

# Introduction to diffusion models

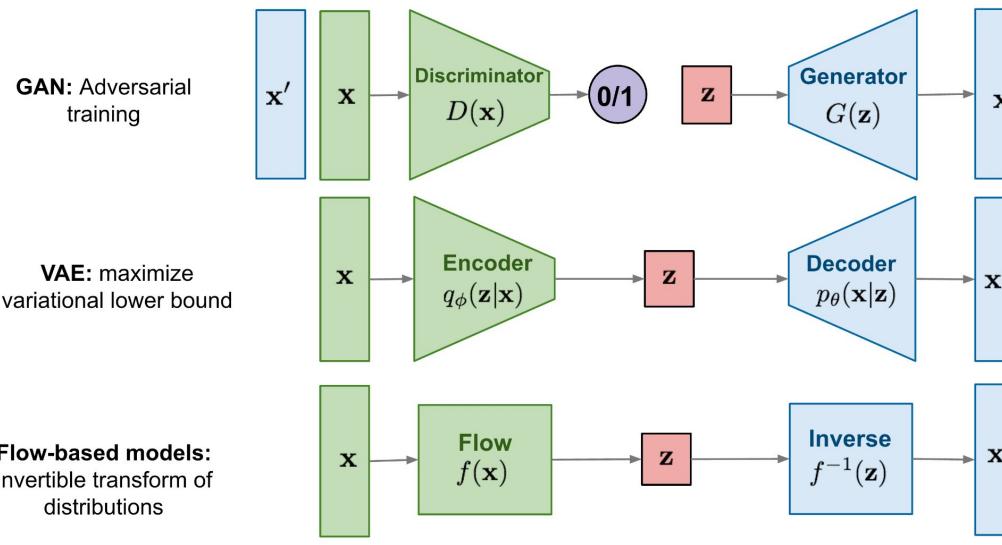


# Generative models

# Generative models

Can you name some family of generative models you know?

# Generative models

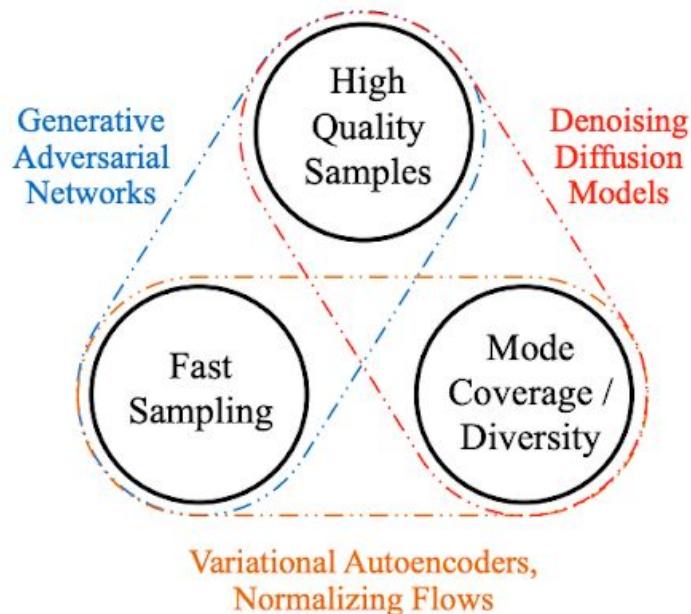


source: [What are Diffusion Models?](#)

# Generative models

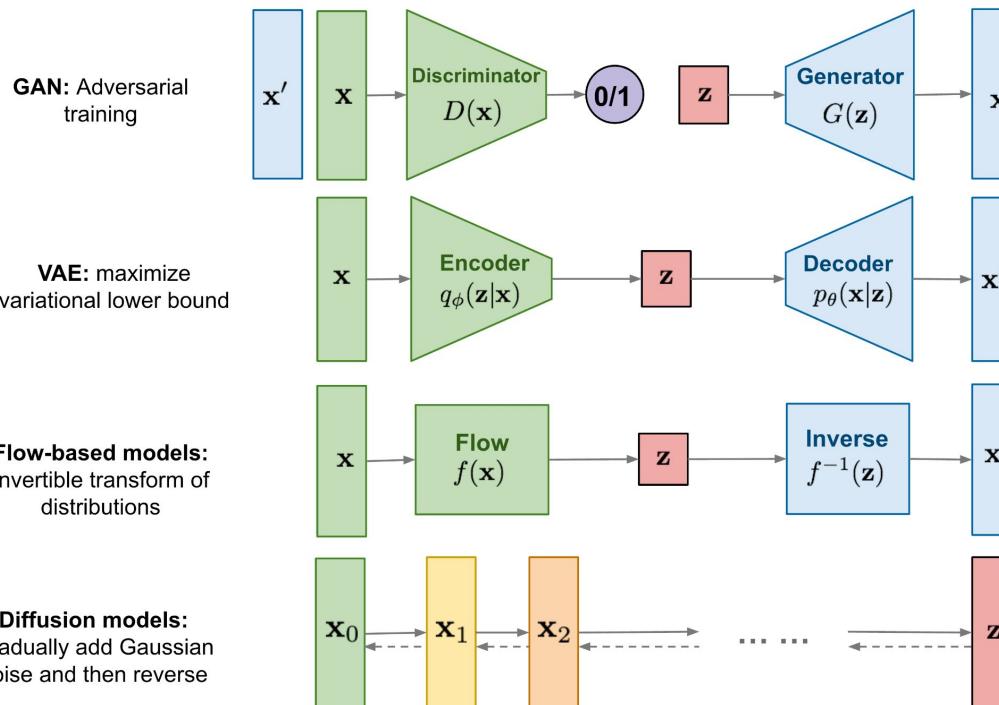
What are the advantages and drawbacks of these models?

# Generative models



source: <https://developer.nvidia.com/blog/improving-diffusion-models-as-an-alternative-to-gans-part-1/>

# Generative models



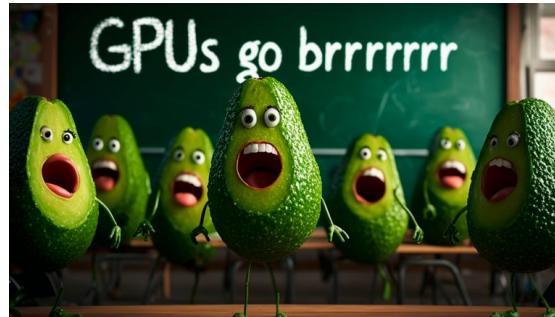
source: [What are Diffusion Models?](#)

# Current best models

Flux



Stable Diffusion



# Diffusion for other modalities

SORA



Stable diffusion audio



# Diffusion models

# Diffusion

- slowly destroy structure in a data distribution through an **iterative forward diffusion process**.
- learn a **reverse diffusion process** that **restores structure in data**



source: [https://www.accessscience.com/highwire\\_display/entity\\_view/node/370286/focus\\_view](https://www.accessscience.com/highwire_display/entity_view/node/370286/focus_view)

# Diffusion

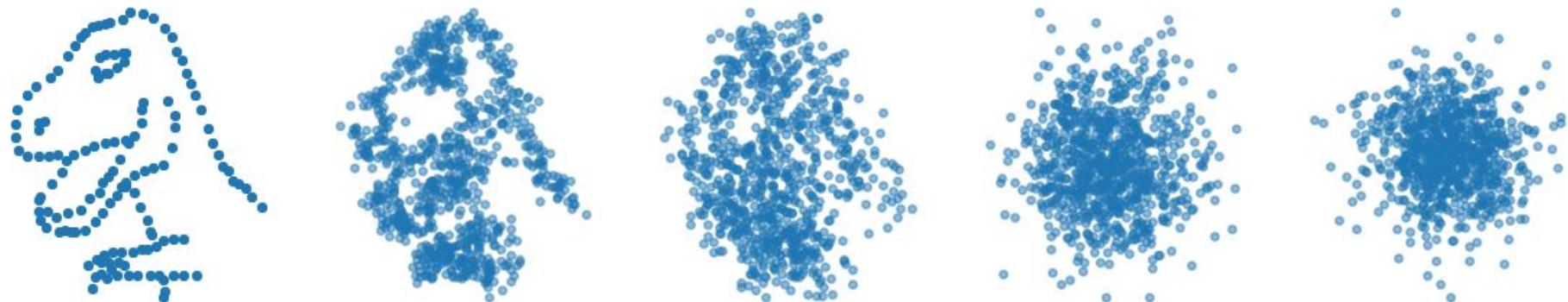
- Start from a data point in the original distribution
- Add a small amount of noise
- Repeat until it looks to pure noise



source: <https://cvpr2022-tutorial-diffusion-models.github.io/>

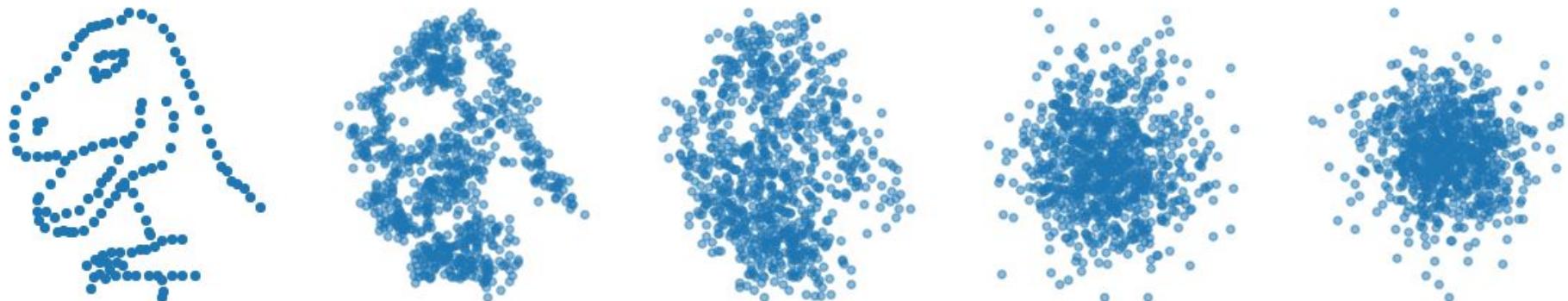
# Diffusion

- At a given step we can kind of guess where was a sample at previous step.
- A sample  $y$  in the target space comes from a sample  $x$  in the original space with a probability close to  $p(x)$
- If we learn how to reverse each step, we can sample from the  $p(x)$



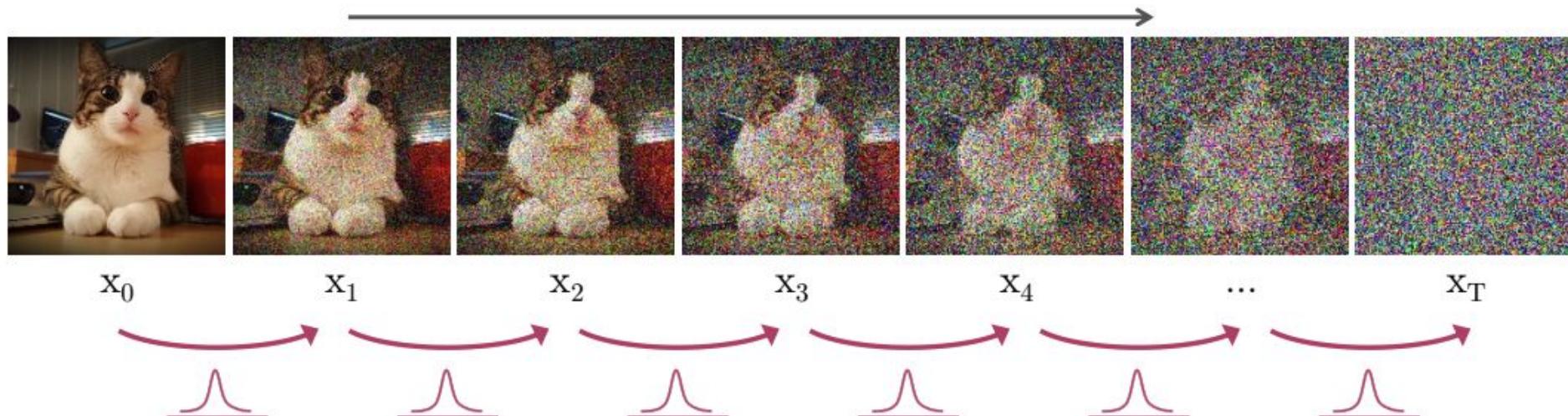
# Diffusion

- **Forward pass** that gradually add noise to the data to move towards a distribution easy to sample
- **Backward or reverse process** that iteratively remove noise to samples to return into the original distribution



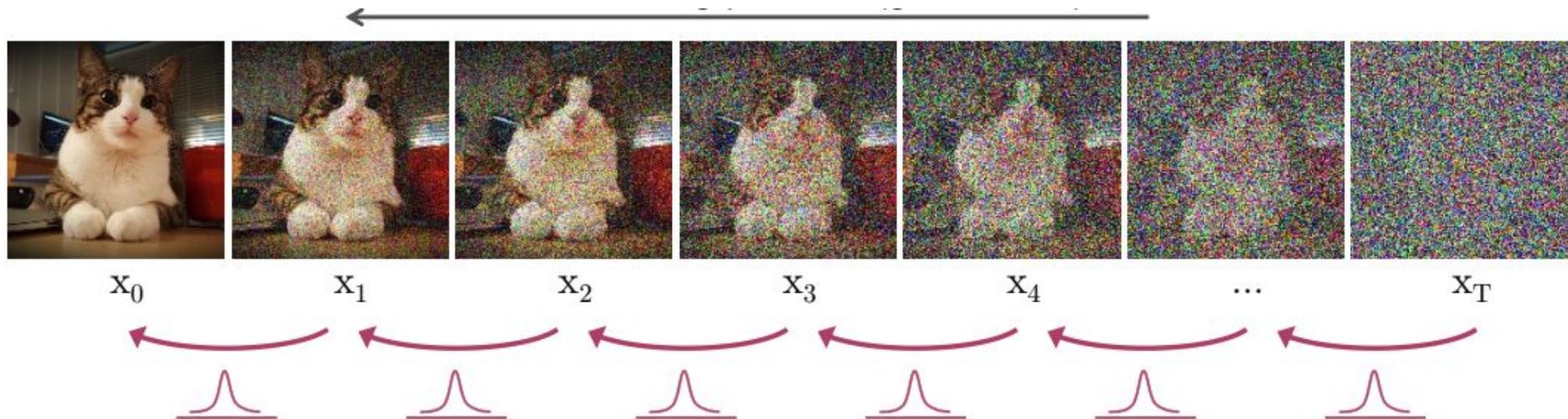
# Forward pass

- Gradually inject noise to the data using a **fixed markov chain**
- Latent variables have **the same dimensionality** as the original ones
- Images are asymptotically transformed to **pure Gaussian noise**



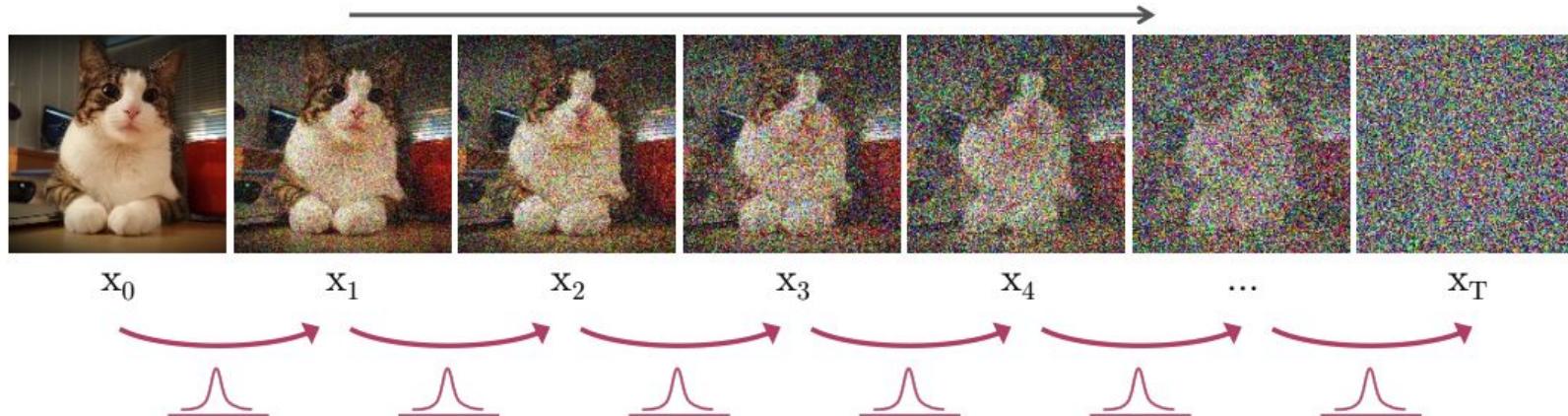
# Backward pass

- Going backward on this chain means that we can sample new images.
- **Problem:** we don't know this reverse process
- **Solution:** estimate it with a neural network parametrized by  $\theta$

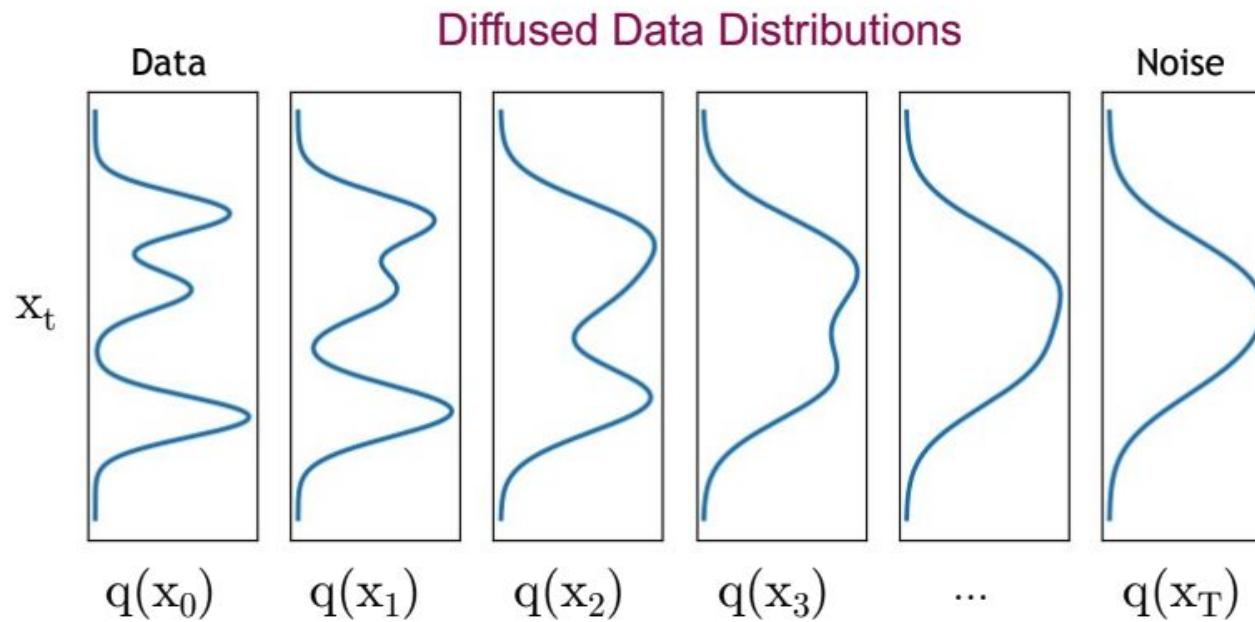


# Forward pass

- Given a data point sampled from a source distribution  $\mathbf{x}_0 \sim q(\mathbf{x})$ , we define a forward diffusion process by adding a small amount of Gaussian noise in  $T$  steps
- This produces a sequence of latent variables  $\mathbf{x}_1, \dots, \mathbf{x}_T$

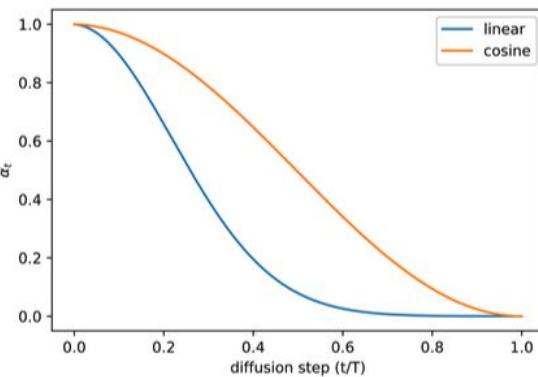


# Forward pass



# Forward pass

- $q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I})$
- Where  $\beta_t$  is determined by a schedule (linear, cosine ...)

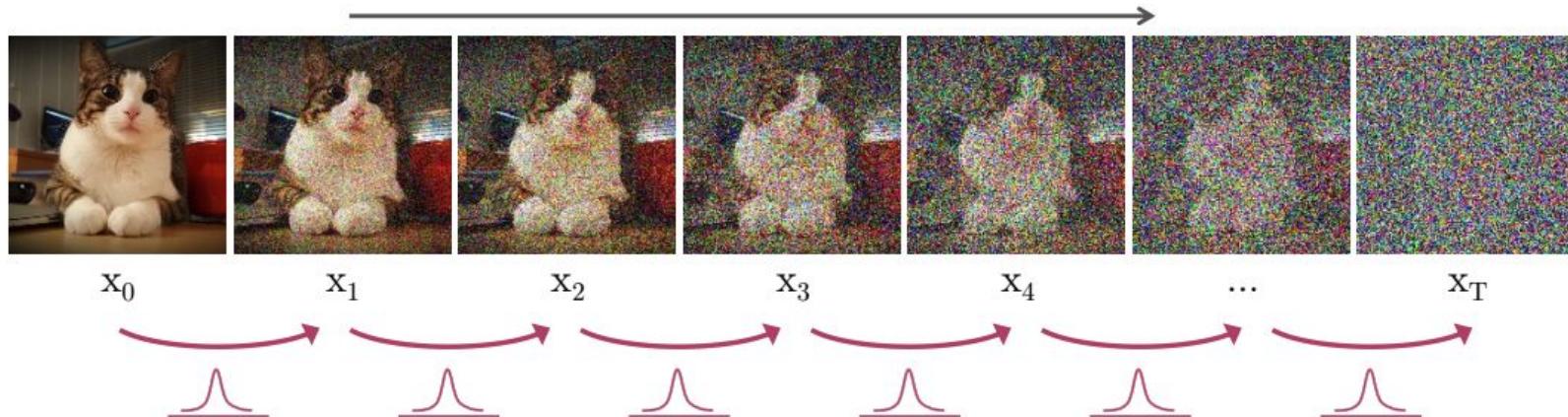


Linear

Cosine

# Sampling

- We want to learn the denoising process using a Neural network
- We need i.i.d data => different images at different time steps
- **Problem:** forward pass is expensive:  $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})$



# Sampling

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

**Reparametrization trick:**  $\mathcal{N}(\mu, \sigma^2) = \mu + \sigma * \epsilon \quad \text{with } \epsilon \sim \mathcal{N}(0, \mathbf{I})$

So we can write

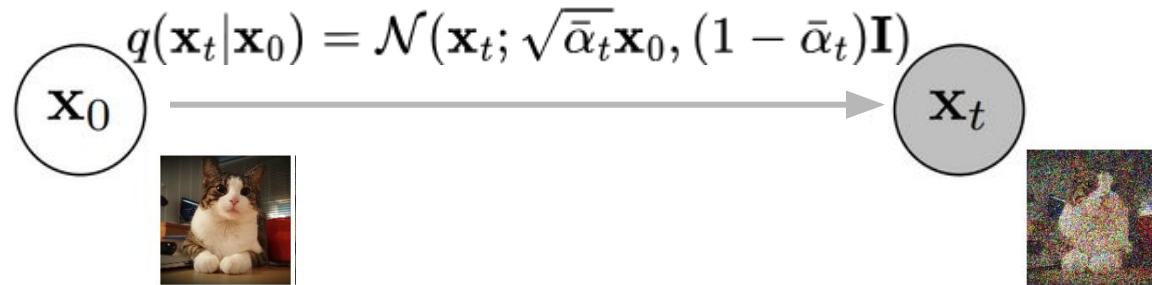
$$\begin{aligned}\mathbf{x}_t &= \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{1 - \alpha_t} \epsilon_{t-1} \\&= \sqrt{\alpha_t} (\sqrt{\alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_{t-1}} \epsilon_{t-2}) + \sqrt{1 - \alpha_t} \epsilon_{t-1} \\&= \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{\alpha_t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

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

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

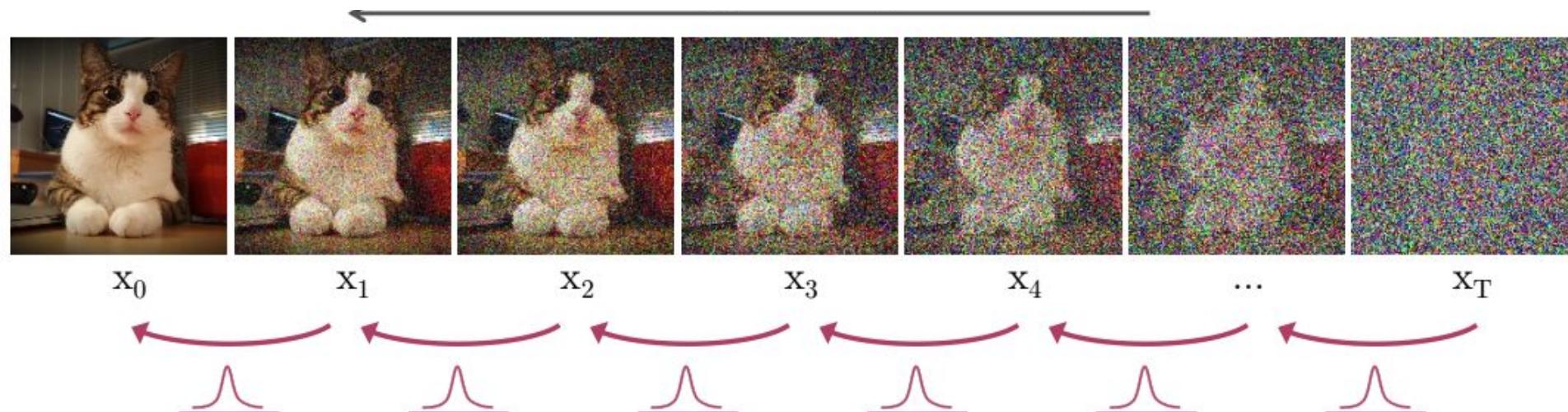
# Forward pass

- We can now directly sample a noisy latent  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$  for any  $t \in [0, \dots,]T$  with  $\bar{\alpha}_t := \prod_{s=1}^t \alpha_s$  and  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$



# Backward pass

- Going backward on this chain means that we can sample new images.
- **Problem:** we don't know this reverse process  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$
- **Solution:** estimate it with a model parametrized by  $\theta$ :  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$



# Backward pass

- Going backward on this chain means that we can sample new images.
- **Problem:** we don't know this reverse process  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$
- **Solution:** estimate it with a model parametrized by  $\theta$ :  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

We want to find  $\theta$  that minimizes:

$$L = \mathbb{E}_{x_0, T \sim q} \left[ \sum_{t=1}^T D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t) || p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \right]$$

It can be proven that minimizing  $L$  maximizes the likelihood of generating the dataset because it optimizes a lower bound for the same, through a process called variational inference (see [Lilian Weng's blog](#) for full derivations).

# Full derivation (from Lilian Weng's blog)

As demonstrated in Fig. 2., such a setup is very similar to VAE and thus we can use the variational lower bound to optimize the negative log-likelihood.

$$\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{\text{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0) \| p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T} \sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)} \right] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q \left[ \log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0) \right] \\
&= \mathbb{E}_q \left[ \log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] \\
\text{Let } L_{\text{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[ \log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0)
\end{aligned}$$

It is also straightforward to get the same result using Jensen's inequality. Say we want to minimize the cross entropy as the learning objective,

$$\begin{aligned}
L_{\text{CE}} &= -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0) \\
&= -\mathbb{E}_{q(\mathbf{x}_0)} \log \left( \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \right) \\
&= -\mathbb{E}_{q(\mathbf{x}_0)} \log \left( \int q(\mathbf{x}_{1:T}|\mathbf{x}_0) \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} d\mathbf{x}_{1:T} \right) \\
&= -\mathbb{E}_{q(\mathbf{x}_0)} \log \left( \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right) \\
&\leq -\mathbb{E}_{q(\mathbf{x}_{0:T})} \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \\
&= \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[ \log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] = L_{\text{VLB}}
\end{aligned}$$

# Full derivation (from Lilian Weng's blog)

To convert each term in the equation to be analytically computable, the objective can be further rewritten to be a combination of several KL-divergence and entropy terms (See the detailed step-by-step process in Appendix B in Sohl-Dickstein et al., 2015):

$$\begin{aligned}
L_{VLB} &= \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[ \log \frac{q(\mathbf{x}_{1:T} | \mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] \\
&= \mathbb{E}_q \left[ \log \frac{\prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} \right] \\
&= \mathbb{E}_q \left[ -\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^T \log \frac{q(\mathbf{x}_t | \mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} \right] \\
&= \mathbb{E}_q \left[ -\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t | \mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1 | \mathbf{x}_0)}{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)} \right] \\
&= \mathbb{E}_q \left[ -\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \left( \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_t | \mathbf{x}_0)}{q(\mathbf{x}_{t-1} | \mathbf{x}_0)} \right) + \log \frac{q(\mathbf{x}_1 | \mathbf{x}_0)}{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)} \right] \\
&= \mathbb{E}_q \left[ -\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t | \mathbf{x}_0)}{q(\mathbf{x}_{t-1} | \mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1 | \mathbf{x}_0)}{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)} \right] \\
&= \mathbb{E}_q \left[ -\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T | \mathbf{x}_0)}{q(\mathbf{x}_1 | \mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1 | \mathbf{x}_0)}{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)} \right] \\
&= \mathbb{E}_q \left[ \log \frac{q(\mathbf{x}_T | \mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0 | \mathbf{x}_1) \right] \\
&= \mathbb{E}_q \underbrace{[D_{KL}(q(\mathbf{x}_T | \mathbf{x}_0) \| p_\theta(\mathbf{x}_T))]}_{L_T} + \underbrace{\sum_{t=2}^T D_{KL}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{L_{T-1}} - \underbrace{\log p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}_{L_0}
\end{aligned}$$

Let's label each component in the variational lower bound loss separately:

$$\begin{aligned}
L_{VLB} &= L_T + L_{T-1} + \cdots + L_0 \\
\text{where } L_T &= D_{KL}(q(\mathbf{x}_T | \mathbf{x}_0) \| p_\theta(\mathbf{x}_T)) \\
L_t &= D_{KL}(q(\mathbf{x}_t | \mathbf{x}_{t+1}, \mathbf{x}_0) \| p_\theta(\mathbf{x}_t | \mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1 \\
L_0 &= -\log p_\theta(\mathbf{x}_0 | \mathbf{x}_1)
\end{aligned}$$

Every KL term in  $L_{VLB}$  (except for  $L_0$ ) compares two Gaussian distributions and therefore they can be computed in closed form.  $L_T$  is constant and can be ignored during training because  $q$  has no learnable parameters and  $\mathbf{x}_T$  is a Gaussian noise. Ho et al. 2020 models  $L_0$  using a separate discrete decoder derived from  $\mathcal{N}(\mathbf{x}_0; \mu_\theta(\mathbf{x}_1, 1), \Sigma_\theta(\mathbf{x}_1, 1))$ .

# Backward pass

$$L = \mathbb{E}_{x_0, T \sim q} \left[ \sum_{t=1}^T D_{KL}(q(\mathbf{x}_{t-1} | \mathbf{x}_t) || p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)) \right]$$

We don't know  $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$  but we actually know  $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ :

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t | \mathbf{x}_{t-1}) q(\mathbf{x}_{t-1} | \mathbf{x}_0) q(\mathbf{x}_0)}{q(\mathbf{x}_t | \mathbf{x}_0) q(\mathbf{x}_0)} \quad (\text{Bayes' rule})$$

$$= \mathcal{N}(\mu(\mathbf{x}_t, \mathbf{x}_0), \Sigma(t)\mathbf{I}) \quad (\text{By subbing in the distribution formulas and doing the algebra  
see next slide for details})$$

$$\text{with } \mu(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\alpha_t}(1-\alpha_{t-1})\mathbf{x}_t + \sqrt{\alpha_{t-1}}(1-\alpha_t)\mathbf{x}_0}{1-\bar{\alpha}_t} \text{ and } \Sigma(t) = \frac{(1-\alpha_t)(1-\alpha_{t-1})}{1-\bar{\alpha}_t}$$

# Full derivation (from Lilian Weng's blog)

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\boldsymbol{\beta}}_t \mathbf{I})$$

Using Bayes' rule, we have:

$$\begin{aligned} q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) &= q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_0)}{q(\mathbf{x}_t | \mathbf{x}_0)} \\ &\propto \exp\left(-\frac{1}{2}\left(\frac{(\mathbf{x}_t - \sqrt{\alpha_t} \mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_0)^2}{1 - \bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0)^2}{1 - \bar{\alpha}_t}\right)\right) \\ &= \exp\left(-\frac{1}{2}\left(\frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t} \mathbf{x}_t \mathbf{x}_{t-1} + \alpha_t \mathbf{x}_{t-1}^2}{\beta_t} + \frac{\mathbf{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_{t-1} \mathbf{x}_0 + \bar{\alpha}_{t-1} \mathbf{x}_0^2}{1 - \bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0)^2}{1 - \bar{\alpha}_t}\right)\right) \\ &= \exp\left(-\frac{1}{2}\left(\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}}\right) \mathbf{x}_{t-1}^2 - \left(\frac{2\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0\right) \mathbf{x}_{t-1} + C(\mathbf{x}_t, \mathbf{x}_0)\right)\right) \end{aligned}$$

where  $C(\mathbf{x}_t, \mathbf{x}_0)$  is some function not involving  $\mathbf{x}_{t-1}$  and details are omitted. Following the standard Gaussian density function, the mean and variance can be parameterized as follows (recall that  $\alpha_t = 1 - \beta_t$  and  $\bar{\alpha}_t = \prod_{i=1}^T \alpha_i$ ):

$$\begin{aligned} \tilde{\beta}_t &= 1 / \left( \frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right) = 1 / \left( \frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\beta_t (1 - \bar{\alpha}_{t-1})} \right) = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \cdot \beta_t \\ \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) &= \left( \frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right) / \left( \frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right) \\ &= \left( \frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right) \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \cdot \beta_t \\ &= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0 \end{aligned}$$

Thanks to the nice property, we can represent  $\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_t)$  and plug it into the above equation and obtain:

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_t &= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1 - \bar{\alpha}_t} \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_t) \\ &= \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_t \right) \end{aligned}$$

# Backward pass

$$L = \mathbb{E}_{x_0, T \sim q} \left[ \sum_{t=1}^T D_{KL}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) || p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)) \right]$$

**Hypothesis:** If  $\beta_t$  is small enough  $q(x_{t-1}|x_t)$  is gaussian.

=> We want to learn  $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}, \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$

**Denoising Diffusion Probabilistic Models** => For numerical stability set  $\Sigma_\theta = \Sigma(t)$   
J. Ho, A. Jain, P. Abbeel. 2020.

# Backward pass

$$L = \mathbb{E}_{x_0, T \sim q} \left[ \sum_{t=1}^T D_{KL}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) || p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)) \right]$$

**KL between two gaussians** (with same variance  $\Sigma(t)$ )

$$\Rightarrow L = \sum_{t=1}^T \mathbb{E}_{x_0, T \sim q} \left[ \frac{1}{2\Sigma(t)} \|\mu(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t)\|^2 \right]$$

With  $\mu(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\alpha_t(1-\alpha_{t-1})}\mathbf{x}_t + \sqrt{\alpha_{t-1}(1-\alpha_t)}\mathbf{x}_0}{1-\bar{\alpha}_t}$

We can use  $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0 \Rightarrow x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_0}{\sqrt{\bar{\alpha}_t}}$  to replace  $x_0$  above

$$\Rightarrow \mu(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon$$

# Backward pass

$$L = \mathbb{E}_{x_0, T \sim q} \left[ \sum_{t=1}^T D_{KL}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) || p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)) \right]$$

**KL between two gaussians** (with same variance  $\Sigma(t)$ )

$$\Rightarrow L = \sum_{t=1}^T \mathbb{E}_{x_0, T \sim q} \left[ \frac{1}{2\Sigma(t)} \|\mu(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t)\|^2 \right]$$

With  $\mu(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}} \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t} \sqrt{\alpha_t}} \epsilon$

We define then  $\mu_\theta(\mathbf{x}_t)$  to match above:  $\mu_\theta(\mathbf{x}_t) = \frac{1}{\sqrt{\alpha_t}} \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t} \sqrt{\alpha_t}} \epsilon_\theta(\mathbf{x}_t, t)$

$$\Rightarrow L = \sum_{t=1}^T \mathbb{E}_{x_0, \epsilon \sim q} \left[ \frac{(1-\alpha_t)^2}{2\Sigma(t)\alpha_t(1-\bar{\alpha}_t)} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right]$$

# Backward pass

$$L = \sum_{t=1}^T \mathbb{E}_{x_0, \epsilon \sim q} \left[ \frac{(1-\alpha_t)^2}{2\Sigma(t)\alpha_t(1-\bar{\alpha}_t)} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right]$$

Denoising Diffusion Probabilistic Models

J. Ho, A. Jain, P. Abbeel. 2020.

=> ignoring the weighting improves the quality of results

$$L = \sum_{t=1}^T \mathbb{E}_{x_0, \epsilon \sim q} \left[ \cancel{\frac{(1-\alpha_t)^2}{2\Sigma(t)\alpha_t(1-\bar{\alpha}_t)}} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right]$$

$$L_{simple} = \sum_{t=1}^T \mathbb{E}_{x_0, \epsilon \sim q} [\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2]$$

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

# Training:

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

---

## 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  
      
$$\nabla_{\theta} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t)\|^2$$
  - 6: **until** converged
-

# Sampling:

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

With  $\mu_{\theta}(\mathbf{x}_t) = \frac{1}{\sqrt{\alpha_t}} \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t} \sqrt{\alpha_t}} \epsilon_{\theta}(\mathbf{x}_t, t)$  and  $\sum_{\theta} = \sum(t)$

---

## Algorithm 2 Sampling

---

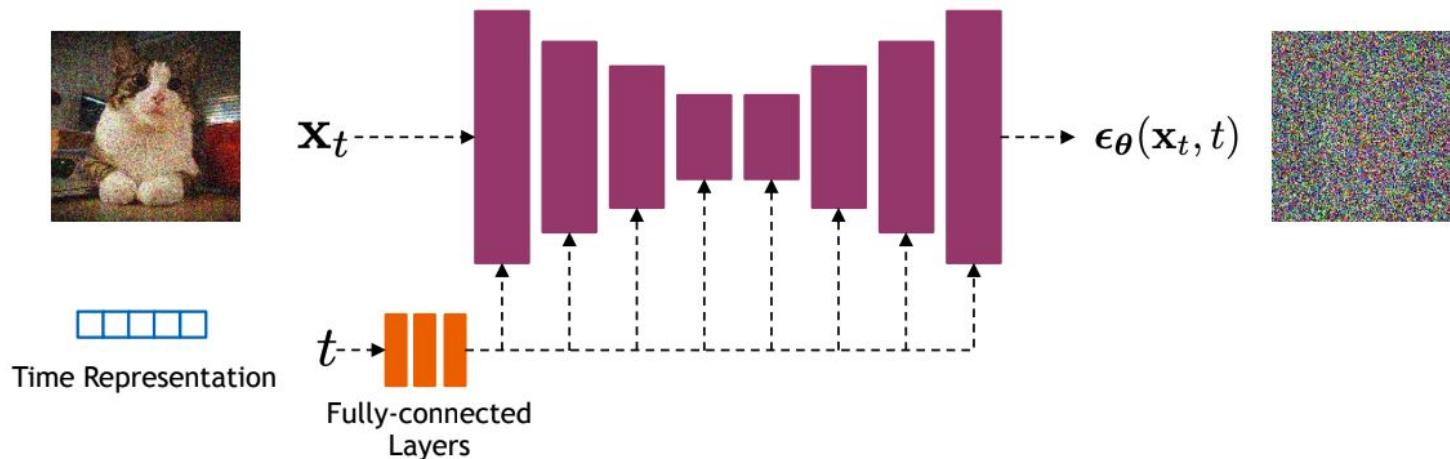
- 1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - 2: **for**  $t = T, \dots, 1$  **do**
  - 3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$
  - 4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}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

# Question:

What does the network predict?

A/ Noise to remove to obtain  $\mathbf{x}_{t-1}$

B/ Noise to remove to obtain  $\mathbf{x}_0$



In practice

# Training:

---

## 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
      
$$\nabla_{\theta} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t)\|^2$$

6: until converged
```

---

---

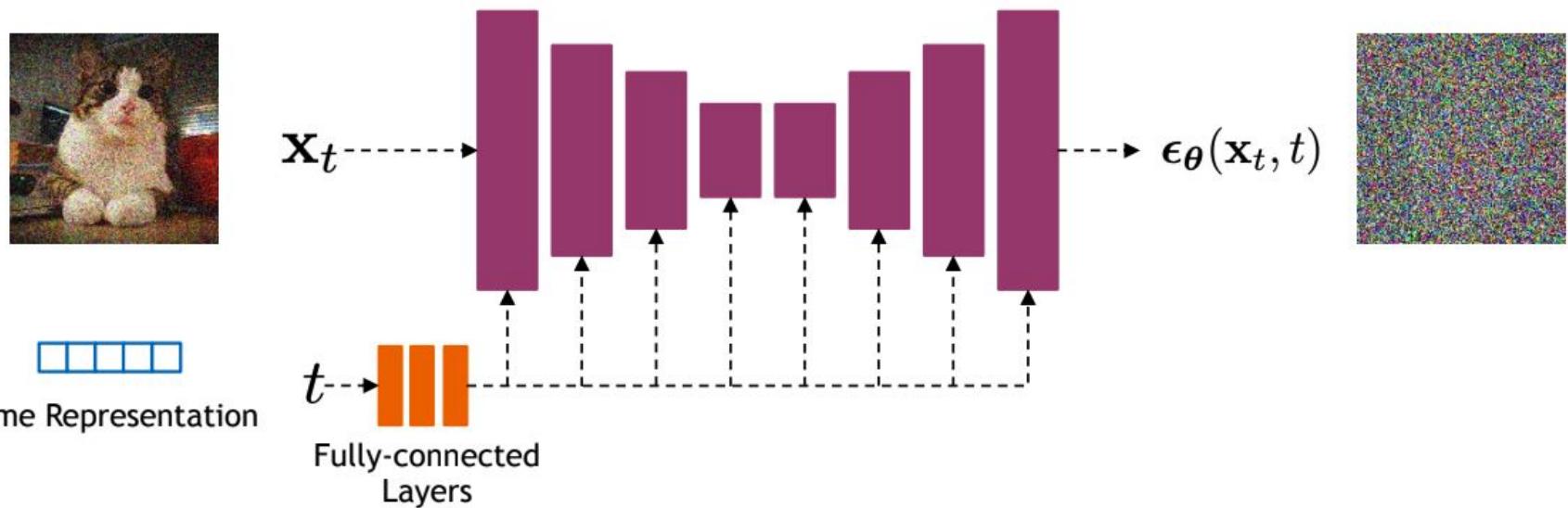
## Algorithm 2 Sampling

---

```
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:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}} \left( \mathbf{x}_t - \frac{1 - \bar{\alpha}_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 
```

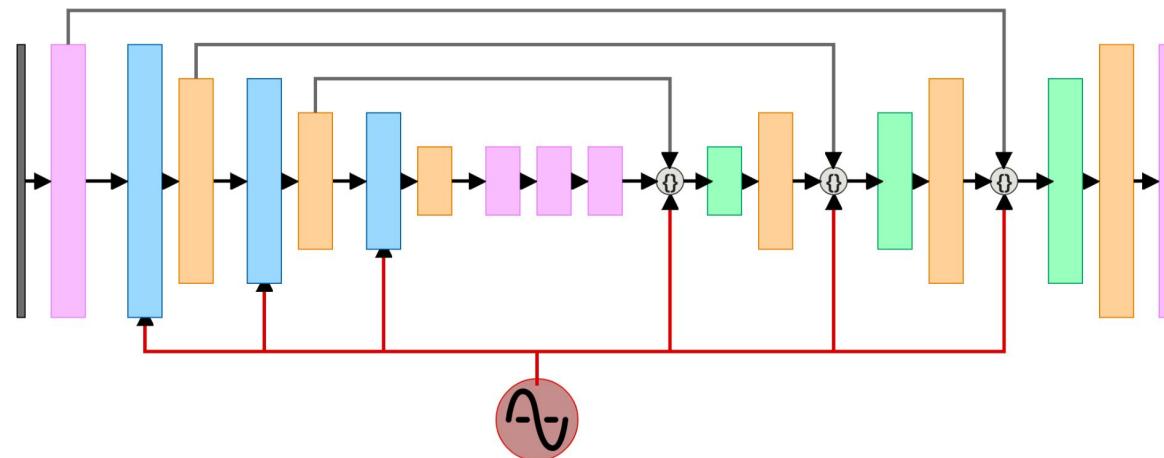
---

# Unet denoiser



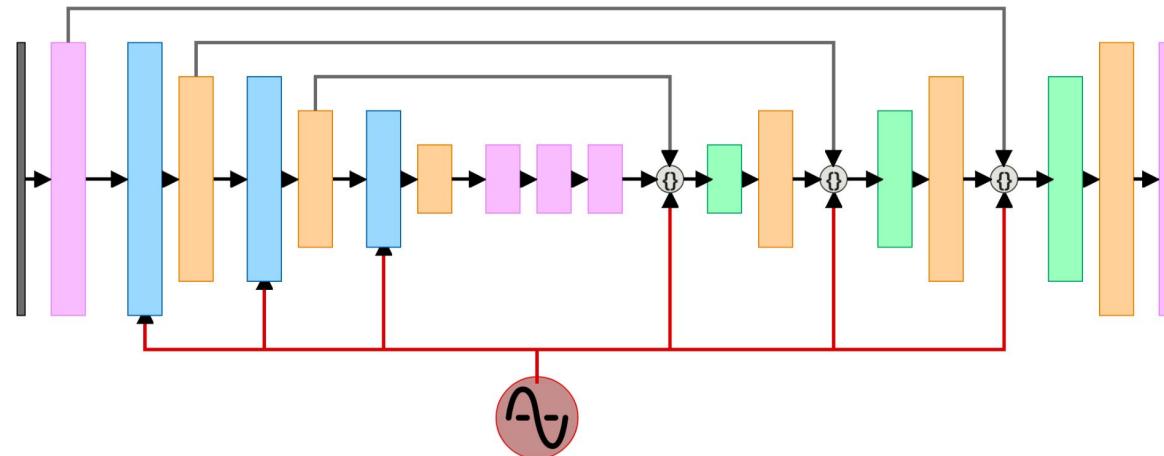
# Unet denoiser

Input
Conv Block
Downsample Block
Self-Attention Block
Upsample Block
Embedding vector



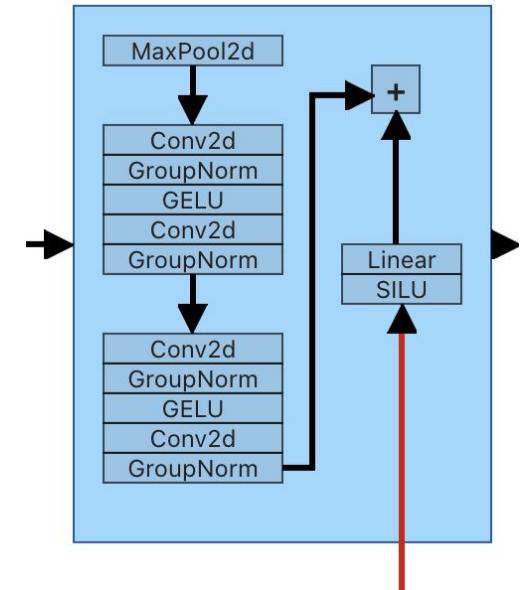
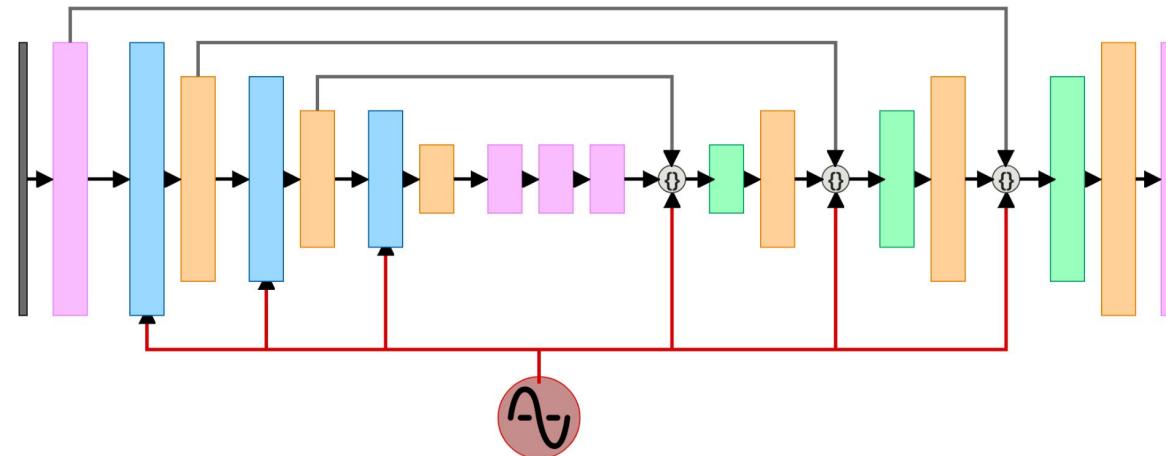
# Unet denoiser

Input  
Conv Block  
Downsample Block  
Self-Attention Block  
Upsample Block  
Embedding vector



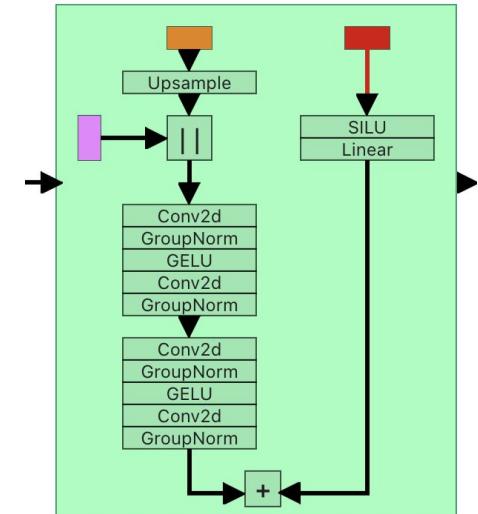
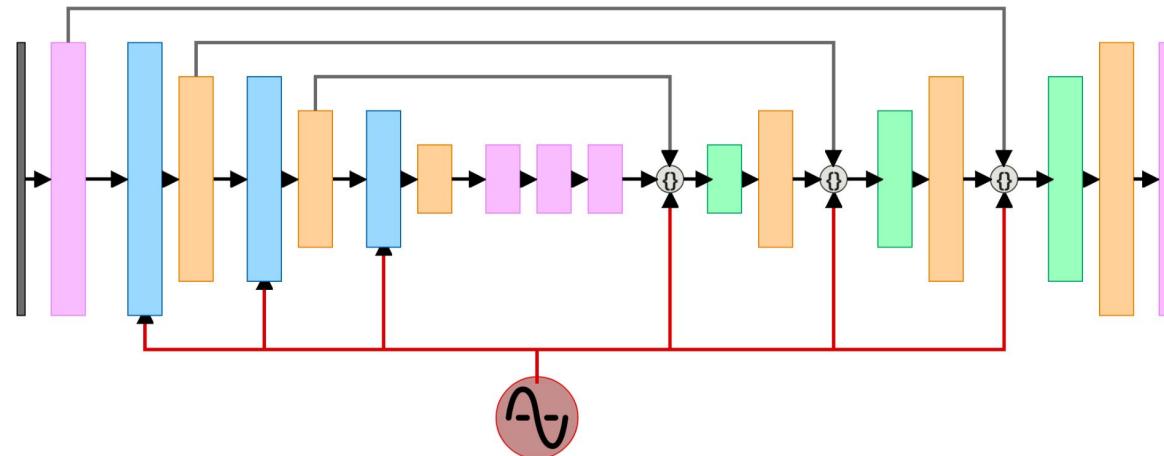
# Unet denoiser

Input  
Conv Block  
Downsample Block  
Self-Attention Block  
Upsample Block  
Embedding vector



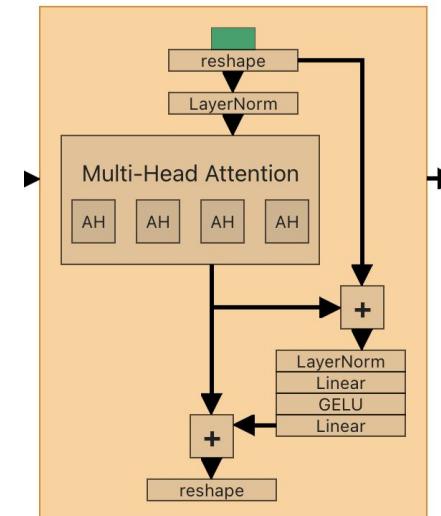
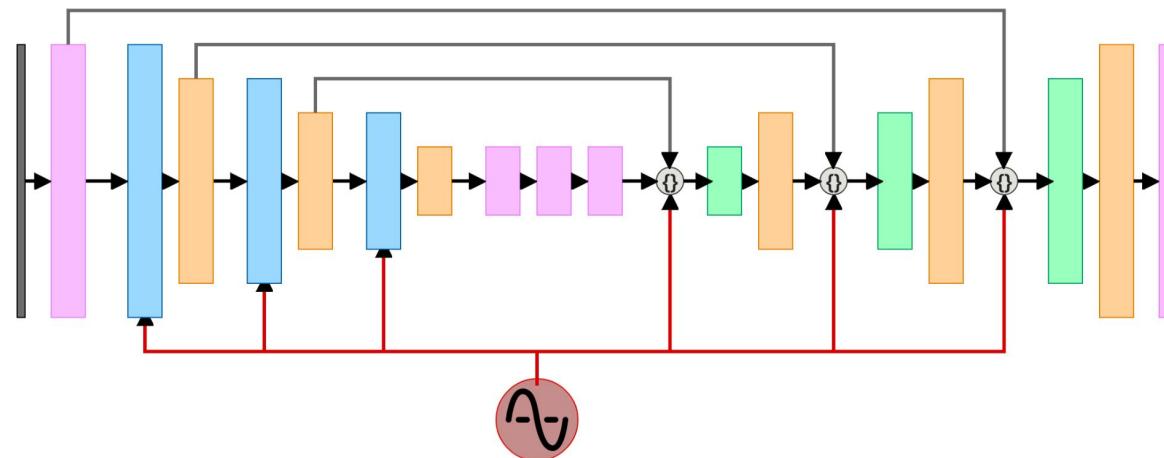
# Unet denoiser

Input  
Conv Block  
Downsample Block  
Self-Attention Block  
Upsample Block  
Embedding vector

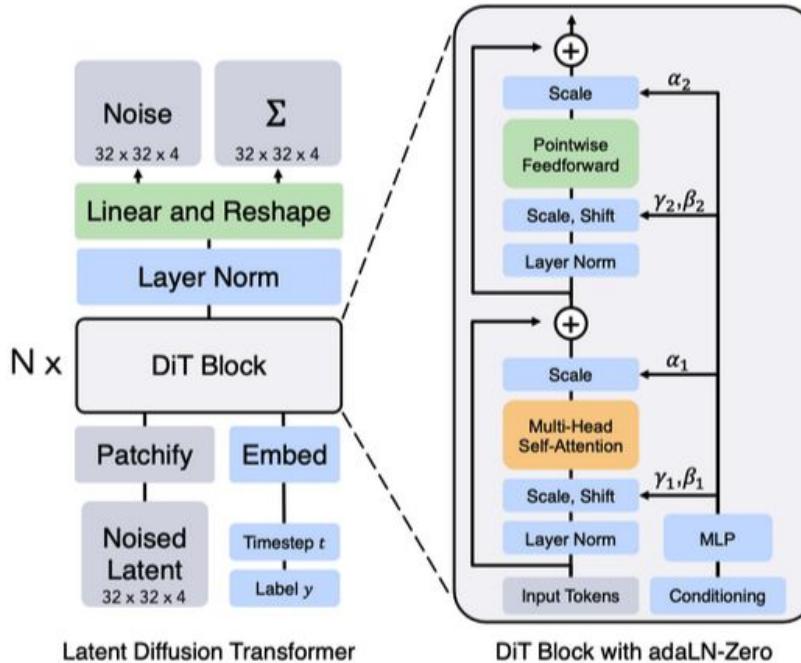


# Unet denoiser

Input  
Conv Block  
Downsample Block  
Self-Attention Block  
Upsample Block  
Embedding vector



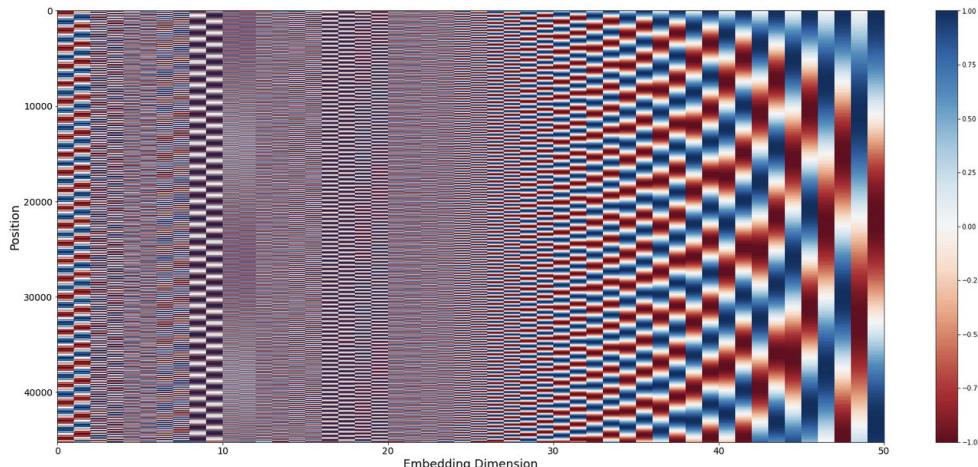
# Diffusion transformer



# Timestep embedding

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$



Generated with : [Tensorflow - Positional encoding code](#)

- **Uniqueness:** allow the model to distinguish different noise levels.
- **Smoothness:** The encoding varies smoothly as the timestep changes, which helps the model interpolate between noise levels.
- **Scale invariance:** Using multiple frequency components (through different powers of 10000) allows the encoding to capture both fine and coarse-grained timestep differences.
- **Fixed bounds:** keep the encoded values bounded between -1 and 1, which helps with numerical stability.

# Conditioning

- Class (e.g. dog)



- Prompt  
(e.g. an astronaut on a horse)



- Other, depth maps, edges, other image...



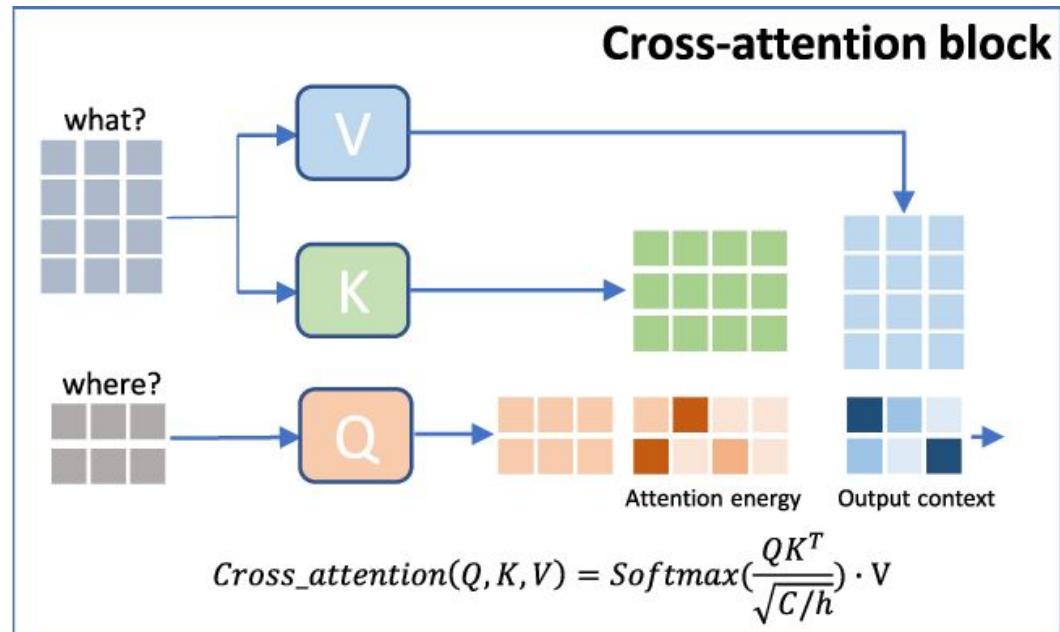
# Class Conditioning

Usually just an embedding summed or concatenated to the timestep embedding or the noisy input

In practice mostly summed to the timestep embedding

# Other types of conditioning

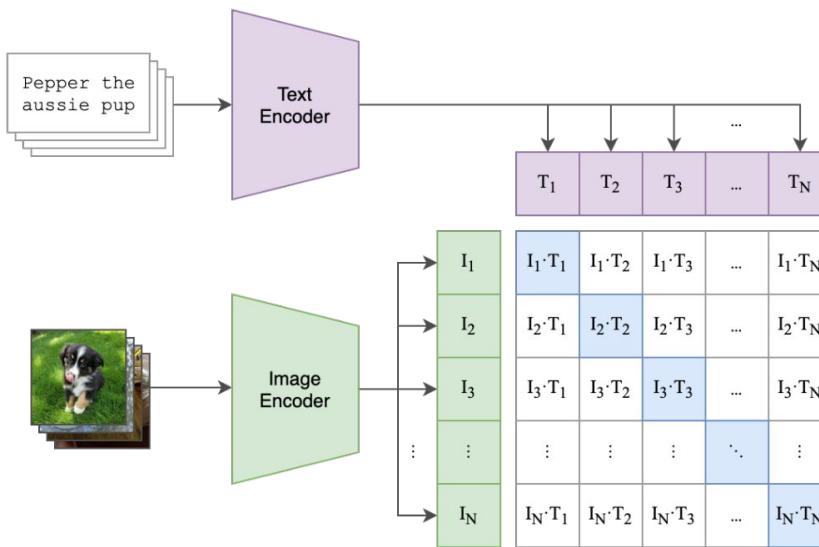
More complex conditioning (i.e. text, images, depth maps...) usually pass through cross attention blocks



# Text conditioning

- Rich text representation space: CLIP, T5, some LLMs now...

(1) Contrastive pre-training



# Cascade networks

- most of the modeling capacity can be dedicated to low resolutions
- individual models can be trained independently with specific architecture choices
- Lower resolution image conditioning is provided by channelwise concatenation of the low resolution image, processed by bilinear or bicubic upsampling to the desired resolution, with the reverse process input  $x_t$
- Apply additional data augmentation on the hi-res training

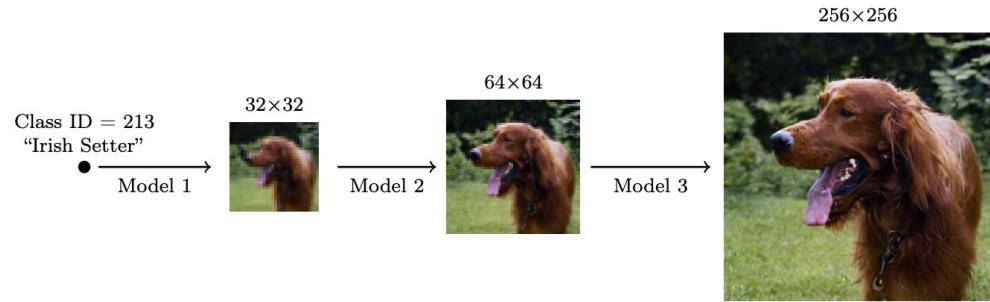
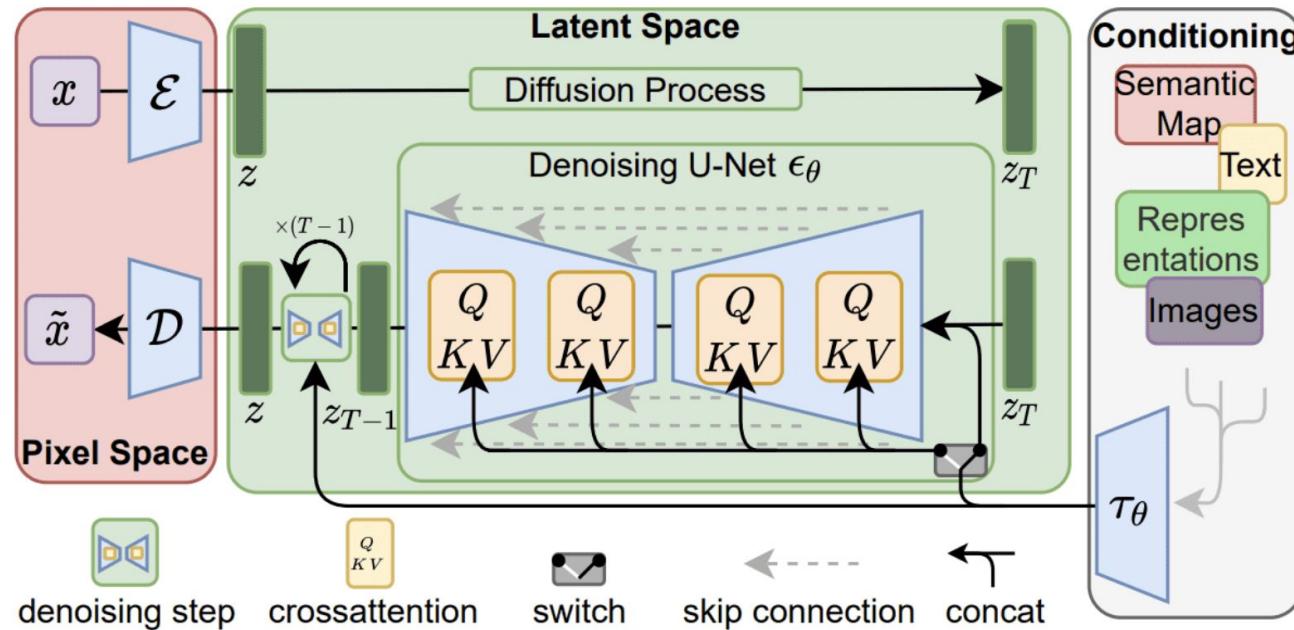


Figure 1: A cascaded diffusion model comprising a base model and two super-resolution models.

# Latent diffusion



# Recap

- **Forward process** to add noise to the input sample using a noise schedule
- **Reverse process** remove small amount of noise at every step
- A **neural network** predicts the noise to be removed to move towards  $X_0$  (not from  $X_t$  to  $X_{t-1}$ !)
- Usually a **U-Net or a DiT**
- **Time** is projected using **sinusoidal encoding**
- Can be **conditioned** with label, text, images, ...
- Recent architectures work in a **latent space**

# Score based models

# DDIM

# Guidance

Sometimes conditioning is just not enough to generate samples that really respect conditioning.

Or sometimes we did not train with conditioning but would want to condition at sampling.

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

# Classifier Guidance

Score based perspective, we aim to learn

$$\begin{aligned}\nabla_{x_t} \log p_\theta(x_t|y) &= \nabla_{x_t} \log\left(\frac{p_\theta(y|x_t)p_\theta(x_t)}{p_\theta(y)}\right) \quad (\text{Bayes rule}) \\ &= \nabla_{x_t} \log p_\theta(x_t) + \nabla_{x_t} \log(p_\theta(y|x_t))\end{aligned}$$

Train a classifier  $f_\phi(y|x_t, t)$

Use its gradients at sampling:

$$\nabla \log p_\theta(x_t|y) = \nabla \log p_\theta(x_t) + \gamma \cdot \nabla \log(p_\theta(y|x_t))$$

# Classifier 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  $s$ 
 $x_T \leftarrow$  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  $x_0$ 
```

---

**Algorithm 2** Classifier guided DDIM sampling, given a diffusion model  $\epsilon_\theta(x_t)$ , classifier  $p_\phi(y|x_t)$ , 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
     $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1 - \bar{\alpha}_t} \nabla_{x_t} \log p_\phi(y|x_t)$ 
     $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left( \frac{x_t - \sqrt{1 - \bar{\alpha}_t} \hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1 - \bar{\alpha}_{t-1}} \hat{\epsilon}$ 
end for
return  $x_0$ 
```

---

# Classifier guidance



Figure 3: Samples from an unconditional diffusion model with classifier guidance to condition on the class "Pembroke Welsh corgi". Using classifier scale 1.0 (left; FID: 33.0) does not produce convincing samples in this class, whereas classifier scale 10.0 (right; FID: 12.0) produces much more class-consistent images.

# GLIDE

Similar to classifier guidance but with text conditioning:

- Fine tune a CLIP model on noisy images
- Compute the prompt CLIP embeddings
- Compute the noisy images CLIP embeddings
- Use the gradient of the dot product between the two embeddings for guidance

# Classifier guidance

- Allows to condition generation even if the model was not trained with conditioning.
- Helps steering generation for a model towards conditioning.
- Often requires training an auxiliary network on noisy inputs
- In for latent diffusion models may require decoding the latents
- Gradients of the conditioner may yield arbitrary or adversarial directions

# Classifier free guidance

Bayes rule:

$$p(y|x) = \frac{p(x|y) \cdot p(y)}{p(x)}$$
$$\log p(y|x) = \log p(x|y) + \log p(y) - \log p(x)$$
$$\nabla_x \log p(y|x) = \nabla_x \log p(x|y) - \nabla_x \log p(x).$$

Replace in the classifier guidance:

$$\begin{aligned}\nabla_x \log p_\gamma(x|y) &= \nabla_x \log p(x) + \gamma(\nabla_x \log p(x|y) - \nabla_x \log p(x)) \\ &= (1 - \gamma)\nabla_x \log p(x) + \gamma\nabla_x \log p(x|y).\end{aligned}$$

# Classifier free guidance

- Train a conditioned diffusion model
- Drop conditioning 10-20% of the time (replace with special input)
- Model can now generate with and without conditioning
- Use CFG at sampling



Without CFG

With CFG

# Continual learning

We would like to add new concepts to our model.

-> Fine tune our general/foundation



OK if we have lots of examples of this concept and an important dataset with other concepts

If not, it leads to overfitting and catastrophic forgetting

# Textual inversion

- Few samples with the new concept to learn.
- Try avoiding catastrophic forgetting
- Being able to make compositions with concepts previously learnt

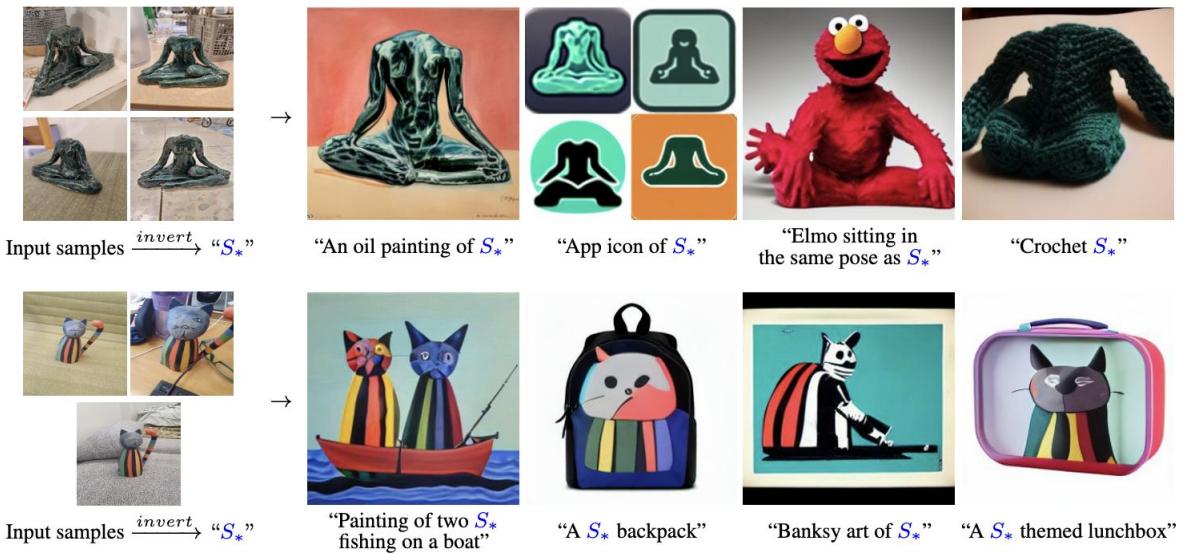


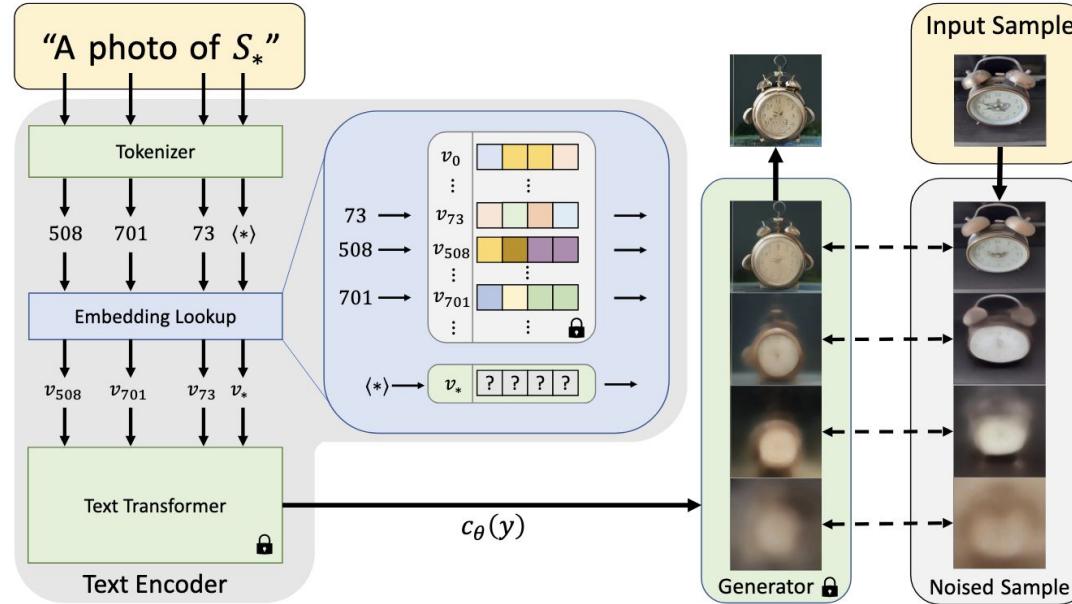
Figure 1: (left) We find new pseudo-words in the embedding space of a pre-trained text-to-image model which describe specific concepts. (right) These pseudo-words can be composed into new sentences, placing our targets in new scenes, changing their style or composition, or ingraining them into new products.

# Textual inversion

## Principle:

- Avoid catastrophic forgetting by freezing all networks
- Create a new token and its corresponding embeddings to represent the new concept
- Using the concept samples, optimize the embeddings such that when provided to the denoiser, generated images are close to the original samples.

# Textual inversion



**Training objective:**

$$v_* = \arg \min_v \mathbb{E}_{z \sim \mathcal{E}(x), y, \epsilon \sim \mathcal{N}(0,1), t} [\|\epsilon - \epsilon_\theta(z_t, t, c_\theta(y))\|_2^2]$$

# Textual inversion

## Image variations:



# Textual inversion

Compositions with previous concepts:



Input samples



"A mosaic depicting  $S_*$ "

"Death metal album cover featuring  $S_*$ "

"Masterful oil painting of  $S_*$  hanging on the wall"

"An artist drawing a  $S_*$ "



Input samples



"A photo of  $S_*$  full of cashew nuts"

"A mouse using  $S_*$  as a boat"

"A photo of a  $S_*$  mask"

"Ramen soup served in  $S_*$ "

# Textual inversion

Style transfer:



# Textual inversion

New concept composition:



$S_{style}$



$S_{clock}$



$S_{cat}$



$S_{craft}$



"Photo of  $S_{clock}$   
in the style of  $S_{style}$ "



"Photo of  $S_{cat}$   
in the style of  $S_{style}$ "



"Photo of  $S_{craft}$   
in the style of  $S_{style}$ "



"Photo of  $S_{clock}$   
in the style of  $S_{cat}$ "



"Photo of  $S_{clock}$   
in the style of  $S_{craft}$ "



"Photo of  $S_{cat}$   
in the style of  $S_{craft}$ "

# Textual inversion

Debiasing concept:



“A stock photo of a doctor” (Base model)



“A photo of  $S_*$ ” (Ours)

# Dreambooth

## Principle:

- Choose a token to describe the new concept
- Fine-tune the denoiser
- Regularize with a “*class-specific prior preservation loss*”



Input images



in the Acropolis



swimming



sleeping



in a bucket



getting a haircut

# Dreambooth

## Choosing a token:

- Existing corresponding token (e.g. "dog")  
=> difficult to disentangle the original meaning of a token and reantangle it to reference the new subject
- Random sequences (e.g. "husiusl") will be separated by the tokenizer in several tokens with a strong prior (e.g. [h, usi, us, l])
- Better to use very rare tokens



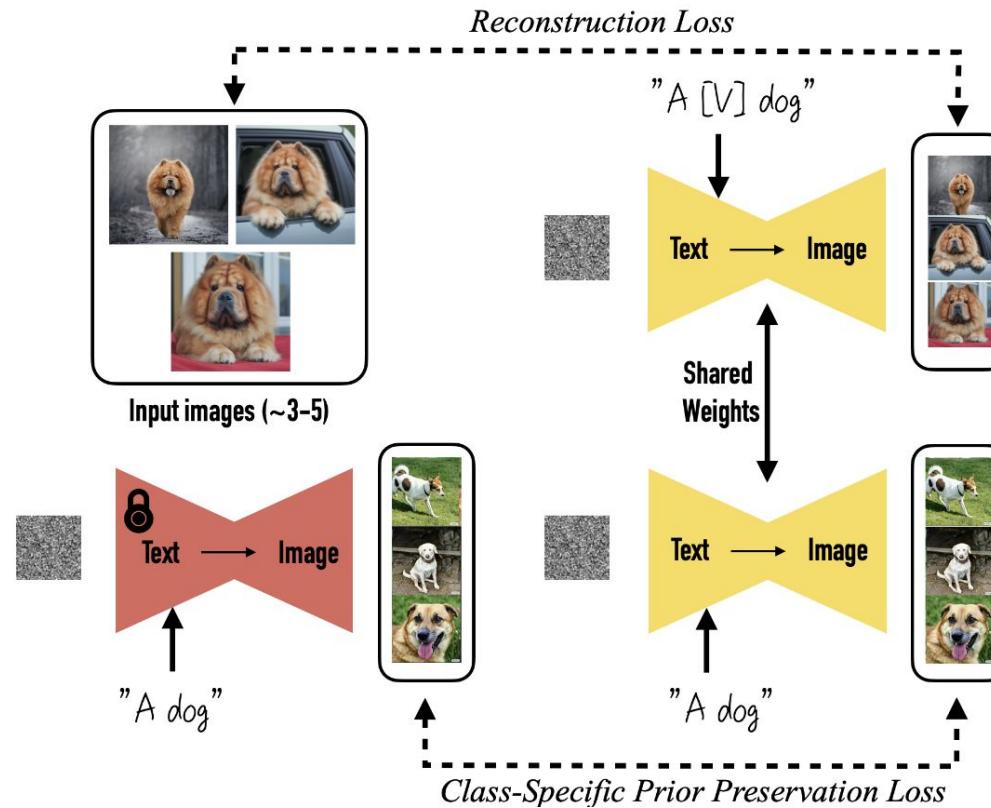
# Dreambooth

## Prevent language drift:

- Fine-tune with text prompts containing a unique identifier followed by the class name of the subject (i.e. "A [V] dog")  
=> class name let the model use prior knowledge on the subject class.
- Regularize to avoid the model to associate the class name with specific concept/instance  
=> ***class-specific prior preservation loss***



# Dreambooth



# Dreambooth

## **Some issues with Dreambooth-like methods:**

- Methods like Dreambooth “overfit” a given concept
- Hurts your model for generating diverse images with other concepts
- Need to save several full copies of the denoiser for each new concept
- Fine-tuning the denoiser requires lot of VRAM

# Low Rank Adaptation (LoRA)

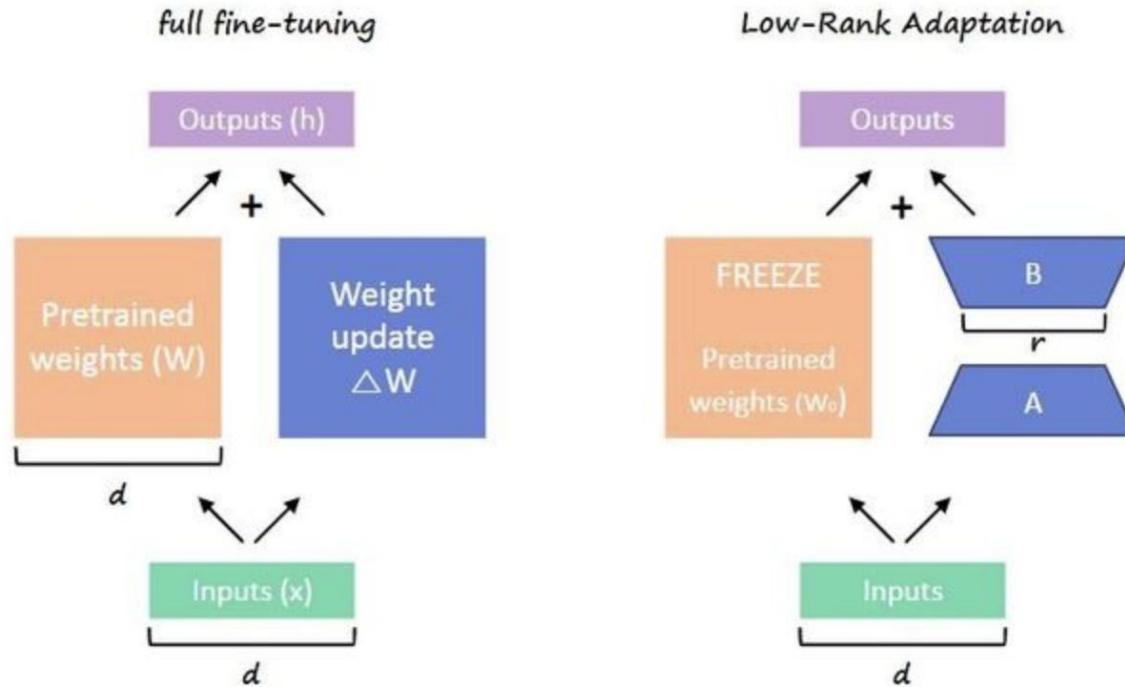
## Intuition:

- Over-parameterized model actually reside in low intrinsic dimension
- Changes in weights during the model adaptation has low intrinsic rank.

## Principle:

- Fine-tuning indirectly by optimizing rank decomposition matrices of the layer changes while freezing the original weights.

# Low Rank Adaptation (LORA)



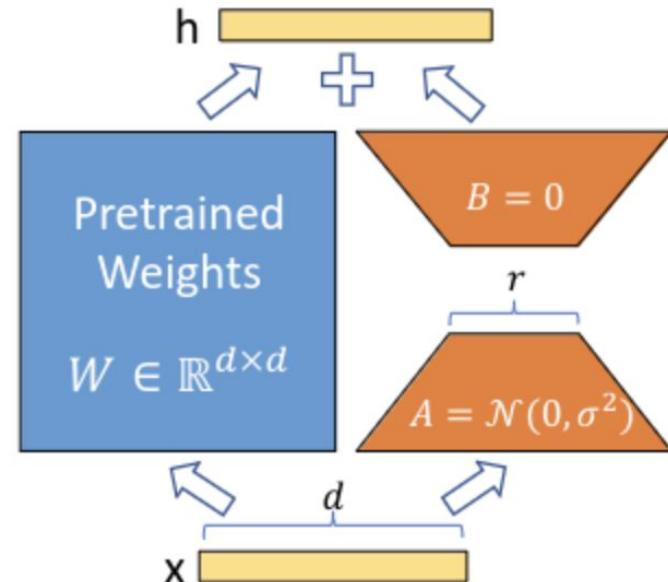
source: <https://tech.hancom.com/2024-05-24-lm/>

LoRA: Low-Rank Adaptation of Large Language Models: Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen

# Low Rank Adaptation (LoRA)

## Training tricks

- A is randomly initialized with small values drawn from a Gaussian and B is initialized with 0s to make sure that the first model inference is not perturbed by LoRA
- Provide good results when applied on Q and K matrices only

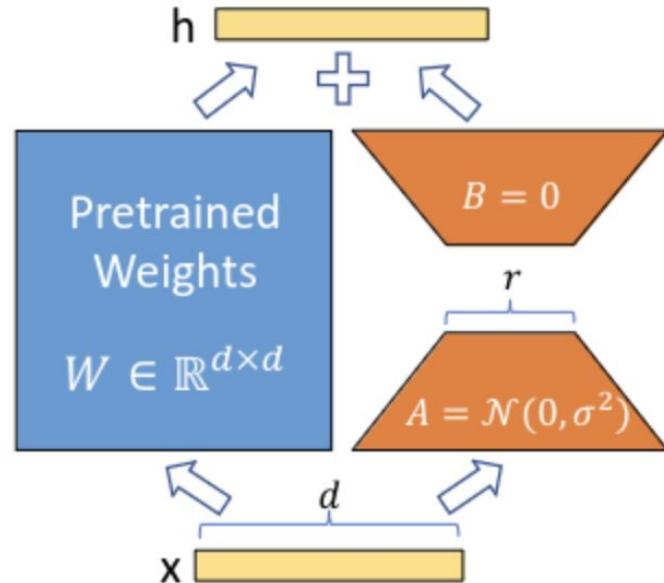


# Low Rank Adaptation (LORA)

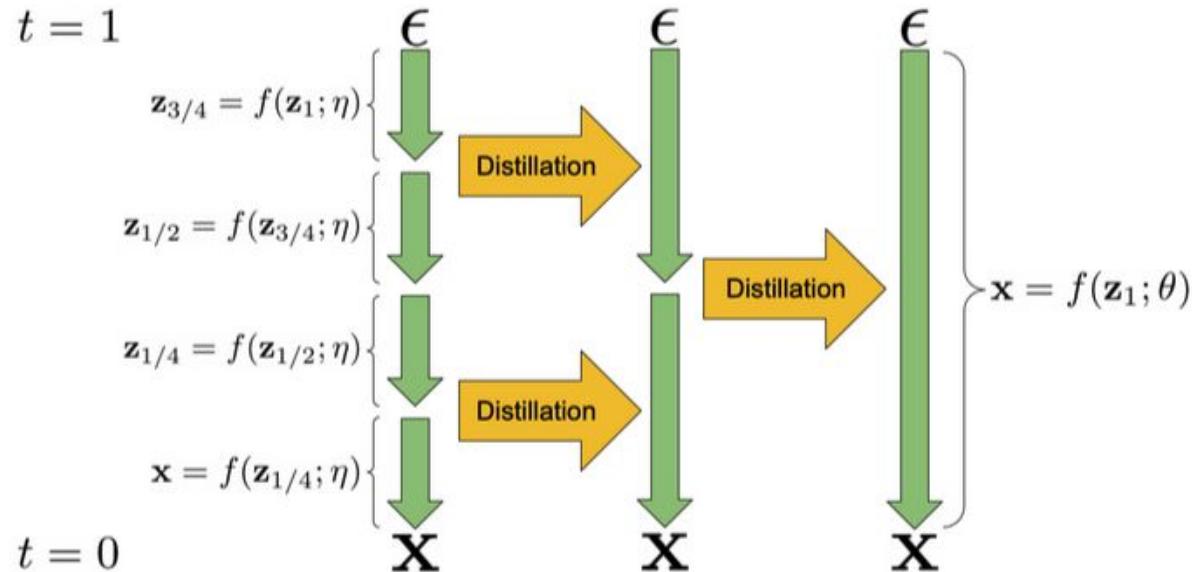
## Benefits:

- Almost as efficient as regular fine-tuning
- Lower VRAM impact
- Impact of the LORA weights can be tuned at inference
- Very small space on disk
- Several LORAs can be combined.

## Examples



# Progressive distillation



# Progressive distillation

---

**Algorithm 1** Standard diffusion training

---

**Require:** Model  $\hat{x}_\theta(z_t)$  to be trained  
**Require:** Data set  $\mathcal{D}$   
**Require:** Loss weight function  $w()$

```

while not converged do
     $x \sim \mathcal{D}$                                  $\triangleright$  Sample data
     $t \sim U[0, 1]$                              $\triangleright$  Sample time
     $\epsilon \sim N(0, I)$                            $\triangleright$  Sample noise
     $z_t = \alpha_t x + \sigma_t \epsilon$        $\triangleright$  Add noise to data

     $\tilde{x} = x$                                  $\triangleright$  Clean data is target for  $\hat{x}$ 
     $\lambda_t = \log[\alpha_t^2 / \sigma_t^2]$            $\triangleright$  log-SNR
     $L_\theta = w(\lambda_t) \|\tilde{x} - \hat{x}_\theta(z_t)\|_2^2$      $\triangleright$  Loss
     $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$          $\triangleright$  Optimization
end while

```

---



---

**Algorithm 2** Progressive distillation

---

**Require:** Trained teacher model  $\hat{x}_\eta(z_t)$   
**Require:** Data set  $\mathcal{D}$   
**Require:** Loss weight function  $w()$   
**Require:** Student sampling steps  $N$

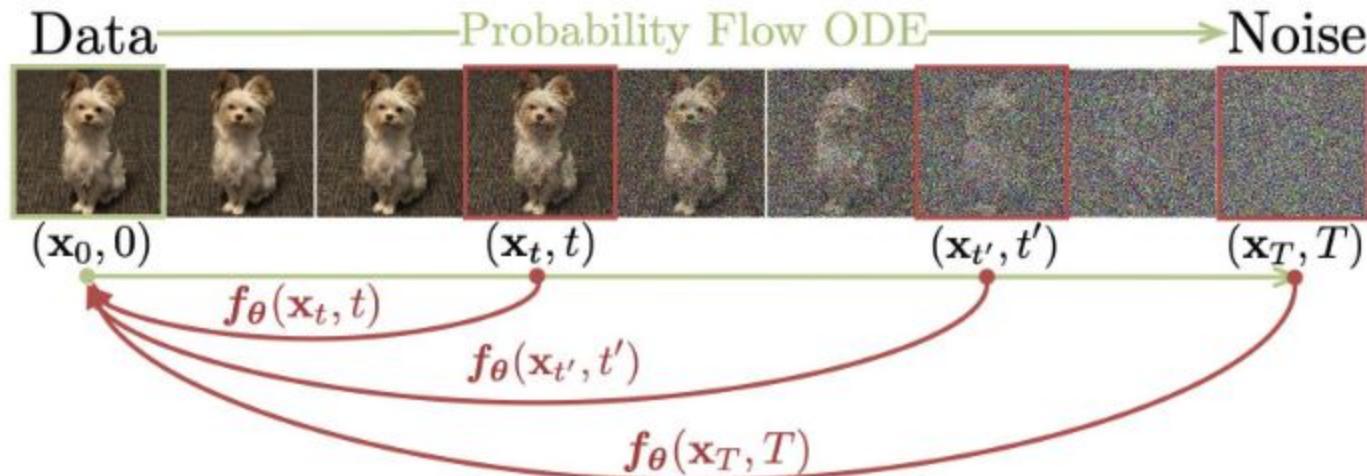
```

for  $K$  iterations do
     $\theta \leftarrow \eta$                                  $\triangleright$  Init student from teacher
    while not converged do
         $x \sim \mathcal{D}$ 
         $t = i/N, i \sim Cat[1, 2, \dots, N]$ 
         $\epsilon \sim N(0, I)$ 
         $z_t = \alpha_t x + \sigma_t \epsilon$ 
        # 2 steps of DDIM with teacher
         $t' = t - 0.5/N, t'' = t - 1/N$ 
         $z_{t'} = \alpha_{t'} \hat{x}_\eta(z_t) + \frac{\sigma_{t'}}{\sigma_t} (z_t - \alpha_t \hat{x}_\eta(z_t))$ 
         $z_{t''} = \alpha_{t''} \hat{x}_\eta(z_{t'}) + \frac{\sigma_{t''}}{\sigma_{t'}} (z_{t'} - \alpha_{t'} \hat{x}_\eta(z_{t'}))$ 
         $\tilde{x} = \frac{z_{t''} - (\sigma_{t''}/\sigma_t) z_t}{\alpha_{t''} - (\sigma_{t''}/\sigma_t) \alpha_t}$        $\triangleright$  Teacher  $\hat{x}$  target
         $\lambda_t = \log[\alpha_t^2 / \sigma_t^2]$ 
         $L_\theta = w(\lambda_t) \|\tilde{x} - \hat{x}_\theta(z_t)\|_2^2$ 
         $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$ 
    end while
     $\eta \leftarrow \theta$                                  $\triangleright$  Student becomes next teacher
     $N \leftarrow N/2$                                  $\triangleright$  Halve number of sampling steps
end for

```

---

# Consistency models



# Consistency models

---

**Algorithm 2** Consistency Distillation (CD)

---

**Input:** dataset  $\mathcal{D}$ , initial model parameter  $\theta$ , learning rate  $\eta$ , ODE solver  $\Phi(\cdot, \cdot; \phi)$ ,  $d(\cdot, \cdot)$ ,  $\lambda(\cdot)$ , and  $\mu$

$$\theta^- \leftarrow \theta$$

**repeat**

    Sample  $\mathbf{x} \sim \mathcal{D}$  and  $n \sim \mathcal{U}[1, N - 1]$

    Sample  $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}; t_{n+1}^2 \mathbf{I})$

$\hat{\mathbf{x}}_{t_n}^\phi \leftarrow \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\Phi(\mathbf{x}_{t_{n+1}}, t_{n+1}; \phi)$

$\mathcal{L}(\theta, \theta^-; \phi) \leftarrow$

$\lambda(t_n)d(\mathbf{f}_\theta(\mathbf{x}_{t_{n+1}}, t_{n+1}), \mathbf{f}_{\theta^-}(\hat{\mathbf{x}}_{t_n}^\phi, t_n))$

$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta, \theta^-; \phi)$

$\theta^- \leftarrow \text{stopgrad}(\mu \theta^- + (1 - \mu) \theta)$

**until** convergence

---

# DDIM inversion