Università di Genova | MaLGa

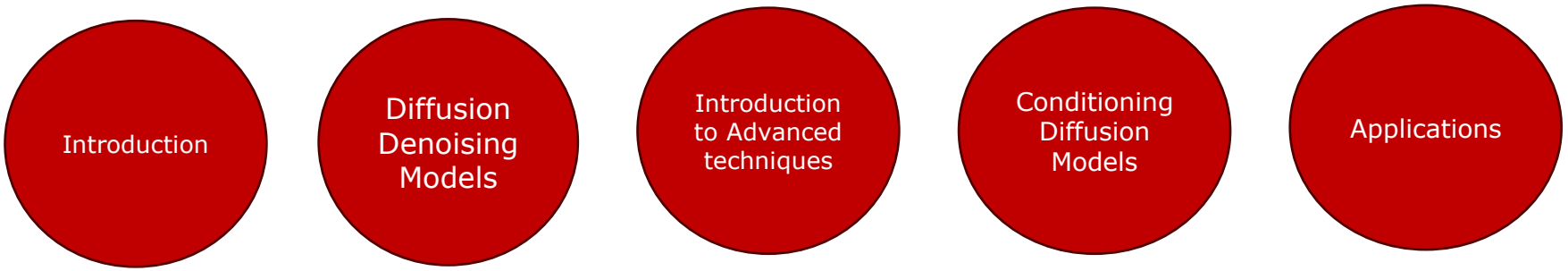
Diffusion models

07/05/2025

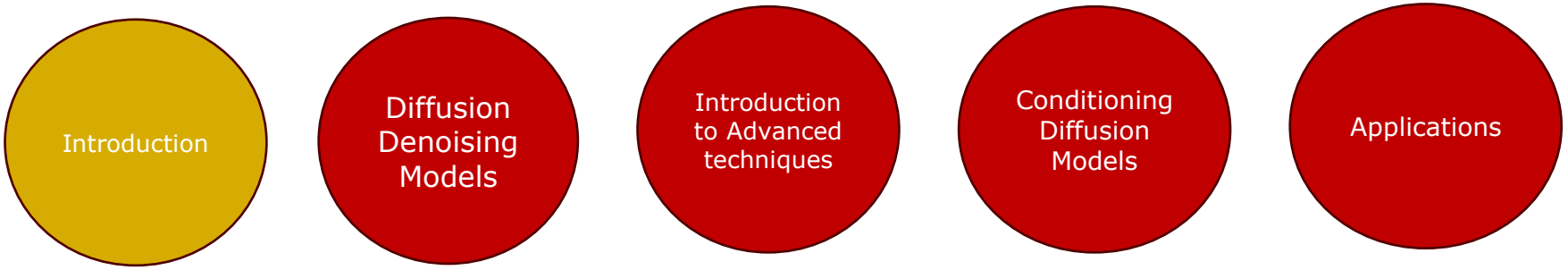
Vito Paolo Pastore

Deep learning a.y. 2024/2025

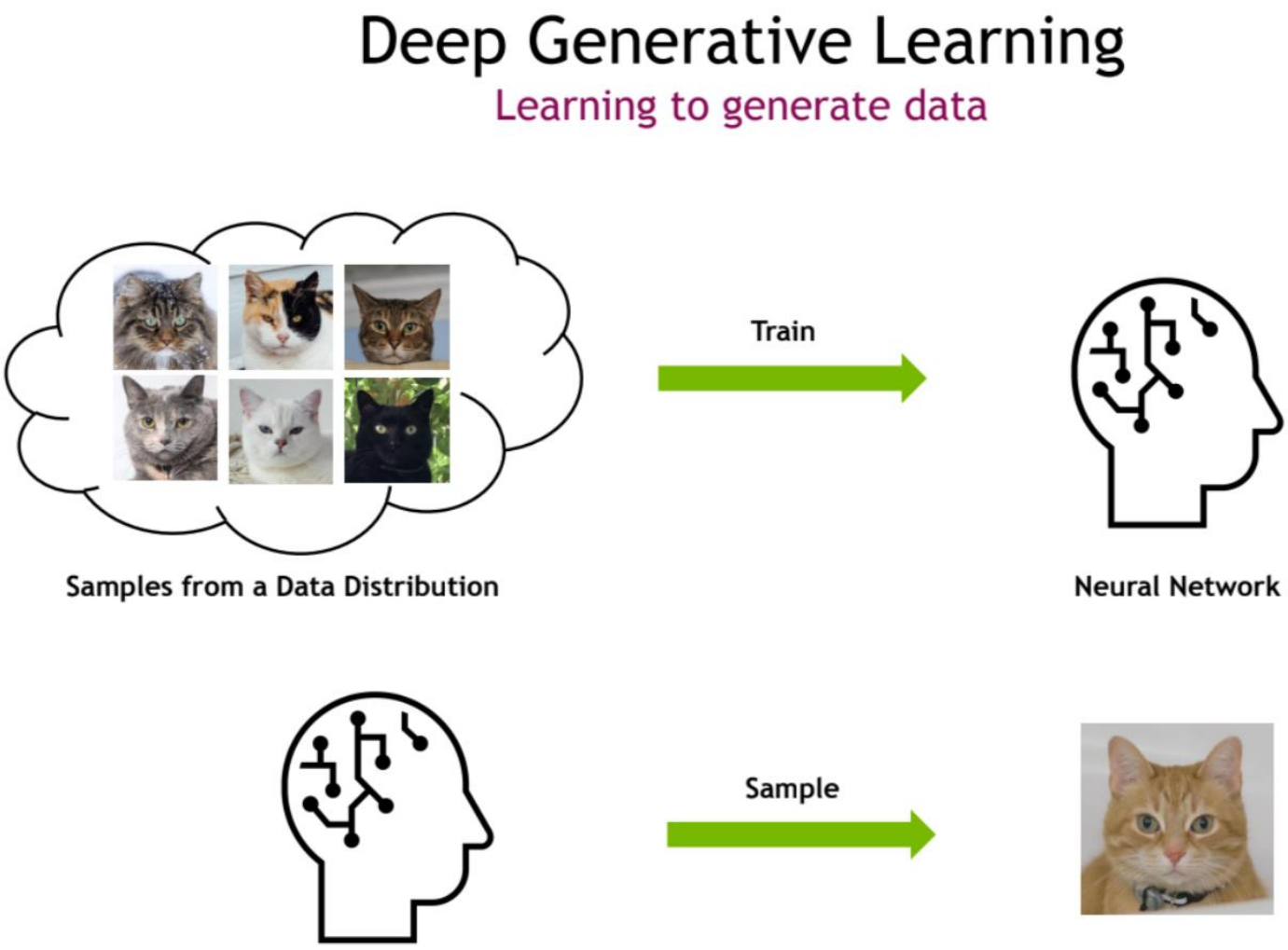
Outline


Introduction
Diffusion Denoising Models
Introduction to Advanced techniques
Conditioning Diffusion Models
Applications

Part 1


Introduction
Diffusion Denoising Models
Introduction to Advanced techniques
Conditioning Diffusion Models
Applications

Introduction: generative models



Slide from https://cvpr2022-tutorial-diffusion-models.github.io/

Introduction: generative models (2)

Given training data, generate new samples from same distribution

Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

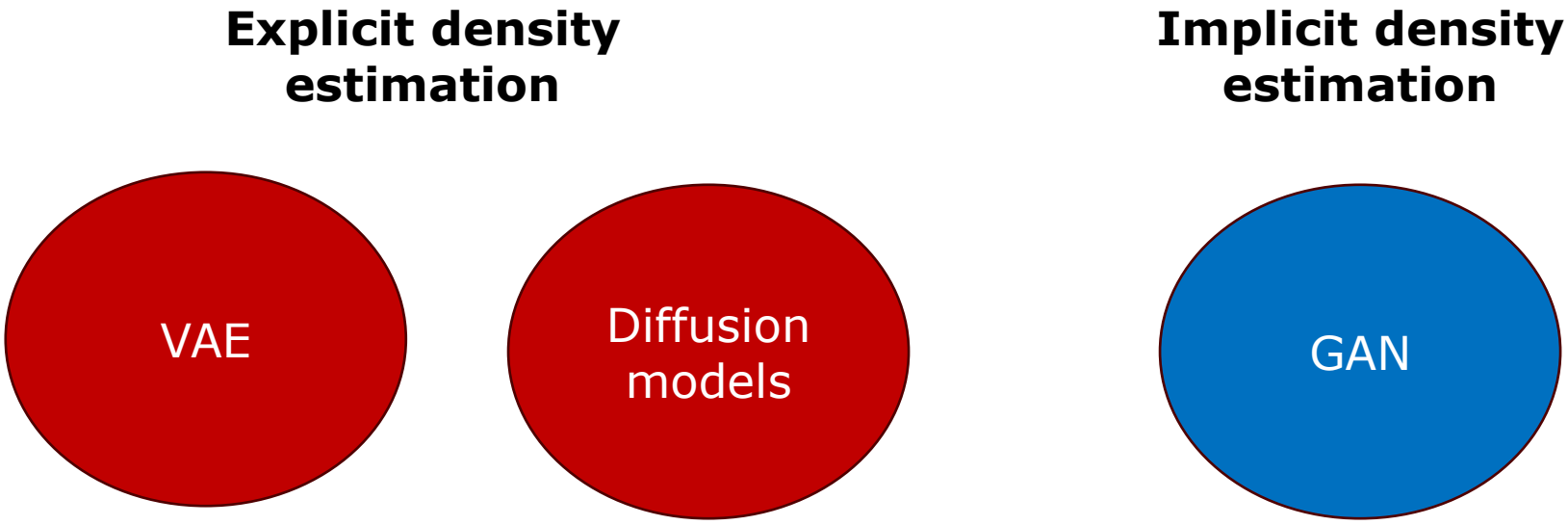
Addresses density estimation, a core problem in unsupervised learning Several flavors:

- **Explicit density estimation**: explicitly define and solve for pmodel(x)

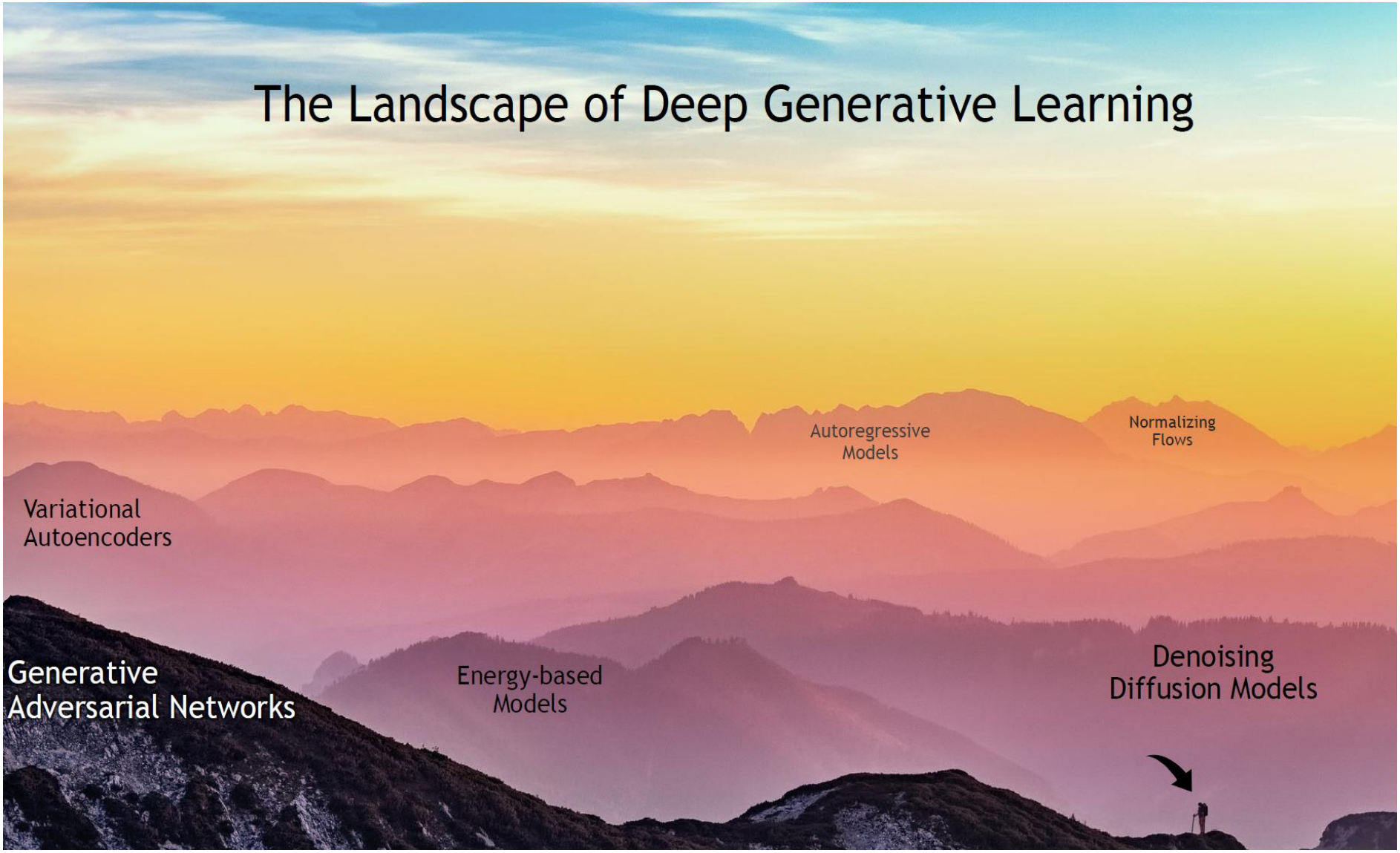
- **Implicit density estimation**: learn model that can sample from pmodel(x) without explicitly defining it

Adapted from https://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Introduction: generative models (3)



Slide from https://cvpr2022-tutorial-diffusion-models.github.io/


The Landscape of Deep Generative Learning
Autoregressive Models
Normalizing Flows
Variational Autoencoders
Generative Adversarial Networks
Energy-based Models
Denoising Diffusion Models

Denoising Diffusion models

DDPM have emerged as powerful generative models, outperforming GANs.

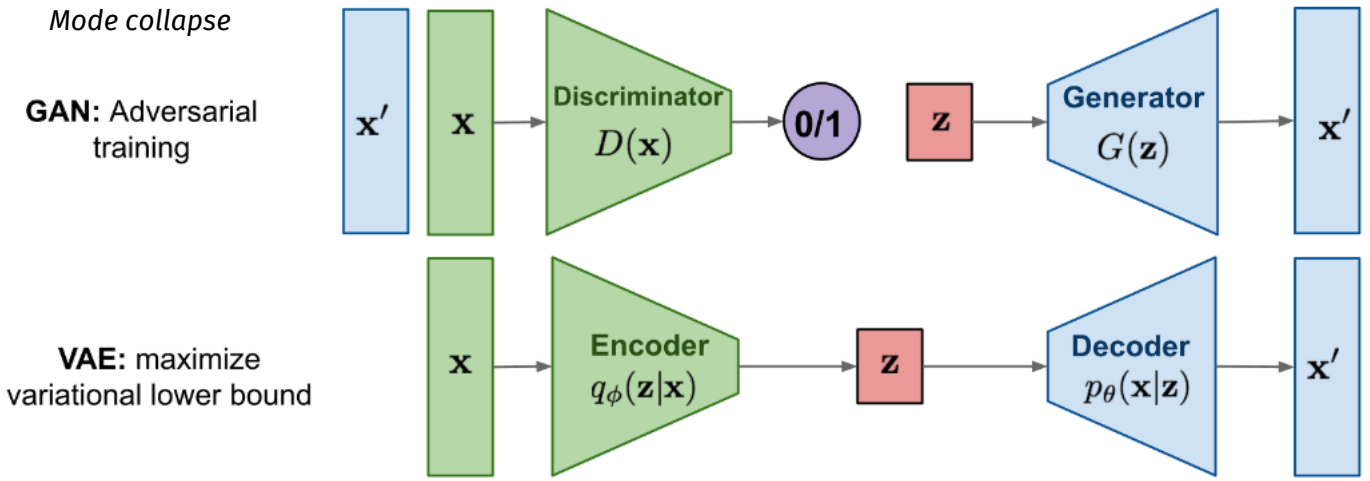
"Diffusion Models Beat GANs on Image Synthesis"
Dhariwal & Nichol, OpenAI, 2021

"Cascaded Diffusion Models for High Fidelity Image Generation"
Ho et al., Google, 2021

So far, GANs and VAEs

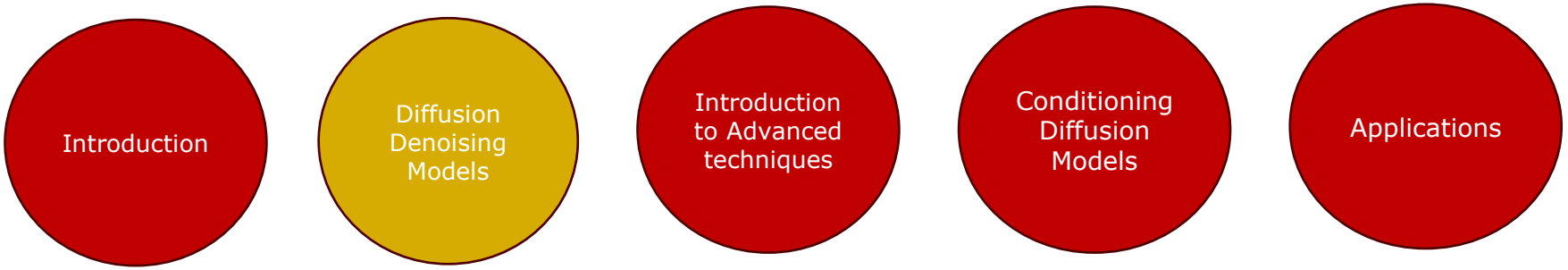
Vanishing gradients;

Mode collapse



https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Part 2


Introduction
Diffusion Denoising Models
Introduction to Advanced techniques
Conditioning Diffusion Models
Applications

Denoising Diffusion Probabilistic Models (DDPM)

- 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:

Diffusion models:
Gradually add Gaussian noise and then reverse

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

https://www.eecs.umich.edu/courses/eecs442-ahowens/fa23/slides/lec11-diffusion.pdf

Reparameterization trick (background)

- The loss function of a VAE is the following:

$$-L_{\text{VAE}} = \log p_\theta(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})$$

Always non-negative

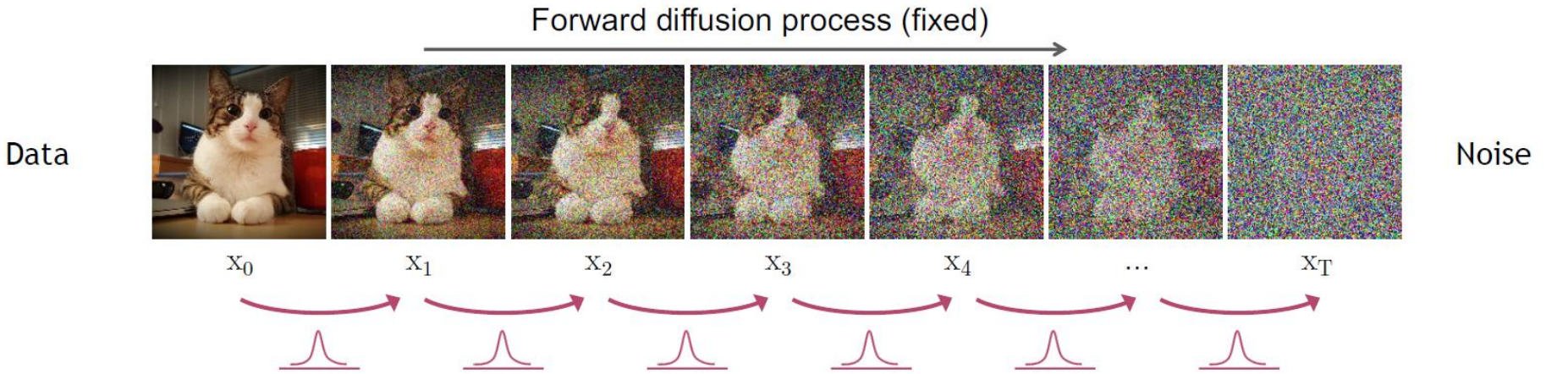
- It is called *variational lower bound;*
- Training the VAE requires to generate samples from $z \sim q_\varphi(z|x)$;
- But sampling is a stochastic process and therefore we cannot backpropagate the gradient -> Reparameterization trick
- If $q_\varphi(z|x)$ is a multivariate gaussian, then we get:

$$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\boldsymbol{I})$$
$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$$

Encodes the stochasticity

https://lilianweng.github.io/posts/2018-08-12-vae/#reparameterization-trick

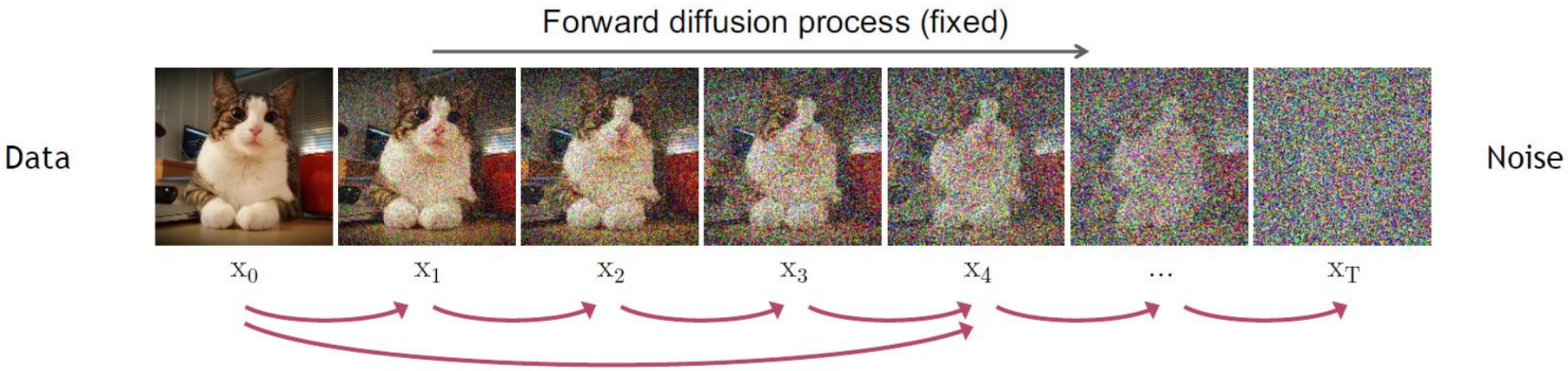
Forward diffusion process (1)



- Diffusion models are latent variable models where the diffusion process is fixed to a Markov chain
- Given an input $X_0 \sim q(x)$;
- We add Gaussian noise with T steps; $\{\beta_t \in (0,1)\}_{t=1}^{T}$.
- Step size is controlled with a variance schedule
- X_0 is degraded at every step.

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

Forward diffusion process (2)



- It is possible to sample x_t in a closed form at any arbitrary step t using a reparameterization trick;

$$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\boldsymbol{I})$$
$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$$

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

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

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

Usually, we can afford a larger update step when the sample gets noisier, so $\beta_1 < \beta_2 < \cdots < \beta_T$ and therefore $\bar{\alpha}_1 > \cdots > \bar{\alpha}_T$.

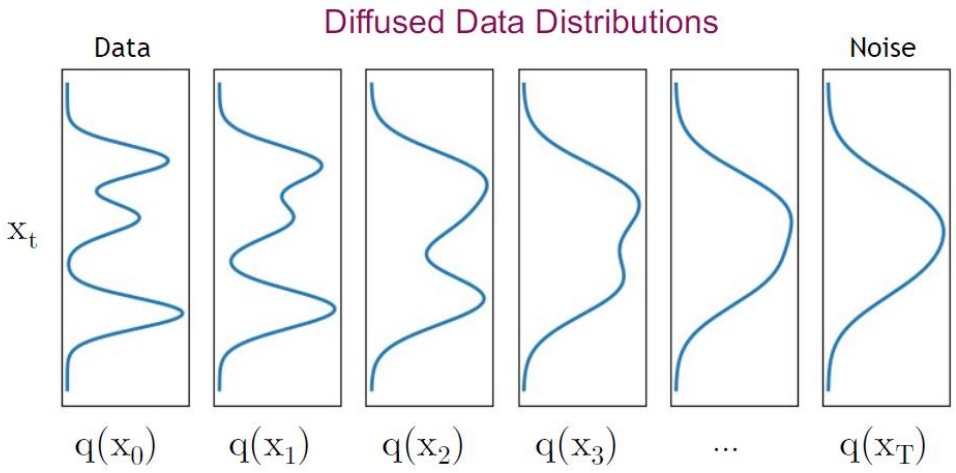
β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Forward diffusion process (3)

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

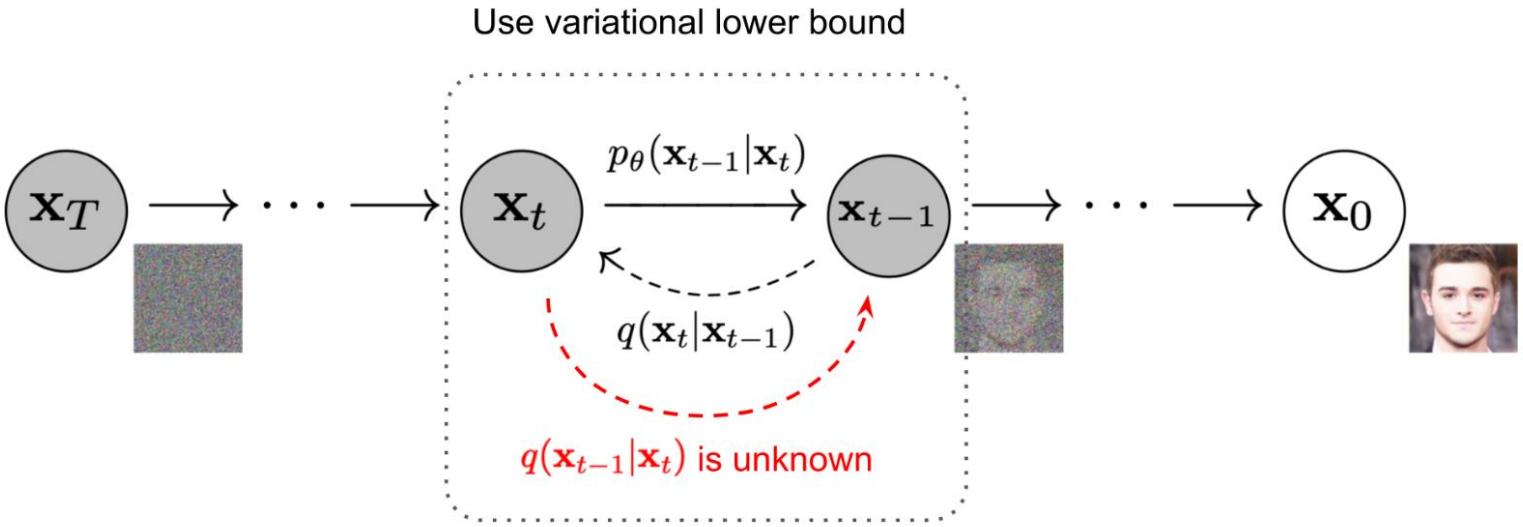
$$\underbrace{q(\mathbf{x}_t)}_{\text{Diffused data dist.}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\text{Joint dist.}} d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\text{Input data dist.}} \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\text{Diffusion kernel}} d\mathbf{x}_0$$

The diffusion kernel is Gaussian convolution.



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).

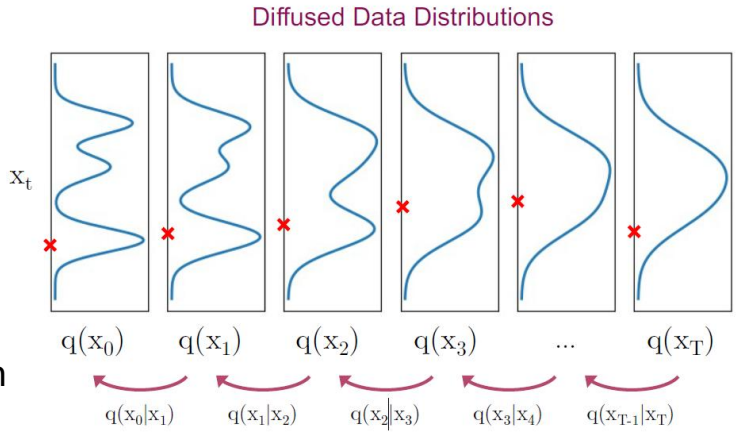
Reverse diffusion process (1)



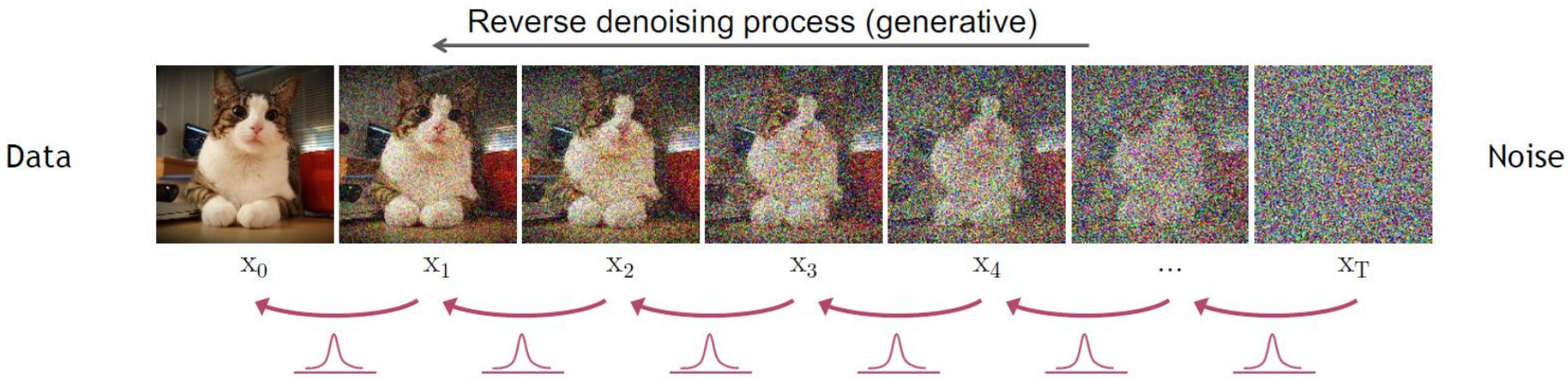
- Reversing the diffusion process, we can generate a true sample starting from pure noise;

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

- Basically, we need to sample from $q(x_{t-1}|x_t)$, that is intractable;
- However, if β_t is small enough, we can use a normal distribution
- We can learn a p_θ distribution to approximate $q(x_{t-1}|x_t)$



Reverse diffusion process (2)



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}, \sigma_t^2\mathbf{I})$$

Trainable network
(U-net, Denoising Autoencoder)

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

Learning a denoising model (1)

For training, we can maximize the variational lower bound, to optimize the negative log-likelihood of as done for VAE:

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

Learning a denoising model (2)

- To make the loss computable, we can rewrite the objective L as a combination of KL and entropy terms.
- We replace the terms in objectives with the definition previously provided:

$$
\begin{aligned}
L_{\text{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \\
&= \mathbb{E}_q\left[\log \frac{\prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\right]
\end{aligned}
$$

- As shown in Sohl-Dickstein et al. ICML 2015 and Ho et al. NeurIPS 2020, we get to:

$$
L_{\text{VLB}} = \mathbb{E}_q[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^{T} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]
$$

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

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

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

Learning a denoising model (2)

- We want to learn a neural network to approximate the conditioned probability distribution in the reverse diffusion problem.

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

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

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:

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

Noise used to generate x_t

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

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

Noise predicted by a neural network

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 simplification

- We can define the first term as a time dependent weight.

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

- Ho et al. NeurIPS 2020 show that ignoring the time weight improves sample quality.

- Thus, we get:

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

For more advanced weighting see Choi et al., Perception Prioritized Training of Diffusion Models, CVPR 2022.

Summary

Algorithm 1 Training

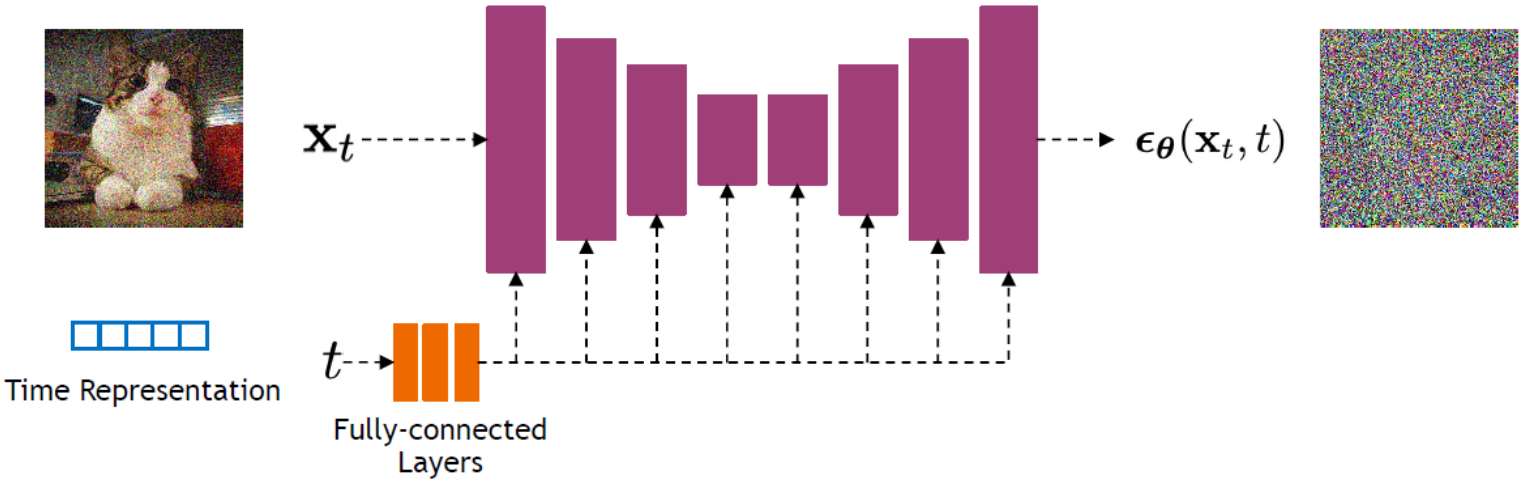
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

Algorithm 2 Sampling

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

Implementation

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)

Diffusion parameters

Diffusion Parameters

Noise Schedule

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

Data

Noise

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

Above, β_t and σ_t^2 control the variance of the forward diffusion and reverse denoising processes respectively.

Often a linear schedule is used for β_t, and σ_t^2 is set equal to β_t.

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

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

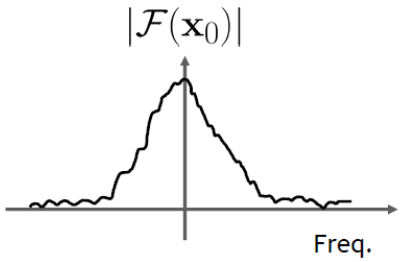
What happen to images in the forward diffusion process?

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

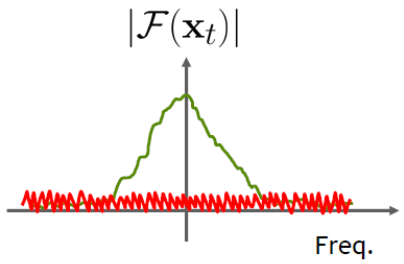
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$$

Fourier Transform

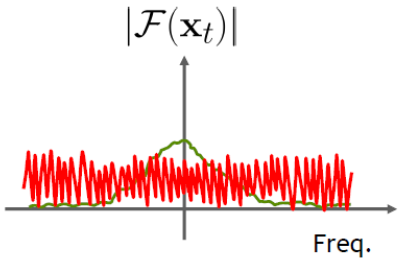
$$\mathcal{F}(\mathbf{x}_t) = \sqrt{\bar{\alpha}_t}\,\mathcal{F}(\mathbf{x}_0) + \sqrt{(1-\bar{\alpha}_t)}\,\mathcal{F}(\epsilon)$$



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

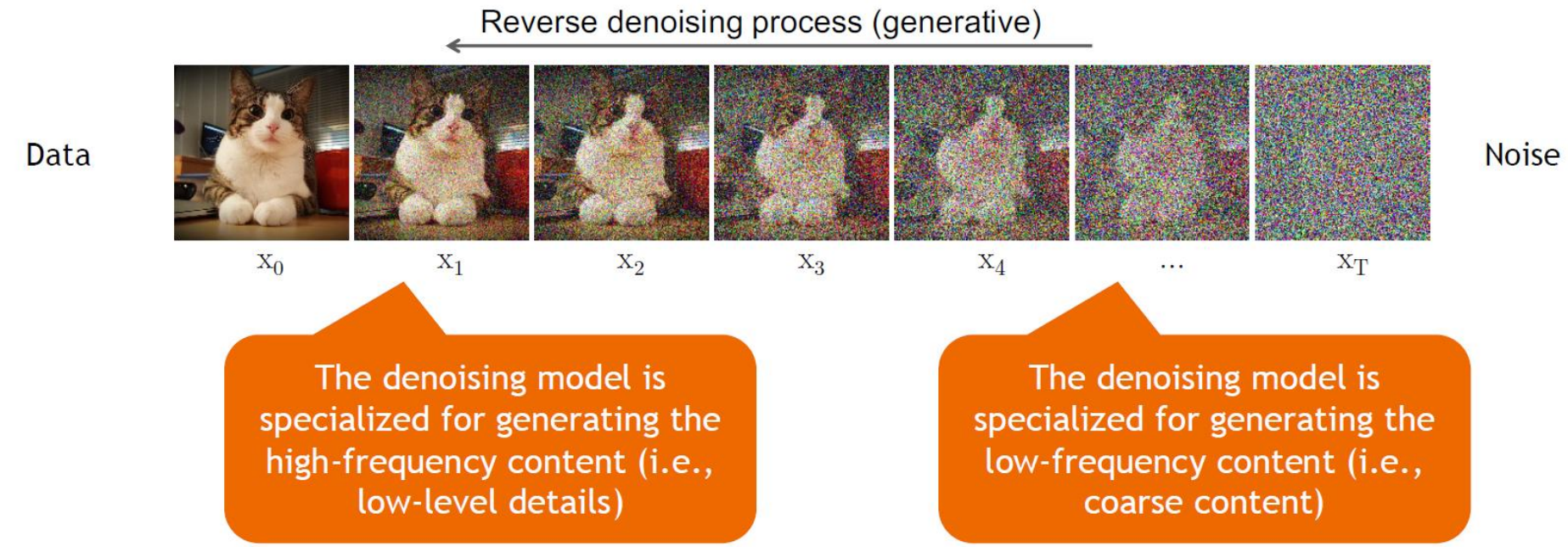


Large t
$\bar{\alpha}_t \sim 0$



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

Content-Detail Tradeoff



Summary

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

Data

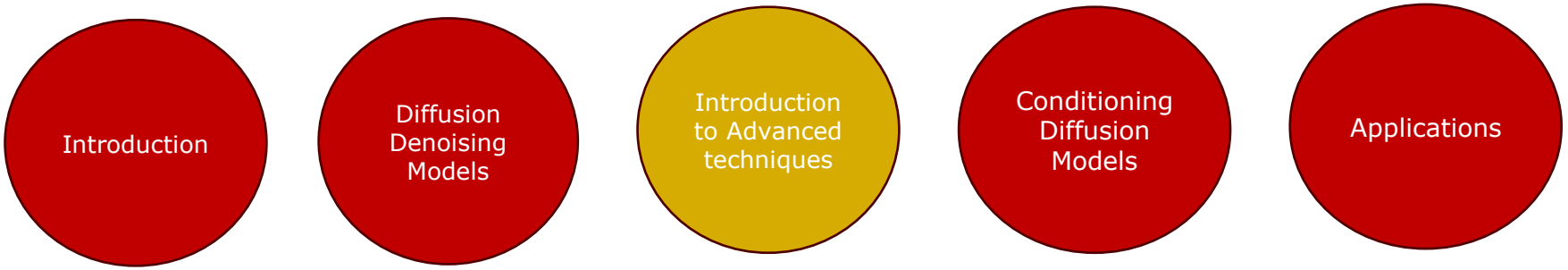
Noise

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

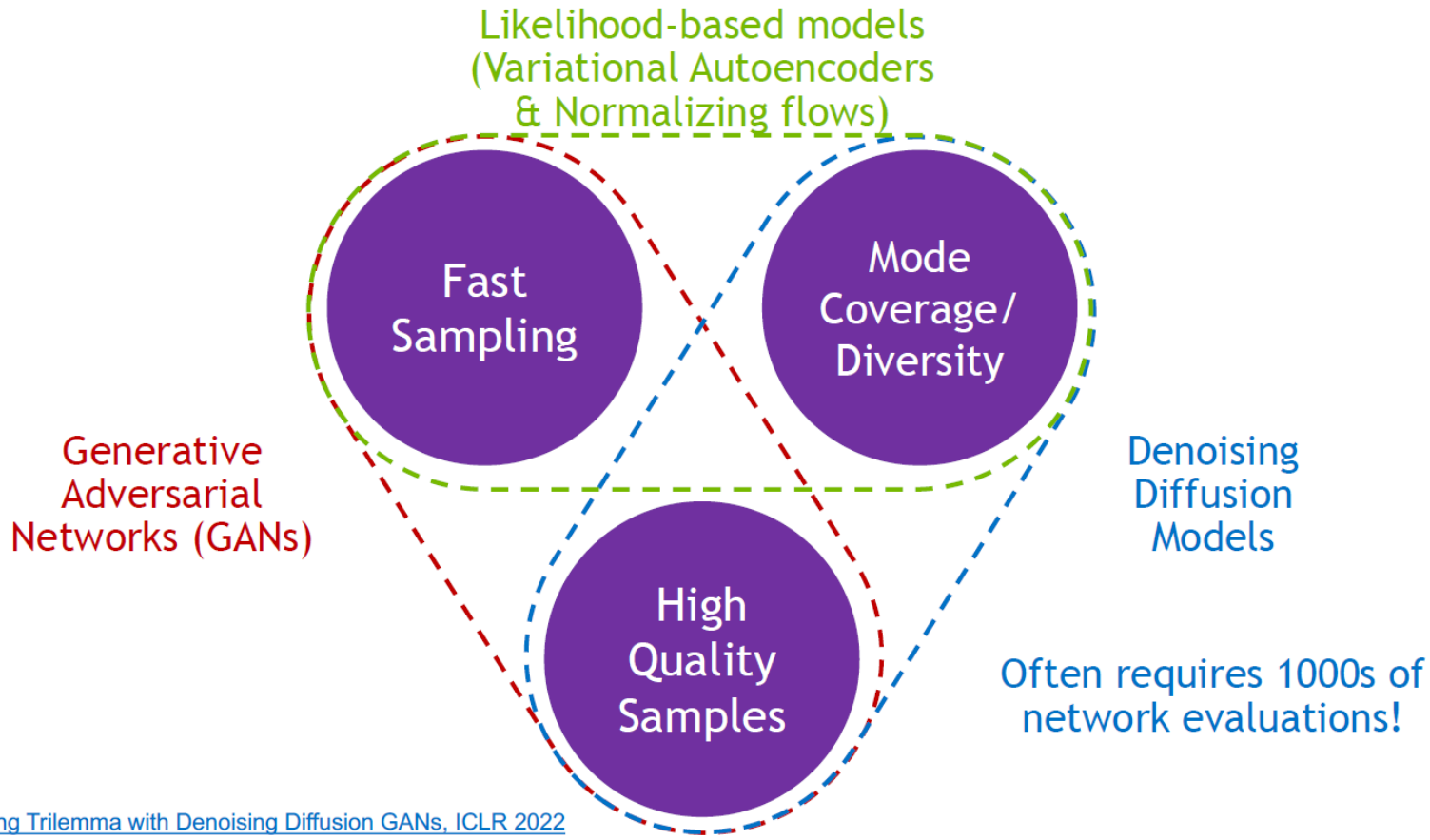
- Diffusion models are Markov chains where noise is added in time steps with small size;
- Diffusion models are composed of a forward (diffusion) and a reverse (generation) process;

- Generation is performed with a neural network that predicts the noise added at each time step;
- For generation we start from random noise and subtract the predicted noise from image iteratively.
- Important details are:
  - Network architecture;
  - Noise schedule;
  - Objective weighting

Part 3

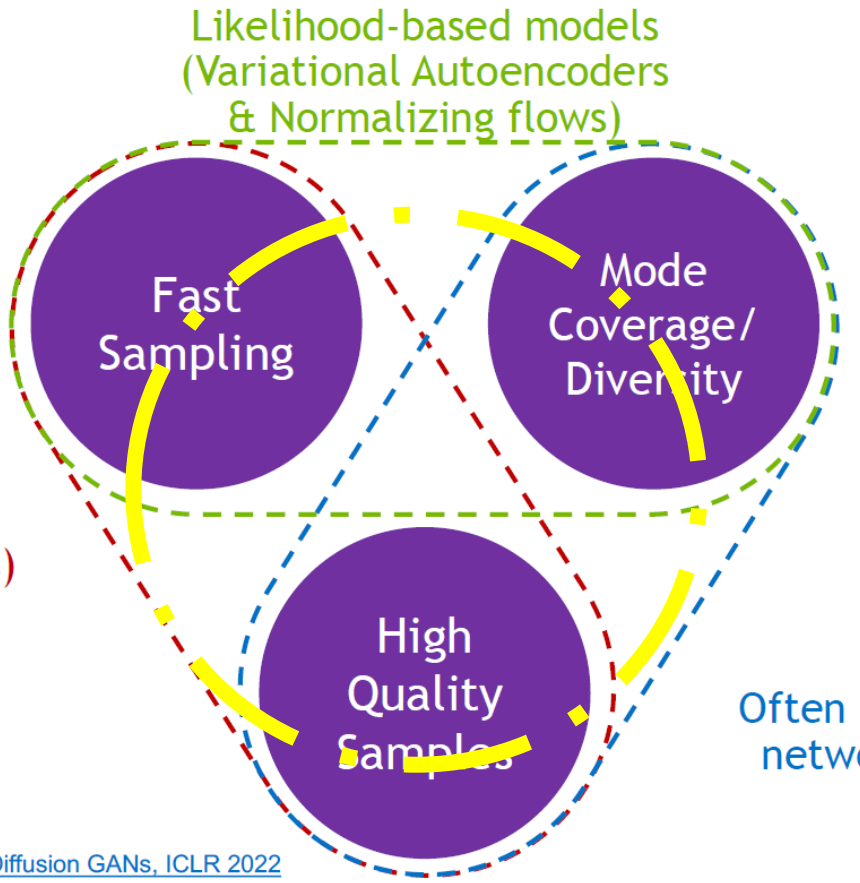

Introduction
Diffusion Denoising Models
Introduction to Advanced techniques
Conditioning Diffusion Models
Applications

The generative learning trilemma



Tackling the Generative Learning Trilemma with Denoising Diffusion GANs, ICLR 2022

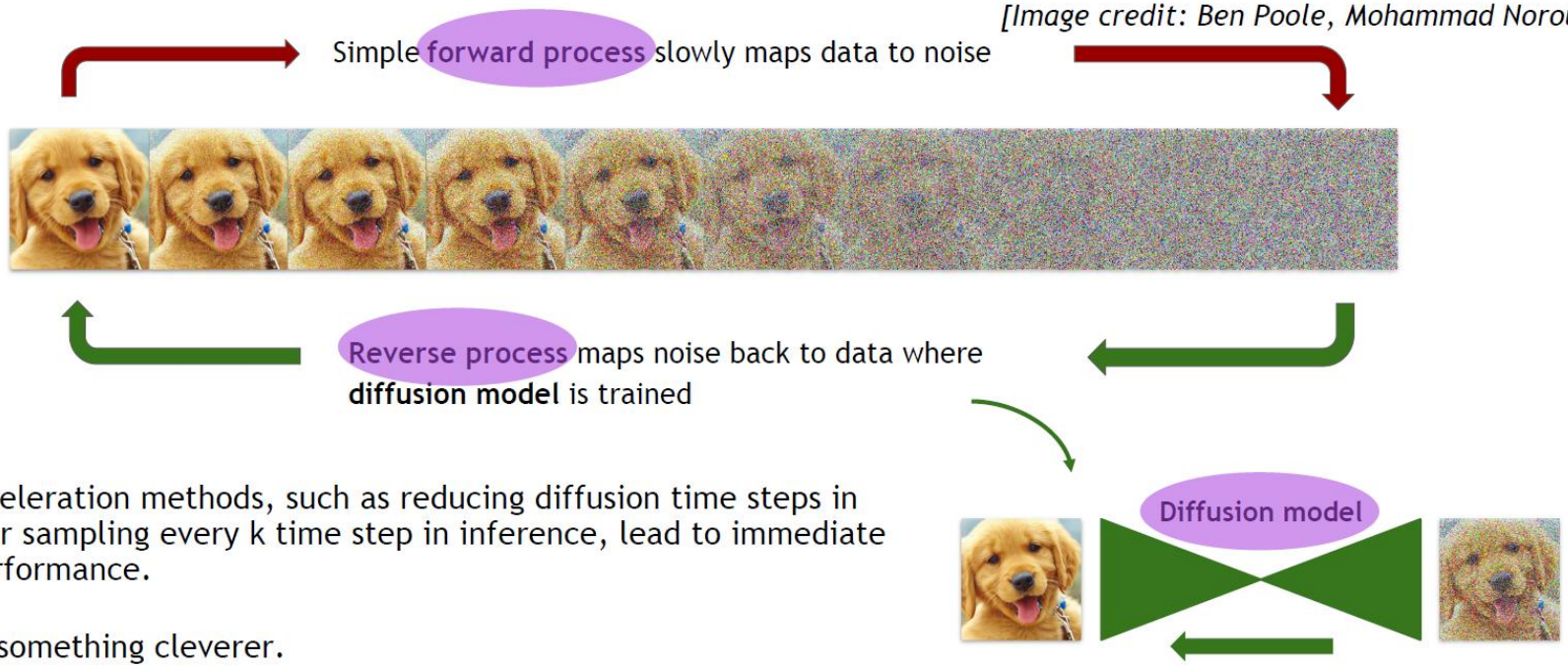
The generative learning trilemma



Tackling the Generative Learning Trilemma with Denoising Diffusion GANs, ICLR 2022

Accelerating the diffusion model for tackle the trilemma

[Image credit: Ben Poole, Mohammad Norouzi]



- Naïve acceleration methods, such as reducing diffusion time steps in training or sampling every k time step in inference, lead to immediate worse performance.
- We need something cleverer.
- Given a limited number of functional calls, usually much less than 1000s, how to improve performance?

Sketch of possibilities and existing solutions (1)

- 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
- Noise schedule is a sigmoid of a monotonic MLP
- Analogous to hierarchical VAE where encoder includes learnable parameters

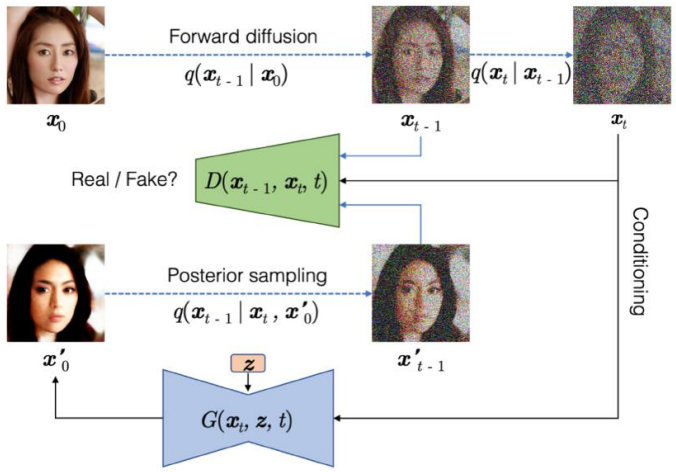
❖ Denoising diffusion implicit models (DDIM)

- Non-Markovian process;
- Deterministic generative process.

❖ Denoising diffusion GANs

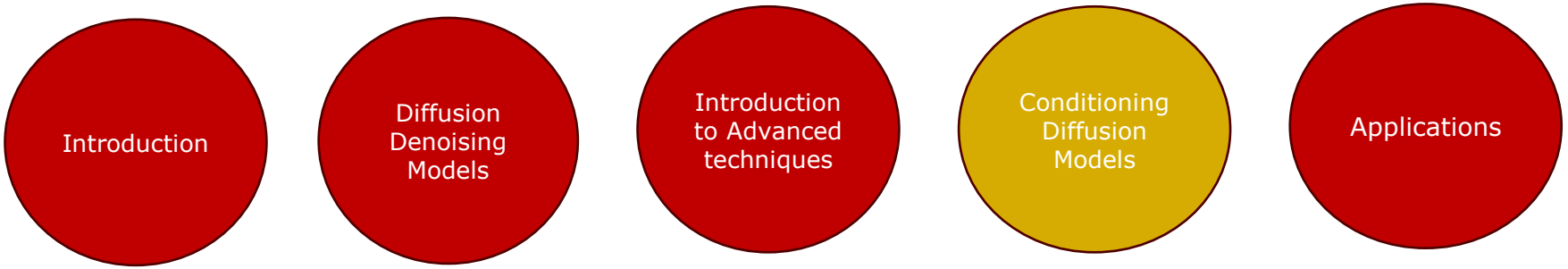
- Generative process is approximated by conditional GANs.

$$\min_{\theta} \sum_{t \geq 1} \mathbb{E}_{q(\mathbf{x}_t)} \left[D_{\text{adv}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t) \| p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \right]$$



Xiao et al., "Tackling the Generative Learning Trilemma with Denoising Diffusion GANs", ICLR 2022.

Part 4


Introduction
Diffusion Denoising Models
Introduction to Advanced techniques
Conditioning Diffusion Models
Applications

Conditional diffusion models examples (1)

Text-to-Image generation

DALL·E 2

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

IMAGEN

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

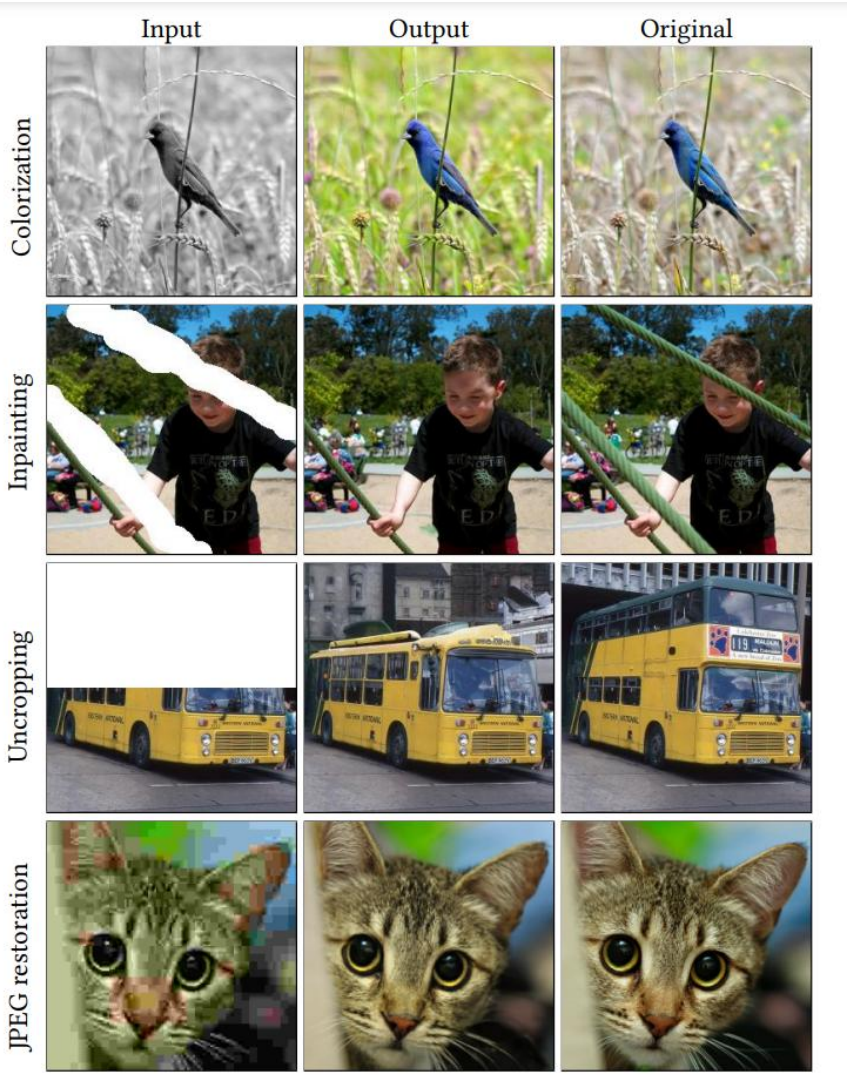
Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.
Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

Conditional diffusion models examples (2)

Image-to-Image diffusion models

Figure 9: Comparison of conditional and unconditional diffusion models for inpainting. Fig. C.7 in the Appendix shows more results.

Saharia et al., “Palette: Image-to-Image Diffusion Models”, arXiv 2021.



Conditional diffusion models examples (3)

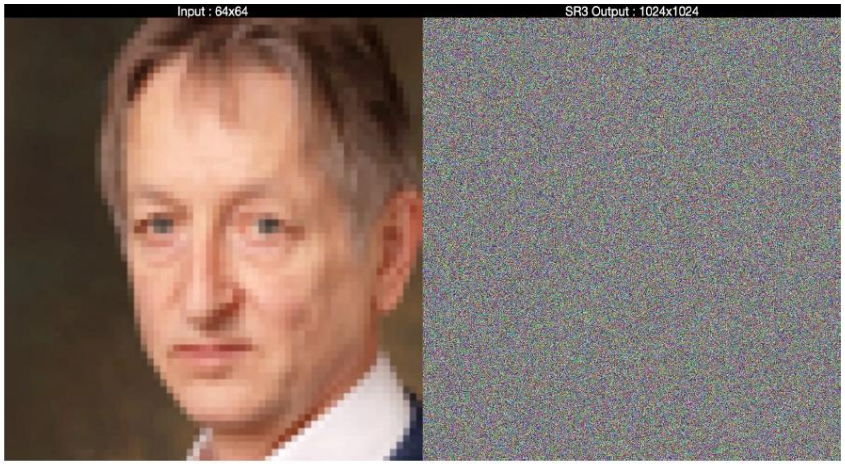
Panorama generation

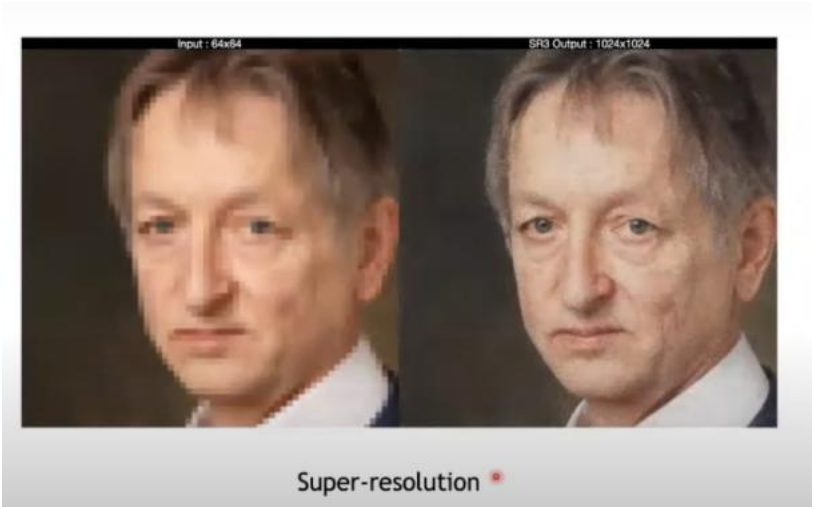
Figure 2: Given the central 256×256 pixels, we extrapolate to the left and right in steps of 128 pixels (2×8 applications of 50% Palette uncropping), to generate the final 256×2304 panorama. Figure D.3 in the Appendix shows more samples.

Saharia et al., "Palette: Image-to-Image Diffusion Models", arXiv 2021.

Conditional diffusion models examples (4)

Super resolution





Saharia et al., “Palette: Image-to-Image Diffusion Models”, arXiv 2021.

How to obtain conditional diffusion models

Reverse process: $p_\theta(\mathbf{x}_{0:T}|\mathbf{c}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t, \mathbf{c}), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t, \mathbf{c}))$

Variational upper bound: $L_\theta(\mathbf{x}_0|\mathbf{c}) = \mathbb{E}_q\left[L_T(\mathbf{x}_0) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c})) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1, \mathbf{c})\right].$

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 seq of vector embeddings - cross-attention.

Classifier-guidance diffusion models

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

Main Idea

For class-conditional modeling of $p(\mathbf{x}_t|\mathbf{c})$, train an extra classifier $p(\mathbf{c}|\mathbf{x}_t)$

Mix its gradient with the diffusion/score model during sampling

Dhariwal and Nichol, "Diffusion models beat GANs on image synthesis", NeurIPS 2021.

Classifier-guidance diffusion models

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

Main Idea

Sample with a modified score: $\nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t|\mathbf{c}) + \omega \log p(\mathbf{c}|\mathbf{x}_t)]$

Approximate samples from the distribution $\tilde{p}(\mathbf{x}_t|\mathbf{c}) \propto p(\mathbf{x}_t|\mathbf{c})p(\mathbf{c}|\mathbf{x}_t)^\omega$

Dhariwal and Nichol, “Diffusion models beat GANs on image synthesis”, NeurIPS 2021.

Classifier-free diffusion models (1)

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

$$p(\mathbf{c}|\mathbf{x}_t) \propto p(\mathbf{x}_t|\mathbf{c})/p(\mathbf{x}_t)$$

Conditional diffusion model — Unconditional diffusion model

- In practice, $p(\mathbf{x}_t|\mathbf{c})$ and $p(\mathbf{x}_t)$ by randomly dropping the condition of the diffusion model at certain chance.

- The modified score with this implicit classifier included is:

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

Ho & Salimans, "Classifier-Free Diffusion Guidance", 2021.

Classifier-free diffusion models (2)

Non-guidance

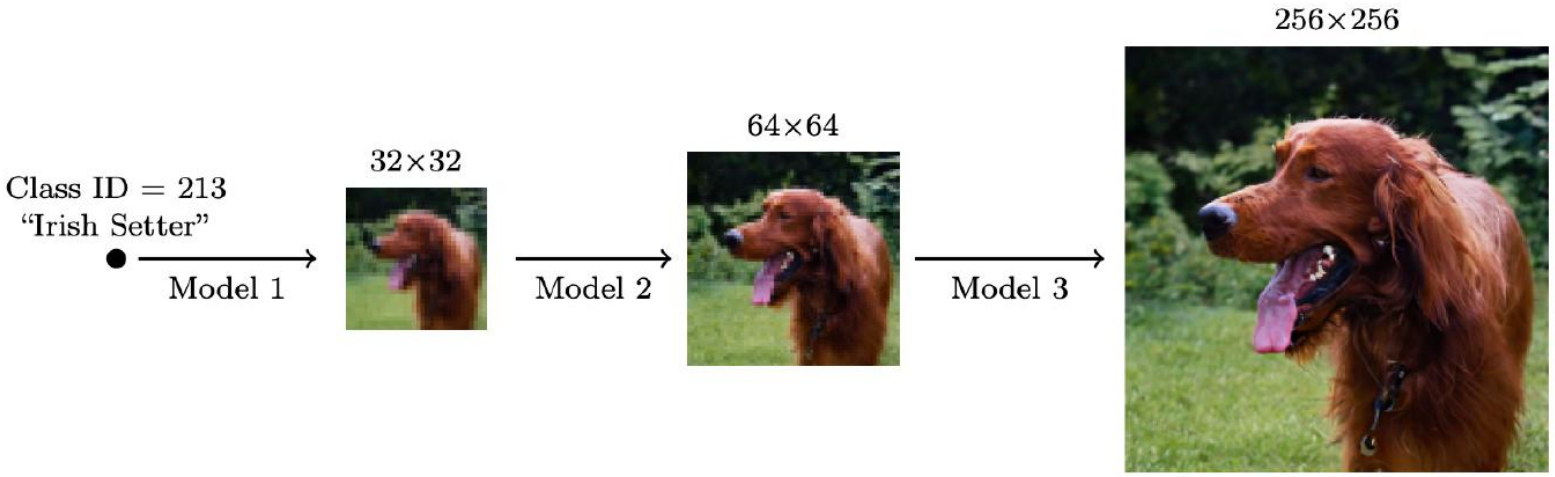
$\omega = 1$

$\omega = 3$

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

Ho & Salimans, “Classifier-Free Diffusion Guidance”, 2021.

Cascaded generation for super resolution



Cascaded Diffusion Models outperform Big-GAN in FID and IS and VQ-VAE2 in Classification Accuracy Score.

Ho et al., "Cascaded Diffusion Models for High Fidelity Image Generation", 2021.

Cascade generation details

- Need robust super-resolution model:
  - Training conditional on original low-res images from the dataset.
  - Inference on low-res images generated by the low-res model.

  } Mismatch issue

- Noise conditioning augmentation:
  - During training, add varying amounts of Gaussian noise (or blurring by Gaussian kernel) to the low-res images.
  - During inference, sweep over the optimal amount of noise added to the low-res images.
  - BSR-degradation process: applies JPEG compressions noise, camera sensor noise, different image interpolations for downsampling, Gaussian blur kernels and Gaussian noise in a random order to an image.

Ho et al., “Cascaded Diffusion Models for High Fidelity Image Generation”, 2021.
Nichol et al., “GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models”, 2021.

Part 5

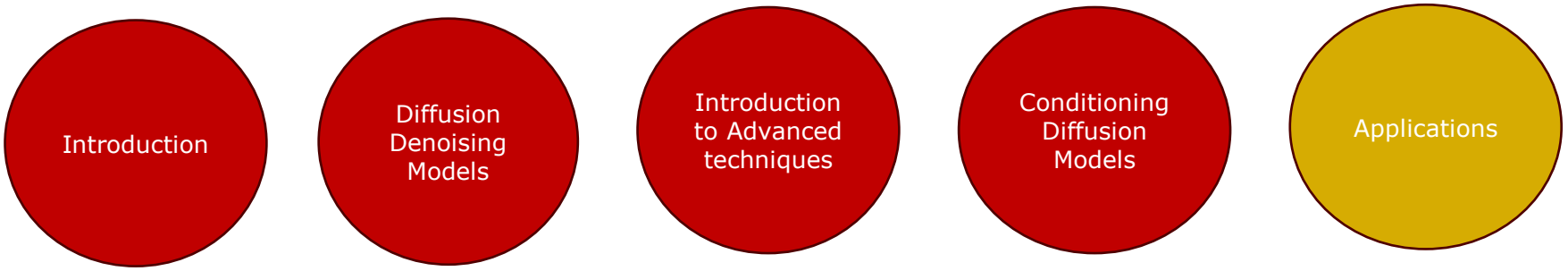

Introduction
Diffusion Denoising Models
Introduction to Advanced techniques
Conditioning Diffusion Models
Applications

Image Synthesis

Conditional generation: given a text prompt *c*, generate high-res images *x*.

A single beam of light enter the room from the ceiling. The beam of light is illuminating an easel. On the easel there is a Rembrandt painting of a raccoon.

https://imagen.research.google/

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

Glide (OpenAI) (1)

A 64x64 base model + a 64x64 → 256x256 super-resolution model.

Tried classifier-free and CLIP guidance. Classifier-free guidance works better than CLIP guidance.

"a hedgehog using a calculator"

"a corgi wearing a red bowtie and a purple party hat"

"robots meditating in a vipassana retreat"

"a fall landscape with a small cottage next to a lake"

Samples generated with classifier-free guidance (256x256)

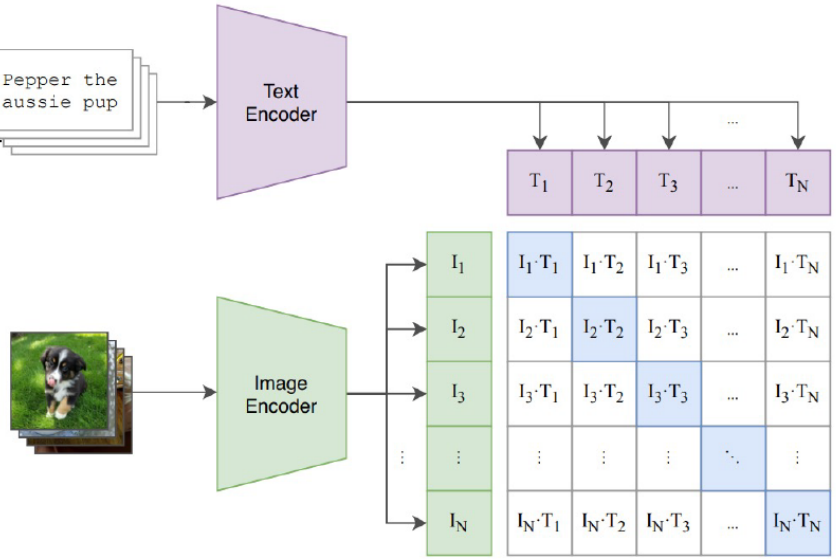
Nichol et al., "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models", 2021.

Glide (2) : CLIP guidance

- (CLIP) Contrastive Language–Image Pre-training
- Trained on image captioning with contrastive approach

- Trained by contrastive cross-entropy loss:

$$-\log \frac{\exp(f(\mathbf{x}_i)\cdot g(\mathbf{c}_j)/\tau)}{\sum_k \exp(f(\mathbf{x}_i)\cdot g(\mathbf{c}_k)/\tau)} - \log \frac{\exp(f(\mathbf{x}_i)\cdot g(\mathbf{c}_j)/\tau)}{\sum_k \exp(f(\mathbf{x}_k)\cdot g(\mathbf{c}_j)/\tau)}$$



- The optimal value of $f(\mathbf{x})\cdot g(\mathbf{c})$ is

$$\log \frac{p(\mathbf{x},\mathbf{c})}{p(\mathbf{x})p(\mathbf{c})} = \log p(\mathbf{c}|\mathbf{x}) - \log p(\mathbf{c})$$

Radford et al., “Learning Transferable Visual Models From Natural Language Supervision”, 2021.
Nichol et al., “GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models”, 2021.

Glide (3) : How do we get CLIP guidance with Diffusion models?

Replace classifier with CLIP model in classifier-guidance

- Sample with a modified score:

$$\nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t|\mathbf{c}) + \omega \log p(\mathbf{c}|\mathbf{x}_t)]$$

$$= \nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t|\mathbf{c}) + \omega(\underbrace{\log p(\mathbf{c}|\mathbf{x}_t) - \log p(\mathbf{c})}_{\text{CLIP model}})]$$

$$= \nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t|\mathbf{c}) + \omega(f(\mathbf{x}_t) \cdot g(\mathbf{c}))]$$

Radford et al., “Learning Transferable Visual Models From Natural Language Supervision”, 2021.
Nichol et al., “GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models”, 2021.

Glide (4) : Image Inpainting

- Fine-tune the model especially for inpainting: feed randomly occluded images with an additional mask channel as the input.

"an old car in a snowy forest"

"a man wearing a white hat"

Text-conditional image inpainting examples

Nichol et al., "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models", 2021.

Acknowledgments

- This lecture has been realized re-adapting slides from the CVPR tutorial on diffusion models accessible at:

https://cvpr2022-tutorial-diffusion-models.github.io/