Week 11

Diffusion Models (2)

[ECEA0649/ECE40049] Deep Learning for Image Processing | Spring 2026

Kihyun Na (Research Professor)
BK21 AI Education and Research Group &
Institute for Information and Communication Technology,
Handong Global University

HANDONG GLOBAL UNIVERSITY

Curriculum

** Adjusted based on survey results*

Wk 1	OT + Introduction
Wk 2	DL Fundamentals Review
Wk 3	CNN
Wk 4	From Sequence Modeling to Transformer
Wk 5	Transformer in Vision
Wk 6	Detection & Segmentation
Wk 7	Self-Supervised Learning
Wk 8	Review Literacy + Role Explanation
Wk 9	Foundation Models (CLIP, SAM) + Paper #1
Wk 10	Diffusion Models + Paper #2
Wk 11	Diffusion Models (2) + Paper #3 (Today)
Wk 12	Vision-Language Models + Paper #4
Wk 13	VLM Applications + Paper #5
Wk 14	Video Understanding + Paper #6
Wk 15	Embodied AI & Robot Vision + Paper #7
Wk 16	Miniconference (Final Project)

Lecture Lecture + Paper Miniconference

Today's Agenda

Wk 11 = Flow Matching, Guidance, and Controllable Generation

01 **Score-Based View & Flow Matching** — 30 min

Score → SDE/ODE → Flow Matching. Simpler math, straighter paths

02 **LDM + CFG + Distillation** — 15 min

Latent space, guidance, and making it fast (FLUX)

03 **Controllable Generation** — 15 min

ControlNet, IP-Adapter, unified approaches → Paper #3 (UniCon)

Part 1

Score-Based View & Flow Matching

From noise prediction to velocity fields

Quick Recap: Wk 10

Generative Models → VAE (ELBO) → GAN → Diffusion

Key takeaways from last week:

1. Generative models learn p(x) — the data distribution
2. VAE: encoder-decoder + ELBO (intractable → lower bound)
3. GAN: adversarial training, sharp but unstable
4. DDPM: gradual denoising, each step = supervised regression

   Forward: $x_t = \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{(1-\bar{\alpha}_t)} \cdot \varepsilon$

   Loss: $L_{\text{simple}} = E[\,||\varepsilon - \varepsilon_\theta(x_t, t)||^2\,]$
5. DDIM: 50× speedup, same model, no retraining
6. DiT: U-Net → Transformer backbone (seminar)

Today: everything that makes this more practical and controllable.

$$\mathcal{L} = \mathbb{E}_{x_0,t,\epsilon}\left[w_t \|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]$$

over p_{data} over $[1, T]$ over $\mathcal{N}(0, \mathbf{I})$

Why Do We Need a Different View?

DDPM works, but can we make it simpler?

DDPM works well, but three things are complex:

1. Noise schedule: $\beta_t, \alpha_t, \bar{\alpha}_t$ — three coupled variables to define
   (linear? cosine? shifted? — sensitive to choice)

2. Slow sampling: 1000 steps by default
   (DDIM helps, but still ~50 steps)

3. Training derivation: ELBO → hierarchical VAE → reparameterize → L_{simple}
   (mathematically elegant but hard to teach and understand)

Question: can we extract the **essence** of what DDPM does and rebuild it with simpler math?

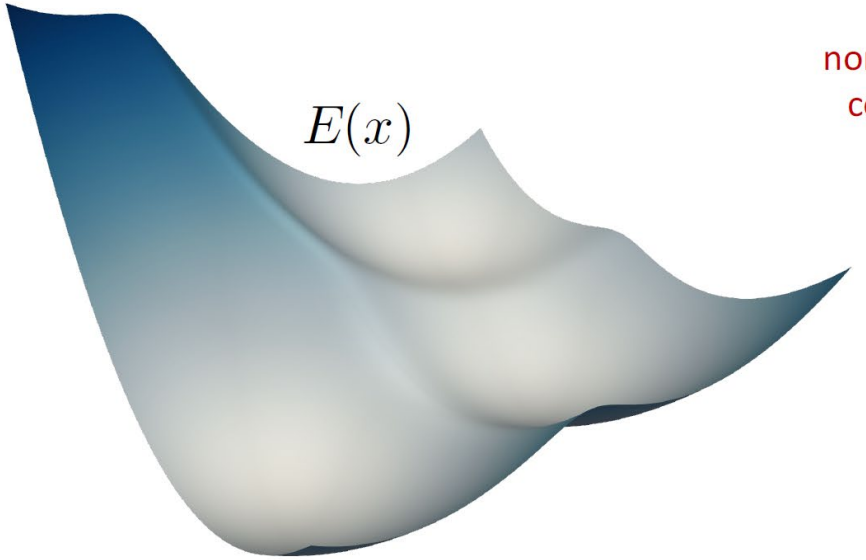
The answer comes from understanding what ε-prediction really means.

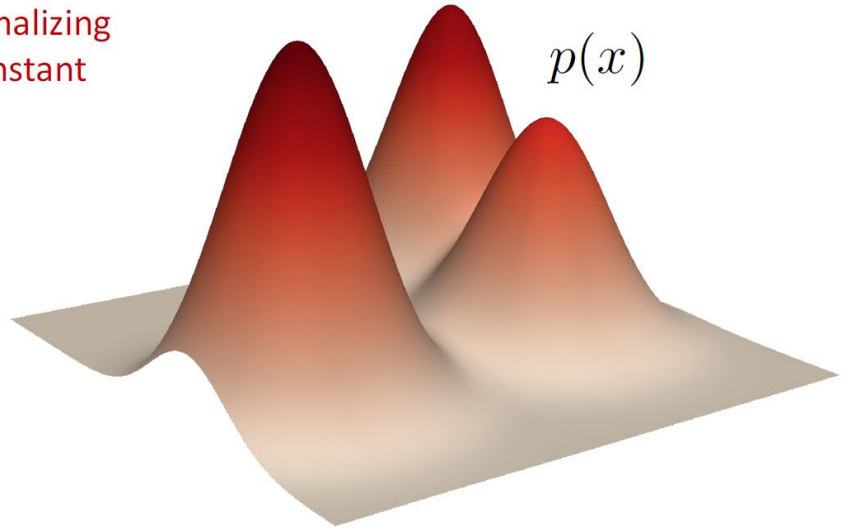
Energy-based Models

- Define a scalar function, called “energy”.
- At inference time, find x that minimizes energy
- We can use an energy to model a probability distribution

$$p(x) = \frac{\exp(-E(x))}{Z}$$

normalizing constant





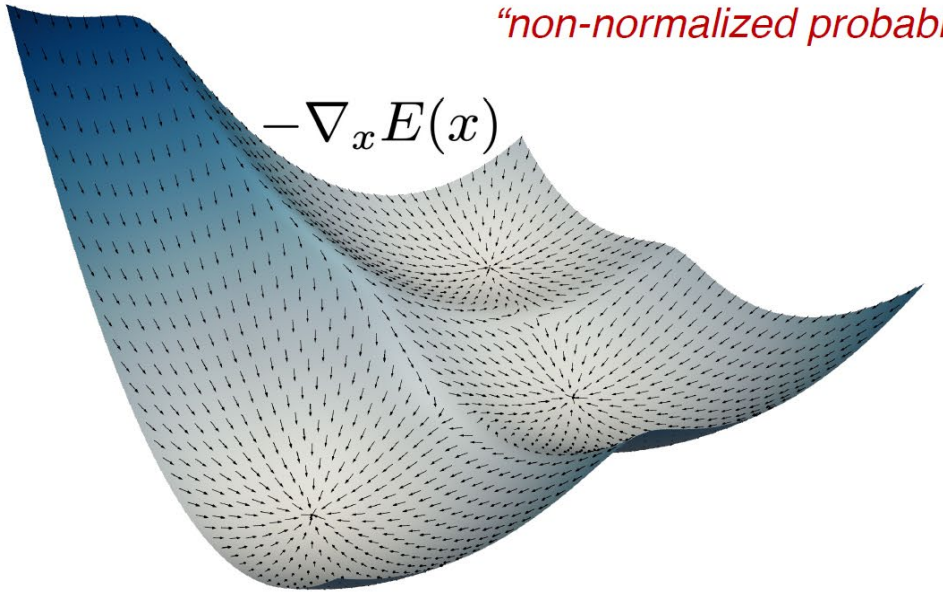
Energy-based Models

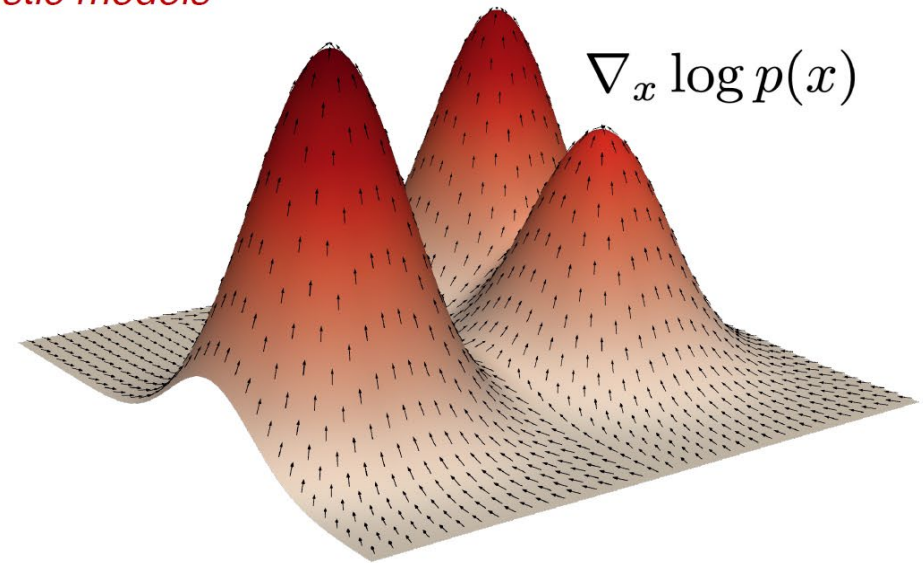
- **"Score function"**: gradient of log-probability

$$\nabla_x \log p(x) = -\nabla_x E(x) - \nabla_x \log Z$$

w.r.t. x $\quad$ $=0$

"non-normalized probabilistic models"





The Score-Based View

ε prediction = score estimation (Song et al., ICLR 2021)

What DDPM Really Does

Score function:

s(x) = $\nabla_x \log p(x)$ → "direction toward higher density"

Key equivalence:

$\varepsilon_\theta \propto -\nabla_x \log p_t(x_t)$

"Predict the noise" = "estimate the direction toward real data"

DDPM was secretly doing score estimation all along!

Two Paths Forward

Path 1: SDE (stochastic)

Reverse the noise process

= DDPM sampling (Random, diverse outputs)

Path 2: ODE (deterministic)

Follow the score deterministically

= DDIM sampling! (Same noise → same image)

Path 2 is the key insight:

ODE has a velocity field Learn that velocity directly

→ Flow Matching!

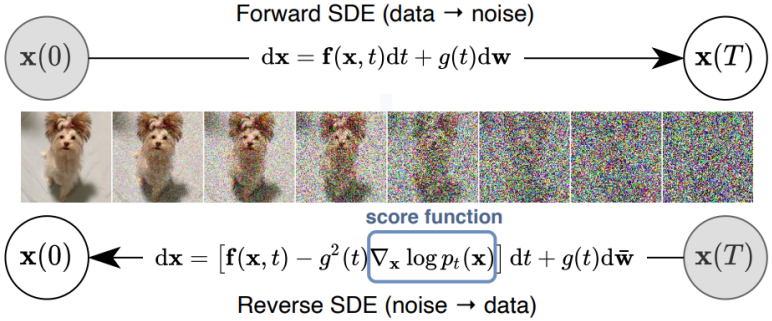


Figure 1: **Solving a reverse-time SDE yields a score-based generative model.** Transforming data to a simple noise distribution can be accomplished with a continuous-time SDE. This SDE can be reversed if we know the score of the distribution at each intermediate time step, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$.

Recognition vs. Generation

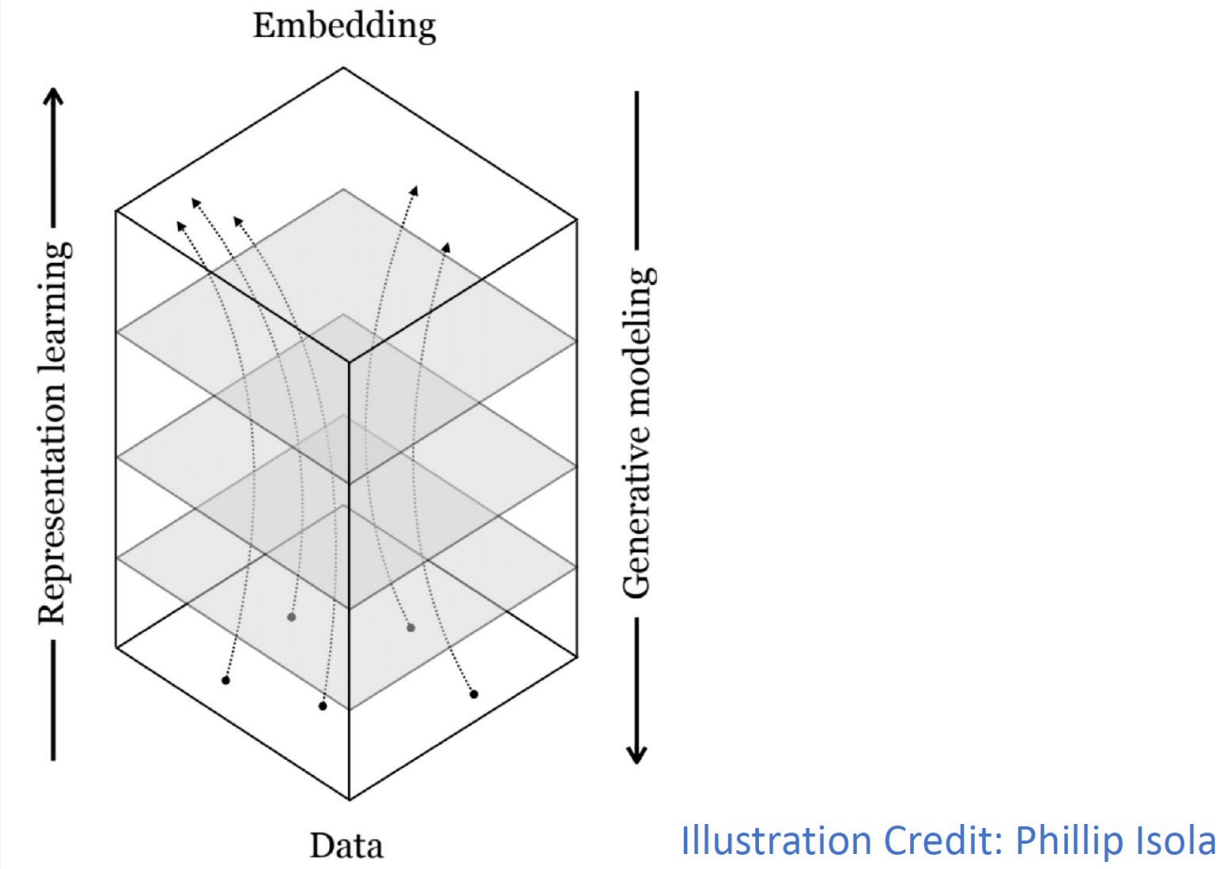
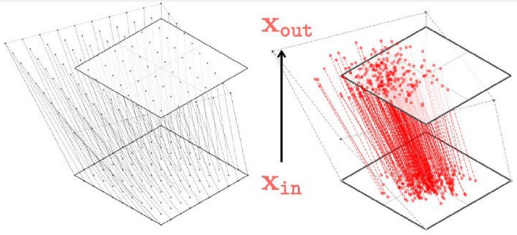


Illustration Credit: Phillip Isola

Recognition vs. Generation

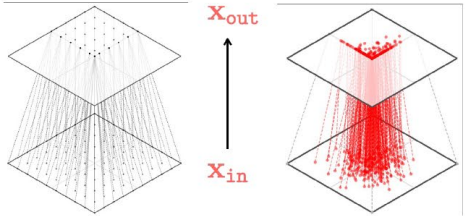
linear

$$\mathbf{x}_{out} = \mathbf{W}\mathbf{x}_{in} + \mathbf{b}$$



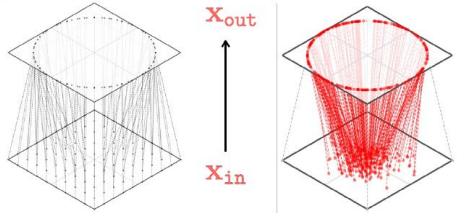
relu

$$x_{out}[i] = \max(x_{in}[i], 0)$$



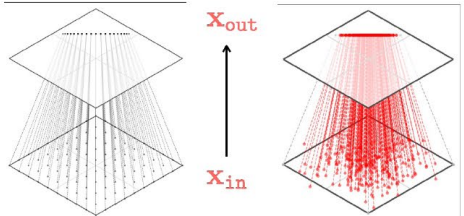
L2-norm

$$x_{out}[i] = \frac{x_{in}[i]}{\|\mathbf{x}_{in}\|_2}$$



softmax

$$x_{out}[i] = \frac{e^{-\tau x_{in}[i]}}{\sum_{k=1}^{K} e^{-\tau x_{in}[k]}}$$



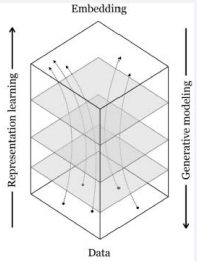
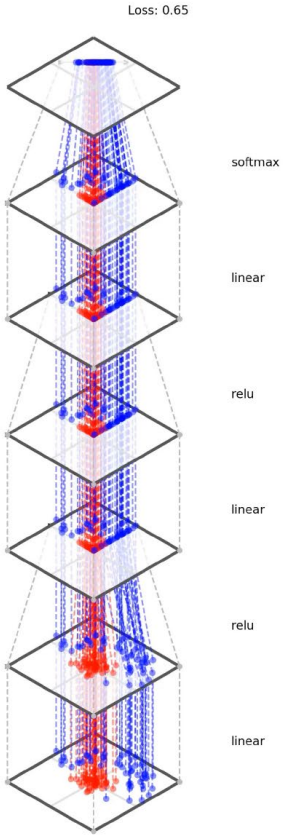


Illustration Credit: Phillip Isola

Recognition vs. Generation



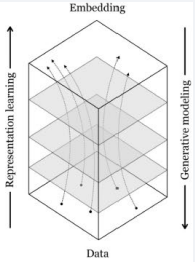
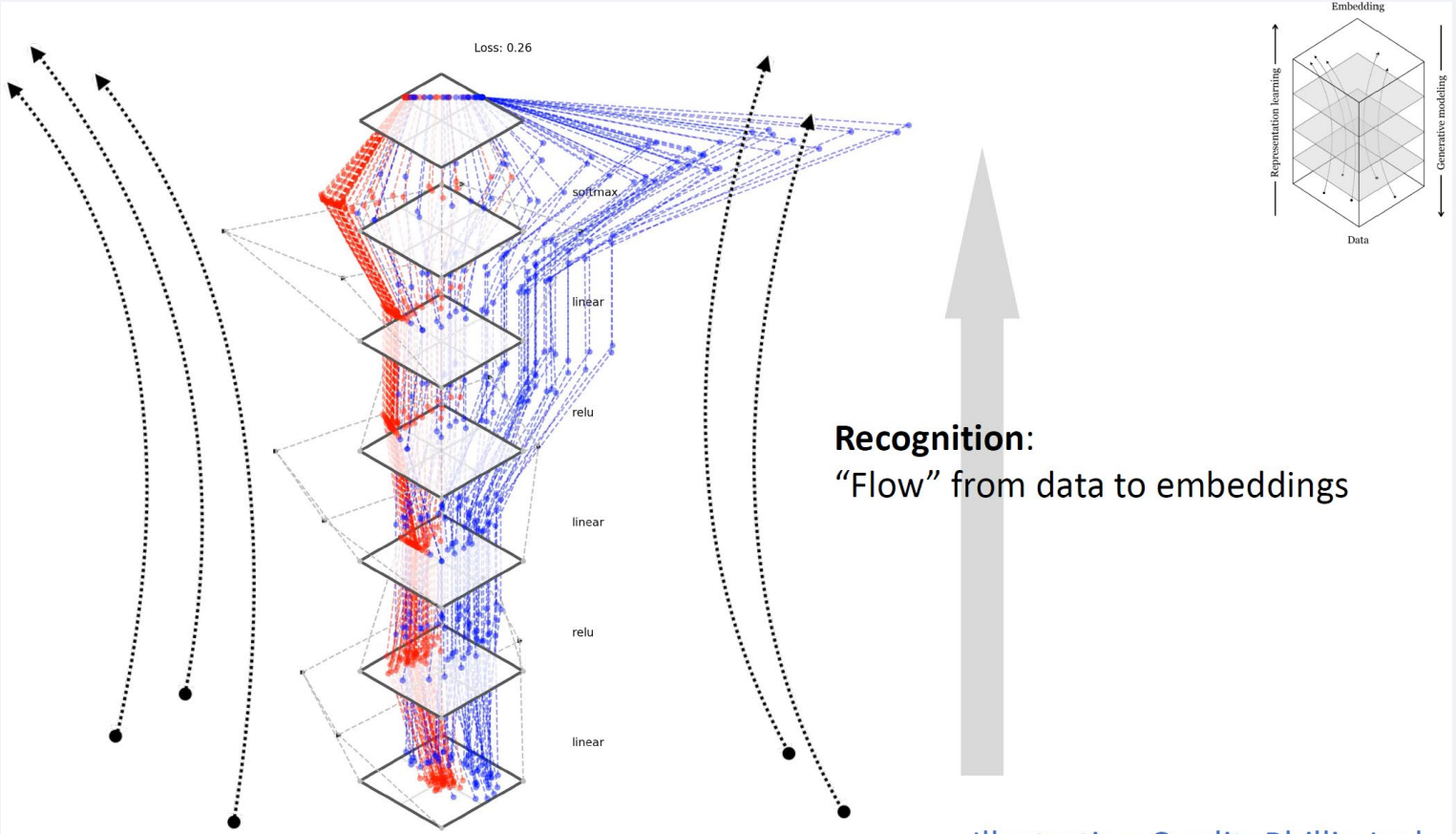


Illustration Credit: Phillip Isola

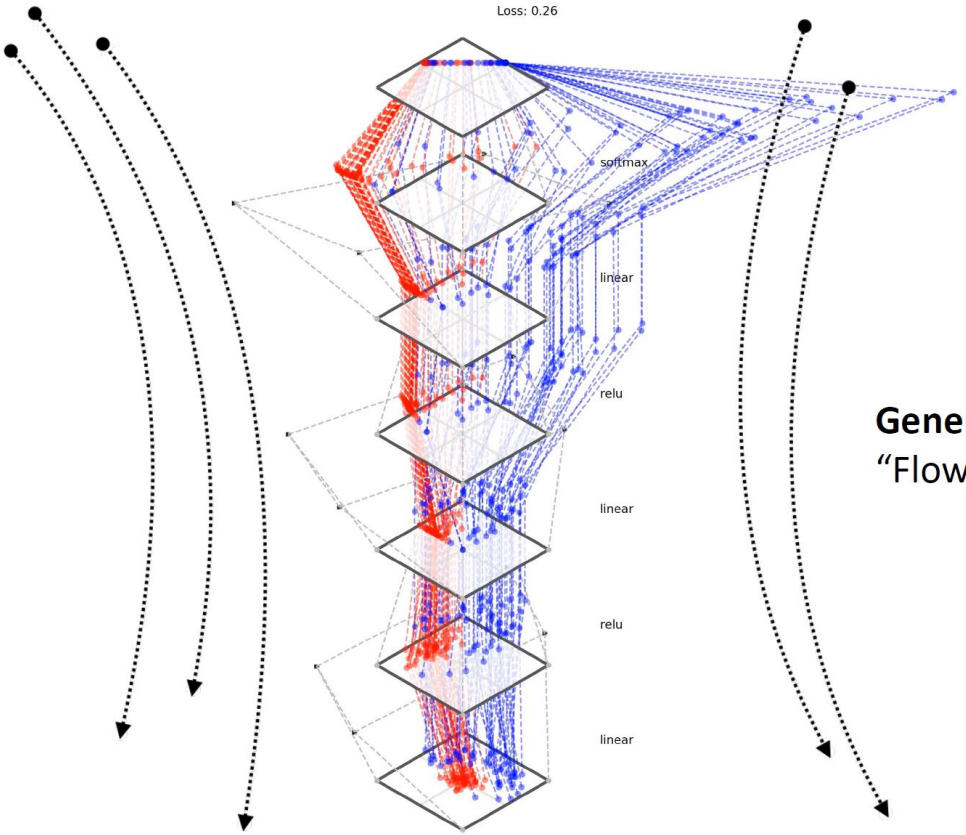
Recognition vs. Generation

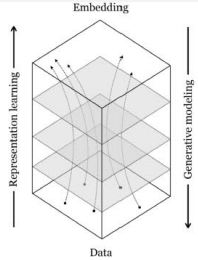


Recognition:
"Flow" from data to embeddings

Illustration Credit: Phillip Isola

Recognition vs. Generation



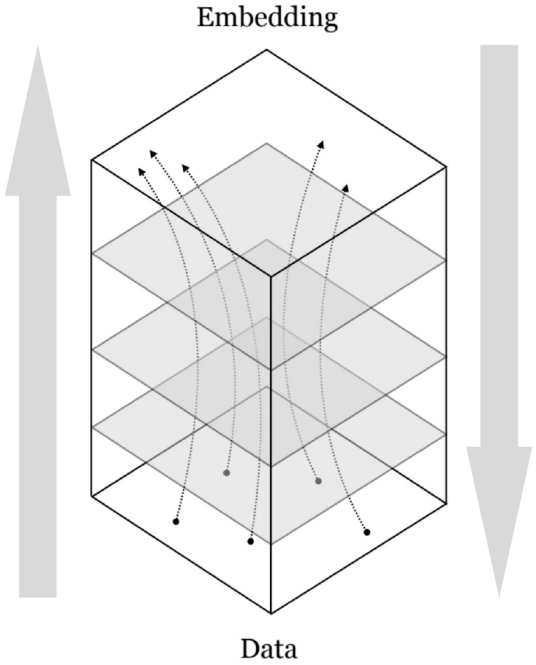


Generation:
"Flow" from embeddings to data

Illustration Credit: Phillip Isola

Recognition vs. Generation

Recognition:
determined
data-to-label mapping



Generation:
unknown
"noise"-to-data mapping
(infinite possibilities)

Construct the mapping?

- Continuous Normalizing Flow (in Neural ODE)
- Flow Matching

Generative models

∪

Flows

∪

Flow Matching

∪

Diffusion Models

Flow Matching: The Core Idea

Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023

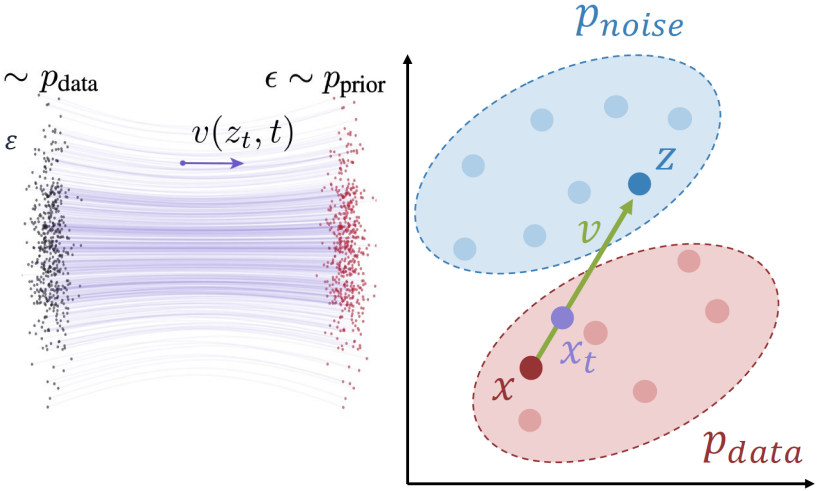
What is Flow Matching?

Learn a velocity field $v_\theta(x, t)$
that transports noise → data

Trajectory: $x_t = t \cdot x_0 + (1 - t) \cdot \varepsilon$
(linear interpolation!)
t=0: pure noise
t=1: clean data

Velocity: $dx_t/dt = x_0 - \varepsilon$
(constant! just a straight line)

Loss: $||v_\theta(x_t, t) - (x_0 - \varepsilon)||^2$
MSE regression, again!



Why is this better?

Simpler math:
No noise schedule (β, ᾱ)
No ELBO derivation needed
Just linear interpolation

Straighter paths:
Linear (Conditonal) → fewer steps needed
DDPM: curved, ~50 steps
FM: straight, ~30 steps

Same quality:
Converges to same distribution
Easier to train and tune

SD3, FLUX, Sora: all FM-based

Relation: Flow Matching and Diffusion

Flow Matching:

$$z_t = a_t x + b_t \epsilon$$

$$v_t = a_t' x + b_t' \epsilon$$

$$\mathcal{L} = \mathbb{E}\left[\|v_\theta(z_t, t) - v_t\|^2\right]$$

Solve ODE: $\frac{d}{dt} z_t = v_\theta(z_t, t)$

same

rewrite ϵ in z, v

in ϵ space

solve ODE in v space

Viewed as Diffusion:

$$z_t = a_t x + b_t \epsilon$$

reparametrize

$$\epsilon_\theta = \frac{a_t}{a_t b_t' - a_t' b_t}\left(v_\theta - \frac{a_t'}{a_t} z_t\right)$$

$$\mathcal{L} = \mathbb{E}\left[w_t^2 \|\epsilon_\theta(z_t, t) - \epsilon\|^2\right]$$

$$v_\theta = \frac{a_t'}{a_t} z_t + \frac{a_t b_t' - a_t' b_t}{a_t} \epsilon_\theta$$

"Scaling Rectified Flow Transformers for High-Resolution Image Synthesis" Esser, et al., ICML 2024

DDPM vs Flow Matching

Same goal, different formulation

Model	DDPM	Flow Matching
Trajectory	$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{(1-\bar{\alpha}_t)}\varepsilon$	$x_t = t \cdot x_0 + (1-t)\varepsilon$
Schedule	$\beta_t, \alpha_t, \bar{\alpha}_t (complex)$	$None\ needed\ (t \in [0,1])$
Predict	$Noise\ \varepsilon_\theta(x_t, t)$	$Velocity\ v_\theta(x_t, t)$
Loss	$\|\|\varepsilon - \varepsilon_\theta\|\|^2$	$\|\|v - v_\theta\|\|^2$
Path shape	$Curved\ (SDE)$	$Linear\ (Conditional)$
Steps	$\sim 50\ (DDIM)$	~ 30
Theory	$ELBO \rightarrow L_{simple}$	$CFM\ (conditional\ FM)$
Used by	SD 1.x, 2.x, DALL − E	SD3, FLUX, Sora, VEO

**cf. Notation: here $x_0 =$ data, $\varepsilon =$ noise (matching DDPM convention). Most FM papers use the opposite: x_0 = noise, x_1 = data.*

Key Takeaway "DDPM and FM differ only in reparameterization and loss reweighting" Kaiming He, MIT 6.7960

Flow Matching in Practice

The backbone of modern generative AI

STABLE DIFFUSION 3

Esser et al., 2024

Rectified Flow (= FM + straight paths)
DiT backbone (MM-DiT)
Dual text encoders
(CLIP + T5)

"Scaling Rectified
Flow Transformers
for High-Resolution
Image Synthesis"

FLUX

Black Forest Labs, 2024

Flow Matching + DiT
Guidance distilled versions:
- FLUX.1-dev (distilled)
- FLUX.1-schnell (4 steps!)

12B parameters
Current SOTA for
open-source T2I

SORA / VEO

OpenAI / Google, 2024-25

Flow Matching + DiT
extended to video

Spatiotemporal patches

Same principles:
learn velocity field,
Transformer backbone,
scale with compute

What Should the Network Predict?

Three parameterizations of the same denoising process

x_0-PREDICTION

Predict the clean image x_0

Also explored in DDPM

Network output: $\hat{x}_0(x_t, t)$

Loss: $||x_0 - \hat{x}_0||^2$

Previously considered equivalent (just reparameterization)

→ But NOT equivalent in practice

More natural target Key to JiT (next slide)

ε-PREDICTION

Predict the noise ε

DDPM (Ho et al., 2020)

Default choice

$$x_t = \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{(1-\bar{\alpha}_t)} \cdot \varepsilon$$

Network output: $\varepsilon_\theta(x_t, t)$

Loss: $||\varepsilon - \varepsilon_\theta||^2$

Works well in latent space

Standard for SD 1.x, 2.x

v-PREDICTION

Predict the velocity v

Salimans & Ho, 2022

$$v = \sqrt{\bar{\alpha}_t} \cdot \varepsilon - \sqrt{(1-\bar{\alpha}_t)} \cdot x_0$$

Network output: $v_\theta(x_t, t)$

Loss: $||v - v_\theta||^2$

Numerically more stable

SD 2.x, SDXL default, Flow Matching

	(a) x-pred $\boldsymbol{x}_\theta := \mathrm{net}_\theta(\boldsymbol{z}_t, t)$	(b) ϵ-pred $\boldsymbol{\epsilon}_\theta := \mathrm{net}_\theta(\boldsymbol{z}_t, t)$	(c) v-pred $\boldsymbol{v}_\theta := \mathrm{net}_\theta(\boldsymbol{z}_t, t)$
(1) x-loss: $\mathbb{E}\|\boldsymbol{x}_\theta - \boldsymbol{x}\|^2$	$\boldsymbol{x}_\theta$	$\boldsymbol{x}_\theta = (\boldsymbol{z}_t - (1-t)\boldsymbol{\epsilon}_\theta)/t$	$\boldsymbol{x}_\theta = (1-t)\boldsymbol{v}_\theta + \boldsymbol{z}_t$
(2) ϵ-loss: $\mathbb{E}\|\boldsymbol{\epsilon}_\theta - \boldsymbol{\epsilon}\|^2$	$\boldsymbol{\epsilon}_\theta = (\boldsymbol{z}_t - t\boldsymbol{x}_\theta)/(1-t)$	$\boldsymbol{\epsilon}_\theta$	$\boldsymbol{\epsilon}_\theta = \boldsymbol{z}_t - t\boldsymbol{v}_\theta$
(3) v-loss: $\mathbb{E}\|\boldsymbol{v}_\theta - \boldsymbol{v}\|^2$	$\boldsymbol{v}_\theta = (\boldsymbol{x}_\theta - \boldsymbol{z}_t)/(1-t)$	$\boldsymbol{v}_\theta = (\boldsymbol{z}_t - \boldsymbol{\epsilon}_\theta)/t$	$\boldsymbol{v}_\theta$

This v is defined under the DDPM schedule ($\bar{\alpha}_t$).
In FM, velocity $v = x_0 - \varepsilon$ (constant).
They are related by reparameterization (slide 18)

Source: Li & He 2025

JiT: Back to Pixel Space

Li & He, 2025 — Just image Transformers

The Problem

In HIGH-dimensional pixel space

Per-patch dim in 256 image: 16×16×3 = 768 ≥ hidden size (768)

Per-patch dim in 512 image: 32×32×3 = 3072 ≫ hidden size (768)

cf. Latent diffusion: per-token dim = 16 ≪ hidden size → no issue

ε-prediction: catastrophic failure

v-prediction: catastrophic failure

x_0-prediction: works!

Why?

ε lives in full D-dimensional space

x_0 lives on a low-dim manifold

As D grows, predicting full-space targets becomes impossible without strong inductive bias (U-Net)

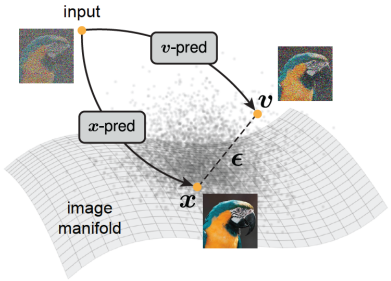


Figure 1. **The Manifold Assumption** [4] hypothesizes that natural images lie on a low-dimensional manifold within the high-dimensional pixel space. While a clean image $\boldsymbol{x}$ can be modeled as on-manifold, the noise $\boldsymbol{\epsilon}$ or flow velocity $\boldsymbol{v}$ (*e.g.*, $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{\epsilon}$) is inherently off-manifold. Training a neural network to predict a clean image (*i.e.*, $\boldsymbol{x}$-prediction) is *fundamentally different* from training it to predict noise or a noised quantity (*i.e.*, $\boldsymbol{\epsilon}$/$\boldsymbol{v}$-prediction).

JiT Architecture

Plain ViT on raw pixels

No VAE, no U-Net, no perceptual loss

Network predicts: x_0 (on-manifold target)

Loss computed in: v-space (better weighting) within flow matching framework

Patch size: 16 or 32

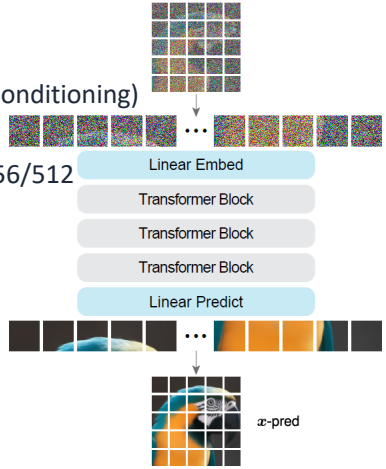
Class tokens replicated 32× (in-context conditioning)

Results: competitive FID on ImageNet 256/512

Implication:

VAE might not be necessary if you predict correctly

Caveat: no text-to-image yet



The Big Picture

Three perspectives, one framework (Lai, Song, Kim, Mitsufuji & Ermon, 2025)

Variational VAE (2014) → DDPM (2020) → Improved DDPM (2021)

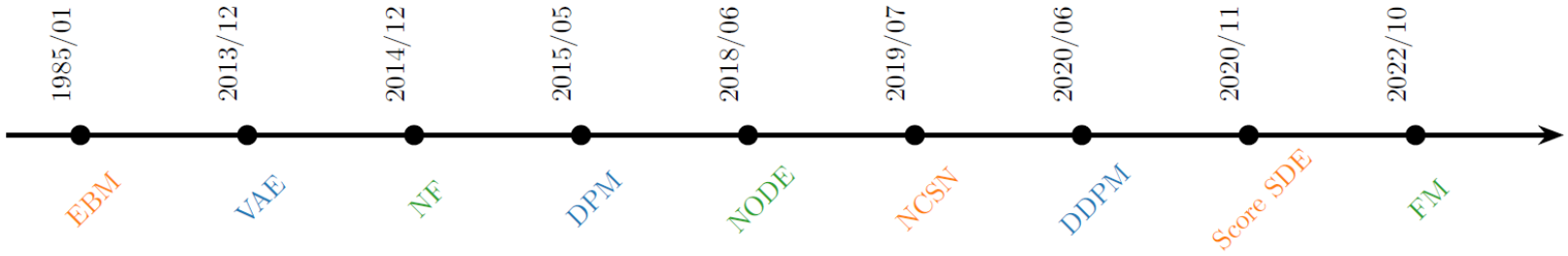
Learn to denoise step by step. ELBO → L_{simple}.

Score-Based Energy-Based Model → Noise Conditional Score Network (2019) → Score SDE (2021)

Learn $\nabla log\ p(x)$. Unifies DDPM and NCSN into SDE/ODE framework.

Flow-Based NF (2015) → CNF/Neural ODE (2018) → Flow Matching (2023)

Learn velocity field directly. Simulation-free, straight paths.



Part 2

LDM, CFG & Distillation

Making diffusion more pracical

Latent Diffusion Models (LDM)

Rombach et al., CVPR 2022

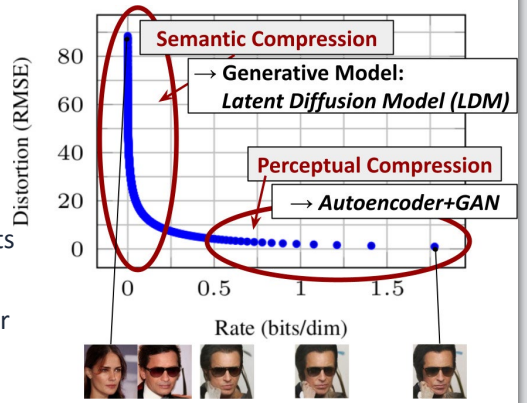
Key Idea

Problem: diffusion in pixel space is very expensive
256×256×3 = 196K dimensions

Solution: work in latent space
256×256 → 32×32 (f=8)
48× fewer dimensions!

VAE (Wk 10 Part 2) encodes image to latent z
→ diffusion operates on z
→ VAE decoder reconstructs

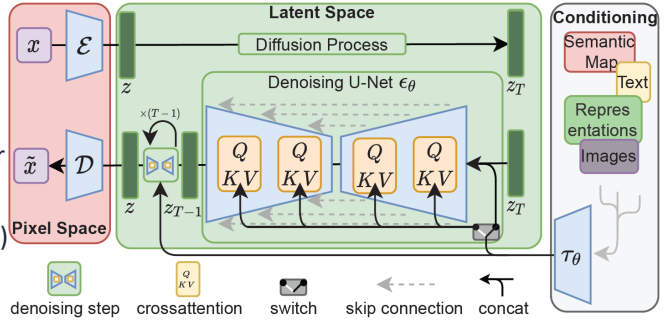
Same DDPM/FM, just smaller



Architecture

Three components:

1. VAE Encoder/Decoder
   Pretrained, frozen
   (VQ-GAN with adv loss)



2. Denoising Network
   U-Net (SD1/2) or DiT (SD3)
   Operates in latent space

$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right]$$

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t)\|_2^2\right]$$

3. Conditioning
   Text: CLIP/T5 → cross-attention
   Class: embedding → adaLN
   Image: various adapters
   (Part 4: ControlNet, IP-Adapter)

Classifier-Free Guidance (CFG)

Ho & Salimans, 2022

CFG formulation: CFG approximates the classifier gradient without a classifier

$$\boldsymbol{\varepsilon_{guided}} = \boldsymbol{\varepsilon_\theta}(\boldsymbol{x_t}, \boldsymbol{t}, \emptyset) + \boldsymbol{w} \cdot \left(\boldsymbol{\varepsilon_\theta}(\boldsymbol{x_t}, \boldsymbol{t}, \boldsymbol{y}) - \boldsymbol{\varepsilon_\theta}(\boldsymbol{x_t}, \boldsymbol{t}, \emptyset)\right)$$

unconditional baseline + w × (condition's pure contribution)

Training: randomly drop condition ($y \rightarrow \emptyset$) with 10-20% probability. One model learns both conditional and unconditional.

Inference: two forward passes per step (conditional + unconditional). w controls quality–diversity tradeoff.

Key insight: unconditional prediction is the baseline. The difference (conditional – unconditional) isolates the condition's contribution. Amplify by w.

Cost: 2× forward passes per step. Solution? → Next slide: Guidance Distillation

Guidance Distillation

Integrate CFG into the model → single forward pass

PROBLEM

CFG requires 2 forward passes per step:

1. $\varepsilon_\theta(x_t, t, y)$ conditional
2. $\varepsilon_\theta(x_t, t, \emptyset)$ unconditional

2× compute cost

For large models (12B+) this is very expensive

SOLUTION

Teacher: full CFG model (2 passes, high quality)

Student: single-pass model learns to match teacher's guided output directly

$$L = \left|\left|student(x_t, t, y) - teacher\mathrm{CFG}(x_t, t, y)\right|\right|^2$$

After distillation:
1 pass = CFG-quality output

FLUX LINEUP

FLUX.1
Full model, CFG, many steps

FLUX.1-dev
Guidance distilled
1 pass, ~30 steps

FLUX.1-schnell
+ Step distillation
1 pass, 4 steps!
~0.5s per image

Same architecture, different distillation levels

FLUX.2 (25.11) has many more variants…

Part 3

Controllable Generation

From WHAT to generate to HOW to generate

Controllable Generation: Taxonomy

How to add spatial/structural/reference control to diffusion models

Text Conditioning

Built into LDM

CLIP / T5 text encoder
→ cross-attention with
denoising network

"a cat sitting on a red couch"

Coarse semantic control
but no spatial precision

SD, DALL-E, FLUX default

Spatial Conditioning

ControlNet
(Zhang et al., 2023)

Structure → Image:
Edge, depth, pose,
segmentation map

Copy encoder weights
+ zero-conv injection

T2I-Adapter:
Lightweight alternative
~77M vs ~360M params

Reference Conditioning

IP-Adapter (Ye et al., 2023)
BrushNet (Ju et al., 2024)

Image → Image:
Style, subject, appearance

Decoupled cross-attention:
text features → text CA
image features → image CA

ReferenceNet:
Full reference encoder

Unified Approaches

One model, many tasks

OminiControl (2024):
DiT-native, single adapter
for multiple modalities

UniCon (Paper #3, ICLR 2025):
Joint distribution over
(image, condition) pairs
→ different inference
schemes for different tasks

ControlNet

Zhang & Agrawala, ICCV 2023

Architecture

Frozen pretrained SD model + Trainable copy of encoder blocks

Input: condition image (edge, depth, pose, seg map)

Zero-convolution:

output initialized to zero → training starts from

pretrained model behavior → gradual learning of control

Injection: add features to decoder skip connections

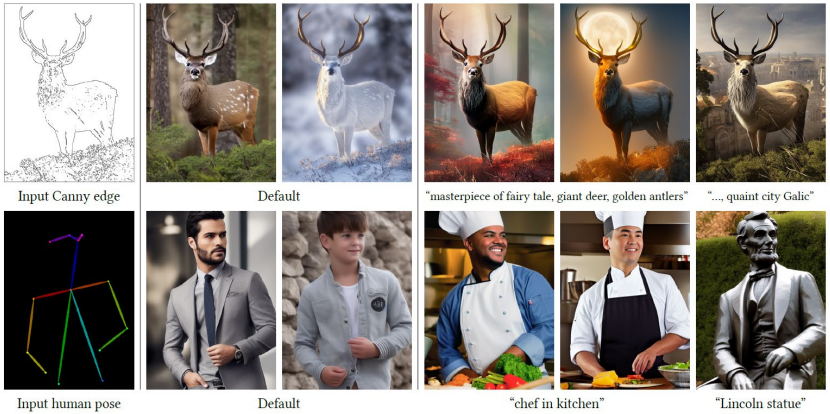


Figure 1: Controlling Stable Diffusion with learned conditions. ControlNet allows users to add conditions like Canny edges (top), human pose (bottom), *etc.*, to control the image generation of large pretrained diffusion models. The default results use the prompt "a high-quality, detailed, and professional image". Users can optionally give prompts like the "chef in kitchen".

Why It Works

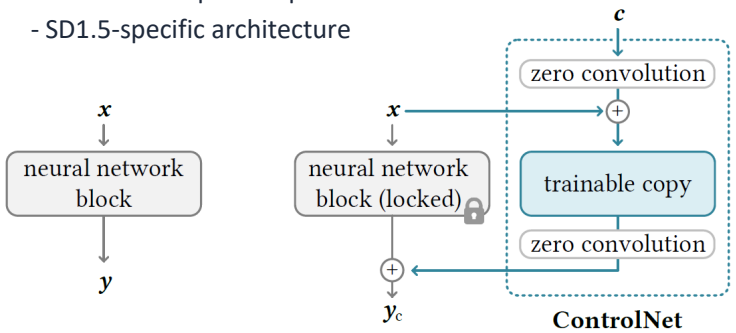
1. Lock the pretrained model → preserve generation quality
2. Zero-init the injection → no harmful noise at start
3. Train on (condition, image) pairs → learn the mapping

*Result:

- Precise spatial control
- Works with any SD checkpoint
- Community-trained 100+ variants

*Limitation:

- ~360M extra params per condition
- SD1.5-specific architecture



IP-Adapter & Beyond

Image-based conditioning through decoupled attention

IP-Adapter (Ye et al., 2023)

Problem: text can't describe everything (style, texture, face)

Solution: use a reference image

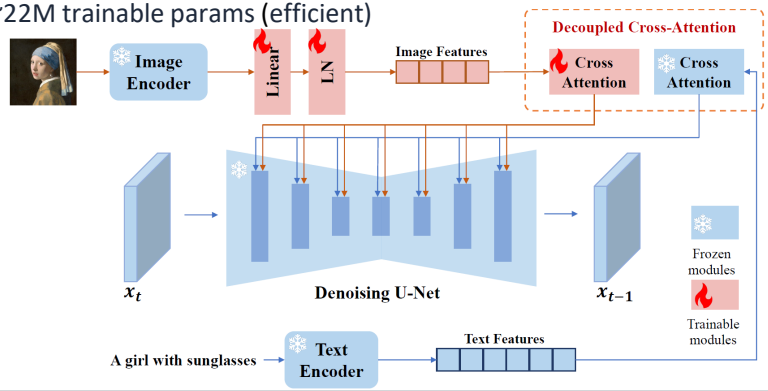
Key innovation:

Decoupled cross-attention

- Text tokens → original CA
- Image tokens → new CA layer

Image encoder: pretrained CLIP image encoder

Only ~22M trainable params (efficient)



Broader Landscape

T2I-Adapter (Mou et al., 2023)

Lightweight spatial control; Adapter modules only

ControlNeXt (2024)

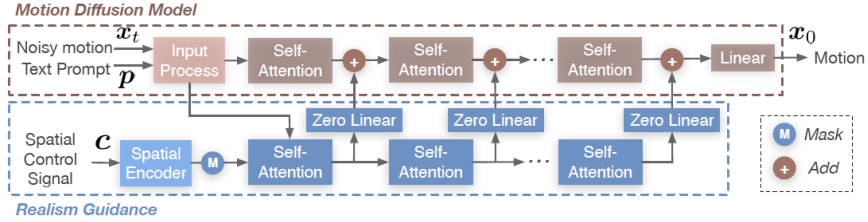
Efficient for video generation; fewer params than ControlNet

Ctrl-Adapter (2024)

Transfer control across models; Plug-and-play

DiT-native approaches:

OminiControl (2024): single adapter for DiT (SD3, FLUX)



Toward Unification: UniCon

Paper #3: A Simple Approach to Unifying Diffusion-based Conditional Generation

The problem with specialized approaches (ControlNet, IP-Adapter, T2I-Adapter):

- Each needs separate training for each condition type
- Architecture-specific ($SD1.5 \neq SDXL \neq SD3$)
- Combining multiple conditions is non-trivial

UniCon's insight: learn the JOINT distribution p(image, condition)

- Single training stage, minimal extra parameters (15% of base model)
- Different inference-time sampling → different tasks:
  depth → image (controllable generation)
  image → depth (estimation)
  (image, depth) jointly (joint generation)
  coarse conditioning (flexible control)

Summary & Resources

What We Covered

Part 1: Score & Flow Matching
- ε prediction = score estimation
- FM: linear interpolation, velocity field, straight paths
- SD3, FLUX, Sora are all FM-based
- $\varepsilon/x_0/v$-prediction: same math, different dynamics
- JiT: pixel-space x-prediction, no VAE needed

Part 2: LDM + CFG + Distillation
- LDM: diffusion in latent space (48× smaller)
- CFG: unconditional baseline + amplified condition
- Guidance distillation: bake CFG into 1 pass

Part 3: Controllable Generation
- Text → Spatial → Reference → Unified
- ControlNet, IP-Adapter, UniCon (Paper #3)

Further Reading

Flow Matching: Lipman et al., ICLR 2023
LDM: Rombach et al., CVPR 2022
CFG: Ho & Salimans, 2022
SD3: Esser et al., 2024
JiT: Li & He, 2025
The Principles of Diffusion Models, Lai et al., 2025

ControlNet: Zhang & Agrawala, ICCV 2023
IP-Adapter: Ye et al., 2023
UniCon: Li et al., ICLR 2025

MIT 6.S184 Course Notes (2026) Ch.3-6
Kaiming He, MIT 6.7960 Lec 16
Kaiming He, CVPR 2025 MeanFlow Talk

Next Week Week 12: Vision Language Model + Paper #4 Seminar

Acknowledgments

Some slides and figures in this course are adapted from or inspired by the following open resources.

- **Stanford CS231n: Deep Learning for Computer Vision (Spring 2025)**
  Fei-Fei Li, Ehsan Adeli, et al. | cs231n.stanford.edu
- **MIT 6.S184: Generative AI with Stochastic Differential Equations (2026)**
  Peter Holderrieth and Ezra Erives | https://diffusion.csail.mit.edu/
- **MIT 6.7960: Deep Learning (Fall 2024, Fall 2025)**
  Phillip Isola, Sara Beery, Jeremy Bernstein | ocw.mit.edu/courses/6-7960-deep-learning-fall-2024/, https://deeplearning6-7960.github.io/
- **3rd Workshop on Generative Models for Computer Vision (CVPR 2025)**
  Kaiming He | https://generative-vision.github.io/workshop-CVPR-25/
- **Key papers: Flow Matching, LDM, CFG, SD3, Jit, ControlNet, IP-Adapter, UniCon, …**