Trustworthy AI Systems

-- Generative Modeling (Part II)

Instructor: Guangjing Wang

guangjingwang@usf.edu

Last Lecture

- Generative Adversarial Network
    - DCGAN
    - Conditional GAN
    - CycleGAN

- Neural Style Transfer

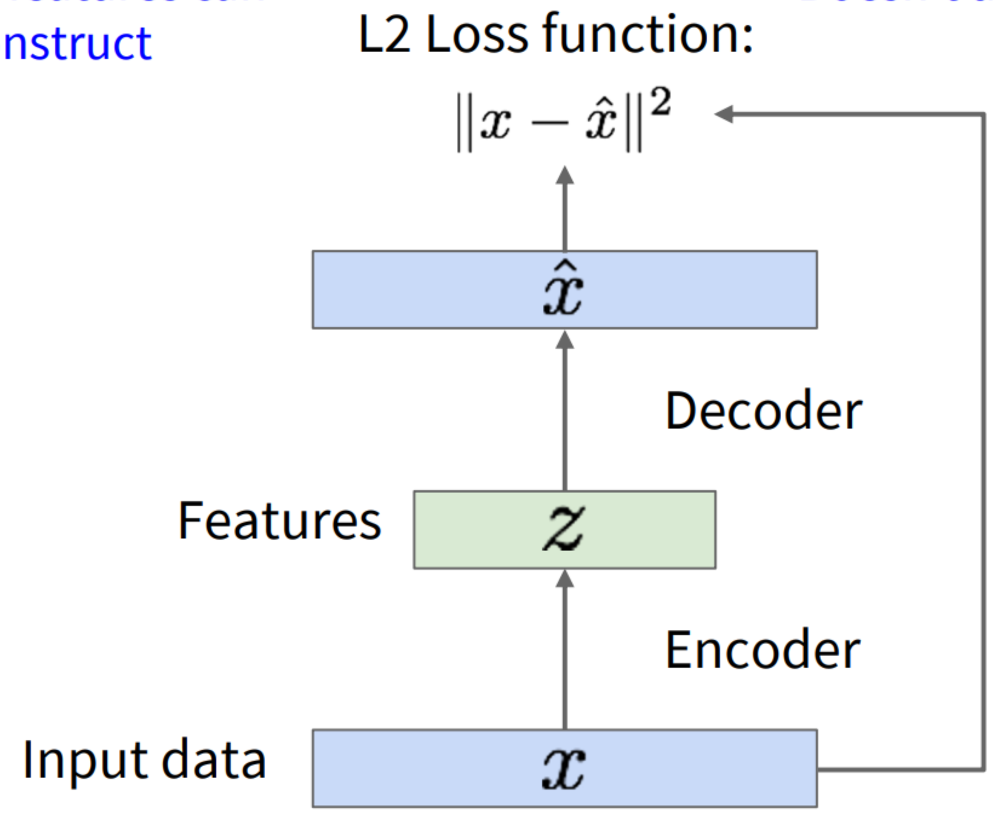
This Lecture

- Variational Autoencoders
- Diffusion Models

Autoencoder

Train such that features can be used to reconstruct original data

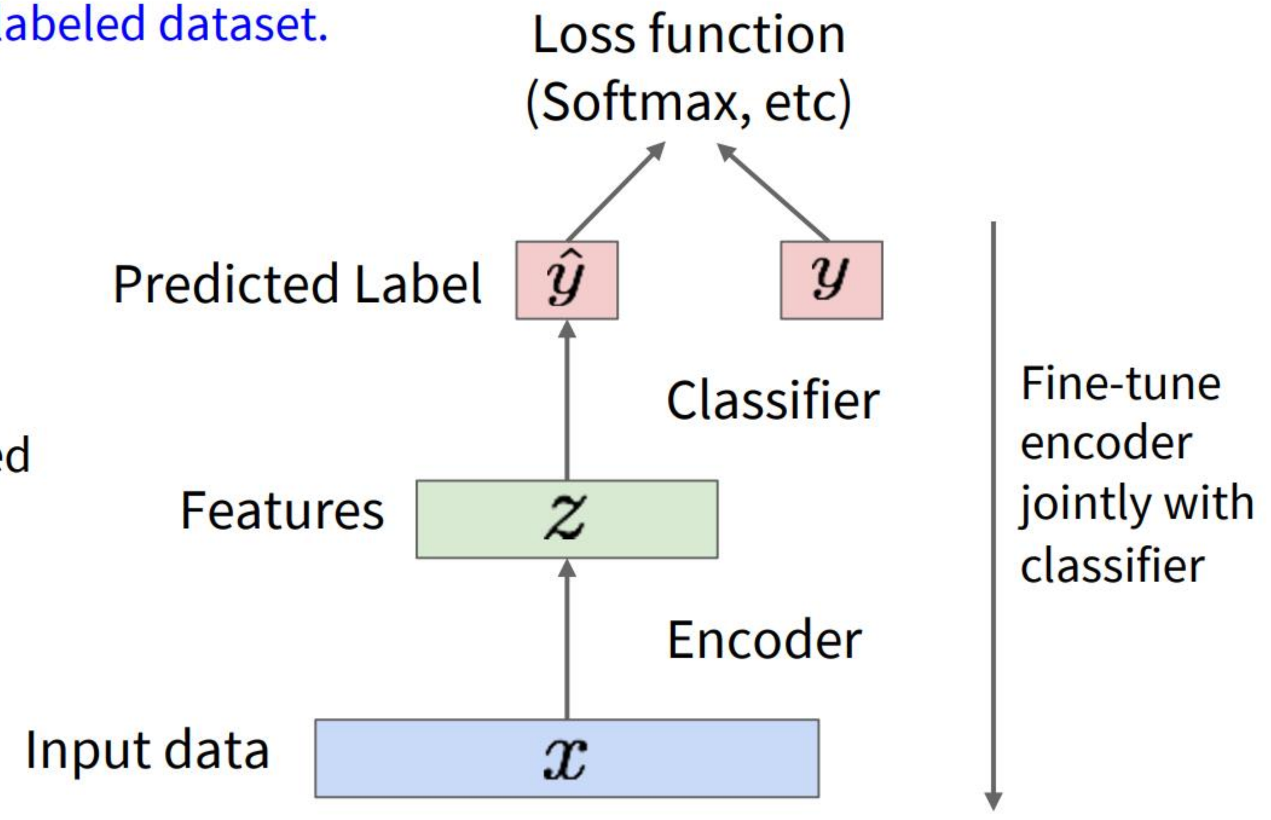
Doesn't use labels!



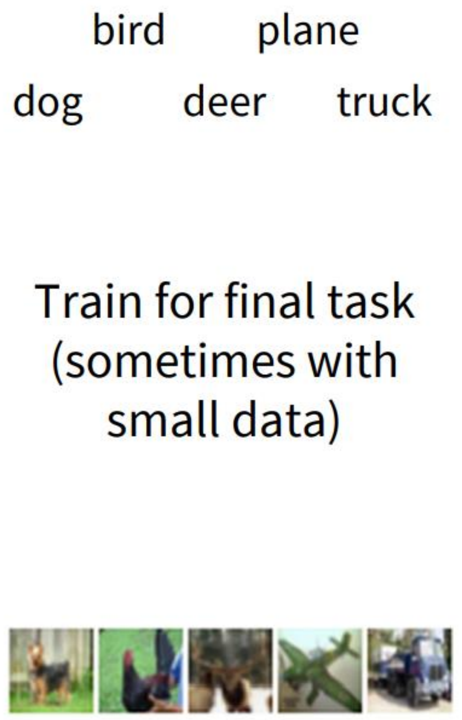
Application of Autoencoder

Transfer from large, unlabeled dataset to small, labeled dataset.

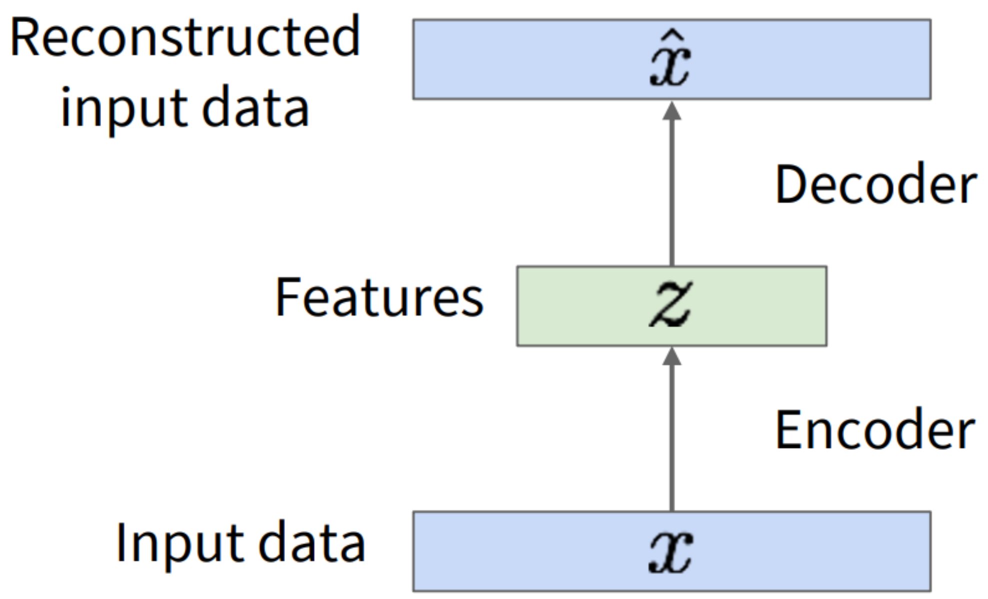
Encoder can be used to initialize a supervised model


Loss function (Softmax, etc)
Predicted Label
$\hat{y}$
y
Classifier
Features
z
Encoder
Input data
x
Fine-tune encoder jointly with classifier

Train for final task (sometimes with small data)



The Limitation of Autoencoder



Autoencoders can reconstruct data, and can learn features to initialize a supervised model

Features capture factors of variation in training data.

But we can’t generate new images from an autoencoder because we don’t know the space of z.

How do we make autoencoder a generative model?

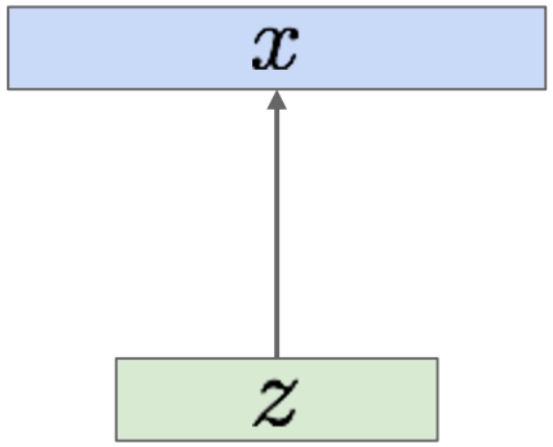
Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from the distribution of unobserved (latent) representation z

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $z^{(i)} \sim p_{\theta^*}(z)$

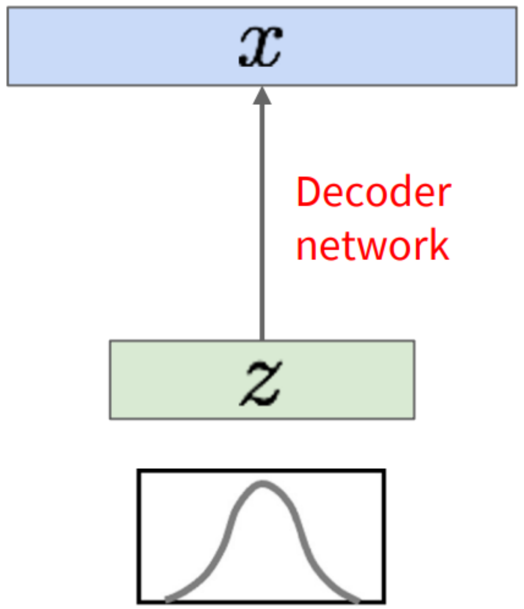


Intuition (remember from autoencoders!): x is an image, z is latent factors used to generate x: attributes, orientation, etc.

Variational Autoencoders

Sample from true conditional

$p_{\theta^*}(x \mid z^{(i)})$



Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$

We want to estimate the true parameters θ^* of this generative model given training data x.

How should we represent this model?

Choose prior p(z) to be simple, e.g. Gaussian. Reasonable for latent attributes, e.g. pose, how much smile.

Conditional p(x|z) is complex (generates image) => represent with neural network

Variational Autoencoders: Intractability

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

$\uparrow$

Intractable to compute p(x|z) for every z!

$$\log p(x) \approx \log \frac{1}{k} \sum_{i=1}^{k} p(x|z^{(i)}), \text{ where } z^{(i)} \sim p(z)$$

Monte Carlo estimation is too high variance

KL measures how one probability distribution *P* is different from a second, Q.

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right).$$

Variational Autoencoders

Math trick: Taking expectation wrt. z (using encoder network)

We want to maximize the data likelihood

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z \sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z) p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z) p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

$\uparrow$ Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling (need some trick to differentiate through sampling).

$\uparrow$ This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$\uparrow$ $p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

Variational Autoencoders

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)}\mid z)\,p_\theta(z)}{p_\theta(z\mid x^{(i)})}\right] && (\text{Bayes' Rule})\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)}\mid z)\,p_\theta(z)}{p_\theta(z\mid x^{(i)})}\frac{q_\phi(z\mid x^{(i)})}{q_\phi(z\mid x^{(i)})}\right] && (\text{Multiply by constant})\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)}\mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z\mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z\mid x^{(i)})}{p_\theta(z\mid x^{(i)})}\right] && (\text{Logarithms})\\
&= \boxed{\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)}\mid z)\right] - D_{KL}(q_\phi(z\mid x^{(i)})\,||\,p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)}} + \underbrace{D_{KL}(q_\phi(z\mid x^{(i)})\,||\,p_\theta(z\mid x^{(i)}))}_{\boxed{\geq 0}}
\end{aligned}$$

Decoder:
reconstruct
the input data

Encoder:
make approximate
posterior distribution
close to prior

Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term differentiable)

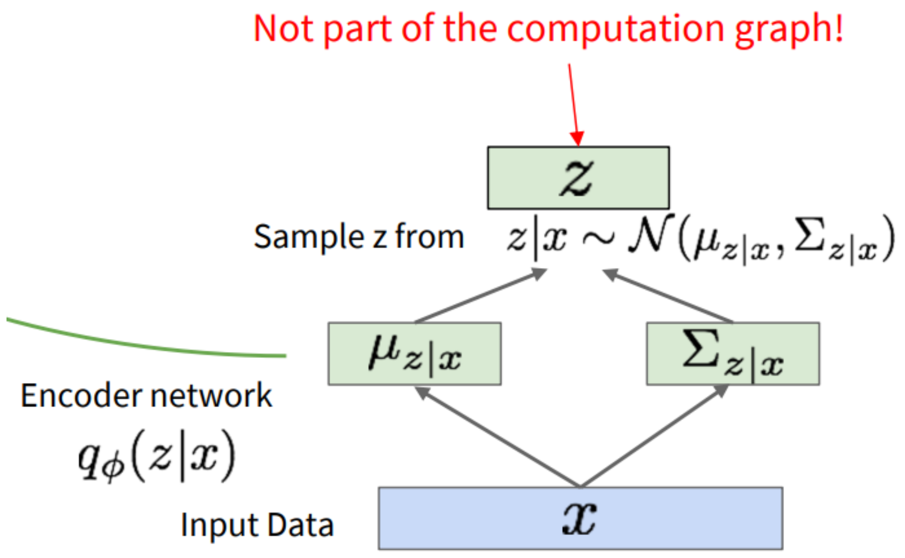
Reparameterization in VAE

- Generate NEW: Sampling is required to model the probabilistic nature of latent space.
- This sampling operation introduce stochasticity and therefore cannot be differentiated.
- Backpropagation relies on computing gradients of deterministic (i.e., non-random) operations.

Reparameterization trick to make sampling differentiable:

Sample $\epsilon \sim \mathcal{N}(0, I)$

$$z = \mu_{z|x} + \epsilon\sigma_{z|x}$$


Not part of the computation graph!
z
Sample z from $z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$
$\mu_{z|x}$
$\Sigma_{z|x}$
Encoder network
$q_\phi(z|x)$
Input Data
x

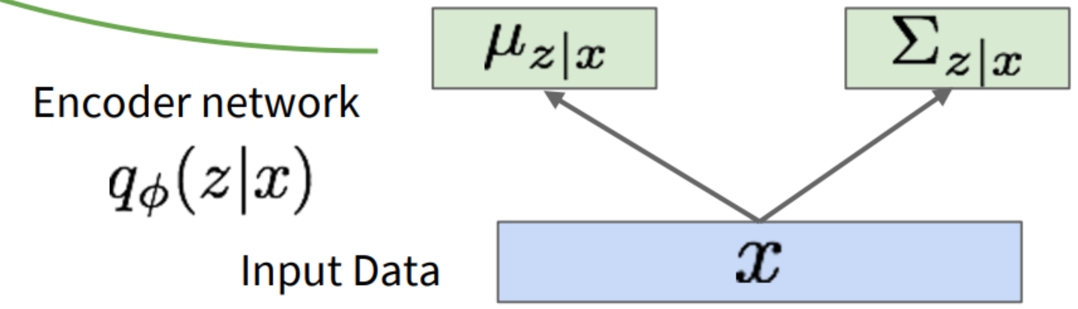
Variational Autoencoders

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

$$D_{KL}(\mathcal{N}(\mu_{z|x}, \Sigma_{z|x})||\mathcal{N}(0, I))$$

Have analytical solution

Make approximate posterior distribution close to prior

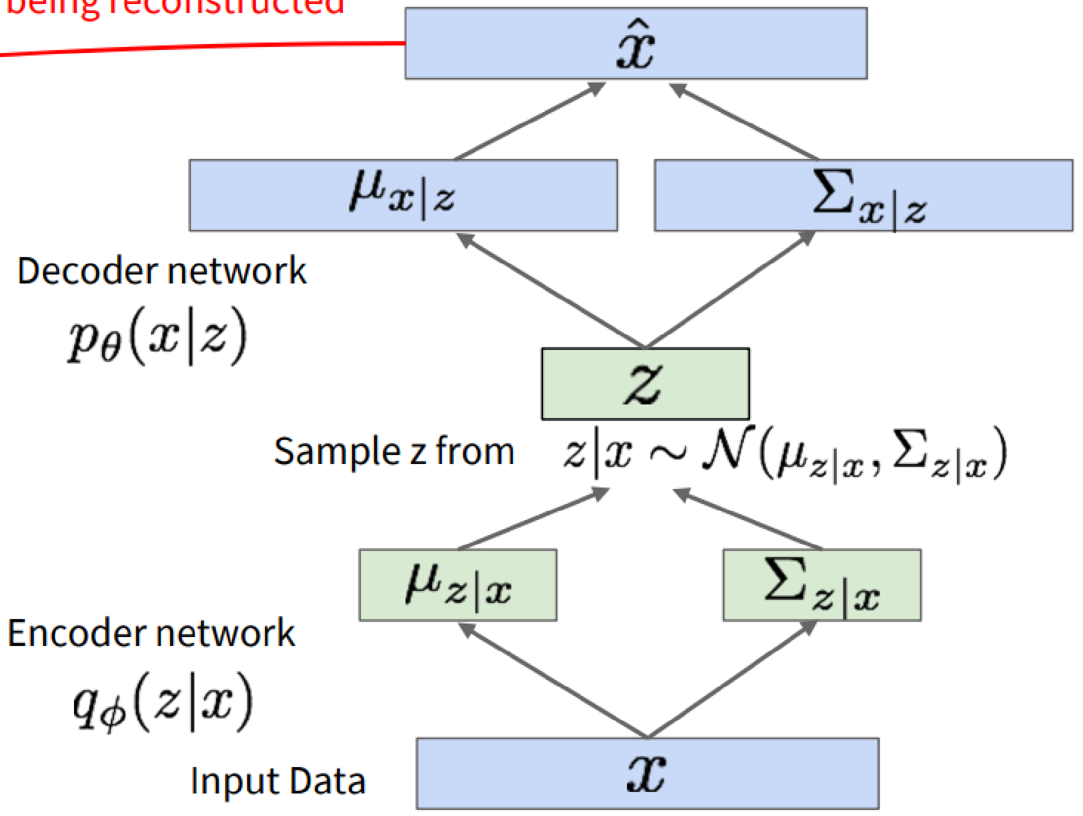


Variational Autoencoders

Maximize likelihood of original input being reconstructed

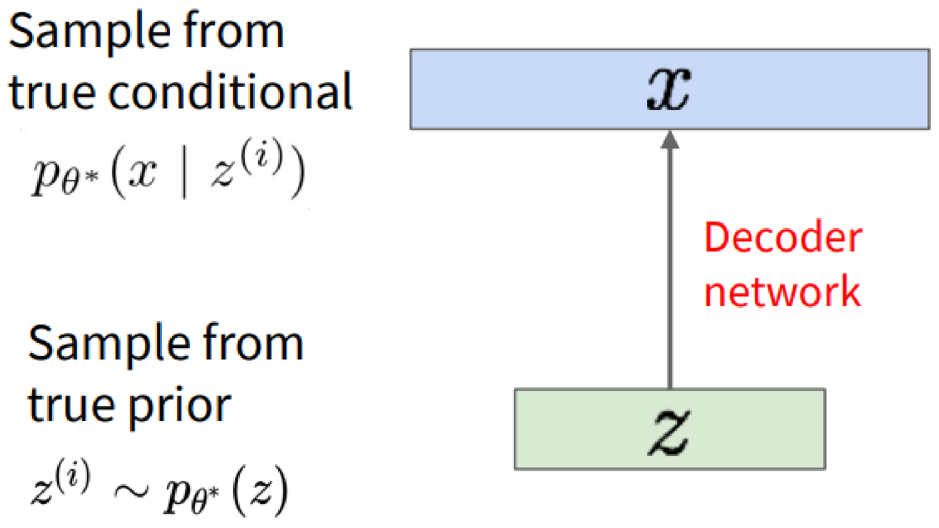
Putting it all together: maximizing the likelihood lower bound

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

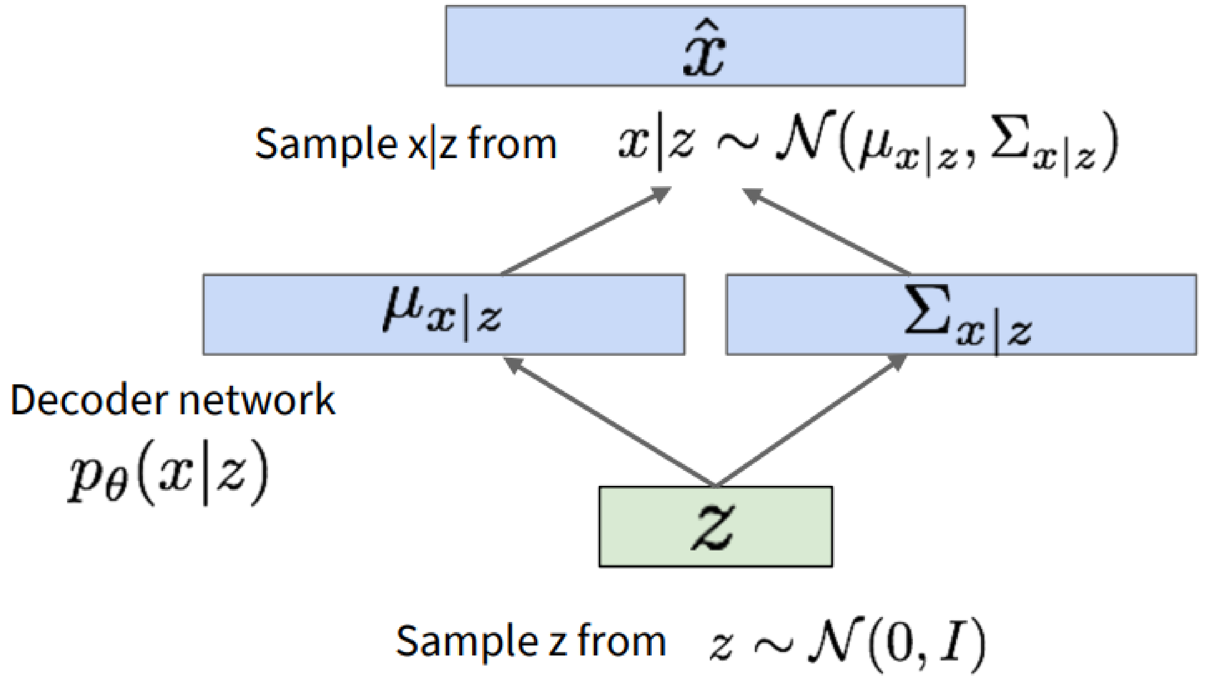


Variational Autoencoders: Generating Data

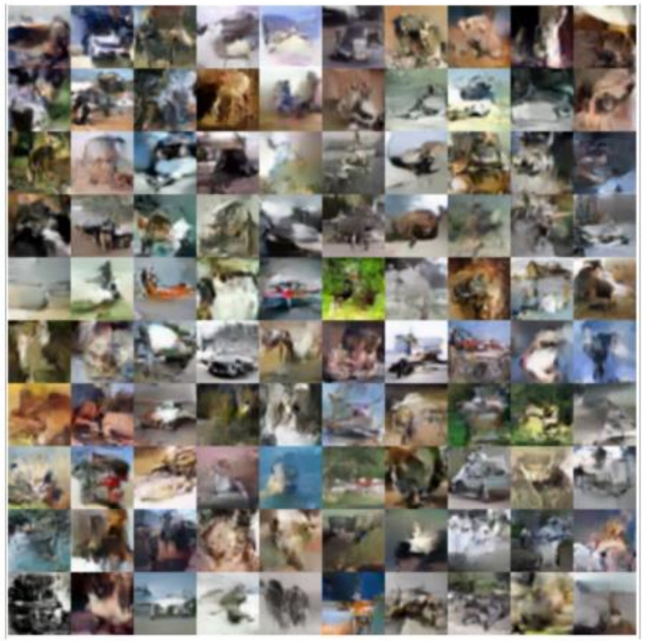
Our assumption about data generation process


Sample from true conditional
$p_{\theta^*}(x \mid z^{(i)})$
x
Decoder network
Sample from true prior
$z^{(i)} \sim p_{\theta^*}(z)$
z

Now given a trained VAE:
use decoder network & sample z from prior!


$\hat{x}$
Sample x|z from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$
$\mu_{x|z}$
$\Sigma_{x|z}$
Decoder network
$p_\theta(x|z)$
z
Sample z from $z \sim \mathcal{N}(0, I)$

Variational Autoencoders: Generating Data!


32x32 CIFAR-10

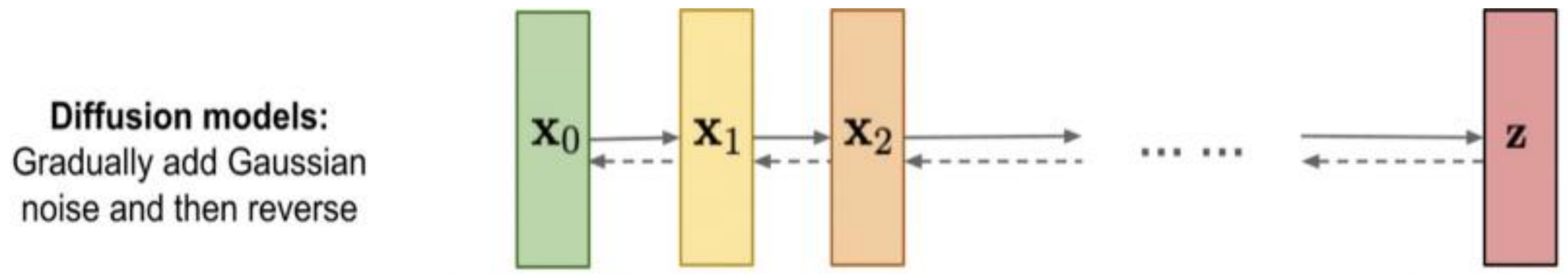
Labeled Faces in the Wild

Take a Break

Diffusion

Idea: Estimating and analyzing small step sizes is more tractable/easier than a single step from random noise to the learned distribution

Convert a well-known and simple base distribution (like a Gaussian) to the target (data) distribution iteratively, with small step sizes, via a Markov chain



Forward Diffusion Process

- Noise added can be parameterized by:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad \{\beta_t \in (0,1)\}_{t=1}^{T}$$

Vary the parameters of the Gaussian according to a *noise schedule*

- You can prove with some math that as T approaches infinity, you eventually end up with an Isotropic Gaussian (i.e. pure random noise)
- Note: forward process is fixed

Forward Diffusion Process

Reparameterization trick:

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

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

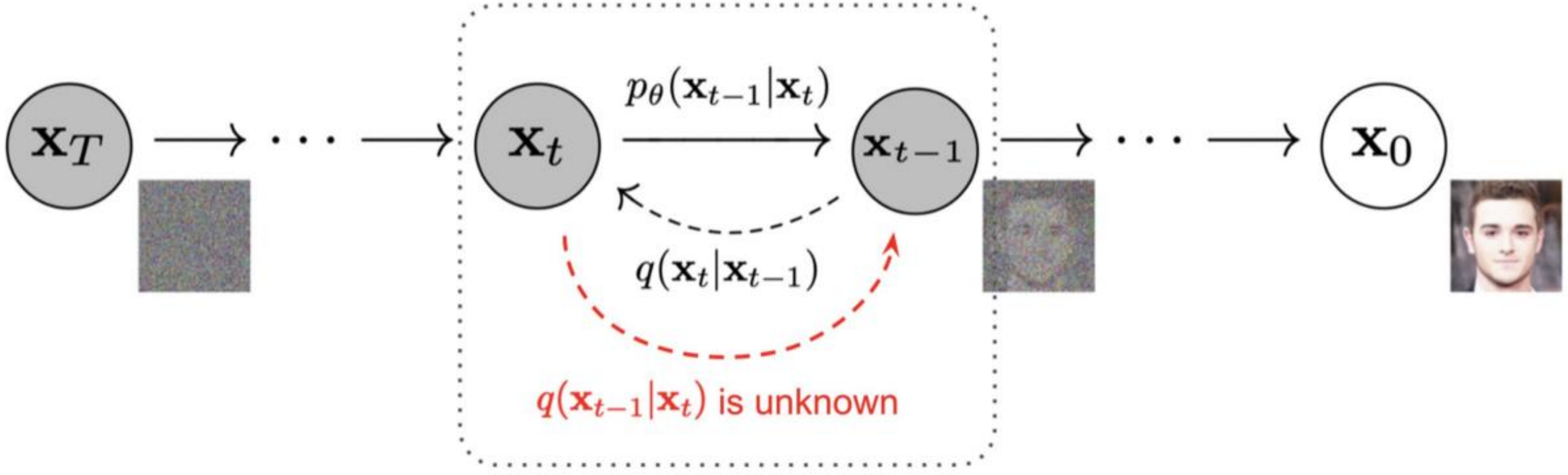
1. Sample an image from the dataset:
2. Sample noise $\epsilon \sim N(0, \mathbf{I})$ (from a **standard** normal distribution)
3. Scale the image by $\sqrt{\overline{\alpha_t}}$: $\sqrt{\overline{\alpha_t}}\, x_0$

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

4. Add $\sqrt{1-\overline{\alpha_t}}\,\epsilon$: $\sqrt{\overline{\alpha_t}}\, x_0 + \sqrt{1-\overline{\alpha_t}}\,\epsilon$ $\longrightarrow$

x_t

Reverse Diffusion Process



$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is unknown

The goal of a diffusion model is to learn the reverse denoising process to iteratively undo the forward process

Distribution in Reverse Process

Turns out that for small enough forward steps, i.e. $\{\beta_t \in (0,1)\}_{t=1}^T$

the reverse process step $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ can be estimated as a Gaussian distribution too

Therefore, we can parametrize the *learned* reverse process as

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

In practice, Σ is just the identify matrix, so we only need to learn the mean of the distribution

Loss Function

$$
\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}
\end{aligned}
$$

$$
\begin{aligned}
L_t &= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t)\|_2^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t,t)\|^2\Big] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\Big) - \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\Big)\|^2\Big] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\|^2\Big] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\Big]
\end{aligned}
$$

Training and Sampling

Algorithm 1 Training

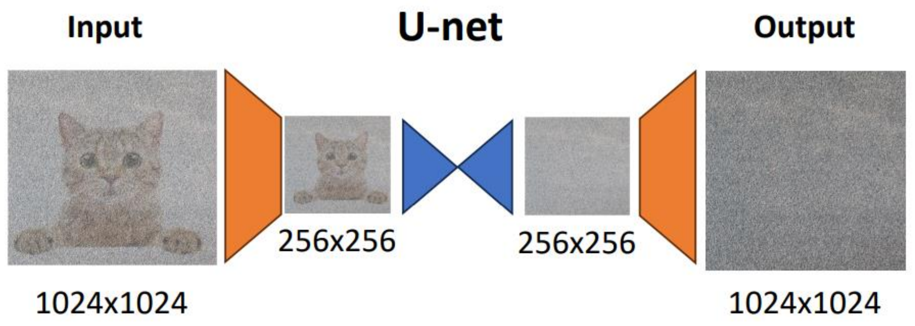
```
1: repeat
2:   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3:   $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
4:   $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:   Take gradient descent step on
         $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

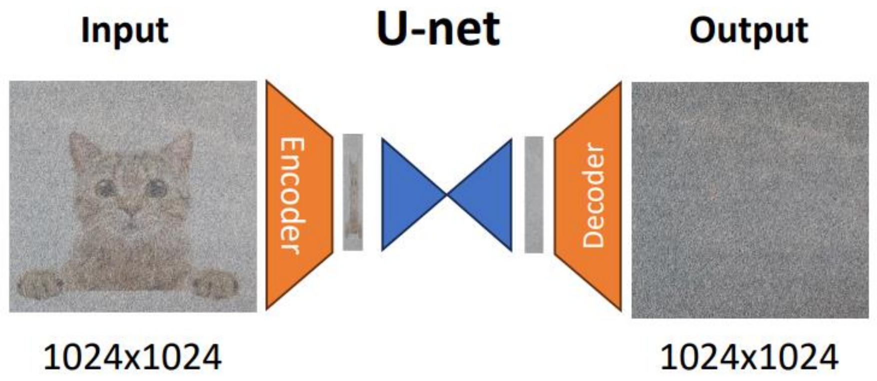
Algorithm 2 Sampling

```
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 $\mathbf{x}_0$
```

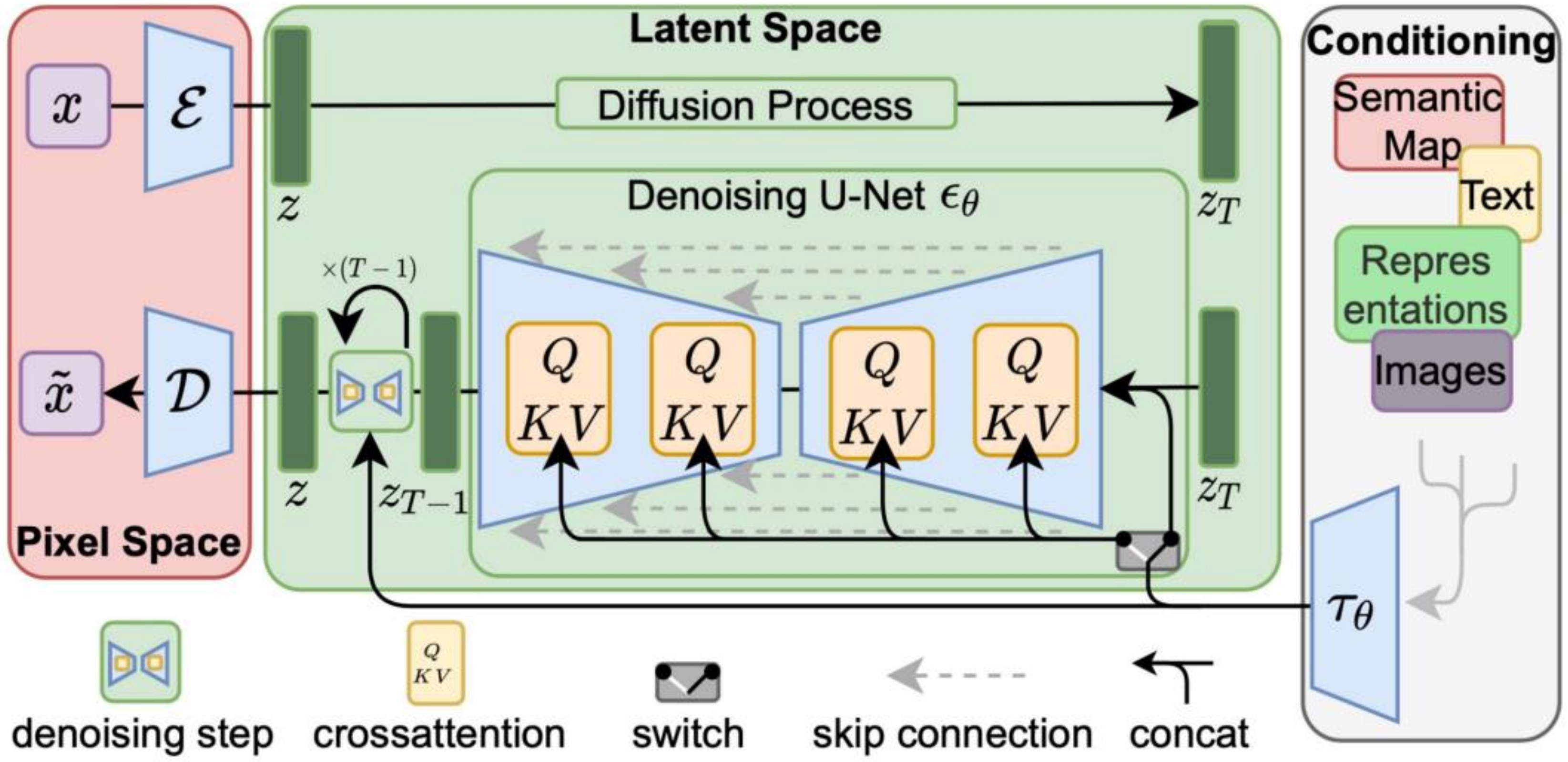
U-Net Problem

- Operating in the input space is very computationally expensive
  - Generate Low-Resolution + Upsample
  - Generate in Latent Space





Stable Diffusion

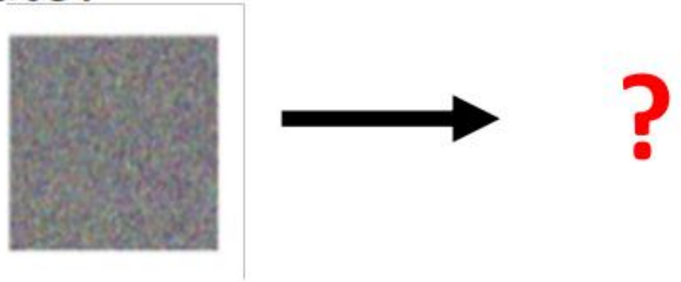


High-Resolution Image Synthesis with Latent Diffusion Models (CVPR 2022)
https://ommer-lab.com/research/latent-diffusion-models/

Guided/Conditioned Diffusion

Lets say we train a diffusion model on images of cats and dogs:

If we start from random noise, and generate a new image, what will the model generate?

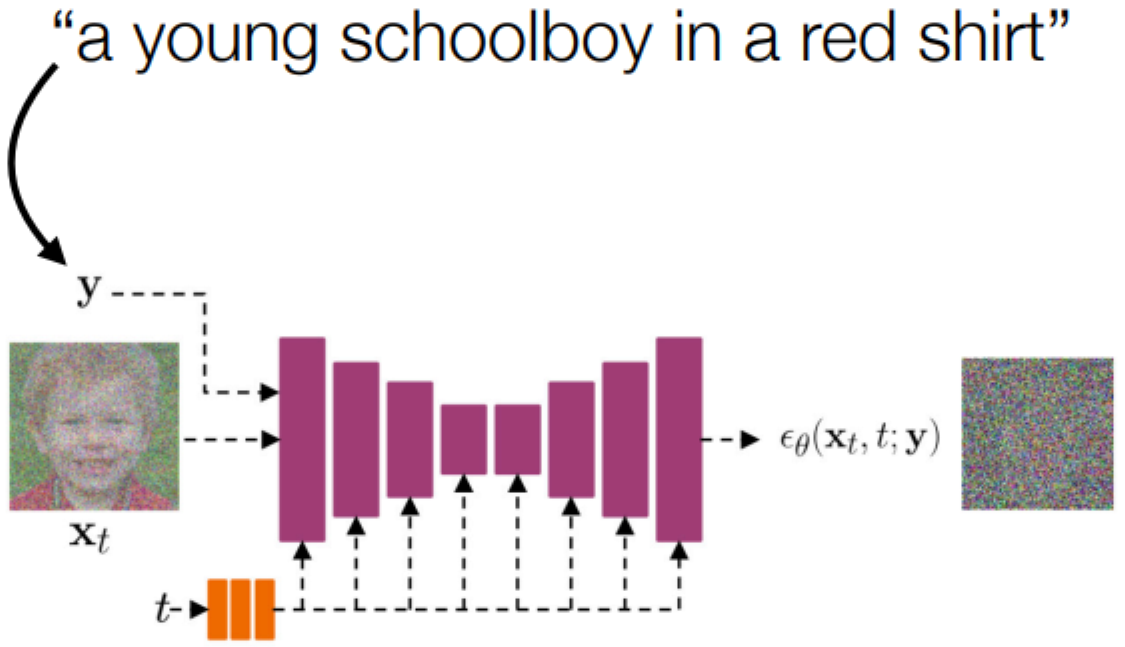


How to Control Diffusion Models?

- Explicit conditioning
- Classifier guidance
- Classifier-free guidance

Explicit conditioning

Use an Image-Text dataset


"a young schoolboy in a red shirt"
$\mathbf{y}$
$\mathbf{x}_t$
t
$\epsilon_\theta(\mathbf{x}_t, t; \mathbf{y})$

Classifier Guidance

Given a Gaussian distribution of x

$\nabla_{\mathbf{x}} \log p(\mathbf{x}) = \nabla_{\mathbf{x}} \Big(-\frac{1}{2\sigma^2}(\mathbf{x} - \boldsymbol{\mu})^2 \Big) = -\frac{\mathbf{x}-\boldsymbol{\mu}}{\sigma^2} = -\frac{\boldsymbol{\epsilon}}{\sigma}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Recall that $q(\mathbf{x}_t|\mathbf{x}_0) \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$ and therefore,

$$\mathbf{s}_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) = \mathbb{E}_{q(\mathbf{x}_0)}[\nabla_{\mathbf{x}_t} q(\mathbf{x}_t|\mathbf{x}_0)] = \mathbb{E}_{q(\mathbf{x}_0)}\Big[-\frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sqrt{1-\bar{\alpha}_t}} \Big] = -\frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sqrt{1-\bar{\alpha}_t}}$$

Score function for the joint distribution:

$$\begin{aligned}
\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t, y) &= \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(y|\mathbf{x}_t) \\
&\approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t) \\
&= -\frac{1}{\sqrt{1-\bar{\alpha}_t}} (\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t} \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t))
\end{aligned}$$

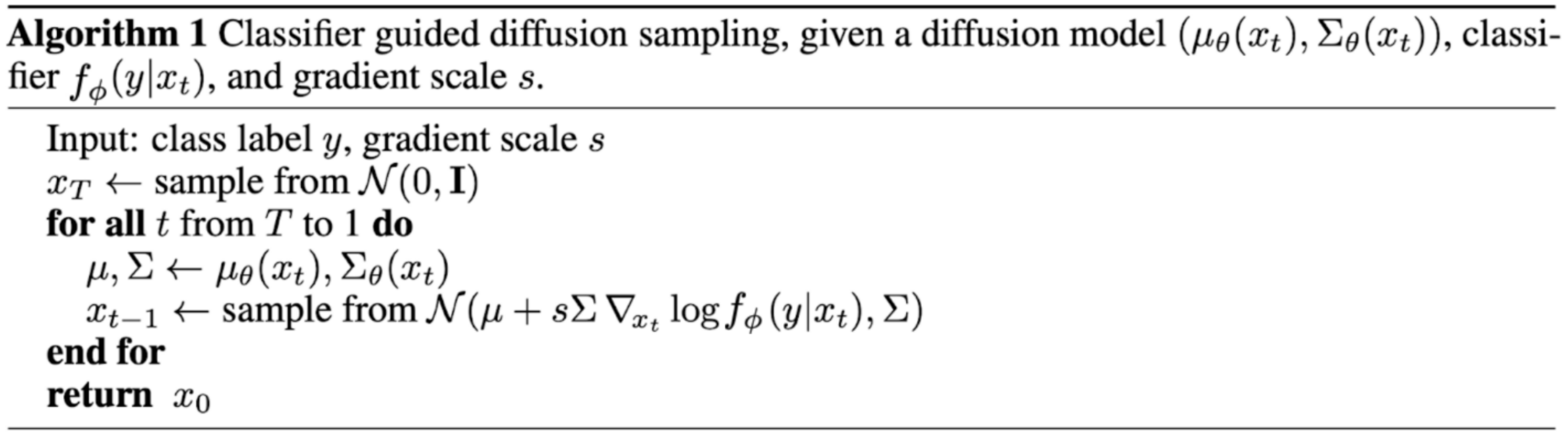
Classifier Guidance

Thus, a new classifier-guided predictor $\bar{\boldsymbol{\epsilon}}_\theta$ would take the form as following,

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

To control the strength of the classifier guidance, we can add a weight w to the delta part,

$$\bar{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t) = \boldsymbol{\epsilon}_\theta(x_t, t) - \sqrt{1 - \bar{\alpha}_t}\; w \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)$$

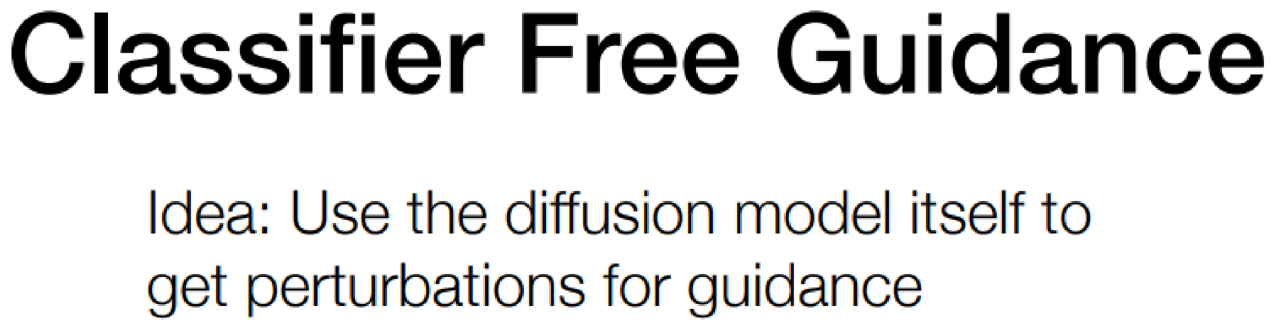


Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $f_\phi(y|x_t)$, and gradient scale s.

```
Input: class label $y$, gradient scale $s$
$x_T \leftarrow$ 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 $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log f_\phi(y|x_t), \Sigma)$
**end for**
**return** $x_0$
```

Problems with Classifier Guidance

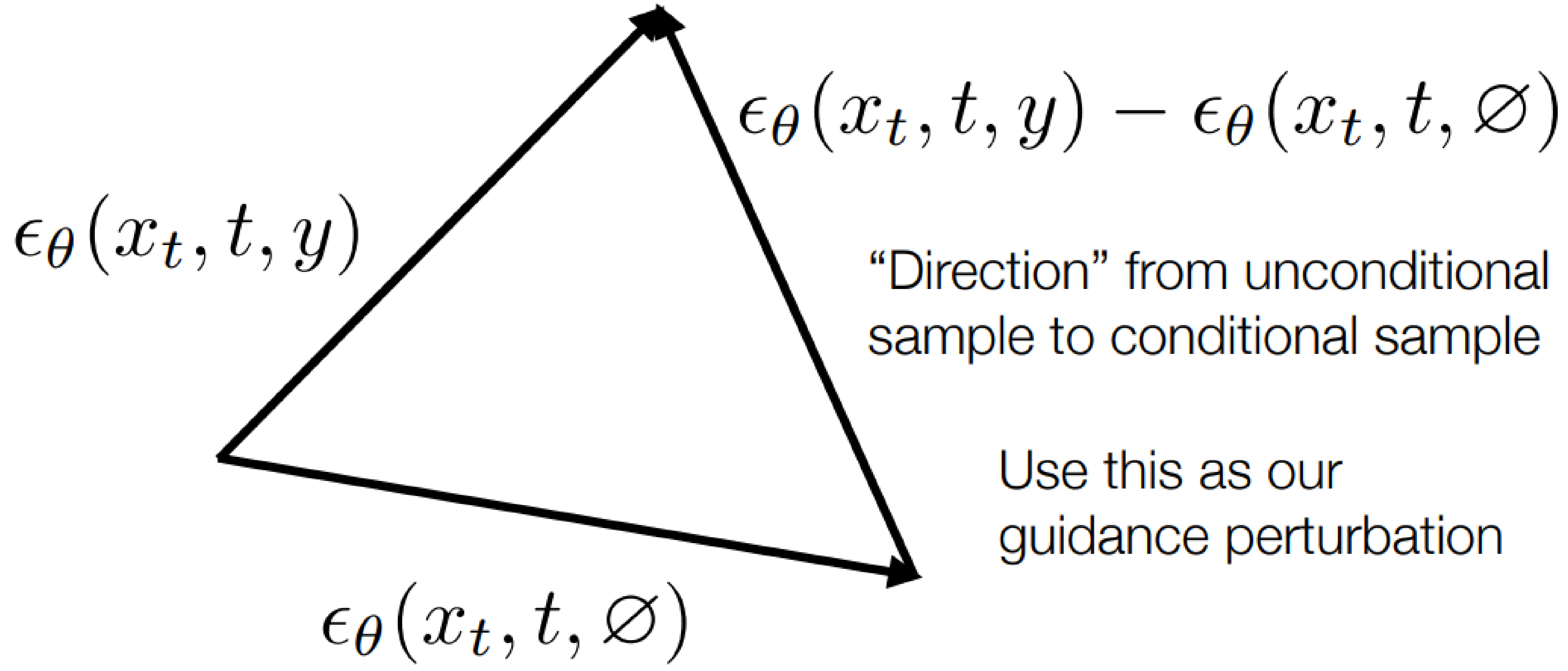
- Need to fine-tune or re-train a classifier on noisy data
- Need a pre-trained classification model
  - What if we want to use any text prompt as input?



Classifier Free Guidance

Idea: Use the diffusion model itself to
get perturbations for guidance

Classifier Free Guidance


$\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t, \varnothing)$
$\epsilon_\theta(x_t, t, y)$
$\epsilon_\theta(x_t, t, \varnothing)$

"Direction" from unconditional sample to conditional sample

Use this as our guidance perturbation

Classifier Free Guidance

- A conditional diffusion model is trained on pair data (x, y), where the conditioning information y get discarded at random

$$
\begin{aligned}
\nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) &= \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|y) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \\
&= -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \Big(\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \Big)
\end{aligned}
$$

Classifier-guided modified score

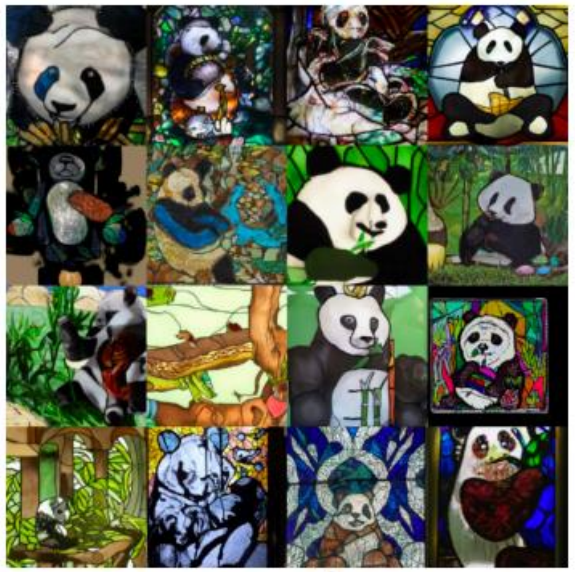
$$
\begin{aligned}
\bar{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t, y) &= \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - \sqrt{1-\bar{\alpha}_t}\, w \nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) \\
&= \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) + w \big(\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \big) \\
&= (w+1)\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - w\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)
\end{aligned}
$$

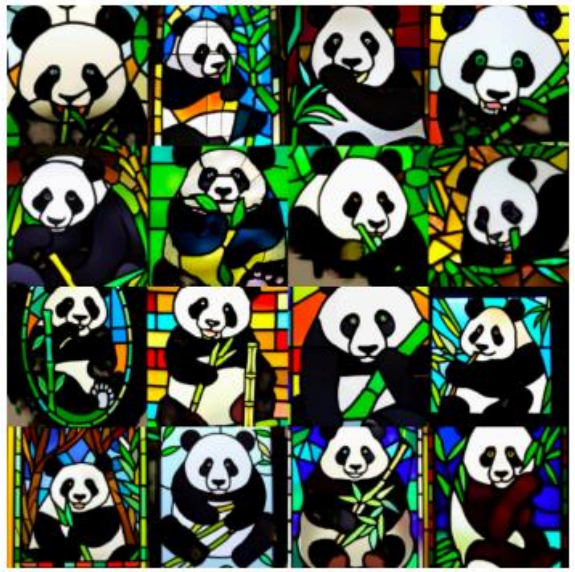
Classifier Free Guidance: Text2Image

Our new noise estimate will then be:

$$\tilde{\epsilon}(x_t, t, y) = \epsilon_\theta(x_t, t, \varnothing) + \gamma\underbrace{(\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t, \varnothing))}_{\text{“Direction” from unconditional to conditional}}$$

“A stained glass window of a panda eating bamboo”


$\gamma = 1$


$\gamma = 3$

References

- https://www.eecs.umich.edu/courses/eecs442-ahowens/fa23/slides/lec11-diffusion.pdf
- https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
- https://cs231n.stanford.edu/slides/2024/lecture_13.pdf