An Introduction to Diffusion Models

Theory and Practice

Wei Huang

Research Scientist · RIKEN AIP & The Institute of Statistical Mathematics

University of Tokyo lecture · Spring 2026 | May 13, 2026 | Public version

Lecture Outline

01

Foundations of Generative Models

Why generative modeling, likelihood, model landscape

02

From VAEs to DDPMs

Variational perspective: ELBO, noising, denoising

03

Score-Based Perspective

EBMs, score matching, DSM, NCSN, Score SDE

04

From Theory to Practice

U-Net, guidance, latent diffusion, applications, exercises

Lecture plan: 105 minutes + Q&A | Public version

What is a Deep Generative Model?

DGMs take as input a large collection of real-world examples (e.g., images, text) drawn from an unknown and complex distribution $p_{data}(x)$ and output a trained neural network that parameterizes an approximate distribution $p_\theta(x)$.

Their goals are twofold:

1

Realistic Generation

Generate novel, realistic samples that are indistinguishable from real data — the model must have truly captured the underlying distribution.

2

Controllable Generation

Enable fine-grained and interpretable control over the generative process — steer outputs by conditioning on text, class labels, or other signals.

Why Do We Need Generative Models?

Creative Generation

Synthesise entirely new images, audio, video, or text that looks and feels real — enabling art, entertainment, and design at scale.

DALL-E 3 · Sora · Stable Diffusion · AudioLDM

Scientific Discovery

Sample novel candidate molecules, proteins, or materials from a learned distribution — dramatically accelerating the design space search.

RFDiffusion · FrameDiff · GenCast

Data Understanding

A model that can generate data must have captured the underlying structure of that data — learning is understanding.

Anomaly detection · Compression · Representation learning

"To understand is to be able to generate." — Richard Feynman (paraphrased)

The Core Problem: Learning p(x)

The Setup

We observe a finite dataset of samples $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$

We assume they were drawn i.i.d. from some unknown distribution

We want a model $p_\theta(\mathbf{x})$ that approximates it well enough to generate new, realistic samples

$p_{\text{data}}(\mathbf{x})$ — the true data distribution

Why Is This Hard?

High dimensionality

A 256×256 image lives in a 196,608-dimensional space — most of which is empty.

Complex structure

Real data lies on low-dimensional manifolds with rich multimodal, compositional structure.

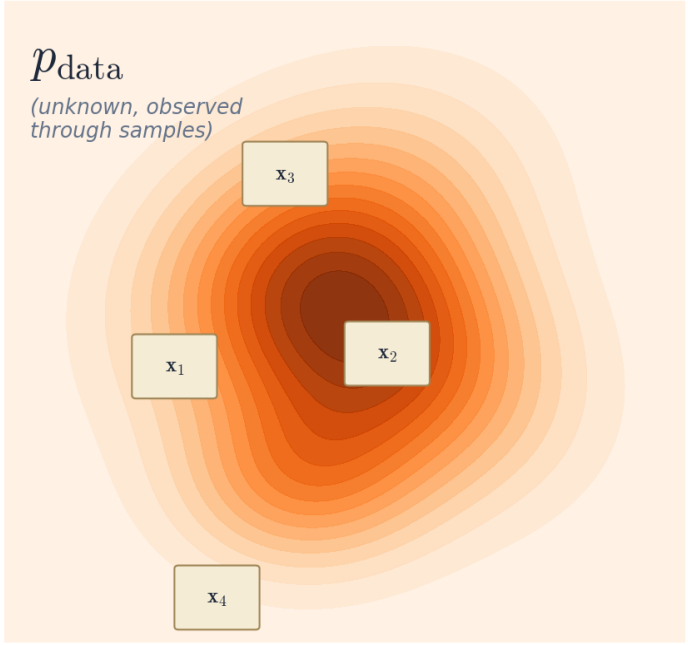
Tractability vs. quality

Computing exact likelihoods is often intractable; approximations trade off coverage and fidelity.

$\mathbf{x}_{\text{new}} \sim p_\theta(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$

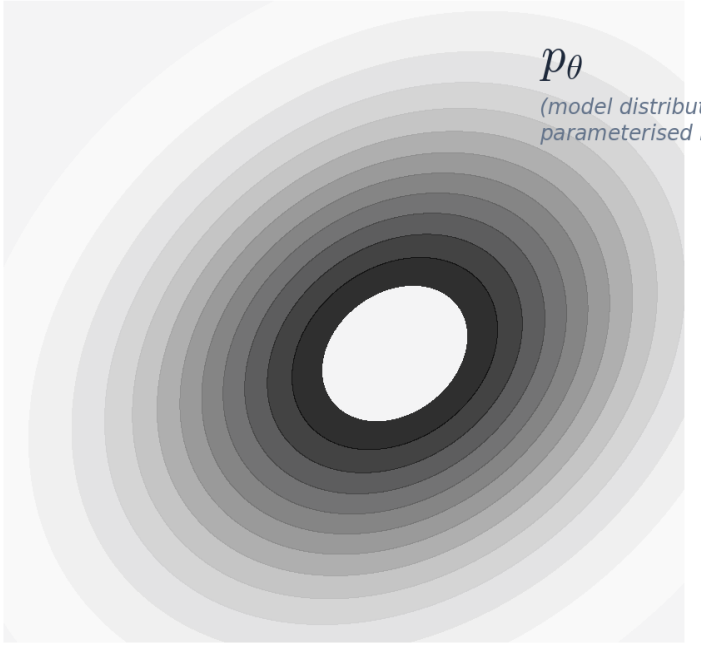
Goal → Learn a model so good we can draw new samples indistinguishable from real data.

What Does a Generative Model Actually Do?



$\mathcal{D}(p_{\text{data}},\ p_\theta)$

minimise discrepancy



Training goal: $\theta^* \in \arg\min_\theta \mathcal{D}(p_{\text{data}},\ p_\theta)$ where $\mathcal{D}$ must be estimable from finite i.i.d. samples $\mathbf{x}_i \sim p_{\text{data}}$

KL Divergence & Maximum Likelihood Estimation

Forward KL divergence — a standard choice for D:

$$\mathcal{D}_{\mathrm{KL}}(p_{\mathrm{data}} \| p_\theta) := \mathbb{E}_{\mathbf{x} \sim p_{\mathrm{data}}}[\log p_{\mathrm{data}}(\mathbf{x}) - \log p_\theta(\mathbf{x})]$$

Decompose — p data term is constant w.r.t. ϑ, so minimising KL = maximising log-likelihood:

$$\mathcal{D}_{\mathrm{KL}}(p_{\mathrm{data}} \| p_\theta) = -\mathbb{E}_{\mathbf{x} \sim p_{\mathrm{data}}}[\log p_\theta(\mathbf{x})] + \mathcal{H}(p_{\mathrm{data}})$$

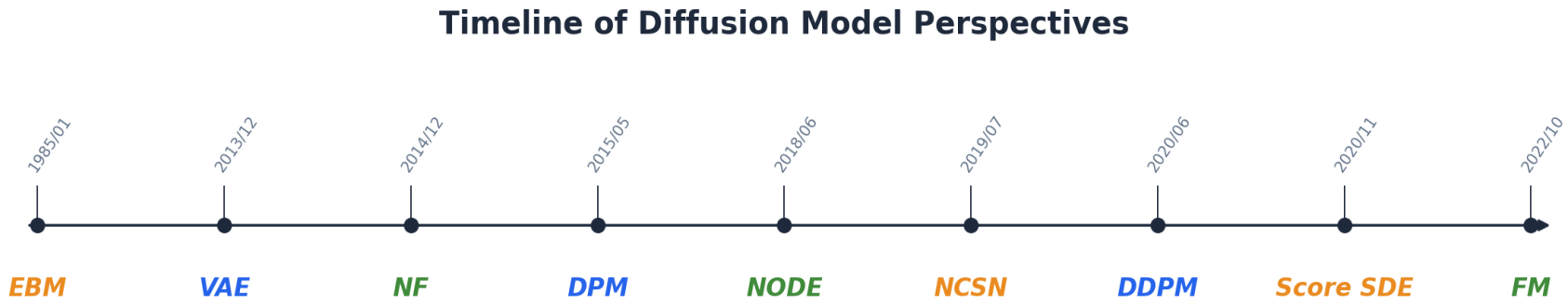
Lemma: Minimising KL ⇔ MLE

$$\min_\theta \mathcal{D}_{\mathrm{KL}}\left(p_{\mathrm{data}} \,\|\, p_\theta\right) \iff \max_\theta \mathbb{E}_{\mathbf{x} \sim p_{\mathrm{data}}}\left[\log p_\theta(\mathbf{x})\right]$$

In practice — replace expectation with Monte Carlo estimate over N i.i.d. samples:

$$\hat{\mathcal{L}}_{\mathrm{MLE}}(\theta) := -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)})$$

A Brief History of Generative Models



VAE perspective: VAE → DPM → DDPM

Score perspective: EBM → NCSN → Score SDE

Flow perspective: NF → NODE → Flow Matching

Sec. 2: VAE/DDPM

Sec. 3: Score-based

Prominent Deep Generative Models (I)

EBM

Energy-Based Models

Define probability through an energy function $E_\theta(\mathrm{x})$ that assigns lower energy to more probable data points. Training maximises log-likelihood, but requires computing the intractable partition function Z(θ).

$$p_\theta(\mathbf{x}) \triangleq (1/Z(\theta))\exp(-E_\theta(\mathbf{x}))$$

→ Diffusion models bypass Z(θ) by learning the gradient of log density instead.

AR

Autoregressive Models

Factorise the joint distribution into a product of conditionals using the chain rule. Each conditional $p_\theta(x_i|x_{<i})$ is parameterised by a neural network. Exact likelihood is tractable since each factor is normalised by design.

$$p(\mathbf{x}) = \prod_i p_\theta(x_i \mid x_1, \ldots, x_{i-1})$$

→ Strong density estimation, but sequential generation limits sampling speed.

VAE

Variational Autoencoders

Introduce latent variables z to capture hidden structure. Encoder $q_\phi(z|x)$ approximates the posterior; decoder $p_\theta(x|z)$ reconstructs data. Training maximises the Evidence Lower Bound (ELBO) on log-likelihood.

$$\mathrm{ELBO} = \mathbb{E}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z}))$$

→ Principled latent-variable framework, but samples may lack sharpness.

Prominent Deep Generative Models (II)

NF

Normalizing Flows

Learn a bijective mapping f_θ between a simple latent distribution z and data x. The change-of-variables formula absorbs normalisation analytically, making likelihood computation exact and tractable. Includes Neural ODEs as a continuous-time variant.

$$\log p_\theta(\mathbf{x}) = \log p(\mathbf{z}) + \log \left| \det \partial f_\theta^{-1}(\mathbf{x}) / \partial \mathbf{x} \right|$$

→ Exact tractable likelihood, but bijectivity constraints can limit expressiveness.

GAN

Generative Adversarial Networks

A generator G creates realistic samples from noise z; a discriminator D tells real from fake. No explicit density — GANs bypass likelihood entirely. The min-max game implicitly minimises the Jensen–Shannon divergence $D_{JS}(p_{data} \| p_G)$.

$$\min_G \max_D \mathbb{E}[\log D(\mathbf{x})] + \mathbb{E}[\log(1 - D(G(\mathbf{z})))]$$

→ High-quality samples, but training is notoriously unstable (mode collapse).

The Generative Model Landscape

	GANs	VAEs	Flows	Diffusion
Training stability	⚠ Hard	✓ Stable	✓ Stable	✓ Stable
Sample quality	✓ High	⚠ Blurry	⚠ Medium	✓✓ **Best**
Likelihood	✗ No	✓ ELBO	✓ Exact	✓ ELBO
Mode coverage	⚠ Drops	✓ Good	✓ Good	✓✓ **Best**
Inference speed	✓ Fast	✓ Fast	✓ Fast	⚠ Slow

→ *Diffusion models lead on sample quality and mode coverage, at a speed cost*

02

From VAEs to DDPMs

Variational Perspective

We view diffusion models through a variational lens. Starting from VAEs — which represent data with latent variables and optimize an evidence lower bound — we show how hierarchical variants stack multiple latent layers. DDPMs follow the same template: a fixed forward noising process replaces the learned encoder, and training learns a decoder that reverses the path in successive denoising steps.

VAEs, hierarchical VAEs, and diffusion models all optimize a likelihood surrogate defined by a variational bound.

From Autoencoders to VAEs

1 The Autoencoder

Deterministic encoder maps x → z; decoder reconstructs x.
Reconstruction only; no structured prior on z.

2 The VAE Solution

Use probabilistic encoder $q_\theta(z \mid x)$, not one code.
Regularize with prior $p(z) = N(0, I)$.
Sample z and decode with $p_\phi(x \mid z)$.

Marginal likelihood: $p_\phi(x) = \int p_\phi(x \mid z)\, p(z)\, dz$

VAE Architecture

$$z \sim \mathcal{N}(0, I) \longrightarrow x \sim p_\phi(x \mid z)$$

Sample a latent code, then decode.

Structured latent space enables meaningful samples

Key Distinction from Autoencoders

Autoencoder: deterministic mapping; unstructured latent space.
VAE: probabilistic mapping; prior + ELBO regularisation.

The Variational Approach

The Problem: Intractable Posterior

Which latent code z produced x?
Bayes requires $p_\phi(x)$, which is intractable:

$$p_\phi(z \mid x) = \frac{p_\phi(x \mid z)p(z)}{p_\phi(x)}$$

→ *intractable for expressive, nonlinear decoders*

Solution: Variational Inference

Use an encoder $q_\theta(z \mid x)$, as a learnable proxy for the true posterior.

Variational approximation $q_\theta(z \mid x) \approx p_\phi(z \mid x)$

The Evidence Lower Bound (ELBO)

Since direct MLE is infeasible, optimize a tractable lower bound on $\log p_\phi(x)$:

$$\mathcal{L}_{\text{ELBO}}(\theta, \phi; x) = \mathbb{E}_{q_\theta(z \mid x)}[\log p_\phi(x \mid z)] - D_{\text{KL}}(q_\theta(z \mid x) \,\|\, p(z))$$

1 Reconstruction term

$\mathbb{E}_{q_\theta(z \mid x)}[\log p_\phi(x \mid z)]$

How well can the decoder recover x from z?

2 Latent regularisation

$D_{\text{KL}}(q_\theta(z \mid x) \,\|\, p(z))$

Keeps $q_\theta(z \mid x)$, close to prior $p(z)=N(0,I)$.

VAE training = balance reconstruction quality and latent structure

Training via the ELBO

Theorem: Evidence Lower Bound

For any data point x, the log-likelihood satisfies:

$$\log p_\phi(x) \;\geq\; \mathcal{L}_{\text{ELBO}}(\theta, \phi; x)$$

$$\mathcal{L}_{\text{ELBO}} = \mathrm{E}_{q_\theta(z \mid x)}[\log p_\phi(x \mid z)] - D_{\text{KL}}(q_\theta(z \mid x) \| p(z))$$

Proof Sketch: Jensen's Inequality

Introduce qθ(z|x) inside the marginal likelihood:

$$\log p_\phi(x) = \log \int q_\theta(z \mid x) \frac{p_\phi(x, z)}{q_\theta(z \mid x)} dz \;\geq\; \mathrm{E}_{q_\theta}\left[\log \frac{p_\phi(x, z)}{q_\theta(z \mid x)}\right]$$

Concavity of log moves the expectation outside the logarithm.

Information-Theoretic View

The ELBO gap decomposes into two error sources. Compare the generative joint pϕ(x,z)=p(z)pϕ(x|z) with the inference joint qθ(x,z)=pdata(x)qθ(z|x):

$$D_{\text{KL}}(q_\theta(x, z) \| p_\phi(x, z)) = D_{\text{KL}}(p_{\text{data}} \| p_\phi) + \mathrm{E}_{p_{\text{data}}} D_{\text{KL}}(q_\theta(z \mid x) \| p_\phi(z \mid x))$$

True modeling error

$$D_{\text{KL}}(p_{\text{data}}(x) \,\|\, p_\phi(x))$$

Inference error

$$\mathbb{E}_{p_{\text{data}}} \mathrm{KL}(q_\theta(z|x) \,\|\, p_\phi(z|x))$$

The gap between log pϕ(x) and the ELBO is exactly:

$$\log p_\phi(x) - \mathcal{L}_{\text{ELBO}} = D_{\text{KL}}(q_\theta(z \mid x) \| p_\phi(z \mid x)) \geq 0$$

→ ***Maximising the ELBO improves both the generator and the encoder.***

★ **Reconstruction**

Accurate recovery of x from z
(like an autoencoder)

★ **Regularisation**

Smooth, structured latent space
(enables generation)

Drawbacks of Standard VAEs

Why Are VAE Outputs Blurry?

A common Gaussian decoder predicts a mean μ(z):

$$p_{\text{dec}}(x \mid z) = \mathcal{N}(x; \mu(z), \sigma^2 I)$$

Optimising the ELBO then becomes a reconstruction-error objective:

Reconstruction = MSE loss

$$\arg\min_{\mu} \mathrm{E}_{p_{\text{data}}, q_{\text{enc}}}[\|x - \mu(z)\|^2]$$

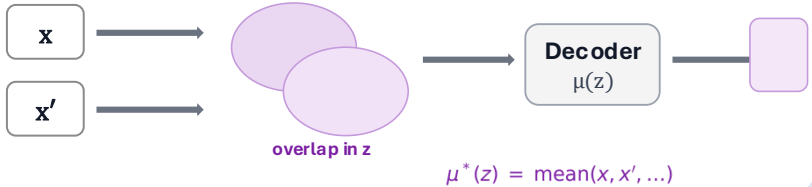
The optimal decoder is the conditional mean:

$$\mu^*(z) = \mathrm{E}_{q_{\text{enc}}(x \mid z)}[x]$$

Conditional means average across plausible modes → non-sharp samples.

The Averaging Problem

If distinct inputs map to overlapping regions in latent space, the decoder cannot choose one mode; μ*(z) averages over multiple inputs.



Implications for Diffusion Models

- Replace single-step decoding with iterative refinement
- Fix the encoder as a forward noising process
- Each denoising step makes small, local corrections

→ avoids averaging over distant modes

1. **Gaussian decoder ⇒ conditional mean**
   MSE loss → blurry outputs
2. **Overlapping latent codes ⇒ averaging**
   conflicting modes collapse
3. **Diffusion ⇒ iterative denoising**
   local corrections → sharper samples

Variational Perspective: DDPM

DDPMs use the same variational template as VAEs, with one crucial change: the encoder is fixed as a known noising process, while the decoder is learned as an iterative denoiser.

1 Forward Pass (Fixed Encoder)

Gradually corrupt data by injecting Gaussian noise via a fixed transition kernel. The data evolves toward an isotropic Gaussian — pure noise.
No parameters to learn: the encoder is predetermined.

$$x_0 \to x_1 \to x_2 \to \cdots \to x_L \sim \mathcal{N}(0, I)$$

2 Reverse Process (Learned Decoder)

A neural network learns $p\phi(x_{i-1}|x_i)$. Starting from pure noise, it iteratively denoises to generate realistic samples.
Each step is a small local correction.

$$x_L \sim \mathcal{N}(0, I) \to x_{L-1} \to \cdots \to x_1 \to x_0$$

A Hierarchical Latent Chain

The DDPM Twist

fixed encoder | iterative decoder | local steps

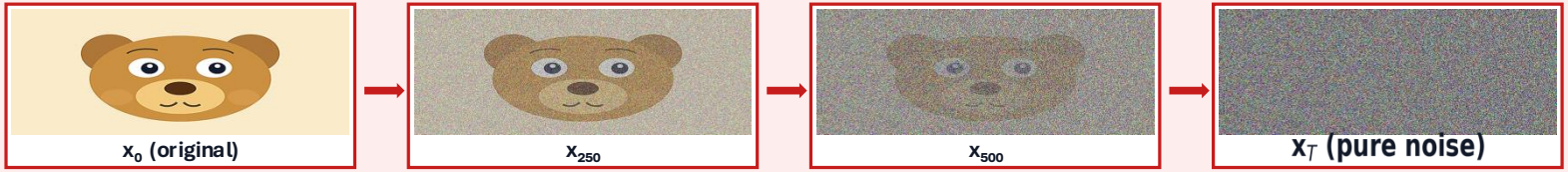
Replace single-step decoding with many small denoising corrections. This avoids asking one decoder pass to average over distant modes.

The Core Idea: Noise as a Creative Tool

① FORWARD PROCESS

Fixed, known process — no learned parameters

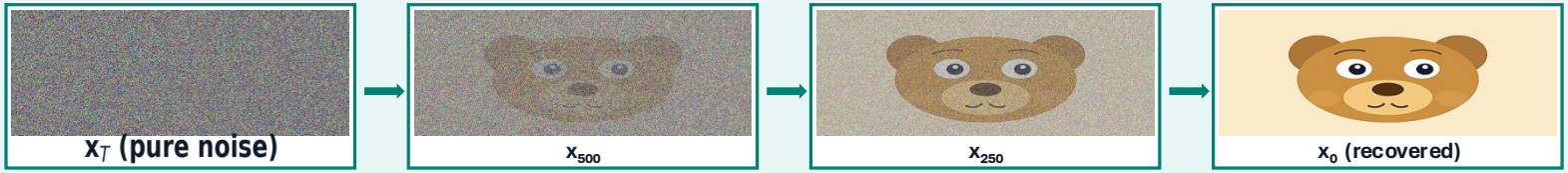
Gradually add Gaussian noise until the data becomes pure random noise:



② REVERSE PROCESS

Learned neural network — this is what we train

Learn to denoise step-by-step, recovering a plausible image from random noise:



Generation = start from noise, then repeatedly subtract the learned noise estimate.

The Forward Process

Markov Chain Formulation

Each step adds a small Gaussian perturbation conditioned only on the previous state.

1 One-Step Gaussian Kernel

For a small β_t, x_t is a slightly noisier version of x_{t-1}:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\right)$$

Markov: $q(x_t|x_{t-1}, x_0) = q(x_t|x_{t-1})$

★ Closed-Form Marginal

No need to simulate T noise steps. Sample x_t directly from x_0:

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

$$\bar{\alpha}_t = \prod_{s=1}^{t} (1-\beta_s)$$

Training samples x_t are cheap to compute: draw one $\varepsilon \sim N(0, I)$, then form x_t in closed form.

Reparameterization & Noise Schedule

Directly sample a noisy state x_t from clean data x_0 — no sequential simulation during training.

Reparameterisation Trick

Sample a timestep t and one Gaussian noise vector ε.
Then construct x_t in closed form:

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

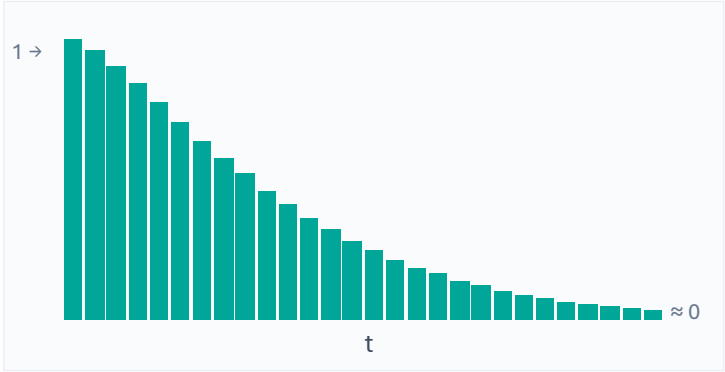
where the cumulative signal kept is

$$\bar{\alpha}_t = \prod_{s=1}^{t} (1 - \beta_s)$$

Training shortcut x_t is cheap to compute, even for T = 1000.

Noise Schedule $\beta_1, \beta_2, \ldots, \beta_T$

Controls how quickly signal is destroyed.



Linear schedule: $\bar{\alpha}_t$ decays from 1 to 0 as t → T.

Cosine schedule gives smoother training in practice.

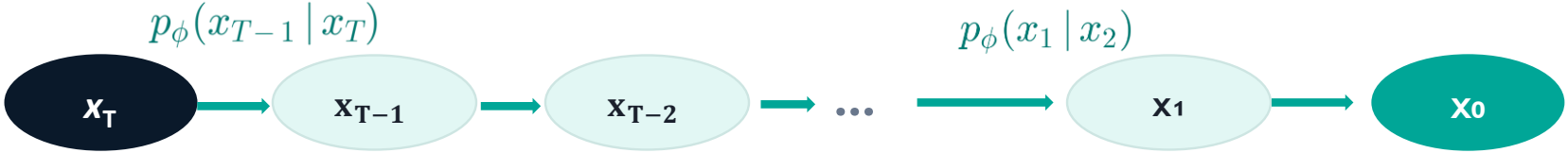
Key takeaway: Noise level t chooses a point on the corruption path; ε tells us exactly what was added.

The Reverse Process

Learn a denoising transition that walks from pure noise back to data.

1 Learned denoising chain

Starting from $x_T \sim N(0, I)$, each step predicts a slightly less noisy sample.



Each reverse step is modelled as a Gaussian:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1};\ \boldsymbol{\mu}_\theta(\mathbf{x}_t, t),\ \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

$\mu_\phi(x_t, t)$

Predicted mean — learned by the neural network.

$\Sigma_\phi(x_t, t)$

Predicted variance — often fixed to β_t I.

DDPM: Modeling & Training Objective

Conditioning on x_0 turns the unreachable reverse target into a closed-form Gaussian target.

1 The Goal

match true reverse

Approximate the true reverse kernel with a learnable model at every noise level.

$$\mathbb{E}\, D_{KL}(p(x_{t-1}|x_t) \| p_\phi(x_{t-1}|x_t))$$

2 The Problem

marginals are unknown

Bayes' rule needs $p_t(x_t)$, which depends on the unknown data distribution.

$$p(x_{t-1} \mid x_t) = \frac{q(x_t \mid x_{t-1})\, p_{t-1}(x_{t-1})}{p_t(x_t)}$$

3 Key Insight

condition on clean x_0

Given x_0, all terms in the reverse kernel are Gaussian and closed-form.

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

From Intractable to Tractable — The DDPM Strategy

Naive objective
match marginal reverse

→

Problem
depends on p_{data}

→

Solution
condition on x_0 → closed forms

Result: the training target is a Gaussian reverse conditional with known mean and variance.

Deriving the DDPM Training Loss

Two facts turn the variational objective into a denoising regression problem.

Theorem: Marginal ⇔ Conditional KL

The intractable marginal KL equals a conditional KL plus a constant:

$$\mathbb{E}_{p_t} D_{KL}(p(x_{t-1}|x_t)\|p_\phi) = \mathbb{E}_{x_0,t} D_{KL}(p(x_{t-1}|x_t, x_0)\|p_\phi) + C$$

→ *Train against the tractable conditioned kernel.*

Lemma: Closed-Form Reverse Kernel

The conditioned reverse transition is Gaussian with schedule-determined parameters:

$$p(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \mu(x_t, x_0, t), \sigma_t^2 I)$$

$$\mu = c_1(t)x_0 + c_2(t)x_t, \quad \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

The DDPM Training Objective

Both distributions are Gaussian, so KL minimisation becomes a weighted MSE between means.

$$L_{DDPM}(\phi) = \sum_t w_t \, \mathbb{E}_{x_0,\varepsilon} \|\mu_\phi(x_t, t) - \mu(x_t, x_0, t)\|^2$$

★ *The network learns a closed-form target at each noise level — no unknown density needed.*

ε-Prediction: From Means to Noise

DDPMs train the network to predict the noise ε rather than the reverse mean directly.

1 **Forward reparameterisation**

Generate a noisy sample from clean data and one Gaussian noise vector:

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

2 **If ε is known, x_0 is recoverable**

The corruption is fully determined by ε:

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

3 **Parameterize μ_θ with ε_θ**

Substitute a learned noise predictor into the reverse mean:

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

4 **Simplified ε-prediction loss**

Mean matching becomes noise matching:

$$L_{simple} = \mathbb{E}_{t,\, x_0,\, \varepsilon} \| \varepsilon - \varepsilon_\theta(x_t, t) \|^2$$

Equivalent views: *mean prediction μ ⇔ noise prediction ε ⇔ clean-data prediction x_0*

“noise detective”

The Training Objective

The DDPM objective collapses to a simple supervised denoising task: predict the noise that was added.

Simplified Training Loss (Ho et al., 2020)

$$\mathcal{L}_{\text{simple}}(\theta) = \mathbb{E}_{t,\, x_0,\, \varepsilon}\left[\|\varepsilon - \varepsilon_\theta(x_t, t)\|_2^2\right]$$

1 Start from ELBO

Maximise $\log p_\theta(x_0)$. The variational lower bound decomposes into KL terms across reverse steps.

2 Simplify to noise prediction

Construct x_t in closed form, then regress the network output back to ε.

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

3 Time conditioning

One denoiser ε_θ handles all noise levels; the timestep t is injected through an embedding.

Minibatch recipe: x_0 → t → ε → x_t → $\varepsilon_\theta(x_t, t)$ → MSE → update θ

DDPM's ELBO & Prediction Equivalences

The variational view and the prediction view are different parameterisations of the same denoising objective.

Theorem: DDPM's ELBO

The negative log-likelihood is upper bounded by three loss terms:

$$-\log p_\phi(x_0) \le L_{\text{prior}} + L_{\text{recon}} + L_{\text{diffusion}}$$

L_prior
prior match
mostly negligible

L_recon
recover x_0
from x_1

L_diffusion
match reverse
conditionals

Data processing: $D_{KL}(p_{data} \| p_\phi) \le D_{KL}(p(x_{0:T}) \| p_\phi(x_{0:T}))$

→ Minimising the ELBO tightens the model–data KL bound.

Three Equivalent Predictions

ε-prediction
predict noise ε
$\|\varepsilon_\theta - \varepsilon\|^2$

x_0-prediction
predict clean data x_0
$\|\hat{x}_{0,\theta} - x_0\|^2$

μ-prediction
predict reverse mean μ
$\|\mu_\theta - \mu\|^2$

Connected by:

$$x_t = \sqrt{\bar{\alpha}_t}\,\hat{x}_{0,\theta}(x_t, t) + \sqrt{1-\bar{\alpha}_t}\,\varepsilon_\theta(x_t, t)$$

The Variational Circle Closes

DDPMs fit the hierarchical VAE template: fixed forward noising chain as encoder, learned reverse chain as decoder. The ELBO decomposes into stable denoising subproblems from coarse to fine.

DDPM: Minimal Training & Sampling Algorithms

Training learns one-step denoising; sampling reuses the same denoiser many times.

TRAINING

Learn to predict the added noise

1. sample x_0 from data
2. sample t and noise ε
3. make noisy x_t from x_0 and ε
4. predict $\varepsilon_\theta(x_t, t)$
5. minimize squared error to ε

same denoiser

SAMPLING

Repeatedly remove predicted noise

1. start from $x_T \sim N(0, I)$
2. for t = T ... 1:
3. predict $\varepsilon_\theta(x_t, t)$
4. compute cleaner x_{t-1}
5. return x_0

Takeaway: training is one-step denoising; generation is the same denoiser applied recursively.

The Speed Problem

High quality comes from many tiny Gaussian reverse steps — but every step costs a full neural network evaluation.

Why So Many Steps?

The reverse transition is modelled as one Gaussian. This approximation is accurate only for small noise increments β_θ, so DDPM uses a long chain.

many small steps

better Gaussian approximation

The Computational Cost

Each step requires one ε_θ evaluation and must be run sequentially because x_{t-1} depends on x_t.

$O(T)$ sequential neural network passes

The Fundamental Trade-off

More steps (large T)

✓ quality ✗ slow

◆

Fewer steps (small T)

✓ fast ✗ rough approximation

Solutions

DDIM · SDE/ODE · distillation

Central challenge: *accelerate sampling without collapsing quality. Viewing sampling as numerical integration of an SDE/ODE opens principled acceleration strategies.*

Faster Sampling: DDIM

DDIM can use a deterministic trajectory (η=0) or a controlled-stochastic variant, enabling skipped timesteps.

Problem: DDPM often uses T = 1000 denoising steps at inference — too slow for practical use.

DDPM (Markov)

- Random noise at every reverse step
- Cannot skip: must run full chain
- High quality, but seconds per sample

1000 steps

DDIM (Non-Markov)

- η=0: deterministic mapping between selected timesteps
- Sub-sample timesteps: 50, 20, or 10 steps
- Similar quality, 10–50× faster inference

50 steps

03

Score-Based Perspective

From EBMs to NCSN

A second, equally fundamental viewpoint: Energy-Based Models represent distributions through an energy landscape. Sampling follows the gradient of this landscape — the score — which points toward higher probability without requiring the intractable normalisation constant.

Score-based diffusion models learn the scores of noise-perturbed distributions, yielding vector fields that guide noisy samples step by step back to data — turning generation into progressive denoising.

Energy-Based Models (EBMs)

Modeling with Energy Functions

EBMs define probability with an energy function: lower energy means higher probability.

EBM density

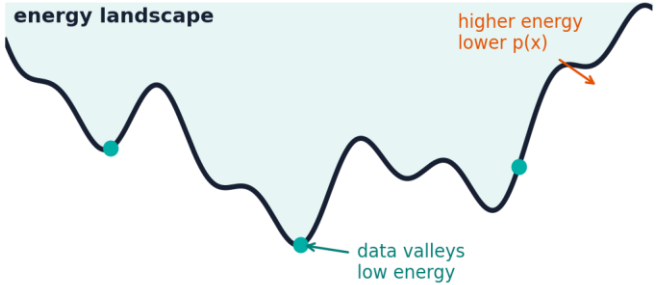
$$p_\phi(x) = \frac{\exp[-E_\phi(x)]}{Z_\phi}$$

Partition function

$$Z_\phi = \int \exp[-E_\phi(x)]\, dx$$

→ Z_ϕ is intractable for expressive networks — the central challenge of EBMs.

Energy Landscape Intuition



Training pushes energy down at real data points and up elsewhere; probabilities are redistributed globally.

Why EBMs Matter for Diffusion Models

EBM challenge

Computing is intractable, so direct sampling and MLE are Z_ϕ difficult.

The score bypasses z_ϕ

$\nabla \log p_\phi(x)$ depends only on the energy gradient, not on Z_ϕ .

$$\nabla_x \log p_\phi(x) = -\nabla_x E_\phi(x)$$

Langevin dynamics

Follow the score plus noise to generate samples without computing Z_ϕ.

The Score Function

MLE Training Is Intractable

The MLE objective for EBMs contains a global normalisation term:

$$\mathcal{L}_{\text{MLE}} = -\mathbb{E}_{p_{\text{data}}}[E_\phi(x)] - \log \int e^{-E_\phi(x)} dx$$

The log Z_φ gradient requires expectations under the model distribution — intractable in high dimensions.

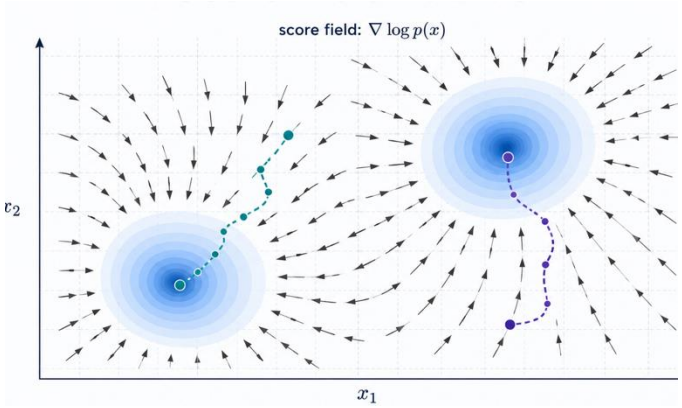
The Score: Gradient of Log-Density

For any density p(x), the score is a vector field in data space:

$$s(x) := \nabla_x \log p(x), \quad s : \mathbb{R}^D \to \mathbb{R}^D$$

For EBMs, Z_ϕ cancels: $\nabla_x \log p_\phi(x) = -\nabla_x E_\phi(x)$.

Score Field Intuition



Direction

toward high-density regions

Sampling

score + noise → samples

The Central Insight

Knowing the score is sufficient for generation: it moves samples toward likely regions without computing Z_ϕ .

Score Matching

1 Match model score to data score

Instead of fitting normalised probabilities, match score vectors directly.

$$\mathcal{L}_{\text{SM}} = \tfrac{1}{2}\mathbb{E}_{p_{\text{data}}}\left[\|\nabla_x \log p_\phi(x) - \nabla_x \log p_{\text{data}}(x)\|^2\right]$$

↓

2 Problem: the data score is unknown

We do not have access to $\nabla x \log p_{data}(x)$ Integration by parts yields an equivalent objective that depends only on the model.

↓

3 Equivalent form — no data score needed

Via integration by parts, the target score disappears.

$$\mathcal{L}_{\text{SM}} = \mathbb{E}_{p_{\text{data}}}[\text{Tr}(\nabla_x^2 E_\phi(x)) + \tfrac{1}{2}\|\nabla_x E_\phi(x)\|^2] + C$$

C is independent of ϕ; the Hessian trace term is the expensive part.

Advantages

✓ eliminates Z_ϕ ✓ no model sampling ✓ uses data + derivatives

Limitation

✗ second-order derivatives ✗ O(D) per sample — prohibitive for images

Score-Based Generative Models

The Key Shift: Learn the Score Directly

Parameterise a neural network $s_\theta(x)$ to approximate the data score field. No energy, no partition function — just a vector field.

$$s_\theta(x) \approx \nabla_x \log p_{\text{data}}(x)$$

Hyvärinen's Tractable Form

Integration by parts gives an equivalent form using only s_θ and data samples:

$$\tilde{\mathcal{L}}_{\text{SM}} = \mathbb{E}_{p_{\text{data}}}\left[\text{Tr}(\nabla_x s_\theta(x)) + \frac{1}{2}\|s_\theta(x)\|^2\right]$$

No access to the true score is needed — only the model's Jacobian.

Intuition of the Two Terms

½ ‖$s_\theta(x)$‖² — norm term
Makes high-density regions stationary.

Tr(∇_x $s_\theta(x)$) — divergence term
Turns stationary points into attractive sinks.

Together: stable, attractive equilibria at data modes.

EBMs → Score Models

learn $E_\phi(x)$ → derive $-\nabla E_\phi$ → learn s_θ directly → sample by score

Denoising Score Matching (DSM)

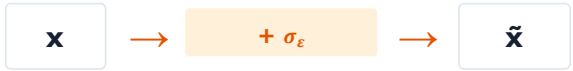
1 Problem: score of clean data is unknown

Score matching needs $\nabla x \log p_{data}(x)$ as a regression target — but the true density is unavailable.

$$\mathcal{L}_{SM} = \tfrac{1}{2}\mathbb{E}\|s_\phi(x) - \nabla_x \log p_{data}(x)\|^2$$

2 Corrupt data, then match a known score

Inject Gaussian noise with a known perturbation kernel. The conditional score is available in closed form.



known denoising target points back to x

Gaussian perturbation target

$$\nabla_{\tilde{x}} \log p_\sigma(\tilde{x}|x) = \frac{x - \tilde{x}}{\sigma^2}$$

DSM loss $\mathcal{L}_{DSM}(\phi;\sigma) = \tfrac{1}{2}\mathbb{E}_{x,\tilde{x}}\|s_\phi(\tilde{x};\sigma) - \nabla_{\tilde{x}} \log p_\sigma(\tilde{x}|x)\|^2$

Equivalence theorem $\mathcal{L}_{SM}(\phi;\sigma) = \mathcal{L}_{DSM}(\phi;\sigma) + C$

Deep connection DSM learns the direction from noisy $\tilde{x}$ back to clean x — exactly DDPM's ε-prediction, up to a scale factor.

Tweedie's Formula

1 The Bayes-optimal denoiser

Given noisy $\tilde{x}$, the MSE-optimal estimate of clean x is the posterior mean.

$$D^*(\tilde{x}) = \mathbb{E}[x \mid \tilde{x}]$$

↓

2 Tweedie: the score gives the denoiser

If $\tilde{x} \mid x \sim N(\alpha x, \sigma^2 I)$, the posterior mean is recovered from the score of the noisy marginal p_σ.

Tweedie identity $$\alpha\,\mathbb{E}[x \mid \tilde{x}] = \tilde{x} + \sigma^2 \nabla_{\tilde{x}} \log p_\sigma(\tilde{x})$$

Special case $\alpha = 1$

$$\mathbb{E}[x \mid \tilde{x}] = \tilde{x} + \sigma^2 s(\tilde{x})$$

Knowing the score is knowing the optimal denoiser.

Bridge to DDPM: score ⇔ ε-prediction

For $x_\sigma = x + \sigma_\varepsilon$ with $\varepsilon \sim N(0, I)$, Tweedie gives:

$$s^*(x_\sigma, \sigma) = -\varepsilon^*(x_\sigma, \sigma)/\sigma$$

The DDPM noise predictor and the score model are the same denoiser under a per-noise-level rescaling.

Why Tweedie matters — DSM training is Bayes-optimal denoising; small score-guided steps gradually clean noise into data.

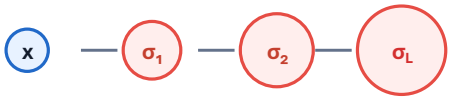
NCSN ⇔ DDPM: One Mechanism

Different discretisations, same idea: learn the direction that removes noise at each noise level.

NCSN / VE view

noise scales $\sigma_L \to \sigma_1$

Noise ladder



1 Corrupt data

Use a chosen noise scale σ; the marginal is explicit.

$$x_\sigma = x + \sigma_\varepsilon$$

2 Train score

Regression target is the direction back to the clean sample.

$$s_\theta(x_\sigma, \sigma) \approx -\varepsilon/\sigma$$

3 Sample

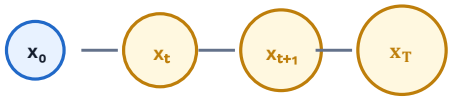
Langevin while σ decreases.

$\sigma_L \to \cdots \to \sigma_1$

DDPM / VP view

timesteps $T \to 0$

Diffusion chain



1 Corrupt data

Use a β schedule; noisy samples are directly computable.

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

2 Train noise predictor

The network predicts the Gaussian noise that was added.

$$\varepsilon_\theta(x_t, t) \approx \varepsilon$$

3 Sample

Reverse chain: noise → data.

$T \to \cdots \to 0$

Tweedie bridge: $s^*_\theta(x_\sigma, \sigma) = -\varepsilon^*_\theta(x_\sigma, \sigma)/\sigma$ ⇒ **same denoising signal, only scaled differently.**

Score SDE: Continuous-Time Unification

DDPM and NCSN unify as a continuous noising SDE plus a learned reverse-time score field.

Forward SDE

A continuous noising process: drift moves the mean, diffusion injects noise.

$$d\,x_t = f(x_t, t)\,d\,t + g(t)\,d\,w_t$$

Reverse-Time SDE

Run backward using the score $\nabla_x \log p_t(x)$.
Train $s_\phi(x,t)$ to approximate it.

$$d\,x_t = [f(x_t, t) - g(t)^2\, s_\phi(x_t, t)]\,d\,t + g(t)\,d\,\bar{w}_t$$

DDPM — **continuous-time score model** — **NCSN**

same training principle: estimate a denoising direction at every noise level

VP-SDE → continuous DDPM

$d\,x_t = -\tfrac{1}{2}\,\beta(t)x_t\,d\,t + \sqrt{\beta(t)}\,d\,w_t$
variance-preserving: $p_T \approx N(0, I)$

VE-SDE → continuous NCSN

$d\,x_t = \sqrt{(d\sigma^2(t)/dt)}\,d\,w_t$
variance-exploding: $p_T \approx N(0, \sigma^2_{max} I)$

Probability Flow ODE

$d\,x_t = [f(x_t,t) - \tfrac{1}{2}\,g(t)^2 s_\phi(x_t,t)]\,d\,t$
deterministic; same marginals

Key message: generation is numerical simulation of a learned reverse-time vector field.

04

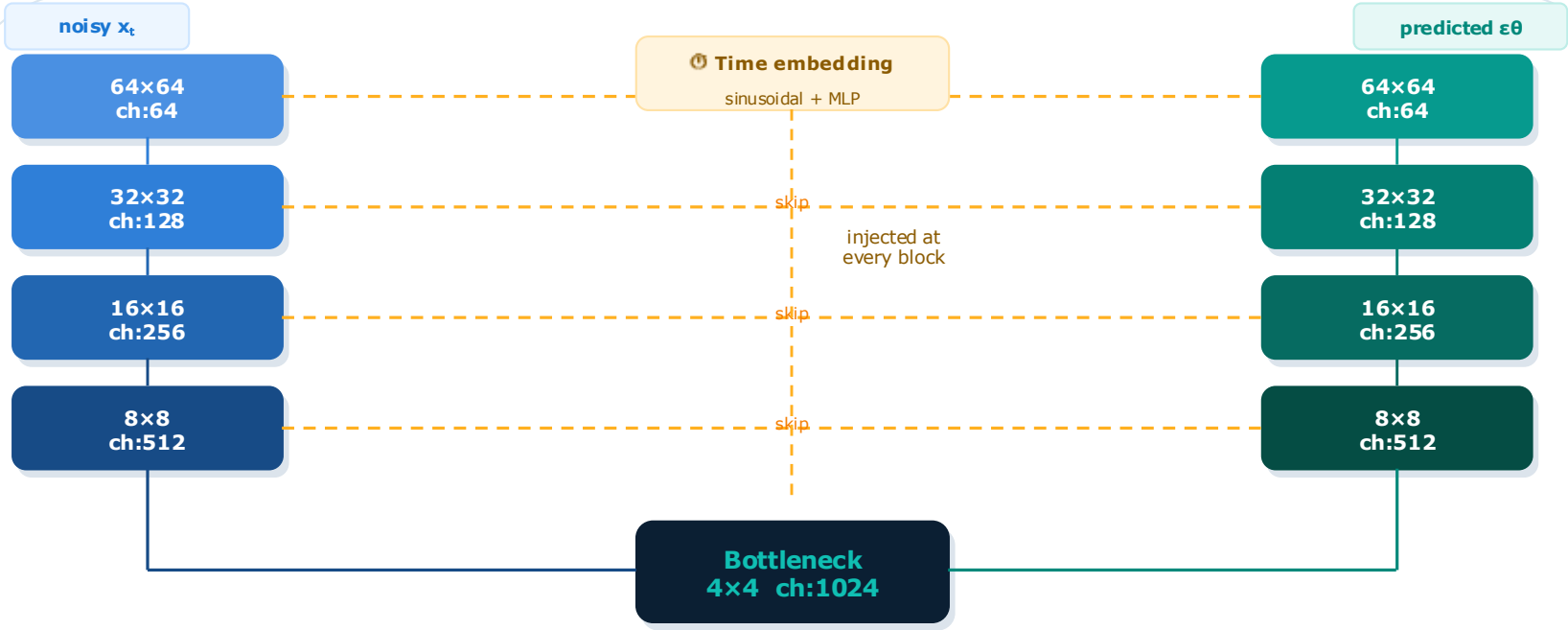
From Theory to Practice

Architectures, Guidance, and Latent Space

From the variational and score-based foundations to working systems. We close the loop on what makes diffusion models scale: time-conditioned U-Nets that handle all noise levels with a single network, classifier-free guidance for fine-grained control, and latent diffusion to bring megapixel generation within reach (Stable Diffusion, DALL-E 3).

Architecture: Time-Conditioned U-Net

One denoiser $\varepsilon\theta(x_t,t)$ is reused across all noise levels; timestep embeddings modulate every block.



Scaling Up: Guidance & Latent Space

Two engineering ideas made diffusion systems controllable and computationally feasible.

Classifier-Free Guidance (CFG)

Problem — Unconditional samples are diverse but may ignore the prompt.

Idea — Train with condition c and with c = ∅.

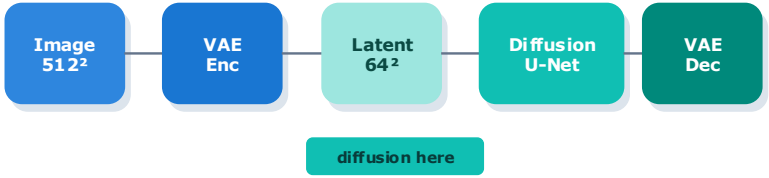
Inference — Blend the two predictions during sampling.

Effect — w>1 improves adherence, with a diversity trade-off.

$$\tilde{\varepsilon}_\theta(x_t, c) = \varepsilon_\theta(x_t, \emptyset) + w[\varepsilon_\theta(x_t, c) - \varepsilon_\theta(x_t, \emptyset)]$$

Amplify the direction that makes the sample match condition c.

Latent Diffusion (Stable Diffusion)



- Compressed latent space (8–16× smaller).
- Much faster training and inference.
- Text conditioning via cross-attention.
- Foundation of Stable Diffusion-style systems.

Applications Across Domains

Image Generation

DALL-E 3, Stable Diffusion, Imagen — text-to-image at photorealistic quality

Video Synthesis

Sora, Stable Video Diffusion — temporal consistency across frames

Audio & Music

AudioLDM, Stable Audio, DiffWave — speech and music generation

Protein Design

RFDiffusion, FrameDiff — generating novel protein backbones

Scientific Data

Weather forecasting (GenCast), molecular conformation, climate modelling

Medical Imaging

MRI / CT synthesis, data augmentation, anomaly detection

Open Problems & Research Frontiers

01 Sampling Speed

Even DDIM needs 50+ steps; distillation methods (consistency models, SDXL-Turbo) aim for 1-4 steps. How to maintain quality?

02 Evaluation Metrics

FID and CLIP scores don't fully capture quality, diversity, or factual accuracy. We lack a reliable, comprehensive benchmark.

03 Controllability

Fine-grained spatial and semantic control remains hard. Current methods (ControlNet, inpainting) are workarounds, not solutions.

04 Training Efficiency

Diffusion requires many forward/backward passes. Reducing compute while preserving generative quality is an active research area.

05 Safety & Bias

Models inherit biases from training data and can generate harmful content. Reliable filtering and auditing pipelines are not yet solved.

06 Continuous Domains

Most work is on images. Extending robustly to 3D scenes, long videos, and scientific data distributions raises fundamental open questions.

Key Takeaways

Forward = fixed, Reverse = learned

Add noise analytically; the neural network only needs to predict what was added.

One loss, one network

The simplified ε-prediction objective is elegant and stable to train.

Architecture matters

Time-conditioned U-Net + skip connections + embeddings make it practical.

Efficiency unlocks scale

Latent diffusion + DDIM bring diffusion to real-time production.

Still wide open

Speed, evaluation, control, safety — plenty of research left to do.

Further Reading

DDPM

Ho, Jain, Abbeel — Denoising Diffusion Probabilistic Models

NeurIPS 2020 · 2020

LDM

Rombach et al. — High-Resolution Image Synthesis with Latent Diffusion Models

CVPR 2022 · 2022

SCORE

Song & Ermon — Generative Modeling by Estimating Gradients of the Data Distribution

NeurIPS 2019 · 2019

CFG

Ho & Salimans — Classifier-Free Diffusion Guidance

NeurIPS 2022 Workshop · 2022

SDE

Song et al. — Score-Based Generative Modeling through SDEs

ICLR 2021 · 2021

BLOG

Lilian Weng — What are Diffusion Models? (lilianweng.github.io)

Recommended Tutorial · —

DDIM

Song, Meng, Ermon — Denoising Diffusion Implicit Models

ICLR 2021 · 2020

BOOK

Lai, Song, Kim, Mitsufuji, Ermon — The Principles of Diffusion Models

arXiv 2510.21890 · 2025 · this lecture follows ch 1–3

Practice Exercises

Optional self-study: core derivations and a minimal MNIST DDPM experiment.

01 Closed-Form Forward Marginal

START

Hint: sums of Gaussians

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\right)$$

SHOW

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

$$\bar{\alpha}_t = \prod_{s=1}^{t} (1-\beta_s)$$

02 ε-Prediction ⇔ x_0-Prediction

REPARAMETERISE

rescale the target

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

IDENTIFY c(t)

$$\|\varepsilon_\phi - \varepsilon\|^2 = c(t)\, \|\hat{x}_\phi - x_0\|^2$$

03 MNIST DDPM Experiment

OPTIONAL CODING

SETUP

MNIST 28×28 grayscale; normalize to [−1, 1].
Use T = 100 or 200 with a linear β-schedule.
Small time-conditioned U-Net/CNN is enough.

OBJECTIVE

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon$$
$$\mathcal{L} = \|\varepsilon - \varepsilon_\theta(x_t, t)\|^2$$

Train εθ to predict the noise added to each image.

WORKFLOW

show forward noise and reverse denoising grids

REPORT

- ✓ forward noising grid
- ✓ reverse denoising trajectory
- ✓ generated digit grid + loss curve
- ✓ short note + code link

Goal: connect the derivations to an end-to-end image diffusion model on MNIST.

Public Use & License

Context

Prepared for a University of Tokyo lecture in Spring 2026 and shared as a public educational version.

Disclaimer

Views are my own and do not represent official statements of the University of Tokyo or RIKEN.

License

Third-party materials

Figures, screenshots, trademarks, and examples credited separately remain under their respective rights.

Suggested citation: Wei Huang, “An Introduction to Diffusion Models: Theory and Practice,” public lecture slides, 2026.