Lecture 3

Training Flow and Diffusion Models

MIT IAP 2025 | Jan 23, 2025

Peter Holderrieth and Ezra Erives

Sponsor: Tommi Jaakkola

Reminder: Flow and Diffusion Models

Flow Model

Initialize:

$$X_0 \sim p_{\text{init}},$$

ODE:

$$\mathrm{d}X_t = u_t^\theta(X_t)\mathrm{d}t$$

E.g. Gaussian

Neural network vector field

Diffusion coeff.

Diffusion Model

Initialize:

$$X_0 \sim p_{\text{init}},$$

SDE:

$$\mathrm{d}X_t = u_t^\theta(X_t)\mathrm{d}t + \sigma_t \mathrm{d}W_t$$

To get samples, simulate ODE/SDE from t=0 to t=1 and return X_1

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	$-\frac{x - \alpha_t z}{\beta_t^2}$

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	$\int \nabla \log p_t(x\|z)\frac{p_t(x\|z)p_{\text{data}}(z)}{p_t(x)}\mathrm{d}z$

Yesterday: Training target

Today: Training algorithm

Marginal Vector Field → *Flow Matching*

Marginal Score Function → *Score Matching*

Reminder - Marginal vector field:

$$u_t^{\text{target}}(x) = \int u_t^{\text{target}}(x|z) \frac{p_t(x|z) p_{\text{data}}(z)}{p_t(x)} \mathrm{d}z$$

Conditional vector field

Marginalization trick: The marginal vector field defines an ODE that "follows" the marginal probability path:

$$X_0 \sim p_{\text{init}}, \qquad \frac{\mathrm{d}}{\mathrm{d}t} X_t = u_t^{\text{target}}(X_t) \mathrm{d}t \qquad \Rightarrow X_t \sim p_t$$

This means that the final point fulfils: $X_1 \sim p_{\text{data}}$

Algorithm 3 Flow Matching Training Procedure (General)

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

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

  Sample a data example z from the dataset.

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

  Sample $x \sim p_t(\cdot|z)$

  Compute loss

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

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

end for

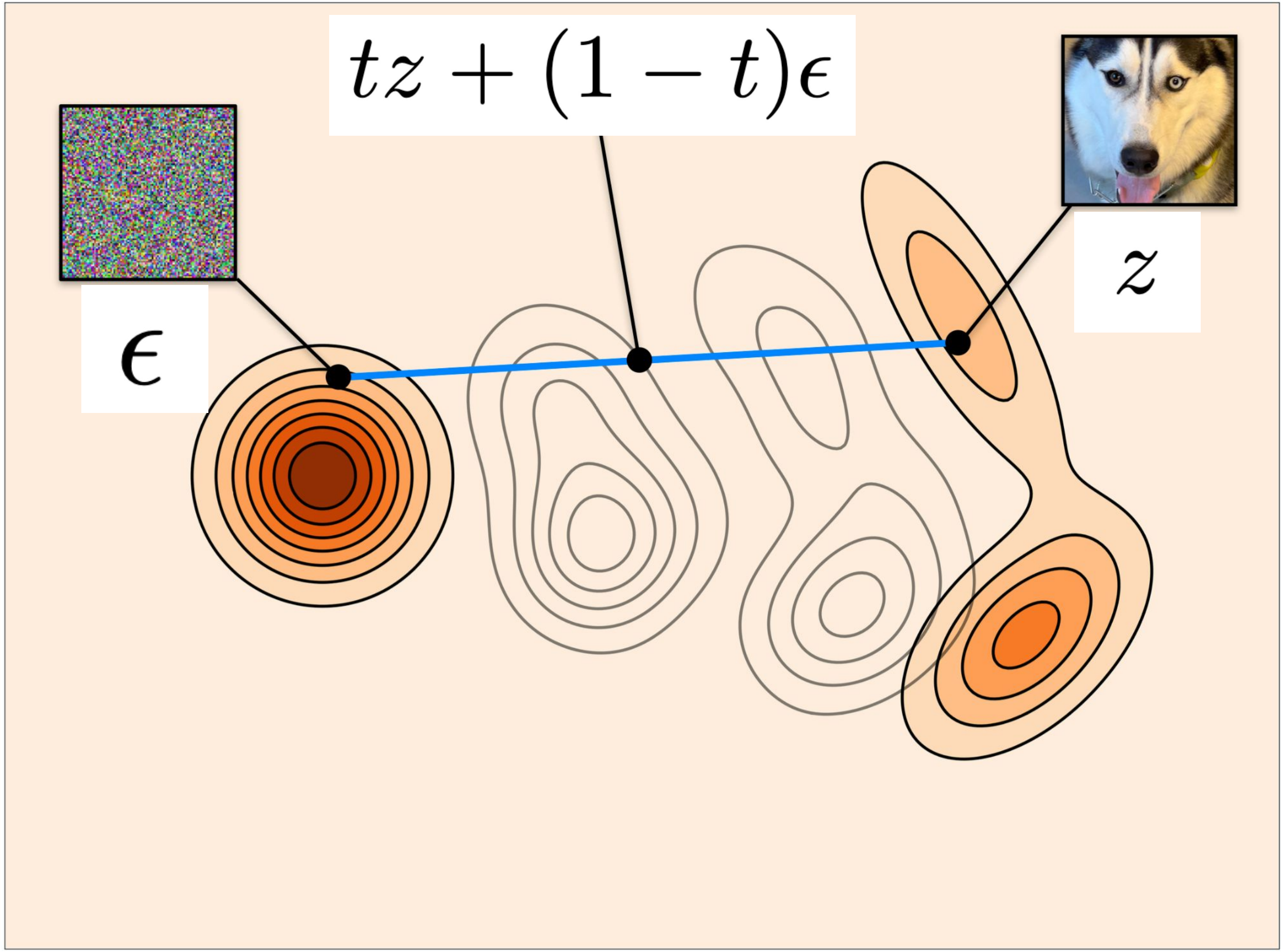


Figure credit: Yaron Lipman

Algorithm 4 Flow Matching Training for CondOT path

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

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

  Sample a data example z from the dataset.

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

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

  Set $x = tz + (1-t)\epsilon$

  Compute loss

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

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

end for

Example Flow Matching - Meta MovieGen

The neural network that generates these videos was trained with the algorithm in the previous slide

Example Flow Matching - Stable Diffusion 3

The neural network that generates these images was trained with the algorithm just shown

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

Set $t = 0$

Set step size $h = \frac{1}{n}$

Draw a sample $X_0 \sim p_{\text{init}}$ *Random initialization!*

for $i = 1, \ldots, n-1$ **do**

  $X_{t+h} = X_t + h u_t^\theta(X_t)$

  Update $t \leftarrow t + h$

end for

return X_1 *Return final point*

Reminder - Marginal score function:

$$\nabla \log p_t(x) = \int \nabla \log p_t(x|z) \frac{p_t(x|z) p_{\text{data}}(z)}{p_t(x)} \mathrm{d}z$$

Conditional score function

SDE extension trick: The marginal score function allows to extend the ODE to a SDE with arbitrary diffusion coefficient:

$$X_0 \sim p_{\text{init}}, \quad \mathrm{d}X_t = \left[u_t^{\text{target}}(X_t) + \frac{\sigma_t^2}{2} \nabla \log p_t(X_t) \right] \mathrm{d}t + \sigma_t \mathrm{d}W_t$$

$$\Rightarrow X_t \sim p_t$$

Algorithm 6 Score Matching Training Procedure (General)

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

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

    Sample a data example z from the dataset.

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

    Sample $x \sim p_t(\cdot|z)$

    Compute loss

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

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

end for

Denoising Score Matching for Gaussian Prob. Path

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

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

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

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 $\alpha_0 = \beta_1 = 0, \alpha_1 = \beta_0 = 1$

```
for each mini-batch of data do
    Sample a data example z from the dataset.
    Sample a random time t ~ Unif_[0,1].
    Sample noise ε ~ N(0, I_d)
    Set x_t = α_t z + β_t ε
    Compute loss
```

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

Numerically unstable for low beta

```
    Update the model parameters θ via gradient descent on L(θ).
end for
```

Stochastic sampling with diffusion models

$$X_0 \sim p_{\text{init}}, \quad \mathrm{d}X_t = \left[u_t^{\text{target}}(X_t) + \frac{\sigma_t^2}{2}\nabla \log p_t(X_t)\right]\mathrm{d}t + \sigma_t \mathrm{d}W_t$$

Plugin neural networks

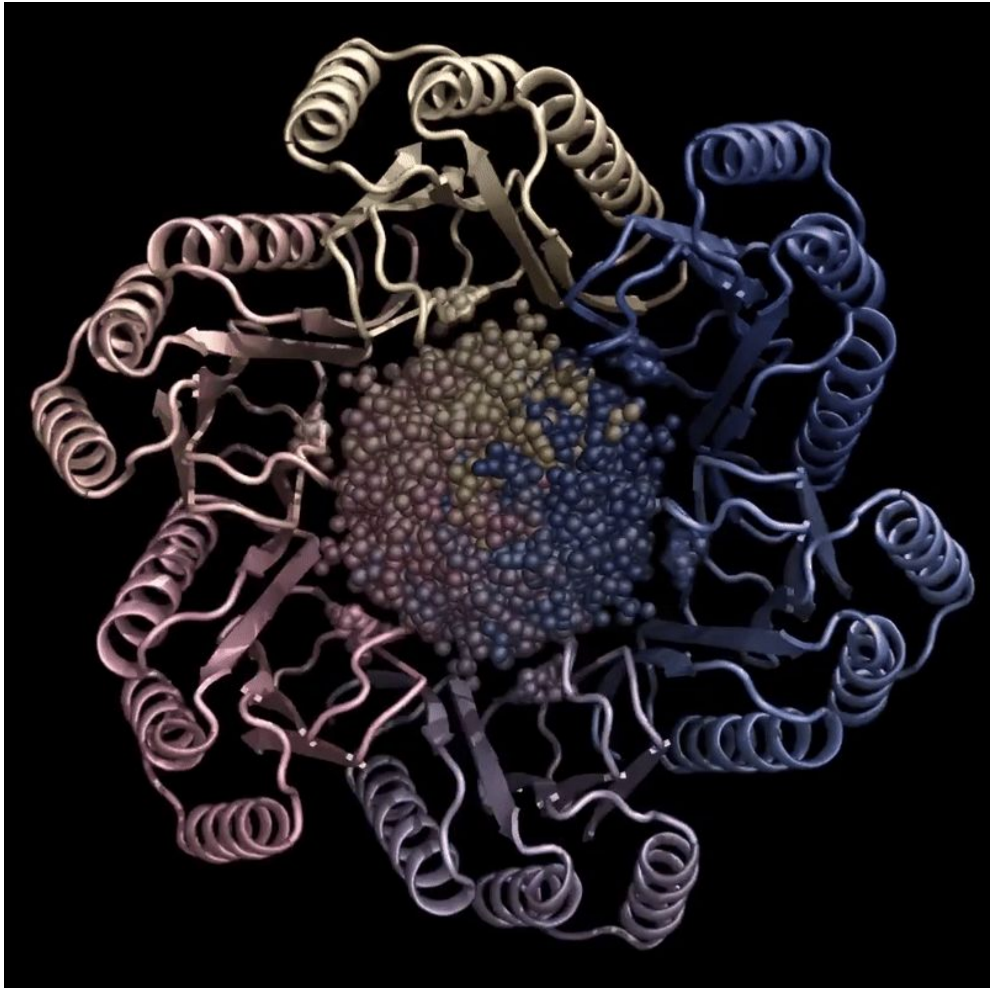
$$\Rightarrow X_t \sim p_t$$

$$X_0 \sim p_{\text{init}}, \quad \mathrm{d}X_t = \left[u_t^{\theta}(X_t) + \frac{\sigma_t^2}{2} s_t^{\theta}(X_t)\right]\mathrm{d}t + \sigma_t \mathrm{d}W_t$$

After training $\Rightarrow X_t \sim p_t$

Stochastic Sampling with Diffusion Models

Sampling a protein structure with stochastic sampling using the score function



Denoising Diffusion Models (DDMs)

Denoising diffusion models

= Diffusion models with a Gaussian probability path

(our standard example) $\mathcal{N}(\alpha_t z, \beta_t^2 I_d)$

Terminology (by many people)

Denoising diffusion model = diffusion model

i.e. many people drop the word "denoising" and just say "diffusion model".

Special property about DDMs: Score for free!

$$u_t^{\text{target}}(x|z) = \left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t}\alpha_t\right) z + \frac{\dot{\beta}_t}{\beta_t} x$$

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

Both are weighted averages of x and z → We can convert them each other!

$$u_t^{\text{target}}(x|z) = \left(\beta_t^2 \frac{\dot{\alpha}_t}{\alpha_t} - \dot{\beta}_t \beta_t\right) \nabla \log p_t(x|z) + \frac{\dot{\alpha}_t}{\alpha_t} x$$

$$u_t^{\text{target}}(x) = \left(\beta_t^2 \frac{\dot{\alpha}_t}{\alpha_t} - \dot{\beta}_t \beta_t\right) \nabla \log p_t(x) + \frac{\dot{\alpha}_t}{\alpha_t} x$$

You can show this by plugging in the equation and do some algebra

Conversion formula for DDMs

One can also convert the score network into vector field network post training:

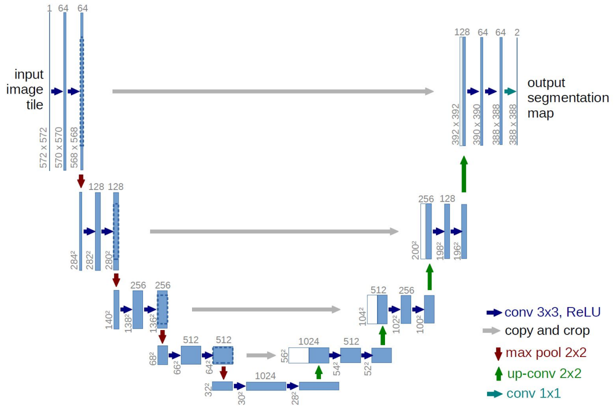
$$u_t^\theta = \left(\beta_t^2 \frac{\dot{\alpha}_t}{\alpha_t} - \dot{\beta}_t \beta_t\right) s_t^\theta(x) + \frac{\dot{\alpha}_t}{\alpha_t} x$$

In denoising diffusion models, we don't have to train the vector field and score network separately

→ they can be converted into one another post-training

The first generation of diffusion models only did score matching!

We arrived at a full end-to-end training and sampling algorithm! We have a generative model!



Next steps:

- Neural networking architectures
- Conditioning on a prompt
- Image generators
- Other applications:
  1. Robotics
  2. Protein design

Brief literature overview (see lecture notes for details)

- There is a whole zoo of different flow and diffusion model (Don't be confused!)
- First diffusion models: **Discrete time** (no ODE/SDEs)
- First SDE-based model: Forward-backward process using **time-reversal** of SDEs → construction of a probability path
- **Inverted time convention**: Data distribution at t=0

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: Our method 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.

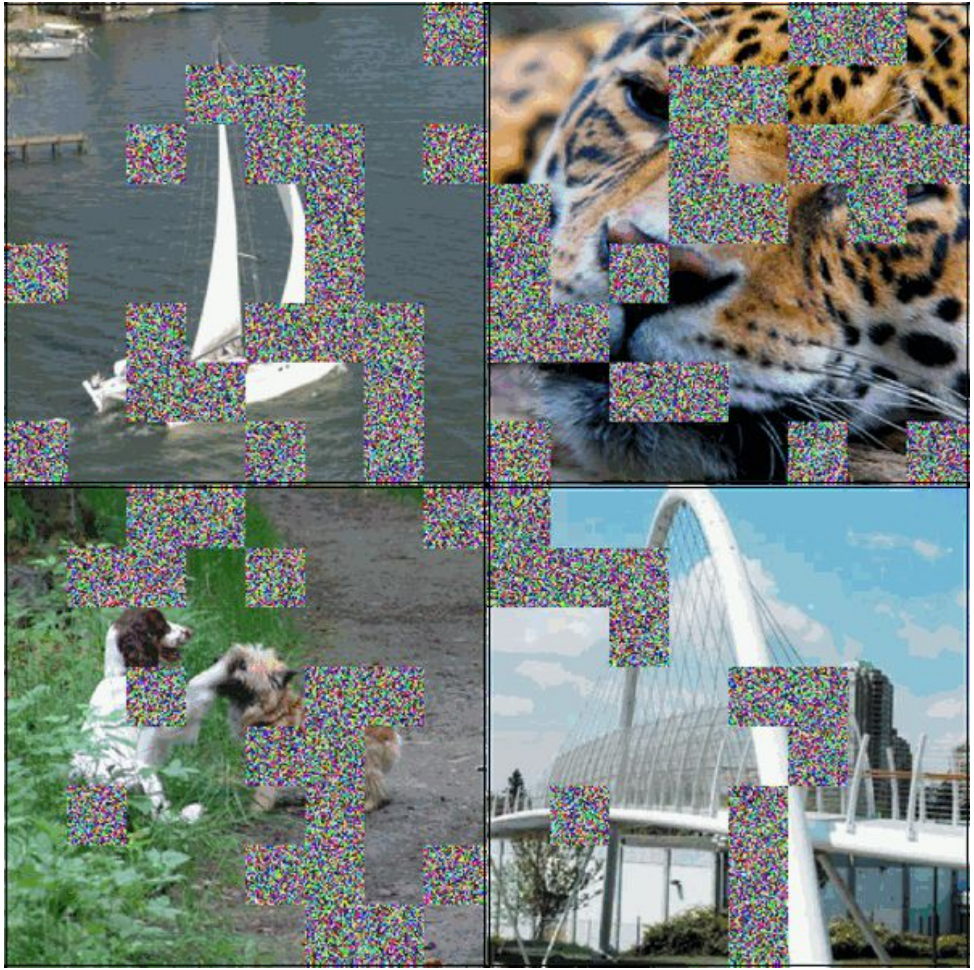


Figure credit: Michael Albergo

Next class:

Monday (Jan 27), 11am-12:30pm

Building an image generator!

E25-111 (same room)

Office hours: Tomorrow (Friday), 11am-12:30pm in 37-212