CS11-711 Advanced NLP

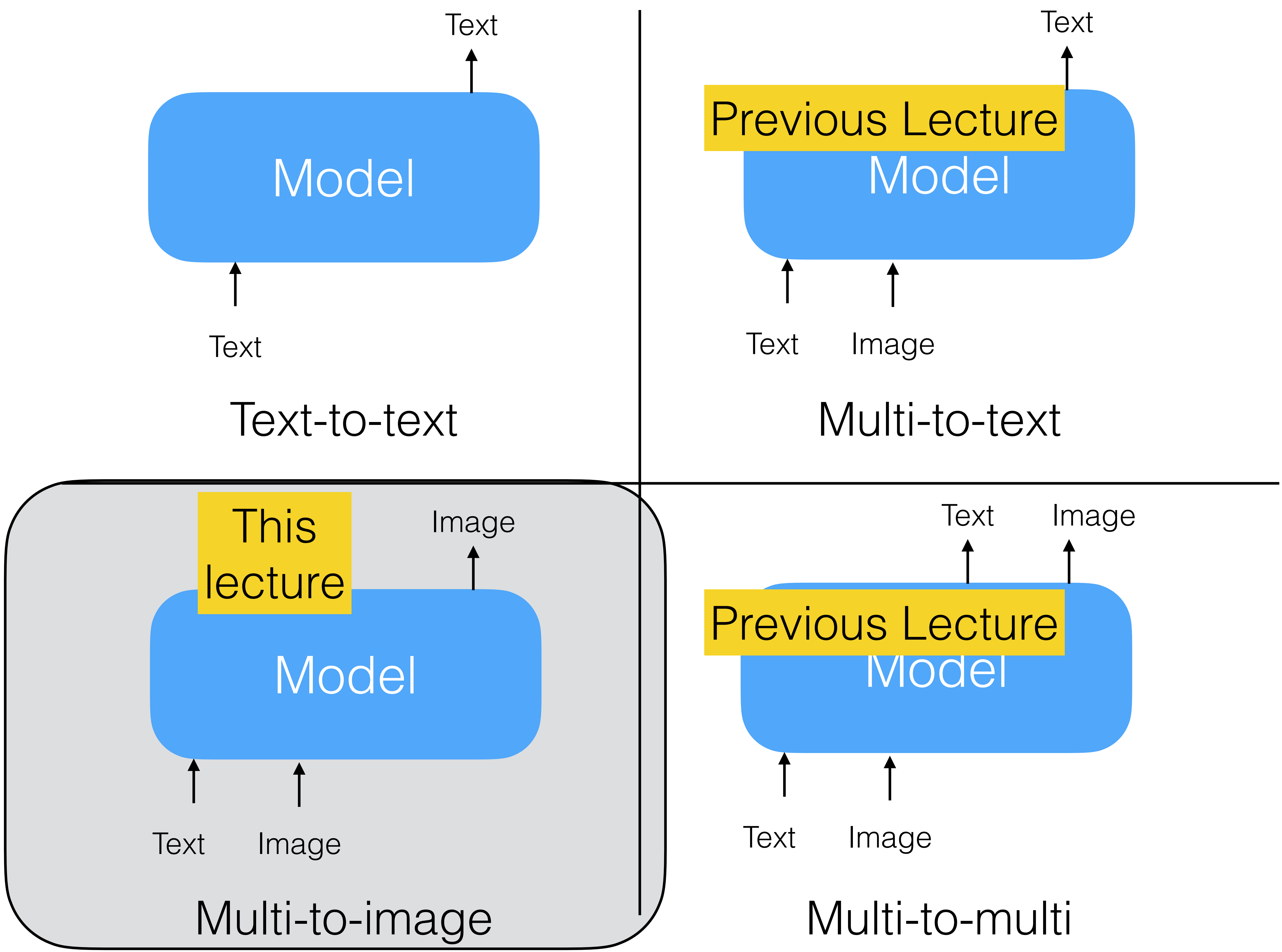
Diffusion and Flows

Multimodal Modeling III

Sean Welleck

https://cmu-l3.github.io/anlp-spring2026/

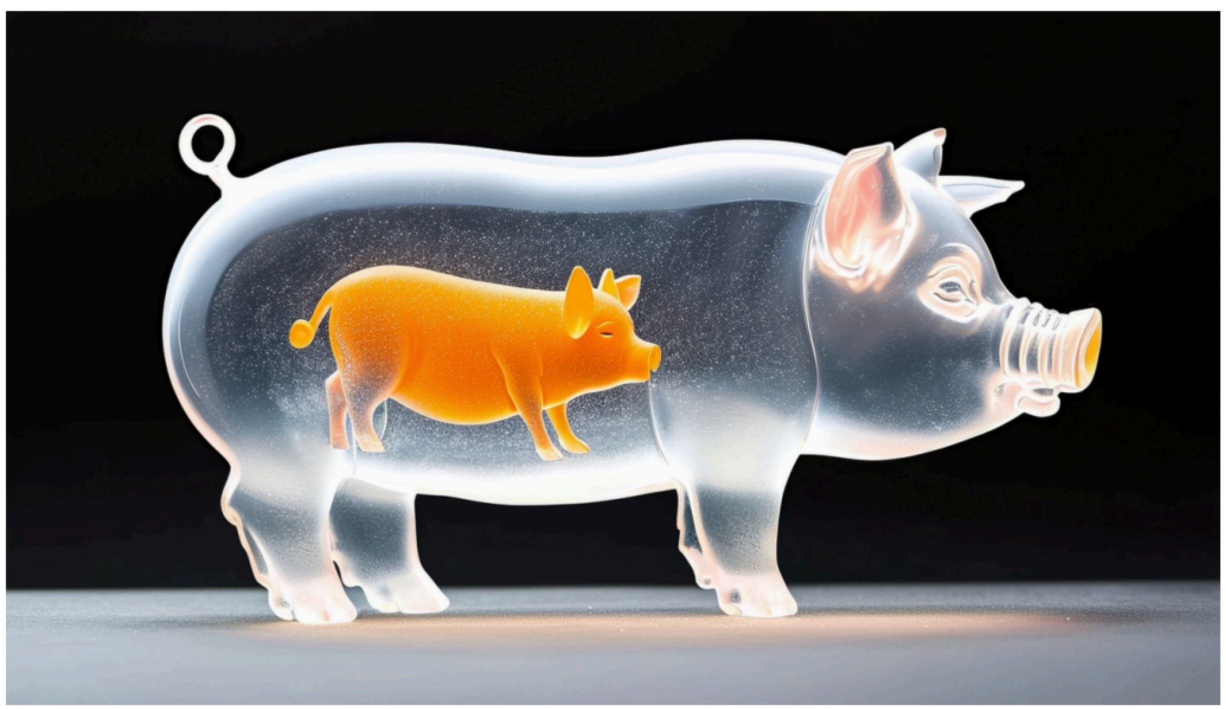
https://github.com/cmu-l3/anlp-spring2026-code



Text
Model
Text
Text-to-text
Text
Previous Lecture
Model
Text
Image
Multi-to-text
This lecture
Image
Model
Text
Image
Multi-to-image
Text
Image
Previous Lecture
Model
Text
Image
Multi-to-multi

A crab made of cheese on a plate

Dystopia of thousand of workers picking cherries and feeding them into a machine that runs on steam and is as large as a skyscraper. Written on the side of the machine: ”SD3 Paper”



translucent pig, inside is a smaller pig.

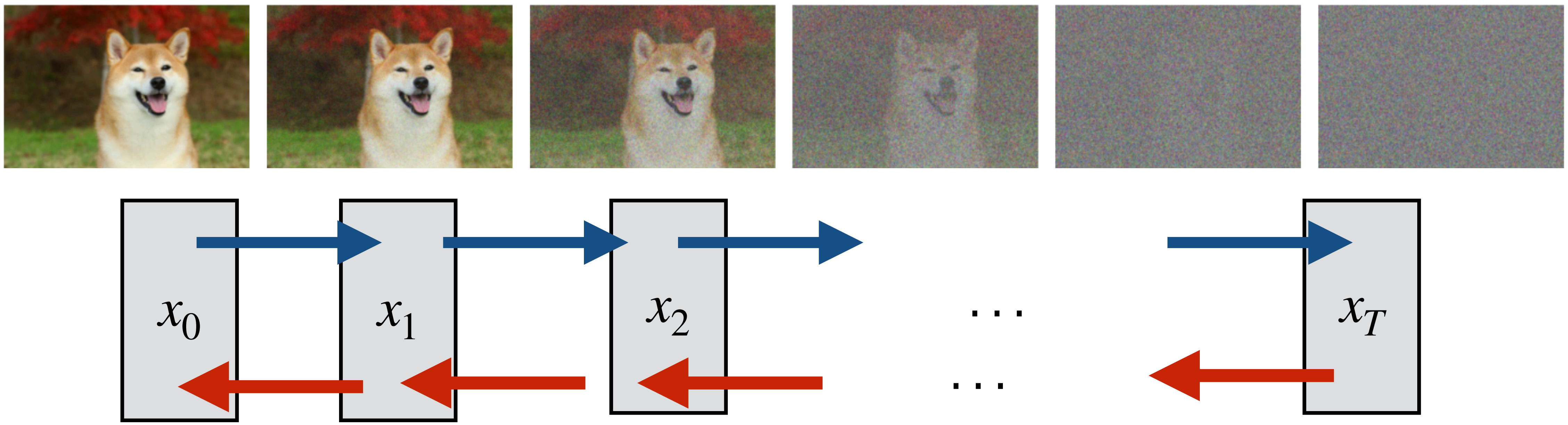
Film still of a long-legged cute big-eye anthropomorphic cheeseburger wearing sneakers relaxing on the couch in a sparsely decorated living room.

Example: Stable Diffusion 3

Today's lecture

- Diffusion basics
- Extensions and generalizations
- Flow matching

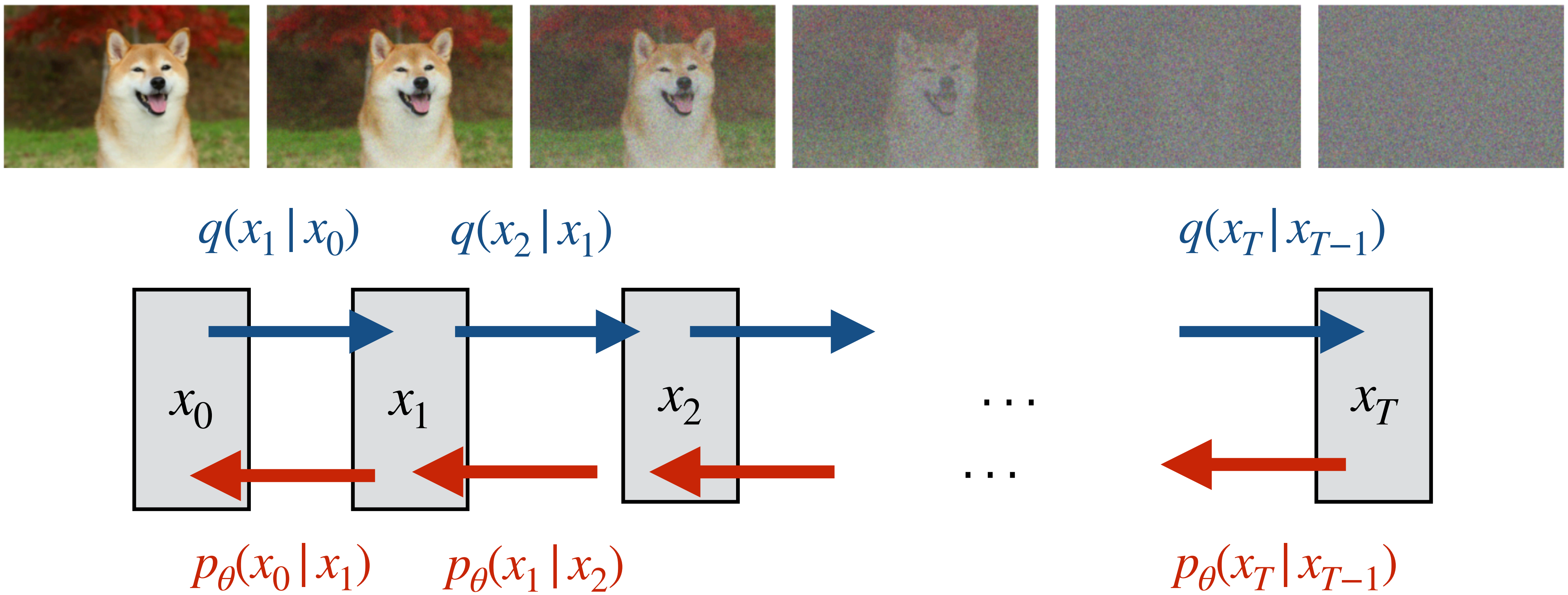
Diffusion: core idea



- Gradually add noise to an image, learn to de-noise

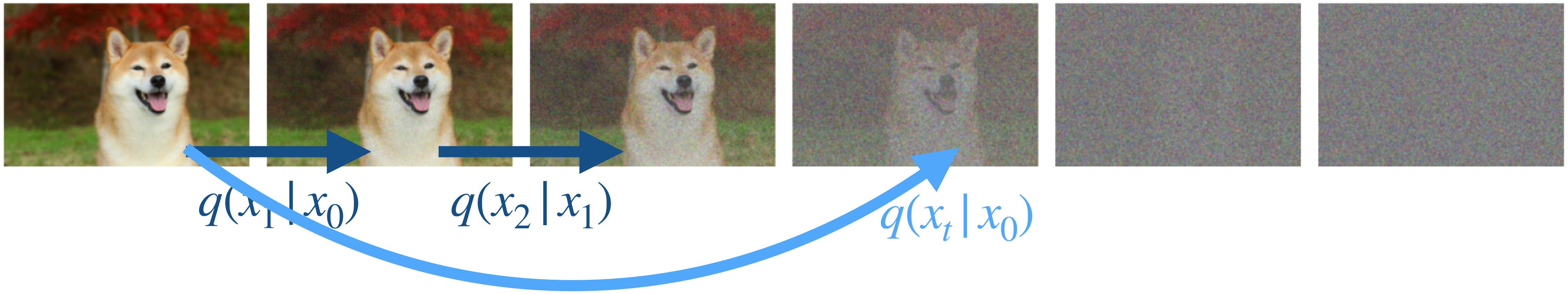
- Forward process
  - Start with data x_0
  - End with pure noise x_T
- Reverse process
  - Start with noise x_T
  - Recover data x_0

Forward and reverse process



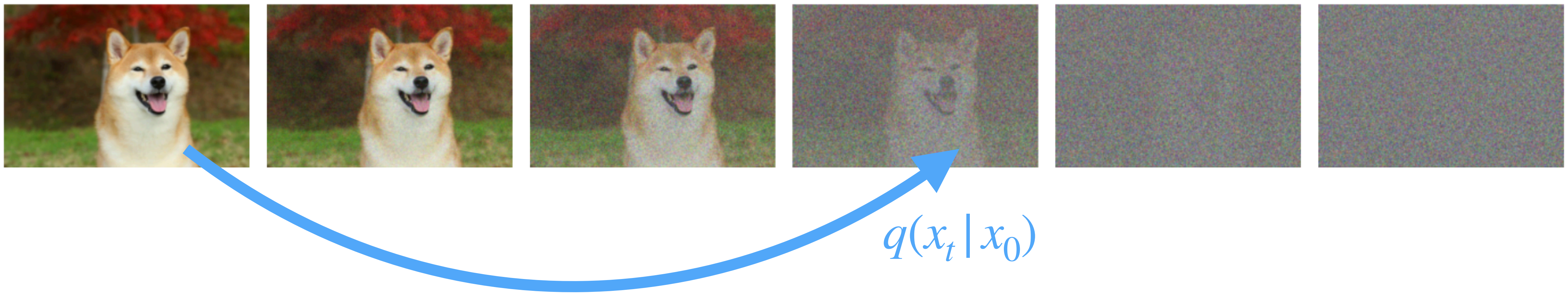
- Forward process (fixed)
  - $q(x_{1:T} | x_0) = \prod_{t=1}^{T} q(x_t | x_{t-1})$
- Reverse process (learned)
  - $p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} | x_t)$

Forward process: Gaussian



- $q(x_t | x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1}, (1 - \alpha_t) I)$
- Key property: we can sample x_t directly from x_0
  - $q(x_t | x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I)$
  - Where $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$
- Re-parameterization: $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$

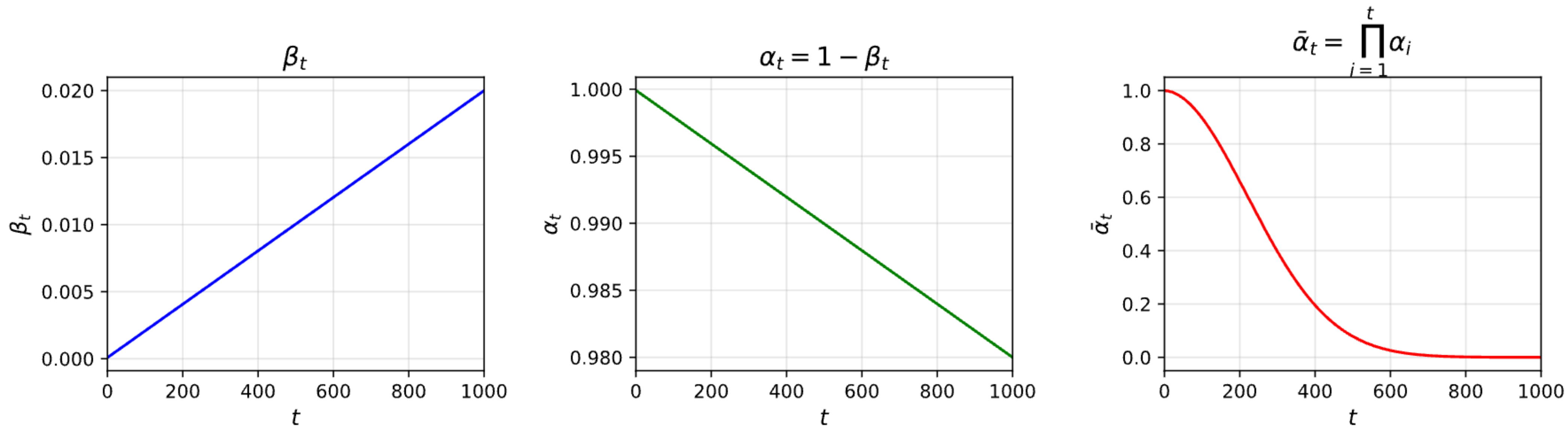
Noise schedule

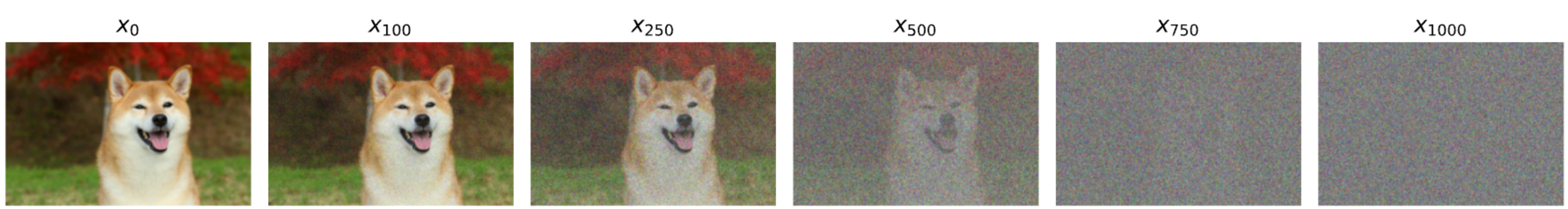


- $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \ \epsilon \sim \mathcal{N}(0, I)$
- Requirements:
  - Start with data: $\bar{\alpha}_0 = 1$
  - End with noise: $\bar{\alpha}_T \approx 0$
  - Monotonically decreasing

Noise schedule

- Example: linear schedule
  - $\beta_t = \beta_1 + \frac{t-1}{T-1}(\beta_T - \beta_1)$
  - Example with $\beta_1 = 0.0001,\ \beta_T = 0.02,\ T = 1000$



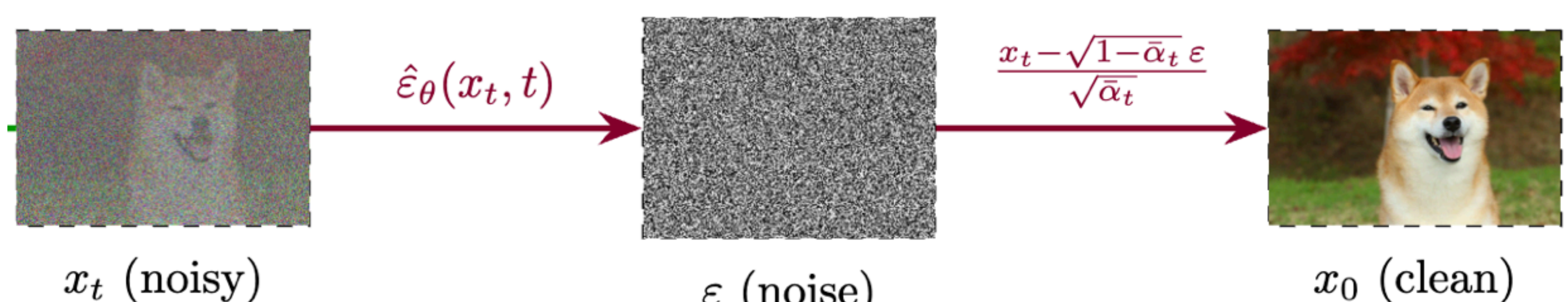


Training: preview

- Before we derive things, here's what training boils down to:
  - Minimize:

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

  - Sample data x_0, timestep t, random noise ϵ
  - Then we can get x_t due to reparameterization
  - Network predicts the noise given x_t



Training: ELBO

- Goal: maximize $\log p_\theta(x_0)$
- Problem: intractable integral over latents $x_{1:T}$

$$\log p_\theta(x_0) = \log \int p_\theta(x_{0:T}) dx_{1:T}$$

- Use importance sampling then Jensen's inequality

$$\log p_\theta(x_0) = \log \mathbb{E}_{q(x_{1:T}|x_0)} \left[\frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right]$$

$$\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right] = \text{ELBO}$$

- After expanding and rearranging, ELBO decomposes into 3 losses (next)

Training: ELBO

- The ELBO decomposes into:

$$L = \mathbb{E}_q\Big[\underbrace{D_{\mathrm{KL}}(q(x_T \mid x_0) \| p(x_T))}_{L_T\text{: prior matching}} + \sum_{t=2}^{T} \underbrace{D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \| p_\theta(x_{t-1} \mid x_t))}_{L_{t-1}\text{: denoising}} \underbrace{- \log p_\theta(x_0 \mid x_1)}_{L_0\text{: reconstruction}} \Big]$$

- Key term: L_{t-1}: match the learned reverse step $p_\theta(x_{t-1} \mid x_t)$ to the posterior $q(x_{t-1} \mid x_t, x_0)$. With Gaussians, leads to a simple loss (next)

Training: ELBO

$$\sum_{t=2}^{T} \underbrace{D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \| p_\theta(x_{t-1} \mid x_t))}_{L_{t-1}\text{: denoising}}$$

- We can show that $q(x_{t-1} \mid x_t, x_0)$ is Gaussian:
  - $q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(x_{t-1}; \mu_q(x_t, x_0), \sigma_q^2(t) I)$
    - $\mu_q(x_t, x_0) = \dfrac{\sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)}{1 - \bar{\alpha}_t} x_0 + \dfrac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} x_t$
- We use $p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1};\ \mu_\theta(x_t, t),\ \sigma_q^2(t)\, I)$
- KL between two Gaussians with the same variance reduces to:

$$L_{t-1} \propto \| \mu_q(x_t, x_0) - \mu_\theta(x_t, t) \|^2$$

Data and noise prediction

- We can reparameterize to predict x_0. Key idea:
  - $\mu_q(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t$
  - $\mu_\theta(x_t, t) = \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t} x_\theta(x_t, t) + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t$
- This leads to the **data prediction loss**

$$L = \mathbb{E}_{x_0, t, \epsilon} \| x_\theta(x_t, t) - x_0 \|^2$$

Data and noise prediction

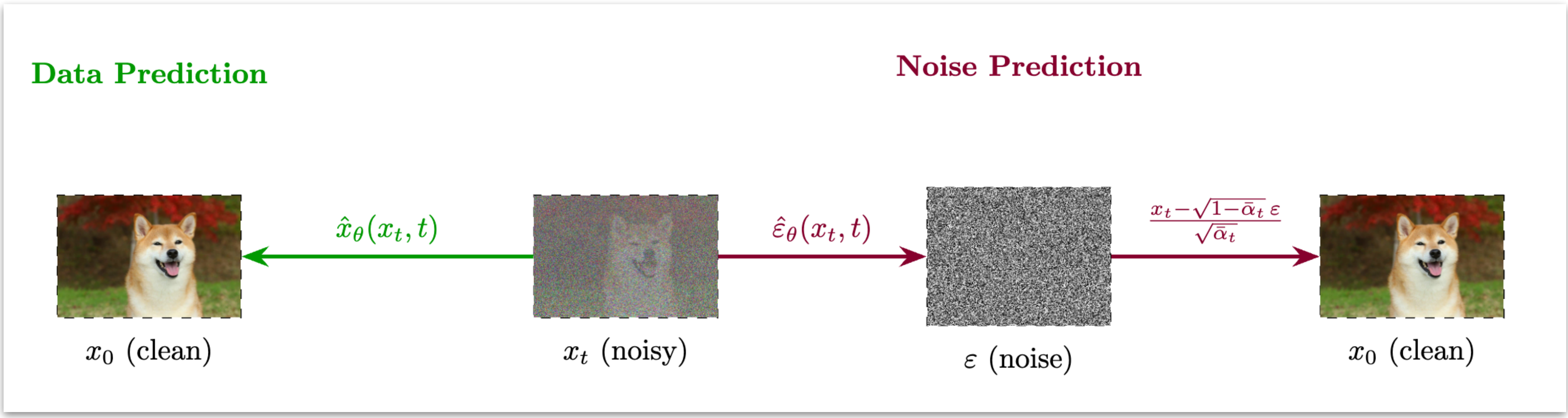
- We can reparameterize again to predict ϵ:
  - Since $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon$, we have:

$$x_0 = \frac{x_t - \sqrt{1 - \bar{\alpha}_t}\, \varepsilon}{\sqrt{\bar{\alpha}_t}}$$

  - $\mu_q(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}\sqrt{\alpha_t}} \epsilon$
  - $\mu_\theta(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}\sqrt{\alpha_t}} \epsilon_\theta(x_t, t)$
  - This leads to the **noise prediction loss**

$$L = \mathbb{E}_{t, x_0, \epsilon} \| \epsilon_\theta(x_t, t) - \epsilon \|^2$$

Data and noise prediction



Training: summary

- **Simplified loss**:

$$L_{\text{simple}}(\theta) = \mathbb{E}_{t,x_0,\varepsilon}\left[\|\varepsilon - \hat{\varepsilon}_\theta(x_t, t)\|^2\right]$$

$$t \sim \text{Uniform}\{1,\ldots,T\},\ \varepsilon \sim \mathcal{N}(0,I),\ x_t = \sqrt{\bar{\alpha}_t}\,x_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon$$

- **Intuition**
  - Sample a random timestep t
  - Add noise to get x_t
  - Train network to predict the noise that was added

Sampling

- Recall our reverse process is
  $$p_\theta(x_{t-1} | x_t) = \mathcal{N}(x_{t-1};\ \mu_\theta(x_t, t), \sigma_q^2(t) I)$$
- $$\mu_\theta(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}\sqrt{\alpha_t}} \epsilon_\theta(x_t, t)$$
- Start with noise
- For t = T…0:
  - Run neural network ϵ_θ to get μ_θ, sample from the resulting Gaussian

Training and sampling summary

Training

1. Sample $x_0 \sim q(x_0)$
2. Sample $t \sim \mathrm{Uniform}\{1, \ldots, T\}$
3. Sample $\varepsilon \sim \mathcal{N}(0, I)$
4. Compute $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon$
5. Gradient step on $\|\varepsilon - \hat{\varepsilon}_\theta(x_t, t)\|^2$

Sampling

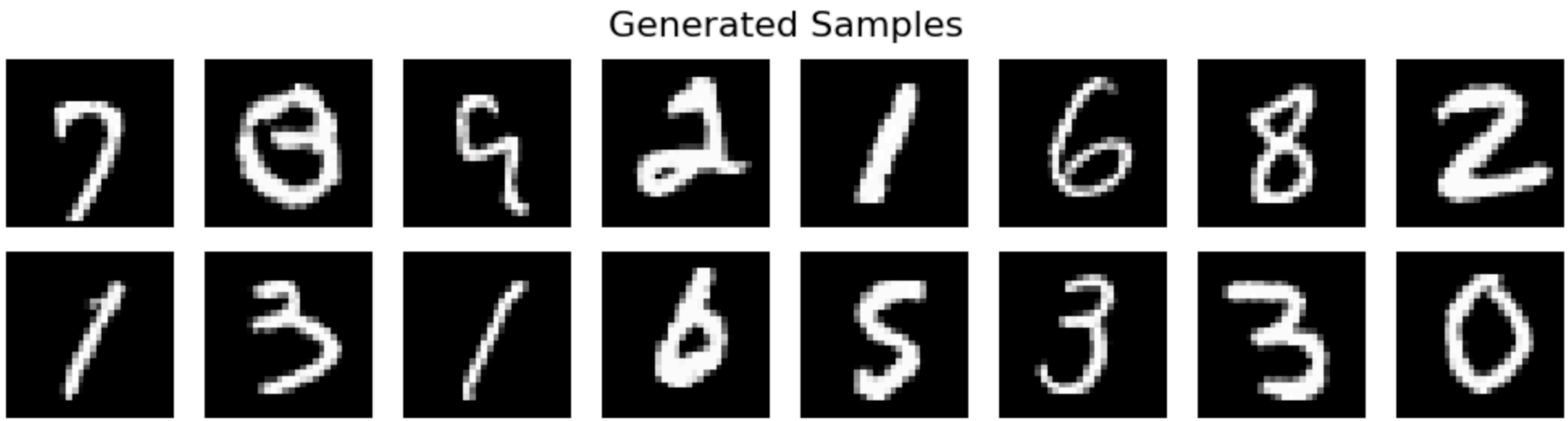
1. Sample $x_T \sim \mathcal{N}(0, I)$
2. For $t = T, \ldots, 1$:
   - $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$
   - $x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \hat{\varepsilon}_\theta(x_t, t) \right) + \sigma_t z$
3. Return x_0

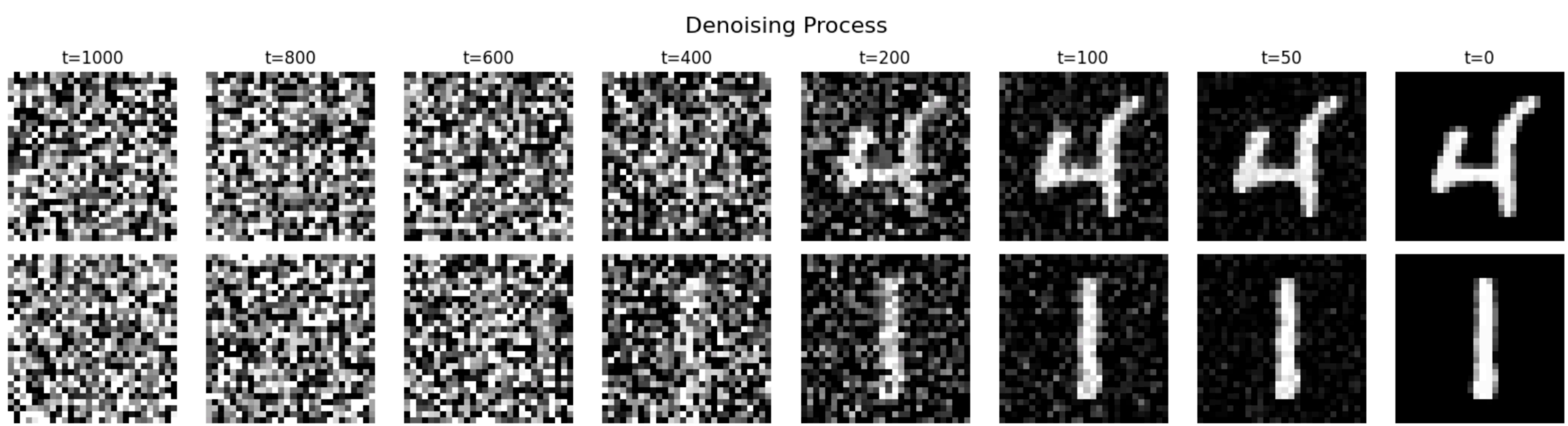
Denoising Diffusion Probabilistic Models (DDPM)

[Ho et al 2020]

Images from: https://hojonathanho.github.io/diffusion/

Code example





https://github.com/cmu-l3/anlp-spring2026-code/tree/main/15_diffusion

From noise to gradients

- **Notation**: let's denote the intermediate diffusion steps as:
  - $z_t = a_t x_0 + \sigma_t \epsilon,$

    where $a_t = \sqrt{\bar{\alpha}_t}, \sigma_t = \sqrt{1 - \bar{\alpha}_t}$
- The **score** of a distribution $p(z_t)$ is defined to be:
  - $\nabla_{z_t} \log p(z_t)$
- Points in the direction that increases the probability of z_t

From noise to gradients

- We can show the following: if $z_t = a_t x_0 + \sigma_t \epsilon$, then the score of the marginal distribution $p(z_t)$ satisfies:
  - $\nabla_{z_t} \log p(z_t) = -\dfrac{\bar{\epsilon}}{\sigma_t}$
  - => $\bar{\epsilon} = -\sigma_t \nabla_{z_t} \log p(z_t)$
  - Where $\bar{\epsilon}$ is the expected noise,
    - $\bar{\epsilon} = \mathbb{E}_{x_0 \sim p(x_0|z_t)}[\epsilon \,|\, z_t]$

"Tweedie's formula"

From noise to gradients

- Since DDPM minimizes $\mathbb{E}_{x_0,t,\epsilon}||\epsilon_\theta(z_t, t) - \epsilon||^2$, the optimal $\epsilon_\theta(z_t, t)$ converges to $\bar{\epsilon} = \mathbb{E}[\epsilon \,|\, z_t]$. Hence:
  - $\epsilon_\theta(z_t, t) \approx -\sigma_t \nabla_{z_t} \log p(z_t)$

DDPM implicitly learns the score function!

- Intuition:
  - Predicting which noise was added tells you which direction to remove noise
    - And removing noise moves you toward higher probability

Basic diffusion (DDPM) recap

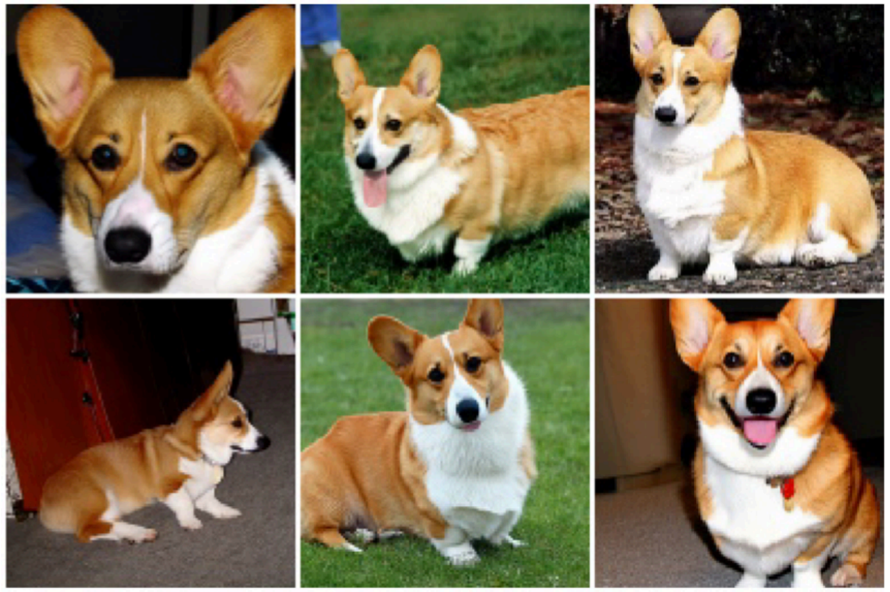
- Define a fixed forward process that gradually destroys data with noise
- Train a neural network to reverse each step (e.g., predict the noise)
- Generate by sampling noise and iteratively denoising
- Intuitively, the model learns to point to directions of higher probability density

Today's lecture

- Diffusion basics
- **Extensions:**
  - Guidance
  - Other extensions
- Flow matching

Conditional generation

- We want to be able to condition on information
- Example:
  - “Dog” -> diffusion model ->



Classifier guidance

[Dhariwal & Nichol 2021]

- Suppose that we have a classifier $p_\phi(c \mid z_t)$
- Recall the connection $\epsilon_\theta(z_t) \approx -\sigma_t \nabla_{z_t} \log p(z_t)$
- We can add an additional term using the classifier gradient:
    - $\tilde{\epsilon}_{\theta,\phi}(z_t, c) = \epsilon_\theta(z_t, c) - w\sigma_t \nabla_{z_t} \log p_\phi(c \mid z_t)$
- Now run diffusion sampling using $\tilde{\epsilon}_{\theta,\phi}$

Classifier-free guidance

[Ho & Salimans 2022]

- Train a single model with and without conditioning
- Sample with a modified score:

$$\tilde{\epsilon}_\theta(z_t, c) = (1 + w)\epsilon_\theta(z_t, c) - w\epsilon_\theta(z_t)$$

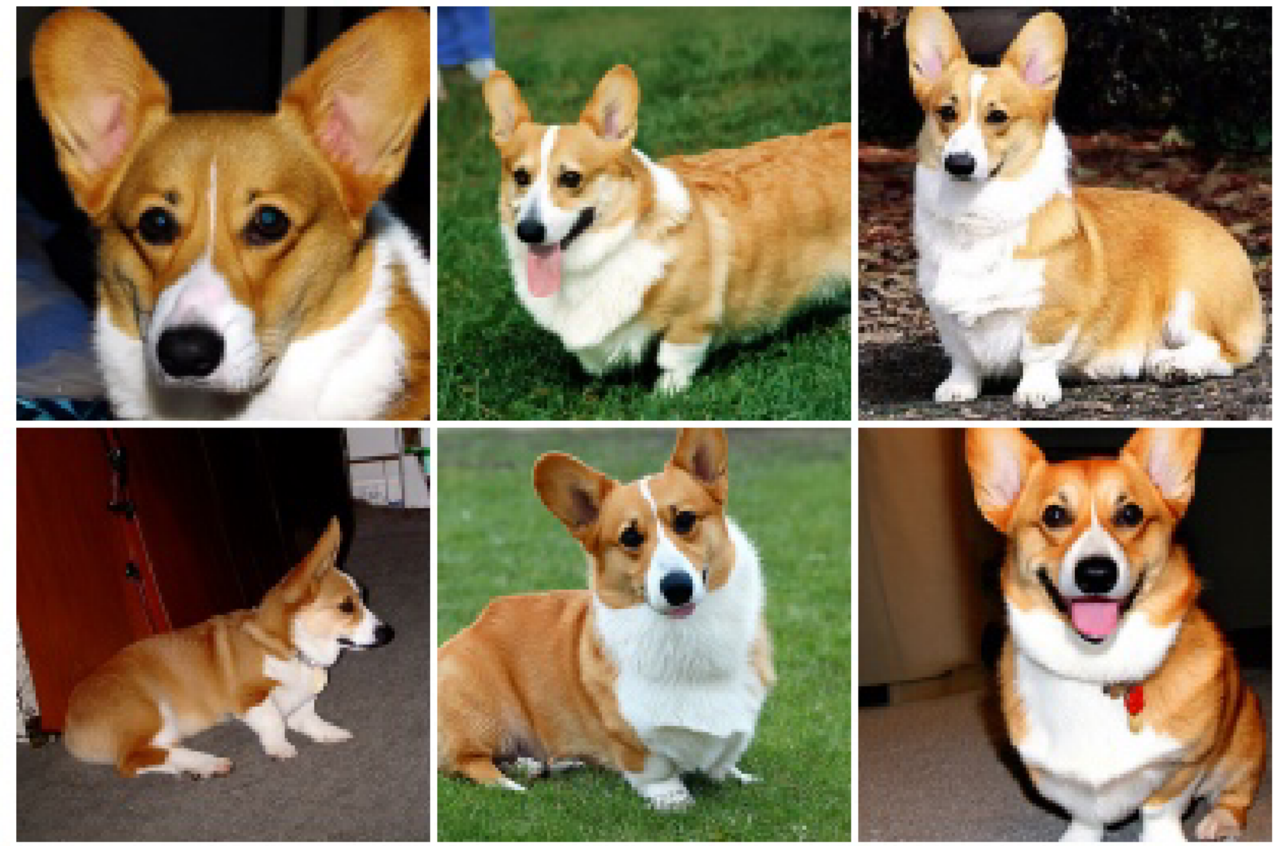
- We can view this as doing classifier guidance with an implicit classifier $p(c \mid z_t) \propto p(z_t \mid c)/p(z_t)$
  - Specifically classifier guidance with the gradient

$$\nabla_{z_t} \log p(c \mid z_t) = -\frac{1}{\sigma_t}[\epsilon^*(z_t, c) - \epsilon^*(z_t)]$$

Classifier-free guidance

[Ho & Salimans 2022]

No guidance

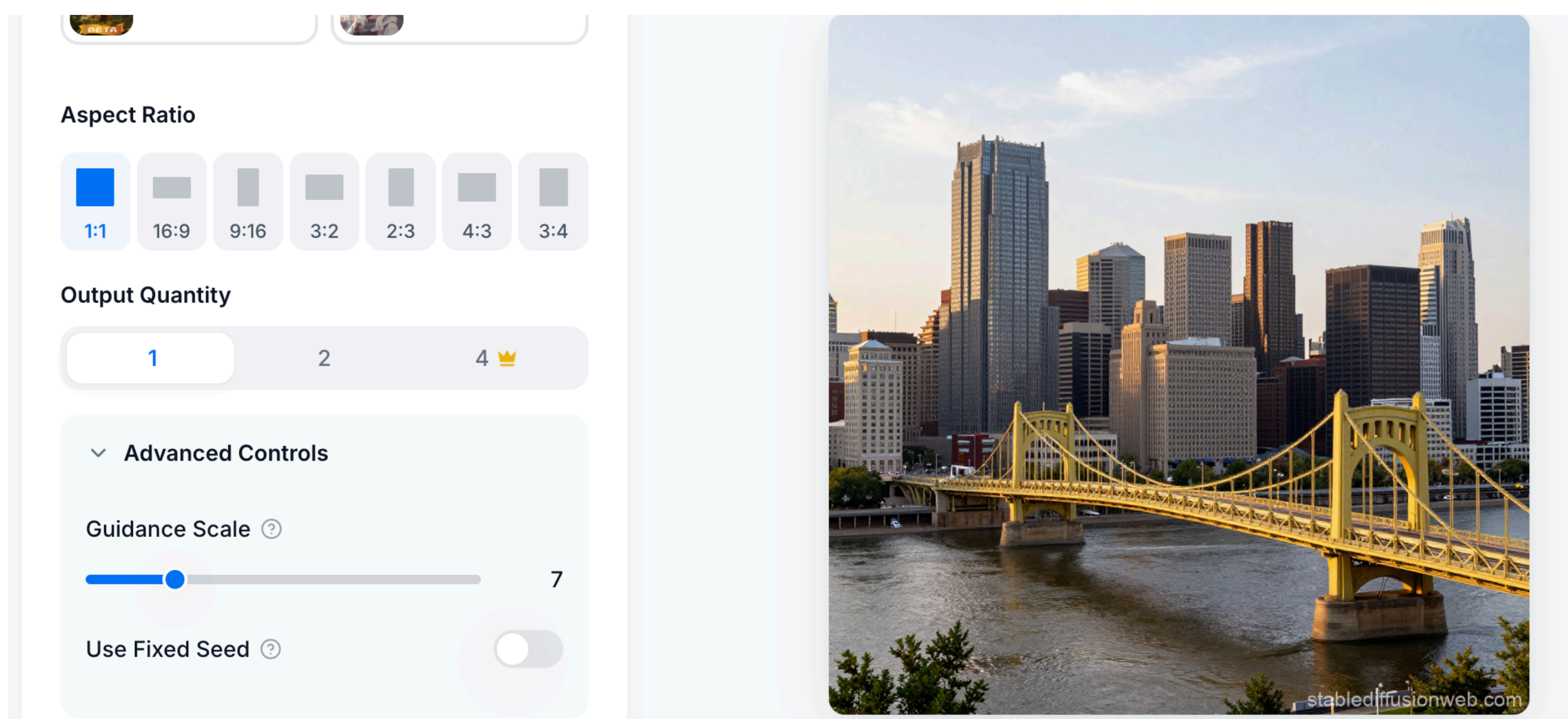


Guidance (w=3.0)

Example: stable diffusion

Impressionist painting of pittsburgh, including its skyline and bridges

realistic photo of pittsburgh, including its skyline and bridges



https://stablediffusionweb.com/app/image-generator

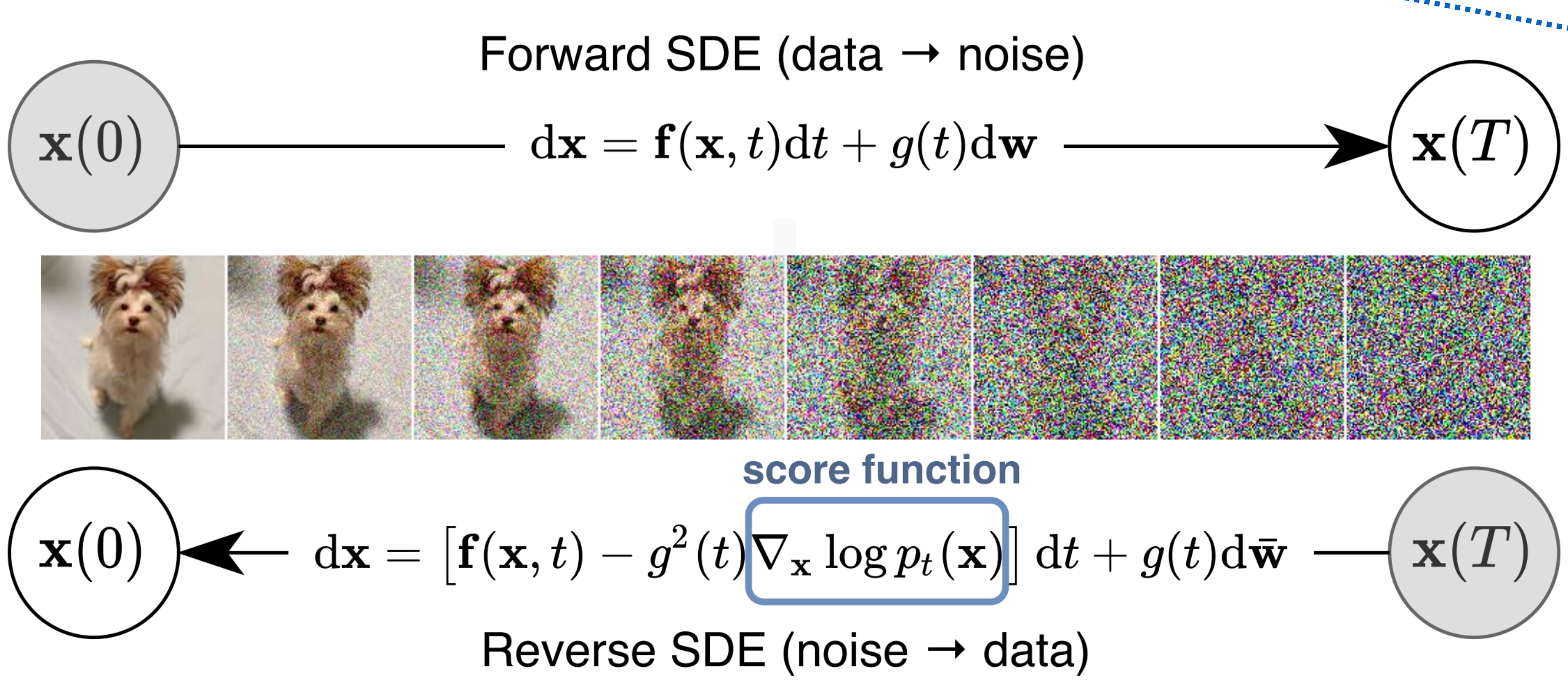
Other extensions

Problem	Solution	Key idea	Key references
Pixel space is expensive	Latent diffusion	Diffuse in an autoencoder’s’s latent space	Rombach et al 2022
Architecture	Diffusion Transformer (DiT)	Transformer replaces U-Net (convolutional)	Peebles & Xie 2022
Stochastic sampling	Deterministic paths and ODE solvers	Deterministic sampling, fewer steps	Song et al 2021
Slow sampling	Distillation	Train a student to match the teacher in fewer steps	Salimans & Ho 2022

The continuous-time view (SDE)

[Song et al 2021]

- If we take the number of steps to ∞, we get a stochastic differential equation (SDE):
    - $dz = f(t)z dt + g(t) dw$
- A classical result states that this SDE can be reversed in time:
    - $dz = [f(t)z - g(t)^2 \nabla_z \log p_t(z)]dt + g(t)d\bar{w}$



Since DDPM's noise predictor estimates the score, we can plug it in

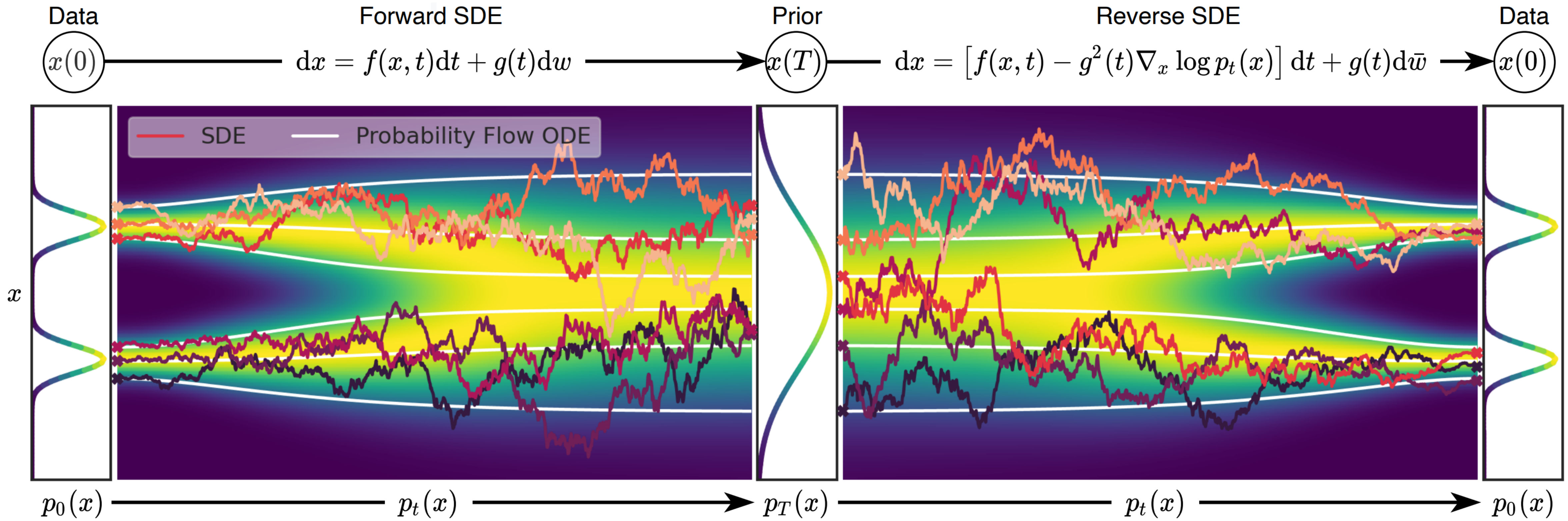
$$\nabla_z \log p_t(z) \approx -\frac{\epsilon_\theta(z, t)}{\sigma_t}$$

The continuous-time view (ODE)

[Song et al 2021]

- **Probability flow ODE**: For any diffusion SDE, there is a deterministic ordinary differential equation (ODE) whose solutions have the same marginals $p_t(z)$:

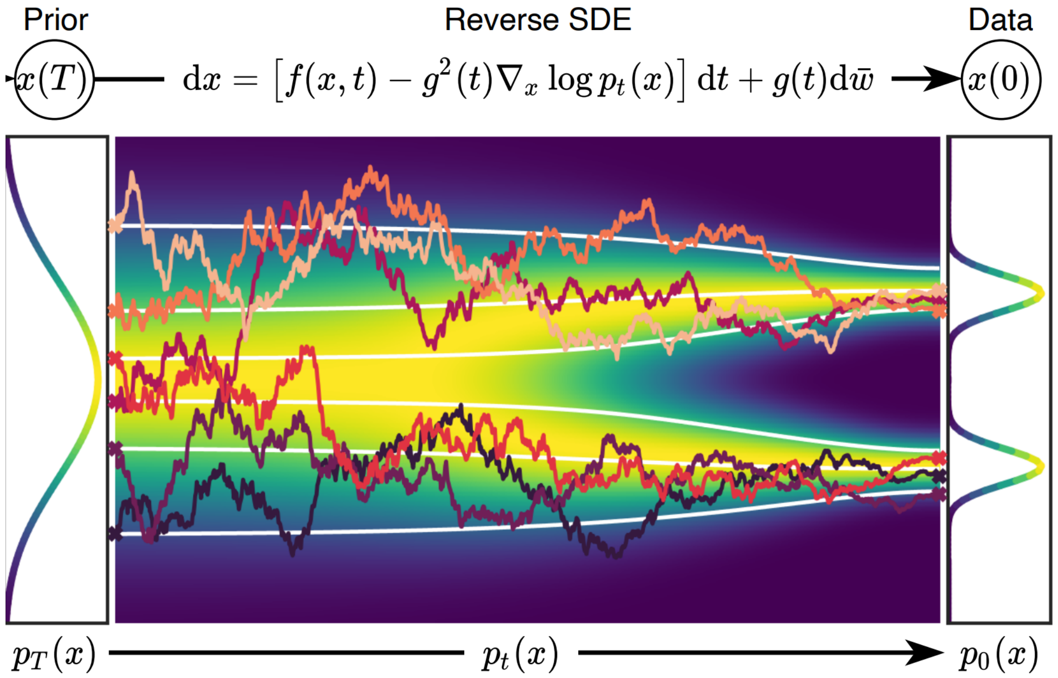
- $$dz = [f(t)z - \frac{1}{2}g(t)^2 \nabla_z \log p_t(z)]dt$$



The continuous-time view (ODE)

[Song et al 2021]

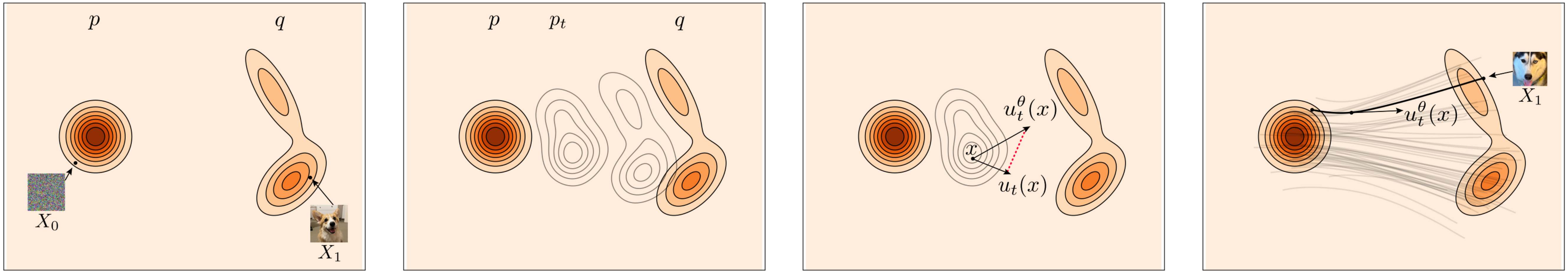
- This view leads to opportunities for better inference
    - Use ODE solvers and related techniques
- It also raises a question: can we design methods that "flow" from noise to data in other ways?



Flows and flow matching

[Lipman et al., 2022; Albergo and Vanden-Eijnden, 2022; Liu et al., 2022]

- Core idea: learn a *velocity* that lets you move from one distribution to another
  - Design a *path* that interpolates between p_0 (e.g., noise) and p_1 (target data distribution)
  - Train model to predict the known velocity along the path
  - To sample, integrate the velocity field from t=0 to 1



(a) Data. **(b)** Path design. **(c)** Training. **(d)** Sampling.

[Lipman et al., 2024]

Flows and flow matching

- Velocity field $v_t : \mathbb{R}^d \to \mathbb{R}^d$
- Flow $\phi_t : \mathbb{R}^d \to \mathbb{R}^d$ defined via the ODE:

$$\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)), \quad \phi_0(x) = x$$

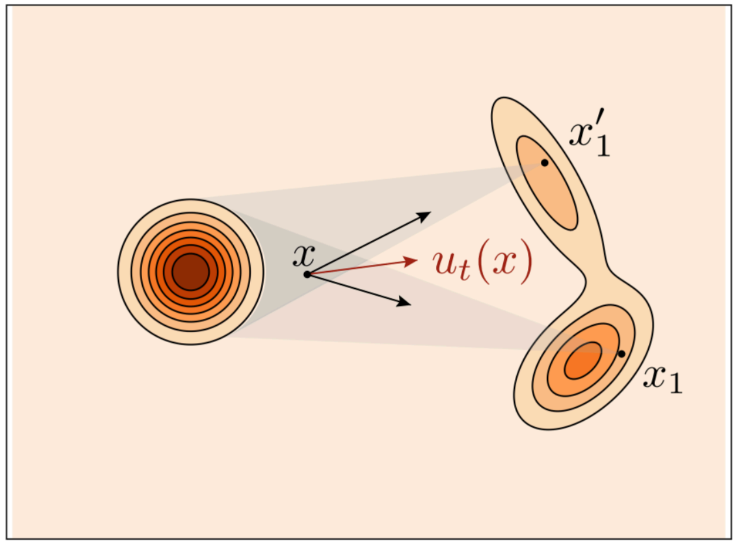
- Starting from $X_0 \sim p_0$ (e.g., noise), the flow transports the sample along the velocity field
- If v_t is chosen correctly, then $X_1 = \phi_1(X_0)$ is a sample from $p_1 = p_{\text{data}}$
  - Hence we want to learn $v_0 \approx v_t$

Flows and flow matching

- A natural choice might be:

$$L_{FM} = \mathbb{E}_{t,X_t} \| v_\theta(X_t, t) - v_t(X_t) \|^2$$

- Since we don't know v_t, define a **conditional path** for each single data point x_1:
  - Simple straight line path: $X_{t|1} = (1-t)X_0 + tx_1$
  - Conditional velocity: $\frac{d}{dt} X_{t|1} = x_1 - X_0$



- Loss:

$$L_{CFM} = \mathbb{E}_{t,x_1,X_0} \| v_\theta(X_t, t) - (x_1 - X_0) \|^2$$

where $X_t = (1-t)X_0 + tx_1$.

Key result: L_{CFM} gives the same optimal v_θ as L_{FM}!

Comparison with DDPM

	DDPM	**Flow Matching**
Path	Curved (from diffusion)	Straight line
Process	Stochastic (SDE)	Deterministic (ODE)
Model predicts	Noise $\boldsymbol{\epsilon}$	Velocity $v = \mathbf{x}_1 - X_0$
Loss	$\|\boldsymbol{\epsilon}_\theta - \boldsymbol{\epsilon}\|^2$	$\|v_\theta - (\mathbf{x}_1 - X_0)\|^2$
Coefficients	$\bar{\alpha}_t$, α_t, etc.	None
Typical steps	$\sim$1000 (or $\sim$50 with tricks)	$\sim$20–50

- Flow matching directly learns an ODE with straight paths that are easier to integrate in fewer steps

Case Study: Stable Diffusion 3

[Esser et al 2024]

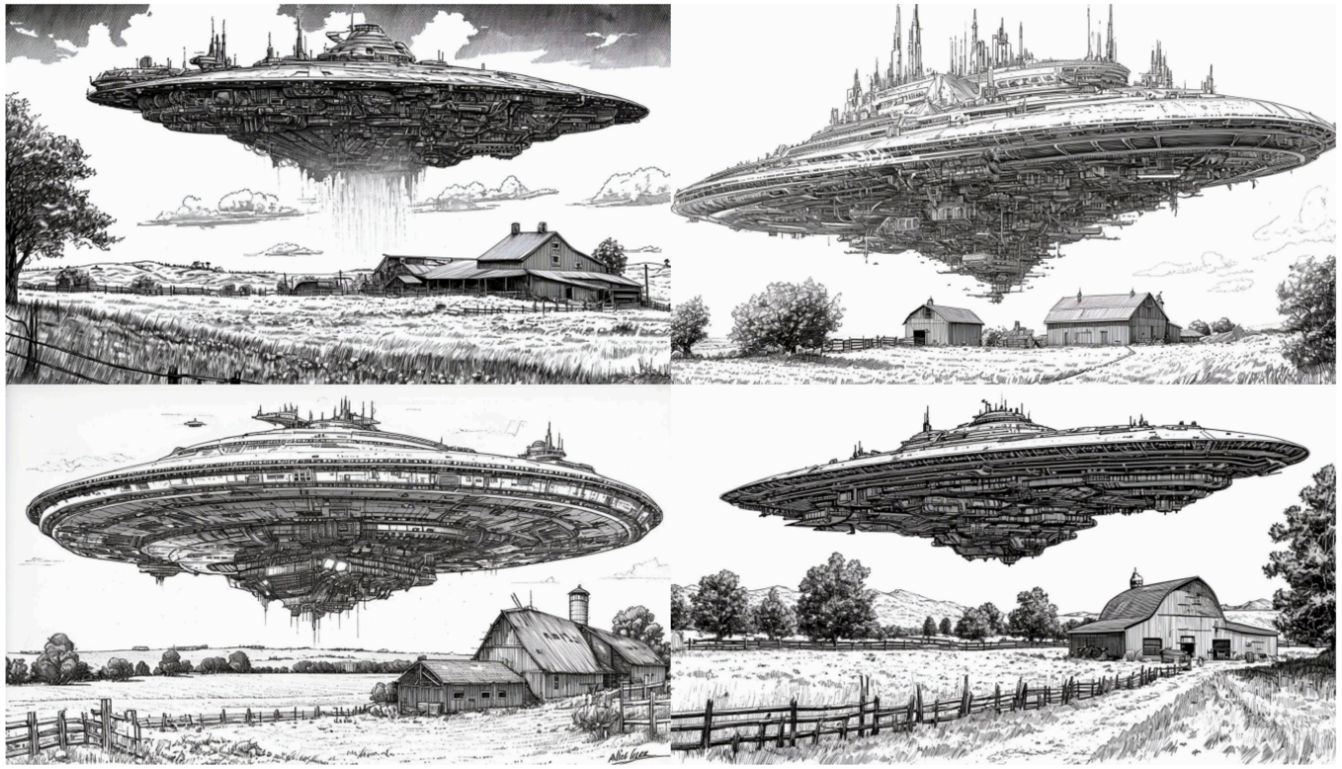
- Flow Matching: sample intermediate time steps more frequently
- Latent: model the latent space of a pretrained auto encoder
- Text: concatenate CLIP text-encoder vector and T5 vectors

Flow-matching variants

DDPM variant

	rank averaged over		
variant	all	5 steps	50 steps
rf/lognorm(0.00, 1.00)	1.54	1.25	1.50
rf/lognorm(1.00, 0.60)	2.08	3.50	2.00
rf/lognorm(0.50, 0.60)	2.71	8.50	1.00
rf/mode(1.29)	2.75	3.25	3.00
rf/lognorm(0.50, 1.00)	2.83	1.50	2.50
eps/linear	2.88	4.25	2.75
rf/mode(1.75)	3.33	2.75	2.75
rf/cosmap	4.13	3.75	4.00
edm(0.00, 0.60)	5.63	13.25	3.25
rf	5.67	6.50	5.75
v/linear	6.83	5.75	7.75
edm(0.60, 1.20)	9.00	13.00	9.00
v/cos	9.17	12.25	8.75
edm/cos	11.04	14.25	11.25
edm/rf	13.04	15.25	13.25
edm(-1.20, 1.20)	15.58	20.25	15.00

Table 1. **Global ranking of variants.** For this ranking, we apply non-dominated sorting averaged over EMA and non-EMA weights, two datasets and different sampling settings.



detailed pen and ink drawing of a massive complex alien space ship above a farm in the middle of nowhere.

photo of a bear wearing a suit and tophat in a river in the middle of a forest holding a sign that says "I cant bear it".

tilt shift aerial photo of a cute city made of sushi on a wooden table in the evening.

dark high contrast render of a psychedelic tree of life illuminating dust in a mystical cave.

Today's lecture

- Diffusion basics
- Extensions:
  - Guidance
  - Continuous-time view
- Flow matching

Thank you!