Chapter 1

Diffusion Models: An Accessible Introduction to the State of the Art in Data Generation

Samir Braga Chaves, José Antônio Fernandes de Macêdo, Regis Pires Magalhães, Vaux Sandino Diniz Gomes, César Lincoln Cavalcante Mattos

Abstract

Generative models have received significant attention in the artificial intelligence community and are expanding their reach beyond the technical sphere. Among the most recent and promising approaches are Diffusion Models, which adopt an innovative generative modeling strategy: they progressively add noise of varying intensities to the data—a technique inspired by stochastic dynamics and known as diffusion—and then learn to reverse these perturbations. Starting from a pure noise sample, these models can generate new images, text, videos, and various other types of data with remarkably high quality, outperforming previous state-of-the-art methods in many of these tasks. This minicourse provides a formal and accessible introduction to the foundations of diffusion models, covering their underlying probabilistic principles—such as Kullback-Leibler divergence, Langevin dynamics, and score matching—as well as visual intuitions to aid comprehension. Then, it presents an overview of the current state of the art in the field, including conditioning techniques and latent diffusion models. Finally, the course discusses the research opportunities and applications that this rapidly evolving area has to offer.

1.1. Introduction

Generative models are currently receiving significant attention within the artificial intelligence community and expanding their impact beyond technical domains. They have enabled the creation of rich, personalized content—such as images, language, and music—with applications ranging from generating realistic photographs and physics-consistent animations to constructing digital worlds for gaming. Their main appeal lies in their ability to handle creative tasks, generate synthetic data, and open new scientific and artistic frontiers.

Various architectures have been proposed for data generation, each with distinct characteristics and modeling strategies. Among the most prominent are Diffusion Models (DMs) [Sohl-Dickstein et al. 2015, Song and Ermon 2019, Ho et al. 2020], which have gained attention for their exceptional sample quality. These models generate data by gradually adding random noise through a process called *diffusion*, and then learning to reverse this corruption. By training a model to denoise, DMs can turn Gaussian noise into realistic samples of images, text, or video [Dhariwal and Nichol 2021a, Gong et al. 2022, Luo et al. 2023]. Due to their solid theoretical foundations and empirical success, DMs have joined the forefront of generative modeling.

Another widely known method is the Generative Adversarial Network (GAN) [Goodfellow et al. 2014], which employs two neural networks (a generator and a discriminator) in a competitive game to improve the quality of generated samples. Variational Autoencoder (VAE) [Kingma and Welling 2022] is an important family that learns a latent probabilistic representation of the data and uses it for generation. Normalizing Flows [Dinh et al. 2015] also rely on latent spaces, but in contrast to VAEs, they use a sequence of invertible transformations to map a simple distribution (e.g., Gaussian) into a complex one that models the data.

Both VAEs and Normalizing Flows use latent spaces as compact representations of data. VAEs associate each data point with a distribution in latent space, promoting diversity in generation, while Flows use deterministic mappings, enabling exact likelihood computation and sharper outputs. Diffusion Models provide a complementary approach: rather than relying on explicit encoder-decoder structures or invertible transformations, they define a generative process by reversing stochastic noise corruption, offering flexibility, robustness, and strong multimodal performance.

Despite their practical success, the variety of formulations and architectures in diffusion models presents challenges for those seeking a clear and unified understanding. This diversity hinders systematic comparison and model selection. Therefore, a comprehensive course is essential to consolidate existing knowledge, explain the mathematical foundations, and provide practical examples to guide implementation.

Along this minicourse, we expect the reader to be familiar with a few basic concepts. From probability and statistics, we expect a good notion of random variables, probability density functions, conditional and marginal probabilities, Bayes' Rule, and the Gaussian distribution. From deep learning, we expect a basic understanding of neural networks, how they are trained, and how they are used for inference. From calculus, the expected background is the gradient operator and the notion of vector fields.

This document presents the background necessary to understand Diffusion Models. Section 1.2 covers mathematical foundations, including latent variable models and variational inference. Section 1.3 introduces Diffusion Models, their forward and reverse processes, and the noise-based generative intuition. Section 1.4 discusses score-based models and Langevin dynamics. Section 1.5 describes common denoising architectures. Section 1.6 outlines recent advances in performance and sampling efficiency. Section 1.7 surveys applications in domains such as image, video, and text generation, bioinformatics, and tabular data. Finally, Section 1.8 summarizes key takeaways and future directions.

1.1.1. Code and resources

The implementation code and supplementary materials for this minicourse are available at the following repository: `https://github.com/InsightLab/diffusion_models_course`. There, readers will find a collection of Jupyter notebooks containing both from-scratch implementations and practical examples using the `diffusers`[1] library developed by Hugging Face. In addition to the main minicourse content, the repository also includes curated references to external implementations, papers, and hands-on examples covering a variety of use cases. This resource is intended to support further exploration, experimentation, and application of Diffusion Models in real-world scenarios.

1.2. Background

1.2.1. Generative modeling

Generative models, such as Diffusion Models, assume that all data come from a joint probability distribution of data $p(\mathbf{x})$. The two main steps of generative modeling are estimating that distribution and sampling from it. The first step consists of training a model to maximize the expected log likelihood of $p(\mathbf{x})$ or trying to minimize some divergence metric between the real distribution of the data $p(\mathbf{x})$ and the distribution $p_\theta(\mathbf{x})$ learned by the model, where θ corresponds to the model's parameters. Alternatively, one could also estimate some quantity related to the data distribution, such as the score function $\nabla \log p(\mathbf{x})$. The second step is to sample points from the learned distribution; we expect these points to be similar to those from the original data distribution. Figure 1.1 illustrates the overall idea of generative modeling.

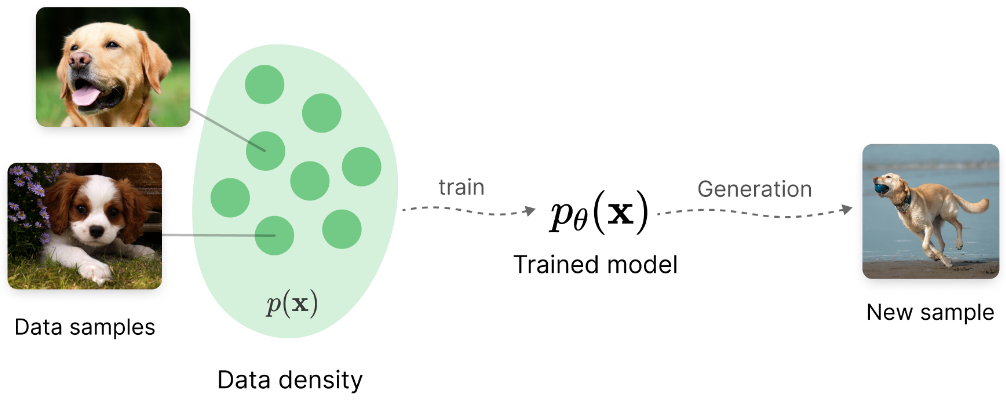


Figure 1.1. Illustration of a generative modeling process. Real data samples are used to estimate a data distribution $p_\theta(\mathbf{x})$ through training. Once trained, the model can generate new samples by sampling from $p_\theta(\mathbf{x})$, ideally matching the characteristics of the original data distribution $p(\mathbf{x})$.

1.2.2. Latent variable models

Generative models often aim to describe complex data distributions by introducing additional variables that are not directly observed, called *latent variables*. Instead of modeling the data distribution $p(\mathbf{x})$ directly, we assume that the observed data $\mathbf{x}$ is generated from

[1] `https://huggingface.co/docs/diffusers/index`

some unobserved latent representation $\mathbf{z}$. The model defines two components: a prior distribution $p(\mathbf{z})$, usually chosen to be simple (e.g., a standard Gaussian), and a likelihood model $p(\mathbf{x} \mid \mathbf{z})$ that specifies how data is generated from its latent representation.

This modeling assumption leads to the following joint distribution $p(\mathbf{x},\mathbf{z}) = p(\mathbf{z}) \cdot p(\mathbf{x} \mid \mathbf{z})$, and from it, we can derive the posterior:

$$\underbrace{p(\mathbf{z} \mid \mathbf{x})}_{\text{posterior}} = \frac{\overbrace{p(\mathbf{x} \mid \mathbf{z})}^{\text{likelihood}} \cdot \overbrace{p(\mathbf{z})}^{\text{prior}}}{\underbrace{p(\mathbf{x})}_{\text{evidence}}}. \tag{1}$$

Here, the latent variable $\mathbf{z}$ captures the hidden structure or semantics that explain the variability in the observed data $\mathbf{x}$. The generation process is a two-step sampling procedure: first sample $\mathbf{z} \sim p(\mathbf{z})$, then sample $\mathbf{x} \sim p(\mathbf{x} \mid \mathbf{z})$. This framework is visualized in Figure 1.2.

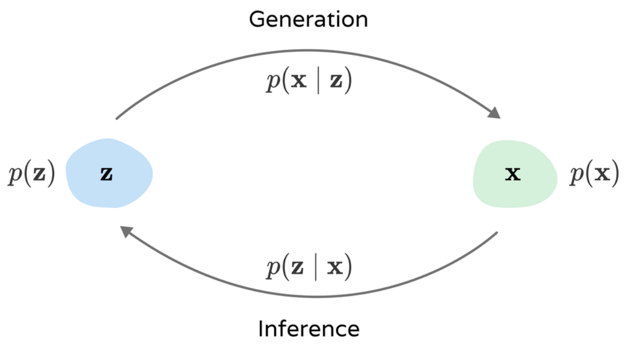


Figure 1.2. Graphical representation of a latent variable model. The top arrow illustrates the generation process: latent variables $\mathbf{z}$ are sampled from a simple prior distribution $p(\mathbf{z})$ and transformed into observed data $\mathbf{x}$ via the likelihood $p(\mathbf{x} \mid \mathbf{z})$. The bottom arrow represents inference: given observed data $\mathbf{x}$, the model estimates the posterior distribution $p(\mathbf{z} \mid \mathbf{x})$, which describes plausible latent causes for the observation.

A classic example of a latent variable model is *Principal Component Analysis* (PCA). In PCA, each data point $\mathbf{x}$ is assumed to be generated by a low-dimensional latent vector $\mathbf{z}$, linearly mapped to the high-dimensional data space by a learned projection matrix. While PCA uses linear transformations, modern approaches, such as Variational Autoencoders, extend this framework to nonlinear mappings using neural networks. Figure 1.3 can be reinterpreted here to show how latent variables mediate between prior assumptions and observed data.

One of the main computational challenges in latent variable models is the evaluation of the marginal likelihood $p(\mathbf{x})$, given by:

$$p(\mathbf{x}) = \int p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z},$$

which is often intractable in high-dimensional settings. To address this, various approximation techniques are used, such as *variational inference*, which introduces a surrogate distribution to estimate the posterior $p(\mathbf{z} \mid \mathbf{x})$.

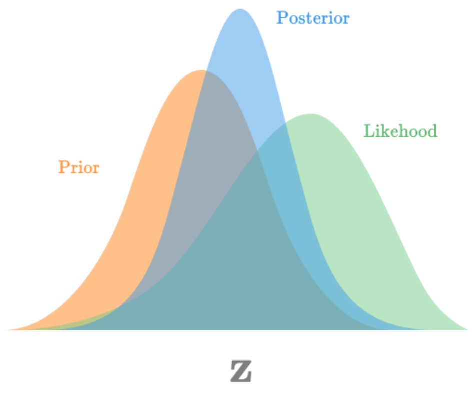


Figure 1.3. Posterior distribution $p(\mathbf{z} \mid \mathbf{x})$ (blue) as the result of combining the prior $p(\mathbf{z})$ (orange) with the likelihood $p(\mathbf{x} \mid \mathbf{z})$ (green) in a latent variable model. The $\mathbf{z}$-axis represents latent factors that explain the observed data $\mathbf{x}$.

In this generative framework, once the model has been trained, new data can be generated by first sampling a latent code $\mathbf{z}$ from the prior $p(\mathbf{z})$, and then generating the corresponding data point $\mathbf{x}$ using the learned decoder $p(\mathbf{x} \mid \mathbf{z})$. This approach enables controllable and interpretable generation, as the latent space can encode meaningful attributes of the data.

1.2.3. Variational Inference

In the context of latent variable models, the posterior $p(\mathbf{z} \mid \mathbf{x})$ is typically intractable to compute. To address this, one can use Variational Inference [Jordan et al. 1998, Wainwright and Jordan 2008] to approximate it with a surrogate distribution $q(\mathbf{z} \mid \mathbf{x})$. To quantify how "close" this approximation is to the true posterior, we use the Kullback-Leibler (KL) divergence: $D_{\mathrm{KL}}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x}))$. This divergence is zero when the two distributions are identical and increases as they differ. The objective is thus to find the distribution q that minimizes the KL divergence:

$$\arg\min_{q} D_{\mathrm{KL}}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})). \tag{2}$$

One might question how it is possible to compute the divergence between an unknown distribution $p(\mathbf{z} \mid \mathbf{x})$ and a distribution $q(\mathbf{z} \mid \mathbf{x})$ we are trying to optimize. This is a valid concern. However, when we expand the KL divergence analytically, a useful identity arises:

$$D_{\mathrm{KL}}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})) = \log p(\mathbf{x}) - \underbrace{\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x} \mid \mathbf{z})] - D_{\mathrm{KL}}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}))}_{\text{ELBO}} \tag{3}$$

$$= \log p(\mathbf{x}) - \underbrace{\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})}\right]}_{\text{ELBO}}. \tag{4}$$

The right-hand side of Equation 3 consists of two components: the log evidence $\log p(\mathbf{x})$, and a term known as the Evidence Lower Bound (ELBO). Since we are optimizing with respect to q, the log evidence can be ignored, as it does not depend on q.

The ELBO includes three expressions: the variational distribution $q(\mathbf{z} \mid \mathbf{x})$ that we aim to optimize; the prior $p(\mathbf{z})$ over the latent variables, which we can choose; and the likelihood $p(\mathbf{x} \mid \mathbf{z})$, which defines the generative model and can also be optimized jointly. Equation 4 provides an alternative but equivalent form of the ELBO, which is particularly useful in the context of diffusion models. With this, the new optimization objective becomes:

$$\arg\min_{q} -\text{ELBO}(q) = \arg\max_{q} \text{ELBO}(q). \tag{5}$$

The acronym ELBO stands for **E**vidence **L**ower **BO**und, indicating that $\log p(\mathbf{x}) \geq \text{ELBO}(q)$, as illustrated in Figure 1.4, which justifies the goal of maximizing it.

A common approach is to choose a parameterized model $q_\theta(\mathbf{z} \mid \mathbf{x})$ for inference and another model $p_\phi(\mathbf{x} \mid \mathbf{z})$ for generation, where θ and ϕ denote the respective parameters. A widely adopted choice is to model both distributions as Gaussians $\mathcal{N}(\mu(\cdot), \Sigma(\cdot))$, where the functions $\mu(\cdot)$ and $\Sigma(\cdot)$ are implemented as neural networks. Figure 1.5 illustrates how a neural network can be used to implement the inference distribution as a Gaussian. The objective in Equation 5 can then be used to optimize both the inference and generative models, as is done in the training of Variational Autoencoders.

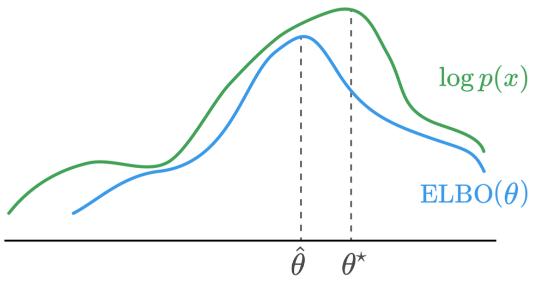
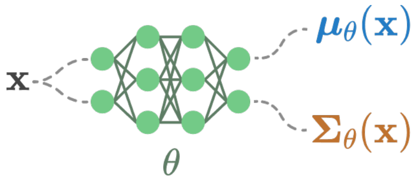


Figure 1.4. Illustration of the Evidence Lower Bound (ELBO) as a lower bound to the log-evidence $\log p(\mathbf{x})$. Maximizing the ELBO provides an approximation to the true posterior.


$\mathbf{x}$
$\boldsymbol{\mu}_\theta(\mathbf{x})$
$\boldsymbol{\Sigma}_\theta(\mathbf{x})$
θ

$$\mathcal{N}(\mathbf{z} \mid \boldsymbol{\mu}_\theta(\mathbf{x}), \boldsymbol{\Sigma}_\theta(\mathbf{x})) =: q_\theta(\mathbf{z} \mid \mathbf{x})$$

Figure 1.5. Neural network implementation of the inference model $q_\theta(\mathbf{z} \mid \mathbf{x})$ as a Gaussian distribution with learnable mean and variance.

1.2.4. History of Diffusion Models

The history of diffusion models begins with [Sohl-Dickstein et al. 2015], who proposed the idea of using a diffusion process to transform a simple prior distribution into the data distribution, where the transition kernel of this process is implemented using a neural network. This work establishes a connection between concepts from nonequilibrium thermodynamics and generative modeling.

After a four-year gap without major follow-up work, [Song and Ermon 2019] introduced a new approach to generative modeling: estimating the gradients (or score) of the data distribution and using iterative stochastic procedures (Langevin dynamics) to sample new data based on those estimated gradients. In this work, the authors employ denoising score matching [Vincent 2011] to train a neural network to estimate the score function. The method also leverages multiple noise scales to facilitate estimation—resulting in a strategy that, while starting from different motivations, ends up resembling the original diffusion framework.

Shortly thereafter, [Ho et al. 2020] popularized modern diffusion-based generative models, demonstrating how a simple neural network can produce high-quality image generations. This work introduced a loss function based on noise prediction, which outperformed previous approaches. It also established connections between diffusion probabilistic models, denoising score matching, and Langevin dynamics, and proposed optimizing a weighted variational bound that both improves sample quality and enhances likelihood estimation.

Following that, [Song et al. 2021b] unified diffusion-based models with continuous-time stochastic processes, enabling more flexible sampling strategies (e.g., Predictor-Corrector samplers). The paper also derived a neural ordinary differential equation (ODE) capable of generating samples from the same distribution as the reverse-time stochastic differential equation (SDE), while also allowing for exact likelihood computation and improved sampling efficiency.

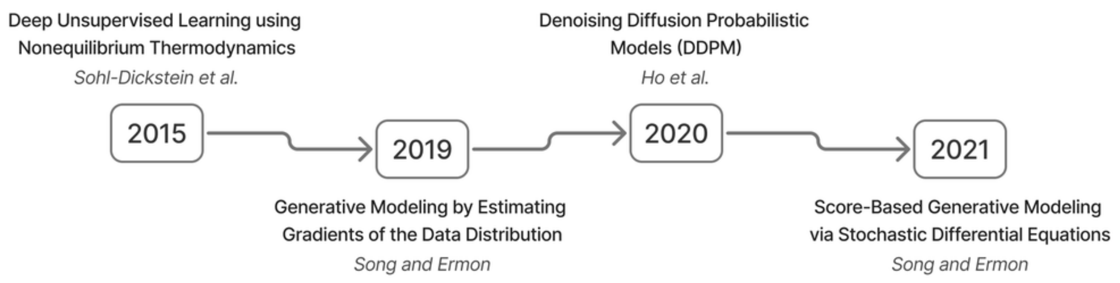

Deep Unsupervised Learning using Nonequilibrium Thermodynamics
Sohl-Dickstein et al.
2015
2019
Generative Modeling by Estimating Gradients of the Data Distribution
Song and Ermon
Denoising Diffusion Probabilistic Models (DDPM)
Ho et al.
2020
2021
Score-Based Generative Modeling via Stochastic Differential Equations
Song and Ermon

Figure 1.6. Timeline of major foundational contributions to diffusion-based generative models.

These four works constitute the foundation of the field of diffusion generative models. Figure 1.6 presents a timeline summarizing it. Since then, many other papers have focused on improving different components of the generative framework, including faster sampling [Song et al. 2021a, Lu et al. 2022, Lu et al. 2023], better conditioning [Dhariwal and Nichol 2021b, Ho and Salimans 2021, Chung et al. 2025, Tang et al. 2025], applications to inverse problems [Chung et al. 2022, Aali et al. 2023, Chung et al. 2023], and latent diffusion models [Rombach et al. 2021]. The volume of recent work in this area is extensive. A comprehensive review of these methods and their applications—ranging from computer vision, natural language processing, and time series modeling to interdisciplinary scientific domains—can be found in [Yang et al. 2024].

1.3. Denoising Diffusion Probabilistic Models

In this section, we present a formulation of diffusion models based on the work of [Ho et al. 2020], which frames diffusion models as latent variable models and employs variational inference to derive their objective function. The core idea is to progressively add noise to data (e.g., images, audio, or text) until it is transformed into pure random noise. This final state corresponds to an isotropic Gaussian distribution, where the variance is equal in all directions. The model aims to learn how to reverse this noising process. The forward and reverse stages are illustrated in Figure 1.7. By sampling from the Gaussian prior and applying the learned reverse process, generating new data that resembles samples from the original distribution is possible.

This section focuses on the key steps and formulas, omitting most derivations to emphasize the underlying motivation and how the core processes are formulated. A more complete treatment can be found in the original paper [Ho et al. 2020] and in books such as [Prince 2023, Bishop and Bishop 2024, Murphy 2022].

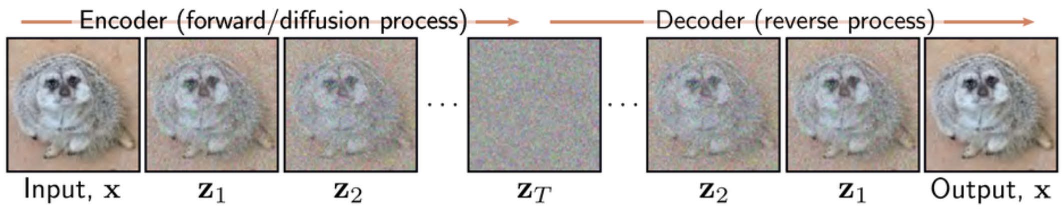


Figure 1.7. Illustration of the forward (noising) and reverse (denoising) diffusion processes used in DDPMs, starting from a data sample and ending in Gaussian noise, and vice versa. Image reproduced from the [Prince 2023].

1.3.1. Forward diffusion process

The forward diffusion process refers to the first stage of the diffusion model framework. In this step, noise is progressively added to the data until it becomes indistinguishable from pure random noise. The result is an isotropic Gaussian distribution with the same variance in all directions and zero covariance, implying that all dimensions are uncorrelated and statistically independent.

Let us define a series of discrete time steps $t = 1, 2, \ldots, T$, and let $\mathbf{x} \sim p(\mathbf{x})$ denote a sample drawn from the data distribution. For instance, $\mathbf{x}$ may be an original image from the training dataset. We introduce T latent variables $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T$, where $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

The noise is added through a Markov chain of diffusion steps, where each noisy version $\mathbf{z}_t$ depends only on the previous step $\mathbf{z}_{t-1}$. As a reminder, smaller values of t correspond to earlier steps in the process (i.e., less noise), and as $t \to T$, the distribution approaches an isotropic Gaussian. Figure 1.8 illustrates how this process unfolds for a simple 2D data distribution.

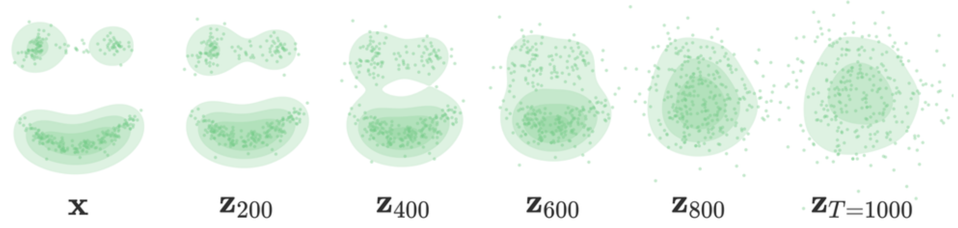


Figure 1.8. Illustration of the forward diffusion process on a 2D data distribution. The images were taken from an interactive visualization tool available at https://alechelbling.com/Diffusion-Explorer.

Then, let us define $q(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ as a Gaussian distribution function conditioned on the $\mathbf{z}_{t-1}$ version of our image, which outputs the noisier $\mathbf{z}_t$ version. We also define $\beta_t \in (0,1)$ as the variance schedule controlling the amount of noise added at time step t. We define the transition at each step as a conditional Gaussian distribution:

$$q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t \mid \sqrt{1-\beta_t}\mathbf{z}_{t-1}, \beta_t \mathbf{I}), \tag{6}$$

where the mean $\sqrt{1-\beta_t}\mathbf{z}_{t-1}$ is just a scaled version of the previous step $\mathbf{z}_{t-1}$ and the variance is β_t. The complete forward process is described as a sequence of conditional Gaussian distributions:

$$q(\mathbf{z}_{1:T} \mid \mathbf{x}) = q(\mathbf{z}_1 \mid \mathbf{x}) \prod_{t=2}^{T} q(\mathbf{z}_t \mid \mathbf{z}_{t-1}). \tag{7}$$

1.3.1.1. One step forward process

One key concept in the forward process is that we can directly find $\mathbf{z}_t$ given $\mathbf{x}$ without iterating over all intermediate time steps. This is only possible because the linear combination of independent Gaussian variables is also a Gaussian. Let us define $\alpha_t = 1 - \beta_t$, the cumulative product of $\alpha_{1:t}$ as $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ and present the reparametrization trick bellow formula:

$$\mathcal{N}(\mu, \sigma^2) = \mu + \sigma \cdot \varepsilon \text{ , where } \varepsilon \sim \mathcal{N}(0,1). \tag{8}$$

Then, we can write Equation 6 at some t as:

$$\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\mathbf{x} + \sqrt{1-\bar{\alpha}_t}\varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{9}$$

Finally, we can rewrite the equation in the original form:

$$q(\mathbf{z}_t \mid \mathbf{x}) = \mathcal{N}(\mathbf{z}_t \mid \sqrt{\bar{\alpha}_t}\mathbf{x}, (1-\bar{\alpha}_t)\mathbf{I}). \tag{10}$$

Note that, to get to any time step t, we only need the original sample image $\mathbf{x}$ and the cumulative noise. Figure 1.9 shows how $q(\mathbf{z}_t \mid \mathbf{x})$ evolves over time.

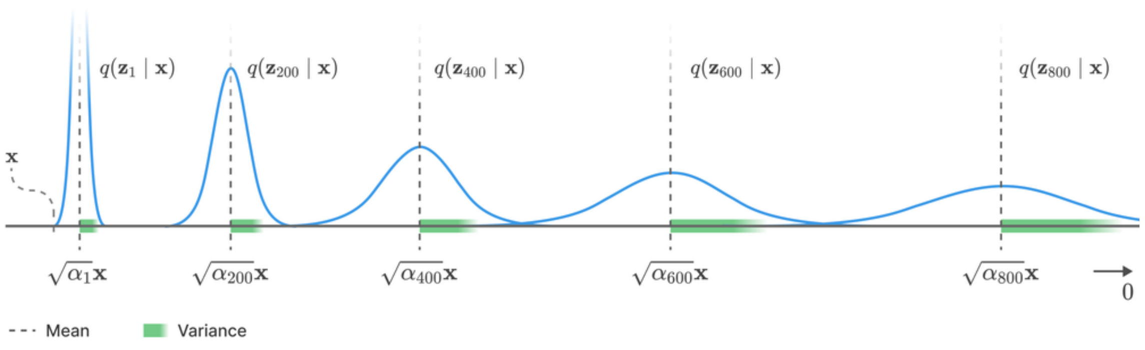


Figure 1.9. Variance evolution in the forward process as a function of time step t. As t increases, the distribution transitions from the original data to pure noise.

1.3.1.2. Noise schedule

In diffusion models, choosing the noise schedule $\{\beta_t\}_{t=1}^T$—which controls the variance added at each diffusion step—is a crucial design decision. The noise schedule directly influences the quality of training and sampling, as it determines how quickly the data distribution is destroyed in the forward process and how effectively the model can learn to reverse that process. Recall that each forward transition is modeled as: $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t \mid \sqrt{1-\beta_t}\mathbf{z}_{t-1}, \beta_t\mathbf{I})$ and the cumulative noise over time is encoded in $\bar{\alpha}_t = \prod_{i=1}^t (1-\beta_i)$.

A well-designed noise schedule adds small amounts of noise in early steps—preserving data structure and enabling meaningful denoising—and progressively increases it until the input becomes an isotropic Gaussian. The *linear schedule*, used in the original DDPM formulation [Ho et al. 2020], linearly increases β_t from a small to a large value, offering a simple and effective strategy. An alternative is the *cosine schedule*, proposed in [Nichol and Dhariwal 2021], which defines the cumulative noise $\bar{\alpha}_t$ via a cosine-shaped curve. This concentrates noise addition toward the end of the process, improving perceptual quality and training stability.

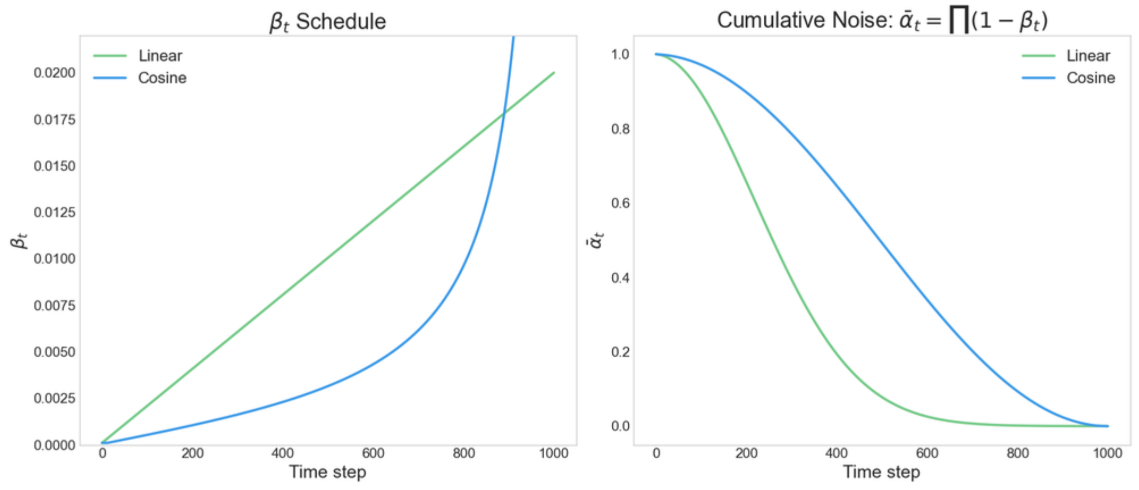


Figure 1.10. Comparison between linear and cosine noise schedules in diffusion models. The left plot shows the values of β_t over time; the right plot shows the cumulative product $\bar{\alpha}_t$, which determines how quickly the signal is destroyed.

Figure 1.10 compares linear and cosine schedules in terms of β_t and the cumulative product $\bar{\alpha}_t$ over time. While most diffusion models use predefined noise schedules, other approaches explore learning the noise schedule directly from data. For example, [Kingma et al. 2021] proposes a variational framework where the parameters of the forward process, including the variance schedule, are optimized jointly with the generative model. In summary, the choice of noise schedule affects the learning dynamics during training and plays a key role in how smoothly the model can reverse the diffusion trajectory during sampling.

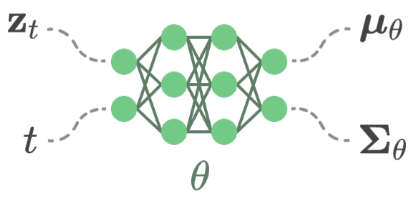


Figure 1.11. Gaussian kernel used in the reverse diffusion process to model $p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t)$.

1.3.2. Reverse diffusion process

In the reverse diffusion process, the goal is to sample from the distribution $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t)$, given that we know how to sample from the forward distribution $q(\mathbf{z}_t \mid \mathbf{z}_{t-1})$. This makes it possible to recover a sample $\mathbf{x} \sim q(\mathbf{x})$ from the original data distribution, starting from a pure noise input $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

A natural first attempt is to apply Bayes' theorem:

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t) = \frac{q(\mathbf{z}_t \mid \mathbf{z}_{t-1})q(\mathbf{z}_{t-1})}{q(\mathbf{z}_t)}. \tag{11}$$

However, as discussed in Section 1.2.2, the marginal distribution $q(\mathbf{z}_t)$ is intractable. To address this, we use variational inference and introduce a surrogate model that approximates the target distribution:

$$q(\mathbf{z}_{1:T}, \mathbf{x}) = q(\mathbf{x})q(\mathbf{z}_1 \mid \mathbf{x}) \prod_{t=2}^{T} q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \tag{12}$$

with the surrogate defined as:

$$p_\theta(\mathbf{z}_{1:T}, \mathbf{x}) = p(\mathbf{z}_T)p_\theta(\mathbf{x} \mid \mathbf{z}_1) \prod_{t=2}^{T} p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t), \tag{13}$$

where the reverse transitions are modeled as Gaussian distributions:

$$p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t) = \mathcal{N}(\mathbf{z}_{t-1} \mid \mu_\theta(\mathbf{z}_t, t), \Sigma_\theta(\mathbf{z}_t, t)). \tag{14}$$

Here, θ represents the learnable parameters. The functions μ_θ and Σ_θ can be implemented as neural networks, as illustrated in Figure 1.11. However, as we will see, this formulation can be simplified so that a neural network does not need to model the full Gaussian parameters directly.

We now use the KL divergence to measure the similarity between the true and surrogate distributions, resulting in the following optimization problem:

$$\hat{\theta} := \arg\min_\theta D_{\mathrm{KL}}(q(\mathbf{x}, \mathbf{z}_{1:T}) \parallel p_\theta(\mathbf{x}, \mathbf{z}_{1:T})). \tag{15}$$

From Equations 4 and 5, this leads to the maximization of the Evidence Lower Bound (ELBO):

$$\hat{\theta} = \arg\max_\theta \underbrace{\mathbb{E}_{q(\mathbf{x}, \mathbf{z}_{1:T})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z}_{1:T})}{q(\mathbf{z}_{1:T} \mid \mathbf{x})}\right]}_{\text{ELBO}}. \tag{16}$$

This ELBO resembles Section 1.2.3, but now with T latent variables. After derivation, the resulting objective becomes:

$$\hat{\theta} = \arg\max_{\theta} \underbrace{\mathbb{E}_{q(\mathbf{x},\mathbf{z}_1)}\left[p_\theta(\mathbf{x} \mid \mathbf{z}_1)\right]}_{L_0} + \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x},\mathbf{z}_t)}\left[D_{\mathrm{KL}}\left(q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x}) \parallel p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t)\right)\right]}_{L_1, L_2, \ldots, L_{T-1}}. \quad (17)$$

We can now use this objective to optimize the parameters θ of the reverse Gaussian distribution from Equation 14. The KL divergence between two multivariate Gaussians admits a closed-form solution, and each L_t term from Equation 17 becomes:

$$L_t = D_{\mathrm{KL}}\left(q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x}) \parallel p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t)\right) \quad (18)$$

$$= D_{\mathrm{KL}}(\mathcal{N}(\mathbf{z}_{t-1} \mid \tilde{\mu}(\mathbf{z}_t, \mathbf{x}), \tilde{\beta}_t \mathbf{I}) \parallel \mathcal{N}(\mathbf{z}_{t-1} \mid \mu_\theta(\mathbf{z}_t, t), \Sigma_\theta(\mathbf{z}_t, t))), \quad (19)$$

where

$$\tilde{\mu}_t(\mathbf{z}_t, \mathbf{x}) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{z}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_t\right), \quad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t, \quad (20)$$

and ε_t comes from the Equation 9. The equalities of Equation 20 come from the definition of the Gaussian density function, assuming the covariance matrix is diagonal $\tilde{\beta}_t \mathbf{I}$.

Algorithm 1 Algorithm for training a denoising diffusion probabilistic model, originally described in [Bishop and Bishop 2024].

```
1:  **Input:** Training data $\mathcal{D} = \{x_n\}$, Noise schedule $\{\beta_1, \ldots, \beta_T\}$
2:  **Output:** Network parameters $\theta$
3:  **for** $t \in \{1, \ldots, T\}$ **do**
4:      $\alpha_t \leftarrow \prod_{\tau=1}^{t}(1-\beta_\tau)$                ▷ Calculate alphas from betas
5:  **end for**
6:  **repeat**
7:      $\mathbf{x} \sim \mathcal{D}$                                      ▷ Sample a data point
8:      $t \sim \{1, \ldots, T\}$                                ▷ Sample a point along the Markov chain
9:      $\varepsilon \sim \mathcal{N}(\varepsilon \mid \mathbf{0}, \mathbf{I})$                          ▷ Sample a noise vector
10:     $\mathbf{z}_t \leftarrow \sqrt{\alpha_t}\mathbf{x} + \sqrt{1-\alpha_t}\,\varepsilon$                   ▷ Evaluate noisy latent variable
11:     $\mathcal{L}(\theta) \leftarrow \|\varepsilon_\theta(\mathbf{z}_t, t) - \varepsilon\|^2$                  ▷ Compute loss term
12:     Take optimizer step
13: **until** converged
14: **return** $\theta$
```

We must learn a neural network to approximate the conditioned probability distributions in the reverse diffusion. We would like to train μ_θ to predict $\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{z}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_t\Big)$. Because $\mathbf{z}_t$ is available as input at training time, we can reparameterize the Gaussian noise term instead to make it predict ε_t from the input $\mathbf{z}_t$ at time step t:

$$\mu_\theta(\mathbf{z}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{z}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(\mathbf{z}_t, t)\right). \quad (21)$$

Given that, L_t becomes:

$$L_t(\theta) = \mathbb{E}_{\mathbf{x},\varepsilon}\left[\frac{1}{2\|\Sigma_\theta(\mathbf{z}_t,t)\|_2^2}\|\tilde{\mu}_t(\mathbf{z}_t,\mathbf{x}) - \mu_\theta(\mathbf{z}_t,t)\|^2\right] \tag{22}$$

$$= \mathbb{E}_{\mathbf{x},\varepsilon}\left[\frac{1}{2\|\Sigma_\theta(\mathbf{z}_t,t)\|_2^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{z}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_t\right) - \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{z}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(\mathbf{z}_t,t)\right)\right\|^2\right] \tag{23}$$

$$= \mathbb{E}_{\mathbf{x},\varepsilon}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\Sigma_\theta(\mathbf{z}_t,t)\|_2^2}\|\varepsilon_t - \varepsilon_\theta(\mathbf{z}_t,t)\|^2\right]. \tag{24}$$

Empirically, [Ho et al. 2020] found that training works better when the weighting term is removed, leading to the simplified loss:

$$L_t^{\text{simple}}(\theta) = \mathbb{E}_{\mathbf{x},\varepsilon_t}\left[\|\varepsilon_t - \varepsilon_\theta(\mathbf{z}_t,t)\|^2\right] \tag{25}$$

$$= \mathbb{E}_{\mathbf{x},\varepsilon_t}\left[\|\varepsilon_t - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x} + \sqrt{1-\bar{\alpha}_t}\varepsilon_t, t)\|^2\right]. \tag{26}$$

Thus, the final complete objective is:

$$\mathcal{L}(\theta) := \mathbb{E}_{t\sim[1,T],\mathbf{x},\varepsilon_t}\left[\|\varepsilon_t - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x} + \sqrt{1-\bar{\alpha}_t}\varepsilon_t, t)\|^2\right]. \tag{27}$$

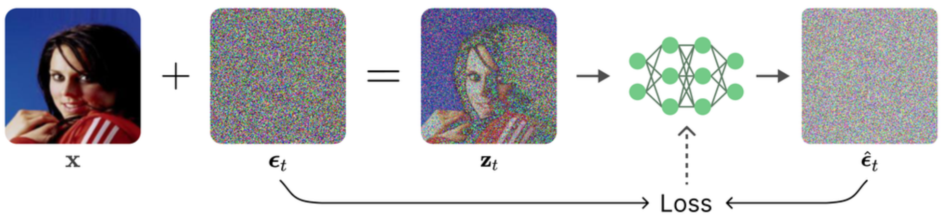


Figure 1.12. Simplified training objective used in DDPM, where the network ε_θ learns to predict the added noise ε_t given z_t and t.

This final training objective from Equation 25 is illustrated in Figure 1.12. Algorithm 1 brings the steps to train the diffusion model using this loss, while Algorithm 2 shows how to generate new data using the trained model ε_θ.

With the DDPM objective established, the next sections explore an alternative formulation (score-based models), practical implementations, and improvements to the base model.

1.4. Diffusion Models as score-based models

Diffusion models belong to the broader class of score-based generative models. The term *score* comes from the (Stein) score function [Cox and Hinkley 1974] that is the gradient of the log probability density function $\nabla \log p(\mathbf{x})$. The score function is a vector field pointing to places with higher data density. Estimating this function is helpful due to methods such as *stochastic gradient Langevin dynamics*, which is a Markov Chain Monte

Algorithm 2 Algorithm for sampling from a denoising diffusion probabilistic model, originally described in [Bishop and Bishop 2024].

```
1: Input: Trained denoising network $\varepsilon_\theta(\mathbf{z},t)$, Noise schedule $\{\beta_1,\dots,\beta_T\}$
2: Output: Sample vector $\mathbf{x}$ in data space
3: $\mathbf{z}_T \sim \mathcal{N}(\mathbf{z} \mid \mathbf{0},\mathbf{I})$                                   ▷ Sample from final latent space
4: for $t \in \{T,\dots,2\}$ do
5:     $\alpha_t \leftarrow \prod_{\tau=1}^{t}(1-\beta_\tau)$                                   ▷ Calculate alpha
6:     $\mu_\theta(\mathbf{z}_t,t) \leftarrow \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\varepsilon_\theta(\mathbf{z}_t,t)\right)$   ▷ Evaluate network output
7:     $\varepsilon \sim \mathcal{N}(\varepsilon \mid \mathbf{0},\mathbf{I})$                                   ▷ Sample a noise vector
8:     $\mathbf{z}_{t-1} \leftarrow \mu_\theta(\mathbf{z}_t,t) + \sqrt{\beta_t}\,\varepsilon$                          ▷ Add scaled noise
9: end for
10: $\mathbf{x} \leftarrow \frac{1}{\sqrt{1-\beta_1}}\left(\mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}}\varepsilon_\theta(\mathbf{z}_1,t)\right)$              ▷ Final denoising step
11: return $\mathbf{x}$
```

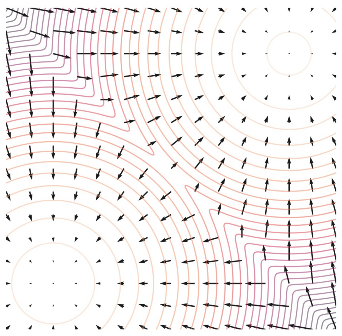


Figure 1.13. A vector field representing the score function and contours representing the density function of a mixture of two Gaussians. Source: https://yang-song.net/blog/2021/score.

Carlo (MCMC) procedure that can sample from a distribution $p(\mathbf{x})$ using only its score function. Figure 1.13 shows a representation of both score and density functions.

To estimate the score, a series of techniques called Score Matching is used. We can train a model to do so by minimizing the Fisher divergence between the model and the score of the data distribution, defined as:

$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2]. \tag{28}$$

The Fisher divergence compares the squared distance between the ground-truth data score and the score-based model. Directly computing this divergence, however, is infeasible because it requires access to the unknown data score $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. That is the reason we need score-matching. Many works have proposed different strategies to perform this score estimation [Hyvärinen 2005, Vincent 2011, Song et al. 2019], but here we will focus on the *denoising score matching*.

1.4.1. Denoising score matching

Denoising score matching [Vincent 2011] is a practical variation of the original score matching technique. The core idea is simple: instead of estimating the score of the original data distribution directly (something often challenging), we first corrupt the data by adding a known amount of noise. Formally, we define a noise distribution $q(\mathbf{z} \mid \mathbf{x})$ that perturbs the original data point $\mathbf{x}$ into a noisy version $\mathbf{z}$.

By applying score matching to this noisy distribution $q(\mathbf{z})$, we aim to train a model $\mathbf{s}_\theta(\mathbf{z})$ that estimates the score of $q(\mathbf{z})$. The objective function turns out to be equivalent to minimizing the difference between the predicted score and the score of the noise distribution itself:

$$\frac{1}{2}\mathbb{E}_{q(\mathbf{z}|\mathbf{x})p_{\text{data}}(\mathbf{x})}\left[\|\mathbf{s}_\theta(\mathbf{z}) - \nabla_\mathbf{z}\log q(\mathbf{z} \mid \mathbf{x})\|_2^2\right]. \tag{29}$$

Intuitively, this means the model learns to "denoise" $\mathbf{z}$ by estimating in which direction the noise moved it from its original clean version $\mathbf{x}$. The optimal solution $\mathbf{s}_{\theta^*}$ to this objective satisfies $\mathbf{s}_{\theta^*}(\mathbf{z}) = \nabla_\mathbf{z}\log q(\mathbf{z})$ almost everywhere, as proved in [Vincent 2011]. Moreover, when the noise level is small, the perturbed distribution $q(\mathbf{z})$ closely approximates the true data distribution $p_{\text{data}}(\mathbf{x})$, so their gradients are also close $\nabla_\mathbf{z}\log q(\mathbf{z}) \approx \nabla_\mathbf{x}\log p_{\text{data}}(\mathbf{x})$.

A common choice for the noise distribution is a simple Gaussian centered at the original data point $q(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\mathbf{z} \mid \mathbf{x}, \sigma^2 I)$, for which we can compute the score analytically:

$$\nabla_\mathbf{z}\log q(\mathbf{z} \mid \mathbf{x}) = -\frac{\mathbf{z}-\mathbf{x}}{\sigma^2}. \tag{30}$$

Plugging this into the objective, we obtain the final denoising score matching loss for a fixed noise level σ:

$$\frac{1}{2}\mathbb{E}_{\mathbf{x},\mathbf{z}\sim\mathcal{N}(\mathbf{x},\sigma^2 I)}\left[\left\|\mathbf{s}_\theta(\mathbf{z},\sigma) + \frac{\mathbf{z}-\mathbf{x}}{\sigma^2}\right\|_2^2\right]. \tag{31}$$

In this formulation, the model learns to predict the direction and intensity of the noise that was added, which effectively teaches it the gradient of the log-density of the data through supervised learning on noisy inputs.

1.4.2. Sampling with Langevin dynamics

Langevin dynamics (LD) provides a way to generate samples from a probability distribution $p(\mathbf{x})$ using only its score function, $\nabla_\mathbf{x}\log p(\mathbf{x})$. The idea is to start from a random point $\tilde{\mathbf{x}}_0$ drawn from some simple prior distribution $\pi(\mathbf{x})$, and then iteratively refine it using both the score function and added noise. Given a fixed step size $\alpha > 0$, the Langevin update rule is:

$$\tilde{\mathbf{x}}_i = \tilde{\mathbf{x}}_{i-1} + \frac{\alpha}{2}\nabla_\mathbf{x}\log p(\tilde{\mathbf{x}}_{i-1}) + \sqrt{\alpha}\,\omega_i, \quad i = 0, 1, \cdots, K, \tag{32}$$

where $\omega_t \sim \mathcal{N}(0, I)$ is Gaussian noise. Intuitively, each update step moves the sample in the direction where the data density $p(\mathbf{x})$ increases (guided by the score), while adding a small amount of random noise to explore the space. Figure 1.14 shows a visualization of Langevin dynamics sampling.

Suppose we apply this update many times (i.e., letting $K \to \infty$) and use a tiny step size ($\alpha \to 0$). In that case, the final sample $\tilde{\mathbf{x}}_K$ will converge to an actual sample from $p(\mathbf{x})$, under certain mathematical conditions. When α is small and K is large, this procedure can

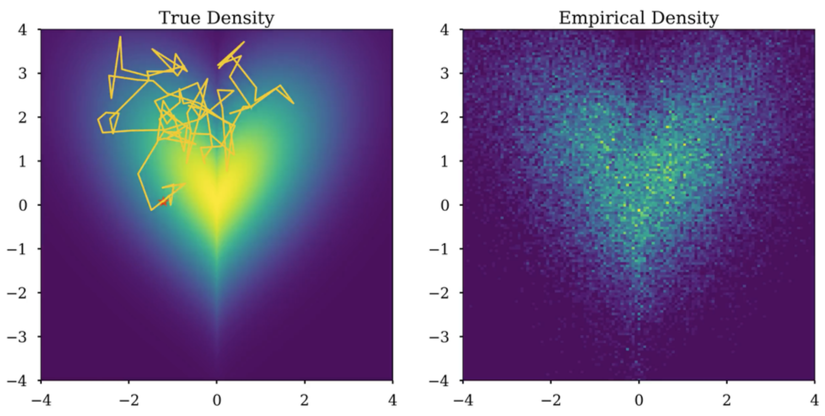


Figure 1.14. Visualization of Langevin dynamics sampling. Left: The true data density is shown as a heatmap, with yellow indicating higher probability. The orange trajectory represents a single sample path generated using LD. Right: The empirical density obtained from multiple samples, showing that the sampling process successfully approximates the true data distribution.

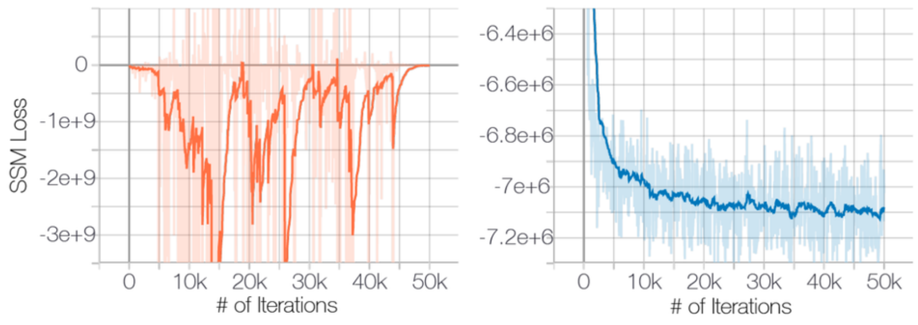


Figure 1.15. Effect of the manifold hypothesis. Left: SSM fails on unperturbed CIFAR-10. Right: Adding Gaussian noise $\mathcal{N}(0, 0.0001)$ enables convergence by giving full support over $\mathbb{R}^D$.

produce high-quality approximate samples even without applying a Metropolis-Hastings correction step, which is sometimes needed to ensure exactness.

An important observation is that the update rule above only depends on the score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. This means that to sample from the true data distribution $p_{\text{data}}(\mathbf{x})$, we can first train a neural network score estimator $\mathbf{s}_\theta(\mathbf{x})$ to approximate this gradient. Once trained, we can use $\mathbf{s}_\theta(\mathbf{x})$ in place of the true score in Langevin dynamics to generate samples that follow $p_{\text{data}}(\mathbf{x})$. This sampling strategy is a key component of the approach known as *score-based generative modeling*.

1.4.3. Limitations of score matching and Langevin dynamics

While score-based generative modeling provides an elegant framework for sampling from complex distributions, it faces challenges when data lies on low-dimensional structures or in sparse regions.

Score undefinedness. Real-world data often lies on lower-dimensional structures in a high-dimensional space (the Manifold Hypothesis [Fefferman et al. 2013]), making the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ undefined off the manifold. Score matching requires full support over $\mathbb{R}^D$, which is rarely satisfied. A common solution is to add slight Gaussian noise to the data, ensuring tractability (Figure 1.15).

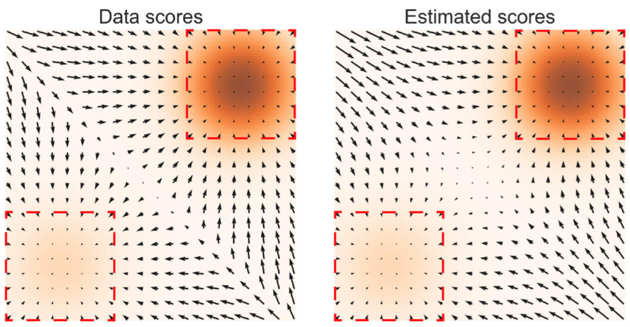


Figure 1.16. Score estimation in a Gaussian mixture. Left: Ground-truth scores. Right: Estimated scores. Accuracy is high only near the dense modes.

Low-density regions. Score matching minimizes a squared error weighted by $p_{\text{data}}(\mathbf{x})$, so errors in low-density areas contribute little:

$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2] = \int p(\mathbf{x}) \| \cdot \|^2 \, d\mathbf{x}.$$

As a result, the model may fail to learn accurate scores in these regions, which can affect global structure (Figure 1.16).

Mode balancing failure in LD. Langevin Dynamics struggles with multimodal distributions, especially when modes are separated by low-density regions. For $p_{\text{data}} = \pi p_1 + (1-\pi) p_2$ with disjoint supports, the score is locally accurate but contains no information about π. LD cannot recover mixture proportions and requires tiny steps to bridge modes. Standard LD fails under these conditions, unlike annealed variants (Figure 1.17).

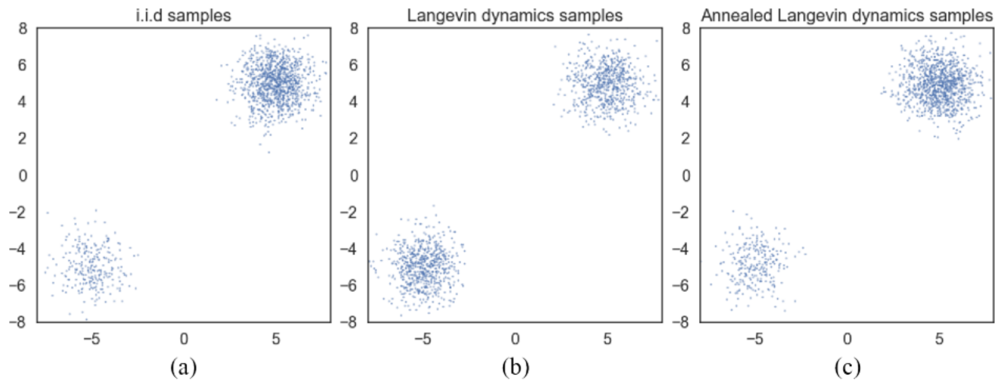


Figure 1.17. Langevin Dynamics vs. Annealed LD. LD fails to reflect correct mode proportions.

1.4.4. Multiple noise scales

One challenge in the presented strategy is choosing the right noise level for perturbing the data. Large noise values help cover low-density regions but overly distort the data, while small noise preserves details but fails to provide coverage in sparse areas.

To address this, we perturb the data with multiple noise scales simultaneously. Let $\{\sigma_1, \sigma_2, \ldots, \sigma_T\}$ be an increasing sequence of standard deviations, and define the perturbed distributions as $q(\mathbf{z}_t \mid \mathbf{x}) = \mathcal{N}(\mathbf{z}_t \mid \mathbf{x}, \sigma_t^2 \mathbf{I})$. Sampling from $q(\mathbf{z}_t \mid \mathbf{x})$ is straightforward: draw $\mathbf{x} \sim p_{\text{data}}$ and add Gaussian noise $\sigma_t \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, I)$.

We then train a score network $\mathbf{s}_\theta(\mathbf{x}, \sigma)$ to estimate the score $\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t \mid \mathbf{x})$ for all noise levels σ_t, using denoising score matching:

$$L_t(\theta) = \frac{1}{2} \mathbb{E}_{\mathbf{x}, \mathbf{z}_t \sim \mathcal{N}(\mathbf{x}, \sigma_t^2 I)} \left[\left\| \mathbf{s}_\theta(\mathbf{z}_t, \sigma_t) + \frac{\mathbf{z}_t - \mathbf{x}}{\sigma_t^2} \right\|_2^2 \right].$$

The full loss combines all scales with a weighting function $w(\sigma)$:

$$\mathcal{L}(\theta) = \frac{1}{2} \mathbb{E}_{t \sim [1,T], \mathbf{x}, \mathbf{z}_t \sim \mathcal{N}(\mathbf{x}, \sigma_t^2 I)} \left[w(\sigma_t) \cdot \left\| \mathbf{s}_\theta(\mathbf{z}_t, \sigma_t) + \frac{\mathbf{z}_t - \mathbf{x}}{\sigma_t^2} \right\|_2^2 \right]. \tag{33}$$

Empirically, it is observed that $\|\mathbf{s}_\theta(\mathbf{x}, \sigma)\|_2 \propto 1/\sigma$, suggesting $w(\sigma) = \sigma^2$ balances the magnitude across scales.

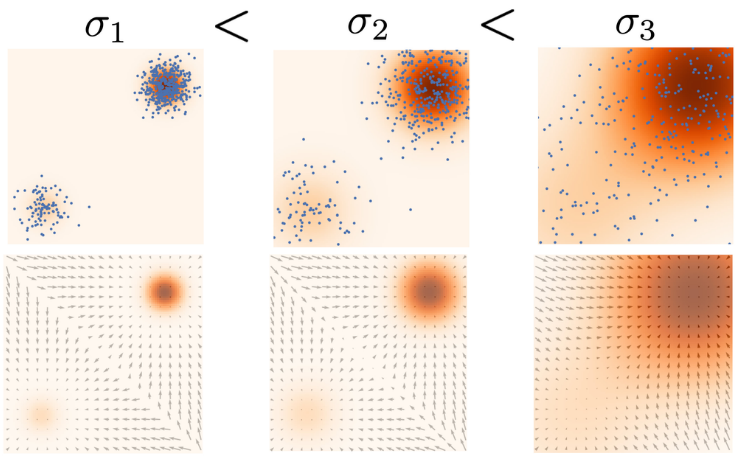


Figure 1.18. Score estimation at multiple noise levels. For each noise scale σ_t, a score network $\mathbf{s}_\theta(\mathbf{x}, \sigma_t)$ is trained to approximate the score $\nabla_{\mathbf{z}_t} \log q(\mathbf{z}_t \mid \mathbf{x})$ of the corresponding noisy distribution. The top row shows the perturbed data distributions for increasing noise levels $\sigma_1 < \sigma_2 < \sigma_3$, and the bottom row shows the estimated scores overlaid on the data density.

This training setup defines the *Noise Conditional Score Network* (NCSN), a model that can estimate the score of perturbed data distributions across multiple noise levels. Figure 1.18 shows a visualization comparing the different noise scales and the estimated score for each scale.

As shown by [Kingma and Gao 2023], the noise prediction loss from Equation 27 is equivalent to the score matching objective when $w(\sigma_t) = \sigma_t^2$.

1.4.4.1. Annealed Langevin dynamics for sampling

Algorithm 3 Annealed Langevin Dynamics (adapted from [Song et al. 2021b])

```
1:  Input: Noise levels $\{\sigma_t\}_{t=1}^{T}$, step size parameter $\alpha$, number of steps $K$
2:  Output: Sample vector $\mathbf{x}$ from the target distribution
3:  Initialize $\tilde{\mathbf{z}}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$                        ▷ Initial Gaussian noise
4:  for $t \in \{1, \dots, T\}$ do
5:      $\varepsilon_t \leftarrow \alpha \cdot \sigma_t^2 / \sigma_T^2$                         ▷ Compute adaptive step size
6:      for $i \in \{1, \dots, K\}$ do
7:          $\omega_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$                                 ▷ Sample Gaussian noise
8:          $\tilde{\mathbf{z}}_i \leftarrow \tilde{\mathbf{z}}_{i-1} + \frac{\varepsilon_t}{2} \mathbf{s}_\theta(\tilde{\mathbf{z}}_{i-1}, \sigma_t) + \sqrt{\varepsilon_t}\, \omega_i$     ▷ Langevin update step
9:      end for
10:     $\tilde{\mathbf{z}}_0 \leftarrow \tilde{\mathbf{z}}_K$                                    ▷ Prepare for next noise scale
11: end for
12: $\mathbf{x} \leftarrow \tilde{\mathbf{z}}_K$
13: return $\mathbf{x}$
```

Once trained, the NCSN can generate data samples via *annealed Langevin dynamics*. This method applies Langevin sampling iteratively, starting from a high noise level σ_T and gradually reducing it down to σ_1. The complete procedure is presented in Algorithm 3. This annealing procedure allows the model to explore the space broadly and then refine details at lower noise levels, overcoming the limitations of standard Langevin dynamics in low-density or disconnected regions.

1.5. Denoising neural network architecture

The denoising neural network is the core component of diffusion models. It learns to reverse the corruption process by estimating the noise added to the data at a given noise level. This model is trained using pairs of noisy inputs and corresponding targets, and is used during sampling to progressively reconstruct clean data from noise.

The efficacy of a diffusion model is critically dependent on the capacity of its neural network to accurately predict the noise component ε from a noisy input $\mathbf{z}_t$ at a given timestep t. This network, denoted as $\varepsilon_\theta(\mathbf{z}_t, t)$ or $\mathbf{s}_\theta(\mathbf{z}_t, \sigma_t)$, serves as the parameterized function approximator that learns to reverse the forward diffusion process. While various architectures could theoretically be employed, the de facto standard, established by [Ho et al. 2020] and since refined, is a time-conditional U-Net architecture. This choice is motivated by the U-Net's proven success in image-to-image translation tasks, its inherent multi-scale processing, and its ability to preserve high-fidelity spatial information.

1.5.1. The U-Net backbone

The foundational architecture is the U-Net, originally proposed by [Ronneberger et al. 2015] for biomedical image segmentation. Its design consists of two main paths: a contracting (or encoder) path and an expansive (or decoder) path, forming a characteristic "U" shape (see Figure 1.19).

The contracting path follows a typical convolutional network structure. It consists

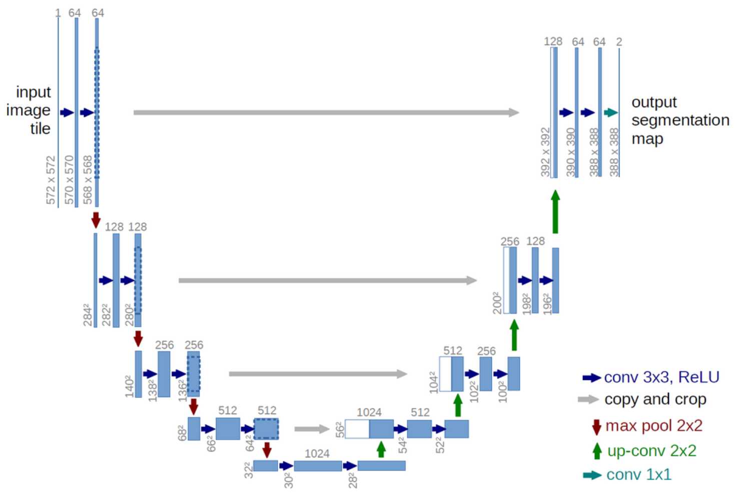


Figure 1.19. The illustration of the original U-Net architecture. The U-shape is clearly visible with a contracting path on the left and an expansive path on the right. Arrows indicate skip connections from the encoder blocks to the decoder blocks. Source: [Ronneberger et al. 2015].

of repeated blocks of standard operations, typically two 3x3 convolutions, each followed by a normalization layer and a non-linear activation function. After each block, a down-sampling operation halves the spatial dimensions of the feature maps while doubling the number of feature channels. This process allows the network to capture contextual information at progressively coarser spatial resolutions.

The expansive path is a mirror image of the contracting path. It systematically upsamples the feature maps, typically using a transposed convolution or an upsample-then-convolve operation. A crucial innovation of the U-Net is the use of skip connections, which concatenate the feature maps from the corresponding level in the contracting path with the upsampled feature maps in the expansive path. These connections provide the decoder with high-resolution spatial information from the early layers, which is vital for precise localization and the reconstruction of fine-grained details that would otherwise be lost during downsampling. The final layer maps the feature channels to the desired output size—in this case, the predicted noise map ε_θ.

1.5.2. Architectural enhancements for diffusion

The standard U-Net is adapted and enhanced with several critical components to meet the specific demands of the diffusion process. These enhancements are key to the remarkable performance of modern generative models.

Timestep conditioning. The network's prediction must depend on the timestep t, as the noise to be estimated varies with it. Simply feeding t as an integer is ineffective. Inspired by Transformer positional encodings [Vaswani et al. 2017], a sinusoidal embedding transforms t into a high-dimensional vector, which is processed by a small MLP and added to the feature maps at each residual block. This lets the network modulate its behavior across the hierarchy according to the noise level.

Attention mechanisms. Convolutions are effective for local patterns but have

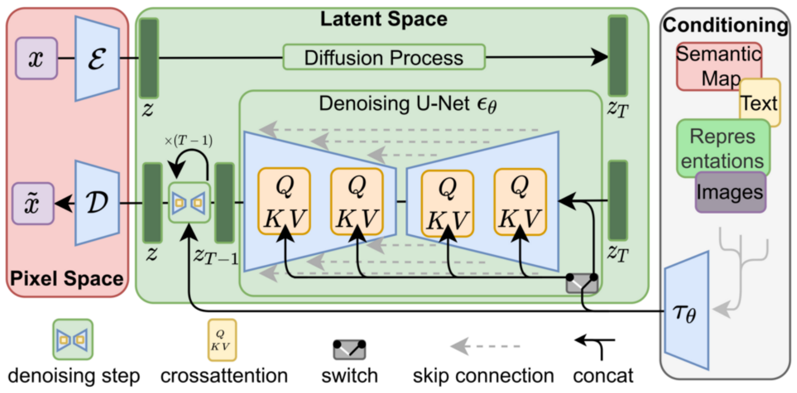


Figure 1.20. Overview of the Stable Diffusion architecture. A denoising UNet operates in latent space, guided by cross-attention over conditioning inputs such as text or image features. The process starts from Gaussian noise and iteratively reconstructs a latent representation, which is decoded into pixel space. Source: [Rombach et al. 2021].

limited receptive fields. To capture long-range dependencies and global context, self-attention blocks are inserted into lower-resolution layers of the U-Net [Dhariwal and Nichol 2021a], where cost is lower and global information is more critical. These blocks enable interactions between distant spatial locations, improving structure, symmetry, and coherence.

Conditional generation via cross-attention. For tasks like text-to-image synthesis, the U-Net is conditioned on external inputs via cross-attention, a key feature of the Latent Diffusion Model (LDM) [Rombach et al. 2021]. Here, the spatial feature maps from the U-Net act as the query vectors, while the conditioning vector (e.g., a text embedding from a pre-trained CLIP model) provides the key and value vectors. Cross-attention layers are distributed across the U-Net, typically alongside self-attention, aligning the denoising process with semantic guidance across scales and effectively steering the generation. Figure 1.20 shows what this conditioning looks like.

1.6. Architectural and methodological advancements

Since their initial formulation, as detailed in Sections 1.3 and 1.4, diffusion models have undergone numerous enhancements aimed at improving sample quality, generation speed, and conditional control. These advancements have transformed them from theoretical curiosities into state-of-the-art generative tools. In this section, we highlight several seminal lines of improvement.

1.6.1. Accelerating sampling via advanced solvers

A primary limitation of early diffusion models like DDPMs was the computationally expensive sampling process, often requiring hundreds or even thousands of sequential network evaluations. Considerable effort has been devoted to accelerating this process without degrading sample quality.

The work of [Song et al. 2021b] introduced a foundational view by framing diffusion as a stochastic differential equation (SDE), showing that DDPMs' discrete forward and reverse processes are discretizations of a continuous-time SDE. This yields a reverse-time ordinary differential equation (ODE) whose solution maps noise into data, reframing sampling as a numerical ODE-solving problem and enabling the use of advanced solvers.

One of the earliest and most impactful methods derived from this view was the Denoising Diffusion Implicit Model (DDIM) [Song et al. 2021a], which defined a non-Markovian forward process leading to a deterministic reverse path. Unlike the stochastic, step-by-step sampling of DDPMs, DDIM allows for large, deterministic transitions through latent space, drastically reducing the number of steps (e.g., from 1000 to 50) while maintaining competitive image quality.

Expanding on this ODE-based direction, DPM-Solver [Lu et al. 2022] and DPM-Solver++ [Lu et al. 2023] developed high-order, semi-linear solvers tailored to the structure of the diffusion ODE. These solvers analytically handle the linear components and approximate the non-linear score term using high-order polynomials. As a result, they achieve accurate integration with just 10–20 steps—an order-of-magnitude improvement over earlier approaches.

More recently, Rectified Flow [Liu et al. 2023] proposed a conceptual shift: instead of simulating denoising, it learns a velocity field that maps noise to data along nearly straight lines in latent space. This simplification allows even basic Euler solvers to perform well, offering both speed and a fresh generative modeling paradigm.

1.6.2. Enhancing controllability with guidance

Enabling precise user control over the generated output is essential for many applications. A key milestone in this direction was Classifier Guidance [Dhariwal and Nichol 2021b], which introduced a mechanism to steer generation using gradients from an external, pre-trained classifier. At each denoising step, the gradient of the classifier's output with respect to the current noisy sample $\mathbf{z}_t$ is added to the diffusion model's noise estimate, effectively guiding the generation towards a desired class.

While effective, this approach depends on a separate noise-aware classifier. Classifier-Free Guidance (CFG) [Ho and Salimans 2021] offered a more practical and now widely adopted alternative. It eliminates the need for an external classifier by training the diffusion model on both conditional inputs (e.g., text prompts) and a null or "unconditional" context. During inference, the model computes both a conditional noise prediction $\varepsilon_\theta(\mathbf{x}_t, c)$ and an unconditional one $\varepsilon_\theta(\mathbf{x}_t, \emptyset)$, and then interpolates between them. This mechanism significantly boosts controllability and fidelity, becoming a standard component in modern systems like Stable Diffusion.

Recent works, including Extended Classifier-Free Guidance [Chung et al. 2025] and Free Guidance [Tang et al. 2025], have refined CFG by analyzing its trade-offs. These studies noted issues like mode collapse at high guidance scales and proposed sampling adjustments to balance prompt adherence and output diversity. Rather than introducing a new paradigm, their contribution lies in improving CFG's robustness and generalization across use cases.

1.6.3. Efficient high-resolution synthesis in latent space

Scaling diffusion models to high-resolution images is computationally prohibitive in pixel space due to the quadratic scaling of self-attention and memory costs. Latent Diffusion Models (LDMs) [Rombach et al. 2021] provided a landmark contribution by shifting the diffusion process from the high-dimensional pixel space to a compact, learned latent space. Their approach is a two-stage process: first, a powerful autoencoder is trained to compress images into a semantically rich but spatially smaller latent representation. Second, a diffusion model is trained exclusively in this latent space. By operating on these compressed latents, the LDM can generate high-resolution outputs with a fraction of the computational resources required by pixel-space models. This decoupling of perceptual compression and generative modeling was a key enabler for the widespread accessibility of high-quality image synthesis.

1.6.4. Architectural evolution: beyond the U-Net

A seminal contribution from [Peebles and Xie 2023] was the introduction of the Diffusion Transformer (DiT). This work challenged the necessity of the convolutional U-Net backbone. Their key insight was to replace the U-Net entirely with a standard Transformer architecture. The model operates on latent patches of the image, similar to a Vision Transformer [Dosovitskiy et al. 2021]. They demonstrated that a well-designed Transformer, when scaled up in terms of depth, width, and training data, can outperform the best U-Net-based diffusion models on class-conditional image synthesis benchmarks. This was a landmark result, suggesting that a unified Transformer-based architecture could be a scalable and superior foundation for future generative models across modalities.

1.7. Applications of Diffusion Models

Table 1.1. Prominent Techniques and Their Application in Industrial Diffusion Models

Technique	Industrial Solution / Model	Resource
Latent Diffusion	Stable Diffusion (Stability AI, Runway, LMU Munich)	[Rombach et al. 2021]
Classifier-Free Guidance	DALL-E 2 (OpenAI)	[Ramesh et al. 2022]
Cascaded Diffusion & Strong Text Encoders	Imagen (Google Research)	[Saharia et al. 2022]
Diffusion Transformer Architecture	Sora (OpenAI)	OpenAI (2024). *Video generation models as world simulators*.[2]
Fast ODE/SDE Solvers (DDIM, DPM-Solver)	Stable Diffusion Ecosystem (Hugging Face Diffusers)	Diffusers: State-of-the-art diffusion models[3]

Diffusion models have demonstrated remarkable versatility, extending far beyond image synthesis into a wide array of scientific and industrial domains. Their strong generative capabilities, ability to model complex distributions, and compatibility with con-

[2]Technical Report: `https://openai.com/research/video-generation-models-as-world-simulators`
[3]Hugging Face Diffusers Documentation: `https://huggingface.co/docs/diffusers/index`

ditioning mechanisms make them a compelling choice for tasks involving structured or multimodal data. Below, we outline key areas where diffusion models are defining the state of the art. A more complete list of applications in a wide range of areas can be found in [Yang et al. 2024]. To illustrate this direct lineage from academic research to practical application, Table 1.1 summarizes several of these key techniques and identifies their implementation in well-known generative models. Each entry is accompanied by a reference to the official technical report or foundational paper, providing a verifiable link between the concepts and their real-world impact.

1.7.1. Diffusion models in database applications

Diffusion models have become essential tools in database applications, providing controllable synthetic data generation, robust imputation, and hybrid vector-relational capabilities. [Liu et al. 2024] showed that SQL-like predicates can steer the denoising process for workload replay or privacy-preserving data sharing, while FinDiff [Sattarov et al. 2023] demonstrated superior utility-privacy trade-offs compared to GAN baselines in regulated finance applications.

For data repair and augmentation, [Villaizán-Vallelado et al. 2025] combined transformer denoisers with dynamic masking to outperform VAE and GAN methods on supporting both data imputation and synthetic data generation while maintaining privacy. [Li et al. 2025] identified four key research themes: data augmentation, data imputation, trustworthy synthesis, and anomaly detection, while highlighting ongoing challenges in mixed-type encoding, fairness, and scalability.

On the systems side, ACORN [Patel et al. 2024] introduced an in-database index that unifies vector search with relational queries, enabling efficient hybrid workloads over vector data (embedded images, text, and video) as well as structured data, such as attributes and keywords. Regarding privacy concerns, [Wu et al. 2025] addressed privacy vulnerabilities in diffusion models used for tabular data synthesis. Their method demonstrated superior capability in detecting privacy leaks in diffusion-based tabular data synthesis compared to traditional heuristic evaluation methods.

[Shi et al. 2025] addresses the critical need for synthetic tabular data generation in machine learning, where real datasets often suffer from scarcity, privacy concerns, and class imbalance issues. The authors organize methods into three main categories (traditional generation, diffusion models, and LLM-based approaches), provide a complete pipeline overview from generation through evaluation, and identify key challenges and real-world applications.

1.7.2. Synthetic data for tabular domains

In enterprise settings, generating high-fidelity synthetic data is critical for data augmentation, privacy preservation, and simulation. Diffusion models have been adapted to handle heterogeneous tabular data. The contribution of methods like TabDDPM [Kotelnikov et al. 2023] was to design a diffusion process specifically for tabular formats containing a mix of continuous and categorical features. This is achieved by using specialized forward noising schemes for each data type and replacing the convolutional U-Net with a standard Multi-Layer Perceptron (MLP) architecture suited for non-spatial data, thereby enabling realistic synthetic table generation.

1.7.3. Text-to-image and text-to-video generation

Perhaps the most culturally significant application of diffusion models is in conditional generation from natural language. The key contribution of models like Stable Diffusion [Rombach et al. 2021] was to make high-resolution synthesis computationally tractable by performing the diffusion process in a compressed latent space learned by a powerful autoencoder. Concurrently, models like Imagen [Saharia et al. 2022] demonstrated another critical insight: the power of leveraging large, pre-trained language models as text encoders, combined with a cascaded diffusion architecture that progressively increases image resolution, leading to unprecedented photorealism and semantic fidelity.

This paradigm has been naturally extended to *text-to-video generation*. The primary challenge here is maintaining temporal consistency across frames. Seminal works like Videofusion [Luo et al. 2023] and Video Diffusion Models [Ho et al. 2022] contributed solutions by extending the image generation architecture with spatiotemporal attention mechanisms. Their key innovation was to enable the model to attend not only to spatial information within a frame but also to temporal information across frames, ensuring that object identity and motion remain coherent over time.

1.7.4. Bioinformatics and molecular design

In the life sciences, diffusion models are driving breakthroughs in problems involving complex 3D structures. DiffDock [Corso et al. 2023] reframed molecular docking as a generative modeling task, predicting how a small molecule binds to a protein. It diffuses over ligand poses, including translational, rotational, and torsional degrees of freedom, to efficiently sample realistic binding conformations. In *de novo* protein design, RFdiffusion [Watson et al. 2023] made a landmark contribution by learning to generate novel, functional protein backbones. It applies a diffusion process directly to the 3D coordinates and orientations of amino acid residues, enabling the design of complex protein structures conditioned on functional motifs or structural constraints.

1.7.5. Text and language modeling

While Transformers dominate language modeling, diffusion models offer a compelling alternative paradigm. The key contribution of Diffusion-LM [Li et al. 2022] was to successfully adapt the continuous denoising process to the discrete domain of text. It achieves this by first embedding discrete word tokens into a continuous space, performing the diffusion process on these embeddings, and then rounding the denoised embeddings back to the nearest token in the vocabulary. This non-autoregressive approach offers unique benefits, such as iterative refinement and strong performance on controlled text generation and infilling tasks, where the model must generate text consistent with a given prefix and suffix.

1.7.6. Graph generation

Generative modeling for graphs is essential in areas like social network analysis, drug discovery, and circuit design. The main challenge is jointly modeling a graph's discrete structure (nodes and edges) and its continuous attributes. DiGress [Vignac et al. 2023] addresses this by proposing a unified framework that diffuses both the adjacency matrix and node/edge features. This joint denoising process captures the interplay between structure and attributes, enabling the generation of realistic and diverse graphs from a simple noise prior.

1.8. Conclusion

In this work, we presented a comprehensive and accessible overview of diffusion models, tracing their development from foundational probabilistic principles to their current role as a leading paradigm in generative artificial intelligence. We began by exploring the theoretical origins of Denoising Diffusion Probabilistic Models (DDPMs) and score-based generative models, showing how both perspectives converge on the goal of learning to reverse a noise process. This unification underpins the core mechanism of diffusion models: training a neural network to denoise samples corrupted through a carefully designed diffusion process.

The practical success of diffusion models is largely attributed to architectural innovations. Central to this is the U-Net backbone, augmented with time-step embeddings and cross-attention modules, enabling both temporal coherence and external conditioning. Alongside this, methodological advances such as Classifier-Free Guidance have improved controllability, while solvers like DDIM and DPM-Solver++ dramatically reduce the sampling time. Latent Diffusion Models (LDMs) further extended the applicability of these techniques to high-resolution outputs with manageable computational cost.

These developments have spurred a broad spectrum of real-world applications. Diffusion models now power systems in domains ranging from image and video generation to protein design, molecular docking, and tabular data synthesis. Their flexibility in modeling complex, multimodal, and structured data highlights their versatility and transformative potential. As the field moves forward, open challenges remain in efficiency, compositionality, and alignment, ensuring that diffusion models will continue to be an active area of research and innovation.

References

[Aali et al. 2023] Aali, A., Arvinte, M., Kumar, S., and Tamir, J. I. (2023). Solving inverse problems with score-based generative priors learned from noisy data. In *2023 57th Asilomar Conference on Signals, Systems, and Computers*, pages 837–843.

[Bishop and Bishop 2024] Bishop, C. M. and Bishop, H. (2024). *Deep Learning - Foundations and Concepts*. Springer.

[Chung et al. 2023] Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. (2023). Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*.

[Chung et al. 2025] Chung, H., Kim, J., Park, G. Y., Nam, H., and Ye, J. C. (2025). CFG++: Manifold-constrained classifier free guidance for diffusion models. In *The Thirteenth International Conference on Learning Representations*.

[Chung et al. 2022] Chung, H., Sim, B., Ryu, D., and Ye, J. C. (2022). Improving diffusion models for inverse problems using manifold constraints. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

[Corso et al. 2023] Corso, G., Stärk, H., Jing, B., Barzilay, R., and Jaakkola, T. S. (2023). Diffdock: Diffusion steps, twists, and turns for molecular docking. In *The Eleventh International Conference on Learning Representations*.

[Cox and Hinkley 1974] Cox, D. and Hinkley, D. (1974). *Theoretical Statistics*. Chapman and Hall/CRC, New York, 1st edition.

[Dhariwal and Nichol 2021a] Dhariwal, P. and Nichol, A. (2021a). Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794.

[Dhariwal and Nichol 2021b] Dhariwal, P. and Nichol, A. Q. (2021b). Diffusion models beat GANs on image synthesis. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*.

[Dinh et al. 2015] Dinh, L., Krueger, D., and Bengio, Y. (2015). NICE: non-linear independent components estimation. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*.

[Dosovitskiy et al. 2021] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

[Fefferman et al. 2013] Fefferman, C., Mitter, S., and Narayanan, H. (2013). Testing the manifold hypothesis.

[Gong et al. 2022] Gong, S., Li, M., Feng, J., Wu, Z., and Kong, L. (2022). Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*.

[Goodfellow et al. 2014] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA. MIT Press.

[Ho et al. 2020] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

[Ho and Salimans 2021] Ho, J. and Salimans, T. (2021). Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

[Ho et al. 2022] Ho, J., Salimans, T., Gritsenko, A. A., Chan, W., Norouzi, M., and Fleet, D. J. (2022). Video diffusion models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

[Hyvärinen 2005] Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *J. Mach. Learn. Res.*, 6:695–709.

[Jordan et al. 1998] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1998). *An Introduction to Variational Methods for Graphical Models*, pages 105–161. Springer Netherlands, Dordrecht.

[Kingma and Gao 2023] Kingma, D. P. and Gao, R. (2023). Understanding diffusion objectives as the ELBO with simple data augmentation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[Kingma et al. 2021] Kingma, D. P., Salimans, T., Poole, B., and Ho, J. (2021). On density estimation with diffusion models. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*.

[Kingma and Welling 2022] Kingma, D. P. and Welling, M. (2022). Auto-encoding variational bayes.

[Kotelnikov et al. 2023] Kotelnikov, A., Baranchuk, D., Rubachev, I., and Babenko, A. (2023). TabDDPM: Modelling tabular data with diffusion models.

[Li et al. 2022] Li, X. L., Thickstun, J., Gulrajani, I., Liang, P., and Hashimoto, T. (2022). Diffusion-LM improves controllable text generation. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

[Li et al. 2025] Li, Z., Huang, Q., Yang, L., Shi, J., Yang, Z., van Stein, N., Bäck, T., and van Leeuwen, M. (2025). Diffusion models for tabular data: Challenges, current progress, and future directions.

[Liu et al. 2024] Liu, T., Fan, J., Tang, N., Li, G., and Du, X. (2024). Controllable tabular data synthesis using diffusion models. *Proc. ACM Manag. Data*, 2(1).

[Liu et al. 2023] Liu, X., Gong, C., and qiang liu (2023). Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*.

[Lu et al. 2022] Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. (2022). DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

[Lu et al. 2023] Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. (2023). DPM-solver++: Fast solver for guided sampling of diffusion probabilistic models.

[Luo et al. 2023] Luo, Z., Chen, D., Zhang, Y., Huang, Y., Wang, L., Shen, Y., Zhao, D., Zhou, J., and Tan, T. (2023). Videofusion: Decomposed diffusion models for high-quality video generation. *arXiv preprint arXiv:2303.08320*.

[Murphy 2022] Murphy, K. P. (2022). *Probabilistic Machine Learning: An introduction*. MIT Press.

[Nichol and Dhariwal 2021] Nichol, A. Q. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models.

[Patel et al. 2024] Patel, L., Kraft, P., Guestrin, C., and Zaharia, M. (2024). Acorn: Performant and predicate-agnostic search over vector embeddings and structured data. *Proc. ACM Manag. Data*, 2(3).

[Peebles and Xie 2023] Peebles, W. and Xie, S. (2023). Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4172–4182.

[Prince 2023] Prince, S. J. (2023). *Understanding Deep Learning*. The MIT Press.

[Ramesh et al. 2022] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents.

[Rombach et al. 2021] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2021). High-resolution image synthesis with latent diffusion models.

[Ronneberger et al. 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

[Saharia et al. 2022] Saharia, C., Chan, W., Saxena, S., Lit, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Gontijo-Lopes, R., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. (2022). Photorealistic text-to-image diffusion models with deep language understanding. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

[Sattarov et al. 2023] Sattarov, T., Schreyer, M., and Borth, D. (2023). Findiff: Diffusion models for financial tabular data generation. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, ICAIF '23, page 64–72, New York, NY, USA. Association for Computing Machinery.

[Shi et al. 2025] Shi, R., Wang, Y., Du, M., Shen, X., Chang, Y., and Wang, X. (2025). A comprehensive survey of synthetic tabular data generation.

[Sohl-Dickstein et al. 2015] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr.

[Song et al. 2021a] Song, J., Meng, C., and Ermon, S. (2021a). Denoising diffusion implicit models. In *International Conference on Learning Representations*.

[Song and Ermon 2019] Song, Y. and Ermon, S. (2019). *Generative modeling by estimating gradients of the data distribution*. Curran Associates Inc., Red Hook, NY, USA.

[Song et al. 2019] Song, Y., Garg, S., Shi, J., and Ermon, S. (2019). Sliced score matching: A scalable approach to density and score estimation. *CoRR*, abs/1905.07088.

[Song et al. 2021b] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021b). Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

[Tang et al. 2025] Tang, Z., Bao, J., Chen, D., and Guo, B. (2025). Diffusion models without classifier-free guidance.

[Vaswani et al. 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

[Vignac et al. 2023] Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. (2023). Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*.

[Villaizán-Vallelado et al. 2025] Villaizán-Vallelado, M., Salvatori, M., Segura, C., and Arapakis, I. (2025). Diffusion models for tabular data imputation and synthetic data generation. *ACM Trans. Knowl. Discov. Data*, 19(6).

[Vincent 2011] Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural Comput.*, 23(7):1661–1674.

[Wainwright and Jordan 2008] Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1–2):1–305.

[Watson et al. 2023] Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., Wicky, B. I. M., Hanikel, N., Pellock, S. J., Courbet, A., Sheffler, W., Wang, J., Venkatesh, P., Sappington, I., Torres, S. V., Lauko, A., De Bortoli, V., Mathieu, E., Ovchinnikov, S., Barzilay, R., Jaakkola, T. S., DiMaio, F., Baek, M., and Baker, D. (2023). De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100.

[Wu et al. 2025] Wu, X., Pang, Y., Liu, T., and Wu, S. (2025). Winning the midst challenge: New membership inference attacks on diffusion models for tabular data synthesis.

[Yang et al. 2024] Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., and Yang, M.-H. (2024). Diffusion models: A comprehensive survey of methods and applications.