



10-423/10-623/10-723 Generative AI

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Diffusion Models

Matt Gormley & Aran Nayebi
Lecture 7
Sep. 17, 2025

Reminders

- **Homework 1: Generative Models of Text**
 - Out: Mon, Sep 8
 - Due: Mon, Sep 22 at 11:59pm
- **Quiz 2:**
 - In-class: Wed, Sep 24
 - Lectures 5-8
- **Homework 2: Generative Models of Images**
 - Out: Mon, Sep 22
 - Due: Sat, Oct 4 at 11:59pm

UNSUPERVISED LEARNING

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. $p_\theta(\mathbf{x}_o) \approx p^*(\mathbf{x}_o)$

Unsupervised Learning

Assumptions:

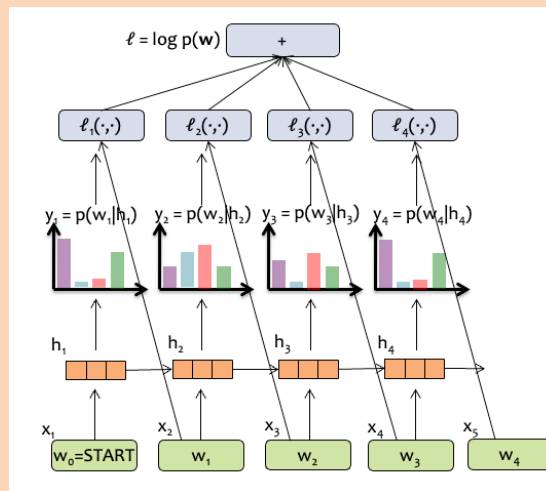
1. our data comes from some distribution $p^*(\mathbf{x}_0)$
2. we choose a distribution $p_\theta(\mathbf{x}_0)$ for which sampling $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

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

Example: 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 $p(\mathbf{x}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1}) \sim \text{Categorical}(\cdot)$ and ancestral sampling is exact/efficient
- learn by finding

$$\theta \approx \operatorname{argmax}_\theta \log(p_\theta(\mathbf{x}_0))$$
 using gradient based updates on $\nabla_\theta \log(p_\theta(\mathbf{x}_0))$



Unsupervised Learning

Assumptions:

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

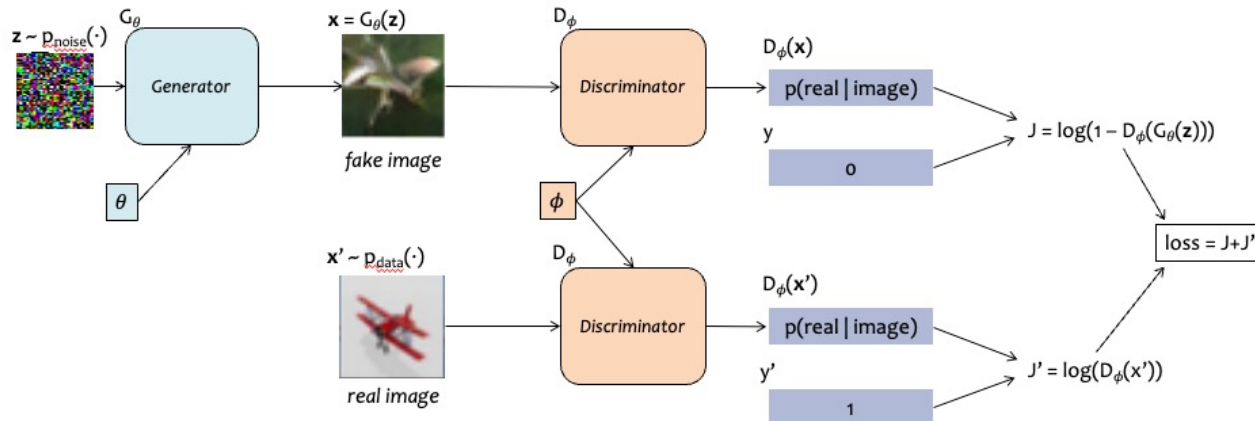
Goal: learn θ s.t. $p_\theta(\mathbf{x}_o) \approx p^*(\mathbf{x}_o)$

Example: GANs

- 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 typically easy:
 $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$ and $\mathbf{x}_o = f_\theta(\mathbf{z})$
- learn by finding $\theta \approx \operatorname{argmax}_\theta \log(p_\theta(\mathbf{x}_o))$?
 - No! Because we can't even compute $\log(p_\theta(\mathbf{x}_o))$ or its gradient
 - Why not? Because the integral is intractable even for a simple 1-hidden layer neural network with nonlinear activation

$$p(\mathbf{x}_o) = \int_{\mathbf{z}} p(\mathbf{x}_o | \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

so optimize a minimax loss instead



Unsupervised Learning

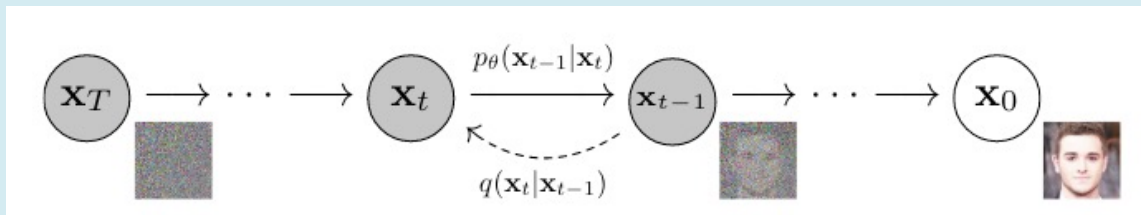
Assumptions:

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



Latent Variable Models

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

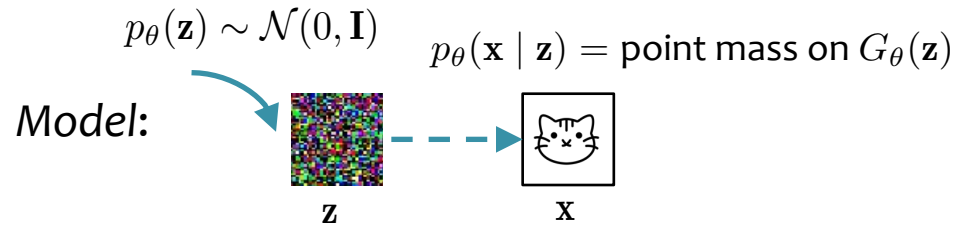
GAN → VAE → Diffusion

(in 15 minutes)

GAN → VAE → Diffusion

(in 5 minutes)

GANs

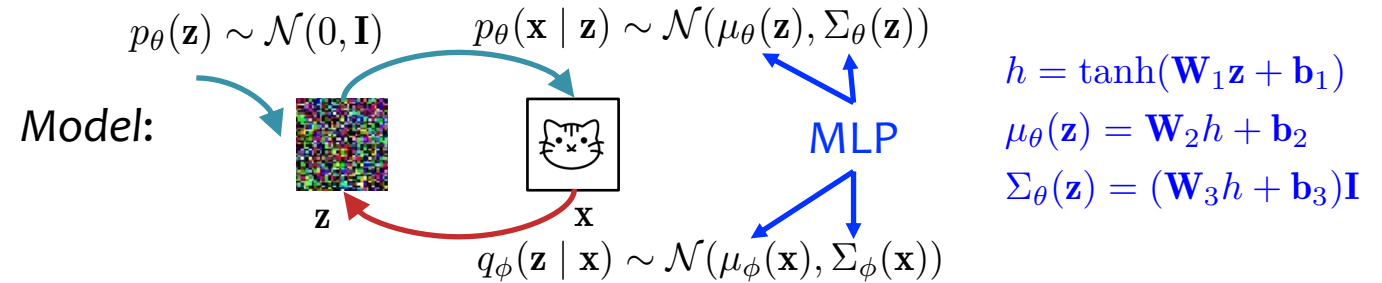


MLE?: $\hat{\theta} = \arg \max_{\theta} \sum_{i=1}^N \log p(\mathbf{x}^{(i)})$ NOPE!

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

Learn: minimax game

VAEs



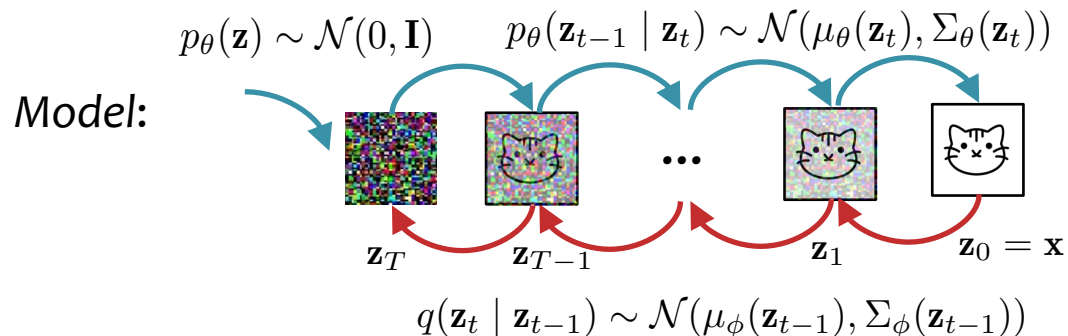
MLE?: $\hat{\theta} = \arg \max_{\theta} \sum_{i=1}^N \log p(\mathbf{x}^{(i)})$ NOPE!

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

Learn: $\theta, \phi = \arg \min_{\theta, \phi} \text{KL}(q_{\phi}(\mathbf{z} | \mathbf{x}) \| p_{\theta}(\mathbf{z} | \mathbf{x}))$

where $p_{\theta}(\mathbf{z} | \mathbf{x})$ is true posterior of $p(\mathbf{x} | \mathbf{z}) p(\mathbf{z})$

Diffusion



Learn: $\theta = \arg \min_{\theta} \text{KL}(q(\mathbf{z}_{1:T} | \mathbf{x}) \| p_{\theta}(\mathbf{z}_{1:T} | \mathbf{x}))$

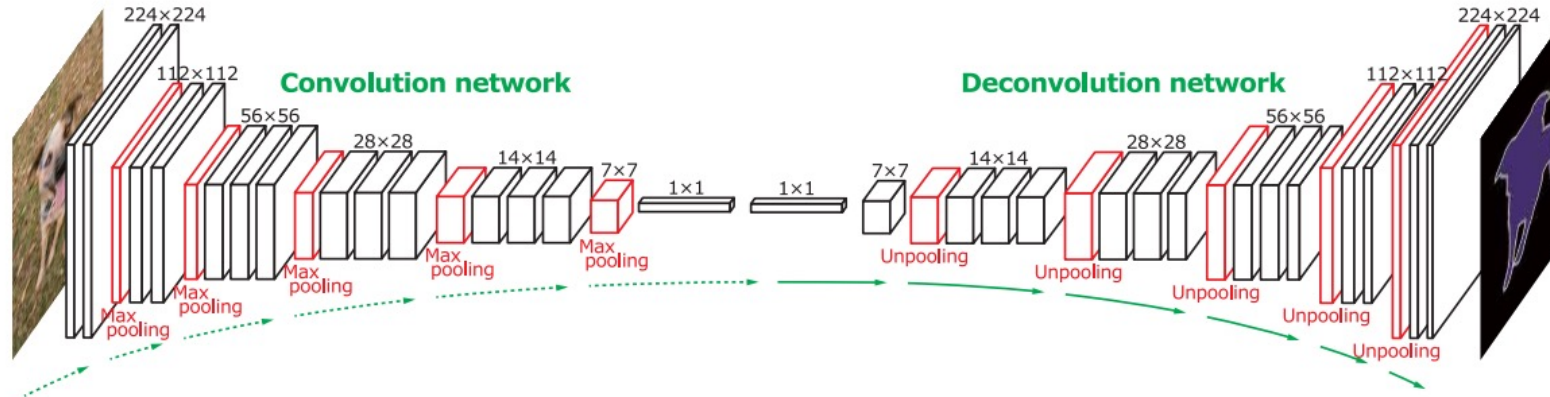
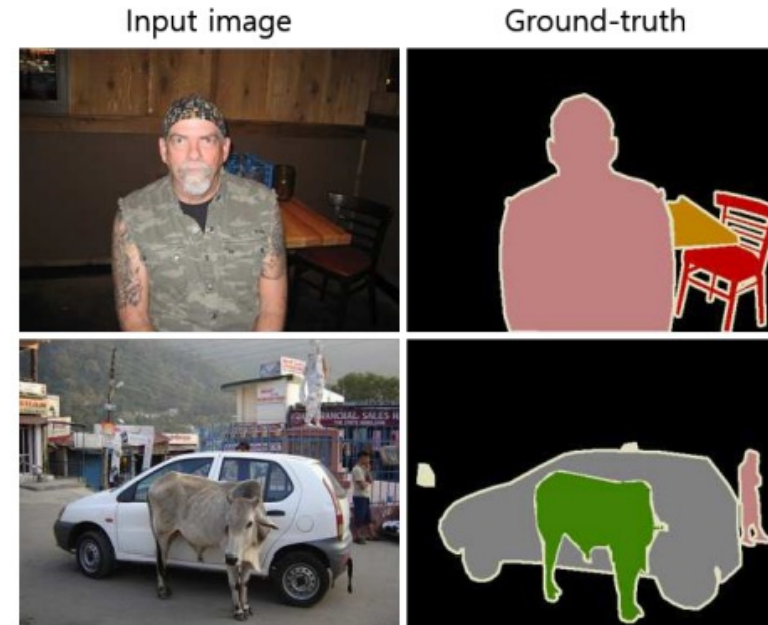
where $p_{\theta}(\mathbf{z}_{1:T} | \mathbf{x})$ is true posterior of $p(\mathbf{x} | \mathbf{z}_1) \cdots p(\mathbf{z}_{T-1} | \mathbf{z}_T) p(\mathbf{z}_T)$

MLE?: NOPE!

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



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**

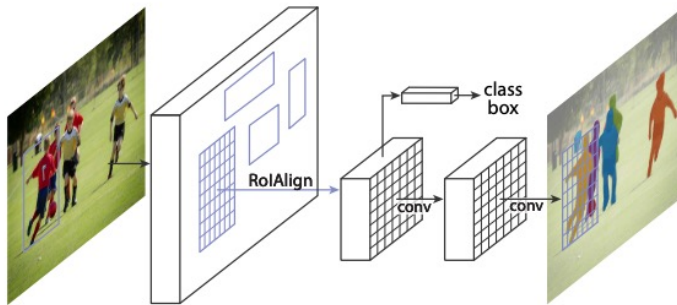
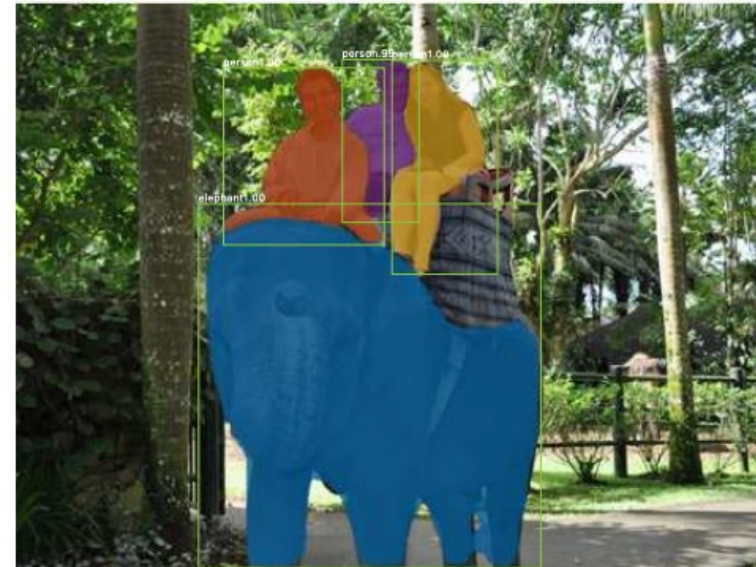
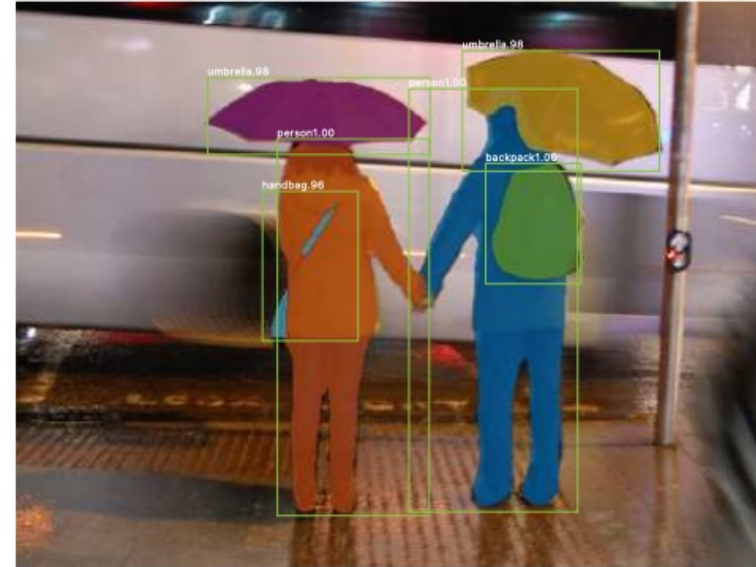


Figure 1. The **Mask R-CNN** framework for instance segmentation.

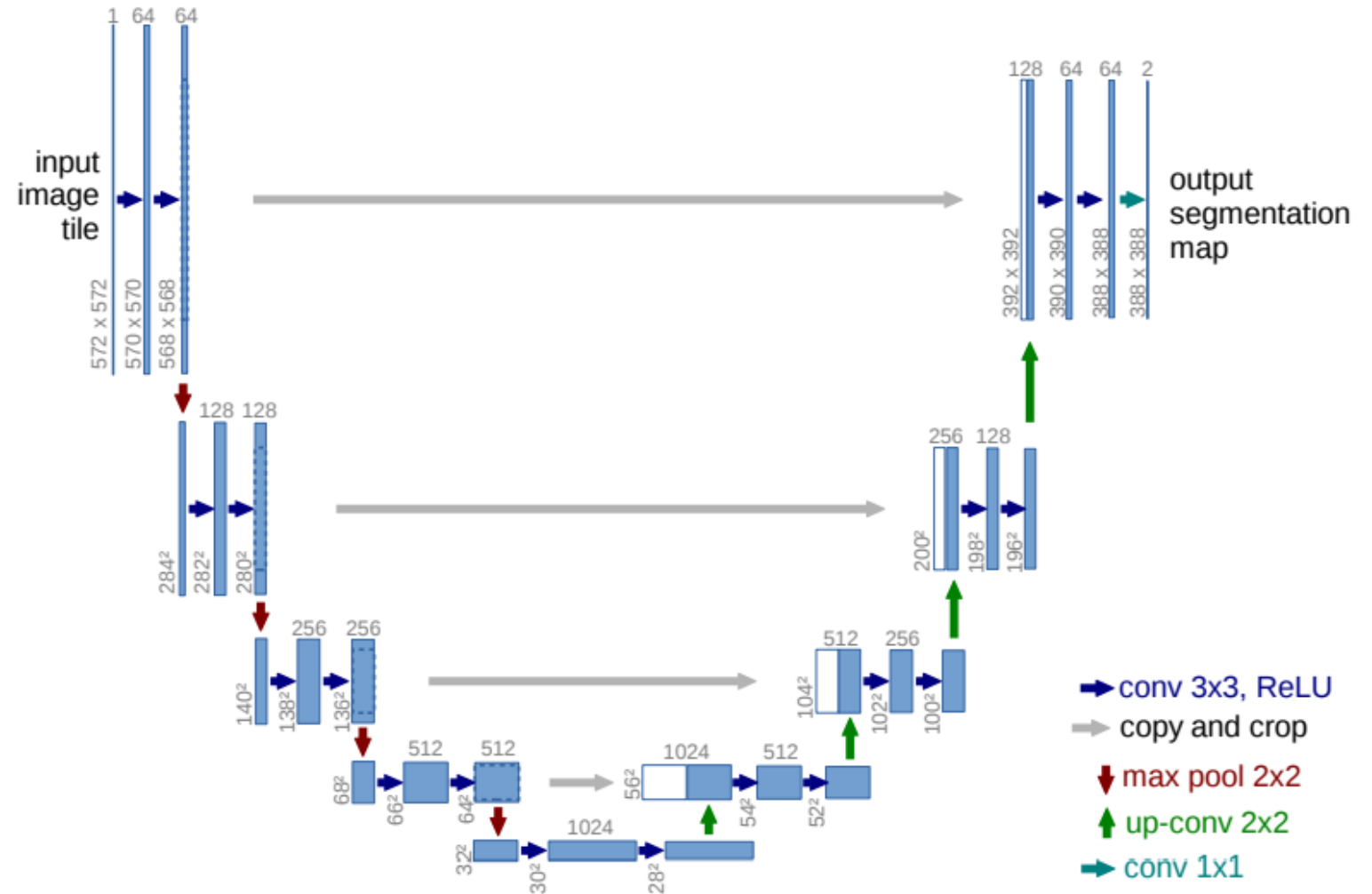
U-Net

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)

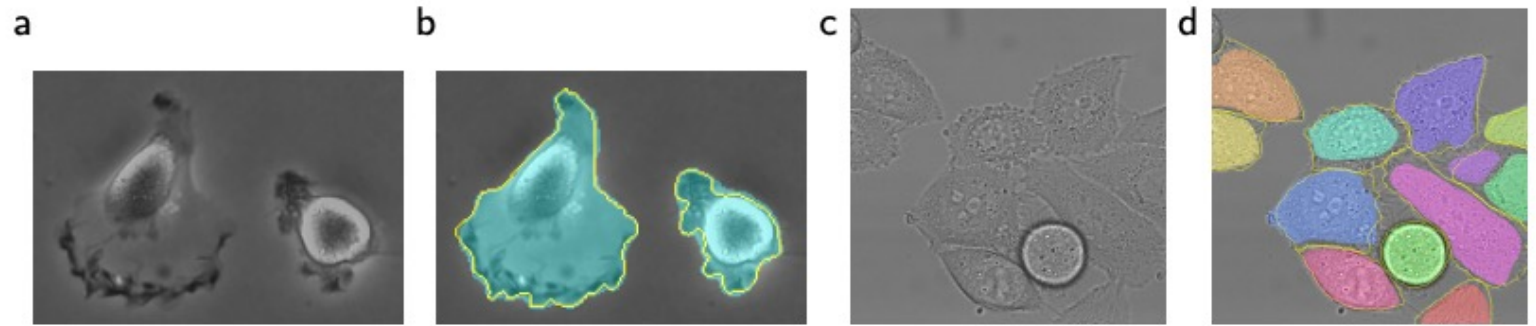
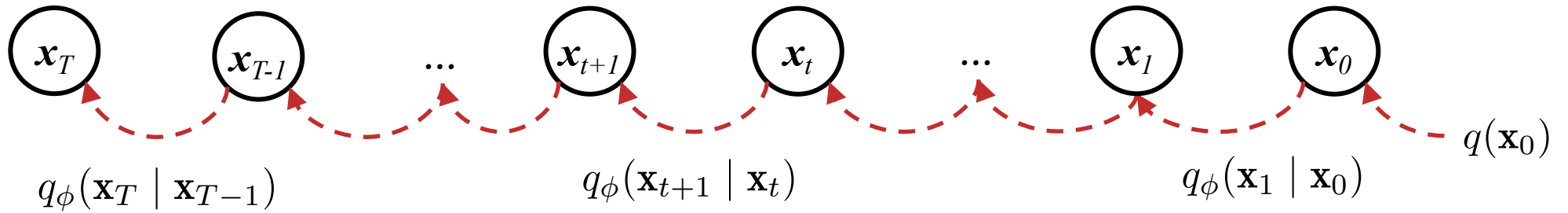


Fig. 4. Result on the ISBI cell tracking challenge. (a) part of an input image of the “PhC-U373” data set. (b) Segmentation result (cyan mask) with manual ground truth (yellow border) (c) input image of the “DIC-HeLa” data set. (d) Segmentation result (random colored masks) with manual ground truth (yellow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

DIFFUSION MODELS

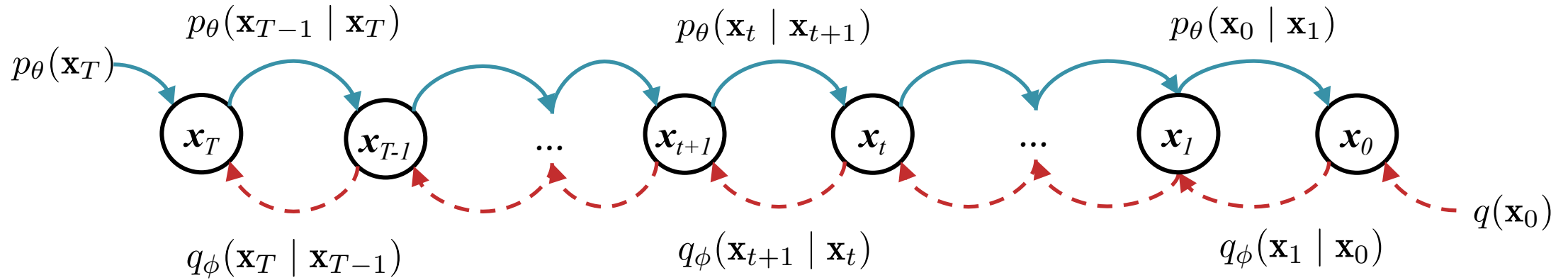
Diffusion Model



Forward Process:

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

Diffusion Model



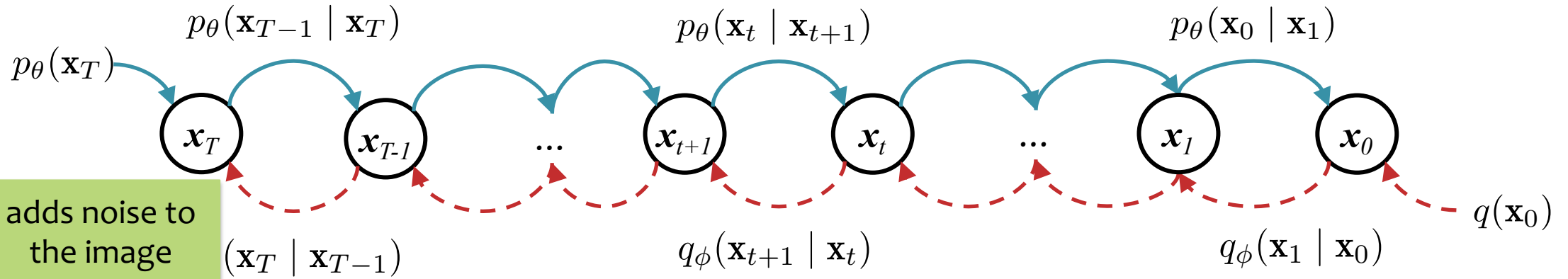
Forward Process:

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

(Learned) Reverse Process:

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

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

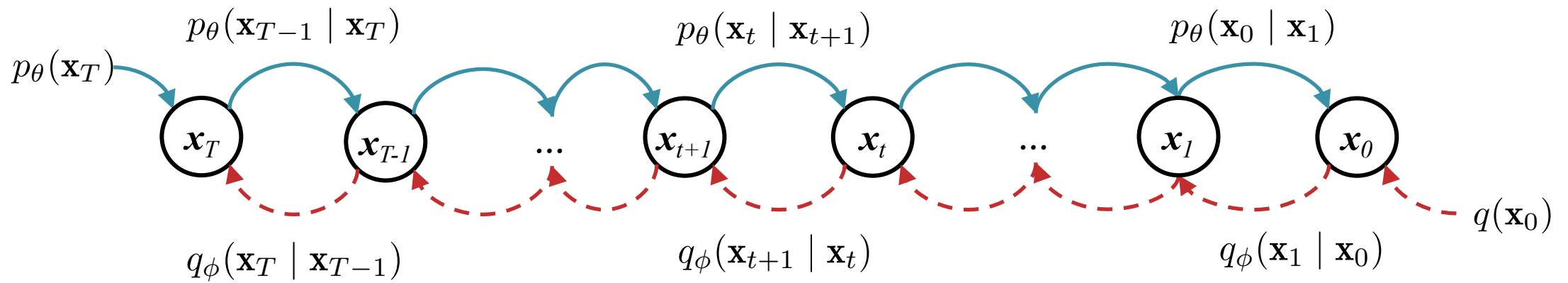
(Learned) Reverse Process:

removes noise

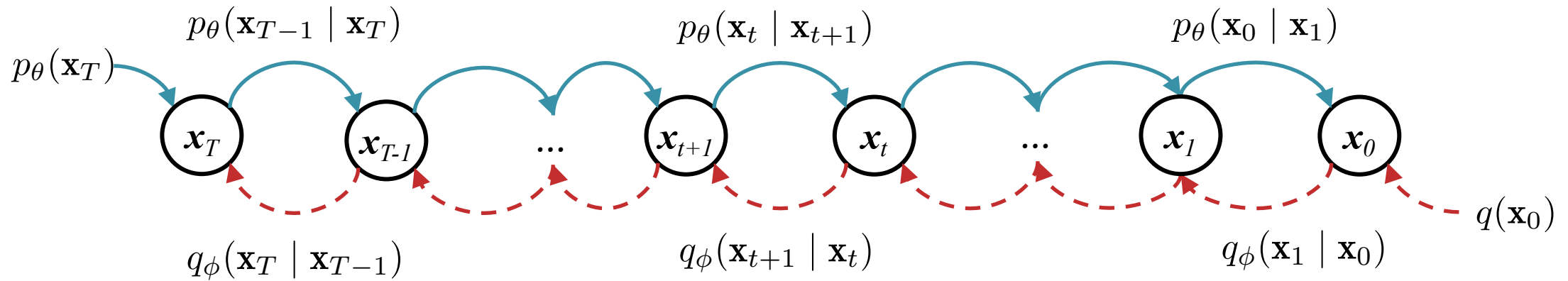
$$p_\theta(\mathbf{x}_{0:T}) = p_\theta(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$$

goal is to learn this

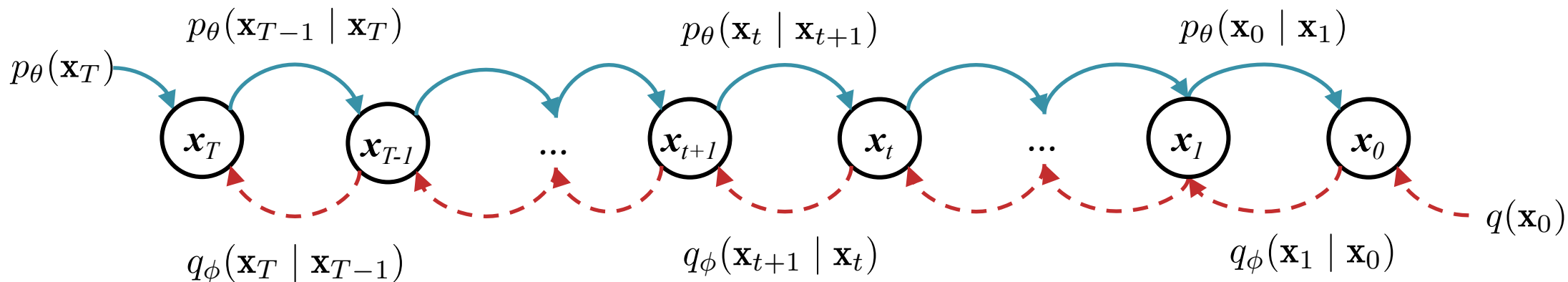
Diffusion Model



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

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

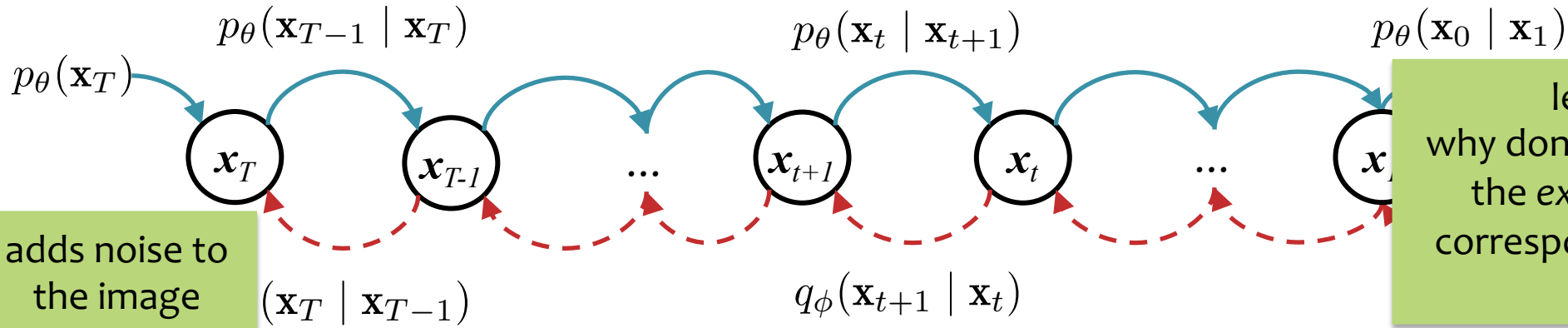
(Learned) Reverse Process:

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

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

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

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

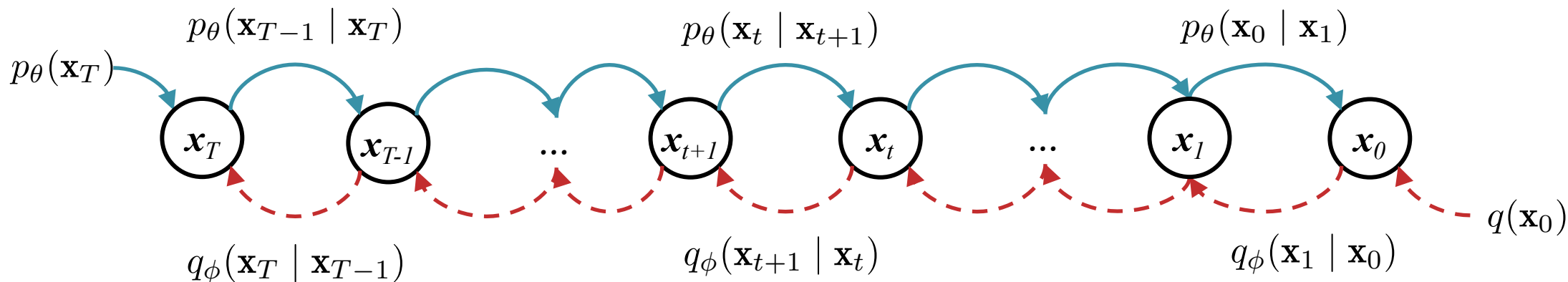
(Exact) Reverse Process:

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

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.

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

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

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

(Learned) Reverse Process:

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

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

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

Defining the Forward Process

Forward Process:

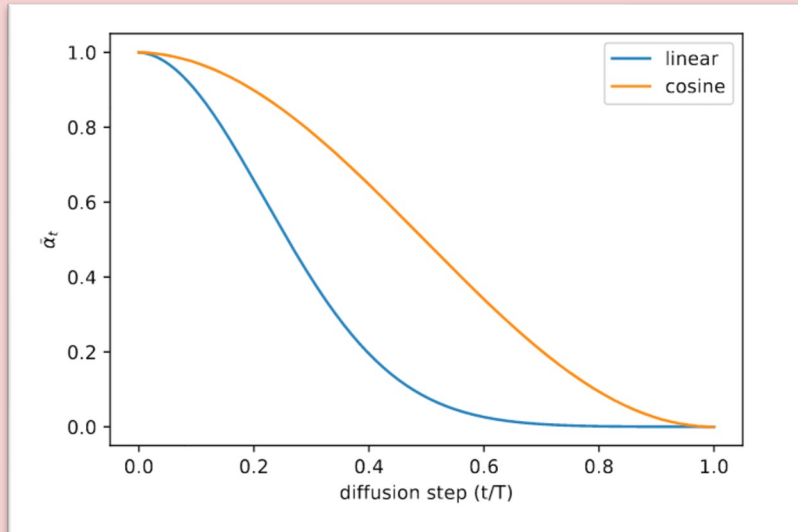
$$q_{\phi}(\mathbf{x}_{0:T}) = q(\mathbf{x}_0) \prod_{t=1}^T q_{\phi}(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

$q(\mathbf{x}_0)$ = 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 $p_{\theta}(\mathbf{x}_T)$.



Gaussians (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.

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

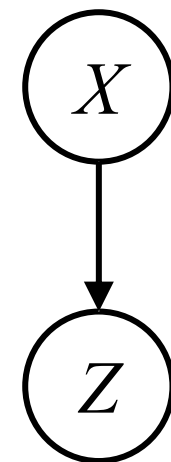
2. Difference of two Gaussians is a Gaussian.

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

Let $Z | X \sim \mathcal{N}(\mu_z = g(X), \sigma_z^2)$

3. Gaussian with a Gaussian mean passed through a *linear/affine* function $g(\cdot)$ has a Gaussian marginal,

$$p_Z(z) = \int_x p_{Z|X}(z|x)p_X(x) dx \text{ is a Gaussian density}$$



Defining the Forward Process

Forward Process:

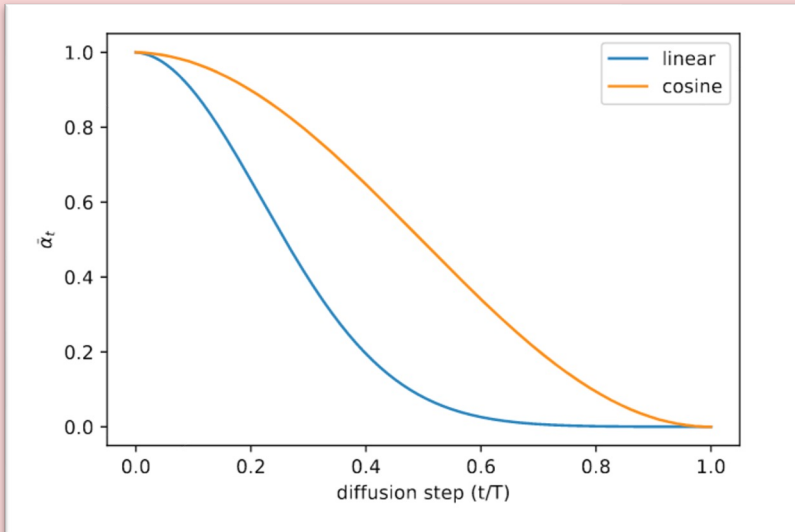
$$q_{\phi}(\mathbf{x}_{0:T}) = q(\mathbf{x}_0) \prod_{t=1}^T q_{\phi}(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

$q(\mathbf{x}_0)$ = 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:

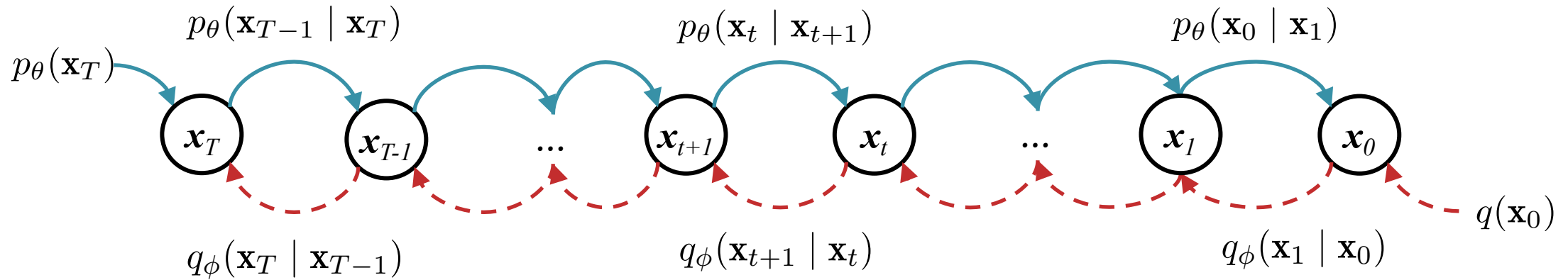
$$q(\mathbf{x}_t \mid \mathbf{x}_0) \sim \mathcal{N}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$$

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

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

A:

Diffusion Model



Forward Process:

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

(Learned) Reverse Process:

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

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:

Gaussians (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.

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

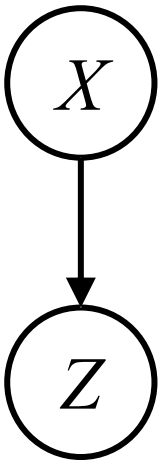
2. Difference of two Gaussians is a Gaussian.

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

Let $Z | X \sim \mathcal{N}(\mu_z = g(X), \sigma_z^2)$

3. Gaussian with a Gaussian mean passed through a *linear/affine* function $g(\cdot)$ has a Gaussian marginal,

$$p_Z(z) = \int_x p_{Z|X}(z|x)p_X(x) dx \text{ is a Gaussian density}$$



Gaussians (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.

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

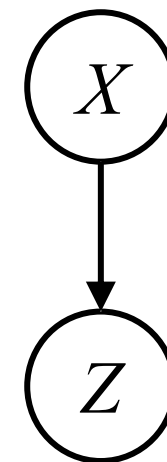
Let $Z | X \sim \mathcal{N}(\mu_z = g(X), \sigma_z^2)$

3. Gaussian with a Gaussian mean passed through a *linear/affine* function $g(\cdot)$ has a Gaussian marginal,

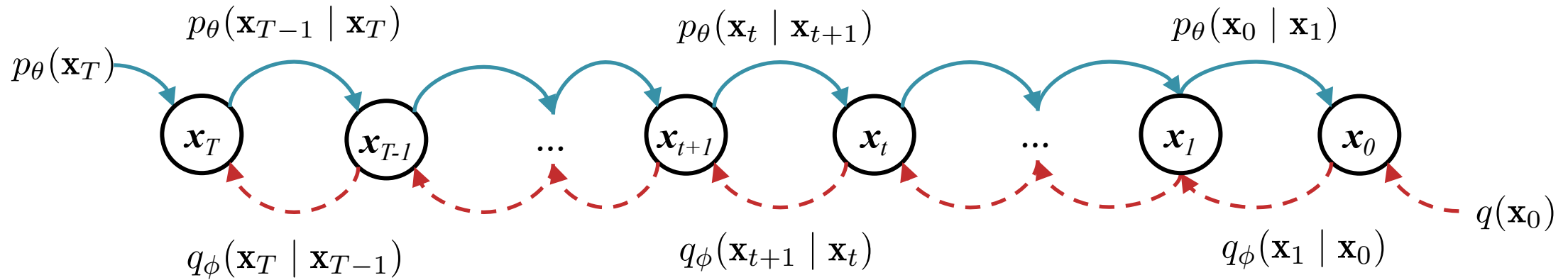
$$p_Z(z) = \int_x p_{Z|X}(z|x)p_X(x) dx \text{ is a Gaussian density}$$

4. Gaussian with a Gaussian mean passed through a *nonlinear* function $g(\cdot)$ does not have a Gaussian marginal,

$$p_Z(z) = \int_x p_{Z|X}(z|x)p_X(x) dx \text{ is **not** a Gaussian density in general}$$



Diffusion Model



Forward Process:

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

(Learned) Reverse Process:

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

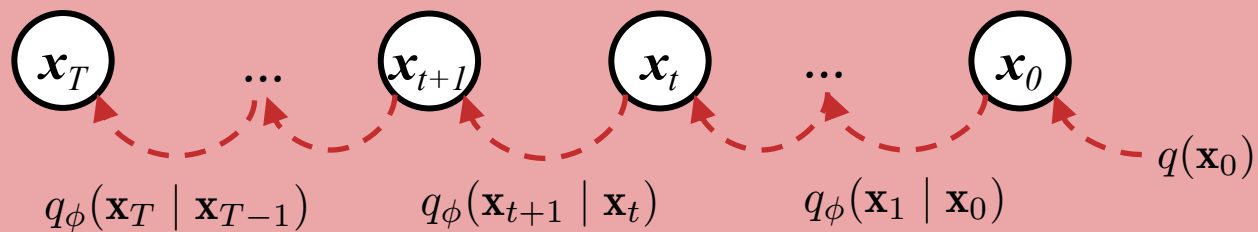
Q: If q_ϕ is just adding noise, how can p_θ be interesting at all?

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:

Asymmetry of Forward/Reverse Processes



Forward Process:

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

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

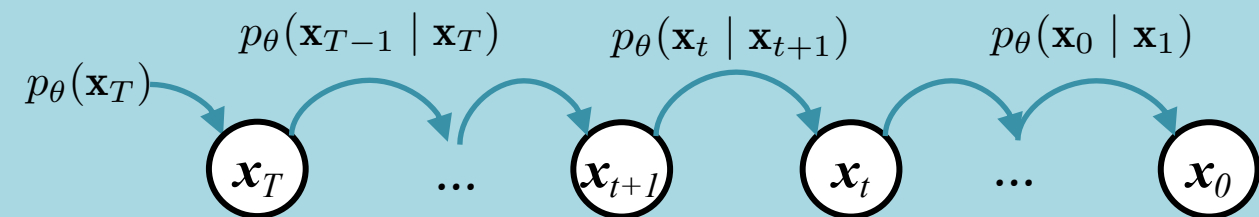
In the forward process, if we marginalize out the intermediate variables of the Markov chain, we get back a Gaussian:

$$\begin{aligned} q(\mathbf{x}_t | \mathbf{x}_0) &= \int \left[\prod_{s=1}^t q(\mathbf{x}_s | \mathbf{x}_{s-1}) \right] d\mathbf{x}_{1:t-1} \\ &= \mathcal{N}(\mathbf{x}_t | \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}) \end{aligned}$$

(Learned) Reverse Process:

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

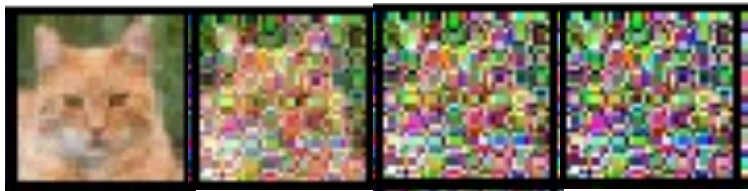
$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) \sim \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$



In the reverse process, by contrast, if we marginalize out the intermediate variables of the Markov chain, we do NOT get a Gaussian.

$$\begin{aligned} p_\theta(\mathbf{x}_0) &= \int \left[p_\theta(\mathbf{x}_T) \prod_{s=1}^T p_\theta(\mathbf{x}_{s-1} | \mathbf{x}_s) \right] d\mathbf{x}_{1:T} \\ &\neq \text{a Gaussian density} \end{aligned}$$

Diffusion Model Analogy



Properties of forward and *exact* reverse processes

Property #1:

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

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

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

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

} — this is the same reparameterization trick from VAEs

Gaussians (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.

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

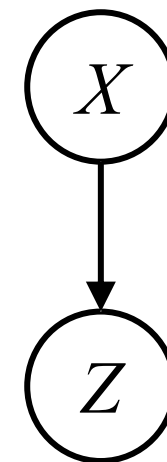
Let $Z | X \sim \mathcal{N}(\mu_z = g(X), \sigma_z^2)$

3. Gaussian with a Gaussian mean passed through a *linear/affine* function $g(\cdot)$ has a Gaussian marginal,

$$p_Z(z) = \int_x p_{Z|X}(z|x)p_X(x) dx \text{ is a Gaussian density}$$

4. Gaussian with a Gaussian mean passed through a *nonlinear* function $g(\cdot)$ does not have a Gaussian marginal,

$$p_Z(z) = \int_x p_{Z|X}(z|x)p_X(x) dx \text{ is **not** a Gaussian density in general}$$



Gaussians (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.

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

Let $Z | X \sim \mathcal{N}(\mu_z = g(X), \sigma_z^2)$

3. Gaussian with a Gaussian mean passed through a *linear/affine* function $g(\cdot)$ has a Gaussian marginal,

$$p_Z(z) = \int_x p_{Z|X}(z|x)p_X(x) dx \text{ is a Gaussian density}$$

and a Gaussian posterior.

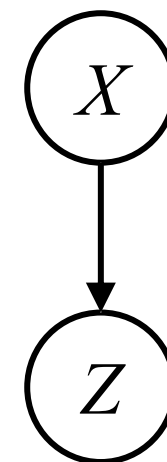
$$p_{X|Z}(x | z) = p_{Z|X}(z|x)p_X(x)/p_Z(z) \text{ is a Gaussian density}$$

4. Gaussian with a Gaussian mean passed through a *nonlinear* function $g(\cdot)$ does not have a Gaussian marginal,

$$p_Z(z) = \int_x p_{Z|X}(z|x)p_X(x) dx \text{ is **not** a Gaussian density in general}$$

nor a Gaussian posterior.

$$p_{X|Z}(x | z) = p_{Z|X}(z|x)p_X(x)/p_Z(z) \text{ is **not** a Gaussian density in general}$$



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

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

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:

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

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

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

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

Parameterizing the *learned* reverse process

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

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

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

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 :

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

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

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

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

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

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

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

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

Parameterizing the *learned* reverse process

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

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 ϵ that gave rise to $\mathbf{x}_t$ from $\mathbf{x}_0$ in the forward process from $\mathbf{x}_t$ and t :

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

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

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

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 :

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

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

Option C is the best empirically

Algorithm 1 Training (Option C)

```
1: initialize  $\theta$ 
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 :

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

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

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