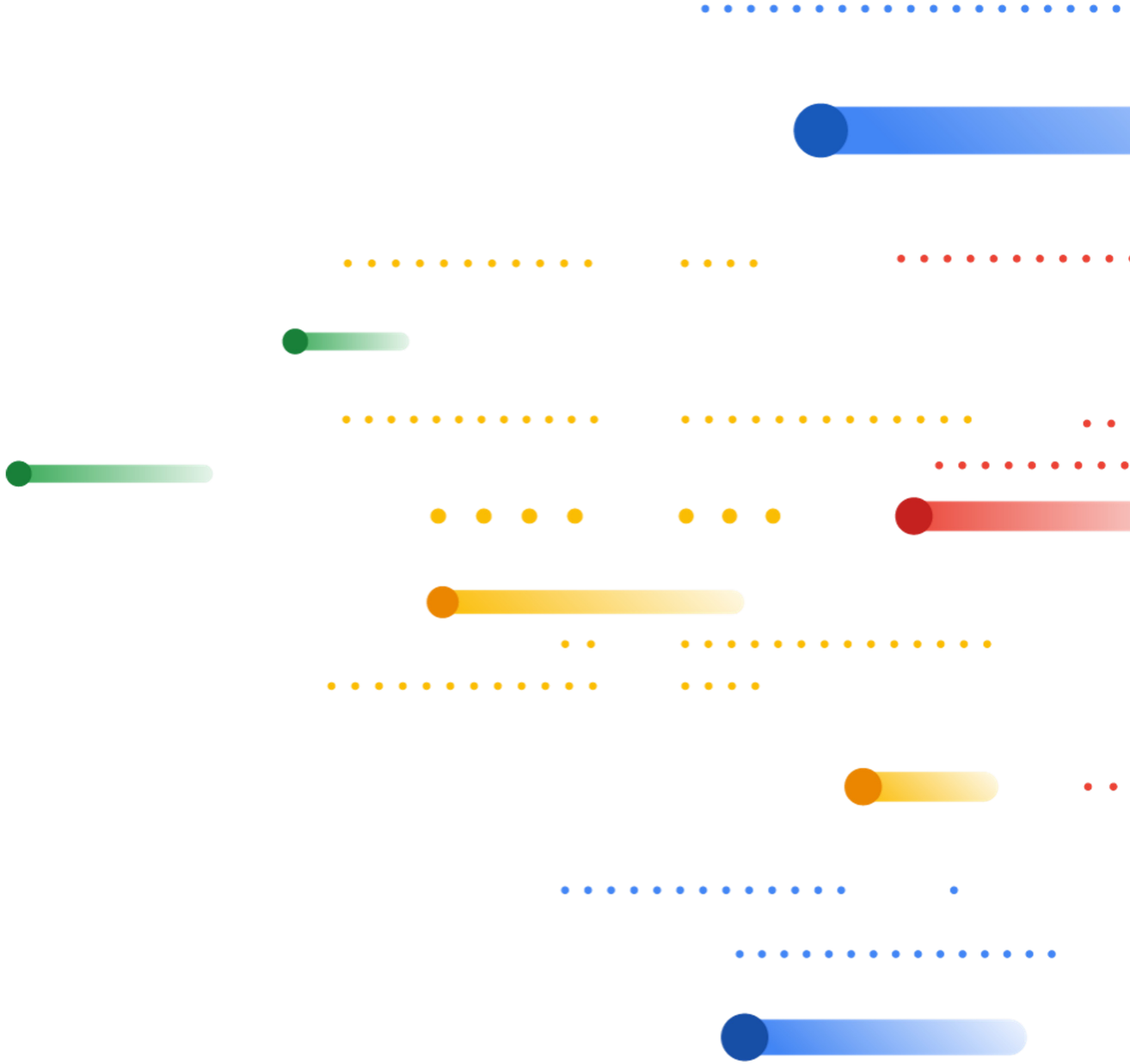
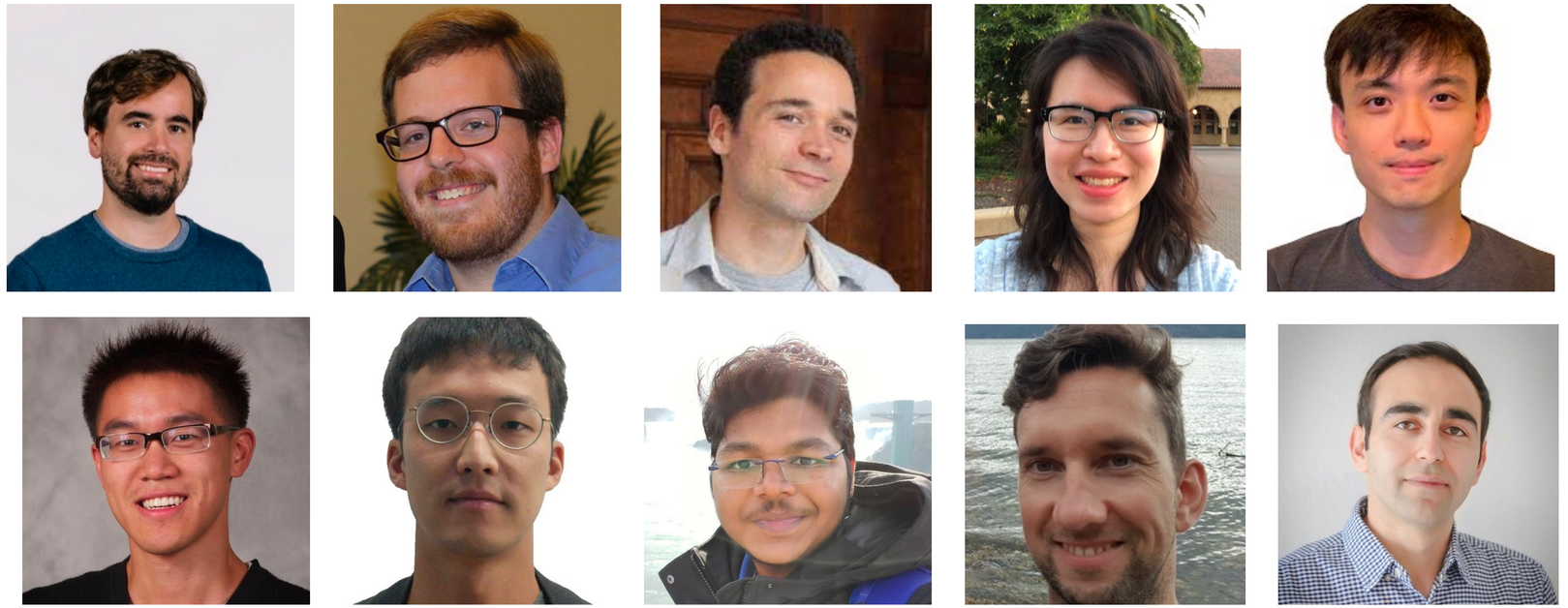
Google DeepMind

Variational Autoencoders and Diffusion Models

Ruiqi Gao @Stanford cs231n
May 25, 2023



Acknowledgements



Some slides were borrowed from
Denoising Diffusion-based Generative Modeling CVPR2022 tutorial
(https://cvpr2022-tutorial-diffusion-models.github.io/)

Contents

- Deep generative models and applications in computer vision
- Variational Autoencoders
- Diffusion models
    - Discrete-time diffusion models
    - Continuous-time diffusion models: differential equation framework
- Case study: Imagen: high-fidelity text-to-image diffusion models

Not intended as a complete review of all recent work!

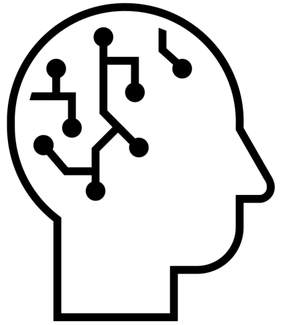
Deep Generative Models

Deep Generative Models

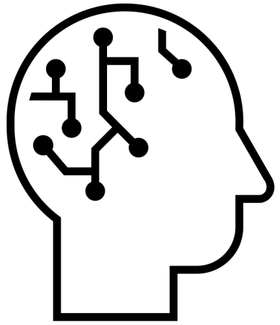
Learning to generate data

Samples from a Data Distribution

Train



Neural Network

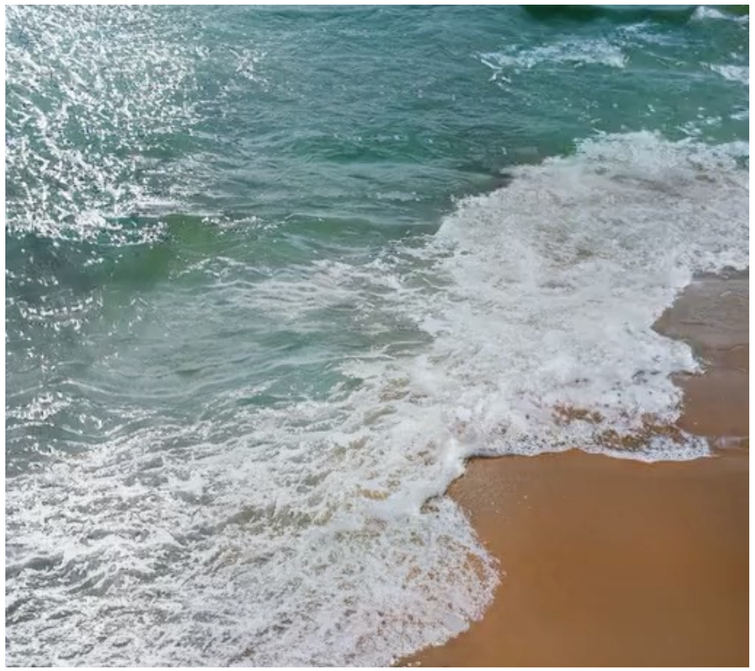


Sample

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

Application (1): Content Generation

StyleGAN3 example images



Karras et al. Alias-Free Generative Adversarial Networks, NeurIPS 2021
slide from https://cvpr2022-tutorial-diffusion-models.github.io/

Application (2): Representation Learning

Learning from limited labels

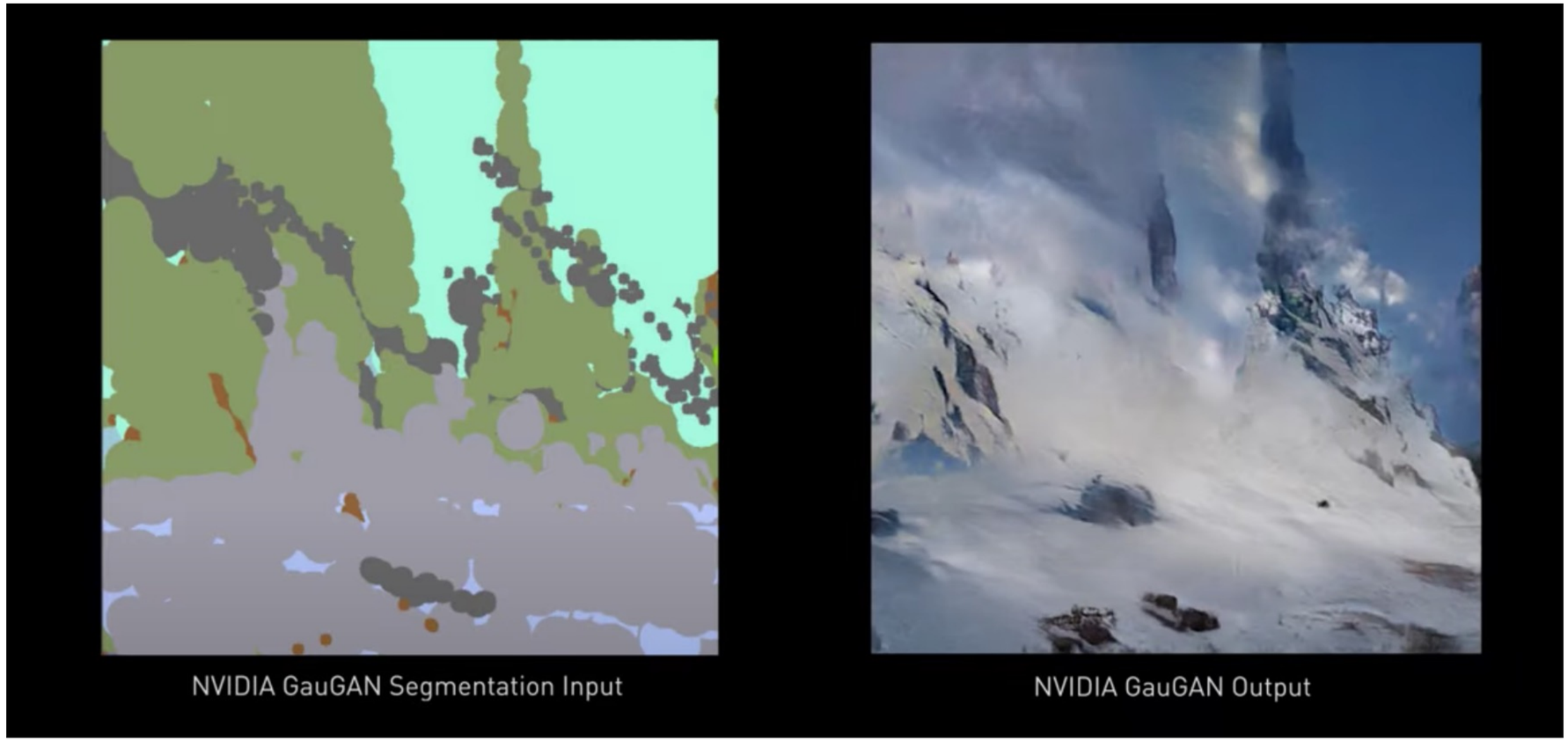
Zhang et al., DatasetGAN: Efficient Labeled Data Factory with Minimal Human Effort, CVPR 2021

Li et al., Semantic Segmentation with Generative Models: Semi-Supervised Learning and Strong Out-of-Domain Generalization, CVPR 2021

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

Application (3): Artistic Tools

NVIDIA GauGAN



Park et al., Semantic Image Synthesis with Spatially-Adaptive Normalization, CVPR 2019

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

2022 / 2023 : The year of generative modeling?

Parti

Pathways Autoregressive Text-to-Image Model

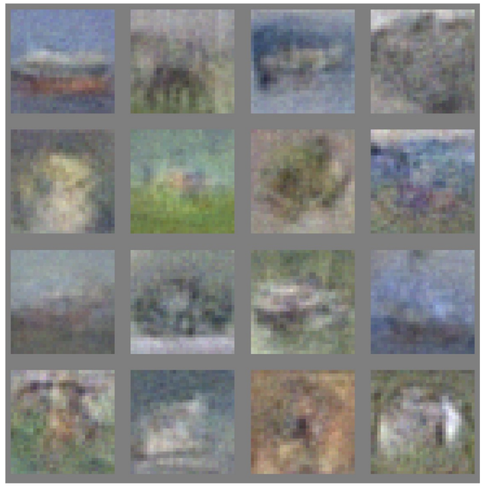
DALL·E 2

Stable Diffusion

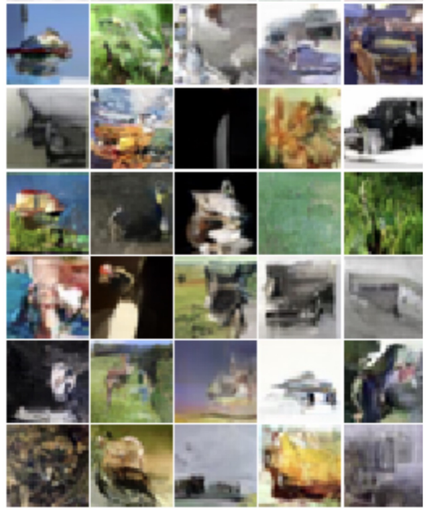
Where we came from

VAEs, 2013

GANs, 2014



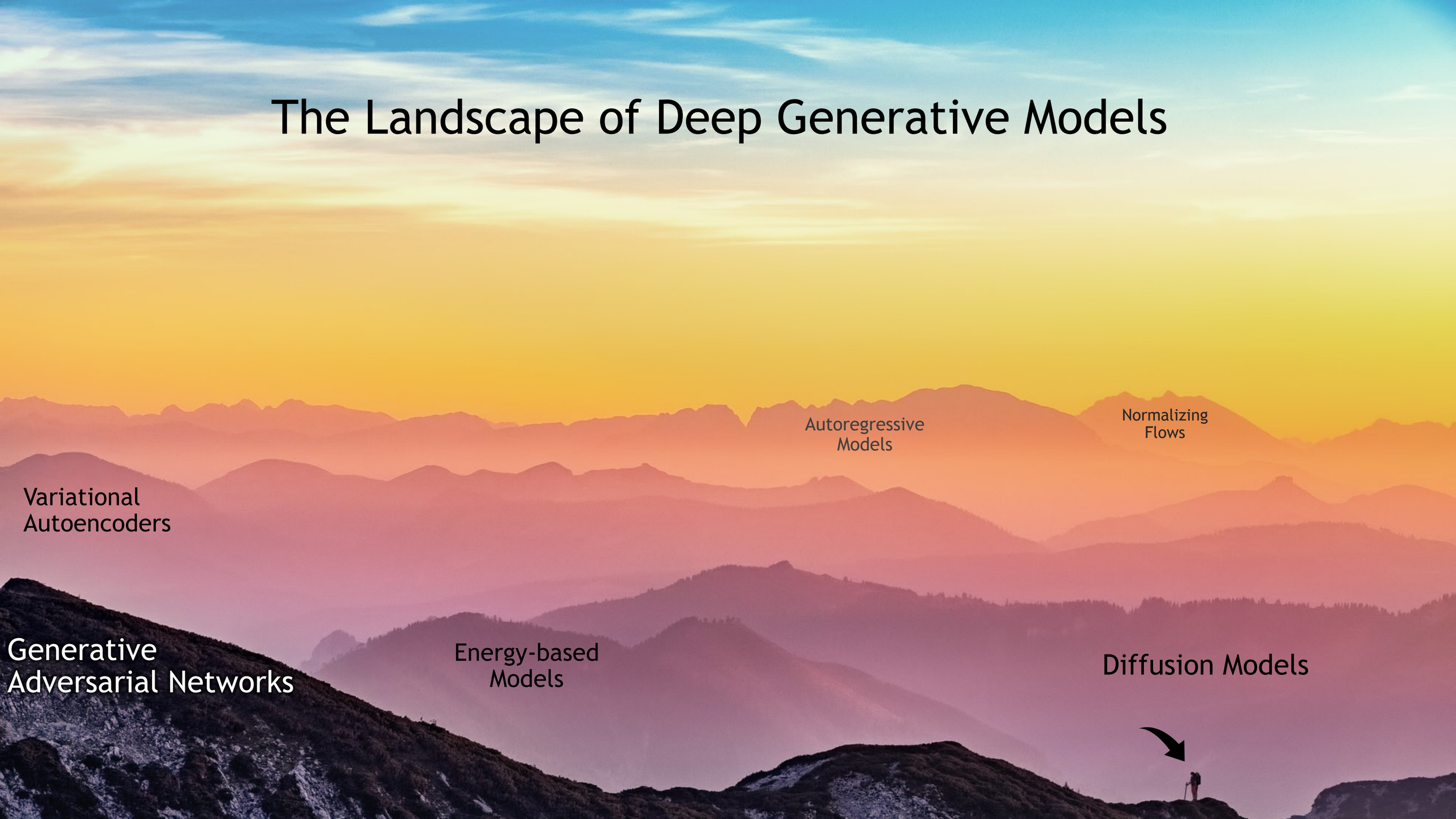
PixelCNN, 2016

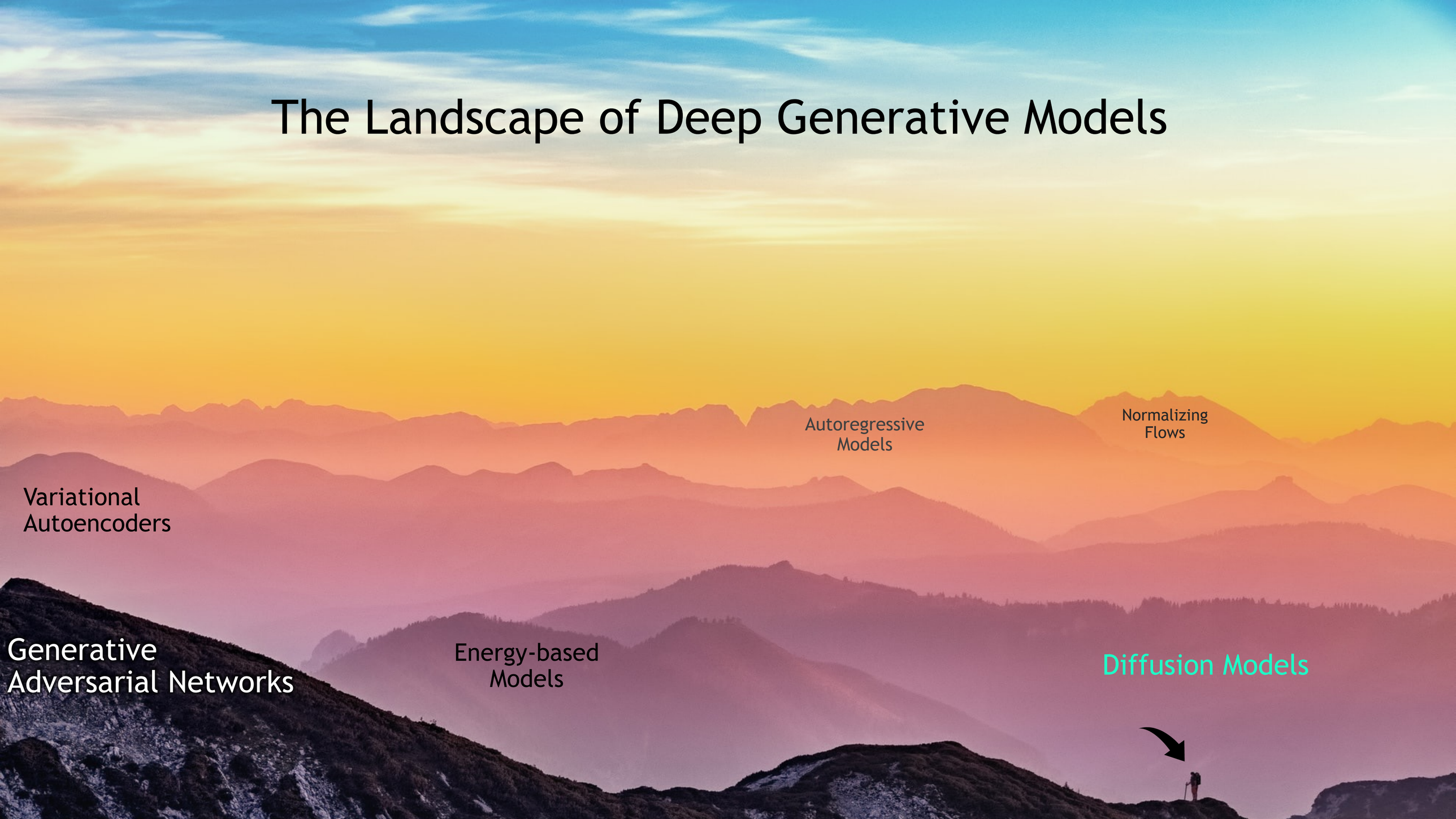


BigGAN, 2019

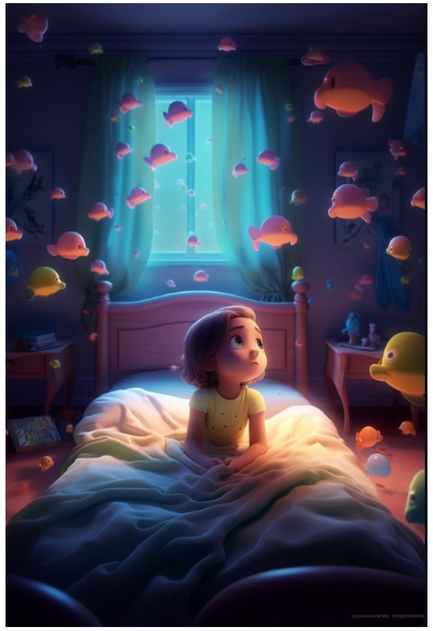
Imagen, 2022

The Landscape of Deep Generative Models



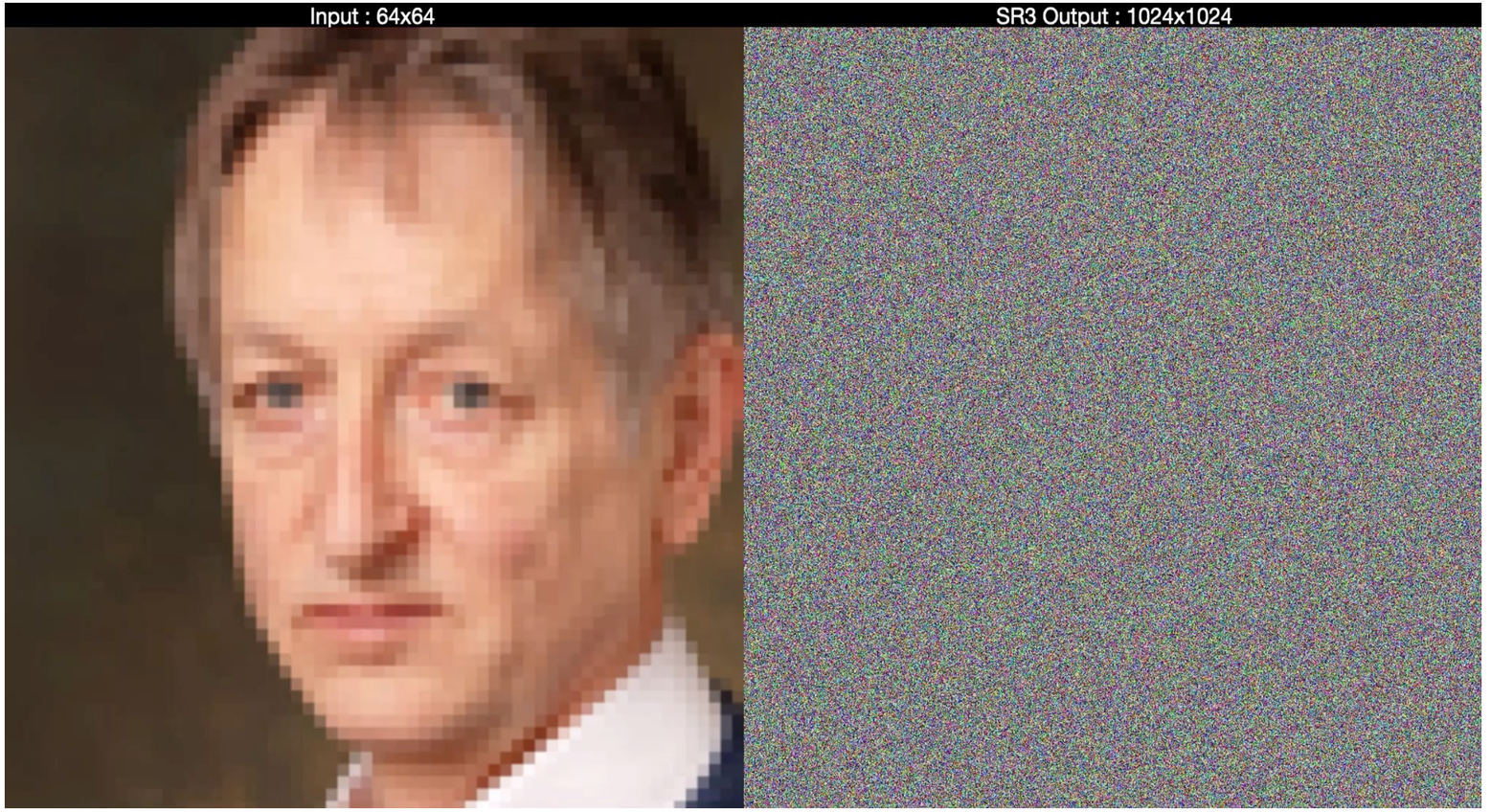

The Landscape of Deep Generative Models
Autoregressive Models
Normalizing Flows
Variational Autoencoders
Generative Adversarial Networks
Energy-based Models
Diffusion Models

AI Art



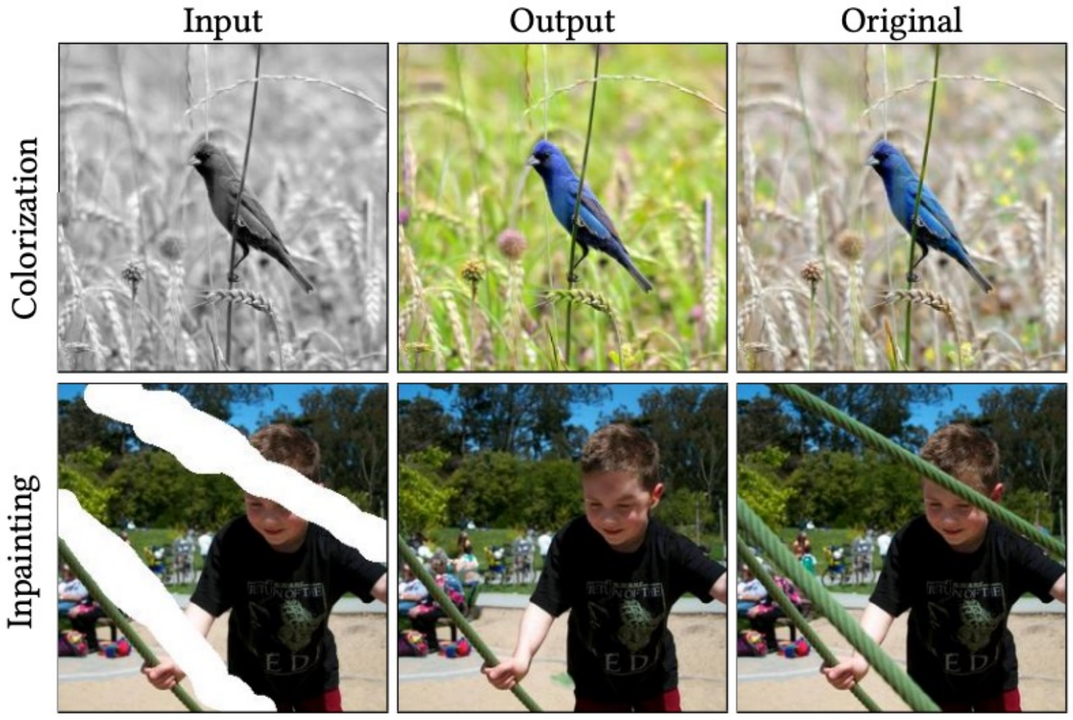
Midjourney v5

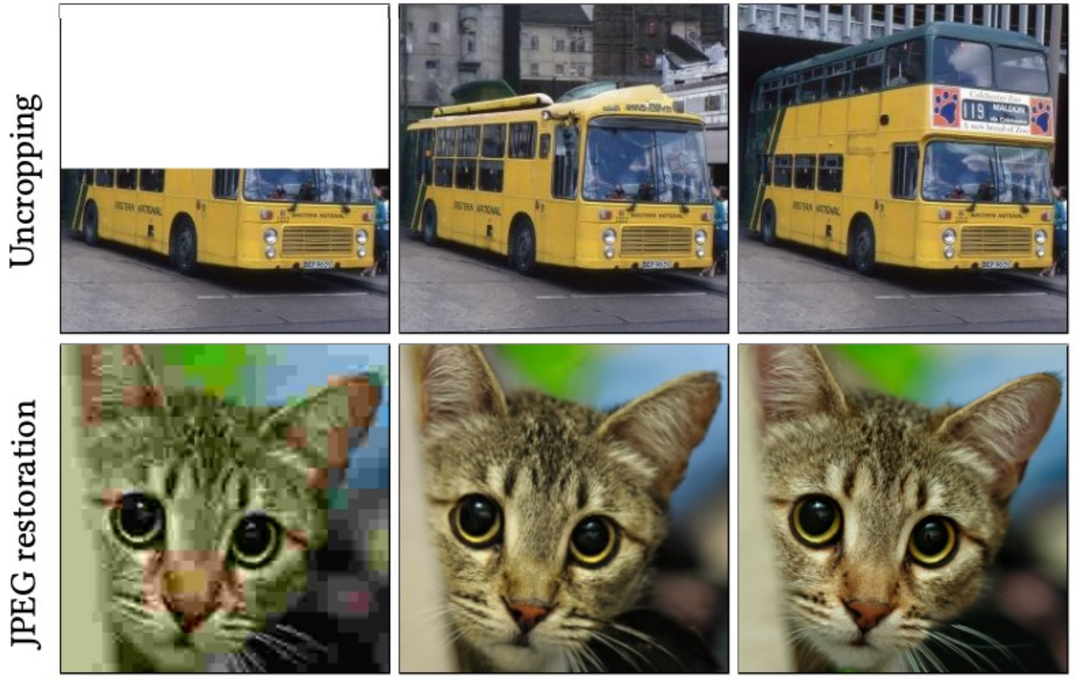
Applications: Super-resolution



Palette: Image-to-Image Diffusion Models, 2022

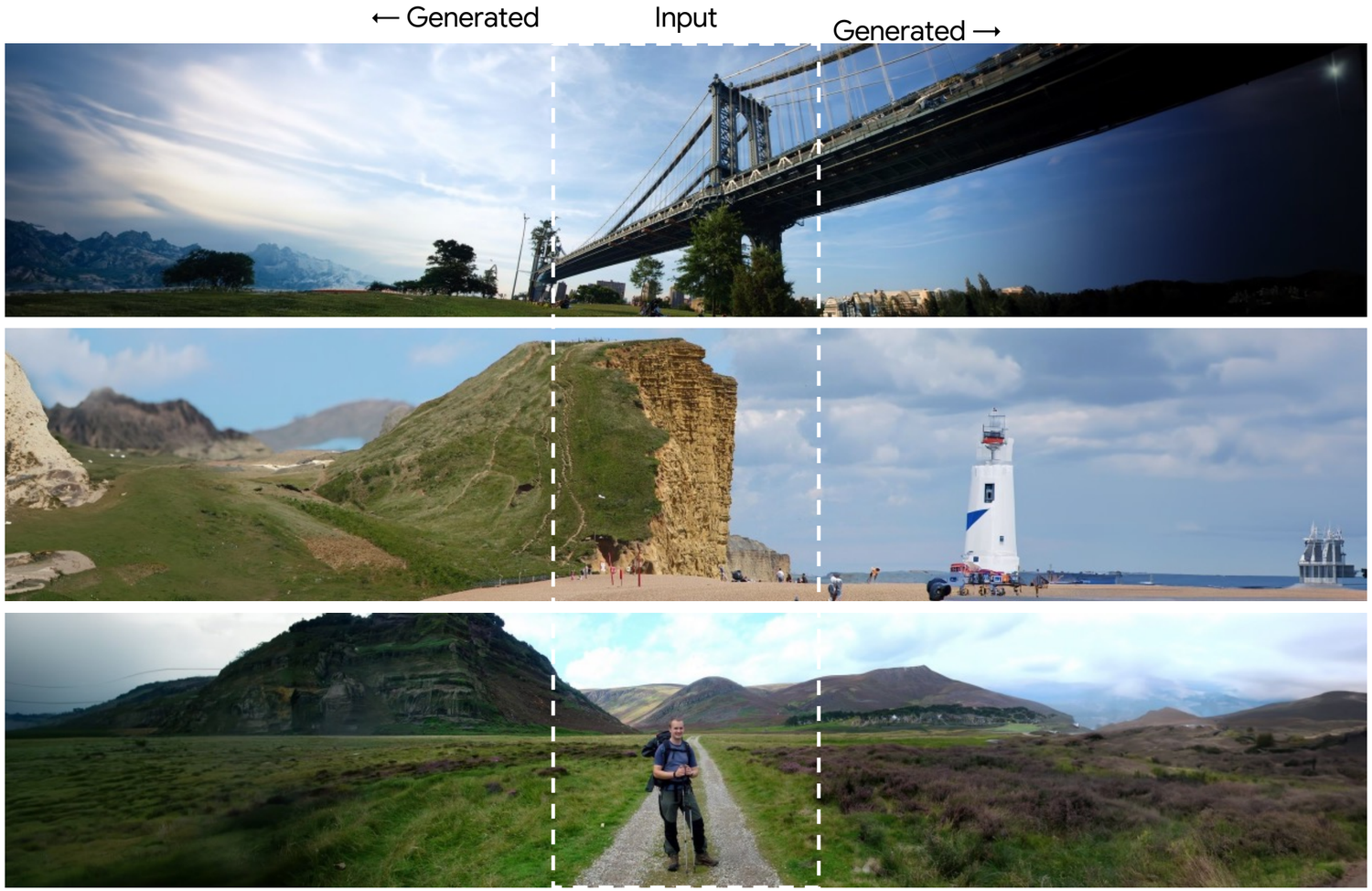
Applications: Colorization, Inpainting, Restoration





Palette: Image-to-Image Diffusion Models, 2022

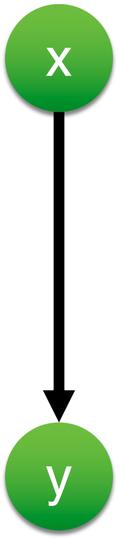
Applications: Outfilling



Palette: Image-to-Image Diffusion Models, 2022

Variational Autoencoders

Classifying chocolate



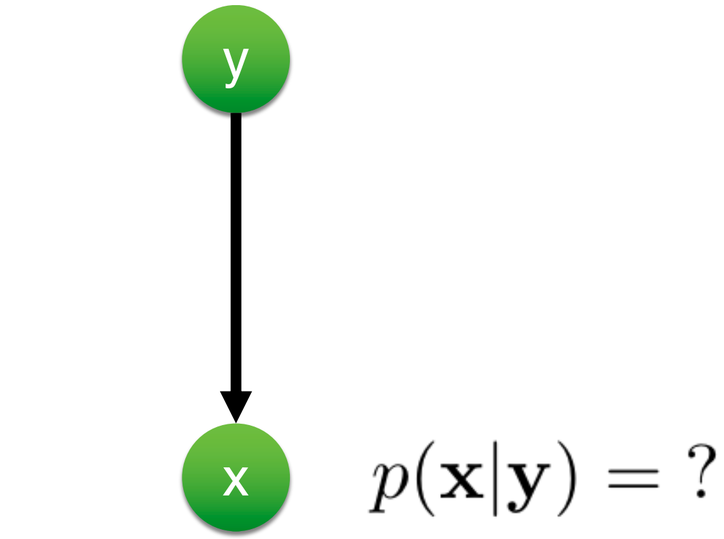
$$p(\mathbf{y}|\mathbf{x}) = \text{Categorical}(\mathbf{y}; \mathbf{f}(\mathbf{x}))$$

“chocolate”

slide by Durk Kingma

Generating chocolate

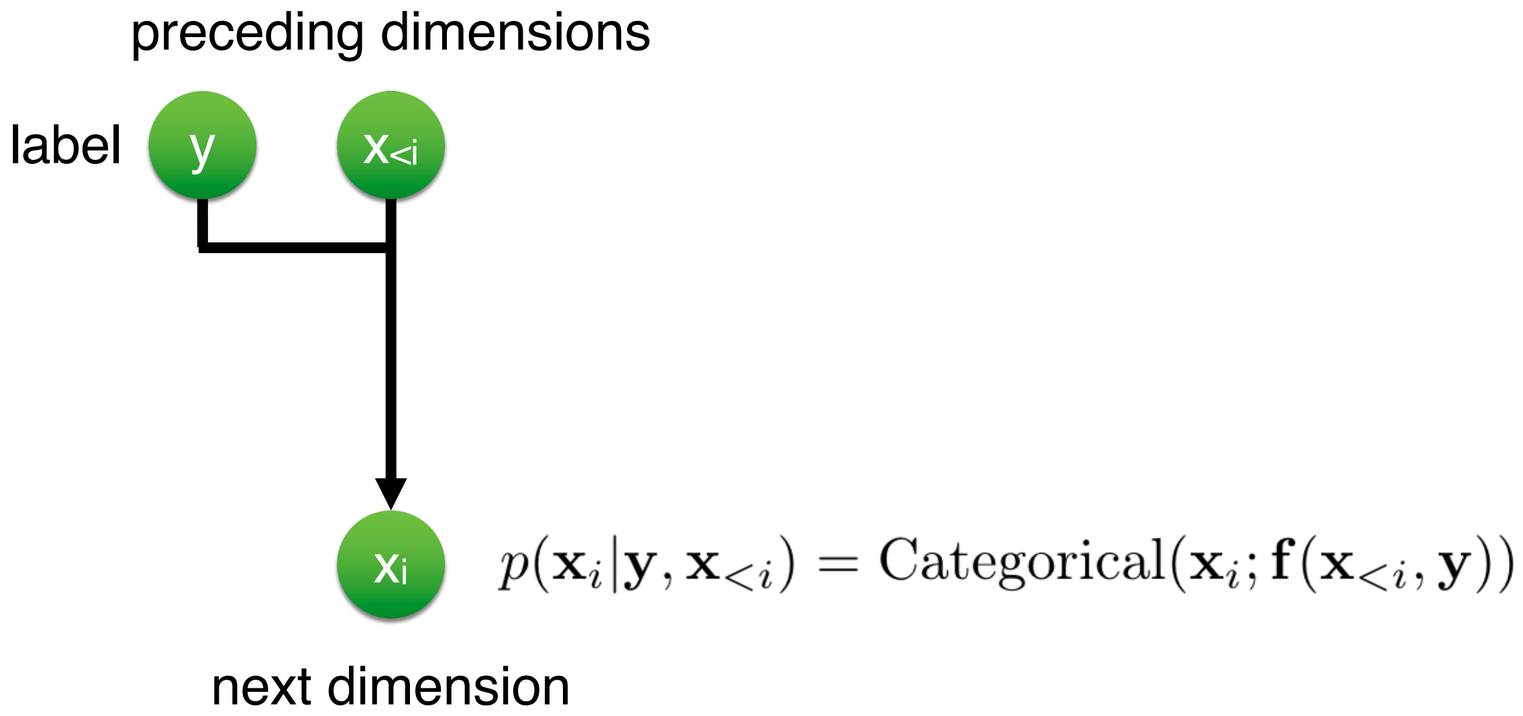
“chocolate”



$$p(\mathbf{x}|\mathbf{y}) = ?$$

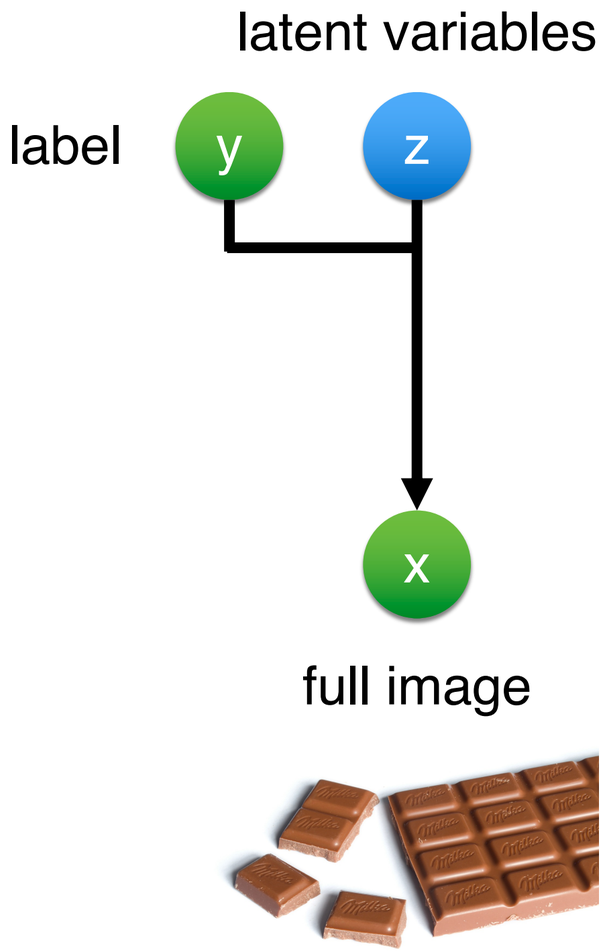
slide by Durk Kingma

Autoregressive approach



slide by Durk Kingma

Latent variable approach

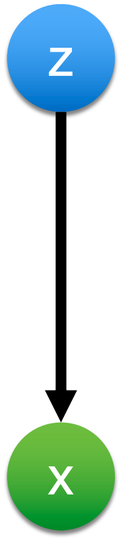


$$
\begin{aligned}
p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}) \\
p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z}) &= \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_\theta(\mathbf{z}, \mathbf{y}), \sigma^2 \mathbf{I}) \\
p_\theta(\mathbf{x}|\mathbf{y}) &= \int_{\mathbf{z}} p(\mathbf{x}|\mathbf{y}, \mathbf{z}) p(\mathbf{z})\, d\mathbf{z} \\
&= \text{flexible distribution}
\end{aligned}
$$

slide by Durk Kingma

Latent variable approach - without y

latent variables



full image

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

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_\theta(\mathbf{z}), \sigma^2\mathbf{I})$$

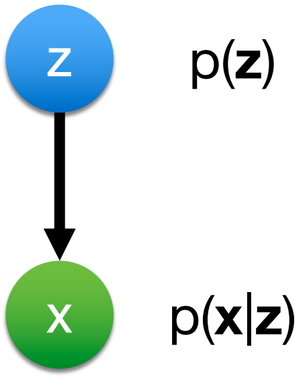
$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})\, d\mathbf{z}$$

$$= \text{flexible distribution}$$

slide by Durk Kingma

Optimization

$$
\begin{aligned}
p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}) \\
p_\theta(\mathbf{x}|\mathbf{z}) &= \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_\theta(\mathbf{z}), \sigma^2 \mathbf{I}) \\
p_\theta(\mathbf{x}) &= \int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})\, d\mathbf{z} \\
&= \text{flexible distribution}
\end{aligned}
$$



Generative model
p(x,z)

- Problem:
  - Marginal likelihood p(x) is intractable
  - So can't do maximum likelihood directly

slide by Durk Kingma

Variational Autoencoders (VAEs)

- We introduce an **inference model** q(z|x)

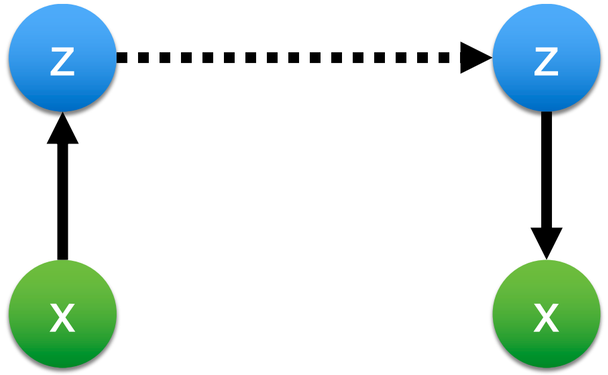
$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\Sigma}_\phi(\mathbf{x}))$$

- This allows us to efficiently optimize the log-likelihood, through the **evidence lower bound** (ELBO).

$$\log p_{\theta,\phi}(\mathbf{x}) \geq \mathrm{ELBO}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]$$

- We optimize q(z|x) and p(x,z) jointly w.r.t. ELBO

- Bound is tight with the right q(z|x)



Inference model
q(z|x)

Generative model
p(x,z)

slide by Durk Kingma

[Kingma and Welling, 2013]

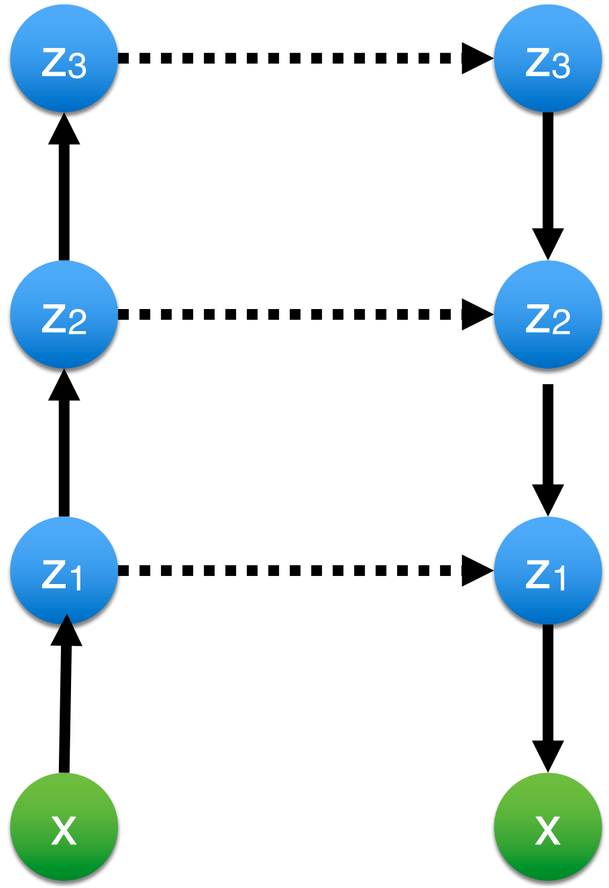
Hierarchical VAEs

- “Flat” VAEs suffer from simple priors
- Making both inference model and generative model hierarchical

$$q_\phi(\mathbf{z}_{1,2,3}|\mathbf{x}) = q_\phi(\mathbf{z}_1|\mathbf{x})q_\phi(\mathbf{z}_2|\mathbf{z}_1)q_\phi(\mathbf{z}_3|\mathbf{z}_2)$$

$$p_\theta(\mathbf{z}_{1,2,3}) = p_\theta(\mathbf{z}_3)p_\theta(\mathbf{z}_2|\mathbf{z}_3)p_\theta(\mathbf{z}_1|\mathbf{z}_2)p_\theta(\mathbf{x}|\mathbf{z}_1)$$

- Better likelihoods are achieved with hierarchies of latent variables



Inference model
q(z|x)

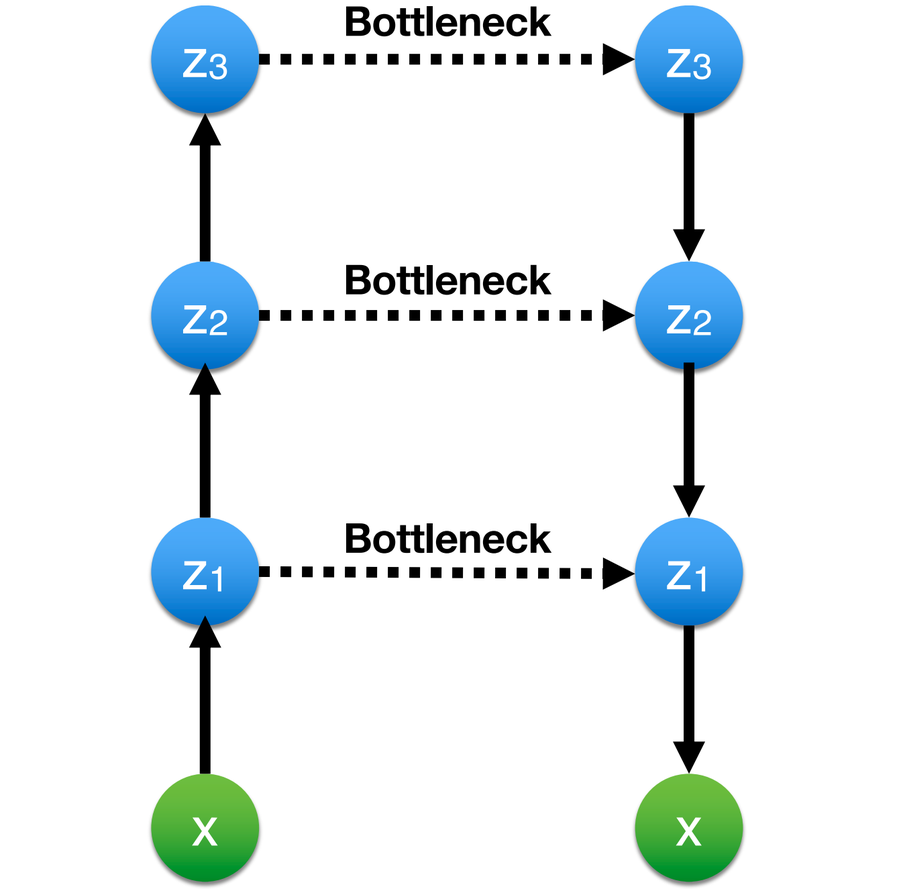
Generative model
p(x,z)

slide by Durk Kingma

[Kingma and Welling, 2017]

VAEs: challenges

- Optimization can be difficult for large models
- The ELBO enforces an **information bottleneck** (through its loss function) at the latent variables 'z', making VAE optimization prone to **bad local minima**.
- **Posterior collapse** is a dreaded bad local minimum where the latents do not transmit any information.



slide by Durk Kingma

[Kingma and Welling, 2013]

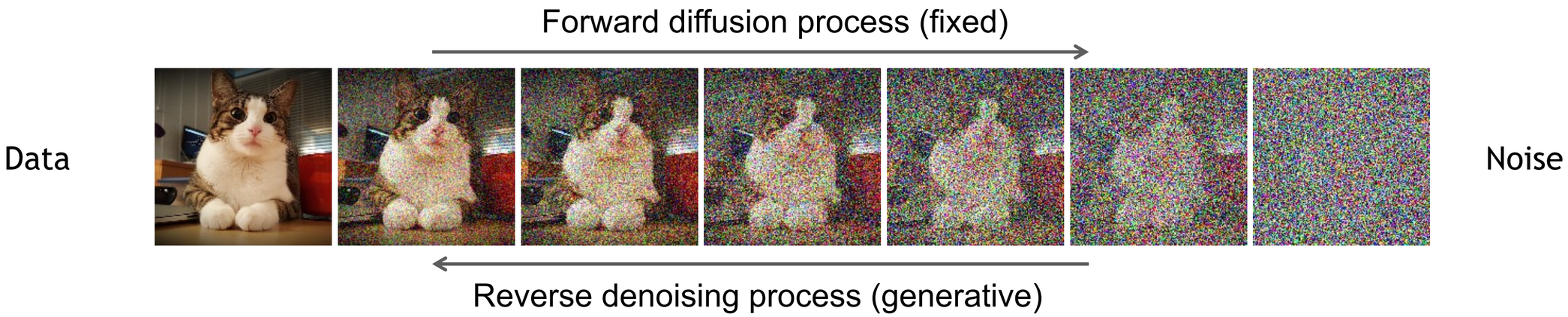
Diffusion Models

Denoising Diffusion Models

Learning to generate by denoising

Denoising diffusion models consist of two processes:

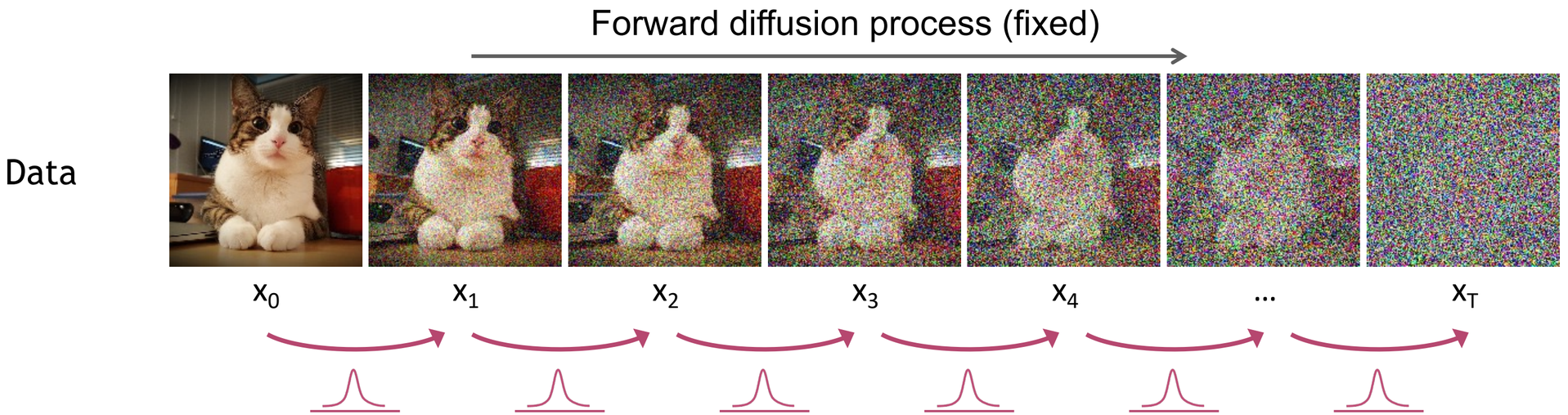
- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015
Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021
slide from https://cvpr2022-tutorial-diffusion-models.github.io/

Forward Diffusion Process

The formal definition of the forward process in T steps:

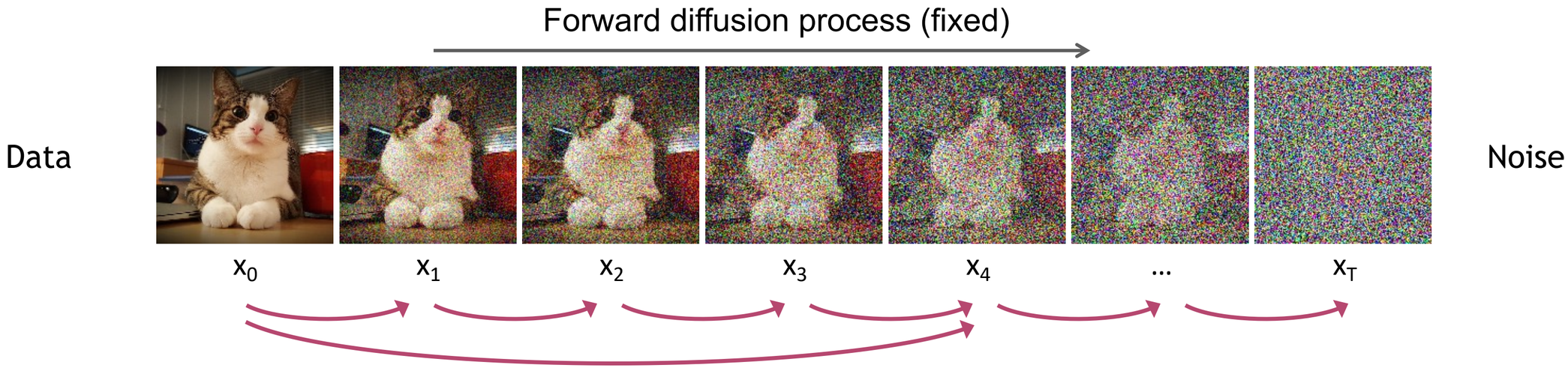


$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

Similar to the inference model in hierarchical VAEs.

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

Diffusion Kernel



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

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

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

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

Generative Learning by Denoising

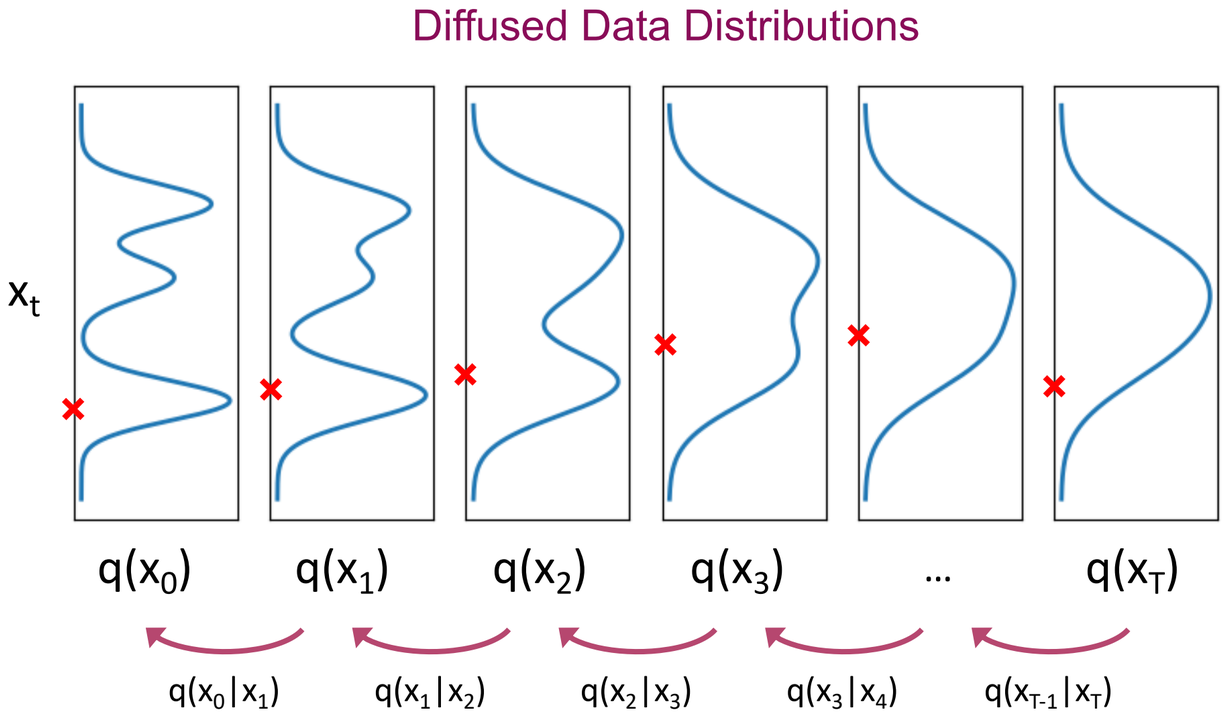
Recall, that the diffusion parameters are designed such that $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Generation:

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

Iteratively sample $\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1}|\mathbf{x}_t)}$

True Denoising Dist.



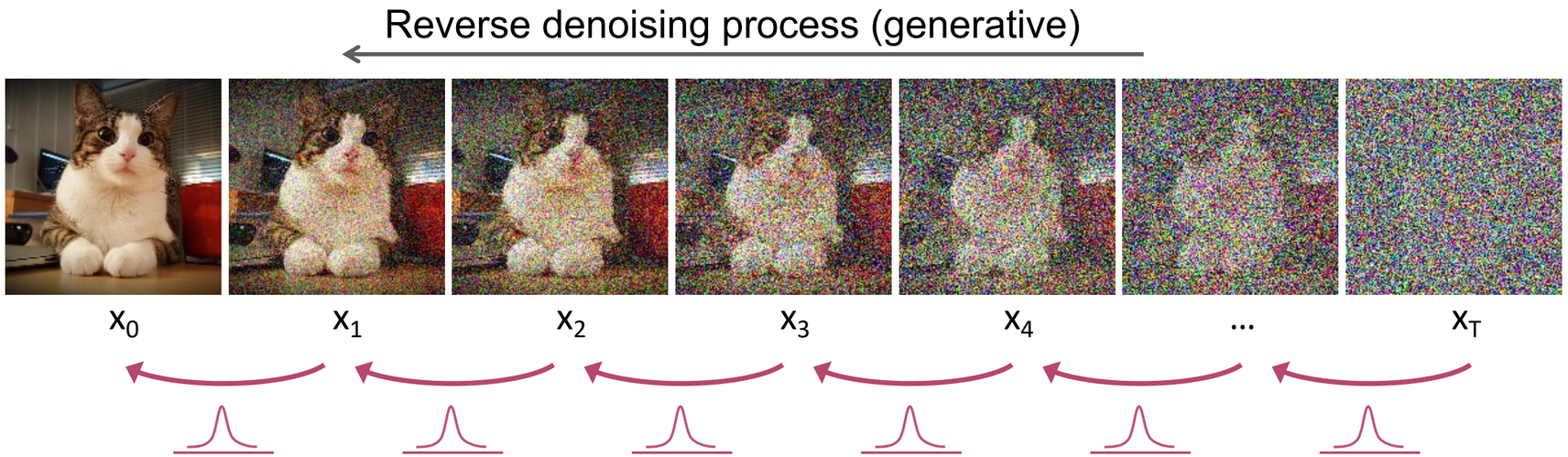
In general, $q(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is intractable.

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

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

Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



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

Trainable network
(U-net, Denoising Autoencoder)

➡

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

Similar to the generative model in hierarchical VAEs.

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

Learning Denoising Model

Variational upper bound

For training, we can form variational upper bound (negative ELBO) 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:

$$L = \mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1))}_{L_0}\right]$$

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

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

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

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

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:

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

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:

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

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

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

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

Training Objective Weighting

Trading likelihood for perceptual quality

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

The time dependent λ_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:

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

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

Summary

Training and Sample Generation

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

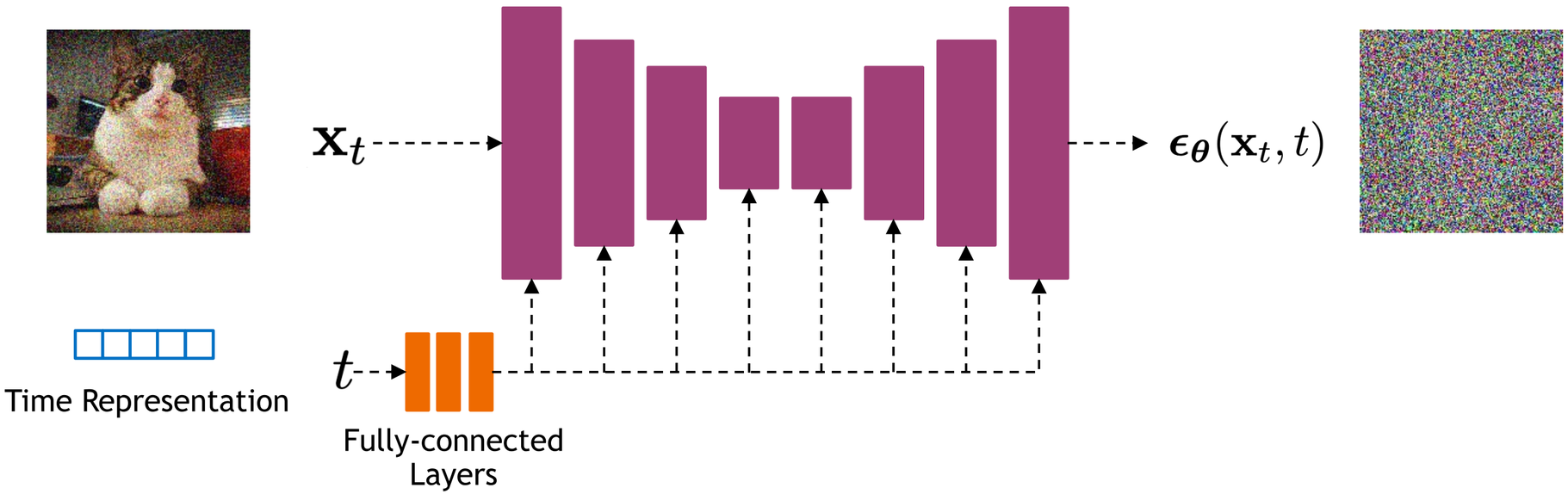
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,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

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

Implementation Considerations

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)

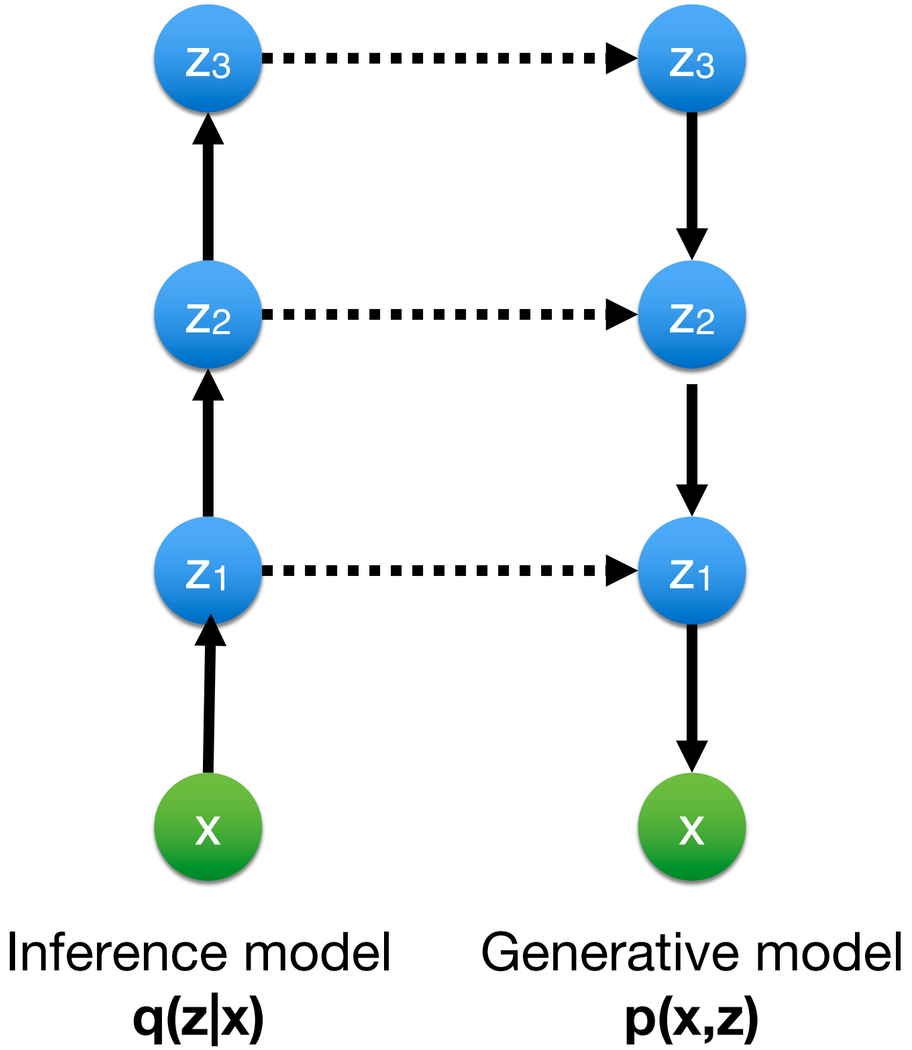
slide from https://cvpr2022-tutorial-diffusion-models.github.io/

Connection to VAEs

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

However, in diffusion models:

- The inference model is fixed: easier to optimize
- The latent variables have the same dimension as the data.
- The ELBO is decomposed to each time step: fast to train
    - Can be made extremely deep (even infinitely deep)
- The model is trained with some reweighting of the ELBO.

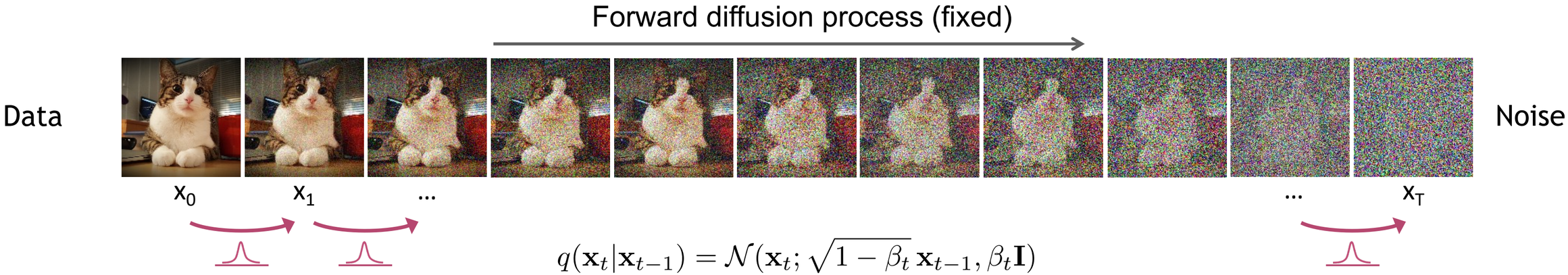


Vahdat and Kautz, NVAE: A Deep Hierarchical Variational Autoencoder, NeurIPS 2020
Sønderby, et al.. Ladder variational autoencoders, NeurIPS 2016.

Continuous-time diffusion models

Stochastic differential equation framework

Consider the limit of **many small steps:**



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

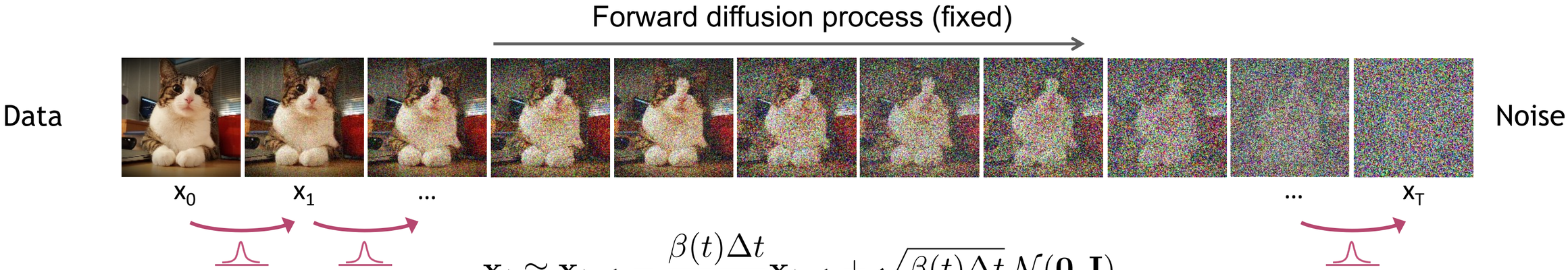
$$\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\,\mathbf{x}_{t-1} + \sqrt{\beta_t}\,\mathcal{N}(\mathbf{0},\mathbf{I}) \\
&= \sqrt{1-\beta(t)\Delta t}\,\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0},\mathbf{I}) && (\beta_t := \beta(t)\Delta t) \\
&\approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0},\mathbf{I}) && \text{(Taylor expansion)}
\end{aligned}$$

Song et al., “Score-Based Generative Modeling through Stochastic Differential Equations”, *ICLR*, 2021

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

Forward Diffusion Process as Stochastic Differential Equation

Consider the limit of **many small steps:**



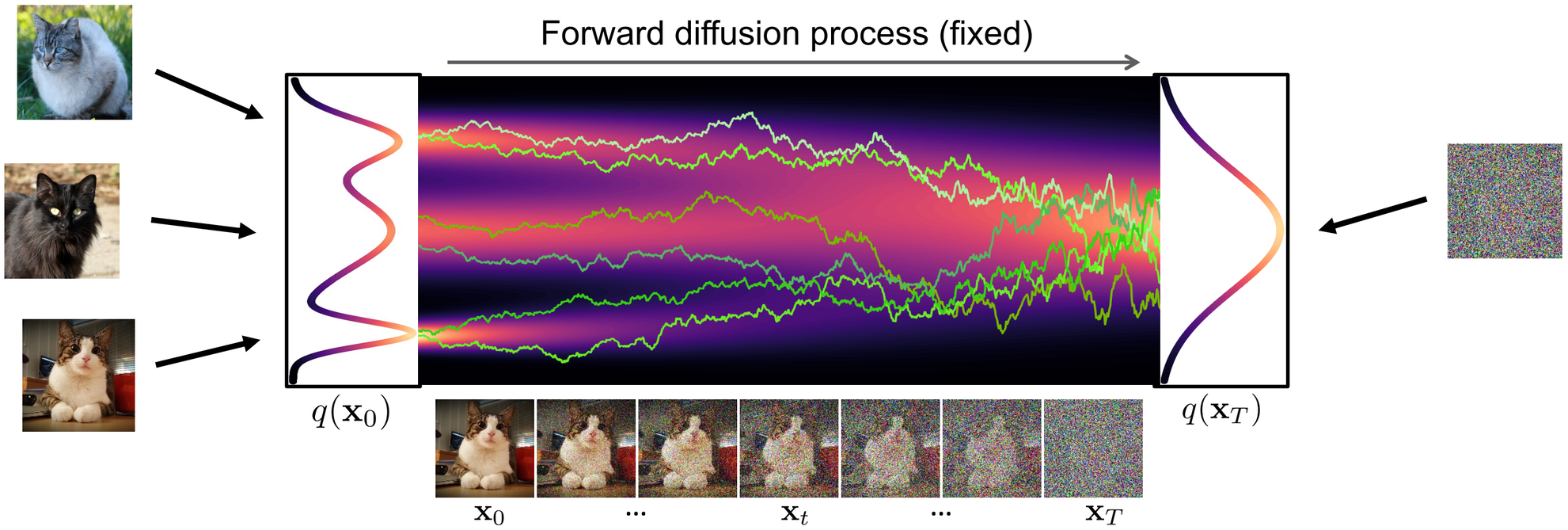
$$\mathbf{x}_t \approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

Stochastic Differential Equation (SDE)
describing the diffusion in infinitesimal limit

Song et al., "Score-Based Generative Modeling through Stochastic Differential Equations", *ICLR*, 2021
slide from https://cvpr2022-tutorial-diffusion-models.github.io/

Forward Diffusion Process as Stochastic Differential Equation



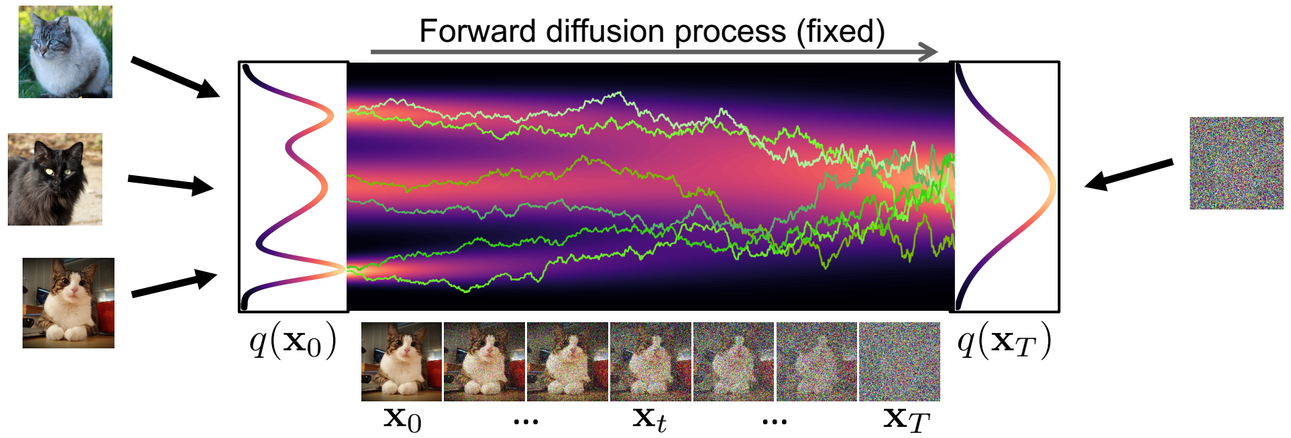
Forward Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t}_{\text{drift term (pulls towards mode)}} + \underbrace{\sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t}_{\text{diffusion term (injects noise)}}$$

Song et al., *ICLR*, 2021

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

The Generative Reverse Stochastic Differential Equation



Forward Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

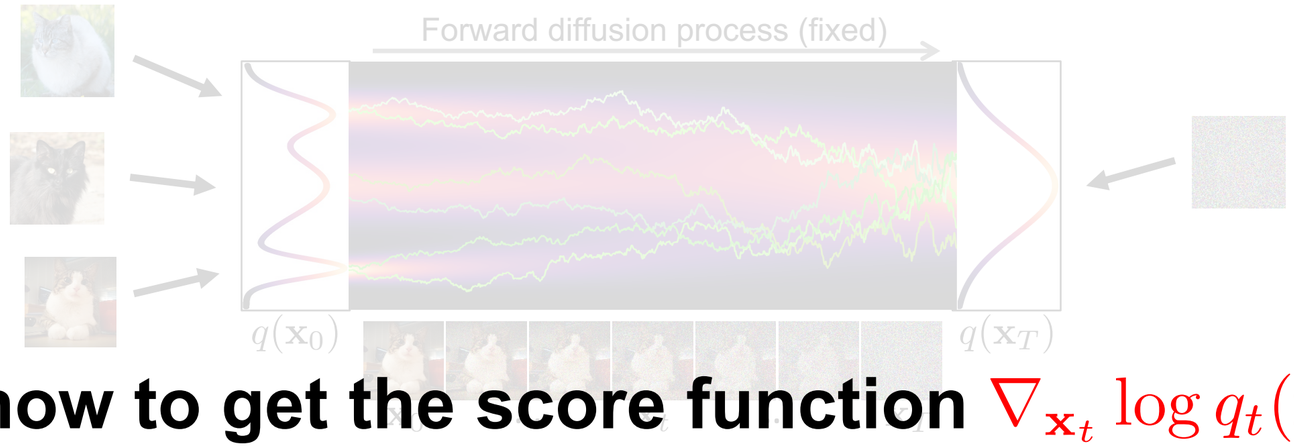
Reverse Generative Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \overbrace{\left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\underbrace{\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)}_{\text{“Score Function”}}\right]}^{\text{drift term}}\mathrm{d}t + \overbrace{\sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t}^{\text{diffusion term}}$$

➡ **Simulate reverse diffusion process: Data generation from random noise!**

Song et al., *ICLR*, 2021
Anderson, in *Stochastic Processes and their Applications*, 1982
slide from https://cvpr2022-tutorial-diffusion-models.github.io/

The Generative Reverse Stochastic Differential Equation



But how to get the score function $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$?

Forward Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

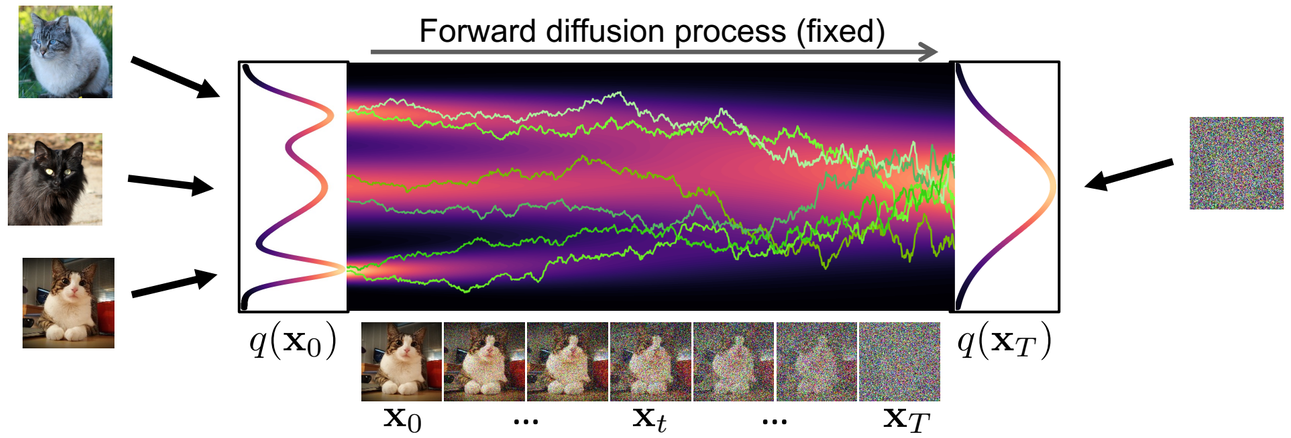
Reverse Generative Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \underbrace{\left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\underbrace{\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)}_{\text{“Score Function”}}\right]}_{\text{drift term}}\mathrm{d}t + \underbrace{\sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t}_{\text{diffusion term}}$$

➡ **Simulate reverse diffusion process: Data generation from random noise!**

Song et al., *ICLR*, 2021
Anderson, in *Stochastic Processes and their Applications*, 1982

Score Matching



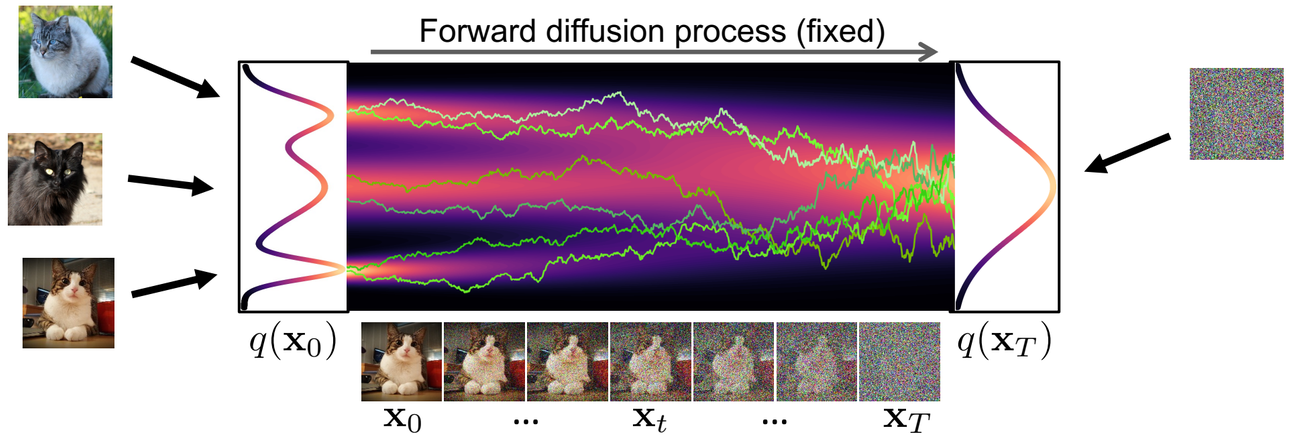
- Naïve idea, learn model for the score function by direct regression?

$$\min_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_{t\sim\mathcal{U}(0,T)}}_{\substack{\text{diffusion}\\ \text{time } t}} \underbrace{\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t)}}_{\substack{\text{diffused}\\ \text{data } \mathbf{x}_t}} \underbrace{\tilde{w}(t)}_{\substack{\text{weighting}\\ \text{function}}} \cdot \underbrace{||\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}_{\substack{\text{neural}\\ \text{network}}} - \underbrace{\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)||_2^2}_{\substack{\text{score of}\\ \text{diffused data}\\ \text{(marginal)}}}$$

➡ **But $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$ (score of the *marginal diffused density* $q_t(\mathbf{x}_t)$) is not tractable!**

Vincent, "A Connection Between Score Matching and Denoising Autoencoders", *Neural Computation*, 2011
Song and Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution", *NeurIPS*, 2019
slide from https://cvpr2022-tutorial-diffusion-models.github.io/

Denoising Score Matching



- Instead, diffuse individual data points $\mathbf{x}_0$. Diffused $q_t(\mathbf{x}_t|\mathbf{x}_0)$ ***is*** tractable!
- **Denoising Score Matching:**

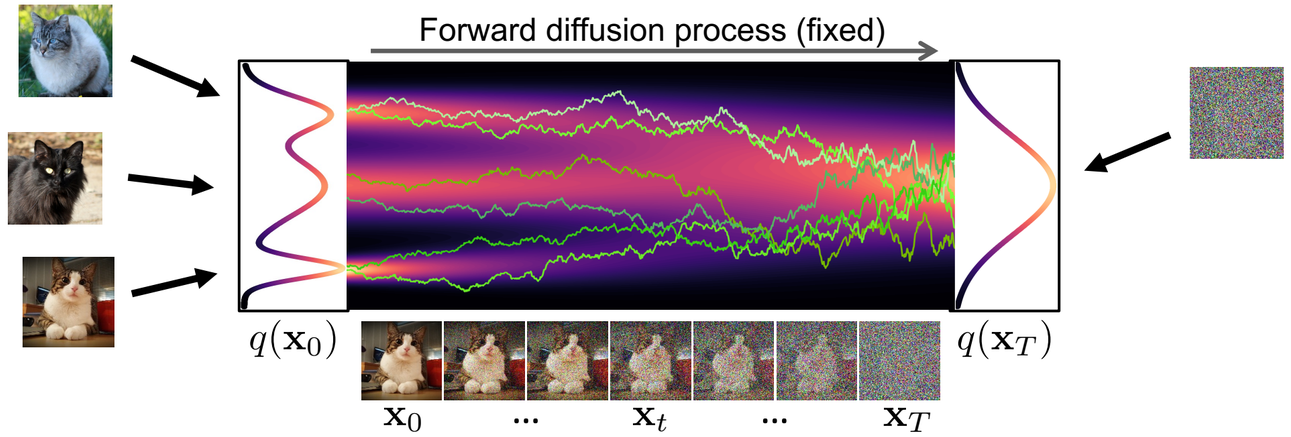
$$\min_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_{t\sim\mathcal{U}(0,T)}}_{\text{diffusion time } t} \underbrace{\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}}_{\text{data sample } \mathbf{x}_0} \underbrace{\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)}}_{\text{diffused data sample } \mathbf{x}_t} \underbrace{\tilde{w}(t)}_{\text{weighting function}} \cdot ||\underbrace{\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}_{\text{neural network}} - \underbrace{\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0)}_{\text{score of diffused data sample}}||_2^2$$

➡ **After expectations,** $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$**!**

Vincent, in *Neural Computation*, 2011
Song and Ermon, *NeurIPS*, 2019
Song et al. *ICLR*, 2021
slide from https://cvpr2022-tutorial-diffusion-models.github.io/

Denoising Score Matching

Epsilon-prediction parametrization



- **Denoising Score Matching:**

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)}\tilde{w}(t)\cdot||\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t|\mathbf{x}_0)||_2^2$$

  - Re-parametrized sampling: $\mathbf{x}_t=\alpha_t\mathbf{x}_0+\sigma_t\boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$
    - Score function: $\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t|\mathbf{x}_0)=-\nabla_{\mathbf{x}_t}\frac{(\mathbf{x}_t-\alpha_t\mathbf{x}_0)^2}{2\sigma_t^2}=-\frac{\mathbf{x}_t-\alpha_t\mathbf{x}_0}{\sigma_t^2}=-\frac{\alpha_t\mathbf{x}_0+\sigma_t\boldsymbol{\epsilon}-\alpha_t\mathbf{x}_0}{\sigma_t^2}=-\frac{\boldsymbol{\epsilon}}{\sigma_t}$
    - Neural network model: $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t,t):=-\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\sigma_t}$

Vincent, in *Neural Computation*, 2011
Song and Ermon, *NeurIPS*, 2019
Song et al. *ICLR*, 2021

➡ $$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\hat{w}(t)\cdot||\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)||_2^2 \qquad \hat{w}(t)=\frac{\tilde{w}(t)}{\sigma_t}$$

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

What is the ELBO for continuous-time diffusion models?

- Denoising Score Matching:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\hat{w}(t)\cdot||\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)||_2^2$$

- [Kingma et al, 2021] showed that the variational upper bound (negative ELBO) can be reduced to a simple variant:

$$-\text{ELBO}(\theta) = \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}-\frac{d\lambda}{dt}\cdot||\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)||_2^2+\text{const}$$

$$\text{where } \mathbf{x}_t = \alpha_t\mathbf{x}_0+\sigma_t\boldsymbol{\epsilon}$$

$$\lambda = \log\frac{\alpha_t^2}{\sigma_t^2} \quad \text{(signal-to-noise ratio of timestep } t\text{)}$$

- In practice, different loss weightings trade off between model with good perceptual quality vs. high log-likelihood:
  - Perceptual quality, e.g., $\hat{w}(t)=1$
  - High likelihood: -ELBO

➡ How to explain such discrepancy?
Can we still trust ELBO / maximum likelihood?

Kingma et al., “Variational diffusion models”, NeurIPS 2021.

Weighted diffusion objectives

Understanding diffusion objectives as the ELBO with data augmentation

- Summarize different types of diffusion objectives as the following **weighted diffusion objective**:

$$\mathcal{L}_w(\boldsymbol{\theta}) = \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[w(t)\cdot -\frac{d\lambda}{dt}\cdot ||\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)||_2^2\right]$$

- [Kingma & Gao, 2023] showed that if $w(t)$ is a monotonically increasing function, then the weighted diffusion objective is equivalent to the **negative ELBO with data augmentation** (Gaussian addition noise):

$$\mathcal{L}_w(\boldsymbol{\theta}) \geq \underbrace{\mathbb{E}_{t\sim p_w(t)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log p(\mathbf{x}_t)\right]}_{\text{Neg. log lik. of noise-perturbed data}} + \text{const}$$

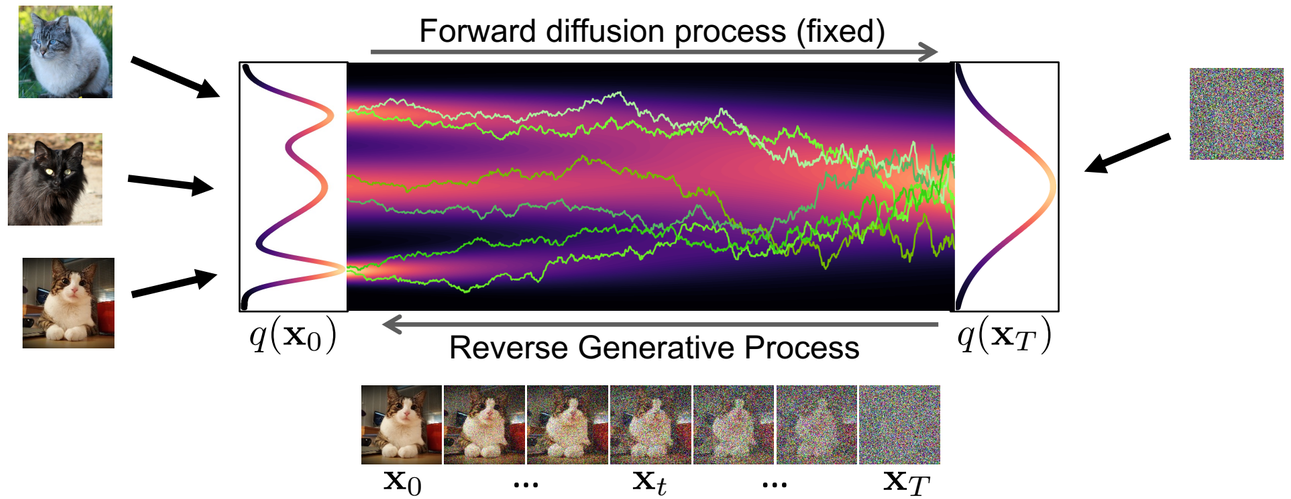
$$\text{where } p_w(t) \propto \frac{dw(t)}{dt}$$

- Therefore, the ELBO objective is compatible with perceptual quality when combined with simple data augmentation (additive noise)

Kingma & Gao, “Understanding Diffusion Objectives\\as the ELBO with Data Augmentation”, 2023.

Probability Flow ODE

Alternative reverse process



- Consider reverse generative diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$$

- In distribution equivalent to **"Probability Flow ODE"**:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)\right]\mathrm{d}t$$

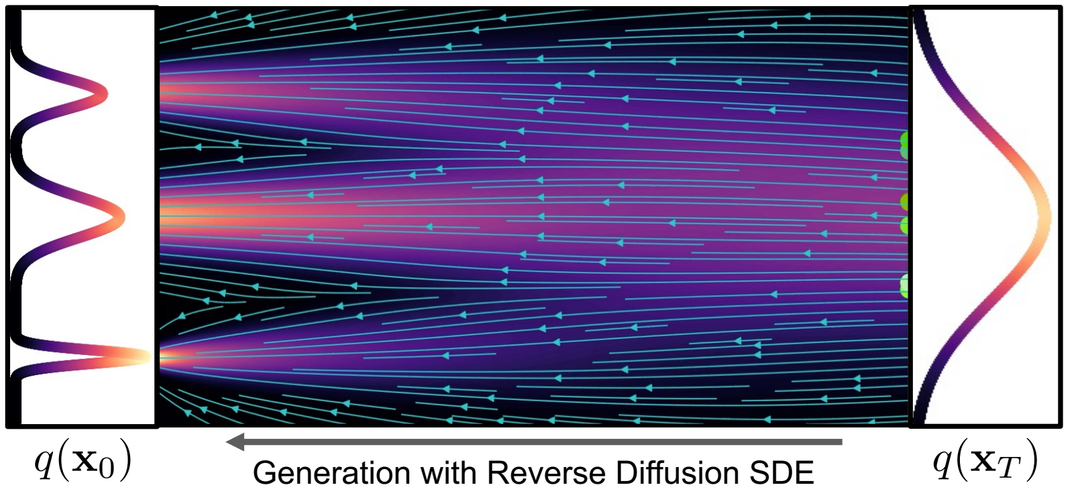
(solving this ODE results in the same $q_t(\mathbf{x}_t)$ when initializing $q_T(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$)

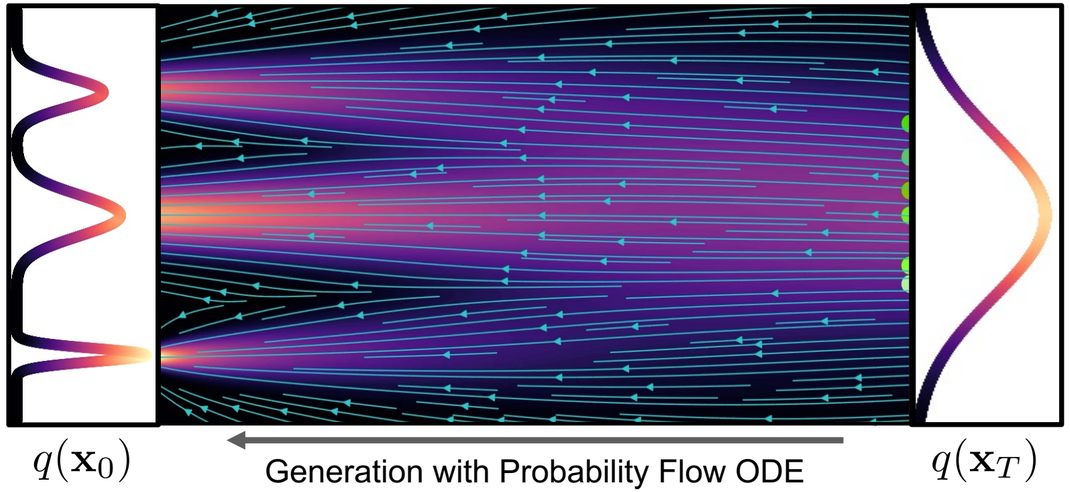
Deterministic mapping from $\mathbf{x}_T$ to $\mathbf{x}_0$

Song et al., *ICLR*, 2021

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

Synthesis with SDE vs. ODE





- **Generative Reverse Diffusion SDE (stochastic):**

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$$

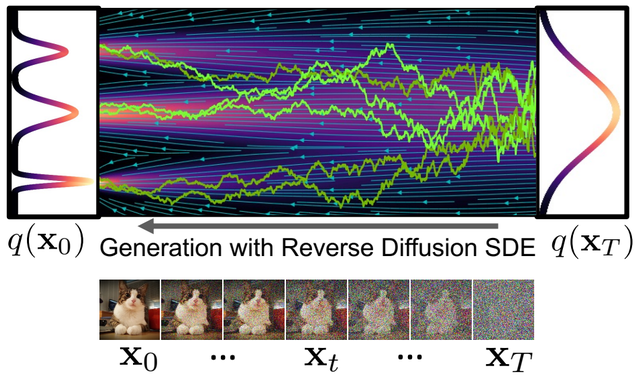
- **Generative Probability Flow ODE (deterministic):**

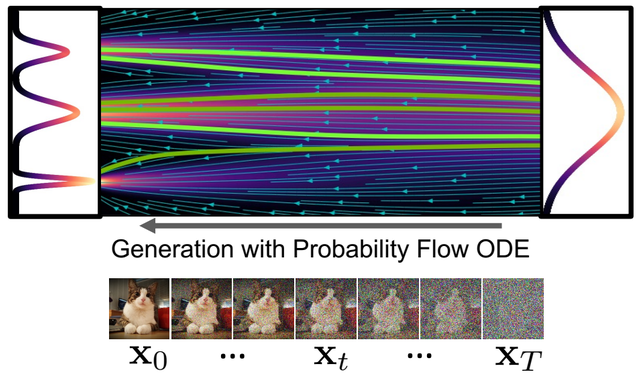
$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t$$

Song et al., *ICLR*, 2021
slide from https://cvpr2022-tutorial-diffusion-models.github.io/

Sampling from “Continuous-Time” Diffusion Models

SDE vs. ODE Sampling: Pro’s and Con’s





Generative Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$$

$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t}_{\text{Probability Flow ODE}} \underbrace{-\frac{1}{2}\beta(t)\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t}_{\text{Langevin dynamics}}$$

Probability Flow ODE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t$$

- *Pro:* Continuous noise injection can help to compensate errors during diffusion process (Langevin sampling actively pushes towards correct distribution).
- *Con:* Often slower, because the stochastic terms themselves require fine discretization during solve.

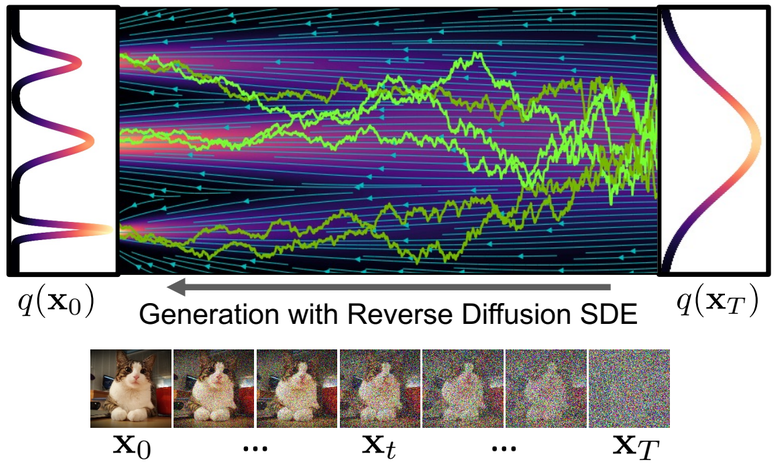
- *Pro:* Can leverage fast ODE solvers. Best when targeting very fast sampling.
- *Con:* No “stochastic” error correction, often slightly lower performance than stochastic sampling.

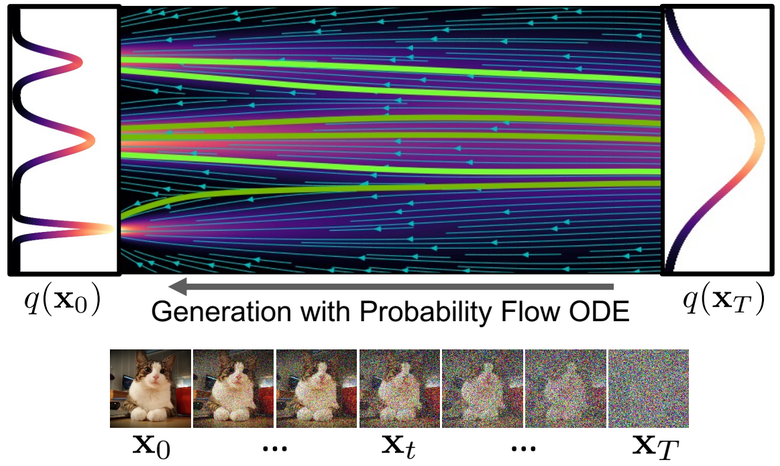
Karras et al., “Elucidating the Design Space of Diffusion-Based Generative Models”, *arXiv*, 2022

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

Sampling from “Continuous-Time” Diffusion Models

How to solve the generative SDE or ODE in practice?





Generative Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$$

- *Euler-Maruyama*:

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\Delta t + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

- *Ancestral Sampler* (discrete-time) is also a generative SDE sampler!

Probability Flow ODE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t$$

- *Euler’s Method*:

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \frac{1}{2}\beta(t)\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\Delta t$$

- In practice: DDIM sampler, another solver of the ODE.

Song et al., *ICLR*, 2021

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

How to make sampling faster?

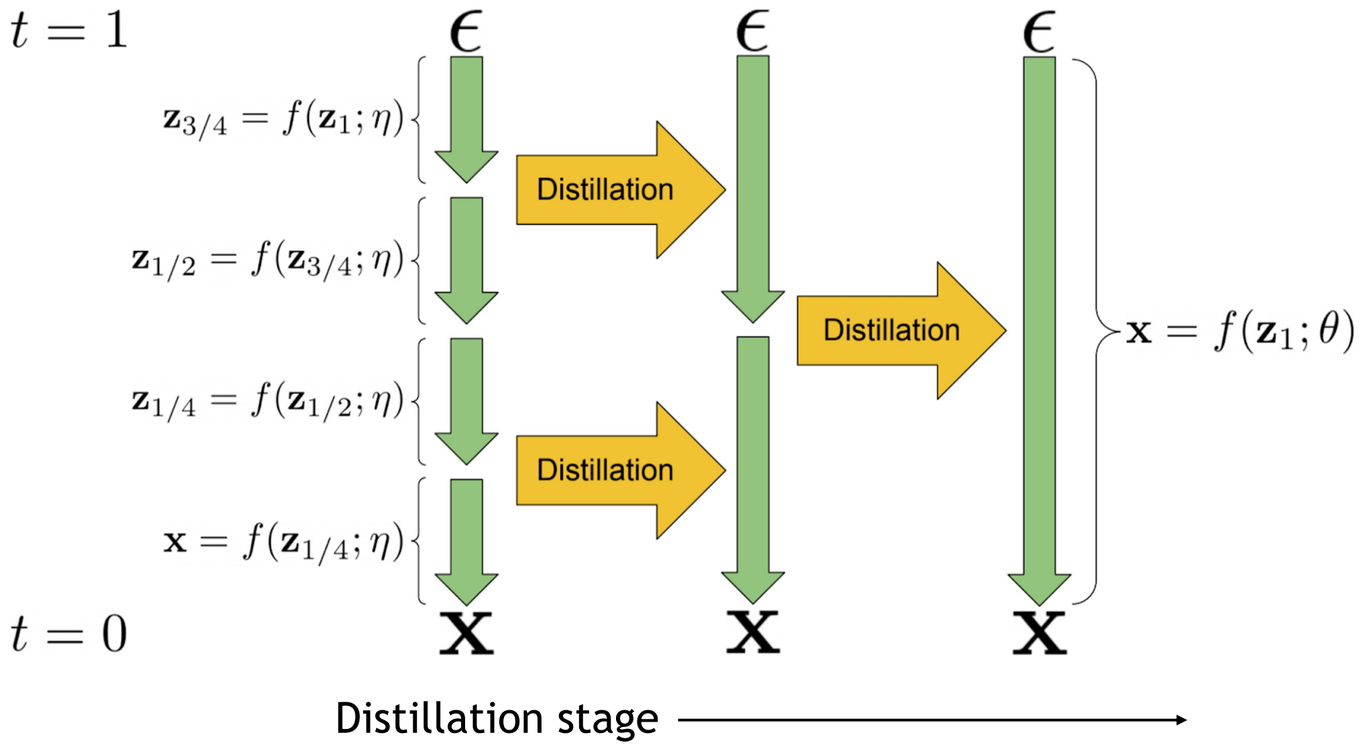
- One bottleneck of diffusion models is its slowness in sampling: need 10-1000+ steps to generate high quality samples
- Generative models need to be fast for practical use.

- One solution: distill diffusion models into models using just 4-8 sampling steps!
  - *Progressive distillation for fast sampling of diffusion models, Salimans & Ho, ICLR 2022*
  - *On Distillation of Guided Diffusion Models, Meng et al., CVPR 2023*

Progressive distillation

How to make sampling faster?

- Distill a deterministic ODE sampler (i.e. DDIM sampler) to the same model architecture.
- At each stage, a “student” model is learned to distill two adjacent sampling steps of the “teacher” model to one sampling step.
- At next stage, the “student” model from previous stage will serve as the new “teacher” model.



Salimans & Ho, “Progressive distillation for fast sampling of diffusion models”, ICLR 2022.

Algorithm 1 Standard diffusion training

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

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

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

Algorithm 2 Progressive distillation

Require: Trained teacher model $\hat{\mathbf{x}}_\eta(\mathbf{z}_t)$
Require: Data set $\mathcal{D}$
Require: Loss weight function $w()$
Require: Student sampling steps N
for K iterations **do**
  $\theta \leftarrow \eta$ ▷ Init student from teacher
  while not converged **do**
    $\mathbf{x} \sim \mathcal{D}$
    $t = i/N, \;\; i \sim Cat[1, 2, \ldots, N]$
    $\epsilon \sim N(0, I)$
    $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$
    `# 2 steps of DDIM with teacher`
    $t' = t - 0.5/N, \;\; t'' = t - 1/N$
    $\mathbf{z}_{t'} = \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_t) + \frac{\sigma_{t'}}{\sigma_t}(\mathbf{z}_t - \alpha_t \hat{\mathbf{x}}_\eta(\mathbf{z}_t))$
    $\mathbf{z}_{t''} = \alpha_{t''} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}) + \frac{\sigma_{t''}}{\sigma_{t'}}(\mathbf{z}_{t'} - \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}))$
    $\tilde{\mathbf{x}} = \frac{\mathbf{z}_{t''} - (\sigma_{t''}/\sigma_t)\mathbf{z}_t}{\alpha_{t''} - (\sigma_{t''}/\sigma_t)\alpha_t}$ ▷ Teacher $\hat{\mathbf{x}}$ target
    $\lambda_t = \log[\alpha_t^2 / \sigma_t^2]$
    $L_\theta = w(\lambda_t) \|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_\theta(\mathbf{z}_t)\|_2^2$
    $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$
  end while
  $\eta \leftarrow \theta$ ▷ Student becomes next teacher
  $N \leftarrow N/2$ ▷ Halve number of sampling steps
end for

Salimans & Ho, “Progressive distillation for fast sampling of diffusion models”, ICLR 2022.

On Distillation of Guided Diffusion Models

Meng et al., CVPR 2023 award nominated

Text-guided generation (1 step)

Text-guided generation (2 steps)

Class-conditional generation (1 step)

Now also works with

- CF-Guidance
- Stochastic sampling
- Text-to-image/video
- Image-to-image
- Inpainting
- Latent Diffusion

Text-guided generation (4 steps)

Input Mask Result 1 Result 2

Image inpainting (2 steps)

Input Output (different styles)

Image to image translation (3 steps)

Case study: Imagen

Imagen: text-to-image diffusion models

By Google (imagen.research.google)

Input: text; Output: 1kx1k images

- An unprecedented degree of photorealism
  - SOTA automatic scores & human ratings
- A deep level of language understanding
- Extremely simple
  - no latent space, no quantization

A brain riding a rocketship heading towards the moon.

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

Imagen

By Google (imagen.research.google)

A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

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

Imagen

By Google (imagen.research.google)

A dragon fruit wearing karate belt in the snow.

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

“toilet paper with real cactus spikes”
by Irina Blok

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

Imagen

What goes to Imagen?

Data

- Image-text pairs
- LAION-400M
- Internal (~500M images)

Model

- Diffusion models
- Cascading super-res
- Frozen text encoders

Sampler

- Classifier-free guidance
- Maximizing text-alignment

Scaling up

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

What goes to Imagen?

Data

- Image-text pairs
- LAION-400M
- Internal (~500M images)

Model

- Diffusion models
- Cascading super-res
- Frozen text encoders

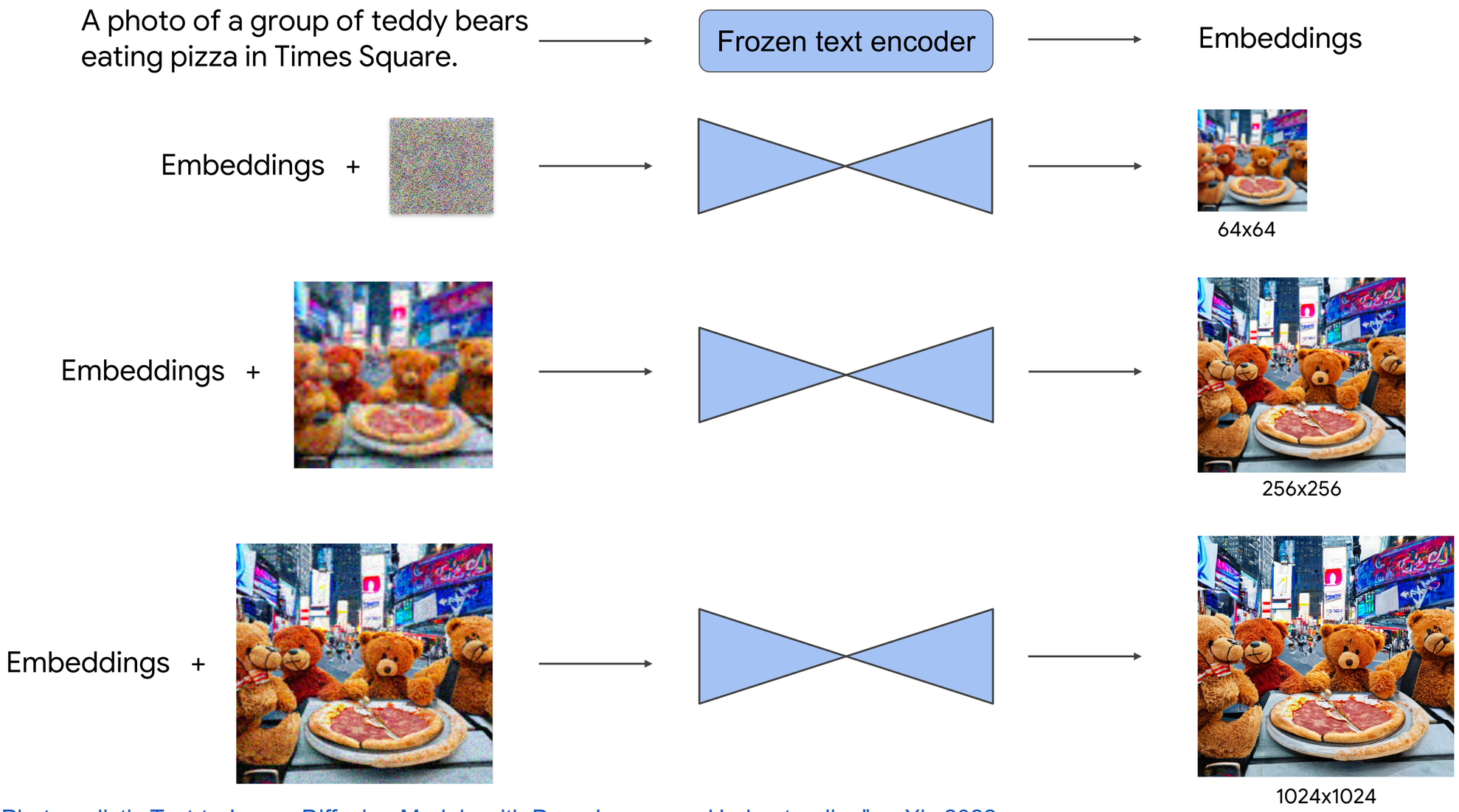
Sampler

- Classifier-free guidance
- Maximizing text-alignment

Scaling up

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

Imagen: Cascaded generation pipeline



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

Classifier Guidance

Sampler technique

- Assume pairs of data (x, c). A classifier guidance diffusion model consists of
  - A trained conditional diffusion model
  - A trained classifier model on noisy data $\mathbf{x}_t$
- During sampling, at each denoising step, modify the score function to

$$\nabla_{\mathbf{x}_t} \log \tilde{p}_{\theta,\phi}(\mathbf{x}_t|\mathbf{c}) = \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{c}) + \omega \nabla_{\mathbf{x}_t} \log p_\phi(\mathbf{c}|\mathbf{x}_t).$$

From the conditional diffusion model

From the classifier model

- Upweight samples that the classifier assigns high probability with, better alignment with c.
- Cons: need to train an additional classifier. Increase model complexity.

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

Classifier-free Guidance

Sampler technique

- Assume there’re two diffusion models, one conditional model and one unconditional model.
- By Bayes’ rule we can define an implicit classifier

$$p_\theta(\mathbf{c}|\mathbf{x}_t) \propto p_\theta(\mathbf{x}_t|\mathbf{c})/p_\theta(\mathbf{x}_t).$$

- The modified score function during sampling then becomes

$$\nabla_{\mathbf{x}_t} \log \tilde{p}_\theta(\mathbf{x}_t|\mathbf{c}) = (1+\omega)\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{c}) - \omega\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t).$$

From the conditional diffusion model

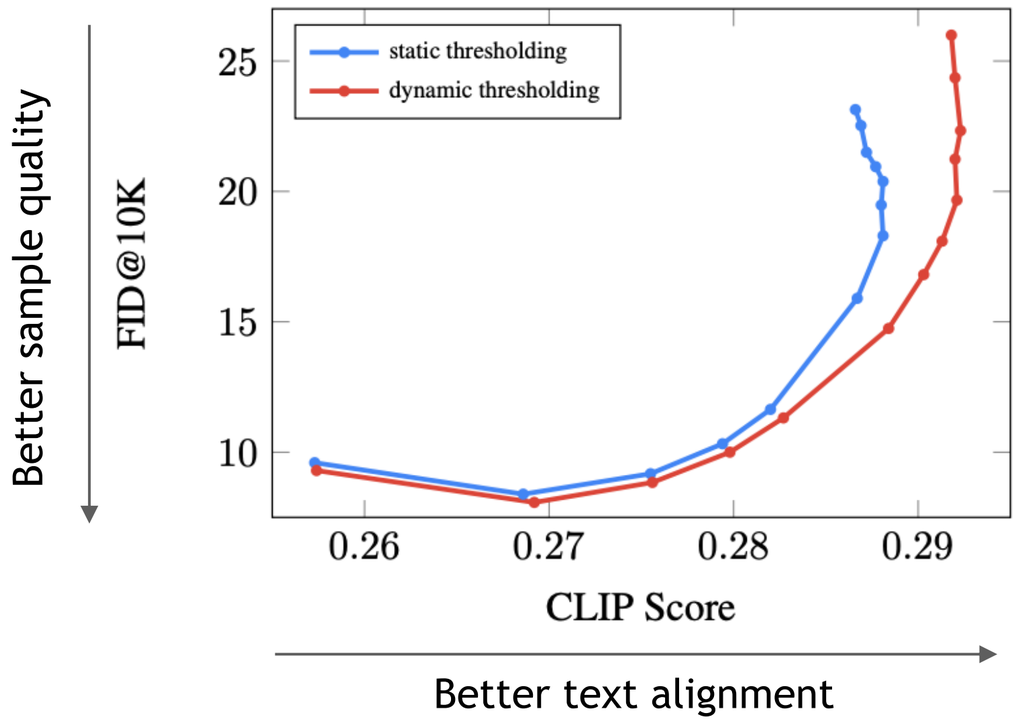
From the unconditional diffusion model

- The two models can share weights, with the unconditional model taking a null class label c.

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

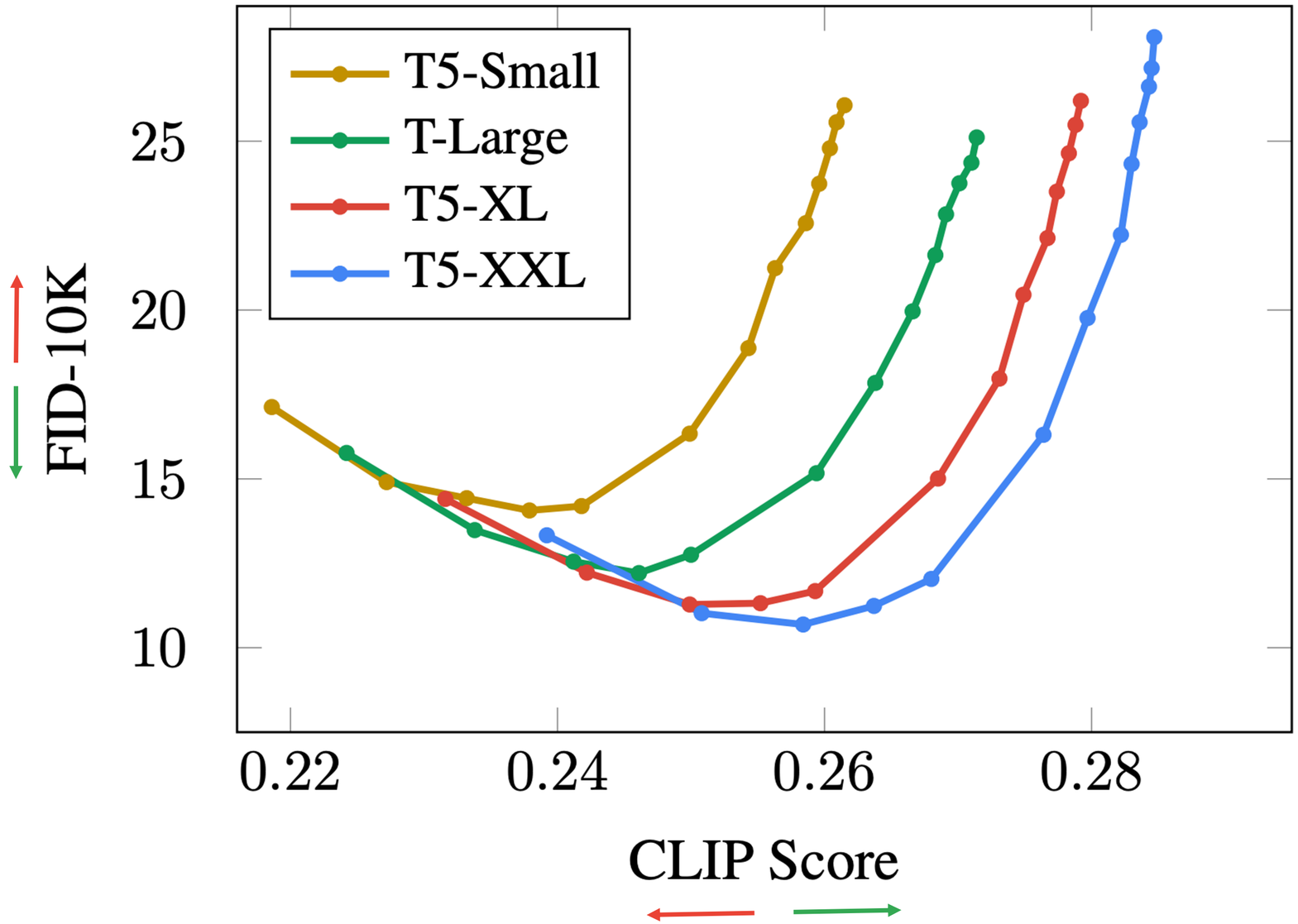
Classifier-free guidance in Imagen

- Large classifier-free guidance weights → better text alignment, worse image fidelity



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

Larger Text Encoders → Better Alignment, Better Fidelity



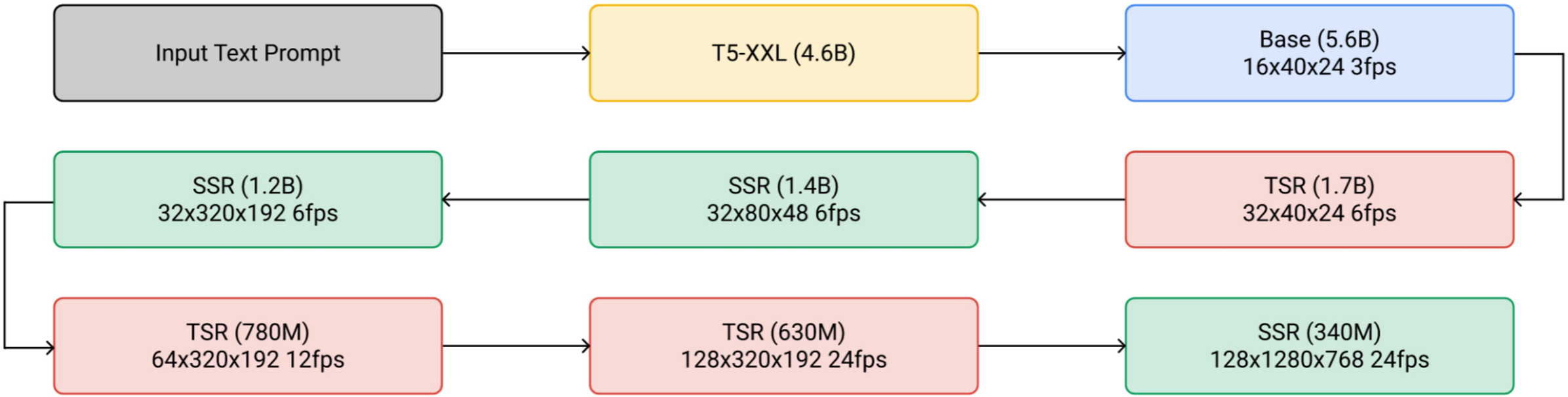
Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

Can we generalize this to video?

Imagen Video (imagen.research.google/video)

Jonathan et al., “Imagen Video: High Definition Video Generation with Diffusion Models”, 2023.

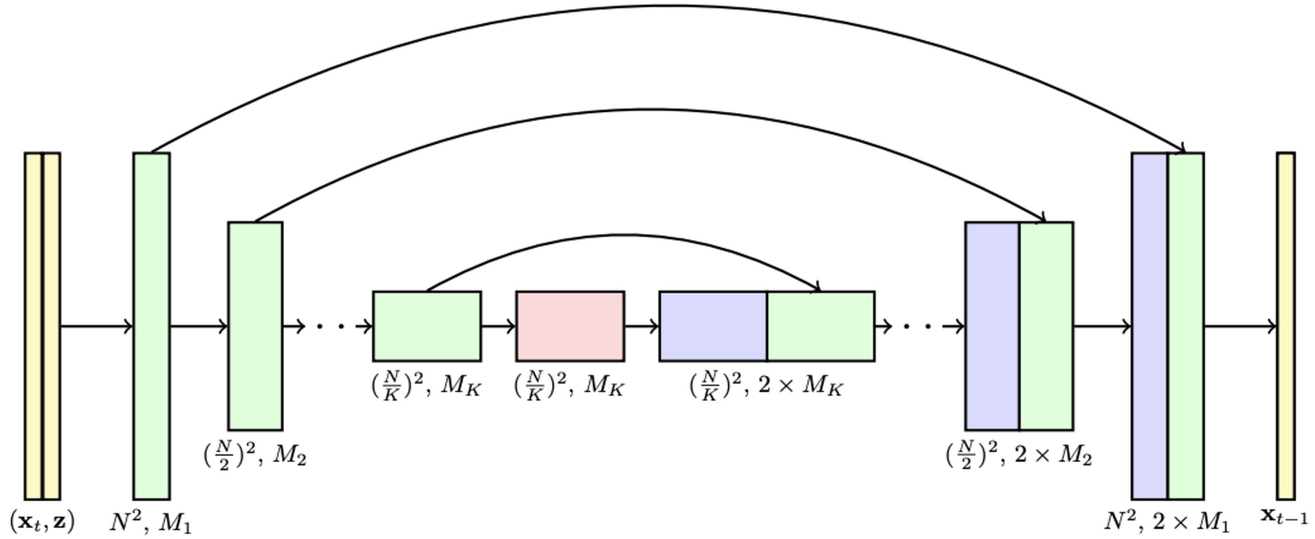
Imagen Video generalizes Imagen to the video domain using a cascade of super-resolution diffusion models in space and time



Video diffusion models

Ho & Salimans, et al. https://video-diffusion.github.io/

- Image ⟶ Video: Just add another dimension to the data tensor
- Image architecture: 2D UNet
- Video architecture: 3D UNet, space-time separable
  - repeat the 2D UNet over frames
  - additional layers to mix over time using attention or convolution



Wooden figurine surfing on a surfboard in space.

Imagen Video

Flying through an intense battle between pirate ships in a stormy ocean.

“watercolor greeting card of thank you so much!”