Lecture 03

Score Matching and Guidance

MIT IAP 2026 | Jan 23, 2025

Peter Holderrieth and Ron Shprints

Sponsor: Tommi Jaakkola

Reminder: Conditional Prob. Path and Cond. Vector Field

	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	$\left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t}\alpha_t\right) z + \frac{\dot{\beta}_t}{\beta_t} x$

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

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

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

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*

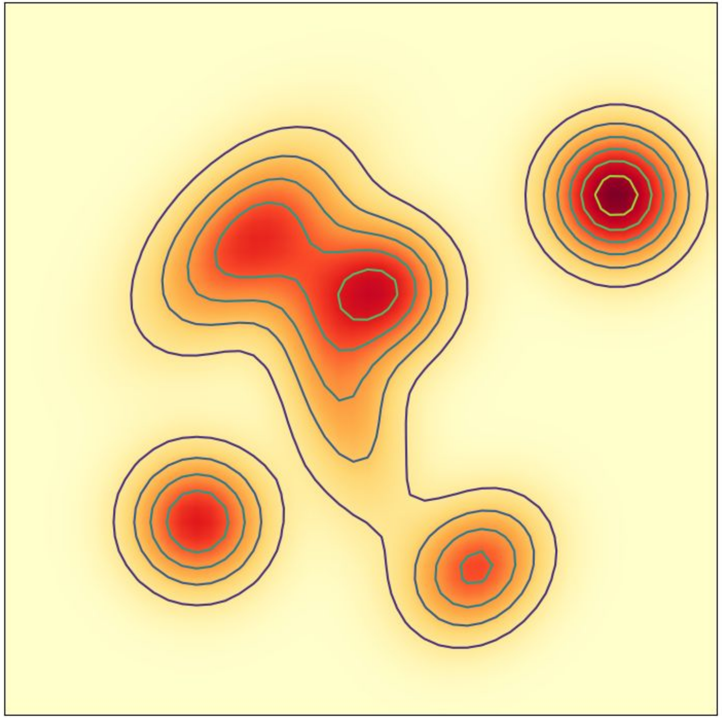
Section 4:

Training algorithms - 2: Score Matching

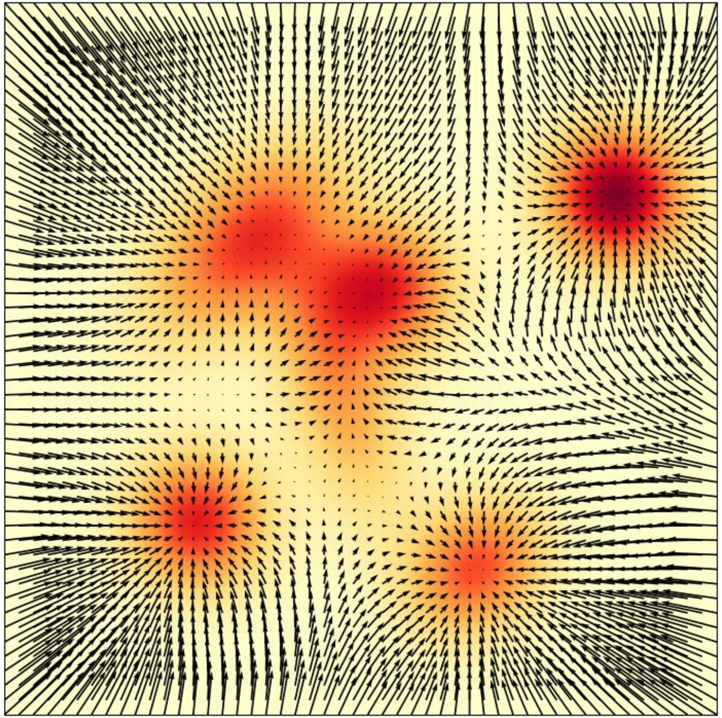
Goal: New perspective on flow and diffusion models.
SDE/Stochastic Sampling.

Score Functions = Gradients of the log-likelihood

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



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



Example - Score of Gaussian Probability Path

$$\nabla \log p_t(x|z) = -\frac{1}{\beta_t^2}x + \frac{\alpha_t}{\beta_t^2}z$$

Proof:

$$p_t(x|z) = \mathcal{N}(x; \alpha_t z, \beta_t^2 I_d) = \frac{1}{(2\pi)^{d/2}\beta_t^d}\exp\left(-\frac{1}{2\beta_t^2}\|x - \alpha_t z\|^2\right)$$

$$\log p_t(x|z) = \log \mathcal{N}(x; \alpha_t z, \beta_t^2 I_d) = -\frac{d}{2}\log(2\pi) - d\log\beta_t - \frac{1}{2\beta_t^2}\|x - \alpha_t z\|^2$$

$$\nabla \log p_t(x|z) = \nabla \log \mathcal{N}(x; \alpha_t z, \beta_t^2 I_d) = -\frac{x - \alpha_t z}{\beta_t^2}$$

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

Observation: Both Conditional VF and Cond Score are linear functions! Just with different coefficients!

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	$\frac{\alpha_t}{\beta_t^2} z - \frac{1}{\beta_t^2} x$

Reparameterization: Velocity Field –> Score Function

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

$$u_t^{\text{target}}(x|z) = a_t \nabla \log p_t(x|z) + b_t x$$

$$u_t^{\text{target}}(x) = a_t \nabla \log p_t(x) + b_t x$$

Proof: Algebra. Insert formulas. See lecture notes.

Early Diffusion Models learnt the score function instead and then just transformed it into the vector field! This is equivalent!

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

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

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

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

*Note what the network does: It needs to predict the noise that was used to corrupt the data point! (**DENOISING** diffusion models)*

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
```

Fokker-Planck equation

Randomly initialized SDE

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

Follow probability path:

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

Marginals are p_t

equivalent

Fokker-Planck equation holds

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

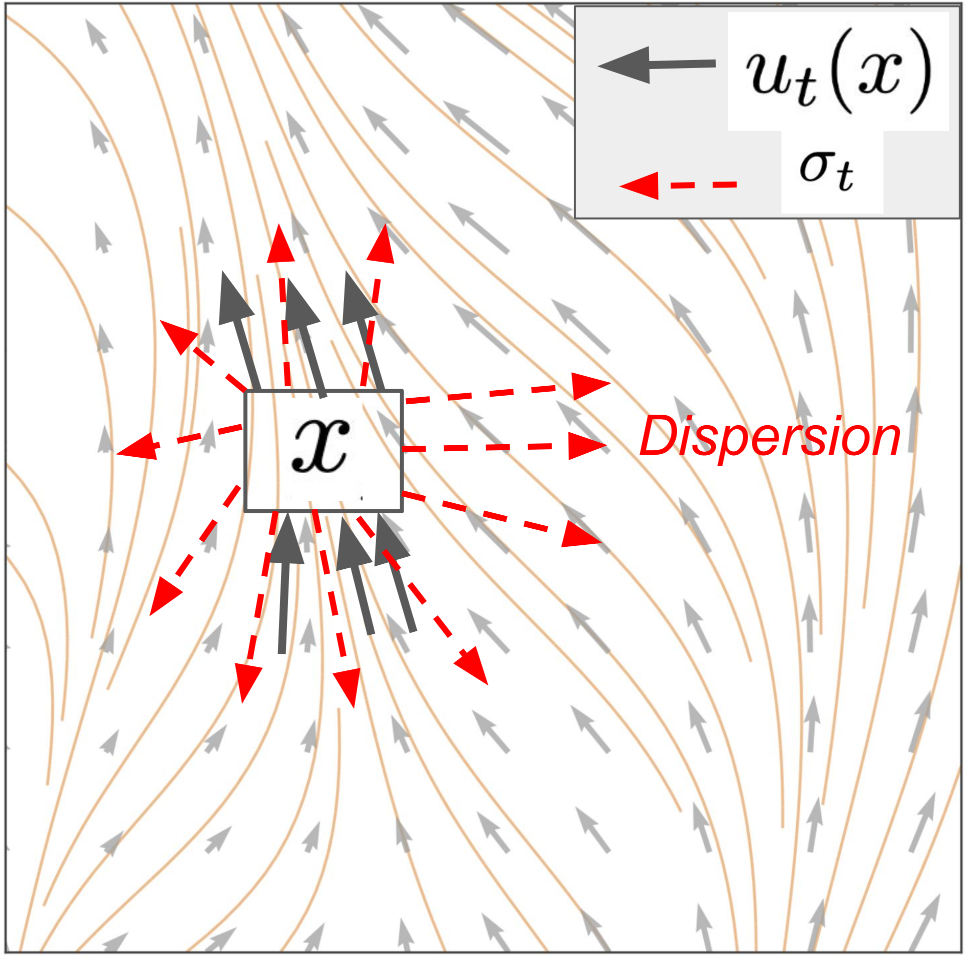
Continuity equ. *Heat equ.*

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 can sample from:

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

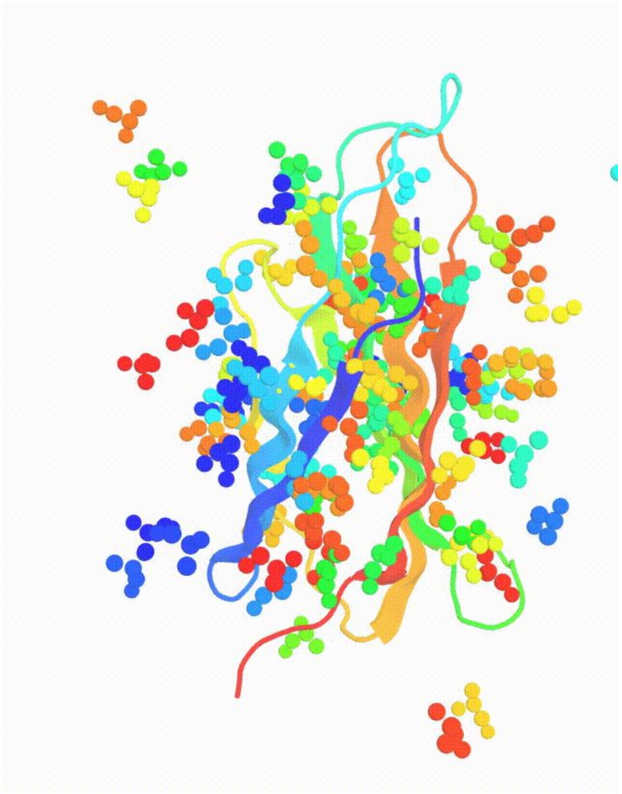
For Gaussian probability paths, we can express this solely in terms of the score:

$$\mathrm{d}X_t = \left[\left(a_t + \frac{\sigma_t^2}{2}\right)\nabla \log p_t(X_t) + b_t X_t\right]\mathrm{d}t + \sigma_t \mathrm{d}W_t$$

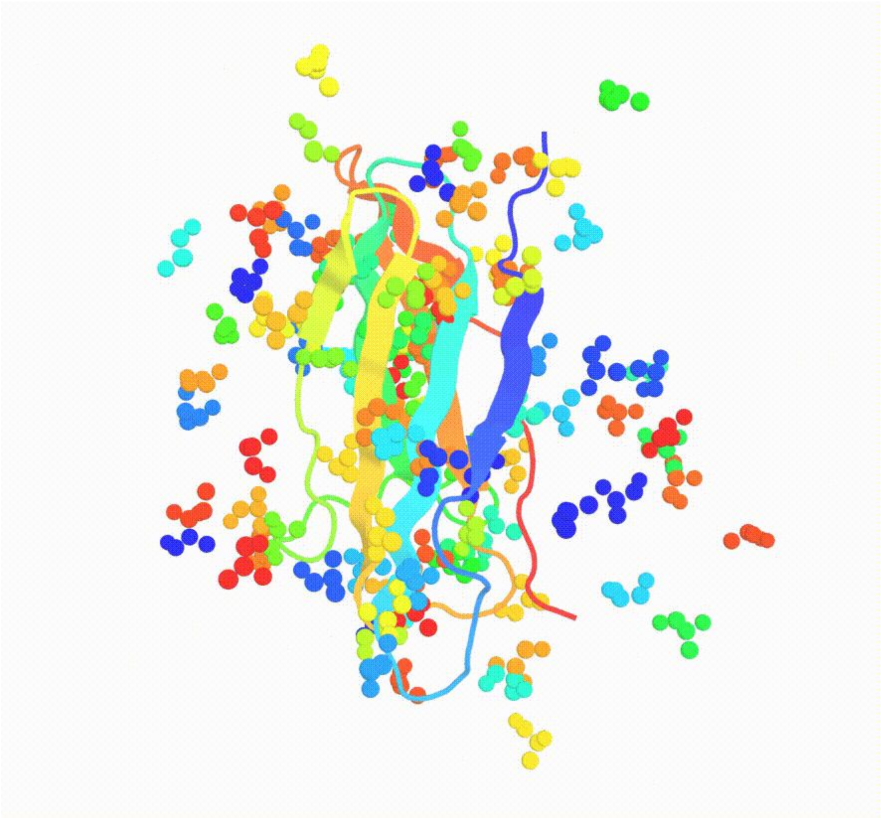
Plugin score network:

$$\mathrm{d}X_t = \left[\left(a_t + \frac{\sigma_t^2}{2}\right)s_t^\theta(X_t) + b_t X_t\right]\mathrm{d}t + \sigma_t \mathrm{d}W_t$$

Deterministic Sampling

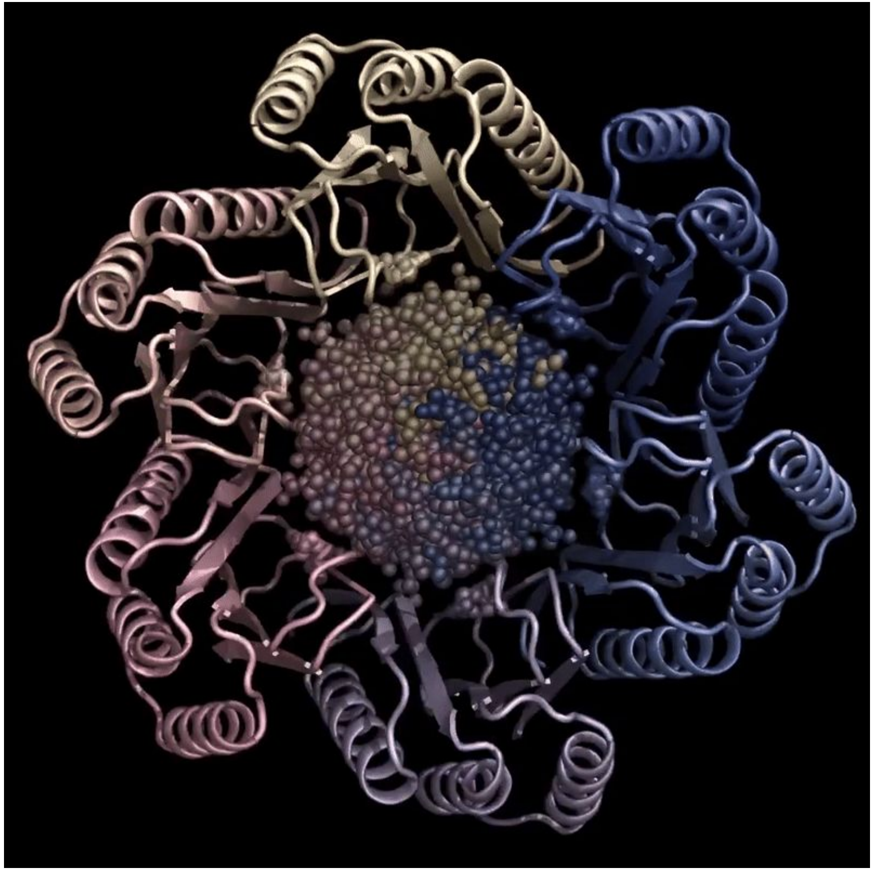


Stochastic Sampling



Stochastic (SDE) Sampling with Diffusion Models

Conversion of of noise into protein structure via SDE sampling



Slide credit: Jason Yim

Why would we want stochastic/SDE dynamics?

In theory: All diffusion coefficients lead to the same result (sample from data distribution).

In practice:

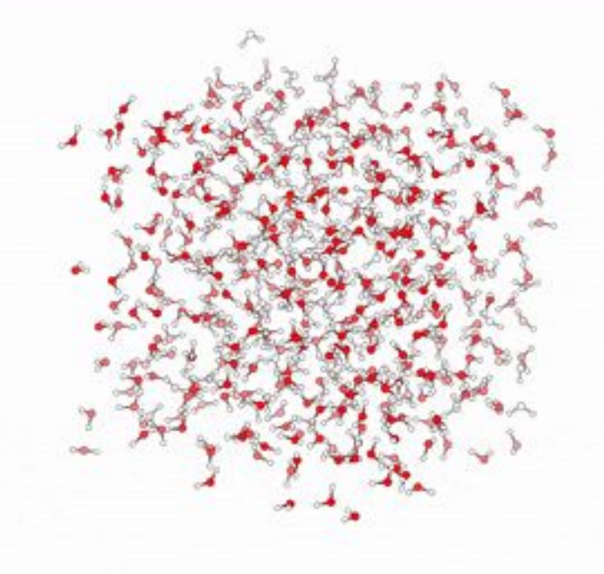
- **Training error:** Neural network has not perfectly learnt the marginal vector field/score.
- **Simulation error:** We need to simulate SDE/ODE leading to discretization error.

Downstream applications: Fine-tuning, inference-time optimization, etc. might require stochastic evolution

Good news: ODE sampling often leads to the best results.

Therefore, SDE sampling is an option, not a must!

Aside: Langevin dynamics - Basis of Molecular Dynamics simulation



Molecular dynamics simulate Langevin dynamics. This equals the SDE extension trick for marginal vector field = zero and a constant probability path.

$$\mathrm{d}X_t = \frac{\sigma_t^2}{2}\nabla \log p_t(X_t)\mathrm{d}t + \sigma_t \mathrm{d}W_t$$

$$p_t(x) = p_{\mathrm{Boltzmann}}(x) = \frac{1}{Z}\exp(-U(x))$$

We have shown:

$$X_0 \sim p_{\mathrm{Boltzmann}} \quad \Rightarrow \quad X_t \sim p_{\mathrm{Boltzmann}}$$

Equilibrium distribution

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:** 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 $t = 1$.

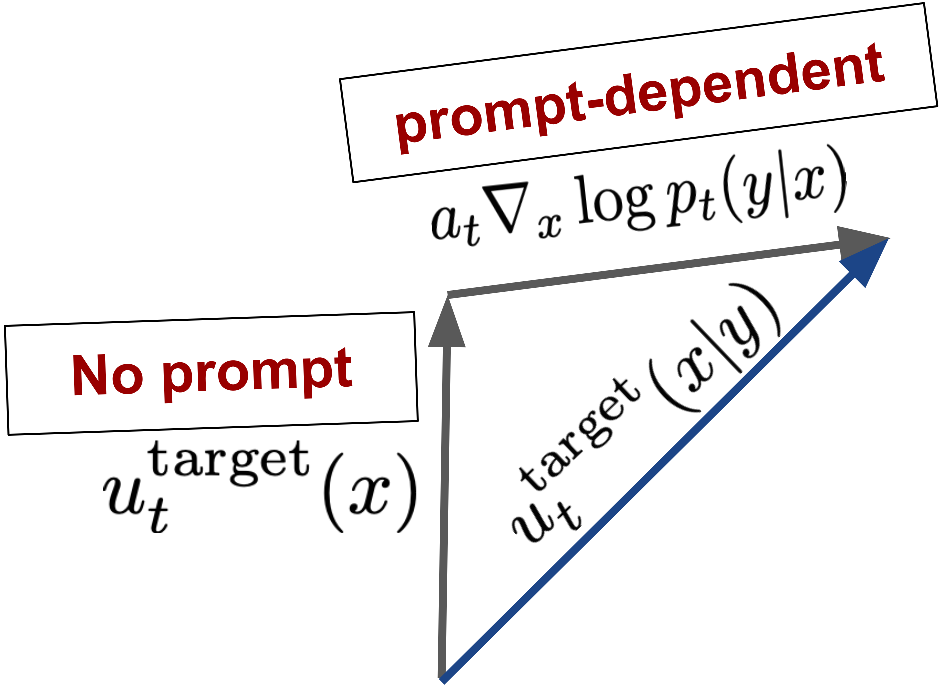
Vanilla Guidance leads to suboptimal results

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

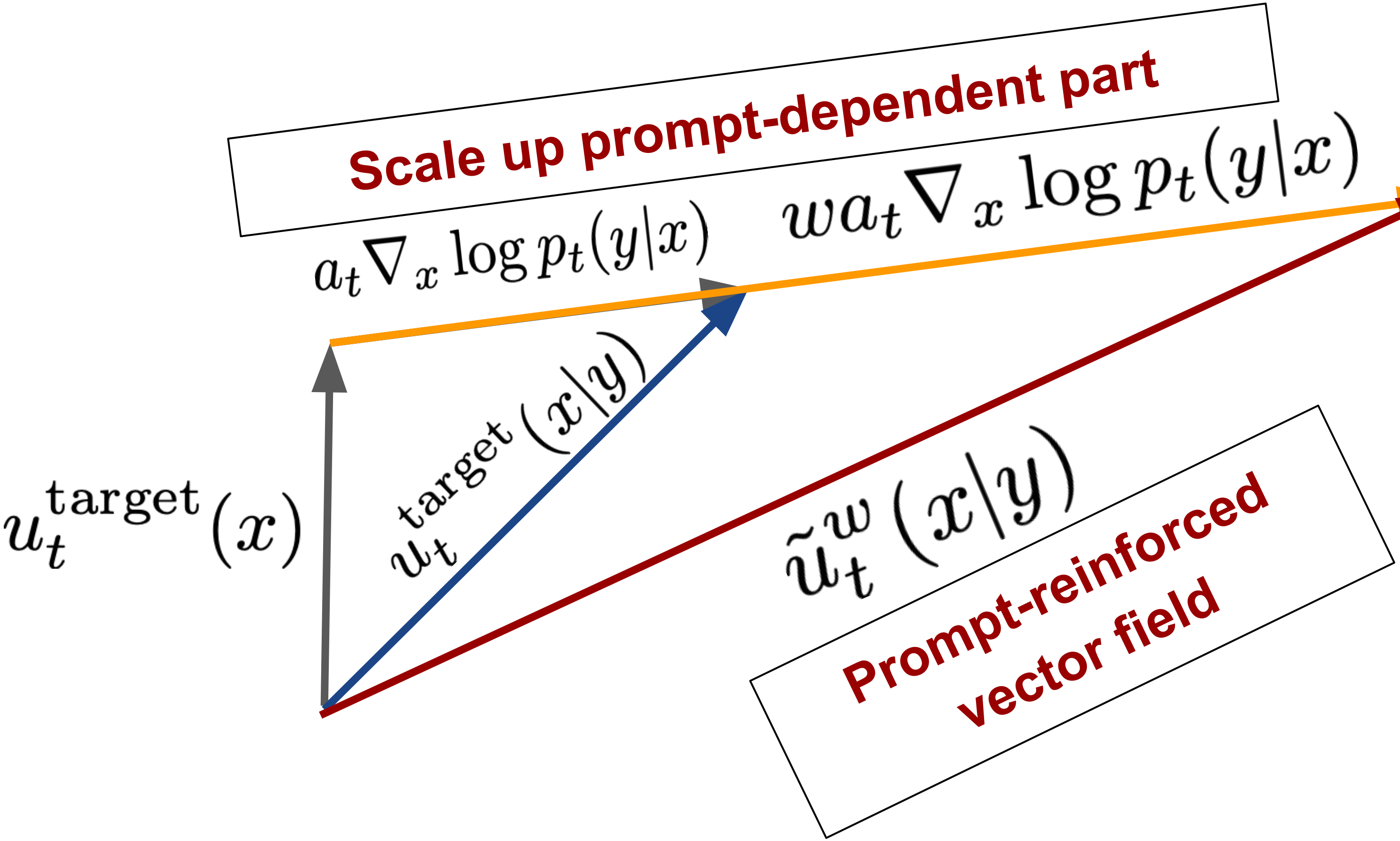
Prompt: ***"Corgi dog"***

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

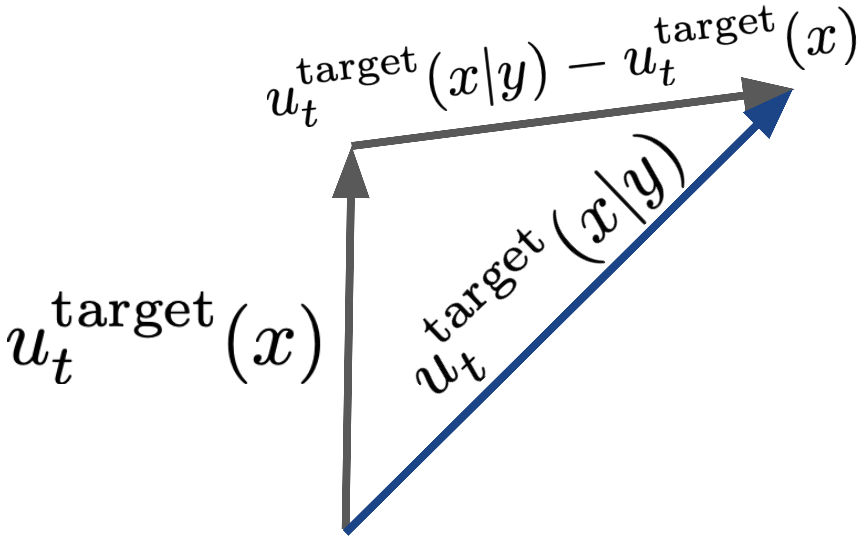
Intuition: Classifier guidance



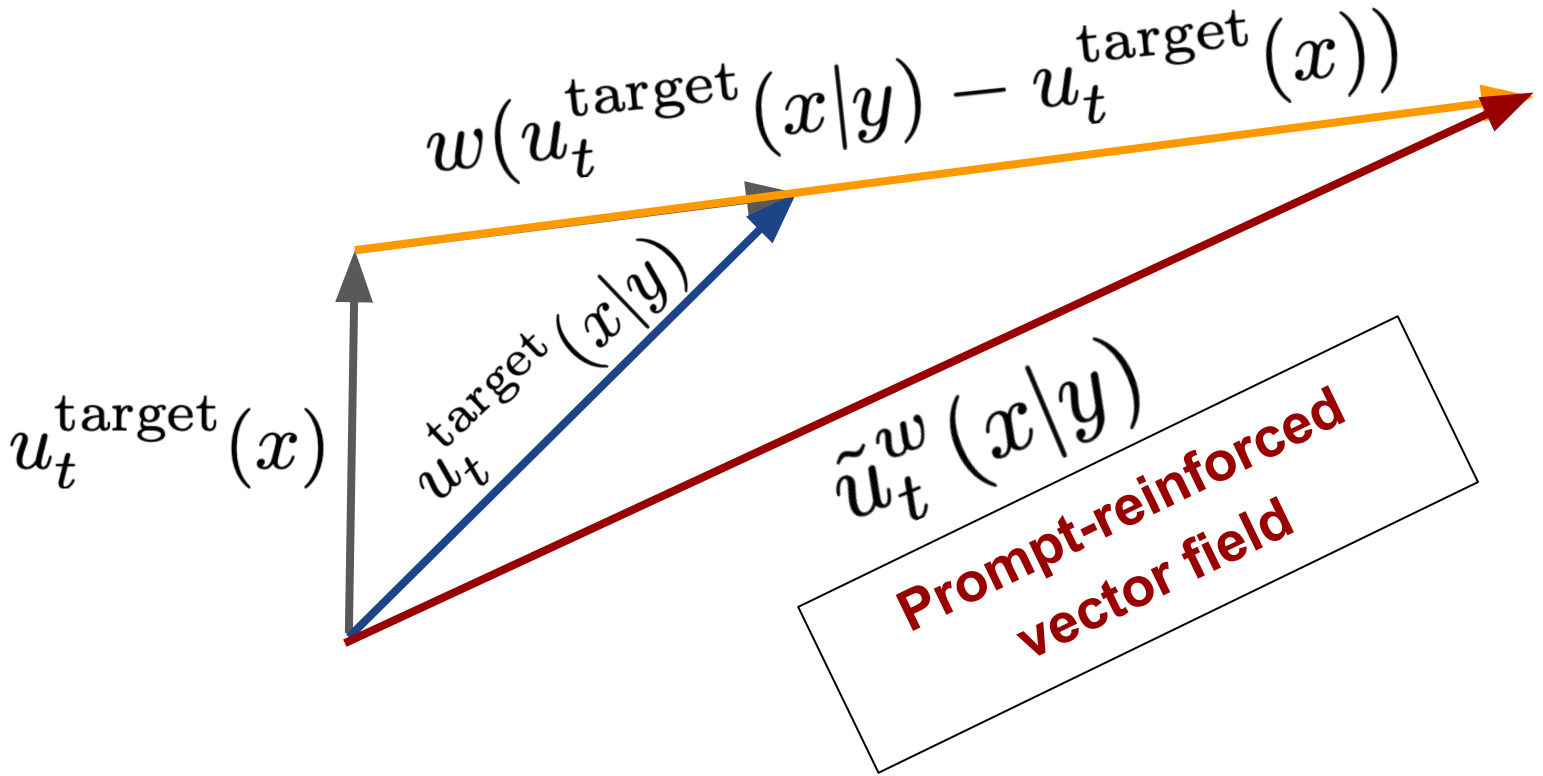
Intuition: Classifier guidance



Classifier-free guidance



Classifier-free guidance



Classifier-free guidance training: Account for empty token $\varnothing$

Algorithm 5 Classifier-free guidance training

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

```
for each mini-batch of data do
    Sample a data example (z, y) from the dataset.
    Sample a random time t ~ Unif_[0,1].
    Sample noise ε ~ N(0, I_d)
    Set x = α_t z + β_t ε
    With probability p drop label: y ← ∅
    Compute loss
```

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

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

Drop label with a certain probability!

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.

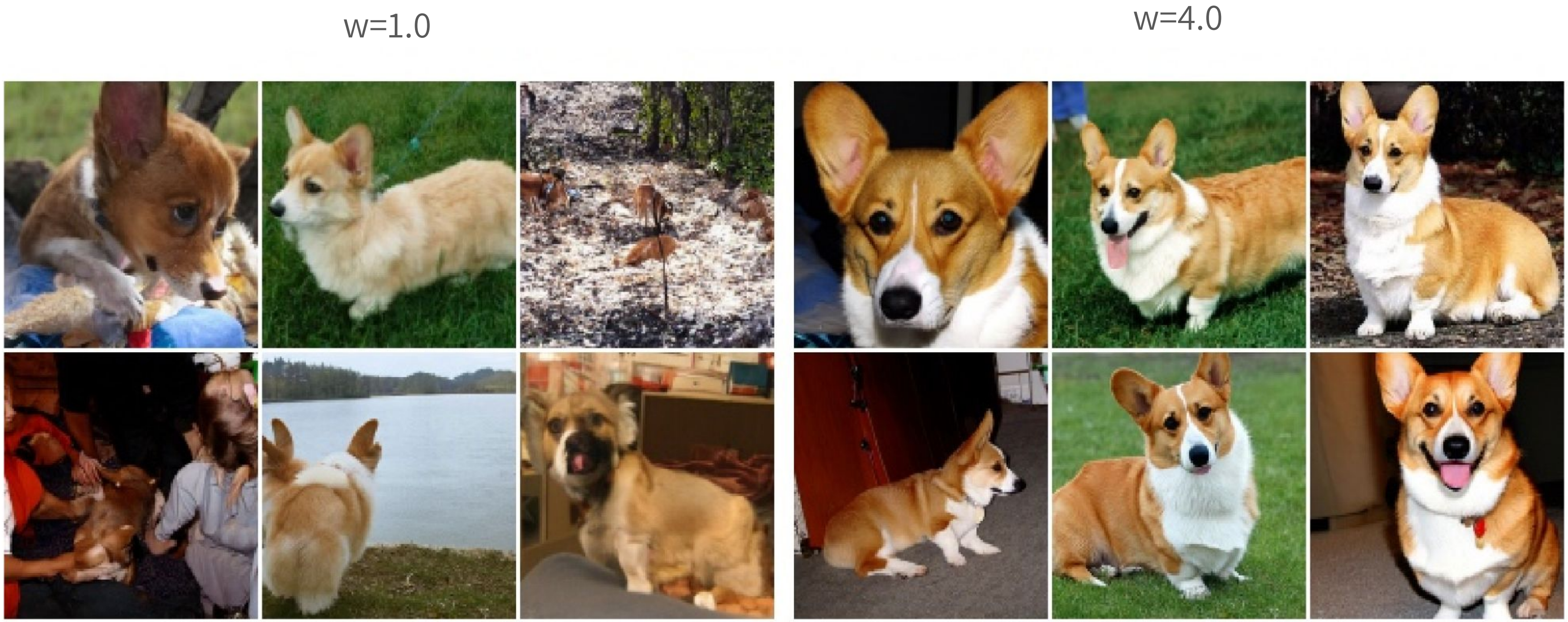
2: Select a **guidance scale** $w > 1$.

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 $t = 1$.

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

Example: Classifier-Free Guidance



Example: Classifier-Free Guidance

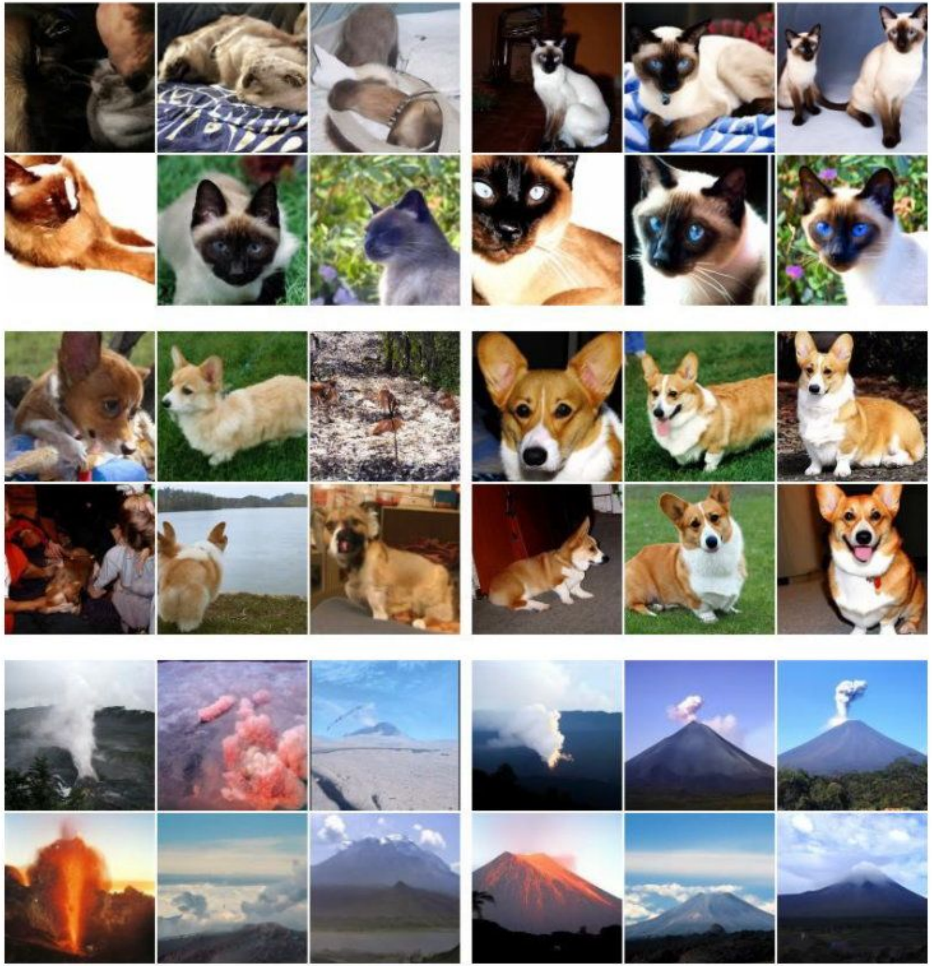


Image source: Classifier-free diffusion guidance [5].

Virtually all Images or Videos that you see use CFG!

- **CFG is key:** Without classifier-free guidance (CFG), almost nothing would work.


Example - Stable Diffusion 3: Classifier-free guidance scale w ~= 4.0

CFG does not model the data distribution anymore!

- **CFG is a heuristic:** We do not model the data distribution anymore. In fact, we go beyond it! It is primarily justified by its good empirical results!

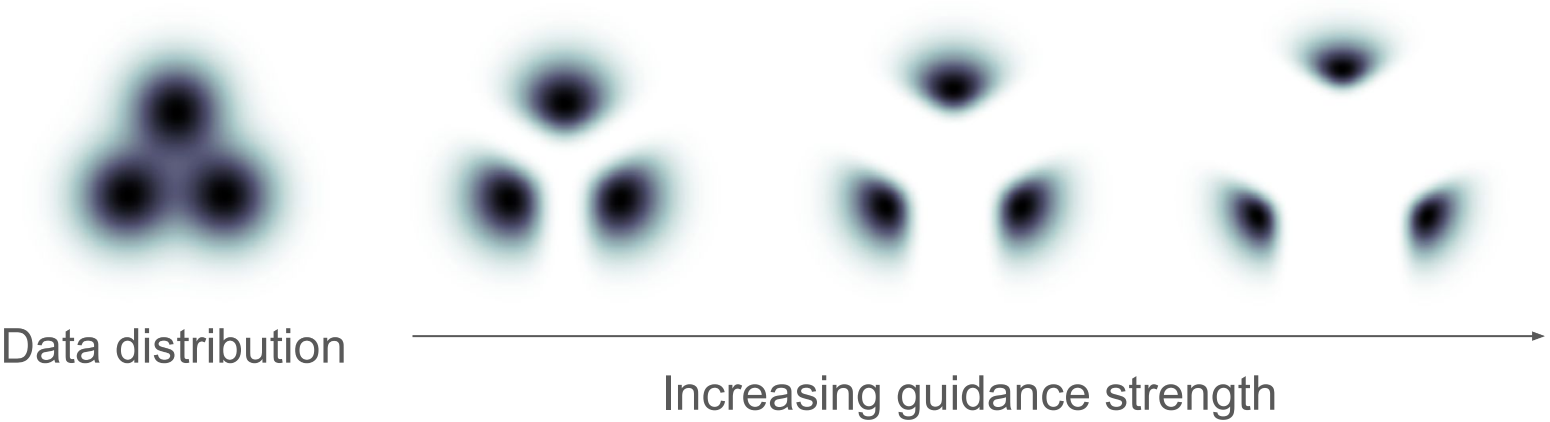


Image source: Classifier-free diffusion guidance [5].

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

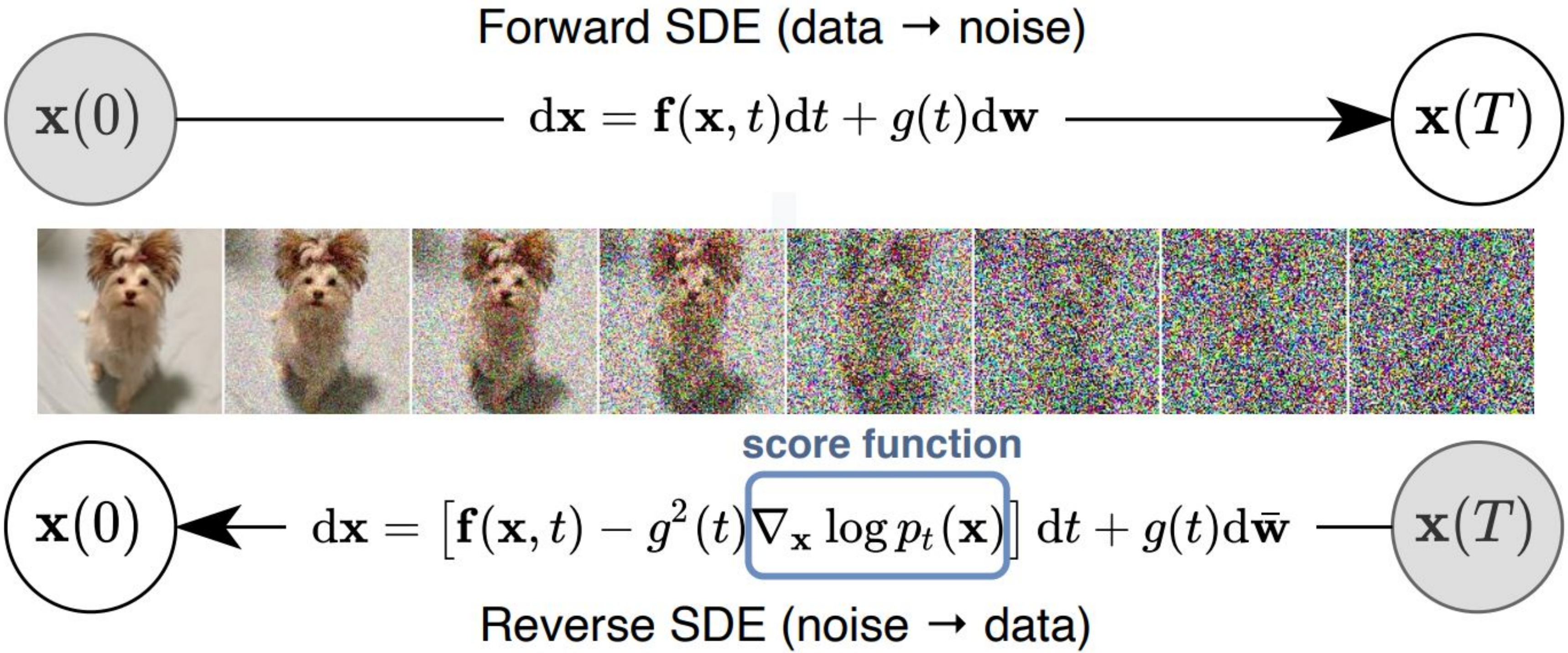
"Forward" diffusion process:

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

Denoising diffusion models

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

Constructing noising procedures via forward noising processes



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

- 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.

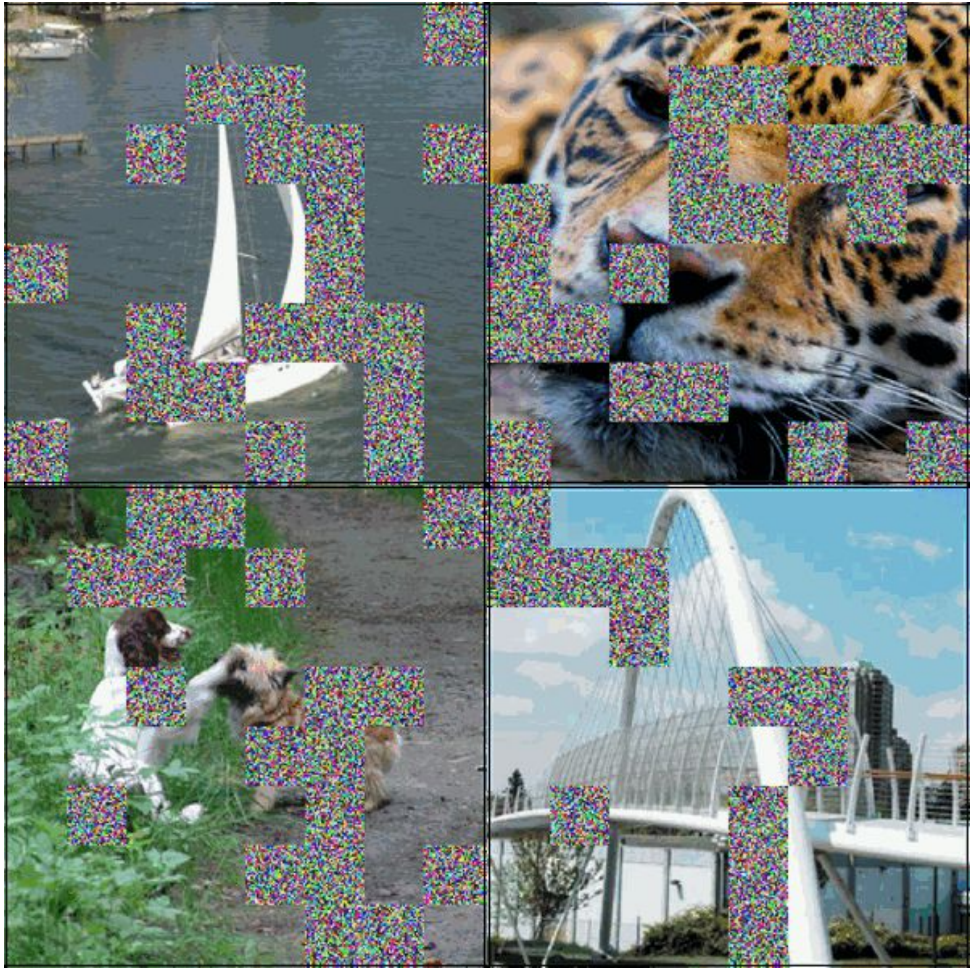


Figure credit: Michael Albergo

Class Overview

- **Lecture 1 -** Generation as Sampling. Flow and Diffusion Models
- **Lecture 2 - Flow Matching:** Training algorithm.
- **Lecture 3 - Score Matching, Guidance:** How to condition on a prompt.

- **Lecture 4 - Build Image Generators:** Network architectures + Latent spaces
- **Lecture 5 - Advanced Topics:** Discrete diffusion models + distilled models

Next class:

Monday, 11am-12:30pm

Neural network architectures + latent spaces!

E25-111 (same room)

Office hours: Today, 3pm-4:30pm in 36-156

References

1. **Scaling Rectified Flow Transformers for High-Resolution Image Synthesis, https://arxiv.org/abs/2403.03206**
2. **An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, https://arxiv.org/abs/2010.11929**
3. **Scalable Diffusion Models with Transformers, https://arxiv.org/abs/2212.09748**
4. **High Resolution Image Synthesis with Latent Diffusion Models, https://arxiv.org/abs/2112.10752**
5. **Classifier-Free Diffusion Guidance, https://arxiv.org/abs/2207.12598**