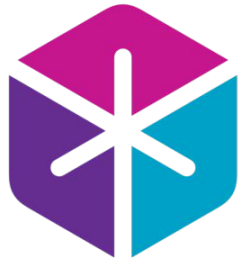
Flow and Diffusion Models - 2

Advances in Computer Vision

MIT Spring 2025 | April 03 2025

Peter Holderrieth



Overview - Today + Next Tuesday

1. **From Generation to Sampling**: Formalize “generating an image/etc.”
2. **Flow and diffusion models as generative models:**
   a. *Flows*: Sampling based on ODEs
   b. *Diffusion*: Sampling based on SDEs
3. **Training algorithms - 1**: Flow Matching — Last class

---

4. **Training algorithms - 2**: Score Matching — Today
5. **Guidance**: How to condition on a prompt
6. **Literature overview**: Various diffusion interpretations
7. **Neural network architectures**
8. **Case studies of large-scale models**

Reminder: Lecture Notes

For your reference, there are lecture notes:
https://diffusion.csail.mit.edu/docs/lecture-notes.pdf

Note that these are bit *more in-depth than required* for this class.

Reminder: Conditional Prob. Path and Cond. Vector Field

	Notation	Key property	Gaussian example
Conditional Probability Path	$p_t(\cdot \mid z)$	Interpolates p_{init} and a data point z	$\mathcal{N}(\alpha_t z, \beta_t^2 I_d)$
Conditional Vector Field	$u_t^{\text{target}}(x \mid z)$	ODE follows conditional path	$\left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t}\alpha_t\right) z + \frac{\dot{\beta}_t}{\beta_t} x$

Reminder: Marginal Prob. Path and Marginal Vector Field

	Notation	Key property	Formula
Marginal Probability Path	p_t	Interpolates p_{init} and p_{data}	$\int p_t(x\|z)p_{\text{data}}(z)\mathrm{d}z$
Marginal Vector Field	$u_t^{\text{target}}(x)$	ODE follows marginal path	$\int u_t^{\text{target}}(x\|z)\frac{p_t(x\|z)p_{\text{data}}(z)}{p_t(x)}\mathrm{d}z$

Algorithm 3 Flow Matching Training Procedure (General)

Require: A dataset of samples $z \sim p_{\text{data}}$, neural network u_t^θ

1: **for** each mini-batch of data **do**
2: Sample a data example z from the dataset.
3: Sample a random time $t \sim \text{Unif}_{[0,1]}$.
4: Sample $x \sim p_t(\cdot|z)$
5: Compute loss

$$\mathcal{L}(\theta) = \|u_t^\theta(x) - u_t^{\text{target}}(x|z)\|^2$$

6: Update the model parameters θ via gradient descent on $\mathcal{L}(\theta)$
7: **end for**

We can learn the marginal vector field by approximating the cond. VF for many different data points z.

Reminder: Sampling Algorithm for Flow Model

Algorithm 1 Sampling from a Flow Model with Euler method

Require: Neural network vector field u_t^θ, number of steps n

1: Set $t = 0$
2: Set step size $h = \frac{1}{n}$
3: Draw a sample $X_0 \sim p_{\text{init}}$ *Random initialization!*
4: **for** $i = 1, \ldots, n-1$ **do**
5: $\quad X_{t+h} = X_t + h u_t^\theta(X_t)$
6: $\quad$ Update $t \leftarrow t + h$
7: **end for**
8: **return** X_1 *Return final point*

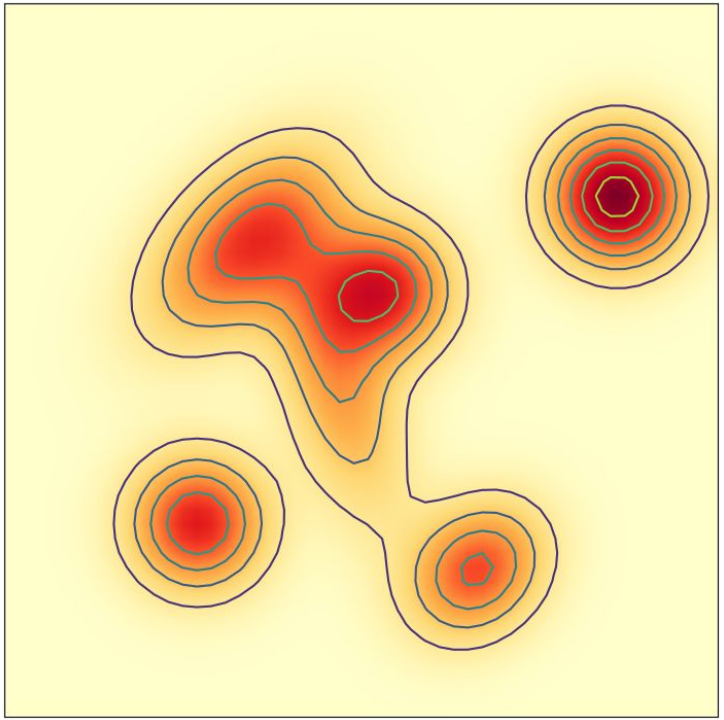
Section 4:

Training algorithms - 2: Score Matching

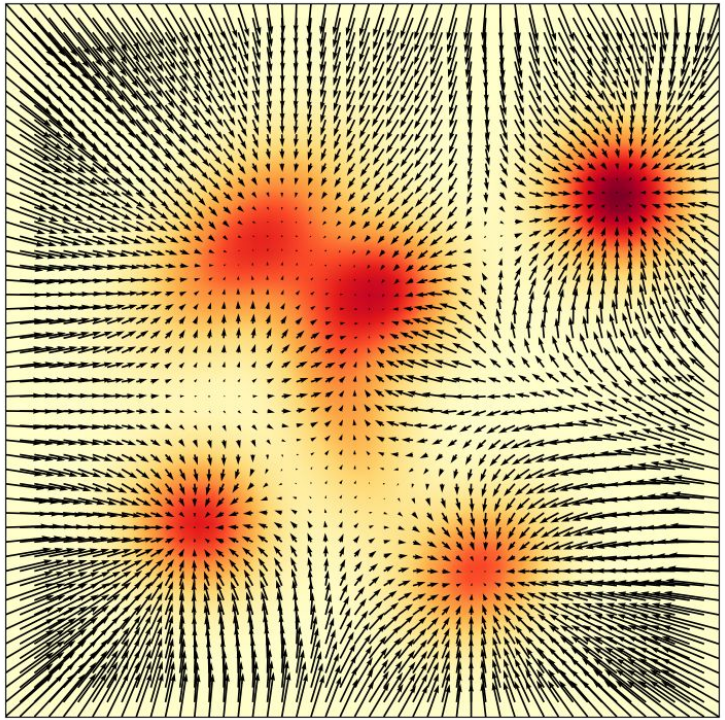
Goal: Different formulation for training flow and diffusion models.

Score Functions = Gradients of the log-likelihood

Log-likelihood: $\log q(x)$



Score function: $\nabla \log q(x)$



Conditional Prob. Path, Vector Field, and Score

	Notation	Key property	Gaussian example
Conditional Probability Path	$p_t(\cdot\|z)$	Interpolates p_{init} and a data point z	$\mathcal{N}(\alpha_t z, \beta_t^2 I_d)$
Conditional Vector Field	$u_t^{\text{target}}(x\|z)$	ODE follows conditional path	$\left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t}\alpha_t\right) z + \frac{\dot{\beta}_t}{\beta_t} x$
Conditional Score Function	$\nabla \log p_t(x\|z)$	Gradient of log-likelihood	$\frac{\alpha_t}{\beta_t^2} z - \frac{1}{\beta_t^2} x$

Marginal Prob. Path, Vector Field, and Score

	Notation	Key property	Formula
Marginal Probability Path	p_t	Interpolates p_{init} and p_{data}	$\int p_t(x\|z)p_{\text{data}}(z)\mathrm{d}z$
Marginal Vector Field	$u_t^{\text{target}}(x)$	ODE follows marginal path	$\int u_t^{\text{target}}(x\|z)\frac{p_t(x\|z)p_{\text{data}}(z)}{p_t(x)}\mathrm{d}z$
Marginal Score Function	$\nabla \log p_t(x)$	Can be used to convert ODE target to SDE	$\int \nabla \log p_t(x\|z)\frac{p_t(x\|z)p_{\text{data}}(z)}{p_t(x)}\mathrm{d}z$

Conversion Formulas

$$\tilde{A}_t = \left(\beta_t^2 \frac{\dot{\alpha}_t}{\alpha_t} - \dot{\beta}_t \beta_t \right), \quad \tilde{B}_t = \frac{\dot{\alpha}_t}{\alpha_t}$$

$$u_t^{\text{target}}(x|z) = \tilde{A}_t \nabla \log p_t(x|z) + \tilde{B}_t x$$

$$u_t^{\text{target}}(x) = \tilde{A}_t \nabla \log p_t(x) + \tilde{B}_t x$$

Proof: Algebra. Insert formulas.

Learning conditional + marginal score functions is equivalent to learning conditional and marginal vector field!

Algorithm 6 Score Matching Training Procedure (General)

Require: A dataset of samples $z \sim p_{\text{data}}$, score network s_t^θ

1: **for** each mini-batch of data **do**
2: Sample a data example z from the dataset.
3: Sample a random time $t \sim \text{Unif}_{[0,1]}$.
4: Sample $x \sim p_t(\cdot|z)$
5: Compute loss

$$\mathcal{L}(\theta) = \|s_t^\theta(x) - \nabla \log p_t(x|z)\|^2$$

6: Update the model parameters θ via gradient descent on $\mathcal{L}(\theta)$
7: **end for**

Denoising Score Matching for Gaussian Prob. Path

$$\nabla \log p_t(x|z) = -\frac{x - \alpha_t z}{\beta_t^2}$$

$$\epsilon \sim \mathcal{N}(0, I_d) \quad \Rightarrow \quad x = \alpha_t z + \beta_t \epsilon \sim \mathcal{N}(\alpha_t z, \beta_t^2 I_d)$$

$$\begin{aligned}
\mathcal{L}_{\text{dsm}}(\theta) =& \mathbb{E}_{t\sim\text{Unif}, z\sim p_{\text{data}}, x\sim p_t(\cdot|z)}[\| s_t^\theta(x) + \frac{x - \alpha_t z}{\beta_t^2} \|^2] \\
=& \mathbb{E}_{t\sim\text{Unif}, z\sim p_{\text{data}}, \epsilon\sim\mathcal{N}(0,I_d)}[\| s_t^\theta(\alpha_t z + \beta_t \epsilon) + \frac{\epsilon}{\beta_t} \|^2]
\end{aligned}$$

Algorithm 5 Score Matching Training Procedure for Gaussian probability path

Require: A dataset of samples $z \sim p_{\text{data}}$, score network s_t^θ or noise predictor ϵ_t^θ

Require: Schedulers α_t, β_t with $\alpha_0 = \beta_1 = 0, \alpha_1 = \beta_0 = 1$

1: **for** each mini-batch of data **do**

2: Sample a data example z from the dataset.

3: Sample a random time $t \sim \text{Unif}_{[0,1]}$.

4: Sample noise $\epsilon \sim \mathcal{N}(0, I_d)$

5: Set $x_t = \alpha_t z + \beta_t \epsilon$

6: Compute loss

$$\mathcal{L}(\theta) = \| s_t^\theta(x_t) + \frac{\epsilon}{\beta_t} \|^2$$

Numerically unstable for low beta!

7: Update the model parameters θ via gradient descent on $\mathcal{L}(\theta)$.

8: **end for**

Sampling with score models

$$\tilde{A}_t = \left(\beta_t^2 \frac{\dot{\alpha}_t}{\alpha_t} - \dot{\beta}_t \beta_t \right), \quad \tilde{B}_t = \frac{\dot{\alpha}_t}{\alpha_t}$$

$$u_t^{\text{target}}(x|z) = \tilde{A}_t \nabla \log p_t(x|z) + \tilde{B}_t x$$

$$u_t^{\text{target}}(x) = \tilde{A}_t \nabla \log p_t(x) + \tilde{B}_t x$$

Fokker-Planck equation

Randomly initialized SDE

Given: $X_0 \sim p_{\text{init}}, \quad \mathrm{d}X_t = u_t(X_t)\mathrm{d}t + \sigma_t \mathrm{d}W_t$

Follow probability path:

$$X_t \sim p_t \quad (0 \le t \le 1)$$

Marginals are p_t

equivalent

Fokker-Planck equation holds

$$\frac{\mathrm{d}}{\mathrm{d}t} p_t(x) = -\mathrm{div}(p_t u_t)(x) + \frac{\sigma_t^2}{2} \Delta p_t(x)$$

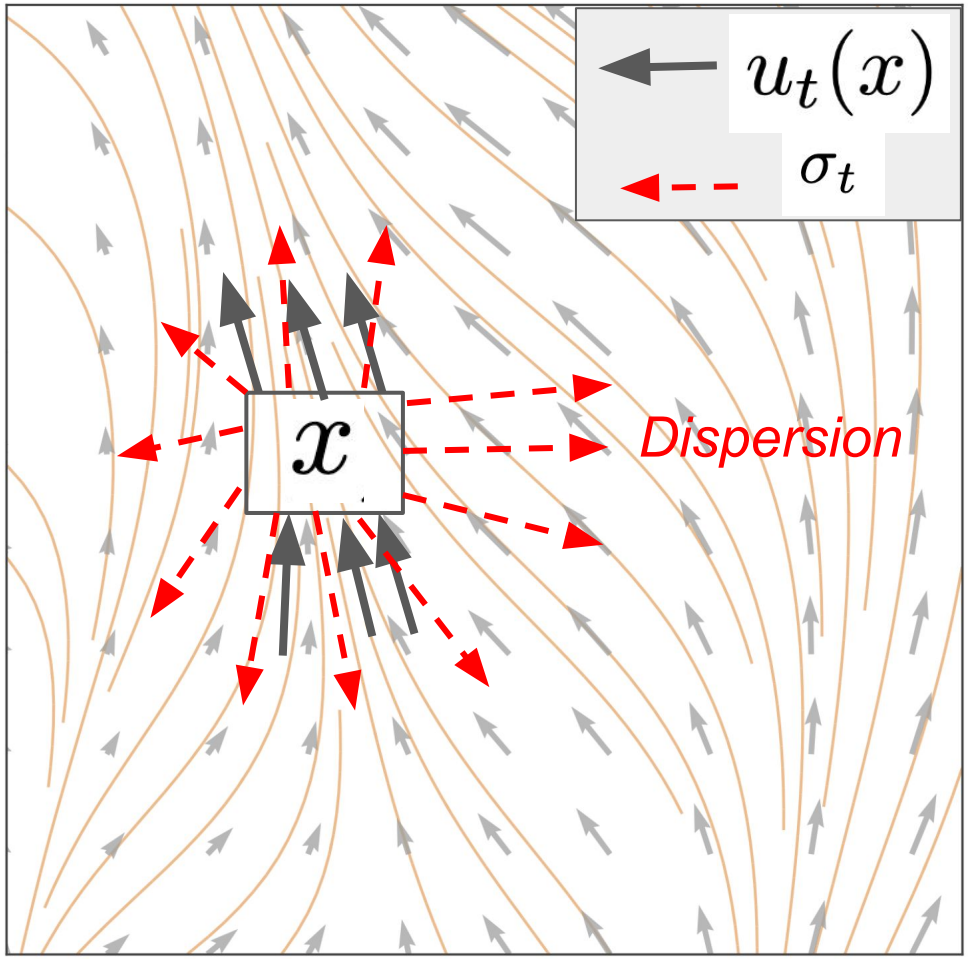
Continuity equ. *Heat equ.*

Fokker-Planck Equation

$$\frac{\mathrm{d}}{\mathrm{d}t}p_t(x) = -\mathrm{div}(p_t u_t)(x) + \frac{\sigma_t^2}{2}\Delta p_t(x)$$

Change of probability mass at x

Heat dispersion



Stochastic Sampling of diffusion models

Choose noise level σ_t. By “SDE extension trick”, we have to sample from

$$\mathrm{d}X_t = \left[(\tilde{A}_t + \frac{\sigma_t}{2})\nabla \log p_t(X_t) + \tilde{B}_t X_t\right]\mathrm{d}t + \sigma_t \mathrm{d}W_t$$

Replace marginal score network to approximately simulate this SDE:

$$\mathrm{d}X_t = \left[(\tilde{A}_t + \frac{\sigma_t}{2})s_t^\theta(X_t) + \tilde{B}_t X_t\right]\mathrm{d}t + \sigma_t \mathrm{d}W_t$$

Simulate SDE via:

$$X_{t+h} = X_t + h\left[(\tilde{A}_t + \frac{\sigma_t}{2})s_t^\theta(X_t) + \tilde{B}_t X_t\right] + \sqrt{h}\sigma_t \epsilon \quad (\epsilon \sim \mathcal{N}(0, I))$$

Key takeaway:

- **Conversion formula:** Learning the marginal vector field and learning the score function is equivalent for Gaussian probability paths.
- **Denoising score matching**: Simple way of learning marginal score functions by approximating conditional score functions.
- **Sampling with score models:**
  - *Probability flow ODE:* Flow matching ODE just reparameterized
  - *Stochastic sampling with SDEs*: Add desired amount of noise + apply correction to vector field

Section 6:

Classifier-free guidance

Goal: Understand how to enforce coherence to prompts

A swamp ogre with a pearl earring by Johannes Vermeer

A car made out of vegetables.

heat death of the universe, line art

Image source: Scaling Rectified Flow Transformers for High-Resolution Image Synthesis [1]

Unguided: “Generate an image.”

Guided: “Generate an image of a cat baking a cake.”

Vanilla Guided Sampling

Algorithm 7 Guided Sampling Procedure

Require: A trained guided vector field $u_t^\theta(x|y)$.

1: Select a prompt $y \in \mathcal{Y}$, such as "a cat baking a cake".
2: Initialize $X_0 \sim p_{\text{init}}$.
3: Simulate $\mathrm{d}X_t = u_t^\theta(X_t|y)\mathrm{d}t$ from $t = 0$ to $t = 1$.

Vanilla Guidance leads to suboptimal results

Image source: Classifier-free diffusion guidance [5].

Prompt: ***“Corgi dog”***

These images do not fit well to the prompt and they have errors!

Classifier-free guidance training: Account for empty token ∅

Algorithm 5 Classifier-free guidance training

Require: Paired dataset $(z, y) \sim p_{\text{data}}$, neural network u_t^θ

1: **for** each mini-batch of data **do**
2: Sample a data example (z, y) from the dataset.
3: Sample a random time $t \sim \text{Unif}_{[0,1]}$.
4: Sample noise $\epsilon \sim \mathcal{N}(0, I_d)$
5: Set $x = \alpha_t z + \beta_t \epsilon$
6: With probability p drop label: $y \leftarrow \varnothing$
7: Compute loss

Drop label with a certain probability!

$$\mathcal{L}(\theta) = \|u_t^\theta(x|y) - u_t^{\text{target}}(x|z)\|^2$$

8: Update the model parameters θ via gradient descent on $\mathcal{L}(\theta)$.
9: **end for**

Sampling with Classifier-Free Guidance simply is the same as before but we use the weighted vector field:

$$u_t^{\theta,w}(x) = (1-w)u_t^\theta(x|\varnothing) + wu_t^\theta(x|y)$$

Algorithm 8 Classifier-Free Guidance Sampling Procedure

Require: A trained guided vector field $u_t^\theta(x|y)$.

1: Select a prompt $y \in \mathcal{Y}$, or take $y = \varnothing$ for unguided sampling.
2: Select a **guidance scale** $w > 1$.
3: Initialize $X_0 \sim p_{\text{init}}$.
4: Simulate $\mathrm{d}X_t = \left[(1-w)u_t^\theta(X_t|\varnothing) + wu_t^\theta(X_t|y)\right]\mathrm{d}t$ from $t = 0$ to $t = 1$.

Example: Classifier-Free Guidance

Image source: Classifier-free diffusion guidance [5].

w=1.0

w=4.0

Section 5:

A guide to the diffusion literature

Goal: Understand different interpretations for diffusion models

Time conventions:

“Flow time convention”:

- Data: t = 1
- Noise: t = 0

Flow matching, rectified flows, stochastic interpolants

“Diffusion time convention”:

- Data t = 0
- Noise: t → infinity

Score-based diffusion models with SDEs

“Discrete time”:

- Use discrete time steps instead of continuous time steps
- No ODE or SDE but Markov chain
- DDIM ~= Probability flow ODE

DDPM
DDIM

Noising Procedure

Probability path (here):

$$p_t(x|z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d)$$

Flow matching, rectified flows

Interpolant function:

$$I_t(\epsilon, z) = \alpha_t \epsilon + \beta_t z$$

Stochastic interpolants

“Forward” diffusion process:

$$\mathrm{d}X_t = a_t(X_t)\mathrm{d}t + \sigma_t \mathrm{d}W_t$$

Denoising diffusion models

Note: For Gaussian probability paths, the above procedures are equivalent.

Constructing noising procedures via forward noising processes

Forward SDE (data → noise)

$\mathbf{x}(0)$ —— $\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$ ——→ $\mathbf{x}(T)$

score function

$\mathbf{x}(0)$ ←—— $\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$ —— $\mathbf{x}(T)$

Reverse SDE (noise → data)

The time-reversed SDE is a specific solution to the SDE extension trick that we discussed for a specific noise level. Empirically, this is often not the best solution in practice.

Here: Flow Matching and Stochastic Interpolants

- Arguably most simple flow and diffusion algorithms
- Allows you to restrict yourself to flows
- Allows you go from arbitrary p_{init} to arbitrary p_{data}

Note: The method presented here allows to convert arbitrary distributions into arbitrary distributions!

Bridging arbitrary distributions - Example

Videos without audio → videos with audio

Low resolution images → high resolution images

Unperturbed cells → perturbed cells

etc.

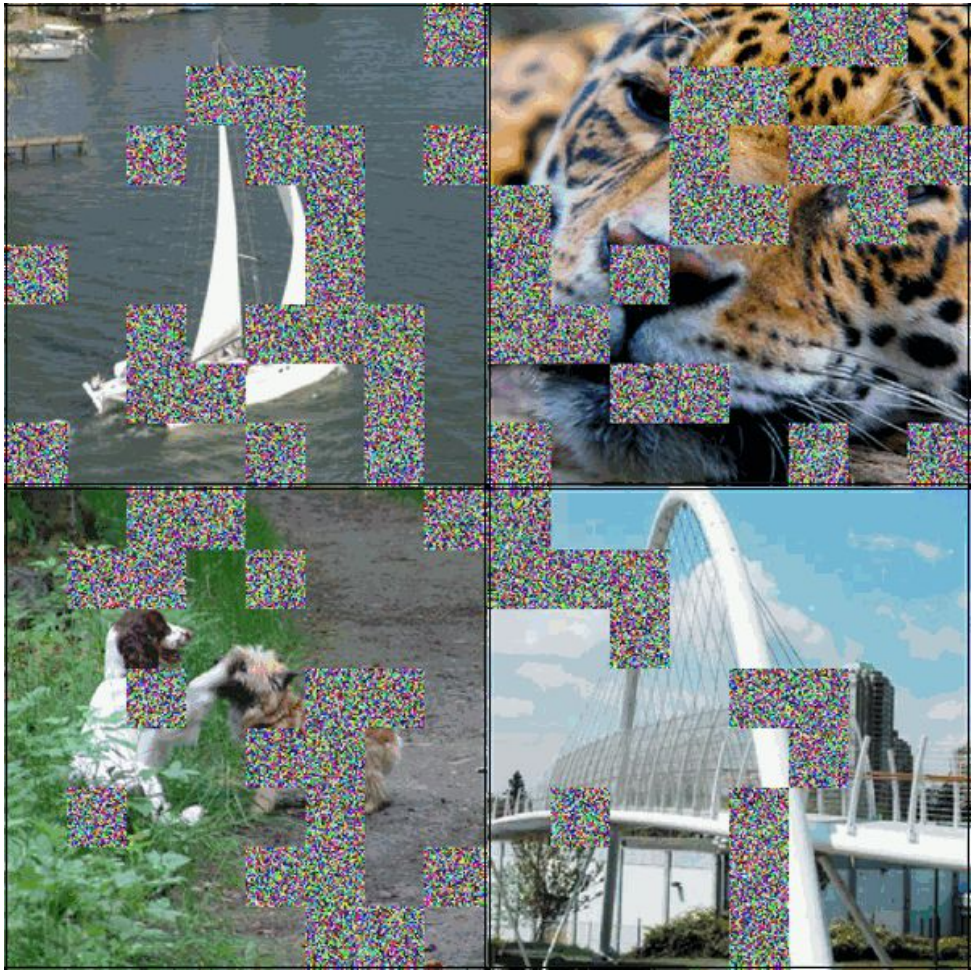


Figure credit: Michael Albergo

Section 7:

Neural network architectures

Goal: Understand different interpretations for diffusion models

Neural network architecture requirements:

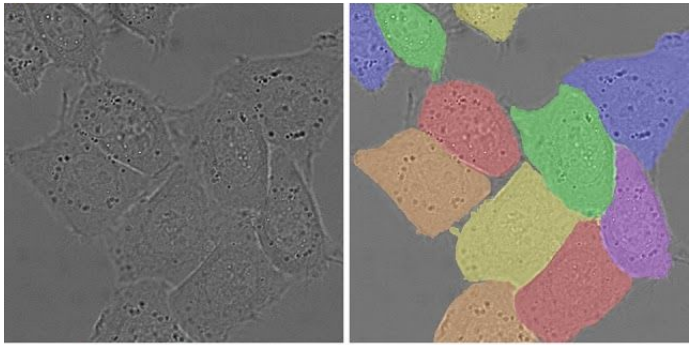
- **Neural network represents vector fields:** $u_t^\theta(x)$
- **3 inputs**: time t, noisy image x, and prompt y
- **Outputs**: Same shape as image x (i.e. we need to output a tensor that has the same shape as the image. E.g. A classification model won't work!)

Various inputs:

- **Time-dependence**: Simple conditioning on 1d variable
- **Prompt y:** Can be complicated language prompt
- **Image x**: Has specific grid shape + translation symmetry → use that topology!

U-Net: Convolutional Architecture

- Originally developed for image segmentation
- Fully convolutional architecture
- Encoder: Number of channels increases while width and height of image tensor decreases
- Decoder: Reverse shape of encoder
- Symmetry of encoder and decoder gives U-like diagram → “U-Net”
- Re-used for diffusion models by adding time-dependent convolutions



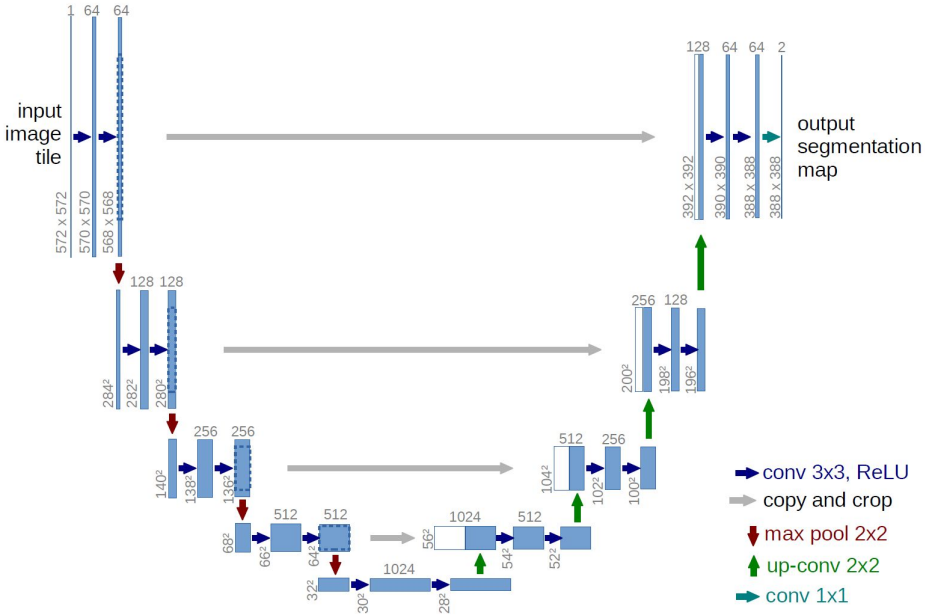
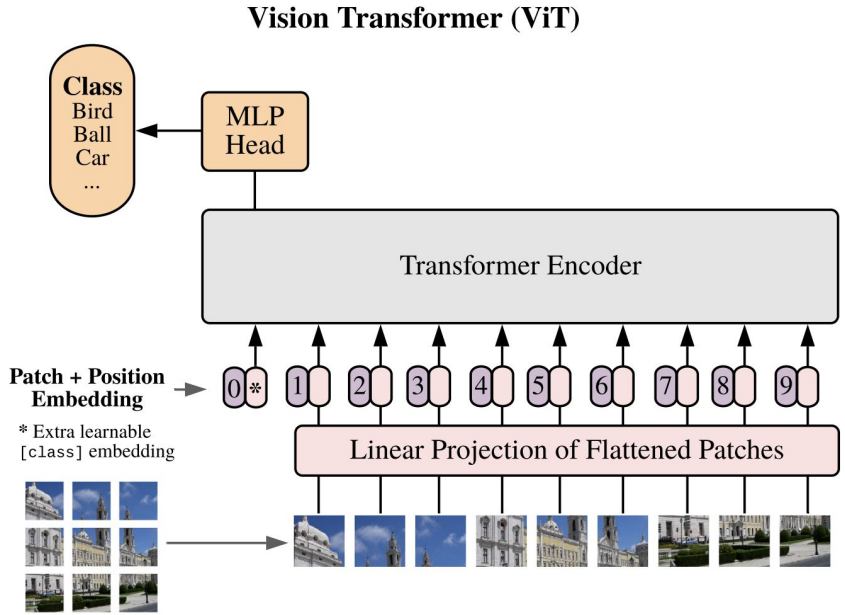


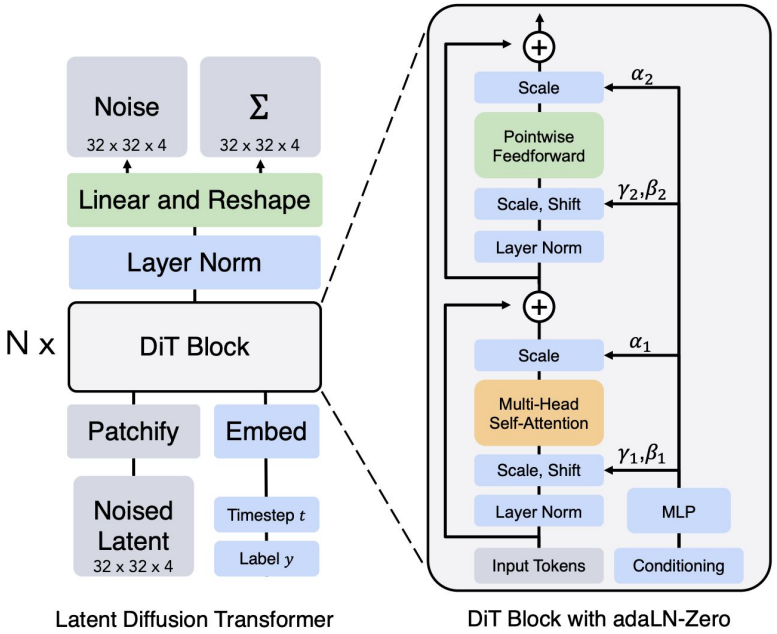
Image sources: Vision transformer paper [2] (left), diffusion transformer paper [3] (right).

Diffusion Transformer (DiT)

Idea: Divide an image into **patches** and **attend** between the patches.

Based on the **vision transformer** (ViT).





Latent Diffusion Transformer

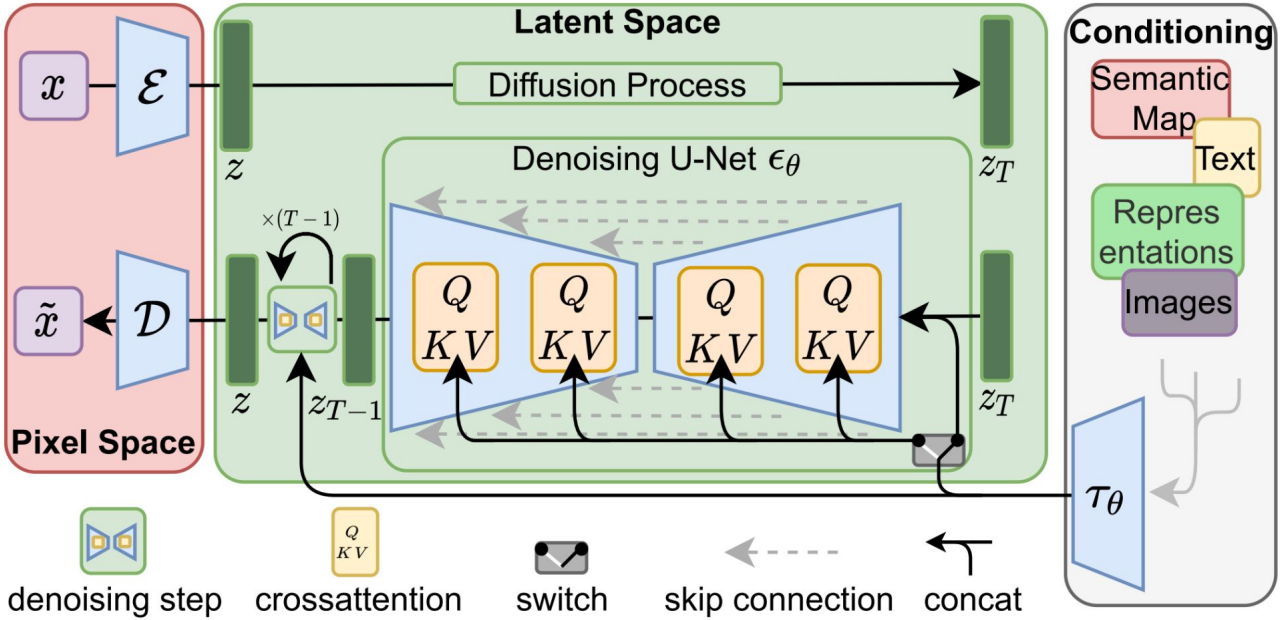
DiT Block with adaLN-Zero

Generative Modeling in Latent Space

Image source: .High Resolution Image Synthesis with Latent Diffusion Models [4]

Problem: Images have too high resolution (even worse for videos)

Solution: Train the generative model in the **latent space** of a pre-trained (variational) autoencoder.



Section 7:

Large-scale diffusion models

Goal: Understand how diffusion models are trained at scale

Case Study: Stable Diffusion 3

Image source: Scaling Rectified Flow Transformers for High-Resolution Image Synthesis [1]

Overview:

- Flow Matching model with “straight line” schedulers (CondOT path)
- Classifier-free guidance with weight 2.0 - 5.0
- Flow Matching in latent space (use pre-trained VAE)
- Number of parameters of model: 8 billion
- Number of simulation/sampling steps: 50
- Example dataset: LAION



Neural network architecture for Stable Diffusion 3

- Conditions on **CLIP** (coarse-grained) and **T5-XXL** (sequence-level) text embeddings via cross-attention.
- **MM-DiT architecture**: Extends DiT from class-conditioning to text-conditioning and processes text and images through the entire network via cross-attention

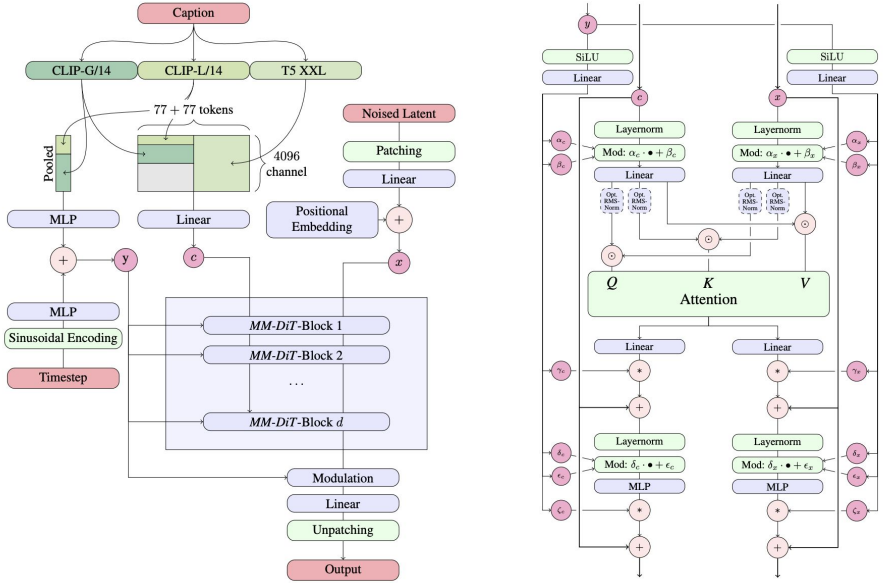


Image source: Scaling Rectified Flow Transformers for High-Resolution Image Synthesis [1]

Case Study - Meta MovieGen

Case Study: Meta MovieGen

Image source: Scaling Rectified Flow Transformers for High-Resolution Image Synthesis [1]

- Flow Matching model with “straight line” schedulers (CondOT path)
- Classifier-free guidance
- Flow Matching in latent space (use pre-trained VAE) - Really crucial for videos because of added time dimension
- Neural network architecture: DiT adapted to videos
- Number of parameters of model: 30 billion
- **6,144 H100 GPUs!**

Recap: Flow and diffusion models

1. **From Generation to Sampling**: Formalize “generating an image/etc.”
2. **Flow and diffusion models as generative models:**
   a. *Flows*: Sampling based on ODEs
   b. *Diffusion*: Sampling based on SDEs
3. **Training algorithms - 1**: Flow Matching
4. **Training algorithms - 2**: Score Matching
5. **Guidance**: How to condition on a prompt
6. **Neural network architectures**
7. **Large-scale SOTA diffusion models**