10-423/10-623 Generative AI

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Diffusion Models

Matt Gormley & Henry Chai
Lecture 8
Sep. 23, 2024

Reminders

- **Homework 1: Generative Models of Text**
  - Out: Mon, Sep 9
  - Due: Mon, Sep 23 at 11:59pm
- **Quiz 2:**
  - In-class: Wed, Sep 25
  - Lectures 5-8
- **Homework 2: Generative Models of Images**
  - Out: Mon, Sep 23
  - Due: Mon, Oct 7 at 11:59pm

UNSUPERVISED LEARNING

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 $x_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

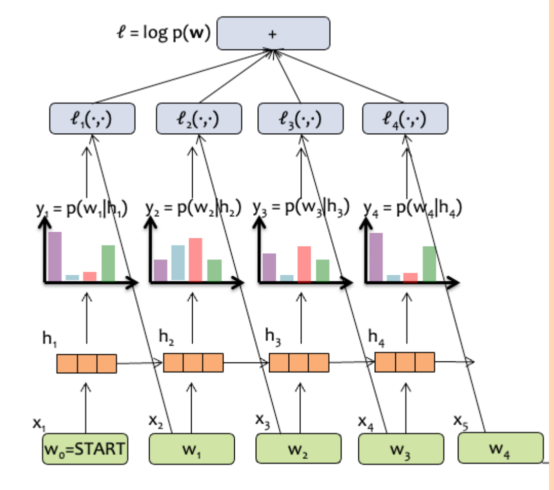
Goal: learn θ s.t. $p_\theta(\mathbf{x}_0) \approx p^*(\mathbf{x}_0)$

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 $x_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

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



Example: autoregressive LMs

- true $p^*(\mathbf{x}_0)$ is the (human) process that produced text on the web
- choose $p_\theta(\mathbf{x}_0)$ to be an autoregressive language model
  - autoregressive structure means that $p(\mathbf{x}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) \sim$ Categorical(.) and ancestral sampling is exact/efficient
- learn by finding
  $$\theta \approx \text{argmax}_\theta \log(p_\theta(\mathbf{x}_0))$$
  using gradient based updates on
  $$\nabla_\theta \log(p_\theta(\mathbf{x}_0))$$

Unsupervised Learning

Assumptions:

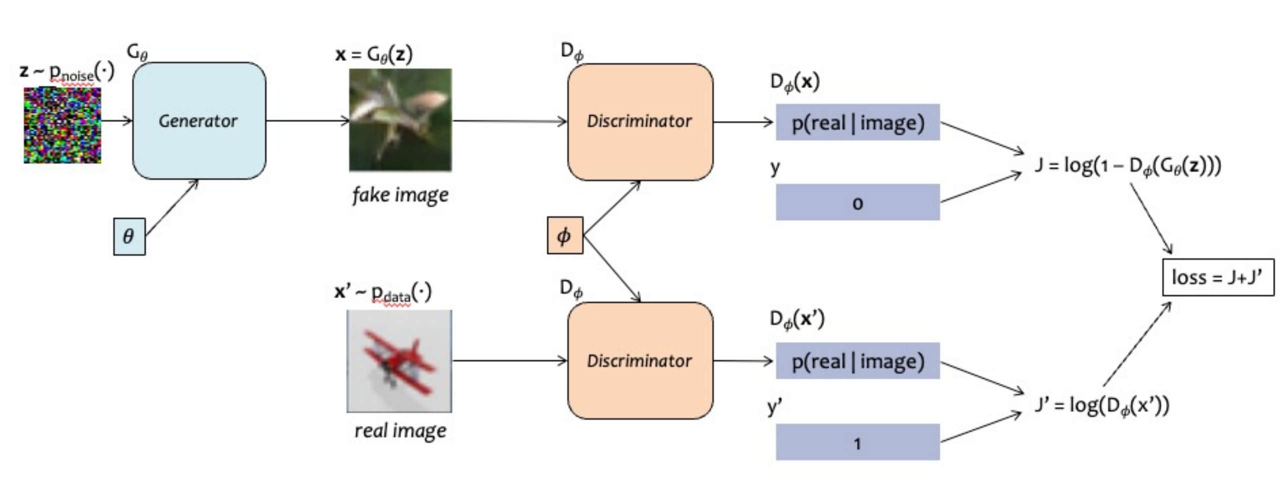
1. our data comes from some distribution $p^*(\mathbf{x}_0)$
2. we choose a distribution $p_\theta(\mathbf{x}_0)$ for which sampling $x_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

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

Example: GANs

- true $p^*(\mathbf{x}_0)$ is distribution over photos taken and posted to Flikr
- choose $p_\theta(\mathbf{x}_0)$ to be an expressive model (e.g. noise fed into inverted CNN) that can generate images
  - sampling is typically easy: $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$ and $\mathbf{x}_0 = f_\theta(\mathbf{z})$
- learn by finding $\theta \approx \text{argmax}_\theta \log(p_\theta(\mathbf{x}_0))$?
  - No! Because we can't even compute $\log(p_\theta(\mathbf{x}_0))$ or its gradient
  - Why not? Because the integral is intractable even for a simple 1-hidden layer neural network with nonlinear activation

$$p(\mathbf{x}_0) = \int_{\mathbf{z}} p(\mathbf{x}_0 \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$



so optimize a minimax loss instead

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 $x_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

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

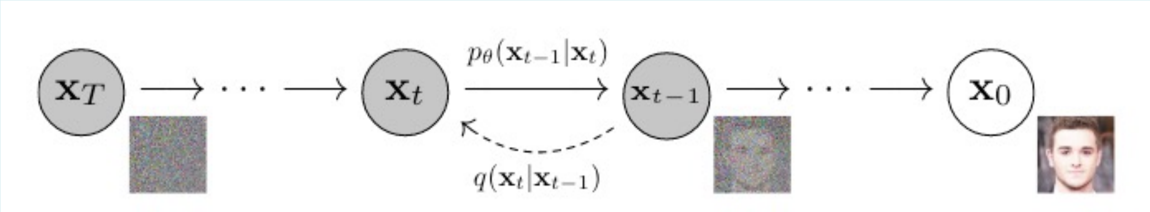


Figure from Ho et al. (2020)

Example: VAEs / Diffusion Models

- true $p^*(\mathbf{x}_0)$ is distribution over photos taken and posted to Flikr
- 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 $\theta \approx \text{argmax}_\theta \log(p_\theta(\mathbf{x}_0))$?
  - 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)

Latent Variable Models

- For GANs and VAEs, we assume that there are (unknown) **latent variables** which give rise to our observations
- The **vector z** are those latent variables
- After learning a GAN or VAE, we can **interpolate** between images in latent **z** space

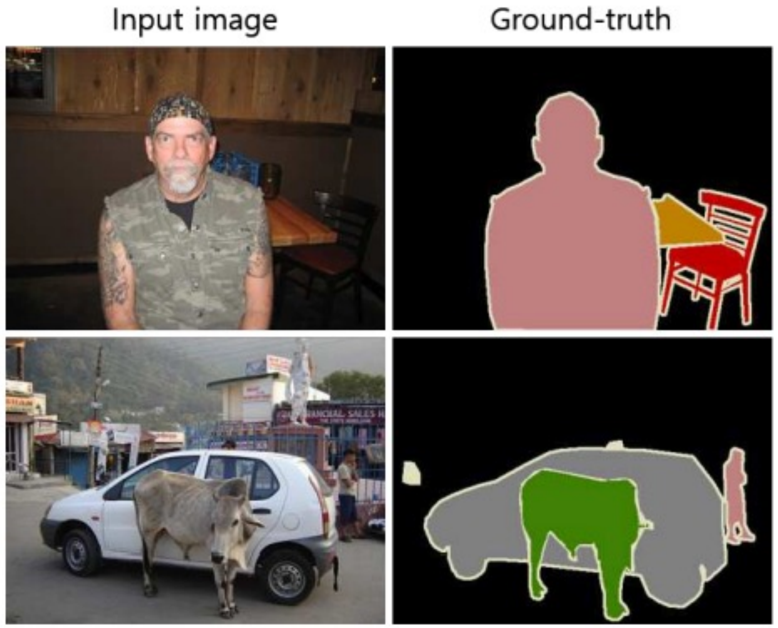
Figure 4: Top rows: Interpolation between a series of 9 random points in Z show that the space learned has smooth transitions, with every image in the space plausibly looking like a bedroom. In the 6th row, you see a room without a window slowly transforming into a room with a giant window. In the 10th row, you see what appears to be a TV slowly being transformed into a window.

Figure from Radford et al. (2016)

U-NET

Semantic Segmentation

- Given an image, predict a label for every pixel in the image
- Not merely a classification problem, because there are strong correlations between pixel-specific labels



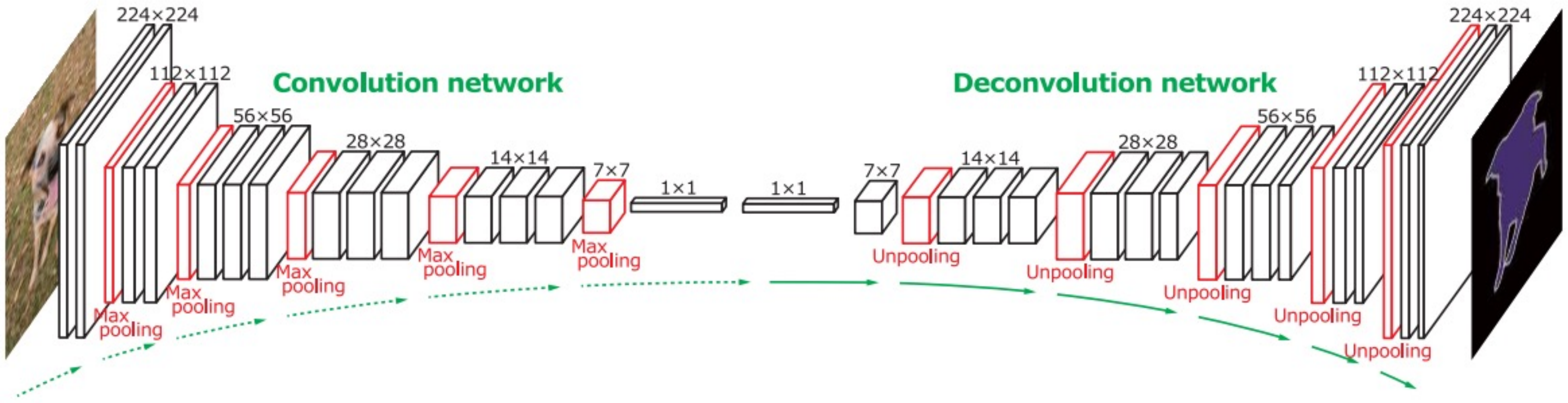


Figure from https://openaccess.thecvf.com/content_iccv_2015/papers/Noh_Learning_Deconvolution_Network_ICCV_2015_paper.pdf

Instance Segmentation

- Predict per-pixel labels as in semantic segmentation, but differentiate between different instances of the same label
- *Example*: if there are two people in the image, one person should be labeled **person-1** and one should be labeled **person-2**

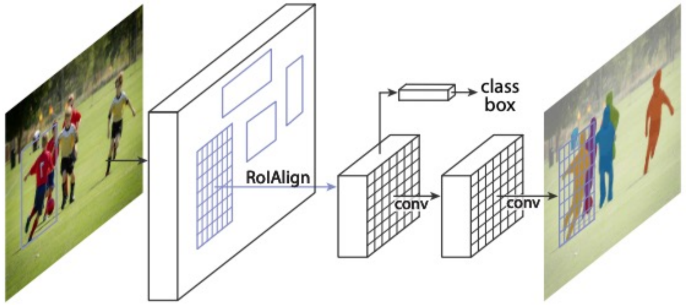
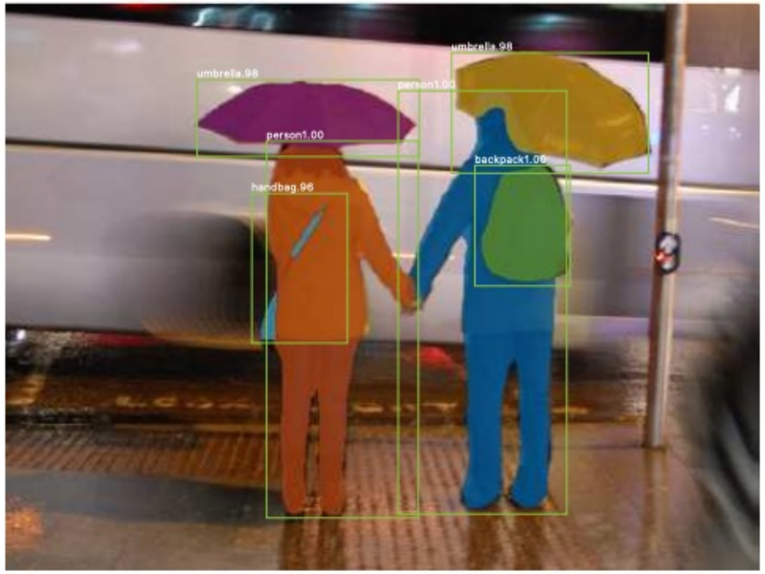


Figure 1. The **Mask R-CNN** framework for instance segmentation.



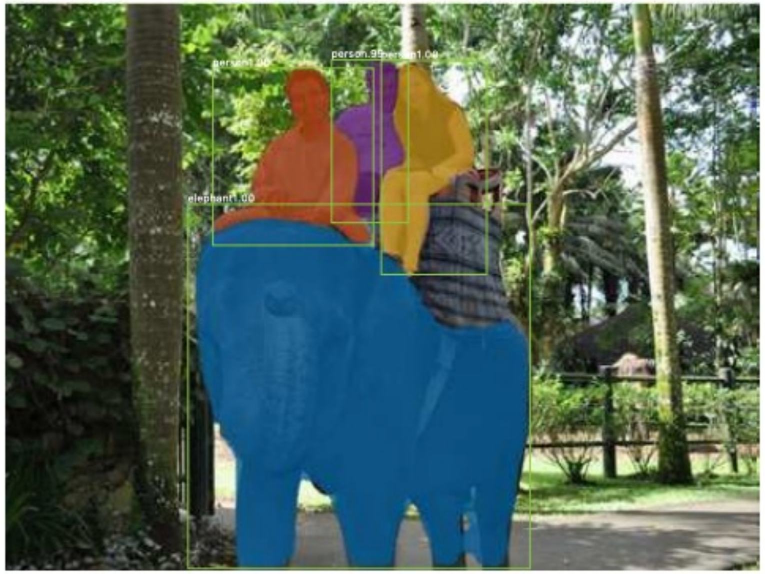


Figure from https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf

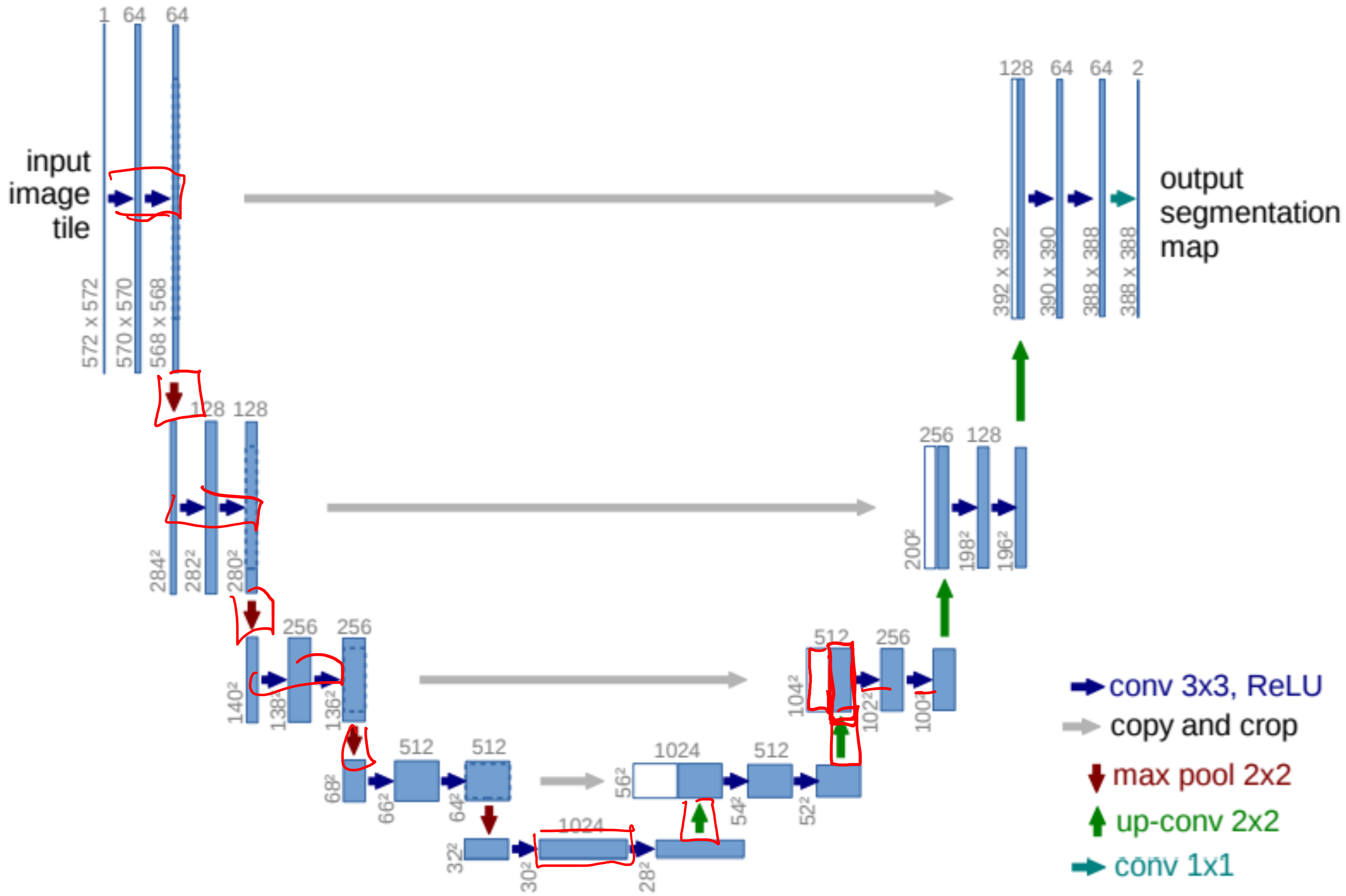
U-Net

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



U-Net

- Originally designed for applications to biomedical segmentation
- Key observation is that the output layer has the **same** dimensions as the input image (possibly with different number of channels)

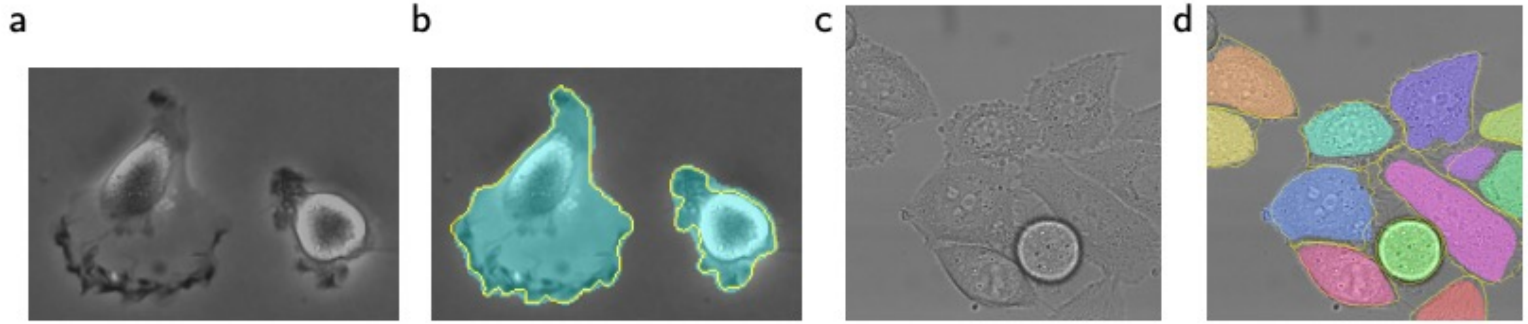


Fig. 4. Result on the ISBI cell tracking challenge. (**a**) part of an input image of the "PhC-U373" data set. (**b**) Segmentation result (cyan mask) with manual ground truth (yellow border) (**c**) input image of the "DIC-HeLa" data set. (**d**) Segmentation result (random colored masks) with manual ground truth (yellow border).

DIFFUSION MODELS

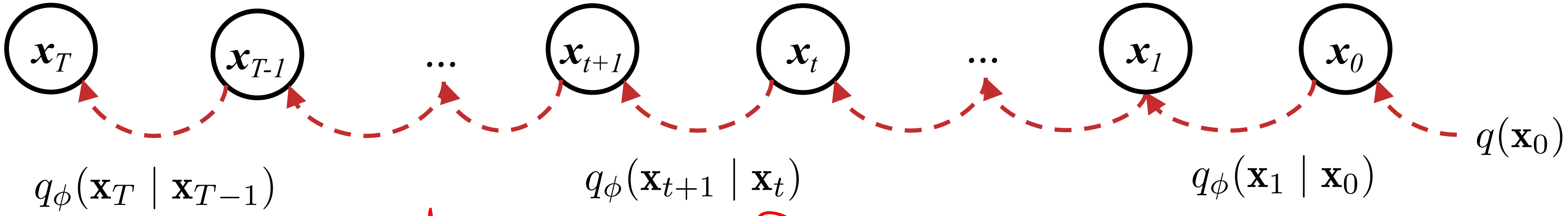
Diffusion Models

- Next we will consider (1) **diffusion mo** **variational autoencoders (VAEs)**
  - Although VAEs came first, we're going to models since they will receive more of o
- The steps in defining these models is
  - Define a probability distribution involvin
  - Use a variational lower bound as an obje
  - Learn the parameters of the probab the objective function
- So what is a variational lower bound?

The standard presentation of diffusion models requires an understanding of variational inference. (we'll do that next time)

Today, we'll do an alternate presentation without variational inference!

Diffusion Model



1

Forward Process:

data distribution

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

$$q(x_{t-1} \mid x_t) = \frac{q(x_t, x_{t-1})}{q(x_t)}$$

2

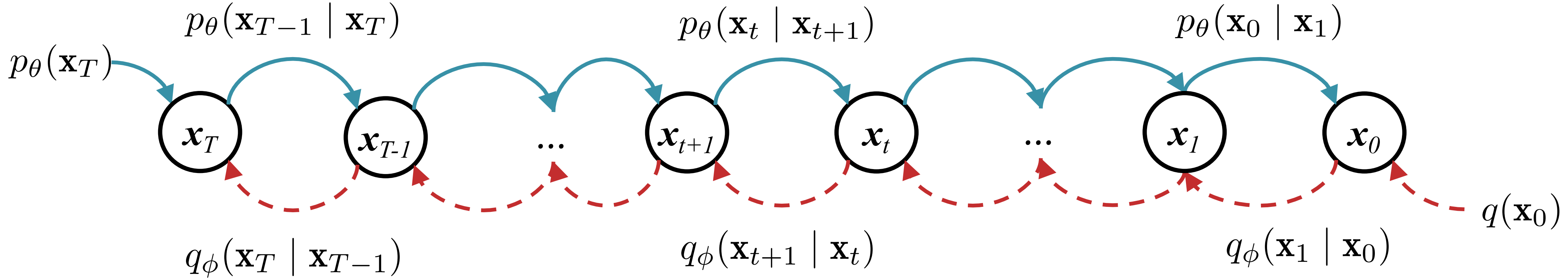
(Exact) Reverse Process:

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

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

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

Diffusion Model



Forward Process:

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

(Learned) Reverse Process:

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

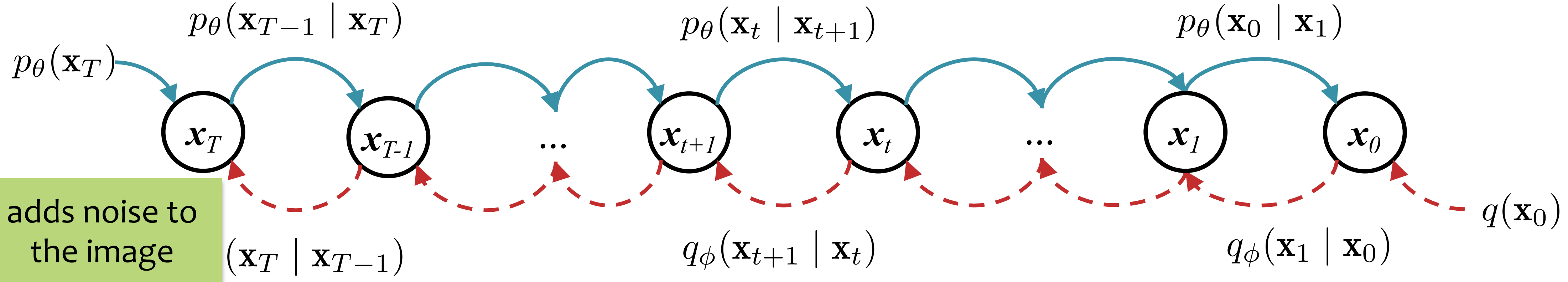
(Exact) Reverse Process:

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

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

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

Diffusion Model



adds noise to the image

Forward Process:

if we could sample from this we'd be done

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

(Learned) Reverse Process:

removes noise

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

goal is to learn this

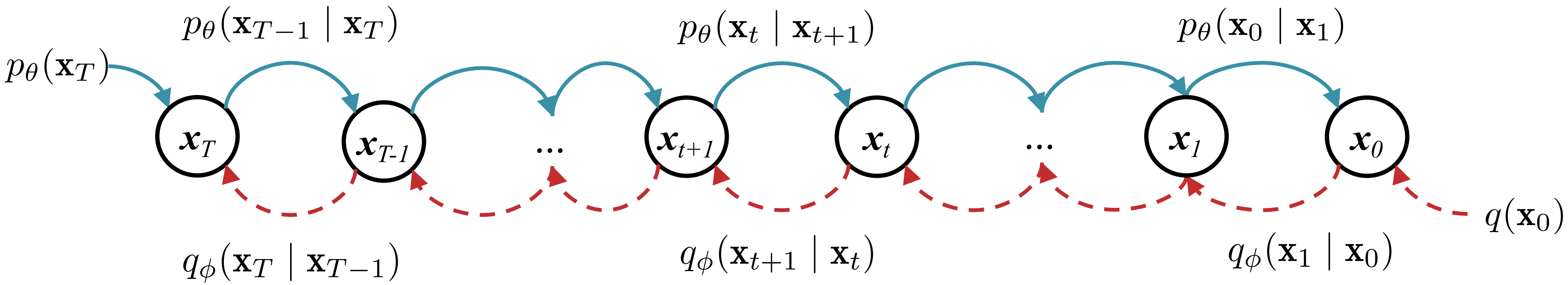
(Exact) Reverse Process:

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

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

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

Diffusion Model



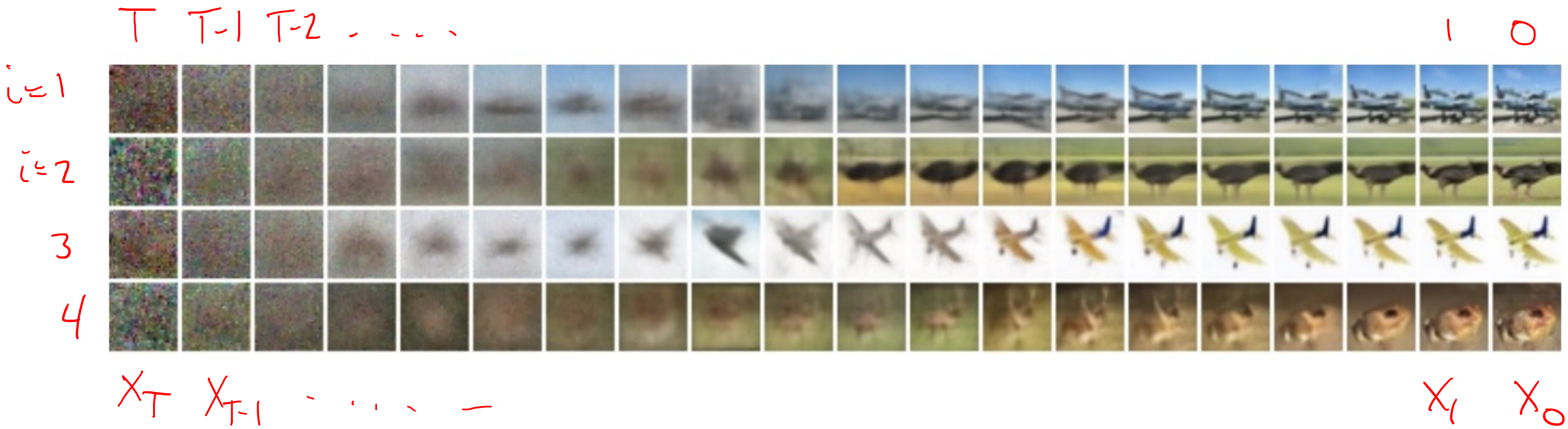
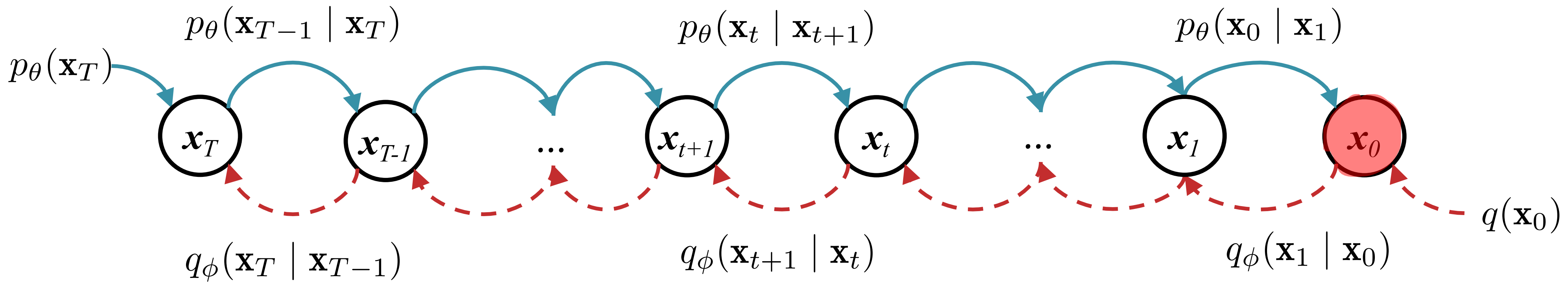
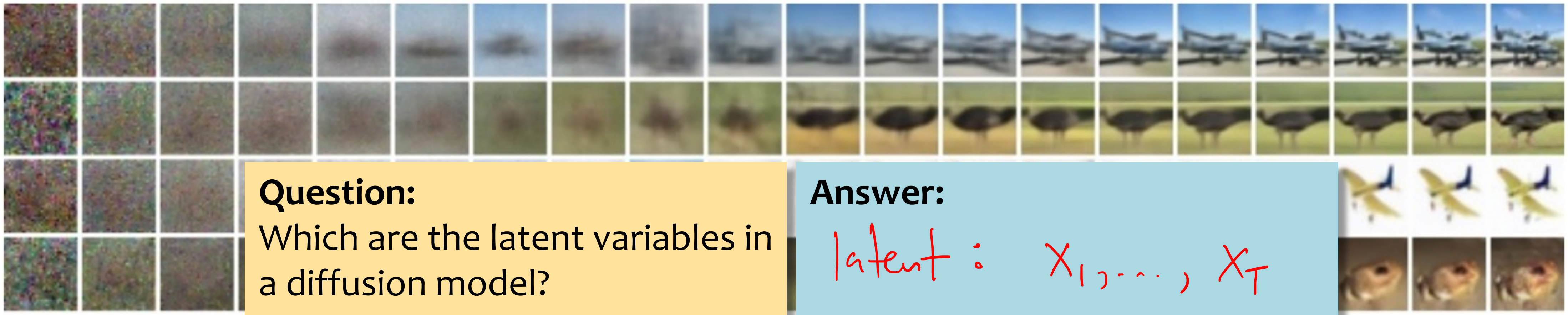


Figure from Ho et al. (2020)

Diffusion Model



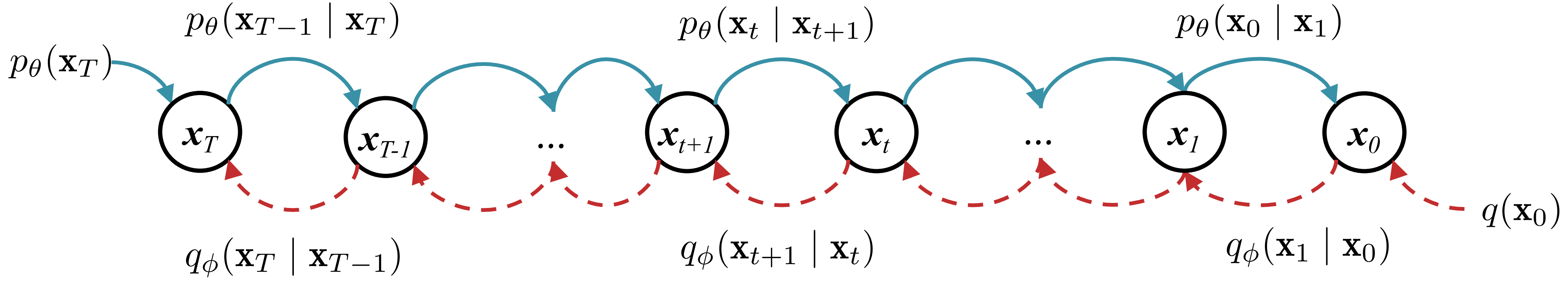


Question:
Which are the latent variables in a diffusion model?

Answer:
latent: $x_1, \ldots, x_T$
observed: x_0

Figure from Ho et al. (2020)

Denoising Diffusion Probabilistic Model (DDPM)



Forward Process:

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

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

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

$x_t = \sqrt{\alpha_t}\, x_{t-1} + \epsilon \sqrt{1-\alpha_t}\, I$ where $\epsilon \sim \mathcal{N}(0, I)$

(Learned) Reverse Process:

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

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

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

(something w/ U-Net

Defining the Forward Process

Forward Process:

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

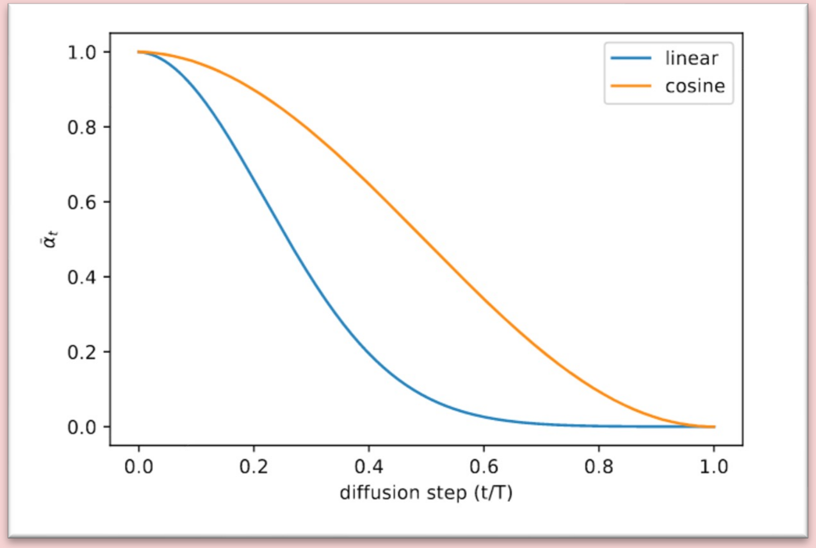
$$q(\mathbf{x}_0) = \text{data distribution}$$

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

$\phi = [\alpha_1, \alpha_2, \ldots, \alpha_T]$

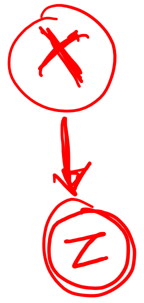
Noise schedule:

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 $p_\theta(\mathbf{x}_T)$.



Gaussian (an aside)

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



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

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

Defining the Forward Process

Forward Process:

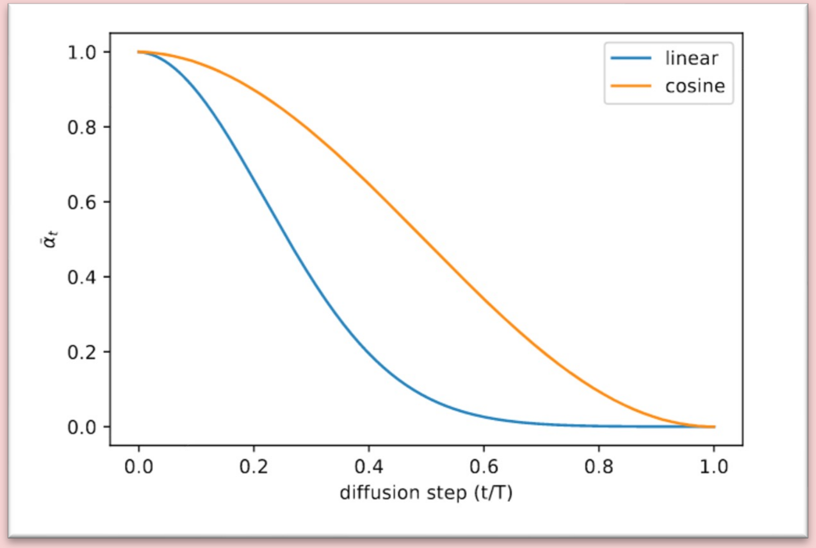
$$q_\phi(\mathbf{x}_{0:T}) = q(\mathbf{x}_0) \prod_{t=1}^{T} q_\phi(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

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

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

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:

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

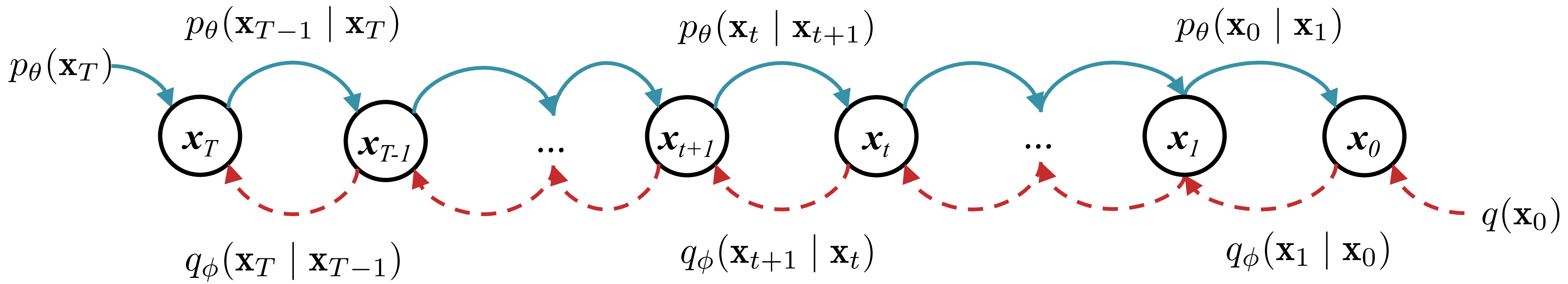
$$\text{where } \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s = a_1 a_2 \dots a_t$$

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

A:

$$q_\phi(x_T \mid x_0) \sim \mathcal{N}(\mu \approx 0, \Sigma \approx I)$$

Diffusion Model



Forward Process:

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

(Learned) Reverse Process:

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

Q: If q_ϕ is just adding noise, how can p_θ be interesting at all?

A:

Q: But if $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is Gaussian, how can it learn a θ such that $p_\theta(\mathbf{x}_0) \approx q(\mathbf{x}_0)$? Won't $p_\theta(\mathbf{x}_0)$ be Gaussian too?

A:

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

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

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

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

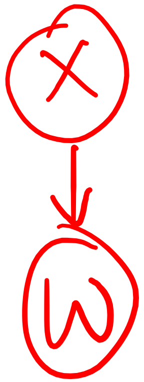
2. Difference of two Gaussians is a Gaussian

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

3. Gaussian with a Gaussian mean has a Gaussian Conditional

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

$$\Rightarrow P(Z) \sim \mathcal{N}(\cdot, \cdot)$$

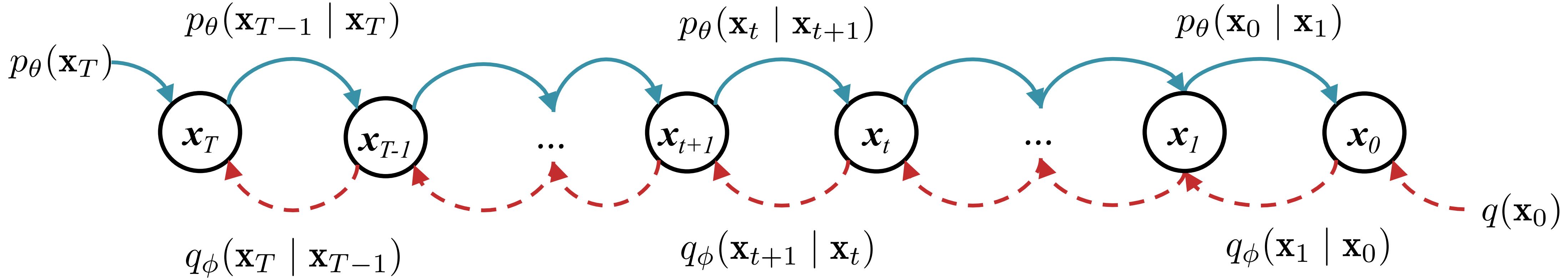


4. But #3 does not hold if X is passed through a nonlinear function f

$$W \sim \mathcal{N}(\mu_z = f(X), \sigma_w^2) \Rightarrow P(W \mid X) \sim \mathcal{N}(\cdot, \cdot)$$

$$\not\Rightarrow P(W) \sim \mathcal{N}(\cdot, \cdot)$$

Diffusion Model



Forward Process:

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

(Learned) Reverse Process:

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

Q: If q_ϕ is just adding noise, how can p_θ be interesting at all?

A:

$\sim \mathcal{N}(\mu_\theta(x_t), \cdot)$

Q: But if $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is Gaussian, how can it learn a θ such that $p_\theta(\mathbf{x}_0) \approx q(\mathbf{x}_0)$? Won't $p_\theta(\mathbf{x}_0)$ be Gaussian too?

A:

Diffusion Model Analogy

Properties of forward and *exact* reverse processes

Property #1:

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

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

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

$\Rightarrow$ $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ — this is the same reparameterization trick from VAEs

Properties of forward and *exact* reverse processes

Property #1:

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

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

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

$\Rightarrow$ $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ 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:

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

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

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

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

Parameterizing the *learned* reverse process

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

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.

Parameterizing the *learned* reverse process

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

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.

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

Parameterizing the *learned* reverse process

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

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.

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

Properties of forward and *exact* reverse processes

Property #1:

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

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

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

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

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

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

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

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

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:

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

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

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

$$= \alpha_t^{(0)}\left(\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}\right) / \sqrt{\bar{\alpha}_t}\right) + \alpha_t^{(t)}\mathbf{x}_t$$

$$= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{(1-\alpha_t)}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right)$$

Parameterizing the *learned* reverse process

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

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.

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

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

where $\mathbf{x}_\theta^{(0)}(\mathbf{x}_t, t) = \left(\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) / \sqrt{\bar{\alpha}_t}$

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

Learning the Reverse Process

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

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.

Algorithm 1 Training (Option A, all timesteps)

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