



10-423/10-623 Generative AI

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Diffusion Models + Variational Autoencoders (VAEs)

Matt Gormley & Pat Virtue

Lecture 8

Feb. 10, 2025

Reminders

- **Homework 1: Generative Models of Text**
 - Out: Mon, Jan 27
 - Due: Mon, Feb 10 at 11:59pm
- **Quiz 2:**
 - In-class: Mon, Feb 17
 - Lectures 5-8
- **Homework 2: Generative Models of Images**
 - Out: Mon, Feb 10
 - Due: Sat, Feb 22 at 11:59pm

Recap of...

DIFFUSION MODELS

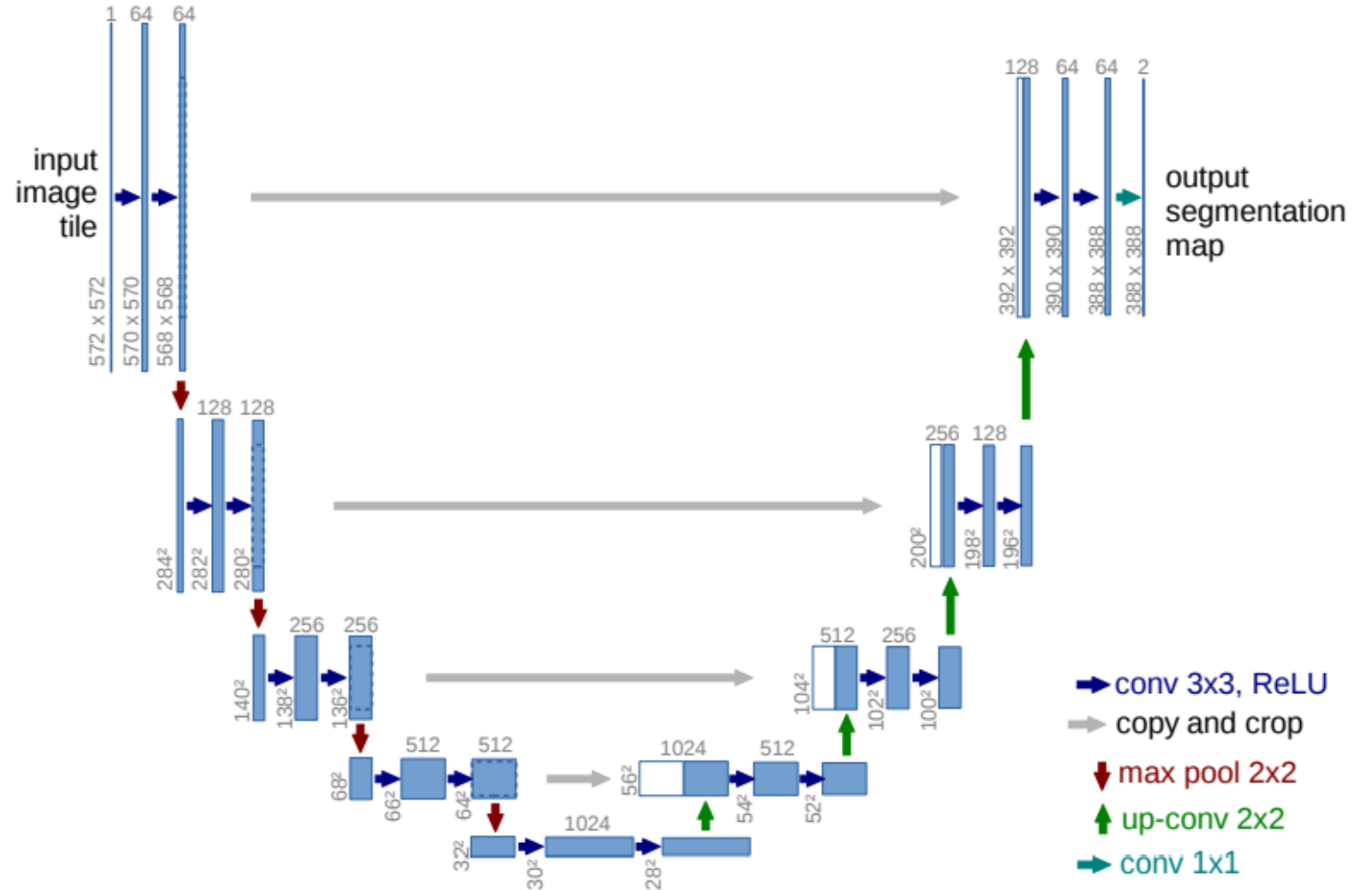
U-Net

Contracting path

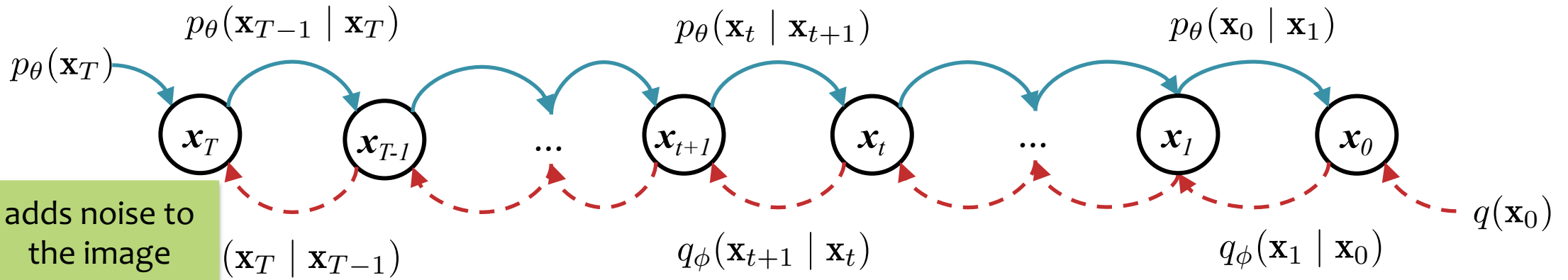
- block consists of:
 - 3x3 convolution
 - 3x3 convolution
 - ReLU
 - max-pooling with stride of 2 (downsample)
- repeat the block N times, doubling number of channels

Expanding path

- block consists of:
 - 2x2 convolution (upsampling)
 - concatenation with contracting path features
 - 3x3 convolution
 - 3x3 convolution
 - ReLU
- repeat the block N times, halving the number of channels



Diffusion Model



Forward Process:

$$q_\phi(\mathbf{x}_{0:T}) = q(\mathbf{x}_0) \prod_{t=1}^T q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1})$$

if we could sample from this we'd be done

(Learned) Reverse Process:

removes noise

$$p_\theta(\mathbf{x}_{0:T}) = p_\theta(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$$

goal is to learn this

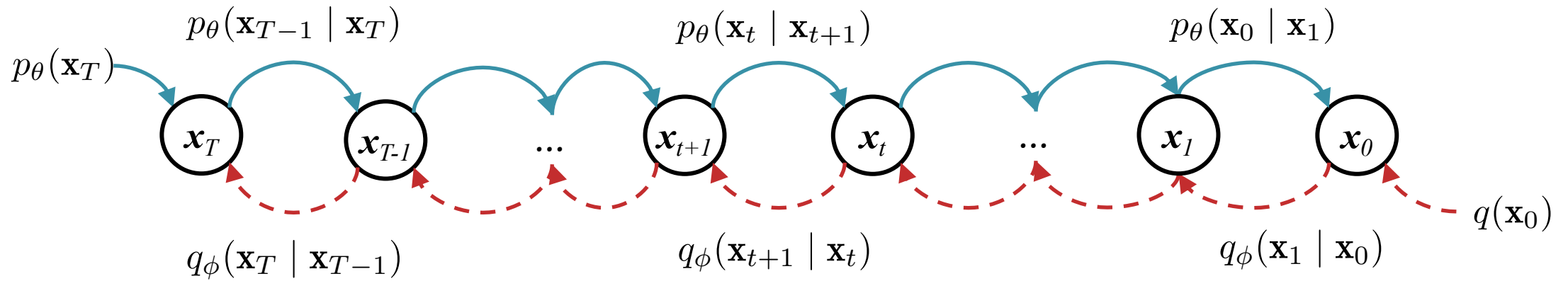
(Exact) Reverse Process:

$$q_\phi(\mathbf{x}_{0:T}) = q_\phi(\mathbf{x}_T) \prod_{t=1}^T q_\phi(\mathbf{x}_{t-1} | \mathbf{x}_t)$$

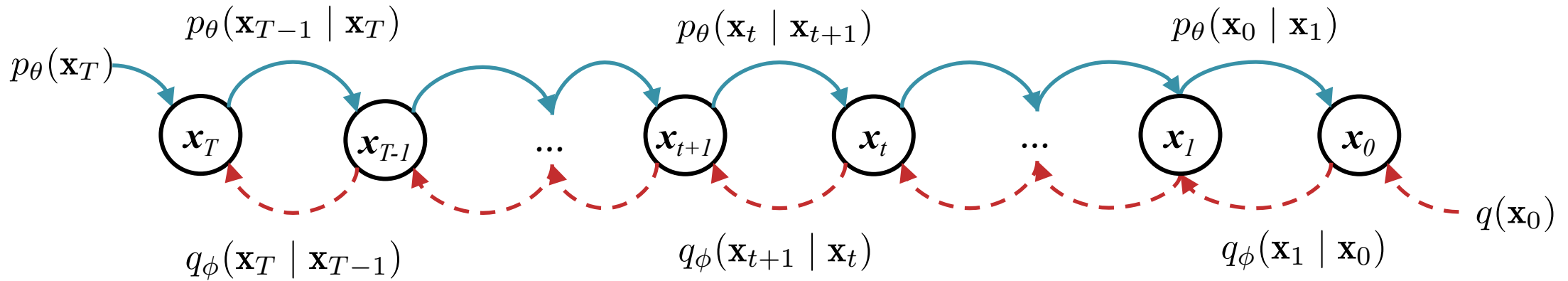
The exact reverse process requires inference. And, even though $q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1})$ is simple, computing $q_\phi(\mathbf{x}_{t-1} | \mathbf{x}_t)$ is intractable! Why? Because $q(\mathbf{x}_0)$ might be not-so-simple.

$$q_\phi(\mathbf{x}_{t-1} | \mathbf{x}_t) = \frac{\int_{\mathbf{x}_{0:t-2,t+1:T}} q_\phi(\mathbf{x}_{0:T}) d\mathbf{x}_{0:t-2,t+1:T}}{\int_{\mathbf{x}_{0:t-2,t:T}} q_\phi(\mathbf{x}_{0:T}) d\mathbf{x}_{0:t-2,t:T}}$$

Diffusion Model



Denoising Diffusion Probabilistic Model (DDPM)



Forward Process:

$$q_\phi(\mathbf{x}_{0:T}) = q(\mathbf{x}_0) \prod_{t=1}^T q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1})$$

$q(\mathbf{x}_0) =$ data distribution

$$q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1}) \sim \mathcal{N}(\sqrt{\alpha_t} \mathbf{x}_{t-1}, (1 - \alpha_t) \mathbf{I})$$

$$\hat{=} \hat{\mathbf{x}}_{t-1} + \epsilon \quad \text{where } \epsilon \sim \mathcal{N}(_, _)$$

(Learned) Reverse Process:

$$p_\theta(\mathbf{x}_{0:T}) = p_\theta(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$$

$$p_\theta(\mathbf{x}_T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) \sim \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

Gaussian (an aside)

Let $X \sim \mathcal{N}(\mu_x, \sigma_x^2)$ and $Y \sim \mathcal{N}(\mu_y, \sigma_y^2)$

1. Sum of two Gaussians is a Gaussian

$$X + Y \sim \mathcal{N}(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2)$$

2. Difference of two Gaussians is a Gaussian

$$X - Y \sim \mathcal{N}(\mu_x - \mu_y, \sigma_x^2 + \sigma_y^2)$$

3. Gaussian with a Gaussian mean has a Gaussian Conditional

$$Z \sim \mathcal{N}(\mu_z = X, \sigma_z^2) \Rightarrow P(Z | X) \sim \mathcal{N}(\cdot, \cdot)$$

Defining the Forward Process

Forward Process:

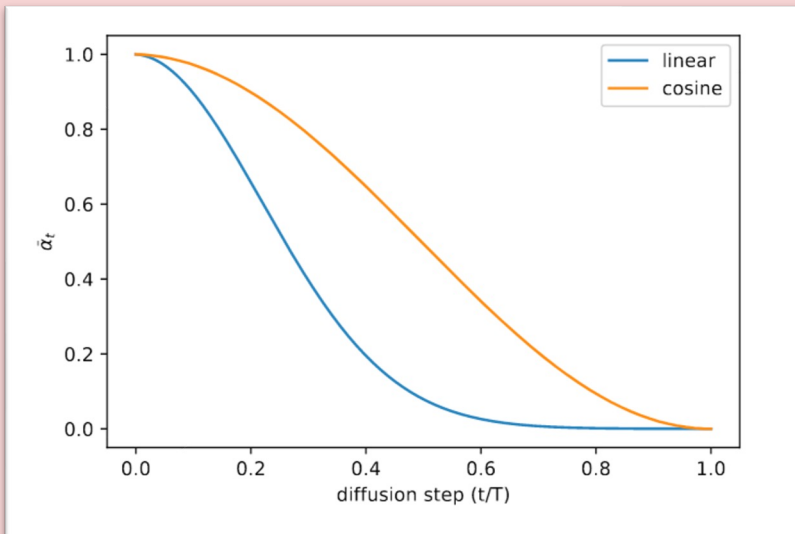
$$q_\phi(\mathbf{x}_{0:T}) = q(\mathbf{x}_0) \prod_{t=1}^T q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1})$$

$q(\mathbf{x}_0)$ = data distribution

$$q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1}) \sim \mathcal{N}(\sqrt{\alpha_t} \mathbf{x}_{t-1}, (1 - \alpha_t) \mathbf{I})$$

Noise schedule:

We choose α_t to follow a fixed schedule s.t. $q_\phi(\mathbf{x}_T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, just like $p_\theta(\mathbf{x}_T)$.



Property #1:

$$q(\mathbf{x}_t | \mathbf{x}_0) \sim \mathcal{N}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$$

$$\text{where } \bar{\alpha}_t = \prod_{s=1}^t \alpha_s$$

Q: So what is $q_\phi(\mathbf{x}_T | \mathbf{x}_0)$? Note the *capital T* in the subscript.

A:

$$q(\mathbf{x}_T | \mathbf{x}_0) \sim \mathcal{N}(\boldsymbol{\mu} \approx \mathbf{0}, \boldsymbol{\Sigma} \approx \mathbf{I})$$

Gaussian (an aside)

Let $X \sim \mathcal{N}(\mu_x, \sigma_x^2)$ and $Y \sim \mathcal{N}(\mu_y, \sigma_y^2)$

1. Sum of two Gaussians is a Gaussian

$$X + Y \sim \mathcal{N}(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2)$$

2. Difference of two Gaussians is a Gaussian

$$X - Y \sim \mathcal{N}(\mu_x - \mu_y, \sigma_x^2 + \sigma_y^2)$$

3. Gaussian with a Gaussian mean has a Gaussian Conditional

$$Z \sim \mathcal{N}(\mu_z = X, \sigma_z^2) \Rightarrow P(Z | X) \sim \mathcal{N}(\cdot, \cdot)$$

4. But #3 does not hold if X is passed through a nonlinear function f

$$W \sim \mathcal{N}(\mu_z = f(X), \sigma_w^2) \not\Rightarrow P(W | X) \sim \mathcal{N}(\cdot, \cdot)$$

Parameterizing the *learned* reverse process

$$\text{Recall: } p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \sim \mathcal{N}(\mu_{\theta}(\mathbf{x}_t, t), \Sigma_{\theta}(\mathbf{x}_t, t))$$

Later we will show that given a training sample \mathbf{x}_0 , we want

$$p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$

to be as close as possible to

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$$

Intuitively, this makes sense: if the *learned* reverse process is supposed to subtract away the noise, then whenever we're working with a specific \mathbf{x}_0 it should subtract it away exactly as *exact* reverse process would have.

Properties of forward and *exact* reverse processes

Property #1:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) \sim \mathcal{N}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$$

$$\text{where } \bar{\alpha}_t = \prod_{s=1}^t \alpha_s$$

⇒ we can sample \mathbf{x}_t from \mathbf{x}_0 at any timestep t efficiently in closed form

$$\Rightarrow \mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Property #2: Estimating $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is intractable because of its dependence on $q(\mathbf{x}_0)$. However, conditioning on \mathbf{x}_0 we can efficiently work with:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0), \sigma_t^2 \mathbf{I})$$

$$\text{where } \tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_t}(1 - \alpha_t)}{1 - \bar{\alpha}_t} \mathbf{x}_0 + \frac{\sqrt{\bar{\alpha}_t}(1 - \bar{\alpha}_t)}{1 - \bar{\alpha}_t} \mathbf{x}_t$$

$$= \alpha_t^{(0)} \mathbf{x}_0 + \alpha_t^{(t)} \mathbf{x}_t$$

$$\sigma_t^2 = \frac{(1 - \bar{\alpha}_{t-1})(1 - \alpha_t)}{1 - \bar{\alpha}_t}$$

Property #3: Combining the two previous properties, we can obtain a different parameterization of $\tilde{\mu}_q$ which has been shown empirically to help in learning p_θ .

Rearranging $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}$ we have that:

$$\mathbf{x}_0 = (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}) / \sqrt{\bar{\alpha}_t}$$

Substituting this definition of \mathbf{x}_0 into property #2's definition of $\tilde{\mu}_q$ gives:

$$\begin{aligned} \tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) &= \alpha_t^{(0)} \mathbf{x}_0 + \alpha_t^{(t)} \mathbf{x}_t \\ &= \alpha_t^{(0)} \left((\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}) / \sqrt{\bar{\alpha}_t} \right) + \alpha_t^{(t)} \mathbf{x}_t \\ &= \frac{1}{\sqrt{\bar{\alpha}_t}} \left(\mathbf{x}_t - \frac{(1 - \alpha_t)}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon} \right) \end{aligned}$$

Parameterizing the *learned* reverse process

Recall: $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \sim \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$

Idea #1: Rather than learn $\Sigma_\theta(\mathbf{x}_t, t)$ just use what we know about $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \sim \mathcal{N}(\tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0), \sigma_t^2 \mathbf{I})$:

$$\Sigma_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$$

Idea #2: Choose μ_θ based on $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$, i.e. we want $\mu_\theta(\mathbf{x}_t, t)$ to be close to $\tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)$. Here are three ways we could parameterize this:

Option A: Learn a network that approximates $\tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)$ directly from \mathbf{x}_t and t :

$$\mu_\theta(\mathbf{x}_t, t) = \text{UNet}_\theta(\mathbf{x}_t, t)$$

where t is treated as an extra feature in UNet

Option B: Learn a network that approximates the real \mathbf{x}_0 from only \mathbf{x}_t and t :

$$\mu_\theta(\mathbf{x}_t, t) = \alpha_t^{(0)} \mathbf{x}_\theta^{(0)}(\mathbf{x}_t, t) + \alpha_t^{(t)} \mathbf{x}_t$$

where $\mathbf{x}_\theta^{(0)}(\mathbf{x}_t, t) = \text{UNet}_\theta(\mathbf{x}_t, t)$

Option C: Learn a network that approximates the ϵ that gave rise to \mathbf{x}_t from \mathbf{x}_0 in the forward process from \mathbf{x}_t and t :

$$\mu_\theta(\mathbf{x}_t, t) = \alpha_t^{(0)} \mathbf{x}_\theta^{(0)}(\mathbf{x}_t, t) + \alpha_t^{(t)} \mathbf{x}_t$$

$$\text{where } \mathbf{x}_\theta^{(0)}(\mathbf{x}_t, t) = (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta(\mathbf{x}_t, t)) / \sqrt{\bar{\alpha}_t}$$

$$\text{where } \epsilon_\theta(\mathbf{x}_t, t) = \text{UNet}_\theta(\mathbf{x}_t, t)$$

DIFFUSION MODEL TRAINING

Learning the Reverse Process

Recall: given a training sample \mathbf{x}_0 , we want

$$p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$

to be as close as possible to

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$$

Depending on which of the options for parameterization we pick, we get a different training algorithm.

Algorithm 1 Training (Option A, all timesteps)

- 1: initialize θ
 - 2: **for** $e \in \{1, \dots, E\}$ **do**
 - 3: **for** $x_0 \in \mathcal{D}$ **do**
 - 4: **for** $t \in \{1, \dots, T\}$ **do**
 - 5: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ←
 - 6: $\mathbf{x}_t \leftarrow \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$ ←
 - 7: $\tilde{\mu}_q \leftarrow \alpha_t^{(0)} \mathbf{x}_0 + \alpha_t^{(t)} \mathbf{x}_t$
 - 8: $\ell_t(\theta) \leftarrow \|\tilde{\mu}_q - \mu_{\theta}(\mathbf{x}_t, t)\|^2$
 - 9: $\theta \leftarrow \theta - \nabla_{\theta} \sum_{t=1}^T \ell_t(\theta)$
-

Option A: Learn a network that approximates $\tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)$ directly from \mathbf{x}_t and t :

$$\mu_{\theta}(\mathbf{x}_t, t) = \text{UNet}_{\theta}(\mathbf{x}_t, t)$$

where t is treated as an extra feature in UNet

Learning the Reverse Process

Recall: given a training sample \mathbf{x}_0 , we want

$$p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$

to be as close as possible to

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$$

Depending on which of the options for parameterization we pick, we get a different training algorithm.

Algorithm 1 Training (Option A)

- 1: initialize θ
 - 2: **for** $e \in \{1, \dots, E\}$ **do**
 - 3: **for** $x_0 \in \mathcal{D}$ **do**
 - 4: $t \sim \text{Uniform}(1, \dots, T)$
 - 5: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 6: $\mathbf{x}_t \leftarrow \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$
 - 7: $\tilde{\mu}_q \leftarrow \alpha_t^{(0)} \mathbf{x}_0 + \alpha_t^{(t)} \mathbf{x}_t$
 - 8: $\ell_t(\theta) \leftarrow \|\tilde{\mu}_q - \mu_{\theta}(\mathbf{x}_t, t)\|^2$
 - 9: $\theta \leftarrow \theta - \nabla_{\theta} \ell_t(\theta)$
-

Option A: Learn a network that approximates $\tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)$ directly from \mathbf{x}_t and t :

$$\mu_{\theta}(\mathbf{x}_t, t) = \text{UNet}_{\theta}(\mathbf{x}_t, t)$$

where t is treated as an extra feature in UNet

Learning the Reverse Process

Recall: given a training sample \mathbf{x}_0 , we want

$$p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$

to be as close as possible to

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$$

Depending on which of the options for parameterization we pick, we get a different training algorithm.

Algorithm 1 Training (Option B)

- 1: initialize θ
 - 2: **for** $e \in \{1, \dots, E\}$ **do**
 - 3: **for** $x_0 \in \mathcal{D}$ **do**
 - 4: $t \sim \text{Uniform}(1, \dots, T)$
 - 5: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 6: $\mathbf{x}_t \leftarrow \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$
 - 7: $\ell_t(\theta) \leftarrow \|\mathbf{x}_0 - \mathbf{x}_{\theta}^{(0)}(\mathbf{x}_t, t)\|^2$
 - 8: $\theta \leftarrow \theta - \nabla_{\theta} \ell_t(\theta)$
-

Option B: Learn a network that approximates the real \mathbf{x}_0 from only \mathbf{x}_t and t :

$$\mu_{\theta}(\mathbf{x}_t, t) = \alpha_t^{(0)} \mathbf{x}_{\theta}^{(0)}(\mathbf{x}_t, t) + \alpha_t^{(t)} \mathbf{x}_t$$

$$\text{where } \mathbf{x}_{\theta}^{(0)}(\mathbf{x}_t, t) = \text{UNet}_{\theta}(\mathbf{x}_t, t)$$

Learning the Reverse Process

Recall: given a training sample \mathbf{x}_0 , we want

$$p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$

to be as close as possible to

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$$

Depending on which of the options for parameterization we pick, we get a different training algorithm.

Option C is the best empirically

Algorithm 1 Training (Option C)

- 1: initialize θ
 - 2: **for** $e \in \{1, \dots, E\}$ **do**
 - 3: **for** $x_0 \in \mathcal{D}$ **do**
 - 4: $t \sim \text{Uniform}(1, \dots, T)$
 - 5: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 6: $\mathbf{x}_t \leftarrow \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$
 - 7: $\ell_t(\theta) \leftarrow \|\epsilon - \epsilon_{\theta}(\mathbf{x}_t, t)\|^2$
 - 8: $\theta \leftarrow \theta - \nabla_{\theta} \ell_t(\theta)$
-

Option C: Learn a network that approximates the ϵ that gave rise to \mathbf{x}_t from \mathbf{x}_0 in the forward process from \mathbf{x}_t and t :

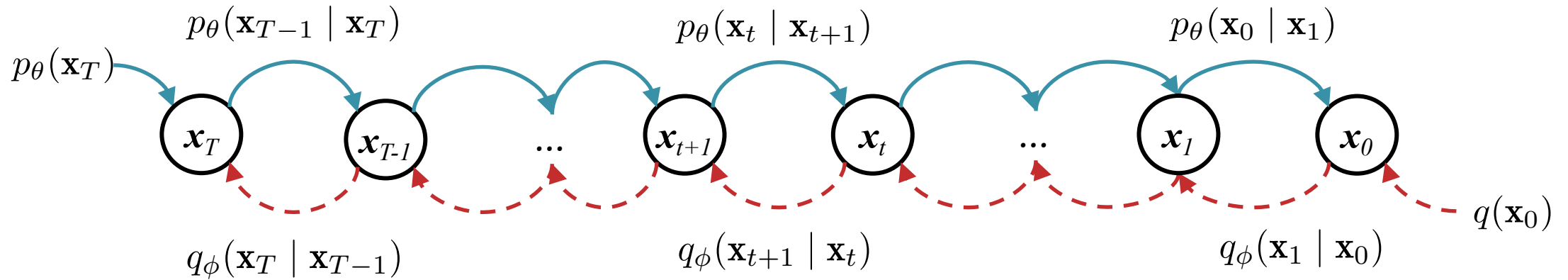
$$\mu_{\theta}(\mathbf{x}_t, t) = \alpha_t^{(0)} \mathbf{x}_{\theta}^{(0)}(\mathbf{x}_t, t) + \alpha_t^{(t)} \mathbf{x}_t$$

$$\text{where } \mathbf{x}_{\theta}^{(0)}(\mathbf{x}_t, t) = (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_{\theta}(\mathbf{x}_t, t)) / \sqrt{\bar{\alpha}_t}$$

$$\text{where } \epsilon_{\theta}(\mathbf{x}_t, t) = \text{UNet}_{\theta}(\mathbf{x}_t, t)$$

Training (Computation Graph)

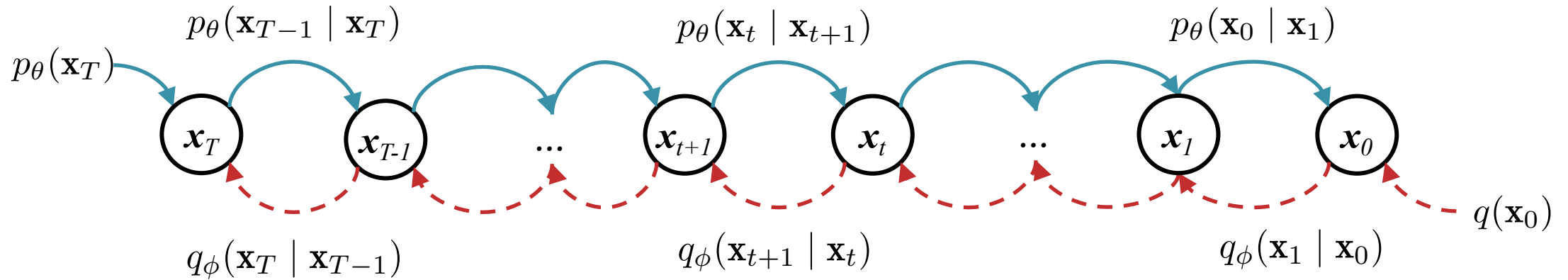
Sampling from the *learned* reverse process



Algorithm 1 Sampling

- 1: $\mathbf{x}_T \sim p_\theta(\mathbf{x}_T)$
 - 2: **for** $t \in \{T, \dots, 1\}$ **do**
 - 3: $\mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1} | \mathbf{x}_t)$
 - 4: **return** \mathbf{x}_0
-

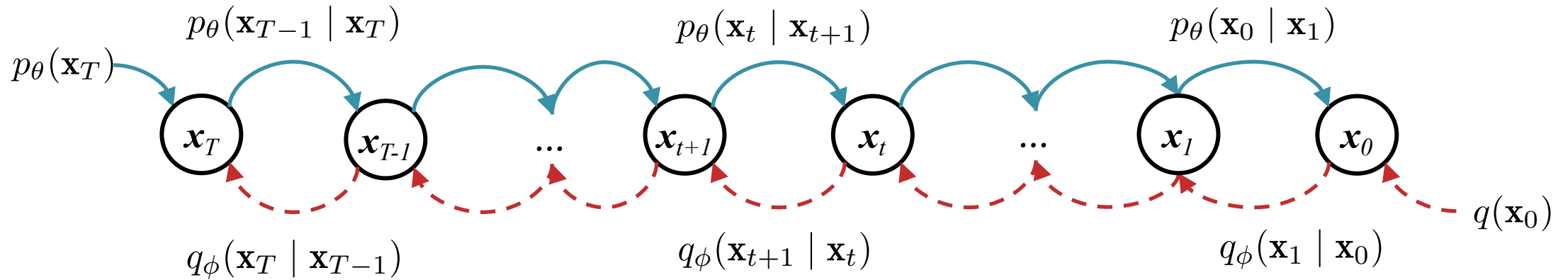
Sampling from the *learned* reverse process



Algorithm 1 Sampling

- 1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 2: **for** $t \in \{T, \dots, 1\}$ **do**
 - 3: $\mathbf{x}_{t-1} \sim \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$
 - 4: **return** \mathbf{x}_0
-

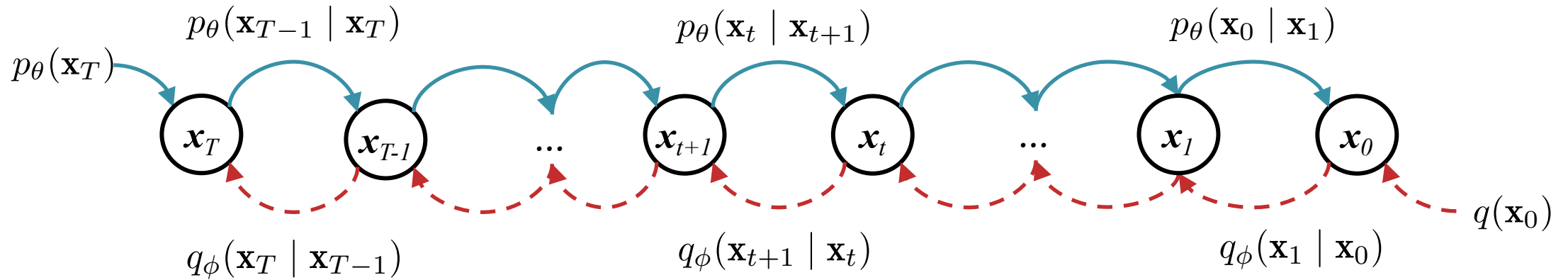
Sampling from the *learned* reverse process



Algorithm 1 Sampling (Option A)

- 1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 2: **for** $t \in \{T, \dots, 1\}$ **do**
 - 3: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 4: $\mathbf{x}_{t-1} \leftarrow \mu_\theta(\mathbf{x}_t, t) + \sigma_t \epsilon$
 - 5: **return** \mathbf{x}_0
-

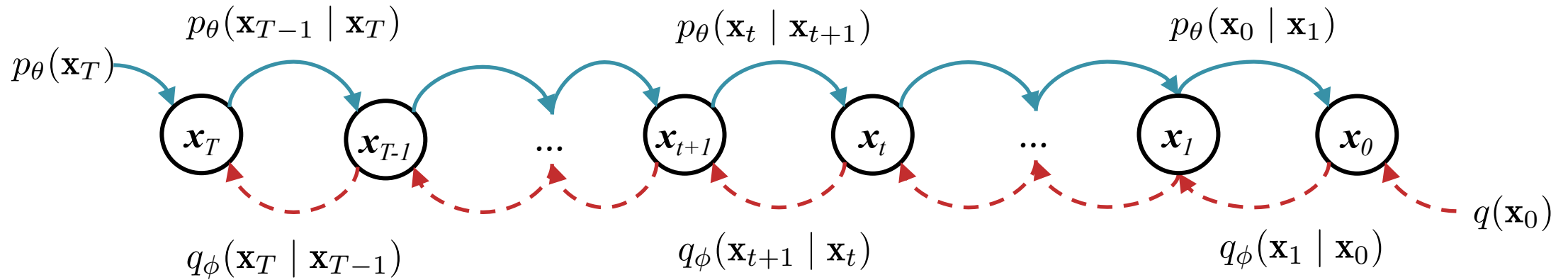
Sampling from the *learned* reverse process



Algorithm 1 Sampling (Option B)

- 1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 2: **for** $t \in \{T, \dots, 1\}$ **do**
 - 3: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 4: $\hat{\boldsymbol{\mu}}_t \leftarrow \alpha_t^{(0)} \mathbf{x}_\theta^{(0)}(\mathbf{x}_t, t) + \alpha_t^{(t)} \mathbf{x}_t$
 - 5: $\mathbf{x}_{t-1} \leftarrow \hat{\boldsymbol{\mu}}_t + \sigma_t \epsilon$
 - 6: **return** \mathbf{x}_0
-

Sampling from the *learned* reverse process



Algorithm 1 Sampling (Option C)

- 1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 2: **for** $t \in \{T, \dots, 1\}$ **do**
 - 3: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 4: $\hat{\mathbf{x}}_0 \leftarrow (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)) / \sqrt{\bar{\alpha}_t}$
 - 5: $\hat{\boldsymbol{\mu}}_t \leftarrow \alpha_t^{(0)} \hat{\mathbf{x}}_0 + \alpha_t^{(t)} \mathbf{x}_t$
 - 6: $\mathbf{x}_{t-1} \leftarrow \hat{\boldsymbol{\mu}}_t + \sigma_t \boldsymbol{\epsilon}$
 - 7: **return** \mathbf{x}_0
-

Unsupervised Learning

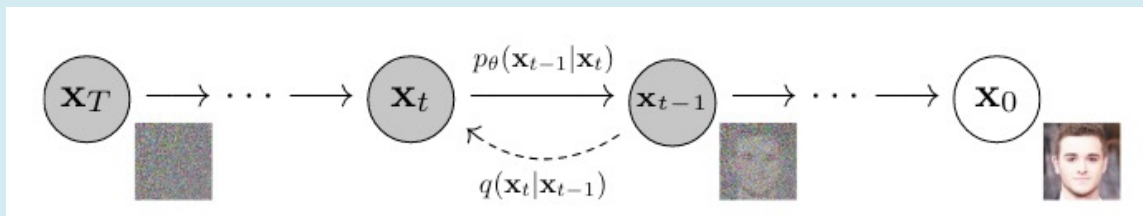
Assumptions:

1. our data comes from some distribution $p^*(\mathbf{x}_0)$
2. we choose a distribution $p_\theta(\mathbf{x}_0)$ for which sampling $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

Goal: learn θ s.t. $p_\theta(\mathbf{x}_0) \approx p^*(\mathbf{x}_0)$

Example: Diffusion Models

- true $p^*(\mathbf{x}_0)$ is distribution over photos taken and posted to Flickr
- choose $p_\theta(\mathbf{x}_0)$ to be an expressive model (e.g. noise fed into inverted CNN) that can generate images
 - sampling is will be easy
- learn by finding $\theta \approx \operatorname{argmax}_\theta \log(p_\theta(\mathbf{x}_0))$?
 - Sort of! We can't compute the gradient $\nabla_\theta \log(p_\theta(\mathbf{x}_0))$
 - So we instead optimize a variational lower bound (more on that later)



DDPM Objective Function

$$\mathbb{E}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] = \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] =: L$$

$$L = \mathbb{E}_q \left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t > 1} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \right]$$

This KL divergence term L_{t-1} wants the two conditional distributions to be as close as possible.

VARIATIONAL AUTOENCODERS

Unsupervised Learning

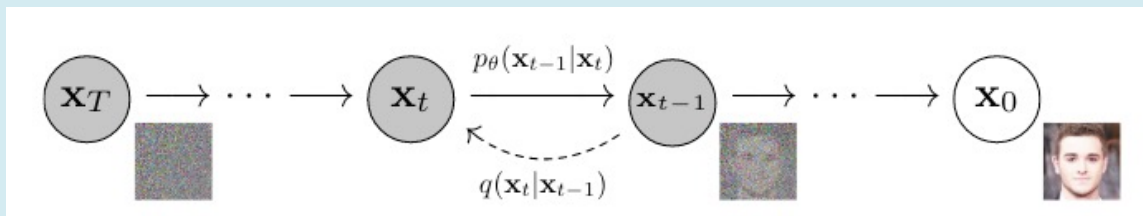
Assumptions:

1. our data comes from some distribution $p^*(\mathbf{x}_0)$
2. we choose a distribution $p_\theta(\mathbf{x}_0)$ for which sampling $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

Goal: learn θ s.t. $p_\theta(\mathbf{x}_0) \approx p^*(\mathbf{x}_0)$

Example: VAEs / Diffusion Models

- true $p^*(\mathbf{x}_0)$ is distribution over photos taken and posted to Flickr
- choose $p_\theta(\mathbf{x}_0)$ to be an expressive model (e.g. noise fed into inverted CNN) that can generate images
 - sampling is will be easy
- learn by finding $\theta \approx \operatorname{argmax}_\theta \log(p_\theta(\mathbf{x}_0))$?
 - Sort of! We can't compute the gradient $\nabla_\theta \log(p_\theta(\mathbf{x}_0))$
 - So we instead optimize a variational lower bound (more on that later)



Unsupervised Learning

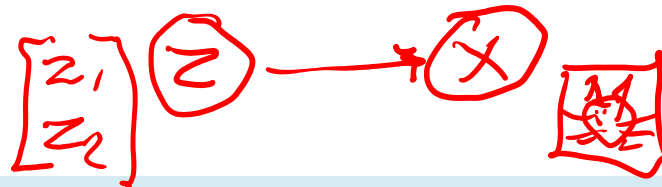
Assumptions:

1. our data comes from some distribution $p^*(\mathbf{x}_o)$
2. we choose a distribution $p_\theta(\mathbf{x}_o)$ for which sampling $\mathbf{x}_o \sim p_\theta(\mathbf{x}_o)$ is tractable

Goal: learn θ s.t. $p_\theta(\mathbf{x}_o) \approx p^*(\mathbf{x}_o)$

Example: VAEs / Diffusion Models

- true $p^*(\mathbf{x}_o)$ is distribution over photos taken and posted to Flickr
- choose $p_\theta(\mathbf{x}_o)$ to be an expressive model (e.g. noise fed into inverted CNN) that can generate images
 - sampling is will be easy
- learn by finding $\theta \approx \operatorname{argmax}_\theta \log(p_\theta(\mathbf{x}_o))$?
 - Sort of! We can't compute the gradient $\nabla_\theta \log(p_\theta(\mathbf{x}_o))$
 - So we instead optimize a variational lower bound (more on that later)



Variational Autoencoders

Outline

- Autoencoders ✓
 - Sampling from Autoencoders???
 - Autoencoders → Variational Autoencoders
- KL Divergence diversion 😊
 - Variational Inference
 - ELBO (Evidence Lower BOund)

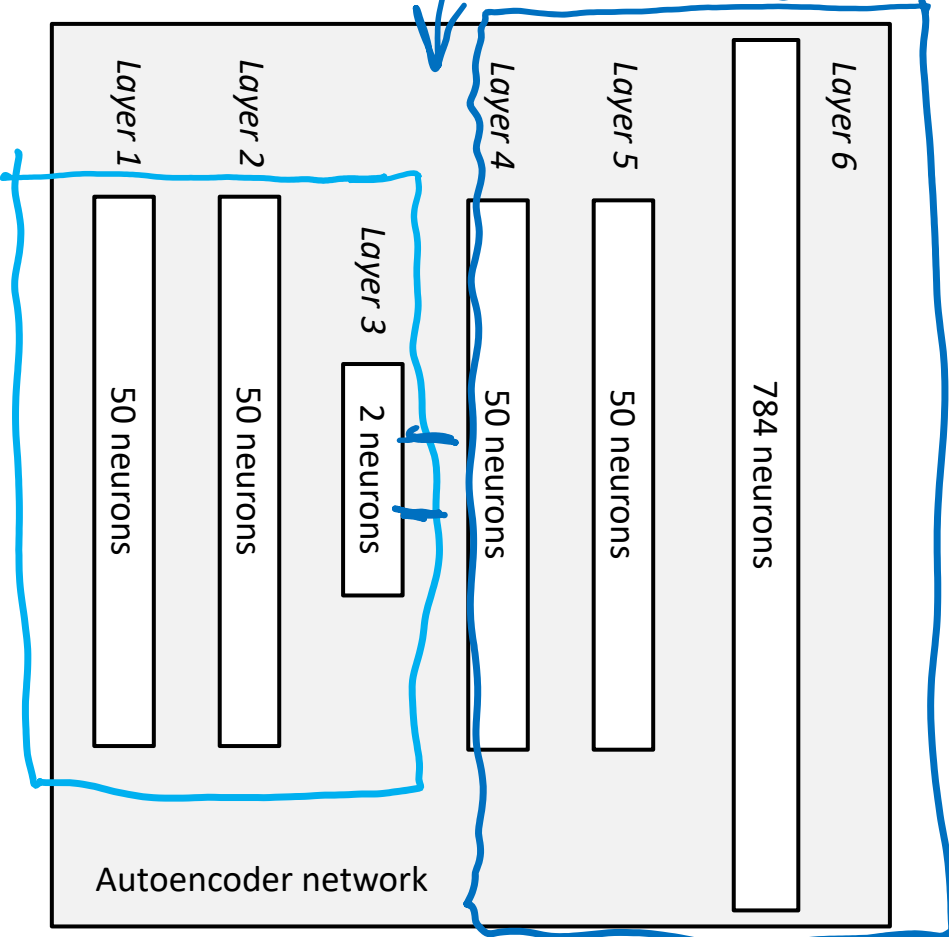
Autoencoders

Force dimensionality reduction bottleneck, then try to reconstruct input image

Input
28x28 = 784 pixels



$f_E(N)$



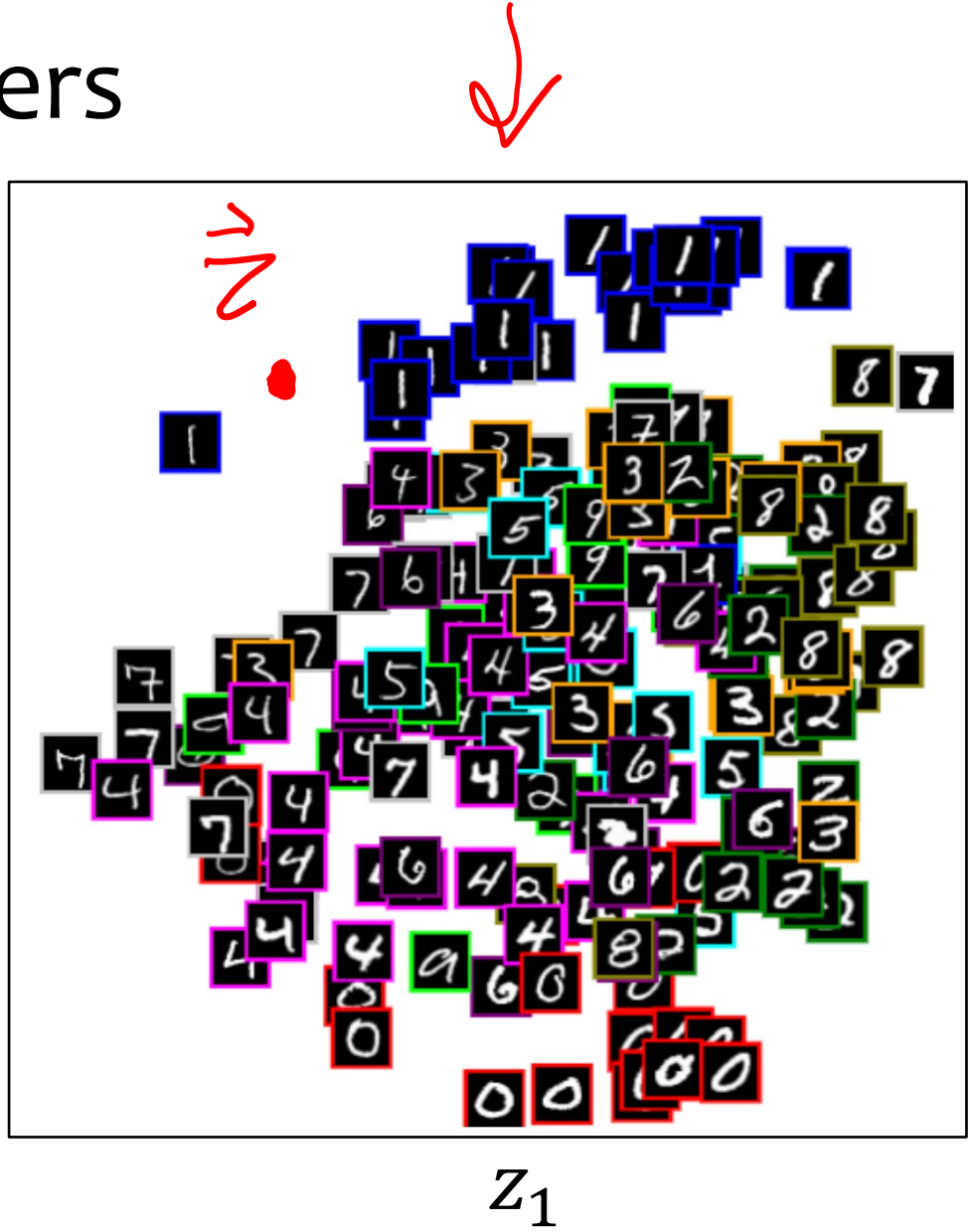
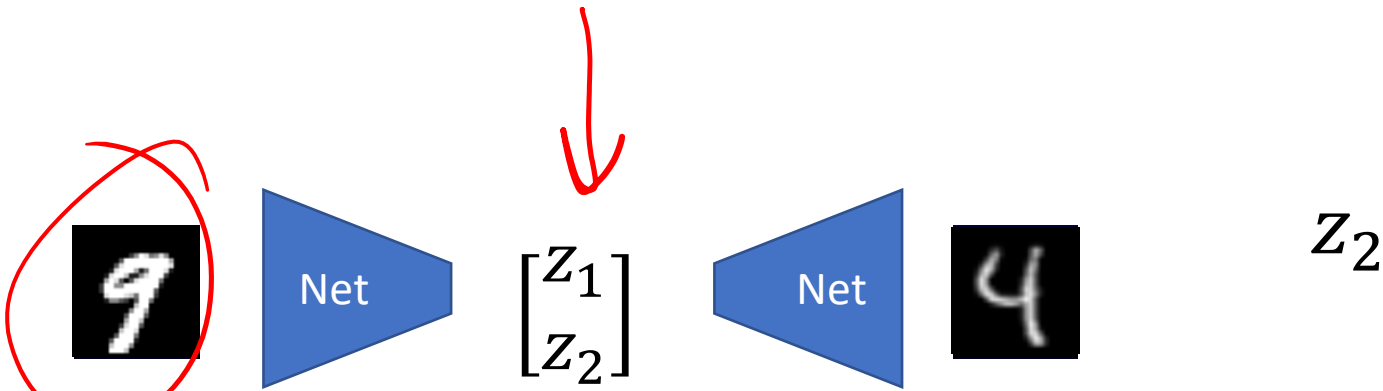
$f_G(N)$

Input
28x28 = 784 pixels



Autoencoders

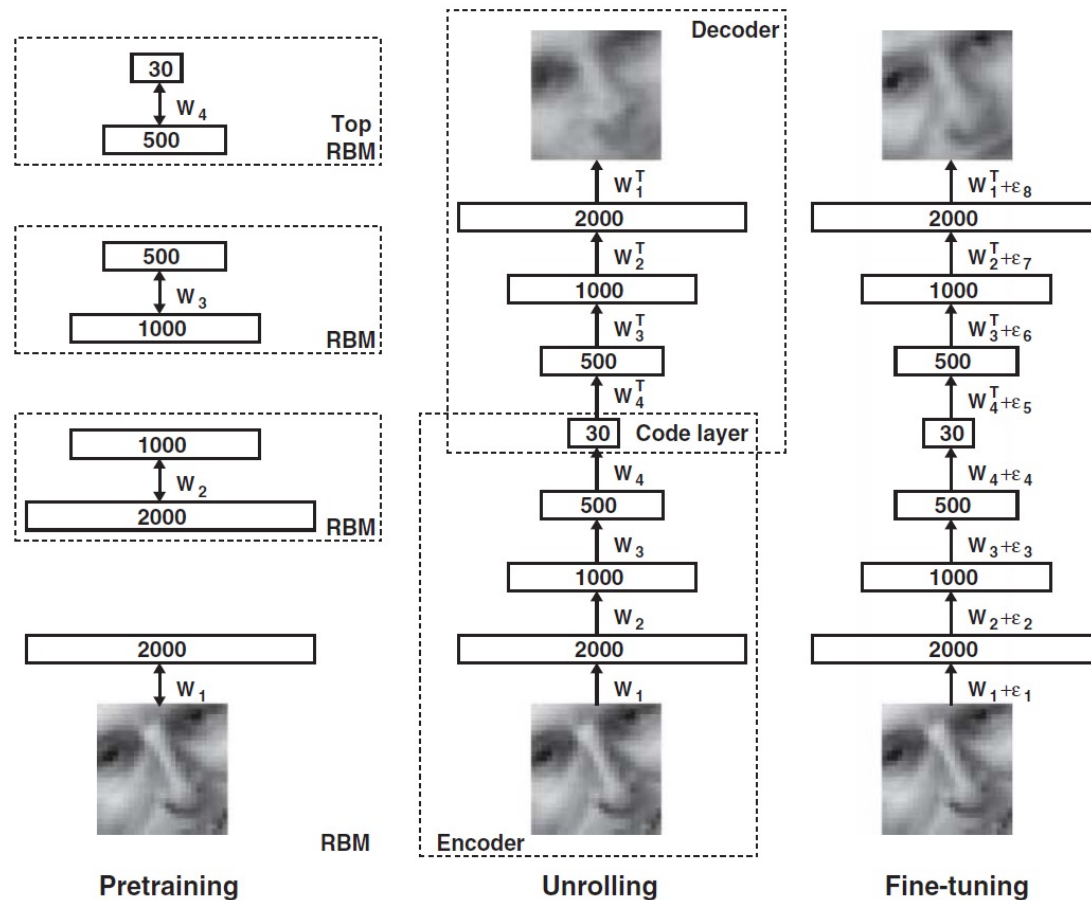
Force dimensionality reduction bottleneck,
then try to reconstruct input image



<https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html>

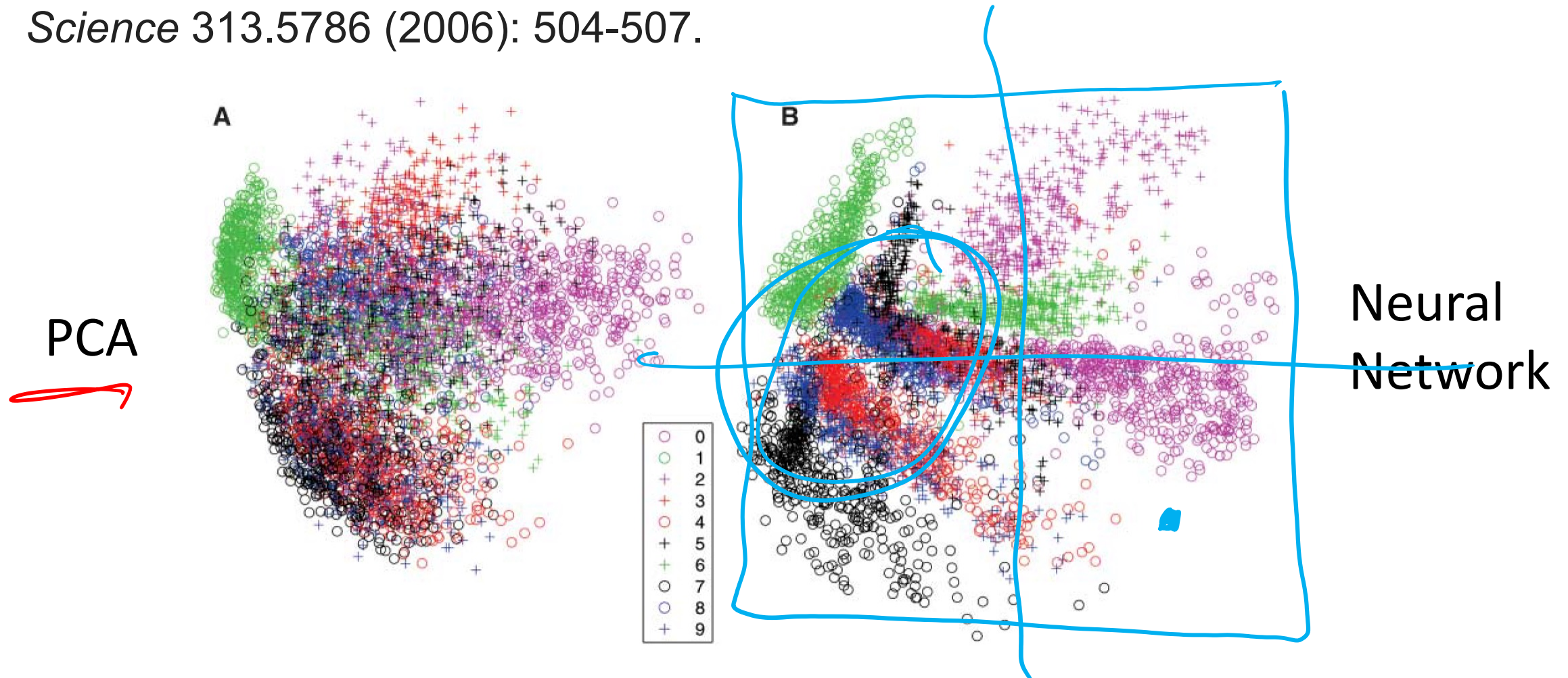
Dimensionality Reduction with Deep Learning

- Hinton, Geoffrey E., and Ruslan R. Salakhutdinov.
- "Reducing the dimensionality of data with neural networks."
- *Science* 313.5786 (2006): 504-507.



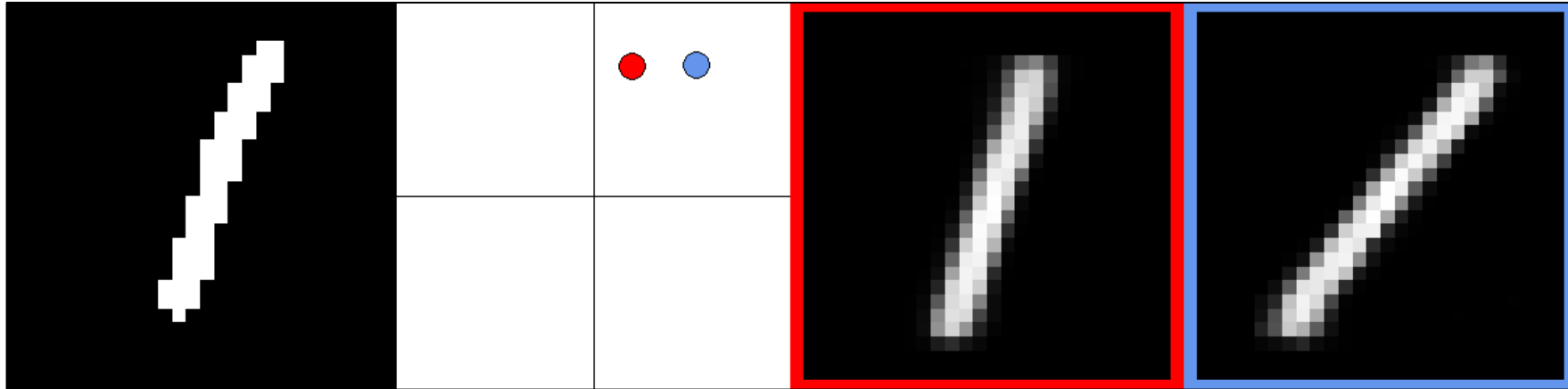
Dimensionality Reduction with Deep Learning

- Hinton, Geoffrey E., and Ruslan R. Salakhutdinov.
- "Reducing the dimensionality of data with neural networks."
- *Science* 313.5786 (2006): 504-507.



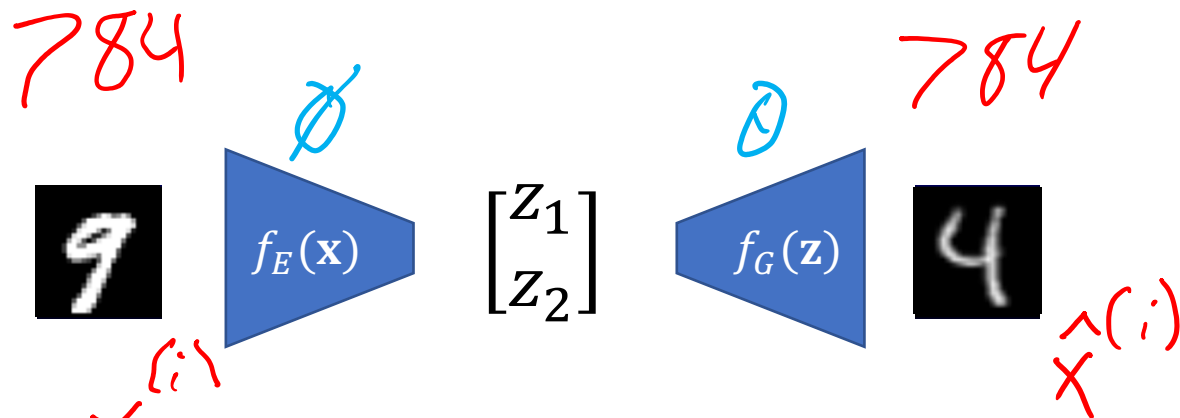
Digit Autoencoder

- Demo: Using a learned feature space



Autoencoder Learning

Objective

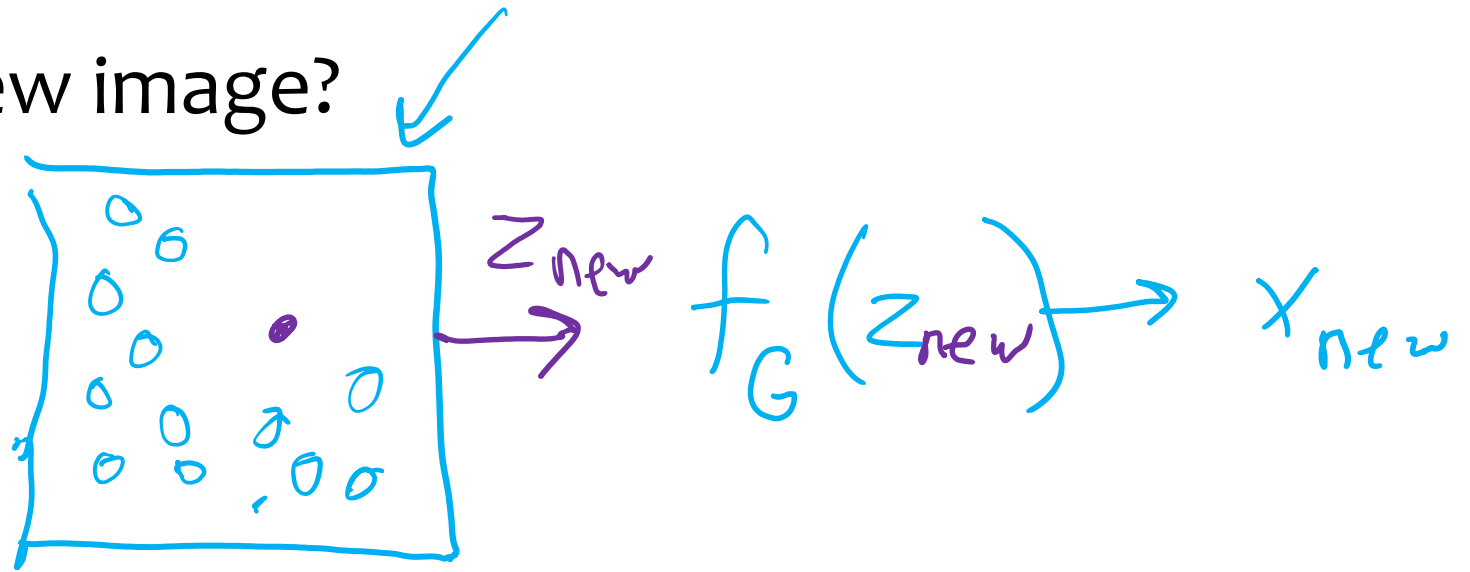


$$J(\theta, \phi) = \frac{1}{N} \sum_{i=1}^N \frac{1}{P} \|x^{(i)} - \hat{x}^{(i)}\|_2^2 \quad \leftarrow$$

$$- \frac{1}{N} \sum \log p(x) \quad \text{or} \\ p(x|z) \dots$$

Sampling $p(x)$ from Autoencoder

How do we sample a new image?

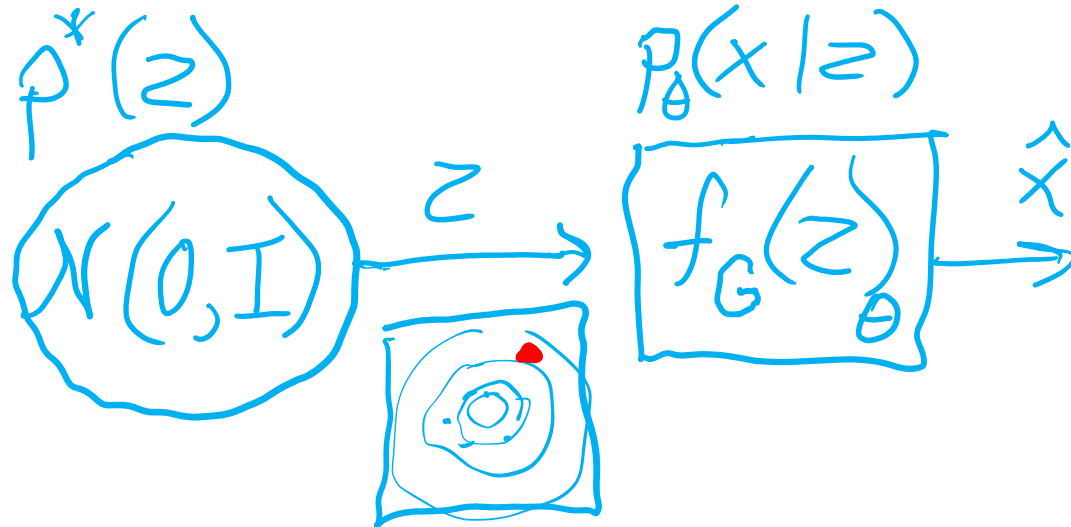


Want: sample \mathbf{z} from a simple distribution, e.g. Gaussian

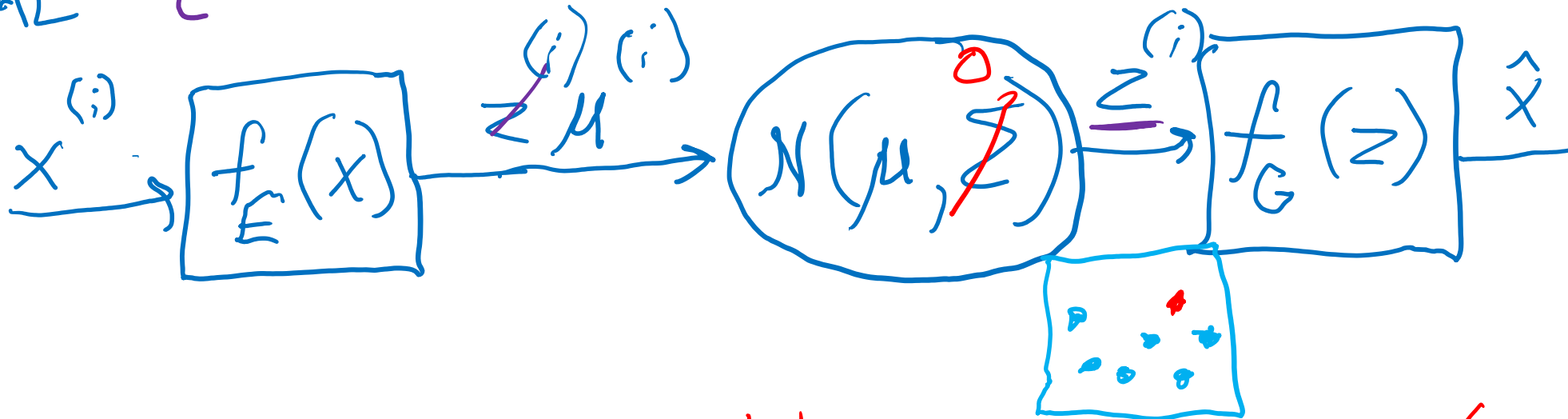
→ then pass that random \mathbf{z} through generator network

From Autoencoders to VAE

Want



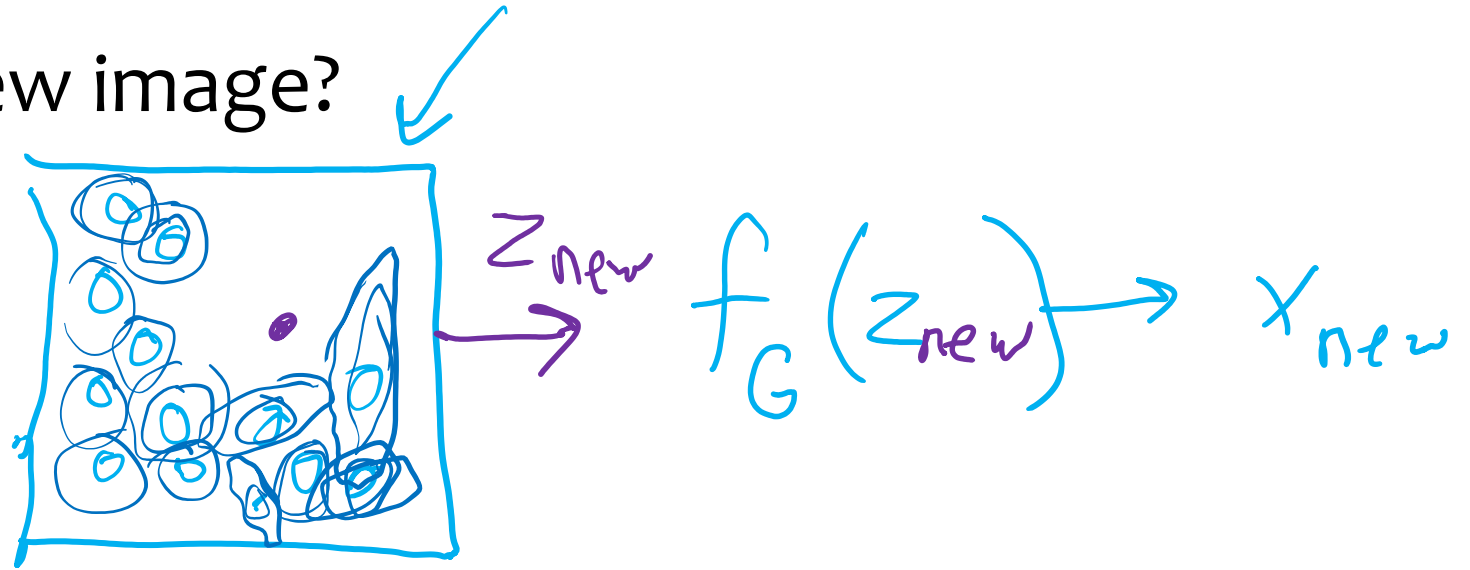
AE $q(z|x)$



∩ point masses (no variance)

Sampling $p(x)$ from Autoencoder

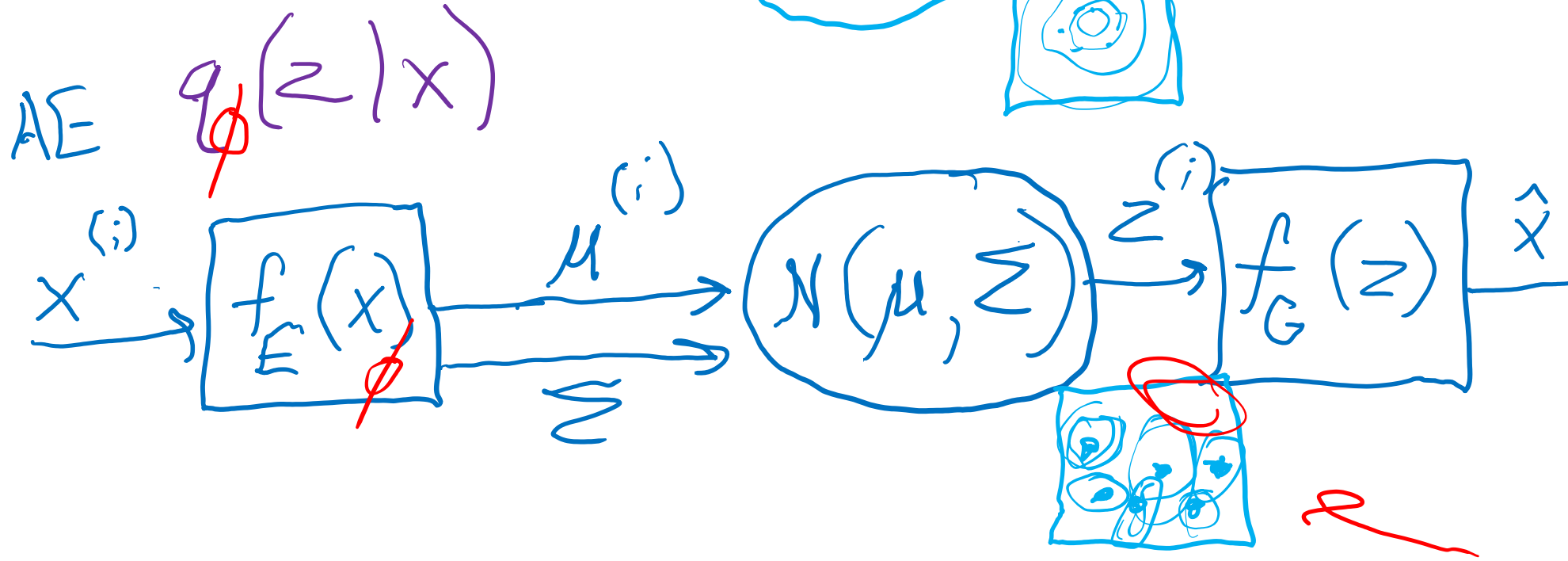
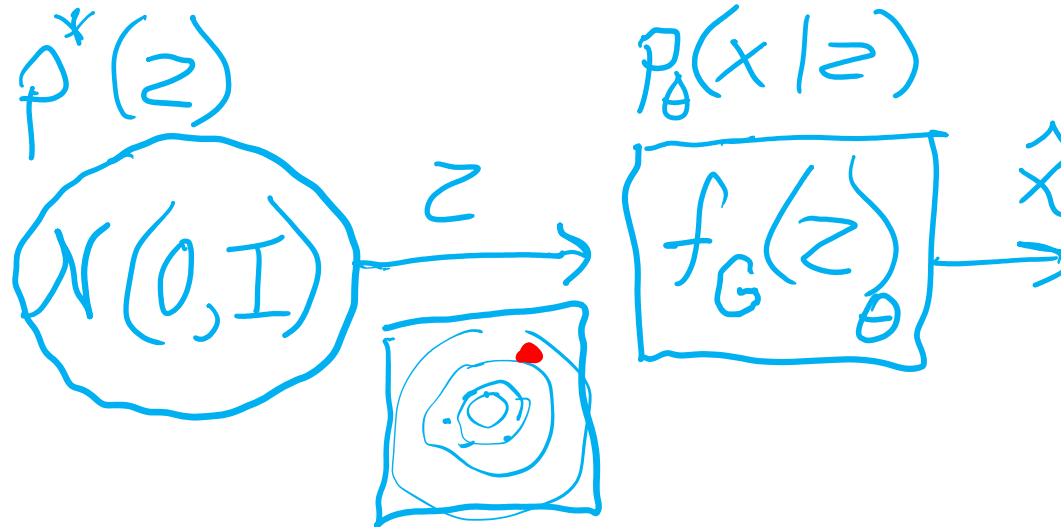
How do we sample a new image?



Let's start by filling in the gaps with some variance

From Autoencoders to VAE

Want



Learn variance from encoder network as well!