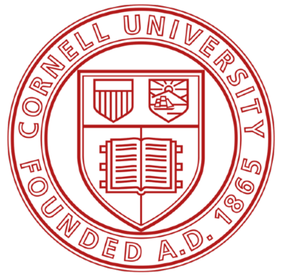
Lecture 20: Midterm review

CS 5788: Introduction to Generative Models



Midterm info

- Midterm on Fri. 4/17, 2-5pm in auditorium.
- You'll have 3 hours.
- Mostly multiple choice, plus some written problems.
- Compared to the homework: more conceptual questions
- 1-page front and back hand-written single-sided "cheat sheet"
- No calculators (also unnecessary).
- No questions directly on:
  - Lec. 4 (neural network review)
  - Lec. 11 (generative models for computer vision guest lecture)

Today

- Practice questions
- Plus quick reviews of the course material behind each one
- Not exhaustive!

PS4

- Implement a transformer.
- Use it to generate text using pretrained LLM.
- Perform post-training using Direct Preference Optimization (DPO), which we'll discuss soon.

Sample question

Which of the following is a maximum likelihood estimate for the mean of a multivariable Gaussian distribution:

$$p_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

Mark all that apply.

A. The vector $\boldsymbol{\mu}$ that maximizes: $\prod_i p_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\boldsymbol{x}_i)$.

B. The average of all the examples in the dataset.

C. The vector $\boldsymbol{\mu}$ for which $\nabla_{\boldsymbol{\mu}} \left[\frac{1}{N}\sum_{i=1}^{N} \log\left(p_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\boldsymbol{x}_i)\right)\right] = 0.$

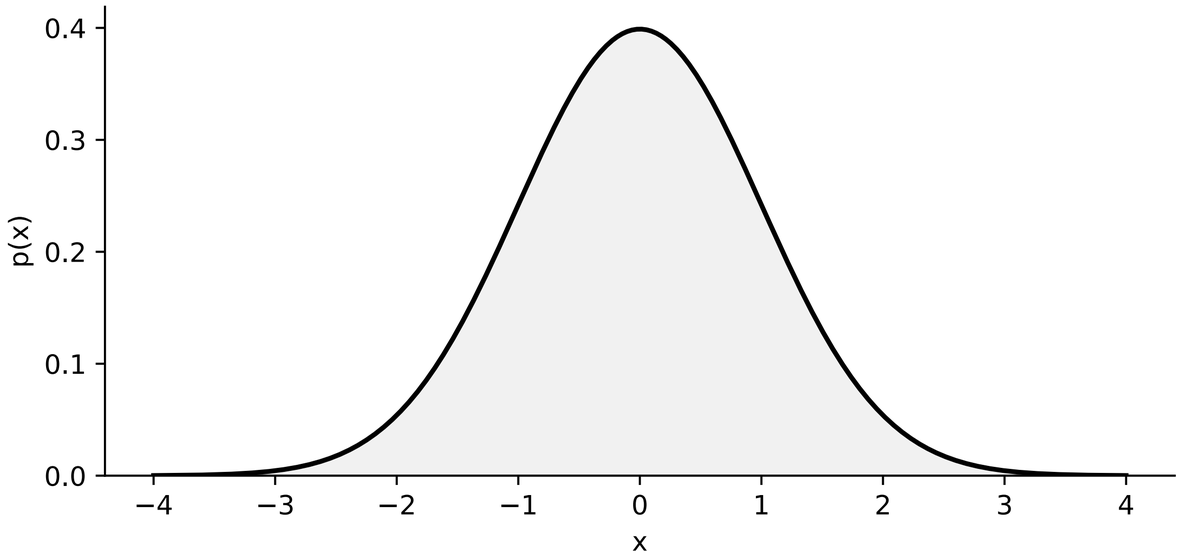
D. The vector $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{T} \boldsymbol{\Sigma}^{-1}\mathbf{x}_{i'}$ given the MLE estimate for the covariance $\boldsymbol{\Sigma}$.

Gaussian distribution

Probability density function (pdf):

$$p_\theta(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Parameters θ: mean μ, standard deviation σ.



Example: $\mu = 0, \sigma = 1$

Maximum likelihood estimation (MLE)

Goal: Find best parameters θ.

How do we quantify this? One option is **maximum likelihood estimation**.

Suppose we have a sample of points $x_1, x_2, \ldots, x_N$ from the distribution. We can find θ by maximizing their likelihood under the model:

$$\text{argmax}_\theta \prod_{i=1}^{N} p_\theta(x_i)$$

Assuming data is **iid** (independent and identically distributed).

$$= \text{argmax}_\theta \sum_{i=1}^{N} \log(p_\theta(x_i))$$

Equivalent to *minimizing* the **negative log likelihood:** $\text{NLL}(\theta) = -\sum_{i=1}^{N} \log(p_\theta(x_i))$

MLE for a Gaussian

$$\text{argmax}_\theta \sum_{i=1}^{N} \log(p_\theta(x_i))$$

Gaussian pdf

$$= \text{argmax}_{\mu,\sigma} \sum_{i=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\right)$$

$$= \text{argmax}_{\mu,\sigma} \sum_{i=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + -\frac{(x_i-\mu)^2}{2\sigma^2}$$

How do we solve for μ and σ? Take derivatives and set to 0!

MLE for a Gaussian

Let's set $\frac{\partial}{\partial \mu} \sum_{i=1}^{N} \log(p_\theta(x_i)) = 0$ and solve for μ

$$0 = \frac{\partial}{\partial \mu} \sum_{i=1}^{N} \log(p_\theta(x_i)) = \sum_{i=1}^{N} \frac{\partial}{\partial \mu} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + -\frac{(x_i - \mu)^2}{2\sigma^2}$$

$$= \sum_{i=1}^{N} \frac{(x_i - \mu)}{\sigma^2} = \frac{1}{\sigma^2}\left[\sum_{i=1}^{N} x_i - N\mu\right] \quad \Longrightarrow \quad \mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Similarly, can show $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$

Sample question

As you add more mixture components to a Gaussian mixture model (GMM), the log likelihood for the training examples:

A. Goes up
B. Goes down
C. Stays the same
D. There’s no clear trend

Sample question

When fitting a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ to a dataset consisting of flattened image patches, the covariance matrix can be decomposed via eigendecomposition as $\Sigma = U\Lambda U^T$. What do the column vectors of the matrix U represent in the context of the image data?

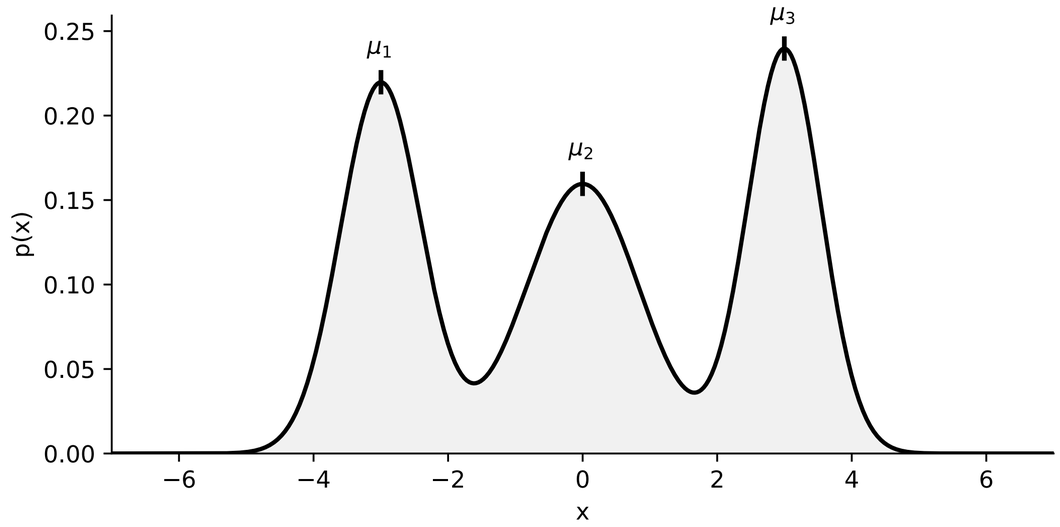
A. The mean pixel intensity values of the localized patches.
B. The individual pixel variances across the entire dataset independent of spatial correlation.
C. The categorical cluster assignment probabilities for each individual patch.
D. They capture the primary structural directions of variance within the image patches.

Gaussian mixture model (GMM)

A useful type of mixture model that mixes multiple Gaussian distributions.

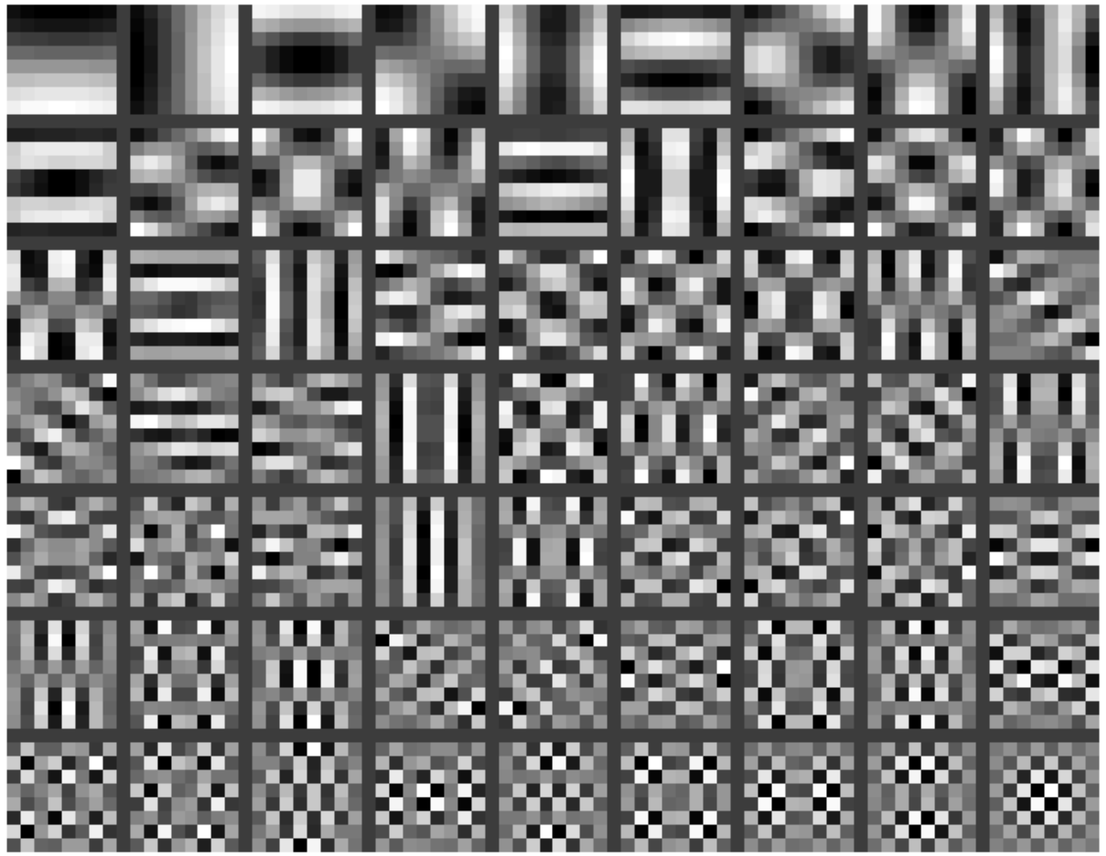
Density function: $p_\theta(\mathbf{x}) = \sum_{i=1}^{K} \pi_i \mathcal{N}(\mathbf{x}; \boldsymbol{\mu_i}, \boldsymbol{\Sigma}_i)$

where π_i is the probability of choosing mixture component i, and $\mathcal{N}$ is the Gaussian pdf with mixture-specific mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

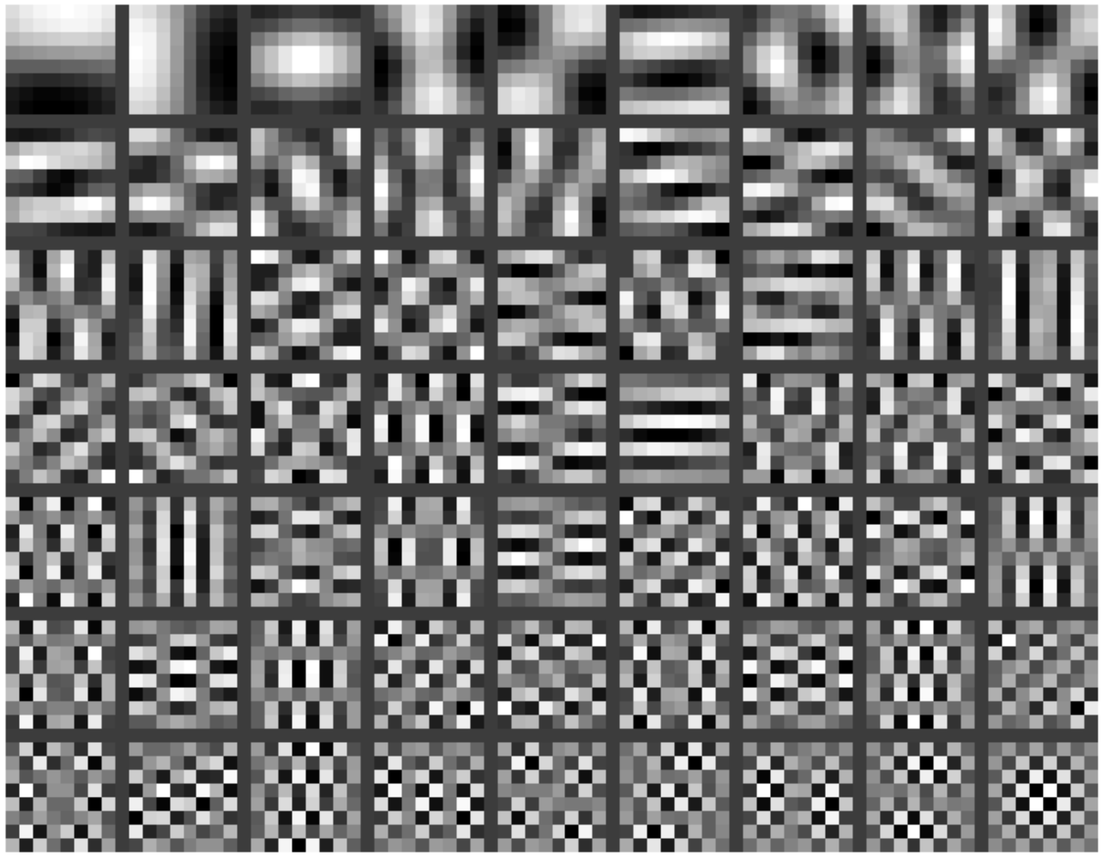


Eigenvectors for 2-component GMM

$\pi_1 = 0.5611$



$\pi_2 = 0.4389$



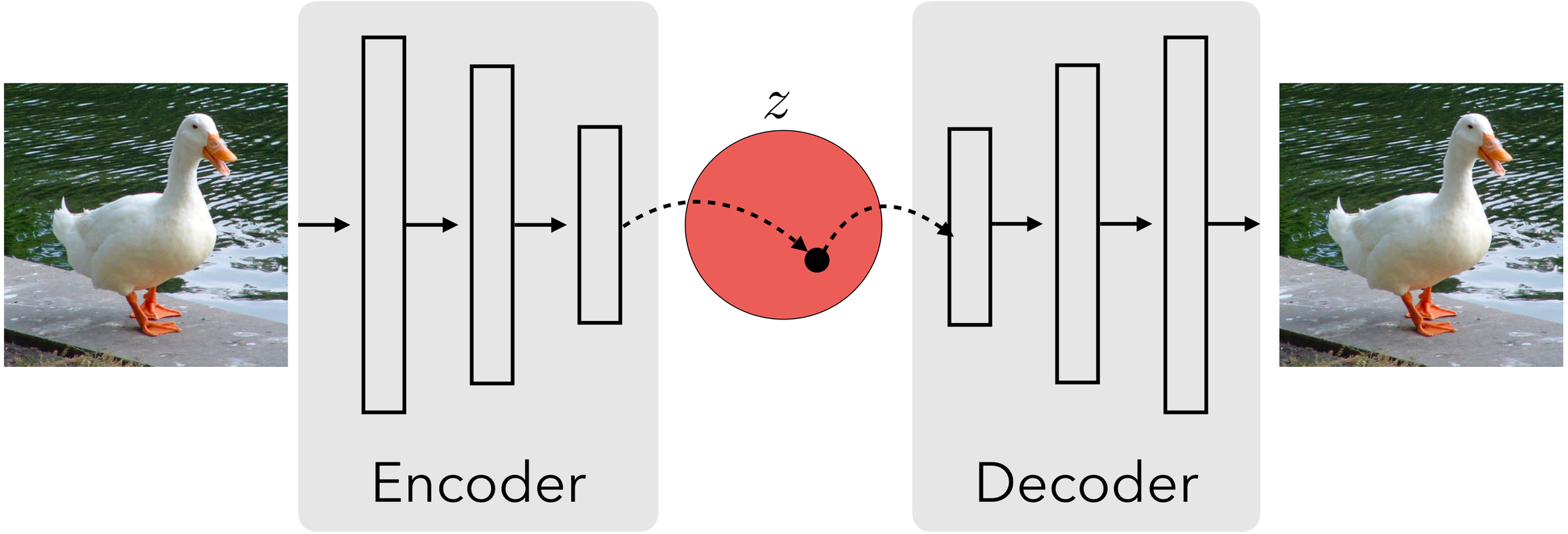
Source: [Zoran & Weiss, 2012]

Sample question

After training a VAE, you can discard the encoder if your only goal is to generate samples.

A. True
B. False

Encoder-decoder models



Assign each image a vector $\mathbf{z}$ that can be decoded to reconstruct the image.

Sample question

Consider the derivation of the VAE objective.

$$
\begin{aligned}
\log p(\mathbf{x}) &= \log \int p(\mathbf{x}, \mathbf{z})\, d\mathbf{z} \\
&= \log \int q(\mathbf{z} \mid \mathbf{x}) \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})}\, d\mathbf{z} \\
&\dots
\end{aligned}
$$

Why do you introduce $q(z \mid x)$ in this step?

Sample question

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x} \mid \mathbf{z})] \; - \; D_{\mathrm{KL}}(q(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z}))$$

The following is the objective function for training a variational autoencoder. Explain the similarities (and dissimilarities) to an autoencoder's training loss.

Evidence lower bound

$$\begin{aligned}
\log p(\mathbf{x}) &= \log \int p(\mathbf{x}, \mathbf{z})\, d\mathbf{z} \\
&\geq \int q(\mathbf{z} \mid \mathbf{x})\, \log p(\mathbf{x} \mid \mathbf{z})\, d\mathbf{z} - \int q(\mathbf{z} \mid \mathbf{x})\, \log \frac{q(\mathbf{z} \mid \mathbf{x})}{p(\mathbf{z})}\, d\mathbf{z} \\
&= \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x} \mid \mathbf{z})] - D_{\mathrm{KL}}(q(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z}))
\end{aligned}$$

Reconstruction loss:

$\mathscr{L}_R = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$ (up to scale factor)

Equivalent to log likelihood of $\mathcal{N}(\hat{\mathbf{x}}, I)$: i.e., how likely is $\mathbf{x}$ under a Gaussian with mean $\hat{\mathbf{x}}$?

KL divergence loss:

$\mathscr{L}_P = \mathrm{D}_{KL}\left(q(\mathbf{z} \mid \mathbf{x}) \parallel \mathcal{N}(0, I)\right)$

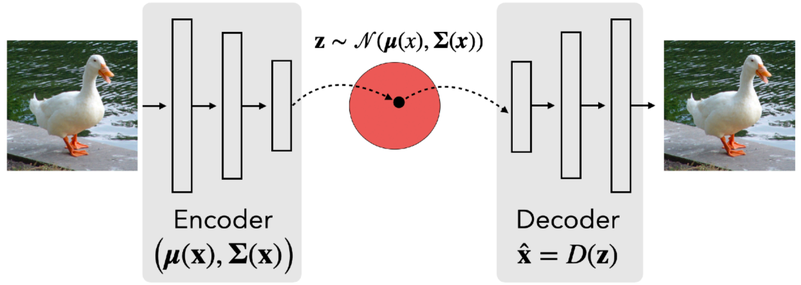
Make $\mathbf{z}$ to match our desired source distribution.

(when we negate this and minimize, rather than maximize)

A closer look

$$\log p(\mathbf{x}) = \log \int p(\mathbf{x}, \mathbf{z})\, d\mathbf{z}$$

$$\geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x} \mid \mathbf{z})] - D_{\mathrm{KL}}(q(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z}))$$



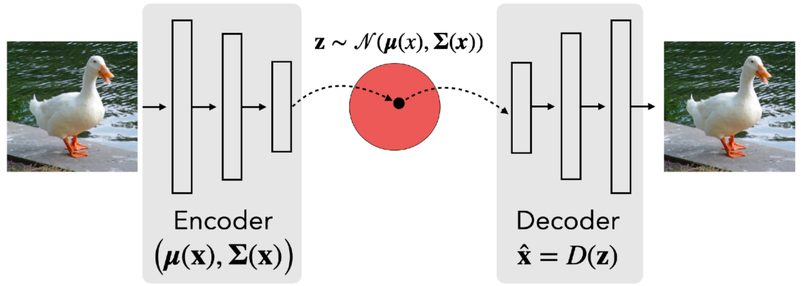
This term describes an *autoencoder*! It says:

1. Choose a random example $\mathbf{x}$ from the dataset.
2. Sample $\mathbf{z}$ given $\mathbf{x}$ using the encoder, $q(\mathbf{z} \mid \mathbf{x})$
3. Decode $\hat{\mathbf{x}}$ and compare it to $\mathbf{x}$, which corresponds to computing $\log p(\mathbf{x} \mid \mathbf{z})$

A closer look

$$\log p(\mathbf{x}) = \log \int p(\mathbf{x}, \mathbf{z})\, d\mathbf{z}$$

$$\geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x} \mid \mathbf{z})] - D_{\mathrm{KL}}(q(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z}))$$



KL divergence. Assuming that $p(\mathbf{z})$ is Gaussian:

1. How far is the latent space that our encoder produces from a Gaussian?
2. Analytical solution if $p(\mathbf{z})$ is Gaussian.

Sample question

Suppose that $\mathbf{X} = g(\mathbf{Z})$ where $p_Z(z) = 2z$ in the range $z \in (0,1)$ and $g(z) = 10z + 5$. What is the density of $\mathbf{X}$?

Change of variables formula

Volume correction

If $\mathbf{X} = g(\mathbf{Z})$, then $p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}\left(g^{-1}(\mathbf{x})\right) \left|\det \mathrm{D}g^{-1}(\mathbf{x})\right|$

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}\left(f(\mathbf{x})\right) \left|\det \mathrm{D}f(\mathbf{x})\right| \text{ for } f \triangleq g^{-1}$$

where $\mathrm{D}f$ is the **Jacobian**, i.e., the matrix of partial derivatives:

$$\mathrm{D}f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}$$

We say that $p_{\mathbf{X}}$ is the **pushforward** of $p_{\mathbf{Z}}$ by g.

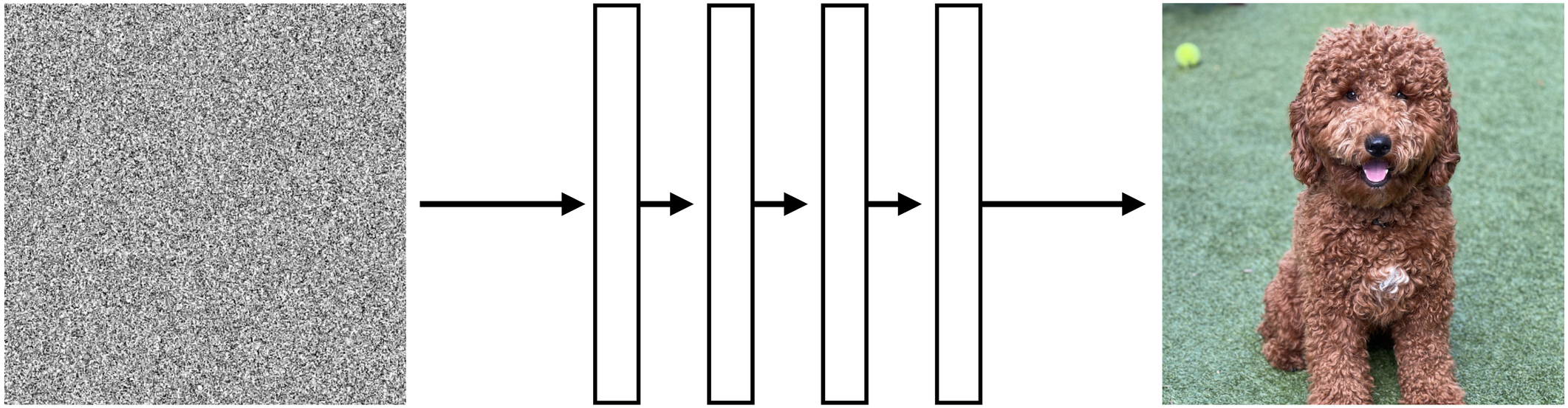
Sample question

True or false: We can implement a normalizing flow model as a U-net that maps image-shaped Gaussian vector $\mathbf{z}$ into an image.

Sample question

True or false: we can put $\tanh$ layers in a normalizing flow model.

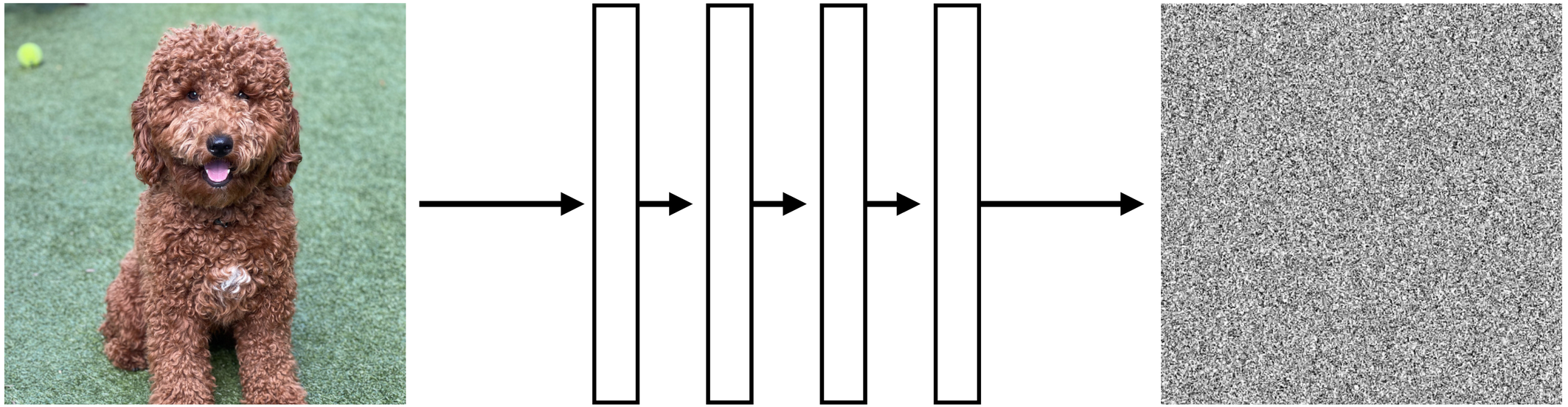
Normalizing flow model



$$g_N \circ g_{N-1} \circ \cdots \circ g_1(\mathbf{z})$$

Normalizing flow model

For estimating the density and learning:



$$\log p_\theta(\mathbf{X}) = \sum_{i=1}^{M} \log p_{\mathbf{Z}}(f(\mathbf{x}_i)) + \log \left| \det(\mathrm{D}f(\mathbf{x}_i)) \right|$$ where $\mathbf{x}_i$ is a training example.

Exact likelihood estimation (unlike VAE, which uses lower bound).

Sample question

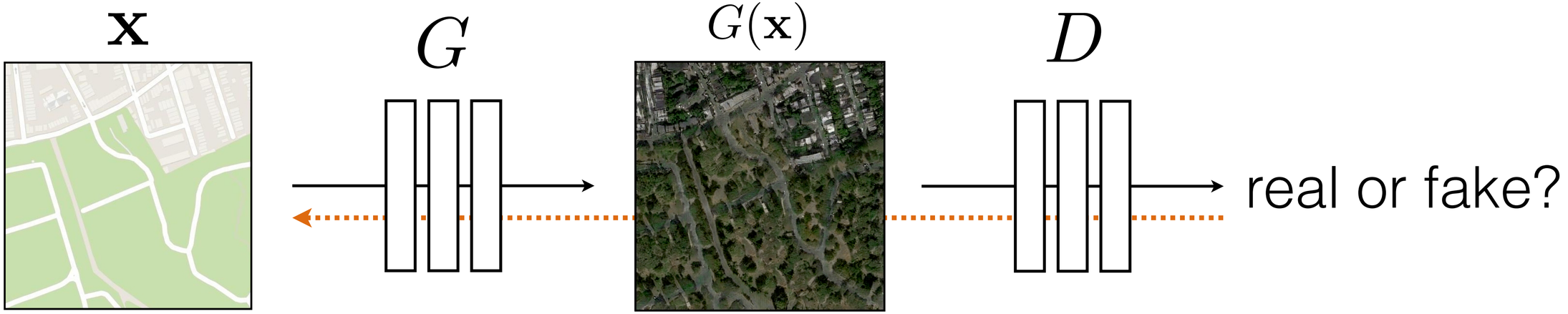
True or false: in a GAN that performs image-to-image translation, the discriminator generally needs the input image to be one of its inputs.

G tries to synthesize fake images that fool **D**

D tries to identify the fakes

$$\arg\max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\, \log D(G(\mathbf{x})) \; + \; \log(1 - D(\mathbf{y}))\,]$$

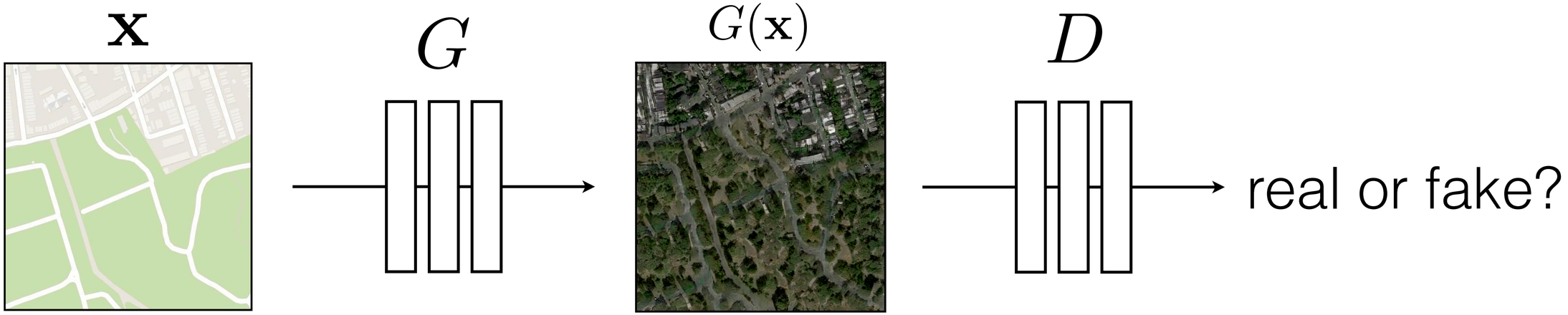
Source: Isola, Freeman, Torralba



G tries to synthesize fake images that ***fool*** **D**:

$$\arg \min_{G} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\ ]$$

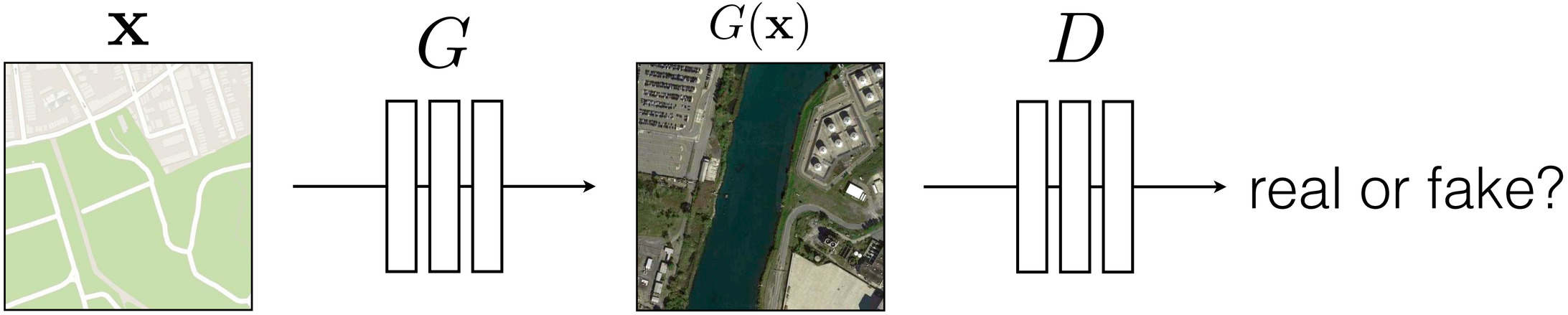
Source: Isola, Freeman, Torralba



G tries to synthesize fake images that ***fool*** the ***best*** **D**:

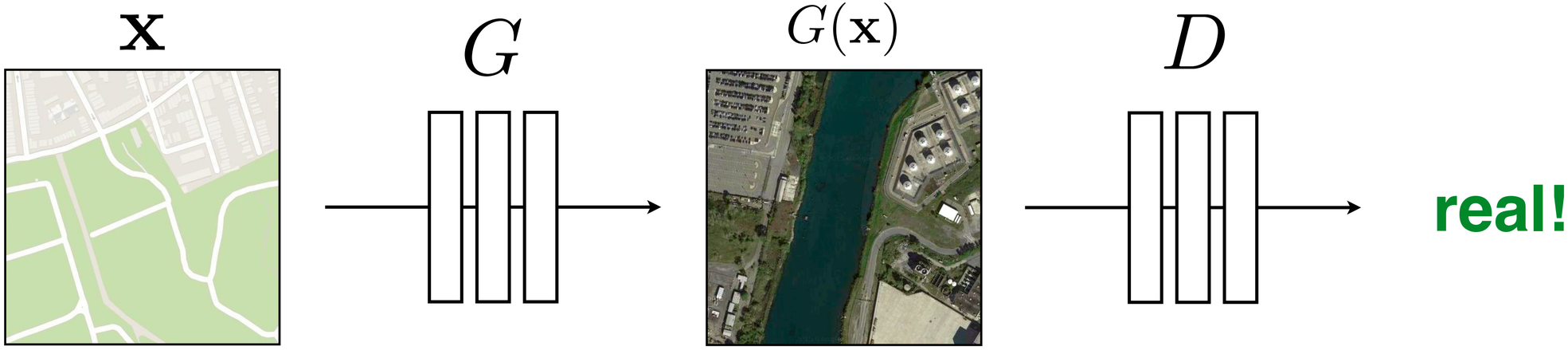
$$\arg \min_G \max_D \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\ ]$$

Source: Isola, Freeman, Torralba



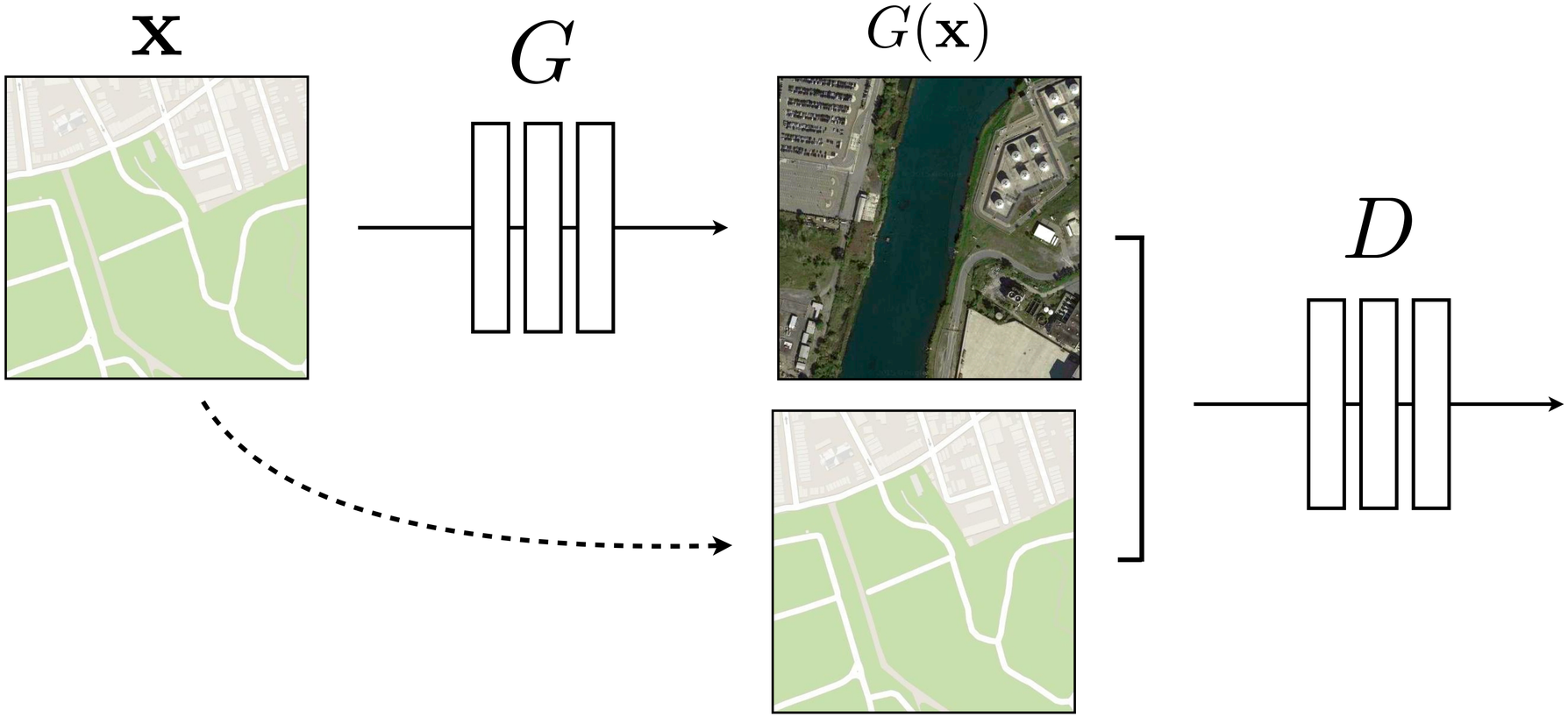
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y})) \;\;]$$

Source: Isola, Freeman, Torralba



$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y})) \;\;]$$

Source: Isola, Freeman, Torralba



$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\;]$$

Source: Isola, Freeman, Torralba

Sample question

In the Generative Adversarial Network (GAN) value function for the minimax game:

$V(G, D) = \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{x \sim p_g}[\log(1 - D(x))]$,

the optimal discriminator $D^*(x)$ is:

A. $D^*(x) = \dfrac{p_{data}(x)}{p_{data}(x) + p_g(x)}$

B. $D^*(x) = \dfrac{p_g(x)}{p_{data}(x)}$

C. $D^*(x) = 0.5$

D. $D^*(x) = 1$

What is the optimal discriminator?

$$D^*(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})}$$

where p_{data} is the data distribution and p_g is the generator's distribution (i.e., the distribution of $G(\mathbf{z})$)

Given a generator G, we are maximizing:

$$V(G, D) = \int_{\mathbf{x}} p_{data}(\mathbf{x}) \log(D(\mathbf{x})) d\mathbf{x} + \int_{\mathbf{z}} p_z(\mathbf{z}) \log(1 - D(G(\mathbf{z}))) d\mathbf{z}$$

$$= \int_{\mathbf{x}} p_{data}(\mathbf{x}) \log(D(\mathbf{x})) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) d\mathbf{x}$$

How do you maximize this? For some (a, b), consider:

$$f(y) = a \log y + b \log(1 - y) \qquad \text{set} \quad \frac{df}{dy} = \frac{a}{y} - b\frac{1}{1 - y} = 0$$

$$\implies \frac{1 - y}{y} = \frac{b}{a} \implies \frac{1}{y} = 1 + \frac{b}{a} \implies y = \frac{a}{a + b}$$

Sample question

True or false: we can learn an energy-based model by *minimizing*:

$$\mathscr{L} = \sum_{i=1}^{N} E_\theta(\mathbf{x}_i)$$

for a dataset of N examples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$.

Sample question

True or false: we can learn an energy-based model by *minimizing*:

$$\mathscr{L} = \sum_{i=1}^{N} E_\theta(\mathbf{x}_i)$$

for a dataset of N examples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$.

Energy-based model

Energy function implemented by neural net, $E_\theta(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z(\theta)}$$

Normalization constant
a.k.a. *partition function*

where $Z(\theta) = \int_{\mathbf{x}} \exp(-E_\theta(\mathbf{x}))d\mathbf{x}$

Normalizing constant

$$Z(\theta) = \int_{\mathbf{x}} \exp(-E_\theta(\mathbf{x}))d\mathbf{x}$$

To compute $Z(\theta)$, we would need to integrate over the full input space $\mathbb{R}^d$! This would be very expensive.

Sample question

Classifier-free guidance (*select all that apply)*:

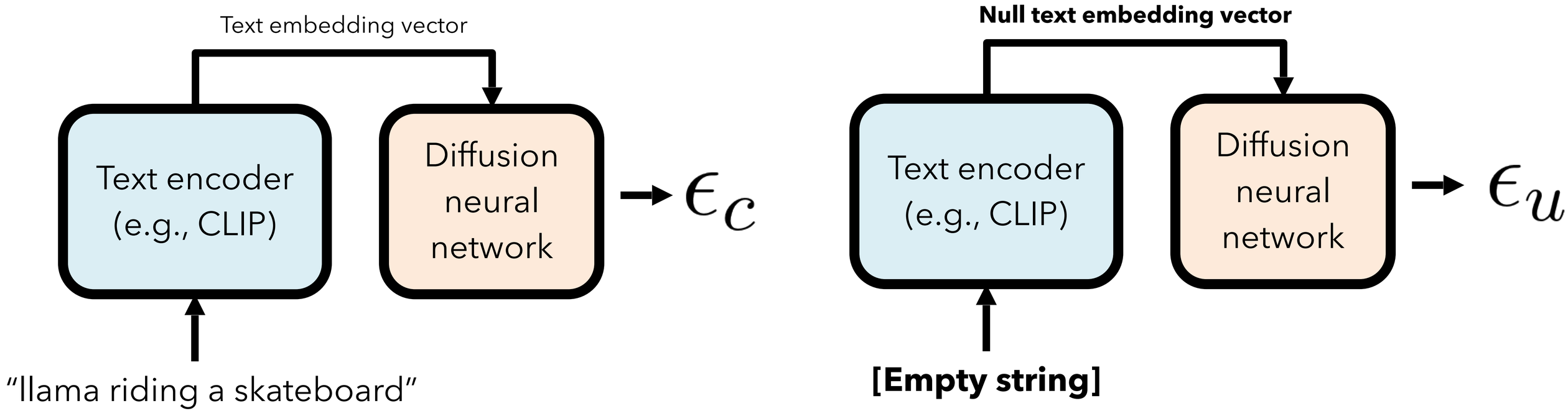
A. Is just as efficient as unconditional generation at test time.

B. Requires training the diffusion model to accept the conditioning signal as an extra input.

C. Requires training the diffusion model to go without any conditioning signal.

D. Leads images that are not very diverse, if too much weight is placed on the conditional noise estimate.

Classifier-free guidance (CFG)



Mix conditional and unconditional noise:

$$\epsilon = \epsilon_u + \gamma(\epsilon_c - \epsilon_u) \quad \text{where } \gamma > 1$$

Classifier-free Guidance

Low γ High γ

Figure source: [Ho and Salimans, 2021]

Sample question

Spot the bug in this diffusion training code:

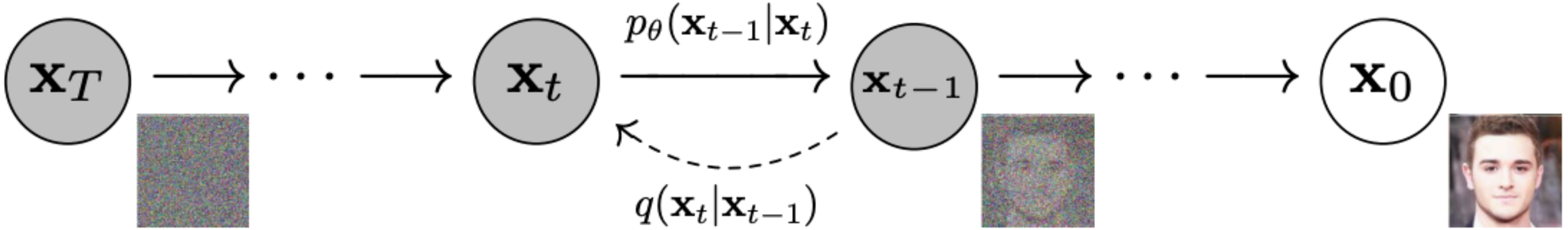
Algorithm 1 Training

1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad$ **for** $t = T, T-1, \ldots, 1$ **do**
4: $\quad\quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad\quad$ Take gradient descent step on

$$\nabla_\theta \left\| \epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon,\ t\right)\right\|^2$$

6: $\quad$ **end for**
7: **until** converged

Training a diffusion model



Algorithm 1 Training

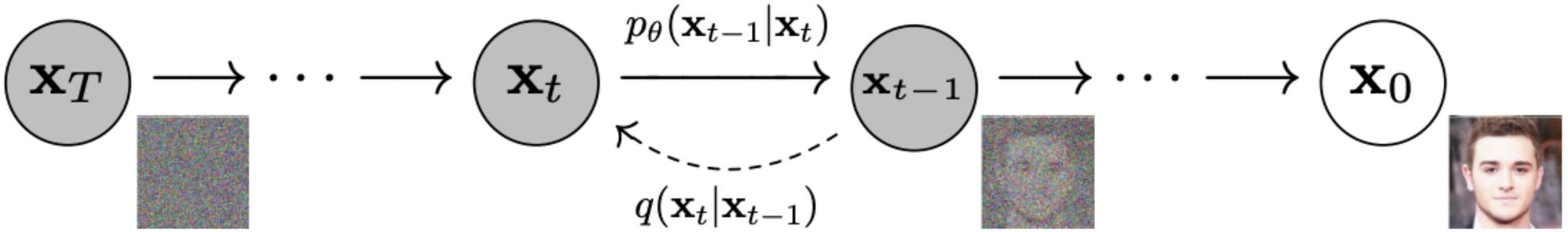
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Training a diffusion model



Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

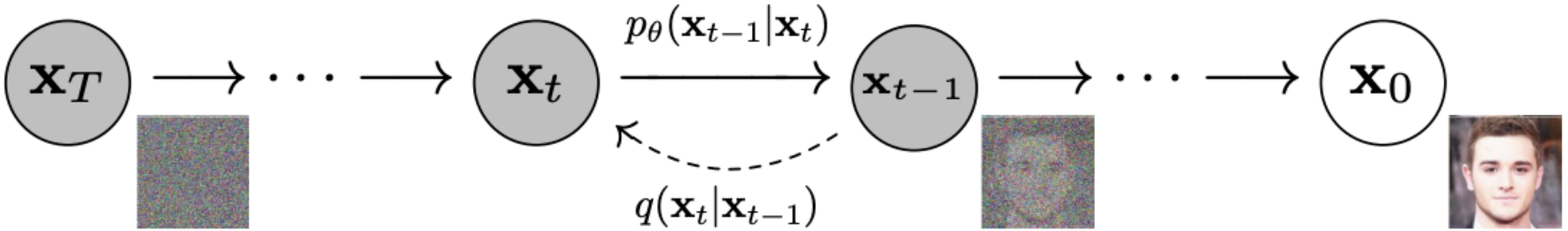
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

Sample x_t by adding noise to a clean image x_0

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Training a diffusion model



Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

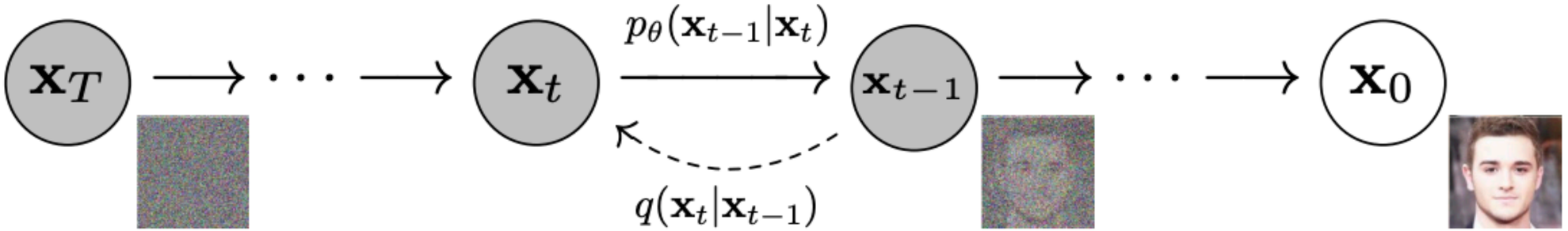
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

Random noise

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Training a diffusion model



Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

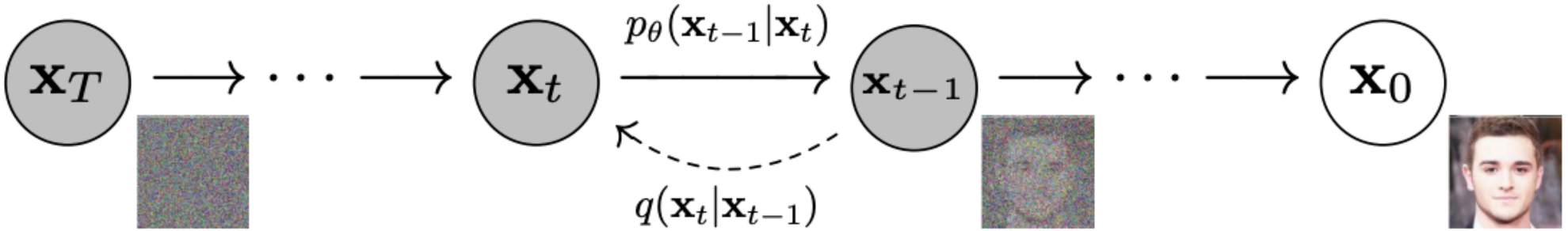
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

Noisy image

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Training a diffusion model



Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2$$

6: **until** converged

Diffusion model ϵ_θ predicts the noise ϵ

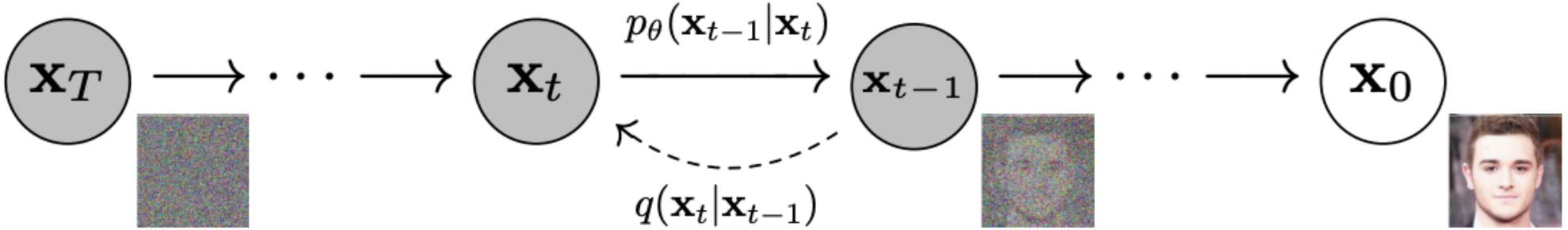
Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Sample question

Write the pseudocode for sampling from a diffusion model, using the standard DDPM formulation we saw in class.

Don’t worry about the constants (you may simply explain what each such constant is doing and assign it a variable name).

Sampling from a diffusion model



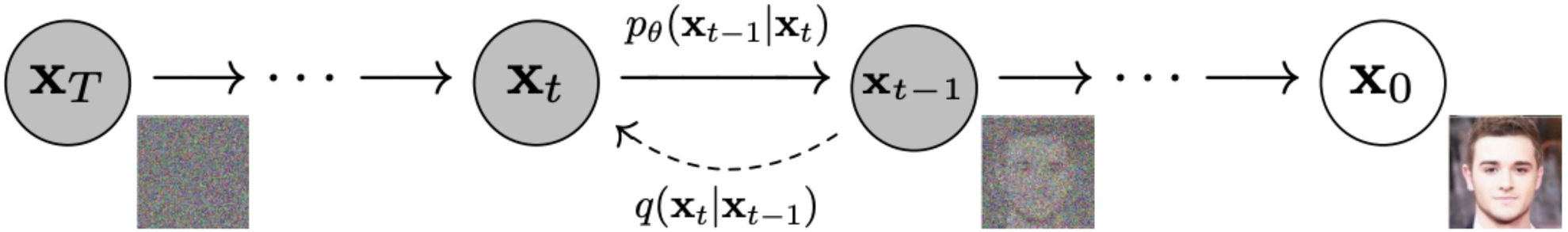
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Remove a little bit of the noise

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Sampling from a diffusion model



Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Add back a bit of noise, too.
(some diffusion variants don't do this)

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Sample question

True or false: diffusion models can be used to fill holes in images without additional training

Inpainting (hole filling)

Input

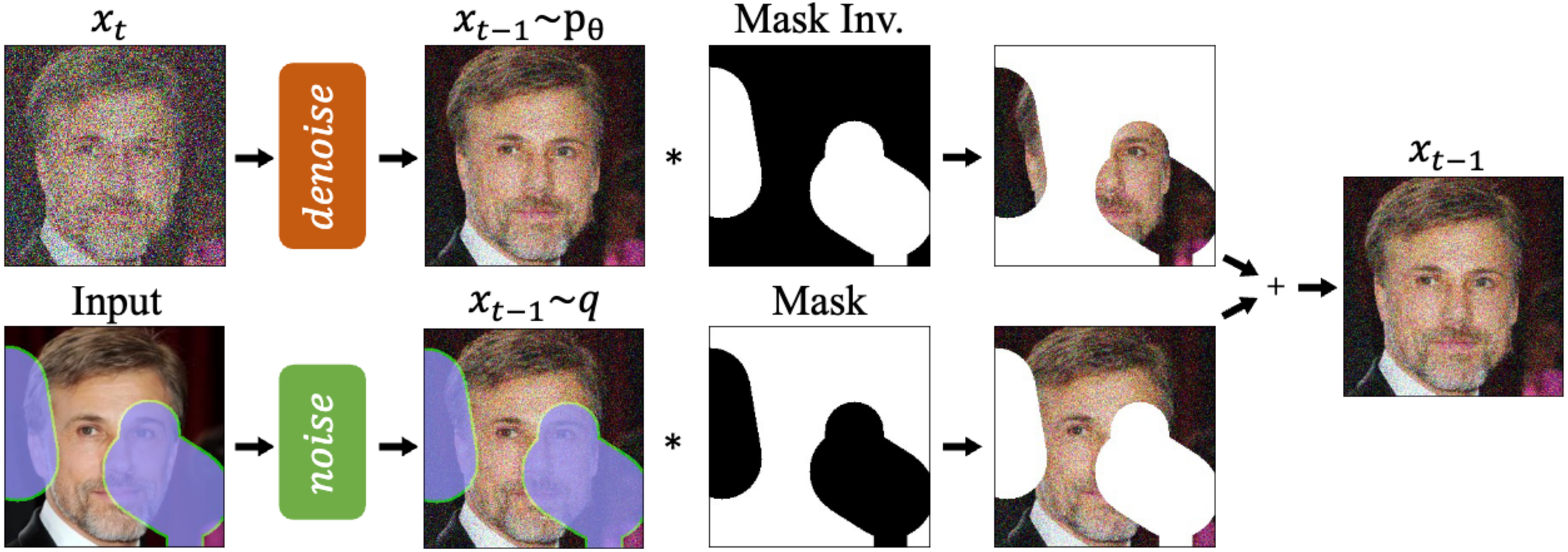
Samples

Do we need a new model for this?

Source: [Lugmayr et al., RePaint]

Zero-shot inpainting

Constrain the pixels each denoising iteration.



Source: [Lugmayr et al., RePaint]

Zero-shot inpainting

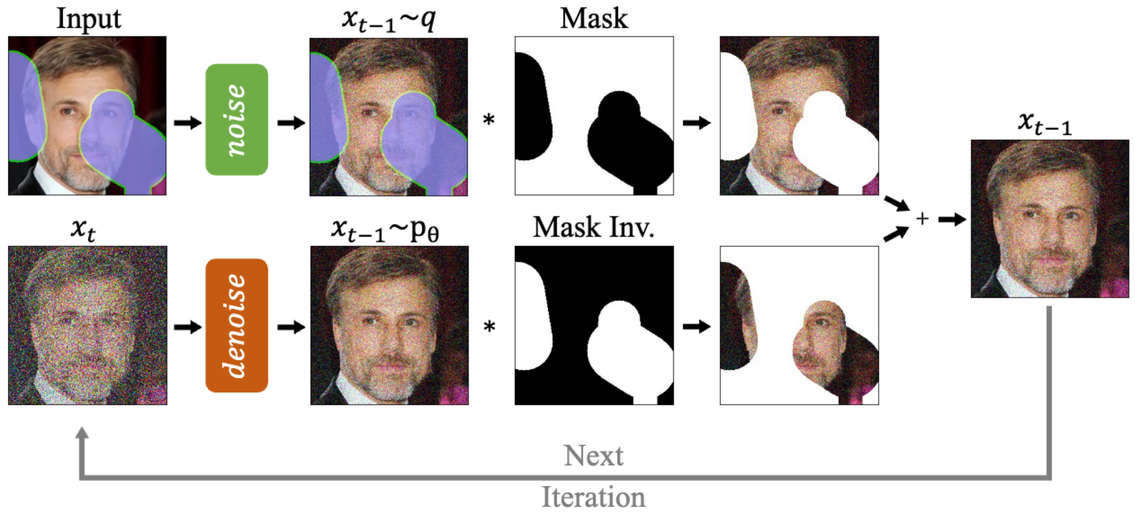
Pseudocode:

$$x_{t-1}^{\text{unknown}} \sim \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

$$x_{t-1}^{\text{known}} \sim \mathcal{N}(\sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)\mathbf{I})$$

$$x_{t-1} = m \odot x_{t-1}^{\text{known}} + (1 - m) \odot x_{t-1}^{\text{unknown}}$$

Constrain pixels each iteration.



Source: [Lugmayr et al., RePaint]

Sample question

True or false: Masked diffusion language models predict text left-to-right (or right-to-left).

Sample question

True or false: Flow matching models allow you to directly estimate the probability density of an input example.

How do we construct x_t?

TLDR: Sample noise, add it, then reconstruct the data

- Flow matching is quite general! For now, just consider time-dependent weighted combinations.
- Flow matching says we can choose any weighted combination, as long we start from a sample in the source (e.g., Gaussian) and end with a sample in the target distribution (image).

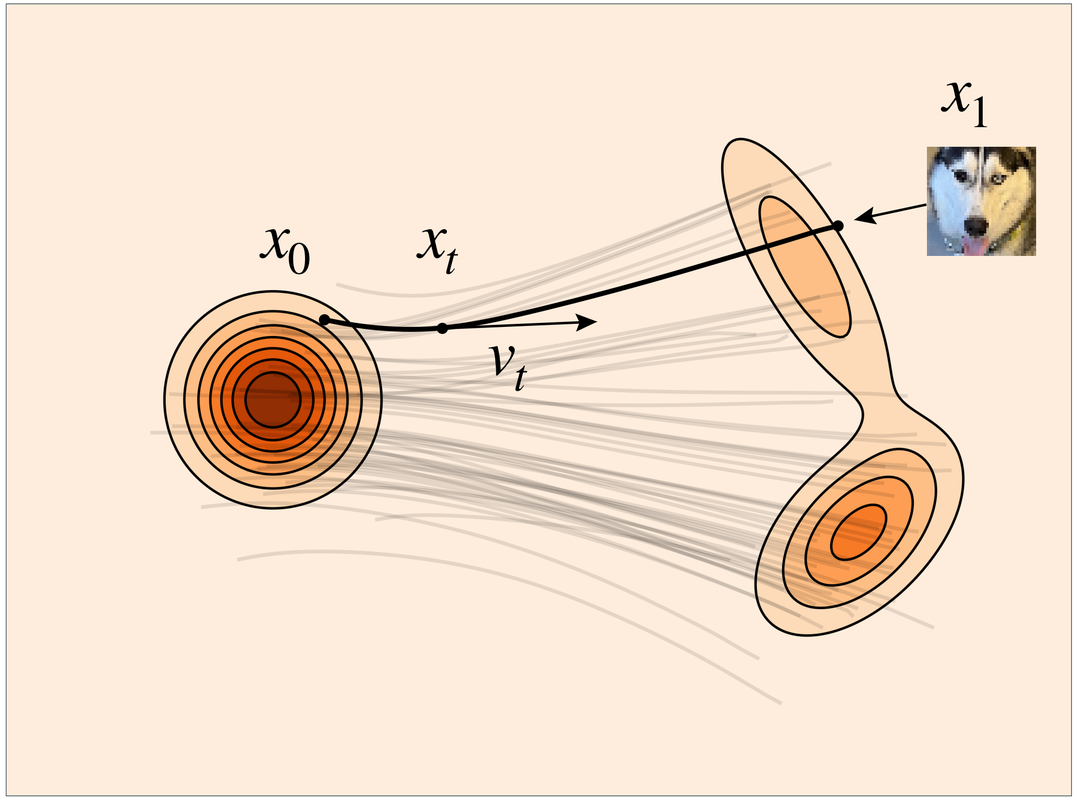
$$x_t = \alpha_t x_0 + \sigma_t x_1$$

$$x_0 \sim p_0(x) \qquad x_1 \sim p_1(x)$$

Adapted from Angjoo Kanazawa

Flow training

- For each image x_1
  - Sample some noise x_0
  - Combine them however you'd like to get x_t
  - Now learn to predict the velocity at x_t
    - What is the velocity? It depends on how you got x_t.



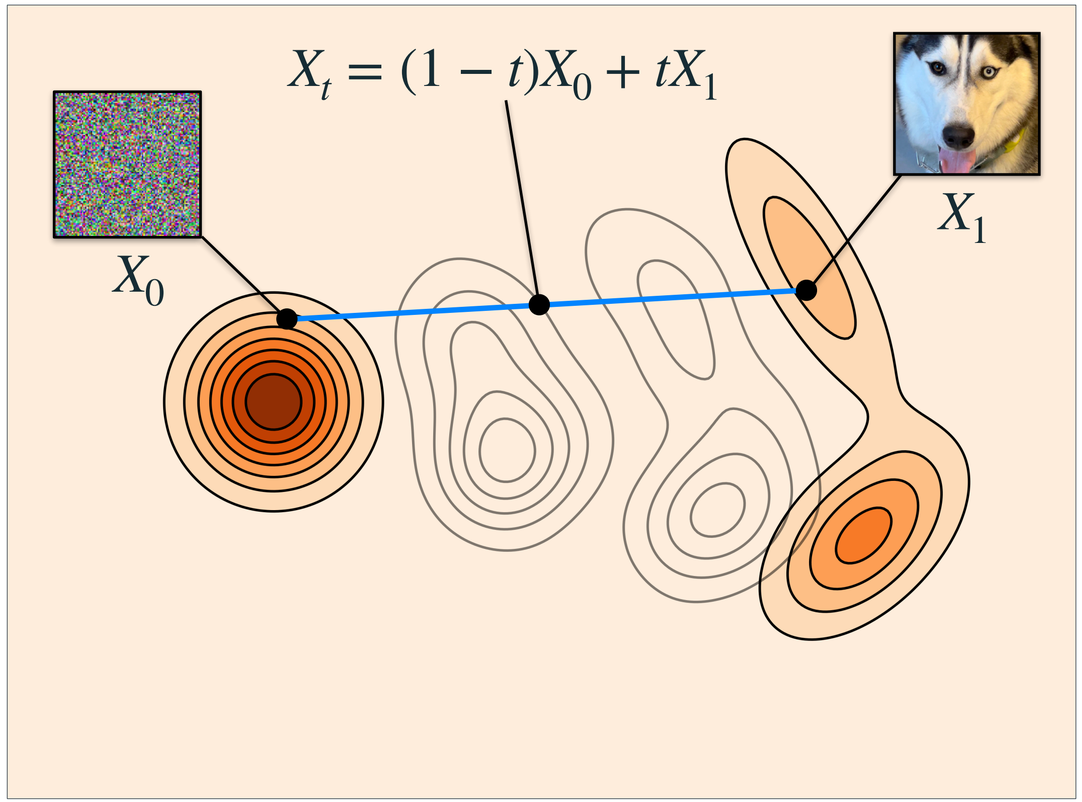
Adapted from Angjoo Kanazawa

A Very Simple Way

Linear interpolation!

$$x_t = \alpha_t x_0 + \sigma_t x_1$$

$$x_t = (1 - t)x_0 + t x_1$$



Source: Angjoo Kanazawa

What is the velocity supervision?

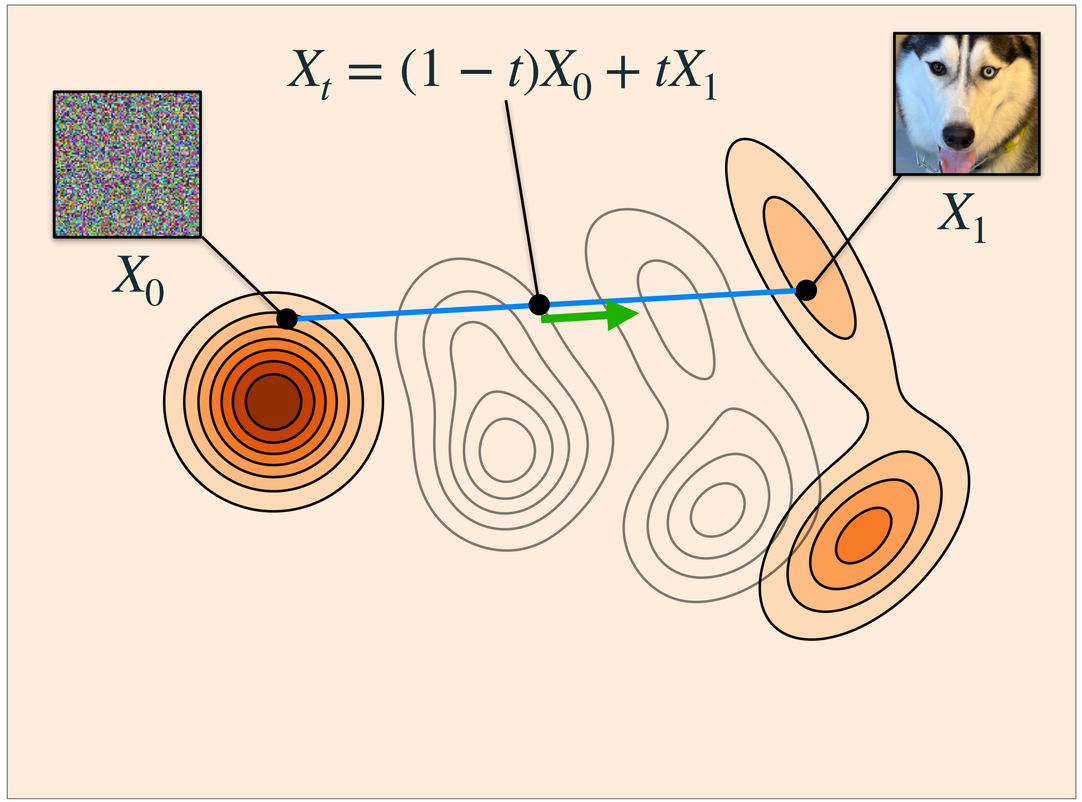
$$x_t = \alpha_t x_0 + \sigma_t x_1$$

$$x_t = (1 - t)x_0 + t x_1$$

$$\frac{dx_t}{dt} = -x_0 + x_1$$

$$= x_1 - x_0$$

$$\mathbb{E}_{t,X_0,X_1}\left\| u_t^{\theta}(X_t) - (X_1 - X_0) \right\|^2$$



*Conditioned on a single sample

Source: Angjoo Kanazawa

Inside a Training Loop

Flow Matching

```
x = next(dataset)
t = torch.rand(1) # Sample timestep (0,1)
noise = torch.randn_like(x) # Sample noise
x_t = (1-t) * noise + (t) * x # Get noisy x_t

flow_pred = model(x_t, t) # Predict noise in x_t
flow_gt = x - noise # ground truth flow (w/ linear sched)
loss = F.mse_loss(flow_pred, flow_gt) # Update model
loss.backward()
optimizer.step()
```

Source: Angjoo Kanazawa

Sample question

Which statement explains why autoregressive language models are trained with teacher forcing?

A. Teacher forcing makes generation at test time parallel across all output tokens.

B. Teacher forcing gives the correct maximum likelihood objective by conditioning on the true prefix of each sequence.

C. Teacher forcing directly optimizes the generated sequence’s reward under human preferences.

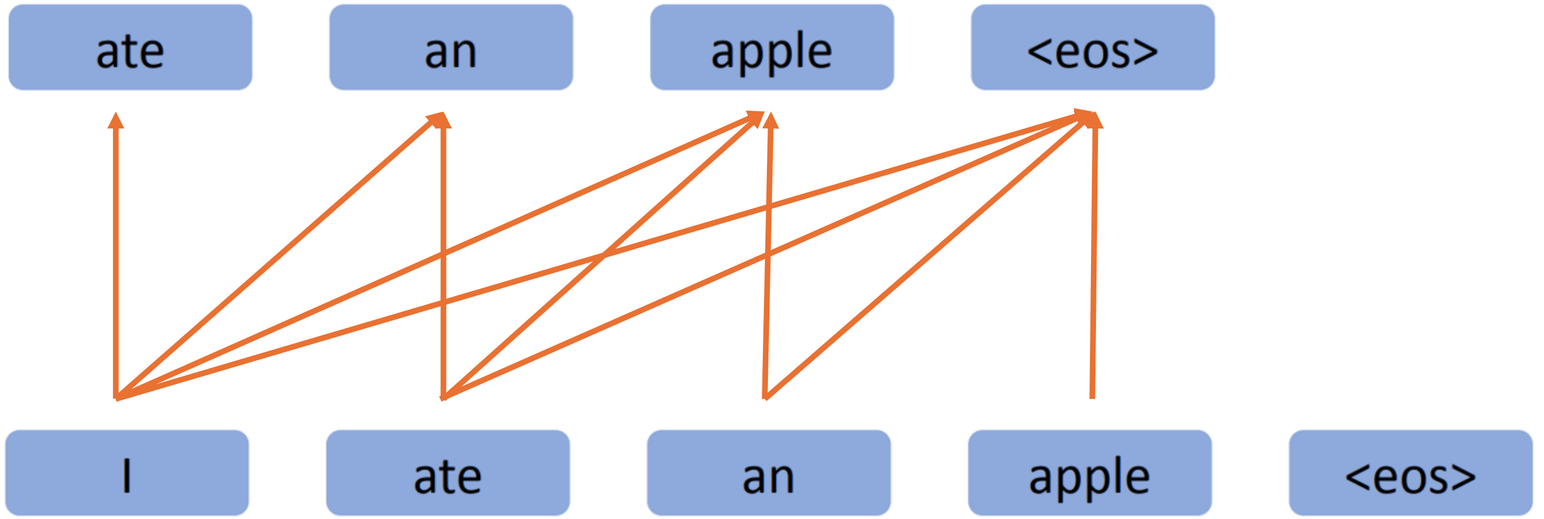
Autoregressive Language Model

$$\pi(y_1, y_2, y_3, y_4, y_5, y_6|x) = \pi(y_1|x)\pi(y_2|x, y_1) \dots \pi(y_6|x, y_1, y_2, y_3, y_4, y_5)$$

$$\underbrace{\overbrace{the}^{y_1}\ \overbrace{capital\ of}^{y_2}\ \overbrace{France}^{y_3}\ \overbrace{is}^{y_4}\ \overbrace{Paris}^{y_5}}_{\text{Response } y} \sim \pi(\cdot\,|\underbrace{what\ is\ the\ capital\ of\ France?}_{\text{Prompt } x})$$

Source: Zhaolin Gao

Training: Teacher forcing



- Feeding the ground truth token from the previous time step as input for the current time step
  - rather than the model's own, potentially incorrect, prediction
- Stabilizes training by preventing early errors from compounding

Source: Zhaolin Gao

Training: Teacher forcing

Given a sequence of tokens from dataset, $x_{1:T}$:

$$\max_{\pi} \pi(x_{2:T}|x_1)$$

$$\max_{\pi} \prod_{t=2}^{T} \pi(x_t|x_{1:t-1})$$

$$\max_{\pi} \log \prod_{t=2}^{T} \pi(x_t|x_{1:t-1})$$

$$\max_{\pi} \sum_{t=2}^{T} \log \pi(x_t|x_{1:t-1})$$

Source: Zhaolin Gao