Conditional Diffusion Model

With expansion to the Time-series data application

Donghyun Ko
Posted: June 2025
Updated: July 2025

Abstract

In this tutorial, we provide a clear and structured introduction to conditional diffusion models, focusing on the key theoretical foundations with practical implementation guidance. We begin with a recap of Denoising Diffusion Probabilistic Models (DDPM) to establish the fundamentals. Next, we explore three core strategies for conditioning diffusion models: Classifier Guidance, Direct Conditioning, and Classifier-Free Guidance (CFG). For each approach, we examine its mathematical formulation, how it is trained, and the trade-offs it involves. Finally, we extend these concepts to time-series data, showing how direct and classifier-free conditioning can be adapted to handle temporally structured inputs. With step-by-step explanations and unified perspectives, this tutorial aims to serve as a comprehensive reference for understanding and applying conditional diffusion models across a variety of domains.

Contents

1 Recap of DDPM Fundamentals

Denoising Diffusion Probabilistic Models (DDPM) reinterpret generation as a learned reversal of a structured corruption process, where simple Gaussian noise is added step by step and then iteratively removed using a deep learning framework. This elegant symmetry between forward noise and reverse denoising defines the model's core training principle, grounded in variational inference. At the heart of DDPM lies a simple yet powerful idea: progressively corrupt clean data through a known Gaussian diffusion process, then train a neural network to learn its exact reversal. This formulation allows the complex data distribution to be learned indirectly by inverting a tractable noising process via variational inference. Before diving into the conditional diffusion model, we briefly recap the fundamentals of DDPM.

1.1 Forward (Noising) Process

Given a data sample $x_0 \sim p_{\text{data}}(x)$, the forward process gradually adds Gaussian noise over T steps where $\beta_t \in (0, 1)$ controls the variance of noise in step t:

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I), \quad \text{for } t = 1, \ldots, T \tag{1}$$

Equivalently, this can be written as:

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \epsilon_{t-1}, \quad \epsilon_{t-1} \sim \mathcal{N}(0, I)$$

This can also be done in a quicker and simpler way using the **reparameterization trick**. By induction, one can obtain a closed-form expression for x_t given x_0:

$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}\, x_0, (1-\bar{\alpha}_t) I), \text{ where } \quad \alpha_t := 1 - \beta_t \quad \text{and} \quad \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s \tag{2}$$

Therefore, this allows us to sample x_t directly from x_0 using:

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

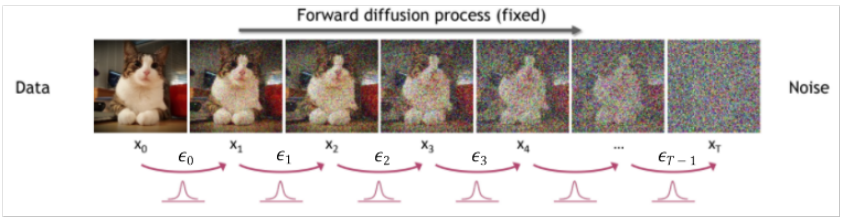

Figure 1: Forward Process

1.2 Reverse (Denoising) Process and ELBO

The goal of the diffusion model is to learn the reverse transition by approximating the intractable $q(x_{t-1} \mid x_t)$ using a neural network framework. We parameterize the reverse process as:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I), \text{ where} \tag{3}$$

μ_θ is learned to approximate the true posterior mean $\tilde{\mu}_t(x_t, x_0)$. 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 complex, high-dimensional integral over latent variables in the reverse process. To circumvent this, we turn to variational inference and instead maximize a tractable surrogate known as the Evidence Lower Bound (ELBO), which provides a principled lower bound on $\log p_\theta(x_0)$ and can be optimized efficiently during training. Using variational inference, the training objective is defined via the ELBO:

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

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

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

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

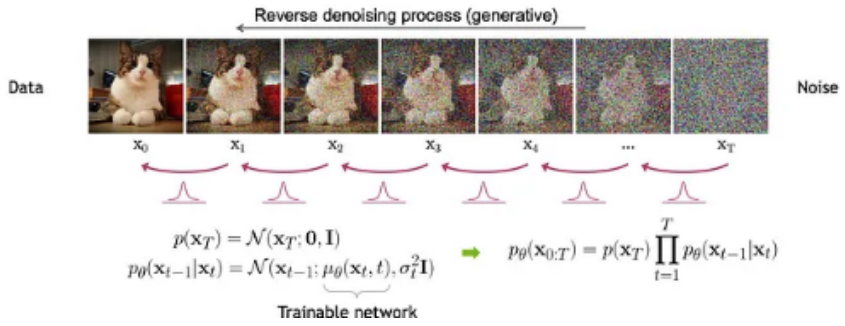

Figure 2: Reverse Process

In practice, the loss terms L_0 and L_T are often ignored during training for computational simplicity and because they contribute relatively little to the total loss compared to the intermediate KL terms L_t. 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 $\mathcal{N}(0, I)$. 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. Therefore, most implementations focus on optimizing L_t for $1 \le t \le T-1$. Furthermore, Ho *et al.* (2020) proposed reparameterizing $\mu_\theta(x_t, t)$ in terms of predicted noise $\epsilon_\theta(x_t, t)$ instead of directly regressing toward 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.

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \sqrt{1-\alpha_t}\, \epsilon_\theta(x_t, t)\right), \tag{4}$$

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:

$$L_t = \mathbb{E}_{t, x_0, \epsilon} \left[\|\epsilon_\theta(x_t, t) - \epsilon\|^2\right] \tag{5}$$

This "noise prediction loss" forms the backbone of DDPM training and remains widely used in modern diffusion models due to its simplicity and empirical success. Having established the

foundations of DDPM—through forward noising, reverse denoising, and the simplified noise prediction objective—we now turn to its conditional extensions that allow more structured and controllable generation.

2 Overview of Conditional Diffusion

Conditional diffusion extends the standard DDPM framework to model conditional distributions of the form $p(x_0 \mid y)$, where y denotes auxiliary information such as class labels, text descriptions, or observed time-series segments. Conditioning enables controlled generation by incorporating domain-specific context into the generative process. Broadly, three major paradigms have emerged to enable conditional generation in diffusion models:

- **Classifier Guidance:** Uses a separately trained classifier to guide the reverse process.
- **Direct Conditioning:** Directly integrates the condition into the model architecture during both training and inference.
- **Classifier-Free Guidance (CFG):** Trains a single model to operate in both conditional and unconditional modes and interpolates between them at inference time.

Each approach introduces different architectural and algorithmic mechanisms, with trade-offs in terms of modularity, fidelity, training complexity, and inference-time flexibility.

2.1 Three Paradigms for Conditional Generation

Classifier Guidance. Originally introduced by Dhariwal and Nichol et al. [2], classifier guidance is a post-training conditioning method that enables a pre-trained unconditional diffusion model to generate conditional samples. This is achieved by augmenting the score function during sampling with the gradient of a separately trained classifier $p_\phi(y \mid x_t)$. Using Bayes' rule, we can write:

$$\nabla_{x_t} \log p(x_t \mid y) = \nabla_{x_t} \log p(x_t) + \nabla_{x_t} \log p(y \mid x_t),$$

where the marginal score $\nabla_{x_t} \log p(x_t)$ corresponds to the unconditional denoising process, and the second term $\nabla_{x_t} \log p(y \mid x_t)$ injects label information into the generation process. Since the true conditional distribution $p(y \mid x_t)$ is generally unknown and intractable, we approximate it using a separately trained classifier $p_\phi(y \mid x_t)$, where ϕ denotes the classifier's parameters. This surrogate allows us to approximate the conditional score using the gradient of the classifier log-likelihood. Thus, the conditional score is approximated as:

$$\nabla_{x_t} \log p(x_t \mid y) \approx \nabla_{x_t} \log p(x_t) + \nabla_{x_t} \log p_\phi(y \mid x_t),$$

which forms the foundation of the *classifier guidance* technique in diffusion models. This allows the model to steer the generation toward the desired condition y by modifying the reverse sampling trajectory. The major advantage of this method is its modularity: a powerful classifier trained on noisy samples can be reused for different generative models. It is particularly effective for class-conditional or image-to-label generation tasks. However, this technique requires backpropagation through the classifier at every sampling step, leading to increased computational cost. Moreover, it is vulnerable to adversarial gradients and suffers from potential distribution mismatch between the classifier and the generative model, which can degrade sample quality or stability.

Direct Conditioning. In this approach, the conditional information y that can represent a class label, text embedding, segmentation map, or past time series observation is directly injected into the denoising model at each timestep. The architecture is explicitly conditioned on y via mechanisms such as concatenation, cross-attention, conditional normalization (e.g., FiLM, AdaGN), or adapter modules. The training dataset consists of pairs (x_0, y) where only x_0 is corrupted by forward diffusion, while y remains fixed and uncorrupted. 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). Classifier-Free Guidance, proposed by Ho and Salimans (2022), eliminates the need for an external classifier while retaining strong conditional control. It is implemented by training a single noise prediction network $\epsilon_\theta(x_t, t, y)$ with **condition dropout**. That is, during training, the condition y is randomly replaced with a null token $\emptyset$ with probability p_{drop}, enabling the model to learn both conditional and unconditional behaviors. At inference time, the model generates two predictions:

$$\epsilon_c = \epsilon_\theta(x_t, t, y), \quad \epsilon_u = \epsilon_\theta(x_t, t, \emptyset)$$

and interpolates them linearly to approximate conditional guidance:

$$\epsilon_{\text{guided}}(x_t, t, y) = (1 + s) \cdot \epsilon_c - s \cdot \epsilon_u \tag{6}$$

where s is a guidance scale hyperparameter that controls the strength of the condition. 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. As a result, it has become the default approach in large-scale conditional diffusion models such as GLIDE, Imagen, and Stable Diffusion. One trade-off, however, is that CFG still relies on direct conditioning during training—just like Direct Conditioning models—and requires careful tuning of p_{drop} and s for optimal performance.

Summary of Trade-offs:

- **Classifier Guidance:** High modularity; suitable for plug-and-play settings but computationally expensive and prone to instability.
- **Direct Conditioning:** Architecturally integrated; stable and effective but inflexible at inference.
- **Classifier-Free Guidance:** Combines architectural simplicity with inference-time control, making it the most widely adopted strategy in practice.

2.2 Mathematical Background and Comparison

All three methods are grounded in score-based generative modeling. In diffusion models, the reverse process is modeled as a series of Gaussian transitions that aim to sample from the conditional distribution $p(x_t \mid y)$ at each timestep. This is done by approximating the score function (i.e., the gradient of the log-density):

$$\nabla_{x_t} \log p(x_t \mid y)$$

By applying Bayes' theorem, this can be decomposed as:

$$\log p(x_t \mid y) = \log p(x_t) + \log p(y \mid x_t) - \log p(y)$$

Differentiating both sides with respect to x_t, we obtain:

$$\nabla_{x_t} \log p(x_t \mid y) = \nabla_{x_t} \log p(x_t) + \nabla_{x_t} \log p(y \mid x_t) \tag{7}$$

This decomposition forms the theoretical foundation for how conditional information is incorporated in diffusion models. Each of the three paradigms handles this score approximation differently:

- **Classifier Guidance** explicitly estimates the second term, $\nabla_{x_t} \log p(y \mid x_t)$, using the gradient of a separately trained classifier $p_\phi(y \mid x_t)$. This gradient is then added to the unconditional score $\nabla_{x_t} \log p(x_t)$ during sampling. The modified sampling process is interpreted as sampling from a perturbed Gaussian with mean shifted in the direction of the classifier's preference. This approach allows for modular reuse of classifiers but requires computing gradients during inference and can suffer from adversarial instability or distributional mismatch.

- **Direct Conditioning** bypasses explicit score estimation completely by directly incorporating y into the noise prediction model $\epsilon_\theta(x_t, t, y)$. The network is trained end-to-end to produce conditional predictions, implicitly learning the conditional score. Since no decomposition such as (7) is used, it treats $p(x_t \mid y)$ as a black-box distribution modeled directly via neural networks. This leads to tight alignment between the condition and the output, but lacks flexibility at inference time.

- **Classifier-Free Guidance** also builds on direct conditioning: the model $\epsilon_\theta(x_t, t, y)$ is trained with condition dropout, allowing it to handle both conditional and unconditional inputs. At inference time, the conditional score is approximated via linear interpolation:

  $$\nabla_{x_t} \log p(x_t \mid y) \approx \epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t) \tag{8}$$

  The effective noise prediction is computed as:

  $$\epsilon_{\text{guided}}(x_t, t, y) = (1 + s) \cdot \epsilon_\theta(x_t, t, y) - s \cdot \epsilon_\theta(x_t, t)$$

  This formulation provides a practical and scalable way to control the influence of y through the guidance scale s, without requiring external classifiers or explicit gradients.

Despite their methodological differences, all three approaches aim to steer the reverse denoising trajectory toward regions in the data manifold that are consistent with the condition y, thereby sampling from the conditional distribution $p(x_0 \mid y)$. In the following chapters, we provide a detailed analysis of each paradigm. Chapter 3 focuses on Classifier Guidance, including its derivation from the score decomposition (7) and how the posterior is approximated using a perturbed Gaussian. Chapter 4 explores Direct Conditioning, explaining various ways of injecting conditional signals into the denoising model and its applications to image and time-series generation. Chapter 5 covers Classifier-Free Guidance, showing how it leverages dropout and interpolation to balance conditional control and sampling flexibility.

3 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.

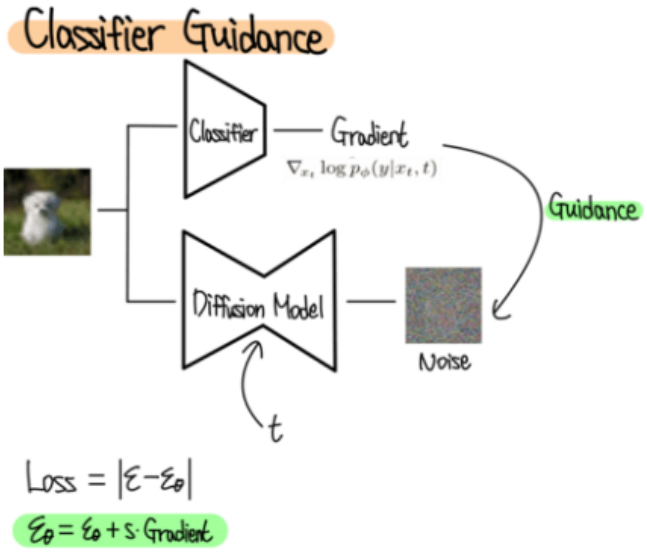

Figure 3: Classifier Guidance Paradigm

In the conditional diffusion model, the objective is to generate samples x_0 that align with a given condition y, such as a class label. Classifier guidance provides a practical mechanism to achieve this goal by steering the unconditional reverse diffusion process using gradients from a separately trained classifier. In this section, we derive this approach from a Bayesian perspective and explain how it modifies the vanilla sampling process.

3.1 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:

$$p_{\theta,\phi}(x_{t-1} \mid x_t, y) = \frac{p_\theta(x_{t-1} \mid x_t)\, p_\phi(y \mid x_{t-1})}{p_\phi(y \mid x_t)} \propto p_\theta(x_{t-1} \mid x_t)\, p_\phi(y \mid x_{t-1}) \tag{9}$$

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. Equation (9) highlights the key idea: 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.

3.2 Perturbed Gaussian Approximation

As established in the previous subsection, the classifier-guided posterior of (8) at each reverse step can be expressed as:

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

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:

$$\log p_\phi(y \mid x_{t-1}) \approx \log p_\phi(y \mid x_t) + \nabla_{x_t} \log p_\phi(y \mid x_t)^\top (x_{t-1} - x_t) \tag{10}$$

Substituting the first-order Taylor approximation (9) into the conditional distribution (8) yields:

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

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.

By completing the square in the exponent (a), we obtain the perturbed posterior:

$$x_{t-1} \sim \mathcal{N}(\mu_t^{\text{guided}}, \Sigma_t), \quad \text{where } \mu_t^{\text{guided}} = \mu_\theta + \Sigma_t \nabla_{x_t} \log p_\phi(y \mid x_t) \tag{11}$$

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.

In summary, by reinterpreting the classifier-augmented posterior as a perturbed Gaussian, we can efficiently perform guided sampling at each reverse step. This approximation enables classifier-based guidance to be practically integrated into diffusion sampling without altering the original training procedure of the denoising model.

3.3 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 (Vanilla model). 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 defined 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 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.

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.

Summary. 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:

$$\epsilon_{\text{new}} = \epsilon_\theta(x_t, t) - \sqrt{1 - \bar{\alpha}_t} \cdot \nabla_{x_t} \log p_\phi(y \mid x_t), \quad \text{or} \quad x_{t-1} \sim \mathcal{N}\left(\mu_\theta + s \cdot \Sigma_t \nabla_{x_t} \log p_\phi(y \mid x_t), \Sigma_t\right).$$

3.4 Trade-Off and Sampling Considerations

Classifier guidance enhances sample fidelity by explicitly pushing generated samples toward regions of high classifier confidence. The scaling parameter s controls this influence: The higher value of s increases the alignment with the condition y (fidelity), but may reduce the sample diversity and cause mode collapse. In contrast, smaller values s preserve diversity, but reduce the influence of the classifier. The method requires evaluating the gradients of the classifier with respect to the current sample x_t at each reverse step, which significantly increases the computational cost during sampling. Nevertheless, this approach is powerful in scenarios where fine control over generated semantics is desired and the classifier can provide reliable gradients.

4 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.

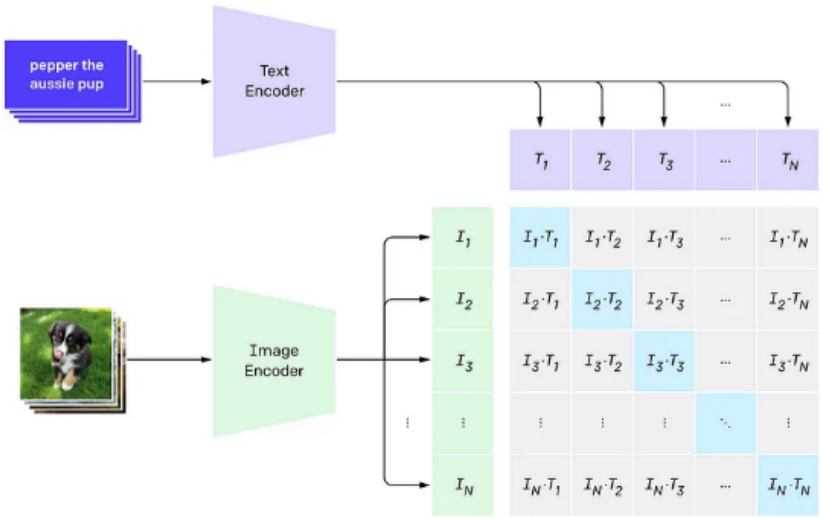

Figure 4: Integrating conditions in a direct way

4.1 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.

4.2 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.

These integration mechanisms can be tailored to the modality and complexity of the condition. For example:

- **Stable Diffusion:** Uses CLIP embeddings for text prompts and injects them via cross-attention layers at multiple levels of the U-Net.
- **ControlNet:** Augments the base U-Net with parallel residual branches that process control signals (e.g., depth, pose, canny edges) before merging them with the main path.
- **Diffusion Transformer (DiT):** Represents both image and condition as token sequences and fuses them via global self-attention across all tokens.

Formally, the denoising function is defined as:

$$\epsilon_\theta : \mathbb{R}^n \times \{0, \ldots, T-1\} \times \mathbb{R}^d \to \mathbb{R}^n$$

where:

- $x_t \in \mathbb{R}^n$ is the noisy data at timestep t,
- t is the diffusion timestep,
- $e_c = E(c) \in \mathbb{R}^d$ is the embedded representation of the condition.

4.3 Trade-Offs and Benefits

Direct Conditioning offers notable advantages in terms of generation fidelity and tight control over conditioning, but it also introduces trade-offs regarding architectural complexity and flexibility during inference.

One of the primary benefits of Direct Conditioning is that it enables end-to-end conditional generation. The model is trained from the outset to generate samples in accordance with a given condition c, which leads to strong alignment between the generated output and the intended condition. Because the condition is directly embedded into the network—typically through mechanisms like cross-attention or conditional normalization—the generated data tends to faithfully reflect the desired attribute or context. This often results in higher fidelity compared to guidance-based approaches that influence generation only during sampling. Additionally, Direct Conditioning does not require any external components such as a separately trained classifier, as is the case with Classifier Guidance, simplifying the overall deployment pipeline.

However, these benefits come with the corresponding limitations. Since the conditioning signal must be explicitly integrated into the model architecture—requiring the addition of attention modules, adaptive normalization, or adapter paths—the resulting model is inherently more complex and often larger in size. Furthermore, Direct Conditioning lacks inference-time flexibility: once the model is trained with a specific conditioning setup, it cannot adjust the strength or presence of that condition without re-training. This contrasts with Classifier-Free Guidance, where the guidance scale can be tuned at sampling time to balance fidelity and diversity. Another important limitation is reusability. A model trained to condition on a particular type of input (such as class labels) cannot be easily repurposed for other conditioning modalities like text prompts or temporal signals without architectural and dataset-level modifications.

In summary, Direct Conditioning is particularly well-suited to settings where the conditioning information is always available and essential, where high-fidelity conditioning is critical, and where sufficient data and resources are available to train a model from scratch. On the other hand, when inference-time adaptability or architectural modularity is more important, guidance-based approaches like Classifier Guidance or Classifier-Free Guidance may be more appropriate alternatives.

5 Classifier-Free Guidance

5.1 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.

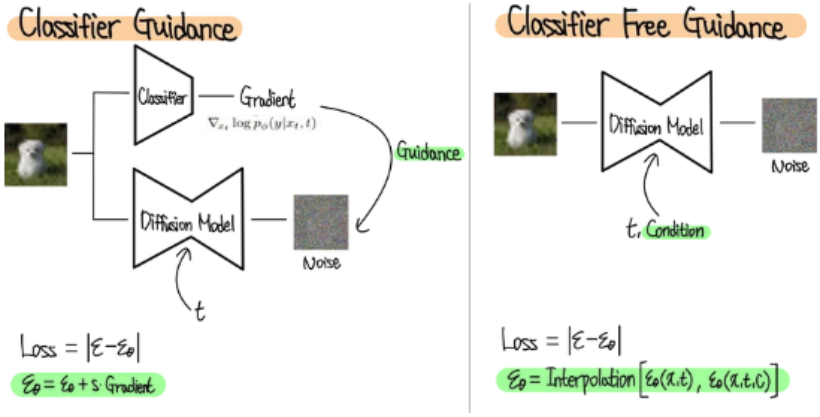

Figure 5: CFG Paradigm compared with CG

5.2 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.

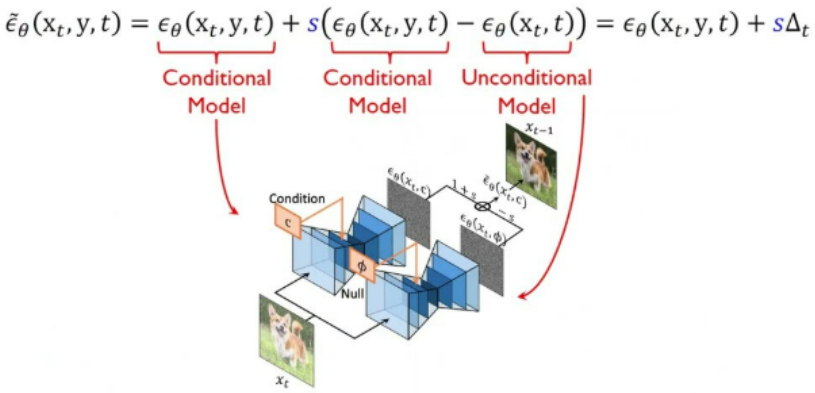

Figure 6: Inference via Linear Combination

5.3 Trade-Offs and Benefits

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 7: 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 state-of-the-art diffusion-based generative models, including Stable Diffusion and its variants.

6 Time-Series Conditional Diffusion

In time-series generation or forecasting, the objective is to predict future values (target segment) based on a partially observed historical window (observation segment). Unlike image generation, the condition is not a static label but a high-dimensional temporal signal with its own complex structure. This makes approaches like classifier guidance less suitable, as applying gradients from external classifiers can break temporal consistency and increase instability. Instead, diffusion models for time series typically rely on directly incorporating the observed past signal as a conditioning input to the model.

6.1 Interpret by Direct Conditioning Framework

The most common and effective approach for time-series conditional diffusion is based on Direct Conditioning. In this framework, the observed sequence $\mathbf{X}_{\text{obs}}$ is treated as an auxiliary input that remains untouched by the forward diffusion process. Only the target segment $\mathbf{X}_{\text{tar}}$ is corrupted with Gaussian noise during the forward process:

$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I), \quad \text{where } x_0 = \mathbf{X}_{\text{tar}}$$

The observation $\mathbf{X}_{\text{obs}}$ is preserved as-is and encoded using an extra neural network E (e.g., RNN or Transformer encoder) into an embedding $e = E(\mathbf{X}_{\text{obs}})$. This embedding is then integrated into the denoising network through one of several mechanisms such as concatenation, conditional normalization (e.g., FiLM), or cross-attention. The denoising model learns the conditional noise prediction as:

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

This prediction is used to compute the mean in the reverse process:

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

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

This method ensures that temporal dependencies are maintained while still allowing flexible and expressive generation conditioned on past observations.

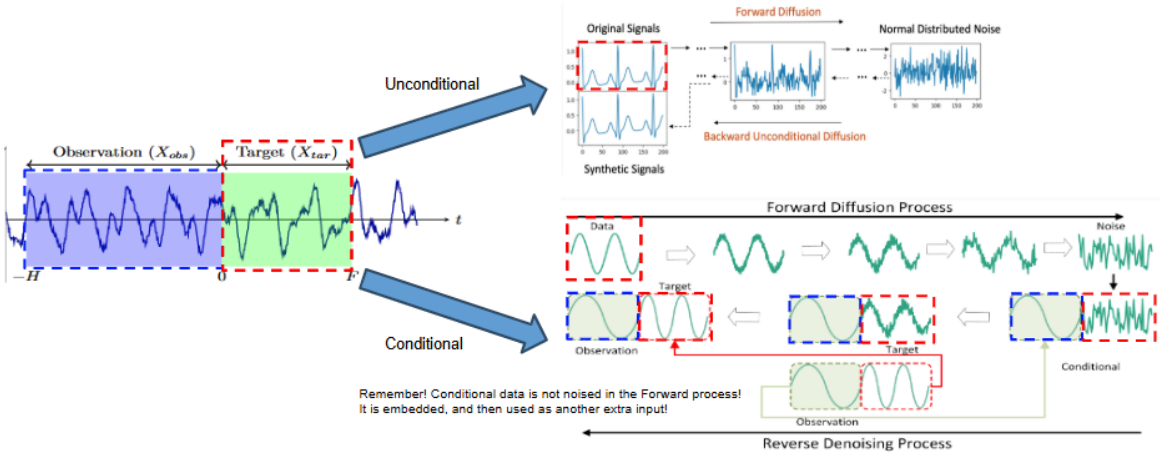

Figure 8: Direct Conditioning Framework

6.2 Training

The model is trained on paired samples $(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{tar}})$ where only the target segment is diffused. The objective function minimizes the mean squared error between the predicted and true noise:

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

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

6.3 Sampling

During sampling, the model generates future trajectories by starting from Gaussian noise and iteratively denoising it using the learned reverse process conditioned on the embedding of $\mathbf{X}_{\text{obs}}$. Since the model is fully conditioned and trained end-to-end, no guidance scale or interpolation between conditional/unconditional outputs is required, as in Classifier-Free Guidance.

6.4 Position in the Design Landscape

As shown in recent literature, the Direct Conditioning approach has become the dominant method in time-series diffusion models such as TimeGrad, ScoreGrad, CSDI, DSPD, and TSDiff. These variants differ mainly in architectural choices (e.g., 1D-UNet, LSTM, Transformer), loss functions (e.g., MSE, CRPS, DWT-based loss), or forward variance schedules (e.g., linear, cosine, learned). However, they all share the core idea of treating $\mathbf{X}_{\text{obs}}$ as a clean, high-dimensional condition that is encoded and injected into the denoising network, defining the direct conditioning paradigm.

While Direct Conditioning provides strong condition fidelity, recent efforts have explored incorporating **Classifier-Free Guidance (CFG)** into time-series diffusion frameworks to improve sampling flexibility. In such approaches, the model is trained with condition dropout, where $\mathbf{X}_{\text{obs}}$ is randomly masked (e.g., replaced with zero or null embeddings) with a fixed probability p_{drop} during training. This enables the model to learn both conditional and unconditional denoising behaviors within the same architecture. At inference time, one can obtain both conditional prediction $\epsilon_c = \epsilon_\theta(x_t, t, E(\mathbf{X}_{\text{obs}}))$ and unconditional prediction $\epsilon_u = \epsilon_\theta(x_t, t, \emptyset)$, and form a linear combination:

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

The guidance scale w can be adjusted to modulate the strength of the condition, enabling controllable generation without retraining the model. This hybrid approach combines the condition fidelity of Direct Conditioning with the sampling-time flexibility of CFG, and is beginning to appear in recent time-series diffusion models such as TSDiff and DiffLoad. Thus, while Direct Conditioning remains the foundation of most existing time-series diffusion architectures, the integration of CFG provides a promising avenue for enhancing controllability and robustness—particularly in settings with noisy or partially observed conditioning inputs.

7 Conclusion

Conditional diffusion models have emerged as powerful and flexible generative frameworks capable of incorporating diverse forms of auxiliary information into the denoising process. Among the three major conditioning strategies,

- **Classifier Guidance** offers modularity and fine-grained semantic control, but at the cost of computational overhead and potential instability.
- **Direct Conditioning** provides a tight alignment between the input condition and the generated output through architectural integration, making it suitable for tasks with well-defined and fixed conditioning structures.
- **Classifier-Free Guidance (CFG)** strikes a practical balance by enabling a single model to support conditional and unconditional sampling with inference-time flexibility.

In the context of time-series forecasting, where conditions are structured and temporally correlated, Direct Conditioning has become the standard. However, the incorporation of CFG into time series models provides additional flexibility by allowing dynamic control over the strength of the condition during sampling.

Overall, this tutorial has highlighted the mathematical principles, algorithmic mechanisms, and design trade-offs underlying each approach. Conditional diffusion remains an active and promising area of research, with growing applications in image synthesis, text generation, multimodal learning, and temporal modeling. Future work will likely explore more adaptive conditioning strategies, hybrid guidance mechanisms, and principled ways to handle partial or noisy conditioning inputs.

References

[1] Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. *NeurIPS*.

[2] Nichol, A., & Dhariwal, P. (2021). Improved Denoising Diffusion Probabilistic Models. *ICML*.

[3] Ho, J., & Salimans, T. (2021). Classifier-Free Diffusion Guidance. *arXiv preprint arXiv:2207.12598*.

[4] Song, Y., & Ermon, S. (2019). Generative Modeling by Estimating Gradients of the Data Distribution. *NeurIPS*.

[5] Song, Y., et al. (2021). Score-Based Generative Modeling through Stochastic Differential Equations. *ICLR*.

[6] Saharia, C., et al. (2022). Imagen: Photorealistic Text-to-Image Generation with Diffusion Models. *arXiv preprint arXiv:2205.11487*.

[7] Rombach, R., et al. (2022). High-Resolution Image Synthesis with Latent Diffusion Models. *CVPR*.

[8] Tashiro, Y., Song, Y., & Ermon, S. (2021). CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation. *NeurIPS*.

[9] Bansal, A., et al. (2023). DiffLoad: A Diffusion-Based Model for Load Forecasting. *arXiv preprint arXiv:2305.15468*.