

# Introduction to Deep Generative Modeling

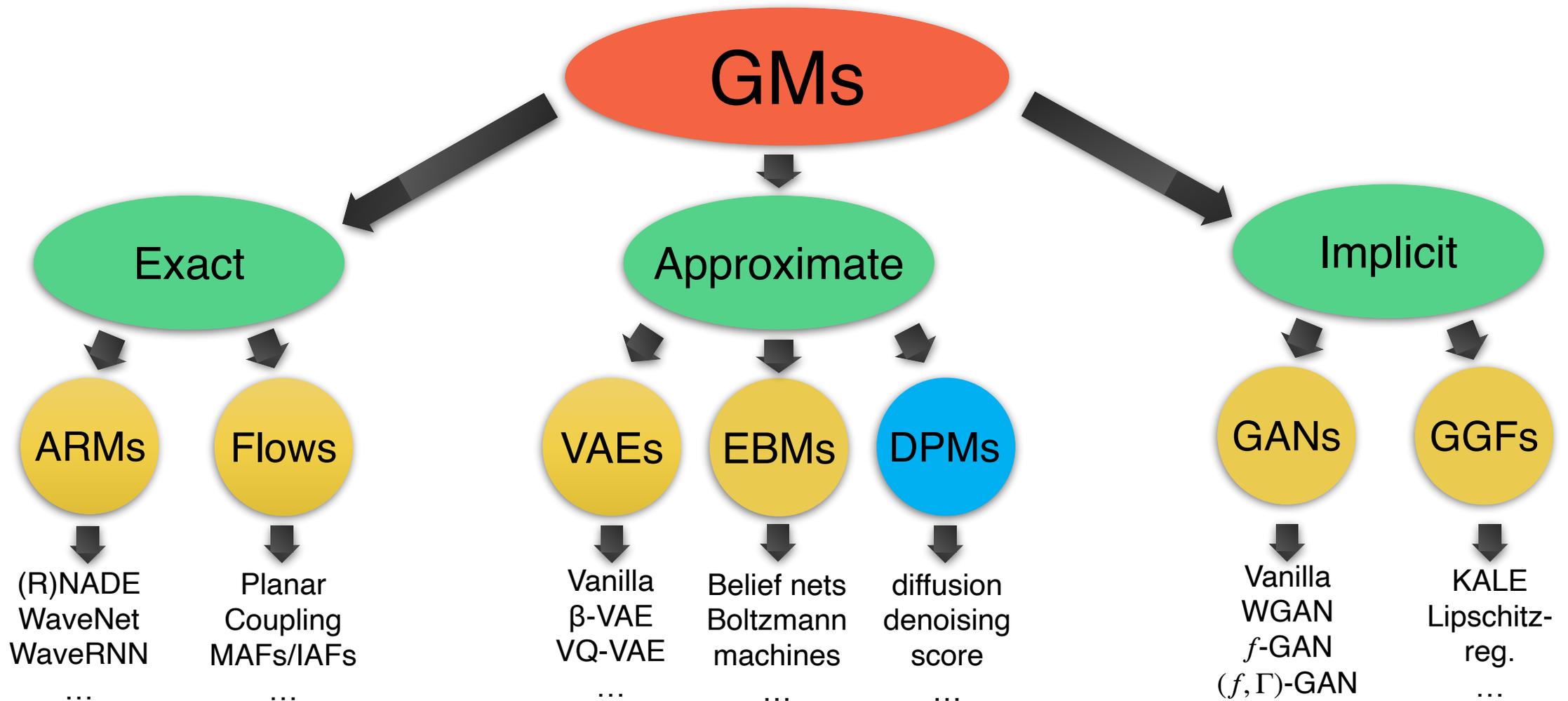
## Lecture #12

**HY-673** – Computer Science Dep., University of Crete

Professors: Yannis Pantazis, Yannis Stylianou

Teaching Assistant: Michail Raptakis

# Taxonomy of GMs



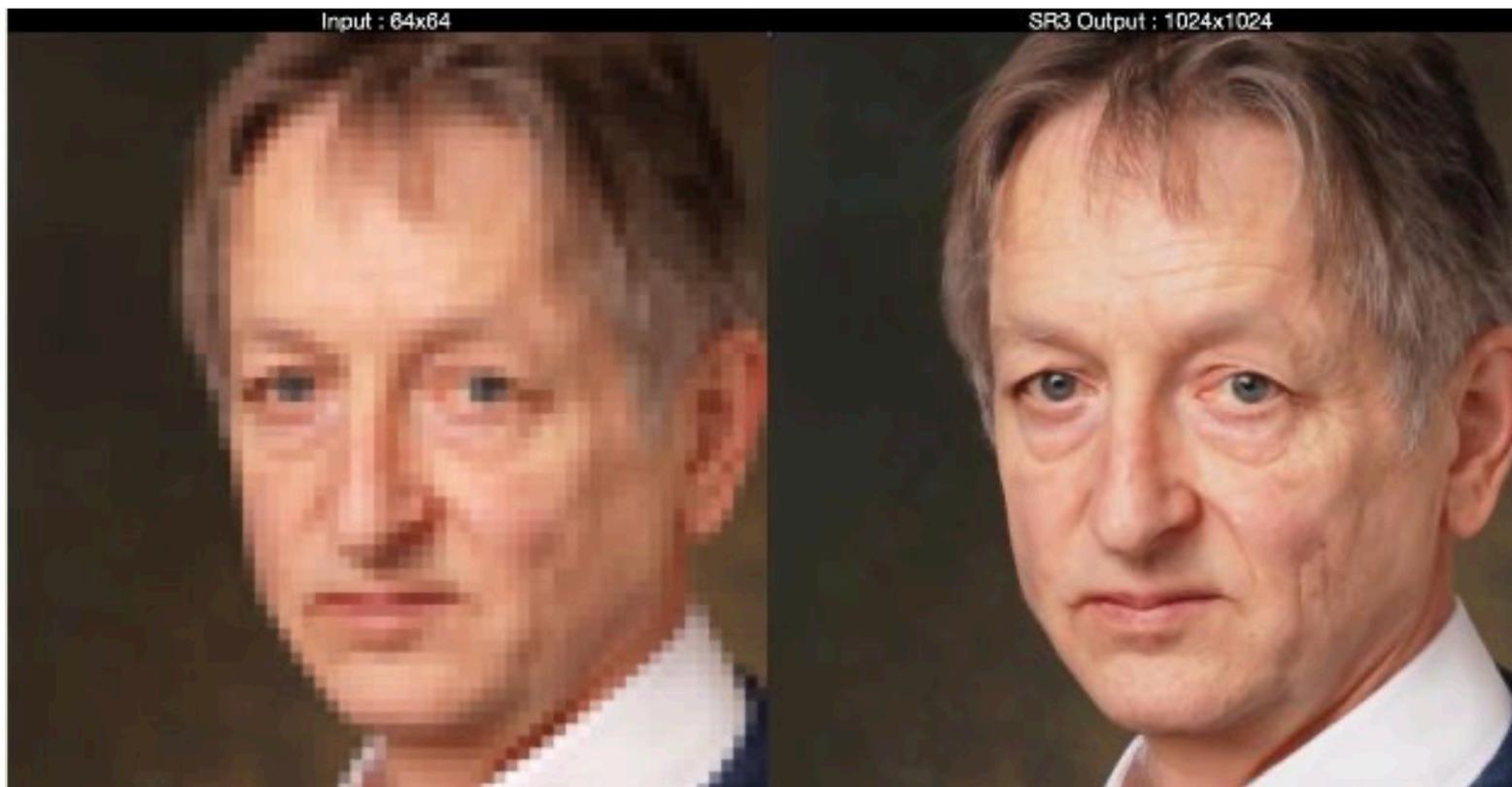
# Denoising Diffusion Models

Lecture #12

Emerging as powerful generative models, outperforming GANs

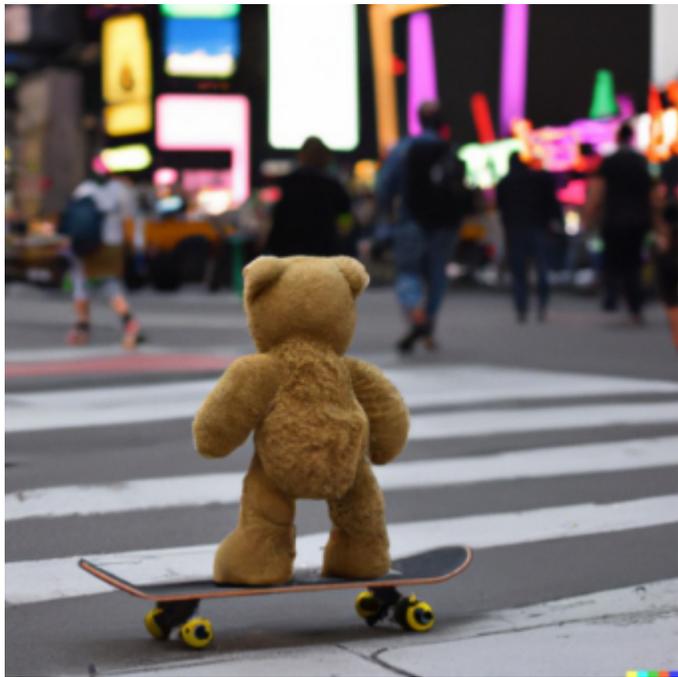


## Successful applications - SR3 Mode



## DALL·E 2

“a teddy bear on a skateboard in times square”



## Imagen

“A group of teddy bears in suit in a corporate office celebrating the birthday of their friend. There is a pizza cake on the desk”.

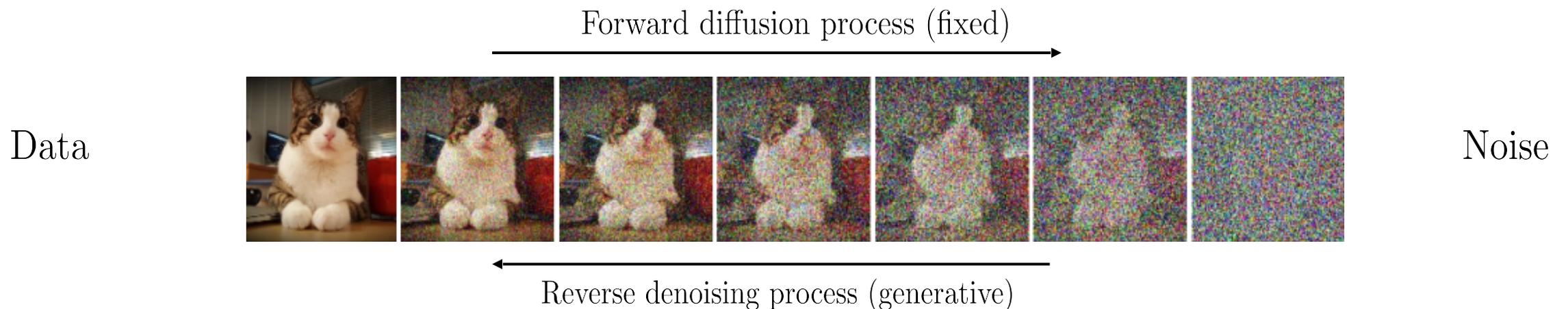


# Denoising Diffusion Models

## Learning to generate by denoising

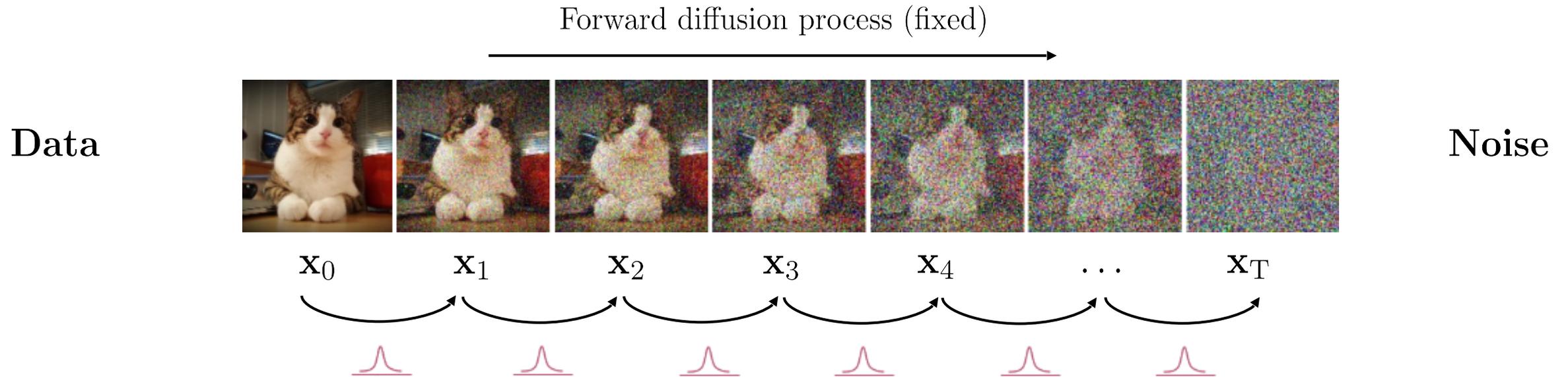
Denoising diffusion models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



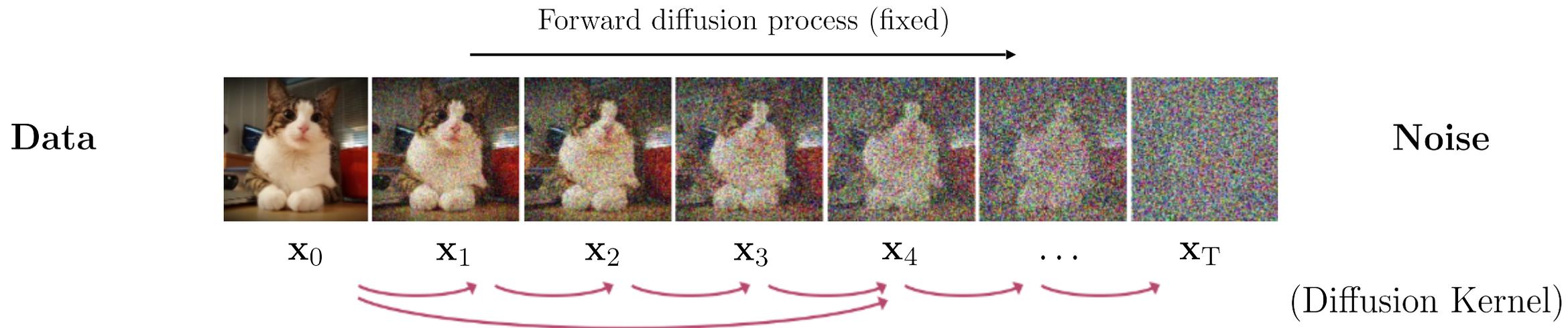
# Forward Diffusion Process

The formal definition of the forward process in  $T$  steps:



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

# Diffusion Kernel



Define  $\bar{\alpha}_t = \prod_{s=1}^t (1 - \beta_s)$   $\Rightarrow$   $q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$ .

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

$\beta_t$  values schedule (i.e., the noise schedule) is designed such that  $\bar{\alpha}_T \rightarrow 0$  and  $q(\mathbf{x}_T | \mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

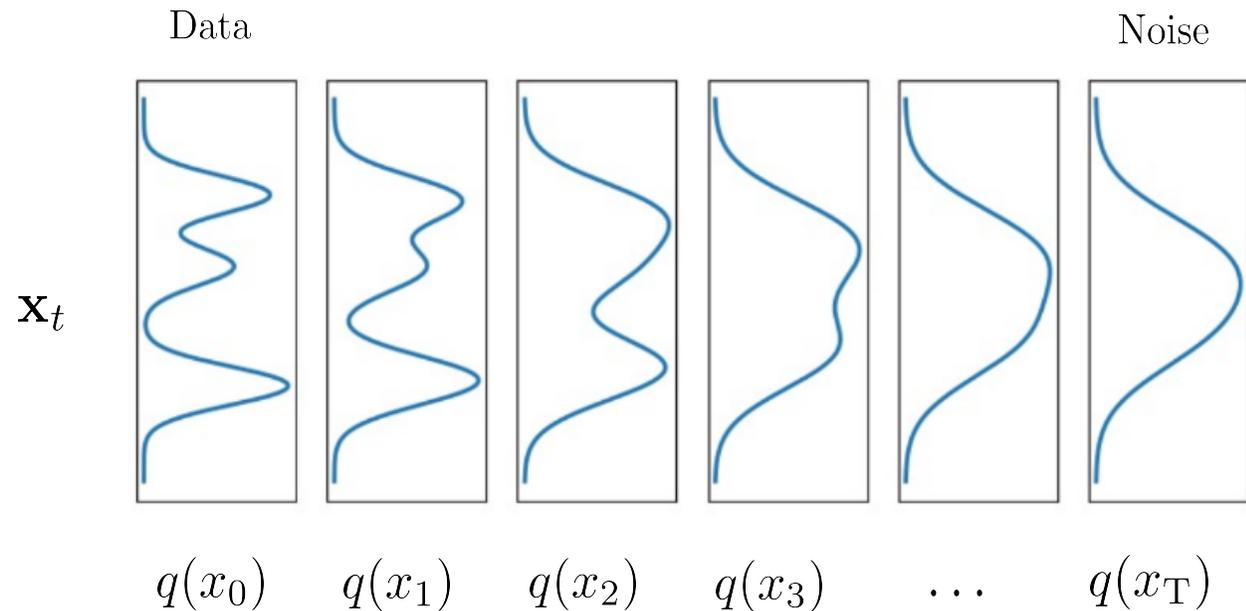
# What happens to a distribution in the forward diffusion?

So far, we discussed the diffusion kernel  $q(\mathbf{x}_t|\mathbf{x}_0)$  but what about  $q(\mathbf{x}_t)$  ?

$$\underbrace{q(\mathbf{x}_t)}_{\text{Diffused data dist.}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\text{Joint dist.}} d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\text{Input data dist.}} \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\text{Diffusion kernel}} d\mathbf{x}_0.$$

The diffusion kernel is Gaussian convolution.

## Diffused Data Distributions



We can sample  $\mathbf{x}_t \sim q(\mathbf{x}_t)$  by first sampling  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$  and then sampling  $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$  (i.e., ancestral sampling).

# Generative Learning by Denoising

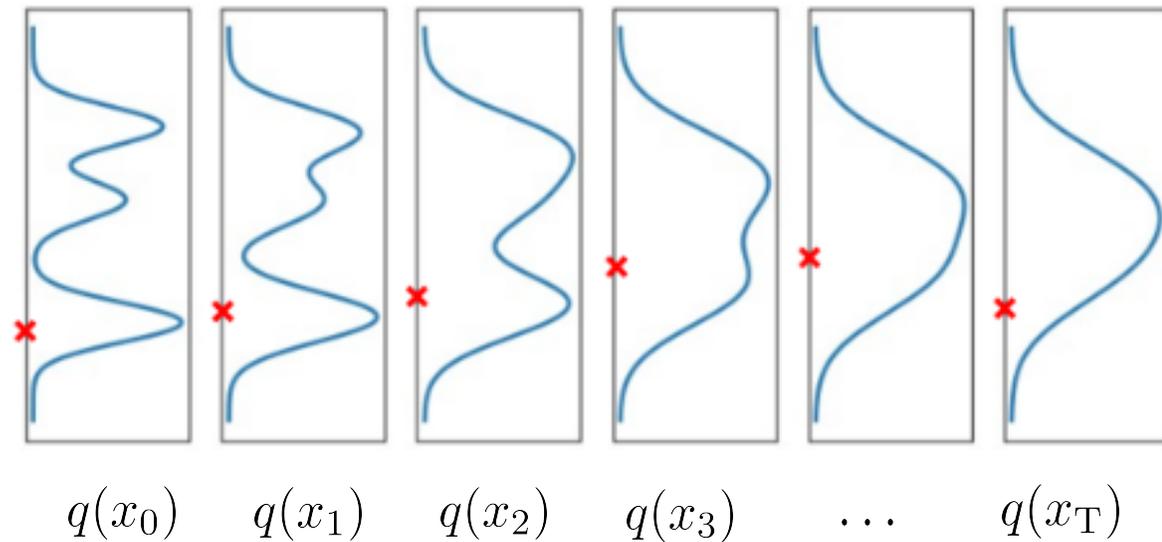
Recall, that the diffusion parameters are designed such that  $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

## Generation:

Sample  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Iteratively sample  $\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1} | \mathbf{x}_t)}_{\text{True Denoising Dist.}}^{\mathbf{x}_t}$

## Diffused Data Distributions

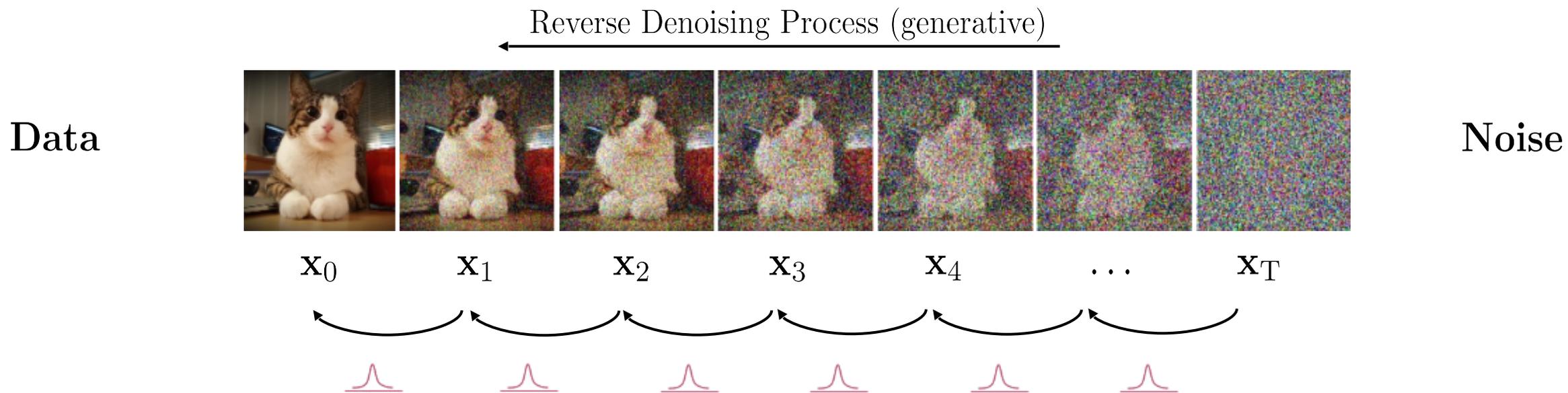


In general,  $q(\mathbf{x}_{t-1} | \mathbf{x}_t) \propto q(\mathbf{x}_t | \mathbf{x}_{t-1})$  is intractable.

Can we approximate  $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$ ? Yes, we can use a **Normal distribution** if  $\beta_t$  is small in each forward diffusion step.

# Reverse Denoising Process

The formal definition of the reverse process in  $T$  steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu(\mathbf{x}_t, t)}_{\text{Trainable network}}, \sigma_t^2 \mathbf{I})$$

Trainable network  
(U-net, Denoising Autoencoder)



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

# Learning Denoising Model

For training, we can form variational upper bound that is commonly used for training variational autoencoders:

$$\mathbb{E}_{q(\mathbf{x}_0)} \left[ -\log p_\theta(\mathbf{x}_0) \right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ -\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] =: L.$$

Sohl-Dickstein et al. ICML 2015 and Ho et al. NeurIPs 2020 show that:

$$L = \mathbb{E}_q \left[ \underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_\theta} \right].$$

where  $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  is the tractable posterior distribution:

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

$$\text{where } \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \mathbf{x}_0 + \frac{\sqrt{1-\beta_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \mathbf{x}_t \text{ and } \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \beta_t.$$

# Parameterizing the Denoising Model

Since both  $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  and  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$  are Normal distributions, the KL divergence has a simple form:

$$L_{t-1} = D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{t}, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) = \mathbb{E}_q \left[ \frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

Recall that  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{(1 - \bar{\alpha}_t)}\epsilon$ . Ho et al. NeurIPS 2020 observe that:

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{1 - \beta_t}} \left( \mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right)$$

They propose to represent the mean of the denoising model using a noise-prediction network:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1 - \beta_t}} \left( \mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right)$$

With this parameterization:

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \frac{\beta_t^2}{2\sigma_t^2(1 - \beta_t)(1 - \bar{\alpha}_t)} \|\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon}_{\mathbf{x}_t}, t)\|^2 \right] + C.$$

# Training Objective Weighting

## Trading likelihood for perceptual quality

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \underbrace{\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}}_{\lambda_t} \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2 \right].$$

The time dependent  $\lambda_t$  ensures that the training objective is weighted properly for the maximum data likelihood training.

However, this weight is often very large for small  $t$ 's.

Ho et al. NeurIPS 2020 observe that simply setting  $\lambda_t = 1$  improves sample quality. So, they propose to use:

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1, T)} \left[ \|\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon}_{\mathbf{x}_t}, t)\|^2 \right].$$

For more advanced weighting see Choi et al., Perception Prioritized Training of Diffusion Models, CVPR 2022 (<https://arxiv.org/abs/2204.00227>)

## Training and Sample Generation

---

### Algorithm 1 Training

---

- 1: **repeat**
- 2:  $\mathbf{x}_0 \sim q(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$
- 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

---

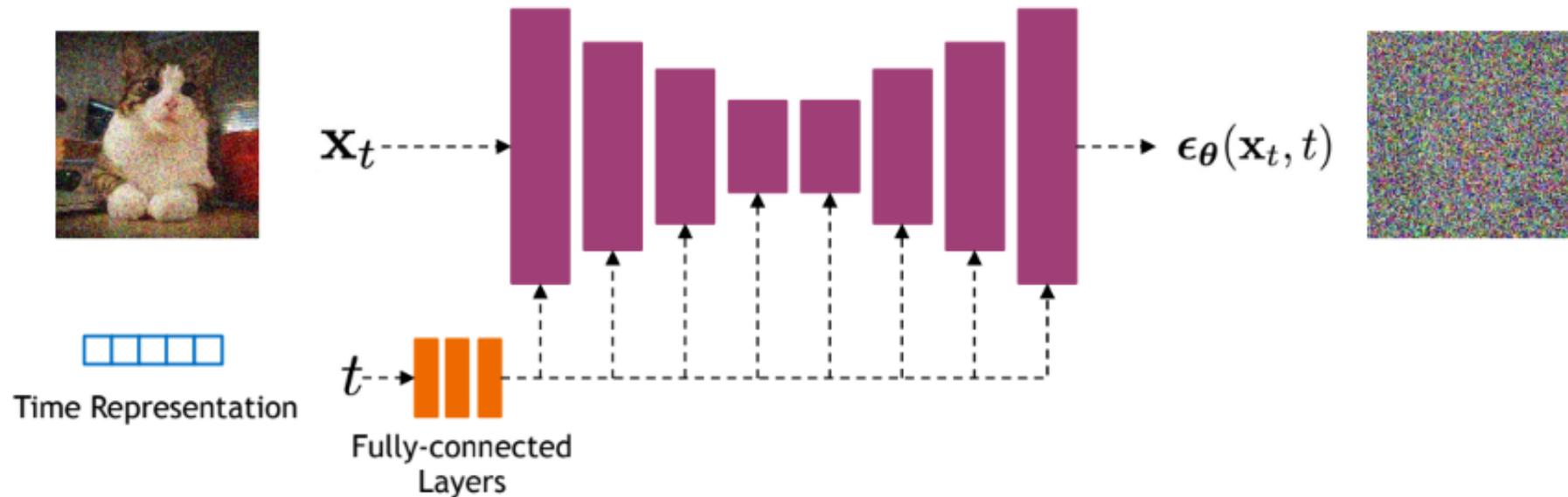
### Algorithm 2 Sampling

---

- 1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - 2: **for**  $t = T, \dots, 1$  **do**
  - 3:      $\mathbf{z} \sim \mathcal{N}$
  - 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**  $\mathbf{x}_0$
-

## Network Architectures

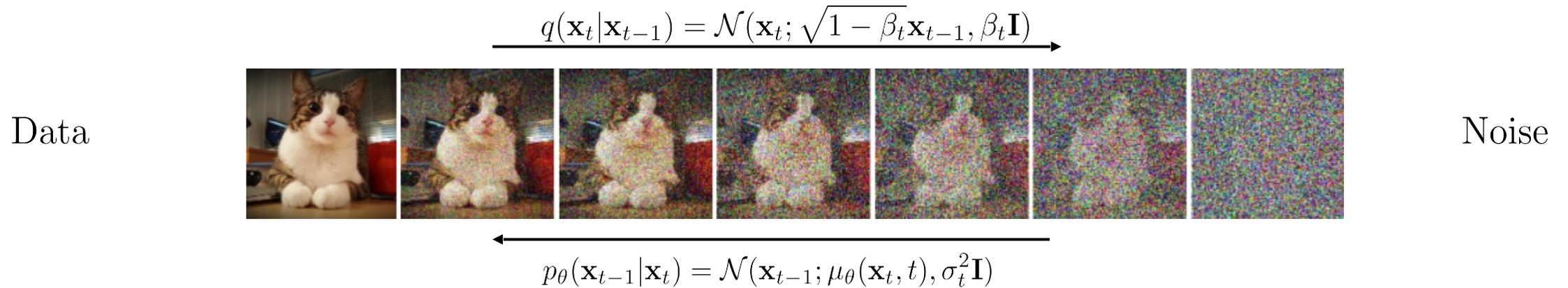
Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent  $\epsilon_{\theta}(\mathbf{x}_t, t)$



Time representation: sinusoidal positional embeddings or random Fourier features.

Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers. (see Dhariwal and Nichol NeurIPS 2021)

## Noise Schedule



Above,  $\beta_t$  and  $\sigma_t^2$  control the variance of the forward diffusion and reverse denoising processes respectively.

Often a linear schedule is used for  $\beta_t$ , and  $\sigma_t^2$  is set equal to  $\beta_t$ .

Kingma et al. NeurIPS 2022 introduce a new parameterization of diffusion models using signal-to-noise ratio (SNR), show how to learn the noise schedule by minimizing the variance of the training objective.

We can also train  $\sigma_t^2$  while training diffusion model by minimizing the variational bound (Improved DPM by Nichol and Dhariwal ICLR 2021) or after training the diffusion model (Analytic-DPM by Bao et al. ICLR 2022).

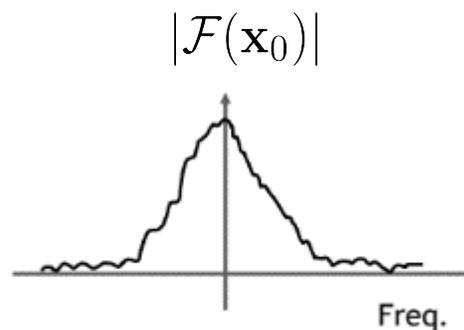
# What happens to an image in the forward diffusion process?

Recall that sampling from  $q(\mathbf{x}_t|\mathbf{x}_0)$  is done using  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{(1 - \bar{\alpha}_t)}\epsilon$  where  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

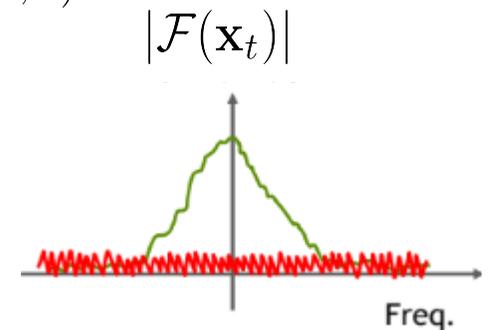
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{(1 - \bar{\alpha}_t)}\epsilon$$

Fourier Transform

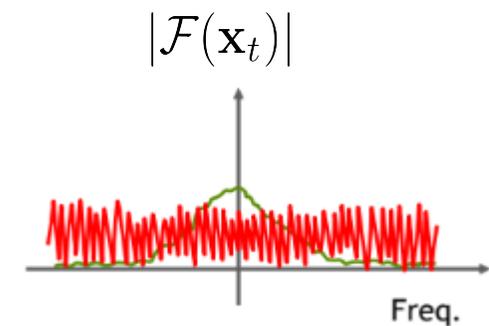
$$\mathcal{F}(\mathbf{x}_t) = \sqrt{\bar{\alpha}_t}\mathcal{F}(\mathbf{x}_0) + \sqrt{(1 - \bar{\alpha}_t)}\mathcal{F}(\epsilon)$$



Small  $t$   
 $\bar{\alpha}_t \sim 1$

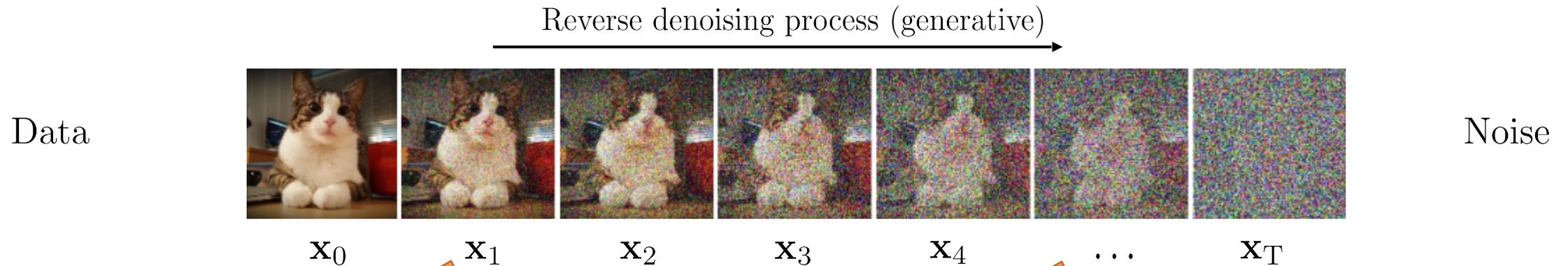


Large  $t$   
 $\bar{\alpha}_t \sim 0$



In the forward diffusion, the high frequency content is perturbed faster.

# Content-Detail Tradeoff



The denoising model is specialized for generating the high-frequency content (i.e., low-level details)

The denoising model is specialized for generating the low-frequency content (i.e., coarse content)

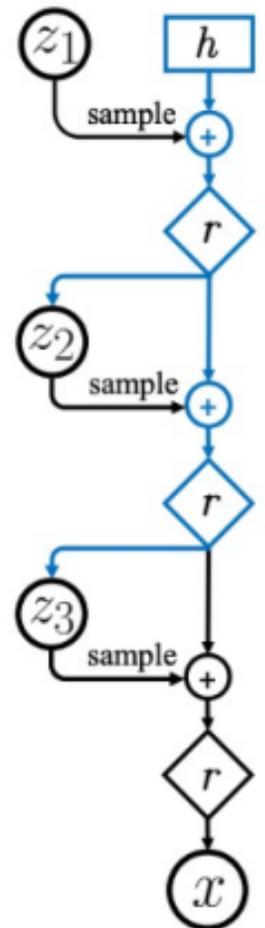
The weighting of the training objective for different timesteps is important!

# Connection to VAEs

Diffusion models can be considered as a special form of hierarchical VAEs.

However, in diffusion models:

- The encoder is fixed
- The latent variables have the same dimension as the data
- The denoising model is shared across different timestep
- The model is trained with some reweighting of the variational bound



## Denoising Diffusion Probabilistic Models

The model is trained by sampling from the forward diffusion process and training a denoising model to predict the noise.

We discussed how the forward process perturbs the data distribution or data samples.

The devil is in the details:

- Network architectures
- Objective weighting
- Diffusion parameters (i.e., noise schedule)

See “Elucidating the Design Space of Diffusion-Based Generative Models” by Karras et al. for important design decision.

# Introduction to Deep Generative Modeling

## Lecture #12

**HY-673** – Computer Science Dep., University of Crete

Professors: Yannis Pantazis, Yannis Stylianou

Teaching Assistant: Michail Raptakis