Diffusion Models

Handout Notes - Generative and Deep Learning (GDL)

Davide Bacciu - University of Pisa

Notation. Observed random vectors are denoted by $\mathbf{X}$ and values by $\mathbf{x} \in \mathbb{R}^D$. Noisy latent variables at diffusion step t are denoted by $\mathbf{Z}_t$ and values by $\mathbf{z}_t$, with $\mathbf{z}_0 = \mathbf{x}$. The number of diffusion steps is T. The forward-process variance schedule is $\beta_t \in (0, 1)$, and we define $\alpha_t = 1 - \beta_t$ and

$$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s.$$

The forward noising distribution is written $q(\mathbf{z}_t \mid \mathbf{z}_{t-1})$, while the learned reverse model is $p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t)$. Gaussian noise variables are written $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I)$. A dataset is $\mathcal{D} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ with size $N = |\mathcal{D}|$.

1 Why diffusion models?

Diffusion models have become central in generative deep learning because they combine strong sample quality with a conceptually clean training principle: learn to generate by *iterative denoising*. Instead of mapping a latent vector directly to a sample in one step, as in GANs, or learning an explicit invertible transformation, as in normalizing flows, diffusion models define generation as the reversal of a gradual corruption process.

This idea is attractive for two reasons. First, the forward corruption process can be made extremely simple: repeatedly add a small amount of Gaussian noise. Second, each reverse step only needs to undo a small amount of corruption. So generation is decomposed into many easy local denoising problems rather than one difficult global synthesis problem.

At a high level, diffusion models posit two stochastic processes:

- a fixed **forward diffusion** process that gradually destroys structure in data by adding noise;
- a learned **reverse denoising** process that reconstructs structure step by step, starting from pure noise.

The resulting model is generative because, after learning the reverse process, one can sample

$$\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, I)$$

and iteratively denoise until reaching a sample in data space.

2 Generation as a denoising process

The central intuition is easiest to state informally. Suppose we start from a data sample $\mathbf{x}$ and repeatedly corrupt it:

$$\mathbf{x} = \mathbf{z}_0 \to \mathbf{z}_1 \to \mathbf{z}_2 \to \cdots \to \mathbf{z}_T,$$

where each $\mathbf{z}_t$ is slightly noisier than the previous one. If the number of steps is large and each noise increment is small, then by time T the variable $\mathbf{z}_T$ is approximately Gaussian noise.

Now reverse the story. If one knew how to remove just the right amount of noise at each step, then one could start from Gaussian noise and reconstruct a data sample:

$$\mathbf{z}_T \to \mathbf{z}_{T-1} \to \cdots \to \mathbf{z}_1 \to \mathbf{z}_0 = \mathbf{x}.$$

This is the generative idea behind diffusion models:

learn the reverse of a simple noising process.

The challenge is not defining the forward process. That part is fixed. The challenge is learning an accurate and efficient approximation to the reverse denoising process.

3 The forward diffusion process

The forward process is a Markov chain that gradually adds Gaussian noise. A standard choice is

$$q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) = \mathcal{N}\Big(\sqrt{1-\beta_t}\,\mathbf{z}_{t-1},\ \beta_t I\Big), \qquad t = 1, \dots, T,$$

with $\mathbf{z}_0 = \mathbf{x}$. Equivalently, one can sample

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

Here β_t is the *noise schedule*. It controls how much new noise is injected at step t. Typically β_t is small, so each step only slightly corrupts the previous state.

The full forward chain factorizes as

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

Worked example — Why the forward process remains Gaussian

Assume

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

Then the conditional sampling equation is

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

Because a linear transformation of a Gaussian plus independent Gaussian noise is again Gaussian, each forward conditional is Gaussian. Moreover, repeated application preserves Gaussianity, which is why many later quantities in diffusion models admit closed forms.

4 Closed-form diffusion kernel

Sequentially simulating the forward chain is conceptually simple, but for training we want a direct formula for sampling $\mathbf{z}_t$ from $\mathbf{x}$ without explicitly constructing all intermediate states.

Define

$$\alpha_t = 1 - \beta_t, \qquad \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s.$$

Then one can show that

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

Equivalently,

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

This closed form is crucial in practice. It means that, during training, one can pick any timestep t and generate the corresponding noisy version $\mathbf{z}_t$ directly from the clean sample $\mathbf{x}$.

Worked example — Deriving the closed-form noising equation

Start from

$$\mathbf{z}_1 = \sqrt{\alpha_1}\mathbf{x} + \sqrt{1-\alpha_1}\,\boldsymbol{\epsilon}_1.$$

Then

$$\mathbf{z}_2 = \sqrt{\alpha_2}\mathbf{z}_1 + \sqrt{1-\alpha_2}\,\boldsymbol{\epsilon}_2 = \sqrt{\alpha_2\alpha_1}\mathbf{x} + \sqrt{\alpha_2(1-\alpha_1)}\,\boldsymbol{\epsilon}_1 + \sqrt{1-\alpha_2}\,\boldsymbol{\epsilon}_2.$$

Because the last two terms are independent Gaussians, they can be combined into a single Gaussian term with the appropriate variance. Continuing inductively yields

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

So the entire forward diffusion process admits a direct one-shot sampling formula.

5 Evolution of the diffused distribution

If $\mathbf{x} \sim q(\mathbf{x})$ denotes the data distribution, then the distribution of the noisy variable at time t is

$$q(\mathbf{z}_t) = \int q(\mathbf{x})\, q(\mathbf{z}_t \mid \mathbf{x})\, d\mathbf{x}.$$

As t increases, $q(\mathbf{z}_t)$ becomes progressively smoother and less structured. For sufficiently large T and suitable schedule β_t, the final distribution $q(\mathbf{z}_T)$ is approximately standard Gaussian.

This is what makes generation possible: if the terminal distribution is simple and known, we can sample from it directly and let the learned reverse process map it back toward data.

6 Why the reverse process is hard

The ideal reverse transition would be

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

If this conditional were known, we could generate samples exactly by starting from $\mathbf{z}_T$ and iterating backward. However, this reverse conditional is intractable in general because it implicitly depends on the unknown data distribution.

The core obstacle is that noise removal is ambiguous. Given a noisy sample $\mathbf{z}_t$, many clean signals might have produced it. Without knowing the data distribution, one cannot uniquely determine how to denoise.

A crucial observation, however, is that if the original clean sample $\mathbf{x}$ were known, then the reverse conditional

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x})$$

is Gaussian and tractable. This makes it possible to derive a variational training objective.

7 The learned reverse process

Diffusion models therefore introduce a parameterized reverse Markov chain:

$$p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t) = \mathcal{N}\big(\boldsymbol{\mu}_\theta(\mathbf{z}_t, t), \sigma_t^2 I\big), \qquad t = 1, \ldots, T,$$

with terminal prior

$$p(\mathbf{z}_T) = \mathcal{N}(\mathbf{0}, I).$$

The mean $\boldsymbol{\mu}_\theta(\mathbf{z}_t, t)$ is predicted by a neural network, typically conditioned both on the noisy input $\mathbf{z}_t$ and on a representation of the timestep t.

Generation then proceeds by ancestral sampling:

1. sample
$$\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, I),$$
2. for $t = T, T-1, \ldots, 1$, sample
$$\mathbf{z}_{t-1} \sim p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t).$$

The final output $\mathbf{z}_0$ is the generated sample.

8 Likelihood training and the ELBO

The diffusion model defines a latent-variable generative model over the chain

$$\mathbf{z}_{1:T}, \mathbf{x}.$$

Formally,
$$p_\theta(\mathbf{x}, \mathbf{z}_{1:T}) = p(\mathbf{z}_T) \prod_{t=1}^{T} p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t), \qquad \mathbf{z}_0 = \mathbf{x}.$$

The marginal likelihood is
$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}_{1:T})\, d\mathbf{z}_{1:T},$$
which is intractable.

As in variational autoencoders, one therefore introduces the forward process as an inference distribution:
$$q(\mathbf{z}_{1:T} \mid \mathbf{x}).$$

Applying Jensen's inequality gives an evidence lower bound (ELBO):
$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z}_{1:T})}{q(\mathbf{z}_{1:T} \mid \mathbf{x})} \right].$$

After algebraic simplification, the ELBO decomposes into:

- a reconstruction term at the final denoising step,
- a sum of KL divergences aligning the learned reverse transitions with the true reverse conditionals.

A convenient schematic form is
$$\mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})} \big[\log p_\theta(\mathbf{x} \mid \mathbf{z}_1) \big] - \sum_{t=2}^{T} \mathrm{KL}\big(q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x}) \parallel p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t) \big).$$

Worked example — Why the ELBO compares reverse Gaussians

Because the forward process is Gaussian, the conditional
$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{x})$$
is Gaussian as well. The learned reverse model is also chosen Gaussian:
$$p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t) = \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{z}_t, t), \sigma_t^2 I).$$
Therefore each KL term in the ELBO is a KL divergence between Gaussians. This is computationally convenient and leads, after simplification, to a quadratic loss on the difference between the true reverse mean and the predicted reverse mean.

9 The practical training objective: predict the noise

Although the ELBO is the principled objective, it can be simplified dramatically. The key trick is to reparameterize the noisy sample as

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

and then train the model to predict the noise $\boldsymbol{\epsilon}$ itself.

Let the neural network be written

$$\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t).$$

The simplified loss is

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\mathbb{E}_t\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},I)}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t)\|_2^2\right].$$

This objective says: given a clean sample, a random timestep, and the corresponding noised sample, predict the exact Gaussian noise that was added.

Worked example — Why noise prediction is equivalent to denoising

From

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

we can solve for the clean sample:

$$\mathbf{x} = \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{z}_t - \frac{\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}\boldsymbol{\epsilon}.$$

So if the network can predict $\boldsymbol{\epsilon}$ accurately from $(\mathbf{z}_t, t)$, then it can also estimate the clean data point $\mathbf{x}$ and hence the mean of the reverse denoising distribution. Thus learning to predict the added noise is a reparameterized form of learning to denoise.

This is the form used in most practical denoising diffusion probabilistic models.

10 A practical training algorithm

The training loop is simple once the closed-form noising formula is available.

For each mini-batch:

1. sample clean data $\mathbf{x}$ from the training set;
2. sample a timestep t uniformly from $\{1, \ldots, T\}$;
3. sample noise
$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I);$$
4. construct the noisy input
$$\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\mathbf{x} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon};$$
5. feed $(\mathbf{z}_t, t)$ into the neural network and minimize
$$\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t)\|_2^2.$$

Thus training does not simulate the entire forward chain explicitly. A single random timestep per example is enough.

11 Network architecture for image diffusion

For images, the standard architecture is a U-Net with residual blocks and attention layers. The noisy image $\mathbf{z}_t$ is processed by an encoder–decoder convolutional network with skip connections. A learned encoding of the timestep t is injected into the residual blocks so that the network knows how much noise is present.

This architectural choice is natural:

- the U-Net captures local and multiscale image structure,
- residual blocks stabilize deep computation,
- self-attention improves modeling of long-range spatial dependencies,
- timestep embeddings allow the denoiser to adapt across noise levels.

So diffusion models combine a probabilistically motivated objective with an image architecture already known to be effective for restoration and dense prediction.

12 Noise schedules

The schedule β_t controls the entire difficulty profile of diffusion. If noise is added too aggressively, the reverse task becomes too hard. If it is added too slowly, generation requires too many steps.

A common basic choice is a linear schedule:

$$\beta_1 < \beta_2 < \cdots < \beta_T,$$

with small positive values. The reverse variance is often tied to the forward process through a simple choice such as

$$\sigma_t^2 = \beta_t,$$

although more refined variants learn or modify these quantities.

Intuitively, early timesteps should preserve most semantic structure while corrupting fine details, and later timesteps should wash the sample into near-isotropic noise.

13 Sampling from the reverse process

Once trained, generation begins from pure Gaussian noise:

$$\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, I).$$

The model then iteratively predicts the reverse mean and samples

$$\mathbf{z}_{t-1} \sim p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t), \qquad t = T, \ldots, 1.$$

At each step, the network estimates the noise present in $\mathbf{z}_t$, and this estimate is used to partially subtract noise and move the sample toward data space.

The overall process is computationally more expensive than one-shot generation, because many reverse steps are needed. This high computational cost is one of the main trade-offs of diffusion models.

14 Conditional generation and guidance

Diffusion models can be conditioned on auxiliary information such as class labels, text prompts, or partial images. The idea is to modify the reverse process so that generated samples are not only realistic, but also compatible with a condition c.

There are two main guidance strategies discussed in modern diffusion modeling.

14.1 Classifier guidance

One can train:

- an unconditional diffusion model,
- a separate classifier $p(c \mid \mathbf{z}_t)$ that receives noisy inputs.

During sampling, the reverse step is adjusted in the direction that increases classifier confidence for the desired condition. Conceptually, the classifier gradient nudges the sample toward the target class.

This works, but it has drawbacks:

- the classifier must operate on noisy intermediate states,
- classifier gradients may become unstable or uninformative,
- a separate model is required.

14.2 Classifier-free guidance

A cleaner alternative is classifier-free guidance. Instead of using a separate classifier, one trains a single conditional diffusion model with occasional dropout of the conditioning signal. The same model therefore learns both:

- unconditional denoising,
- conditional denoising.

At sampling time, the two predictions are combined:

$$\text{guided score} = (1-\gamma)\,\text{unconditional score} + \gamma\,\text{conditional score},$$

where γ is a guidance scale.

This usually yields better practical behavior and is now the standard mechanism in text-to-image diffusion systems.

Worked example — Why classifier-free guidance trades diversity for fidelity

Suppose the conditional model predicts a denoising direction that strongly favors samples consistent with condition c, while the unconditional model captures the broader data distribution. Combining them with guidance scale $\gamma > 1$ amplifies condition-consistent directions. As γ increases, generated samples usually become more aligned with the prompt or label, but they also become less diverse because sampling is pushed toward fewer highly compatible modes. Thus guidance improves controllability at the cost of some variability.

15 Conditional and multimodal generation

Once conditioning is available, diffusion becomes a general-purpose framework for structured generation. Examples include:

- class-conditional image generation,
- text-to-image synthesis,
- image completion and panorama completion,
- multimodal generation combining text, image, or other inputs.

The conditioning signal can be incorporated in different ways:

- scalar or vector conditions via embeddings added to the network,
- image conditions via channel-wise concatenation,
- text conditions via embeddings, cross-attention, or both.

This flexibility is one of the major reasons diffusion models have become the dominant generative framework in image synthesis.

16 Latent diffusion

A major efficiency improvement is to perform diffusion not in pixel space, but in a learned latent space. Instead of diffusing the full image $\mathbf{x}$ directly, one first encodes it into a lower-dimensional latent representation:

$$\mathbf{h} = E(\mathbf{x}),$$

runs diffusion in latent space, and then decodes back to the image domain:

$$\tilde{\mathbf{x}} = D(\mathbf{h}).$$

This has two major benefits:

- diffusion operates on smaller tensors, reducing computational cost;
- the latent space can focus on semantically meaningful structure rather than raw pixel redundancy.

This idea underlies latent diffusion models and the family of large-scale systems built on them.

Worked example — Why latent diffusion is cheaper

Suppose an image has spatial size $H \times W \times C$. If an autoencoding stage compresses it to latent size

$$h \times w \times c, \qquad h \ll H, \quad w \ll W,$$

then each denoising step processes far fewer activations. Since diffusion requires many repeated neural evaluations, even a moderate compression factor yields a large total computational saving. This is why latent diffusion makes high-resolution generation much more practical than pixel-space diffusion.

17 Strengths and limitations of diffusion models

Diffusion models have several notable strengths:

- excellent sample quality, especially for images,
- stable training relative to adversarial methods,
- strong compatibility with conditional and multimodal generation,
- flexible probabilistic interpretation through denoising and score estimation.

They also have important limitations:

- generation is computationally expensive because many denoising steps are required,
- vanilla diffusion acts in a latent space of the same size as the data, which can be costly,
- guidance often improves fidelity but can reduce diversity.

Latent diffusion and accelerated samplers address some of these issues, but the basic trade-off remains: diffusion models are powerful because generation is decomposed into many easy denoising steps, yet that same decomposition makes them slower than one-shot generators.