

Introduction to Deep Generative Modeling

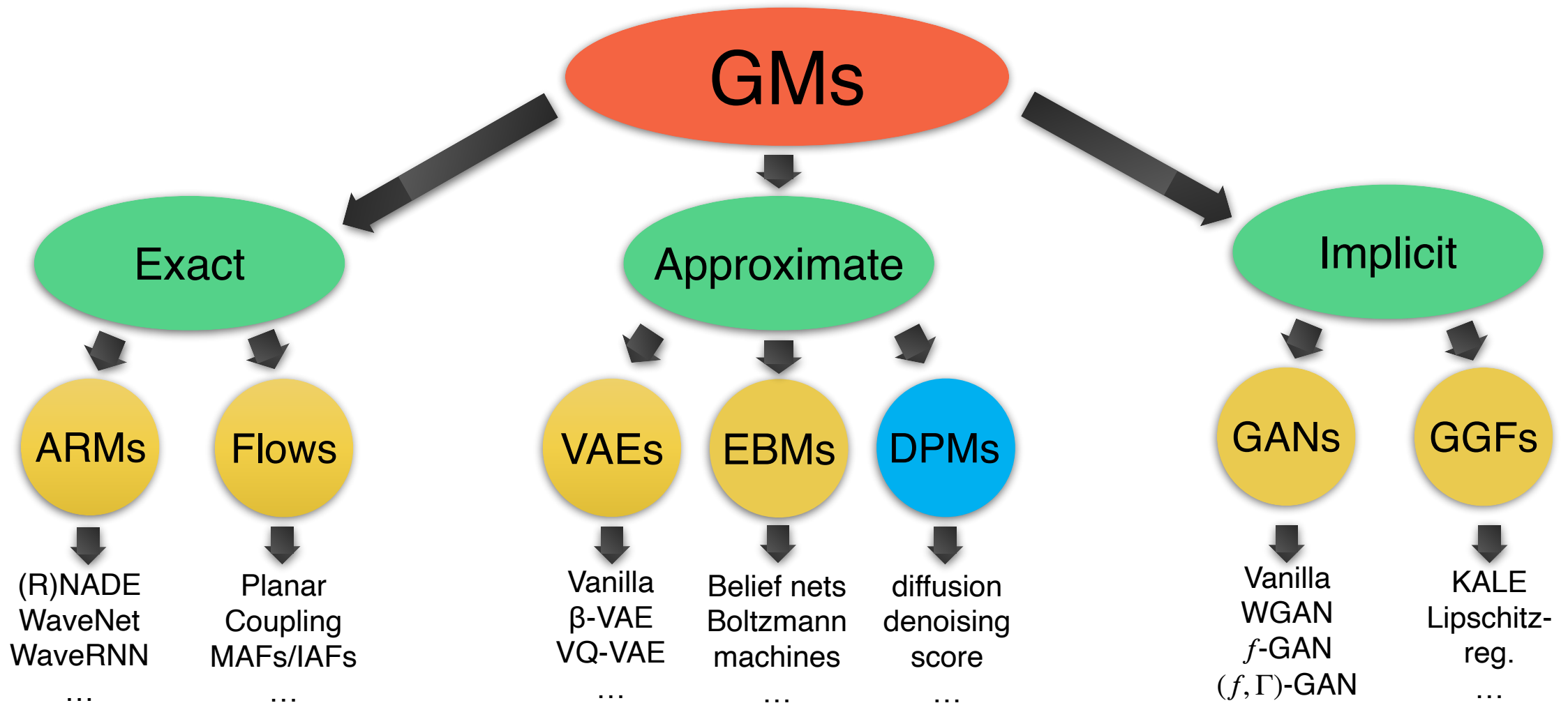
Lecture #13

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

Professors: Yannis Pantazis, Yannis Stylianou

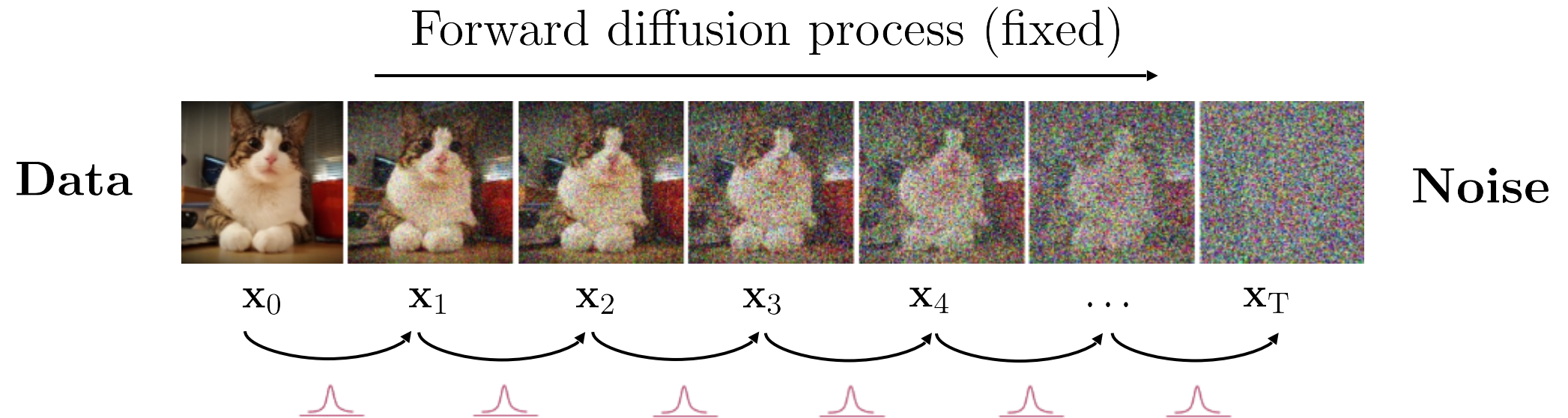
Teaching Assistant: Michail Raptakis

Taxonomy of GMs



Recap: Forward Diffusion Process

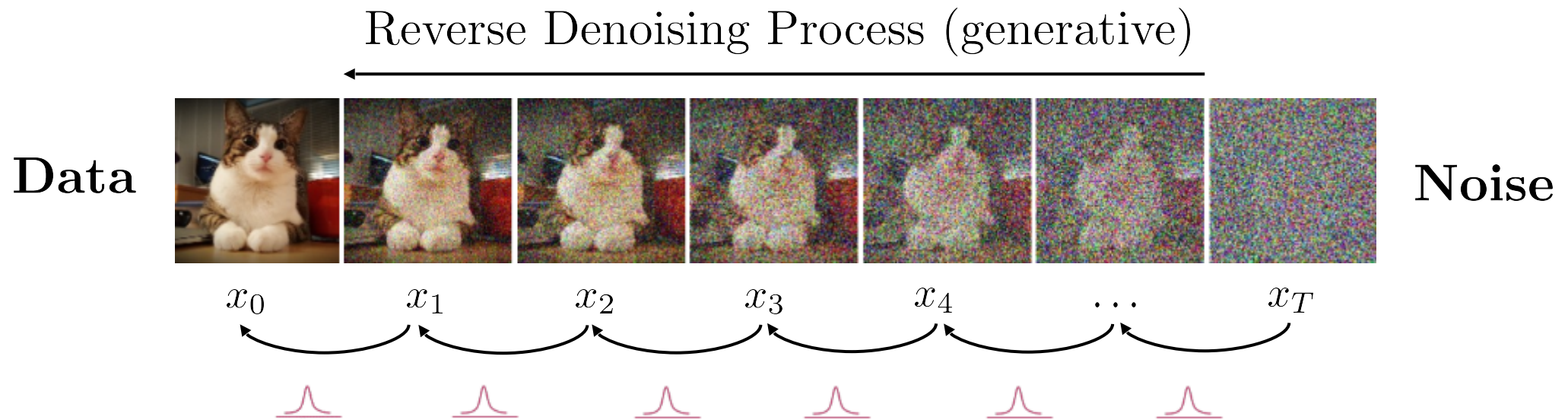
The forward diffusion process:



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

Recap: Reverse Denoising Process

The formal definition of the reverse process in T steps:



$$p(x_T) = \mathcal{N}(x_T; 0, I_d)$$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \underbrace{\mu_\theta(x_t, t)}_{\text{Trainable network (U-net, Denoising Autoencoder)}}, \sigma_t^2 I_d) \quad \longrightarrow \quad p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t).$$

$\approx q(x_{t-1}|x_t)$ (true posterior; intractable)

Recap: Training and Sampling

Lecture #13

Minimize a simplification of negative ELBO:

$$L_{\text{simple}} = \mathbb{E}_{x_0 \sim p_d(x_0), \epsilon \sim \mathcal{N}(0, I_d), t \sim \mathcal{U}(1, T)} \left[\left\| \epsilon - \epsilon_{\theta} \left(\underbrace{\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon}_{x_t}, t \right) \right\|^2 \right].$$

Algorithm 1 Training

- 1: **repeat**
- 2: $x_0 \sim p_d(x_0)$
- 3: $t \sim \text{Uniform}(1, \dots, T)$
- 4: $\epsilon \sim \mathcal{N}(0, I_d)$
- 5: Take gradient descent step on

$$\nabla_{\theta} \left\| \epsilon - \epsilon_{\theta} \left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t \right) \right\|^2$$

- 6: **until** converged
-

Algorithm 2 Sampling

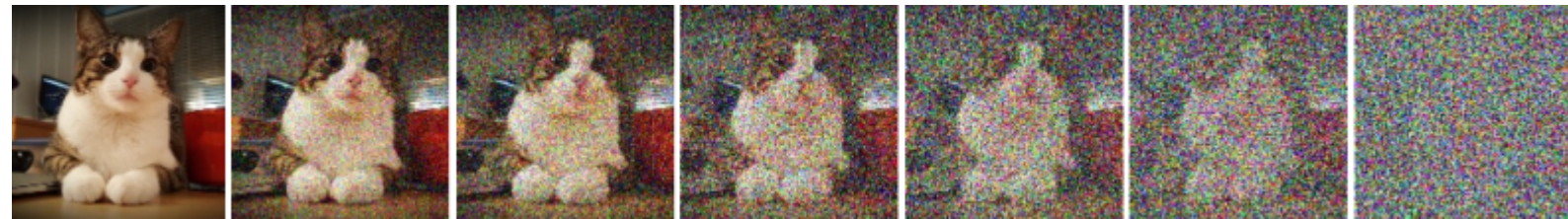
- 1: $x_T \sim \mathcal{N}(0, I_d)$
 - 2: **for** $t = T, \dots, 1$ **do**
 - 3: $z \sim \mathcal{N}(0, I_d)$
 - 4: $x_{t-1} = \frac{1}{\sqrt{1 - \beta_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(x_t, t) \right) + \sigma_t z$
 - 5: **end for**
 - 6: **return** x_0
-

Forward Diffusion Process Limit

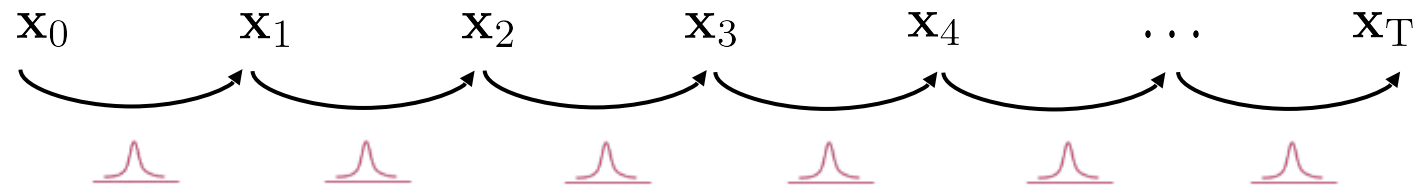
Consider the limit
of many small steps:

Forward diffusion process (fixed)

Data



Noise



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

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

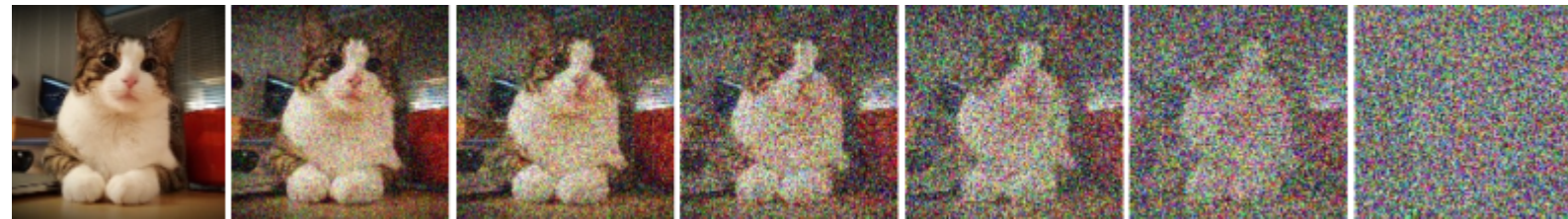
$$= \sqrt{1 - \beta(t)\Delta t}x_{t-1} + \sqrt{\beta(t)\Delta t}z_t \quad (\beta_t := \beta(t)\Delta t)$$

$$\approx x_{t-1} - \frac{\beta(t)\Delta(t)}{2}x_{t-1} + \sqrt{\beta(t)\Delta t}z_t. \quad (\text{Taylor expansion})$$

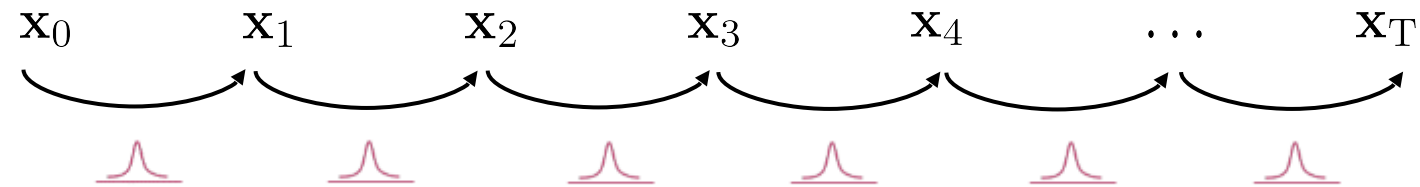
Forward Diffusion Process as an SDE

Consider the limit
of many small steps:
Data

Forward diffusion process (fixed)



Noise



$$x_t - x_{t-1} \approx \frac{\beta(t)\Delta(t)}{2}x_{t-1} + \sqrt{\beta(t)\Delta t}z_t$$

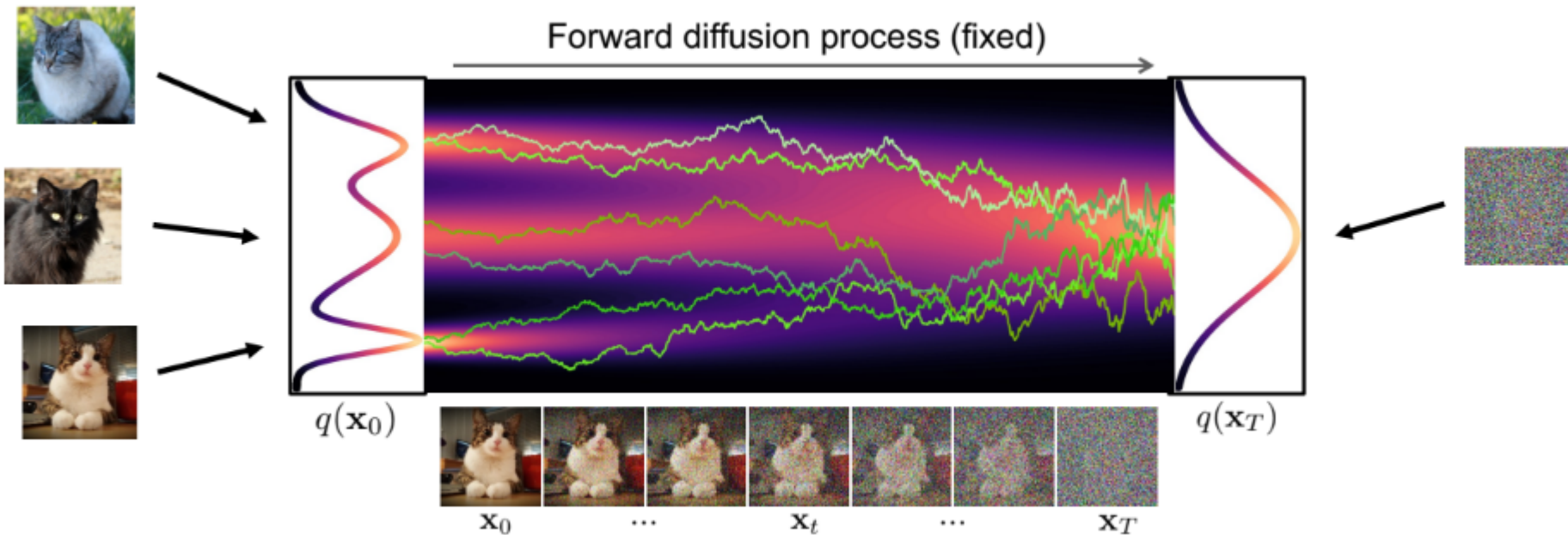


Stochastic Differential Equation (SDE)

describing the diffusion
process in the infinitesimal limit

$$dx_t = -\frac{1}{2}\beta(t)x_t dt + \sqrt{\beta(t)}dW_t \quad W_t: \text{Weiner process}$$

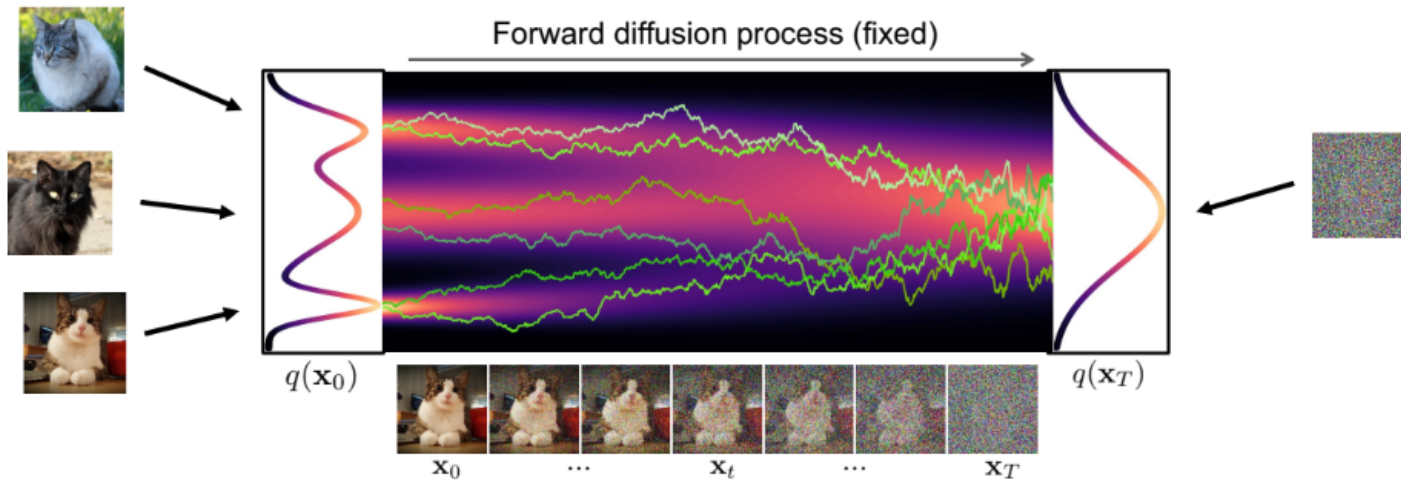
Forward Diffusion Process as an SDE



Forward Diffusion SDE:

$$dx_t = \underbrace{-\frac{1}{2}\beta(t)x_t dt}_{\text{drift term (pulls towards mode)}} + \underbrace{\sqrt{\beta(t)}dW_t}_{\text{diffusion term (injects noise)}}.$$

Denoising Score Matching

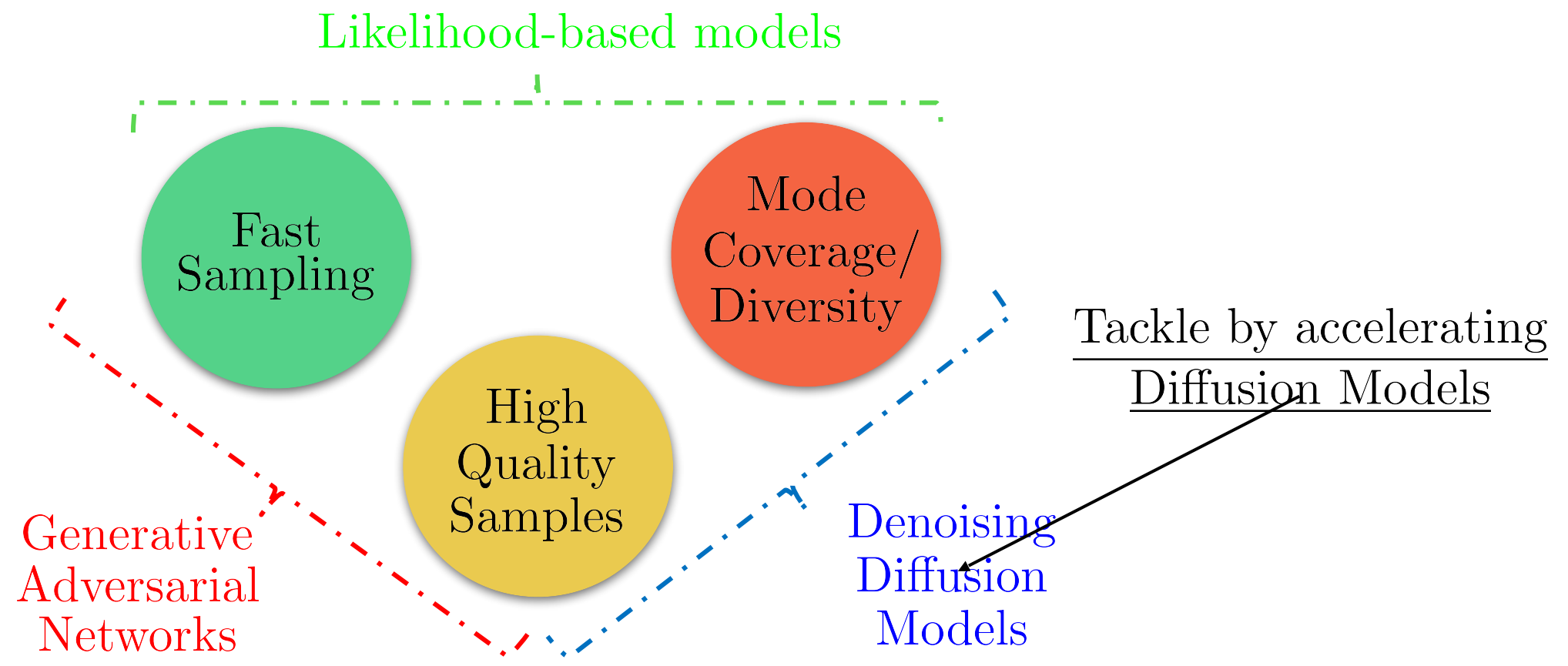


- Instead, diffuse individual data points x_0 . Conditional $q_t(x_t|x_0)$ *is* tractable!
- **Denoising Score Matching:**

$$\min_{\theta} \underbrace{\mathbb{E}_{t \sim \mathcal{U}(0, T)}}_{\text{diffusion time } t} \underbrace{\mathbb{E}_{x_0 \sim q_0(x_0)}}_{\text{diffused data } x_0} \underbrace{\mathbb{E}_{x_t \sim q_t(x_t|x_0)}}_{\text{diffused data sample } x_t|x_0} \left\| \underbrace{s_{\theta}(x_t, t)}_{\text{neural network}} - \underbrace{\nabla_x \log q_t(x_t|x_0)}_{\text{score of diffused data sample}} \right\|_2^2.$$

➔ After expectations, $s_{\theta}(x_t, t) \approx \nabla_x \log q_t(x_t)$!

The Generative Learning Trilemma



- Naive acceleration methods: Reduce diffusion time steps in training every k -th time step in inference.
Unfortunately, it leads to immediate worse performance.
- We need something more clever.
- Given a limited number of functional calls, usually much less than 1000, how to improve the performance?

- Does the noise schedule have to be predefined?
- Does it have to be a Markovian process?
- Is there any faster mixing diffusion process?

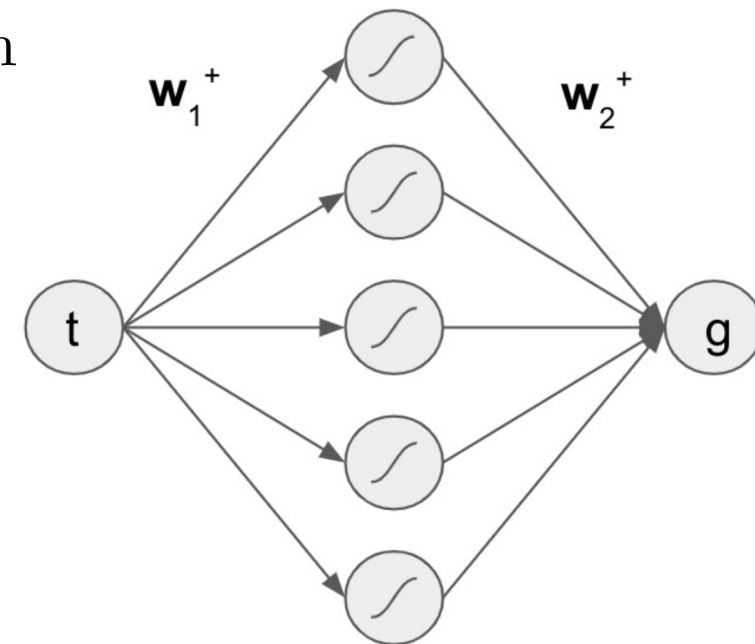
Variational Diffusion Models: Learnable Diffusion Process

Lecture #13

- Given the forward process $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I)$:
- Directly parametrize the variance through a learned function

$$1 - \bar{\alpha}_t = \text{sigmoid}(\gamma_\eta(t)).$$

- $\gamma_\eta(t)$: A monotonic MLP.
 - Strictly positive weights & monotonic activations (e.g., sigmoid).
- Analogous to hierarchical VAE: unlike diffusion models using a fixed encoder, we include learnable parameters in the encoder.



Variational Diffusion Models: New Parametrization of Training Objectives

- Optimizing variational upper bound of diffusion models can be simplified to the following training objective:

$$\mathcal{L}_T = \frac{T}{2} \mathbb{E}_{x_0, \epsilon, t} \left[(\exp(\gamma_\eta(t) - \gamma_\eta(t-1)) - 1) \|\epsilon - \epsilon_\theta(x_t, t)\|_2^2 \right]$$

- Learning noise schedule improves likelihood estimation of diffusion models, given fewer diffusion steps.

Variational Diffusion Models: New Parametrization of Training Objectives

- Letting $T \rightarrow \infty$ leads to variational upper bound in continuous time:

$$\mathcal{L}_\infty = \frac{1}{2} \mathbb{E}_{x_0, \epsilon, t} [\gamma'_\eta \|\epsilon - \epsilon_\theta(x_t, t)\|_2^2], \quad \gamma'_\eta(t) = \frac{d}{dt} \gamma_\eta(t).$$

- It is shown to be only related to the signal-to-noise ratio (SNR):

$$\text{SNR}(t) = \frac{\bar{\alpha}_t}{1 - \bar{\alpha}_t} = \exp(-\gamma_\eta(t)).$$

at endpoints, invariant to the noise schedule in-between.

- The continuous time noise schedule can be learned to minimize the variance of the training objective for faster training.

- Key factor: Appending Fourier features to the input of U-Net:

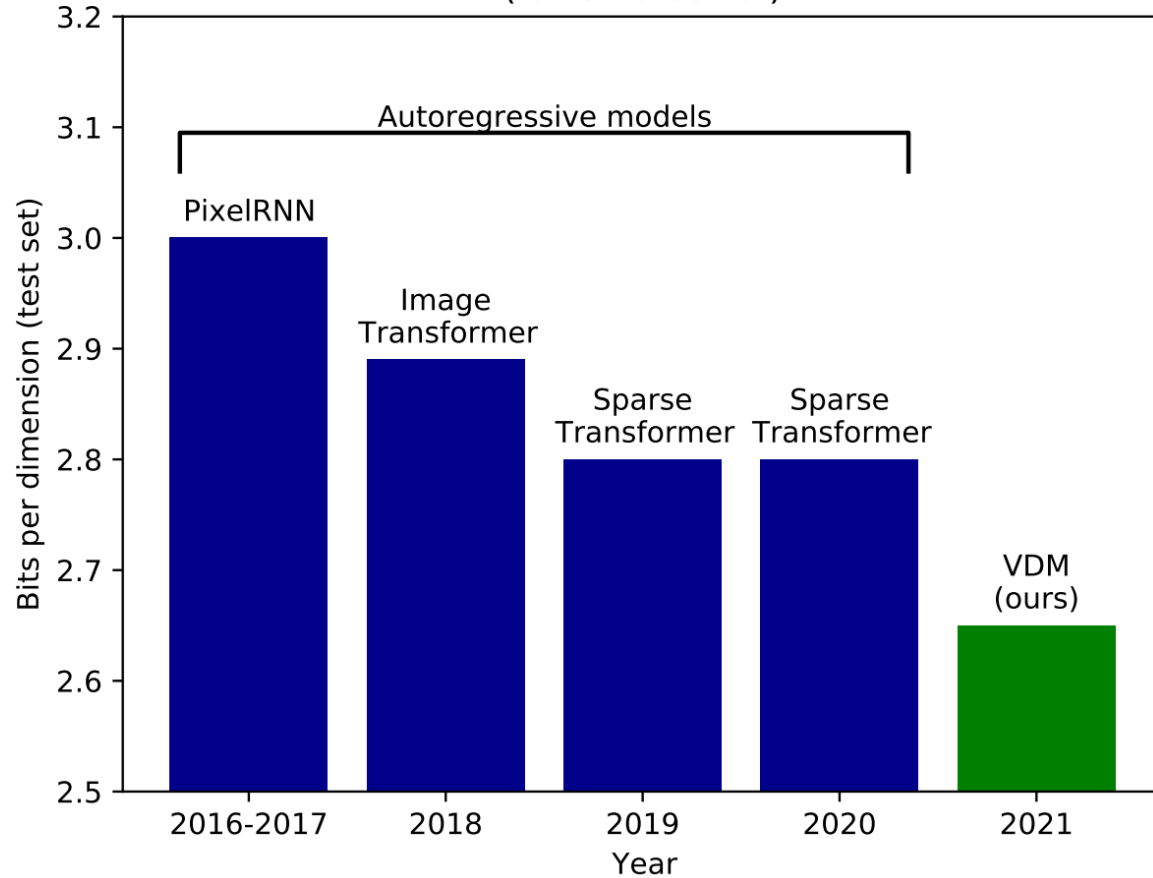
$$f_{i,j,k}^n = \sin(x_{i,j,k} 2^n \pi), \quad g_{i,j,k}^n = \cos(x_{i,j,k} 2^n \pi), \quad n = 7, 8.$$

- Good likelihoods require modeling all bits, even the ones corresponding to very small changes in the input.
- But: Neural Networks are usually bad at modeling small changes to the inputs.
- Significant improvements in log-likelihoods.

Variational Diffusion Models: SOTA Likelihood Estimation

CIFAR-10 without data augmentation

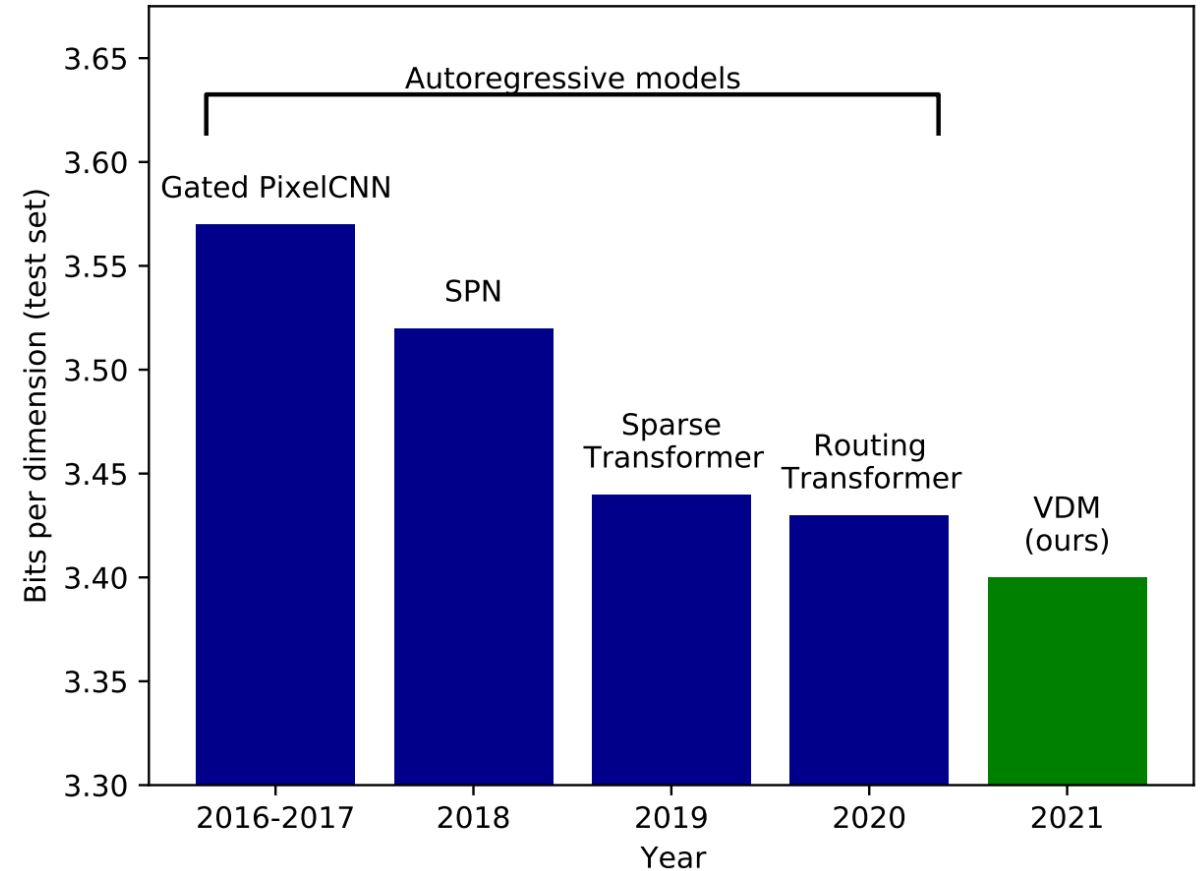
State-of-the-art models in each of the 5 past years
(lower is better)



(a) CIFAR-10 without data augmentation

ImageNet 64x64

State-of-the-art models in each of the 5 past years
(lower is better)

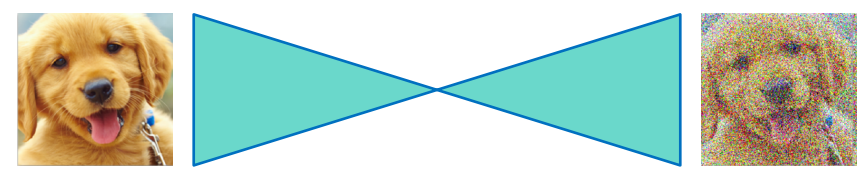


(b) ImageNet 64x64

Advanced Reverse Process: Approximating transition probabilities with more complicated distributions



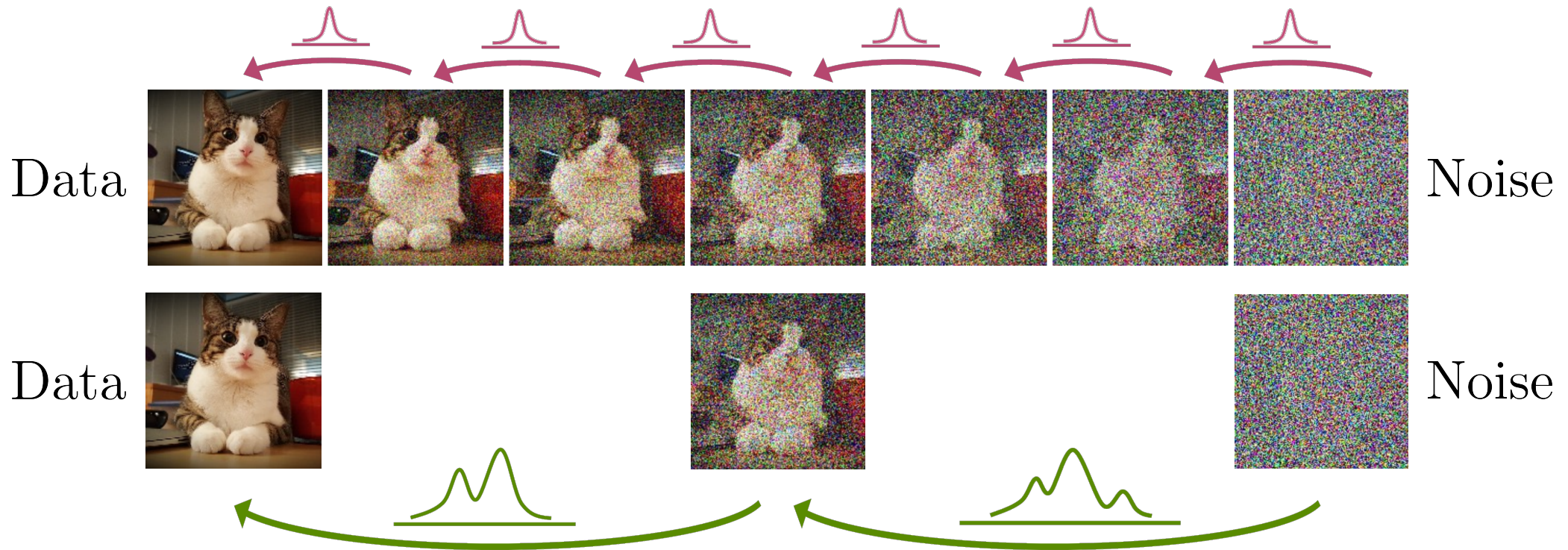
Reverse process maps noise back to data
where the diffusion model is trained



- Question: Is normal approximation of the reverse process still accurate when there are less diffusion time steps?

Advanced Reverse Process: Normal Assumption in Denoising Distribution Holds Only for Small Step

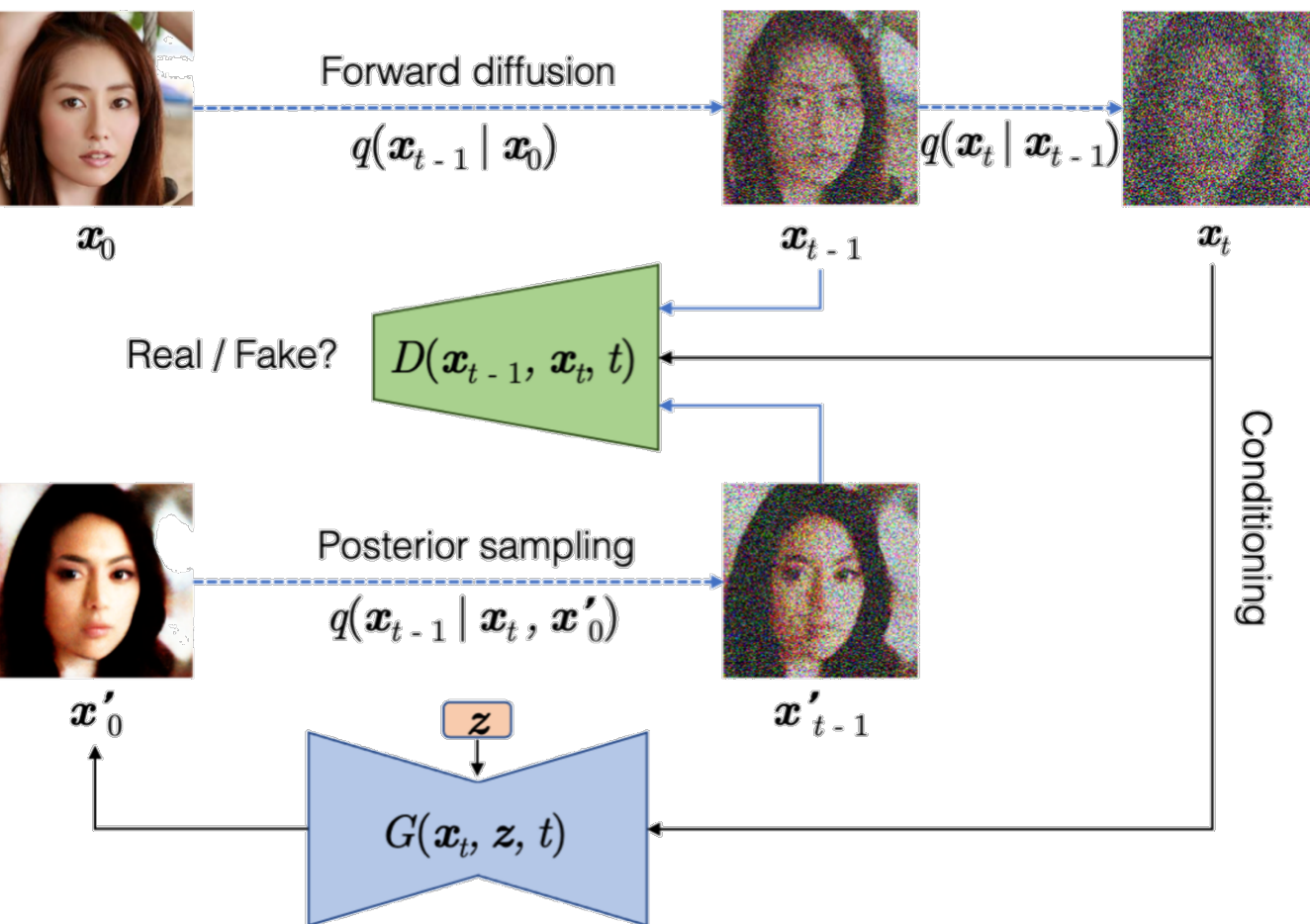
- Denoising Process with unimodal normal distribution:



- Requires more complicated functional approximators!

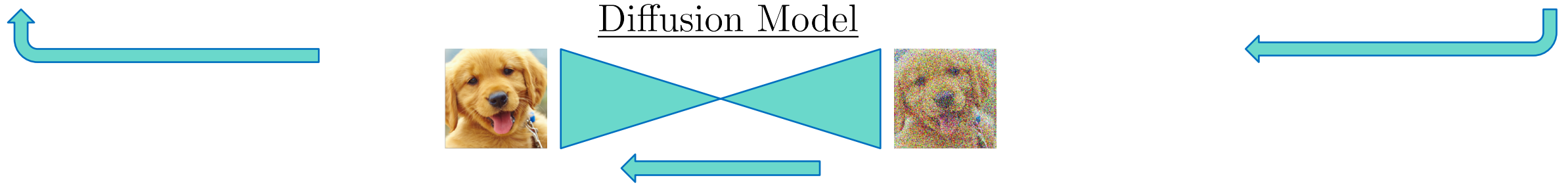
Denoising Diffusion GANs: Approximating Reverse Process by Conditional GANs

$$\min_{\theta} \sum_{t \geq 1} \mathbb{E}_{q(x_t)} [D_{\text{adv}} (q(x_{t-1} | x_t) || p_{\theta}(x_{t-1} | x_t))].$$



Compared to a one-shot GAN generator:

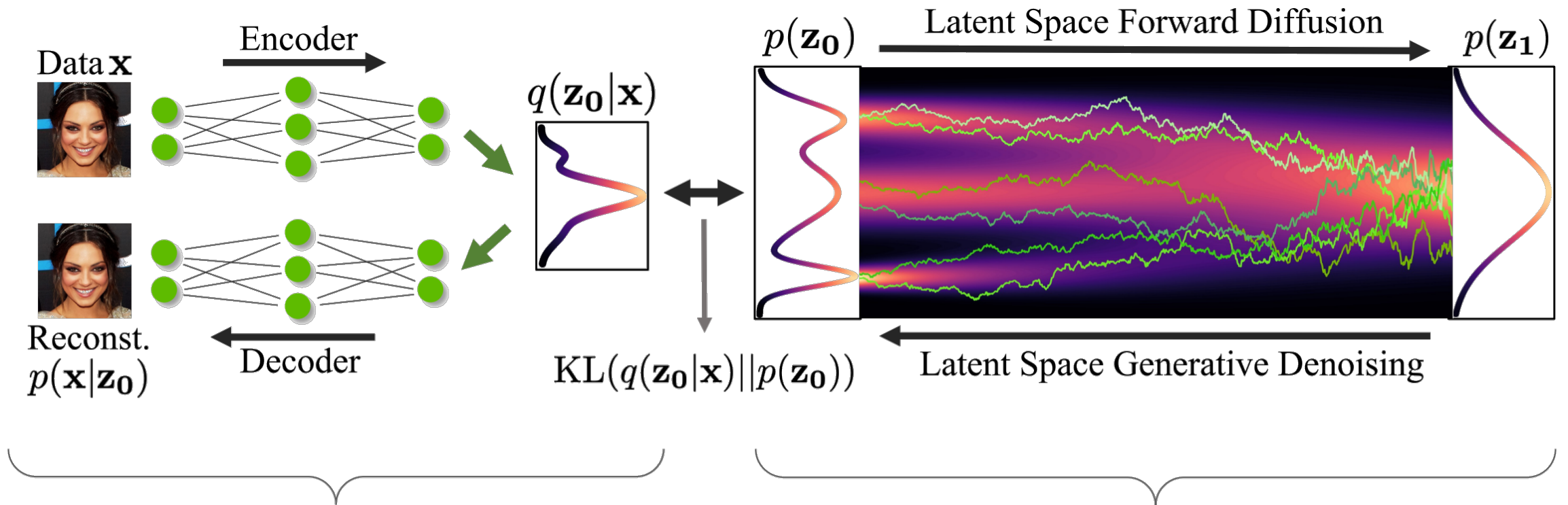
- Both generator and discriminator are solving a much simpler problem.
- Stronger mode coverage.
- Better training stability.



- Q: Can we lift the diffusion model to a latent space that is faster to diffuse?

Latent-Space Diffusion Models: Variational Autoencoder + Score-Based Prior

Lecture #13



Variational Autoencoder

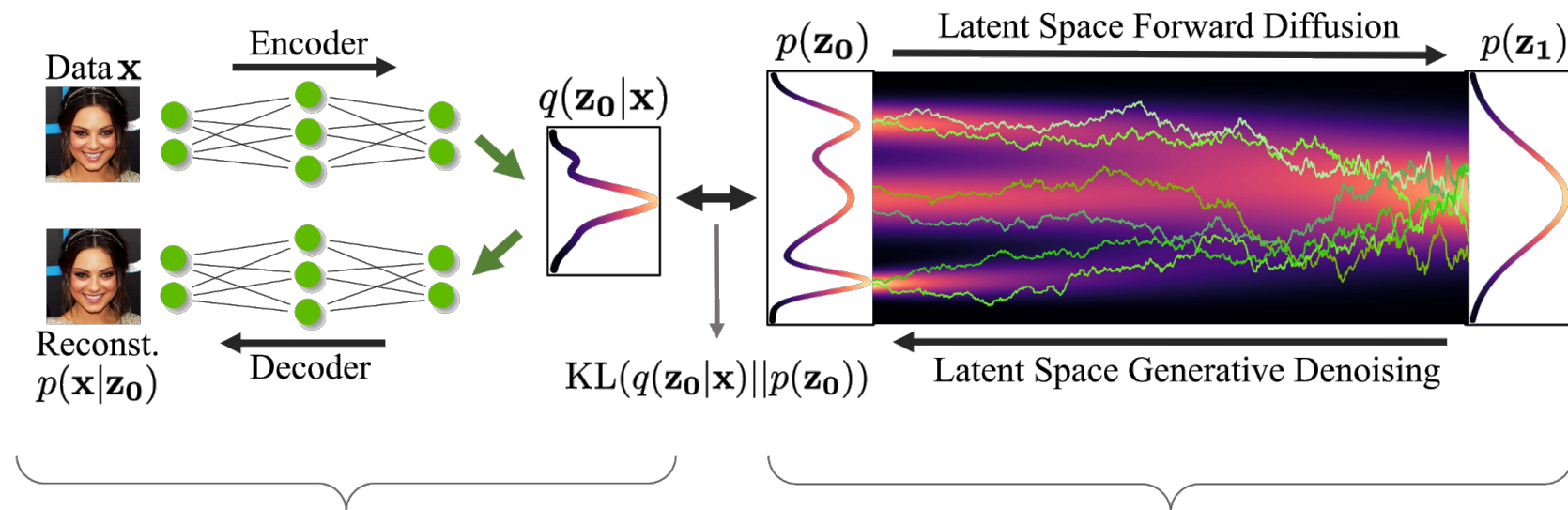
Denoising Diffusion Prior



- Encoder maps the input data to an embedding space.
- Denoising diffusion models are applied in the latent space.

Latent-Space Diffusion Models: Variational Autoencoder + Score-Based Prior

Lecture #13



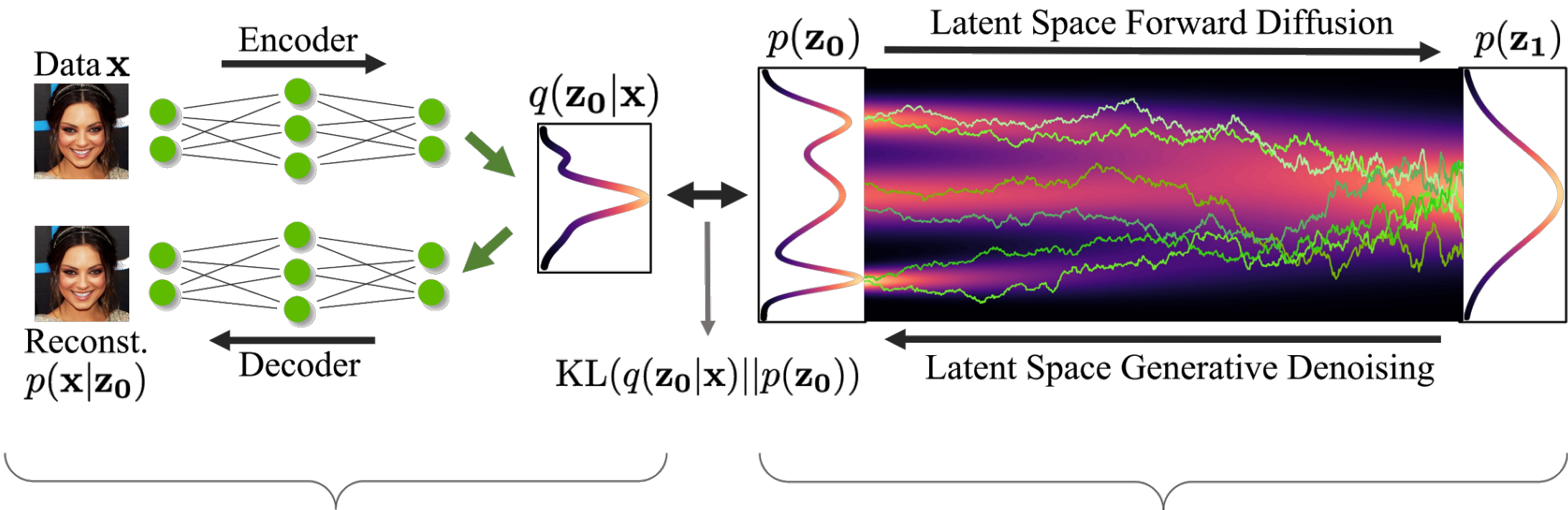
Variational Autoencoder

Denoising Diffusion Prior

- Advantages:
 - The distribution of latent embeddings close to Normal distribution \rightarrow Simpler denoising and faster synthesis.
 - Augmented latent space \rightarrow More expressivity.
 - Tailored Autoencoders \rightarrow More expressivity, application to any data type, e.g., graphs, text, 3D data, etc.

Latent-Space Diffusion Models: Variational Autoencoder + Score-Based Prior

Lecture #13



Variational Autoencoder

Denoising Diffusion Prior

$$\begin{aligned}
 \mathcal{L}(x, \phi, \theta, \psi) &= \mathbb{E}_{q_\phi(z_0|x)} [-\log p_\psi(x|z_0)] + D_{\text{KL}}(q_\phi(z_0|x) || p_\theta(z_0)) \\
 &= \underbrace{\mathbb{E}_{q_\phi(z_0|x)} [-\log p_\psi(x|z_0)]}_{\text{reconstruction term}} + \underbrace{\mathbb{E}_{q_\phi(z_0|x)} [\log q_\phi(z_0|x)]}_{\text{negative encoder entropy}} + \underbrace{\mathbb{E}_{p_\theta(z_0)} [-\log p_\psi(x|z_0)]}_{\text{cross entropy}}.
 \end{aligned}$$

- There are many successful applications of diffusion models (in constantly growing numbers):
 - **Image generation**, text-to-image generation, **controllable generation**.
 - Image editing, image-to-image translation, super-resolution, segmentation, adversarial robustness.
 - Discrete models, 3D generation, medical imaging, video synthesis.
- Key enabler by diffusion models: Perform high-resolution conditional generation!

High Resolution Conditional Generation: Impressive Text-to-Image Conditional Diffusion Models

“a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese”



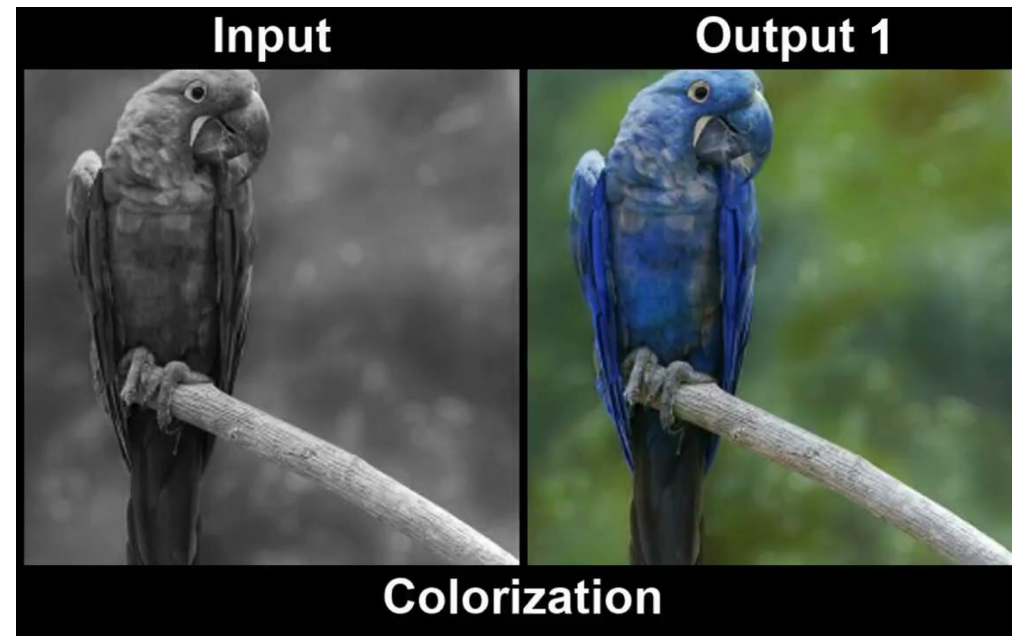
“A photo of a raccoon wearing an astronaut helmet, looking out of the window at night.”



High Resolution Conditional Generation: Impressive Super-Resolution & Colorization Diffusion Models



Super-Resolution



Colorization

High Resolution Conditional Generation: Impressive Panorama Generation Diffusion Models

Lecture #13

Generated

Input

Generated



Conditional Diffusion Models: Include Condition as Input to Reverse Process

- Reverse Process:

$$p_{\theta}(x_{0:T}|c) = p(x_T) = \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t, c), \quad p_{\theta}(x_{t-1}|x_t, c) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t, c), \Sigma(x_t, t, c)).$$

- Variational Upper Bound:

$$L_{\theta}(x_0|c) = \mathbb{E}_q[L_T(x_0) + \sum_{t>1} D_{\text{KL}}(q(x_{t-1}|x_t, x_0) || p(x_{t-1}|x_t, c)) - \log p_{\theta}(x_0|x_1, c)].$$

- Incorporate Conditions into U-Net:

- Scalar conditioning: Encode scalar as a vector embedding, simple spatial addition or adaptive group normalization layers.
- Image conditioning: Channel-wise concatenation of the conditional image.
- Text conditioning: Single vector embedding – spatial addition or adaptive group norm / a sequence of vector embeddings - cross-attention.

Classifier-Guided Conditional Diffusion Models: Using the Gradient of a Trained Classifier as Guidance

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

Input: class label y , gradient scale s

$x_T \leftarrow$ 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

Score Model

Classifier Gradient



- For class-conditional modeling of $p(x_t|c)$, train an extra classifier $p(c|x_t)$.
- Mix its gradient with the diffusion/score model during sampling.

Classifier-Guided Conditional Diffusion Models: Using the Gradient of a Trained Classifier as Guidance

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

Input: class label y , gradient scale s

$x_T \leftarrow$ 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

Score Model

Classifier Gradient



- Sample with a modified score: $\nabla_{x_t} [\log p(x_t|c) + \omega \log p(c|x_t)]$.

- Approximate samples from the distribution $\tilde{p}(x_t|c) \propto p(x_t|c)p(c|x_t)^\omega$.

Classifier-Free Conditional Diffusion Models: Guidance by Bayes' Rule on Conditional Diffusion Models

- Instead of training an additional classifier, get an “implicit classifier” by jointly training a conditional and unconditional diffusion model:

$$p(c|x_t) \propto \frac{p(x_t|c)}{p(x_t)}.$$

← Conditional Diffusion Model
← Unconditional Diffusion Model

- In practice, compute $p(x_t|c)$ and $p(x_t)$ by randomly dropping the condition of the diffusion model at certain chance.
- The modified score with this implicit classifier included is:

$$\begin{aligned} \nabla_{x_t} [\log p(x_t|c) + \omega \log p(c|x_t)] &= \nabla_{x_t} [\log p(x_t|c) + \omega(\log p(x_t|c) - \log p(x_t))] \\ &= \nabla_{x_t} [(1 + \omega) \log p(x_t|c) - \omega \log p(x_t)]. \end{aligned}$$

Classifier-Free Conditional Diffusion Models: Trade-Off for Sample Quality and Sample Diversity

Non-Guidance

$\omega = 1$

$\omega = 3$



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

Introduction to Deep Generative Modeling

Lecture #13

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

Professors: Yannis Pantazis, Yannis Stylianou

Teaching Assistant: Michail Raptakis