PHASE 5
Generative Models

Autoencoders, VAEs, GANs, and diffusion — how machines create

Phase goal

Understand how models learn to *generate* data: the latent-variable view (VAEs), the adversarial game (GANs), and the denoising process (diffusion) behind modern image and audio generation.

Estimated time: 4–5 weeks | **Track:** Comprehensive ML & Deep Learning Curriculum

A self-paced, depth-first path from mathematical foundations to production deep learning.

Contents

1 Discriminative vs. generative

Everything so far has been mostly *discriminative*: learn $p(y \mid \mathbf{x})$ to predict a label. **Generative** models learn the data distribution $p(\mathbf{x})$ itself, so you can sample new examples that look like the training data — images, audio, molecules, text. The central difficulty: $p(\mathbf{x})$ for real data (say, 256×256 images) is absurdly high-dimensional and complex. The three model families in this phase are three different bargains for making that tractable.

Key idea

All three families exploit the same observation: real data lives near a low-dimensional *manifold* inside its high-dimensional space (the SVD/PCA intuition from Phase 0). VAEs learn an explicit latent code for that manifold; GANs learn to map noise onto it; diffusion learns to walk back onto it from noise. Master the manifold picture and the rest follows.

2 Autoencoders

An autoencoder learns to compress and reconstruct: an **encoder** maps $\mathbf{x}$ to a low-dimensional latent $\mathbf{z}$, a **decoder** reconstructs $\hat{\mathbf{x}}$, trained to minimize reconstruction error $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$. The bottleneck forces the network to keep only the essential structure — a nonlinear generalization of PCA, useful for denoising, anomaly detection, and representation learning.

Common pitfall

A plain autoencoder is *not* a good generator. Its latent space has "holes" — sampling a random $\mathbf{z}$ usually decodes to garbage because the encoder never organized the latent space to be continuous or complete. Fixing exactly this is the motivation for the VAE.

3 Variational Autoencoders (VAEs)

A VAE makes the latent space *probabilistic and well-structured* so you can sample from it. The encoder outputs a distribution $q(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ rather than a point, and we regularize that distribution toward a standard Gaussian prior.

Definition

VAEs maximize the **Evidence Lower Bound (ELBO)**:

$$\mathcal{L} = \underbrace{\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x} \mid \mathbf{z})]}_{\text{reconstruction}} - \underbrace{D_{\mathrm{KL}}\big(q(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z})\big)}_{\text{regularize latent to prior}}.$$

The KL term pulls each encoded distribution toward $\mathcal{N}(\mathbf{0}, \mathbf{I})$, filling the latent space so that sampling $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and decoding produces plausible new data.

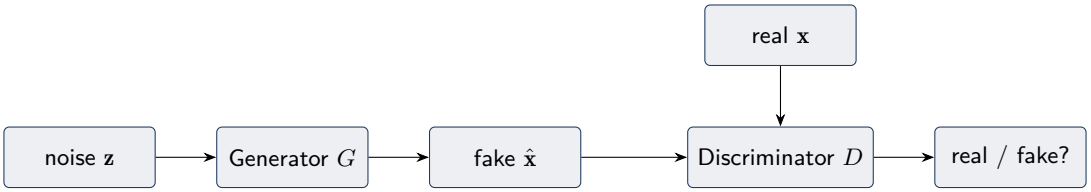
Intuition

The **reparameterization trick** — writing $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ — moves the randomness "outside" the network so gradients can flow through $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. This is the key engineering trick that makes VAEs trainable by ordinary backprop. VAEs produce smooth, interpolatable latent spaces but somewhat blurry samples (a consequence of the Gaussian likelihood and the averaging it encourages).

4 Generative Adversarial Networks (GANs)

A GAN pits two networks against each other. The **generator** G maps noise $\mathbf{z}$ to fake samples; the **discriminator** D tries to distinguish real from fake. They play a minimax game:

$$\min_G \max_D \; \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))].$$



Intuition

As D gets better at spotting fakes, G is pushed to make more convincing ones — an arms race that drives sample quality up. At the theoretical optimum, G's distribution matches the data and D is reduced to guessing (50%). GANs produce sharp, high-fidelity images but are notoriously finicky to train.

Common pitfall

GAN failure modes you must recognize: **mode collapse** (the generator produces only a few outputs, ignoring data diversity); **training instability / non-convergence** (the two losses oscillate rather than settle); and **vanishing discriminator gradients** (if D wins too easily, G gets no signal). Mitigations developed over years: Wasserstein loss (WGAN-GP), spectral normalization, two-timescale updates (TTUR), and careful architecture (DCGAN/StyleGAN). Evaluate with FID, not loss values.

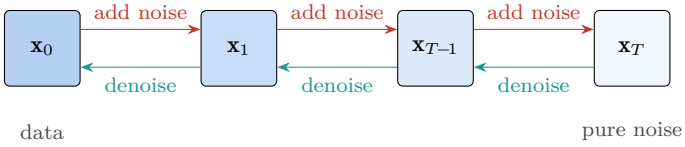
5 Diffusion models

Diffusion is the backbone of modern image and audio generation (Stable Diffusion, DALL · E, Imagen, Midjourney). The idea is strikingly simple: **gradually add noise to data, then learn to reverse it**.

Definition

Forward process: over T steps, progressively add Gaussian noise to a sample until it is pure noise — a fixed, parameter-free Markov chain $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$. **Reverse process:** train a neural network $\epsilon_\theta(\mathbf{x}_t, t)$ to predict the noise added at each step, so you can iteratively *denoise* from random noise back to a clean sample. The DDPM training loss is beautifully simple:

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_0,t,\boldsymbol{\epsilon}}\left[\; \|\boldsymbol{\epsilon} - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \;\right].$$



Key idea

Diffusion trades the GAN's unstable adversarial game for a *stable regression* problem (predict the noise), which is why it scaled so well and largely displaced GANs for high-quality image generation. The price is slow sampling — many denoising steps — which spawned fast samplers (DDIM, DPM-Solver) and distillation. The denoising network is typically a **U-Net** (Phase 3) or, increasingly, a Transformer (DiT).

5.1 Conditional and text-to-image generation

To control generation, condition the model on a signal: a class label, or text. **Latent diffusion** (Stable Diffusion) runs the diffusion process in a compressed VAE latent space — far cheaper than pixel space. Text conditioning uses a text encoder (often **CLIP**, which aligns image and text embeddings in a shared space) and **classifier-free guidance** to steer samples toward the prompt. Conceptually: CLIP says "how well does this image match the text," and guidance nudges each denoising step to increase that match.

	VAE	**GAN**	**Diffusion**
Training	stable	unstable	stable
Sample quality	blurry	sharp	state-of-the-art
Sampling speed	fast	fast	slow (many steps)
Latent space	smooth, usable	less structured	(in latent diffusion)
Mode coverage	good	can collapse	good

6 Checkpoint project

Train a small generative model

Goal: watch generation emerge firsthand on a simple dataset (Fashion-MNIST or MNIST).

- **Track A — VAE:** build encoder/decoder, implement the ELBO with the reparameterization trick, train, then (1) sample new images from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and (2) interpolate between two latent codes to see the smooth manifold.
- **Track B — DDPM:** implement the forward noising schedule and a small U-Net noise predictor; train on the simple ϵ-prediction loss; sample by iterative denoising and visualize the reverse trajectory.

Definition of done: a grid of generated samples, a latent interpolation (VAE) or a denoising trajectory (DDPM), the loss curve, and a paragraph comparing what each approach was like to train.

Stretch: add class conditioning (generate a specified digit/garment) and, for DDPM, compare DDPM vs. DDIM sampling speed/quality.

Phase 5

- **Lilian Weng**'s blog — "What are Diffusion Models?" and her GAN/VAE posts (exceptionally clear).
- **Hugging Face** *diffusers* library + its tutorials (the practical entry point).
- Papers (skim for intuition): *Auto-Encoding Variational Bayes* (VAE); *Generative Adversarial Networks*; *Denoising Diffusion Probabilistic Models* (DDPM); *High-Resolution Image Synthesis with Latent Diffusion* (Stable Diffusion).

- Karpathy / fast.ai diffusion-from-scratch walkthroughs.

7 Phase 5 self-check

- Explain why a plain autoencoder cannot generate and how the VAE fixes it.
- Write the ELBO and explain the reconstruction vs. KL terms and the reparameterization trick.
- Describe the GAN minimax game and name three failure modes with mitigations.
- Explain the forward/reverse diffusion processes and the simple ϵ-prediction loss.
- Compare VAE/GAN/diffusion on quality, stability, and sampling speed.
- Explain latent diffusion and classifier-free guidance at a conceptual level.