

CS 4644-DL / 7643-A

ZSOLT KIRA

Generative Models:

Denoising Diffusion Probabilistic Models (DDPMs)

Slides adapted from those by Danfei Xu

- **Assignment 4 – Generative models**
 - Due **April 5th 11:59pm EST**
- **Quiz** next class! (18th)
- **Projects**
 - Project proposal due **March 20th**
- **Meta office hours**
 - Wed. 3pm ET on Embeddings

Taxonomy of Generative Models

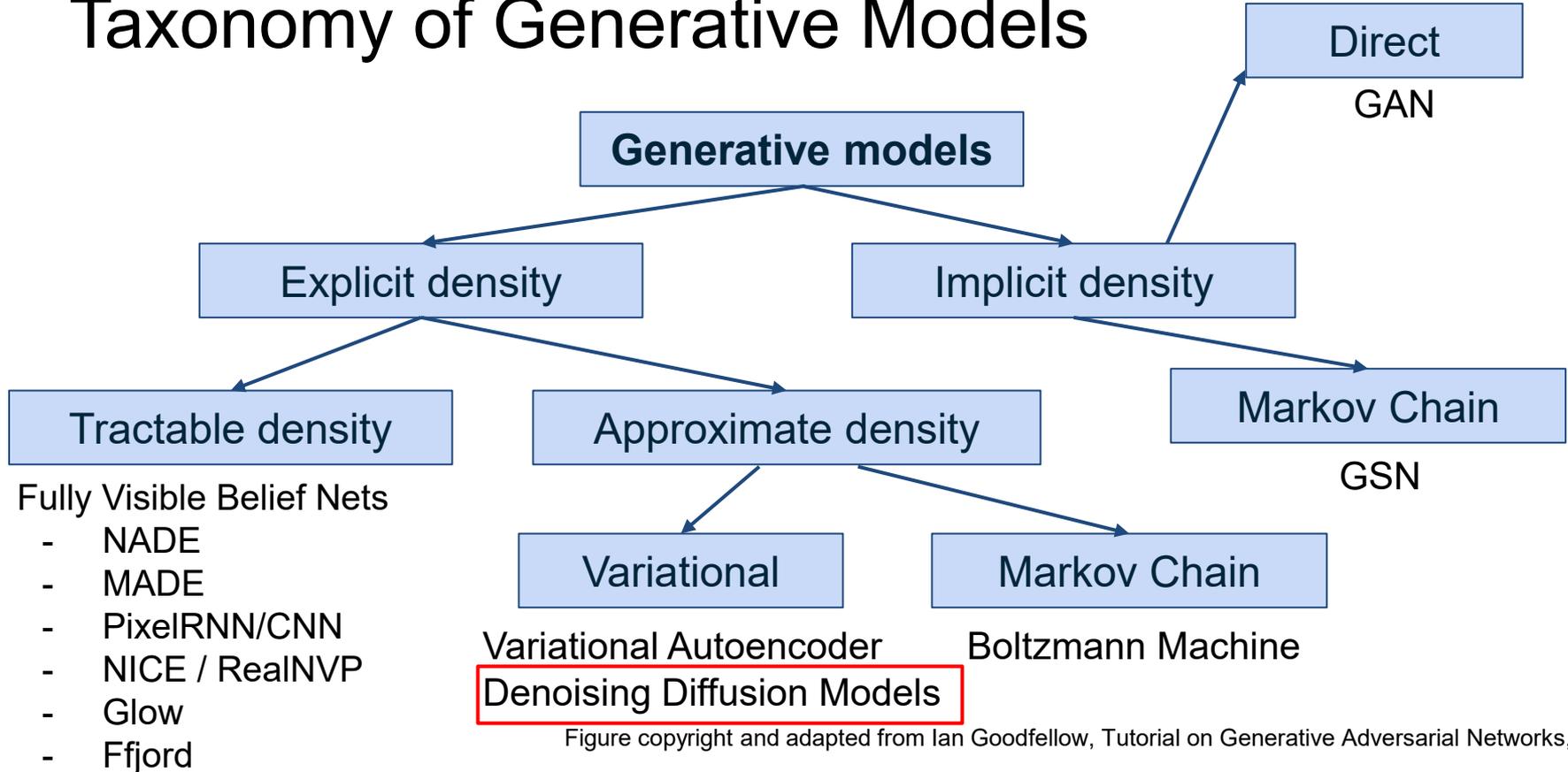


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Putting it all together: maximizing the likelihood lower bound

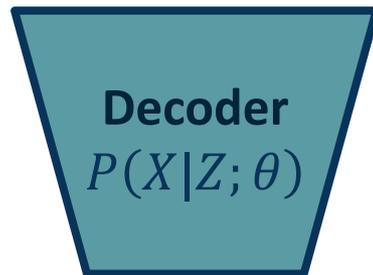
$$\underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Maximize likelihood of original input being reconstructed

\hat{X}

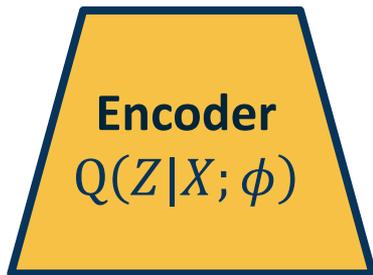
Sample from $P(X|Z; \theta) \sim N(\mu_x, \sigma_x)$

μ_x σ_x

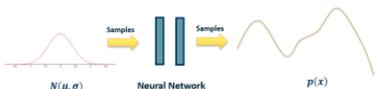


Z

μ_z σ_z



X



1. Output dist. Params
2. Sample from simpler distn -> complex distn

3. Max Log Likelihood -> Bayes -> KL -> ELBO
4. Reparameterization trick
5. Result is simple! KL or MSE if fixed covariance

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Forward and Backward Passes

Denoising Diffusion Probabilistic Models (DDPMs)

And Conditional Diffusion Models

TEXT DESCRIPTION

An astronaut riding a horse

Teddy bears
A bowl of soup
lounging in a tropical resort
in space
playing basketball with cats in space

in a photorealistic style
in the style of Andy Warhol
as a pencil drawing



DALL-E 2



<https://openai.com/dall-e-2/>

Landscape Highlights of Diffusion Models (Nov 2022)

basic principles

- *Diffusion probabilistic models* ([Sohl-Dickstein et al., 2015](#))
- *Noise-conditioned score network (NCSN)*; [Yang & Ermon, 2019](#))
- *Denoising diffusion probabilistic models (DDPM)*; [Ho et al. 2020](#))

conditional & high-res image generation

- *Classifier-guided conditional generation* ([Dhariwal and Nichole, 2021](#))
- *Classifier-free Diffusion Guidance* ([Ho and Salimans, 2022](#))
- *Latent-space Diffusion (StableDiffusion)*; [Rombach and Blattmann et al., 2022](#))

new applications

- *Planning with Diffusion for Flexible Behavior Synthesis (Diffuser)*; [Janner et al., 2022](#))
- *DreamFusion: Text-to-3D using 2D Diffusion* ([Poole and Jain et al., 2022](#))
- *Make-A-Video: Text-to-Video Generation without Text-Video Data* ([Singer et al., 2022](#))

How to make a new generative model

- **Setting:** Given unlabeled dataset of data, I want to learn to sample from $P(x)$
- Define the generative process
- Parameterize it
- Maximum likelihood (often \rightarrow KL-divergence)
- Approximations
- Optimize parameters!
- Add conditioning, e.g. text

Landscape Highlights of Diffusion Models (Nov 2022)

basic principles

- *Diffusion probabilistic models* ([Sohl-Dickstein et al., 2015](#))
- *Noise-conditioned score network (NCSN)*; [Yang & Ermon, 2019](#))
- *Denoising diffusion probabilistic models (DDPM)*; [Ho et al. 2020](#))

conditional & high-res image generation

- *Classifier-guided conditional generation* ([Dhariwal and Nichole, 2021](#))
- *Classifier-free Diffusion Guidance* ([Ho and Salimans, 2022](#))
- *Latent-space Diffusion (StableDiffusion)*; [Rombach and Blattmann et al., 2022](#))

new applications

- *Planning with Diffusion for Flexible Behavior Synthesis (Diffuser)*; [Janner et al., 2022](#))
- *DreamFusion: Text-to-3D using 2D Diffusion* ([Poole and Jain et al., 2022](#))
- *Make-A-Video: Text-to-Video Generation without Text-Video Data* ([Singer et al., 2022](#))

The Denoising Diffusion Process

image from
dataset

x_0



The Denoising Diffusion Process

image from
dataset

The “forward diffusion” process:
add Gaussian noise each step

x_0 → x_1 →



...

The Denoising Diffusion Process

image from
dataset

The “forward diffusion” process:
add Gaussian noise each step

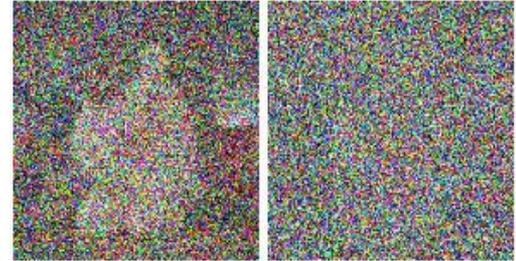
noise $\mathcal{N}(0, I)$

$x_0 \longrightarrow x_1 \longrightarrow$



...

$\longrightarrow x_{T-1} \longrightarrow x_T$

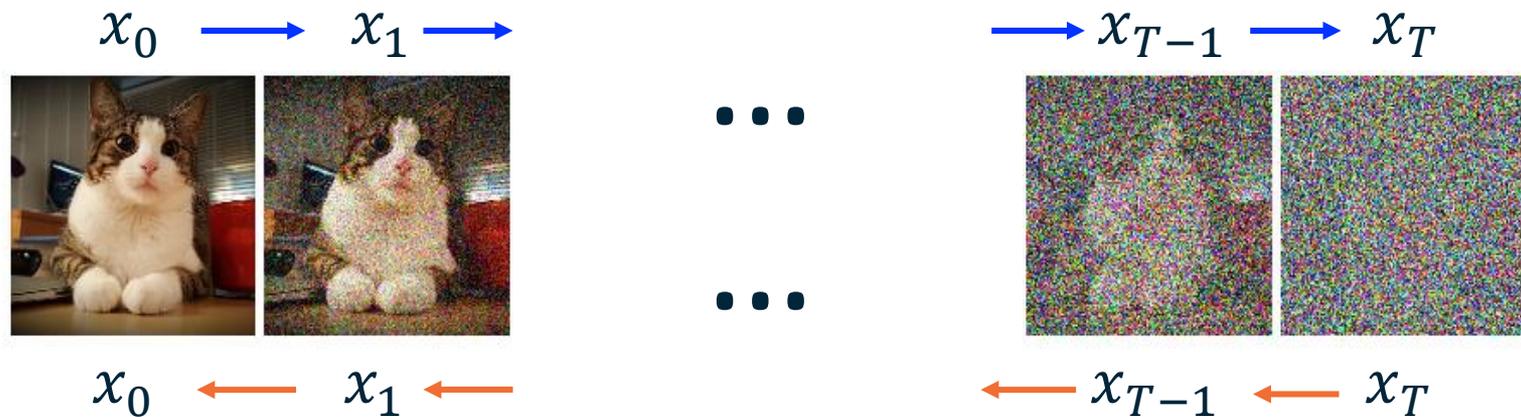


The Denoising Diffusion Process

image from
dataset

The “forward diffusion” process:
add Gaussian noise each step

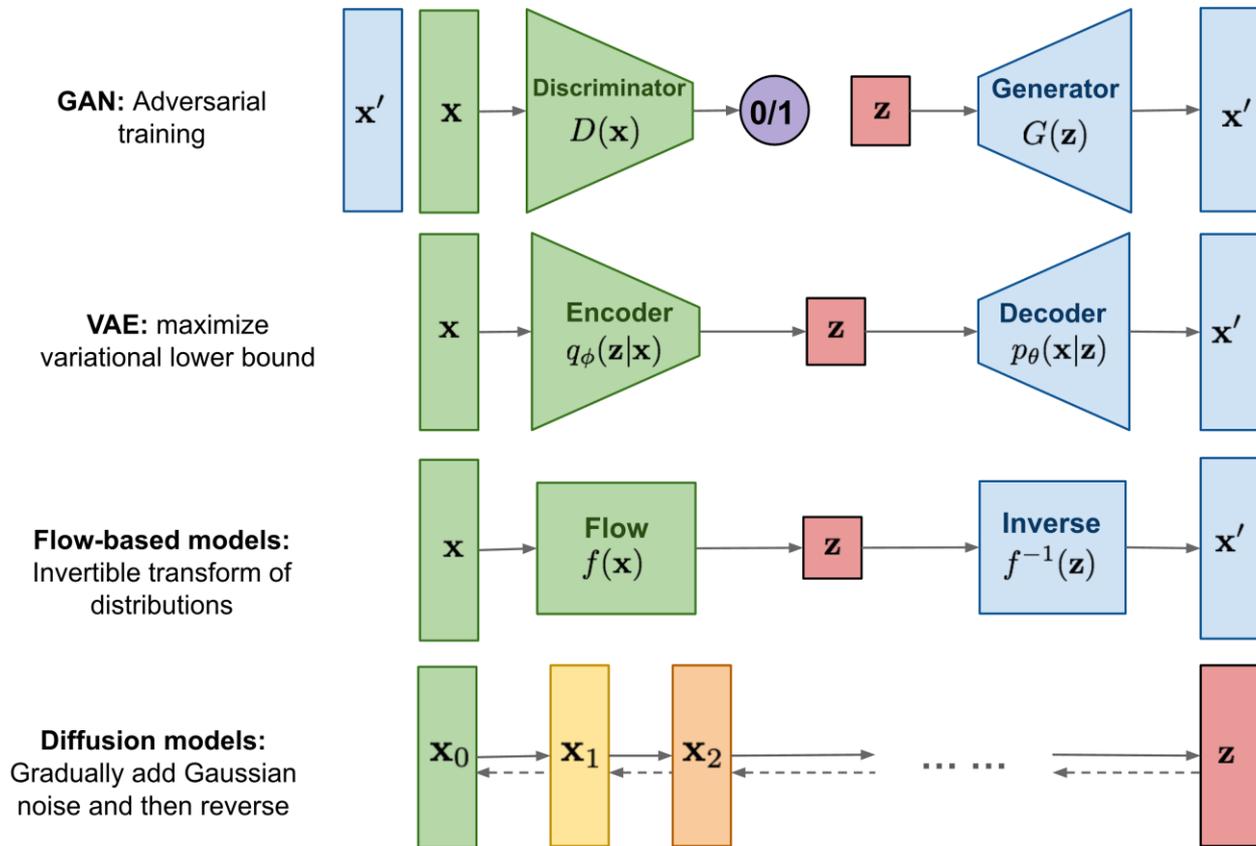
noise $\mathcal{N}(0, I)$



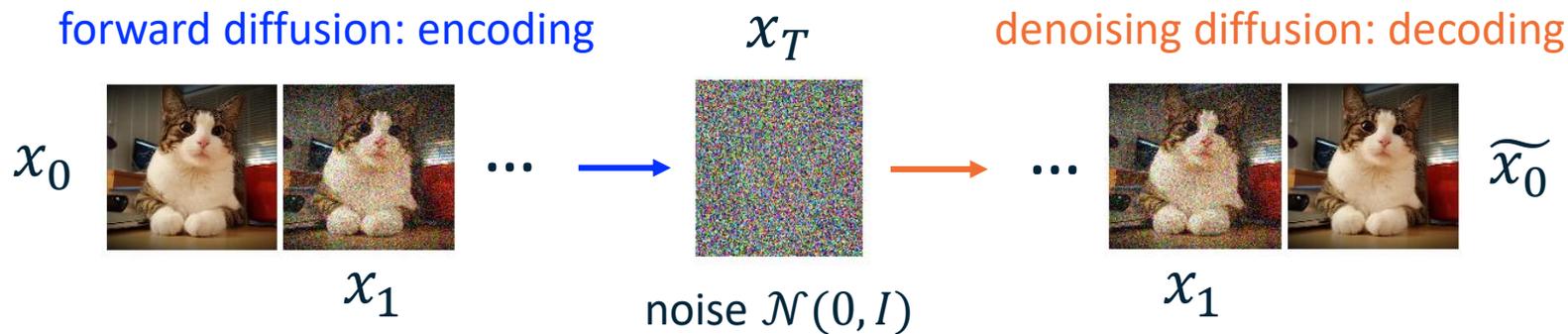
The “denoising diffusion” process:
generate an image from noise by
denoising the gaussian noises

Ties/inspiration from Annealed
Importance Sampling in physics

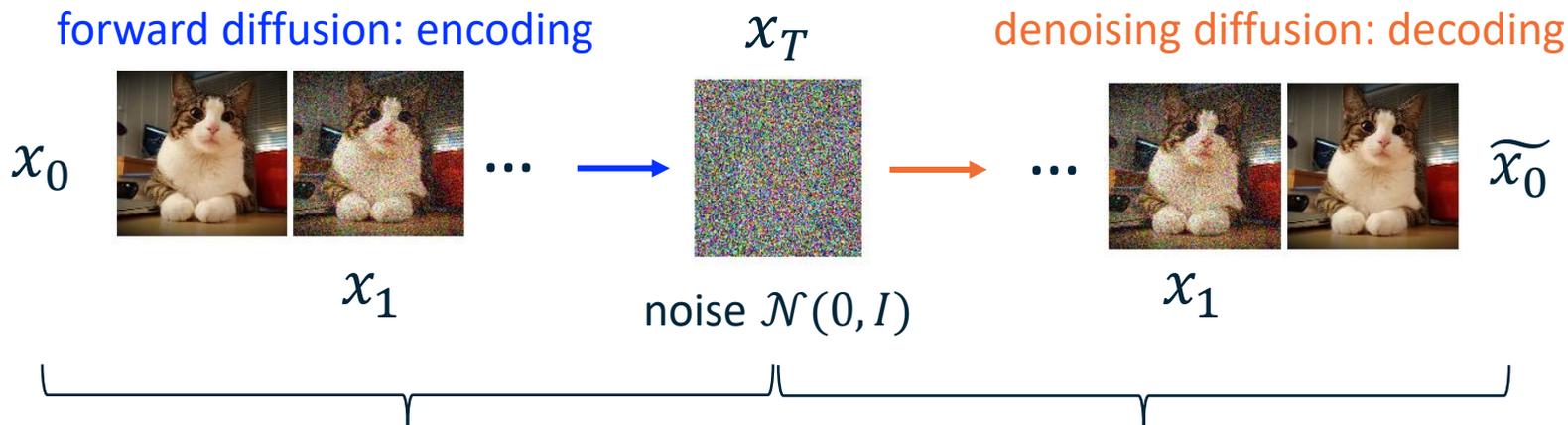
Comparison



Forward/Reverse Processes



Forward/Reverse Processes

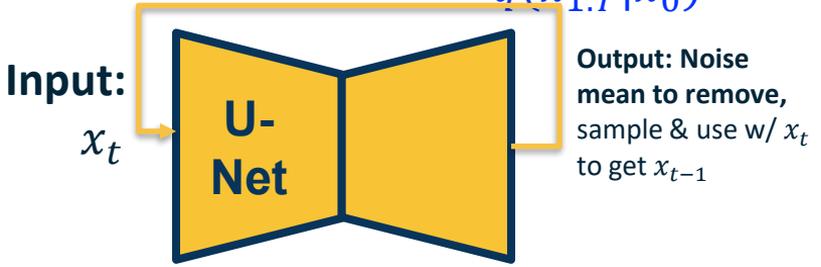


Known / predefined:

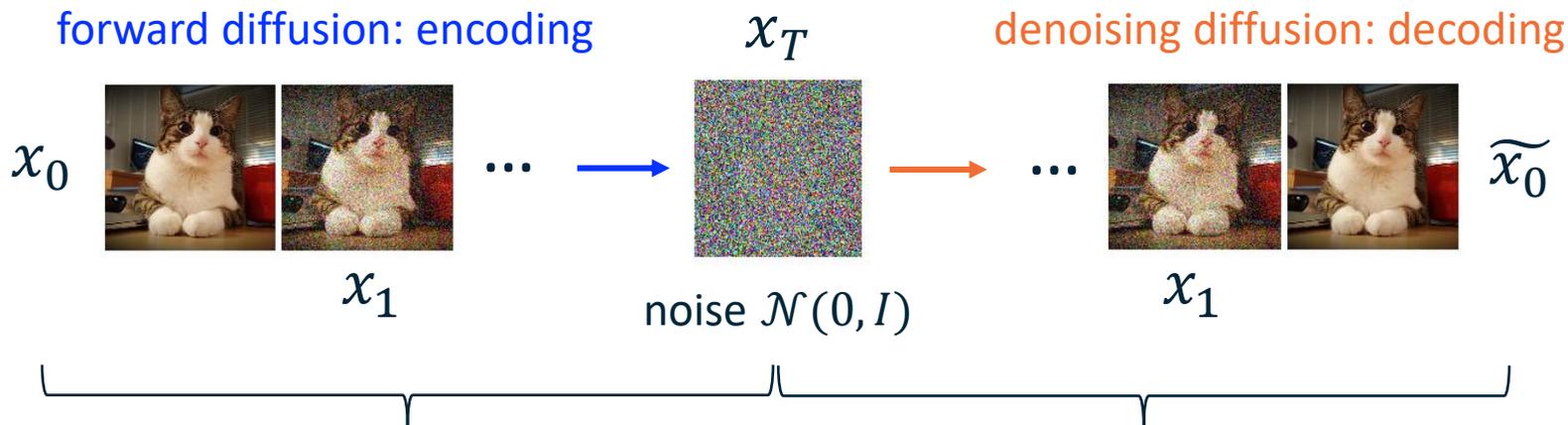
$$q(x_{1:T} | x_0)$$

Unknown / learned:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1} | x_t)$$



Forward/Reverse Processes

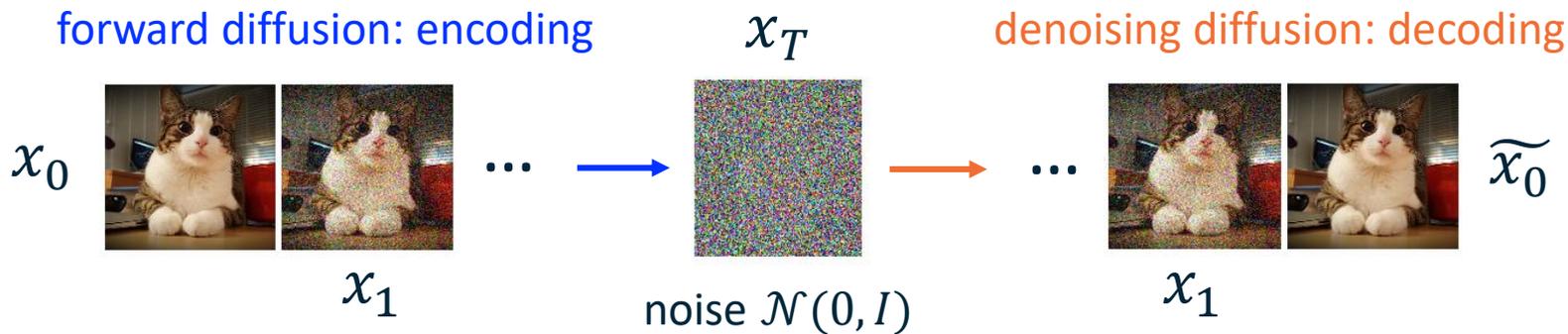


$$q(x_{1:T}|x_0)$$

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t)$$

Use the denoising decoding process to generate new images.

Forward/Reverse Processes



Known / predefined:
 $q(x_{1:T}|x_0)$

Unknown / learned:
 $p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t)$

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \dots \longrightarrow x_T$

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \dots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1}) \quad \text{Probability Chain Rule (Markov Chain)}$$

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \dots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1}) \quad \text{Probability Chain Rule (Markov Chain)}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1 - \beta_t})x_{t-1}, \beta_t I) \quad \text{Conditional Gaussian}$$

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \dots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1}) \quad \text{Probability Chain Rule (Markov Chain)}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1 - \beta_t})x_{t-1}, \beta_t I) \quad \text{Conditional Gaussian}$$

Notation: A Gaussian distribution “for” x_t

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \dots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1}) \quad \text{Probability Chain Rule (Markov Chain)}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1 - \beta_t})x_{t-1}, \beta_t I) \quad \text{Conditional Gaussian}$$

β_t is the *variance schedule* at the diffusion step t

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \dots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1}) \quad \text{Probability Chain Rule (Markov Chain)}$$

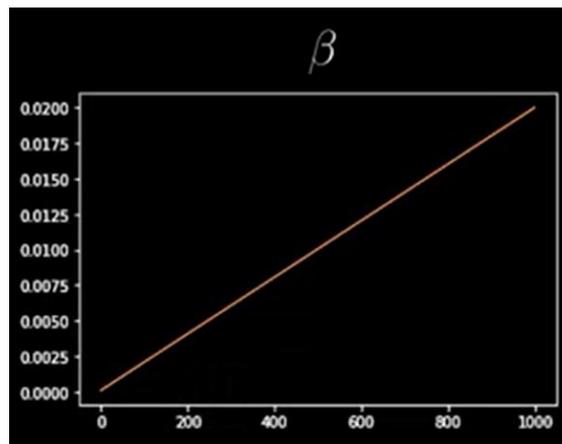
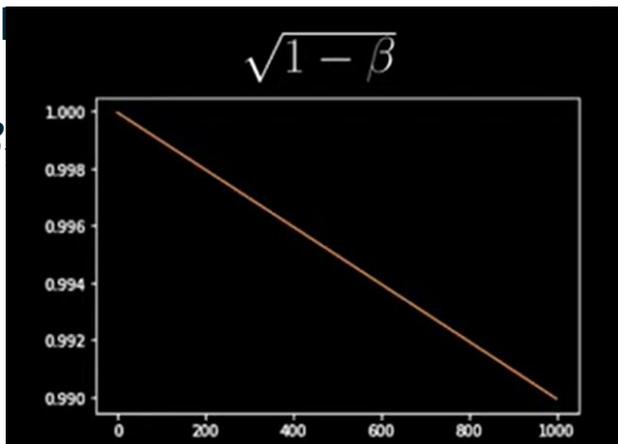
$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1 - \beta_t})x_{t-1}, \beta_t I) \quad \text{Conditional Gaussian}$$

β_t is the

diffusion

$0 < \beta$

value



$= 1000$

The Diffusion (Encoding) Process

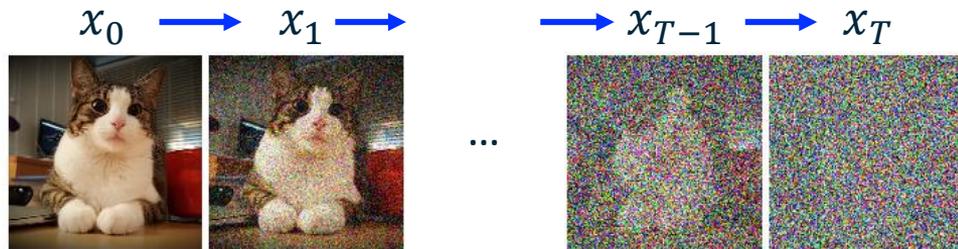
The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \dots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1}) \quad \text{Probability Chain Rule (Markov Chain)}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1 - \beta_t})x_{t-1}, \beta_t I) \quad \text{Conditional Gaussian}$$

β_t is the *variance schedule* at the diffusion step t

$0 < \beta_1 < \beta_2 < \dots < \beta_T < 1$, typical value range $[0.0001, 0.02]$, with $T = 1000$



The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \dots \longrightarrow x_T$

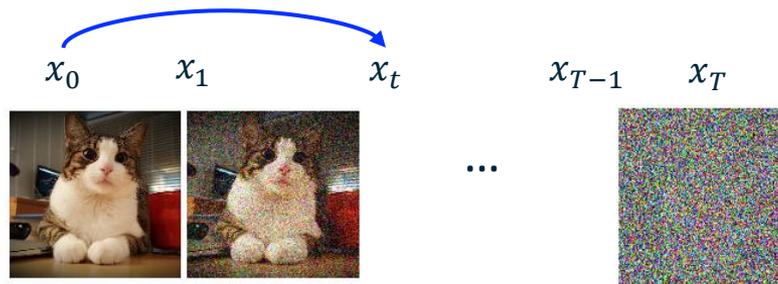
$$q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1}) \quad \text{Probability Chain Rule (Markov Chain)}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1 - \beta_t})x_{t-1}, \beta_t I) \quad \text{Conditional Gaussian}$$

Nice property: samples from an *arbitrary forward step* are also Gaussian-distributed!

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I)$$

, where $\alpha_t = (1 - \beta_t)$, $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$



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

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t, \sqrt{1 - \beta_t}x_{t-1}, \beta_t I)$$

$$= \sqrt{1 - \beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon$$

$$= \sqrt{\alpha_t}x_{t-1} + \sqrt{1 - \alpha_t}\epsilon$$

$$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1 - \alpha_t\alpha_{t-1}}\epsilon$$

$$= \sqrt{\alpha_t\alpha_{t-1}\alpha_{t-2}}x_{t-3} + \sqrt{1 - \alpha_t\alpha_{t-1}\alpha_{t-2}}\epsilon$$

$$= \sqrt{\alpha_t\alpha_{t-1}\dots\alpha_1}x_0 + \sqrt{1 - \alpha_t\alpha_{t-1}\dots\alpha_1}\epsilon$$

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I) \leftarrow \boxed{= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon}$$

The Diffusion (Encoding)

The **known** forward process x_0 —

$$q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1}) \quad \text{Probab}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1 - \beta_t})x_{t-1}, \beta_t I)$$

Nice property: samples from an *arbitrary forward step* are also Gaussian-distributed!

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I)$$

Gaussian reparameterization trick:

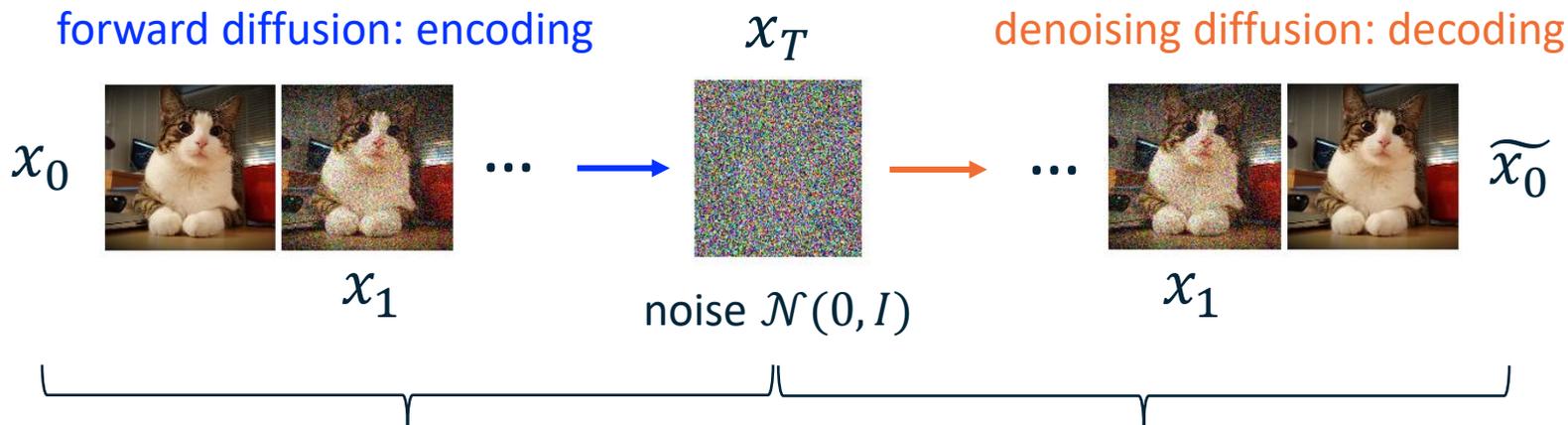
$$z = \mu + \epsilon * \sigma, \epsilon \sim N(0,1)$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

Intuition: We know all distributions in forward process, and can in fact directly compute for any t based on x_0

(square root appears because reparameterization trick has just σ)

The Diffusion and Denoising Process



Known / predefined:
 $q(x_{1:T} | x_0)$

Unknown / learned:

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1} | x_t)$$

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \dots \leftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t) \quad \text{Probability Chain Rule (Markov Chain)}$$

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \dots \leftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t) \quad \text{Probability Chain Rule (Markov Chain)}$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_{\theta}(x_t, t)) \quad \text{Conditional Gaussian}$$

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \dots \leftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t) \quad \text{Probability Chain Rule (Markov Chain)}$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_q(t)) \quad \text{Conditional Gaussian}$$

Want to learn time-
dependent mean

Assume fixed / known variance
(simplification)

**What is the shape of
the mean?**

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \dots \leftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t) \quad \text{Probability Chain Rule (Markov Chain)}$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_q(t)) \quad \text{Conditional Gaussian}$$

Want to learn time-
dependent mean

Assume fixed / known variance
(simplification)

How do we form a learning objective?

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_q(t))$$

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_q(t))$$

High-level intuition: derive a *ground truth denoising distribution* $q(x_{t-1}|x_t, x_0)$ and train a neural net $p_{\theta}(x_{t-1}|x_t)$ to match the distribution.

$$p(x) = \int p(x|z)p(z)dz \quad \text{Intractable to estimate!}$$

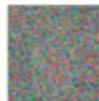
$$\begin{aligned} \log p(x) &= \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + D_{KL}(q(z|x) || p(z|x)) \\ &\geq \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] \quad \text{Evidence Lower Bound (ELBO)} \end{aligned}$$

$$\log p(x_0) \geq \mathbb{E}_q \left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)} \right] \quad x = x_0, z = x_{1:T}$$

... (derivation omitted, see [Sohl-Dickstein et al., 2015 Appendix B](#))

$$= -\mathbb{E}_q [D_{KL}(q(x_T|x_0) || p(x_T))] - \sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

fixed



Easy to optimize / sometimes omitted

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_q(t))$$

High-level intuition: derive a *ground truth denoising distribution* $q(x_{t-1}|x_t, x_0)$ and train a neural net $p_{\theta}(x_{t-1}|x_t)$ to match the distribution.

The learning objective: $\operatorname{argmin}_{\theta} D_{KL}(q(x_{t-1}|x_t, x_0) || p_{\theta}(x_{t-1}|x_t))$

What does it look like? $q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \mu_q(t), \Sigma_q(t))$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{(1 - \bar{\alpha}_t)}} \epsilon \right), \quad \epsilon \sim \mathcal{N}(0, I)$$

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

High-level intuition: derive a *ground truth denoising distribution* $q(x_{t-1}|x_t, x_0)$ and train a neural net $p_\theta(x_{t-1}|x_t)$ to match the distribution.

The learning objective: $\operatorname{argmin}_\theta D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t))$

What does it look like? $q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \mu_q(t), \Sigma_q(t))$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{(1 - \bar{\alpha}_t)}} \epsilon \right), \quad \epsilon \sim \mathcal{N}(0, I) \longleftarrow \text{Recall: Gaussian reparameterization trick}$$

The “ground truth” noise that brought x_{t-1} to x_t

$$\begin{aligned} q(x_{t-1}|x_t) &= \frac{q(x_{t-1}|x_t, x_0)}{q(x_t|x_{t-1}, x_0)} \quad (\text{markov assumption}) \\ &= \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)} \quad (\text{Bayes rule}) \\ &= \frac{\mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, \beta_t I) \mathcal{N}(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}x_0, (1 - \bar{\alpha}_{t-1})I)}{\mathcal{N}(x_t; \sqrt{\alpha_t}x_0, (1 - \alpha_t)I)} \\ &\propto \mathcal{N}\left(x_{t-1}; \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)x_0}{1 - \sqrt{\alpha_t}}, \Sigma_q(t)\right) \quad (\text{Property of Gaussian}) \end{aligned}$$

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_q(t))$$

High-level intuition: derive a *ground truth denoising distribution* $q(x_{t-1}|x_t, x_0)$ and train a neural net $p_{\theta}(x_{t-1}|x_t)$ to match the distribution.

The learning objective: $\operatorname{argmin}_{\theta} D_{KL}(q(x_{t-1}|x_t, x_0) || p_{\theta}(x_{t-1}|x_t))$

What does it look like? $q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \mu_q(t), \Sigma_q(t))$

Assuming identical variance $\Sigma_q(t)$, we have:

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{(1 - \bar{\alpha}_t)}} \epsilon \right)$$

$$\operatorname{argmin}_{\theta} D_{KL}(q(x_{t-1}|x_t, x_0) || p_{\theta}(x_{t-1}|x_t)) = \operatorname{argmin}_{\theta} w || \mu_q(t) - \mu_{\theta}(x_t, t) ||^2$$

Should be variance-dependent, but constant works better in practice

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_q(t))$$

High-level intuition: derive a *ground truth denoising distribution* $q(x_{t-1}|x_t, x_0)$ and train a neural net $p_{\theta}(x_{t-1}|x_t)$ to match the distribution.

The learning objective: $\operatorname{argmin}_{\theta} D_{KL}(q(x_{t-1}|x_t, x_0) || p_{\theta}(x_{t-1}|x_t))$

What does it look like? $q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \mu_q(t), \Sigma_q(t))$

Assuming identical variance $\Sigma_q(t)$, we have:

$$\operatorname{argmin}_{\theta} D_{KL}(q(x_{t-1}|x_t, x_0) || p_{\theta}(x_{t-1}|x_t)) = \operatorname{argmin}_{\theta} w ||\mu_q(t) - \mu_{\theta}(x_t, t)||^2$$

Simplified learning objective: $\operatorname{argmin}_{\theta} ||\epsilon - \epsilon_{\theta}(x_t, t)||^2$

Predict the one-step noise that was added (and remove it!)

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \dots \leftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t) \quad \text{Probability Chain Rule (Markov Chain)}$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_q(t)) \quad \text{Conditional Gaussian}$$

Assume fixed / known variance

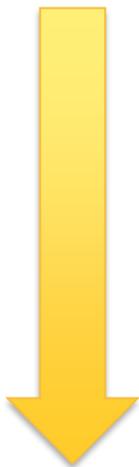
How did we arrive at the learning objective?

See slides at the end! Variational models ...

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_q(t))$$



Simplified learning objective: $\operatorname{argmin}_{\theta} \|\epsilon - \epsilon_{\theta}(x_t, t)\|^2$

Predict the one-step noise that was added (and remove it!)

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \dots \leftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t) \quad \text{Probability Chain Rule (Markov Chain)}$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_q(t)) \quad \text{Conditional Gaussian}$$

We know how to learn

Assume fixed / known variance

$$\text{Inference time: } \mu_{\theta}(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{(1-\alpha_t)}} \epsilon_{\theta}(x_t, t) \right)$$



The Denoising (Decoding) Process

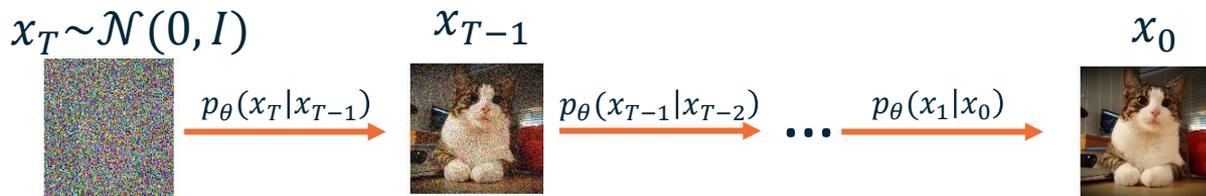
The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \dots \leftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t) \quad \text{Probability Chain Rule (Markov Chain)}$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_q(t)) \quad \text{Conditional Gaussian}$$

We know how to learn

Assume fixed / known variance



Generate new images!

The Denoising Diffusion Algorithm

Algorithm 1 Training

- 1: **repeat**
 - 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
 - 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
 - 4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 5: Take gradient descent step on
$$\nabla_{\theta} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t) \right\|^2$$
 - 6: **until** converged
-

The Denoising Diffusion Algorithm

Algorithm 1 Training

- 1: **repeat**
 - 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
 - 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
 - 4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 5: Take gradient descent step on
$$\nabla_{\theta} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t) \right\|^2$$
 - 6: **until** converged
-

Algorithm 2 Sampling

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

The Denoising Diffusion Algorithm

Algorithm 1 Training

- 1: **repeat**
 - 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
 - 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
 - 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 5: Take gradient descent step on
 $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\|^2$
 - 6: **until** converged
-

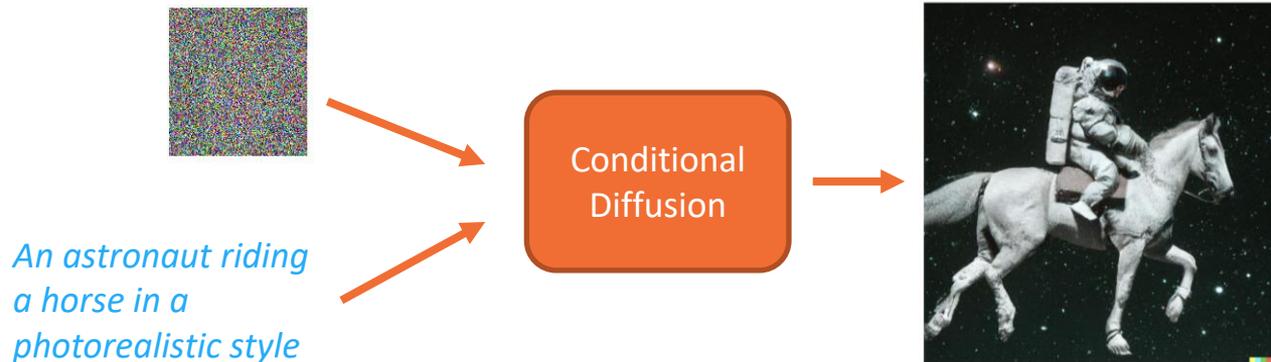

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, I)$$

Algorithm 2 Sampling

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


$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma(t))$$

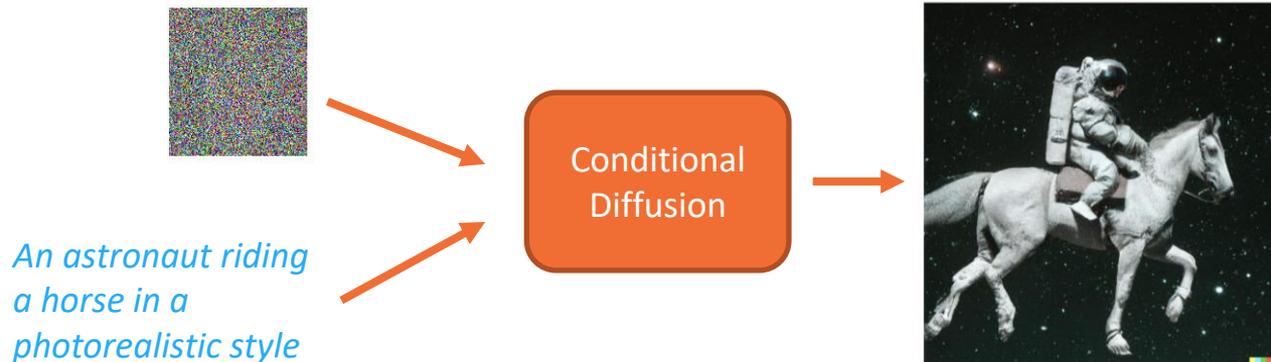
Conditional Diffusion Models



Simple idea: just condition the model on some text labels y !

$$\epsilon_{\theta}(x_t, y, t)$$

Conditional Diffusion Models

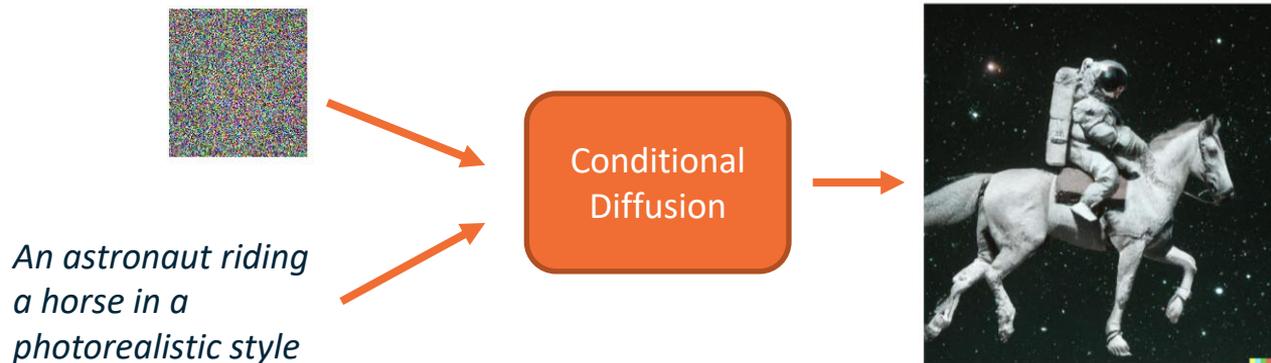


Simple idea: just condition the model on some text labels y !

$$\epsilon_{\theta}(x_t, y, t)$$

Problem: Very blurry generation

Classifier-guided Diffusion



Better idea: use the *gradients* from a image captioning model $f_\varphi(y|x_t)$ to guide the diffusion process!

$$\bar{\epsilon}_\theta(x_t, t) = \epsilon_\theta(x_t, t) - \sqrt{1 - \bar{\alpha}_t} \nabla_{x_t} \log f_\varphi(y|x_t)$$

Classifier guidance

Using the gradient of a trained classifier as guidance

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s .

Input: class label y , gradient scale s Score model Classifier gradient

$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$

for all t from T to 1 **do**

$\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$

$x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$

end for

return x_0

- Train unconditional Diffusion model
- Take your favorite classifier, depending on the conditioning type
- During inference / sampling mix the gradients of the classifier with the predicted score function of the unconditional diffusion model.

Classifier guidance

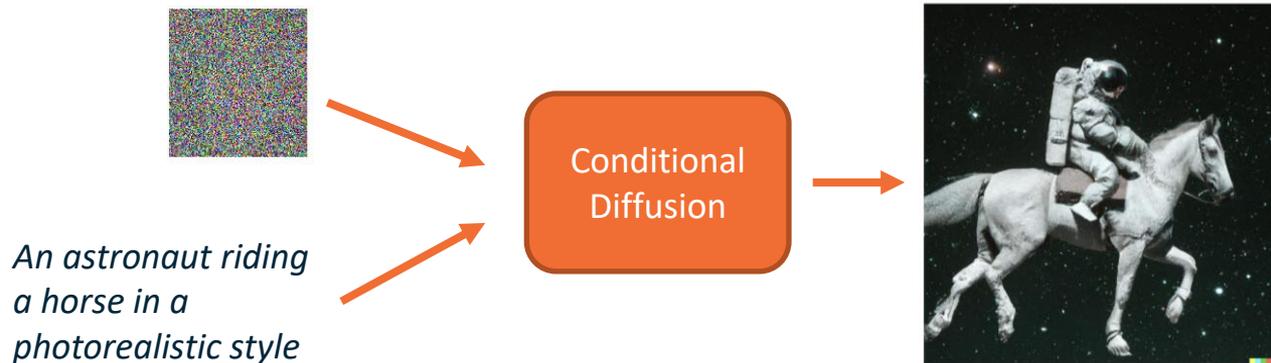
Using the gradient of a trained classifier as guidance

$$\nabla_x \log p_\gamma(x | y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y | x).$$



Samples from an unconditional diffusion model with classifier guidance, for guidance scales 1.0 (left) and 10.0 (right), taken from Dhariwal & Nichol (2021).

Classifier-free Guided Diffusion



Classifier-free Guided Diffusion: estimate the gradient of the classifier model with conditional diffusion models!

$$\nabla_{x_t} \log f_{\varphi}(y|x_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}} (\epsilon_{\theta}(x_t, t, y) - \epsilon_{\theta}(x_t, t))$$

Classifier-free guidance

Trade-off for sample quality and sample diversity



Non-guidance



Guidance scale = 1

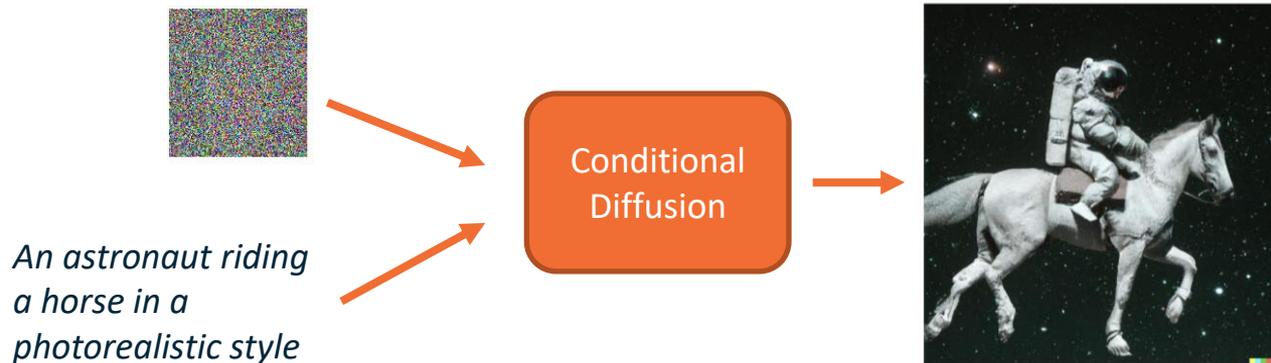


Guidance scale = 3

Large guidance weight (ω) usually leads to better individual sample quality but less sample diversity.

[Ho & Salimans, "Classifier-Free Diffusion Guidance", 2021.](#)

Classifier-free Guided Diffusion



Classifier-free Guided Diffusion: estimate the gradient of the classifier model with conditional diffusion models!

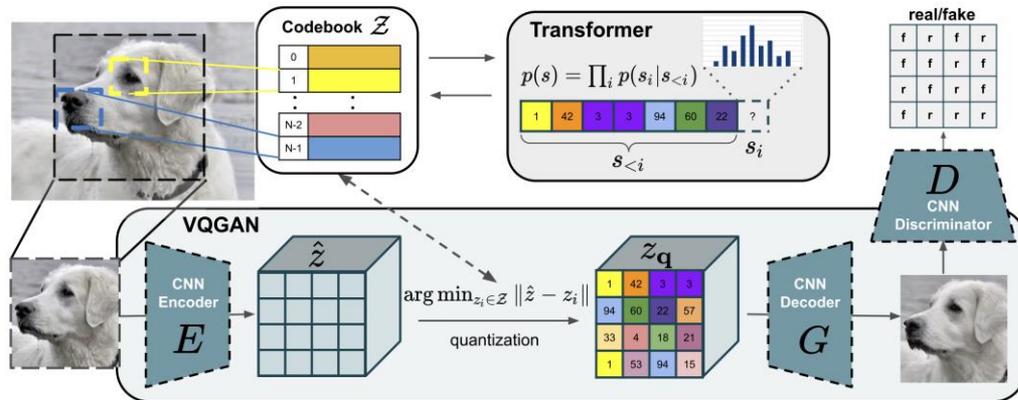
$$\nabla_{x_t} \log f_{\varphi}(y|x_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}} (\epsilon_{\theta}(x_t, t, y) - \epsilon_{\theta}(x_t, t))$$

$$\bar{\epsilon}_{\theta}(x_t, t, y) = (w + 1)\epsilon_{\theta}(x_t, t, y) - w\epsilon_{\theta}(x_t, t)$$

Latent-space Diffusion

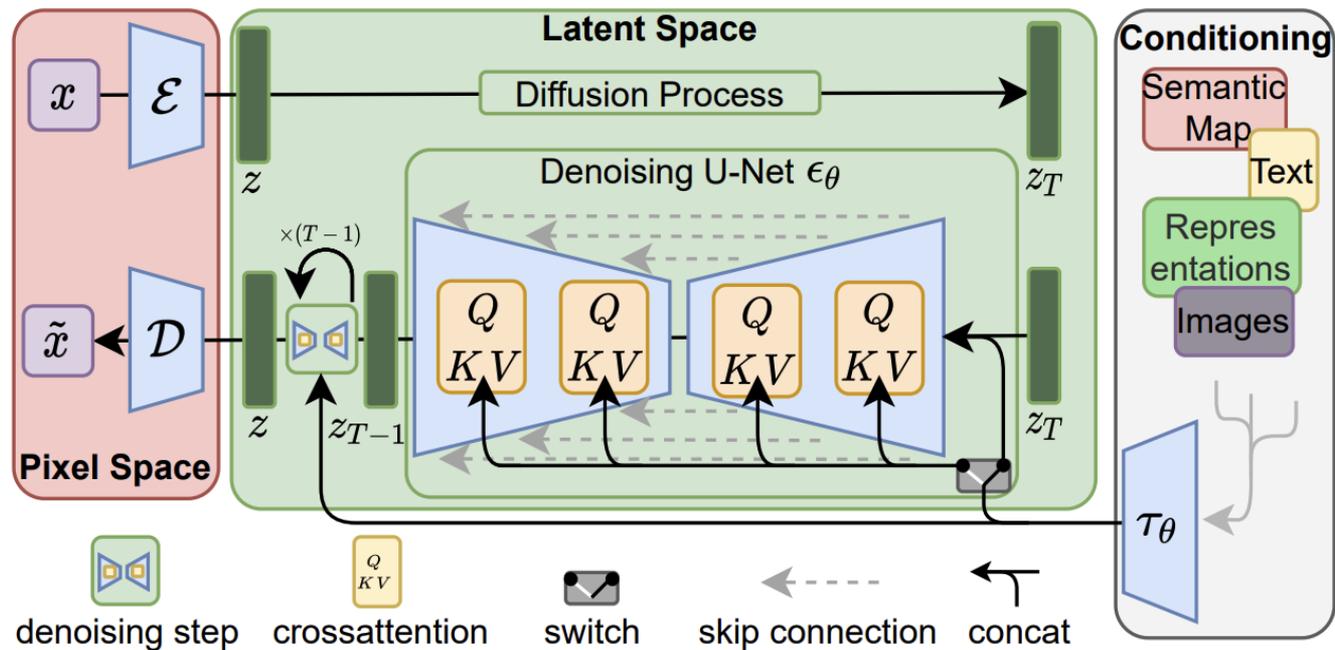
Problem: Hard to learn diffusion process on high-resolution images

Solution: learn a low-dimensional latent space using a transformer-based autoencoder and *do diffusion on the latent space!*



The latent space autoencoder

“StableDiffusion”



“StableDiffusion”



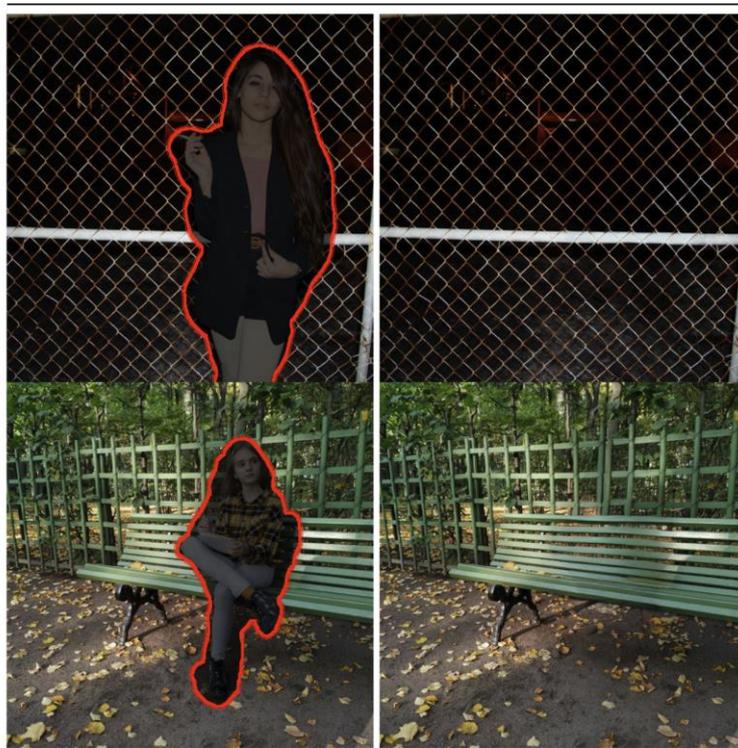
Layout-Conditional Generation

“StableDiffusion”



Segmentation-Conditional Generation

“StableDiffusion”



Inpainting



<https://openai.com/dall-e-2/>

Additional resources / tutorials

- Overview of the research landscape: [What are Diffusion Models?](#)
- More math! [Understanding Diffusion Models: A Unified Perspective](#)
- Tutorial with hands-on example: [The Annotated Diffusion Model](#)
- Nice introduction videos:
 - [What are Diffusion Models?](#)
 - [Diffusion Models | Math Explained](#)
 - Three hours of the math! <https://www.youtube.com/watch?v=rLepfNziDPM>
- CVPR Tutorial: [Denoising Diffusion-based Generative Modeling: Foundations and Applications](#)
- Score functions:
 - [In general](#)
 - For [Diffusion models](#)

Summary

- Denoising Diffusion model is a type of generative model that learns the process of “denoising” a known noise source (Gaussian).
- We can construct a learning problem by deriving the evidence lower bound (ELBO) of the denoising process.
- The learning objective is to minimize the KL divergence between the “ground truth” and the learned denoising distribution.
- A simplified learning objective is to estimate the noise of the forward diffusion process.
- The diffusion process can be guided to generate targeted samples.
- Can be applied to many different domains. Same underlying principle.
- Very hot topic!

Comparing the different generative models

Q. Which ones are VAEs good at?

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations			
Fast sampling			
High quality samples			

Comparing the different generative



VAEs are bad at generating high quality samples

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✓		
Fast sampling	✓		
High quality samples	✗		

Comparing the different generative models

Q. Which ones are GANs good at?

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✓		
Fast sampling	✓		
High quality samples	✗		

Comparing the different generative models

GANs suffer from mode collapse

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✓	✗	
Fast sampling	✓	✓	
High quality samples	✗	✓	

Comparing the different generative models

Q. Which ones are Diffusion models good at?

	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✓	✗	
Fast sampling	✓	✓	
High quality samples	✗	✓	

Comparing the different generative models

Diffusion models are bad at sampling fast.

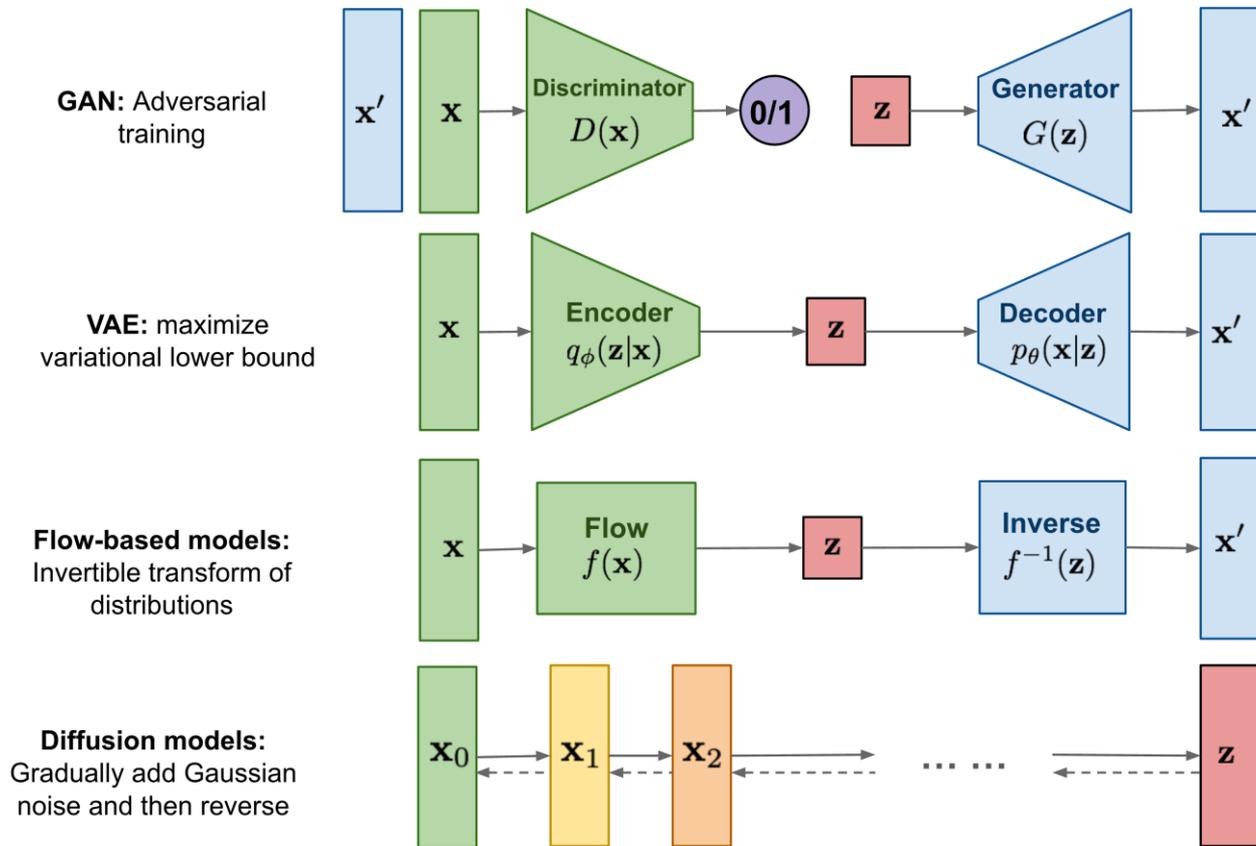
	Autoregressive (VAEs)	GANs	Diffusion
Mode coverage / diversity of generations	✓	✗	✓
Fast sampling	✓	✓	✗
High quality samples	✗	✓	✓

- ◆ Several ways to learn *generative* models via deep learning
- ◆ **Variational Autoencoders:**
 - ◆ Pro: Principled mathematical formulation
 - ◆ Pro: Results in disentangled latent representations
 - ◆ Con: Approximation inference, results in somewhat lower quality reconstructions
- ◆ **Generative Adversarial Networks (GANs):**
 - ◆ Pro: Amazing results across many image modalities
 - ◆ Con: Unstable/difficult training process, computationally heavy for good results
 - ◆ Con: Limited success for discrete distributions (language)
 - ◆ Con: Hard to evaluate (implicit model)
- ◆ **Diffusion Models**
 - ◆ Pro: Great results and diversity!
 - ◆ Con: Slow generation (though lots of tricks to address)

Ha & Schmidhuber, World Models, 2018

Overall Summary

Comparison



(Quick) Derivation!



$p(x) = \int p(x|z)p(z)dz$ Intractable to estimate!

$$p(x) = \int p(x|z)p(z)dz \quad \text{Intractable to estimate!}$$

$$\begin{aligned} \log p(x) &= \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + D_{KL}(q(z|x) || p(z|x)) \\ &\geq \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] \end{aligned} \quad \text{Evidence Lower Bound (ELBO)}$$

Variational
Inference



Simplify to
KL

$$p(x) = \int p(x|z)p(z)dz \quad \text{Intractable to estimate!}$$

$$\begin{aligned} \log p(x) &= \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + D_{KL}(q(z|x) || p(z|x)) \\ &\geq \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] \end{aligned} \quad \text{Evidence Lower Bound (ELBO)}$$

$$\log p(x_0) \geq \mathbb{E}_q \left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)} \right] \quad x = x_0, z = x_{1:T}$$

Variational
Inference



Simplify to
KL

$$p(x) = \int p(x|z)p(z)dz \quad \text{Intractable to estimate!}$$

$$\begin{aligned} \log p(x) &= \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + D_{KL}(q(z|x) || p(z|x)) \\ &\geq \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] \end{aligned}$$

Evidence Lower Bound (ELBO)

$$\log p(x_0) \geq \mathbb{E}_q \left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)} \right] \quad x = x_0, z = x_{1:T}$$

... (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= \mathbb{E}_q \left[\log \frac{p(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t)}{\prod_{t=1}^T q(x_t|x_{t-1})} \right]$$

← reverse denoising
← forward diffusion

Variational
Inference



Simplify to
KL

$$p(x) = \int p(x|z)p(z)dz \quad \text{Intractable to estimate!}$$

$$\begin{aligned} \log p(x) &= E_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + D_{KL}(q(z|x) || p(z|x)) \\ &\geq E_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] \end{aligned} \quad \text{Evidence Lower Bound (ELBO)}$$

$$\log p(x_0) \geq E_q \left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)} \right] \quad x = x_0, z = x_{1:T}$$

... (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

Variational
Inference



Simplify to
KL

$$p(x) = \int p(x|z)p(z)dz \quad \text{Intractable to estimate!}$$

$$\begin{aligned} \log p(x) &= \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + D_{KL}(q(z|x) || p(z|x)) \\ &\geq \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] \end{aligned} \quad \text{Evidence Lower Bound (ELBO)}$$

$$\log p(x_0) \geq \mathbb{E}_q \left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)} \right] \quad x = x_0, z = x_{1:T}$$

... (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -\mathbb{E}_q [D_{KL}(q(x_T|x_0) || p(x_T))] - \sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

Variational Inference



Simplify to KL

$$p(x) = \int p(x|z)p(z)dz \quad \text{Intractable to estimate!}$$

$$\begin{aligned} \log p(x) &= \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + D_{KL}(q(z|x) || p(z|x)) \\ &\geq \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] \end{aligned} \quad \text{Evidence Lower Bound (ELBO)}$$

$$\log p(x_0) \geq \mathbb{E}_q \left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)} \right] \quad x = x_0, z = x_{1:T}$$

... (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -\mathbb{E}_q [D_{KL}(q(x_T|x_0) || p(x_T))] - \sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

fixed



Easy to optimize / sometimes omitted

Variational Inference



Simplify to KL

$$p(x) = \int p(x|z)p(z)dz \quad \text{Intractable to estimate!}$$

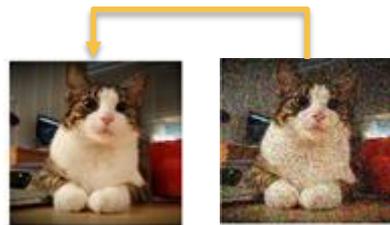
$$\begin{aligned} \log p(x) &= \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + D_{KL}(q(z|x) || p(z|x)) \\ &\geq \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] \end{aligned} \quad \text{Evidence Lower Bound (ELBO)}$$

$$\log p(x_0) \geq \mathbb{E}_q \left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)} \right] \quad x = x_0, z = x_{1:T}$$

... (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -\mathbb{E}_q [D_{KL}(q(x_T|x_0) || p(x_T))] - \sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

Maximize the agreement between the predicted reverse diffusion distribution p_θ and the “ground truth” reverse diffusion distribution q



Variational Inference



Simplify to KL



Reverse Process => Normal

$$p(x) = \int p(x|z)p(z)dz \quad \text{Intractable to estimate!}$$

$$\begin{aligned} \log p(x) &= \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + D_{KL}(q(z|x) || p(z|x)) \\ &\geq \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] \end{aligned} \quad \text{Evidence Lower Bound (ELBO)}$$

$$\log p(x_0) \geq \mathbb{E}_q \left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)} \right] \quad x = x_0, z = x_{1:T}$$

... (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -\mathbb{E}_q [D_{KL}(q(x_T|x_0) || p(x_T))] - \sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

Variational Inference



Simplify to KL



Reverse Process => Normal

$$p(x) = \int p(x|z)p(z)dz \quad \text{Intractable to estimate!}$$

$$\begin{aligned} \log p(x) &= E_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + D_{KL}(q(z|x) || p(z|x)) \\ &\geq E_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] \end{aligned} \quad \text{Evidence Lower Bound (ELBO)}$$

$$\log p(x_0) \geq E_q \left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)} \right] \quad x = x_0, z = x_{1:T}$$

... (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -E_q [D_{KL}(q(x_T|x_0) || p(x_T))] - \sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

$$\begin{aligned} q(x_{t-1}|x_t) &= q(x_{t-1}|x_t, x_0) \quad \text{(markov assumption)} \\ &= \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)} \quad \text{(Bayes rule)} \end{aligned}$$

$$= \frac{\mathcal{N}(x_t; \sqrt{a_t}x_{t-1}, \beta_t I) \mathcal{N}(x_{t-1}; \sqrt{\bar{a}_{t-1}}x_{t-1}, (1-\bar{a}_{t-1})I)}{\mathcal{N}(x_t; \sqrt{\bar{a}_t}x_0, (1-\bar{a}_{t-1})I)}$$

$$\propto \mathcal{N} \left(x_{t-1}; \frac{\sqrt{a_t}(1-\bar{a}_{t-1})x_t + \sqrt{\bar{a}_{t-1}}(1-a_t)x_0}{1-\sqrt{\bar{a}_t}}, \Sigma_q(t) \right) \quad \text{(Property of Gaussian)}$$



$$p(x) = \int p(x|z)p(z)dz \quad \text{Intractable to estimate!}$$

$$\log p(x) = \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + D_{KL}(q(z|x) || p(z|x))$$

$$\geq \mathbb{E}_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] \quad \text{Evidence Lower Bound (ELBO)}$$

$$\log p(x_0) \geq \mathbb{E}_q \left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)} \right] \quad x = x_0, z = x_{1:T}$$

... (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -\mathbb{E}_q [D_{KL}(q(x_T|x_0) || p(x_T))] - \sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \mu_q(t), \Sigma_q(t))$$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{(1 - \bar{\alpha}_t)}} \epsilon \right), \quad \epsilon \sim \mathcal{N}(0, I)$$

Proof using bayes rule and gaussian reparameterization trick



$$p(x) = \int p(x|z)p(z)dz \quad \text{Intractable to estimate!}$$

$$\log p(x) = E_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + D_{KL}(q(z|x) || p(z|x))$$

$$\geq E_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] \quad \text{Evidence Lower Bound (ELBO)}$$

$$\log p(x_0) \geq E_q \left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)} \right] \quad x = x_0, z = x_{1:T}$$

... (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -E_q [D_{KL}(q(x_T|x_0) || p(x_T))] - \sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \mu_q(t), \Sigma_q(t))$$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{(1 - \bar{\alpha}_t)}} \epsilon \right), \quad \epsilon \sim \mathcal{N}(0, I)$$

Proof using bayes rule and gaussian reparameterization trick

The “ground truth” noise that brought x_{t-1} to x_t



$$p(x) = \int p(x|z)p(z)dz \quad \text{Intractable to estimate!}$$

$$\log p(x) = E_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + D_{KL}(q(z|x) || p(z|x))$$

$$\geq E_q \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] \quad \text{Evidence Lower Bound (ELBO)}$$

$$\log p(x_0) \geq E_q \left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)} \right] \quad x = x_0, z = x_{1:T}$$

... (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -E_q [D_{KL}(q(x_T|x_0) || p(x_T))] - \sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t, x_0) || p_{\theta}(x_{t-1}|x_t)) + \log p_{\theta}(x_0|x_1)$$

Minimize the difference of distribution means (assuming identical variance)

$$\operatorname{argmin}_{\theta} w ||\mu_q(t) - \mu_{\theta}(x_t, t)||^2$$



Learning the Denoising Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \dots \leftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t)$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma(t)) \quad \text{Conditional Gaussian}$$

Learning objective: $\operatorname{argmin}_{\theta} \|\mu_q(t) - \mu_{\theta}(x_t, t)\|^2$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{(1 - \bar{\alpha}_t)}} \epsilon \right), \quad \epsilon \sim \mathcal{N}(0, I)$$



Learning the Denoising Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t)$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma(t)) \quad \text{Conditional Gaussian}$$

Learning objective: $\operatorname{argmin}_{\theta} \|\mu_q(t) - \mu_{\theta}(x_t, t)\|^2$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{(1 - \bar{\alpha}_t)}} \epsilon \right), \quad \epsilon \sim \mathcal{N}(0, I)$$

Do we actually need to learn the entire $\mu_{\theta}(x_t, t)$?



Learning the Denoising Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t)$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma(t)) \quad \text{Conditional Gaussian}$$

Learning objective: $\operatorname{argmin}_{\theta} \|\mu_q(t) - \mu_{\theta}(x_t, t)\|^2$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{(1 - \bar{\alpha}_t)}} \epsilon \right), \quad \epsilon \sim \mathcal{N}(0, I)$$

known during inference

Unknown during
inference

Recall: this is the “ground truth”
noise that brought x_{t-1} to x_t



Learning the Denoising Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t)$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma(t)) \quad \text{Conditional Gaussian}$$

Learning objective: $\operatorname{argmin}_{\theta} \|\mu_q(t) - \mu_{\theta}(x_t, t)\|^2$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{(1 - \bar{\alpha}_t)}} \epsilon \right), \quad \epsilon \sim \mathcal{N}(0, I)$$

known during inference

Unknown during
inference

Recall: this is the “ground truth”
noise that brought x_{t-1} to x_t

Idea: just learn ϵ with $\epsilon_{\theta}(x_t, t)$!



Learning the Denoising Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t)$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma(t)) \quad \text{Conditional Gaussian}$$

Simplified learning objective: $\operatorname{argmin}_{\theta} \|\epsilon - \epsilon_{\theta}(x_t, t)\|^2$



Learning the Denoising Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t)$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma(t)) \quad \text{Conditional Gaussian}$$

Simplified learning objective: $\operatorname{argmin}_{\theta} \|\epsilon - \epsilon_{\theta}(x_t, t)\|^2$

Recall: the simplified t -step forward sample:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$$



Learning the Denoising Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t)$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma(t)) \quad \text{Conditional Gaussian}$$

Simplified learning objective: $\operatorname{argmin}_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\|^2$

Recall: the simplified t -step forward sample:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$$



Learning the Denoising Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

$$p_{\theta}(x_{0:T}) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t)$$

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma(t)) \quad \text{Conditional Gaussian}$$

Simplified learning objective: $\operatorname{argmin}_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\|^2$

Math for Classifier Guidance

Conditional diffusion models

Include condition as input to reverse process

Reverse process: $p_{\theta}(\mathbf{x}_{0:T}|\mathbf{c}) = p(\mathbf{x}_T) \prod_{t=1}^T p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}), \quad p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_{\theta}(\mathbf{x}_t, t, \mathbf{c}), \Sigma_{\theta}(\mathbf{x}_t, t, \mathbf{c}))$

Variational upper bound: $L_{\theta}(\mathbf{x}_0|\mathbf{c}) = \mathbb{E}_q \left[L_T(\mathbf{x}_0) + \sum_{t>1} D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c})) - \log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1, \mathbf{c}) \right]$.

Incorporate conditions into U-Net

- Scalar conditioning: encode scalar as a vector embedding, simple spatial addition or adaptive group normalization layers.
- Image conditioning: channel-wise concatenation of the conditional image.
- Text conditioning: single vector embedding - spatial addition or adaptive group norm / a seq of vector embeddings - cross-attention.

Classifier guidance

Using the gradient of a trained classifier as guidance

Applying Bayes rule to obtain conditional score function $\nabla_{x_t} \log q_t(x_t/y)$

$$p(x | y) = \frac{p(y | x) \cdot p(x)}{p(y)}$$

$$\implies \log p(x | y) = \log p(y | x) + \log p(x) - \log p(y)$$

$$\implies \nabla_x \log p(x | y) = \nabla_x \log p(y | x) + \nabla_x \log p(x),$$

$$\nabla_x \log p_\gamma(x | y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y | x). \quad \leftarrow \text{Classifier}$$

Guidance scale: value >1 amplifies the influence of classifier signal.

$$p_\gamma(x | y) \propto p(x) \cdot p(y | x)^\gamma.$$

Classifier guidance

Problems of classifier guidance

$$\nabla_x \log p_\gamma(x | y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y | x). \quad \leftarrow \text{Classifier}$$

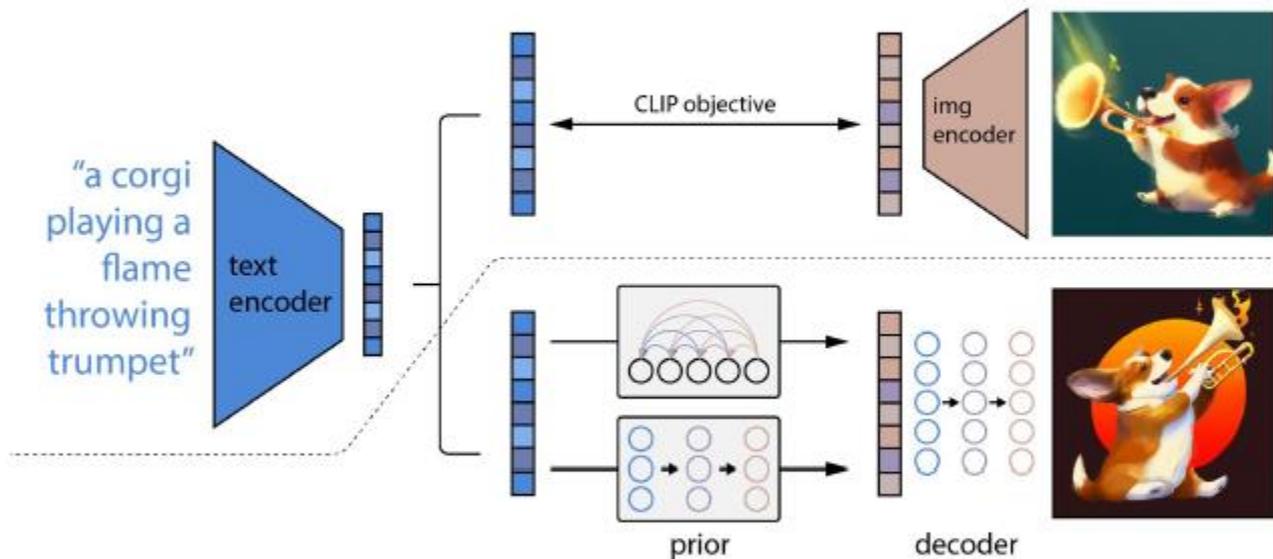
Guidance scale: value >1 amplifies the influence of classifier signal.

- At each step of denoising the input to the classifier is a noisy image x_t . Classifier is never trained on noisy image. So one needs to re-train classifier on noisy images! Can't use existing pre-trained classifiers.
- Most of the information in the input x is not relevant to predicting y , and as a result, taking the gradient of the classifier w.r.t. its input can yield arbitrary (and even adversarial) directions in input space.

Solution 1 (DALL-E 2): Use CLIP Model

DALL·E 2

Model components



Why conditional on CLIP image embeddings?

CLIP image embeddings capture high-level semantic meaning.

Slide by Soumyadip (Roni) Sengupta

Classifier-free guidance

Get guidance by Bayes' rule on conditional diffusion models

$$p(y | x) = \frac{p(x | y) \cdot p(y)}{p(x)}$$

$$\implies \log p(y | x) = \log p(x | y) + \log p(y) - \log p(x)$$

$$\implies \nabla_x \log p(y | x) = \nabla_x \log p(x | y) - \nabla_x \log p(x).$$

We proved this in classifier guidance.

$$\nabla_x \log p_\gamma(x | y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y | x).$$

$$\nabla_x \log p_\gamma(x | y) = \nabla_x \log p(x) + \gamma (\nabla_x \log p(x | y) - \nabla_x \log p(x)),$$

$$\nabla_x \log p_\gamma(x | y) = (1 - \gamma) \nabla_x \log p(x) + \gamma \nabla_x \log p(x | y).$$

↑
Score function
for unconditional
diffusion model

↑
Score function
for conditional
diffusion model

Classifier-free guidance

Get guidance by Bayes' rule on conditional diffusion models

$$\hat{\epsilon} = (1 + \omega)\epsilon_{\theta}(x_t, y) - \omega\epsilon_{\theta}(x_t) \quad \nabla_x \log p_{\gamma}(x | y) = (1 - \gamma)\nabla_x \log p(x) + \gamma\nabla_x \log p(x | y).$$



Score function for
unconditional
diffusion model



Score function for
conditional diffusio
model

In practice:

- Train a conditional diffusion model $p(x|y)$, with *conditioning dropout*: some percentage of the time, the conditioning information y is removed (10-20% tends to work well).
- The conditioning is often replaced with a special input value representing the absence of conditioning information.
- The resulting model is now able to function both as a conditional model $p(x|y)$, and as an unconditional model $p(x)$, depending on whether the conditioning signal is provided.
- During inference / sampling simply mix the score function of conditional and unconditional diffusion model based on guidance scale.