ARTIFICIAL INTELLIGENCE AND NEURAL NETWORKS

Lecture 8b – Generative Models: GANs, Diffusion, and Control

Chizhi Chris ZHANG

zhangchizhi@ciomp.ac.cn

Advanced Computing and Digital Technology Research Center

University of Chinese Academy of Sciences

Spring 2026

Today's Question

What we are trying to answer

How do neural networks learn to generate realistic content instead of only classifying inputs?

Why this lecture matters

AI8 explained why generative AI became visible to ordinary users. NN8 explains the main model families behind that change and why different families make different tradeoffs.

What changes from NN7

NN7 focused on transformers for language. NN8 shifts to generation more broadly and asks how neural networks can synthesize new samples in image and multimodal settings.

From NN7 to NN8

Last time

We studied how transformer models process sequences and generate language token by token.

Today

We move to generative modeling as a broader neural problem: how to learn a distribution well enough to sample convincing new outputs.

One sentence

NN7 was about understanding and generating sequences. NN8 is about learning how to generate whole samples.

Discriminative and Generative Goals

Discriminative modeling

Learn to answer a question about an input, such as “what class is this” or “what number should we predict”.

Generative modeling

Learn a distribution well enough that the model can create new samples that look like they belong to the same world.

The shift is from deciding among options to producing plausible new content.

Why Generation Is Hard

The difficulty

Natural images, sound, and language live in high-dimensional spaces with a lot of hidden structure.

What the model must get right

It has to produce outputs that look realistic, stay diverse, and still respond to conditions or control signals.

Why this is not a small extension

Generation is not only about learning a label boundary. It is about learning what kinds of full samples are possible.

A Map of the Main Families

GANs

Learn through competition between a generator and a discriminator. Often visually impressive, but training can be unstable.

Autoregressive models

Generate one piece at a time. Natural for language and sequence tasks, but slower for large images.

VAEs

Learn a latent representation with an explicit probabilistic structure. Usually easier to train, but often softer in output quality.

Diffusion models

Learn to reverse noise step by step. They proved strong in image quality and controllability, especially when scaled well.

Why keep all four in mind

They are solving the same broad problem with very different training stories, and those choices shape quality, speed, controllability, and stability.

Latent Variable Intuition

Simple sampling view

Draw a latent variable z from a simple distribution, then let a neural network map z to an output sample x.

Why latent space is useful

It gives the model a compact internal space where interpolation, control, and variation become easier.

This idea appears in several generative families, even when the details differ.

No Single Objective Solves Everything

What we want

Outputs should look real, cover many possible modes, follow the condition, and be useful for the task.

Why this is tricky

One metric rarely captures realism, diversity, controllability, and safety all at once.

Engineering consequence

Generative modeling is always partly about tradeoffs, not only about maximizing one number.

GAN as a Competitive Game

The story

One network generates samples. Another network tries to detect whether each sample is real or fake.

Why this idea was powerful

The generator improves because the discriminator becomes a moving critic instead of a fixed hand-written rule.

Mathematical form

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

Training Loop Intuition

Step 1

Update the discriminator on real and generated batches.

Step 2

Freeze the discriminator, then update the generator so its outputs look harder to reject.

Where instability comes from

If one side becomes much stronger than the other too early, gradients can become weak or misleading.

A GAN Example

Why this example was memorable

CycleGAN showed that image-to-image translation could work even without perfectly matched training pairs. That made the idea easy to demonstrate and easy to remember.

Mode Collapse

The symptom

The generator keeps producing many near-duplicates because one narrow region of output space happens to fool the discriminator.

Why students should care

This is a vivid example of a model looking successful at first glance while actually losing diversity.

In generative modeling, realism alone is not enough. Variety matters too.

How People Stabilized GANs

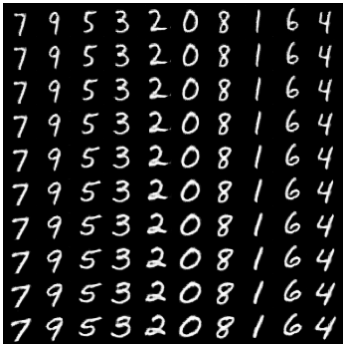
Typical ideas

Wasserstein-style objectives, gradient penalties, spectral normalization, and better architectures all tried to make the training signal more informative.

Big lesson

Good generative performance often depends as much on optimization design as on raw neural-network size.

Can Latent Factors Become More Interpretable



Info-style idea

Encourage parts of the latent code to align with interpretable features such as rotation, thickness, or style.

Why this matters

Control becomes easier when the latent space is not only compact, but also somewhat organized.

Diffusion Starts with a Different Story

The core idea

Instead of generating a full sample in one shot, gradually add noise during training and learn how to reverse that noising process.

Why this was appealing

The model gets to solve many small denoising problems rather than one giant generation jump.

That shift in viewpoint is one reason diffusion became so influential.

Forward Noising Process

Single-step form

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I)$$

Read in words

At each step, keep a little of the old signal and mix in a little more noise.

Result

If you repeat this long enough, structure disappears.

Reverse Denoising Process

What the model learns

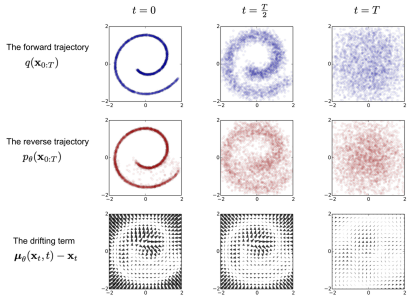
Given a noisy state, predict the noise or the score so the sample can move one step back toward structure.

Why this is powerful

Generation becomes iterative refinement. The model improves a rough guess again and again until a coherent sample appears.

This is why diffusion is often explained as “start from noise, then clean it up”.

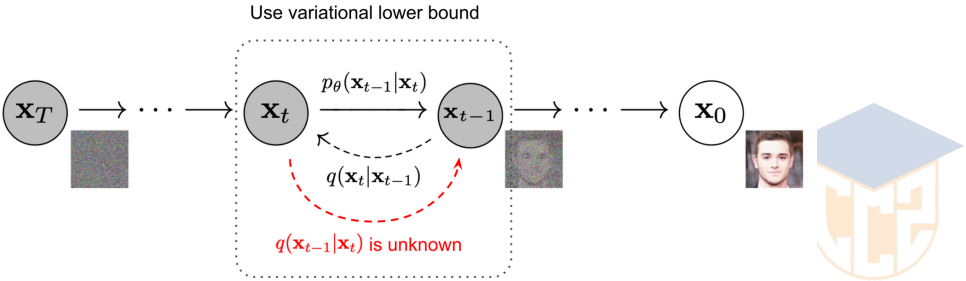
Visual Intuition



How to read this picture

The left side shows structure breaking apart under noise. The right side suggests the inverse idea: from noise back toward a meaningful image.

DDPM Pipeline



Training and sampling are mirror images

Training shows the model examples of structured images plus added noise. Sampling starts from noise and repeatedly applies what the model has learned.

Algorithm View

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

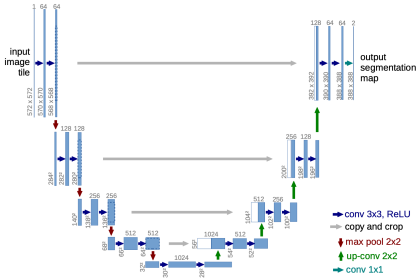
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

What this algorithm is really saying

Repeat a simple denoising update many times. The details matter for performance, but the classroom-level idea is still iterative refinement.

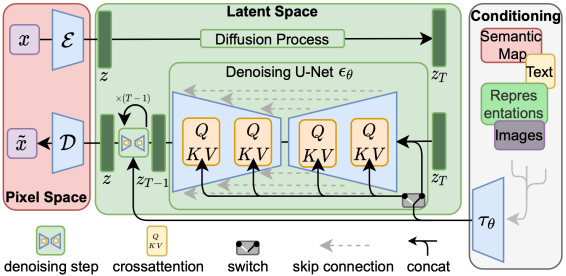
Why U-Net Fit So Well



Reason

U-Net mixes local detail with broader context across multiple scales. That makes it a strong denoiser for image generation.

Latent Diffusion Changed the Cost Curve



Why teams cared

Latent diffusion reduced the cost of working at full image resolution, which made larger and more practical image generation systems easier to deploy.

Conditioning Channels

Common condition types

- text prompts
- class labels
- masks and edges
- pose or layout signals

Key effect

Conditioning turns generation from free sampling into controlled generation.

Conditional Diffusion View

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $f_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log f_\phi(y|x_t), \Sigma)$
end for
return x_0

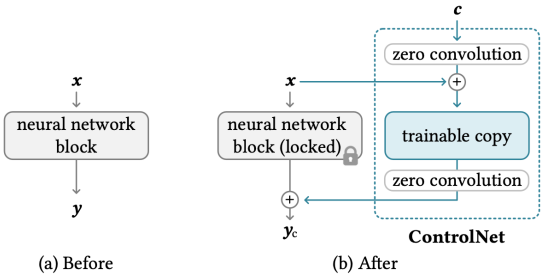
Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $f_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log f_\phi(y|x_t)$
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$
end for
return x_0

Practical lesson

The same denoising engine becomes much more useful once it can respond to conditions instead of producing random but unrelated outputs.

Structured Control



Why industry likes this

Prompt text may describe intent, but structure hints such as edges, depth, or pose tell the model where things should go. That greatly improves predictability.

Why Prompt Alone Is Not Enough

Prompt-only limitation

Natural language is flexible, but often too vague to fully specify layout, geometry, identity, or consistency across several outputs.

Engineering response

Add stronger control signals, better interfaces, editing loops, or external tools instead of relying on prompt wording alone.

This is where many real products move from impressive demo to usable system.

Failure Types to Watch

Semantic	Structural	Safety
The picture does not match the request.	Hands, symmetry, perspective, or object counts break.	The output violates policy, privacy, or trust.

Why this page matters

Generative errors are not one thing. Different failures need different checks and different mitigation strategies.

How We Evaluate Generative Models

Automatic measures

- sample quality
- diversity
- condition alignment
- speed and cost

Human evaluation

People still need to judge realism, usefulness, preference, and whether the result actually serves the application.

One Metric Can Mislead You

Pitfall

An automatic score can improve even while diversity drops, artifacts increase, or the model becomes worse for a real product need.

Better rule

Use metrics as instruments, not as substitutes for judgment.

Good generative evaluation is always partly contextual.

Data-Centric Problems

Dataset risks

- biased representation
- poor captions or labels
- copyright problems
- private or sensitive material

Consequence

A powerful model trained on weak or risky data can still produce outputs that are socially or legally problematic.

Red-Teaming Questions

Ask before deployment

- Can the model be used for deception?
- Does it fail on certain groups or styles?
- Can users force unsafe content?
- How easy is it to misuse the system at scale?

Goal

We are not only testing quality. We are testing how the model behaves under pressure and misuse.

When a Small Model Beats a Big One

Where a big model wins

Better quality ceiling, broader capability, and stronger generalization.

Where a small model can win

Lower cost, lower latency, simpler deployment, easier on-device use, and sometimes easier control for a narrow task.

The best model is the one that fits the job, not the one with the largest name.

Compute Budget Is a Design Choice

Training side

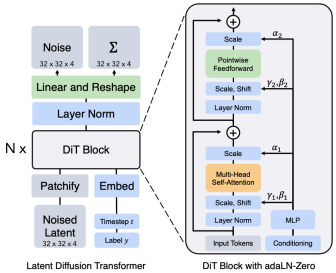
Large datasets, long schedules, and expensive hardware can improve model quality, but only at real cost.

Inference side

Users care about response time, memory use, and operating cost. Product teams care about all three.

That is why generative modeling is always both a neural-network problem and a systems problem.

From U-Net to DiT



Trend

As transformers proved useful in other settings, researchers started adapting transformer-style blocks to image generation too.

Deployment Is More Than the Model

Model

Choose the architecture, training recipe, and control path.

System

Add serving, caching, editing tools, moderation, and user workflow.

Governance

Add policy, logging, review, and incident response.

Why AI8 and NN8 Belong Together

AI8 view

Generative AI looked like prompts, collaboration, workflow, risk, and social impact.

NN8 view

Under that surface sit concrete neural mechanisms: GAN competition, diffusion denoising, conditioning channels, and architecture tradeoffs.

The application story and the model story are two halves of the same lecture pair.

Summary

- Generative modeling is harder than ordinary prediction because the model must produce full samples, not only labels.
- GANs framed generation as a competition, which was powerful but often unstable.
- Diffusion framed generation as iterative denoising, which proved easier to scale and control.
- Conditioning and structured control turn raw generation into something closer to a usable tool.
- Evaluation, data quality, safety, and compute tradeoffs matter as much as model architecture.

Next Time

Where the course is going

We have now covered the application story and the mechanism story of modern generative AI.

Next lecture

The next step is to build from this foundation and examine the later directions that extend, specialize, or operationalize these generative ideas.

Thank You