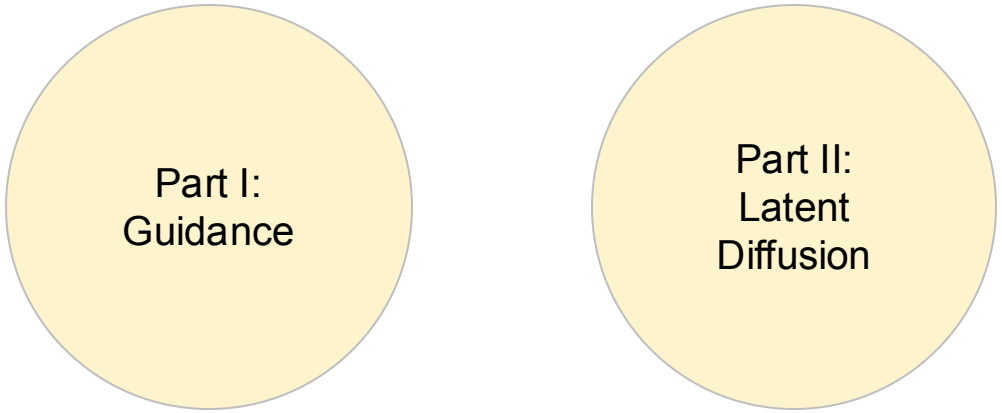
Multimodal Generative AI 2025
Guided Diffusion & Stable Diffusion

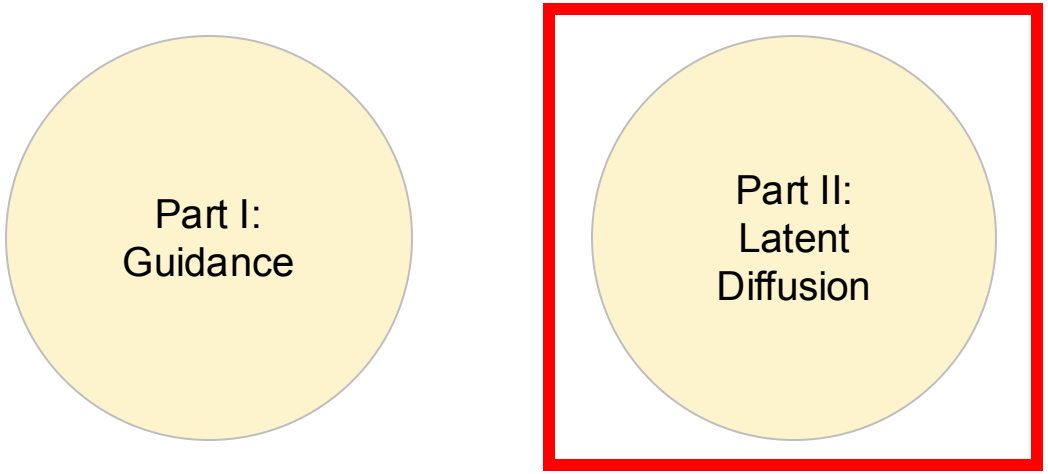
VISTA
École Polytechnique

Today's lecture



Slides adapted from many resources:
[Xi Wang, Fei-Fei Li & Andrej Karpathy & Justin Johnson & Ross Girshick & VGG]

Today's lecture



Slides adapted from many resources:
[Xi Wang, Fei-Fei Li & Andrej Karpathy & Justin Johnson & Ross Girshick & VGG]

Part I: Outline

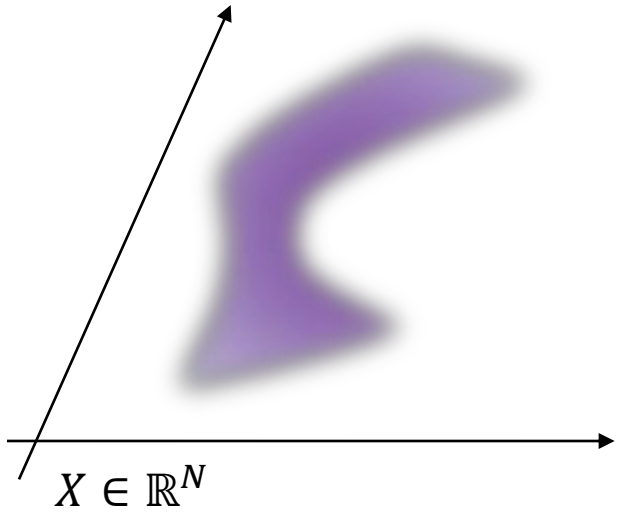
Recap: Diffusion Models

Guidance

- Control the diffusion
- Explicit condition
- Guided diffusion
- Why not guided diffusion?
- Classifier-free guidance
- Negative prompting

Last time
RECAP: Diffusion Models

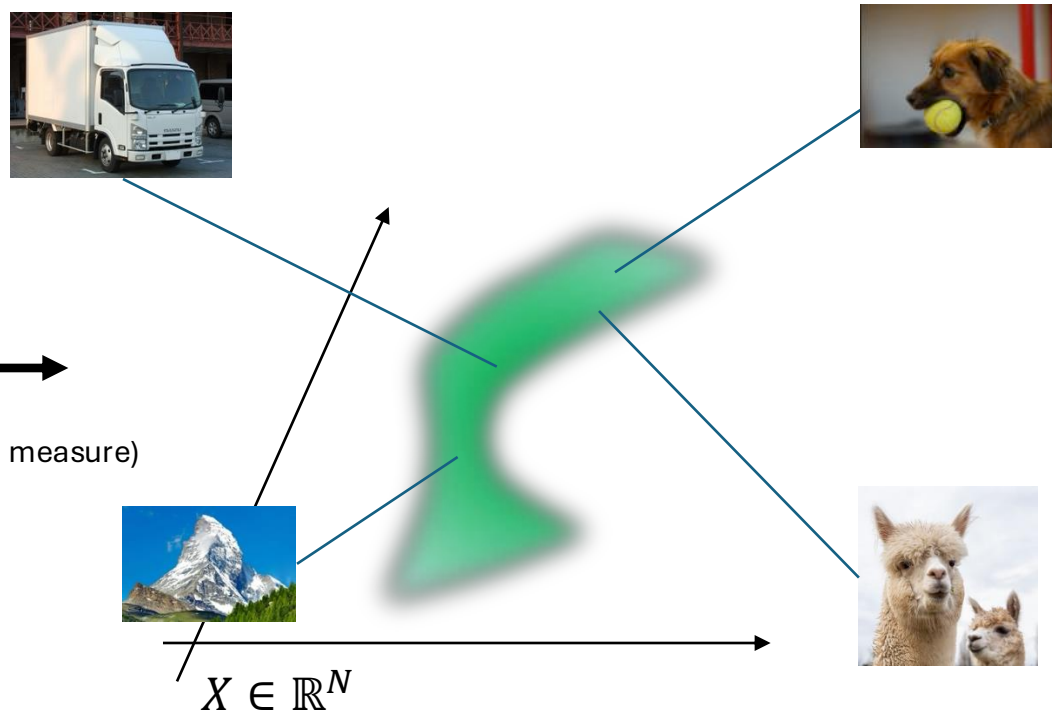
Generative Objective: Learn the distribution



Distribution of Learnt Data $P_\theta(X)$
with parameter $\theta \in \mathbb{R}^M$

find best θ

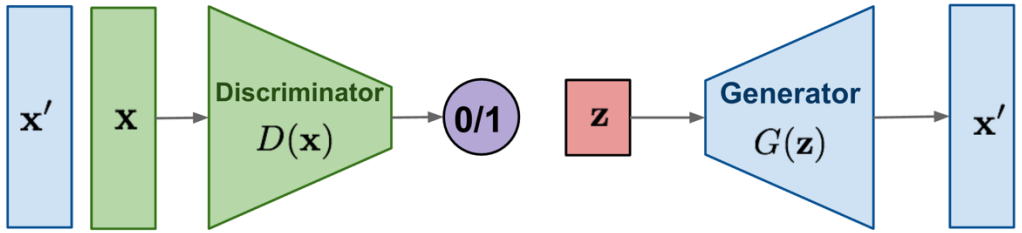
s.t. $P_\theta(X) \approx P(X)$ (with a certain measure)



Distribution of Real Data $P(X)$

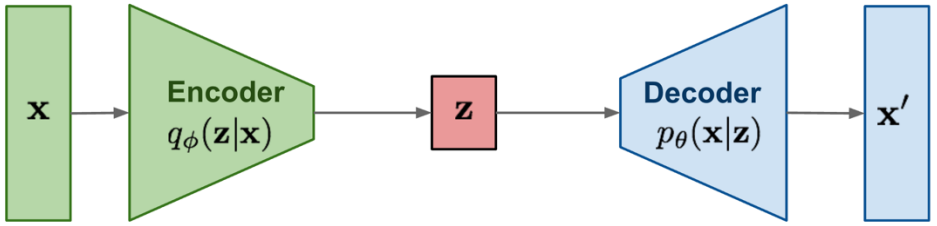
Generative Models

GAN: Adversarial training



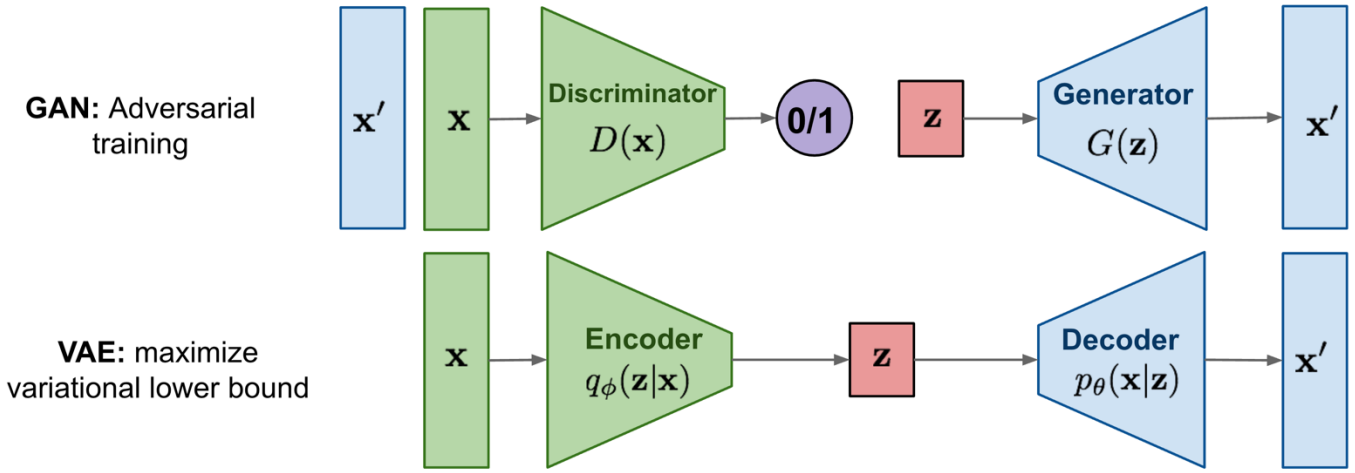
$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$
$$= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]$$

VAE: maximize variational lower bound



$$L_{\text{VAE}}(\theta,\phi) = -\log p_\theta(\mathbf{x}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$$
$$= -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$$
$$\theta^*,\phi^* = \arg\min_{\theta,\phi} L_{\text{VAE}}$$

Generative Models



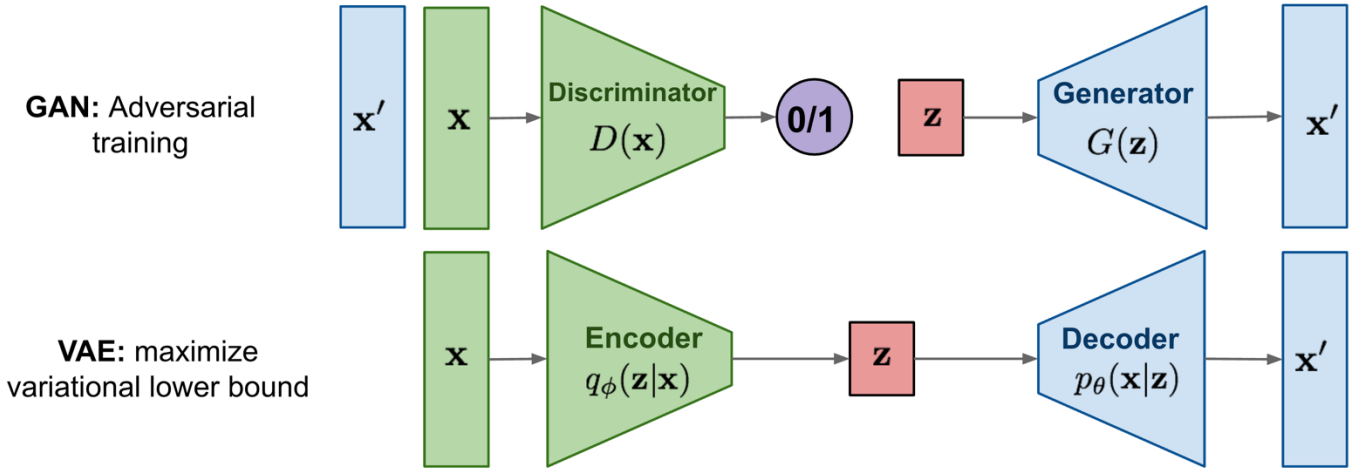
$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$
$$= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]$$

unstable training

and mode collapse (learning data, instead of distribution)

$$L_{\text{VAE}}(\theta,\phi) = -\log p_\theta(\mathbf{x}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$$
$$= -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$$
$$\theta^*,\phi^* = \arg\min_{\theta,\phi} L_{\text{VAE}}$$

Generative Models



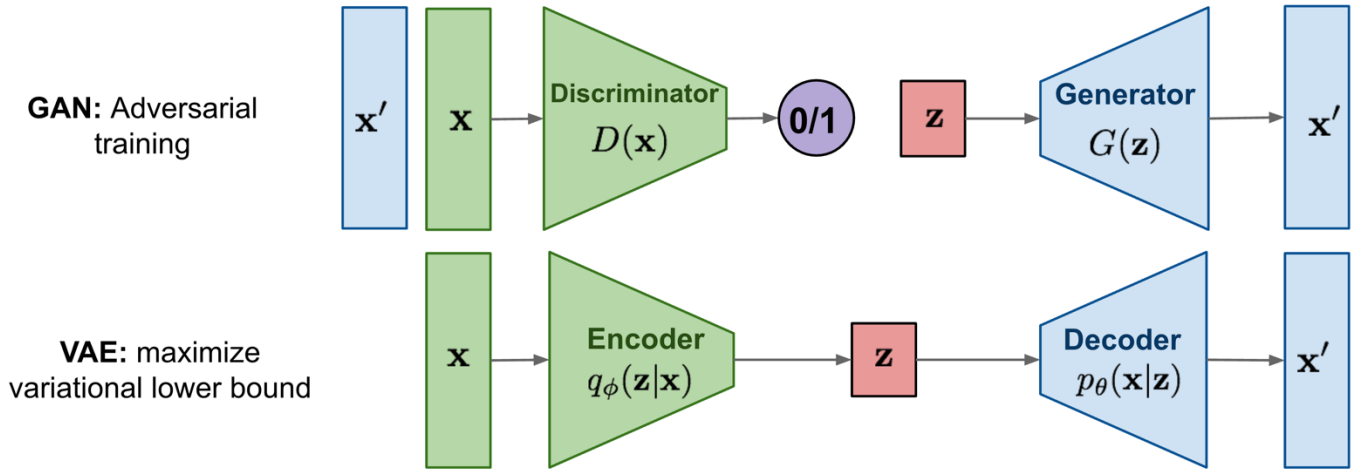
$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$
$$= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]$$

unstable training
and mode collapse (learning data, instead of distribution)

$$L_{\text{VAE}}(\theta,\phi) = -\log p_\theta(\mathbf{x}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$$
$$= -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$$
$$\theta^*,\phi^* = \arg\min_{\theta,\phi} L_{\text{VAE}}$$

under-represtation of the distribution,
posteriori collapse (Gaussian Priori is not realistic)

Generative Models

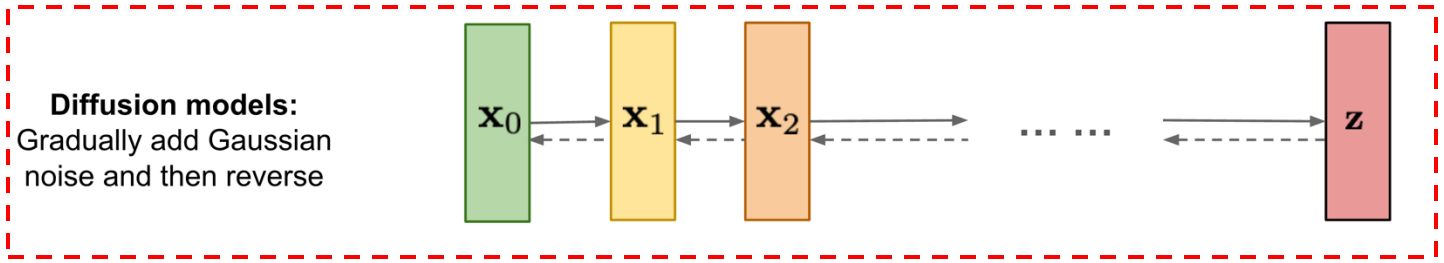


$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$
$$= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]$$

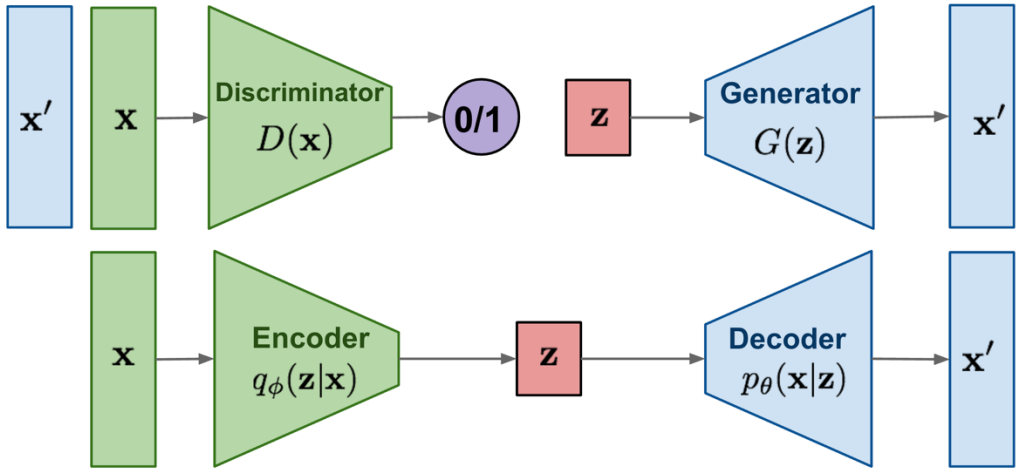
unstable training
and mode collapse (learning data, instead of distribution)

$$L_{\text{VAE}}(\theta,\phi) = -\log p_\theta(\mathbf{x}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$$
$$= -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$$
$$\theta^*,\phi^* = \arg\min_{\theta,\phi} L_{\text{VAE}}$$

under-represtation of the distribution,
posteriori collapse (Gaussian Priori is not realistic)



Generative Models

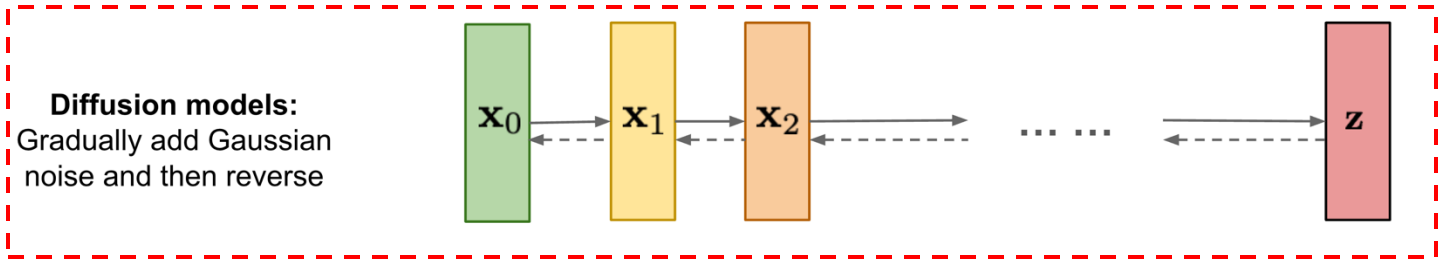


$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$
$$= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]$$

unstable training
and mode collapse (learning data, instead of distribution)

$$L_{\text{VAE}}(\theta,\phi) = -\log p_\theta(\mathbf{x}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$$
$$= -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$$
$$\theta^*,\phi^* = \arg\min_{\theta,\phi} L_{\text{VAE}}$$

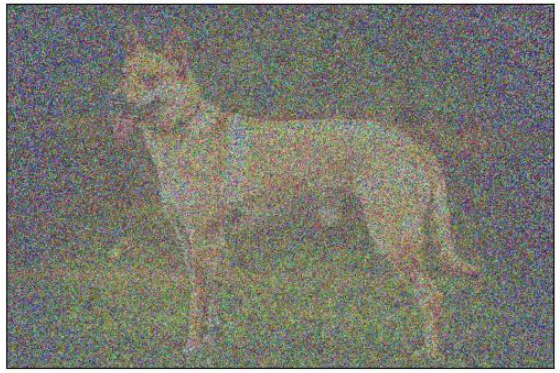
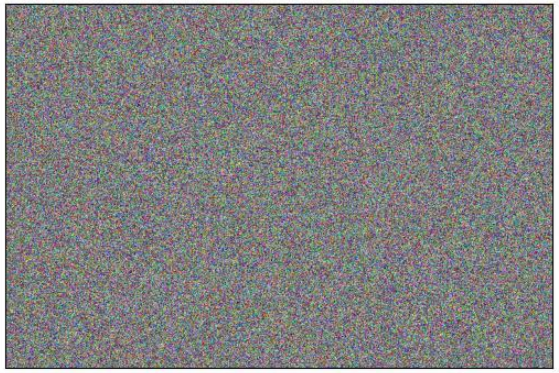
under-represtation of the distribution,
posteriori collapse (Gaussian Priori is not realistic)



Better representation capacity,
and learn the whole distribution.

Generative Objective: Forward Process

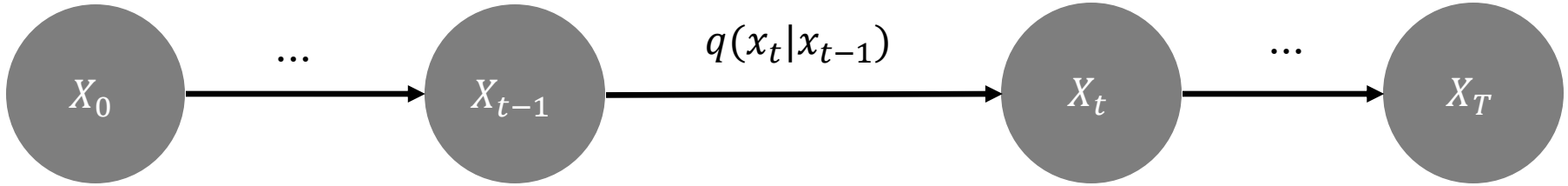
Generative Objective: Forward Process

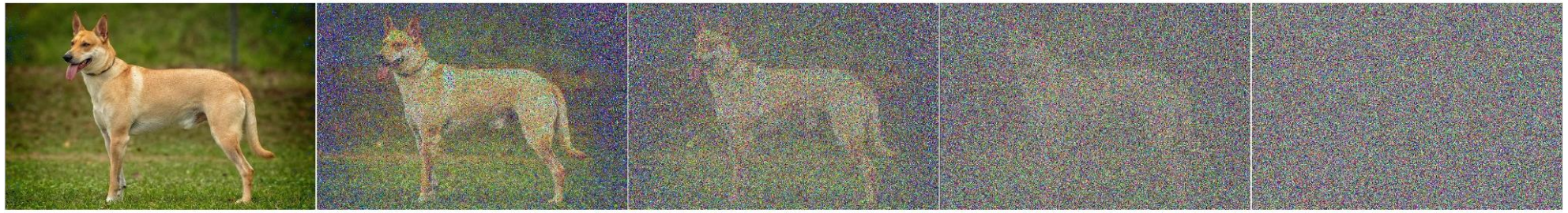




How to push an image to a Gaussian?

Easy, let's add noise !

Generative Objective: Forward Process



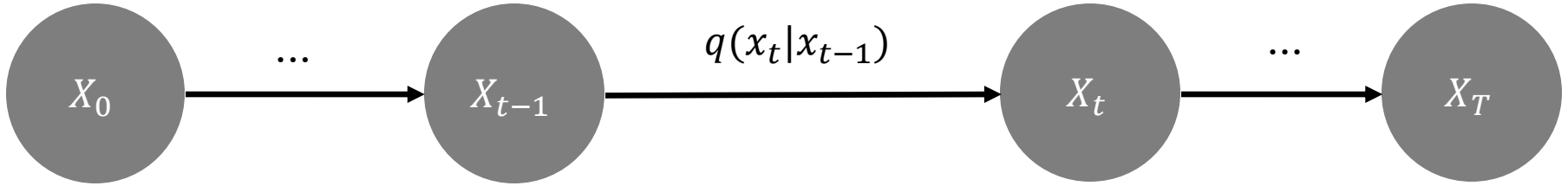


We call this **a Forward Process.**

- Original image at X_0 and pure noise at X_T
- We repeat the noising T times
- $\beta_t \in (0,1)$ is a noise schedule, ie. linear

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

Generative Objective: Forward Process



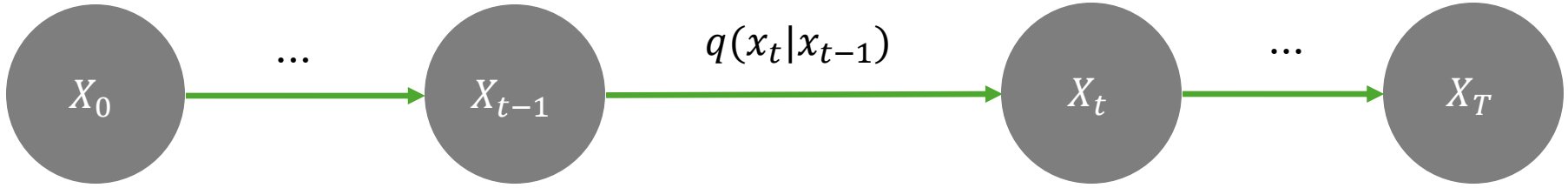
We call this **<u>a Forward Process.</u>**

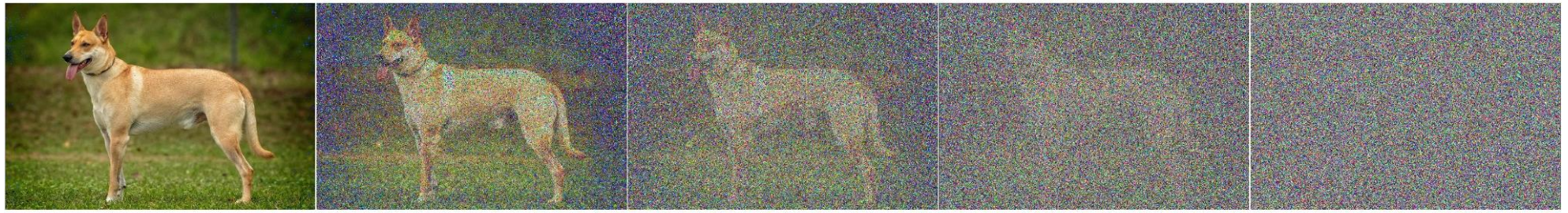
- Original image at X_0 and pure noise at X_T
- We repeat the noising T times
- $\beta_t \in (0,1)$ is a noise schedule, ie. linear

$$q(x_t|x_{t-1})$$

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

DDPM: Forward Process





- Original image at X_0 and pure noise at X_T
- We repeat the noising T times
- $\beta_t \in (0,1)$ is a noise schedule

Forward:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

("Shortcut")
Sample any step using x_0 :

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$$

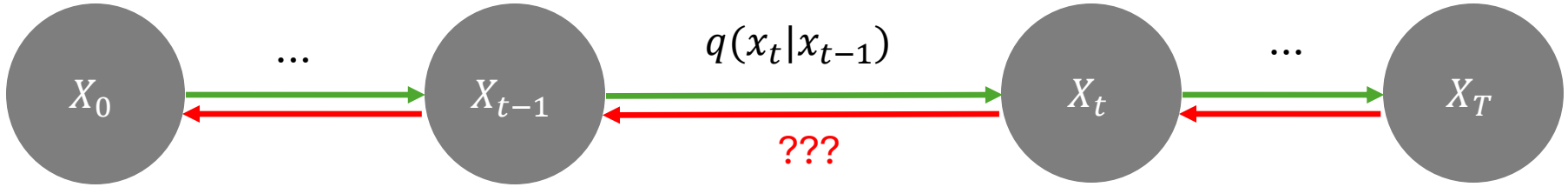
$$\alpha_t = 1 - \beta_t \qquad \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

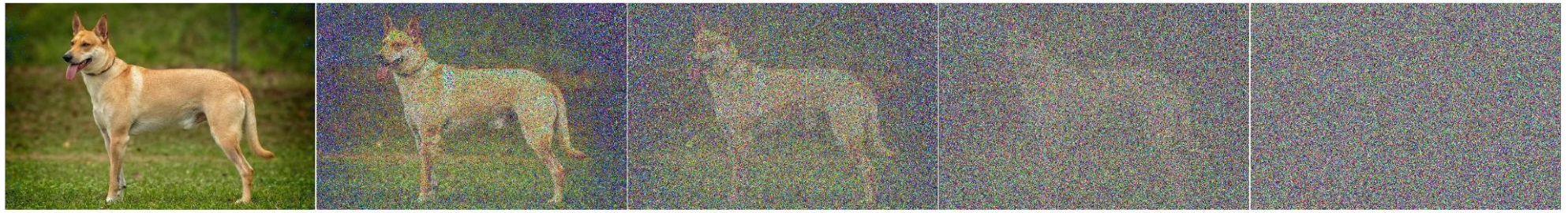
Generative Objective: Reverse Process *Summary*

1. In the "Reverse Diffusion process," the idea is to reverse the forward diffusion process
2. Slowly, iteratively try to reverse the corruption performed on images in the forward process
3. The reverse process starts where the forward process ends
    1. The benefit of starting from a simple space is that we know how to get/sample a point from this simple distribution (think of it as any point outside the data subspace)
4. Our goal here is to figure out how to return to the data subspace.
5. However, the problem is that we can take infinite paths starting from a point in this "simple" space, but only a fraction of them will take us to the "data" subspace
6. In diffusion, this is done by referring to the small iterative steps taken during forward process
7. The PDF that satisfies the corrupted images in the forward process differs slightly at each step
8. So, reverse: use a DNN at each step to predict the PDF parameters of forward process
9. And once we train the model, we can start from any point in the simple space and use the model to iteratively take steps to lead us back to the data subspace
10. In reverse, we iteratively perform the ***"denoising"*** in small steps, starting from a noisy image
11. This approach for training and generating new samples is much more stable than GANs and better than previous approaches like variational autoencoders (VAE) and normalizing flows

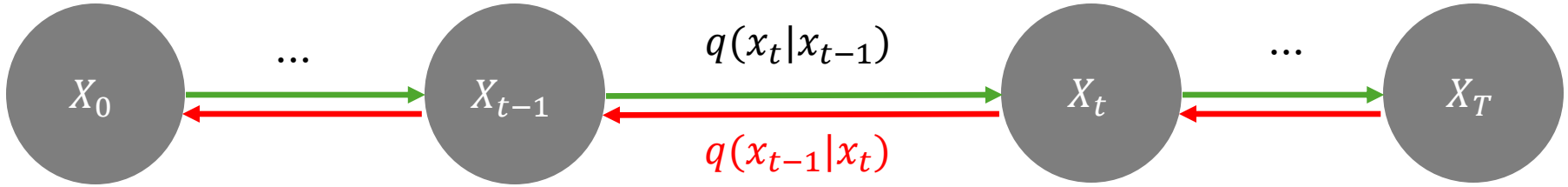
[Vaibhav Singh]

Generative Objective: Reverse Process

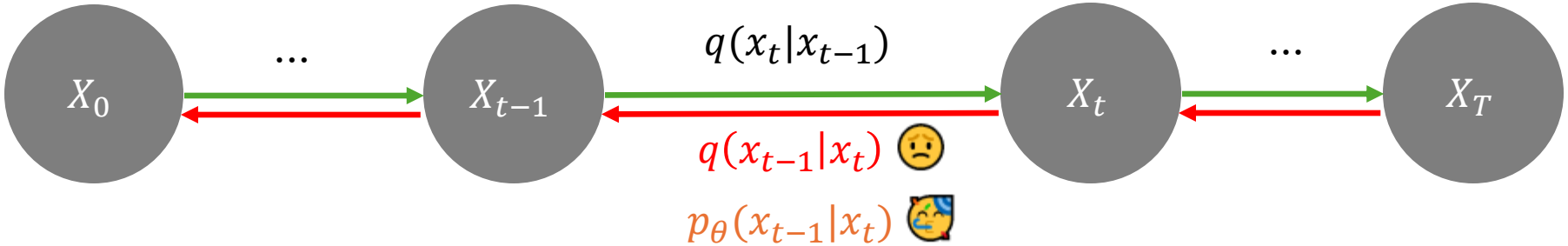




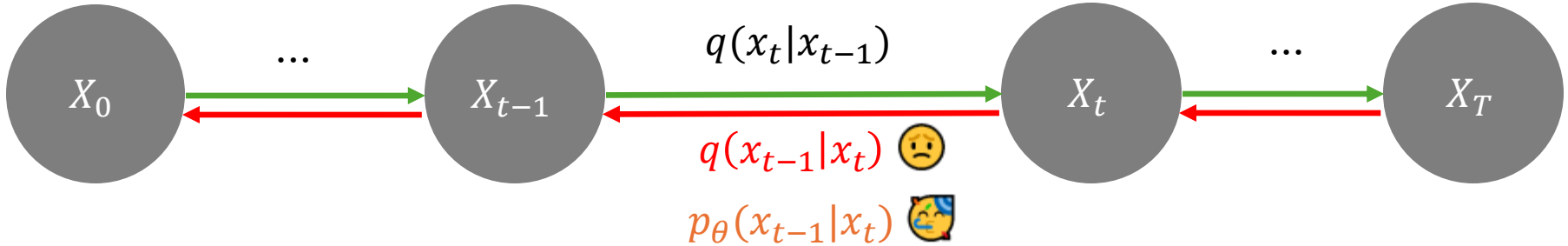
Generative Objective: Reverse Process



Generative Objective: Reverse Process



Generative Objective: Reverse Process



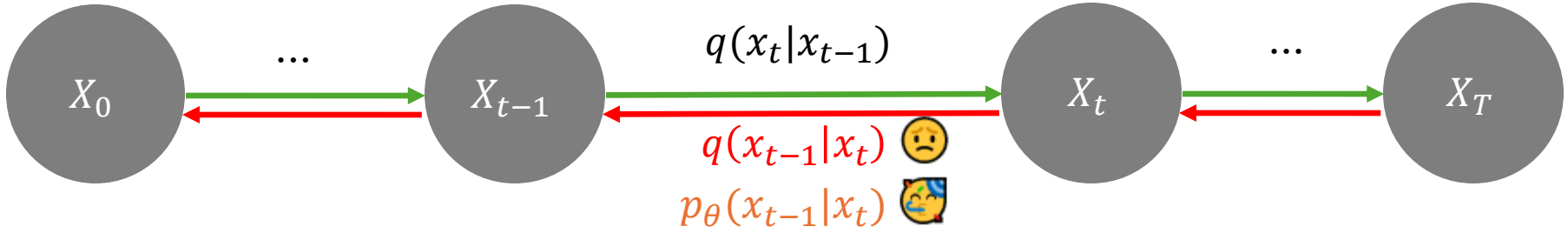
A very nice property of Gaussian:
if $q(x_t|x_{t-1})$ is a Gaussian with small β (another reason we need many steps!)
→ then, $q(x_{t-1}|x_t)$ is also a Gaussian.
Therefore, we learn this Gaussian's mean and variance by a network approximated $p_\theta(x_{t-1}|x_t)$

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

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Learnable parameters

DDPM: Reverse (=Generative) Process



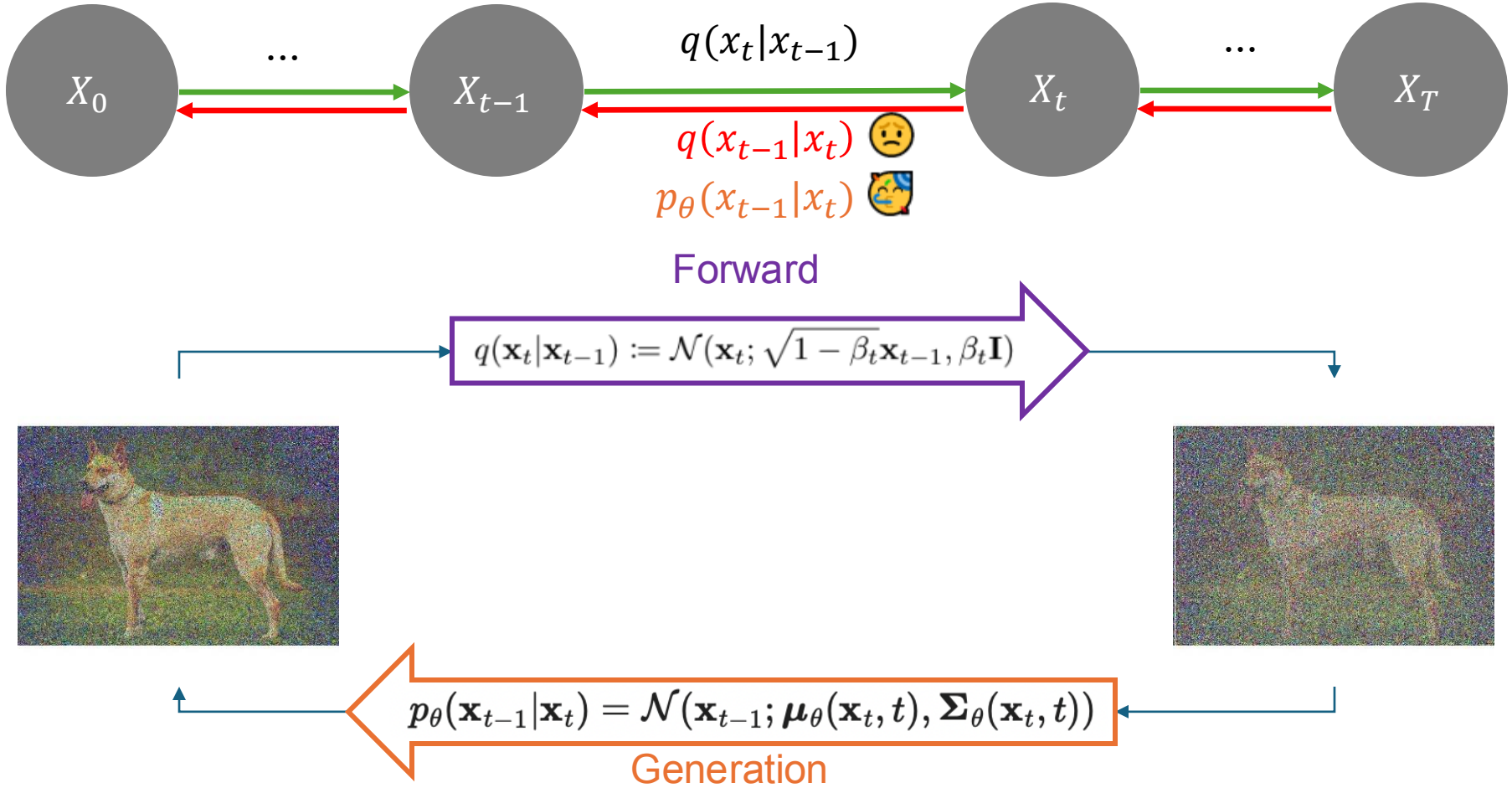
Generation:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

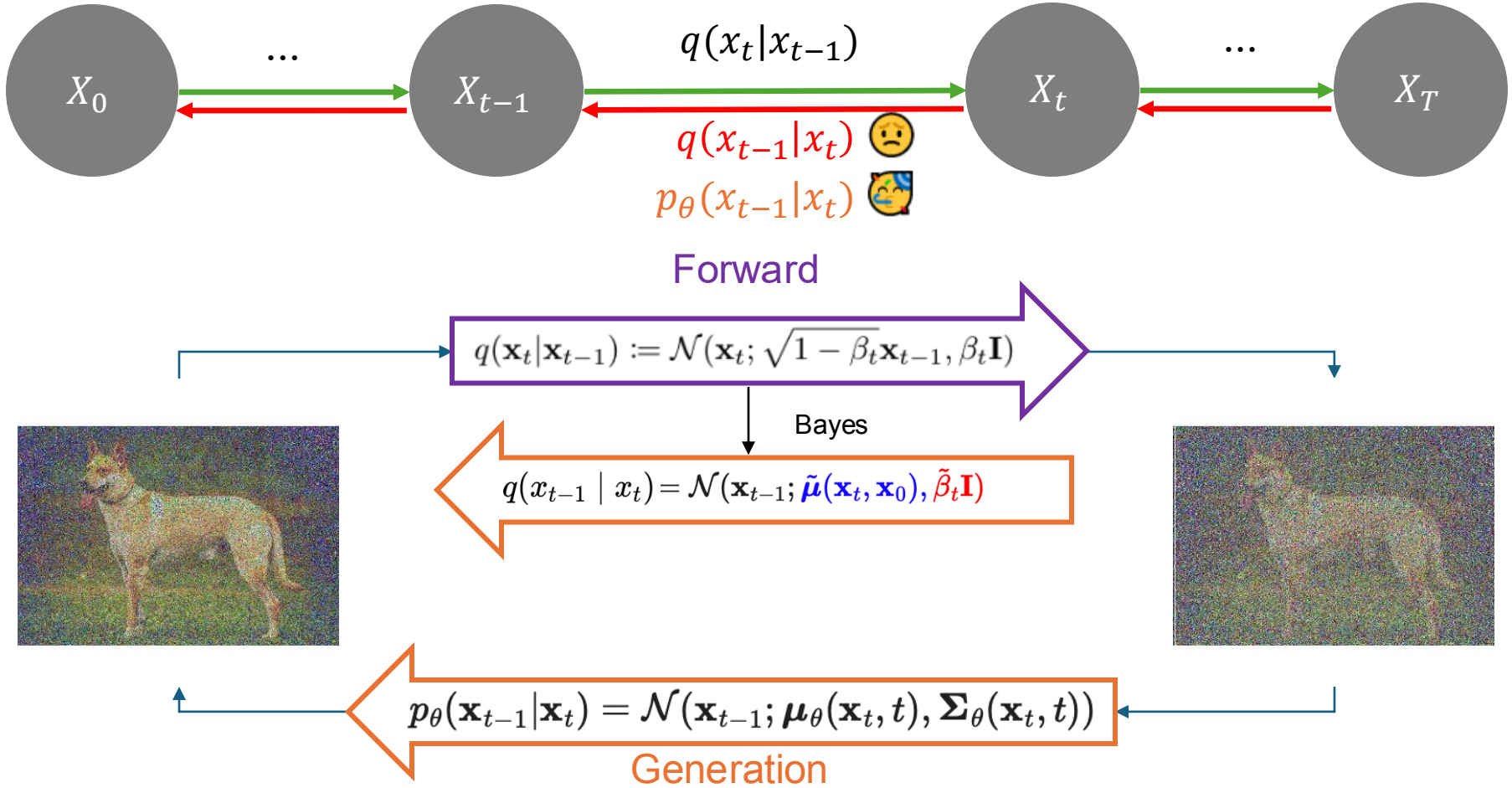
Learnable parameters

A very nice property of Gaussian:
if $q(x_t|x_{t-1})$ is a Gaussian with small β (another reason we need many steps!)
→ then, $q(x_{t-1}|x_t)$ is also a Gaussian.
Therefore, we learn this Gaussian's mean and variance by a network approximated $p_\theta(x_{t-1}|x_t)$

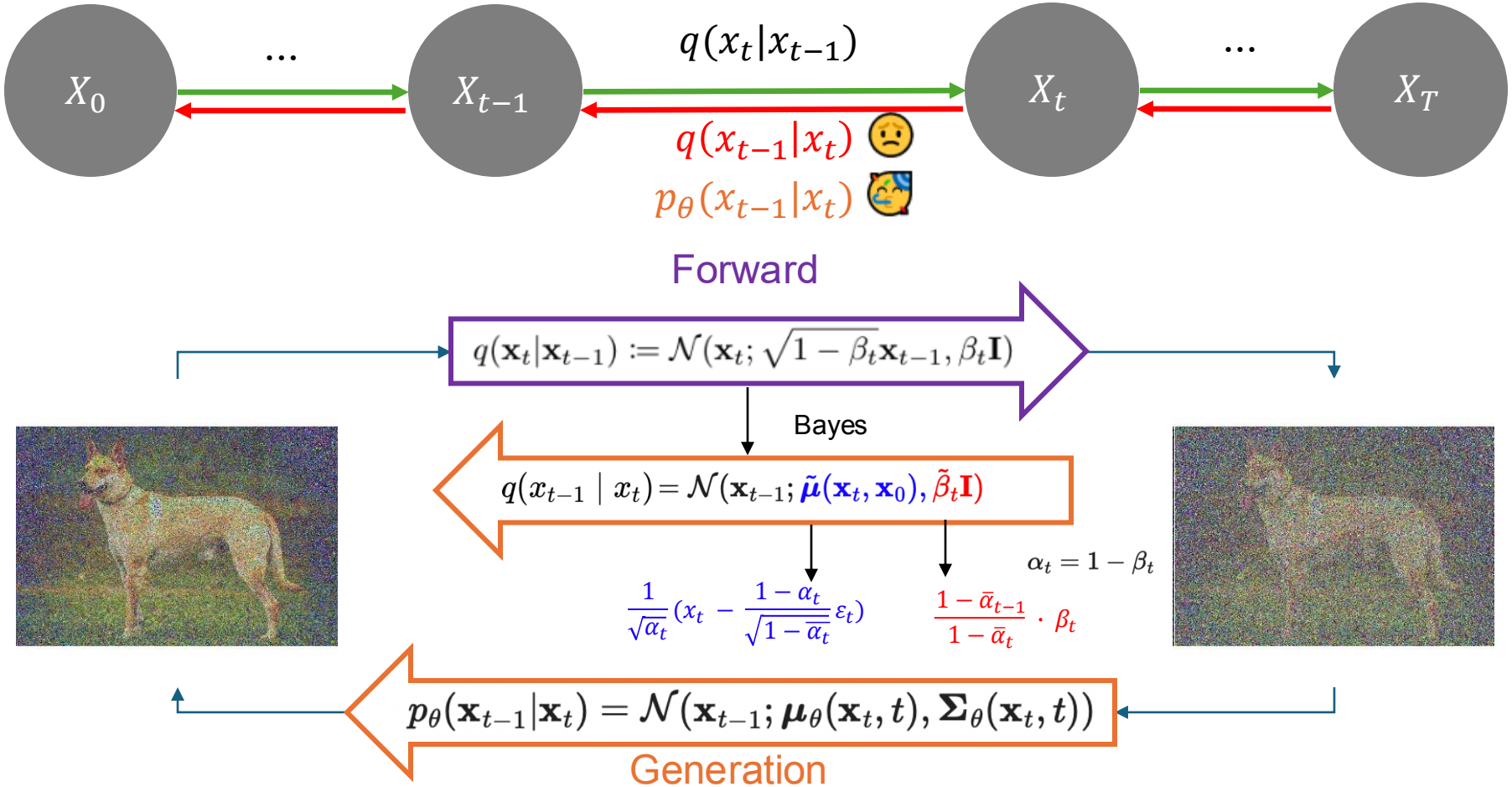
DDPM: Generative Process



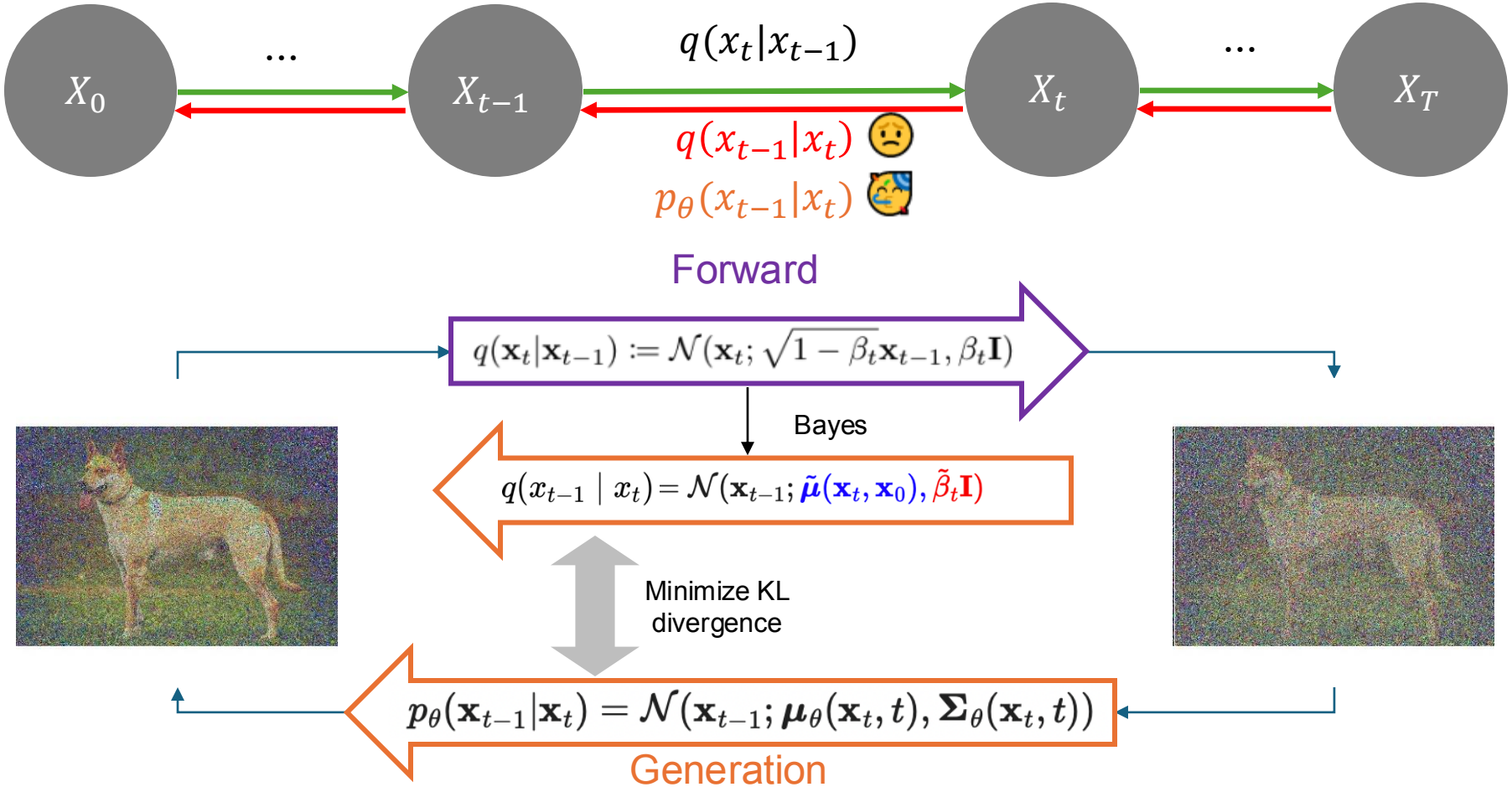
DDPM: Reverse Process



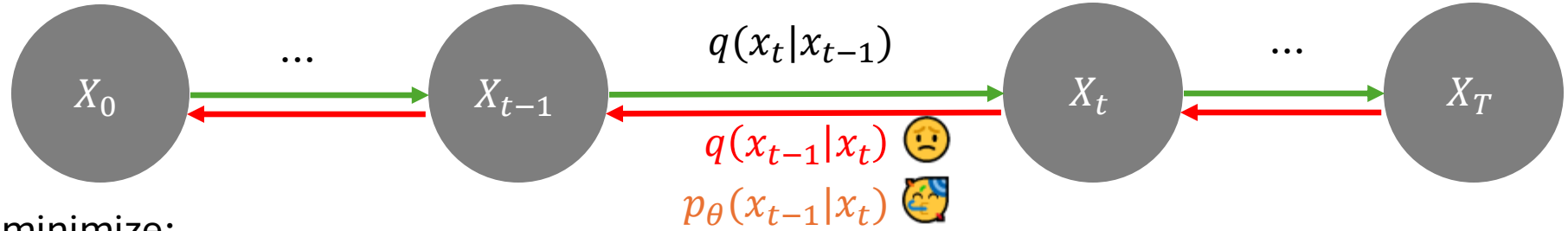
DDPM: Reverse/Generative Process



DDPM: Reverse/Generative Process



Generative Objective: Loss



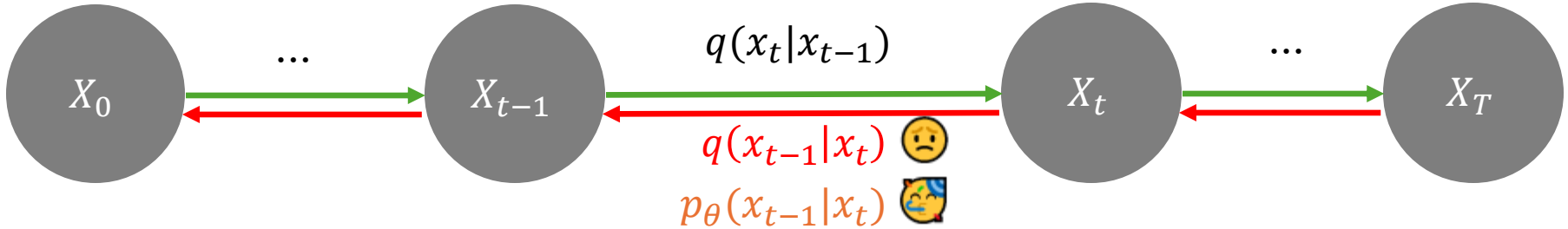
We want to minimize:

$$L_{\text{VLB}} = L_T + L_{T-1} + \cdots + L_0$$

$$\text{where } L_T = D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))$$

$$L_t = D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1$$

Generative Objective: Loss



We want to minimize:

$$L_{\text{VLB}} = L_T + L_{T-1} + \cdots + L_0$$
$$\text{where } L_T = D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))$$
$$L_t = D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \le t \le T-1$$

=

Minimizing predicted noise based on data:

$$L_t^{\text{simple}} = \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\left[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\right]$$

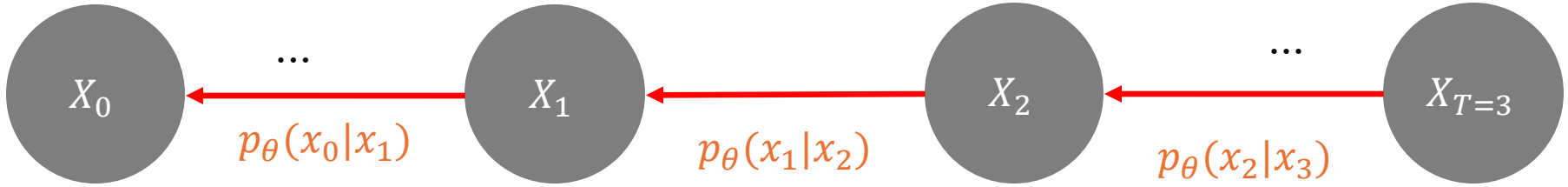
$$L_{t-1} = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2\right]$$

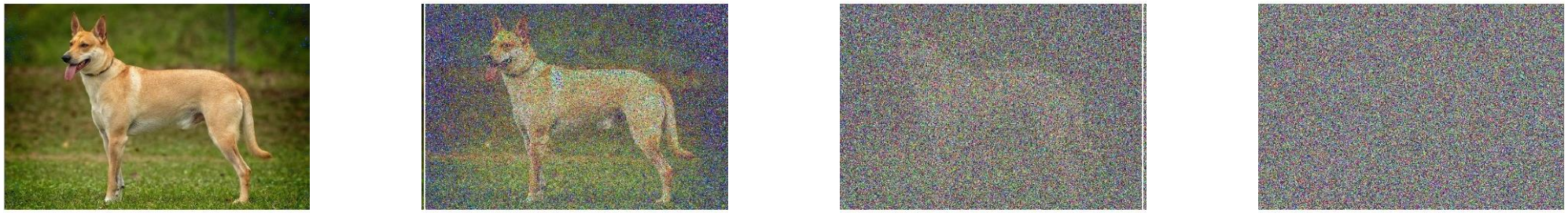
In code:

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

DDPM: Sampling

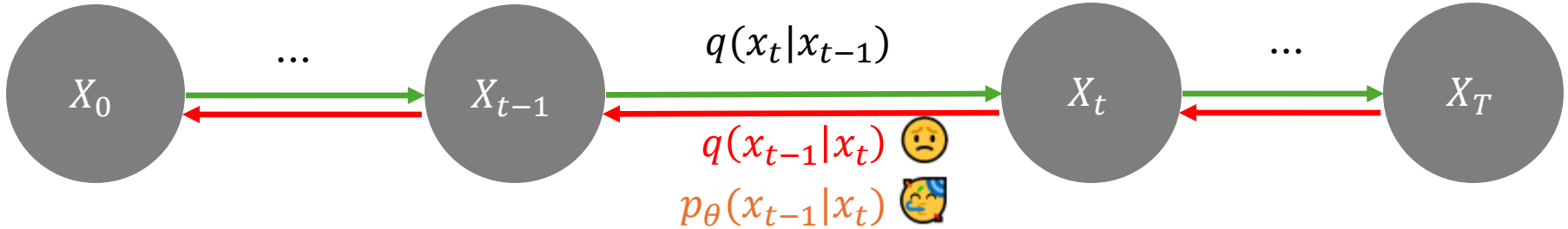




$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1};\ \mu_\theta(x_t,t),\ \boxed{\mathbf{\Sigma}_\theta(\mathbf{x}_t,t)})$$

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha_t}}}\varepsilon_\theta(x_t,t)\right) + \boxed{\sigma_t \mathrm{z}}$$

Generative Objective: Loss



We want to minimize:

$$L_{\text{VLB}} = L_T + L_{T-1} + \cdots + L_0$$
$$\text{where } L_T = D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))$$
$$L_t = D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \le t \le T-1$$

=

Minimizing predicted noise based on data:

$$L_t^{\text{simple}} = \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\left[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\right]$$

In code:

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

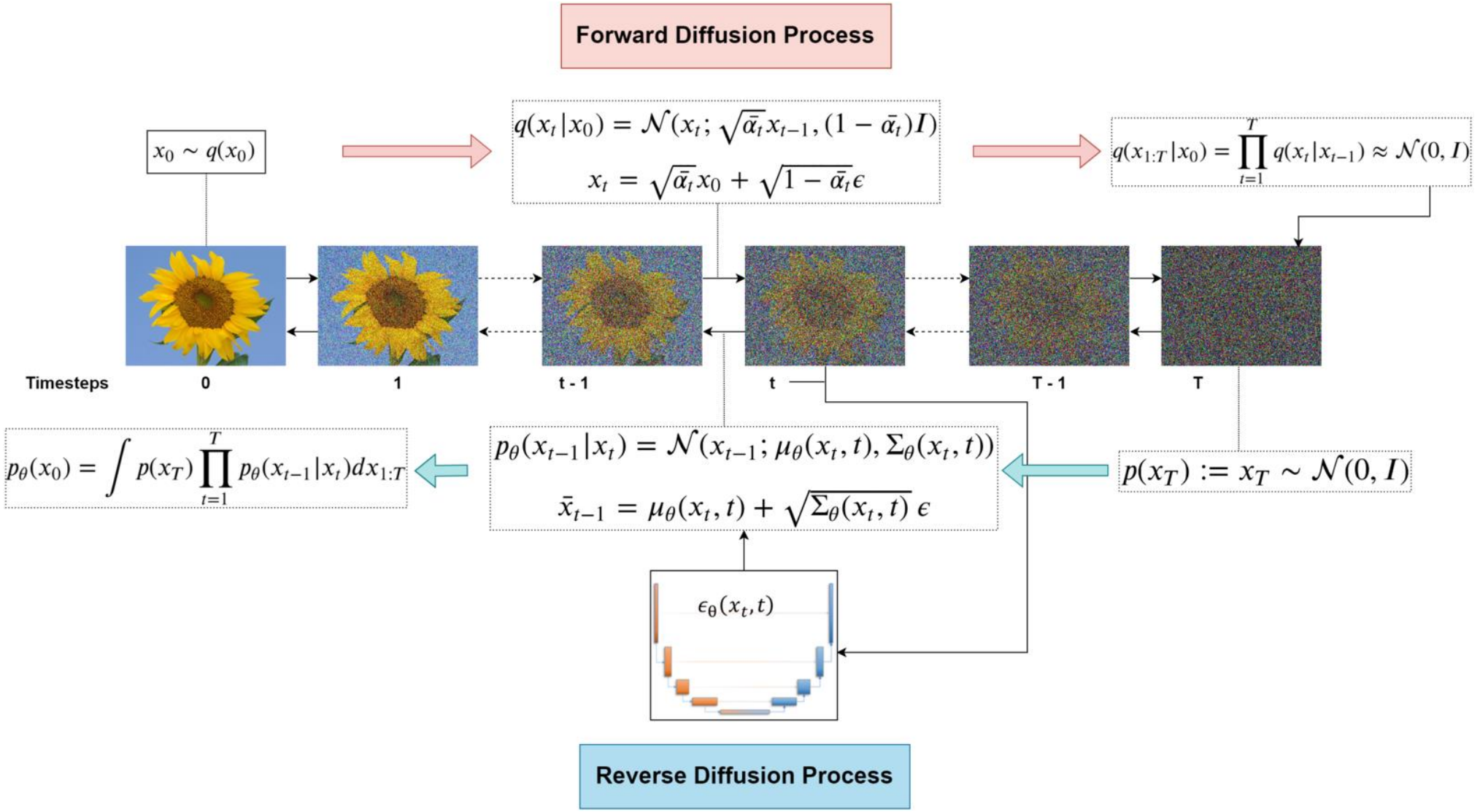
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,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

$x \sim N(\mu, \sigma^2)$
$z \sim N(0,1)$
$x = \mu + \sigma z$

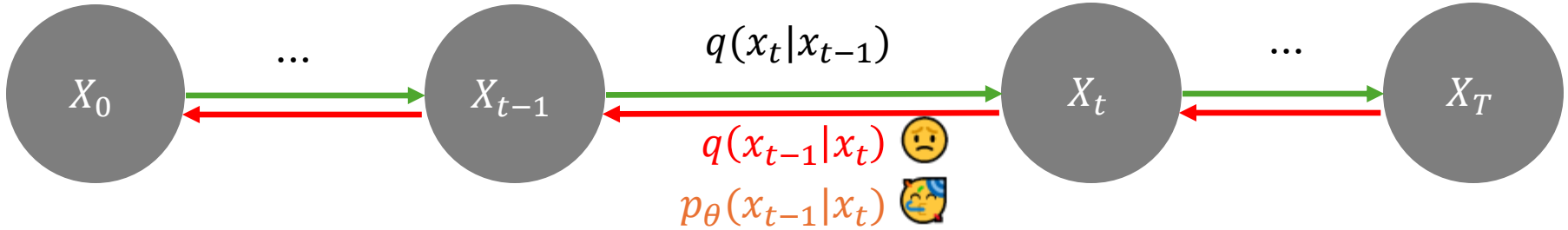
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

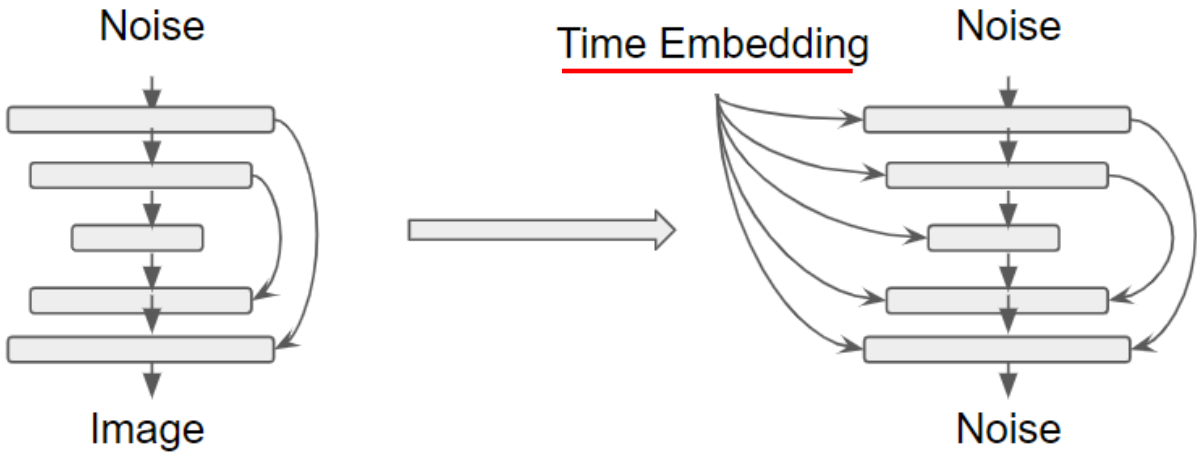
$$\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta\right) \qquad \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$

Denoising Diffusion Probabilistic Model (DDPM)



Generative Objective: Which networks?



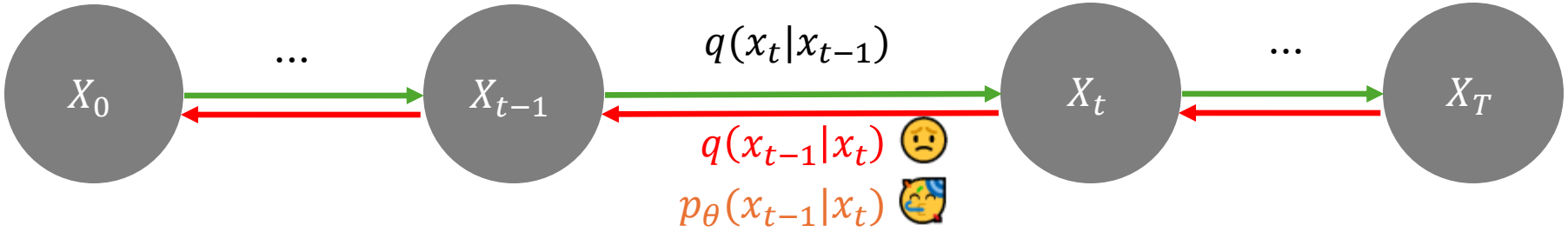


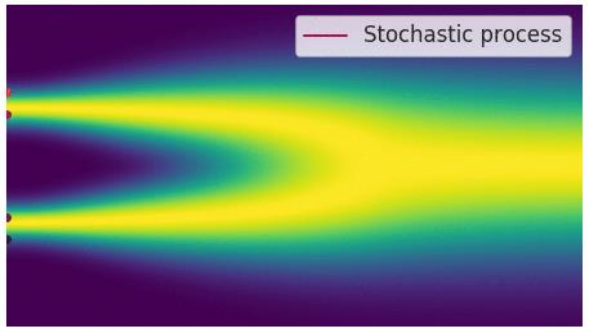
A standard U-Net to predict noise from previous noise and time information.

U-Net for standard encoding

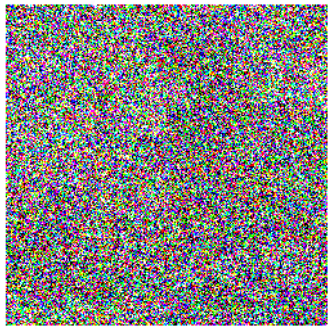
A (very simple) U-Net for diffusion Model

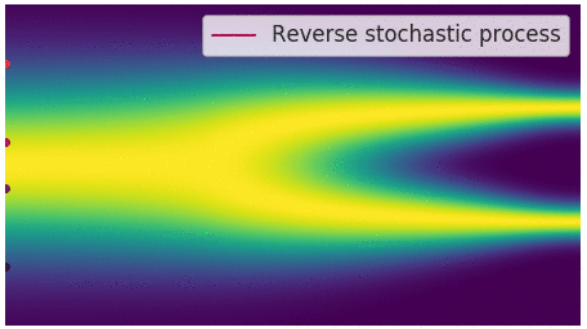
Generative Objective: Reverse Process





Stochastic noising





Stochastic denoising

Generative Objective: Reverse Process
Convergence

- GANs: if the Discriminator can successfully differentiate between real/fake then we stop the training

- VAE: reconstruction loss (meaningful)

- Diffusion models: complicated: we need to measure the distance between two distributions → FID

Results: Denoising Diffusion Probabilistic Model (DDPM)

Sampled results:
LSUN Church Dataset

Sampled results:
LSUN Church Bedroom

Results: Denoising Diffusion Probabilistic Model (DDPM)

Sampled results:
CelebA-HQ Dataset

Part I: Outline

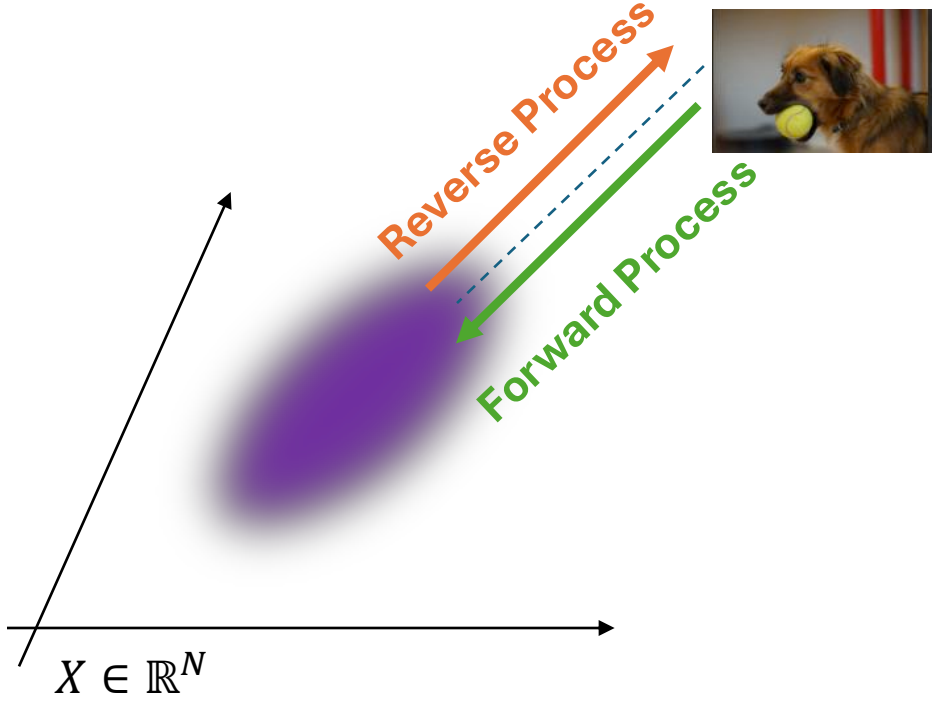
Recap: Diffusion Models

Guidance

- Control the diffusion
- Explicit condition
- Guided diffusion
- Why not guided diffusion?
- Classifier-free guidance
- Negative prompting

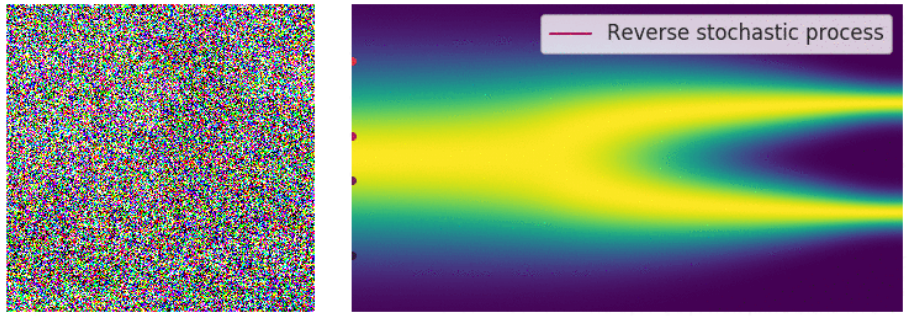
Control the diffusion

Control the Diffusion Model



Distribution of Learnt Data $P_\theta(X)$ with parameter $\theta \in \mathbb{R}^M$

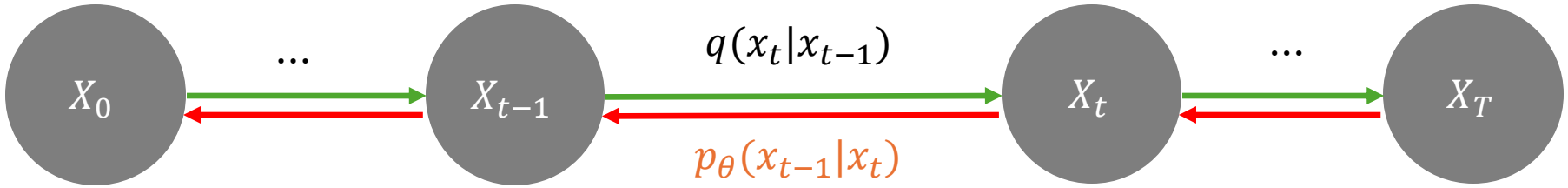
Good, it means one noise gives me an image!



But how can I achieve **control** on this? For example, I want a cat image, rather than others.

Or even more complicated: "*A stained glass window of a panda eating bamboo*." – text-to-image generation

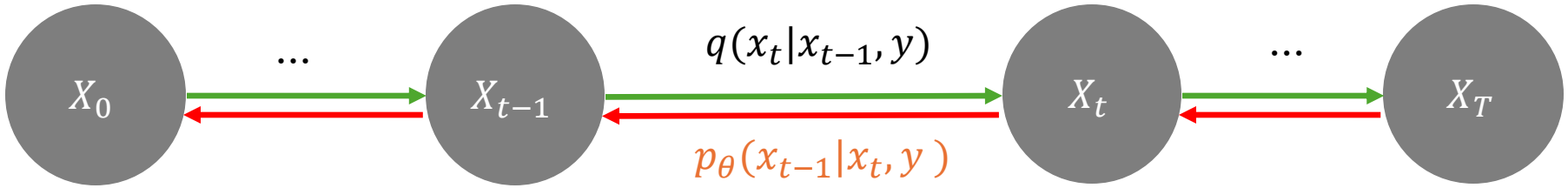
Control the Diffusion Model

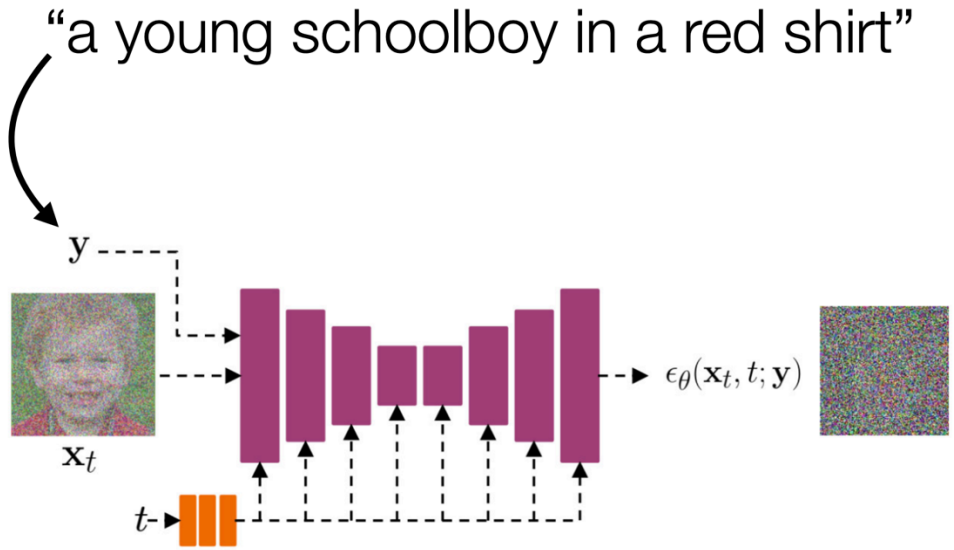


Where is the control?
How did we do with VAE? This sounds a familiar question.

Explicit condition

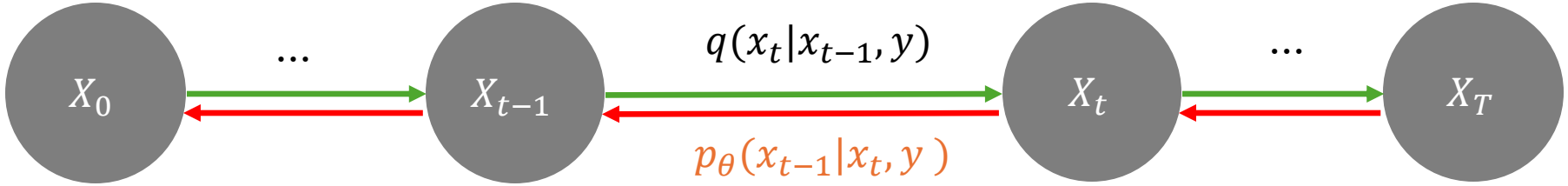
Control the Diffusion Model: Explicit Condition

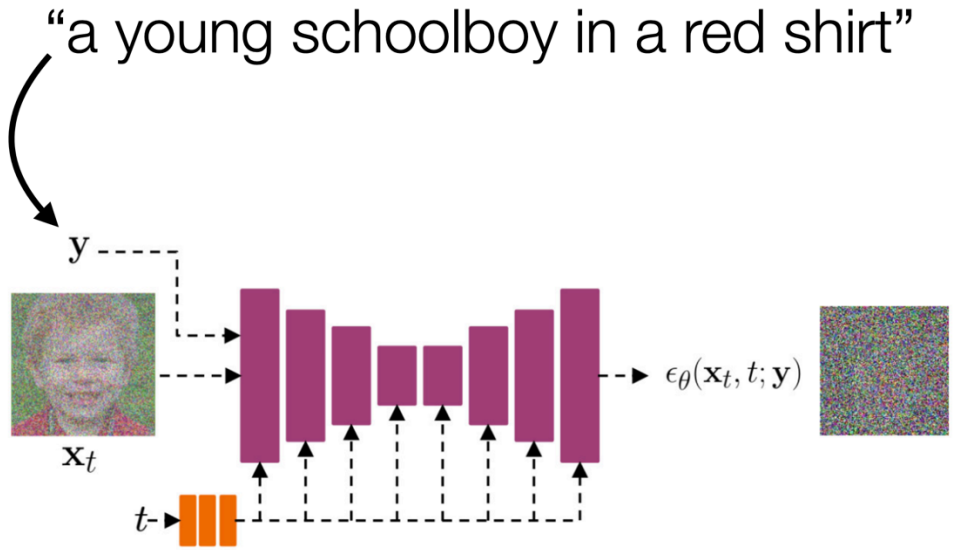




We can add it directly.

Control the Diffusion Model: Explicit Condition

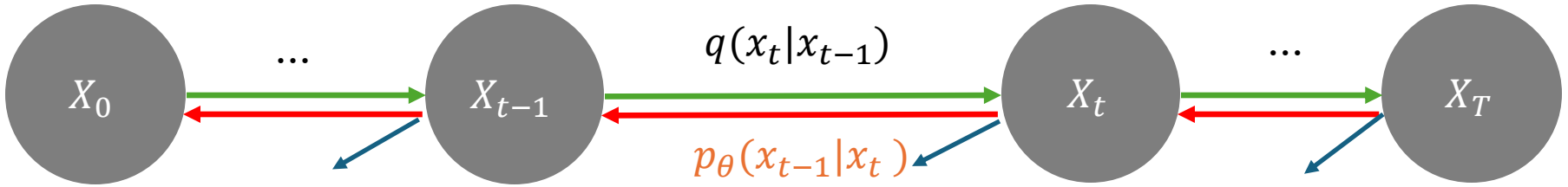




We can add it directly, but is this an effective way? Why?

Guided diffusion

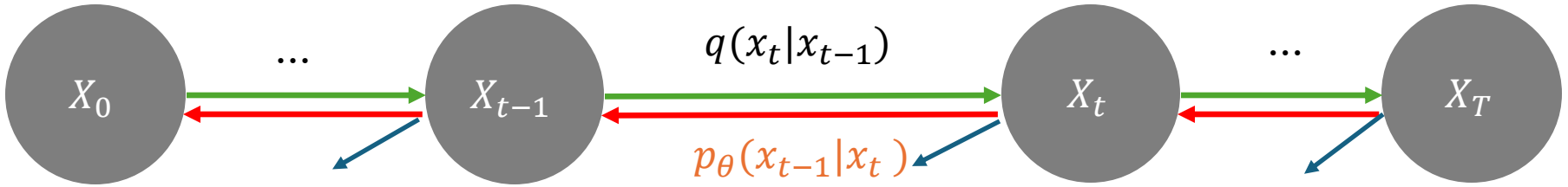
Control the Diffusion Model: Guided Diffusion



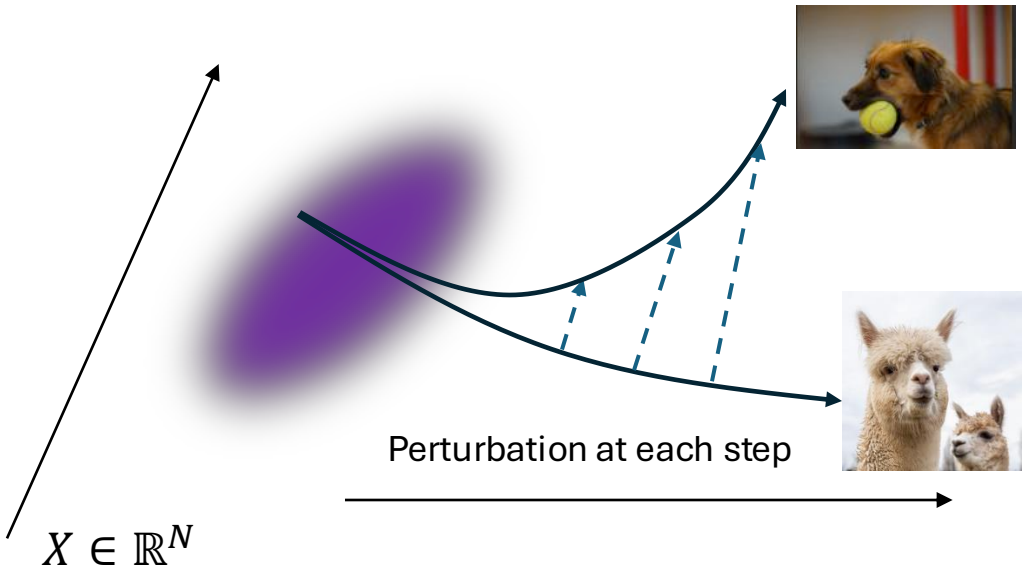
Shift to a dog label!

Let's perturb it step-by-step during the generation!

Control the Diffusion Model: Guided Diffusion

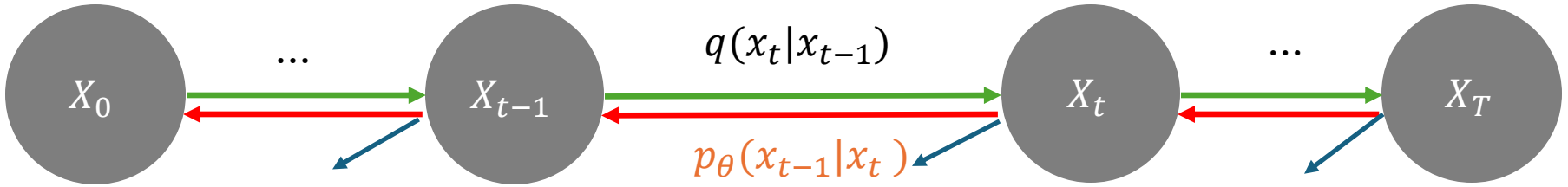


Shift to a dog label!

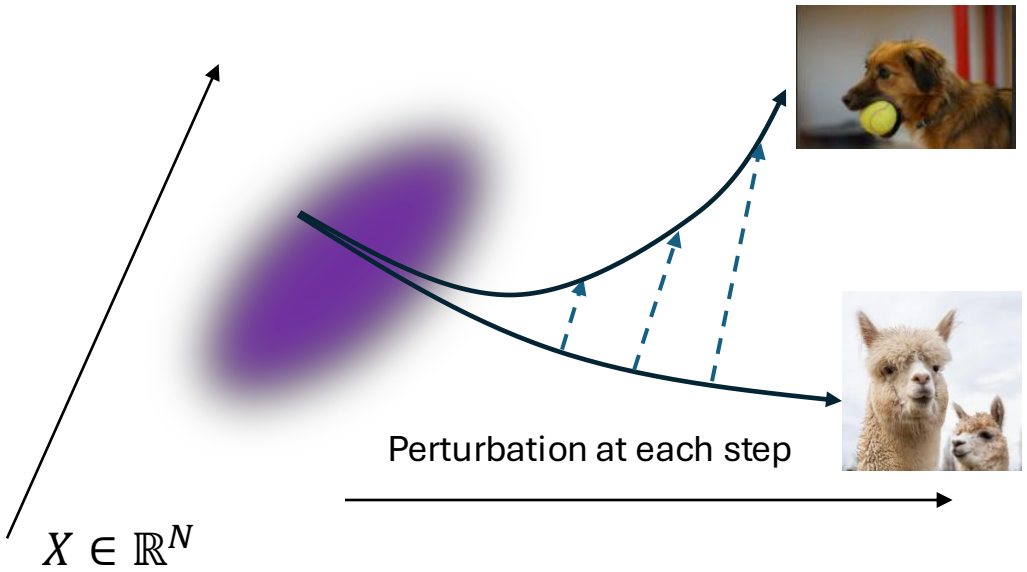


How to do it?

Control the Diffusion Model: Guided Diffusion



Shift to a dog label!



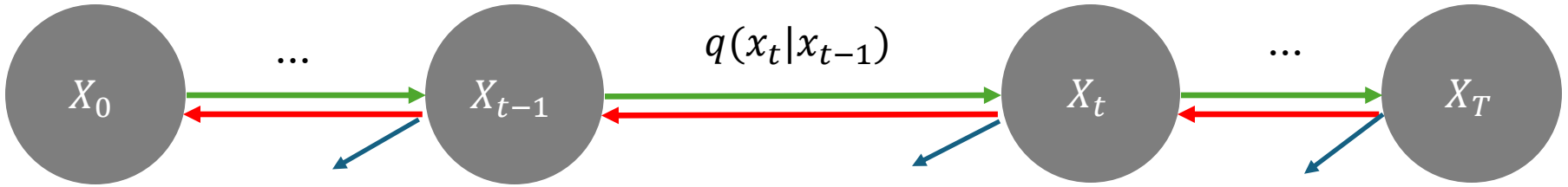
It's like a classifier: $p_\phi(y|x)$

We just look at its gradient:

$$\nabla_x \log p_\phi(y|x)$$

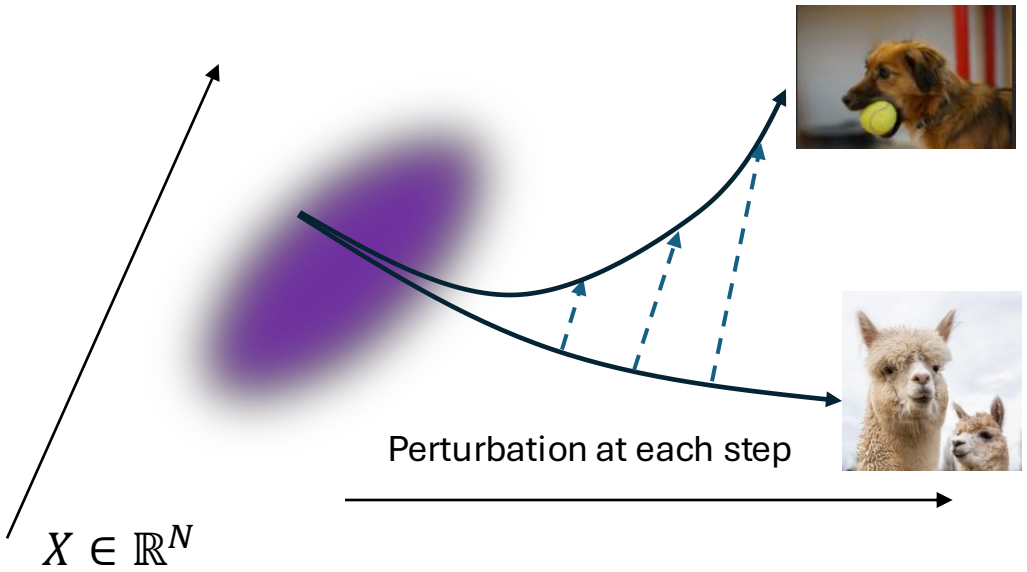
Such that the generated x is similar to the condition label.

Control the Diffusion Model: Guided Diffusion



$$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$$

Shift to a dog label!



It's like a classifier: $p_\phi(y|x)$

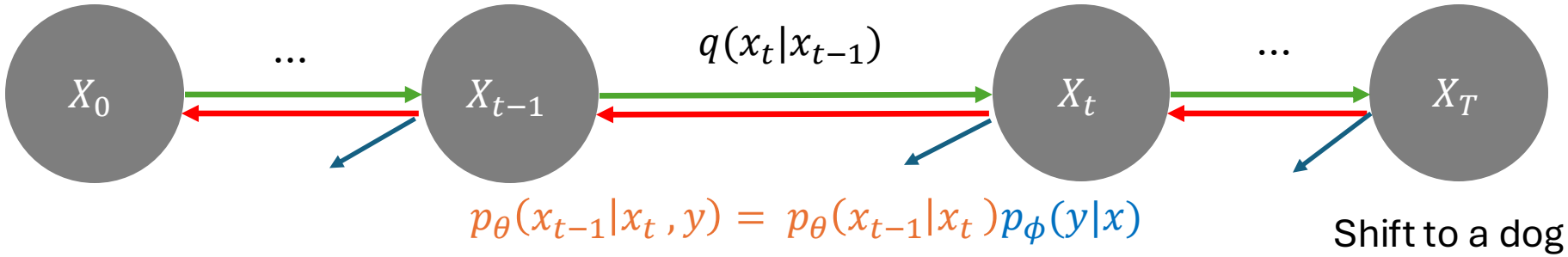
We just look at its gradient:

$$\nabla_x \log p_\phi(y|x)$$

Such that the generated x is similar to the condition label.

In sampling: $\epsilon_\theta(x_t, t) + \nabla_x \log p(y|x)$

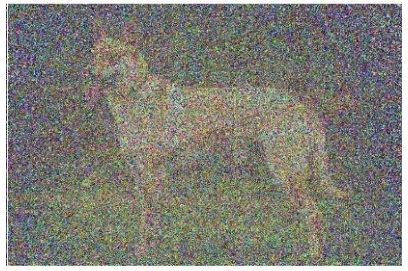
Control the Diffusion Model: Guided Diffusion



$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$

Shift to a dog label!

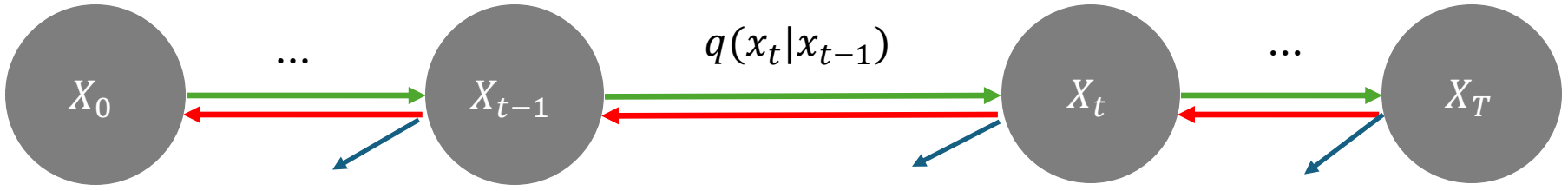
In sampling: $\epsilon_\theta(x_t, t) + \nabla_x \log p_\phi(y|x)$



$p_\phi(y|x)$

A Dog

Control the Diffusion Model: Guided Diffusion

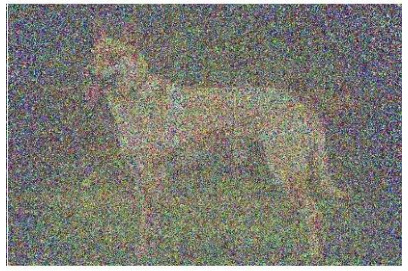


$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$

Shift to a dog label!

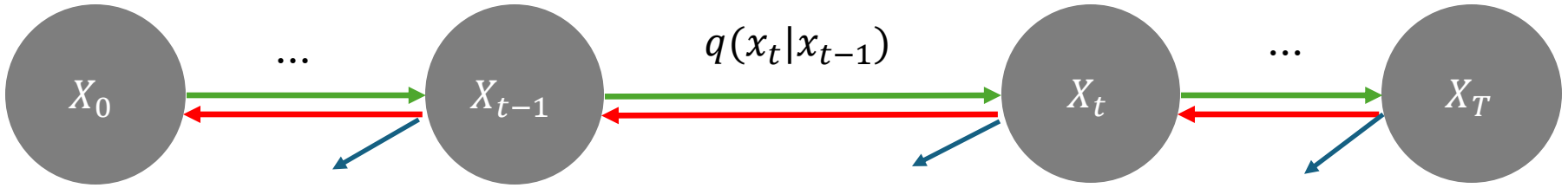
In sampling: $\epsilon_\theta(x_t, t) + \nabla_x \log p_\phi(y|x)$. ← **Guided Diffusion**

We need to train a classifier: $p_\phi(y|x)$, with the awareness of noise



$p_\phi(y|x)$ → A Dog

Control the Diffusion Model: Guided Diffusion

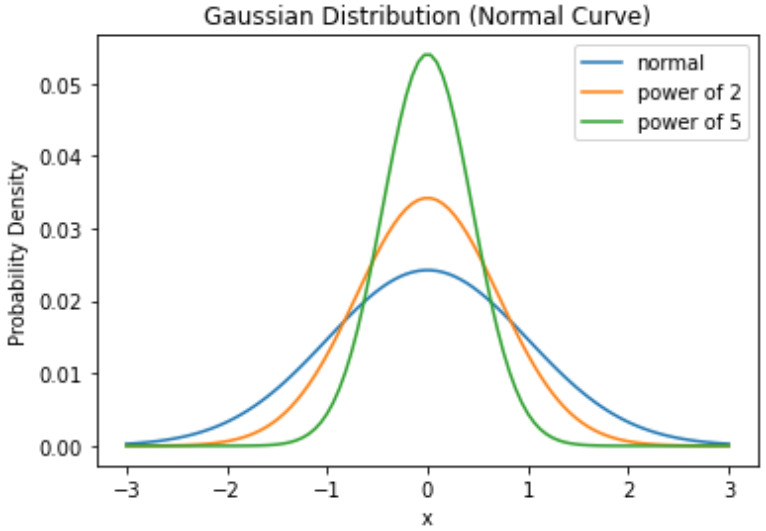


$$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$$

Shift to a dog label!

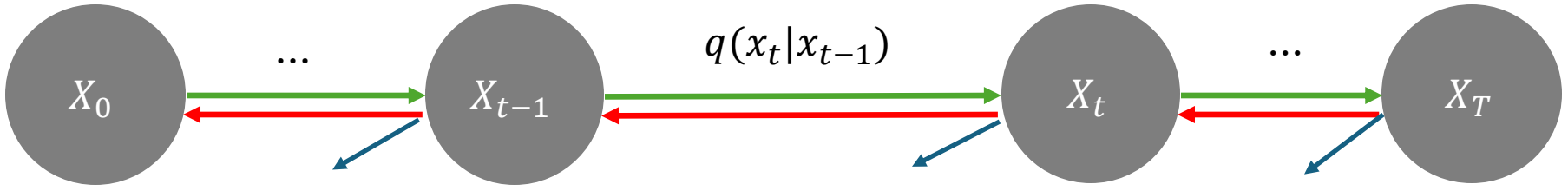
In sampling: $\epsilon_\theta(x_t, t) + \gamma\nabla_x \log p_\phi(y|x)$. ← **Guided Diffusion**

$$\gamma\nabla_x \log p_\phi(y|x) \sim \nabla_x \log p_\phi(y|x)^\gamma$$



Dhariwal and Nichol, 2021

Control the Diffusion Model: Guided Diffusion



$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$

Shift to a dog label!

In sampling: $\epsilon_\theta(x_t, t) + \gamma\nabla_x \log p_\phi(y|x)$. ← **Guided Diffusion**

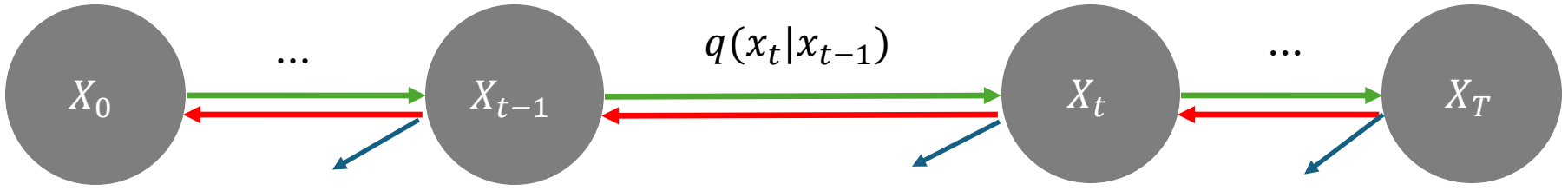
Label: Corgi

γ = 1

γ = 3

Dhariwal and Nichol, 2021

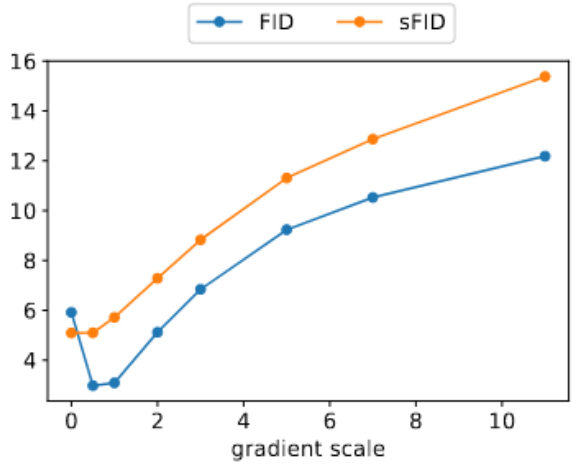
Control the Diffusion Model: Guided Diffusion



$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$

Shift to a dog label!

In sampling: $\epsilon_\theta(x_t, t) + \gamma\nabla_x \log p_\phi(y|x)$. ← **Guided Diffusion**



Dhariwal and Nichol, 2021

Guided Diffusion: Nearest Neighbors for Samples

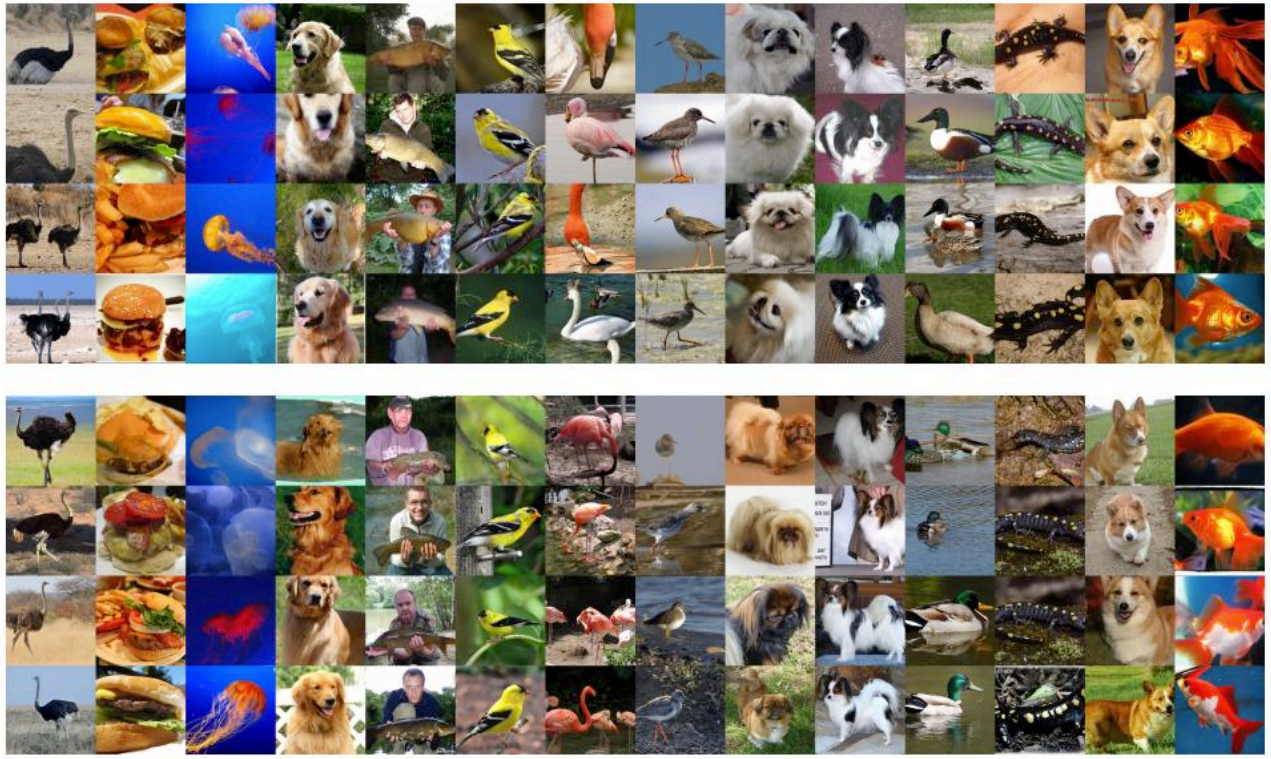


Figure 7: Nearest neighbors for samples from a classifier guided model on ImageNet 256×256. For each image, the top row is a sample, and the remaining rows are the top 3 nearest neighbors from the dataset. The top samples were generated with classifier scale 1 and 250 diffusion sampling steps (FID 4.59). The bottom samples were generated with classifier scale 2.5 and 25 DDIM steps (FID 5.44).

Guided Diffusion: Effect of Varying the Classifier Scale

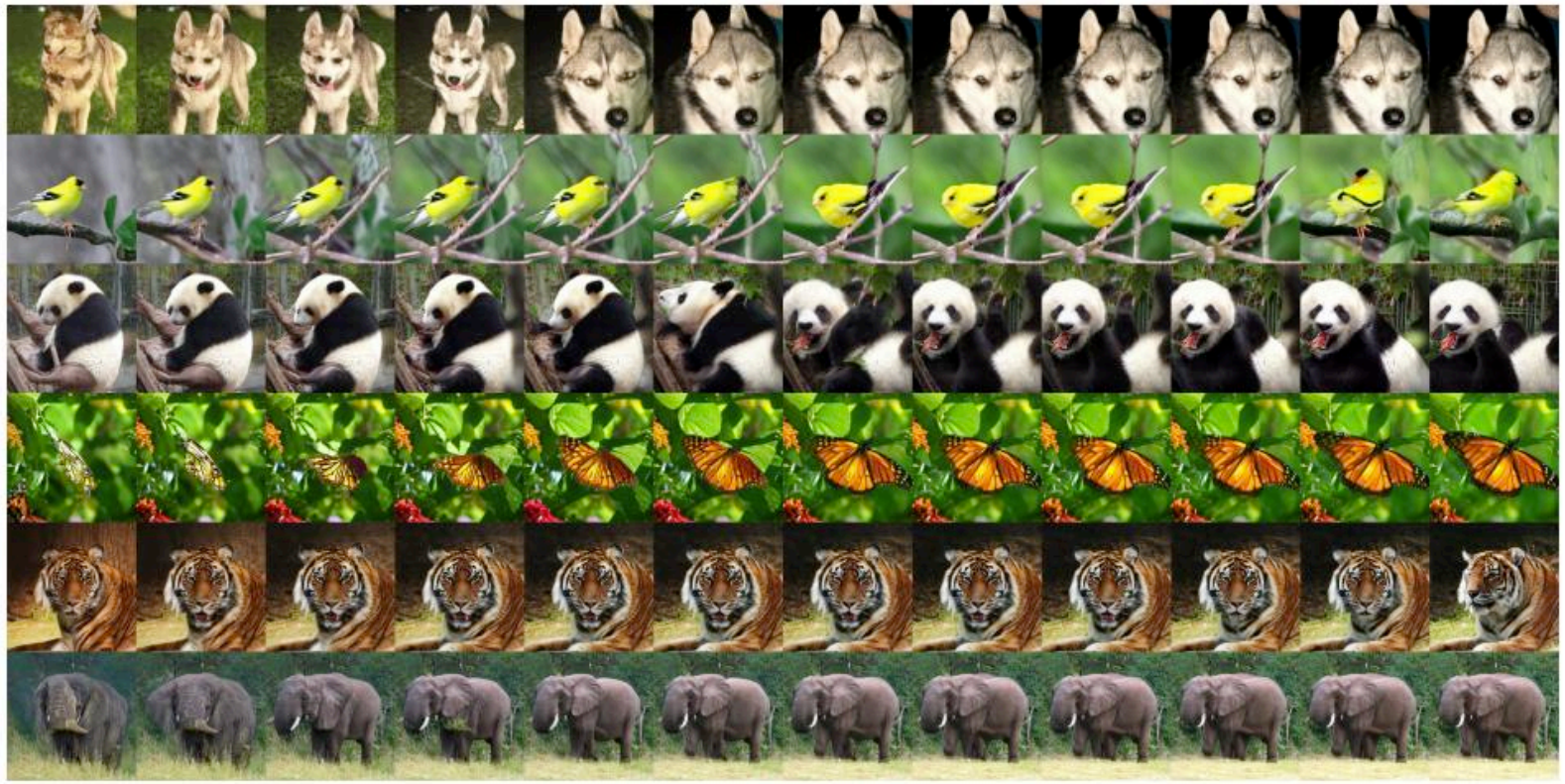


Figure 8: Samples when increasing the classifier scale from 0.0 (left) to 5.5 (right). Each row corresponds to a fixed noise seed. We observe that the classifier drastically changes some images, while leaving others relatively unaffected.

Guided Diffusion: Examples

Figure 13: Samples from our best 512×512 model (FID: 3.85). Classes are 1: goldfish, 279: arctic fox, 323: monarch butterfly, 386: african elephant, 130: flamingo, 852: tennis ball.

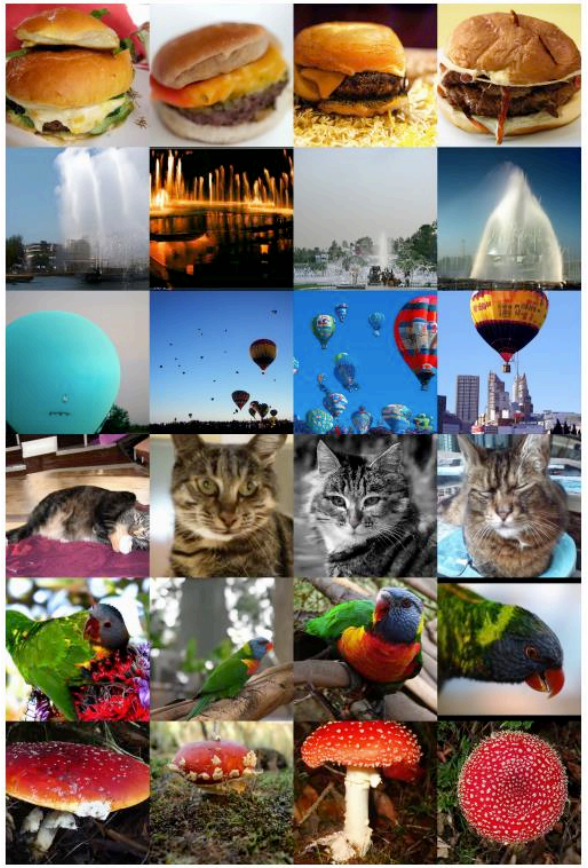
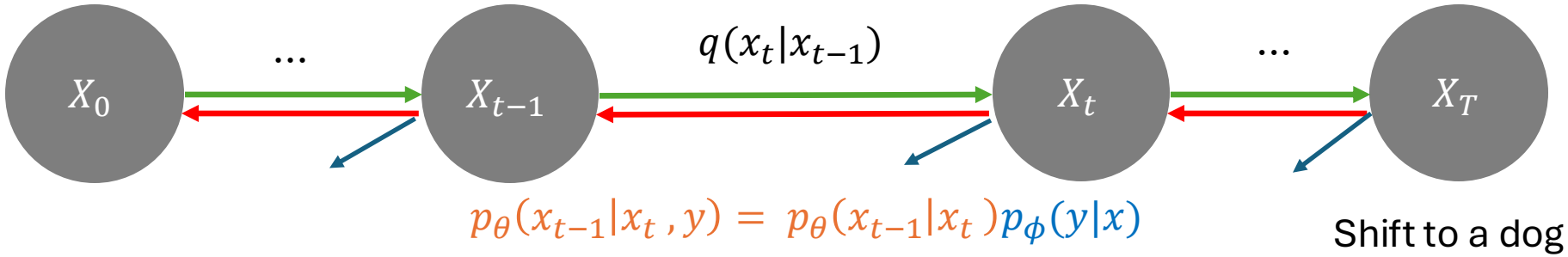


Figure 14: Samples from our best 512×512 model (FID: 3.85). Classes are 933: cheeseburger, 562: fountain, 417: balloon, 281: tabby cat, 90: lorikeet, 992: agaric.

Why not guided diffusion?

Control the Diffusion Model: Guided Diffusion



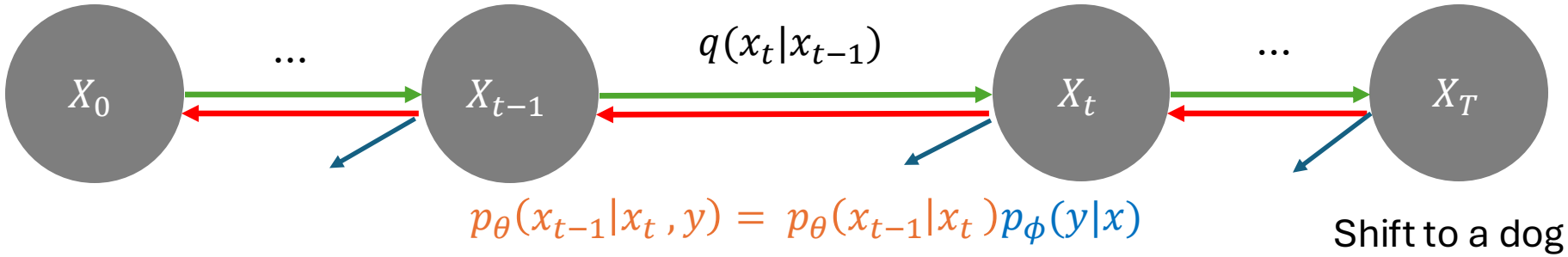
In sampling: $\epsilon_\theta(x_t, t) + \gamma \nabla_x \log p_\phi(y|x)$. ← **Guided Diffusion**

What do we NOT like in guided diffusion?

,

Dhariwal and Nichol, 2021

Control the Diffusion Model: Guided Diffusion



In sampling: $\epsilon_\theta(x_t, t) + \boldsymbol{\gamma}\nabla_x \log p_\phi(y|x)$. ← **Guided Diffusion**

What do we NOT like in guided diffusion?

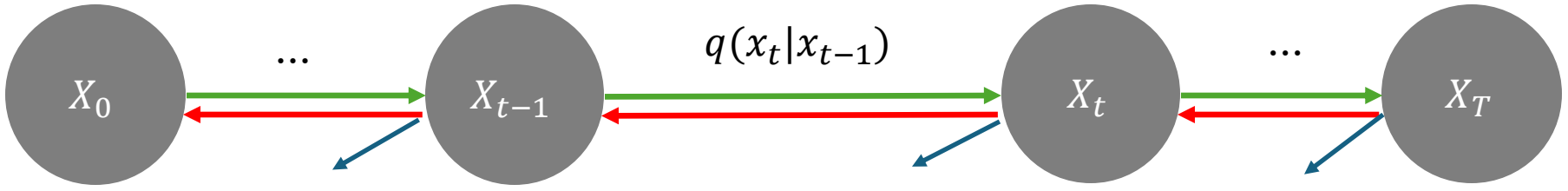
- Need to fine-tune and train a classifier
- Condition can only be label-based, hard to support other conditions like “text input”

Because for text, the classifier $p_\phi(y|x)$ ***does not*** exist.

Dhariwal and Nichol, 2021

Classifier-free guidance

Control the Diffusion Model: Classifier-Free Guidance



$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$

Shift to a dog label!

At training: $p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$

In sampling: $\hat{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + \gamma \nabla_x \log p(y|x)$

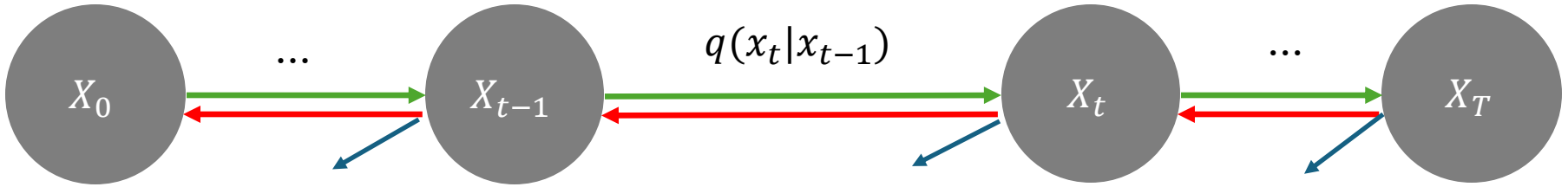
$$p(y|x) \propto \frac{p(x|y)}{p(x)}$$

$$\nabla_x \log p(y|x) \propto \nabla_x \log p(x|y) - \nabla_x \log p(x)$$

Thanks to Bayes

Ho and Salimans, 2022

Control the Diffusion Model: Classifier-Free Guidance



$$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$$

Shift to a dog label!

At training: $p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$

In sampling: $\hat{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + \gamma \nabla_x \log p(y|x)$

$$\nabla_x \log p(y|x) \propto \epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t)$$

Finally: $\hat{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + \gamma(\underbrace{\epsilon_\theta(x_t, t, y)}_{\text{conditional generation}} - \underbrace{\epsilon_\theta(x_t, t)}_{\text{unconditional generation}})$

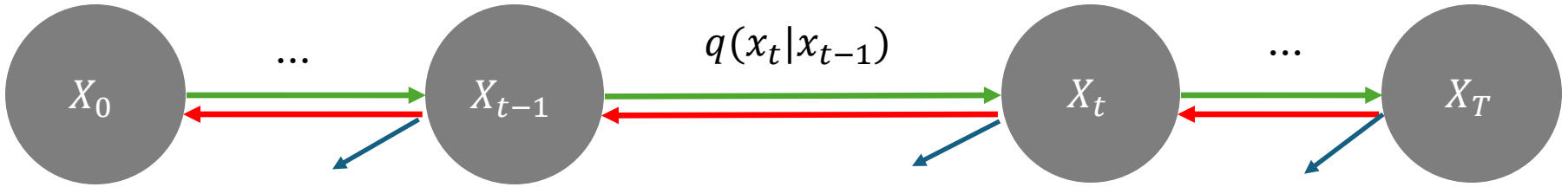
$$p(y|x) \propto \frac{p(x|y)}{p(x)}$$

$$\nabla_x \log p(y|x) \propto \nabla_x \log p(x|y) - \nabla_x \log p(x)$$

Thanks to Bayes

Ho and Salimans, 2022

Control the Diffusion Model: Classifier-Free Guidance



$$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)\, p_\phi(y|x)$$

Shift to a dog label!

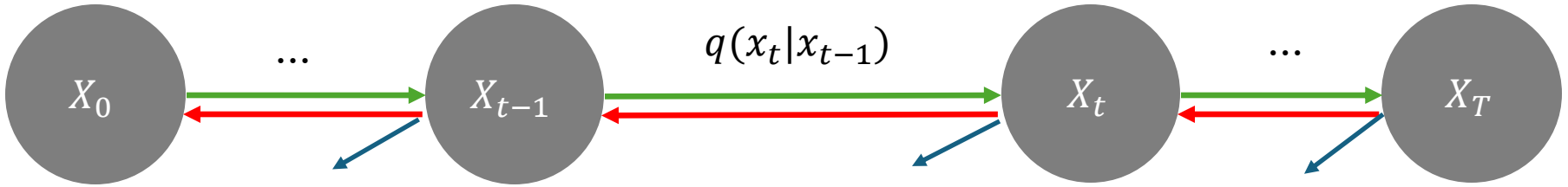
$$\hat{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + \gamma(\underbrace{\epsilon_\theta(x_t, t, y)}_{\text{conditional generation}} - \underbrace{\epsilon_\theta(x_t, t)}_{\text{unconditional generation}})$$

How to compute: $\epsilon_\theta(x_t, t, y)$ →
- explicit condition

How to compute: $\epsilon_\theta(x_t, t)$ →

Ho and Salimans, 2022

Control the Diffusion Model: Classifier-Free Guidance



$$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$$

Shift to a dog label!

$$\hat{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + \gamma(\underbrace{\epsilon_\theta(x_t, t, y)}_{\text{conditional generation}} - \underbrace{\epsilon_\theta(x_t, t)}_{\text{unconditional generation}})$$

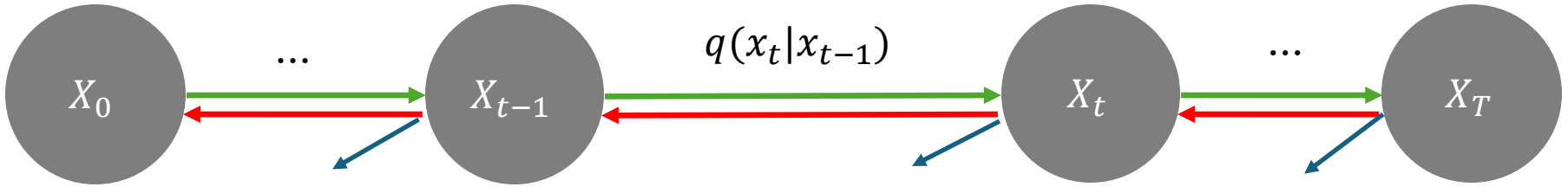
Implicit classifier

How to compute: $\epsilon_\theta(x_t, t, y)$ →
- explicit condition

How to compute: $\epsilon_\theta(x_t, t)$ →
- We randomly set the condition to null (drop-out condition)
- $\epsilon_\theta(x_t, t, y)$ → $\epsilon_\theta(x_t, t, \emptyset)$

Ho and Salimans, 2022

Control the Diffusion Model: Classifier-Free Guidance



$$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$$

Shift to a dog label!

$$\hat{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + \gamma(\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t))$$

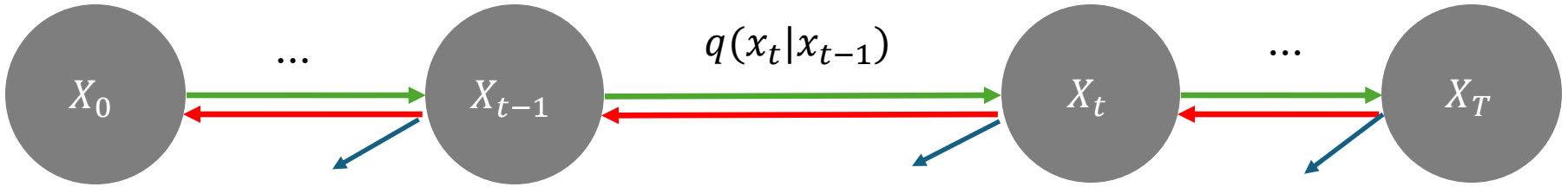
Label: Husky

γ=1

γ=3

Ho and Salimans, 2022

Control the Diffusion Model: Classifier-Free Guidance



$$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$$

Shift to a dog label!

$$\hat{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + \gamma(\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t))$$

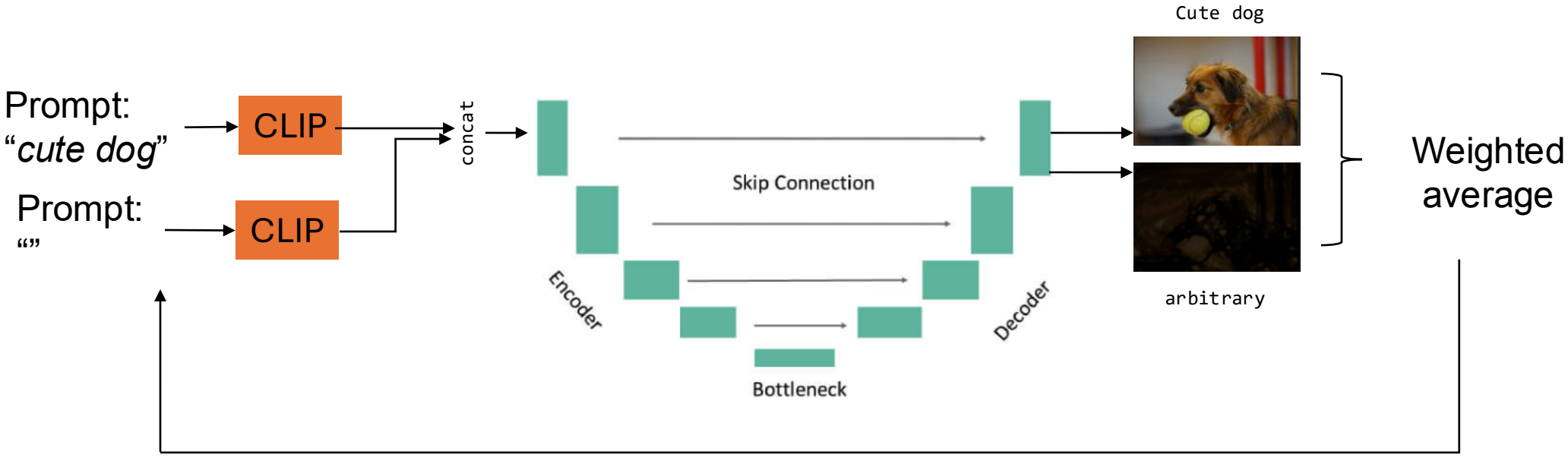
We do not need the explicit classifier: we can use text-encoder to condition on text.
Called : Classifier-Free Guidance (CFG)

"A panda is eating ice-cream" → CLIP/T5 → 768x1 dim

Condition Latent

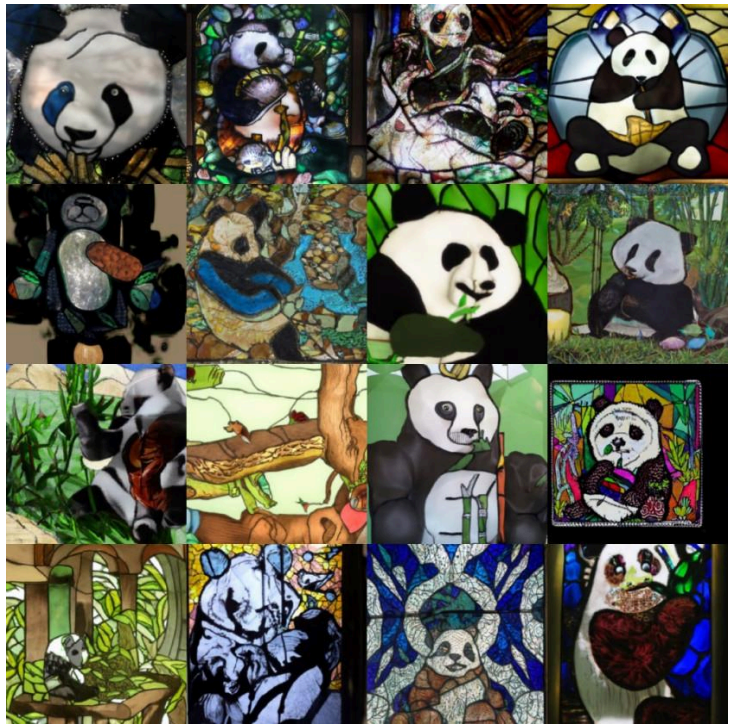
Ho and Salimans, 2022

Classifier free guidance



Classifier free guidance

Control the Diffusion Model: Classifier-Free Guidance



$\gamma = 1$

$\gamma = 3$

Caption: "*A stained glass window of a panda eating bamboo.*"

Control the Diffusion Model: Classifier-Free Guidance

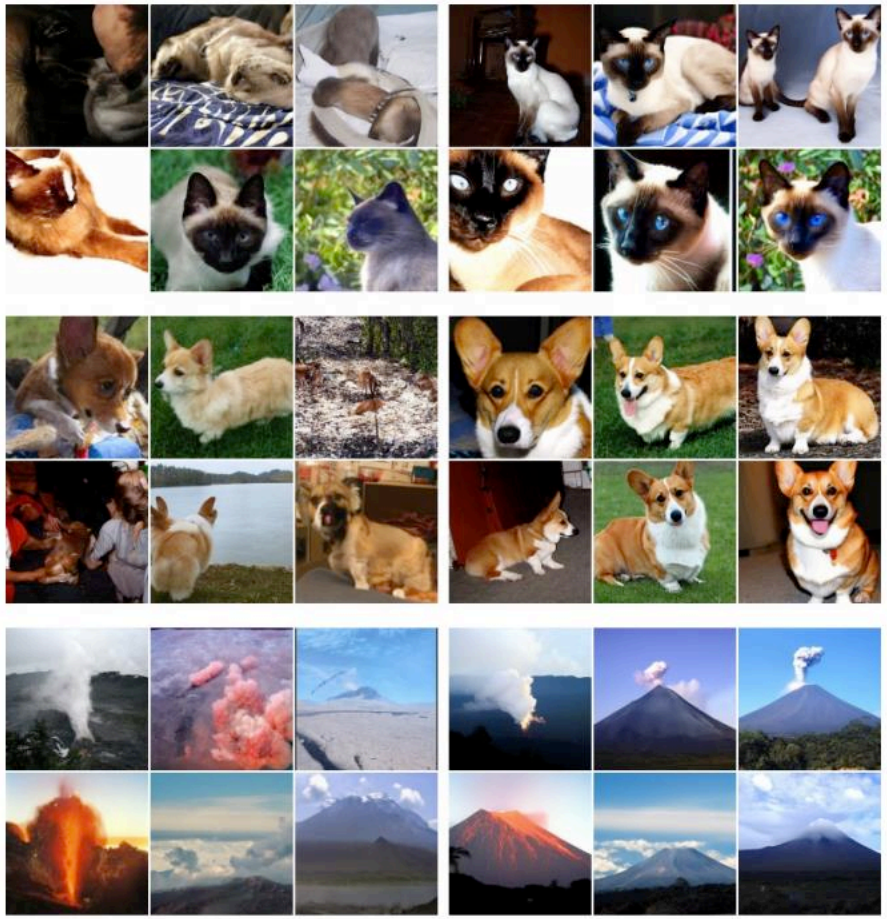


Figure 3: Classifier-free guidance on 128x128 ImageNet. Left: non-guided samples, right: classifier-free guided samples with $w = 3.0$. Interestingly, strongly guided samples such as these display saturated colors. See Fig. 8 for more.

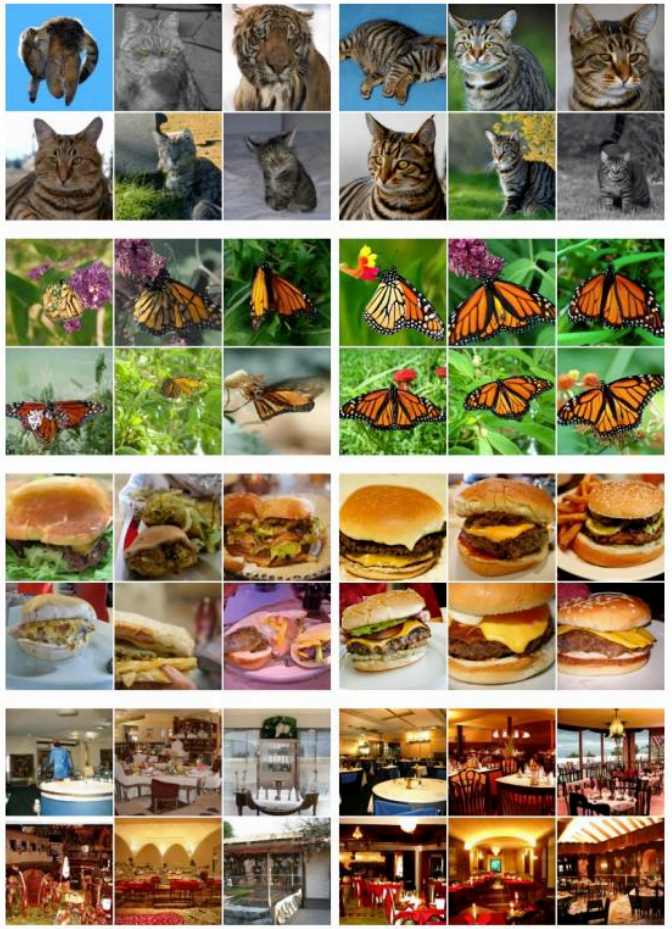
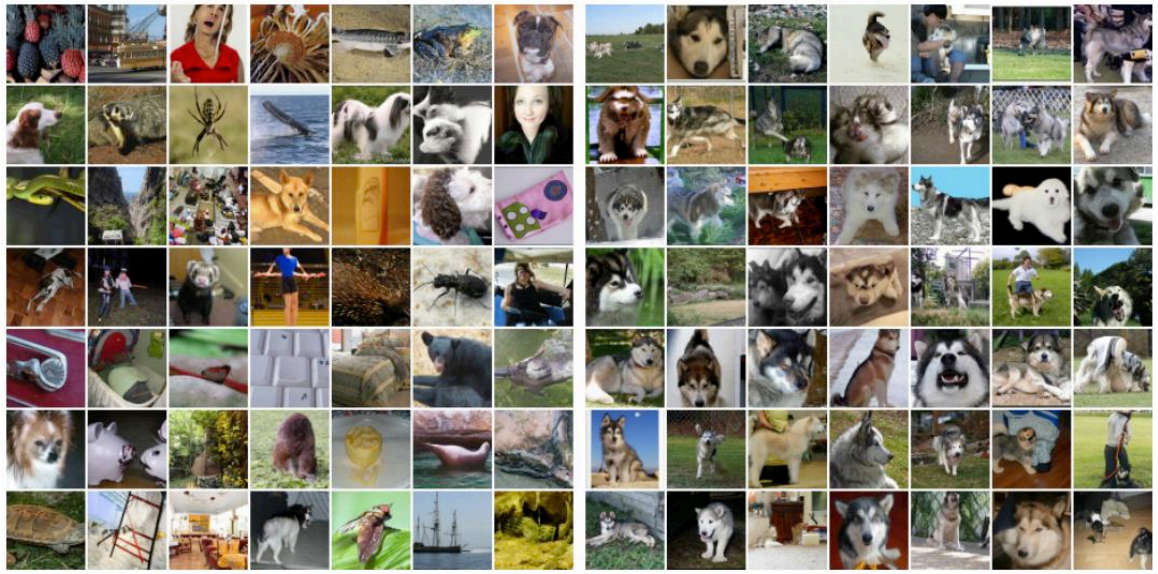
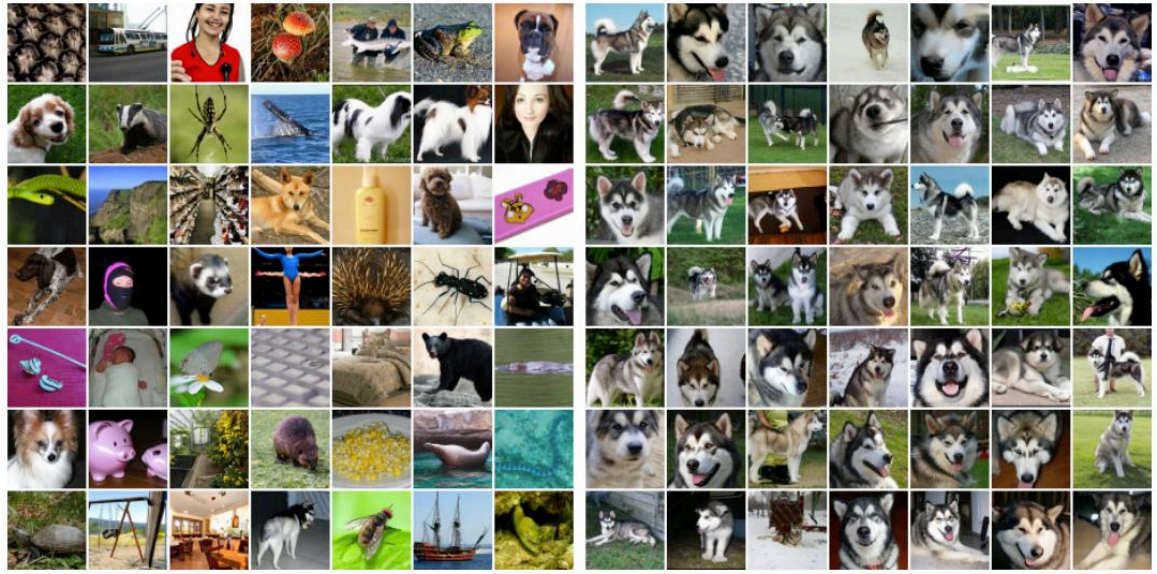


Figure 8: More examples of classifier-free guidance on 128x128 ImageNet. Left: non-guided samples, right: classifier-free guided samples with $w = 3.0$.

Control the Diffusion Model: Classifier-Free Guidance

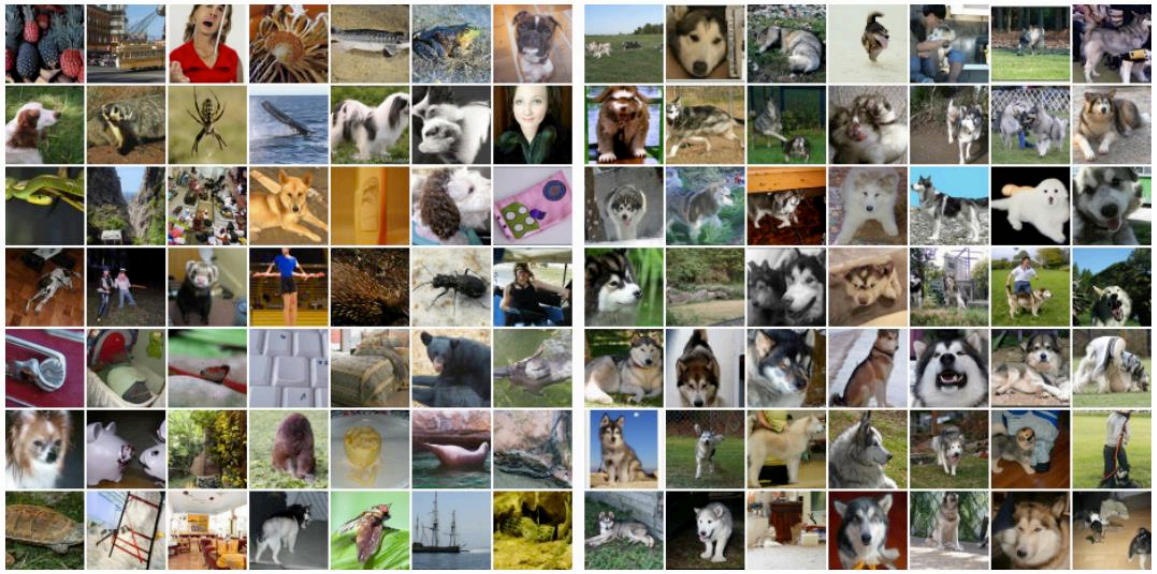


(a) Non-guided conditional sampling: FID=1.80, IS=53.71

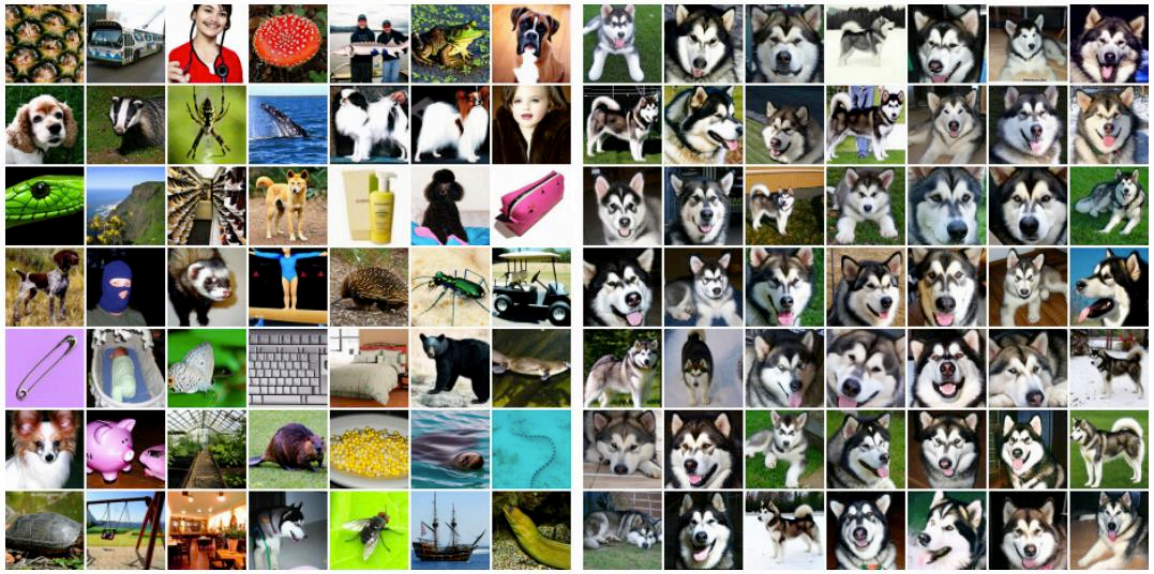


(b) Classifier-free guidance with $w = 1.0$: FID=12.6, IS=170.1

Control the Diffusion Model: Classifier-Free Guidance



(a) Non-guided conditional sampling: FID=1.80, IS=53.71



(c) Classifier-free guidance with $w = 3.0$: FID=24.83, IS=250.4

Negative prompting

Control the Diffusion Model: Classifier-Free Guidance

$$\hat{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + \gamma(\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t))$$

Negative prompting:

$$\hat{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + \underbrace{\gamma(\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t))}_{\text{positive CFG}} \underbrace{- \gamma(\epsilon_\theta(x_t, t, y_{neg}) - \epsilon_\theta(x_t, t))}_{\text{negative CFG}}$$

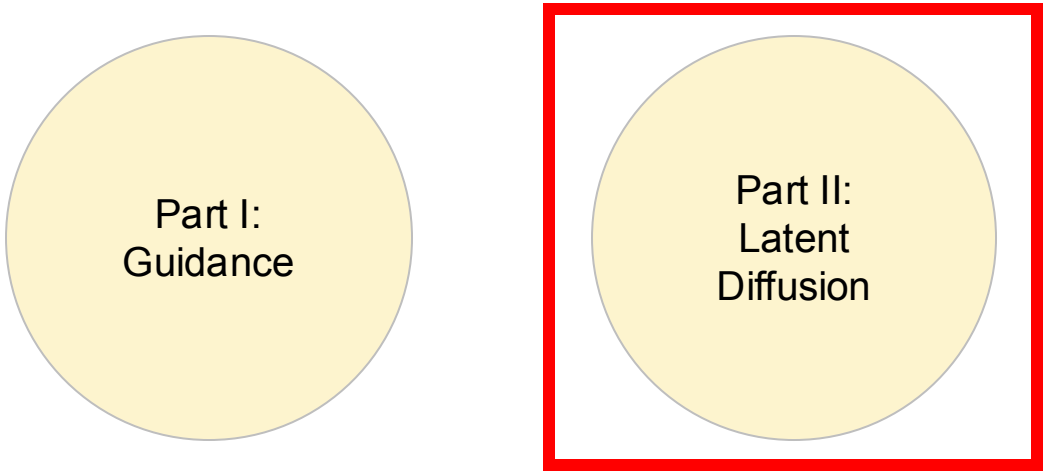
Control the Diffusion Model: Classifier-Free Guidance

w/o negative prompting

w/ negative prompting:
Disfigured, cartoon, blurry, nude

Control the Diffusion Model: Classifier-Free Guidance

Today's lecture



Slides adapted from many resources:
[Xi Wang, Fei-Fei Li & Andrej Karpathy & Justin Johnson & Ross Girshick & VGG]

Latent Diffusion Model

What don't we like in Denoising Diffusion Probabilistic Model (DDPM)?

- Training models in the pixel space is excessively computationally expensive — it can easily take multiple GPU days (measured in terms of a V100 GPU)
  - Even image synthesis at inference time is very slow compared to GANs because of the iterative nature of diffusion models
  - Image are high-dimensional → high-dimensional is hard

Latent Diffusion Model

What don't we like in Denoising Diffusion Probabilistic Model (DDPM)?

- Training models in the pixel space is excessively computationally expensive — it can easily take multiple GPU days (measured in terms of a V100 GPU)
  - Even image synthesis at inference time is very slow compared to GANs because of the iterative nature of diffusion models
  - Image are high-dimensional → high-dimensional is hard
- Researchers observed that most "bits" of an image contribute only to its perceptual characteristics (i.e., how the image looks) compared to semantic and conceptual composition
  - In layman's terms, there are more "bits" for describing pixel-level details while less "bits" are sufficient for describing the "meaning" of an image
  - Generative models should ideally focus on the latter more
- Can we separate the two components?

Latent Diffusion Model

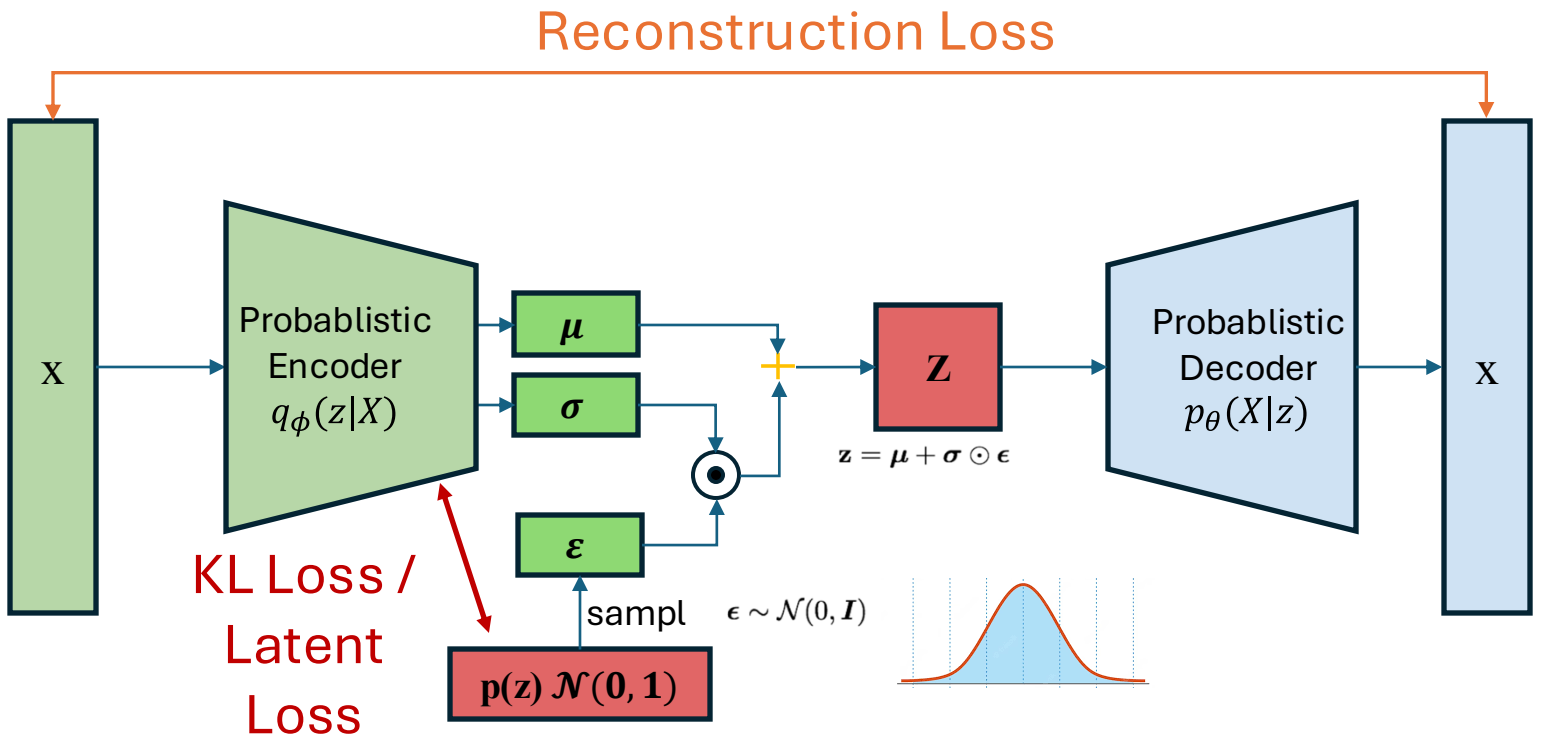
- Train a compression model that strips away irrelevant high-level pixel-space details from an image and encodes it into a semantically equivalent low-dimensional latent

  - Also need a way to convert back from the latent space to the pixel space — naturally, we want an encoder-decoder architecture
  - Any idea? VAE/VQVAE/VQGAN !!!

Latent Diffusion Model

- Train a compression model that strips away irrelevant high-level pixel-space details from an image and encodes it into a semantically equivalent low-dimensional latent
  - Also need a way to convert back from the latent space to the pixel space — naturally, we want an encoder-decoder architecture
  - Any idea? VAE/VQVAE/VQGAN !!!
- Perform the diffusion process in this latent space. There are several benefits to this:
  - The diffusion process is only focusing on the relevant semantic bits of the data
  - Performing diffusion in a low-dimensional space is significantly faster

Variational AutoEncoder (VAE)



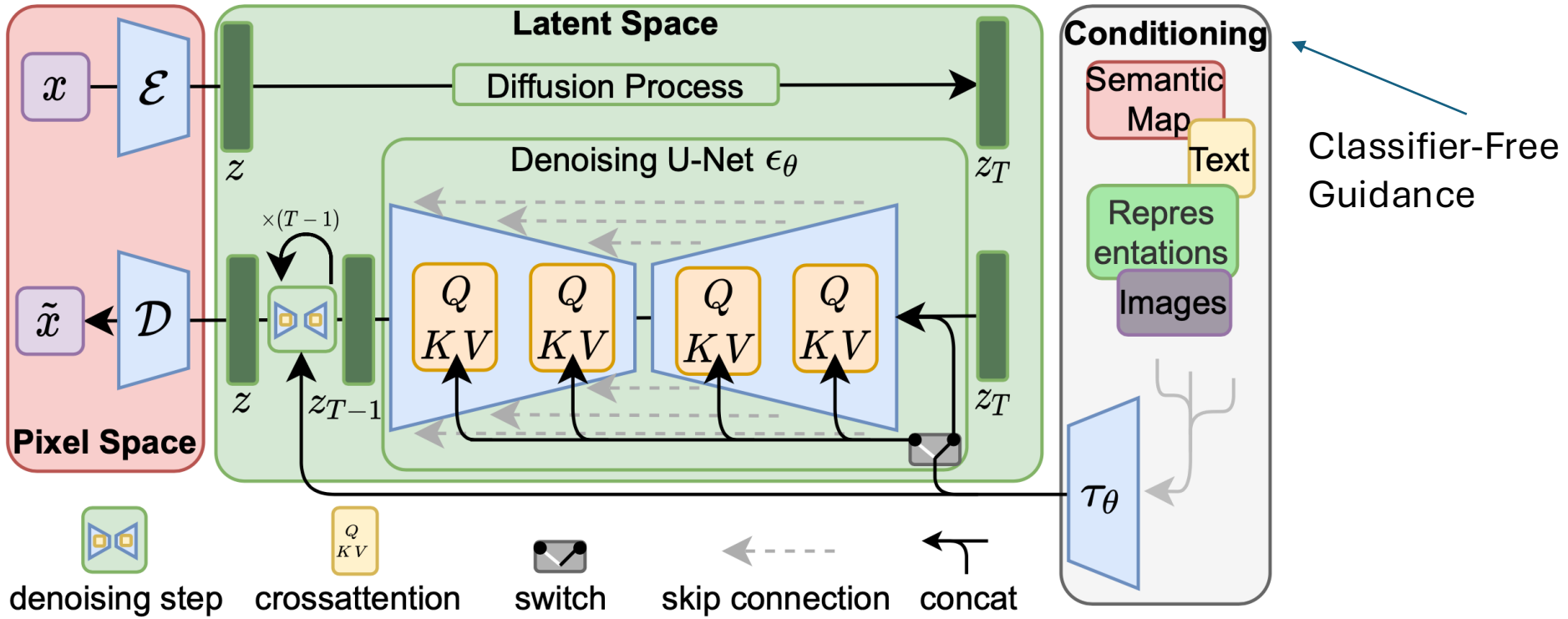
$$L_{\text{VAE}}(\theta, \phi) = -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

θ: Parameter of decoder

ϕ: Parameter of encoder

Kingma & Welling, 2014

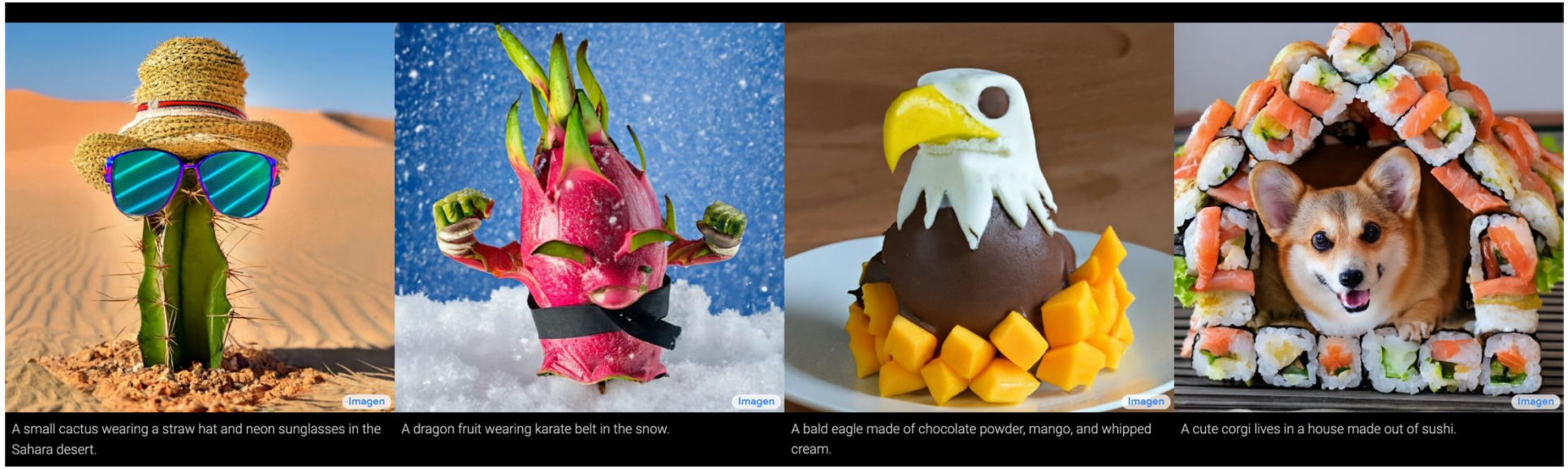
Latent Diffusion Model



Instead of denoising on pixels, we could denoise on latent space with could be constructed from a VAE/VQGAN.

Latent Diffusion Model

Imagen by Google AI



https://imagen.research.google/

Make-A-Video (Text-to-Video)

A confused grizzly bear in a calculus class

A golden retriever eating ice cream on a beautiful tropical beach at sunset, high resolution

A panda playing on a swing set

Thank you