Deep Learning

Class 6: Autoencoders and Diffusion Models

Hédi Hadiji

Université Paris-Saclay - CentraleSupelec

hedi.hadiji@l2s.centralesupelec.fr

November 2025

Table of Contents

Before today we

learned to train models for supervised tasks:

- Feedforward Neural Networks
- Convolutional Neural Networks, Tranformers, etc.
- Training techniques

Supervised learning: learn to apply transformation to data with a goal in mind.
Today: Unsupervised learning: learn useful representations of data without labels. Compression, generation, etc.
Material:

- [Reference paper on VAEs]
- [This blog post]

Table of Contents

Dimensionality Reduction and Encoding

Given a dataset of high-dimensional data points (images, audio, etc.) $Z_1, \ldots, Z_n \in \mathbb{R}^d$.

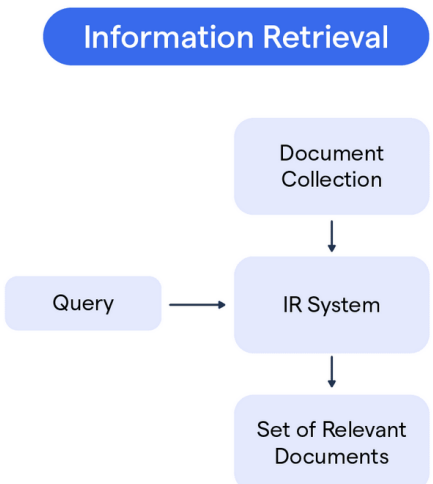
We want to find ways to represent these data points in a lower-dimensional space $\mathbb{R}^k$ with $k \ll d$ while preserving as much information as possible.

Possible applications:

- Data compression
- Visualization
- Noise reduction
- Feature extraction for downstream tasksj

Example application: Information Retrieval

Example pipeline:
Large set of text documents. Train an autoencoder to get low-dimensional representations.



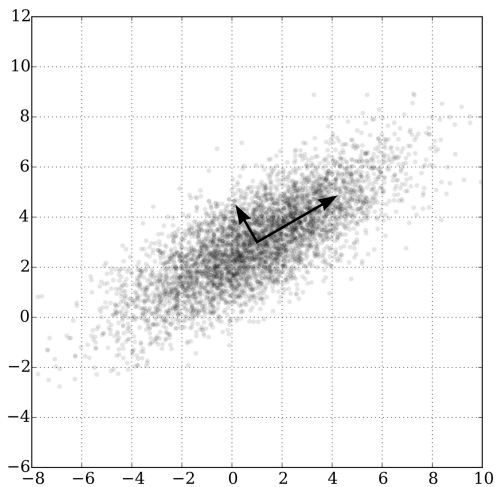
Goal: Given a new document, want to find related documents in the dataset.

Approach: Encode new document to low-dimensional space, then find nearest neighbors among encoded dataset.

Solution 0: PCA

Principal Component Analysis (PCA) is a linear dimensionality reduction technique. PCA finds orthogonal directions (principal components) that capture the most variance in the data.

Project data onto the top k principal components to get low-dimensional representations.



Limitations:

- Only captures linear relationships
- May not perform well on complex, high-dimensional data like images

Solution 1: Task-specific Encoding

Train a model for a specific task (e.g., image classification).

Grab intermediate features from a hidden layer as low-dimensional representation.

Limitations: Representations may not capture all relevant information for other tasks.

(e.g. BERT, GPT, any trained model)

Solution 2: Contrastive Learning

Example: CLIP
Learns representations by contrasting similar and dissimilar pairs of data points.

How do we define similar/dissimilar pairs?

Uses data augmentations to create positive pairs (similar) and treats other samples as negative pairs (dissimilar).

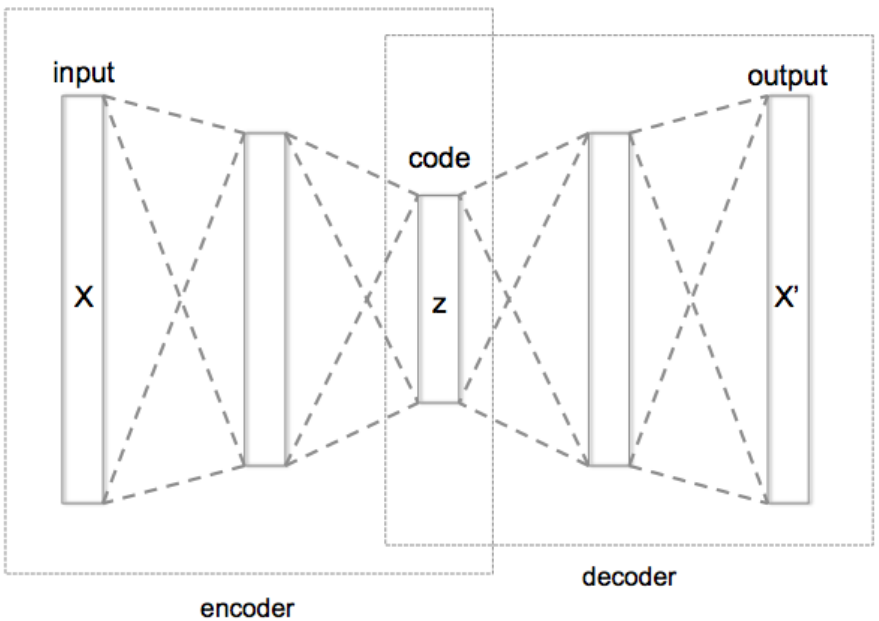
Limitations:

- Requires careful design of augmentations, or at least positive pairs
- May not capture all aspects of the data distribution

Solution 3: Autoencoders

Learn to reconstruct input data from a lower-dimensional representation.
Consists of two parts:

- Encoder: Maps input data to low-dimensional latent space
- Decoder: Reconstructs input data from latent representation



Autoencoder Training

Two networks: Encoder $E : \mathbb{R}^d \to \mathbb{R}^k$ and Decoder $D : \mathbb{R}^k \to \mathbb{R}^d$.

Then $\widehat{x} = D(E(x))$ is the reconstructed input.
Train the autoencoder to minimize reconstruction loss:

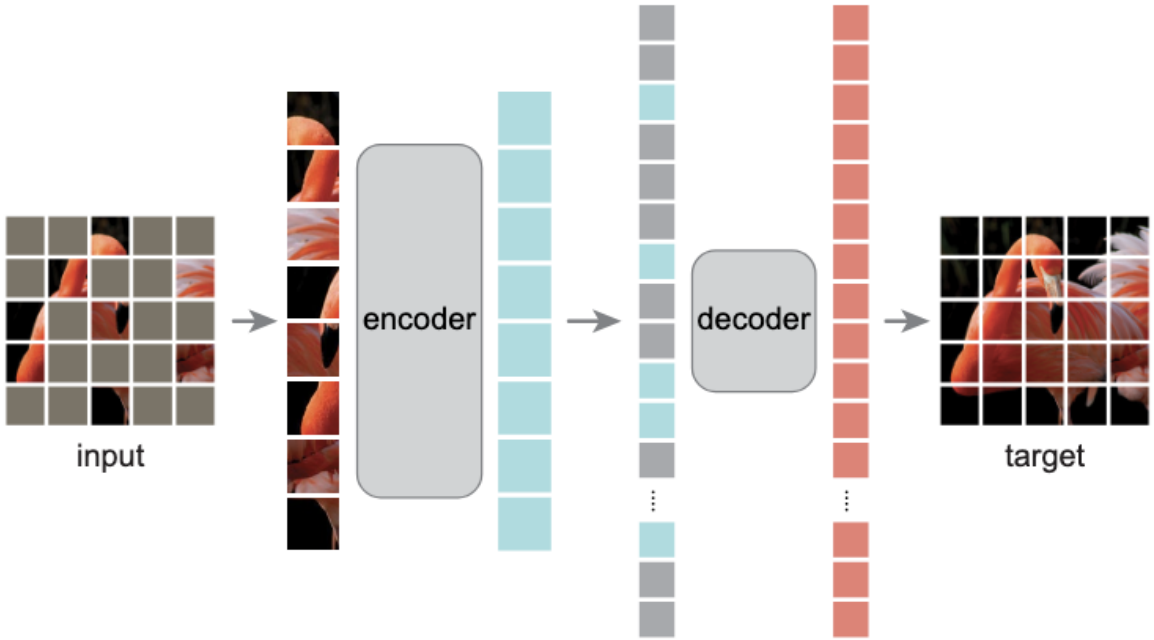
$$L(x, \widehat{x}) = \|x - \widehat{x}\|^2$$

where x is the input data and $\widehat{x}$ is the reconstructed output.

The encoder learns to compress data into a lower-dimensional representation, while the decoder learns to reconstruct the original data from this representation.

After training, use the encoder to obtain low-dimensional representations for new data points.

Focus: Masked AutoEncoder



- Mask images by removing a large portion of patches (e.g., 75%).
- Encoder: processes only visible patches. Output is latent representation of visible patches.
- Decoder: reconstructs the original image from encoded visible patches and mask tokens. Output is full image.
- Loss: Mean squared error between reconstructed and original pixels, computed only on masked patches.

[the paper]

Table of Contents

Generative Models and Density Estimation

Classification: $(X_i, Y_i) \sim P^{\otimes n}(x, y)$, want to predict future Y from X.

Generative modelling: $X_i \sim P^{\otimes n}(X)$, want to learn the distribution $P_{data}(X)$. In statistical terms, this is called density estimation.

In particular, generative modelling focuses on trying to generate new samples from the data distribution. (e.g. generate new images that look like real images)

Why generative models?

- Creative applications (art, music, etc.) (???)
- AGI (????)
- Scientific knowledge: simulate molecules, physics, etc.

Using Autoencoders for Generation

Points in the the embedding space correspond to data points.

But training only to reconstruct input data does not guarantee that the latent space has a nice structure for generation. Arbitrary points in embedding space may not correspond to interesting points for the decoder.

Idea: add some randomness in the training process to encourage smoothness and structure in the latent space.

Possibly in a principled way, using Bayesian statistics.

Idea: model the probability distribution of the embedding, from now on, we call it a latent variable Z.

Latent Variable Generative Model

We define a generative process. We assume a prior distribution over latent variables Z which we will update using Bayes' rule after observing data X.

$$Z \sim p(Z) \quad \textit{and} \quad X|Z \sim p_\theta(X|Z).$$

The marginal distribution of data is:

$$p_\theta(x) = \int p_\theta(x|Z)p(Z)\, dZ.$$

We want to maximize over θ but this integral is typically intractable.

Bayesian viewpoint:

- Z is a random variable (latent cause),
- $p_\theta(X|Z)$ defines the likelihood,
- $p(Z)$ is a prior of our choice over latent variables.

We want to update our belief about Z given the observed data X. The true posterior over latent variables is given by Bayes' rule:

$$p_\theta(Z|X) = \frac{p_\theta(X|Z)p(Z)}{p_\theta(X)}\,.$$

Evidence Lower Bound (ELBO)

General theorem from convex analysis / statistical physics: for any $g = g(z)$,

$$\log \int g(z)p(z)dz = \sup_{q \text{ prob}} \int \log\left(g(z)\right) q(z)dz - \mathrm{KL}(q \,||\, p)\,.$$

where KL is the Kullback-Leibler divergence.
We apply this to

$$g(z) = p_\theta(x|z)$$

so the argument depends on x and θ. Since it's a probability distribution that depends on x, θ and we want to consider them all, we write the argument of the sup as $q(x|z,\theta)$

$$\log \int p_\theta(x|z)p(z)dz = \sup_{q(z|x,\theta)} \int \log\left(p_\theta(x|z)\right) q(z|x,\theta)dz - \mathrm{KL}\left(q(z|x,\theta) \,||\, p(z)\right)\,.$$

Rewriting the maximum likelihood objective:

Maximum Likelihood via ELBO

$$\sup_\theta \sup_{\text{probs } q(z|x,\theta)} \int \log\left(p_\theta(x|z)\right) q(z|x,\theta)dz - \mathrm{KL}\left(q(z|x,\theta) \,||\, p(z)\right)\,.$$

Still, the second supremum is over an infinite dimensional space.

Variational Inference

Idea: parameterize a subset of all conditional distributions and optimize over this subset. Most natural family of distributions: Gaussians, with weights and variances parameterized by the weights of a neural net.

$$\sup_{\text{probs } q(z|x,\theta)} \int \log\left(p_\theta(x|z)\right) q(z|x,\theta) dz - \mathrm{KL}\left(q(z|x,\theta) \,||\, p(z)\right)$$

$$\approx \sup_{q_\phi(z|x)=\mathcal{N}(\mu_\phi(x),\Sigma_\phi(x))} \int \log\left(p_\theta(x|z)\right) q_\phi(z|x) dz - \mathrm{KL}\left(q_\phi(z|x) \,||\, p(z)\right).$$

The first supremum is attained at the true posterior distribution $p_\theta(z|x)$.

This gives a procedure for learning our distribution. Have two networks:

- an encoder that models $p_\theta(x|z)$
- a decoder that models $q_\phi(z|\theta)$

VAE

ELBO objective:

$$\sup_{\theta} \sup_{q_\phi(z|x)=\mathcal{N}(\mu_\phi(x),\Sigma_\phi(x))} \int \log\left(p_\theta(x|z)\right) q_\phi(z|x) dz - \mathrm{KL}\left(q_\phi(z|x) \,||\, p(z)\right).$$

Still uncomputable given θ, ϕ. But we can sample, $Z \sim q_\phi(\cdot \mid X)$ and replace

$$\int \log\left(p_\theta(x|z)\right) q_\phi(z|x) dz \approx \log p_\theta(x|Z)$$

One final trick: **Reparameterization** so that the objective is differentiable:

$$Z \sim \mathcal{N}\left(\mu_\phi(x), \Sigma_\phi(x)\right) \quad \text{iff} \quad Z = \mu_\phi(x) + \Sigma_\phi^{1/2}(x)\varepsilon \quad \text{with} \quad \varepsilon \sim \mathcal{N}(0, I_d)$$

VAE: Recap

- Encoder network, weights ϕ. Inputs: x. Outputs $\mu_\phi(x)$, $\Sigma(x)$, parameters of the posterior distribution $q_\phi(Z|X)$.
- Decoder network, weights θ. Inputs: Z. Outputs $p_\theta(X|Z)$.

Variational Autoencoder

VAR objective:

$$\mathcal{L}(X, Z, \theta, \phi) = -\log p_\theta(X|Z) + \mathrm{KL}\left(q_\phi(z|X) \,||\, p(z)\right).$$

Training:

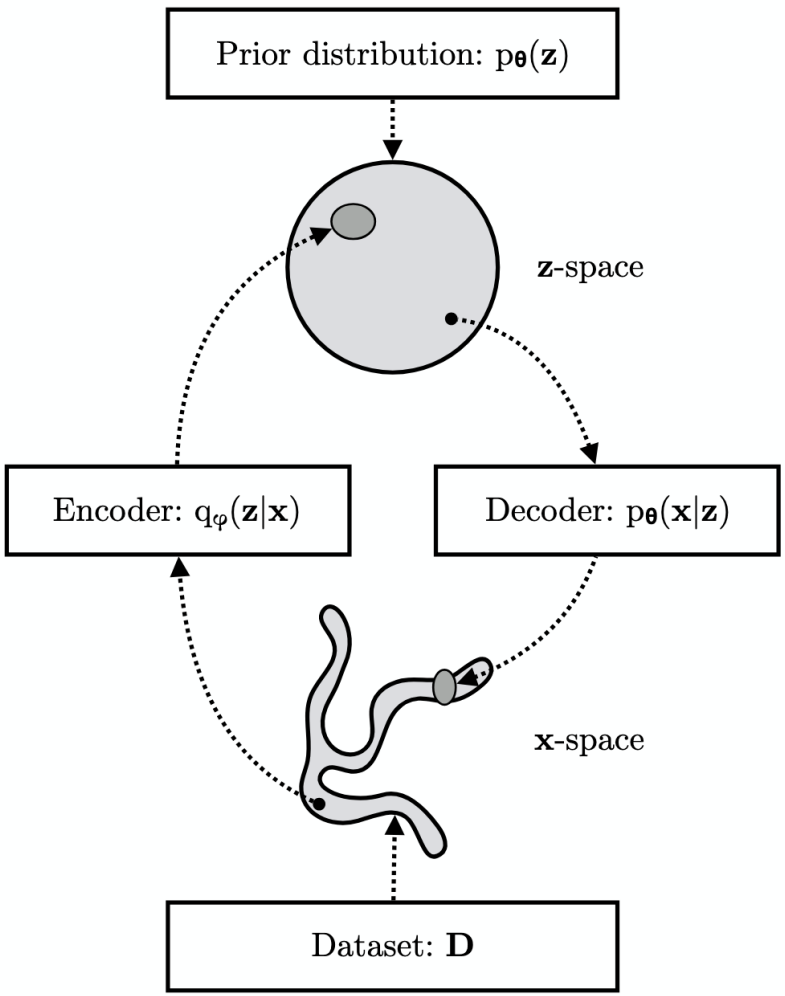
- Take data X.
- Sample Z from the posterior distribution $q_\phi(\cdot|X)$
- Compute VAR objective and do a gradient step.

Note: with gaussian prior and diagonal covariance $\Sigma_\phi^{1/2}(x) = \mathrm{Diag}(\sigma_{\phi,i}(x))$ we have:

$$\mathrm{KL}\left(\mathcal{N}(\mu, \mathrm{Diag}(\sigma^2_{\phi,i}(x)) \,||\, \mathcal{N}(0, I)\right) = \frac{1}{2}\left(\|\mu\|^2 + \sum_{i=1}^{k} \sigma_i^2 - k - \sum_{i=1}^{k} \log \sigma_i^2\right)$$

VAE

[The survey paper by the inventors]



Sampling and Generation

After training:

- The support of the prior $p(Z) = \mathcal{N}(0, I)$ defines a latent space.
- Sampling $Z \sim p(Z)$ and decoding yields synthetic $X \sim p_\theta(X|Z)$.

$$Z \sim \mathcal{N}(0, I), \quad \widehat{X} = g_\theta(Z).$$

Compared to a deterministic auto-encoder, the latent space is better populated: any region of the latent space can be decoded meaningfully.

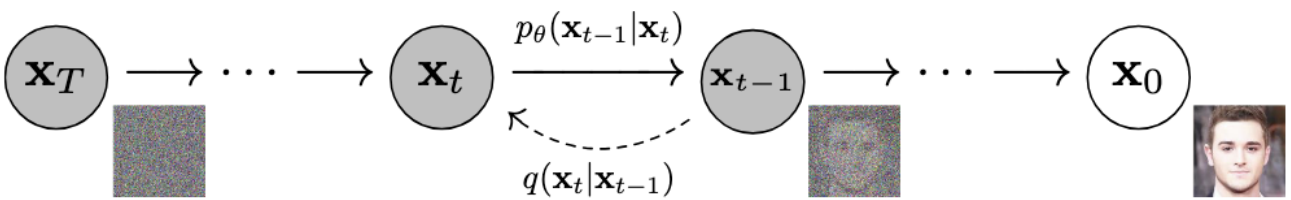
Table of Contents

From VAEs to Diffusion Models

Common goal: learn a generative model of data $p_\theta(X)$.

VAE: assumes latent variable Z with explicit likelihood $p_\theta(X|Z)$.
Diffusion models: define a stochastic process that gradually destroys and then learn to reconstruct the data.

$$X_0 \text{ (data)} \xrightarrow[\text{forward}]{q} X_1, X_2, \ldots, X_T \xrightarrow[\text{reverse}]{p_\theta} \tilde{X}_0.$$



[Idea from 2015] [Breakthrough in 2020]
Now in all major image generation models rely on this.

Forward (Diffusion) Process

We define a Markov chain, **the diffusion process** that progressively adds Gaussian noise to the original image:

$$X_0 \text{ data}, \qquad q(X_t|X_{t-1}) = \mathcal{N}\big(\sqrt{1-\beta_t}\, X_{t-1}, \beta_t I\big),$$

with small noise levels $0 < \beta_t < 1$.

This defines a family of random variables

$$X_0 \sim p_{\text{data}}, \qquad X_t = \sqrt{\bar{\alpha}_t}\, X_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon,$$

where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ and $\varepsilon \sim \mathcal{N}(0, I)$.

As $t \to T$, $X_T \approx \mathcal{N}(0, I)$.

Reverse (Generative) Process

To generate new samples, we need to *reverse the diffusion*:

$$p_\theta(X_{t-1}|X_t) \approx q(X_{t-1}|X_t, X_0).$$

Each reverse step is modeled as a Gaussian:

$$p_\theta(X_{t-1}|X_t) = \mathcal{N}\big(\mu_\theta(X_t, t),\ \Sigma_\theta(X_t, t)\big),$$

where μ_θ is predicted by a neural network.

The joint generative process is:

$$p_\theta(X_{0:T}) = p(X_T) \prod_{t=1}^{T} p_\theta(X_{t-1}|X_t), \qquad p(X_T) = \mathcal{N}(0, I).$$

Learning objective: Variational Lower Bound in Diffusion Models

We want to estimate the log-likelihood of data X_0:

$$\log p_\theta(X_0) = \log \int p_\theta(X_{0:T})\, dX_{1:T} = \log \int \frac{p_\theta(X_{0:T})}{q(X_{1:T} \mid X_0)}\, q(X_{1:T} \mid X_0)\, dX_{1:T}.$$

Applying Jensen's inequality gives the **Evidence Lower Bound (ELBO):**

$$\log p_\theta(X_0) \geq \mathbb{E}_{q(X_{1:T}|X_0)} \left[\log \frac{p_\theta(X_{0:T})}{q(X_{1:T} \mid X_0)} \right].$$

Hence the diffusion ELBO can be written as

$$\mathcal{L}_{\text{diffusion}}(X_0) = \mathbb{E}_{q(X_{1:T}|X_0)} \Big[\log p_\theta(X_{0:T}) - \log q(X_{1:T} \mid X_0) \Big].$$

Expanding the chain factors:

$$\mathcal{L}_{\text{diffusion}}(X_0) = \mathbb{E}_q \Big[\log p(X_T) + \sum_{t=1}^{T} \log p_\theta(X_{t-1} \mid X_t) - \sum_{t=1}^{T} \log q(X_t \mid X_{t-1}) \Big].$$

Equivalent KL Form of the Diffusion ELBO

The previous expression can be rearranged into a sum of Kullback–Leibler divergences between the forward and reverse transitions.

Starting from

$$\mathcal{L}_{\text{diffusion}}(X_0) = \mathbb{E}_q\left[\log \frac{p_\theta(X_{0:T})}{q(X_{1:T} \mid X_0)}\right],$$

expand and regroup terms using the Markov structure of p_θ and q:

$$\mathcal{L}_{\text{diffusion}}(X_0) = \mathbb{E}_q[\log p(X_T)] + \sum_{t=1}^{T} \mathbb{E}_q\left[\log \frac{p_\theta(X_{t-1} \mid X_t)}{q(X_t \mid X_{t-1})}\right].$$

After conditioning on X_t and X_0, each term can be expressed as a KL divergence:

$$\mathbb{E}_q\left[\log \frac{p_\theta(X_{t-1} \mid X_t)}{q(X_t \mid X_{t-1})}\right] = -\,\mathbb{E}_q\left[\mathrm{KL}\big(q(X_{t-1} \mid X_t, X_0) \,\|\, p_\theta(X_{t-1} \mid X_t)\big)\right].$$

Hence the **KL form** of the diffusion ELBO:

$$\log p_\theta(X_0) \geq -\,\mathrm{KL}\big(q(X_T \mid X_0) \,\|\, p(X_T)\big) - \sum_{t>1} \mathbb{E}_q\left[\mathrm{KL}\big(q(X_{t-1} \mid X_t, X_0) \,\|\, p_\theta(X_{t-1} \mid X_t)\big)\right].$$

(Intuition:) Each term encourages the learned reverse step $p_\theta(X_{t-1} \mid X_t)$ to match the true posterior $q(X_{t-1} \mid X_t, X_0)$ of the known noising process.

From ELBO to the ε-prediction loss

Forward (noising) process. Fix $\{\beta_t\}_{t=1}^T \subset (0,1)$, let

$$q(x_t \mid x_{t-1}) = \mathcal{N}\big(\sqrt{1-\beta_t}\, x_{t-1},\ \beta_t I\big), \qquad \alpha_t := 1-\beta_t, \quad \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s.$$

Then the closed form marginal is

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

Posterior of one step of the forward chain. For $t \geq 2$,

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\big(\mu_t(x_t, x_0),\ \sigma_t^2 I\big),$$

with

$$\mu_t(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\, x_0, \qquad \sigma_t^2 = \frac{\beta_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}.$$

Reverse (generative) model. Parameterize

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

i.e. we *match* the posterior variance and learn only the mean.

Exact diffusion ELBO (VLB).

$$\mathcal{L}_{\mathrm{VLB}}(x_0) = \underbrace{\mathbb{E}_q\big[-\log p_\theta(x_0 \mid x_1)\big]}_{\text{“reconstruction” at } t=1} + \sum_{t=2}^{T} \mathbb{E}_q\Big[\mathrm{KL}\left(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\right)\Big] + \underbrace{\mathrm{KL}\left(q(x_T \mid x_0) \,\|\, p(x_T)\right)}_{\text{const if } p=\mathcal{N}(0,I)}.$$

From ELBO to the ε-prediction loss 2

With matched variances, each KL becomes a quadratic in the mean difference:

$$\mathrm{KL}\left(\mathcal{N}(\mu_t, \sigma_t^2 I) \,\|\, \mathcal{N}(\mu_\theta, \sigma_t^2 I)\right) = \frac{1}{2\sigma_t^2} \left\|\mu_t(x_t, x_0) - \mu_\theta(x_t, t)\right\|^2 + \text{const}.$$

Use the reparametrization $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon$ and the standard mean parameterization

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

A short calculation shows

$$\mu_t(x_t, x_0) - \mu_\theta(x_t, t) = \frac{\beta_t}{\sqrt{\alpha_t}\,\sqrt{1-\bar{\alpha}_t}} \left(\varepsilon - \varepsilon_\theta(x_t, t)\right).$$

Hence, for $t \geq 2$,

$$\mathrm{KL}\left(q(\cdot \mid x_t, x_0) \,\|\, p_\theta(\cdot \mid x_t)\right) = \underbrace{\frac{\beta_t^2}{2\,\sigma_t^2\,\alpha_t(1-\bar{\alpha}_t)}}_{=:w_t} \left\|\varepsilon - \varepsilon_\theta(x_t, t)\right\|^2 + \text{const}.$$

The ε-prediction objective. Up to additive constants and the $t = 1$ term,

$$\mathcal{L}_{\text{simple}}(\theta) = \mathbb{E}_{x_0 \sim p_{\text{data}},\, t \sim \mathrm{Unif}\{2,\dots,T\},\, \varepsilon \sim \mathcal{N}(0,I)} \left[w_t \left\| \varepsilon - \varepsilon_\theta\left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon,\ t\right) \right\|^2 \right].$$

In practice one often sets $w_t \equiv 1$ (or uses a simple schedule), yielding the standard DDPM loss.

Learning Objective

Simplifying choice: we fix the variance of each reverse step to match the true posterior.

By reparameterization, we train the model to predict directly the (mean of) noise ε added at each step

$$\varepsilon_\theta(X_t, t)$$

Out network takes as input a noisy image X_t and a timestep t and returns a prediction of the noise ε.

Then the simplified training objective is:

Diffusion Model Loss

$$\mathcal{L}(\theta) = \mathbb{E}_{t,x_0,\varepsilon}\left[\|\varepsilon - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}X_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon, t)\|^2\right].$$

Diffusion Model Training and Sampling Algorithm (Practical Form)

Goal: Learn a network $\varepsilon_\theta(x_t, t)$ that predicts the noise added at any timestep t in the forward diffusion process.

Training objective (simple loss):

$$\mathcal{L}_{\text{simple}}(\theta) = \mathbb{E}_{x_0,t,\varepsilon}\left[\|\varepsilon - \varepsilon_\theta(x_t, t)\|^2\right].$$

Forward (training-time) process: Sample $t \sim \text{Uniform}\{1, \ldots, T\}$, noise $\varepsilon \sim \mathcal{N}(0, I)$, and construct

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon,$$

Learn to invert the noising process at every scale of noise.

- To incorporate time step t, use a positional encoding.
- The network sees partially noised images x_t and predicts the noise component ε.

Algorithm Summary: Training

Training phase

1: Sample data $x_0 \sim p_{\text{data}}$
2: Sample timestep $t \sim \text{Uniform}\{1, \ldots, T\}$
3: Sample noise $\varepsilon \sim \mathcal{N}(0, I)$
4: Compute noisy input: $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon$
5: Predict noise: $\widehat{\varepsilon} = \varepsilon_\theta(x_t, t)$
6: Compute loss: $\mathcal{L} = \|\varepsilon - \widehat{\varepsilon}\|^2$
7: Take gradient step on $\nabla_\theta \mathcal{L}$

Algorithm Summary (contd.)

Sampling (reverse process)

Sample $x_T \sim \mathcal{N}(0, I)$
for $t = T, \ldots, 1$ **do**
    Predict noise: $\widehat{\varepsilon} = \varepsilon_\theta(x_t, t)$
    Compute mean: $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \widehat{\varepsilon} \right)$
    if $t > 1$ **then**
        Sample $z \sim \mathcal{N}(0, I)$
        $x_{t-1} = \mu_\theta(x_t, t) + \sigma_t z$
    else
        $x_0 = \mu_\theta(x_1, 1)$
return x_0

Remarks:

- Typical T: 1000-4000 steps (coarser for fast sampling variants).
- ε_θ is often a U-Net conditioned on t.

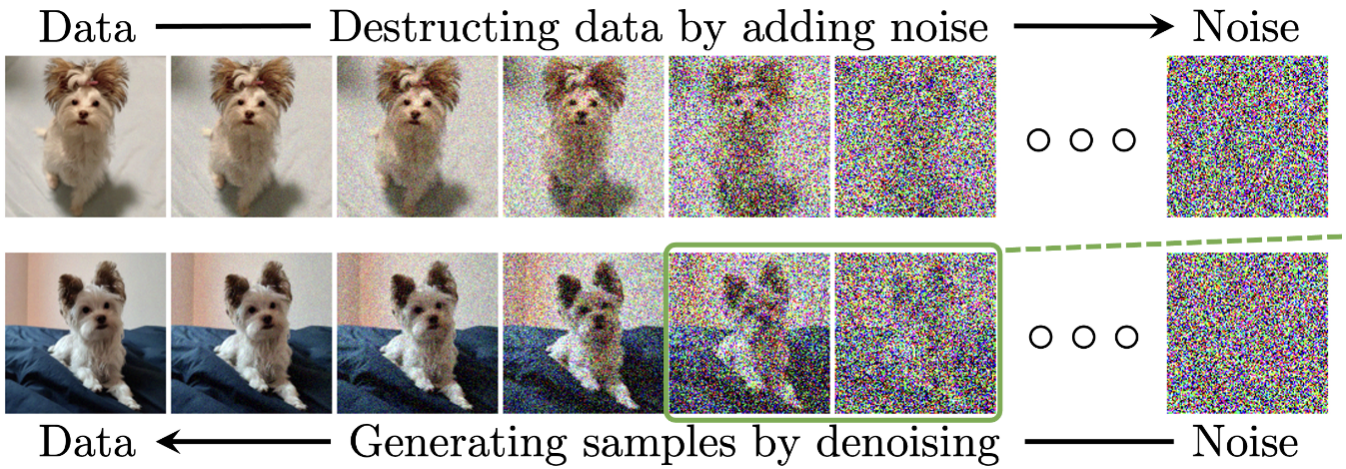
Sampling from a Diffusion Model

Once trained:

$$X_T \sim \mathcal{N}(0, I), \quad X_{t-1} \sim p_\theta(X_{t-1}|X_t), \quad t = T, \ldots, 1.$$

The process produces $X_0 \sim p_\theta(X_0)$.

Need T successive denoising steps, each using the learned network ε_θ.



[Example implementation of diffusion sampling]

Relation to Variational Inference

The diffusion model's training objective is also a variational bound:

$$\log p_\theta(X_0) \geq \mathbb{E}_q \left[\log \frac{p_\theta(X_{0:T})}{q(X_{1:T}|X_0)} \right],$$

analogous to the ELBO in VAEs.

Key difference:

- In VAEs, Z is a single latent variable with learned posterior $q_\phi(Z|X)$.
- In diffusion models, the latent trajectory (X_t) has known forward posteriors $q(X_t|X_0)$ and learned reverse transitions $p_\theta(X_{t-1}|X_t)$.

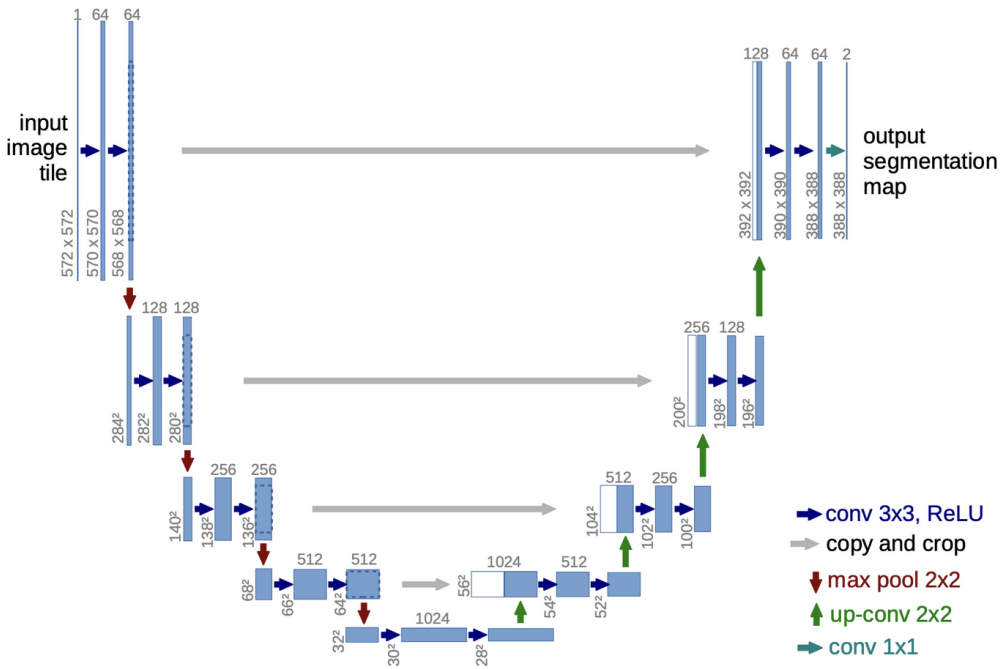
U-Net Architecture

Goal: Map noisy inputs to clean outputs while preserving fine spatial detail.

- **Encoder:** successive convolutions and downsamplings capture global context.
- **Decoder: transposed** convolutions reconstruct the image resolution.
- **Skip connections:** pass high-resolution features from encoder to decoder to recover detail lost during downsampling.

Structure:

$$\text{Input} \rightarrow \text{Encoder (down)} \leftrightarrow \text{Bottleneck} \leftrightarrow \text{Decoder (up)} \rightarrow \text{Output}$$



Why U-Nets for Diffusion Models

Goal in diffusion: learn a denoising function

$$\varepsilon_\theta(x_t, t) \approx \text{noise added at step } t.$$

- Each denoising step requires both **global context** (what object is being drawn) and **local precision** (pixel details).
- The U-Net's symmetric encoder–decoder with skip connections provides exactly this:
  - encoder $\Rightarrow$ captures large-scale structure,
  - decoder $\Rightarrow$ reconstructs fine textures,
  - skip paths $\Rightarrow$ fuse global and local information efficiently.

Better for image generation than standard CNNs because we do not lose the fine-grained details thanks to the skip connections.

Summary

VAEs:

- Introduce latent random variable Z and learn $p_\theta(X|Z)$.
- Optimize a variational lower bound (ELBO).

Diffusion Models:

- Define a Markov chain of noise corruption and denoising.
- Learn reverse transitions $p_\theta(X_{t-1}|X_t)$ to invert diffusion.