A Conditional Diffusion-Based Generative AI Model for Industrial Predictive Analytics

Donghyun Ko* Xiaolei Fang†

Abstract

Accurate prediction of remaining useful life (RUL) or time-to-failure (TTF) from degradation signals is a central problem in prognostics and health management across domains such as aerospace, manufacturing, and energy. Existing approaches largely fall into two categories: model-based methods that explicitly model degradation trajectories under parametric assumptions, and data-driven methods that directly map observed signals to failure time via dimensionality reduction and regression. Representative examples of these approaches considered in this paper include Bayesian updating and PCA-based lognormal regression. While effective for smooth and monotonic degradation patterns, both paradigms often degrade when applied to irregular, nonstationary, or multimodal degradation processes commonly encountered in real-world environments. To address these limitations, we propose a conditional diffusion–based generative framework that learns the full conditional distribution of future degradation trajectories given partial observations, enabling flexible and probabilistic TTF prediction. By modeling the data distribution directly, diffusion models can approximate nonlinear, heavy-tailed, and multimodal degradation behaviors. To rigorously evaluate performance, we construct synthetic degradation datasets with controlled increases in structural complexity, incorporating Brownian-motion noise, localized perturbations, and multi-regime dynamics. Extensive benchmarks against Bayesian updating and PCA-based lognormal regression demonstrate that the proposed diffusion model consistently achieves lower error under complex settings, particularly near failure thresholds and under complex, heteroscedastic conditions. These results position diffusion model as a robust and scalable alternative for predictive analytics in realistic prognostic.

Keywords: Remaining Useful Life (RUL); Time-to-Failure (TTF); Degradation signals; Generative time-series modeling; Diffusion models; prognostics and health management (PHM).

1 Introduction

1.1 Prognostics and Health Management

Degradation is an irreversible process of damage accumulation that ultimately leads to system failure. Although degradation itself is often latent and not directly observable, it produces measurable manifestations through vibration, temperature, acoustic emission, or other sensing

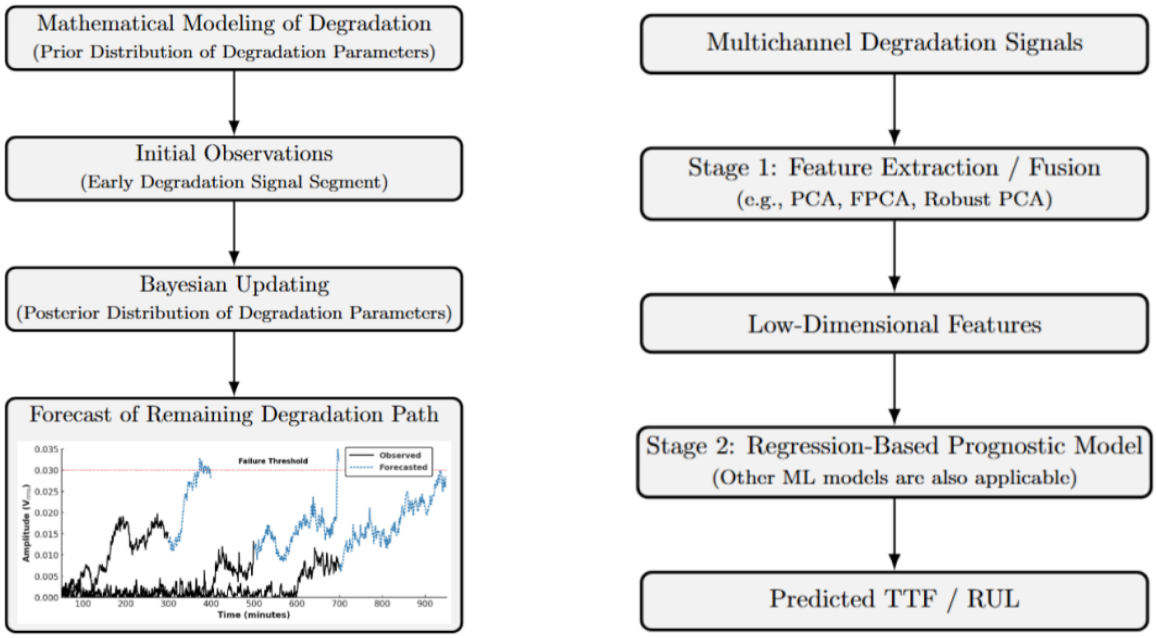
*Ph.D. Student, Edward P. Fitts Department of Industrial and Systems Engineering, North Carolina State University, Raleigh, NC, USA. This work was conducted under the supervision of Prof. Xiaolei Fang.

†Associate Professor, Edward P. Fitts Department of Industrial and Systems Engineering, North Carolina State University, Raleigh, NC, USA.

modalities, forming what are known as degradation signals. When properly modeled, these signals enable prognostic methods to estimate the time-to-failure (TTF) of engineering systems [1, 2, 3, 6]. Accurate TTF prediction is essential across manufacturing, aerospace, and energy sectors because it supports proactive maintenance scheduling, prevents unexpected shutdowns, and improves operational efficiency [4, 7, 8].

1.2 Existing work and limitations

A wide array of prognostic methods has been developed to interpret degradation signals and use them to predict the TTF, and these methods broadly fall into two categories. *Model-based* methods rely on explicit mathematical representations of the degradation process and often—particularly in Bayesian approaches—update model parameters as new observations arrive. For example, Gebraeel et al. [5] proposed a conditional Bayesian updating framework that forecasts the remaining trajectory given initial observations and declares failure when the predicted path crosses a predefined threshold (see Fig. 1a); derivations are provided in Appendices A.1 and A.2. Representative examples include exponential or Wiener-process degradation models with sequential posterior updating [6, 7, 8]. These methods are valued for their interpretability and analytical tractability. In contrast, *data-driven* methods directly models the relationship between historical signals and failure time using statistical or machine learning (ML) algorithms. For example, He et al. [9] developed a two-stage data-driven approach that first fuses multichannel degradation signals to extract low-dimensional features and then constructs a prognostic model by regressing the failure time on these features (see Fig. 1b); derivations are provided in Appendices B. Classical variants commonly employ PCA, FPCA, or robust PCA to extract low-dimensional health indicators prior to regression modeling [10, 11, 12, 13]. More recently, deep learning (DL) models have advanced the data-driven paradigm by automatically extracting latent features and capturing nonlinear dependencies without relying on externally imposed dimension reduction [14, 15, 16, 17, 18].

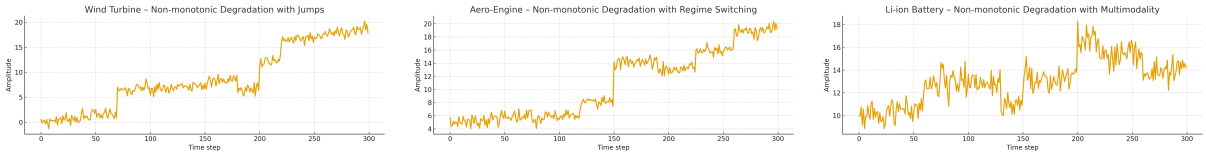


(a) Bayesian updating framework

(b) Two-stage prognostics framework

Figure 1: Examples of classical prognostic modeling approaches

Despite these advancements, both model-based and data-driven prognostic frameworks exhibit important limitations when applied to real-world degradation processes. Model-based approaches, which rely on restrictive parametric forms such as linear or exponential trends, often break down when actual degradation trajectories exhibit nonlinear drifts, abrupt jumps (Fig. 2a), discontinuities (Fig. 2b), or multimodal regime-switching behaviors (Fig. 2c) which are widely observed in lithium-ion batteries [19, 20, 21], wind turbines [22, 23, 24], and aero-engines [25, 26]. These models implicitly assume that degradation evolves smoothly (e.g., with linear or exponential trends) over time and therefore cannot accommodate sudden jumps, discontinuous signal segments, or transitions between distinct operating regimes, leading to incorrect extrapolations and unreliable TTF estimates. Classical data-driven approaches suffer from complementary limitations: they depend on low-rank linear subspace assumptions during dimensionality reduction, which distort fine-grained geometric structure in the data and lead to information loss when the underlying trajectories evolve on a curved, highly nonlinear manifold rather than a single linear subspace. As illustrated in Fig. 2, such trajectories bend, branch, and cluster in ways incompatible with any fixed linear subspace, causing PCA variants to collapse distinct regimes and obscure essential dynamical features, ultimately leading downstream ML models to miss turning points and regime changes. Such branching trajectories and regime-dependent remaining-useful-life (RUL) distributions have been reported in real-world aero-engine prognostics studies, including datasets such as C-MAPSS [25], highlighting the limitations of parametric and linear subspace-based approaches. DL methods mitigate some of these issues by learning nonlinear representations directly from raw signals, but they remain fundamentally limited: most are trained in a discriminative manner that outputs point predictions rather than full predictive distributions, and they do not explicitly model the generative process governing degradation evolution, preventing them from producing realistic future trajectories. To make these challenges explicit, we construct synthetic datasets that preserve domain-informed behaviors such as jumps, discontinuities, and multimodal regimes, thereby highlighting the need for generative, manifold-aware prognostic frameworks capable of representing complex degradation structure.



(a) Wind turbine (b) Aero-engine (c) Lithium-ion battery

Figure 2: Domain-informed synthetic degradation examples
[1]

1.3 Goals and novelty of the current work

The challenges outlined in Section 1.2 highlight the need for more flexible and expressive prognostic frameworks capable of modeling the nonlinear, multimodal, and regime-switching behavior commonly observed in real degradation signals. Recent advances in generative artificial intelligence (AI), specifically diffusion models, offer a principled mechanism for learning complex data manifolds and generating realistic samples conditioned on auxiliary information.

[1] These trajectories emulate behaviors reported in publicly available datasets for wind turbines, aero-engines, and Lithium-ion batteries showing non-monotonic drifts, jumps, and multi-modality. Since these open source datasets are limited in size and breadth for training diffusion models, we synthesize data with similar patterns.

Diffusion models [31, 33, 34] learn to reverse a gradual forward noising process defined as a Markov chain that progressively transforms data into Gaussian noise; a neural network is then trained to approximate the reverse denoising transitions, enabling probabilistic sampling by iteratively reconstructing clean signals from noise to recover intrinsic nonlinear structure in the data. This framework has recently been extended to sequential domains, too. Early generative approaches for time-series data, using VAEs [27] and GANs [28, 29], often suffer from instability and mode collapse. Diffusion models avoid these issues and have shown strong performance in temporal tasks such as time-series forecasting [30], imputation [39], and even degradation modeling [53, 54]. These methods typically employ *direct conditioning*, in which observed segments are injected into the denoising network through concatenation, cross-attention, or conditional normalization, enabling the model to learn the conditional distribution of the missing future portion of a sequence.

Following this paradigm, our task is to generate future degradation trajectories conditioned on partially observed signals and infer the distribution of the TTF for each unit. Once trained on historical degradation data, the conditional diffusion model generates a large set of plausible future continuations for any partially observed trajectory. Applying a predefined failure threshold to each generated sample yields an empirical predictive distribution of TTF rather than a single deterministic estimate (see Fig. 3). Compared with the rigidity of parametric model-based approaches and the geometric limitations of classical ML / DL methods, the generative formulation of diffusion models naturally accommodates nonlinear, multimodal, and abrupt degradation behaviors.

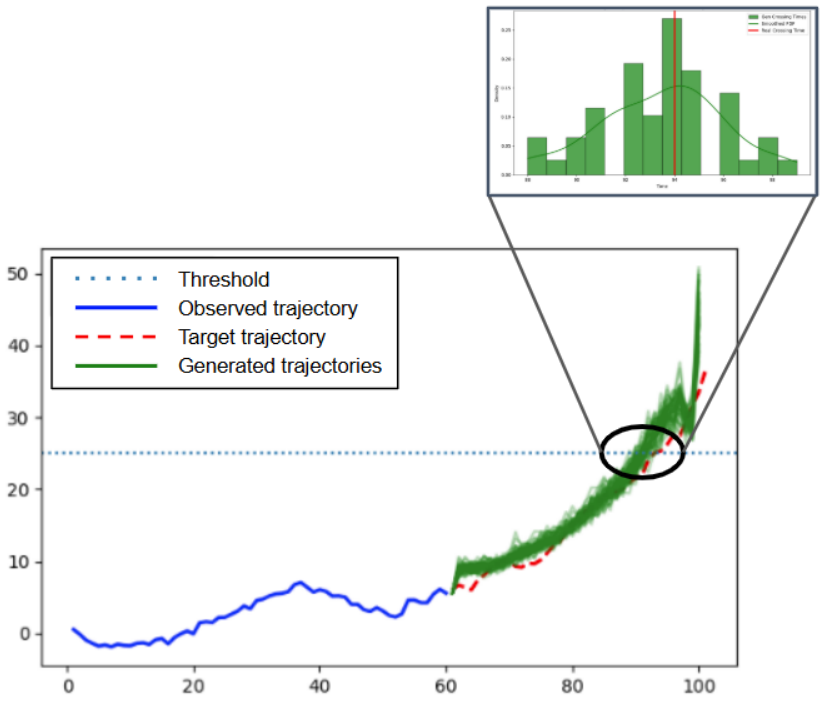


Figure 3: Generated trajectories by diffusion model with distribution of TTF

Although prior work has explored diffusion-based TTF prediction through conditional score-based generation [53], latent diffusion with uncertainty quantification [54], class-conditional generation [55], and Bayesian score-based modeling [56], existing approaches exhibit several fundamental limitations. Most notably, they directly apply diffusion models originally developed for image synthesis or generic time-series generation to degradation-driven TTF forecast-

ing without considering their underlying modeling assumptions. Consequently, (i) evaluations are often restricted to benchmark datasets exhibiting relatively smooth and near-monotonic degradation trajectories, such as linear or exponential trends; (ii) the forward diffusion process is typically defined using i.i.d. Gaussian noise and model architectures that do not explicitly enforce time-aligned conditioning between observed prefixes and future segments, thereby limiting the representation of temporal correlation and regime-dependent dynamics; and (iii) empirical validation against classical reliability baselines is usually conducted on a limited set of degradation scenarios, without systematically controlling degradation complexity such as abrupt transitions, multimodal progression, or nonlinear manifold structure.

This study addresses these gaps and makes three key contributions by revising the diffusion framework itself for the specific demands of time-series TTF prediction. First, we conduct rigorous and controlled benchmarking of diffusion-based TTF prediction against classical *model-based approach* (e.g., Bayesian updating models) and *data-driven approach* (e.g., PCA-based lognormal regression) across increasing levels of degradation complexity, providing the first systematic performance comparison under realistic experimental settings. Second, we introduce a *Gaussian process–based forward noising mechanism* that explicitly encodes temporal correlation across degradation trajectories, replacing the i.i.d. noise assumption—a design choice that, to our knowledge, has not been explored in diffusion-based TTF prediction. Third, we develop a *time-aware conditional reverse denoising architecture* that combines a hybrid multiscale Temporal U-Net and Transformer backbone [45, 49] with cross-attention, tailored temporal embeddings based on the log signal-to-noise ratio (log-SNR), and classifier-free guidance [37]. This design enables precise, time-aligned conditioning on partially observed signals while capturing both global degradation trends and local fluctuations. Together, these innovations position the proposed framework as a principled generative alternative for TTF prediction in systems exhibiting heterogeneous operating conditions, regime shifts, and sensor-induced irregularities. The remainder of the paper is organized as follows. Section 2 introduces diffusion fundamentals and their adaptation to time series. Section 3 describes the proposed conditional diffusion architecture and training procedure. Section 4 presents the datasets, experimental design, metrics, and empirical evaluations. Section 5 discusses limitations and future directions, and Section 6 concludes the paper.

2 Preliminaries

This section reviews denoising diffusion probabilistic models (DDPM), their conditional variants, and extensions to time-series modeling for TTF prediction. We present formal definitions and their mathematical structure, while full derivations are deferred to the appendix.

2.1 Diffusion Model Fundamentals

Diffusion models [31, 33] learn to reverse a gradual forward noising process to approximate complex data distributions. Given a data sample $\mathbf{x}_0 \sim p_{\text{data}}$, the forward process defines a Markov chain $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\big(\sqrt{1-\beta_t}\,\mathbf{x}_{t-1},\, \beta_t\mathbf{I}\big)$ for $t = 1, \ldots, T$ with variance schedule $\{\beta_t\}$ such that $\mathbf{x}_T \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$. The reverse process is parameterized as $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\big(\boldsymbol{\mu}_\theta(\mathbf{x}_t, t),\, \sigma_t^2\mathbf{I}\big)$, where a neural network architecture(e.g., U-Net) learns the distribution of the noise added at each diffusion step, and the parameters θ are learned through a variational ob-

jective that reduces to a noise-prediction MSE [35]. Once trained, one can draw $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iteratively remove noise to generate realistic data samples $\mathbf{x}_0$ by applying the learned reverse process.

Forward process. Given $\mathbf{x}_0 \sim p_{\text{data}}$, where p_{data} denotes the unknown real data distribution, we define a forward (noising) process (see Figure 4) as a first-order Markov chain that gradually adds Gaussian noise over T discrete steps:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\big(\sqrt{1-\beta_t}\,\mathbf{x}_{t-1},\ \beta_t \mathbf{I}\big), \quad t = 1, \ldots, T,$$

where each $\beta_t \in (0, 1)$ is a variance schedule hyperparameter. Intuitively, the factor $\sqrt{1-\beta_t}$ shrinks the previous sample $\mathbf{x}_{t-1}$, and Gaussian noise with variance β_t is added. Equivalently, each step can be written in generative form as

$$\mathbf{x}_t = \sqrt{1-\beta_t}\,\mathbf{x}_{t-1} \ + \ \sqrt{\beta_t}\,\boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \text{ i.i.d. across } t.$$

Let $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s = \prod_{s=1}^{t}(1-\beta_s)$. A useful closed-form marginal then follows (derivation in Appendix C.1):

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0,\ (1-\bar{\alpha}_t)\,\mathbf{I}\big)\,, \qquad \mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \ \ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

which allows sampling $\mathbf{x}_t$ from $\mathbf{x}_0$ in one shot without iterating intermediate states [31, 35]. By using this, the full forward process can be factorized as

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) \ = \ \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

Typical choices for the schedule $\{\beta_t\}$ include linear [31] and cosine [34]; other learned or continuous-time parameterizations are also used in practice [36] for better quality generations.

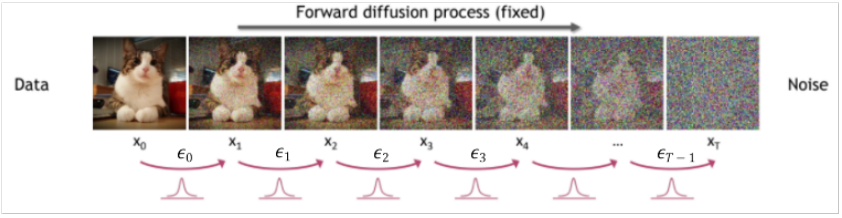


Figure 4: Forward Process

Reverse process and training objective. As T grows, x_T becomes approximately distributed as $\mathcal{N}(0, I)$ regardless of the original x_0. The reverse process(see Figure 5) aims to iteratively recover the clean sample x_0 from a fully corrupted latent variable $x_T \sim \mathcal{N}(0, I)$. The goal of the diffusion model is to learn this reverse process by approximating the intractable $q(x_{t-1} \mid x_t)$ using a neural network framework for which we parameterize the reverse process as:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I),$$

where μ_θ is learned to approximate the true posterior mean $\tilde{\mu}_t(x_t, x_0)$. Then, the generative model factorizes as a Markov chain such that:

$$p_\theta(\mathbf{x}_{0:T}) \ = \ p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$

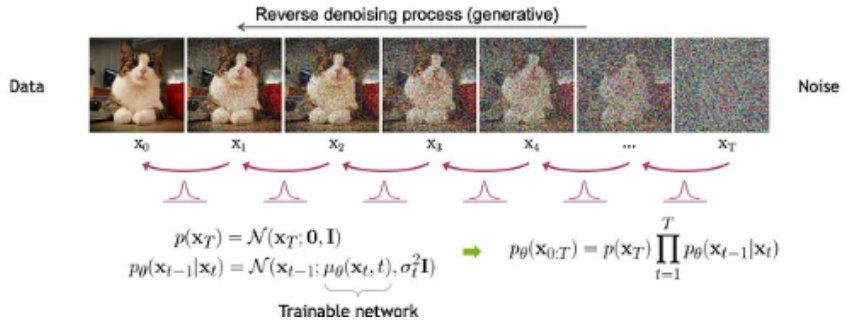


Figure 5: Reverse Process

Although our ultimate goal is to maximize the likelihood of the data $\log p_\theta(x_0)$, directly computing this quantity is intractable due to the high-dimensional integral over latent variables in the reverse process. To circumvent this, we turn to variational inference and instead maximize a tractable surrogate objective known as the Evidence Lower Bound (ELBO), which provides a principled lower bound on $\log p_\theta(x_0)$ [31, 35]. In the process of defining the ELBO, we encounter the intractable term $\log q(x_{t-1} \mid x_t)$ when trying to compute the KL divergence between the true and parameterized reverse processes. To overcome this, we utilize the tractable conditional posterior $q(x_{t-1} \mid x_t, x_0)$, which admits a closed-form Gaussian expression (See Appendix C.2). Specifically, using the Markov structure of the forward process, the posterior can be written as:

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I),$$

where both the mean $\tilde{\mu}_t(x_t, x_0)$ and the variance $\tilde{\beta}_t$ are derived analytically from the forward noising process. This allows us to define a tractable training objective function by minimizing the KL divergence between the true posterior and the model (The full derivation is provided in Appendix C.3). Moreover, each term in the objective function can be decomposed by individual pairs along the time step and this makes the implementation easier and quicker (See Appendix C.4 for details):

$$\mathcal{L}_t := \mathbb{E}_{q(x_t|x_0)} \left[D_{\mathrm{KL}} \left(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t) \right) \right]$$

Using variational inference, the training objective function is defined via the ELBO such that:

$$\begin{aligned}
-\mathrm{ELBO}(x_0; \theta) &= L_T + \sum_{t=1}^{T-1} L_t + L_0, \quad \text{where} \\
L_0 &:= -\mathbb{E}_{q(x_1|x_0)} \left[\log p_\theta(x_0 \mid x_1) \right] \\
L_t &:= \mathbb{E}_{q(x_t|x_0)} \left[D_{\mathrm{KL}} \left(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t) \right) \right], \quad 1 \le t \le T-1 \\
L_T &:= D_{\mathrm{KL}} \left(q(x_T \mid x_0) \,\|\, p(x_T) \right), \quad \text{where } q(x_T \mid x_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_T}\, x_0, (1 - \bar{\alpha}_T) I \right)
\end{aligned}$$

In practice, the loss terms L_0 and L_T are often ignored during training for computational simplicity. The term L_0 depends on the decoder $p_\theta(x_0 \mid x_1)$, which can be replaced by simpler objectives, such as predicting the added noise. The L_T term, which compares $q(x_T \mid x_0)$ to the standard Gaussian prior $p(x_T)$, becomes negligible when T is large and $q(x_T \mid x_0)$ closely approximates $\mathcal{N}(0, I)$. Therefore, most implementations focus on optimizing L_t for $1 \le t \le T-1$. For L_t, both $q(x_{t-1} \mid x_t, x_0)$ and $p_\theta(x_{t-1} \mid x_t)$ are assumed to be multivariate Gaussians with diagonal covariance matrices, enabling closed-form KL evaluation (See

Appendix C.5). Furthermore, Ho et al. [31] proposed reparameterizing $\mu_\theta(x_t, t)$ in terms of predicted noise $\epsilon_\theta(x_t, t)$ (See Appendix C.5) instead of directly regressing towards x_0 or $\tilde{\mu}_t(x_t, x_0)$, thus simplifying the training objective to a mean squared error between the true noise and the predicted noise such that:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \sqrt{1 - \alpha_t}\, \epsilon_\theta(x_t, t) \right)$$

This formulation, known as the ϵ-prediction objective, underlies most practical DDPM implementations. The KL divergence between the true and model posteriors reduces to a weighted mean-squared error on the predicted noise (See Appendix C.5):

$$\mathcal{L}_t^{\text{simple}} := \mathbb{E}_{q(x_t|x_0)} \left[\|\epsilon_\theta(x_t, t) - \epsilon_t\|^2 \right],$$

where $\epsilon_\theta(x_t, t)$ is the model's prediction of the noise added at step t. This formulation, popularized by Ho et al. [31], has become the standard objective for training diffusion models due to its simplicity and empirical success. Finally, the total training objective aggregates the loss over all diffusion steps:

$$\mathcal{L}_{\text{total}} = \sum_{t=1}^{T} \mathbb{E}_{q(x_t, x_0)} \left[\|\epsilon_\theta(x_t, t) - \epsilon_t\|^2 \right]$$

This objective enables the model to learn an effective approximation of the true reverse process $q(x_{t-1} \mid x_t)$, allowing realistic data generation from pure noise via iterative denoising. Below (Figure 6) is a concise overview of the full training and sampling algorithms, including all steps needed to implement a diffusion model.

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure 6: Algorithm summary

2.2 Conditional Diffusion Models

Conditional diffusion extends the standard diffusion framework to model conditional distributions $p(\mathbf{x}_0 \mid \mathbf{y})$, where $\mathbf{y}$ denotes auxiliary information (e.g., class labels, text prompts, or observed time–series segments). By injecting $\mathbf{y}$ into the denoising network, conditioning enables controlled generation aligned with domain context. In practice, three paradigms are widely used: (i) *classifier guidance*, which steers the sampling using $\nabla_{\mathbf{x}_t} \log p_\phi(\mathbf{y} \mid \mathbf{x}_t)$ from a separately trained classifier [41]; (ii) *direct conditioning*, which feeds $\mathbf{y}$ into the denoiser through architectural mechanisms such as concatenation, cross-attention, or FiLM [39, 40, 49, 51]; and (iii) *classifier-free guidance*, which interpolates predictions from jointly trained conditional and unconditional denoisers [37].

Classifier Guidance. Introduced by Dhariwal and Nichol [41], classifier guidance is a post-training conditioning method that enables a pretrained unconditional diffusion model to generate conditional samples. This is achieved by augmenting the sampling-time score function with the gradient of a separately trained, noise-aware classifier $p_\phi(y \mid \mathbf{x}_t, t)$. By Bayes' rule,

$$\log p(\mathbf{x}_t \mid y) \;=\; \log p(\mathbf{x}_t) \;+\; \log p(y \mid \mathbf{x}_t) \;-\; \log p(y)$$

Differentiating w.r.t. $\mathbf{x}_t$ (noting $\nabla_{\mathbf{x}_t} \log p(y) = 0$) gives

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid y) \;=\; \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \;+\; \nabla_{\mathbf{x}_t} \log p(y \mid \mathbf{x}_t), \tag{1}$$

where the marginal score $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$ corresponds to the unconditional denoiser, and the second term injects label information. Since $p(y \mid \mathbf{x}_t)$ is unknown, we approximate it with a separately trained, noise-aware classifier $p_\phi(y \mid \mathbf{x}_t, t)$ (parameters ϕ), yielding

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid y) \;\approx\; \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \;+\; \nabla_{\mathbf{x}_t} \log p_\phi(y \mid \mathbf{x}_t, t), \tag{2}$$

which is the basis of classifier guidance [41]. Practically, a guidance scale $s \geq 0$ is often applied to the second term to trade off fidelity and diversity. (See Appendix D.1 for details). This guidance steers generation toward the desired condition y by modifying the reverse trajectory; in the DDPM parameterization, it is equivalent to sampling from a perturbed Gaussian whose mean is shifted along the classifier gradient, e.g., $\boldsymbol{\mu}_\theta^{\text{guided}}(\mathbf{x}_t, t) = \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) + s\, \sigma_t^2 \nabla_{\mathbf{x}_t} \log p_\phi(y \mid \mathbf{x}_t, t)$ (cf. Eq. (2)). Its main advantage is modularity: a single noise-conditioned classifier trained across timesteps 't' can be reused to guide different generators, and it is particularly effective for class-conditional image generation [41]. However, it requires a backward pass through the classifier at every sampling step (increased cost) and can be sensitive to adversarial or miscalibrated gradients, as well as distribution mismatch between the classifier and the generator, which may degrade sample quality or stability. In practice, a guidance scale 's' together with gradient clipping or other regularization is used to balance fidelity and diversity.

Direct Conditioning. In this approach, the conditioning variable $\mathbf{y}$ is injected into the denoiser at every timestep, so the network predicts $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{y})$ (or equivalently $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t, \mathbf{y})$). The architecture is explicitly conditioned on $\mathbf{y}$ via concatenation, cross-attention, or conditional normalization (e.g., FiLM; conditional GroupNorm) [47, 49, 51] (see Appendix D.2). Training uses pairs $(\mathbf{x}_0, \mathbf{y})$ in which only $\mathbf{x}_0$ is noised by the forward process while $\mathbf{y}$ remains fixed. Since no decomposition in Eq. (1) is used, $p(x_t \mid y)$ is treated as a black-box distribution modeled directly via neural networks. This method enables the model to tightly couple the generation with the condition and has been widely adopted in text-to-image synthesis, semantic image manipulation, and time-series forecasting. Because the condition is embedded directly into the model, no guidance signal or interpolation is required at inference time. However, it lacks sampling-time flexibility, as the conditioning mechanism is hard-coded into the network.

Classifier-Free Guidance (CFG). Proposed by Ho and Salimans [37], CFG removes the external classifier and builds on direct conditioning. A single noise prediction network $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{y})$ is trained with *condition dropout*: with probability p_{drop}, condition $\mathbf{y}$ is replaced by a null token $\emptyset$, so the model learns both conditional and unconditional behaviors. At inference, we compute

$$\boldsymbol{\epsilon}_c = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{y}), \qquad \boldsymbol{\epsilon}_u = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \emptyset),$$

and interpolates them linearly to approximate conditional guidance:

$$\hat{\boldsymbol{\epsilon}}(\mathbf{x}_t, t, \mathbf{y}) = (1 + s)\, \boldsymbol{\epsilon}_c - s\, \boldsymbol{\epsilon}_u,$$

where 's' is a guidance scale hyperparameter that controls the strength of the condition (See Appendix D.3 for details). This allows users to dynamically balance condition fidelity and sample diversity at sampling time without modifying the model architecture. CFG offers several key benefits: it requires only a single model, avoids classifier-induced instability, and provides stable, high-quality outputs. One trade-off, however, is that CFG still relies on direct conditioning during training and requires careful tuning of p_{drop} and s for optimal performance.

2.3 Conditional Diffusion for Time-Series

In time series generation, the goal is to predict a future segment from a partially observed history. i.e., given $\mathbf{y}_{\text{obs}} \in \mathbb{R}^{L \times d}$, produce $\mathbf{y}_{\text{tar}} \in \mathbb{R}^{H \times d}$. Unlike image settings where y is often a low-dimensional label, here the condition itself is a high-dimensional temporal signal with structures such as ordering, autocorrelation, seasonality. Consequently, classifier guidance is rarely used: differentiating through an external sequence classifier at every step is costly and can destabilize temporal coherence. Instead, time series diffusion models prefer to directly incorporate the observed past into the denoiser through concatenation, cross-attention, or together with masks that mark observed positions so the network learns $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{y}_{\text{obs}})$ [39, 40]. This direct conditioning preserves temporal structure and supports both forecasting and imputation within a unified objective. In *direct conditioning*, the observed sequence $\mathbf{X}_{\text{obs}}$ is treated as auxiliary input and is not noised by the forward process; only the target segment $\mathbf{X}_{\text{tar}}$ is diffused (see Figure 7). Let $\mathbf{x}_0 \equiv \mathbf{X}_{\text{tar}}$. The forward marginal is

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0,\ (1-\bar{\alpha}_t)\,\mathbf{I}\big)$$

The observation $\mathbf{X}_{\text{obs}}$ is encoded by an auxiliary network E (e.g., RNN or Transformer encoder) into an embedding $\mathbf{e} = E(\mathbf{X}_{\text{obs}})$, which is injected into the denoiser via concatenation, conditional normalization (e.g., FiLM), or cross-attention [49, 51]. Then, the denoiser predicts the conditional noise $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{e})$, which defines the reverse mean as:

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t, \mathbf{e}) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{e})\right),$$

and the sampling step as:

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_\theta(\mathbf{x}_t, t, \mathbf{e}) + \sqrt{\sigma_t^2}\cdot\mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

where σ_t^2 is typically set to the posterior variance $\tilde{\beta}_t$ (or learned) [34]. This direct conditioning preserves temporal dependencies while enabling flexible, expressive generation conditioned on past observations [39, 40].

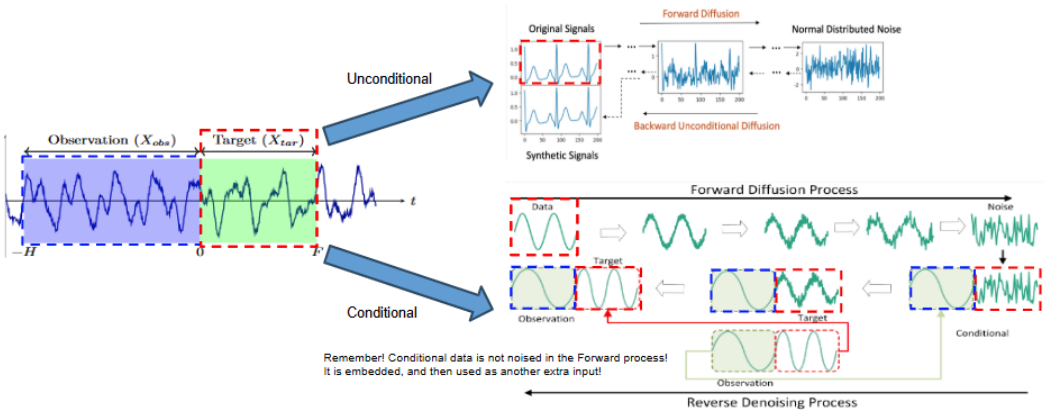


Figure 7: Direct Conditioning Framework

Training. The model is trained on paired samples $(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{tar}})$ where only $\mathbf{X}_{\text{tar}}$ is diffused. Let $\mathbf{x}_0 \equiv \mathbf{X}_{\text{tar}}$ and $\mathbf{e} = E(\mathbf{X}_{\text{obs}})$. At each step, we sample $t \sim \mathcal{U}\{1, \dots, T\}$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, to form

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\, \boldsymbol{\epsilon},$$

and feed $(\mathbf{x}_t, t, \mathbf{e})$ to the denoiser to predict $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{e})$. The objective function minimizes the mean squared error between the predicted and true noise:

$$\mathbb{E}_{t, x_0, \epsilon, \mathbf{X}_{\text{obs}}} \left[\| \epsilon_\theta(x_t, t, \mathbf{e}) - \epsilon \|^2 \right]$$

This training strategy ensures that the model learns to denoise target signals based on the contextual information provided by the observed sequence.

Sampling. At inference, we condition on the observed history via $\mathbf{e} = E(\mathbf{X}_{\text{obs}})$, draw $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and iteratively apply the learned reverse transitions:

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_\theta(\mathbf{x}_t, t, \mathbf{e}) + \sigma_t\, \mathbf{z}_t, \qquad \mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

yielding $\hat{\mathbf{X}}_{\text{tar}} = \mathbf{x}_0 \sim p_\theta(\mathbf{X}_{\text{tar}} \mid \mathbf{X}_{\text{obs}})$. This formulation naturally supports varying observation ratios, captures nonlinear or multimodal degradation, and admits faster samplers (e.g., DDIM) when fewer steps are desired [32].

3 Diffusion Model Architecture

Let $\mathbf{x}^{(i)} = \left(x_1^{(i)}, \dots, x_{T_i}^{(i)}\right)$ denote the degradation signal of unit i for $i = 1, \dots, N$. Given a predefined failure threshold x_{th}, the time-to-failure (TTF) $t_f^{(i)}$ is defined as the first threshold-crossing time such that $t_f^{(i)} := \inf\{t : x_t^{(i)} \geq x_{\text{th}}\}$. The full dataset of N units can be partitioned into a training set $\mathcal{D}_{\text{tr}}$ and a test set $\mathcal{D}_{\text{te}}$ using arbitrary splitting strategies depending on the application; in this study, we adopt a fixed $8{:}2$ train / test split to ensure consistent and reproducible evaluation. The training set is used exclusively to learn the conditional generative model, while the test set is reserved for sampling-based prediction and performance assessment. For each training unit, we construct paired segments $(\mathbf{X}_{\text{obs}}^{(i)}, \mathbf{x}_0^{(i)})$, where the partially observed prefix is $\mathbf{X}_{\text{obs}}^{(i)} = \left(x_1^{(i)}, \dots, x_{T_{\text{obs}}}^{(i)}\right)$ and the future target segment to be generated is $\mathbf{x}_0^{(i)} = \left(x_{T_{\text{obs}}+1}^{(i)}, \dots, x_{T_{\text{obs}}+T_{\text{tar}}}^{(i)}\right) \in \mathbb{R}^{T_{\text{tar}}}$. We then train a conditional diffusion model on $\mathcal{D}_{\text{tr}}$ to learn $p(\mathbf{x}_0 \mid \mathbf{X}_{\text{obs}})$ by inverting a forward Gaussian noising process with a learned denoiser, following the *direct conditioning* paradigm. The denoiser adopts a hybrid Temporal U-Net/Transformer backbone [45, 46, 49, 58], equipped with Fourier/positional embeddings and diffusion-step embeddings based on the log signal-to-noise ratio (log-SNR), $\lambda_\tau := \log \frac{\bar{\alpha}_\tau}{1 - \bar{\alpha}_\tau}$, which parameterizes the noise level at diffusion step τ [36, 49, 52]. Cross-attention modules inject a representation of $\mathbf{X}_{\text{obs}}$ after each downsampling stage to ensure time-aligned conditional generation. To enable sampling-time control without retraining, we incorporate classifier-free guidance (CFG) [37]: during training, we apply *condition dropout* by replacing $\mathbf{X}_{\text{obs}}$ with $\emptyset$ with probability p_{drop} (tuned via Optuna [57]) so that a single denoiser learns both conditional and unconditional behaviors, and at inference we guide sampling by linearly combining the conditional and unconditional noise predictions to balance fidelity and diversity. In deployment (testing/real-time prediction), for each unseen test unit with a partially observed signal $\mathbf{X}_{\text{obs}}$, we draw M conditional continuations $\{\hat{\mathbf{x}}_0^{(m)}\}_{m=1}^M \sim p(\mathbf{x}_0 \mid \mathbf{X}_{\text{obs}})$ using the trained model and stitch them with $\mathbf{X}_{\text{obs}}$ to form plausible full trajectories; applying the same threshold rule to each generated trajectory yields Monte Carlo samples of the corresponding failure times, from which we obtain an empirical predictive distribution of TTF.

3.1 Forward Process and Noise Injection

Given a target degradation segment $\mathbf{x}_0 \in \mathbb{R}^{T_{\text{tar}}}$, we define a forward diffusion process that injects temporally correlated Gaussian noise. Whereas the original DDPM assumes i.i.d. noise at each diffusion step [31], degradation signals often exhibit temporal continuity, making i.i.d. perturbations suboptimal for sequence modeling. We therefore draw noise from a zero-mean Gaussian process (GP) on the time grid [42, 43] with a squared-exponential (SE) kernel

$$K_{\text{SE}}(t_i, t_j) = \exp\left(-\frac{(t_i - t_j)^2}{2\ell^2}\right),$$

where $K_{\text{SE}}(\cdot,\cdot)$ denotes the SE covariance kernel, i.e., a Gaussian process covariance function that encodes temporal correlation between time indices by assuming that noise values at temporally closer points are more similar, and ℓ is the length-scale parameter controlling the degree of smoothness. At each diffusion step $\tau \in \{1, \ldots, T\}$, we sample $\boldsymbol{\epsilon}_\tau \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\text{SE}})$ (independent across τ but correlated across time indices) and use a cosine scheduler [34] to obtain $\{\beta_\tau\}_{\tau=1}^T$ with $\alpha_\tau = 1 - \beta_\tau$ and $\bar{\alpha}_\tau = \prod_{s=1}^{\tau} \alpha_s$. For a fixed time grid, we precompute a numerically stable Cholesky factorization $\mathbf{K}_{\text{SE}} + \delta\mathbf{I} = \mathbf{L}\mathbf{L}^\top$ (with jitter $\delta \approx 10^{-6}$), sample white noise $\mathbf{z}_\tau \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and set $\boldsymbol{\epsilon}_\tau = \mathbf{L}\mathbf{z}_\tau$. The forward marginal then becomes

$$q(\mathbf{x}_\tau \mid \mathbf{x}_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_\tau}\,\mathbf{x}_0,\ (1 - \bar{\alpha}_\tau)\,\mathbf{K}_{\text{SE}}\big),$$

equivalently written in generative form as

$$\mathbf{x}_\tau = \sqrt{\bar{\alpha}_\tau}\,\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_\tau}\,\boldsymbol{\epsilon}_\tau, \qquad \boldsymbol{\epsilon}_\tau \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\text{SE}}).$$

This induces smooth and temporally coherent perturbations that better match degradation dynamics; diffusion models with GP-structured noise have also shown benefits in related settings by improving temporal consistency, stabilizing the reverse denoising process, and yielding more realistic sample paths compared to i.i.d. noise assumptions [44].

3.2 Time and Positional Encoding

To guide the reverse diffusion process for time-series degradation modeling, we explicitly distinguish two notions of time: the diffusion step index $\tau \in \{0, \ldots, T\}$ and the physical timestamp $t \in \mathcal{T} \subset \mathbb{R}$. The diffusion step τ determines the noise level in the generative process, whereas t represents the actual sampling times of the degradation signal. Encoding both into the reverse process is essential: τ governs the progression of the forward noising process, while t specifies when the signal is observed along the degradation timeline. We therefore design complementary embeddings that jointly condition the reverse process, allowing the model to align noise-level progression with physical time context.

- **Diffusion-step embedding (for τ).** We encode the diffusion step using a combination of sinusoidal features and the log signal-to-noise ratio (log-SNR), so that the network is informed of both the progression along the diffusion trajectory and the current noise intensity. Specifically, the normalized step τ/T is mapped to fixed sinusoidal features

  $$\gamma(\tau) = \big[\sin(\omega_1 \tfrac{\tau}{T}),\ \cos(\omega_1 \tfrac{\tau}{T}),\ \ldots,\ \sin(\omega_K \tfrac{\tau}{T}),\ \cos(\omega_K \tfrac{\tau}{T})\big],$$

  where $\{\omega_k\}_{k=1}^K$ are predefined frequencies that span multiple temporal scales of the diffusion process. Low-frequency components encode coarse progression along the diffusion

trajectory, while high-frequency components resolve fine-grained differences between nearby diffusion steps, allowing the model to represent diffusion time continuously rather than as a discrete index. These features are concatenated with the log-SNR

$$\lambda_\tau = \log \frac{\bar{\alpha}_\tau}{1 - \bar{\alpha}_\tau},$$

which offers a schedule-invariant and physically meaningful measure of noise intensity at step τ. The combined vector is projected through a small multilayer perceptron (MLP),

$$\mathbf{e}_\tau = \mathrm{MLP}\big([\gamma(\tau),\, \lambda_\tau]\big),$$

yielding a compact embedding that supports smooth interpolation across diffusion steps and stable conditioning across different noise regimes [31, 34, 35, 36].

- **Physical-time embedding (for t).** To encode the actual sampling times of the degradation signal, we use Fourier (positional) features that map scalar timestamps to a multi-frequency representation [49, 52],

$$\gamma_{\text{phys}}(t_i) = \big[\sin(\omega_1 t_i),\, \cos(\omega_1 t_i),\, \ldots,\, \sin(\omega_K t_i),\, \cos(\omega_K t_i)\big].$$

This encoding allows the network to represent long-term degradation trends and local temporal variations using linear operations in the embedding space. Each $\gamma_{\text{phys}}(t_i)$ is passed through a simple MLP,

$$\mathbf{u}_i = \mathrm{MLP}\big(\gamma_{\text{phys}}(t_i)\big),$$

which learns an adaptive nonlinear projection of the fixed Fourier features. The resulting vectors are aggregated over the window using a pooling operator (e.g., mean pooling) to form a fixed-size physical-time embedding

$$\mathbf{e}_t = \mathrm{Pool}_i\, \mathbf{u}_i$$

This embedding summarizes the temporal location of the target segment along the degradation timeline, independently of the diffusion noise level.

We then concatenate the diffusion-step and physical-time embeddings to form a unified time representation $\mathbf{h}_{\tau,t} = [\,\mathbf{e}_\tau\,;\,\mathbf{e}_t\,]$, which is injected into the denoising network via feature-wise linear modulation (FiLM) [51]. At each denoiser stage b, a MLP maps $\mathbf{h}_{\tau,t}$ to FiLM parameters $(\boldsymbol{\alpha}_b, \boldsymbol{\beta}_b)$ that modulate intermediate feature maps $\mathbf{f}_b$ as

$$\tilde{\mathbf{f}}_b = \boldsymbol{\alpha}_b \odot \mathbf{f}_b + \boldsymbol{\beta}_b.$$

This mechanism allows the network to adapt its computations jointly to the current noise level (controlled by τ) and the physical time context (encoded by t), thereby enabling noise-aware and time-aware denoising of degradation trajectories.

3.3 Network Architecture

The denoising network follows a modified *Temporal U-Net* [45] tailored to degradation dynamics. To better incorporate conditional context and long-range dependencies, we augment the backbone architecture with Transformer-based sequence modeling [50, 58] and cross-attention [46, 47, 49, 51].

- **Inputs per reverse step.** At each reverse step τ, the network consumes (i) the noised target $\mathbf{x}_\tau \in \mathbb{R}^{1\times T_{\text{tar}}}$, (ii) the previous-step prediction $\hat{\mathbf{x}}_0^{(\tau+1)}$ (self-conditioning) [34], and (iii) the observed context $\mathbf{X}_{\text{obs}} \in \mathbb{R}^{1\times T_{\text{obs}}}$. We concatenate $\mathbf{x}_\tau$ and $\hat{\mathbf{x}}_0^{(\tau+1)}$ along channels to obtain $[\,\mathbf{x}_\tau; \hat{\mathbf{x}}_0^{(\tau+1)}\,] \in \mathbb{R}^{2\times T_{\text{tar}}}$ and project it to $\mathbb{R}^{T_{\text{tar}}\times C}$ via a 1D convolutional encoder. In parallel, $\mathbf{X}_{\text{obs}}$ is encoded by a 1D convolution followed by a Transformer encoder, yielding context features $\mathbf{H}_{\text{obs}} \in \mathbb{R}^{T_{\text{obs}}\times C}$.

- **Backbone and time conditioning.** The backbone is a hierarchical encoder–decoder with downsampling stages, a bottleneck, and upsampling stages. Each stage uses residual 1D blocks with GroupNorm [47] and SiLU [46, 48]. The hybrid embedding $\mathbf{h}_{\tau,t}$ is injected via FiLM [51] to enable diffusion step and time aware adaptation.

- **Cross-attention for conditioning.** After each downsampling stage, we fuse the observed context into the target representation via cross-attention. Let $\mathbf{H} \in \mathbb{R}^{T\times C}$ denote the latent features of the target segment (queries) and $\mathbf{H}_{\text{obs}} \in \mathbb{R}^{T_{\text{obs}}\times C}$ denote the latent features of the observed segment (keys and values). Cross-attention computes weights that quantify how much each target timestep i should attend to each observed timestep j, producing a context-aware update that is added back to $\mathbf{H}$ through a residual connection followed by layer normalization [49]. To do so, we first apply linear projections

$$\mathbf{Q} = \mathbf{H}\mathbf{W}^Q \in \mathbb{R}^{T\times d}, \qquad \mathbf{K} = \mathbf{H}_{\text{obs}}\mathbf{W}^K \in \mathbb{R}^{T_{\text{obs}}\times d}, \qquad \mathbf{V} = \mathbf{H}_{\text{obs}}\mathbf{W}^V \in \mathbb{R}^{T_{\text{obs}}\times d_v},$$

where $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V$ are learnable projection matrices. The attention weights are then computed as

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right) \in \mathbb{R}^{T\times T_{\text{obs}}},$$

where the softmax is applied row-wise so that each row satisfies $\sum_j a_{ij} = 1$. The context-aware update is obtained by

$$\mathbf{C} = \mathbf{A}\mathbf{V} \in \mathbb{R}^{T\times d_v},$$

and incorporated into the target features via

$$\mathbf{H}' = \text{LN}\big(\mathbf{H} + \mathbf{C}\mathbf{W}^O\big),$$

where $\mathbf{W}^O \in \mathbb{R}^{d_v\times C}$ is a learnable output projection and $\text{LN}(\cdot)$ denotes layer normalization. Intuitively, this operation aligns the latent representation of the target sequence with informative portions of the observed context, improving conditional generation for imputation and forecasting while the residual connection maintains training stability.

- **Heads and outputs.** The decoder terminates with two 1×1 convolutional heads. The first head predicts the diffusion noise $\hat{\boldsymbol{\epsilon}}_\tau$ and is trained using the standard noise mean-squared error objective of diffusion models [31, 34]. The second head reconstructs the clean signal $\hat{\mathbf{x}}_0$ and is used to compute auxiliary temporal-consistency losses (e.g., trend-alignment and boundary-consistency penalties), which encourage physically plausible degradation trajectories.

3.4 Training & Optimization

We train the model using a composite objective that balances accurate denoising with temporally coherent reconstruction of degradation trajectories. While the standard diffusion loss

enforces correct noise prediction, additional signal-level constraints are introduced to encourage physically plausible and temporally consistent generations. At reverse step τ, the per-step objective function is

$$\mathcal{L}_\tau = \mathcal{L}_{\text{noise}} + \lambda_{x_0}\,\mathcal{L}_{x_0} + \lambda_{\text{trend}}\,\mathcal{L}_{\text{trend}} + \lambda_{\text{bound}}\,\mathcal{L}_{\text{bound}},$$

where each term serves a distinct modeling purpose:

- **Noise matching (primary).** With GP covariance $\mathbf{K}_{\text{SE}} = \mathbf{L}\mathbf{L}^\top$, we predict the whitened noise $\tilde{\boldsymbol{\epsilon}}_\tau := \mathbf{L}^{-1}\boldsymbol{\epsilon}_\tau$ and minimize

  $$\mathcal{L}_{\text{noise}} = \left\| \hat{\tilde{\boldsymbol{\epsilon}}}_\tau - \tilde{\boldsymbol{\epsilon}}_\tau \right\|_2^2.$$

  This is the primary diffusion objective and is equivalent to a Mahalanobis-weighted MSE in the original coordinates, ensuring that the denoiser correctly inverts the GP-correlated forward noising process.

- **Signal reconstruction.** This term encourages accurate recovery of the clean degradation signal and stabilizes training by directly supervising the reconstruction of $\mathbf{x}_0$ with

  $$\mathcal{L}_{x_0} = \|\hat{\mathbf{x}}_0 - \mathbf{x}_0\|_2^2\,.$$

- **Trend alignment.** Let Δ denote first-order differencing, $(\Delta\mathbf{x})_i = \mathbf{x}_i - \mathbf{x}_{i-1}$. We define

  $$\mathcal{L}_{\text{trend}} = \|\Delta\hat{\mathbf{x}}_0 - \Delta\mathbf{x}_0\|_2^2\,,$$

  which penalizes discrepancies in local temporal slopes and helps preserve physically meaningful degradation trends, such as monotonic wear or gradual acceleration.

- **Boundary consistency.** Let $\mathbf{x}_0^{(1)}$ be the first element of the target window and $\hat{\mathbf{x}}_0^{(1)}$ its reconstruction. We impose

  $$\mathcal{L}_{\text{bound}} = \left\| \hat{\mathbf{x}}_0^{(1)} - \mathbf{x}_0^{(1)} \right\|_2^2,$$

  to enforce continuity between the observed prefix and the generated future segment, preventing spurious jumps at the observation–prediction boundary.

The weighting coefficients λ_{x_0}, λ_{trend}, and λ_{bound} control the trade-off between denoising accuracy and temporal coherence, and are selected using a validation split within $\mathcal{D}_{\text{tr}}$.

Classifier-Free Guidance (CFG) for Conditional Generation. To enable CFG while retaining generative flexibility, we apply *condition dropout* during training: with probability p_{drop}, the conditioning input is replaced by null condition $\emptyset$. Concretely, in each iteration, we sample $\tau \sim \mathcal{U}\{1,\dots,T\}$, draw $\boldsymbol{\epsilon}_\tau \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\text{SE}})$, sample $\delta \sim \text{Bernoulli}(p_{\text{drop}})$, and set $\mathbf{X}_{\text{obs}}^{(\delta)} \in \{\mathbf{X}_{\text{obs}}, \emptyset\}$. The overall objective becomes

$$\min_\theta \; \mathbb{E}_{\tau,\,\boldsymbol{\epsilon}_\tau,\,\delta}\Big[\, \mathcal{L}_\tau\big(\mathbf{x}_\tau, \mathbf{X}_{\text{obs}}^{(\delta)}; \theta\big)\,\Big].$$

At inference, CFG is implemented by evaluating the denoiser with and without the condition and combining the two predictions [37]:

$$\hat{\boldsymbol{\epsilon}}_{\text{cond}} = \boldsymbol{\epsilon}_\theta(\mathbf{x}_\tau, \tau, E(\mathbf{X}_{\text{obs}})), \quad \text{and} \quad \hat{\boldsymbol{\epsilon}}_{\text{uncond}} = \boldsymbol{\epsilon}_\theta(\mathbf{x}_\tau, \tau, \emptyset)$$
$$\hat{\boldsymbol{\epsilon}}^{\text{guided}} = \hat{\boldsymbol{\epsilon}}_{\text{uncond}} + w\big(\hat{\boldsymbol{\epsilon}}_{\text{cond}} - \hat{\boldsymbol{\epsilon}}_{\text{uncond}}\big),\; w > 1,$$

where w is the guidance scale controlling the trade-off between conditional fidelity and sample diversity.

Bayesian Hyperparameter Optimization via Optuna. Model hyperparameters are selected using Bayesian optimization with **Optuna** [57], which efficiently explores high-dimensional search spaces. The search ranges, chosen empirically to cover practically relevant regimes, are

$$\begin{aligned} \lambda_{\text{trend}} &\sim \mathcal{U}[0.1,\, 5.0], & \lambda_{x_0} &\sim \mathcal{U}[0.1,\, 2.0], \\ \lambda_{\text{bound}} &\sim \mathcal{U}[0.1,\, 20.0], & p_{\text{drop}} &\sim \mathcal{U}[0.0,\, 0.3], \\ \beta_{\text{end}} &\sim \text{LogU}[10^{-3},\, 5\times 10^{-2}], & \ell_{\text{SE}} &\sim \text{LogU}[10^{-3},\, 5\times 10^{-2}], \\ \texttt{lr} &\sim \text{LogU}[10^{-5},\, 10^{-3}], & \texttt{ch} &\in \{64,\, 96, \ldots,\, 256\}, \end{aligned}$$

where β_{end} controls the terminal noise level of the cosine diffusion schedule, ℓ_{SE} is the length scale of the squared-exponential kernel used in GP-based noise injection, `lr` is the learning rate, and `ch` denotes the base channel width of the U-Net backbone. For each Optuna trial, the model is trained from scratch on the training split with early stopping, and performance is evaluated using the validation estimate of $\mathbb{E}_{\tau,\boldsymbol{\epsilon},\delta}[\mathcal{L}_\tau]$. After selecting the best hyperparameter configuration, we further sweep the classifier-free guidance scale $w \in \{1.0, 1.5, 2.0, 3.0, 4.0\}$ on the validation set to control the fidelity–diversity trade-off at inference.

3.5 Real-time Prediction & TTF Estimation

This subsection outlines the practical use of the trained conditional diffusion model for real-time prediction and failure-time inference on an unseen unit in the test data. Given a test unit with a partially observed prefix $\mathbf{X}_{\text{obs}} = \big(x_1, \ldots, x_{T_{\text{obs}}}\big)$, our objective is to obtain a predictive distribution of the time-to-failure (TTF) under a predefined threshold x_{th}.

- **Step 1. Conditional sampling of future trajectories.** We draw M Monte Carlo samples from the learned conditional distribution,

$$\hat{\mathbf{x}}_0^{(m)} \sim p_\theta(\mathbf{x}_0 \mid \mathbf{X}_{\text{obs}}), \qquad m = 1, \ldots, M,$$

  and concatenate each sampled future segment with the observed prefix to form plausible full degradation trajectories $\hat{\mathbf{x}}^{(m)} = [\,\mathbf{X}_{\text{obs}}; \hat{\mathbf{x}}_0^{(m)}\,]$. Classifier-free guidance with a guidance scale w may be applied during sampling to control the fidelity–diversity trade-off.

- **Step 2. Threshold-crossing time estimation.** For each generated trajectory $\hat{\mathbf{x}}^{(m)}$, we compute the first threshold-crossing time

$$\hat{t}_f^{(m)} := \inf\{t :\, \hat{x}_t^{(m)} \geq x_{\text{th}}\}.$$

- **Step 3. Predictive distribution of TTF.** The set $\{\hat{t}_f^{(m)}\}_{m=1}^M$ constitutes Monte Carlo samples from the predictive TTF distribution. We summarize this distribution using standard statistics such as the mean, median, and credible intervals, providing both point predictions and uncertainty quantification for performance comparison against benchmarks.

Section 4 will present the detailed experimental design, benchmark comparisons, and evaluation metrics (e.g., RMSE and MAPE), as well as implementation-level choices for sampling and performance assessment.

4 Experiment

4.1 Dataset

We define *parametric models* as approaches that assume a fixed, finite-dimensional family for the RUL/TTF distribution, such as Bayesian updating or lognormal regression with PCA-based dimension reduction. In contrast, we refer to diffusion-based approaches as *nonparametric models*, which learn the data distribution directly without prescribing a specific functional form. To compare these models under controlled yet progressively challenging conditions, we simulate synthetic degradation datasets that reflect degradation patterns commonly observed in real-world engineering systems as described in Section 1.2. Specifically, we consider degradation processes whose mean behavior follows an exponential trend, as widely used to model wear-out, fatigue accumulation, and aging phenomena in mechanical components, while observations are perturbed by Brownian-motion noise to capture temporally correlated uncertainty arising from measurement error, environmental fluctuations, and operational variability. This construction yields globally increasing degradation trajectories with local stochastic variability, providing a physically plausible and reproducible testbed for benchmarking diffusion-based methods against parametric baselines. We begin with two representative baseline types that reveal contrasting behaviors in monotone and non-monotone regimes.

- **Type 1 (Monotone)**: Exponential growth with Brownian-motion noise. The degradation signal is globally increasing, while fluctuations around the exponential trend arise from temporally correlated noise, representing local variability without trend reversal.
- **Type 2 (Non-monotone)**: An initial exponential increase, a short linear pullback, and a subsequent exponential rise, with Brownian perturbations throughout. This structure introduces explicit trend reversals and regime changes, enabling evaluation of model robustness to nonstationarity and mode shifts.

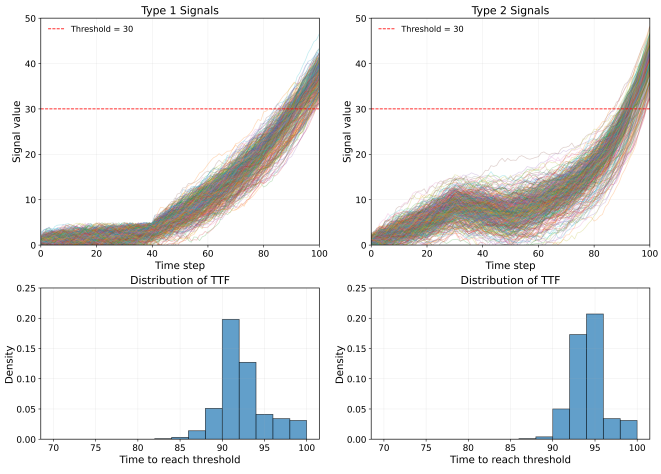


Figure 8: Examples of synthetic degradation signals (Type 1 – 2)

To further assess robustness under more realistic and challenging conditions, we introduce two additional families of synthetic datasets beyond the Type 1–2 baselines:

- **Complexity 1–4**. From the Type 1, we inject K localized perturbations at randomly selected time points, with $K \in \{1, 2, 3, 4\}$ corresponding to Complexity 1–4, respectively.

Each perturbation has an amplitude sampled from $\mathcal{U}[10, 20]$ and induces a short-lived increase in the local slope of the degradation signal. These perturbations introduce abrupt but transient changes in the local trend, emulating phenomena like thermal spikes, external shocks, or temporary control instabilities commonly observed in operational systems.

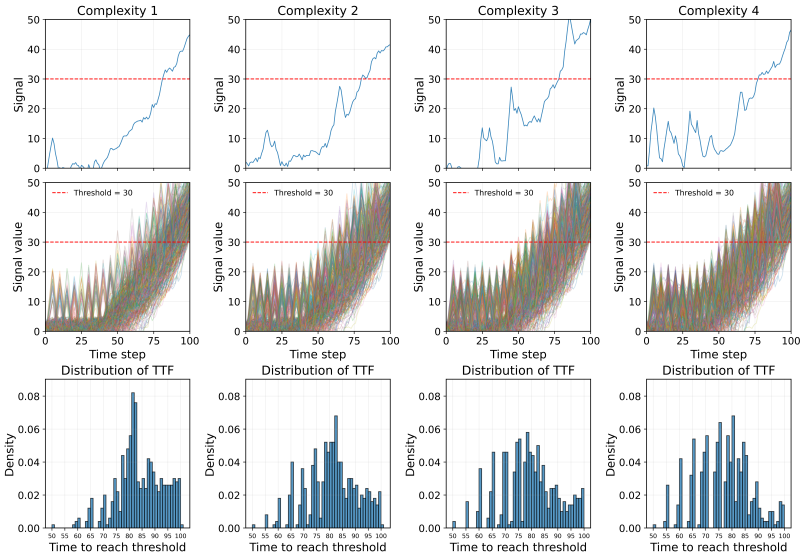


Figure 9: Examples of synthetic degradation signals (Complexity 1–4)

- **Modality 1–4**. Trajectories are generated under multiple operating regimes. The number of extra regimes increases with the level $M=1, 2, 3, 4$ for Modality 1–4. This construction produces a population-level mixture of degradation behaviors, resulting in multimodal TTF distributions and heteroscedastic variability. Such settings enable evaluation of model performance under regime-dependent dynamics arising from shifts in operating conditions or gradual sensor drift.

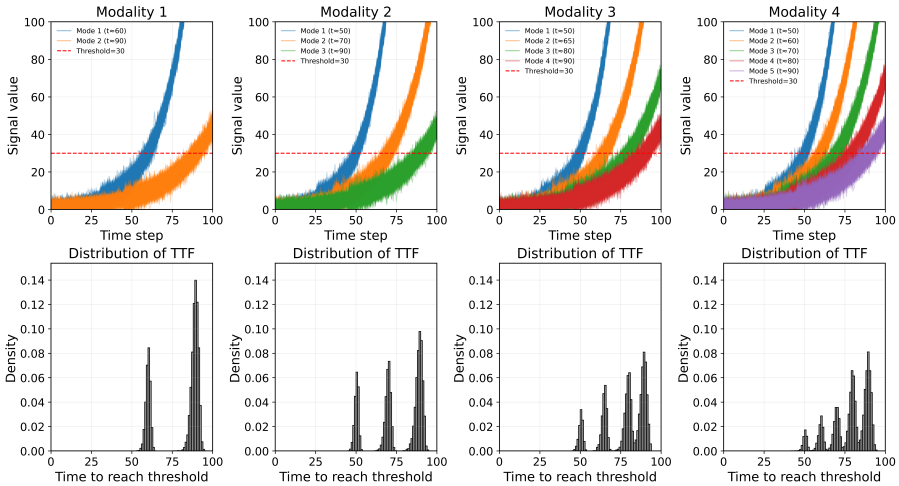


Figure 10: Examples of synthetic degradation signals (Modality 1–4)

Unlike Type 1–2, these families are intentionally irregular and nonstationary. They provide a stronger test of whether a method can be generalized beyond simplified assumptions and

recover the structure of complex degradation processes. In particular, they expose the limits of parametric baseline models, while favoring diffusion-based non-parametric models that learn the distribution without a fixed functional form. We use these datasets as diagnostic tools to study model behavior under realistic uncertainty and operational variability.

4.2 Design of Experiment

We pursue primary benchmark comparisons in terms of prediction performance, corresponding to the two classical methods. Specifically, **Benchmark 1** is *Modeling-based approaches* that explicitly model the temporal evolution of degradation trajectories and infer TTF through sequential updating; this category is represented by Bayesian updating and compared against the proposed diffusion-based model. In addition, **Benchmark 2** corresponds to *data-driven approaches* that directly map partially observed degradation signals to TTF; this category is represented by PCA–lognormal regression and compared against the diffusion model. To ensure robust performance assessment, we adopt a 5-fold cross-validation strategy, where each dataset is split into training and validation sets using a 9:1 ratio.

Point 1: Bayesian Update vs. Diffusion. We evaluated the predictive performance of Bayesian update and diffusion models to estimate TTF. The experiments are conducted using both Type 1 and Type 2 datasets. For each signal, four different initial observation ratios—20%, 40%, 60%, and 80%—are used as conditions, and the correspondingly remaining portions as targets.

- **Bayesian update model:** The prior parameters (θ, β) are estimated from the training set, and the posterior distributions of the parameters are updated using the observed portion of the test signals. The median of remaining-useful life (RUL) is then computed analytically from the posterior and TTF is inferred by adding the predicted RUL to the last observed time point.(See the details in Appendix A)

- **Diffusion model:** A conditional diffusion model is pre-trained on the training set using the specified observation ratio. During evaluation, the model generates the unobserved target portion of each test signal conditioned on its observed segment. TTF is then estimated based on when the generated signal reaches a predefined failure threshold.

The predicted TTF of both models is compared with the ground truth of the test data to assess predictive accuracy. This evaluation is repeated with varying training data sizes—100, 1,000, 5,000, and 10,000—to examine the robustness of the model at different levels of data availability. Given the structured nature of the Type 1 and Type 2 datasets, the diffusion model is expected to exhibit superior performance over the Bayesian update model across most conditions. As such, the evaluation for this comparison is limited to these two datasets, since they already offer a sufficiently clear basis for assessing relative model capabilities.

Point 2: PCA+Lognormal vs. Diffusion We also compared the performance of a PCA-based lognormal regression against the diffusion model for predicting TTF. Following the setup in Point 1, experiments are conducted on Type 1 and Type 2 datasets using four different observation ratios to emulate scenarios where only a portion of the degradation process is observed.

- **PCA+Lognormal model:** Principal component analysis (PCA) is applied to reduce the dimensionality of the observed segments. A lognormal regression is then trained on the

truncated principal components to predict TTF. In inference, the test data are projected onto the PCA space and the fitted model is used to sample the RUL values from the distribution of $\log(RUL) \sim \mathcal{N}(\mu, \sigma^2)$, where $\mu = \mathbf{w}^\top \mathbf{x} + b$. TTF is then computed by adding the sampled RUL to the last observed time point.(See the details in Appendix B)

- **Diffusion model:** The same conditional diffusion framework from Point 1 is used.

To assess the impact of signal complexity and observation ratio on model performance, we further extend the evaluation to the *Complexity* and *Modality* datasets. These datasets contain more irregular and nonstationary patterns compared to Type 1 and Type 2 where PCA-based lognormal regression models are expected to perform competitively by effectively capturing key features via dimensionality reduction due to the relatively smooth and monotonic degradation structures. However, to evaluate model robustness under more challenging conditions, we apply higher observation ratios—50, 60, 70, and 80%—of the *Complexity* and *Modality* datasets, where irregular, multimodal, or nonstationary behaviors begin to emerge.

4.3 Evaluation Metrics

Model performance is quantitatively assessed using two primary error metrics: Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). These metrics are computed based on the predicted and true TTF values across all test samples.

- **Root Mean Squared Error (RMSE)**

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$$

RMSE quantifies the average size of prediction errors in the same units. Because it squares the errors before averaging, it gives more weight to large errors, making it especially effective for evaluating overall accuracy in TTF prediction.

- **Mean Absolute Percentage Error (MAPE)**

$$\text{MAPE} = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right|$$

MAPE quantifies prediction error as a percentage of the true value, making it a scale-independent metric suitable for comparing performance across signals with different ranges or time horizons.

4.4 Results and Analysis

We present experimental findings for both Point 1 and Point 2 in different observation ratios and further complexities. The analysis focuses on comparing model performance under varying signal characteristics, and assessing how performance evolves as more of the signal is revealed.

Point 1: Diffusion vs. Bayesian Updating Figures 11 and 12 present the RMSE and MAPE of the TTF predictions of the diffusion model and the Bayesian update approach (BM model) in the Type 1 and Type 2 datasets. In Type 1 signals, which follow a relatively well-structured and monotonic degradation trend, the Bayesian update demonstrates competitive performance under limited training conditions (sample size < 5000) or when the observation ratio is high (e.g., 80%). These results reflect the alignment between the parametric assumptions of the model and the structured nature of the data. As the training set grows, the Bayesian model shows a marginal improvement. However, gains remain small because the fixed parametric form cannot adapt beyond its assumptions as the dataset grows. In contrast, the diffusion model benefits substantially from larger training sets, consistently outperforming the baseline.

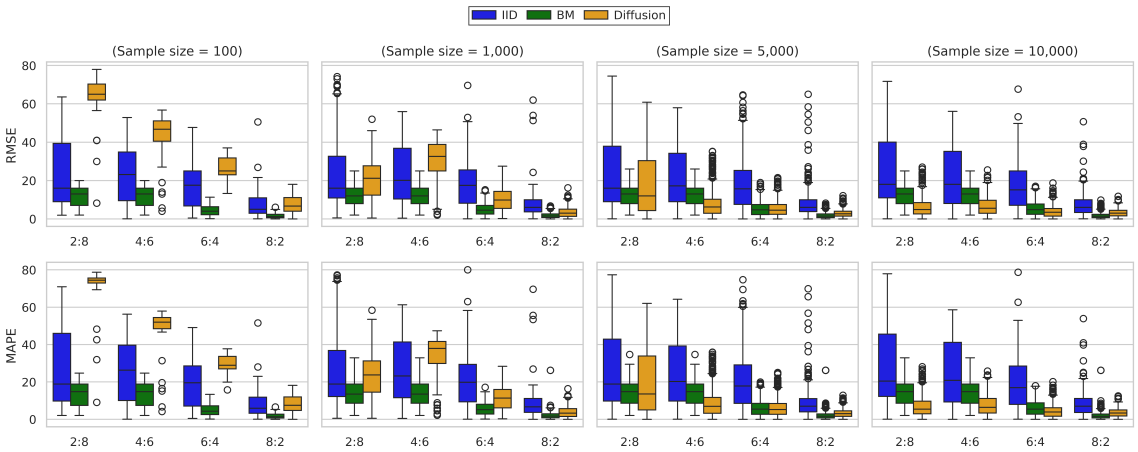


Figure 11: Result of Point 1 with Type 1 data

In Type 2 signals, which exhibit more irregular and nonlinear patterns, the diffusion model demonstrates clear and robust superiority across all observation ratios including 8:2. Provided a moderate training size (≥ 1000), the diffusion model consistently yields lower RMSE and MAPE than the Bayesian update. This performance gap widens as the training set becomes larger, highlighting the ability of diffusion to learn complex degradation dynamics that deviate from conventional patterns.

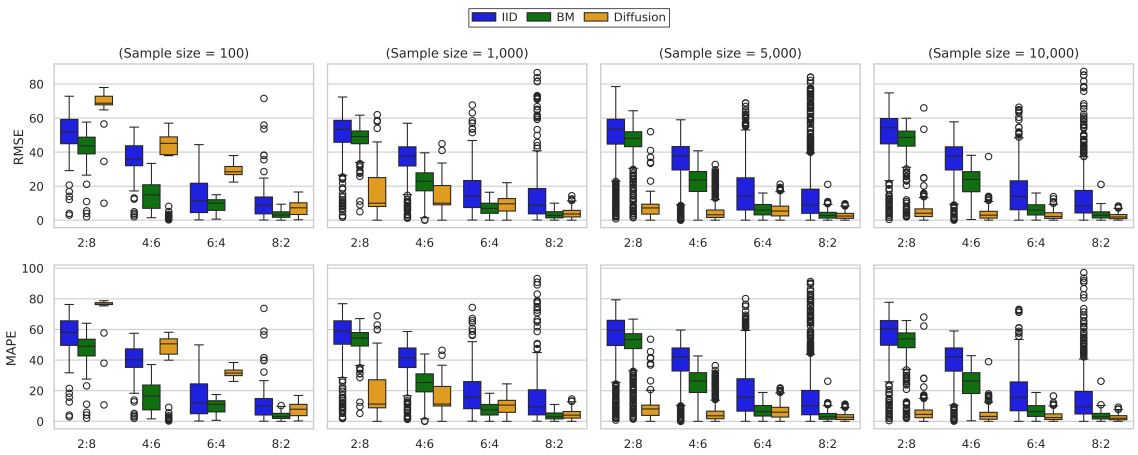


Figure 12: Result of Point 1 with Type 2 data

Except at very small training sizes ($N \leq 100$), the diffusion model outperforms i.i.d. baseline at

nearly all observation ratios for both Type 1 and Type 2; we therefore omit further comparisons. For the BM model with $N < 5000$, the relative performance varies with the observation ratio. In Type 1, BM consistently attains lower RMSE and MAPE across all ratios; the gap is largest at low ratios (2:8) and shrinks as the observed portion and the sample size increase. In Type 2, BM still outperforms diffusion by a large margin at very small training sizes ($N \leq 100$). From $N \geq 1000$, diffusion begins to surpass BM at low–mid ratios (2:8, 4:6), whereas BM retains a slight advantage at higher ratios (6:4, 8:2). At $N = 5000$, diffusion is superior overall in Type 2, although BM still wins in some high-ratio settings; in Type 1, BM remains ahead at 8:2. For $N \geq 10000$, diffusion outperforms BM in nearly all ratios for both types even though, in Type 1, the margins are small and sometimes reverse at 8:2 where the trajectories are close to linear. In Type 2, the advantage of diffusion is pronounced and persistent. Table 1 summarizes the large sample regime. In Type 2, the relative improvement which is defined as $\Delta = \frac{\text{BM}-\text{Diff}}{\text{BM}} \times 100\,\%$ averages about 63% for both RMSE and MAPE, that is, the diffusion model attains roughly 63% lower error than BM across observation ratios and folds. In Type 1, improvements are relatively modest with variability that exceeds the mean, consistent with practical parity and occasional BM advantages at high observation ratios.

Table 1: Aggregate improvements of Diffusion over BM when $N \geq 10{,}000$ (Mean $\pm$ Std)

Setting	ΔRMSE (%)	ΔMAPE (%)
Type 1	10.04 ± 56.89	3.97 ± 68.77
Type 2	63.35 ± 27.08	63.12 ± 27.40

Point 2: Diffusion vs. PCA+Lognormal Regression PCA+lognormal regression performs well in Type 1 and Type 2 for observation ratios between 2:8 and 6:4. In this range, threshold crossings are drift-dominated with approximately homoscedastic variability, so the RUL distribution remains well behaved and a PCA feature set with a lognormal fit is adequate. At 8:2, the ranking reverses: diffusion attains lower RMSE and MAPE in both types because near-threshold crossings are noise-dominated and heteroscedastic, yielding skewed and unstable RUL distributions that a linear PCA feature set cannot well represent. (Figure 13)

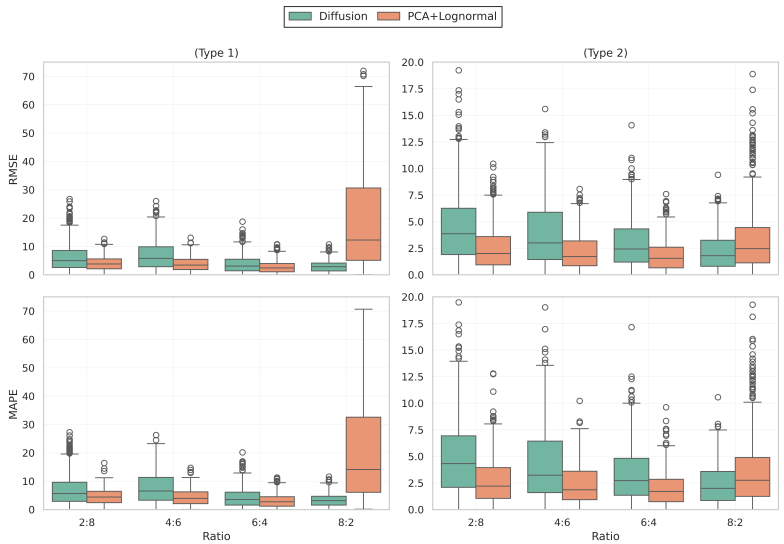


Figure 13: Result of Point 2 with Type 1 and 2

To evaluate robustness under challenging conditions, we test both methods on datasets with rising structural complexity (Complexity 1–4) and multiple operating regimes (Modality 1–4). These prolonged experiments aim to evaluate how each model generalizes to more complex degradation patterns that move beyond smooth single-regime exponential trends and include nonlinear segments, localized trend perturbations, regime switching, and heteroscedastic noise.

- **Complexity 1–4 Results:** For observation ratios between 2:8 and 4:6 (Figure 14), threshold crossings are largely drift-dominated with approximately homoscedastic variability. In this regime, the RUL distribution is still well behaved, and PCA features with a lognormal TTF fit remain competitive across complexity levels. From 6:4, the situation changes: crossings become noise-dominated and heteroscedastic near the threshold, producing skewed and unstable RUL distributions. Diffusion achieves lower RMSE and MAPE in all complexity levels under this condition, as it learns the conditional dynamics near the threshold that are not captured by linear projections with a parametric lognormal fit. Follow-up experiments (Figure 15, 5:5–8:2) confirm that the performance gap widens as the observed portion increases where structural complexity rises.

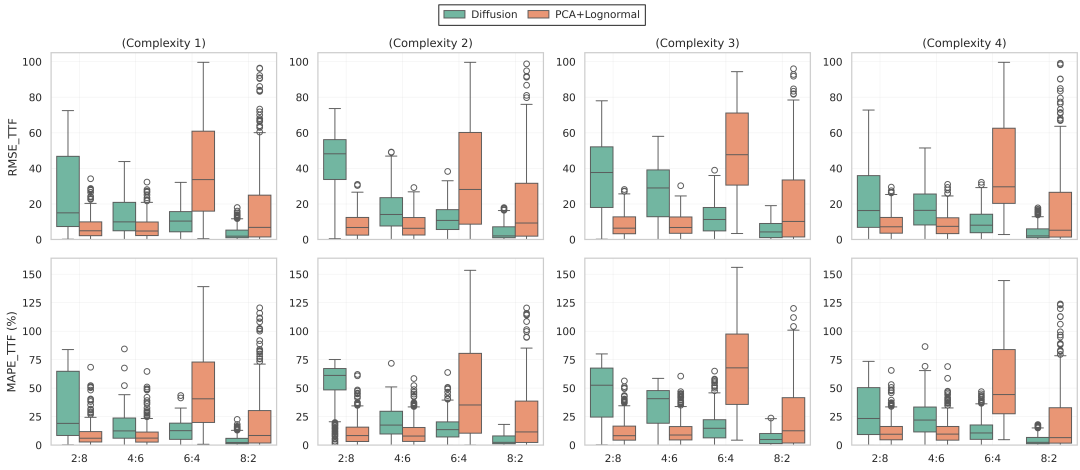


Figure 14: Result of Point 2 with Complexity 1-4 data (2:8 to 8:2)

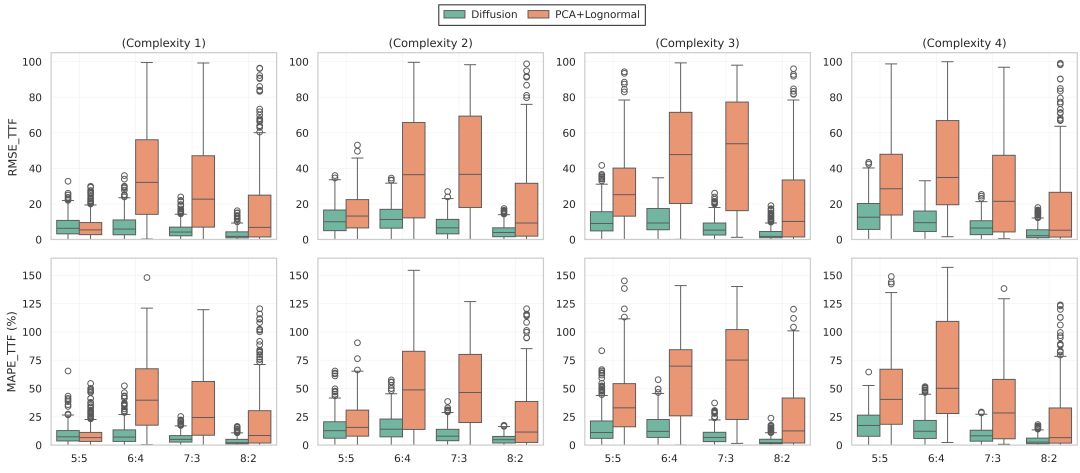


Figure 15: Result of Point 2 with Complexity 1-4 data (5:5 to 8:2)

- **Modality 1–4 Results:** For observation ratios between 2:8 and 8:2 in Modality dataset (Figure 16), threshold crossings are largely *drift-dominated* with homoscedastic variability. In this regime, the RUL distribution remains well behaved, so a linear PCA feature set with a lognormal TTF fit remains competitive across modality levels. In Modality 1, the multi-modality even ends near the 6:4 ratio and this helps the PCA+lognoraml model retains superiority at 6:4. As modality increases (Modality 2–4), regime switching and sensor drift raise structural variance and introduce between-regime mixtures. Even at these mid ratios, PCA+lognormal begins to degrade because linear projections blur regime boundaries and a single lognormal component mis-specifies first-passage distributions that are mixtures or skewed. At 8:2 the mechanism changes: crossings become *near-threshold regime*, yielding skewed and unstable RUL distributions. Under this condition, diffusion attains lower RMSE and MAPE across Modality 1–4 because it learns the conditional dynamics near the threshold and accommodates regime-dependent variability that the lognormal pipeline cannot represent. Follow-up experiments at intermediate ratios (Figure 17, 5:5–8:2) confirm that the performance gap widens as the observed portion increases and modality-induced nonstationarity intensifies.

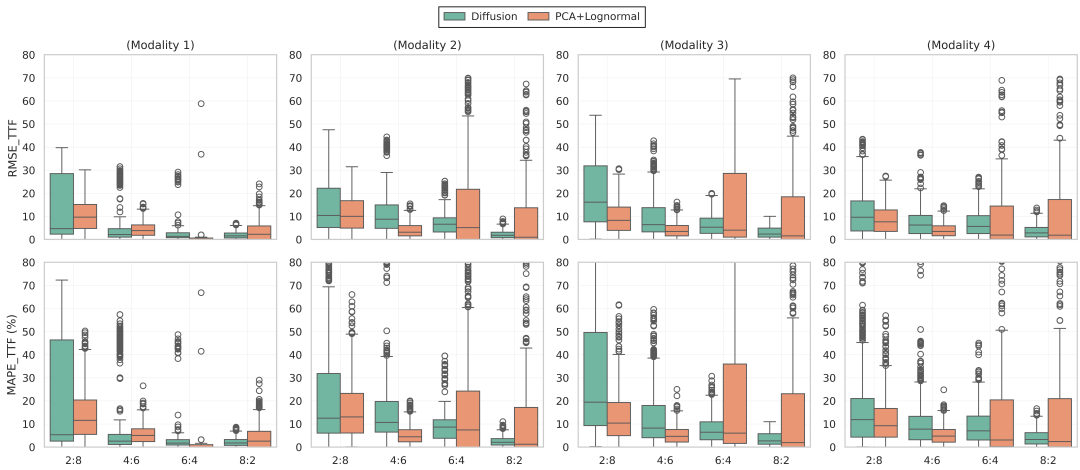


Figure 16: Result of Point2 with Modality 1-4 data. 2:8 to 8:2

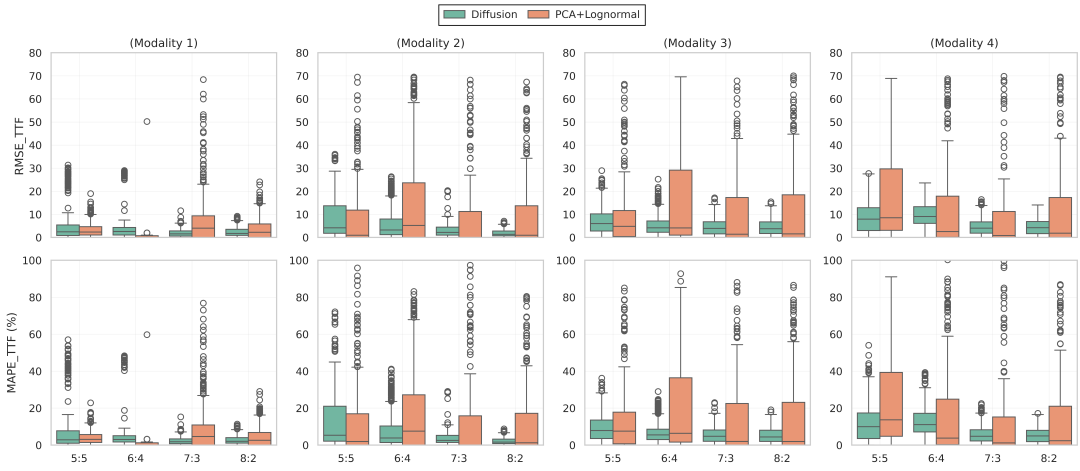


Figure 17: Result of Point2 with Modality 1-4 data. 5:5 to 8:2

In both Type 1 and Type 2, observation ratios from 2:8 to 6:4 are largely drift dominated with near-homoscedastic variability, so the PCA–lognormal baseline remains competitive. Near 8:2, the crossings become noise dominated and heteroscedastic, and diffusion attains lower RMSE and MAPE by modeling the near-threshold dynamics that linear projections with a parametric lognormal fit cannot capture. The same pattern appears in the *Complexity* and *Modality* suites (Fig. 16); a zoom in over 5:5–8:2 confirms that the advantage increases from 6:4 to 8:2 as nonlinear segments, localized perturbations, regime switching, and variance changes intensify (Fig. 17). Quantitatively, Table 2 shows that at 8:2 diffusion achieves about 90% lower error than PCA–lognormal in Type 1 and roughly 32% in Type 2. Across *Complexity* 1–4, the average improvement over 6:4–8:2 is $74.47 \pm 6.17\%$ (RMSE) and $75.94 \pm 6.50\%$ (MAPE), rising to $78.62 \pm 5.09\%$ and $79.14 \pm 5.75\%$ in the 5:5–8:2 zoom in (Table 3). Across *Modality* 1–4, the corresponding gains are $57.21 \pm 19.17\%$ and $56.53 \pm 20.51\%$ for 6:4–8:2, and rising to $59.50 \pm 15.71\%$ and $59.81 \pm 17.54\%$ for the 5:5–8:2 zoom in (Table 4).

Table 2: Improvements of Diffusion over PCA+Lognormal in Type 1-2 (Mean $\pm$ Std)

Setting	ΔRMSE (%)	ΔMAPE (%)
Type 1	90.55 ± 2.10	90.05 ± 2.13
Type 2	33.14 ± 18.51	31.74 ± 19.42

Table 3: Improvements of Diffusion over PCA+Lognormal on Complexity 1–4 (Mean $\pm$ Std)

Setting	ΔRMSE (%)	ΔMAPE (%)
Average at 6:4 and 8:2	74.47 ± 6.17	75.94 ± 6.50
Zoom-in (5:5–8:2)	78.62 ± 5.09	79.14 ± 5.75

Table 4: Improvements of Diffusion over PCA+Lognormal on Modality 1–4 (Mean $\pm$ Std)

Setting	ΔRMSE (%)	ΔMAPE (%)
Average at 6:4 and 8:2	57.21 ± 19.17	56.53 ± 20.51
Zoom-in (5:5–8:2)	59.50 ± 15.71	59.81 ± 17.54

Summary. Across datasets, the diffusion model is the most reliable choice once the training set is moderate to large and the observed segment extends from 6:4 toward 8:2. Gains are largest near 8:2 and in structurally complex settings, as quantified in Tables 1, 2, 3, and 4. All methods improve as more of the trajectory is observed, but diffusion improves more rapidly because it captures nonlinear segments, localized perturbations, regime switching, and heteroscedastic noise. Parametric Bayesian updating and PCA+lognormal remain competitive mainly when trajectories are smooth, single-regime, and low in variance, or when training data are scarce with high observation ratios. These results underscore that predictive accuracy is governed by signal structure and support diffusion-based TTF prediction for assets operating under complex or near-threshold conditions, while retaining classical baselines as lightweight options for simple monotone regimes.

5 Limitations and Further Research

While our results indicate that conditional diffusion is a strong approach for TTF prediction, several limitations remain and need to point out concrete next steps. **First**, our evaluation relies on simulated datasets that emulate a range of trends, complexities, and regime changes. Although these simulations let us precisely control trend shape, stochastic variability, and regime changes, they cannot yet fully reproduce real sensor noise, covariate shift, or operational interventions observed in practice. To enhance external validity, we plan to include real-world benchmarks such as NASA C-MAPSS [26], enabling a more complete assessment of generalizability under realistic operating conditions. **Second**, diffusion-based inference is more computationally demanding than baselines because it requires iterative reverse sampling [31]. To meet real-time constraints, few-step samplers (e.g., DDIM), progressive distillation, and optimized noise schedules reduce the number of reverse steps required at inference [32, 36, 38]. **Third**, the current framework assumes a single-sensor degradation stream, whereas many assets provide multivariate measurements across sensors or subsystems. Extending the model to multi-sensor fusion with spatiotemporal and cross-modal dependencies is therefore an important direction for scalability. **Lastly**, deployment often requires online/streaming inference as new observations arrive. Adapting diffusion models to such settings (e.g., lightweight online fine-tuning) remains open. Methods for federated learning for privacy-preserving deployment [59, 60], and meta-learning or few-shot adaptation [61, 62] provide complementary tooling and baselines for this line of work.

6 Conclusion

We propose a conditional diffusion framework for TTF prediction and benchmark it against Bayesian updating and PCA+lognormal regression across varying synthetic families (Type 1–2, Complexity 1–4, Modality 1–4) and observation ratios from $2{:}8$ to $8{:}2$. Performance improves as training data and observed length increase. With larger training data, diffusion reliably surpasses Bayesian update model; at $N \geq 10{,}000$, it yields approximately 63% lower error in Type 2, while Type 1 is near parity with small, variable margins $4 \sim 10\%$(Table 1). For drift-dominated and near-homoscedastic ratios of $2{:}8 \sim 6{:}4$, PCA+lognormal is competitive; near 8:2 where crossings are near-threshold regime, diffusion wins decisively ($\approx$90% in Type 1; $\approx$32% in Type 2; Table 2). The advantage intensifies under structural complexity and regime mixtures: across *Complexity* 1–4, gains are 74–76% at 6:4–8:2 and 78–79% at 5:5–8:2 (Table 3); across *Modality* 1–4, gains are 56–59% (Table 4). Overall, diffusion is preferred when observation windows are long or when dynamics exhibit nonlinear segments, localized perturbations, regime switching, or heteroscedastic noise; classical baselines remain viable in smooth, single-regime, low-variance settings or with small data and early ratios. As a nonparametric data-driven model, diffusion scales well with large data size and complex structure. Future work will validate on real-world datasets (e.g., CMAPSS), accelerate sampling, and extend to multi-sensor and online settings.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions, which helped improve the quality and clarity of this manuscript.

References

[1] A. Heng, S. Zhang, A. Tan, J. Mathew, "Rotating machinery prognostics: State of the art, challenges and opportunities," *Mechanical Systems and Signal Processing*, 23(3): 724–739, 2009.

[2] A. K. S. Jardine, D. Lin, D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical Systems and Signal Processing*, 20(7): 1483–1510, 2006.

[3] X. Si, W. Wang, C. Hu, D. Zhou, "Remaining useful life estimation – A review on the statistical data driven approaches," *European Journal of Operational Research*, 213(1): 1–14, 2011.

[4] J. Lee, J. Wu, W. Zhao, M. Ghaffari, L. Liao, D. Siegel, "Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications," *Mechanical Systems and Signal Processing*, 42(1–2): 314–334, 2014.

[5] N. Gebraeel, M. Lawley, B. Wang, M. Zhang, "Residual life predictions from vibration-based degradation signals: a Bayesian approach," *IEEE Transactions on Reliability*, 54(3): 539–550, 2005.

[6] X. Fang, J. Shi, "Profile monitoring and prognostics of nonlinear degradation processes: A nonparametric modeling approach," *IIE Transactions*, 44(6): 501–514, 2012.

[7] X. Fang, N. Z. Gebraeel, M. A. Lawley, "A Bayesian degradation modeling approach for prognostics of multiple degradation signals," *Reliability Engineering & System Safety*, 134: Qiu2024331–337, 2015.

[8] X. Fang, N. Z. Gebraeel, "An adaptive approach for prognostics of nonlinear degradation signals," *IEEE Transactions on Reliability*, 66(4): 1207–1216, 2017.

[9] S. He, R. Li, W. Li, "RUL prediction for aircraft engines using a PCA based regression model," *Mechanical Systems and Signal Processing*, 93: 470–484, 2017.

[10] B. Zhang, H. Li, H. Zhang, G. Li, "Aircraft engine prognostics based on informative sensor selection and adaptive degradation modeling with functional principal component analysis," *Sensors*, 20(3): 920, 2020.

[11] N. Li, M. Wang, Y. Lei, X. Li, "Remaining useful life prediction of lithium-ion battery with nonparametric degradation modeling using functional principal component analysis," *Reliability Engineering & System Safety*, 241: 109629, 2024.

[12] S. Nian, W. Kang, Y. Wen, Z. Du, "Health index-based remaining useful life prediction using functional principal component analysis with multivariate sensor data," *Reliability Engineering & System Safety*, 239: 111985, 2025.

[13] M. Vila Forteza, A. C. Capistrán, H. N. Nounou, "Data reduction in proportional hazards models applied to remaining useful life prediction using sparse robust PCA," *Machines*, 13(3): 215, 2025.

[14] X. Li, K. Wolf, J. Reese, "Remaining useful life estimation in prognostics using deep bidirectional LSTM neural network," *Reliability Engineering & System Safety*, 172: 1–12, 2018.

[15] W. Wang, H. Zhang, Z. Wang, "Remaining Useful Life Prediction using Deep Learning: A Survey," *IEEE Access*, 8: 3570–3588, 2020.

[16] Y. Qin, N. Cai, C. Gao, Y. Zhang, Y. Cheng, X. Chen, "Remaining Useful Life Prediction Using Temporal Deep Degradation Network for Complex Machinery with Attention-based Feature Extraction," *arXiv preprint arXiv:2202.10916*, 2022.

[17] H. Hafizhahullah, et al., "A Hybrid CNN–LSTM for Battery Remaining Useful Life Prediction," *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2022.

[18] L. Ma, et al., "Accurate and efficient remaining useful life prediction of lithium-ion batteries by integrating domain knowledge and deep neural networks," *Journal of Energy Storage*, 63: 106807, 2024.

[19] K. A. Severson, P. M. Attia, N. J. Jiang, C. Kurth, J. C. Moore, M. H. Chen, S. J. Krause, M. Z. Bazant, R. D. Braatz, "Data-driven prediction of battery cycle life before capacity degradation," *Nature Energy*, 4: 383–391, 2019.

[20] P. M. Attia, A. Grover, N. J. Jiang, Z. K. Yang, K. A. Severson, P. J. Harris, W. Chueh, S. Ermon, R. D. Braatz, "Closed-loop optimization of fast-charging protocols for batteries with machine learning," *Nature*, 578: 397–402, 2020.

[21] M. Berecibar, I. Gandiaga, I. Villatoro, N. Omar, J. Van Mierlo, P. Van den Bossche, "Critical review of state of health estimation methods of Li-ion batteries for real applications," *Renewable and Sustainable Energy Reviews*, 56: 572–587, 2016.

[22] Z. Hameed, Y. H. Hong, Y. M. Cho, S. H. Ahn, C. K. Shin, "Condition monitoring and fault detection of wind turbines and related algorithms: A review,"*Renewable and Sustainable Energy Reviews*, 13(1): 1–39, 2009.

[23] J. Tautz-Weinert, S. J. Watson, "Using SCADA data for wind turbine condition monitoring: A review," *Renewable and Sustainable Energy Reviews*, 81: 1965–1975, 2018.

[24] A. Kusiak, A. Verma, "A data-mining approach to monitoring wind turbines," *IEEE Transactions on Energy Conversion*, 27(1): 136–144, 2012.

[25] A. Saxena, K. Goebel, D. Simon, N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Proc. IEEE Int. Conf. on Prognostics and Health Management*, pp. 1–9, 2008.

[26] A. Saxena and K. Goebel, "Turbofan Engine Degradation Simulation Data Set (CMAPSS)," *NASA Ames Prognostics Data Repository (PHM08 Challenge Data)*, 2008.

[27] D. P. Kingma, M. Welling, "Auto-encoding variational Bayes," *International Conference on Learning Representations (ICLR)*, 2014.

[28] C. Esteban, S. L. Hyland, G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional GANs," *arXiv preprint* arXiv:1706.02633, 2017.

[29] Z. Che, Y. Li, C. Quirk, X. Li, Y. Liu, "Time series generative adversarial networks," *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[30] K. Rasul, C. Seward, I. Schuster, R. Vollgraf, "Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting," *Proceedings of the 38th International Conference on Machine Learning (ICML)*, vol. 139, pp. 8857–8868, 2021.

[31] J. Ho, A. Jain, P. Abbeel, "Denoising Diffusion Probabilistic Models," *NeurIPS*, 33, 2020.

[32] J. Song, C. Meng, S. Ermon, "Denoising Diffusion Implicit Models," arXiv:2010.02502, 2020.

[33] Y. Song, S. Ermon, "Score-Based Generative Modeling through Stochastic Differential Equations," *ICLR*, 2021.

[34] A. Q. Nichol, P. Dhariwal, "Improved Denoising Diffusion Probabilistic Models," *ICML*, 2021.

[35] D. P. Kingma, T. Salimans, B. Poole, J. Ho, "Variational Diffusion Models," *NeurIPS*, 2021.

[36] T. Karras, M. Aittala, S. Laine, E. Veach, J. Lehtinen, T. Aila, "Elucidating the Design Space of Diffusion-Based Generative Models," *NeurIPS*, 2022.

[37] J. Ho, T. Salimans, "Classifier-Free Diffusion Guidance," arXiv:2207.12598, 2022.

[38] T. Salimans, J. Ho, "Progressive Distillation for Fast Sampling of Diffusion Models," *ICLR*, 2022.

[39] Y. Tashiro, A. Gritsenko, S. Ermon, "CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation," arXiv:2107.03502, 2021.

[40] K. Rasul, J. Schröter, I. Valera, "Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting," arXiv:2107.06615, 2021.

[41] P. Dhariwal, A. Nichol, "Diffusion Models Beat GANs on Image Synthesis," *NeurIPS*, 2021.

[42] C. E. Rasmussen, C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[43] M. A. Álvarez, L. Rosasco, N. D. Lawrence, "Kernels for Vector-Valued Functions: A Review," *Foundations and Trends in Machine Learning*, 4(3): 195–266, 2012.

[44] V. Fortuin, H. Salimbeni, F. Wenzel, J. P. Cunningham, S. Mandt, "GP-Score: Gaussian Process Regression for Scalable Amortized Inference in Score-Based Generative Models," arXiv:2202.00512, 2022.

[45] O. Ronneberger, P. Fischer, T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *MICCAI*, 2015.

[46] K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition," *CVPR*, 2016.

[47] Y. Wu, K. He, "Group Normalization," *ECCV*, 2018.

[48] P. Ramachandran, B. Zoph, Q. V. Le, "Searching for Activation Functions," arXiv:1710.05941, 2017.

[49] A. Vaswani *et al.*, "Attention is All You Need," *NeurIPS*, 2017.

[50] A. Dosovitskiy *et al.*, "An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale," *ICLR*, 2021.

[51] E. Perez, F. Strub, H. de Vries, V. Dumoulin, A. Courville, "FiLM: Visual Reasoning with a General Conditioning Layer," *AAAI*, 2018.

[52] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, R. Ng, "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains," *NeurIPS*, 2020.

[53] L. Tang, A. Che, "RUL Prediction Using Conditional Score-based Diffusion Models," *Proc. PHM Society Conference*, 15(1), 2023.

[54] H. Wen, Y. Wu, Y. Zhao, X. Xu, "Diffusion Models for Remaining Useful Life Prediction with Uncertainty Quantification," arXiv:2402.01971, 2024.

[55] C. Zhang, Y. Zhao, H. Wen, X. Xu, Y. Wu, "Remaining useful life prediction using class-conditional diffusion models," in *Proc. PHM Society Conference*, 15(1): 1–9, 2023.

[56] Z. Li, Y. Xiao, J. Chen, J. Wang, "Bayesian score-based diffusion models for uncertainty-aware remaining useful life prediction," arXiv:2310.02085, 2023.

[57] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pp. 2623–2631, 2019.

[58] J. Su, Z. Lu, H. Pan, "RoFormer: Enhanced Transformer with Rotary Position Embedding," arXiv:2104.09864, 2021.

[59] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. AISTATS*, 2017.

[60] P. Kairouz *et al.*, "Advances and Open Problems in Federated Learning," *Foundations and Trends in Machine Learning*, 14(1–2): 1–210, 2021.

[61] C. Finn, P. Abbeel, S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," in *Proc. ICML*, 2017.

[62] J. Snell, K. Swersky, R. Zemel, "Prototypical Networks for Few-shot Learning," in *Proc. NeurIPS*, 2017.

A Bayesian Updating

In this appendix, we provide the full mathematical derivation of Bayesian updating and closed-form RUL distribution under both Exponential-IID and Exponential-Brownian Motion (BM) degradation models, following the framework of [5].

A.1 Exponential-IID Model

Prior and Likelihood

The degradation signal is modeled as:

$$S(t) = \phi + \theta \exp\left(\beta t + \epsilon(t) - \frac{\sigma^2}{2}\right), \quad \epsilon(t) \sim \mathcal{N}(0, \sigma^2)$$

Log-transformed signal:

$$L(t_i) = \ln(S(t_i) - \phi) = \theta' + \beta t_i + \epsilon(t_i), \quad \theta' = \ln\theta$$

Likelihood of observations $\{L(t_1), \ldots, L(t_k)\}$:

$$f(L_1, \ldots, L_k \mid \theta', \beta) = \prod_{i=1}^{k} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(L(t_i) - \theta' - \beta t_i)^2}{2\sigma^2}\right)$$

Posterior Update

Assuming Gaussian priors:

$$\theta' \sim \mathcal{N}(\mu_\theta, \sigma_\theta^2), \quad \beta \sim \mathcal{N}(\mu_\beta, \sigma_\beta^2)$$

The posterior becomes:

$$(\theta', \beta) \mid L_1, \ldots, L_k \sim \mathcal{N}\left(\begin{bmatrix}\mu_{\theta'|post} \\ \mu_{\beta|post}\end{bmatrix}, \Sigma_{post}\right), \text{ where}$$

Σ_{post} is analytically derived via conjugate prior updates.

RUL Distribution

Failure condition:

$$L(t_k + T) = \ln(D - \phi)$$

Thus:

$$T \sim \Phi\left(\frac{\mu_{\theta'|post} + \mu_{\beta|post}(t_k + T) - \ln(D - \phi)}{\sigma_L(t_k + T)}\right)$$

where:

$$\sigma_L(t_k + T) = \sqrt{\sigma_{\theta'}^2 + (t_k + T)^2\sigma_\beta^2 + 2\rho(t_k + T)\sigma_{\theta'}\sigma_\beta}$$

A.2 Exponential-Brownian Motion (BM) Model

Prior and Likelihood

The degradation signal is modeled as:

$$S(t) = \phi + \theta \exp\left(\beta t + \sigma W(t) - \frac{\sigma^2 t}{2}\right)$$

Log-transformed signal:

$$L(t_i) = \ln(S(t_i) - \phi) = \theta' + \beta t_i + \sigma W(t_i) - \frac{\sigma^2 t_i}{2}$$

Define:

$$\epsilon(t_i) = \sigma W(t_i) - \frac{\sigma^2 t_i}{2}, \quad \epsilon(t_i) \sim \mathcal{N}(0, \sigma^2 t_i)$$

Thus:

$$L(t_i) = \theta' + \beta t_i + \epsilon(t_i)$$

Posterior Update

The priors remain Gaussian. The posterior is updated considering time-varying noise variance $\sigma^2 t_i$:

$$(\theta', \beta) \mid L_1, \ldots, L_k \sim \mathcal{N}\left(\begin{bmatrix}\mu_{\theta'|post} \\ \mu_{\beta|post}\end{bmatrix}, \Sigma_{post}(t)\right)$$

RUL Distribution

Failure condition:

$$L(t_k + T) = \ln(D - \phi)$$

RUL CDF:

$$T \sim \Phi\left(\frac{\mu_{\theta'|post} + \mu_{\beta|post}(t_k + T) - \ln(D - \phi)}{\sigma_L(t_k + T)}\right)$$

with:

$$\sigma_L^2(t_k + T) = \sigma_{\theta'}^2 + (t_k + T)^2 \sigma_\beta^2 + 2\rho(t_k + T)\sigma_{\theta'}\sigma_\beta + \sigma^2(t_k + T)$$

Both IID and BM models follow similar Bayesian updating procedures, with the key difference being the noise structure:

- IID: constant noise σ^2
- BM: time-varying noise $\sigma^2 t_i$

B PCA + Lognormal Regression

In this appendix, we provide a detailed derivation of the PCA + lognormal regression method for the prediction of RUL, following the framework of [9].

Step 1: PCA in training data

- Apply PCA on X_{train}:
$$X_{\text{obs}}^{\text{PCA}} = \Phi^\top X_{\text{obs}}$$
- Select number of components explaining 95% of total variance.
- The truncation here refers to retaining only the d-dimensional PCA space:
$$X_{\text{obs}}^{\text{PCA}} \in \mathbb{R}^d$$

Step 2: Lognormal regression

- Log-normal regression fit:
$$\ln(\text{RUL}) = w^\top X_{\text{obs}}^{\text{PCA}} + b + \eta, \quad \eta \sim \mathcal{N}(0, \sigma^2)$$

Step 3: Prediction on Test Data

- Project X_{test} to PCA space:
$$X_{\text{test}}^{\text{PCA}} = \Phi^\top X_{\text{test}}$$
- Predict:
$$\ln(\text{RUL}) \sim \mathcal{N}(w^\top X_{\text{test}}^{\text{PCA}} + b, \sigma^2)$$
- Back-transform:
$$\text{RUL} \sim \text{LogNormal}(\mu, \sigma^2)$$

Step 4: Sampling and RUL Distribution

- Draw M samples of RUL from the predictive distribution.
- Compute:
$$\hat{T} = \text{mean/median(RUL samples)}$$

C Diffusion Model

C.1 Reparameterized marginal form expression derivation: $q(x_t \mid x_0)$

We now derive the exact marginal distribution of x_t given the initial x_0. Define again $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. We claim that

$$\boxed{q(x_t \mid x_0) = \mathcal{N}\big(x_t; \sqrt{\bar{\alpha}_t}\, x_0,\, (1 - \bar{\alpha}_t)\, I\big)}$$

In other words, one can sample x_t directly from x_0 in one step as

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

Below is a short proof by induction.

Proof by induction

- **Base case ($t = 1$).** By definition,

  $$q(x_1 \mid x_0) = \mathcal{N}(x_1; \sqrt{1 - \beta_1}\, x_0,\ \beta_1 I) = \mathcal{N}\big(x_1; \sqrt{\alpha_1}\, x_0,\ (1 - \alpha_1) I\big)$$

  and indeed $\bar{\alpha}_1 = \alpha_1$.

- **Inductive step.** Suppose at time $t - 1$ we have

  $$q(x_{t-1} \mid x_0) = \mathcal{N}\big(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\, x_0,\ (1 - \bar{\alpha}_{t-1})\, I\big)$$

  Then

  $$\begin{aligned} q(x_t \mid x_0) &= \int q(x_t \mid x_{t-1})\, q(x_{t-1} \mid x_0)\, dx_{t-1} \\ &= \int \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1},\ \beta_t I)\, \mathcal{N}\big(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\, x_0,\ (1 - \bar{\alpha}_{t-1}) I\big)\, dx_{t-1} \end{aligned}$$

  Combining two Gaussians in the integral yields a Gaussian with mean and variance:

  $$\mathbb{E}[x_t] = \sqrt{\alpha_t}\, \mathbb{E}[x_{t-1}] = \sqrt{\alpha_t}\, (\sqrt{\bar{\alpha}_{t-1}}\, x_0) = \sqrt{\alpha_t\, \bar{\alpha}_{t-1}}\, x_0 = \sqrt{\bar{\alpha}_t}\, x_0$$

  $$\mathrm{Var}(x_t) = \alpha_t\, \mathrm{Var}(x_{t-1}) + \beta_t I = \alpha_t \big((1 - \bar{\alpha}_{t-1}) I\big) + \beta_t I = \big(\alpha_t (1 - \bar{\alpha}_{t-1}) + \beta_t\big)\, I$$

  Since $\alpha_t\, \bar{\alpha}_{t-1} = \bar{\alpha}_t$, one checks

  $$\alpha_t (1 - \bar{\alpha}_{t-1}) + \beta_t = \alpha_t - \alpha_t\, \bar{\alpha}_{t-1} + \beta_t = 1 - \bar{\alpha}_t$$

  Therefore, $q(x_t \mid x_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}\, x_0,\ (1 - \bar{\alpha}_t) I\big)$

This closed-form expression is extremely useful because it allows us to sample x_t in one shot given x_0, without iterating all the intermediate steps.

C.2 Deriving the Posterior $q(x_{t-1} \mid x_t, x_0)$ closed form

It is intractable to obtain a closed form solution for the reverse process $q(x_{t-1} \mid x_t)$, because it requires integrating the entire distribution of x_0 conditioned on x_t, which is generally intractable due to the complex and unknown nature of the data distribution $p_{\text{data}}(x_0)$. However, the good news is that $q(x_{t-1} \mid x_t, x_0)$ is tractable, which we can use to mathematically define the objective(loss) function to train the diffusion model.

Mathematical derivation. Using the Markov property of the reverse step, we can replace this intractable marginal posterior with a tractable conditional Gaussian that depends on the original sample x_0. Specifically:

- $q(x_{t-1} \mid x_t)$ **(intractable):** Using the conditional marginalization formula,

$$q(x_{t-1} \mid x_t) = \int q(x_{t-1}, x_0 \mid x_t)\,\mathrm{d}x_0 = \int q(x_{t-1} \mid x_t, x_0)\, q(x_0 \mid x_t)\,\mathrm{d}x_0,$$

  where $q(x_0 \mid x_t) = \frac{q(x_t|x_0)\, p_{\text{data}}(x_0)}{q(x_t)}$ is intractable due to unknown p_{data}, so we cannot evaluate this integral directly.

- $q(x_{t-1} \mid x_t, x_0)$ **(tractable):** By Bayes' rule on the forward process (using the Markov property and reparameterization trick),

$$q(x_{t-1} \mid x_t, x_0) = \frac{q(x_t \mid x_{t-1})\, q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)}, \text{ where}$$
$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}\, x_{t-1}, \beta_t I)$$
$$q(x_{t-1} \mid x_0) = \mathcal{N}(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\, x_0, (1-\bar{\alpha}_{t-1}) I)$$
$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}\, x_0, (1-\bar{\alpha}_t) I)$$

  Completing the square yields the known Gaussian form such that:

$$\boxed{q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\big(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I\big)}$$

  with

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t$$
$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\, x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, x_t$$

  In practice, we often rewrite $\tilde{\mu}_t(x_t, x_0)$ in terms of the actual noise ϵ_t that was used to sample x_t from x_0. Recall from closed-form forward sampling:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I)$$

  Solving for ϵ_t:

$$\epsilon_t = \frac{x_t - \sqrt{\bar{\alpha}_t}\, x_0}{\sqrt{1-\bar{\alpha}_t}}$$

  One can then verify that

$$\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}\big(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t\big)$$

  This alternate form will be crucial when we parameterize the mean of our model to predict ϵ_t directly in the reverse process.

Key insight (approximation). Building on the closed-form conditional posterior $q(x_{t-1} \mid x_t, x_0)$, we approximate the intractable marginal $q(x_{t-1} \mid x_t)$ by a Gaussian distribution whose parameters are learned by a neural network. Concretely, we define

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t),\, \sigma_\theta^2(x_t, t)\, I\big)$$

train μ_θ and σ_θ by minimizing the KL divergence;

$$\mathrm{KL}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\big)$$

so that

$$p_\theta(x_{t-1} \mid x_t) \;\approx\; q(x_{t-1} \mid x_t)$$

by matching them(μ_θ and σ_θ) to the tractable $\tilde{\mu}_t(x_t, x_0)$ and $\tilde{\beta}_t$ derived above. In practice, this means that the network learns to undo each forward noising step iteratively, denoising x_t one timestep at a time.

C.3 How to Define an Objective(Loss) Function

To derive a tractable objective function for training the diffusion model, we apply variational inference techniques based on Jensen's inequality. Begin with the forward joint distribution:

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}),$$

and the reverse generative model:

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

Then, the marginal likelihood becomes:

$$p_\theta(x_0) = \int q(x_{1:T} \mid x_0) \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)}\, dx_{1:T}$$

By taking log on both sides,

$$\begin{aligned}
\log p_\theta(x_0) &= \log \left[\int q(x_{1:T} \mid x_0) \cdot \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)}\, dx_{1:T} \right] \\
&= \log \left[\mathbb{E}_{q(x_{1:T}|x_0)} \left(\frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right) \right] \\
&\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \left(\frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right) \right] \quad \text{(by Jensen's Inequality)} \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0) \right]
\end{aligned}$$

$$\boxed{\log p_\theta(x_0) \;\geq\; \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)} \Big[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0) \Big]}_{\mathcal{L}(\theta; x_0)\ \text{(ELBO)}}} \tag{3}$$

In another sense,

$$
\begin{aligned}
\log p_\theta(x_0) &\geq \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log\left(\frac{p_\theta(x_0)\cdot p_\theta(x_{1:T} \mid x_0)}{q(x_{1:T} \mid x_0)}\right)\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log p_\theta(x_0) - \log\left(\frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{1:T} \mid x_0)}\right)\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}[\log p_\theta(x_0)] - \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log\left(\frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{1:T} \mid x_0)}\right)\right] \\
&= \log p_\theta(x_0) - \int \log\left(\frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{1:T} \mid x_0)}\right)\cdot q(x_{1:T} \mid x_0)\, dx_{1:T} \quad \text{(by KL-divergence)} \\
&= \log p_\theta(x_0) - D_{\mathrm{KL}}\left(q(x_{1:T} \mid x_0) \,\|\, p_\theta(x_{1:T} \mid x_0)\right)
\end{aligned}
$$

$$
\boxed{\log p_\theta(x_0) \;\geq\; \log p_\theta(x_0) - D_{\mathrm{KL}}\Big(\underbrace{q(x_{1:T} \mid x_0)}_{\substack{\text{True forward}\\ \text{process}}} \;\Big\|\; \underbrace{p_\theta(x_{1:T} \mid x_0)}_{\substack{\text{Modeled forward}\\ \text{process}}}\Big)} \tag{4}
$$

By combining 3, 4, and the definition of KL divergence, we have the exact identity s.t:

$$
\log p_\theta(x_0) = \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)}\big[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0)\big]}_{\mathcal{L}(\theta;x_0)} \;+\; \underbrace{D_{\mathrm{KL}}\big(q(x_{1:T} \mid x_0) \,\|\, p_\theta(x_{1:T} \mid x_0)\big)}_{\geq 0}
$$

Taking the negative on both sides and using $D_{\mathrm{KL}} \geq 0$ gives

$$
-\log p_\theta(x_0) \;\leq\; -\mathcal{L}(\theta; x_0) \;+\; D_{\mathrm{KL}}\big(q(x_{1:T} \mid x_0) \,\|\, p_\theta(x_{1:T} \mid x_0)\big)
$$

so that, in practice, we optimize the tractable surrogate negative ELBO $[-\mathcal{L}(\theta; x_0)]$ as a proxy to minimize the true negative log-likelihood of $p_\theta(x_0)$. In diffusion model, the ELBO further decomposes into a sum of timestep-wise KL terms:

$$
-\mathcal{L}(\theta; x_0) = \underbrace{D_{\mathrm{KL}}\big(q(x_T \mid x_0) \,\|\, p_\theta(x_T)\big)}_{\text{prior matching}} \;+\; \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)}\Big[D_{\mathrm{KL}}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\big)\Big]
$$

By minimizing each timestep's KL divergence individually, we teach the model's denoising kernel $p_\theta(x_{t-1} \mid x_t)$ to match the true conditional $q(x_{t-1} \mid x_t, x_0)$ at every step. Although this procedure does not guarantee a perfect match across the entire sequence at once, reducing the error step by step results in an overall close alignment of the full reverse process. Because time-step decomposition makes the KL of each reverse step tractable, minimizing $-\mathcal{L}(\theta; x_0)$ serves as an effective and computationally feasible proxy to reduce the true negative logarithmic likelihood of $p_\theta(x_0)$.

C.4 ELBO Decomposition and KL-Based Training Objective

In the previous subsection, we introduced the Evidence Lower Bound (ELBO) as a principal surrogate objective to circumvent the intractability of directly maximizing the marginal log-likelihood $\log p_\theta(x_0)$. By Jensen's inequality, we have:

$$\log p_\theta(x_0) \geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0)\right],$$

which justifies using the ELBO as a lower bound on $\log p_\theta(x_0)$ for model training. To turn this abstract formulation into a practical training objective, we now expand and decompose each term of the ELBO based on the known structure of the forward and reverse processes in diffusion models. The forward process q is a fixed Markov chain that progressively adds noise, while the reverse process p_θ is parameterized by a neural network trained to remove noise. Recall the generative and inference processes:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t), \quad q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}),$$

where q is the fixed forward process and p_θ is the learnable reverse denoising process. Substituting these into the ELBO gives:

$$\begin{aligned}
\mathrm{ELBO}(x_0;\theta) &= \mathbb{E}_{q(x_{1:T}|x_0)}\Big[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0)\Big] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log p(x_T) + \sum_{t=1}^{T} \log p_\theta(x_{t-1} \mid x_t) - \sum_{t=1}^{T} \log q(x_t \mid x_{t-1})\right] \\
&= \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)}[\log p(x_T)]}_{\text{prior term}} + \sum_{t=1}^{T} \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)}[\log p_\theta(x_{t-1} \mid x_t)]}_{\text{reverse transitions}} \\
&\quad - \sum_{t=1}^{T} \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)}[\log q(x_t \mid x_{t-1})]}_{\text{forward transitions}}
\end{aligned}$$

The prior term $\mathbb{E}_{q(x_{1:T}|x_0)}[\log p(x_T)]$ in the ELBO does not depend on the model parameters θ. As a result, when we take the gradient of the ELBO with respect to θ during optimization, this term contributes nothing—the gradient is zero. Therefore, it can be safely omitted from the loss function used in training. In practice, only the terms involving θ, such as the reverse and forward transition terms, are retained and optimized. To simplify, consider the term

$$\mathbb{E}_{q(x_{1:T}|x_0)}\big[\log q(x_t \mid x_{t-1}) - \log p_\theta(x_{t-1} \mid x_t)\big]$$

Using the law of total expectation with the Markov property, we can rewrite ELBO term as:

$$\mathbb{E}_{q(x_t|x_0)}\left[\mathbb{E}_{q(x_{t-1}|x_t,x_0)}\left[\log q(x_t \mid x_{t-1}) - \log p_\theta(x_{t-1} \mid x_t)\right]\right]$$

Equivalently, in integral form:

$$\begin{aligned}
&= \int q(x_t \mid x_0) \left[\int q(x_{t-1} \mid x_t, x_0) \left(\log q(x_t \mid x_{t-1}) - \log p_\theta(x_{t-1} \mid x_t)\right) dx_{t-1}\right] dx_t \\
&= \int q(x_t \mid x_0) \left[\int q(x_{t-1} \mid x_t, x_0) \log \frac{q(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)} dx_{t-1}\right] dx_t \\
&= \mathbb{E}_{q(x_t|x_0)}\left[D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t))\right]
\end{aligned}$$

We now elaborate on the transition from the expression:

$$\int q(x_t \mid x_0) \left[\int q(x_{t-1} \mid x_t, x_0) \left(\log q(x_t \mid x_{t-1}) - \log p_\theta(x_{t-1} \mid x_t) \right) dx_{t-1} \right] dx_t$$

to the KL divergence form:

$$\int q(x_t \mid x_0) \left[\int q(x_{t-1} \mid x_t, x_0) \log \frac{q(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)} dx_{t-1} \right] dx_t$$

The key step involves replacing the term $\log q(x_t \mid x_{t-1})$ with an expression involving $q(x_{t-1} \mid x_t, x_0)$. To see why this is possible, consider the identity of the joint distribution under the forward process:

$$q(x_t \mid x_{t-1}) q(x_{t-1} \mid x_0) = q(x_t, x_{t-1} \mid x_0) = q(x_{t-1} \mid x_t, x_0) q(x_t \mid x_0)$$

Taking the logarithm on both sides yields:

$$\log q(x_t \mid x_{t-1}) = \log \frac{q(x_{t-1} \mid x_t, x_0) q(x_t \mid x_0)}{q(x_{t-1} \mid x_0)}$$

This can be rearranged as:

$$\log q(x_t \mid x_{t-1}) = \log q(x_{t-1} \mid x_t, x_0) + \log q(x_t \mid x_0) - \log q(x_{t-1} \mid x_0)$$

Now consider plugging this into the earlier expectation:

$$\int q(x_t \mid x_0) \left[\int q(x_{t-1} \mid x_t, x_0) \Big(\log q(x_{t-1} \mid x_t, x_0) + \log q(x_t \mid x_0) - \log q(x_{t-1} \mid x_0) - \log p_\theta(x_{t-1} \mid x_t) \Big) dx_{t-1} \right] dx_t$$

Observe that both $\log q(x_t \mid x_0)$ and $\log q(x_{t-1} \mid x_0)$ are independent of x_{t-1} and can be pulled out of the inner integral. Furthermore, since they do not depend on θ, they vanish when computing gradients with respect to θ, and thus can be safely ignored in optimization. This simplifies the expression to:

$$\int q(x_t \mid x_0) \left[\int q(x_{t-1} \mid x_t, x_0) \log \frac{q(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)} dx_{t-1} \right] dx_t$$

Finally, we identify this expression as the expected KL divergence:

$$\mathbb{E}_{q(x_t \mid x_0)} \left[D_{\mathrm{KL}} \left(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t) \right) \right]$$

The KL divergence formulation derived earlier becomes a building block for constructing the training loss. Specifically, we define a per-timestep loss function L_t for each time step $t = 1, \ldots, T-1$ as follows:

$$L_t := \mathbb{E}_{q(x_t \mid x_0)} \left[D_{\mathrm{KL}} (q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)) \right]$$

This term measures how well the learned reverse transition distribution $p_\theta(x_{t-1} \mid x_t)$ approximates the true posterior $q(x_{t-1} \mid x_t, x_0)$ for each denoising step in the diffusion process. The expectation over $q(x_t \mid x_0)$ ensures that this comparison is averaged over possible noisy inputs.

At the final diffusion step $t = T$, the forward marginal distribution $q(x_T \mid x_0)$ and the prior $p(x_T)$ are both Gaussian, and their KL divergence can be computed in closed form:

$$q(x_T \mid x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_T}x_0, (1-\bar{\alpha}_T)I), \quad L_T := D_{\mathrm{KL}}(q(x_T \mid x_0) \,\|\, p(x_T))$$

This term quantifies how far the final noised sample x_T deviates from the assumed prior, and since both distributions are Gaussian, L_T is often easy to precompute or implement analytically. At the starting point of the reverse process $t = 0$, our aim is to reconstruct the clean data x_0 from the first denoised latent x_1. This is expressed as a negative log-likelihood:

$$L_0 := -\mathbb{E}_{q(x_1|x_0)} \left[\log p_\theta(x_0 \mid x_1)\right]$$

Unlike the other L_t terms, L_0 is not a KL divergence—it directly measures how well the model can generate x_0 given x_1. In practice, this is often approximated by a simple Mean Squared Error (MSE) loss between x_0 and its predicted reconstruction. Therefore, the original ELBO can be decomposed like the following box, and this decomposition enables modular training, allowing approximations (e.g., skip L_0), surrogate losses, and per-timestep control over reverse process learning.

Negative ELBO as Training Objective:

$$-\mathrm{ELBO}(x_0;\theta) = L_T + \sum_{t=1}^{T-1} L_t + L_0$$

C.5 Closed-Form of KL Terms

We now derive an explicit expression for each intermediate KL divergence term that appears in the objective(loss) function to train a diffusion model:

$$L_t = \mathbb{E}_{q(x_t|x_0)} \left[D_{\mathrm{KL}} \left(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\right)\right], \quad 1 \le t \le T-1$$

Both the true posterior $q(x_{t-1} \mid x_t, x_0)$ and the model $p_\theta(x_{t-1} \mid x_t)$ are assumed to be multivariate Gaussians with diagonal (often isotropic) covariance matrices. Therefore, we can use the closed form formula for the KL divergence between two Gaussians.

Gaussian-to-Gaussian KL Formula

The KL divergence between two multivariate Gaussian distributions in $\mathbb{R}^k$, each with diagonal covariance, is given by:

$$D_{\mathrm{KL}} \left(\mathcal{N}(\mu_1, \sigma_1^2 I) \,\|\, \mathcal{N}(\mu_2, \sigma_2^2 I)\right) = \frac{1}{2}\left(\frac{\sigma_1^2}{\sigma_2^2} + \frac{\|\mu_1 - \mu_2\|^2}{\sigma_2^2} - k + k\log\frac{\sigma_2^2}{\sigma_1^2}\right)$$

This expression compares two Gaussian distributions:

- $\mathcal{N}(\mu_1, \sigma_1^2 I)$: the **true posterior** $q(x_{t-1} \mid x_t, x_0)$
- $\mathcal{N}(\mu_2, \sigma_2^2 I)$: the **learned reverse model** $p_\theta(x_{t-1} \mid x_t)$

This closed-form is particularly useful in diffusion models, where the mean and variance of both $q(x_{t-1} \mid x_t, x_0)$ and $p_\theta(x_{t-1} \mid x_t)$ are explicitly available. This allows us to compute the KL divergence for each timestep, without sampling or numerical approximation. Moreover, since all involved distributions are Gaussian, the entire loss can be evaluated analytically. Using the known forms:

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I), \quad p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

The KL divergence is then:

$$D_{\mathrm{KL}}(q \,\|\, p_\theta) = \frac{1}{2}\left[\mathrm{tr}(\Sigma_\theta^{-1}\tilde{\beta}_t I) + (\mu_\theta - \tilde{\mu}_t)^\top \Sigma_\theta^{-1}(\mu_\theta - \tilde{\mu}_t) - k + \ln \frac{\det \Sigma_\theta}{(\tilde{\beta}_t)^k}\right]$$

In practice, we often set $\Sigma_\theta(x_t, t) = \tilde{\beta}_t I$. Then:

$$\mathrm{tr}((\tilde{\beta}_t I)^{-1}\tilde{\beta}_t I) = k, \quad \ln \frac{\det(\tilde{\beta}_t I)}{(\tilde{\beta}_t)^k} = 0$$

The KL divergence simplifies to:

$$D_{\mathrm{KL}} = \frac{1}{2\tilde{\beta}_t}\|\mu_\theta(x_t, t) - \tilde{\mu}_t(x_t, x_0)\|^2$$

Thus,

$$\boxed{L_t \approx \mathbb{E}_{q(x_t|x_0)}\left[\frac{1}{2\tilde{\beta}_t}\|\mu_\theta(x_t, t) - \tilde{\mu}_t(x_t, x_0)\|^2\right] + \mathbf{C}}$$

We can also rewrite $\tilde{\mu}_t$ in Terms of Noise by using the reparameterization trick:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I)$$

$$\epsilon_t = \frac{x_t - \sqrt{\bar{\alpha}_t}x_0}{\sqrt{1-\bar{\alpha}_t}}$$

$$\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}(x_t - \sqrt{1-\alpha_t}\,\epsilon_t)$$

We now parameterize:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \sqrt{1-\alpha_t}\,\epsilon_\theta(x_t, t))$$

Thus,

$$\mu_\theta - \tilde{\mu}_t = -\frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}(\epsilon_\theta - \epsilon_t)$$

$$\|\mu_\theta - \tilde{\mu}_t\|^2 = \frac{1-\alpha_t}{\alpha_t}\|\epsilon_\theta - \epsilon_t\|^2$$

Hence,

$$\boxed{\frac{1}{2\tilde{\beta}_t}\|\mu_\theta - \tilde{\mu}_t\|^2 \propto \|\epsilon_\theta - \epsilon_t\|^2}$$

Putting all the derivations together, we arrive at a much simpler training objective by assuming that the model variance $\Sigma_\theta(x_t, t)$ is fixed and equal to the posterior variance $\tilde{\beta}_t I$. In this setting, the KL divergence at each timestep reduces to a weighted squared error between the predicted mean $\mu_\theta(x_t, t)$ and the posterior mean $\tilde{\mu}_t(x_t, x_0)$:

$$L_t \approx \mathbb{E}_{q(x_t|x_0)} \left[\frac{1}{2\tilde{\beta}_t} \|\mu_\theta(x_t, t) - \tilde{\mu}_t(x_t, x_0)\|^2 \right] + \text{const}$$

By expressing both means in terms of the underlying noise using the reparameterization trick:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I)$$

we showed that:

$$\mu_\theta(x_t, t) - \tilde{\mu}_t(x_t, x_0) = -\frac{\sqrt{1 - \alpha_t}}{\sqrt{\alpha_t}} \left(\epsilon_\theta(x_t, t) - \epsilon_t \right)$$

and thus,

$$\|\mu_\theta - \tilde{\mu}_t\|^2 = \frac{1 - \alpha_t}{\alpha_t} \|\epsilon_\theta(x_t, t) - \epsilon_t\|^2$$

Substituting this into the loss expression reveals that minimizing the KL divergence is equivalent to minimizing the squared difference between the predicted noise of the model $\epsilon_\theta(x_t, t)$ and the true noise ϵ_t. This yields the simplified training loss:

$$L_t^{\text{simple}} = \mathbb{E}_{q(x_t|x_0)} \left[\|\epsilon_\theta(x_t, t) - \epsilon_t\|^2 \right]$$

This elegant reformulation has become the standard objective in Denoising Diffusion Probabilistic Models (DDPM). Instead of directly predicting x_0 or x_{t-1}, the model learns to predict the noise ϵ_t added to x_0 at timestep t. This noise-prediction formulation forms the foundation of most modern diffusion models and enables efficient and scalable training across high-dimensional data domains such as images, audio, and time-series.

$$\boxed{L_t^{\text{simple}} = \mathbb{E}_{q(x_t|x_0)} \left[\|\epsilon_\theta(x_t, t) - \epsilon_t\|^2 \right]}$$

The full training objective aggregates the loss across all timesteps, which forms the foundation of DDPM training.

$$\boxed{L_{\text{total}} = \sum_{t=1}^{T} \mathbb{E}_{q(x_t, x_0)} \left[\|\epsilon_\theta(x_t, t) - \epsilon_t\|^2 \right] + \text{const}}$$

D Conditional Diffusion Model

D.1 Classifier Guidance

In *"Improved Denoising Diffusion Probabilistic Models"* by Dhariwal and Nichol (2021), the authors introduce a powerful method known as **Classifier Guidance**, which enables conditional generation in diffusion models without modifying the original training process. This approach leverages a separately trained classifier to guide the reverse diffusion process during sampling by computing gradients of the log-probability $\log p_\phi(y \mid x_t)$. By injecting this gradient into the denoising update, the model can steer samples toward the desired condition y.

Bayesian Reformulation of the Posterior. The objective of classifier guidance is to sample from the class-conditional distribution $p(x_0 \mid y)$, where x_0 denotes the clean data and y is the desired condition such as a class label. Since direct sampling from $p(x_0 \mid y)$ is intractable, we approximate it via a reverse diffusion process that incorporates the guidance of a separately trained classifier $p_\phi(y \mid x_t)$ at each time step t. The diffusion model defines the unconditional reverse process as:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)), \text{ where}$$

μ_θ and Σ_θ are outputs of the neural network trained to approximate the true reverse transitions. To incorporate condition y, we aim to sample from the conditional posterior:

$$p_{\theta,\phi}(x_{t-1} \mid x_t, y),$$

which is difficult to model directly. However, by applying Bayes' rule, the joint distribution over x_{t-1} and y can be factorized as:

$$p_{\theta,\phi}(x_{t-1}, y \mid x_t) = p_\theta(x_{t-1} \mid x_t)\, p_\phi(y \mid x_{t-1})$$

Thus, the conditional distribution becomes:

$$p_{\theta,\phi}(x_{t-1} \mid x_t, y) = \frac{p_\theta(x_{t-1} \mid x_t)\, p_\phi(y \mid x_{t-1})}{p_\phi(y \mid x_t)} \propto p_\theta(x_{t-1} \mid x_t)\, p_\phi(y \mid x_{t-1})$$

Here, $p_\phi(y \mid x_t)$ serves as a normalization constant with respect to x_{t-1} and does not influence the sampling trajectory. This implies that the reverse transition distribution is modulated by the classifier's preference for samples belonging to the target class y. In practice, the classifier $p_\phi(y \mid x)$ is trained using noisy inputs x_t that are obtained by diffusing clean samples x_0 through the forward process. The training objective typically adopts a cross-entropy loss, particularly suited for classification tasks. In the case of multiclass classification with C classes, the classifier $p_\phi(y \mid x_t)$ is modeled using a softmax function applied to class logits $z = (z_1, z_2, \ldots, z_C)$ such that:

$$p_\phi(y = c \mid x_t) = \frac{\exp(z_c)}{\sum_{j=1}^{C} \exp(z_j)}, \text{ where}$$

$z_c = f_\phi^{(c)}(x_t)$ denotes the logit output for class c produced by the classifier network. The cross-entropy loss then measures the negative log-likelihood of the true class label y under this softmax distribution:

$$\mathcal{L}_{\text{clf}} = -\mathbb{E}_{(x_0,y)} \left[\log p_\phi(y \mid x_t)\right] = -\mathbb{E}_{(x_0,y)} \left[\log \frac{\exp(z_y)}{\sum_{j=1}^{C} \exp(z_j)}\right]$$

This formulation ensures that the classifier learns to assign a high probability to the correct class, even when the input is a noisy intermediate sample x_t. More importantly, it enables the computation of the gradient $\nabla_{x_t} \log p_\phi(y \mid x_t)$ during sampling. These gradients act as guidance signals that adjust the reverse denoising trajectory in the diffusion process, guiding the generated samples toward regions that are likely to be under the target class label y, thus allowing controlled generation of conditions of the class.

Perturbed Gaussian Approximation. As established in the previous subsection, the classifier-guided posterior at each reverse step can be expressed as:

$$p_{\theta,\phi}(x_{t-1} \mid x_t, y) \propto p_\theta(x_{t-1} \mid x_t)\, p_\phi(y \mid x_{t-1}) \tag{5}$$

The base reverse distribution $p_\theta(x_{t-1} \mid x_t)$ is modeled as a Gaussian and the likelihood of the classifier $p_\phi(y \mid x_{t-1})$ acts as a guidance term. While this formulation is theoretically sound, direct sampling from such a distribution is generally intractable due to the non-Gaussian nature of the classifier term. To make sampling practical, we approximate the conditional posterior using a perturbed Gaussian. Specifically, we assume that the base model outputs a Gaussian distribution:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t), \Sigma_t), \text{ where}$$

μ_θ and Σ_t are computed from the diffusion model. Then, we apply a first-order Taylor expansion to the classifier log-probability term $\log p_\phi(y \mid x_{t-1})$ around x_t:

$$\log p_\phi(y \mid x_{t-1}) \approx \log p_\phi(y \mid x_t) + \nabla_{x_t} \log p_\phi(y \mid x_t)^\top (x_{t-1} - x_t) \tag{6}$$

Substituting the first-order Taylor approximation (6) into the conditional distribution (5) yields:

$$\begin{aligned}
\log p_{\theta,\phi}(x_{t-1} \mid x_t, y) &\approx \log p_\theta(x_{t-1} \mid x_t) + \log p_\phi(y \mid x_{t-1}) \\
&= \log \mathcal{N}(x_{t-1}; \mu_\theta, \Sigma_t) + \log p_\phi(y \mid x_{t-1}) \\
&\approx -\tfrac{1}{2}(x_{t-1} - \mu_\theta)^\top \Sigma_t^{-1}(x_{t-1} - \mu_\theta) + \nabla_{x_t} \log p_\phi(y \mid x_t)^\top (x_{t-1} - x_t) + C \\
&= -\tfrac{1}{2}(x_{t-1} - \mu_\theta)^\top \Sigma_t^{-1}(x_{t-1} - \mu_\theta) + (x_{t-1} - \mu_\theta)^\top \Sigma_t^{-1} g + C_1, \text{ where} \\
&\quad g := \nabla_{x_t} \log p_\phi(y \mid x_t) \\
&= -\tfrac{1}{2}(x_{t-1} - \mu_{\text{new}})^\top \Sigma_t^{-1}(x_{t-1} - \mu_{\text{new}}) + C_2, \text{ where } \mu_{\text{new}} := \mu_\theta + \Sigma_t g
\end{aligned}$$

Hence, the classifier-guided conditional posterior can be effectively approximated by a Gaussian distribution with a perturbed mean:

$$x_{t-1} \sim \mathcal{N}(\mu_{\text{new}}, \Sigma_t), \quad \text{where} \quad \mu_{\text{new}} = \mu_\theta + \Sigma_t \nabla_{x_t} \log p_\phi(y \mid x_t)$$

This result demonstrates that the effect of the classifier is to shift the mean of the reverse diffusion distribution toward regions of higher likelihood under the target class label y, while maintaining the same covariance structure Σ_t as in the unconditional model. The guidance signal $\nabla_{x_t} \log p_\phi(y \mid x_t)$ steers the denoising trajectory by modulating the reverse transitions in a way that increases class consistency, thereby enabling conditional sample generation without retraining the diffusion model itself. To flexibly control the degree of guidance, a user-defined scaling factor $s > 0$ is introduced:

$$\mu_t^{\text{guided}} = \mu_\theta + s\Sigma_t \nabla_{x_t} \log p_\phi(y \mid x_t),$$

which allows tuning the balance between class-conditioning fidelity and sample diversity. The higher values of s produce samples more aligned with the target condition y, at the expense of reduced diversity, while the lower values allow for broader exploration but looser conditioning.

Gradient Computation and Losses. The classifier guidance mechanism relies on the gradient $\nabla_{x_t} \log p_\phi(y \mid x_t)$, which is computed from a **pretrained classifier** p_ϕ. This classifier is trained independently of the diffusion model, using cross-entropy loss on noisy inputs x_t that are generated by diffusing clean samples x_0 through the forward process.

- **Classifier training loss.** Formally, the classifier is optimized using:

$$\mathcal{L}_{\text{clf}} = -\mathbb{E}_{(x_0,y)\sim p_{\text{data}},\, t\sim\text{Unif}[1,T]} \left[\log p_\phi(y \mid x_t)\right], \text{ where}$$

  $x_t \sim q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$ is obtained via the forward diffusion process.

- **Denoising model loss.** The diffusion model ϵ_θ is trained in a standard (unconditional) DDPM setting by minimizing the MSE between the predicted noise and the true noise:

$$\mathcal{L}_{\text{denoise}}^{\text{vanilla}} = \mathbb{E}_{x_0,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right], \quad \text{where } x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

- **Classifier-guided gradient adjustment.** During sampling, the classifier guidance modifies the reverse denoising trajectory. Instead of sampling from the unconditional reverse process, we want to steer the sample x_{t-1} toward the target class y using the classifier's guidance. This is done by modifying the mean of the Gaussian reverse transition:

$$\mu_t^{\text{guided}} = \mu_\theta(x_t, t) + s \cdot \Sigma_t \nabla_{x_t} \log p_\phi(y \mid x_t),$$

  which results in a shift in the expected clean sample $\hat{x}_0$ toward the class manifold by y. Given that the predicted clean sample $\hat{x}_0$ can be expressed from the noise prediction as:

$$\hat{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t} \cdot \epsilon_\theta(x_t, t)\right)$$

  We can reinterpret the effect of classifier guidance as modifying the predicted noise ϵ_θ to a guided noise ϵ_{new} such that:

$$\hat{x}_0^{\text{guided}} = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t} \cdot \epsilon_{\text{new}}\right)$$

  The modified noise term ϵ_{new} arises from adjusting the predicted clean sample $\hat{x}_0$ in the direction of the classifier gradient, thus guiding the reverse denoising process toward samples of the target class. Recall that in standard DDPM, the clean sample estimate is reconstructed from the noisy input x_t and the predicted noise $\epsilon_\theta(x_t, t)$ as:

$$\hat{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t} \cdot \epsilon_\theta(x_t, t)\right)$$

  In classifier guidance, this estimate is modified to steer sampling toward high-probability regions under the classifier $p_\phi(y \mid x_t)$:

$$\hat{x}_0^{\text{guided}} = \hat{x}_0 + \eta \cdot \nabla_{x_t} \log p_\phi(y \mid x_t), \text{ where}$$

  η is a scaling factor. Substituting this into the inversion formula of $\hat{x}_0$ gives:

$$\hat{x}_0^{\text{guided}} = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t} \cdot \epsilon_{\text{new}}\right)$$

Solving for ϵ_{new} yields:

$$\epsilon_{\text{new}} = \frac{x_t - \sqrt{\bar{\alpha}_t} \cdot \hat{x}_0^{\text{guided}}}{\sqrt{1-\bar{\alpha}_t}}$$

Substituting $\hat{x}_0^{\text{guided}} = \hat{x}_0 + \eta \cdot \nabla_{x_t} \log p_\phi(y \mid x_t)$, and using the expression for $\hat{x}_0$, we obtain:

$$\begin{aligned}
\epsilon_{\text{new}} &= \frac{x_t - \sqrt{\bar{\alpha}_t} \cdot (\hat{x}_0 + \eta \cdot \nabla_{x_t} \log p_\phi(y \mid x_t))}{\sqrt{1-\bar{\alpha}_t}} \\
&= \frac{x_t - \sqrt{\bar{\alpha}_t} \cdot \left(\frac{1}{\sqrt{\bar{\alpha}_t}} (x_t - \sqrt{1-\bar{\alpha}_t} \cdot \epsilon_\theta) + \eta \cdot \nabla_{x_t} \log p_\phi(y \mid x_t) \right)}{\sqrt{1-\bar{\alpha}_t}} \\
&= \epsilon_\theta(x_t, t) - \eta \cdot \sqrt{1-\bar{\alpha}_t} \cdot \nabla_{x_t} \log p_\phi(y \mid x_t)
\end{aligned}$$

This formulation reflects how the classifier guidance modifies the original noise prediction ϵ_θ to incorporate the label information during sampling.

- **Classifier-guided denoising loss.** With this new target, the denoising loss becomes:

$$\mathcal{L}_{\text{denoise}}^{\text{guided}} = \mathbb{E}_{x_0, \epsilon, t} \left[\|\epsilon - \epsilon_{\text{new}}\|^2 \right],$$

which encourages the model to not only denoise, but also to generate outputs consistent with the target class. This formulation allows the classifier's gradient to act as a corrective signal that shifts the denoising direction. The diffusion model does not need to be retrained; instead, this guidance is applied only during sampling by modifying the mean or noise prediction using:

$$\epsilon_{\text{new}} = \epsilon_\theta(x_t, t) - \sqrt{1-\bar{\alpha}_t} \cdot \nabla_{x_t} \log p_\phi(y \mid x_t), \text{ or}$$

$$x_{t-1} \sim \mathcal{N}\left(\mu_\theta + s \cdot \Sigma_t \nabla_{x_t} \log p_\phi(y \mid x_t), \Sigma_t\right)$$

Classifier-Guided Noise Modification

When the guidance strength $\eta = 1$, the adjusted noise estimate simplifies to the widely adopted form:

$$\epsilon_{\text{new}} = \epsilon_\theta(x_t, t) - \sqrt{1-\bar{\alpha}_t} \cdot \nabla_{x_t} \log p_\phi(y \mid x_t)$$

This formulation shows how classifier guidance modifies the original noise prediction $\epsilon_\theta(x_t, t)$ by incorporating the gradient signal from a pretrained classifier $p_\phi(y \mid x_t)$. The gradient $\nabla_{x_t} \log p_\phi(y \mid x_t)$ effectively steers the reverse diffusion trajectory toward regions in the data space that are more likely to be associated with the desired label y.
Importantly, the hyperparameter η allows interpolation between unconditional and conditional generation. When $\eta = 0$, the classifier gradient term vanishes, and the expression reduces to:

$$\epsilon_{\text{new}} = \epsilon_\theta(x_t, t),$$

which corresponds exactly to the original DDPM denoising formulation. Thus, setting $\eta = 0$ disables classifier guidance, and samples are generated purely based on the learned denoising model without label conditioning.

D.2 Direct Conditioning

In *"Cascaded Diffusion Models for High Fidelity Image Generation"* by J. Ho et al. (2022), the authors propose a conditional variant of diffusion models in which the generative process is explicitly guided by auxiliary information such as class labels, text descriptions, or low-resolution images. A key approach introduced in this work is to incorporate the condition directly into the reverse denoising process — a method commonly referred to as **Direct Conditioning**. Unlike **Classifier Guidance**, which utilizes the gradient of an external classifier during sampling and does not require modifying the original diffusion model architecture, Direct Conditioning integrates the condition into the model itself at both training and inference time. This results in a unified, end-to-end conditional generative model without reliance on a separate classifier. In conditional diffusion models, the goal is to generate samples that align with a specific condition or label, such as a class, text prompt, segmentation mask, or observed time series data. The most straightforward way to incorporate such conditions is to directly condition the denoising network on the additional input. Let x_0 be a clean data sample and 'c' be the associated condition (e.g., class label or text embedding). The forward diffusion process remains unchanged from the unconditional setting:

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

Here, no noise is applied to the condition c. We treat training data as paired tuples (x_0, c), and only x_0 undergoes noise corruption.

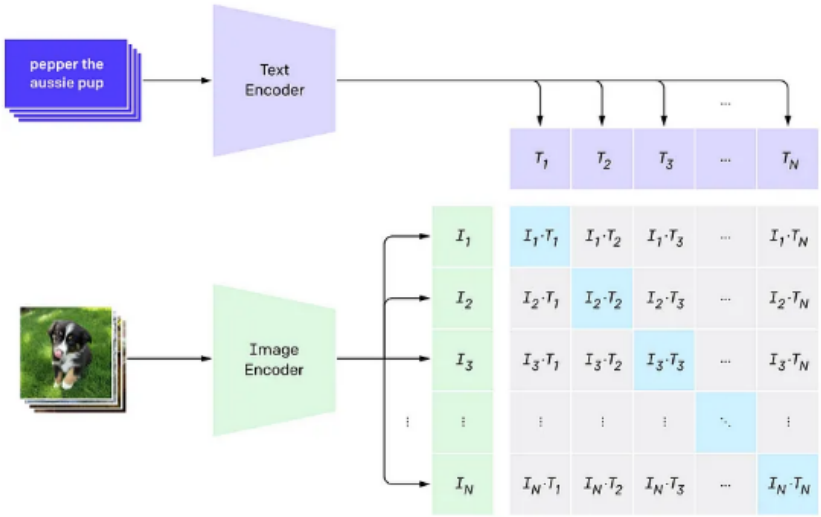


Figure 18: Integrating conditions in a direct way

Conditioned Reverse Process. The reverse process is modified to incorporate an external condition c (e.g., a class label or text description). At each timestep t, the model predicts the noise ϵ added to the data using not only the noisy input x_t and timestep t, but also the clean condition c as an extra input. To do this, we re-train the model to receive the paired input (x_t, c) and learn the conditional noise predictor:

$$\epsilon_\theta(x_t, t, c)$$

This prediction is used to compute the denoised mean in the Gaussian reverse transition:

$$p_\theta(x_{t-1} \mid x_t, c) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t, c), \sigma_t^2 I)$$

where

$$\mu_\theta(x_t, t, c) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t, c) \right)$$

This approach explicitly alters the generative trajectory according to the condition c, allowing the model to generate samples that are consistent with the specified label or context. Importantly, the condition c remains uncorrupted throughout the process (i.e., no noise is added to it), and the model is trained on paired data (x_0, c) where only x_0 is diffused.

Implementation Details. To effectively inject the condition c into the denoising network, the first step is to embed it into a dense vector representation $e_c = E(c)$. The embedding function E depends on the nature of the condition and can take various forms:

- **Learned embedding layer:** Typically used for discrete class labels or small vocabularies.
- **Pretrained encoder:** For more complex or structured conditions such as text or time-series, encoders like BERT, LSTM, or GRU are commonly used.
- **Multimodal encoders:** In vision-language models, CLIP or ViT (Vision Transformer) encoders are often used to embed image-text pairs.
- **Positional/temporal encoding:** For sequential conditions (e.g., observed sensor data), positional encodings or Transformer-style temporal embeddings can be added to retain order information.

Once the condition is embedded into e_c, it is integrated into the denoising architecture using one or more of the following strategies:

- **Concatenation:** The simplest approach is to concatenate e_c with the noisy input x_t or intermediate feature maps within the network.
- **Conditional Normalization:** Feature-wise modulation is performed by injecting e_c into normalization layers such as BatchNorm, GroupNorm, or LayerNorm. Common implementations include Feature-wise Linear Modulation (FiLM) and Adaptive Group Normalization (AdaGN).
- **Cross-Attention:** The embedded condition e_c is used as the key and value in a cross-attention mechanism within the denoising U-Net or Transformer. This allows the model to dynamically attend to the relevant parts of the condition at each spatial or temporal location. Cross-attention is the dominant method in large-scale text-to-image models such as Stable Diffusion.
- **Feature-wise Affine Transformation:** Beyond FiLM, some architectures apply a learned affine transformation on intermediate feature maps conditioned on e_c.
- **Adapter or Gating Modules:** In Transformer-based diffusion models, lightweight adapter layers (e.g., LoRA) or gated residual paths (e.g., Gated AdaLN) are used to selectively inject condition information while minimizing the number of learnable parameters.

D.3 Classifier-Free Guidance

Dropout-Based Training. Classifier-Free Guidance (CFG), introduced by Ho and Salimans (2021), enables conditional generation using a single denoising model $\epsilon_\theta(x_t, t, y)$ trained with both conditional and unconditional objectives. During training, the condition y is randomly dropped with probability p_{drop}, and replaced with a special null token $\emptyset$ to simulate an unconditioned setting. The model is then trained to minimize the standard noise prediction loss under both settings:

$$\mathbb{E}_{x_0,y,t,\epsilon,\tilde{y}\sim\text{Drop}(y)}\left[\|\epsilon_\theta(x_t, t, \tilde{y}) - \epsilon\|^2\right], \text{ where}$$

$\tilde{y} = y$ with probability $1 - p_{\text{drop}}$ and $\tilde{y} = \emptyset$ with probability p_{drop}. This dropout-style training encourages the model to learn both conditional and unconditional denoising behavior, enabling flexible guidance at inference time using only a single network.

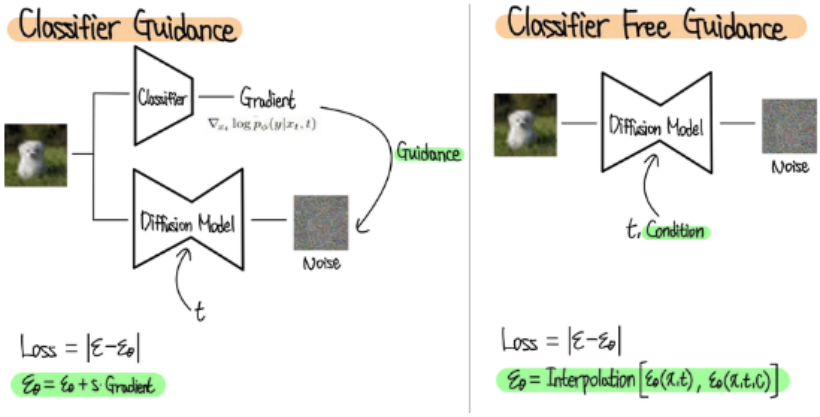


Figure 19: CFG Paradigm compared with CG

Inference via Linear Combination. At inference time, the model outputs two noise predictions for each noisy sample x_t:

$$\epsilon_c = \epsilon_\theta(x_t, t, y) \quad \text{(conditional prediction)}$$

$$\epsilon_u = \epsilon_\theta(x_t, t, \emptyset) \quad \text{(unconditional prediction)}$$

These are then combined via a linear interpolation to amplify the influence of the condition:

$$\hat{\epsilon} = \epsilon_u + w(\epsilon_c - \epsilon_u), \quad \text{where } w > 1$$

The modified denoising step uses $\hat{\epsilon}$ in place of ϵ such that:

$$\mu_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \sqrt{1-\alpha_t}\,\hat{\epsilon}\right)$$

$$x_{t-1} = \mu_t + \sqrt{\sigma_t^2}\cdot z, \quad z \sim \mathcal{N}(0, I)$$

Therefore, the guidance scale w controls how strongly the generation is influenced by the condition y. Larger values of w yield more condition-faithful outputs, but may also increase sample distortion or reduce diversity.

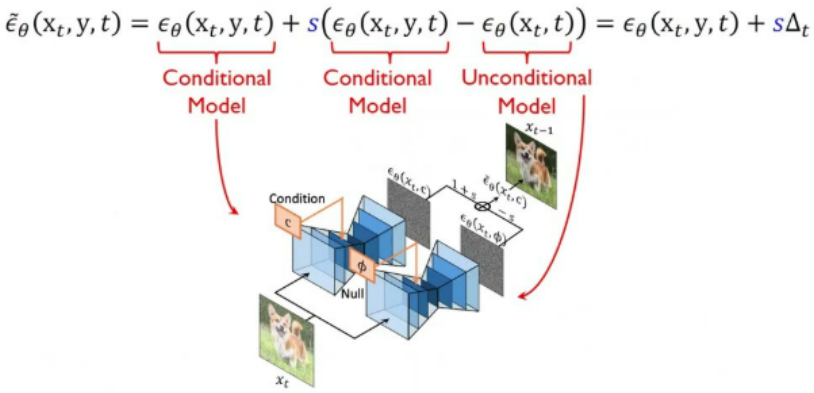


Figure 20: Inference via Linear Combination

Classifier-Free Guidance (CFG) offers several practical advantages that make it particularly appealing for large-scale generative models. Most notably, CFG eliminates the need for an external classifier, in contrast to traditional classifier guidance methods that rely on a separately trained model during sampling. This simplification reduces architectural dependencies and improves portability. Additionally, CFG uses a single denoising model that is trained to handle both conditional and unconditional scenarios. This unified approach enables efficient training and deployment. Perhaps most importantly, CFG introduces the ability to flexibly control the strength of conditioning at inference time via a guidance scale parameter w. By adjusting w, practitioners can modulate the trade-off between fidelity to the condition and output diversity without retraining the model.

Figure 21: Role of guidance scale parameter

Despite these advantages, CFG introduces a couple of key hyperparameters that must be carefully selected. One such parameter is the dropout rate p_{drop}, typically set between 0.1 and 0.2, which determines how often the condition is randomly dropped during training. This parameter directly affects the model's ability to generalize across both conditional and unconditional modes. Another important hyperparameter is the guidance scale w, often set in the range of 1.5 to 3.0, which controls the relative weighting between conditional and unconditional predictions during sampling. Tuning this value is essential for balancing the trade-off between condition faithfulness and generative diversity. In practice, these trade-offs have proven to be manageable, and the flexibility and simplicity of CFG have made it the standard conditioning approach in many diffusion-based generative models, including Stable Diffusion and its variants.