnvInfer supplies learned proxy weights: for interventional generation $q_\phi(e \mid x_t, c)$ approximates $r_{e,t}$, and for counterfactual generation $q_\phi(e \mid x_0, c)$ provides factual abduction weights.

The clean-space interventional identity can be summarized as follows:

$$\begin{aligned}
\nabla_x \log p(x \mid \mathrm{do}(c)) &= \nabla_x \log \left( \sum_e p(x \mid c, e)\, p(e) \right) && \textit{(Using the backdoor adjustment in Eq. 2)} \\
&= \sum_e \frac{p(x \mid c, e)\, p(e)}{\sum_{e'} p(x \mid c, e')\, p(e')} \cdot \nabla_x \log p(x \mid c, e) && \textit{(Softmax trick in Eq. 26)} \\
&= \mathbb{E}_{e \sim r_e(x,c)} \left[ \nabla_x \log p(x \mid c, e) \right] && \textit{(Eq. 29)}
\end{aligned}$$

Applying the same identity to $p_t$ gives the unguided noised interventional score:

$$s_t^{\text{int}} = \nabla_{x_t} \log p_t(x_t \mid \mathrm{do}(c)) \approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \mathbb{E}_{e \sim r_{e,t}(x_t,c)} \left[ \varepsilon_\theta(x_t, t, c, e) \right]. \tag{31}$$

**Counterfactual score function.** We now detail the counterfactual score function $s_t^{\text{cf}} = \nabla_{x'_t} \log p_t(x'_t \mid x, c, c')$ for a fixed factual pair $(x, c)$. Following the abduction-action-prediction procedure (Eq. 10), the exact mixture and fixed-weight approximation can be summarized as follows:

$$\begin{aligned}
\nabla_{x'_t} \log p_t(x'_t \mid x, c, c') &= \nabla_{x'_t} \log \left( \sum_e p_t(x'_t \mid c', e)\, p(e \mid x, c) \right) && \textit{(AAP factorization in Eq. 10)} \\
&= \sum_e \frac{p_t(x'_t \mid c', e)\, p(e \mid x, c)}{\sum_{e'} p_t(x'_t \mid c', e')\, p(e' \mid x, c)} \cdot \nabla_{x'_t} \log p_t(x'_t \mid c', e) && \textit{(Softmax trick in Eq. 26)} \\
&= \mathbb{E}_{e \sim p(e \mid x'_t, x, c, c')} \left[ \nabla_{x'_t} \log p_t(x'_t \mid c', e) \right] && \textit{(State-dependent responsibility)} \\
&\approx \mathbb{E}_{e \sim p(e \mid x, c)} \left[ \nabla_{x'_t} \log p_t(x'_t \mid c', e) \right] && \textit{(Fixed factual weights).}
\end{aligned}$$

The last line fixes the factual posterior weights rather than updating the exact responsibilities with $x'_t$.

**Scope of the learned instantiation.** The derivations above are exact under the assumed SCM for the true responsibilities and the true environment-specific component scores $\nabla_x \log p(x \mid c, e)$, apart from the fixed-weight step in the last counterfactual line. **CaTSG** operationalizes them with learned quantities, and the denoising objective (Eq. 7) supervises only their weighted aggregate: no part of it constrains an individual environment-conditioned branch, whose implied score is $-\varepsilon_\theta(x_t, t, c, \mathbf{e}_k)/\sigma_t$, to equal a true component score $\nabla_{x_t} \log p_t(x_t \mid c, e)$, nor $q_\phi$ to equal $r_{e,t}$ or $p(e \mid x, c)$. The environment-conditioned branches are therefore latent model components, and we do not claim the decomposition is identified at the branch level (Appendix H.3).

### C.3 Practical Discretization Implementation

Having distinguished the unguided scores in Eq. 4 and Eq. 5 from the fixed-weight approximation, we now describe CFG-adjusted generation in practice. Both score functions rely on the conditional noise prediction network $\varepsilon_\theta(x_t, t, c, e)$, its unconditional counterpart $\varepsilon_\theta(x_t, t)$, and a guidance scale $\omega$ that balances the strength of the guidance. The key computational challenge lies in evaluating the expectation

$$\mathbb{E}_{e \sim r_{e,t}(x_t,c)} \left[ \varepsilon_\theta(x_t, t, c, e) \right],$$

which requires marginalizing over a continuous latent confounder space. This is computationally expensive and difficult to parameterize directly.

To address this, inspired by prior works (Xia et al., 2023b; Huang et al., 2025), we discretize the latent environment space into a finite set of embeddings. Specifically, we introduce a learnable environment bank $\boldsymbol{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_K\} \in \mathbb{R}^{K \times H}$, where each embedding $\mathbf{e}_k \in \mathbb{R}^H$ indexes one environment-conditioned branch of the shared denoiser, and the embeddings are encouraged to be mutually orthogonal, which promotes distinct environment codes but does not constrain the corresponding branch outputs of the shared denoiser to differ. The index $k$ is a model-internal label and is not placed in one-to-one correspondence with a value of the true latent $E$.

Under this discretization, the practical CFG-adjusted noise predictions are approximated by:

$$\hat{\boldsymbol{\epsilon}}_t^{\text{int}} \approx (1+\omega) \sum_{k=1}^{K} q_\phi(\mathbf{e}_k \mid \mathbf{x}_t, \mathbf{c})\, \varepsilon_\theta(x_t, t, c, e_k) - \omega\, \varepsilon_\theta(x_t, t), \tag{32}$$

$$\hat{\boldsymbol{\epsilon}}_t^{\text{cf}} \approx (1+\omega) \sum_{k=1}^{K} q_\phi(\mathbf{e}_k \mid \mathbf{x}_0, \mathbf{c})\, \varepsilon_\theta(x'_t, t, c', e_k) - \omega\, \varepsilon_\theta(x'_t, t), \tag{33}$$

where $q_\phi$ outputs a normalized weight vector over the $K$ environments, serving as a learned proxy for $r_{e_k,t}$ in the interventional case and as learned factual abduction weights in the counterfactual case.

To estimate these learned weights, we design an environment inference module **EnvInfer** $q_\phi$, which takes as input the current sample $\mathbf{x}_t$ and condition $\mathbf{c}$, and outputs a normalized weight vector:

$$(\mathbf{x}_t, \mathbf{c}) \xrightarrow{q_\phi} \mathbf{w} = \{w_1, \ldots, w_K\} \in \mathbb{R}^K, \quad \text{where } \sum_{k=1}^{K} w_k = 1. \tag{34}$$

The weight vector $\mathbf{w}$ represents learned mixture weights over the $K$ environment embeddings $\{\mathbf{e}_k\}$, and is used to select or mix environment embeddings for subsequent generation. Note that the two procedures differ in what $q_\phi$ conditions on. Interventional sampling uses stepwise weights $w_k(t) = q_\phi(\mathbf{e}_k \mid \mathbf{x}_t, \mathbf{c})$ recomputed at each reverse step, whereas counterfactual abduction computes $w_k = q_\phi(\mathbf{e}_k \mid \mathbf{x}_0, \mathbf{c})$ once from the observed sample and holds it fixed (Algorithm 3).

Building on this inference step, we employ a **Denoiser** $\varepsilon_\theta(\cdot)$ to realize the actual generative process. This network is parameterized by $\theta$ and takes as input the noisy sequence, time step, and (optionally) conditioning information, and outputs noise predictions that guide the reverse diffusion trajectory. Specifically, it produces two forms of predictions:

$$\begin{aligned} (\mathbf{x}_t, t, \mathbf{c}, \mathbf{e}_k) &\xrightarrow{\varepsilon_\theta(\cdot)} \boldsymbol{\epsilon}^{\text{env}} \in \mathbb{R}^{T \times D} \\ (\mathbf{x}_t, t) &\xrightarrow{\varepsilon_\theta(\cdot)} \boldsymbol{\epsilon}^{\text{base}} \in \mathbb{R}^{T \times D} \end{aligned} \tag{35}$$

Here, $\boldsymbol{\epsilon}^{\text{env}}$ denotes the environment-conditioned noise prediction that incorporates both $\mathbf{c}$ and $\mathbf{e}_k$, while $\boldsymbol{\epsilon}^{\text{base}}$ denotes the all-null-context baseline prediction, obtained when every temporal token of the fused context is replaced by the learned null context.

Together, EnvInfer $q_\phi$ and the denoising network $\varepsilon_\theta$ form the core pipeline (Figure 3): the former produces learned environment mixture weights, and the latter performs denoising under the corresponding environment-conditioned guidance. The architecture of these networks is detailed in Appendix C.4.

With the environment inference module and the denoising network, we can now approximate the interventional and counterfactual targets in practice. Although Eq. 32 and Eq. 33 are derived for different causal queries, their discrete forms share the same computational structure. Specifically, at each timestep $t$, both can be approximated by:

$$\hat{\boldsymbol{\epsilon}}_t^{\text{int}} \text{ or } \hat{\boldsymbol{\epsilon}}_t^{\text{cf}} \;\approx\; (1+\omega) \sum_{k=1}^{K} w_k \, \boldsymbol{\epsilon}_k^{\text{env}} - \omega \, \boldsymbol{\epsilon}^{\text{base}}, \tag{36}$$

where the input pair $(\mathbf{x}_t, \mathbf{c})$ corresponds to interventional guidance and $(\mathbf{x}'_t, \mathbf{c}')$ corresponds to counterfactual guidance.

### C.4 Module Architecture

Based on the above discussion, we need one denoising network $\varepsilon_\theta$ and one EnvInfer module $q_\phi$, will detailed below.

**EnvInfer $q_\phi$. (Figure 7)** Take $\mathbf{x} \in \mathbb{R}^{N \times T \times D}$, $\mathbf{c} \in \mathbb{R}^{N \times T \times D_c}$ and $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_K\} \in \mathbb{R}^{K \times H}$ (a set of learnable embeddings) as input, EnvInfer $q_\phi$ produces (1) a latent environment embedding $\mathbf{h} \in \mathbb{R}^{N \times D_h}$ and (2) a probability $\mathbf{w} = \{w_1, \ldots, w_K\} \in \mathbb{R}^K$ over the environment bank $\mathbf{E}$. First, a Temporal Convolutional Network (TCN) (Bai et al., 2018) as encoder maps the concatenation $[\mathbf{x}, \mathbf{c}]$ to $\mathbf{h}' \in \mathbb{R}^{N \times T \times H}$. Then, we extract three types of features from $\mathbf{h}'$:

- **Temporal statistics:** $\boldsymbol{\mu} = \text{mean}_t(\mathbf{h}')$, $\boldsymbol{\sigma} = \text{std}_t(\mathbf{h}')$, $\boldsymbol{m} = \max_t(\mathbf{h}')$, concatenate them we have $\mathbf{h}_{\text{stat}} = [\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{m}] \in \mathbb{R}^{N \times 3H}$.
- **Attention pooling:** a linear score on channel vectors yields weights $\mathbf{a} = \text{softmax}_t(\text{Linear}(\mathbf{h}'^{\top})) \in \mathbb{R}^{N \times T \times 1}$; the attention feature is $\mathbf{h}_{\text{att}} = (\mathbf{h}' \odot \mathbf{a}^{\top})\text{sum}_t \in \mathbb{R}^{N \times H}$.
- **Spectral analysis:** power spectrum $\text{psd} = |\text{rFFT}_t(\mathbf{h}')|^2 \in \mathbb{R}^{N \times H \times T_f}$ with $T_f = \lfloor T/2 \rfloor + 1$; spectral centroid per channel $\mathbf{c}_{\text{spec}} = \frac{(\text{psd} \cdot f)\,\text{sum}_t}{\text{psd}\,\text{sum}_t} \in \mathbb{R}^{N \times H}$, and top-$K_p$ global peaks from

psd averaged over channels, $\mathbf{p}_{\text{top}} \in \mathbb{R}^{N\times K_p}$. Concatenate and we have $\mathbf{h}_{\text{spec}} = [\mathbf{c}_{\text{spec}}, \mathbf{p}_{\text{top}}] \in \mathbb{R}^{N\times(H+K_p)}$.

Concatenating all representations, we have $\mathbf{h}'' = [\mathbf{h}_{\text{stat}}, \mathbf{h}_{\text{att}}, \mathbf{h}_{\text{spec}}] \in \mathbb{R}^{N\times(5H+K_p)}$. We then project this representation of features to latent and normalization. A LayerNorm and MLP projects features $\mathbf{h} = \tanh\big(\text{MLP}(\mathbf{h}'')\big) \in \mathbb{R}^{N\times H}$. We apply row-wise $\ell_2$-normalization to $\mathbf{h}$ and $\mathbf{E}$, yielding unit-norm vectors for bank matching. Next, temperature-scaled dot-product scores and softmax produce the environment probabilities:

$$\mathbf{s} = \frac{\mathbf{h}\mathbf{E}^\top}{\tau} \in \mathbb{R}^{N\times K}, \qquad \mathbf{w} = \text{softmax}(\mathbf{s}) \in \mathbb{R}^{N\times K},$$

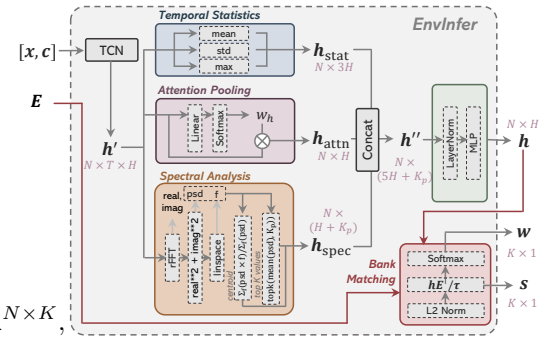



Figure 7: Architecture of **EnvInfer** $q_\phi$. The module takes input sequences $(\mathbf{x}, \mathbf{c})$ (or their augmented versions $(\mathbf{x}_v, \mathbf{c}_v)$) together with the environment bank $\mathcal{E}$, and produces latent representations $\mathbf{h}$, sample weights $\mathbf{w}$, and logits $\mathbf{s}$.

where $\mathbf{w} = \{w_1, \ldots, w_K\}$ and $\sum_{k=1}^K w_k = 1$. In training, these scores $\mathbf{s}$ can be combined with a balanced assignment (e.g., Sinkhorn) to form targets, while $\mathbf{w}$ serves as learned mixture weights in our guidance rules at inference, which will be detailed in the following section. Therefore, EnvInfer outputs a tuple consisting of latent representations, scores, and mixture weights $(\mathbf{h}, \mathbf{s}, \mathbf{w})$. The first two components are only used to compute the swapped prediction loss $\mathcal{L}_{\text{sw}}$ (Eq. 8) and do not participate in the generation process. In contrast, the estimated environment probability $\mathbf{w}$ is employed both for computing the noise prediction loss $\mathcal{L}_{\text{eps}}$ (Eq. 7) during training and for backdoor-adjusted guidance in generation (Eq. 28).

**Denoiser $\varepsilon_\theta$.** The denoiser $\varepsilon_\theta$ in **CaTSG** is implemented as a conventional U-Net architecture for 1D time series. It consists of a downsampling encoder path, a bottleneck module, and an upsampling decoder path with skip connections. This standard design has proven effective in diffusion-based generative models, and we adopt it directly as the backbone of our framework.

# D More Details on Training and Inference

We now describe how **CaTSG** is trained and used during inference. We first detail the training objective and optimization strategy for EnvInfer $q_\phi$, the environment bank $E$, and the denoising network $\varepsilon_\theta$, then explain the interventional and counterfactual inference procedures. Algorithms 1, 2, and 3 summarize the respective training and sampling procedures.

## D.1 Training Objective & Optimization

We optimize three components jointly: (1) the denoising network $\varepsilon_\theta$, (2) the environment inference network (i.e., EnvInfer $q_\phi$), (3) the learnable environment bank $E$.

**Training Denoiser.** For the backbone of the proposed framework, i.e., denoising network $\varepsilon_\theta$, we train it via $\mathcal{L}_{\text{eps}} = \mathbb{E}_{\mathbf{x}_0,t,\boldsymbol{\epsilon},\tilde{\mathbf{d}}}\Big[\big\|\boldsymbol{\epsilon} - \sum_{k=1}^K \tilde{w}_k \varepsilon_\theta(\mathbf{x}_t, t, \tilde{\mathbf{d}}_k)\big\|_2^2\Big]$ with $\tilde{\mathbf{d}}_k$ denoting the token-wise masked fused context of branch $k$, as described in Eq. 7. In every mini-batch, all $K$ branches are evaluated and the EnvInfer weights $\mathbf{w}$ are balanced across the mini-batch as $\tilde{\mathbf{w}} = \text{Sinkhorn}_{\epsilon=0.05,\, n=3}(\log \text{clamp}(\mathbf{w}, 10^{-12}))$ before mixing. The loss therefore supervises the $\tilde{\mathbf{w}}$-weighted aggregate of the $K$ environment-conditioned predictions, rather than any branch separately. The Sinkhorn step is applied without gradient, so $\mathcal{L}_{\text{eps}}$ does not update EnvInfer, and $\mathcal{L}_{\text{sw}}$ is the only term that does. The balancing applies to training only: interventional and counterfactual sampling mix the branches with the raw learned weights $\mathbf{w}$ returned by $q_\phi$ (Algorithms 2 and 3), and no Sinkhorn step is applied at inference.

**Algorithm 1** `CaTSG` Training procedure

**Input:** Training data $\mathcal{D} = \{(\mathbf{x}_0, \mathbf{c})\}$; #env $K$; temperature $\tau$; Sinkhorn reg. $\eta = 0.05$ with 3 iterations; weights $\alpha, \beta$, drop probability $p_{\text{drop}}$
**Output:** Trained Denoiser $\varepsilon_\theta$, EnvInfer $q_\phi$, Env. Bank $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_K\}$
**repeat**
  Sample data $(\mathbf{x}_0, \mathbf{c}) \in \mathcal{D}$
  // For optimizing EnvInfer
  Construct two augmentations $\{(\mathbf{x}_0', \mathbf{c}'), (\mathbf{x}_0'', \mathbf{c}'')\}$ ▷ Two-view synchronized augmentations
  $\mathcal{V} \leftarrow \{(\mathbf{x}_0, \mathbf{c}), (\mathbf{x}_0', \mathbf{c}'), (\mathbf{x}_0'', \mathbf{c}'')\}, V \leftarrow |\mathcal{V}| = 3$ ▷ Clean view and its two augmentations
  $\{(\mathbf{h}^{(v)}, \mathbf{s}^{(v)}, \mathbf{w}^{(v)})\}_{v=1}^{V} \leftarrow q_\phi(\mathcal{V}, \mathbf{E})$ ▷ Obtain representations, logits, probabilities
  $\{\hat{\mathbf{w}}^{(v)}\}_{v=1}^{V} \leftarrow \text{sg}\left[\text{Sinkhorn-Knopp}(\{\mathbf{s}^{(v)}\}_{v=1}^{V})\right]$ ▷ Balanced soft assignments, sg = stop-gradient
  $\mathcal{L}_{\text{sw}} \leftarrow \frac{1}{V(V-1)} \sum_{u \neq v} \ell(\mathbf{w}^{(v)}, \hat{\mathbf{w}}^{(u)})$ with $\ell(\cdot, \cdot)$ denoting cross-entropy ▷ Swapped prediction loss (Eq. 8)
  // For optimizing Denoiser
  Sample $t \sim \mathcal{U}(\{1, \ldots, T\})$, noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$
  Corrupt data: $\mathbf{x}_t \leftarrow \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
  $(\mathbf{h}, \mathbf{s}, \mathbf{w}) \leftarrow q_\phi(\mathbf{x}_t, \mathbf{c}, \mathbf{E})$, where $\mathbf{w} = \{w_1, \ldots, w_K\}$ ▷ Infer mixture weights at the noisy state $\mathbf{x}_t$
  $\tilde{\mathbf{w}} \leftarrow \text{Sinkhorn}_{\eta=0.05,\, n=3}(\log \text{clamp}(\mathbf{w}, 10^{-12}))$ ▷ Balance weights within the mini-batch
  **for** $k = 1$ **to** $K$ **do**
    Construct $\tilde{\mathbf{d}}_k$ by independently replacing each temporal token of branch $k$'s fused context with the learned null context with probability $p_{\text{drop}}$ ▷ Token-wise, branch-wise CFG masking
    $\hat{\boldsymbol{\epsilon}}_k \leftarrow \varepsilon_\theta(\mathbf{x}_t, t, \tilde{\mathbf{d}}_k)$ ▷ Predicting environment-aware noises
  $\mathcal{L}_{\text{eps}} \leftarrow \|\boldsymbol{\epsilon} - \sum_{k=1}^{K} \tilde{w}_k \hat{\boldsymbol{\epsilon}}_k\|_2^2$ ▷ Noise prediction loss (Eq. 7)
  // For optimizing Env. Bank
  $\bar{\mathbf{e}}_k \leftarrow \mathbf{e}_k / \|\mathbf{e}_k\|_2$ for $k = 1, \ldots, K$; $\mathcal{L}_{\text{orth}} \leftarrow \sum_{k \neq \ell} (\bar{\mathbf{e}}_k^\top \bar{\mathbf{e}}_\ell)^2$ ▷ Pairwise cosine penalty
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{eps}} + \alpha \mathcal{L}_{\text{sw}} + \beta \mathcal{L}_{\text{orth}}$
  Update $\theta$, $\phi$, $\mathbf{E}$ by one optimizer step on $\mathcal{L}$
**until** *converged*

**Training EnvInfer.** For training EnvInfer $q_\phi$, inspired by SwAV (Caron et al., 2020) we adopted a swapped prediction loss. Intuitively, the SwAV-style loss encourages agreement within the same environment and diversity across environments: two augmentations of the same sample are encouraged to produce consistent environment assignments (capturing shared causal context), while the orthogonality regularization keeps different environment embeddings distinct. This "consistency-under-augmentation" principle aligns closely with the causal goal of discovering latent environmental factors that remain stable across perturbations. In a word, we take the clean input together with augmented versions of it and force the assignment predicted from one view to match the target assignment of another, so all views agree on an environment embedding despite different temporal perturbations. Specifically, for each sample $(\mathbf{x}, \mathbf{c})$ we form two synchronized augmentations and use them together with the clean sample, giving the view set $\{(\mathbf{x}_v, \mathbf{c}_v)\}_{v=1}^{V}$ with $V = 3$. For view $v$, via EnvInfer, we obtain latent representation $\mathbf{h}_v \in \mathbb{R}^{N \times H}$, logits $\mathbf{s}_v = \frac{1}{\tau}\mathbf{h}_v \mathbf{E}^\top \in \mathbb{R}^{N \times K}$ and the predicted probability $\mathbf{w}_v = \text{softmax}(\mathbf{s}_v) \in \mathbb{R}^{N \times K}$, where $\mathbf{E} = \{e_1, \ldots, e_K\} \in \mathbb{R}^{K \times H}$ is the learnable environment bank and $\tau > 0$ is a temperature. For each ordered pair of distinct views $u \neq v$, the loss enforces the distribution $\mathbf{w}_v$ predicted from view $v$ to match the target assignment $\hat{\mathbf{w}}_u$ of view $u$, and the $V(V-1)$ ordered pairs are averaged:

$$\mathcal{L}_{\text{sw}} = \frac{1}{V(V-1)} \sum_{u \neq v} \ell(\mathbf{w}_v, \hat{\mathbf{w}}_u),$$

where $\ell(\mathbf{w}_v, \hat{\mathbf{w}}_u) = -\sum_{k=1}^{K} \hat{w}_{u,k} \log w_{v,k}$ is the cross-entropy between the target assignment $\hat{\mathbf{w}}_u$ and the predicted distribution $\mathbf{w}_v$. Following (Caron et al., 2020), the target assignments $\hat{\mathbf{w}}_u$ are computed online using the Sinkhorn-Knopp algorithm (Cuturi, 2013) to enforce balanced partitioning of samples across environment embeddings, and they are treated as constants, so no gradient flows through them. This encourages different augmentations of the same input to agree on the same

environment embeddings, while avoiding degenerate solutions where all samples collapse to a single embedding. In our practice we use two augmentations, so $V = 3$ views and $V(V-1) = 6$ ordered pairs contribute to $\mathcal{L}_{\text{sw}}$. Note that this swapped-prediction path is the only one that consumes the clean $\mathbf{x}_0$. The denoising path of Eq. 7 instead queries EnvInfer at the noisy state $\mathbf{x}_t$ (Algorithm 1).

**Training Environment Bank.** The environment bank $E$ is initialized with orthogonal initialization, followed by L2 normalization along the feature dimension. In addition, to encourage diversity among environment embeddings, we row-normalize the learnable bank $\mathbf{E} \in \mathbb{R}^{K \times H}$ and regularize it with:

$$\mathcal{L}_{\text{orth}} = \sum_{k \neq \ell} (\bar{\mathbf{e}}_k^\top \bar{\mathbf{e}}_\ell)^2,$$

where $\bar{\mathbf{e}}_k = \mathbf{e}_k / \|\mathbf{e}_k\|_2$. This loss penalizes pairwise cosine similarity between different environment embeddings.

The total training objective combines the denoising and environment regularization terms: $\mathcal{L} = \mathcal{L}_{\text{eps}} + \alpha \mathcal{L}_{\text{sw}} + \beta \mathcal{L}_{\text{orth}}$, where $\alpha$ and $\beta$ are balance coefficients.

**Computational considerations.** The mixture-style denoising loss in Eq. 7 requires $K$ forward passes per sample and step. When $K$ is moderate, this overhead is acceptable. In our tasks, $K = 4$, and the end-to-end cost remains modest (Appendix G.3). Potential future reductions include top-$r$ pruning or sparse gating, neither of which is used in the reported experiments.

### D.2 Intervention generation

We show how backdoor-adjusted guidance is used for intervention generation. Given a context $\mathbf{c}$, our goal is to generate samples from the interventional distribution $P(X \mid \text{do}(C = \mathbf{c}))$. To achieve this, we combine the environment inference module $q_\phi$ with the environment bank $\mathcal{E}$ to obtain learned mixture weights, and the denoiser $\varepsilon_\theta$ to predict noise. Here, $q_\phi$ produces a latent embedding from $(\mathbf{x}_t, \mathbf{c})$, which is matched against $\mathbf{E}$ to obtain mixture weights $\mathbf{w} = \{w_1, \ldots, w_K\}$ with $\sum_{k=1}^K w_k = 1$.

At each denoising step, the denoiser produces both an all-null-context baseline prediction $\boldsymbol{\epsilon}^{\text{base}} = \varepsilon_\theta(x_t, t)$ and environment-aware predictions $\boldsymbol{\epsilon}_k^{\text{env}} = \varepsilon_\theta(x_t, t, c, e_k)$ for each embedding $e_k \in \mathcal{E}$. These predictions are then aggregated according to the backdoor-adjusted rule

$$\hat{\boldsymbol{\epsilon}} = (1 + \omega) \sum_{k=1}^K w_k \, \boldsymbol{\epsilon}_k^{\text{env}} - \omega \, \boldsymbol{\epsilon}^{\text{base}}, \tag{37}$$

which corresponds to Eq. 36. Intuitively, this adjustment interpolates between environment-conditioned and all-null-context predictions, preventing trivial dominance of either branch. The resulting $\hat{\boldsymbol{\epsilon}}$ is supplied to the second-order single-step DPM-Solver update. The full procedure is summarized in Algorithm 2.

**Algorithm 2** Interventional Generation via Backdoor-adjusted Guidance

**Input:** Guidance scale $\omega$, context $\mathbf{c}$, Env. Bank $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_K\}$, Denoiser $\varepsilon_\theta$, EnvInfer $q_\phi$
**Output:** Generated interventional sample $\hat{\mathbf{x}}$ under context $\mathbf{c}$
Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
**for** *each second-order single-step DPM-Solver step* $t$ **do**
  $\hat{\boldsymbol{\epsilon}}^{\text{base}} \leftarrow \varepsilon_\theta(\mathbf{x}_t, t)$ ▷ Predict unconditional noise
  $(\mathbf{h}, \mathbf{w}) \leftarrow q_\phi(\mathbf{x}_t, \mathbf{c}, \mathbf{E})$, where $\mathbf{w} = \{w_1, \ldots, w_K\}$ ▷ Infer learned mixture weights
  **for** $k = 1$ **to** $K$ **do**
    $\hat{\boldsymbol{\epsilon}}_k^{\text{env}} \leftarrow \varepsilon_\theta(\mathbf{x}_t, t, \mathbf{c}, \mathbf{e}_k)$ ▷ Predict env-aware noise
  $\hat{\boldsymbol{\epsilon}} \leftarrow (1 + \omega) \sum_{k=1}^K w_k \cdot \hat{\boldsymbol{\epsilon}}_k^{\text{env}} - \omega \cdot \hat{\boldsymbol{\epsilon}}^{\text{base}}$ ▷ Backdoor-adjusted guidance in Eq. 6
  $\mathbf{x}_{t-} \leftarrow \text{DPM-SolverStep}(\mathbf{x}_t, \hat{\boldsymbol{\epsilon}}, t)$ ▷ DPM-Solver update
**return** $\hat{\mathbf{x}} \leftarrow \mathbf{x}_{t_{\text{final}}}$

### D.3 COUNTERFACTUAL GENERATION

We show how AAP (Eq. 10) works in counterfactual generation. Given an observed target $\mathbf{x}_0$ with context $\mathbf{c}$, our goal is to generate a counterfactual sample corresponding to an alternative context $\mathbf{c}'$, i.e., $P(X' \mid X = \mathbf{x}_0, C = \mathbf{c}, C' = \mathbf{c}')$. To this end, we follow the AAP procedure. In the *abduction* step, the environment inference module EnvInfer $q_\phi$ obtains learned factual mixture weights $q_\phi(\mathbf{e}_k \mid \mathbf{x}_0, \mathbf{c})$ by matching the latent embedding of $(\mathbf{x}_0, \mathbf{c})$ against the environment bank $\mathbf{E}$. In the *action* step, we replace the original context $\mathbf{c}$ with the counterfactual context $\mathbf{c}'$. Finally, in the *prediction* step, we run the denoising diffusion process with backdoor-adjusted guidance. At each denoising step, the denoiser produces both an all-null-context baseline prediction $\boldsymbol{\epsilon}^{\text{base}} = \varepsilon_\theta(\mathbf{x}'_t, t)$ and environment-aware predictions $\boldsymbol{\epsilon}^{\text{env}}_k = \varepsilon_\theta(\mathbf{x}'_t, t, \mathbf{c}', \mathbf{e}_k)$ for each environment embedding $\mathbf{e}_k \in \mathbf{E}$. These predictions are aggregated using the same rule as in intervention generation, which gives the guided noise prediction corresponding to the fixed-weight counterfactual score approximation. The exact AAP mixture has state-dependent responsibilities $p(e \mid x'_t, x, c, c')$; during counterfactual generation, the implementation instead holds the learned factual proxy weights $q_\phi(e \mid x_0, c)$ fixed. The resulting $\hat{\boldsymbol{\epsilon}}$ is supplied to the second-order single-step DPM-Solver update, yielding a counterfactual trajectory that preserves the learned factual environment information while reflecting the counterfactual context. The full procedure is summarized in Algorithm 3.

**Algorithm 3** Counterfactual Generation via Abduction-Action-Prediction

**Input:** Guidance scale $\omega$, observed target $\mathbf{x}_0$, observed context $\mathbf{c}$, counterfactual context $\mathbf{c}'$, Env. Bank $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_K\}$, Denoiser $\varepsilon_\theta$, EnvInfer $q_\phi$
**Output:** Generated counterfactual sample $\hat{\mathbf{x}}'$ under context $\mathbf{c}'$
**Step 1 - Abduction:** $(\mathbf{h}, \mathbf{w}) \leftarrow q_\phi(\mathbf{x}_0, \mathbf{c}, \mathbf{E})$, where $\mathbf{w} = \{w_1, \ldots, w_K\}$ ▷ Infer learned factual mixture weights

**Step 2 - Action:** Replacing $\mathbf{c} \leftarrow \mathbf{c}'$ ▷ Construct new world

**Step 3 - Prediction:**
Sample $\mathbf{x}'_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
**for** *each second-order single-step DPM-Solver step* $t$ **do**
  $\hat{\boldsymbol{\epsilon}}_{\text{base}} \leftarrow \varepsilon_\theta(\mathbf{x}'_t, t)$ ▷ Predict unconditional noise
  **for** $k = 1$ **to** $K$ **do**
    $\hat{\boldsymbol{\epsilon}}^{\text{env}}_k \leftarrow \varepsilon_\theta(\mathbf{x}'_t, t, \mathbf{c}', \mathbf{e}_k)$ ▷ Predict env-aware noise
  $\hat{\boldsymbol{\epsilon}} \leftarrow (1+\omega)\sum_{k=1}^{K} w_k \cdot \hat{\boldsymbol{\epsilon}}^{\text{env}}_k - \omega \cdot \hat{\boldsymbol{\epsilon}}^{\text{base}}$ ▷ Fixed-weight counterfactual guidance in Eq. 6
  $\mathbf{x}'_{t-} \leftarrow \text{DPM-SolverStep}(\mathbf{x}'_t, \hat{\boldsymbol{\epsilon}}, t)$ ▷ DPM-Solver update
**return** $\hat{\mathbf{x}}' \leftarrow \mathbf{x}'_{t_{\text{final}}}$

## E MORE DETAILS ON DATASETS

### E.1 SYNTHETIC DATASET DESCRIPTION

To systematically evaluate causal time series generation under known causal mechanisms, we construct two synthetic benchmark datasets: **Harmonic-VM** and the more complex **Harmonic-VP**. These datasets simulate a class of damped mechanical oscillators (Figure 8a) governed by second-order differential equations (Eq. 38). We first describe the governing equation and then explain how the datasets are constructed.

**Governing Equation.** We consider a damped harmonic oscillator illustrated in Figure 8a, governed by the following second-order differential equation:

$$m \cdot \ddot{x}(t) + \gamma \cdot \dot{x}(t) + k \cdot x(t) = 0, \tag{38}$$

where:

- $x(t)$ denotes the position at time $t$,
- $\dot{x}(t) \equiv v(t)$ denotes the velocity at time $t$,

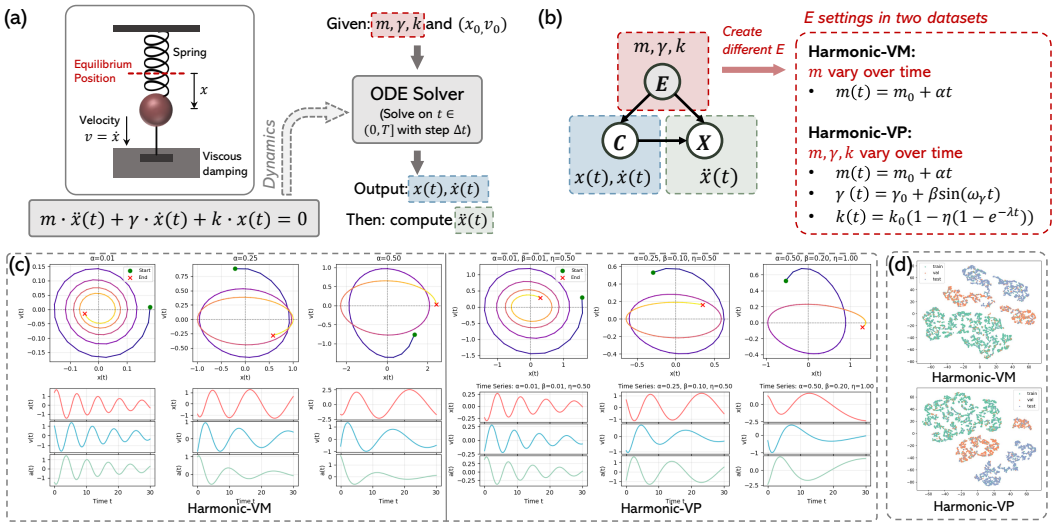



Figure 8: (a) A damped harmonic oscillator governed by $m\ddot{x}(t) + \gamma\dot{x}(t) + kx(t) = 0$, where an ODE solver, given $(m, \gamma, k)$ and initial state $(x_0, v_0)$, produces trajectories of position $x(t)$, velocity $\dot{x}(t)$, and acceleration $\ddot{x}(t)$. (b) Mapping the variables in this physical system to our SCM (Figure 2b), where the latent environment $E$ specifies physical parameters $(m, \gamma, k)$. By varying $E$, we create two datasets: **Harmonic-VM**, with mass $m$ varying over time, and **Harmonic-VP**, with mass, damping, and stiffness all varying over time. (c) Example trajectories under different parameter settings, shown in phase space (*top*) and as time series of position, velocity, and acceleration (*bottom*). (d) t-SNE visualization showing that different environments yield distinct clusters in the data distribution.

- $\ddot{x}(t) \equiv a(t)$ denotes the acceleration at time $t$,
- $m$ is the mass of the object,
- $\gamma$ is the damping coefficient,
- $k$ is the stiffness coefficient.

**Variable Roles.** We then introduce how we define the variable to construct the dataset under the SCM in Figure 2b based on Eq. 38.

- **Choice of $X$, $C$, and $E$.** As shown in Figure 8b (*left* part), we define the target series $X$ as the acceleration sequence $a(t)$ (or $\ddot{x}(t)$), the contextual variables $C$ as the instantaneous position $x(t)$ and velocity $v(t)$ (or $\dot{x}(t)$), and the environment-level parameters $E$ as the physical coefficients governing the dynamical system, which will be introduced below. Note that we choose $a(t)$ as $X$ because, according to the governing second-order differential equation of motion, acceleration is uniquely determined by the current position, velocity, and system parameters (Lehrman, 1998). This ensures a clear causal direction $E \to X$ and $C \to X$ without instantaneous feedback from $X$ to $C$.
- **Design of $E$.** As defined above, $E$ corresponds to the physical coefficients $(m, \gamma, k)$ in Eq. 38, which govern the system's mass, damping, and stiffness, respectively. These coefficients can either remain constant or vary over time, allowing us to represent a wide range of dynamical behaviors. For example, an increasing $m$ simulates mass growth (e.g., a spring absorbing water), a periodically varying $\gamma$ models oscillatory damping, and a decaying $k$ captures progressive stiffness loss.

**Dataset Design.** We consider two levels of environment diversity to create two datasets, shown in Figure 8b (*right* part). For both scenarios, we simulate environment diversity by sampling coefficients from non-overlapping intervals for training, validation, and testing, detailed as follows.

- **Harmonic-VM** (short for *Harmonic Oscillator with Variable Mass*): the mass evolves over time according to $m(t) = m_0 + \alpha\, t$, where $m_0 = 0.5$ is the initial mass. The damping coefficient $\gamma$ and

the stiffness $k$ remain constant. This scenario is analogous to a sponge suspended from a spring gradually absorbing water, resulting in an increasing mass over time.

We assign *dominant, non-overlapping* intervals of $\alpha$ to each split, detailed in Table 6. For each sample, $\alpha$ is drawn from an *80–20 mixture*: with probability $0.8$ from the split's dominant interval and with probability $0.2$ from the union of the other splits' intervals.

- **Train range** (slow mass growth): represents relatively slow mass growth, where the system remains close to its original oscillation characteristics.
- **Validation range** (moderate mass growth): moderate mass increase that noticeably reduces oscillation frequency over time.
- **Test range** (rapid mass growth): rapid mass growth leading to substantial reduction in oscillation frequency and stronger damping effects.

- **Harmonic-VP** (short for *Harmonic Oscillator with Variable Parameters*): all three physical parameters ($m(t)$, $\gamma(t)$, and $k(t)$) are allowed to evolve over time according to smooth nonlinear functions:

$$m(t) = m_0 + \alpha\, t, \quad \gamma(t) = \gamma_0 + \beta\, \sin(\omega_\gamma t), \quad k(t) = k_0\,(1 - \eta\,(1 - e^{-\lambda t})). \tag{39}$$

where $m_0 = 0.5$ is the initial mass, $\gamma_0 = 0.1$ is the damping coefficient, $\omega_\gamma = 0.2$ controls the damping oscillation frequency, $k_0 = 1.0$ is the stiffness, and $\lambda = 0.05$ governs the stiffness decay rate.

This setting reflects more complex environments in which the mass grows at a constant rate (controlled by $\alpha$), the damping exhibits periodic modulation (amplitude $\beta$ and frequency $\omega_\gamma$), and the stiffness undergoes an exponential decay (rate $\lambda$).

We assign *dominant, non-overlapping* parameter intervals for $(\alpha, \beta, \eta)$ to train/validation/test to create clearly distinguishable oscillatory regimes (see Table 6). For each split and each parameter, samples are drawn from an 80–20 mixture: with probability $0.8$ from the split's dominant interval and with probability $0.2$ from the union of the other splits' intervals (tails). This mild cross-split mixing preserves regime separability while providing limited overlap to test robustness under small distribution shift.

- **Train range** (*light damping regime*): low $(\alpha, \beta, \eta)$ values correspond to slow mass saturation, gentle periodic damping modulation, and gradual stiffness decay. This regime resembles a lightly damped system where the mass increases slowly, damping changes are minimal, and stiffness remains nearly constant over long durations, resulting in sustained oscillations with only mild amplitude reduction.
- **Validation range** (*moderate damping regime*): moderate $(\alpha, \beta, \eta)$ values lead to faster mass growth, more pronounced damping oscillations, and noticeable stiffness weakening over time. The system exhibits a balance between energy retention and dissipation, with observable but not abrupt changes in both oscillation amplitude and frequency.
- **Test range** (*strong damping regime*): high $(\alpha, \beta, \eta)$ values produce rapid mass saturation, strong periodic damping fluctuations, and accelerated stiffness decay. This regime mimics a harsh environment where the system quickly loses stiffness and experiences significant damping variation, causing oscillations to decay rapidly and frequencies to shift noticeably from those in the training regime.

By explicitly parameterizing environment-level factors (e.g., $\alpha$, $\beta$, $\eta$), our synthetic datasets support counterfactual queries. Given an observed trajectory under one environment configuration, we keep these parameters fixed and change the initial state, enabling controlled simulation of alternative outcomes. This property is crucial for evaluating counterfactual time series generation models in a setting with known ground truth.

For both synthetic datasets, trajectories are integrated with SciPy's default RK45 solver over $[0, 30]$ and recorded at 96 uniformly spaced time points. Harmonic-VM uses $m_0 = 0.5$, $\gamma = 0.1$, and $k = 1.0$; Harmonic-VP uses $m_0 = 0.5$, $\gamma_0 = 0.1$, $k_0 = 1.0$, $\omega_\gamma = 0.2$, and $\lambda = 0.05$.

**Simulation and Construction Process.** We construct the dataset in two stages:

**Step 1: Factual trajectory simulation.**

- *Parameter sampling.* Sample the environment-level factors from predefined ranges: in Harmonic-VM each environment is associated with a unique $\alpha$, and in Harmonic-VP with a triplet $(\alpha, \beta, \eta)$.

- *Initial-state sampling.* For each trajectory, sample the initial position $x_0$ and velocity $v_0$ from designated factual ranges $(x_{\min}, x_{\max})$ and $(v_{\min}, v_{\max})$.
- *Numerical integration.* With the initial state $(x_0, v_0)$, solve the ODE in Eq. 38 to obtain:

$$x = [x_1, \ldots, x_T] \in \mathbb{R}^T, \quad v = [v_1, \ldots, v_T] \in \mathbb{R}^T, \quad a = [a_1, \ldots, a_T] \in \mathbb{R}^T.$$

The acceleration sequence $a$ is taken as the target time series $\mathbf{x}$, and the contextual sequence is defined as $\mathbf{c} = [\mathbf{c}_1, \ldots, \mathbf{c}_T] \in \mathbb{R}^{T\times 2}$, where $\mathbf{c}_t = (x_t, v_t)$. Example trajectories under different parameter settings are shown in Figure 8c.

**Step 2: Counterfactual trajectory construction.**

- *Intervention on initial state.* Given an observed trajectory $(\mathbf{x}, \mathbf{c})$ simulated under environment parameter $\alpha$ or $(\alpha, \beta, \eta)$, keep these parameters fixed and independently redraw the initial state $(x_0', v_0')$ from the designated counterfactual ranges. The counterfactual position range is non-overlapping with the factual position range; the velocity ranges overlap on $[-1.5, -1.0]$.
- *Re-simulation.* Re-solve Eq. 38 under the same environment parameters but with the modified initial state to obtain the counterfactual trajectory $(\mathbf{x}', \mathbf{c}')$.
- *Pairing.* Store the factual–counterfactual pair: $\big[(\mathbf{x}, \mathbf{c}), (\mathbf{x}', \mathbf{c}')\big]$.

In our implementation, the factual initial states are sampled from $x_0 \in [-2.0, 2.0]$ and $v_0 \in [-1.5, 1.5]$ (expanded for diversity), while counterfactual initial states are drawn from $x_0' \in [2.2, 4.0]$ and $v_0' \in [-2.5, -1.0]$.

Table 6: Summary of datasets and splitting strategies. Synthetic datasets are constructed from physical simulations with controlled parameter variations, while real-world datasets are split based on selected environmental factors. #Samples: number of samples in Train/Val/Test splits.

| Type | Dataset | Variables | Split Strategy | #Samples | |
|---|---|---|---|---|---|
| *Synthetic* | **Harmonic-VM** | Target ($x$): Acceleration<br>Context ($c$): Velocity, Position | $\alpha$-based:<br>Train $[0.0, 0.2]$; Val $[0.3, 0.5]$; Test $[0.6, 1.0]$ | 3,000/<br>1,000 | 1,000/ |
| | **Harmonic-VP** | Target ($x$): Acceleration<br>Context ($c$): Velocity, Position | Combination-based:<br>Train: $\alpha \in [0.0, 0.2]$, $\beta \in [0.0, 0.01]$, $\eta \in [0.002, 0.08]$<br>Val: $\alpha \in [0.3, 0.5]$, $\beta \in [0.018, 0.022]$, $\eta \in [0.18, 0.22]$<br>Test: $\alpha \in [0.6, 1.0]$, $\beta \in [0.035, 0.04]$, $\eta \in [0.42, 0.5]$ | 3,000/<br>1,000 | 1,000/ |
| *Real-world* | **Air Quality** | Target ($x$): $PM_{2.5}$<br>Context ($c$): TEMP, PRES, DEWP, WSPM, RAIN, wd | Station-based:<br>Train (Dongsi, Guanyuan, Tiantan, Wanshouxigong, Aotizhongxin, Nongzhanguan, Wanliu, Gucheng);<br>Val (Changping, Dingling); Test (Shunyi, Huairou) | 11,664/<br>2,916 | 2,916/ |
| | **Traffic** | Target ($x$): traffic_volume<br>Context ($c$): rain_1h, snow_1h, clouds_all, weather_main, holiday | Window-mean temperature:<br>Train $[-50, 12)$°C; Val $[12, 22)$°C; Test $[22, 50)$°C | 26,477/<br>5,578 | 16,054/ |

**Discussion.** In our dataset construction, the contextual variables $C$ are represented by the instantaneous position $x(t)$ and velocity $v(t)$ derived from the simulated trajectories. Although in many real-world conditional time series generation tasks $C$ corresponds to exogenous variables (e.g., weather, holiday), here $C$ is an intermediate state variable determined jointly by the environment-level parameters $E$ and the system's dynamics. The target series $X$ is defined as the acceleration sequence $a(t)$, which, according to the governing second-order differential equation, is uniquely determined by the current $C$ and $E$. This design ensures a clear causal direction $E, C \to X$ without instantaneous feedback from $X$ to $C$, thereby creating a controlled setting where $E$ acts as a latent confounder influencing both $C$ and $X$. To further illustrate this effect, Figure 8d shows t-SNE visualizations of two datasets, where data from different environments form clearly separated clusters, confirming that the constructed environments induce distinct joint distributions of $C$ and $X$. Such a setup faithfully instantiates the causal graph in Figure 2b and offers the following advantages:

- **Clean causal structure.** By defining $X$ as acceleration, the instantaneous feedback loop from $X$ to $C$ is eliminated, simplifying the SCM and aligning it with physical laws.

- **Controlled confounding.** The environment-level parameters $E$ influence both $C$ and $X$, enabling systematic evaluation of methods under known and controllable confounding.
- **Counterfactual tractability.** Because $X$ is fully determined by $C$ and $E$, counterfactual trajectories can be generated by intervening on either $C$ or $E$ with clear physical interpretation.
- **Physical interpretability.** The mapping from $(E, C)$ to $X$ directly follows from the second-order differential equation, ensuring that generated data adhere to physically consistent dynamics.

### E.2 Real World Dataset Description

We further evaluate our framework on two real-world datasets, i.e., **Air Quality** and **Traffic**. Across datasets, we construct *environment splits* by using selected observable variables as proxies to delineate latent contexts (i.e., treated as unobserved environments). These variables are used *only* for splitting Train/Val/Test, and the environment bank remains latent and learnable during training and inference.

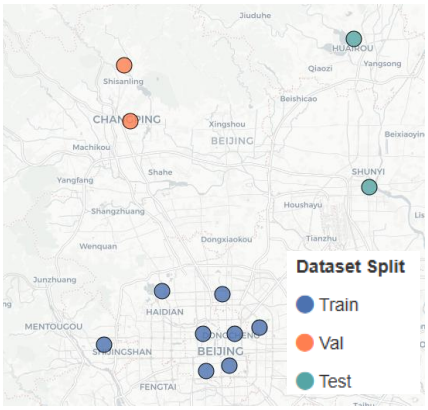



Figure 9: Geographic distribution of stations by split in the Air Quality dataset.

**Air Quality** dataset (Chen, 2019) is a comprehensive environmental monitoring dataset containing four years of hourly air quality and meteorological measurements from 12 monitoring stations across Beijing, China. The time period is from March 1st, 2013 to February 28th, 2017. We set the target variable as $\mathrm{PM}_{2.5}$ concentration, while contextual variables include temperature (TEMP), pressure (PRES), dew point (DEWP), wind speed (WSPM), rainfall (RAIN), and wind direction (wd). We form 96-hour sliding windows with a 24-hour stride, yielding 11,664/2,916/2,916 train/validation/test windows. Continuous station-level channels (PM2.5, PM10, SO2, NO2, CO, O3, TEMP, PRES, DEWP, RAIN, and WSPM) are linearly interpolated within station; wind direction is filled by the station-wise mode. After cleaning TEMP, RAIN, and WSPM, rows retaining at least 50% missing values are removed, and any residual normalized numerical missing values are represented by $-1$. We use station clusters (central vs. peripheral, derived from station coordinates) as observable proxies for latent environments to define the Train/Val/Test splits. Details are provided in Table 6, and the geographic distribution of selected stations is shown in Figure 9.

**Traffic** dataset (Hogue, 2019) is an hourly traffic dataset collected on the westbound I-94 Interstate Highway between Minneapolis and St. Paul, MN, USA. It contains 48,204 instances with traffic volume as the prediction target, and contextual variables including rainfall (rain_1h), snowfall (snow_1h), cloud coverage (clouds_all), main weather type (weather_main), and holiday indicators. We form 96-hour sliding windows with a stride of 1 hour over the full hourly series, and then assign each window to a split by its mean temperature, using temperature regimes as proxies of latent environments (details in Table 6). Temperature is excluded from $\mathbf{c}$. It defines the splits and is also used to select factual samples in the Traffic case study. Because the windows are formed before the split is applied, windows in different splits can contain the same raw hours, while no identical $(\mathbf{x}, \mathbf{c})$ window appears in more than one split. The target StandardScaler is fit on all constructed windows before the split and applied to all splits. The unit of evaluation is therefore the window, and the protocol is a temperature-stratified environment-regime evaluation rather than a chronological extrapolation over disjoint time points.

### E.3 Empirical Evidence of Spurious Correlations

To diagnose spurious correlations, we examine the stability of the conditional law $P(X \mid C)$ across environmental regimes. We treat the dataset splits $\{\mathrm{train}, \mathrm{val}, \mathrm{test}\}$ as proxies for environments; the splitting method is reported above and summarized in Table 6. We proceed in two steps:

- **Local conditional expectation (LOWESS).** For each contextual variable $C_j$ and each split $s \in \{\text{train}, \text{val}, \text{test}\}$, we fit a univariate locally weighted regression of $X$ on $C_j$ using only data from $s$. This provides an estimate of the conditional mean $\mu_s^{(j)}(c) \equiv \mathbb{E}_s[X \mid C_j = c]$, where all other components of $C$ are implicitly averaged out under their empirical distribution within $s$. We then plot the three fitted curves $\{\mu_s^{(j)}(c)\}$ to assess whether the $X$–$C_j$ relationship is stable across environments.

- **Binned conditional distribution.** For each $C_j$, we first define bin edges on the training split. Within each bin and each split, we compute the conditional mean (and variance) of $X$, and compare these per bin estimates across splits. For each bin $b$ and split $s$, we compute the sample mean $\mu_s^{(j)}(b) \equiv \mathbb{E}_s[X \mid C_j \in b]$ (bar height) and its standard error (error bar). Thus, each bar represents the average target value for samples whose $C_j$ falls in that interval. Large differences across splits for the same bin indicate potentially spurious correlation.

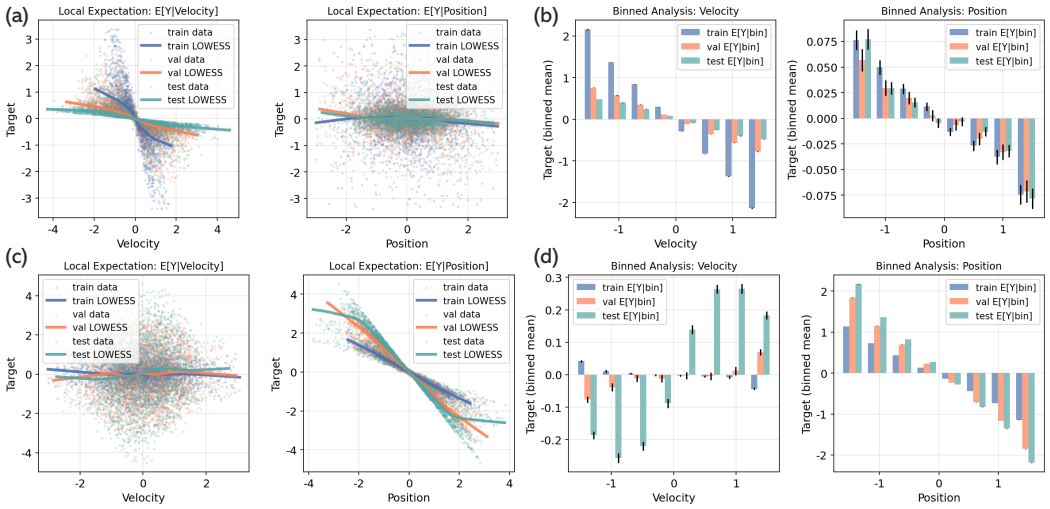



Figure 10: Empirical diagnostics on synthetic datasets. (a,b) Harmonic-VM and (c,d) Harmonic-VP with $C \in \{\text{position}, \text{velocity}\}$. (a,c) Local conditional expectation $E[X \mid C]$ with LOWESS for train, validation, and test. (b,d) Binned conditional means with standard-error bars for the same splits.

**Synthetic datasets.** Figure 10 contrasts Harmonic-VM *(a,b)* and Harmonic-VP *(c,d)*. Under our SCM, $X = a(t) = -(\gamma(t)/m(t))\, v(t) - \big(k(t)/m(t)\big)\, x(t)$, hence the conditional relation with velocity is governed by $\gamma/m$ while the relation with position is dominated by $k/m$. In Harmonic-VM, the LOWESS curves for $E[X \mid v]$ *(a, left)* separate by split, whereas $E[X \mid x]$ *(a, right)* remains close to a shared linear trend. The binned means *(b)* confirm this: the velocity–target association changes sign and magnitude across bins and splits, while the position–target association is comparatively stable. In Harmonic-VP the divergence is amplified *(c,d)* because both $\gamma(t)$ and $k(t)$ vary, leading to stronger environment dependence in $E[X \mid v]$ and mild level or slope shifts in $E[X \mid x]$.

**Real-world datasets.** For the Air Quality dataset (Figure 11a and 11b), the conditional relation with temperature shows moderate between–split shifts, and wind direction exhibits pronounced nonmonotonic drift, both evident in the LOWESS curves and confirmed by the binned means. For the Traffic dataset (Figure 11c and 11d), the association with cloud coverage shows mild shifts, whereas the relation with hourly rainfall is more strongly nonstationary across splits. Note that the missing bins in Figure 11d arise because `rain_1h` is zero-inflated with a heavy-tailed positive part; under equal-width binning many mid-range intervals contain no observations and are omitted, producing the visible gap.

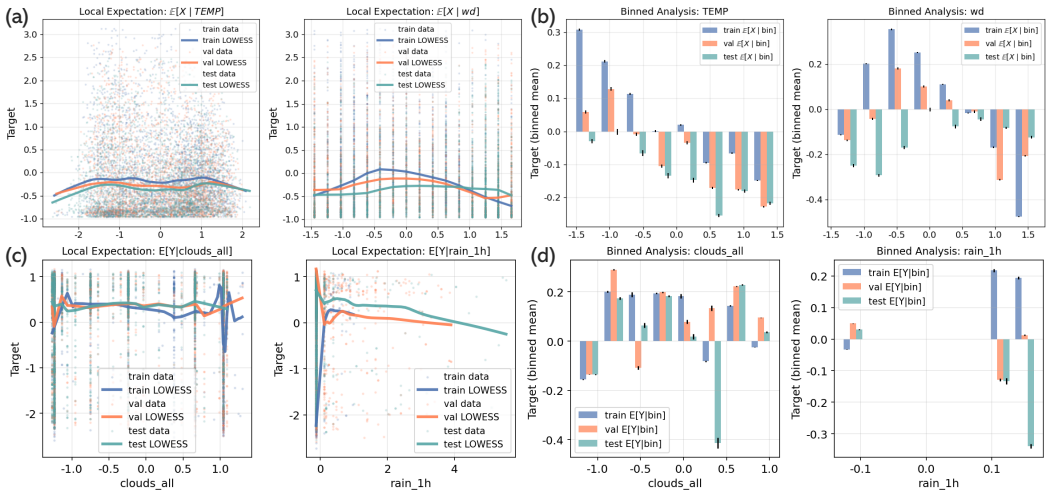



Figure 11: Empirical diagnostics on real-world datasets. (a) and (b) Air Quality with $C \in \{\text{TEMP}, \text{wd}\}$, (c) and (d) Traffic with $C \in \{\text{clouds_all}, \text{rain_1h}\}$. (a,c) Local conditional expectation $E[X \mid C]$ with LOWESS for train, validation, and test. (b,d) Binned conditional means with standard-error bars for the same splits.

# F More Details on Experiments

## F.1 Baselines

We compare our method against five representative baselines for conditional time series generation. Their hyperparameters are selected on the validation split of each dataset under the same tuning protocol used for **CaTSG**, and the configurations listed below are the selected ones.

- **TimeGAN** (Yoon et al., 2019): A generative adversarial framework tailored for time series with embedding and recovery modules. Its architecture consists of an *embedder*, *recovery*, *generator*, and *discriminator*, enabling both autoencoding and adversarial learning. The training follows three stages: autoencoder pretraining, supervised alignment of latent trajectories, and joint adversarial optimization. We use hidden dimension 64, noise dimension 32, 3 layers, and train with learning rate $10^{-4}$.
- **WaveGAN** (Donahue et al., 2019): A convolutional GAN originally designed for audio, adapted for time series via 1D convolutions. The generator is composed of stacked 1D transposed convolutional layers (512, 256, 128, output channels), while the discriminator mirrors this with strided convolutions and global pooling. Conditional information is injected through a shared ConditionalEncoder. We set hidden dimension 64, 4 convolutional layers, and train with learning rate $10^{-4}$.
- **Pulse2Pulse** (Thambawita et al., 2021): A conditional UNet-based GAN for time series. The generator is a 1D UNet with four downsampling blocks, a bottleneck, and four upsampling blocks with skip connections, while the discriminator is a CNN with global pooling. Training combines adversarial loss with an $L_1$ reconstruction loss to enforce fidelity. We set hidden dimensions 64, $\lambda_{L1} = 10$, and learning rate $10^{-4}$.
- **TimeWeaver-CSDI** (Narasimhan et al., 2024): A score-based diffusion model adapted for conditional time series generation. It employs a 1D UNet backbone with residual blocks (64–512 hidden channels) for noise prediction, conditioned on processed context embeddings. Training follows DDPM with $T = 1000$ diffusion steps. For the reported evaluations, sampling uses the same DPM-Solver protocol as **CaTSG**. We use hidden dimensions 64 and learning rate $10^{-4}$.
- **TimeWeaver-SSSD** (Narasimhan et al., 2024): A diffusion-based model that replaces the UNet with a structured state space (S4) backbone to capture long-range dependencies. The architecture stacks four S4 layers with state dimension 64 and dropout 0.1, conditioned via the ConditionalEncoder. Training follows score matching under continuous-time diffusion. We set $d_{\text{model}} = 64$, $d_{\text{state}} = 64$, 4

S4 layers, and learning rate $10^{-4}$. For the reported evaluations, sampling uses the same DPM-Solver protocol as **CaTSG**.

### F.2 Evaluation Metrics

**Marginal Distribution Distance (MDD).** This metric measures the discrepancy between the marginal distributions of generated and real sequences. Concretely, we compute the distance between empirical marginals $p_{\text{gen}}(x)$ and $p_{\text{real}}(x)$ over each variable, then average across variables to assess distributional fidelity.

**Kullback–Leibler divergence (KL).** The KL divergence quantifies how much the generated distribution $p_{\text{gen}}$ diverges from the real distribution $p_{\text{real}}$, defined as

$$\mathrm{KL}(p_{\text{real}} \,\|\, p_{\text{gen}}) = \sum_x p_{\text{real}}(x) \log \frac{p_{\text{real}}(x)}{p_{\text{gen}}(x)}.$$

Lower KL values indicate closer alignment between generated and real samples.

**Maximum Mean Discrepancy (MMD).** MMD is a kernel-based distance that compares two distributions by embedding them into a reproducing kernel Hilbert space (RKHS). It is defined as the squared distance between mean embeddings:

$$\mathrm{MMD}^2(p_{\text{real}}, p_{\text{gen}}) = \Big\| \mathbb{E}_{x\sim p_{\text{real}}}[\,\phi(x)\,] - \mathbb{E}_{x\sim p_{\text{gen}}}[\,\phi(x)\,] \Big\|^2,$$

where $\phi(\cdot)$ is the kernel feature map. Smaller MMD indicates greater similarity between the two distributions.

**Joint Frechet Time Series Distance (J-FTSD).** To evaluate the quality of generated time series and metadata jointly, we adopt the Joint Frechet Time Series Distance (J-FTSD) proposed by Narasimhan et al. (2024). J-FTSD compares the joint distributions of real and generated samples in a learned embedding space. Specifically, the time series $x_i$ and the conditional data $c_i$ are mapped into embeddings via $\phi_{\text{time}}(\cdot)$ and $\phi_{\text{meta}}(\cdot)$, which are then concatenated to form a joint representation $z_i$. Let $\mu_d$ and $\Sigma_d$ ($d \in \{r, g\}$) denote the mean and covariance of the embeddings for real and generated data respectively. J-FTSD is defined as the Fréchet distance between the two Gaussian approximations:

$$\text{J-FTSD}(D_g, D_r) = \|\mu_r - \mu_g\|^2 + \mathrm{Tr}\Big(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}\Big). \tag{40}$$

This metric thus quantifies the similarity between real and generated joint distributions of time series and metadata in the embedding space.

*Note:* In practice, we observed instability during training when learning the joint embedding space. To address this, we applied $\ell_2$ normalization to the embedding vectors during training, which stabilizes the contrastive learning objective. At inference time, however, we compute the final J-FTSD using the raw (unnormalized) embeddings. This design choice is motivated by two considerations: (1) normalization during training improves optimization stability, while (2) avoiding normalization at inference preserves the original scale of embeddings, which is crucial for reliable Fréchet-based distance estimation.

### F.3 Implementation Detail

**Compute resources.** All experiments were conducted on a single NVIDIA A100 GPU. Each training run takes approximately 1 hour 26 minutes; inference using the second-order DPM-Solver with 20 steps completes in under a minute per generation batch. Including ablations and seed averaging across all four datasets, the total compute for the reported experiments is on the order of 100–200 GPU-hours.

**Selection of $K$.** In the proposed method, there is a hyperparameter $K$, which is the number of latent environments. For both synthetic and real-world datasets, we define "environments" splits by using selected variables to delineate latent contexts that differ across splits (see Table 6 for details). These variables are used only to define the splits; the environments themselves remain unobserved (latent). Because the environment bank is latent, we do not tie the number of environments to the number of split rules (e.g., fix $K = 3$). Instead, we treat $K$ as a dataset-specific hyperparameter. This flexibility allows the model to represent more complex scenarios (e.g., submodes within a split) and to capture richer latent environment information. We also conducted the experiments under different $K$ in Section 5.5. Since both $K$ and $H$ control the capacity of the representation of $E$, if multiple unobserved confounders are expected, we recommend increasing both $K$ and $H$ to allow the environment bank to capture combinatorial variation.

**Preparation of c.** To ensure a consistent $\mathbf{c}$ representation and handle different types contextual variables especially for real world dataset, we preprocess $\mathbf{c}$ as follows. Categorical attributes (e.g., weather type) are embedded into dense vectors using learnable embeddings. We then augment the sequence with a time-of-day phase encoding by adding sinusoidal features $\sin(2\pi t/T)$ and $\cos(2\pi t/T)$ for each time step $t$, which capture periodic temporal patterns. Finally, the resulting tensor concatenated with $\mathbf{e}_k$ for input to the denoising network.

**Dataset-specific settings of `CaTSG`.** We evaluate **`CaTSG`** on four datasets (Harmonic-VM, Harmonic-VP, Air Quality and Traffic). For all these datasets, we set $p_{\text{drop}} = 20\%$ for training, and we use the DPM-Solver sampler (Lu et al., 2022) for generation, enabling the single-step variant with second order. For evaluation, we use five independent training runs per method with random seeds fixed to {42, 123, 456, 789, 999}, as stated in Section 5.1. Harmonic-VM and Harmonic-VP are synthetic datasets and we use the same setting. In every case the warm-up phase optimizes the swapped-prediction and orthogonality terms, i.e., $\mathcal{L}_{\text{sw}}$ and $\mathcal{L}_{\text{orth}}$. The settings below differ in the warm-up length and in which losses the normal phase optimizes.

- **Harmonic-VM & Harmonic-VP.** For both synthetic datasets, we set the batch size to 512, the hidden dimension to 64, and the base learning rate to $1 \times 10^{-3}$. We set the number of environments $K$ to 4. We warm up the model for 200 optimization steps, and in the normal phase we optimize the swapped, MSE, and orthogonality losses, i.e., $\mathcal{L}_{\text{sw}}$, $\mathcal{L}_{\text{orth}}$ and $\mathcal{L}_{\text{eps}}$ .
- **Air Quality.** For the air quality dataset, we set the batch size to 256, the hidden dimension to 32, and the base learning rate to $1 \times 10^{-3}$. We set the number of environments $K$ to 4. We use a short warm-up of 50 steps and, in the normal training phase, we optimize the MSE and orthogonality losses, i.e., $\mathcal{L}_{\text{orth}}$ and $\mathcal{L}_{\text{eps}}$ .
- **Traffic.** For the traffic dataset, we set the batch size to 256, the hidden dimension to 32, and the base learning rate to $1 \times 10^{-3}$. We set the number of environments $K$ to 4. We use a 50-step warm-up and optimize the MSE and orthogonality losses during the normal phase, i.e., $\mathcal{L}_{\text{orth}}$ and $\mathcal{L}_{\text{eps}}$ .

All main synthetic and real-world experiments use $K = 4$; $K = 8$ is used only for visualization and interpretability analysis.

**Ablation study experimental setup.** We compare four **`CaTSG`** variants to assess the roles of the model components and training stages. The configuration of all variants is summarized in Table 7.

# G More Experimental Results

## G.1 Direct Evaluation against the Interventional Target

The metrics of Table 2 compare generated samples with held-out observational data. This section instead compares them with $P(X \mid \text{do}(C))$ itself, on Harmonic-VM, where the structural equations are known and the interventional target can therefore be constructed directly.

**Construction of the target.** For each held-out context $\mathbf{c}$, we implement the intervention by replacing the structural mechanism for $C$ with $C := \mathbf{c}$, sampling the environment independently from the target marginal $P(E)$, and evaluating the structural equation for $X$ given in Appendix E.1. This

Table 7: Ablation settings details. $w_k$ denotes environment mixture weights.

| Mode | Env. Bank $E$ | Weights $w_k$ | $\mathcal{L}_{\mathrm{sw}}$ | $\mathcal{L}_{\mathrm{orth}}$ | Detail |
|---|---|---|---|---|---|
| **CaTSG** | Learnable | Predicted | ✓ | ✓ | Full model. |
| **w/o SW** | Learnable | Predicted | ✗ | ✓ | Omit the swapped-prediction loss and skip the warm-up stage. The environment bank is retained, but since the mixture weights are stop-gradient, EnvInfer receives no training gradient in this variant. |
| **RandEnv** | Learnable | Random | ✓ | ✓ | Randomize $w_k$. |
| **FrozenEnv** | Frozen | Predicted | ✓ | ✗ | Freeze the environment bank (no learning). |
| **w/o Env** | None | None | ✗ | ✗ | Remove the environment bank. |

is a fixed-context structural response under the sequence-level SCM, not a trajectory obtained by re-integrating the oscillator with the new environment. This differs from conditioning on $C = \mathbf{c}$, which would retain the observational association between $C$ and $E$. Because $\mathbf{c}$ is held fixed while $E$ is drawn independently, the resulting context-environment pairs need not carry observational support, so this comparison probes controlled interventional extrapolation under the known SCM rather than identification from observational data. We use $n_{\mathrm{mc}} = 100$ environment draws per context. As a consistency check on the factual trajectories, the known structural equation allows $\alpha$ to be recovered from the stored trajectories, and the reconstructed acceleration matches the data to a mean squared error of $2.9 \times 10^{-15}$ over $1{,}000$ trajectories.

**Choice of $P(E)$.** Backdoor adjustment marginalizes over the environment distribution of the population the query concerns. Our datasets assign dominant, non-overlapping parameter intervals to each split and draw each sample from an 80-20 mixture of its own interval and the others (Appendix E.1), so we take that mixture as the primary choice. Because this is a modelling decision rather than an observed quantity, we repeat the evaluation under three further choices: the full parameter range, the test interval alone, and an equal mixture over the three splits.

**Metrics.** Alongside the three distributional metrics of Section F.2, we report Cond-MSE, which compares the two conditional means at each held-out context:

$$\text{Cond-MSE} = \frac{1}{NT} \sum\nolimits_{i=1}^{N} \sum\nolimits_{t=1}^{T} \Big( \frac{1}{n_{\mathrm{mc}}} \sum\nolimits_{j=1}^{n_{\mathrm{mc}}} y_{i,j,t}^{\mathrm{gt}} - \frac{1}{R} \sum\nolimits_{r=1}^{R} y_{i,r,t}^{\mathrm{gen}} \Big)^2, \tag{41}$$

where $N$ is the number of held-out contexts, $T$ the sequence length, $y_{i,j,t}^{\mathrm{gt}}$ the constructed target under the $j$-th environment draw, and $y_{i,r,t}^{\mathrm{gen}}$ the $r$-th sampling repeat of the method under evaluation, with $R = 5$ repeats per method. Note that the three distributional metrics are fed differently, pooling the individual environment draws and sampling repeats as separate samples rather than averaging them, with at most $10{,}000$ pooled sequences per side. They therefore compare distributions, while Cond-MSE compares conditional means.

**Result.** **CaTSG** obtains the lowest distance or error in all 16 metric–environment-marginal combinations on Harmonic-VM (Table 8). The comparison is end to end: it compares the generated distribution against a directly constructed interventional target, and so it does not separate the contributions of the learned weights and the environment-conditioned branches, nor test whether $q_\phi$ recovers the target responsibility $r_{e,t}$ (Appendix C.2).

**Guidance setting.** The **CaTSG** rows of Table 8, both **CaTSG** and the **CaTSG** w/o Env variant, are generated at $\omega = 0$, the unguided setting that Eq. 4 states, rather than at the $\omega = 0.2$ used for the **CaTSG** samples of Table 2. TW-CSDI and TW-SSSD are evaluated without classifier-free guidance, and $\omega$ is a **CaTSG**-specific parameter that does not apply to these baselines. Table 9 compares **CaTSG** at $\omega = 0$ and $0.2$ against the same targets and metrics, from the same trained checkpoint and the same five matched generation seeds, so that $\omega$ is the only quantity that changes. The baselines are not regenerated, so their Table 8 numbers carry over unchanged. Which setting wins depends on the environment marginal: in the pooled five-repeat values, $\omega = 0$ is better on all four metrics under the primary and test-interval marginals, and $\omega = 0.2$ is better on all four under the full-range and equal-mixture marginals. One ordering depends on the aggregation, namely Cond-MSE under the full parameter range: the pooled value favours $\omega = 0.2$, whereas averaging the per-seed differences over the five matched generation seeds favours $\omega = 0$, with all five seeds in that direction. Pooling

Table 8: Direct evaluation against the interventional target on Harmonic-VM. Rows are methods and blocks are the environment marginal $P(E)$ used to construct the target: the 80-20 split mixture used for the primary result, then the full parameter range, the test interval alone, and an equal mixture over the three splits. Lower is better for all four metrics, with the best value in **bold** and the second best underlined. Metrics are computed from the available generated samples under a fixed trained model. Both **CaTSG** rows, **CaTSG** and **CaTSG** w/o Env, are generated at $\omega = 0$, the unguided setting of Eq. 4; Table 9 compares **CaTSG** at $\omega = 0$ and $0.2$ against the same targets and metrics. No across-training-run uncertainty is reported.

| Method | MDD ↓ | KL ↓ | MMD ↓ | Cond-MSE ↓ |
|---|---|---|---|---|
| *80-20 split mixture (primary)* | | | | |
| **CaTSG** | **0.0059** | **0.0102** | **0.0055** | **0.1275** |
| TimeGAN | 0.0988 | 4.5358 | 1.0762 | 1.8667 |
| WaveGAN | 0.0506 | 3.6138 | 0.5245 | 0.9338 |
| Pulse2Pulse | 0.0344 | 3.3639 | 0.2218 | 0.6454 |
| TW-CSDI | 0.0326 | 0.3624 | 0.1042 | 0.7898 |
| TW-SSSD | 0.0361 | 0.4297 | 0.1311 | 0.7405 |
| **CaTSG** w/o Env | 0.0168 | 0.2424 | 0.1085 | 0.6181 |
| *Full parameter range* | | | | |
| **CaTSG** | **0.0117** | **0.0571** | **0.0159** | **0.2834** |
| TimeGAN | 0.0832 | 6.1525 | 1.0170 | 2.2667 |
| WaveGAN | 0.0444 | 4.4787 | 0.5538 | 1.3320 |
| Pulse2Pulse | 0.0356 | 6.0443 | 0.2992 | 1.0446 |
| TW-CSDI | 0.0233 | 0.2921 | 0.0697 | 1.1784 |
| TW-SSSD | 0.0260 | 0.3282 | 0.0850 | 1.0915 |
| **CaTSG** w/o Env | 0.0236 | 0.5112 | 0.1666 | 0.9968 |
| *Test interval only* | | | | |
| **CaTSG** | **0.1684** | **0.0287** | **0.0122** | **0.1018** |
| TimeGAN | 0.7147 | 5.2925 | 1.1291 | 1.6721 |
| WaveGAN | 0.9919 | 3.1550 | 0.5196 | 0.7401 |
| Pulse2Pulse | 0.6504 | 3.2496 | 0.1843 | 0.4511 |
| TW-CSDI | 0.4562 | 0.4607 | 0.1400 | 0.6038 |
| TW-SSSD | 0.4758 | 0.5516 | 0.1746 | 0.5819 |
| **CaTSG** w/o Env | 0.2171 | 0.0903 | 0.0835 | 0.4399 |
| *Equal mixture over splits* | | | | |
| **CaTSG** | **0.0150** | **0.1262** | **0.0297** | **0.4920** |
| TimeGAN | 0.0734 | 5.7250 | 0.9898 | 2.6309 |
| WaveGAN | 0.0467 | 6.7814 | 0.5746 | 1.6936 |
| Pulse2Pulse | 0.0391 | 6.0708 | 0.3451 | 1.4055 |
| TW-CSDI | 0.0183 | 0.4004 | 0.0562 | 1.5312 |
| TW-SSSD | 0.0203 | 0.4196 | 0.0653 | 1.4169 |
| **CaTSG** w/o Env | 0.0268 | 0.8459 | 0.2036 | 1.3406 |

the repeats before computing the metric and averaging per-seed differences are two aggregation conventions, and the other fifteen combinations agree under both. Holding the baseline numbers of Table 8 fixed, the $\omega = 0.2$ samples still attain the best value in 15 of the 16 metric–environment-marginal combinations, the exception being MDD under the test-interval marginal. That count is a descriptive comparison against those baseline numbers under one checkpoint, not a robustness claim across training runs.

### G.2 Environmental Interpretability

In Section 5.4, we analyzed the distribution and embedding structure of the learned environments on the Harmonic-VM dataset with $K = 8$. Here, we provide additional visualizations on the Harmonic-VM and Air Quality datasets with $K = 4$, as shown in Figure 12. These results further illustrate the consistency of the learned mixture weights across splits and the clustering patterns around the learned embeddings.

Table 9: Guidance sensitivity of **CaTSG** against the directly constructed interventional target on Harmonic-VM, under the four environment marginals $P(E)$ of Table 8. The $\omega = 0$ rows are the **CaTSG** rows of Table 8. Both settings share one trained checkpoint and the same five matched generation seeds, pooled as in Table 8, so the comparison isolates $\omega$ and carries no across-training-run uncertainty. Lower is better, with the better of the two settings in **bold**.

| $P(E)$ | $\omega$ | MDD ↓ | KL ↓ | MMD ↓ | Cond-MSE ↓ |
|---|---|---|---|---|---|
| 80-20 split mixture (primary) | 0 | **0.0059** | **0.0102** | **0.0055** | **0.1275** |
| | 0.2 | 0.0097 | 0.0243 | 0.0168 | 0.1610 |
| Full parameter range | 0 | 0.0117 | 0.0571 | 0.0159 | 0.2834 |
| | 0.2 | **0.0089** | **0.0298** | **0.0117** | **0.2737** |
| Test interval only | 0 | **0.1684** | **0.0287** | **0.0122** | **0.1018** |
| | 0.2 | 0.2510 | 0.0704 | 0.0335 | 0.1655 |
| Equal mixture over splits | 0 | 0.0150 | 0.1262 | 0.0297 | 0.4920 |
| | 0.2 | **0.0123** | **0.0771** | **0.0184** | **0.4546** |

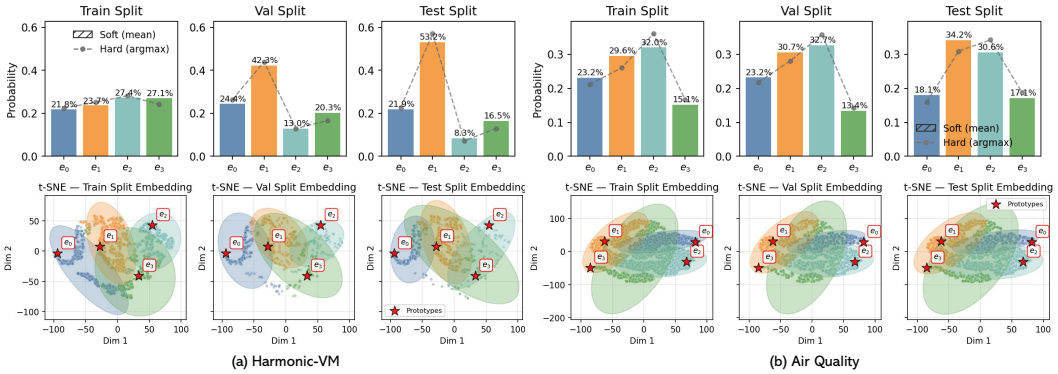



Figure 12: EnvInfer mixture weights and t-SNE of latent representations on (a) Harmonic-VM and (b) Air Quality when $K = 4$.

### G.3 Efficiency & Scalability

**Efficiency Comparison.** Table 10 compares **CaTSG** with classical baselines under matched loaders and precision. The TimeWeaver variants use the same DPM-Solver protocol as **CaTSG**. As expected, GAN-based models perform a single forward pass at inference and therefore achieve very high sampling throughput. Under this common sampler protocol, **CaTSG** reaches 144 seq/s, compared with TW-CSDI (90 seq/s) and TW-SSSD (14 seq/s). **CaTSG**'s parameter count and model size are comparable to peers. During training, CaTSG processes 16 seq/s, lower than TW-CSDI and TW-SSSD, reflecting the additional cost of environment-aware components.

Table 10: Efficiency summary. Params: parameter counts. Memory: a model-size proxy in megabytes. Thpt: throughput during training or sampling, sequences per second.

| Method | Params (M) | Memory (MB) | Training | Sampling | | |
|---|---|---|---|---|---|---|
| | | | Thpt (seq/s) | Solver | Steps | Thpt (seq/s) |
| **TimeGAN** | 10.8 | 41.1 | 45 | - | - | 2,250 |
| **WaveGAN** | 9.2 | 35.2 | 180 | - | - | 11,349 |
| **Pulse2Pulse** | 20.3 | 77.4 | 120 | - | - | 4,638 |
| **TW-CSDI** | 18.9 | 72.2 | 25 | DPM-Solver | – | 90 |
| **TW-SSSD** | 11.4 | 43.3 | 35 | DPM-Solver | – | 14 |
| **CaTSG** | 21.0 | 80.2 | 16 | DPM-Solver | 20 | 144 |

**Trade-off between speed and quality.** We vary the number of DPM-Solver steps $\{5, 10, 20, 50, 100\}$ on the Harmonic-VM dataset and report five-run runtime and accuracy in Table 11. Increasing steps markedly improves quality at roughly linear cost in runtime. From 5 to 10 steps, MDD/KL/MMD/J-FTSD drop by 47.2%/85.7%/81.7%/91.1% while runtime rises $\sim$1.8$\times$ (5.49 s$\rightarrow$9.96 s). Going to 20 steps yields a further 19.7%/51.4%/45.5%/50.7% reduction at another

$\sim$1.9$\times$ cost (9.96 s$\rightarrow$19.30 s), forming a clear knee. Beyond this point the gains are marginal: 20 to 50 steps improves by only 10.2-25.0% at $\sim$2.8$\times$ time (19.30 s$\rightarrow$54.63 s), and 50 to 100 steps yields $\leq$7.1% additional improvement with $\sim$2.0$\times$ time (54.63 s$\rightarrow$107.01 s). Overall, 20 steps offers the best speed–quality trade-off.

Table 11: Ablation on the number of DPM-Solver steps on Harmonic-VM: runtime and accuracy (mean$\pm$std over 5 runs).

| Steps | Runtime (s) | MDD | KL | MMD | J-FTSD |
|---|---|---|---|---|---|
| **5** | 5.49 $\pm$ 0.45 | 0.231 $\pm$ 0.002 | 0.244 $\pm$ 0.006 | 0.120 $\pm$ 0.003 | 12.489 $\pm$ 2.031 |
| **10** | 9.96 $\pm$ 1.39 | 0.122 $\pm$ 0.002 | 0.035 $\pm$ 0.001 | 0.022 $\pm$ 0.001 | 1.111 $\pm$ 0.168 |
| **20** | 19.30 $\pm$ 2.85 | 0.098 $\pm$ 0.001 | 0.017 $\pm$ 0.000 | 0.012 $\pm$ 0.000 | 0.548 $\pm$ 0.085 |
| **50** | 54.63 $\pm$ 2.22 | 0.088 $\pm$ 0.001 | 0.014 $\pm$ 0.000 | 0.009 $\pm$ 0.000 | 0.447 $\pm$ 0.068 |
| **100** | 107.01 $\pm$ 4.09 | 0.087 $\pm$ 0.001 | 0.013 $\pm$ 0.000 | 0.009 $\pm$ 0.000 | 0.424 $\pm$ 0.062 |

We also visualize the results in Figure 13, which shows that increasing steps improves accuracy with near-linear runtime growth, with a clear knee around 20 steps.

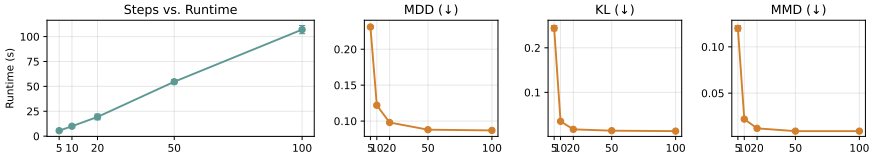



Figure 13: Speed-quality trade-off on Harmonic-VM: runtime vs. steps (*left*) and MDD/KL/MMD vs. steps (*right*).

### G.4 Uncertainty Analysis

**Confidence Interval Analysis.** We report 95% Confidence Intervals (CIs) for descriptive per-task comparisons against the best competing baseline (Table 12). CI separation is summarized in the last column, where ✓ denotes non-overlapping intervals favoring **CaTSG**, $\approx$ denotes overlapping intervals, and $\times$ denotes non-overlapping intervals favoring the baseline. Across the 16 dataset-metric comparisons, 12 have non-overlapping intervals favoring **CaTSG**, three have overlapping intervals and are inconclusive, and one has non-overlapping intervals favoring the baseline. On the synthetic datasets, Harmonic-VM shows consistent and large improvements across all metrics with non-overlapping CIs, while Harmonic-VP has non-overlapping intervals for most metrics except J-FTSD. On real-world benchmarks, non-overlapping intervals favor **CaTSG** on MDD and KL for Air Quality, and on MDD, MMD, and J-FTSD for Traffic. Overall, the largest margins occur in synthetic settings.

### G.5 Counterfactual Evaluation for Real-world Data

Real-world counterfactual ground truth is fundamentally unobservable, so the evaluation in this section is indirect and supportive rather than definitive. Unlike settings with readily inspectable targets, counterfactual quality in time series cannot be judged by inspection, and no reference trajectory exists against which a generated counterfactual can be scored. In Table 13 we summarize our evaluation of **CaTSG** and its ablation variants (RandEnv, w/o SW, FrozenEnv) under a comprehensive set of metrics. We report the standard distributional similarity metrics (MDD, KL, MMD, J-FTSD) that are also used in the main text. Because some metrics measure distributional alignment between generated samples and the underlying data distribution, while others rely on comparing generated samples with held-out observations, we report results on both the validation and test sets. The results consistently show that **CaTSG** achieves the best or near-best scores on both sets, while the ablation variants show noticeable degradation.

### G.6 Variance Analysis

As causal estimation typically involves a bias–variance trade-off, we investigate whether incorporating a latent environment variable $E$ introduces unstable variance. To this end, Figure 14 visualizes the

Table 12: Per-task comparison against the best competing baseline. Entries show the 95% CI, relative improvement of **CaTSG**, and whether CIs are non-overlapping. CI sep.: ✓ = non-overlapping (favoring **CaTSG**), ≈ = overlapping (inconclusive), × = non-overlapping (favoring the baseline).

| Dataset | Metric | Best baseline | Baseline (95% CI) | **CaTSG** (95% CI) | Rel. Δ | CI sep. |
|---|---|---|---|---|---|---|
| **Harmonic-VM** | MDD | TW-CSDI | [0.266, 0.268] | [0.097, 0.099] | 63.3% | ✓ |
| | KL | TW-CSDI | [0.202, 0.214] | [0.017, 0.017] | 91.8% | ✓ |
| | MMD | TW-CSDI | [0.080, 0.080] | [0.012, 0.012] | 85.0% | ✓ |
| | J-FTSD | TW-CSDI | [2.476, 3.088] | [0.442, 0.654] | 80.3% | ✓ |
| **Harmonic-VP** | MDD | TW-CSDI | [0.077, 0.077] | [0.052, 0.054] | 31.2% | ✓ |
| | KL | TW-CSDI | [0.201, 0.215] | [0.062, 0.070] | 68.3% | ✓ |
| | MMD | TW-SSSD | [0.084, 0.084] | [0.034, 0.034] | 59.5% | ✓ |
| | J-FTSD | TW-CSDI | [12.583, 13.645] | [9.660, 15.694] | 3.3% | ≈ |
| **Air Quality** | MDD | Pulse2Pulse | [0.081, 0.081] | [0.069, 0.069] | 14.8% | ✓ |
| | KL | TW-SSSD | [0.357, 0.357] | [0.300, 0.302] | 15.7% | ✓ |
| | MMD | TW-SSSD | [0.058, 0.060] | [0.070, 0.070] | -18.6% | × |
| | J-FTSD | TW-CSDI | [2.970, 4.544] | [2.312, 3.546] | 22.0% | ≈ |
| **Traffic** | MDD | TW-SSSD | [0.091, 0.091] | [0.071, 0.071] | 22.0% | ✓ |
| | KL | TW-CSDI | [0.154, 0.156] | [0.116, 0.228] | -11.0% | ≈ |
| | MMD | TW-SSSD | [0.038, 0.038] | [0.006, 0.006] | 84.2% | ✓ |
| | J-FTSD | TW-CSDI | [5.912, 7.216] | [0.655, 0.791] | 89.0% | ✓ |

Table 13: Preliminary evaluation results on the real-world Air Quality dataset.

| | **CaTSG** | **RandEnv** | **w/o SW** | **FrozenEnv** |
|---|---|---|---|---|
| **MDD** (val) | 0.064 ± 0.000 | 0.086 ± 0.000 | 0.085 ± 0.000 | 0.089 ± 0.000 |
| **KL** (val) | 0.290 ± 0.001 | 2.549 ± 0.002 | 2.542 ± 0.001 | 2.631 ± 0.002 |
| **MMD** (val) | 0.058 ± 0.000 | 0.091 ± 0.001 | 0.086 ± 0.001 | 0.061 ± 0.000 |
| **J-FTSD** (val) | 0.806 ± 0.608 | 1.622 ± 0.552 | 2.868 ± 2.780 | 0.953 ± 0.785 |
| **MDD** (test) | 0.064 ± 0.000 | 0.086 ± 0.000 | 0.084 ± 0.000 | 0.089 ± 0.000 |
| **KL** (test) | 0.264 ± 0.002 | 1.907 ± 0.002 | 1.893 ± 0.001 | 1.957 ± 0.001 |
| **MMD** (test) | 0.070 ± 0.000 | 0.095 ± 0.001 | 0.090 ± 0.001 | 0.069 ± 0.000 |
| **J-FTSD** (test) | 2.905 ± 0.457 | 4.900 ± 0.313 | 4.757 ± 0.606 | 3.586 ± 0.799 |

standard deviation of evaluation metrics across the five runs per method (as reported in Table 2) for each method, focusing on **KL** and **MMD**, which reflect distributional alignment.

We plot results separately for each metric across three datasets (Harmonic-VM, Harmonic-VP, Air Quality). These standard deviations are descriptive summaries of the five independent training runs per method; they are not used to establish a cross-method stability ranking. The results are consistent with the main performance comparison, while the environment bank and learned mixture weights remain subject to the same run-to-run variation as the other learned components.

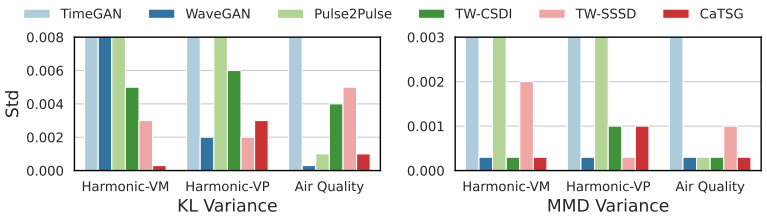



Figure 14: Standard deviation across seeds on KL and MMD metrics. Lower is better.

### G.7 Guidance Coefficient Sensitivity

To assess the robustness of classifier-free guidance in our model, we conduct a sensitivity analysis over the guidance coefficient $\omega$, which controls the trade-off between fidelity and diversity during sampling. We evaluate $\omega \in \{0.1, 0.2, 0.5, 1.0\}$ on two datasets using standard generation metrics (MDD and KL divergence). As shown in Table 14, small positive guidance weights perform best,

with the optimum varying by dataset: $\omega = 0.1$ is best on Harmonic-VM, whereas $\omega = 0.2$ is best on Air Quality, and larger values degrade both. We use $\omega = 0.2$ as the common default, including for the **CaTSG** samples of Table 2. The direct interventional comparison of Appendix G.1 instead uses the unguided setting $\omega = 0$ (Table 8), and Table 9 reports how that comparison responds to the choice.

Table 14: Sensitivity analysis of guidance coefficient $\omega$.

| $\omega$ | Harmonic-VM | | Air Quality | |
|---|---|---|---|---|
| | MDD ↓ | KL ↓ | MDD ↓ | KL ↓ |
| 0.1 | 0.077 ± 0.001 | 0.011 ± 0.001 | 0.081 ± 0.000 | 0.316 ± 0.003 |
| 0.2 | 0.098 ± 0.001 | 0.017 ± 0.000 | 0.069 ± 0.000 | 0.301 ± 0.001 |
| 0.5 | 0.169 ± 0.001 | 0.069 ± 0.002 | 0.092 ± 0.000 | 0.409 ± 0.003 |
| 1.0 | 0.243 ± 0.002 | 0.171 ± 0.001 | 0.101 ± 0.000 | 0.512 ± 0.002 |

# H More Discussion

## H.1 Causal Time Series Generation: What, Why, and How

We introduce a family of causal time series generation tasks. Below we provide a concise takeaway via three aspects: *What*, *Why*, and *How*.

**What are they?** We extend conditional TSG from observational setting to causal targets at two levels on Pearl's causal ladder (Section 4.1):

- **Observational TSG (Obs-TSG):** given observed context $C$, generate sequences $X$ from the observational distribution $P(X \mid C)$.
- **Interventional TSG (Int-TSG):** simulate how $X$ responds to setting $C$ while averaging over latent environments (i.e., altering the *data-generating mechanism* rather than just conditioning), so that the result reflects a *population-level* outcome distribution across typical environments/individuals.
- **Counterfactual TSG (CF-TSG):** given a realized $(X, C)$, ask "what would the outcome distribution have been if $C$ had been $C'$?" The target is $P(X' \mid X, C, C')$, an environment-conditioned distributional counterfactual.

Intuitively, Obs-TSG *replays* the world as it is; Int-TSG reports how the *population* responds to $C$ averaged over environments; CF-TSG reports an alternative outcome distribution under $C'$ while preserving learned factual environment information.

**Why are they needed?**

- **Avoiding confounding bias.** In many domains (traffic, air quality, energy, healthcare), $c$ and $x$ are jointly influenced by unobserved factors and thus may have spurious correlations (Appendix E.3). Reporting $P(X \mid C)$ alone risks telling the story of a particular environment/individual. Int-TSG targets $p(x \mid \mathrm{do}(c))$ and, under the assumed SCM, gives an environment-neutral response that averages over typical environments rather than any single setting.
- **Case-specific explanation.** Many decisions require a case-specific "what if". CF-TSG infers learned factual environment information from an observed target series $X$ and context $C$, changes the context from $C$ to $C'$, and generates a counterfactual sequence $X'$ while resampling residual uncertainty. For example, given a patient's vital-sign trajectory under dose $C$, CF-TSG estimates an alternative outcome distribution under dose $C'$ while holding the learned factual environment information fixed.
- **Robust generalization.** By averaging over $E$ (Int-TSG) and preserving learned factual environment information (CF-TSG), the learned generator becomes less sensitive to shifts in season, region, or load regime, improving portability and stability.
- **Unifying generation–decision–explanation.** Obs-TSG can replay data but cannot support intervention planning or counterfactual reasoning. Int/CF-TSG turn a generator into a tool that can *act* (simulate environment-neutral responses to $C$) and *explain* (case-conditioned alternatives under $C'$), closing the loop from modeling to decision support.

**How to realize them?** We adopt diffusion as the backbone, given its strong performance on time series generation (Yang et al., 2024). Our unified model, **CaTSG**, enables Obs-/Int-/CF-TSG by (1) introducing a learnable environment bank that captures latent background conditions and (2) replacing standard conditional guidance with backdoor-adjusted guidance during sampling (Eq. 6). Specifically, training follows Algorithm 1. For Int-TSG, **CaTSG** aggregates per-environment predicted noise with environment weights (Algorithm 2). For CF-TSG, it follows an Abduction-Action-Prediction routine: obtain the instance's learned factual mixture weights from $(X, C)$, set the alternative control $C'$, and predict the noise (Algorithm 3). When the environment module is disabled, **CaTSG** reduces to a standard observational generator for Obs-TSG.

### H.2 More Examples Across Domains

Table 15: Representative TSG examples under observational (Seeing), interventional (Doing), and counterfactual (Imagining) settings (Pearl, 2009) across different application domains.

| | Variable | Seeing | Doing | Imagining |
|---|---|---|---|---|
| *General* | $X$: Target time series<br>$C$: Contextual variable<br>$E$: Latent confounder influencing both $X$ and $C$ | Observe that $X$ is correlated with $C$, but this may be confounded by latent factor $E$. | Estimate how $X$ would change under an intervention on $C$, adjusting for $E$. | Given observed $(X, C)$, simulate what $X'$ would have been under alternative $C'$, keeping $E$ fixed. |
| *Healthcare* | $X$: Patient physiological indicators (e.g., temperature, heart rate)<br>$C$: Treatment (e.g., type and dosage of medication)<br>$E$: Health condition (e.g., disease severity, comorbidities) | Observe that patients who received Treatment A recover faster, but this may reflect differences in health condition. | Estimate how recovery would look under Treatment A for any patient, regardless of their health condition. | For a patient who received Treatment B and recovered slowly, generate how they would have recovered under Treatment A. |
| *Energy* | $X$: Electricity demand (hourly/daily)<br>$C$: Energy-saving policy level<br>$E$: City profile (e.g., industrial vs. residential) | Observe that stricter policies are associated with lower demand, but this may be biased by city profile. | Estimate demand if all cities are given strict policy, regardless of city-level differences. | For a city with observed demand under lenient policy, ask what the demand would have been under stricter policy. |
| *Transportation* | $X$: Traffic flow<br>$C$: Traffic Contextual strategy (e.g., signal timing, lane closures)<br>$E$: Road network structure / commuter behavior | Observe that Strategy A reduces traffic jams, but that strategy may be applied only in low-density areas. | Estimate traffic flow under Strategy A, adjusting for road and commuter profiles. | Ask what traffic would have looked like on this road if a different Contextual strategy had been applied. |
| *Education* | $X$: Test score<br>$C$: Learning aid (e.g., reminders, videos)<br>$E$: Student type (self-discipline, background knowledge) | Observe that students who got reminders learned faster (higher test scores), but self-disciplined students may opt into them more. | Estimate learning outcomes if all students received the same intervention, regardless of student type. | For a student without reminders, simulate their learning outcome if reminders had been provided. |
| *Finance* | $X$: Stock price sequence<br>$C$: Public policy signals (e.g., interest rate change)<br>$E$: Market context (e.g., economic cycle, volatility regime) | Observe that rate hikes tend to reduce stock prices, but this may be due to broader economic downturns. | Estimate how stock price sequences would evolve under a given policy, adjusting for latent market regimes. | Given observed stock prices under a no-hike policy, simulate the price trajectory that would have occurred under a rising-rate policy, assuming the same market condition. |

To further illustrate the applicability of the proposed causal time series generation tasks, i.e., Int-TSG and CF-TSG in real-world scenarios, we provide additional examples across various domains in Table 15. These complement the transportation case discussed in the Causal Ladder (Table 1). Together, these examples highlight the significance and versatility of causal time series generation across different application settings.

### H.3 Identifiability Scope

This subsection clarifies what the backdoor decomposition provides, what **CaTSG** approximates, and the scope of the resulting causal interpretation.

**What the decomposition gives.** Under assumptions (A1) and (A2) stated in Section 4.1, the interventional distribution admits the backdoor decomposition of Proposition C.1, which expresses $P(X \mid \mathrm{do}(C))$ in terms of $P(X \mid C, E)$ and $P(E)$ under the assumed SCM. Because $E$ is latent, A1-A2 alone do not imply nonparametric identification of $P(X \mid \mathrm{do}(C))$ from observed $(X, C)$ alone, and **CaTSG** therefore operationalizes the decomposition through a learned environment representation rather than through a measured adjustment set. The neural causal models do not

expand the identifiability class beyond the underlying SCM (Xia et al., 2021), so our diffusion-based instantiation inherits whatever the assumed graph and model assumptions entail, and no more.

**What is not claimed.** We do not claim identifiability of the latent confounder $E$ itself. The environment bank is a finite discrete approximation to a potentially continuous confounder space (see Appendix H.4), not an identified latent variable. The t-SNE clusters and learned mixture-weight visualizations reported in Section 5.4 should accordingly be read as supportive structural evidence, not as identifiability of $E$. By the Dual Identifiability theorem (Xia et al., 2023a), quantities that are unidentifiable in the symbolic SCM remain unidentifiable in any neural implementation; our framework cannot recover the latent confounder when the assumed SCM does not entail its identifiability.

**When the assumptions fail.** The validity of the decomposition relies on (A1)-(A2). When these are violated, e.g., due to unmodeled mediators, feedback loops $C \leftrightarrow X$, time-varying confounding, or cross-domain shifts in causal structure (see Appendix H.4 for a detailed enumeration), **CaTSG** continues to execute its backdoor-adjusted generation procedure under the *assumed* SCM. The resulting samples therefore constitute a model-based approximation whose causal interpretation is no longer guaranteed: under misspecification, **CaTSG** produces a misspecified approximation to the intended causal target, and the generated distribution may remain confounded or biased.

### H.4 Causal Assumption and Confounding Treatments

**Fixed SCM assumption.** In our **CaTSG** solution for causal time series generation, we adopt a predefined SCM (Section 4.1) to operationalize interventional and counterfactual objectives under the assumed SCM. While this SCM connects observational, interventional, and counterfactual generation under a single formulation, it also introduces several limitations: By the Dual Identifiability theorem (Xia et al., 2023a), quantities that are unidentifiable in the symbolic SCM remain unidentifiable in any neural implementation. Therefore, the validity of CaTSG's causal inferences is bounded by the correctness of the assumed SCM. First, if true mechanisms violate the assumed edges or independences, e.g., unmodeled mediators, feedback loops ($C \leftrightarrow X$), time-varying confounding, nonstationarity, or an intervention that alters the mechanism rather than only setting $C$, the backdoor set may fail to block spurious paths and counterfactual inferences may be biased. Second, we approximate confounding $E$ by a finite environment bank with $K$ embeddings, implicitly assuming that conditioning on $E$ renders $C$ ignorable for $X$. When $K$ is misspecified or $E$ cannot summarize all relevant confounding, residual bias can remain; the learned embedding labels can be permuted without changing the model, and the representation may be entangled. Third, if certain $(\mathbf{c}, \mathbf{e})$ combinations are rarely or never observed, estimating $P(X \mid \mathrm{do}(C{=}\mathbf{c}))$ may be much more challenging. Fourth, applying a fixed SCM across domains (e.g., finance or healthcare) presumes mechanism invariance and similar support; violations (shifts in causal structure, intervention ranges, or measurement processes) can degrade robustness and external validity.

Overall, the predefined SCM assumption should be regarded as a simplifying choice that enables tractable formulation and analysis of causal time series generation. Our framework thus represents only an initial step toward this new direction, i.e., causal time series generation. We anticipate that future work will develop more expressive and adaptive models, e.g., with learned or dynamic causal structures, that can support richer forms of interventional and counterfactual generation in complex real-world settings.

**Genuine and spurious relationship.** In many observational time series systems, an observed dependence combines both a genuine causal effect and a spurious component induced by external contextual factors. The edge $C \to X$ in our running example (i.e., Figure 1) reflects this mixed structure: while $C$ (e.g., environmental conditions) may exert a true influence on $X$, the dominant variation in school-neighborhood traffic is also strongly shaped by the school calendar, producing a confounded association between $C$ and $X$. The resulting dependence therefore has a mixture of structural and spurious effects. Therefore, our model addresses this issue by introducing the environment variable $E$, which models the spurious calendar-driven variation, thereby isolating the stable causal component of the $C \to X$ relationship.

**Finite environment bank.** We do not claim to identify the latent confounder $E$. The environment bank is a finite discrete approximation to a potentially continuous confounder space, designed to make backdoor-adjusted generation tractable and end-to-end trainable. As a design intuition (not a theoretical guarantee), discrete codebooks have been shown to capture useful structure in high-dimensional continuous representations in VQ-VAE (Van Den Oord et al., 2017) and SwAV (Caron et al., 2020); we use this representation-learning precedent as motivation for the present confounder approximation, not as a justification of its causal validity.

Several design choices reinforce the approximation: (1) we allow soft environment inference with probabilistic weights $w_k$, reducing the impact of hard mis-assignment; (2) we apply diversity-promoting regularization (e.g., orthogonality loss) across the environment embeddings to discourage factor entanglement; (3) the adopted swapped loss promotes environment consistency under stochastic augmentations, discouraging over-fragmentation or collapse. These strategies aim to ensure that, even with moderate $K$, the environment bank captures a diverse and functionally sufficient set of latent factors for adjustment.

For settings with continuous or high-dimensional confounders, richer representations such as continuous latent variables with normalizing flows, or hierarchical and multi-factor latent environment models, are natural extensions beyond the scope of the present work.

### H.5 Positioning & Scope

**CaTSG** is designed to target interventional and counterfactual objectives, i.e., $P(X \mid \mathrm{do}(C))$ and $P(X' \mid X, C, C')$. This is particularly useful in regimes with environment shifts, where a latent environment $E$ varies and acts as a confounder for the observational relation between $C$ and $X$ to mirrors real-world scenarios (i.e., under SCM in Figure 2b instead of Figure 2a). Our curated datasets and split strategy are constructed to highlight this setting by inducing changes in $E$. The goal is to evaluate generative targets aligned with interventions and counterfactuals, rather than to optimize observational fit under near-iid conditions. In such shifted regimes, modeling $P(X \mid \mathrm{do}(C))$ with latent $E$ yields observable gains in fidelity metrics. When $E$ is stable and fully observed, standard conditional Obs-TSG models that target $P(X \mid C)$ will still be competitive. In scenarios where $E$ is effectively constant, **CaTSG** reduces to the conditional case (a single or degenerate environment), and performance improvements are expected to be small or negligible. When $E$ is fully observed, it can be incorporated into $C$ directly, and the backdoor adjustment reduces to standard conditioning. When $E$ is partially observed, the observed portion can be included in $C$ while the environment bank captures the residual latent confounding. The framework's core mechanism remains unchanged in both cases. Typical applications of the targeted regime include policy simulation and what-if planning under seasonal or regime changes (e.g., weather patterns in traffic, meteorology in air quality), and cross-region transfer. Examples across different domains are presented in Table 15.

### H.6 Comparison with Related Works

Prior causal generative models are mostly developed for static modalities, e.g., images via label-graph–consistent GANs or modular recombinations (Kocaoglu et al., 2018; Sauer & Geiger, 2021), text via interventions on latent narrative factors (Hu & Li, 2021). Compared to prior causal generative work, which mostly targets a single rung of Pearl's ladder or non-temporal modalities, our framework formulates observational, interventional, and counterfactual generation jointly for time series within a unified diffusion-based instantiation. Methodologically, within an SCM-based formulation, we realize causality through a backdoor-adjusted score guidance that injects confounder adjustment directly into the diffusion sampling field, and we couple it with a learnable environment bank to model latent environments that drive nonstationarity. This is different from designs that either intervene on observed, single-shot factors or require acyclicity-constrained latent DAGs as in CausalVAE (Yang et al., 2021).

### H.7 Relation to Temporal Causality Literature

**Granger causality.** A long line of work studies causality in time series through Granger causality (Granger, 1969; Eichler & Didelez, 2010; Assaad et al., 2022), which is classically framed around predictive precedence: $X$ is said to "Granger-cause" $Y$ if past values of $X$ improve the prediction of

$Y$ beyond what is achievable from $Y$'s own history. By itself, this is not an interventional semantics; an interventional interpretation requires additional structural assumptions and graphical conditions, as discussed in detail by Eichler & Didelez (2010). Our work targets a complementary objective: rather than testing whether one series predicts another, we model interventional and counterfactual distributions ($P(X \mid \mathrm{do}(C))$ and $P(X' \mid X, C, C')$) under an assumed SCM, conditional on those structural assumptions being satisfied.

**Temporal SCM frameworks.** Several frameworks formalize causal structure in time series, including models linking Granger causality and interventions (Eichler & Didelez, 2010), and restricted structural equation models for time series such as TiMINo (Peters et al., 2013). These model causal relationships at time points and specific lags. **CaTSG**, in contrast, operates on *sequence-level* variables: $X$ and $C$ are entire trajectories, and the intervention $\mathrm{do}(C)$ corresponds to setting the full contextual sequence rather than to a time-indexed structural equation. As a consequence, **CaTSG** does not cover time-point-level interventions or finer-grained temporal causal structure; this is a deliberate scope limitation rather than an oversight.

**What an extension would require.** Supporting time-point-level interventions would require time-indexed structural equations, additional acyclicity or temporal-ordering constraints, and a richer SCM that explicitly represents lagged confounders and mediators. Causal graph discovery from observational time series is likewise an orthogonal direction. We view both as natural future extensions, and emphasize that **CaTSG**'s causal formulation (Appendix H.3) is confined to the sequence-level setting.

### H.8 Limitations and Future Directions

**CaTSG** assumes time series conditions and a predefined sequence-level SCM, and represents latent environments with a finite learned bank. Extending these assumptions to multimodal conditions and changing causal mechanisms is a direction for future work (Appendix H.4). On real-world data, CF-TSG holds learned factual environment weights fixed while resampling residual uncertainty, so its target is distributional rather than generally unit-level. Because our real-world datasets lack counterfactual ground truth, their counterfactual evaluations provide indirect distributional diagnostics rather than direct validation (Appendix G.5).

### H.9 Broader Impact

**CaTSG** aims to support what-if analysis and policy-relevant simulation in domains where collecting interventional data is costly or infeasible, with example scenarios discussed in Appendix H.2 and Appendix H.5. Like other generative time series models, it carries generic misuse risks (e.g., synthetic outputs being presented as factual records); a risk specific to causally guided generation is that outputs produced under a misspecified SCM may inherit biases or spurious associations from the assumed causal graph (failure modes are discussed in Appendix H.4). The released artifacts are limited to method code on public datasets and do not include pre-trained models for general public use.