Denoising Diffusion-based Generative Modeling: Foundations and Applications

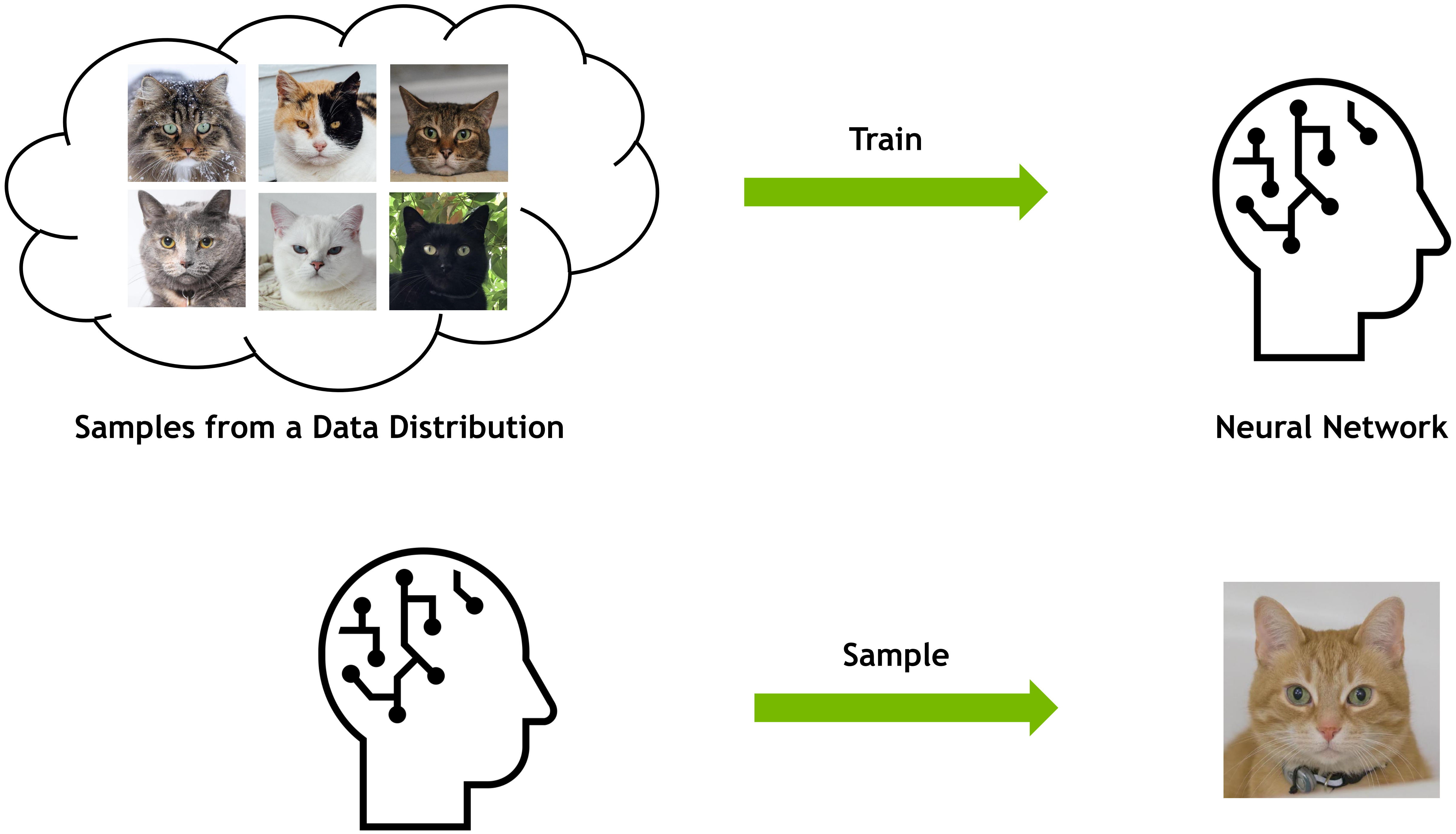
Karsten Kreis

Ruiqi Gao

Arash Vahdat

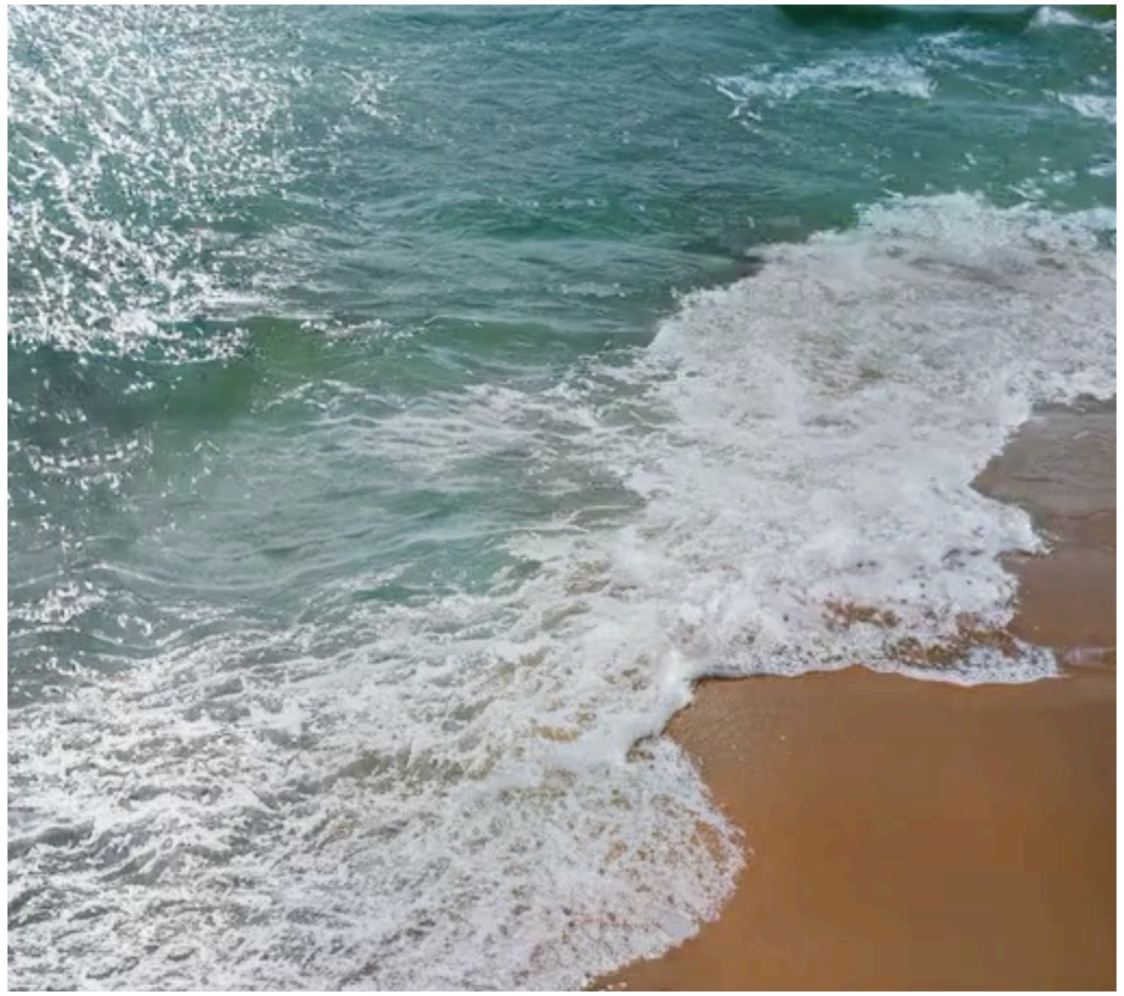
Deep Generative Learning

Learning to generate data



Application (1): Content Generation

StyleGAN3 example images



Karras et al. Alias-Free Generative Adversarial Networks, NeurIPS 2021

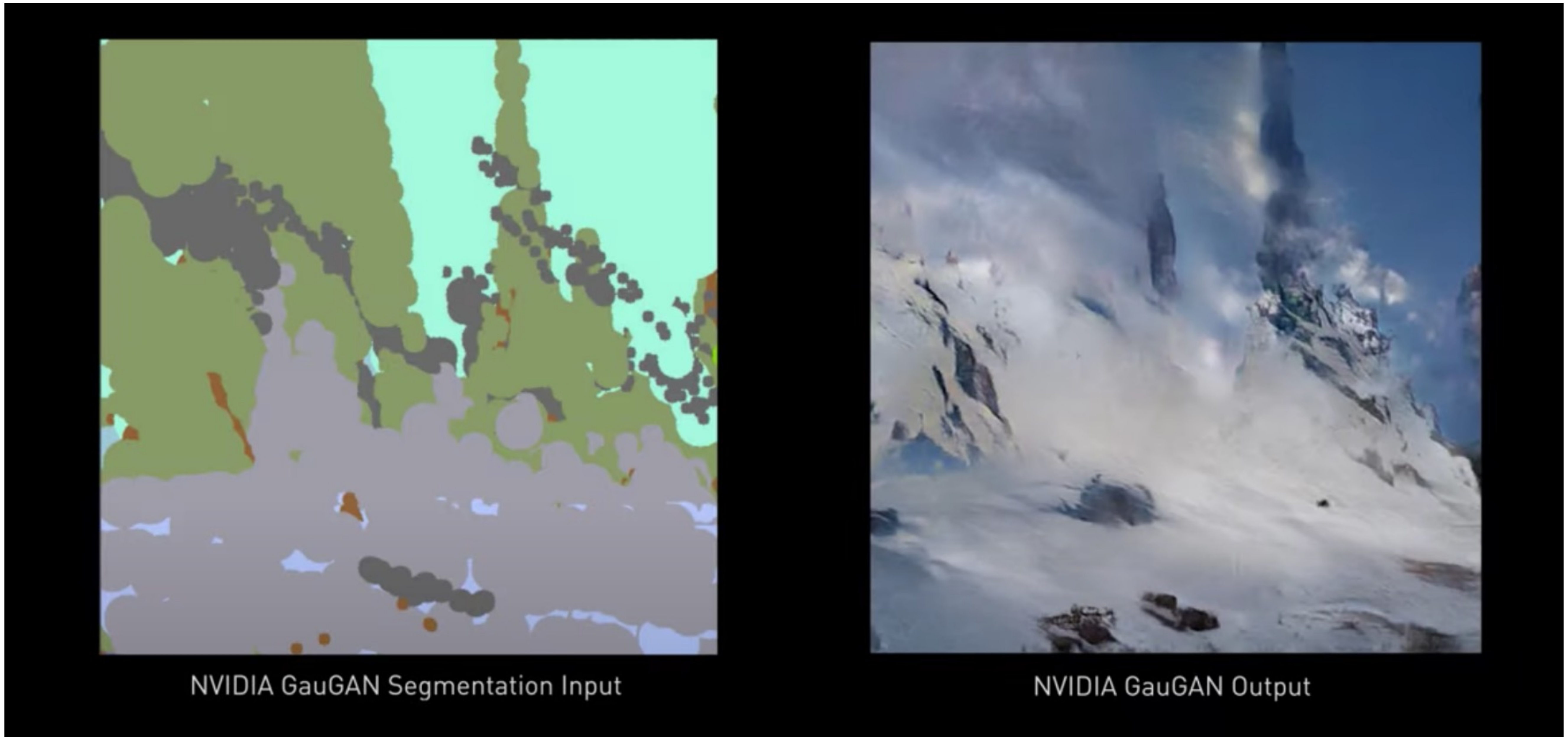
Application (2): Representation Learning

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]

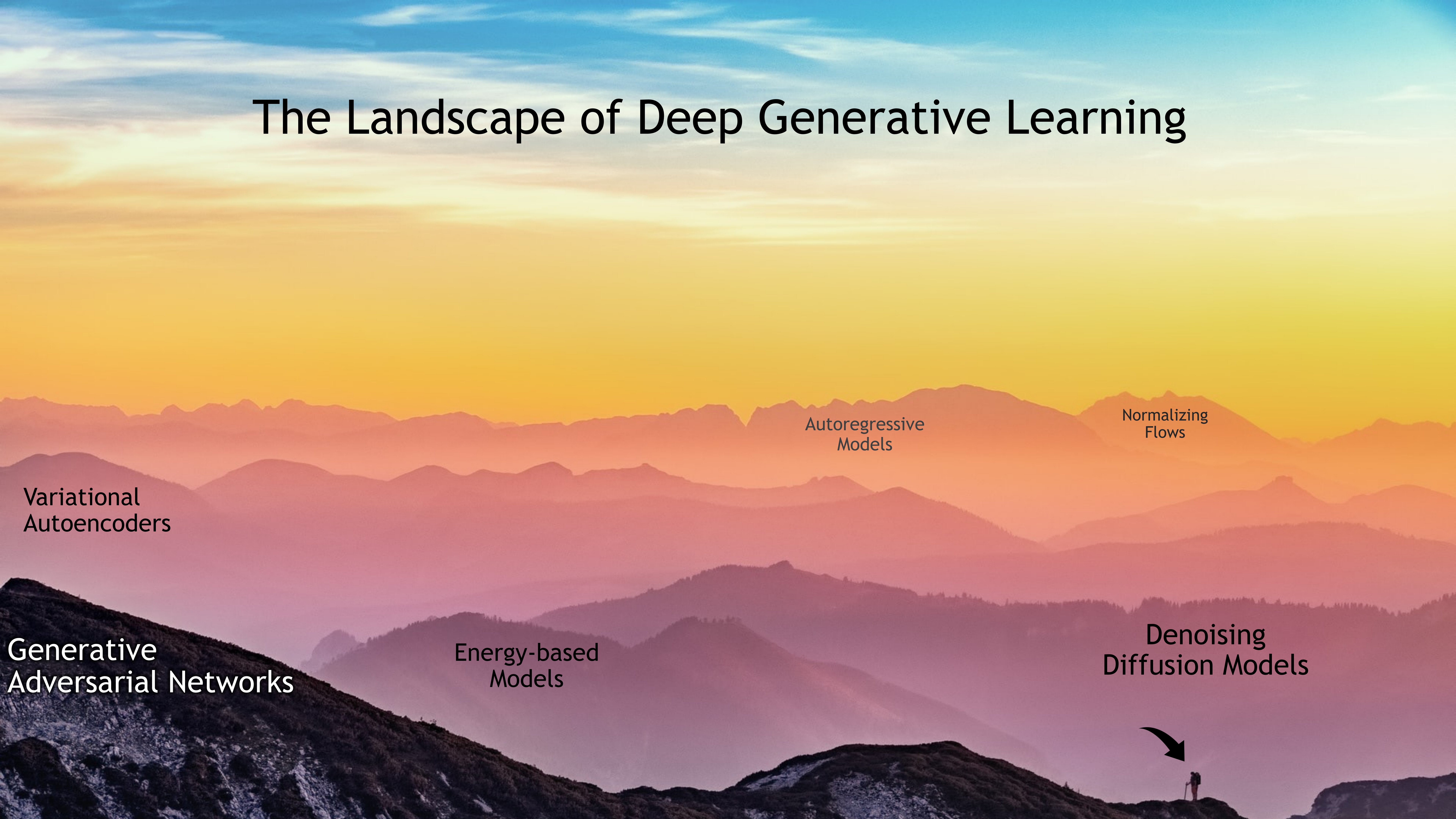
Application (3): Artistic Tools

NVIDIA GauGAN

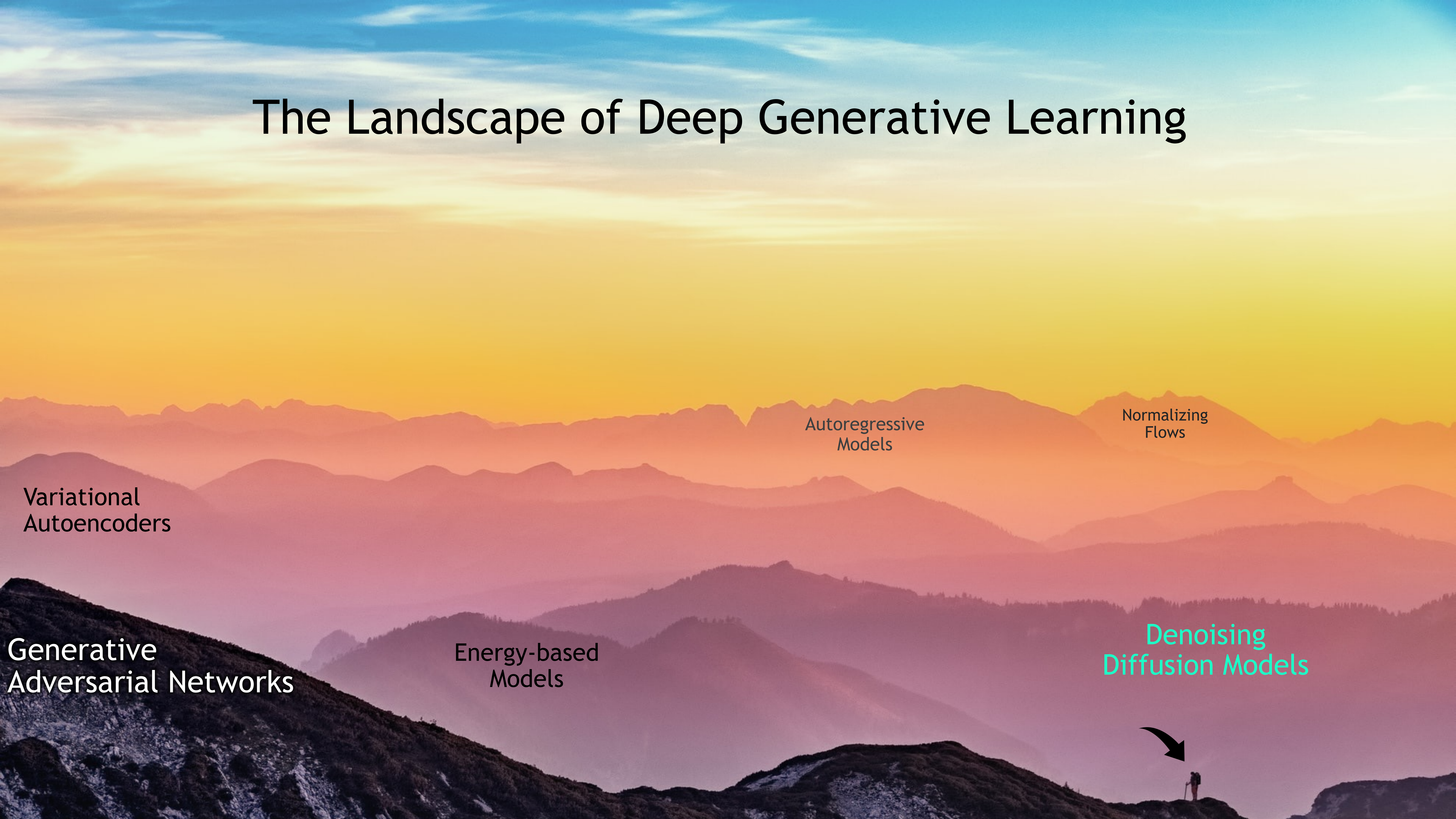


[Park et al., Semantic Image Synthesis with Spatially-Adaptive Normalization, CVPR 2019](#)

The Landscape of Deep Generative Learning



The Landscape of Deep Generative Learning



Denoising Diffusion Models

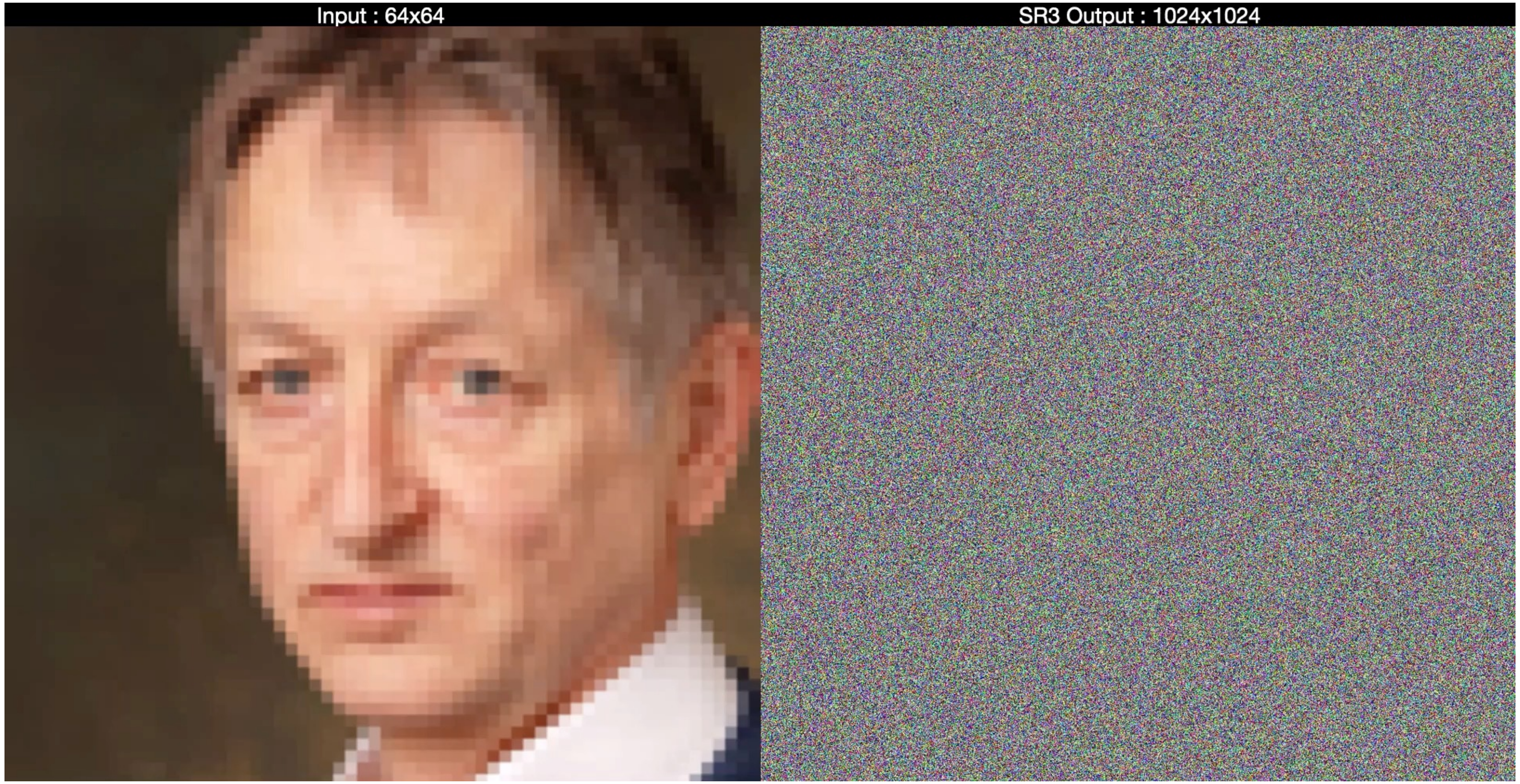
Emerging as powerful generative models, outperforming GANs

[“Diffusion Models Beat GANs on Image Synthesis”
Dhariwal & Nichol, OpenAI, 2021]

[“Cascaded Diffusion Models for High Fidelity Image Generation”
Ho et al., Google, 2021]

Image Super-resolution

Successful applications



[Saharia et al., Image Super-Resolution via Iterative Refinement, ICCV 2021](#)

Text-to-Image Generation

DALL·E 2

“a teddy bear on a skateboard in times square”

[“Hierarchical Text-Conditional Image Generation with CLIP Latents” Ramesh et al., 2022]

Imagen

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

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

Today's Program

Title	Speaker	Time
Introduction	Arash	10 min
Part (1): Denoising Diffusion Probabilistic Models	Arash	35 min
Part (2): Score-based Generative Modeling with Differential Equations	Karsten	45 min
Part (3): Advanced Techniques: Accelerated Sampling, Conditional Generation, and Beyond	Ruiqi	45 min
Applications (1): Image Synthesis, Text-to-Image, Controllable Generation	Ruiqi	15 min
Applications (2): Image Editing, Image-to-Image, Super-resolution, Segmentation	Arash	15 min
Applications (3): Video Synthesis, Medical Imaging, 3D Generation, Discrete State Models	Karsten	15 min
Conclusions, Open Problems and Final Remarks	Arash	10 min

cvpr2022-tutorial-diffusion-models.github.io

Today's Program

Title	Speaker	Time
Introduction	Arash	10 min
Part (1): Denoising Diffusion Probabilistic Models	Arash	35 min
Part (2): Score-based Generative Modeling with Differential Equations	Karsten	45 min
Part (3): Advanced Techniques: Accelerated Sampling, Conditional Generation, and Beyond	Ruiqi	45 min
Applications (1): Image Synthesis, Text-to-Image, Controllable Generation	Ruiqi	15 min
Applications (2): Image Editing, Image-to-Image, Super-resolution, Segmentation	Arash	15 min
Applications (3): Video Synthesis, Medical Imaging, 3D Generation, Discrete State Models	Karsten	15 min
Conclusions, Open Problems and Final Remarks	Arash	10 min

cvpr2022-tutorial-diffusion-models.github.io

Disclaimer


You didn't include my arXiv submission that will come out next week?
Imagen

Today's Program

Title	Speaker	Time
Introduction	Arash	10 min
Part (1): Denoising Diffusion Probabilistic Models	Arash	35 min
Part (2): Score-based Generative Modeling with Differential Equations	Karsten	45 min
Part (3): Advanced Techniques: Accelerated Sampling, Conditional Generation, and Beyond	Ruiqi	45 min
Applications (1): Image Synthesis, Text-to-Image, Controllable Generation	Ruiqi	15 min
Applications (2): Image Editing, Image-to-Image, Super-resolution, Segmentation	Arash	15 min
Applications (3): Video Synthesis, Medical Imaging, 3D Generation, Discrete State Models	Karsten	15 min
Conclusions, Open Problems and Final Remarks	Arash	10 min

cvpr2022-tutorial-diffusion-models.github.io

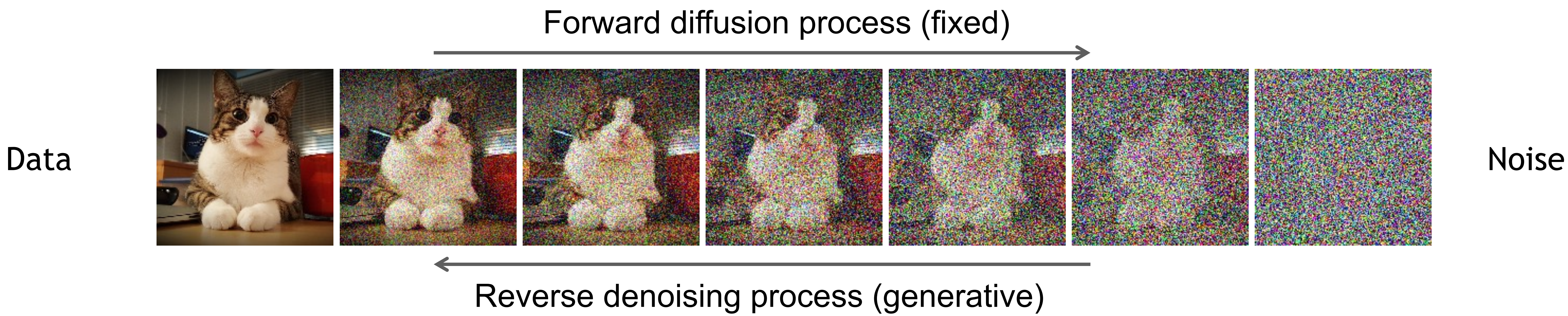
Part (1):
Denoising Diffusion Probabilistic Models

Denoising Diffusion Models

Learning to generate by denoising

Denoising diffusion models consist of two processes:

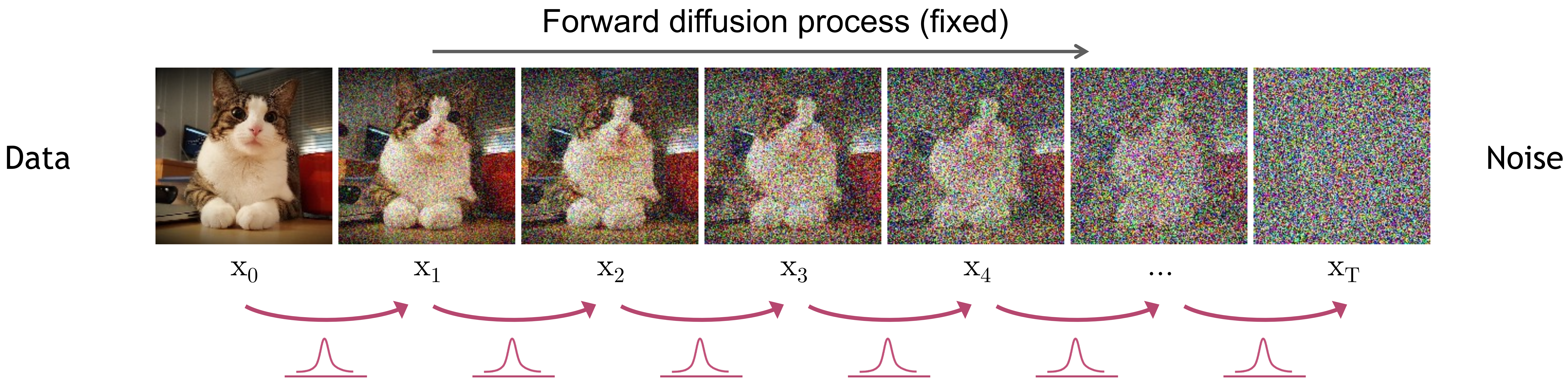
- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



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

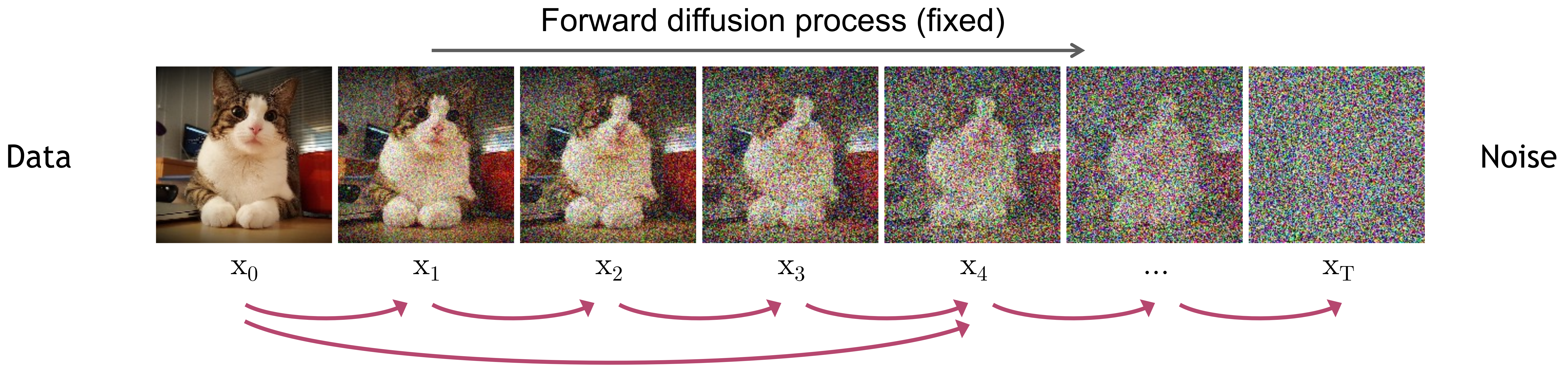
Forward Diffusion Process

The formal definition of the forward process in T steps:



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

Diffusion Kernel



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

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

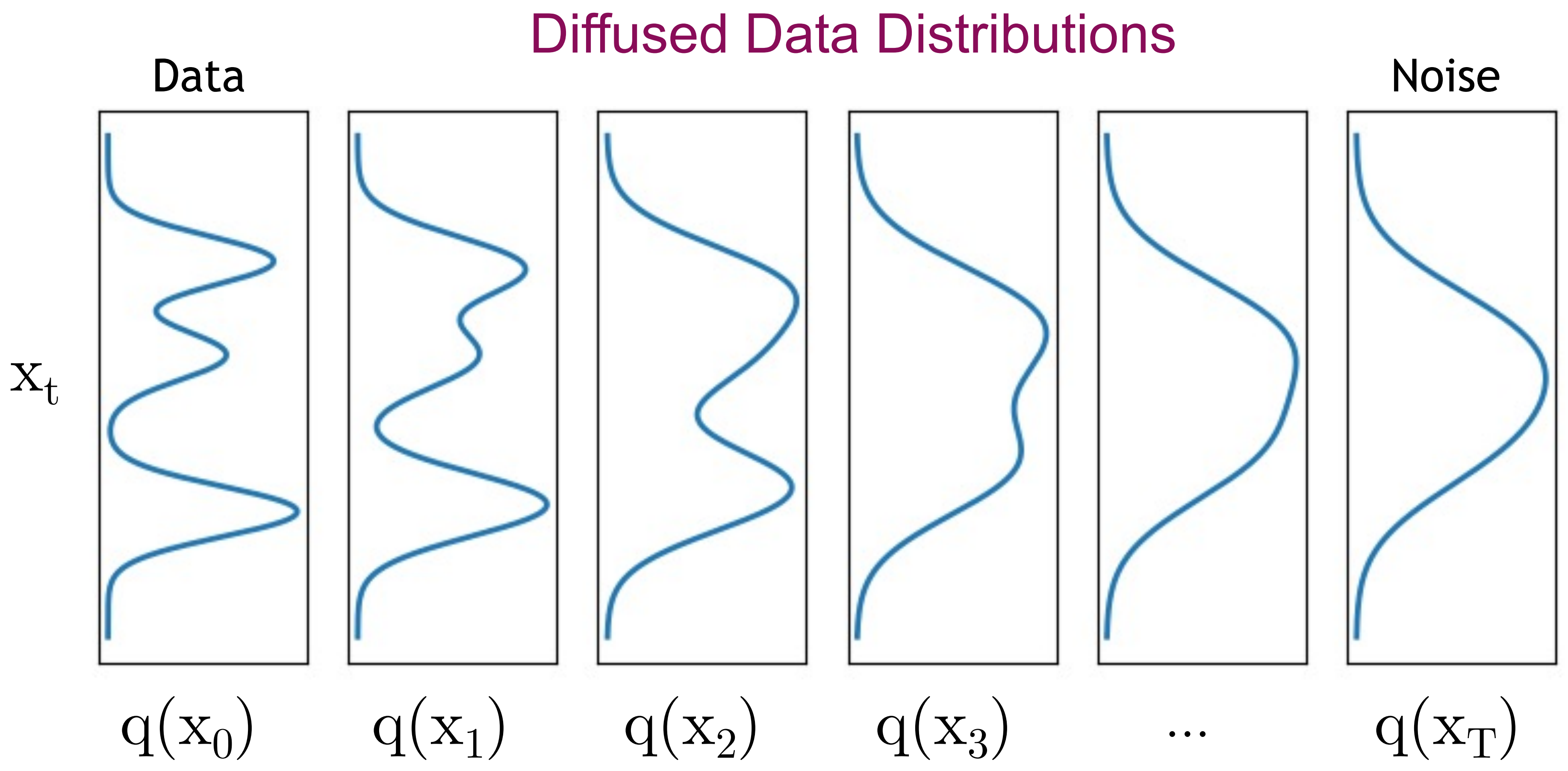
β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

What happens to a distribution in the forward diffusion?

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

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

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).

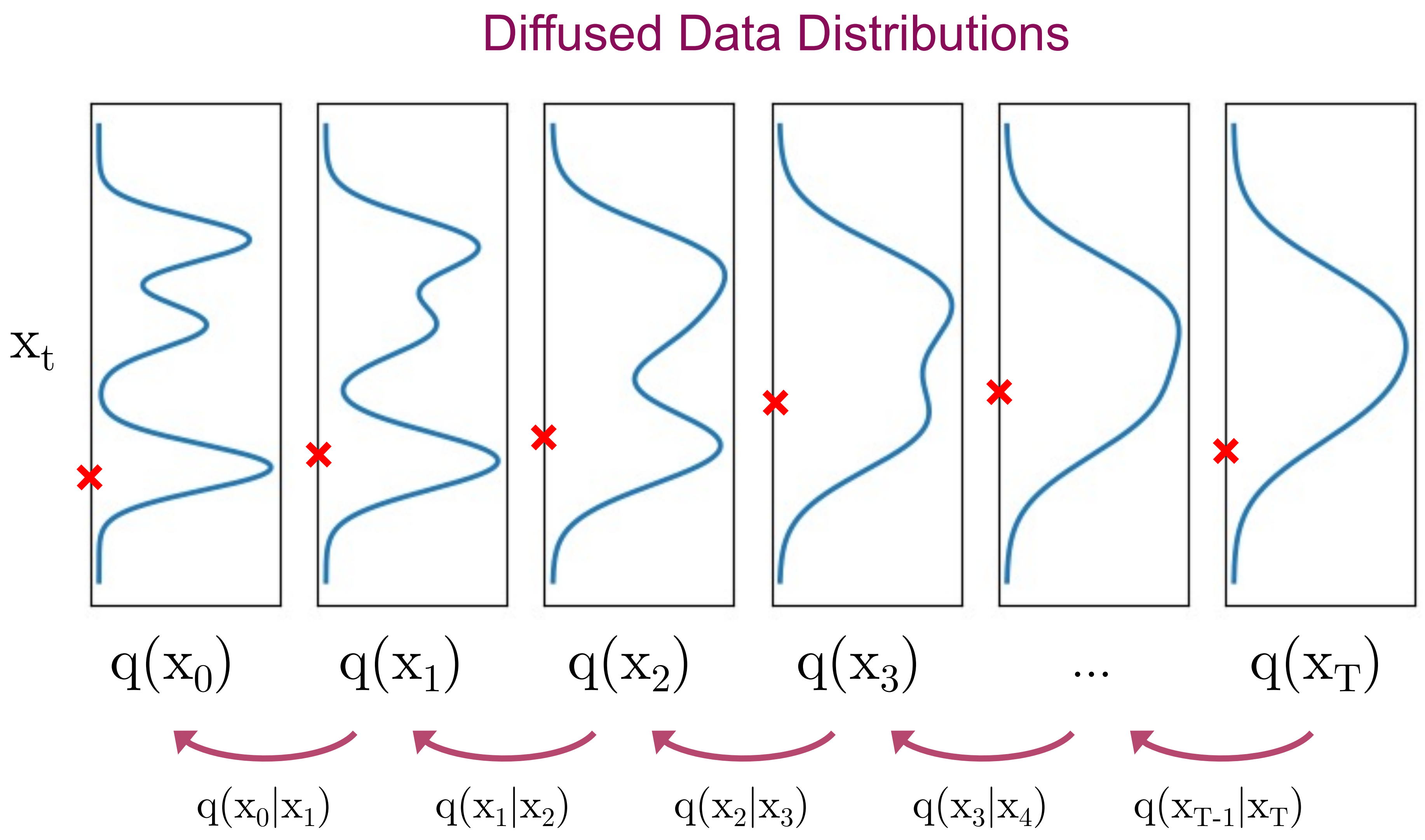
Generative Learning by Denoising

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

Generation:

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

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

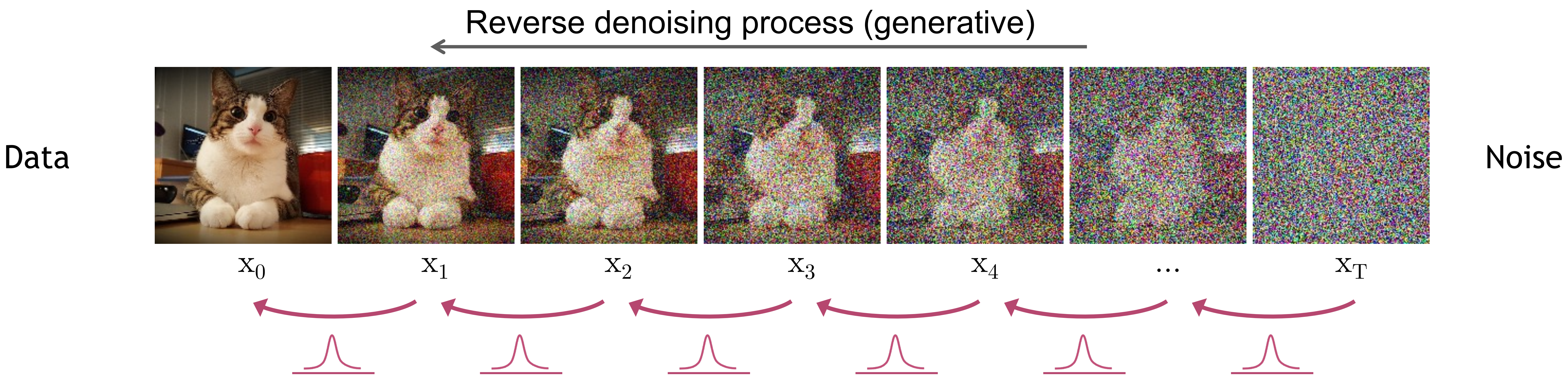


In general, $q(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is intractable.

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

Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



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

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

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

Learning Denoising Model

Variational upper bound

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

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

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

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

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

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

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

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

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

Training Objective Weighting

Trading likelihood for perceptual quality

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

The time dependent 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:

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

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

Summary

Training and Sample Generation

Algorithm 1 Training

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

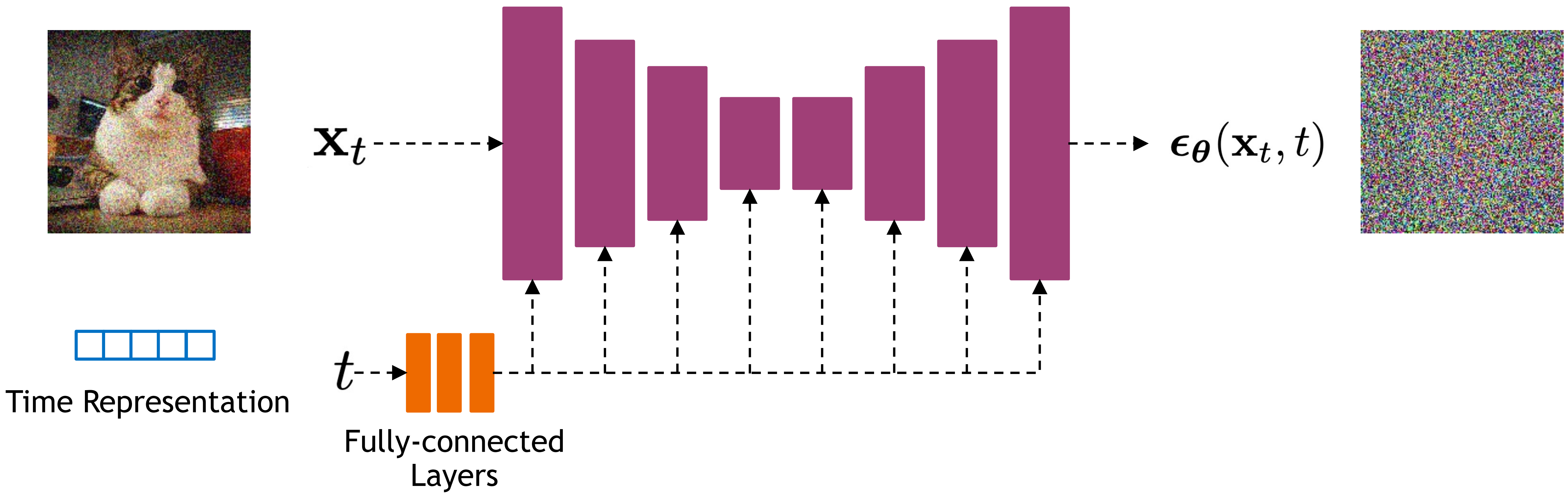
Algorithm 2 Sampling

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

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 Dharivwal and Nichol NeurIPS 2021)

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).

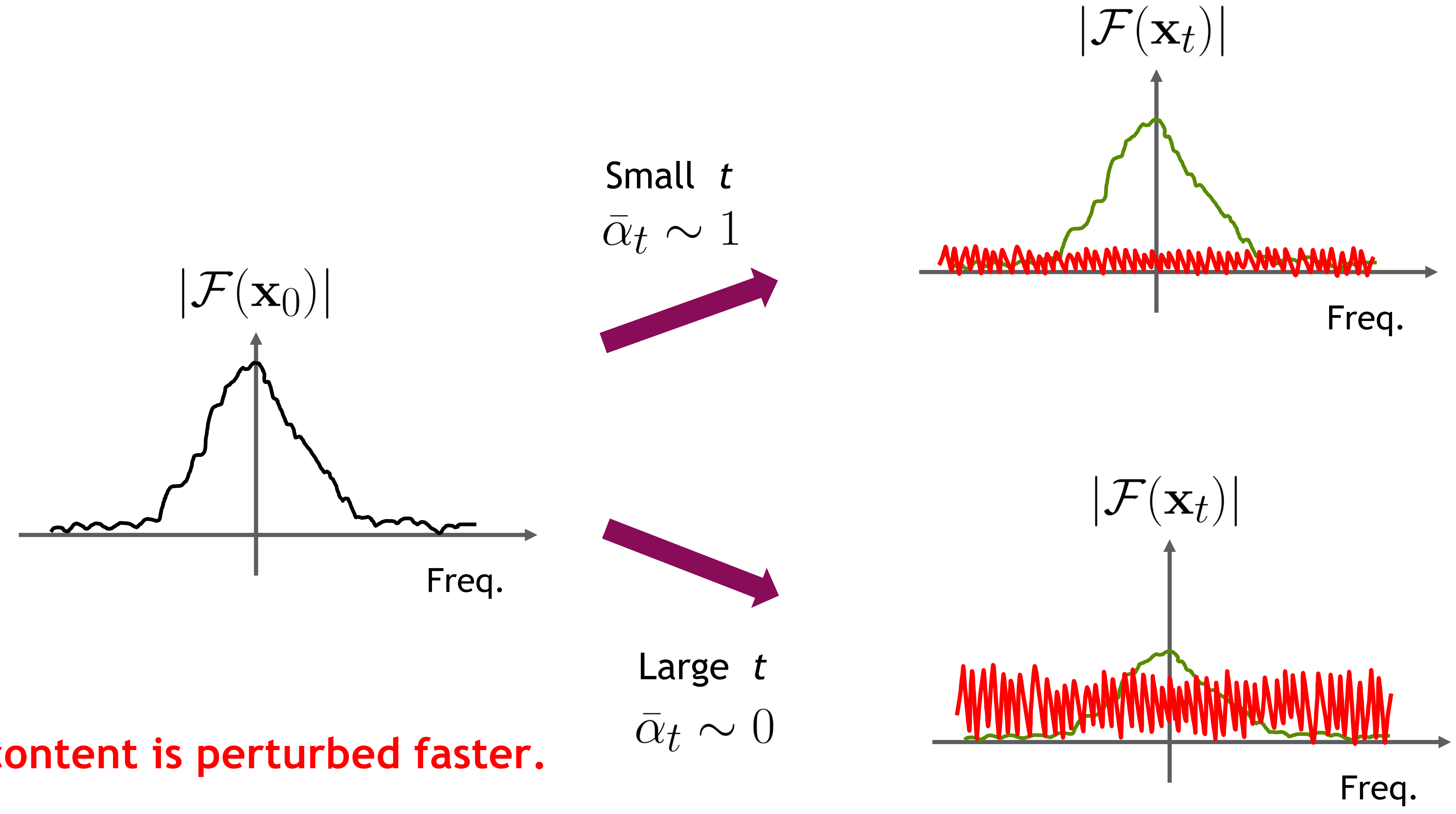
What happens to an image in the forward diffusion process?

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

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

Fourier Transform

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



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

Content-Detail Tradeoff

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

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

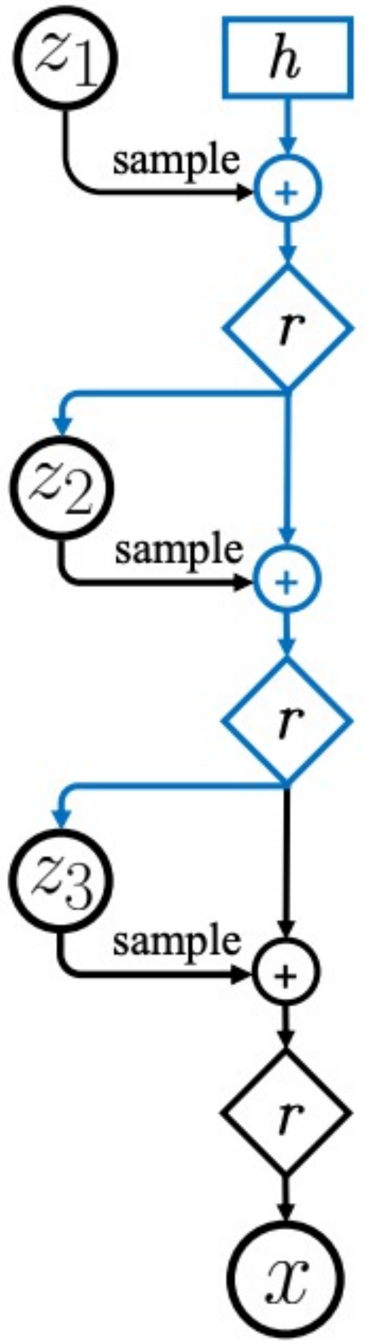
The weighting of the training objective for different timesteps is important!

Connection to VAEs

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

However, in diffusion models:

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



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

Summary
Denoising Diffusion Probabilistic Models

In this part, we reviewed denoising diffusion probabilistic models.

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

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

The devil is in the details:

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

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

Today's Program

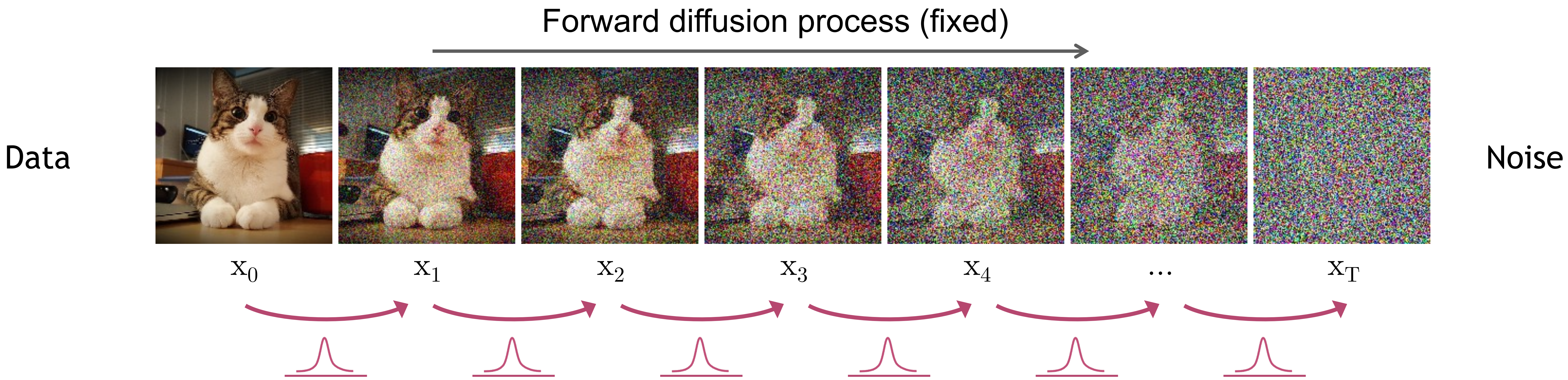
Title	Speaker	Time
Introduction	Arash	10 min
Part (1): Denoising Diffusion Probabilistic Models	Arash	35 min
Part (2): Score-based Generative Modeling with Differential Equations	Karsten	45 min
Part (3): Advanced Techniques: Accelerated Sampling, Conditional Generation, and Beyond	Ruiqi	45 min
Applications (1): Image Synthesis, Text-to-Image, Controllable Generation	Ruiqi	15 min
Applications (2): Image Editing, Image-to-Image, Super-resolution, Segmentation	Arash	15 min
Applications (3): Video Synthesis, Medical Imaging, 3D Generation, Discrete State Models	Karsten	15 min
Conclusions, Open Problems and Final Remarks	Arash	10 min

cvpr2022-tutorial-diffusion-models.github.io

Part (2):
Score-based Generative Modeling with Differential Equations

Forward Diffusion Process

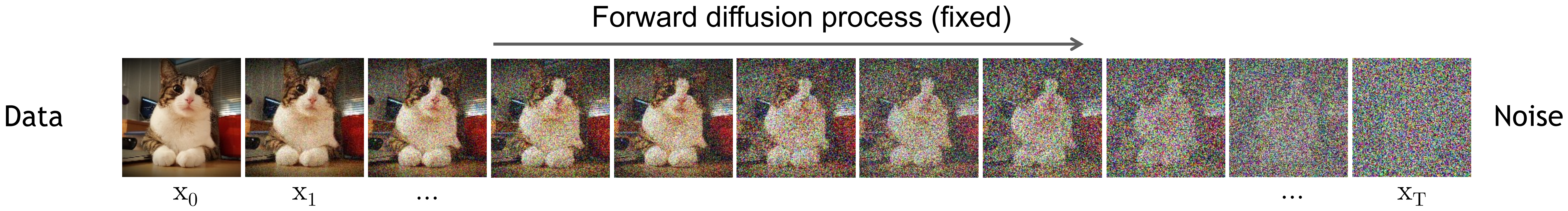
Consider the forward diffusion process again:



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

Forward Diffusion Process

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



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

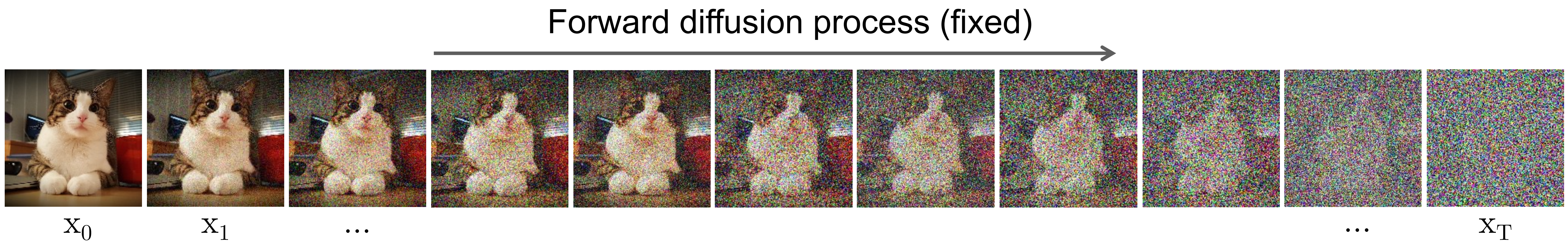
$$\Rightarrow \quad \mathbf{x}_t = \sqrt{1-\beta_t}\,\mathbf{x}_{t-1} + \sqrt{\beta_t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

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

Forward Diffusion Process

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

Data



Noise

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

$$\begin{aligned}\mathbf{x}_t &= \sqrt{1-\beta_t}\,\mathbf{x}_{t-1} + \sqrt{\beta_t}\,\mathcal{N}(\mathbf{0},\mathbf{I}) \\ \Rightarrow \quad &= \sqrt{1-\beta(t)\Delta t}\,\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0},\mathbf{I})\end{aligned} \qquad (\beta_t := \beta(t)\Delta t)$$

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

Forward Diffusion Process

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

Forward diffusion process (fixed)

Data

Noise

x_0 x_1 $\cdots$ $\cdots$ x_T

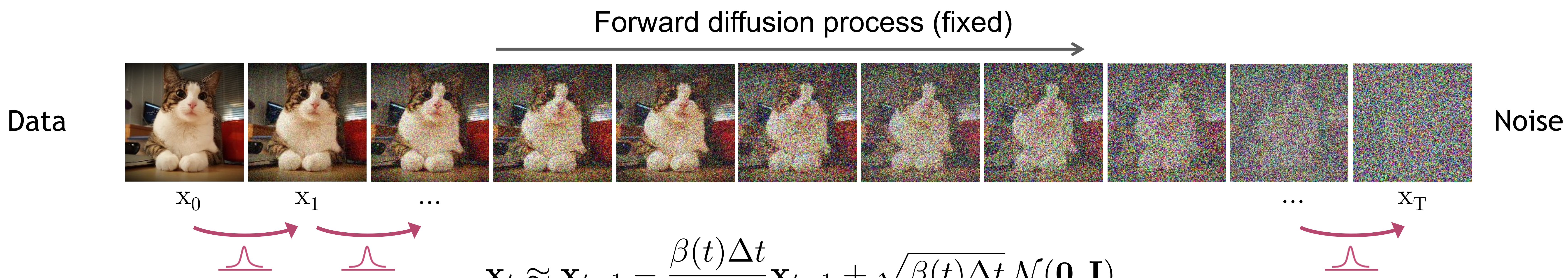
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

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

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

Forward Diffusion Process as Stochastic Differential Equation

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



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

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

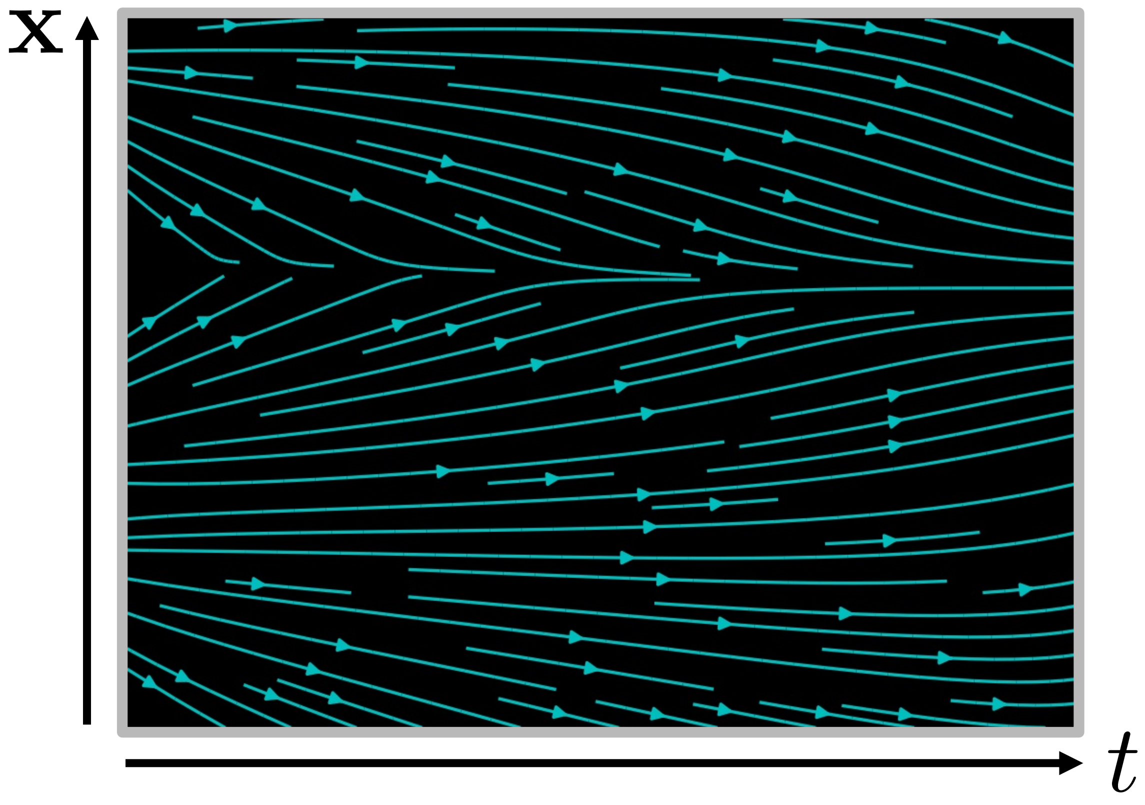
Stochastic Differential Equation (SDE)
describing the diffusion in infinitesimal limit

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

Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

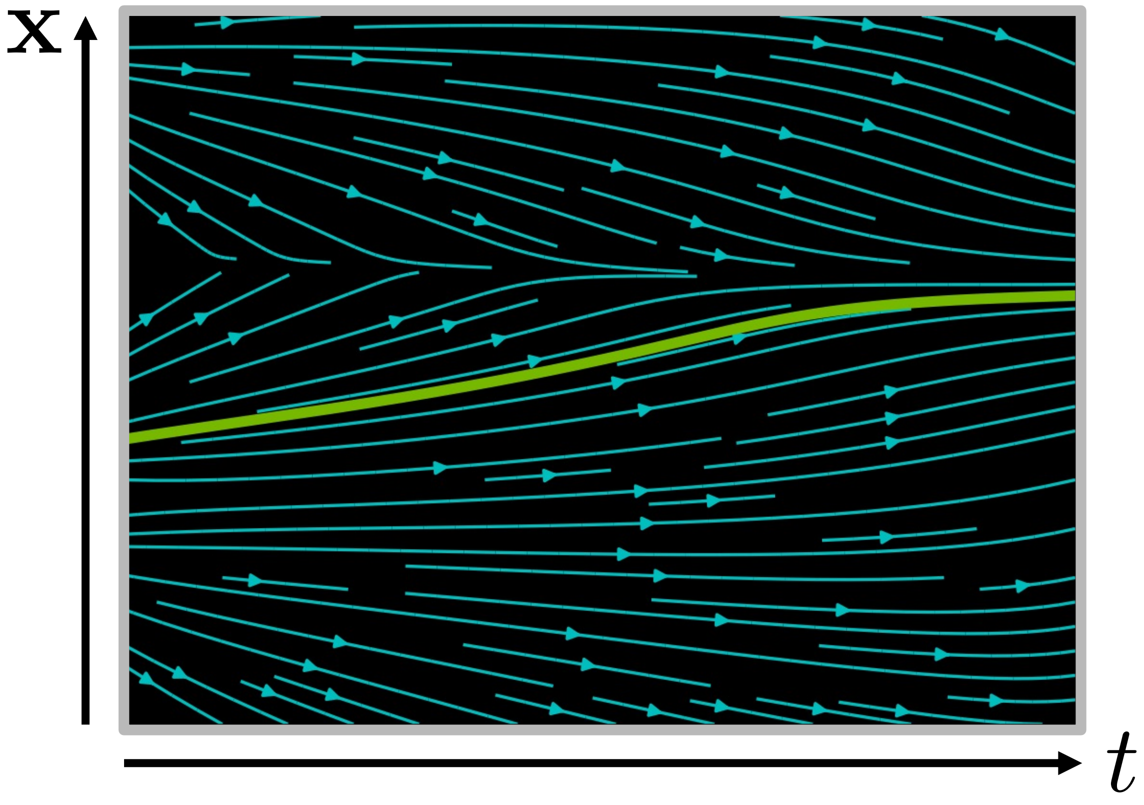
$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) \quad \text{or} \quad \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t$$



Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) \quad \text{or} \quad \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t$$



Analytical Solution:

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$$

Iterative Numerical Solution:

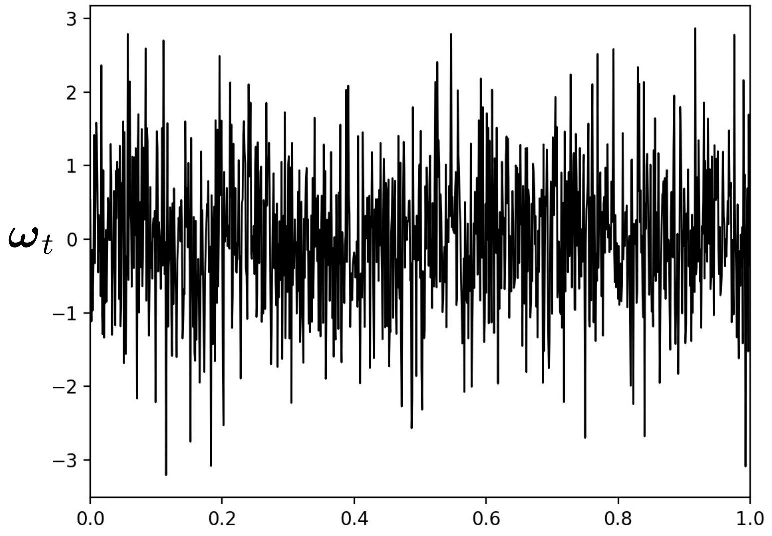
$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$$

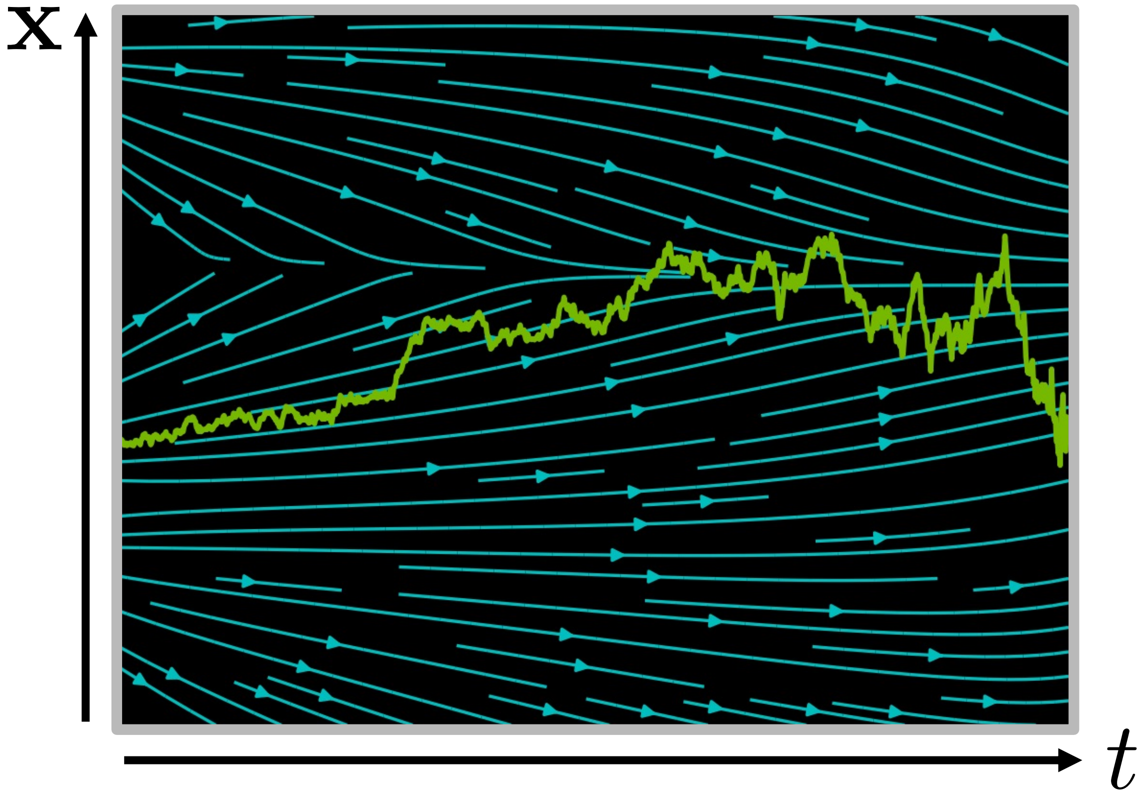
Stochastic Differential Equation (SDE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \underbrace{\mathbf{f}(\mathbf{x}, t)}_{\text{drift coefficient}} + \underbrace{\sigma(\mathbf{x}, t)}_{\text{diffusion coefficient}}\boldsymbol{\omega}_t$$

$$\left(\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + \sigma(\mathbf{x}, t)\mathrm{d}\boldsymbol{\omega}_t \right)$$

Wiener Process (Gaussian White Noise)



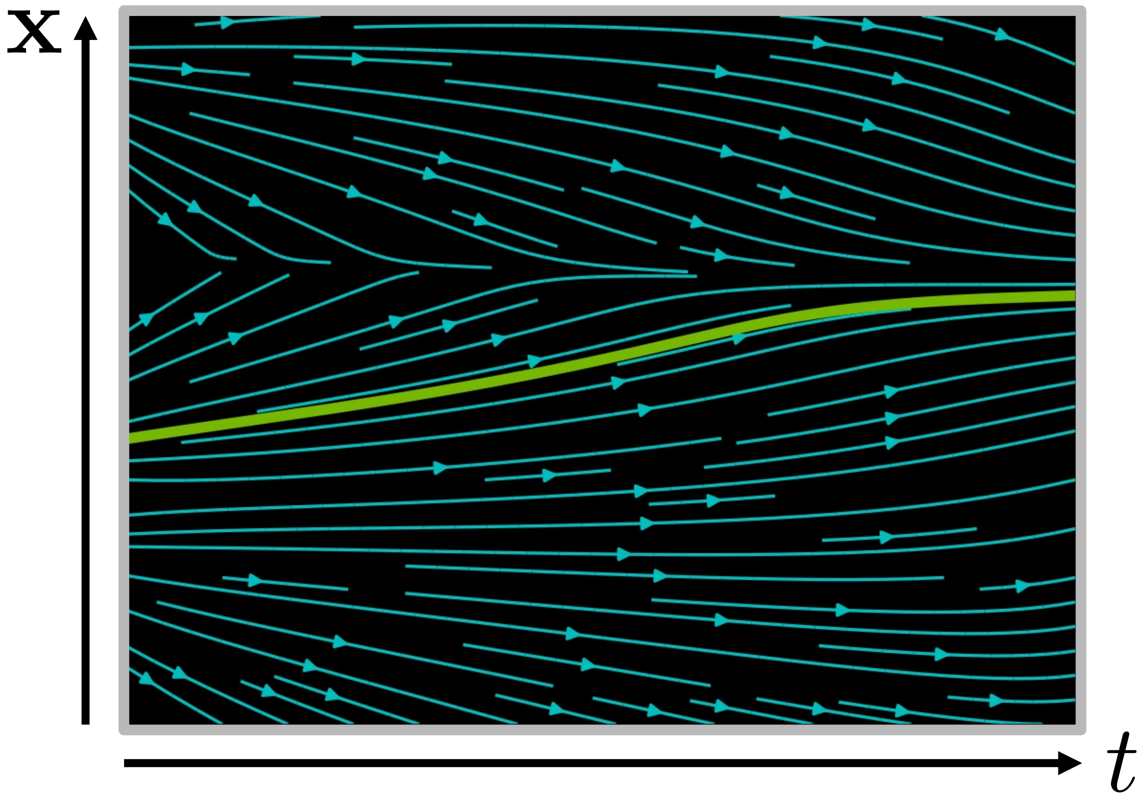


$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + \sigma(\mathbf{x}(t), t)\sqrt{\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

Crash Course in Differential Equations

Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) \quad \text{or} \quad \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t$$



Analytical Solution:

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$$

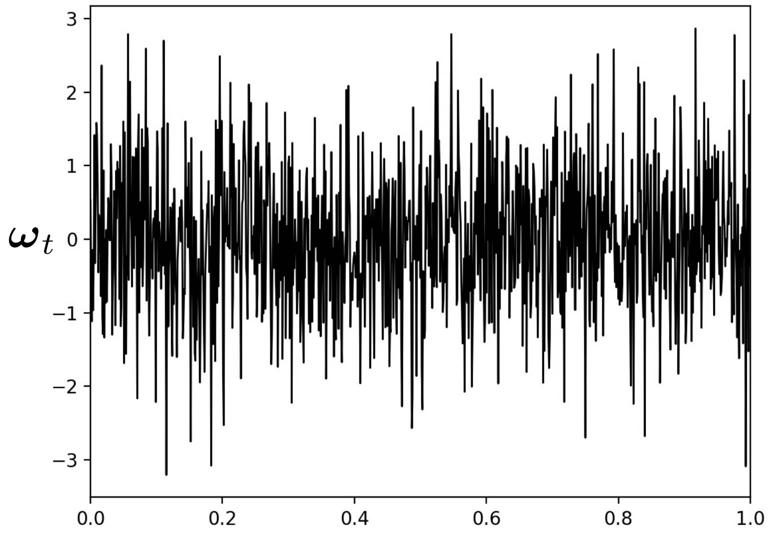
Iterative Numerical Solution:

$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$$

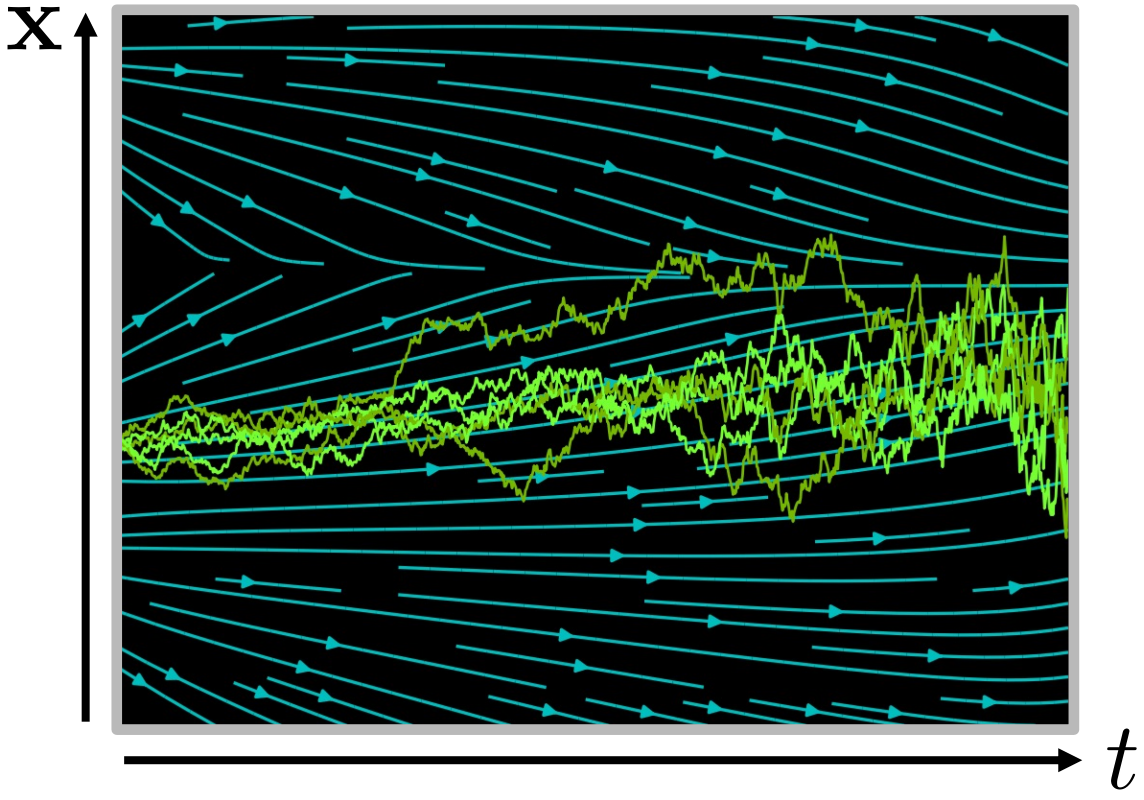
Stochastic Differential Equation (SDE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \underbrace{\mathbf{f}(\mathbf{x}, t)}_{\text{drift coefficient}} + \underbrace{\sigma(\mathbf{x}, t)}_{\text{diffusion coefficient}}\boldsymbol{\omega}_t$$

Wiener Process (Gaussian White Noise)



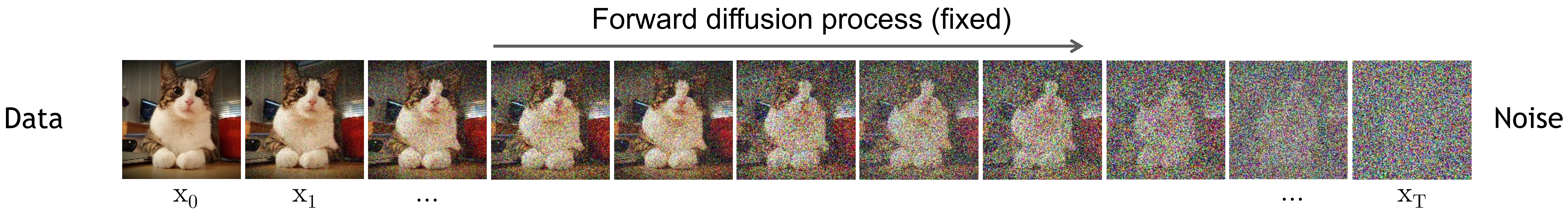
$$\left(\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + \sigma(\mathbf{x}, t)\mathrm{d}\boldsymbol{\omega}_t \right)$$



$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + \sigma(\mathbf{x}(t), t)\sqrt{\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

Forward Diffusion Process as Stochastic Differential Equation

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



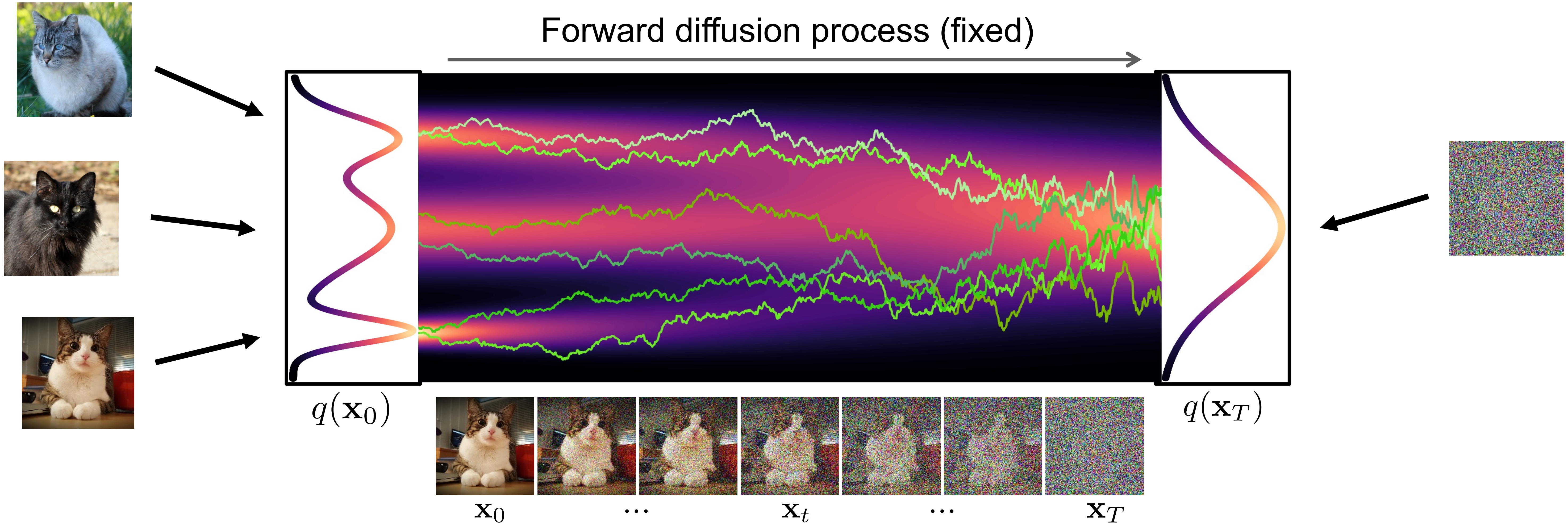
$$\mathbf{x}_t \approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

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

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

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

Forward Diffusion Process as Stochastic Differential Equation

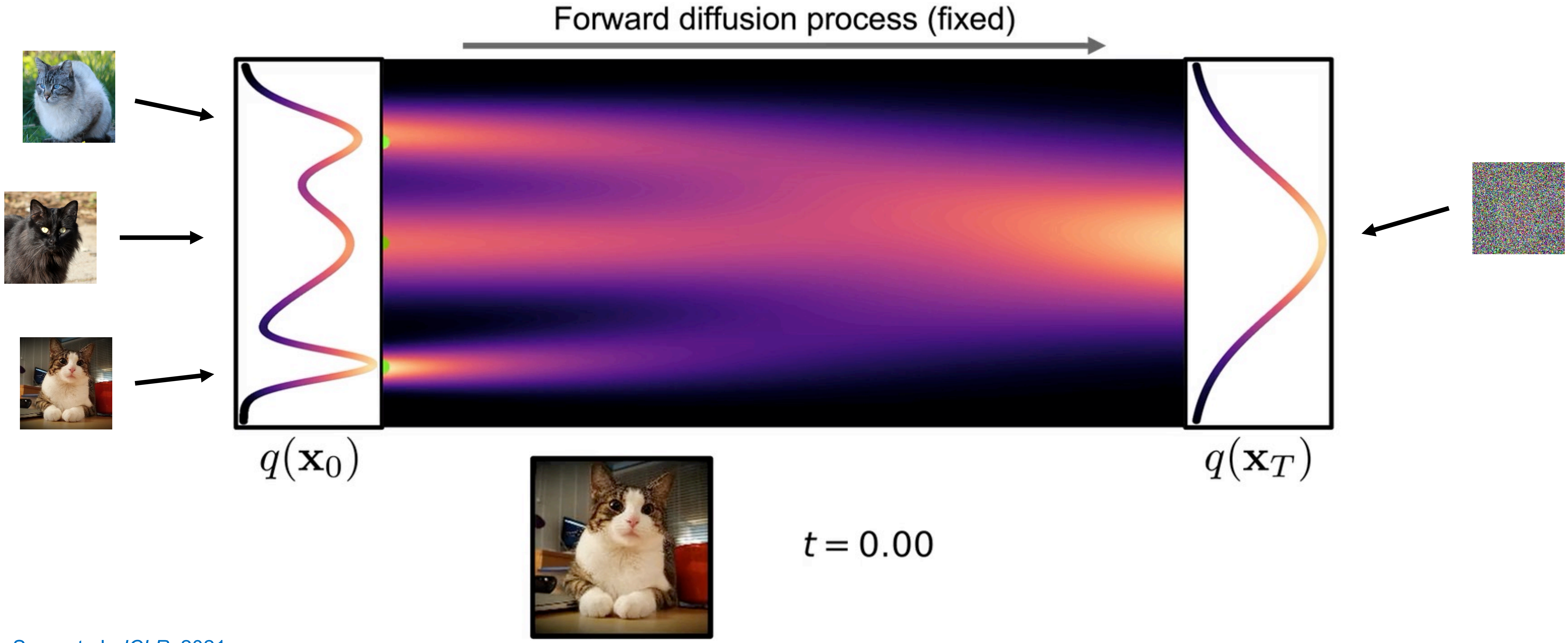


Forward Diffusion SDE:

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

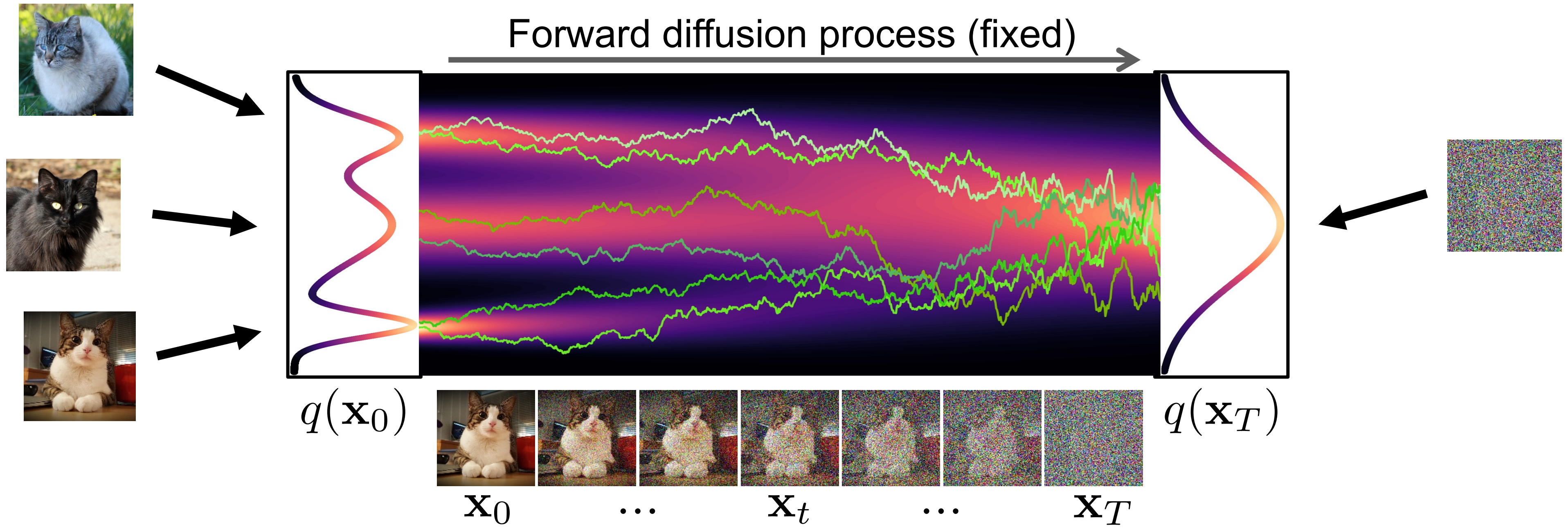
Song et al., *ICLR*, 2021

Forward Diffusion Process as Stochastic Differential Equation



Song et al., *ICLR*, 2021

Forward Diffusion Process as Stochastic Differential Equation



Forward Diffusion SDE:

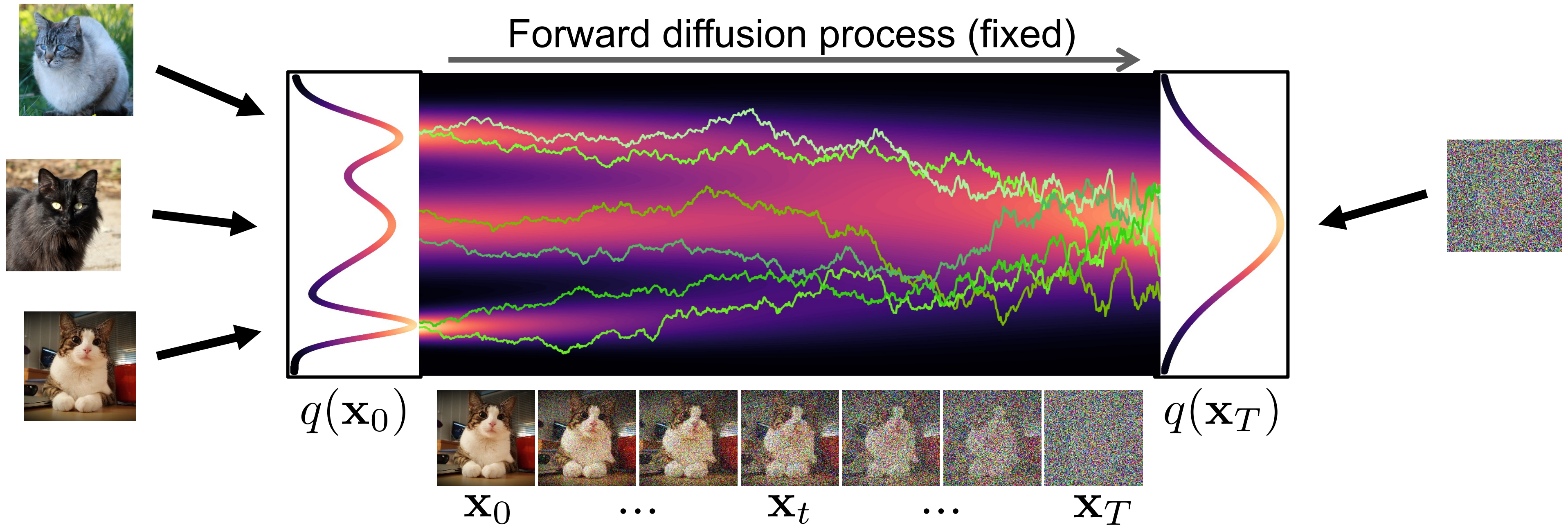
$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t}_{\substack{\text{drift term}\\ \text{(pulls towards mode)}}} + \underbrace{\sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t}_{\substack{\text{diffusion term}\\ \text{(injects noise)}}}$$

Special case of more general SDEs used in generative diffusion models:

$$\mathrm{d}\mathbf{x}_t = f(t)\mathbf{x}_t\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{\omega}_t$$

Song et al., *ICLR*, 2021

The Generative Reverse Stochastic Differential Equation



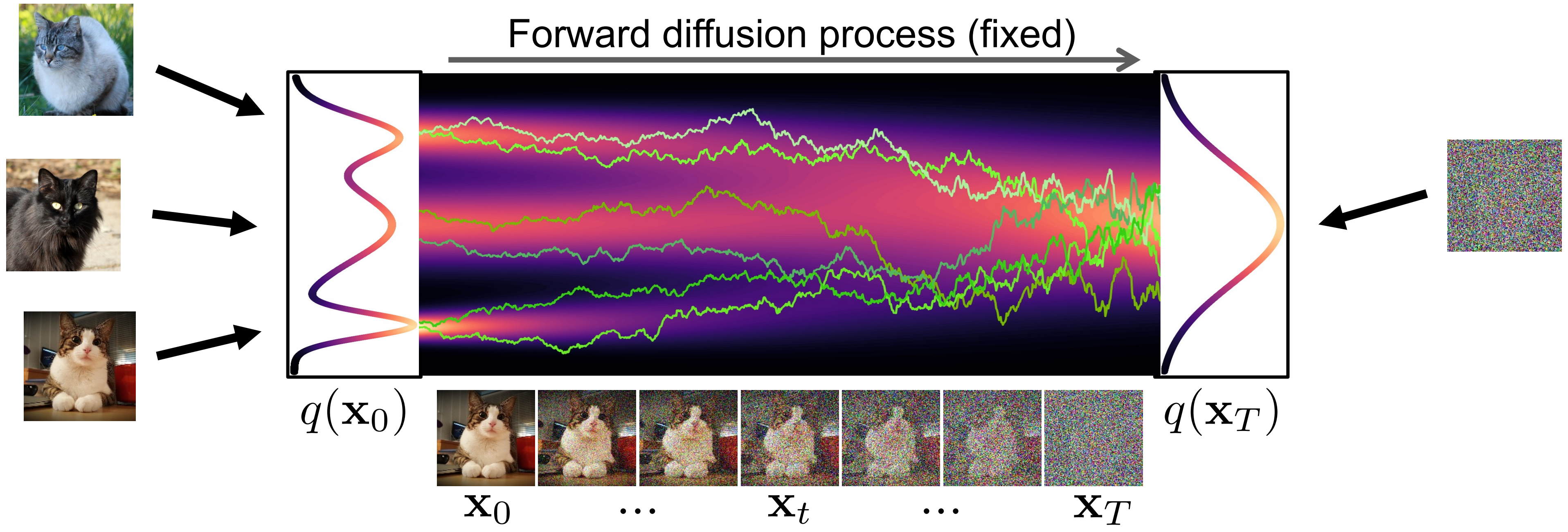
Forward Diffusion SDE:

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

But what about the reverse direction, necessary for generation?

Song et al., *ICLR*, 2021

The Generative Reverse Stochastic Differential Equation



Forward Diffusion SDE:

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

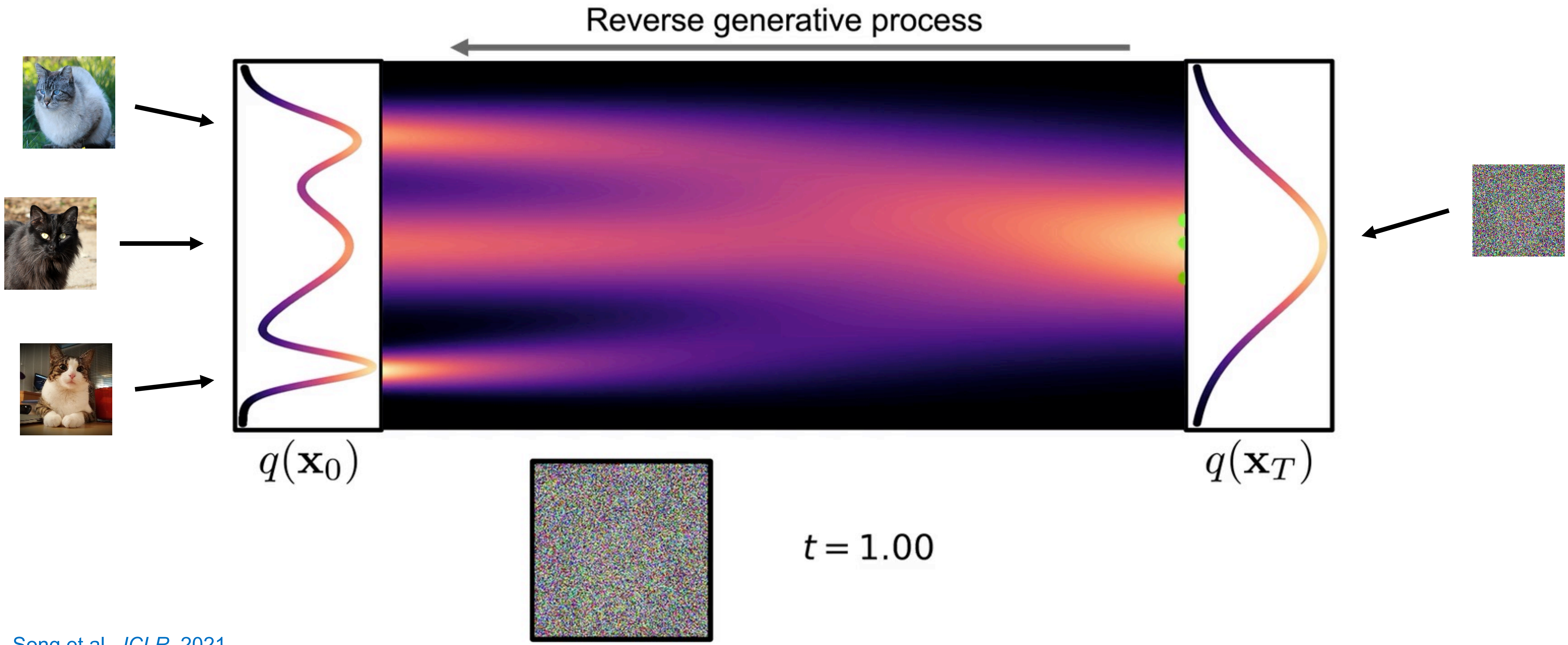
Reverse Generative Diffusion SDE:

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

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

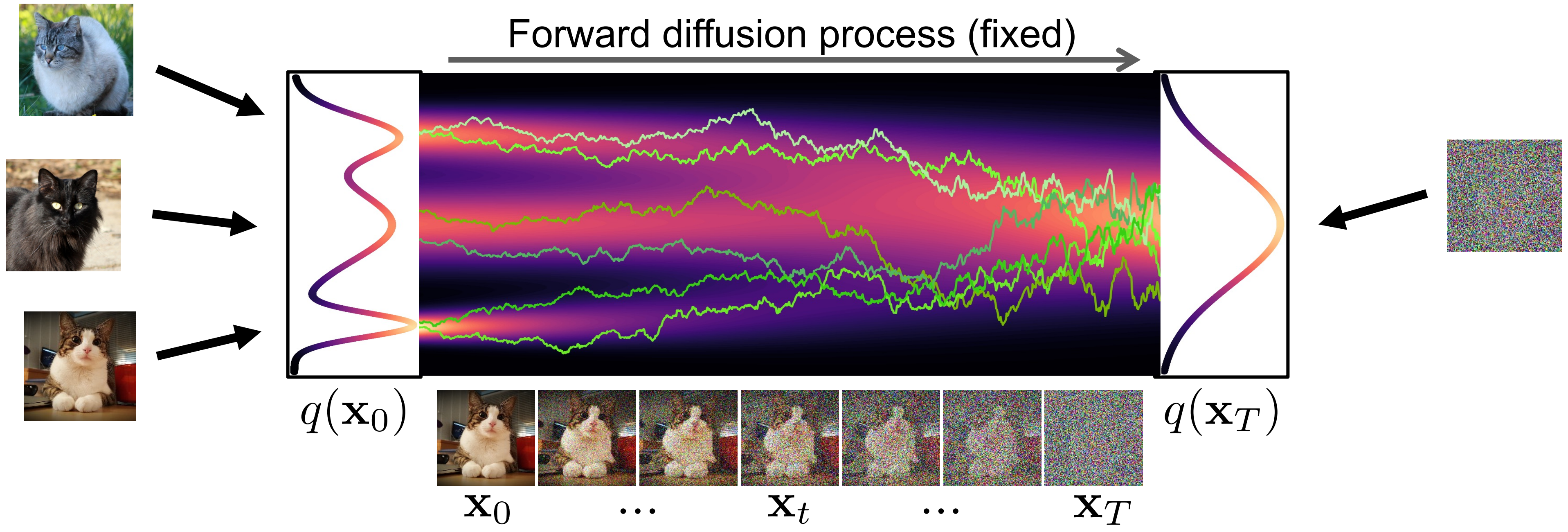
Song et al., *ICLR*, 2021
Anderson, in *Stochastic Processes and their Applications*, 1982

The Generative Reverse Stochastic Differential Equation



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

The Generative Reverse Stochastic Differential Equation



Forward Diffusion SDE:

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

Reverse Generative Diffusion SDE:

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

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

[Song et al., *ICLR*, 2021](#)
[Anderson, in *Stochastic Processes and their Applications*, 1982](#)

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: $d\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,dt + \sqrt{\beta(t)}\,d\omega_t$

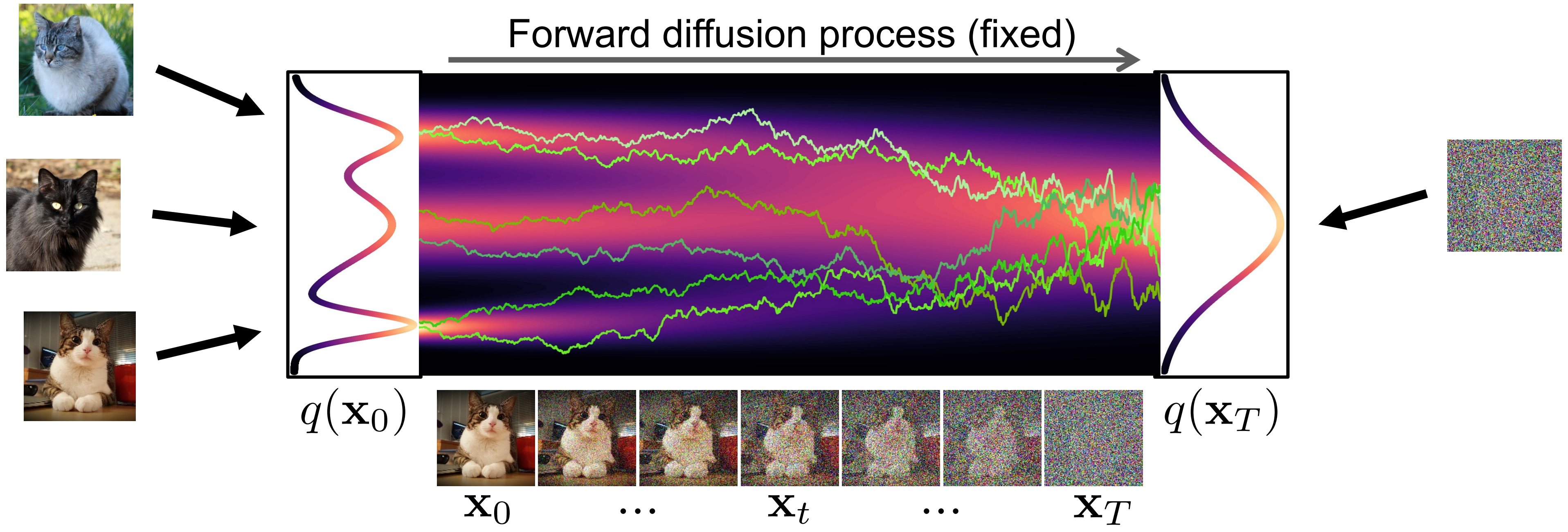
Reverse Generative Diffusion SDE:

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

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

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

Score Matching



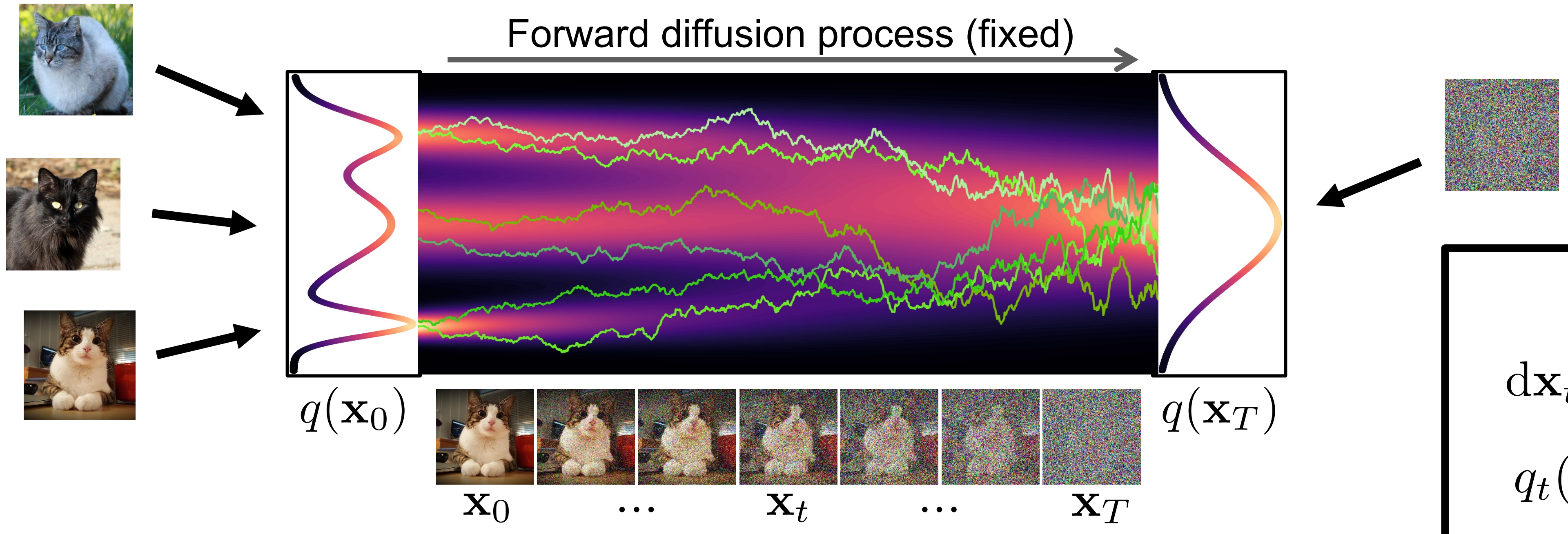
- Naïve idea, learn model for the score function by direct regression?

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

➡ **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

Denoising Score Matching



"Variance Preserving" SDE:

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

$$q_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \gamma_t\mathbf{x}_0, \sigma_t^2\mathbf{I})$$

$$\gamma_t = e^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

$$\sigma_t^2 = 1 - e^{-\int_0^t \beta(s)ds}$$

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

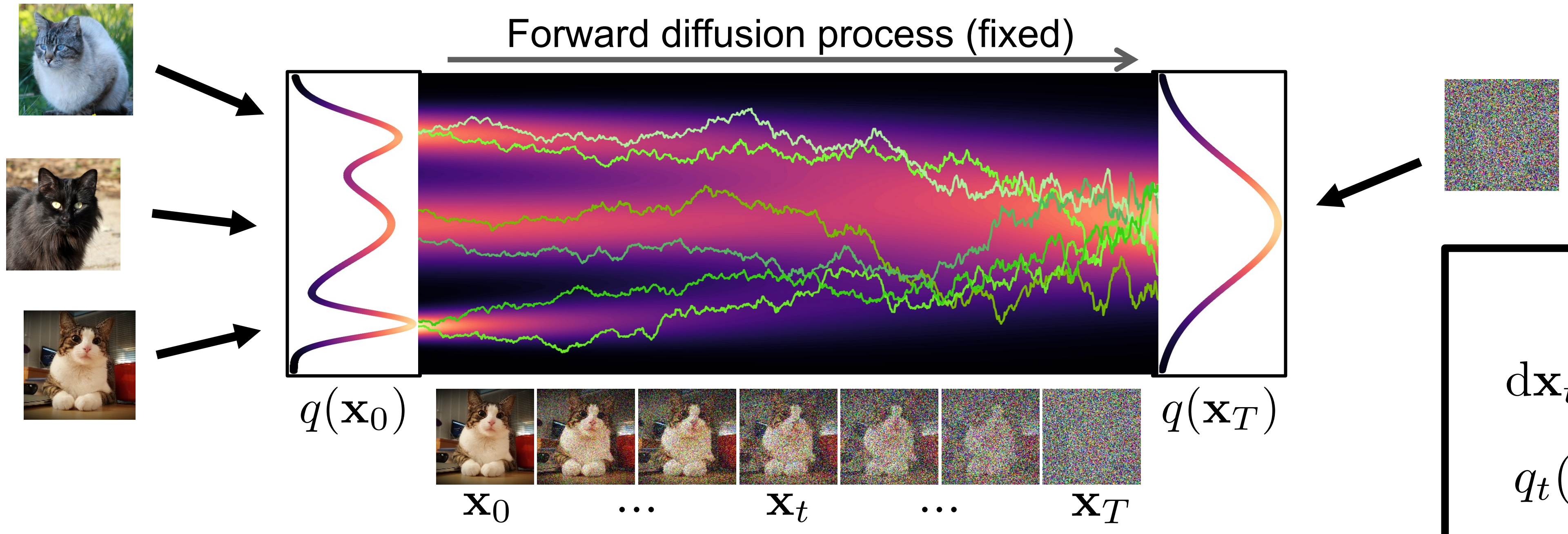
$$\min_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_{t\sim\mathcal{U}(0,T)}}_{\text{diffusion time } t} \underbrace{\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}}_{\text{data sample } \mathbf{x}_0} \underbrace{\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)}}_{\text{diffused data sample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

➡ **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

Denoising Score Matching

Implementation 1: Noise Prediction



"Variance Preserving" SDE:

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

$$q_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \gamma_t\mathbf{x}_0, \sigma_t^2\mathbf{I})$$

$$\gamma_t = e^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

$$\sigma_t^2 = 1 - e^{-\int_0^t \beta(s)ds}$$

- **Denoising Score Matching:**

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

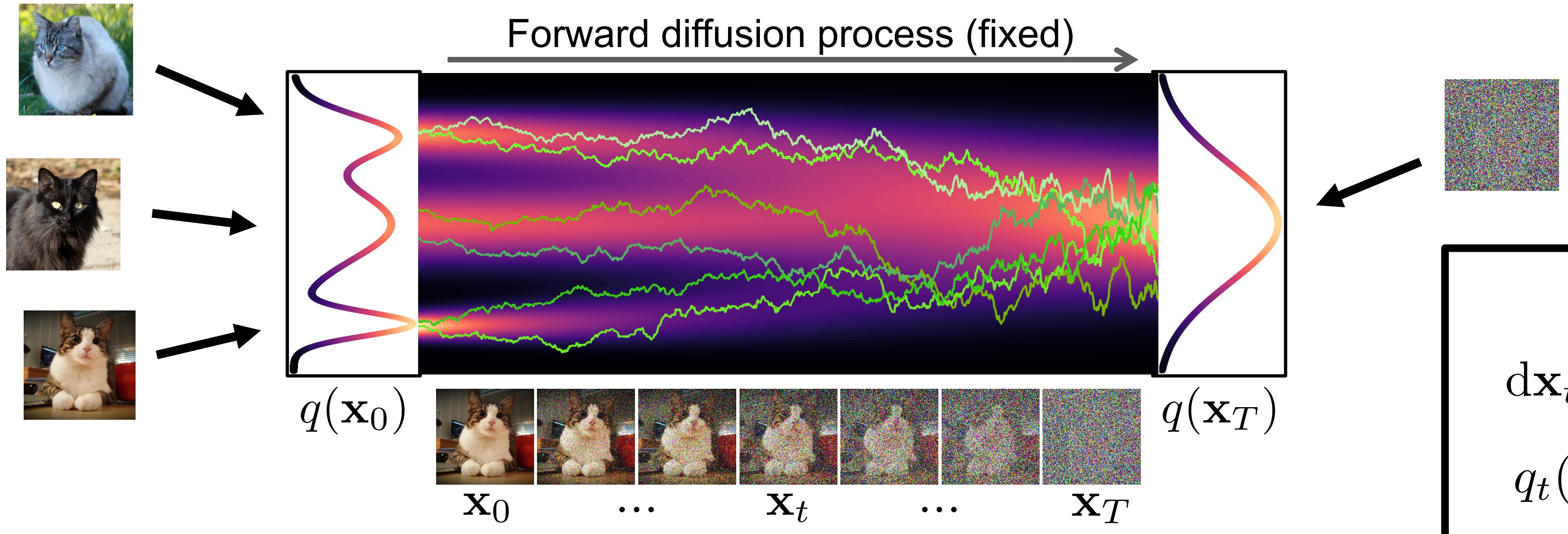
  - Re-parametrized sampling: $\mathbf{x}_t = \gamma_t\mathbf{x}_0 + \sigma_t\boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$
    - Score function: $\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t|\mathbf{x}_0) = -\nabla_{\mathbf{x}_t}\frac{(\mathbf{x}_t-\gamma_t\mathbf{x}_0)^2}{2\sigma_t^2}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

$$\Rightarrow \min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\frac{1}{\sigma_t^2}||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)||_2^2$$

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

Denoising Score Matching

Implementation 2: Loss Weightings



"Variance Preserving" SDE:

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

$$q_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \gamma_t\mathbf{x}_0, \sigma_t^2\mathbf{I})$$

$$\gamma_t = e^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

$$\sigma_t^2 = 1 - e^{-\int_0^t \beta(s)ds}$$

- **Denoising Score Matching** objective with loss weighting $\lambda(t)$:

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

Different loss weightings trade off between model with good perceptual quality vs. high log-likelihood

- *Perceptual quality:* $\lambda(t) = \sigma_t^2$
- *Maximum log-likelihood:* $\lambda(t) = \beta(t)$ *(negative ELBO)*

➡ **Same objectives as derived with variational approach in Part (1)!**

Ho et al, *NeurIPS*, 2020
Song et al., *NeurIPS*, 2021
Kingma et al., *NeurIPS*, 2021
Vahdat et al., *NeurIPS*, 2021
Huang et al., *NeurIPS*, 2021
Karras et al., *arXiv*, 2022

Denoising Score Matching

Implementation 2: Loss Weightings

More sophisticated model parametrizations and loss weightings possible!

➡ [Karras et al., "Elucidating the Design Space of Diffusion-Based Generative Models", *arXiv*, 2022]

Denoising Score Matching

Implementation 3: Variance Reduction and Numerical Stability

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

- Notice $\sigma_t^2 \to 0$, as $t \to 0$. Loss heavily amplified when sampling t close to 0 (for $\lambda(t)=\beta(t)$). **High variance!**

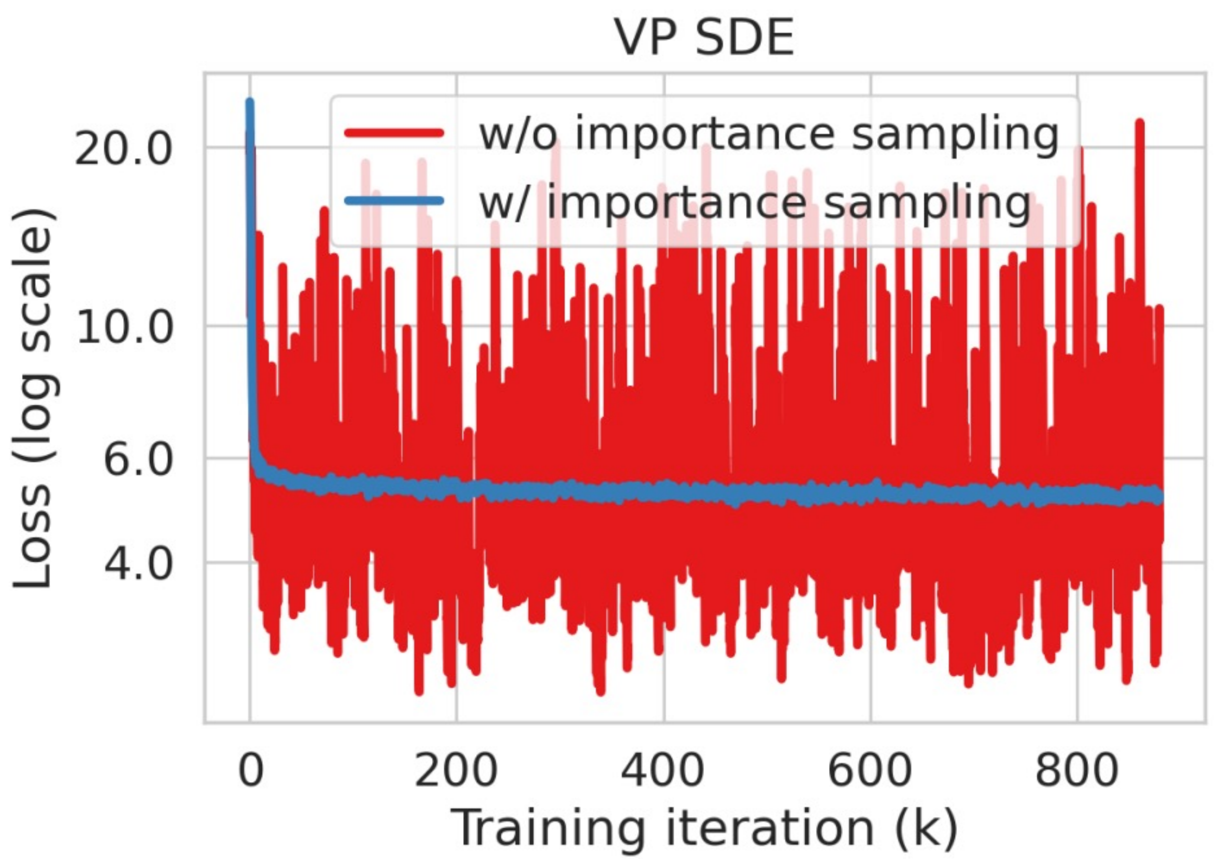
- 1. Train with small **time cut-off** η ($\approx 10^{-5}$):

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

- 2. Variance reduction by **Importance Sampling:**

Importance sampling distribution: $r(t) \propto \frac{\lambda(t)}{\sigma_t^2}$

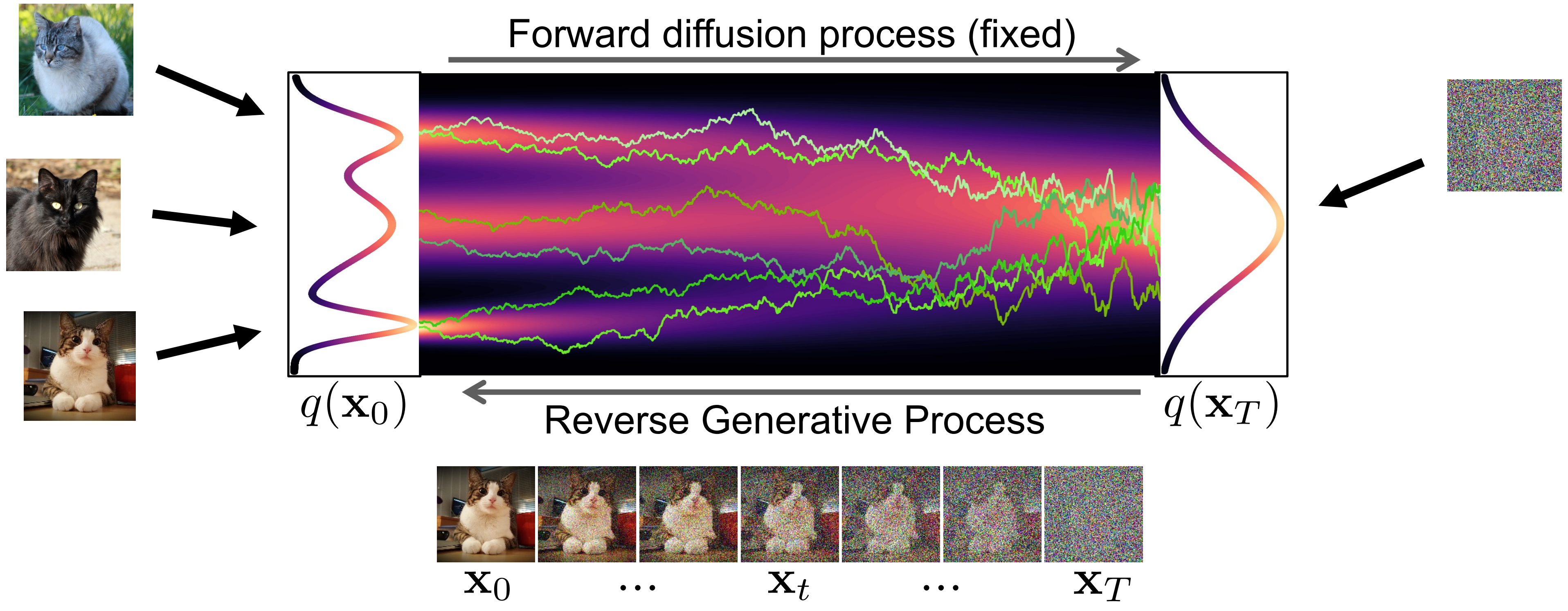
$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim r(t)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\frac{1}{r(t)}\frac{\lambda(t)}{\sigma_t^2}||\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)||_2^2$$



(image from: Song et al., “Maximum Likelihood Training of Score-Based Diffusion Models“, *NeurIPS*, 2021)

Song et al., *NeurIPS*, 2021
Vahdat et al., *NeurIPS*, 2021
Huang et al., *NeurIPS*, 2021

Probability Flow ODE



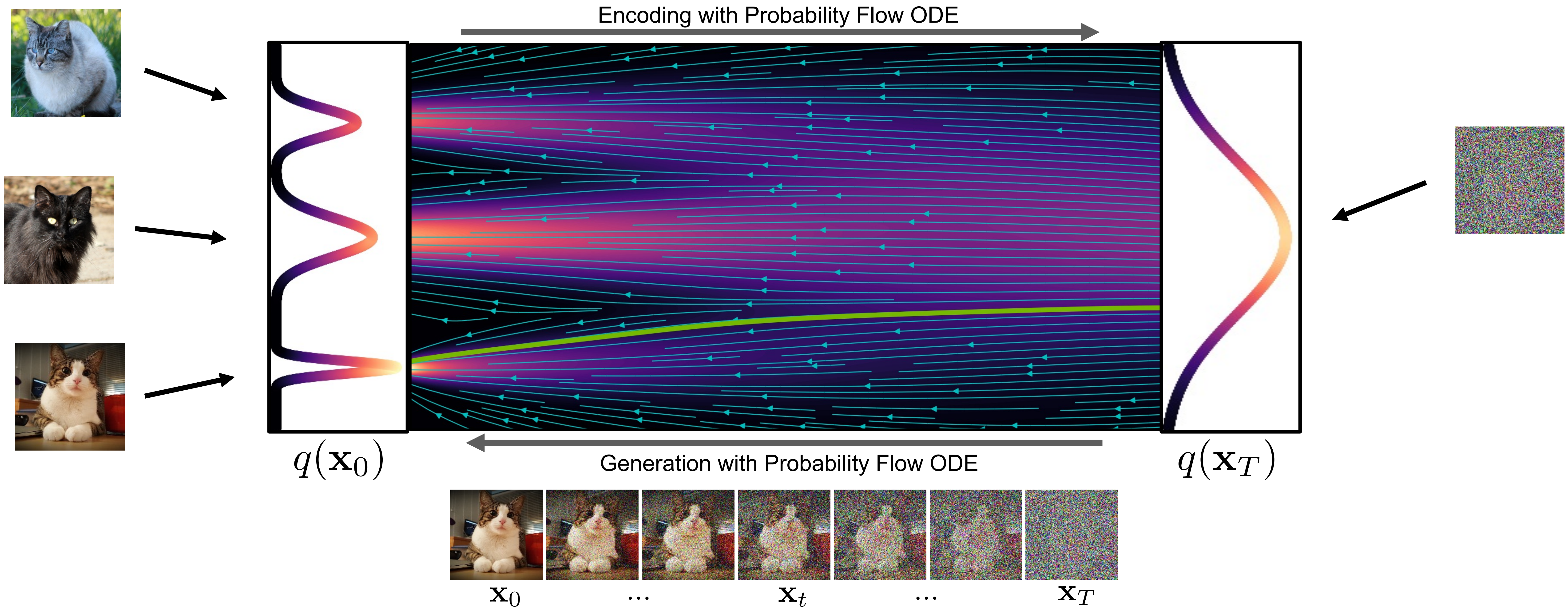
- Consider reverse generative diffusion SDE: $$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)\right]dt + \sqrt{\beta(t)}\,d\bar{\boldsymbol{\omega}}_t$$

- In distribution equivalent to "**Probability Flow ODE**": $$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)\right]dt$$

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

Song et al., *ICLR*, 2021

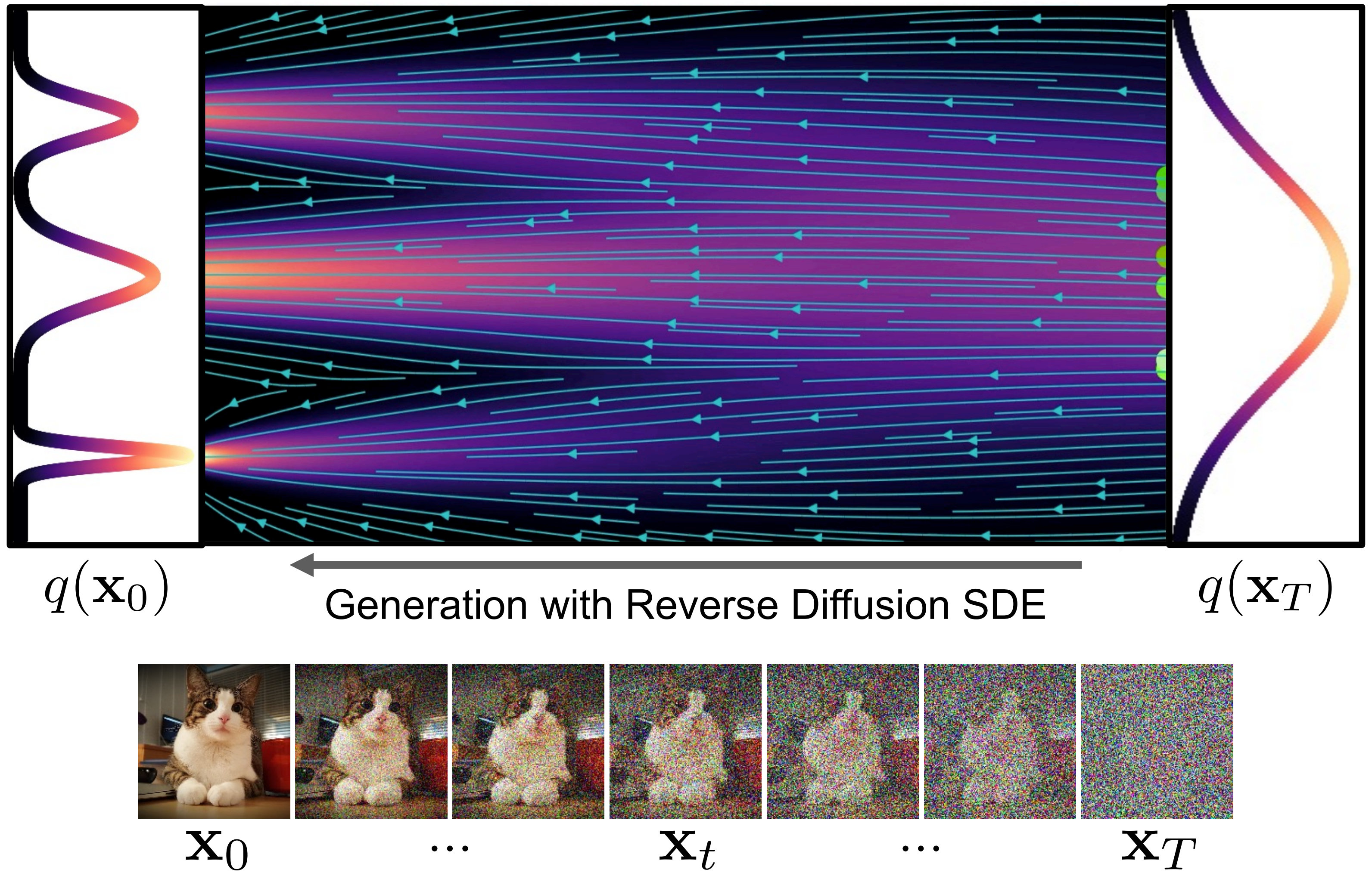
Probability Flow ODE

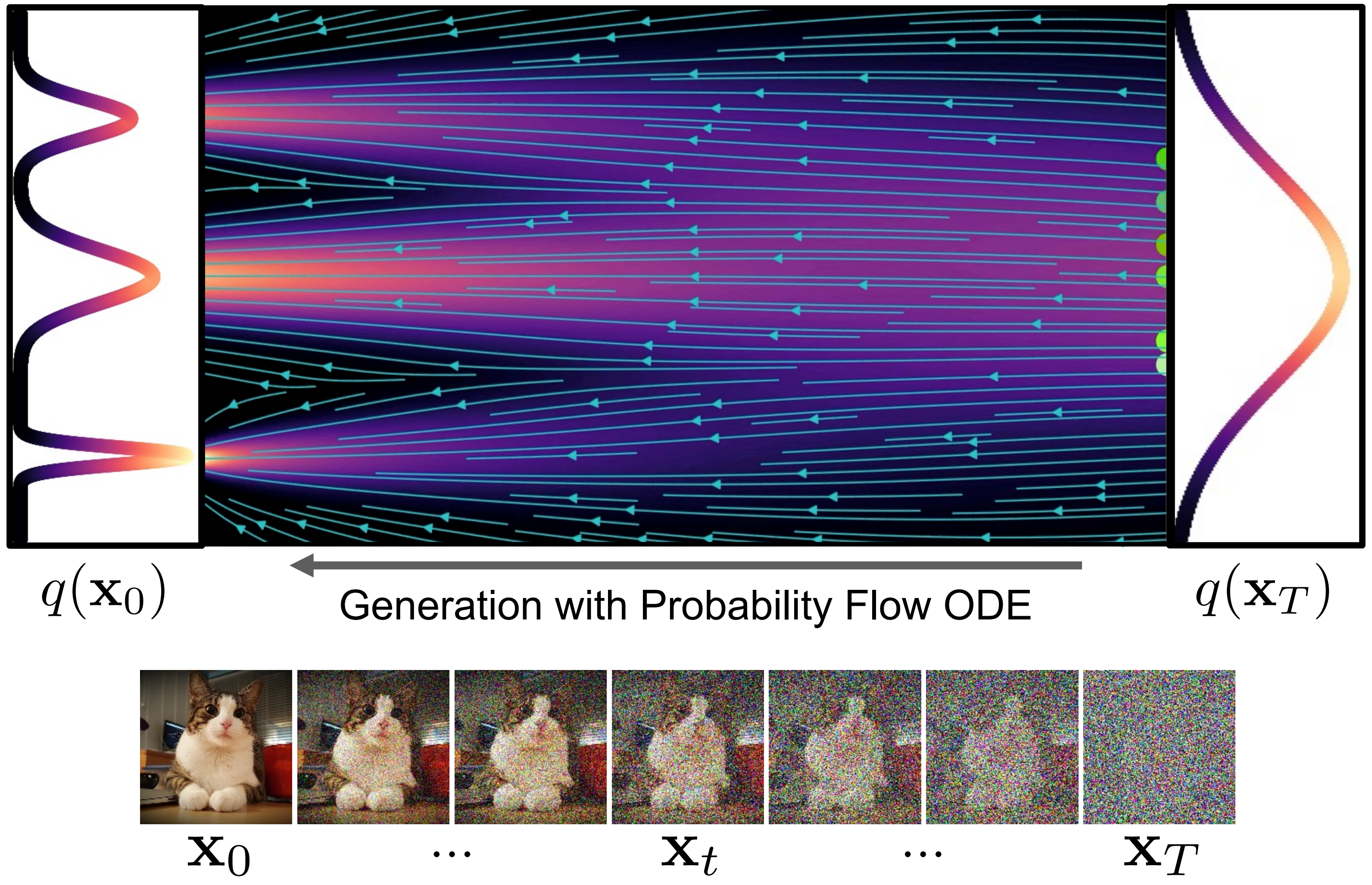


$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t$$

Song et al., *ICLR*, 2021

Synthesis with SDE vs. ODE





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

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

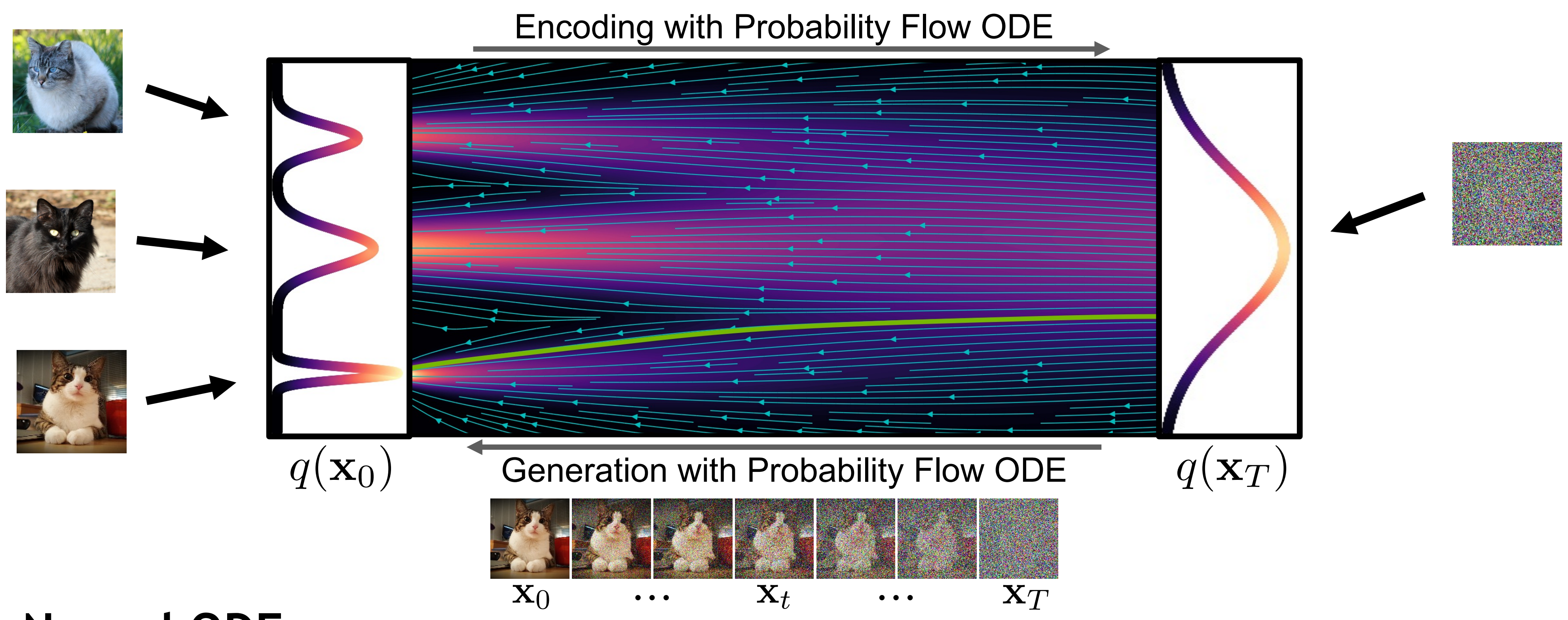
- **Generative Probability Flow ODE (deterministic):**

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

Song et al., *ICLR*, 2021

Probability Flow ODE

Diffusion Models as Continuous Normalizing Flows



- **Probability Flow ODE as Neural ODE or Continuous Normalizing Flow (CNF):**

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

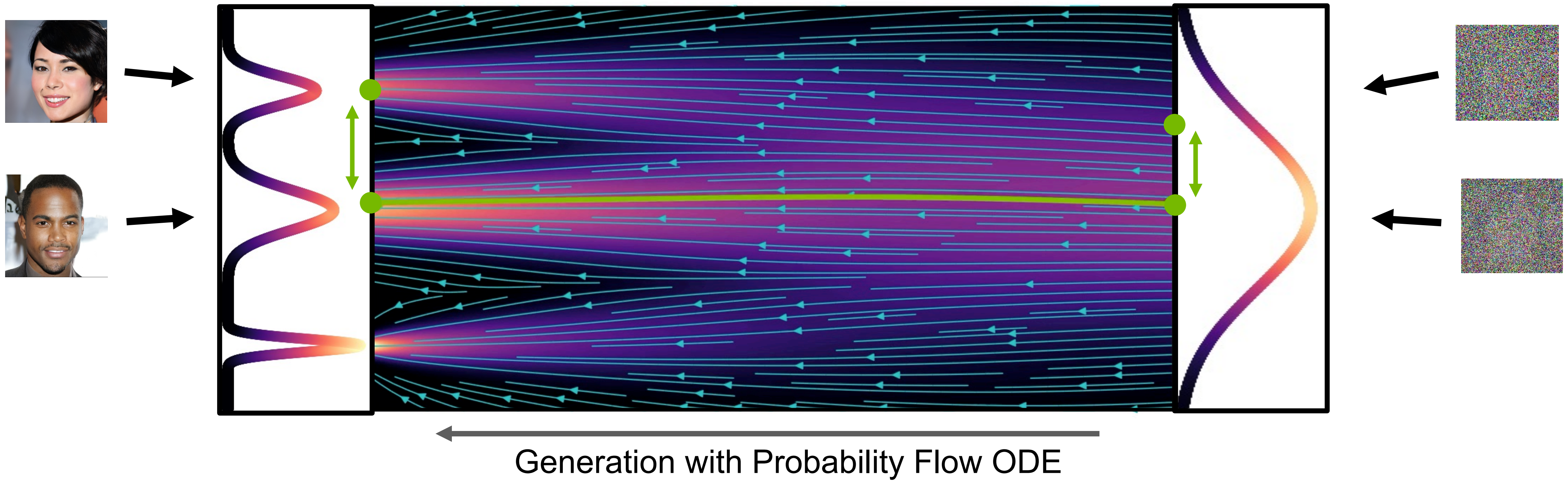
$$\left(\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\right)$$

➡ Enables use of **advanced ODE solvers**

➡ **Deterministic encoding and generation** (semantic image interpolation, etc.)

Chen et al., *NeurIPS*, 2018
Grathwohl, *ICLR*, 2019
Song et al., *ICLR*, 2021

Semantic Image Interpolation with Probability Flow ODE

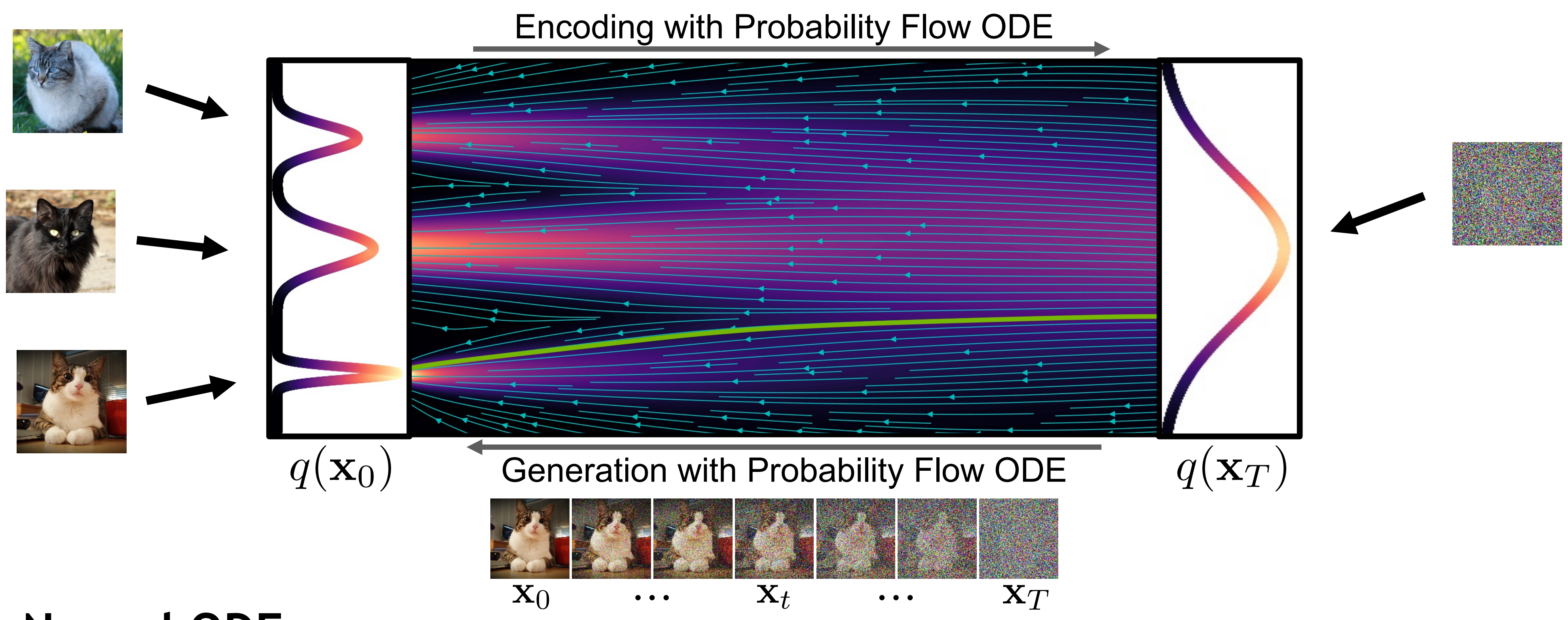


Continuous changes in latent space ($\mathbf{x}_T$) result in continuous, semantically meaningful changes in data space ($\mathbf{x}_0$)!

Dockhorn et al., *ICLR*, 2022

Probability Flow ODE

Diffusion Models as Continuous Normalizing Flows



- **Probability Flow ODE as Neural ODE or Continuous Normalizing Flow (CNF):**

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

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

➡ Enables use of **advanced ODE solvers**

➡ **Deterministic encoding and generation** (semantic image interpolation, etc.)

➡ **Log-likelihood computation** (instantaneous change of variables):

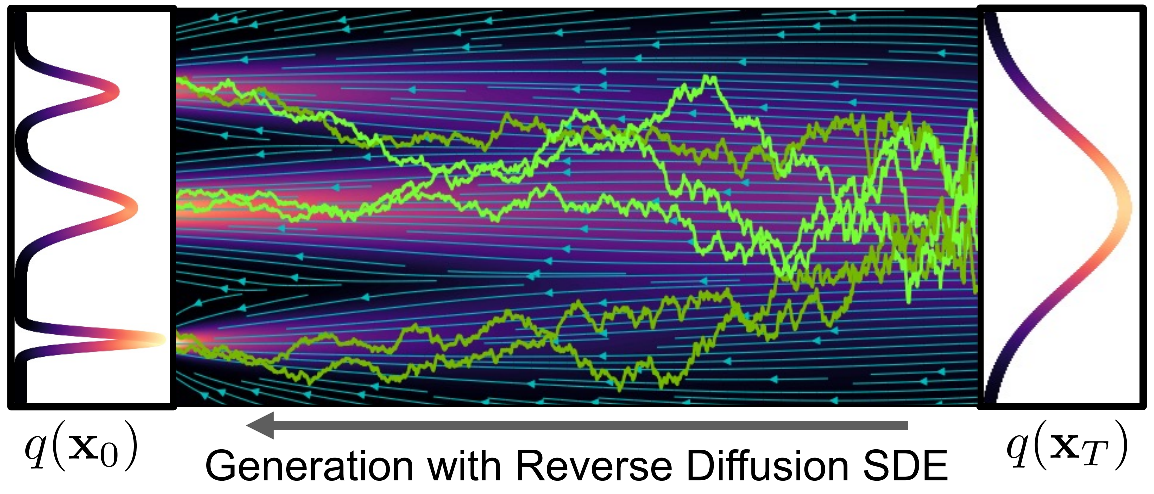
$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_0) = \log p_T(\mathbf{x}_T) - \int_0^T \mathrm{Tr}\left(\frac{1}{2}\beta(t)\frac{\partial}{\partial \mathbf{x}_t}\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\right)\mathrm{d}t$$

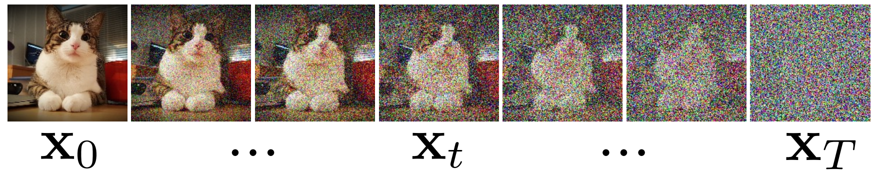
Chen et al., *NeurIPS*, 2018
Grathwohl, *ICLR*, 2019
Song et al., *ICLR*, 2021

➡ *Diffusion models can be considered CNFs trained with score matching!*

Sampling from "Continuous-Time" Diffusion Models

How to solve the generative SDE or ODE in practice?





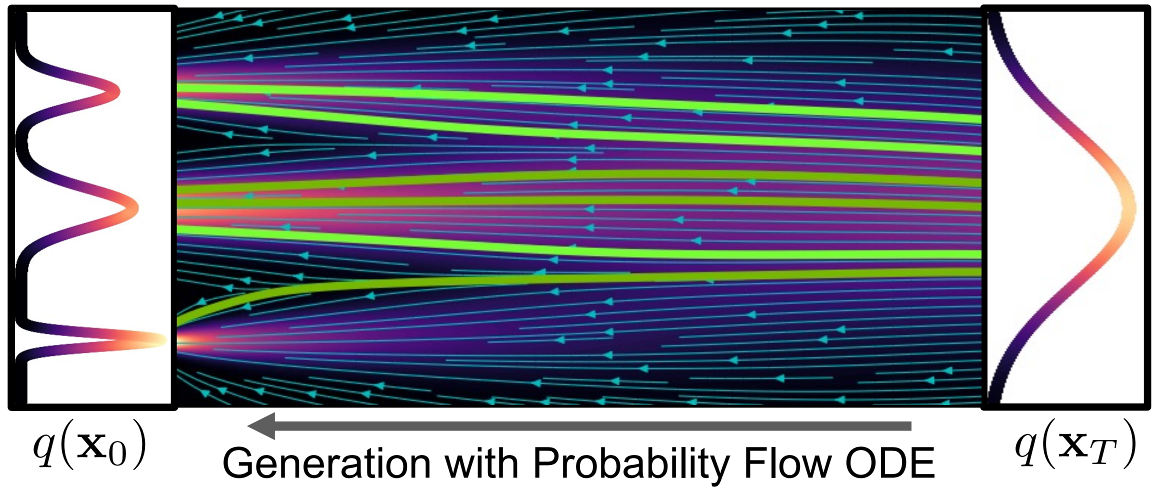
Generative Diffusion SDE:

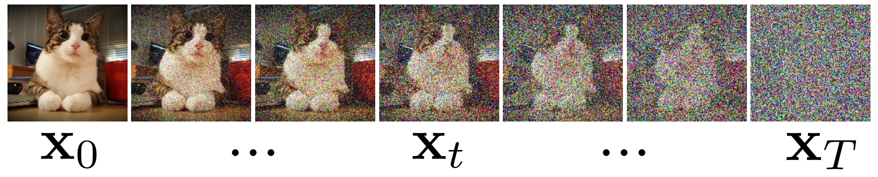
$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$$

➡ *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 Sampling* (Part 1) 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: Higher-Order ODE solvers (*Runge-Kutta, linear multistep methods, exponential integrators, …*)

[Song et al., *ICLR*, 2021](#)

Sampling from "Continuous-Time" Diffusion Models

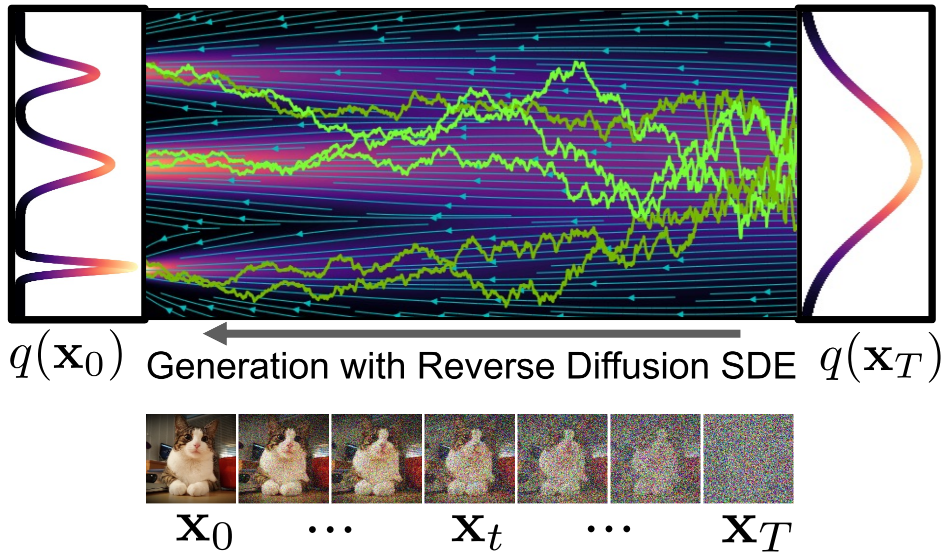
How to solve the generative SDE or ODE in practice?

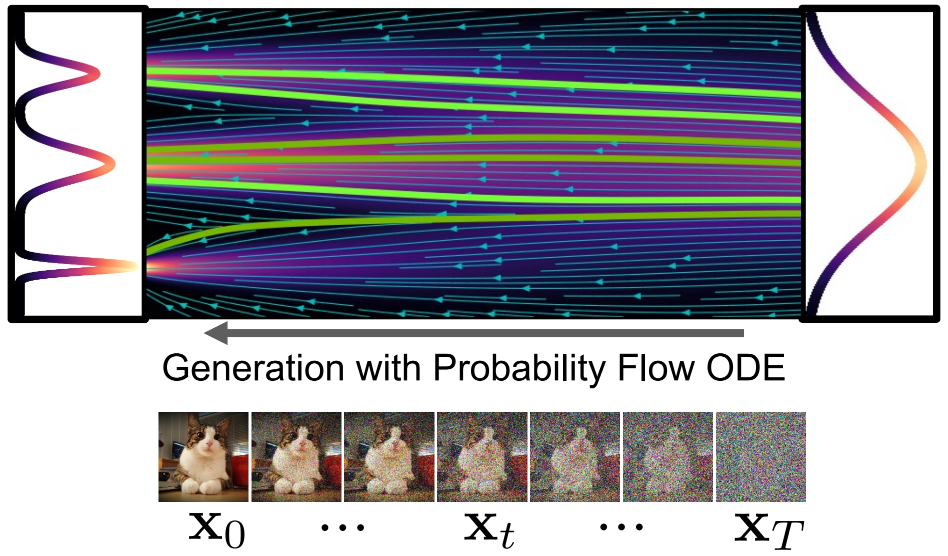
- Runge-Kutta adaptive step-size ODE solver [1]
- Higher-Order adaptive step-size SDE solver [2]
- Reparametrized, smoother ODE [3]
- Higher-Order ODE solver with linear multistepping [4]
- Exponential ODE Integrators [5,6]
- Higher-Order ODE solver with Heun's Method [7]

[1] Song et al., "Score-Based Generative Modeling through Stochastic Differential Equations", *ICLR*, 2021
[2] Jolicoeur-Martineau et al., "Gotta Go Fast When Generating Data with Score-Based Models", *arXiv*, 2021
[3] Song et al., "Denoising Diffusion Implicit Models", *ICLR*, 2021
[4] Liu et al., "Pseudo Numerical Methods for Diffusion Models on Manifolds", *ICLR*, 2022
[5] Zhang and Chen, "Fast Sampling of Diffusion Models with Exponential Integrator", *arXiv*, 2022
[6] Lu et al., "DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps", *arXiv*, 2022
[7] Karras et al., "Elucidating the Design Space of Diffusion-Based Generative Models", *arXiv*, 2022

Sampling from “Continuous-Time” Diffusion Models

SDE vs. ODE Sampling: Pro’s and Con’s





Generative Diffusion SDE:

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

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

- *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.

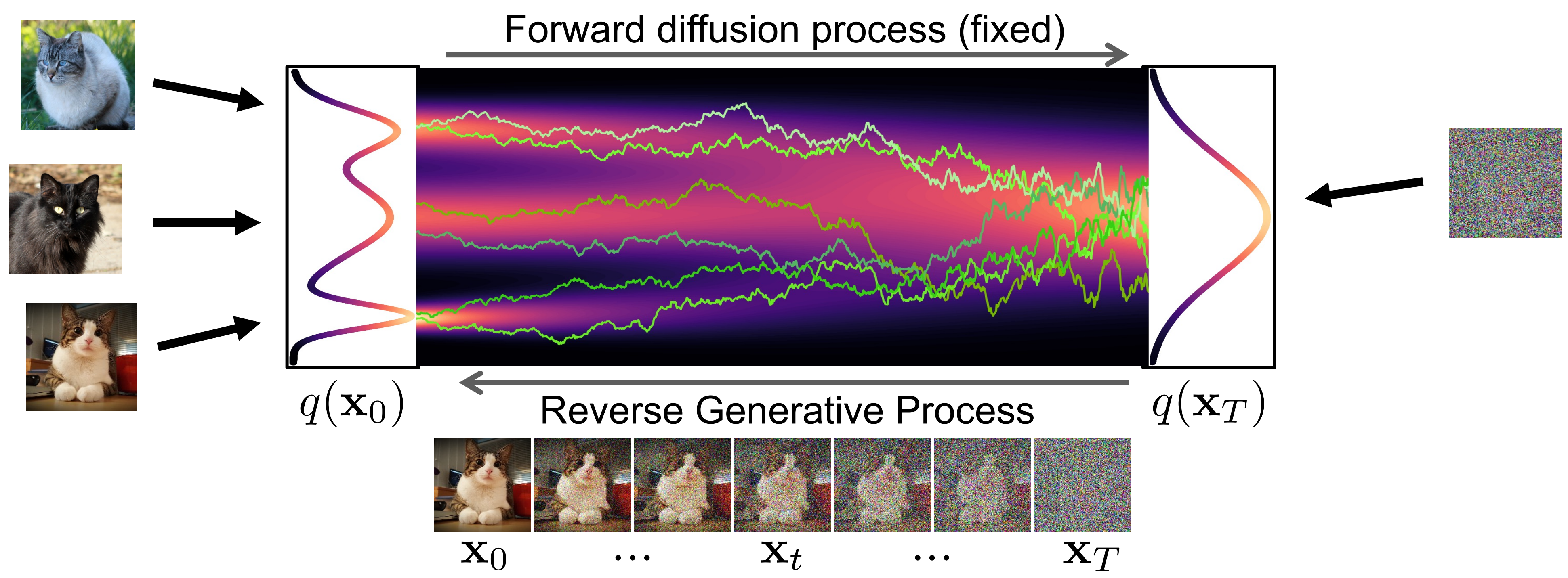
Probability Flow ODE:

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

- *Pro:* 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

Diffusion Models as Energy-based Models



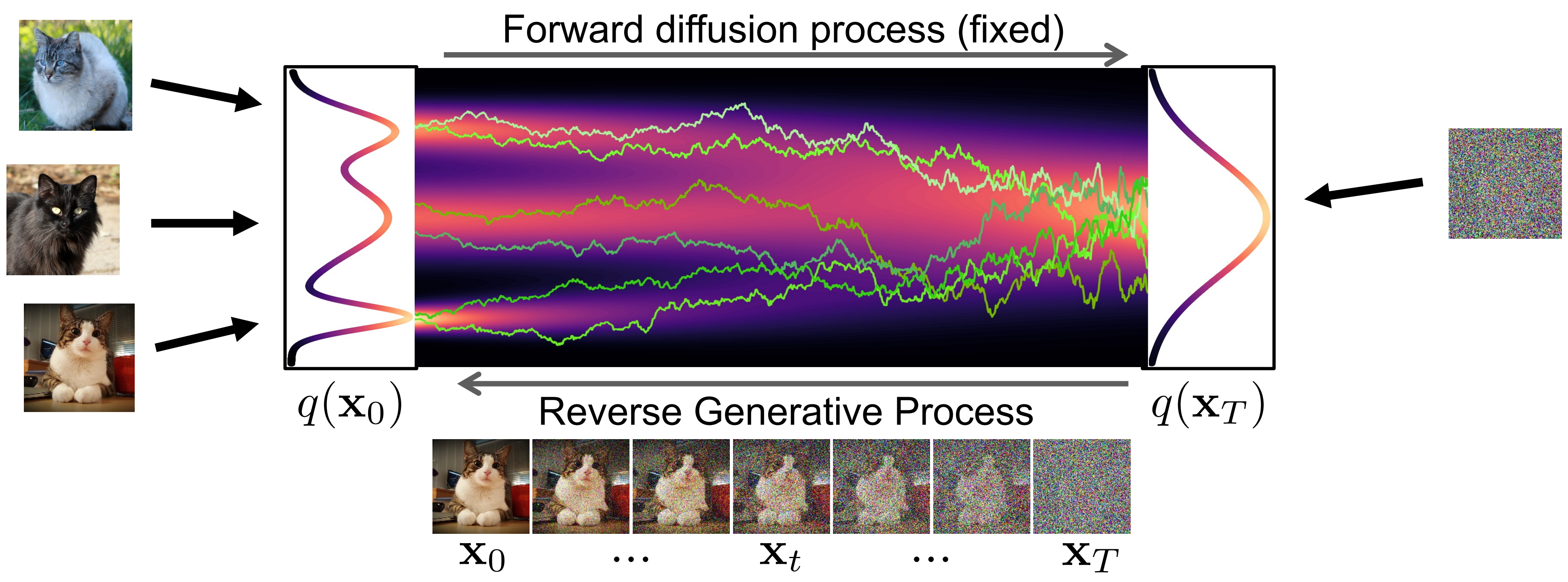
- Assume an Energy-based Model (EBM): $p_{\boldsymbol{\theta}}(\mathbf{x}, t) = \dfrac{e^{-E_{\boldsymbol{\theta}}(\mathbf{x},t)}}{\mathcal{Z}_{\boldsymbol{\theta}}(t)}$
- Sample EBM via Langevin dynamics: $\mathbf{x}_{i+1} = \mathbf{x}_i - \eta \nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x}_i, t) + \sqrt{2\eta}\, \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Requires only gradient of energy $-\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x}, t)$, not $E_{\boldsymbol{\theta}}(\mathbf{x}, t)$ itself, nor $\mathcal{Z}_{\boldsymbol{\theta}}(t)$!

In diffusion models, we learn “energy gradients” for all diffused distributions directly:

$$\nabla_{\mathbf{x}} \log q_t(\mathbf{x}) \approx \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) =: \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, t) = -\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x}, t) - \underbrace{\nabla_{\mathbf{x}} \log \mathcal{Z}_{\boldsymbol{\theta}}(t)}_{=0} = -\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x}, t)$$

➡ Diffusion Models model energy gradient directly, along entire diffusion process, and avoid modeling partition function. Different noise levels along diffusion are analogous to annealed sampling in EBMs.

Unique Identifiability



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:

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

- Denoising model $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ and deterministic data encodings uniquely determined by **data** and **fixed forward diffusion**!
- Even with different architectures and initializations, we recover identical model outputs and encodings (given sufficient training data, model capacity and optimization accuracy), in contrast to GANs, VAEs, etc.

Song et al., *ICLR*, 2021

Unique Identifiability

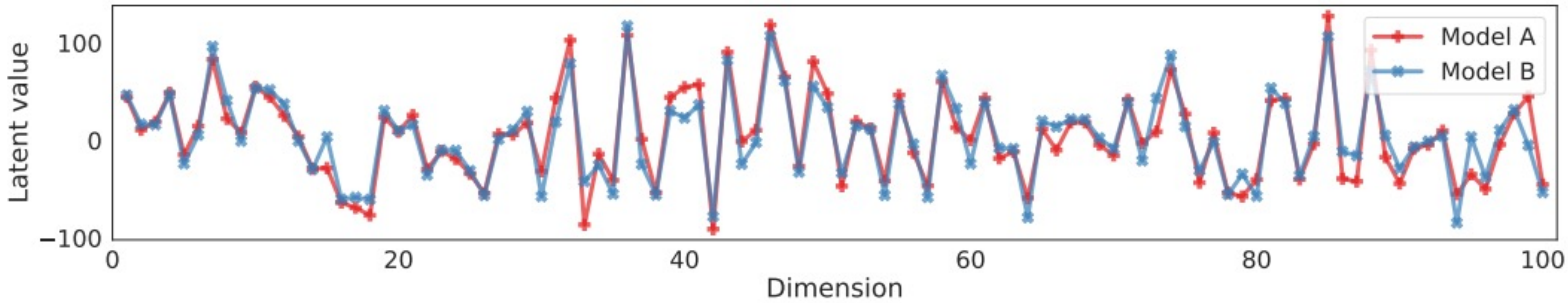
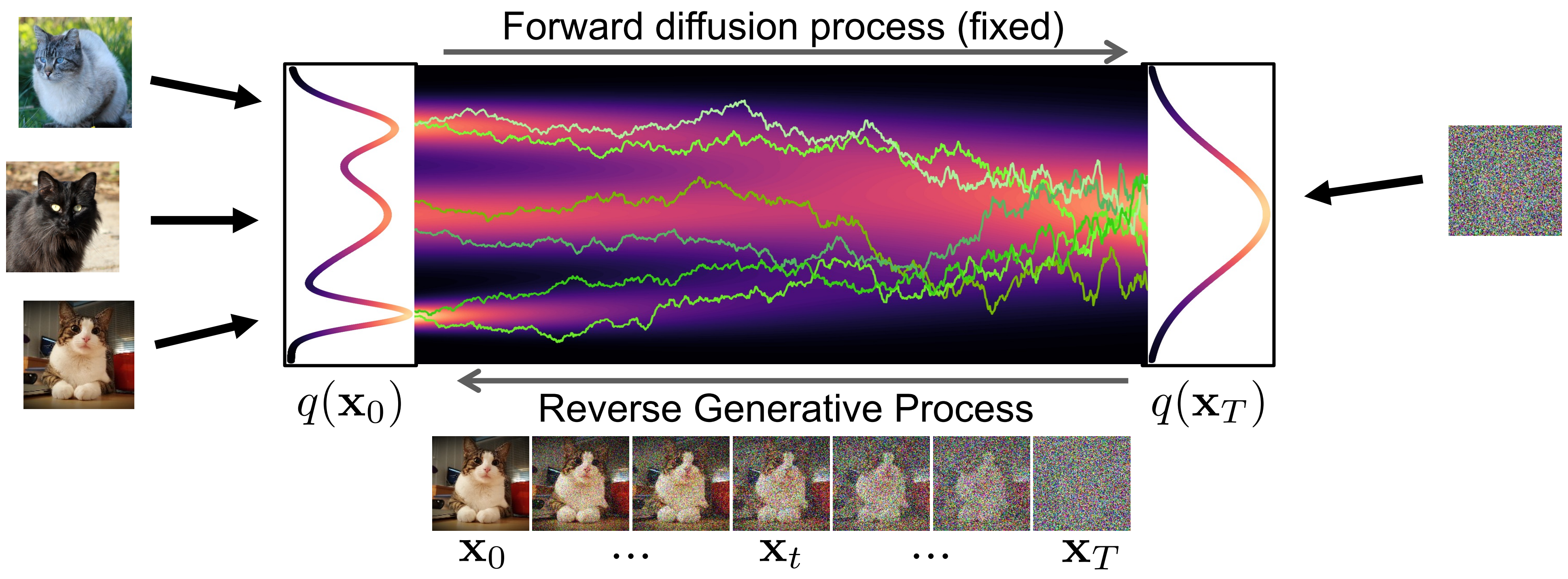


Figure 7: Comparing the first 100 dimensions of the latent code obtained for a random CIFAR-10 image. "Model A" and "Model B" are separately trained with different architectures.

(image from: Song et al., "Score-Based Generative Modeling through Stochastic Differential Equations", *ICLR*, 2021)

Song et al., *ICLR*, 2021

Why use Differential Equation Framework?



Advantages of the Differential Equation framework for Diffusion Models:

- Can leverage broad existing literature on **advanced and fast SDE and ODE solvers**
- Allows us to construct deterministic **Probability Flow ODE**
    - Deterministic Data Encodings
    - Log-likelihood Estimation
- **Clean mathematical framework** based on Diffusion Processes and Score Matching; connections to Neural ODEs, Continuous Normalizing Flows and Energy-based Models

Today's Program

Title	Speaker	Time
Introduction	Arash	10 min
Part (1): Denoising Diffusion Probabilistic Models	Arash	35 min
Part (2): Score-based Generative Modeling with Differential Equations	Karsten	45 min
Part (3): Advanced Techniques: Accelerated Sampling, Conditional Generation, and Beyond	Ruiqi	45 min
Applications (1): Image Synthesis, Text-to-Image, Controllable Generation	Ruiqi	15 min
Applications (2): Image Editing, Image-to-Image, Super-resolution, Segmentation	Arash	15 min
Applications (3): Video Synthesis, Medical Imaging, 3D Generation, Discrete State Models	Karsten	15 min
Conclusions, Open Problems and Final Remarks	Arash	10 min

cvpr2022-tutorial-diffusion-models.github.io

Part (3):

Advanced Techniques: Accelerated Sampling, Conditional Generation, and Beyond

Outline

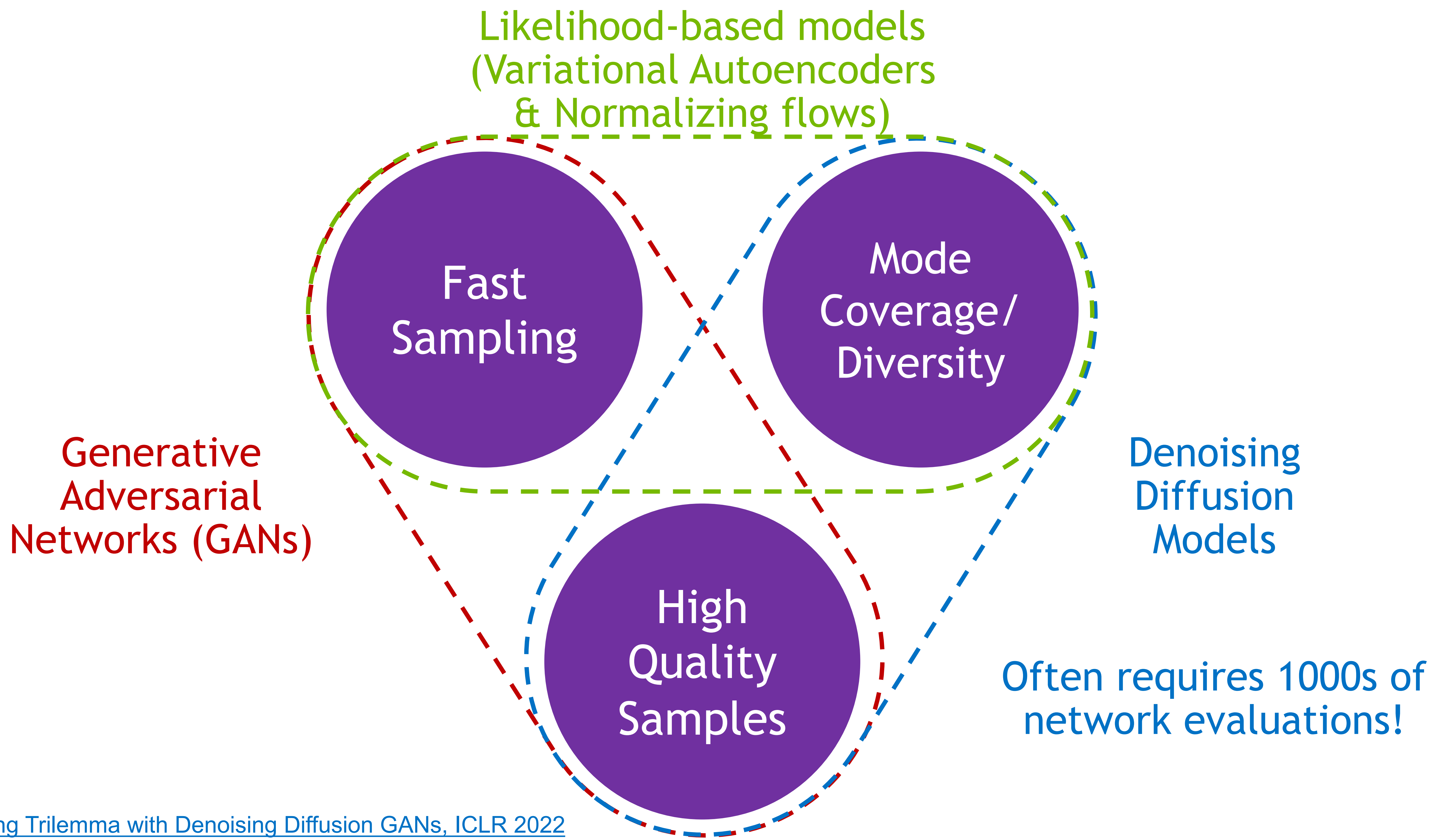
Questions to address with advanced techniques

- Q1: How to accelerate the sampling process?
  - Advanced forward diffusion process
  - Advanced reverse process
  - Hybrid models & model distillation
- Q2: How to do high-resolution (conditional) generation?
  - Conditional diffusion models
  - Classifier(-free) guidance
  - Cascaded generation

Part (3)-1:
Q: How to accelerate sampling process?

What makes a good generative model?

The generative learning trilemma

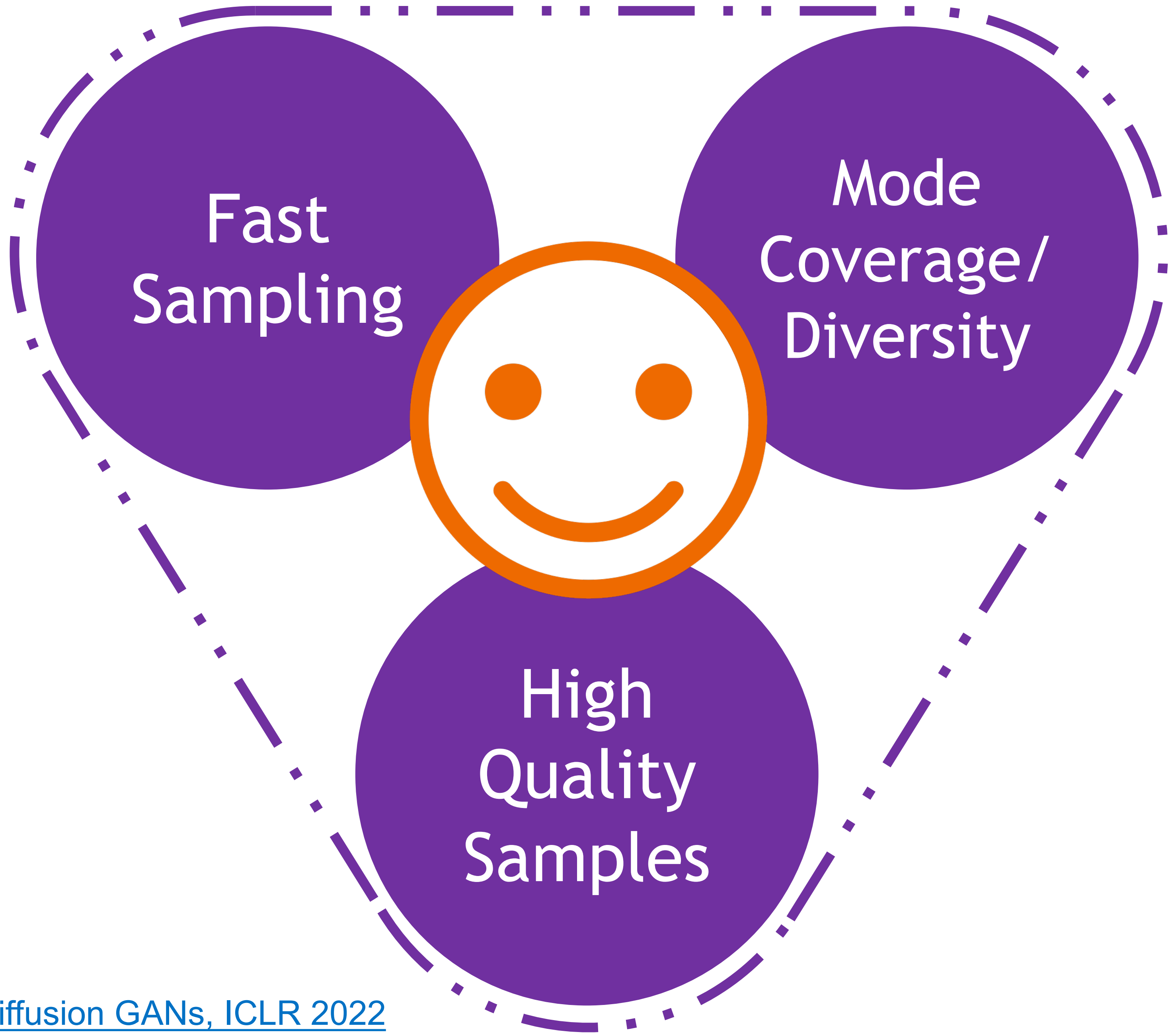


[Tackling the Generative Learning Trilemma with Denoising Diffusion GANs, ICLR 2022]

What makes a good generative model?

The generative learning trilemma

Tackle the trilemma by accelerating diffusion models



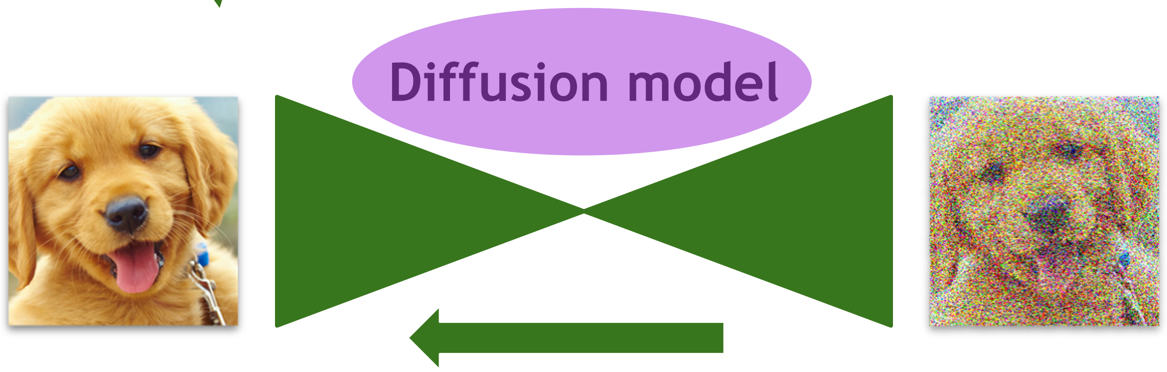
Tackling the Generative Learning Trilemma with Denoising Diffusion GANs, ICLR 2022

How to accelerate diffusion models?

[Image credit: Ben Poole, Mohammad Norouzi]

Simple **forward process** slowly maps data to noise

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



- 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?

(1/3) Advanced forward process

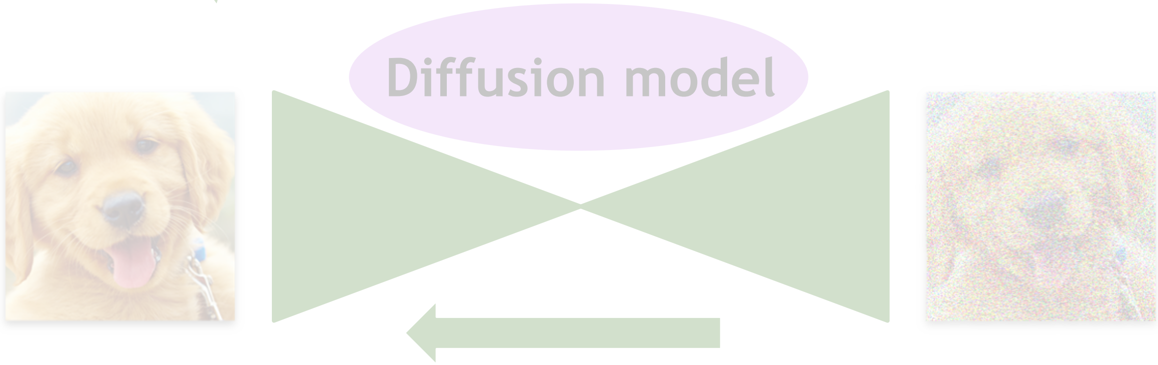
The reverse process will be changed accordingly

Simple **forward process** slowly maps data to noise

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

$$\mathbf{x}_0 \rightarrow \ldots \rightarrow \mathbf{x}_t \rightarrow \mathbf{x}_{t+1} \rightarrow \ldots \rightarrow \mathbf{x}_T$$

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



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

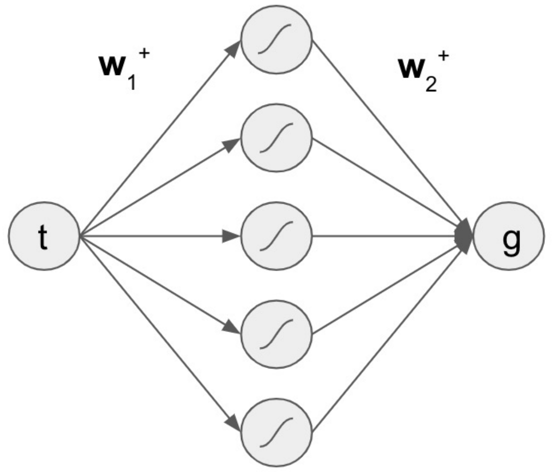
Variational diffusion models

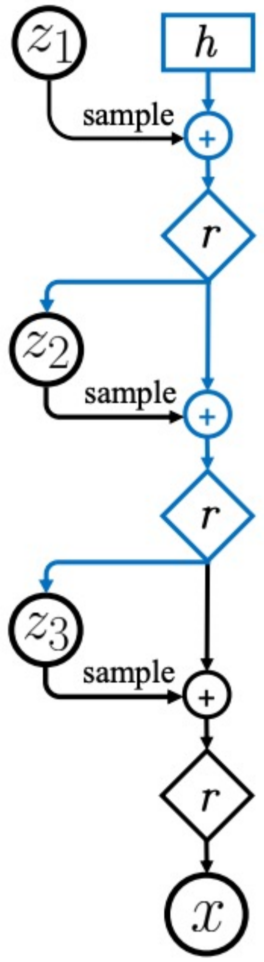
Learnable diffusion process

- Given the forward process $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$
- Directly parametrize the variance through a learned function $\gamma_{\boldsymbol{\eta}}$:

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

- $\gamma_{\boldsymbol{\eta}}(t)$: a monotonic MLP.
    - Strictly positive weights & monotonic activations (e.g. sigmoid)
- Analogous to hierarchical VAE (part 1): unlike diffusion models using a fixed encoder, we include learnable parameters in the encoder.





Kingma et al., "Variational diffusion models", NeurIPS 2021.
Vahdat and Kautz, NVAE: A Deep Hierarchical Variational Autoencoder, NeurIPS 2020

Variational diffusion models

New parametrization of training objectives

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

$$\mathcal{L}_T = \frac{T}{2}\mathbb{E}_{\mathbf{x}_0,\epsilon,t}\left[(\exp(\gamma_\eta(t) - \gamma_\eta(t-1)) - 1)||\epsilon - \epsilon_\theta(\mathbf{x}_t, t)||_2^2\right]$$

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

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

$$\mathcal{L}_\infty = \frac{1}{2}\mathbb{E}_{\mathbf{x}_0,\epsilon,t}\left[\gamma'_\eta(t)||\epsilon - \epsilon_\theta(\mathbf{x}_t, t)||_2^2\right], \quad \gamma'_\eta(t) = \mathrm{d}\gamma_\eta(t)/\mathrm{d}t$$

- it is shown to be only related to the signal-to-noise ratio $\mathrm{SNR}(t) = \bar{\alpha}_t/(1-\bar{\alpha}_t) = \exp(-\gamma_\eta(t))$ 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.

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

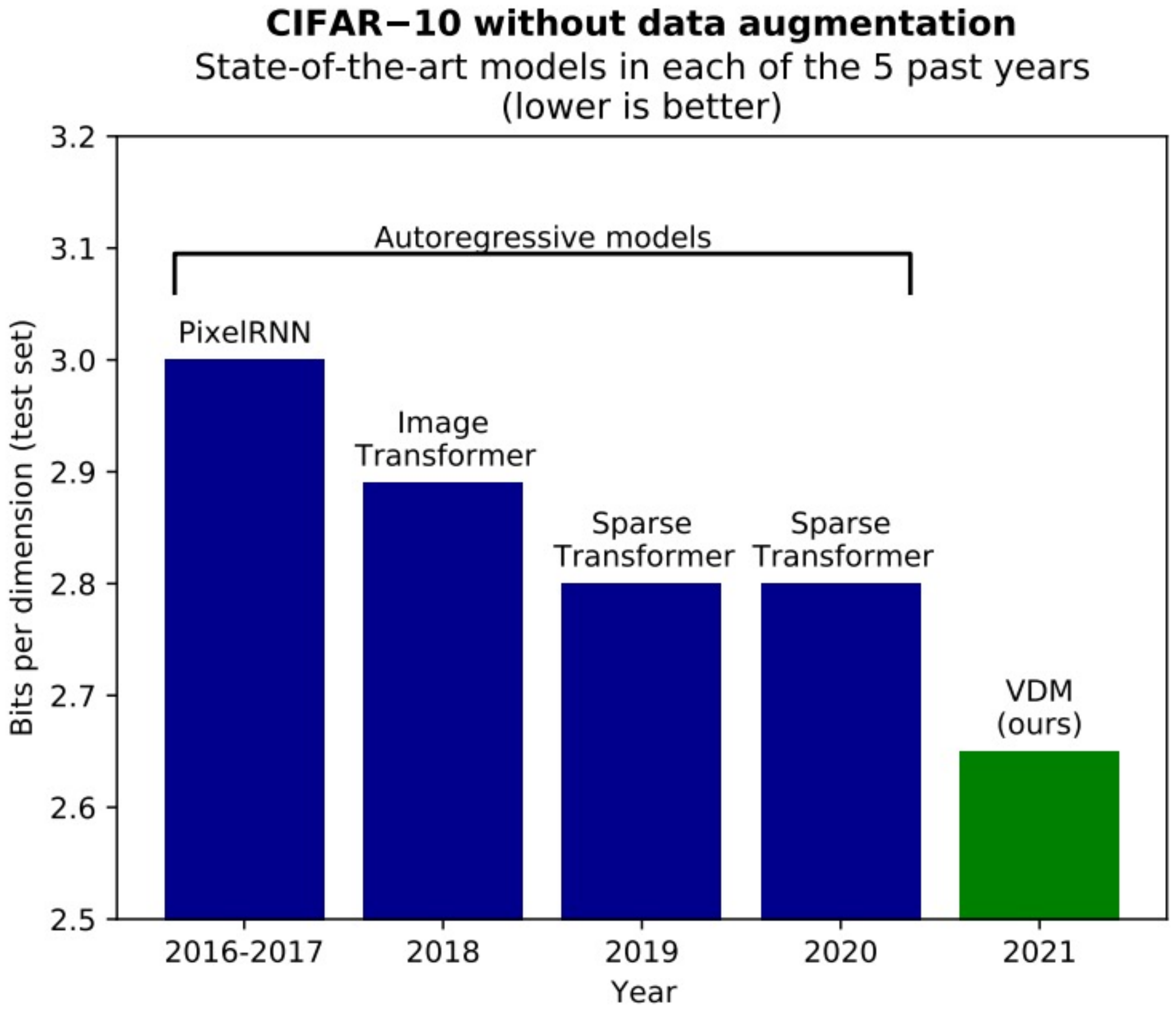
Variational diffusion models

SOTA likelihood estimation

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

$$f^n_{i,j,k} = \sin(x_{i,j,k} 2^n \pi), g^n_{i,j,k} = \cos(x_{i,j,k} 2^n \pi), n = 7, 8.$$

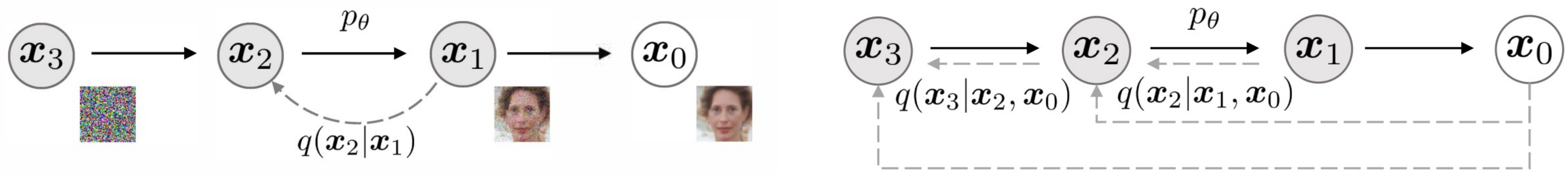
- Good likelihoods require modeling all bits, even the ones corresponding to very small changes in input.
- But: neural nets are usually bad at modeling small changes to inputs.
- Significant improvements in log-likelihoods.



Kingma et al., "Variational diffusion models", NeurIPS 2021.

Denoising diffusion implicit models (DDIM)

Non-Markovian diffusion process



Main Idea

Design a family of non-Markovian diffusion processes and corresponding reverse processes.

The process is designed such that the model can be optimized by the same surrogate objective as the original diffusion model.

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

Therefore, can take a pretrained diffusion model but with more choices of sampling procedure.

Song et al., "Denoising Diffusion Implicit Models", ICLR 2021.

Denoising diffusion implicit models (DDIM)

How to define the non-Markovian forward process?

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

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

Recall that the KL divergence in the variational upper bound can be written as:

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

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

If we assume loss weighting λ_t can be arbitrary values, the above formulation holds as long as

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

Forward process: $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\sigma}_t^2\mathbf{I}), \quad \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = a\mathbf{x}_t + b\epsilon = a\mathbf{x}_t + b\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}$

Reverse process: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \tilde{\sigma}_t^2\mathbf{I}), \quad \mu_\theta(\mathbf{x}_t, t) = a\mathbf{x}_t + b\epsilon_\theta(\mathbf{x}_t, t) = a\mathbf{x}_t + b\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\hat{\mathbf{x}}_0}{\sqrt{1-\bar{\alpha}_t}}$

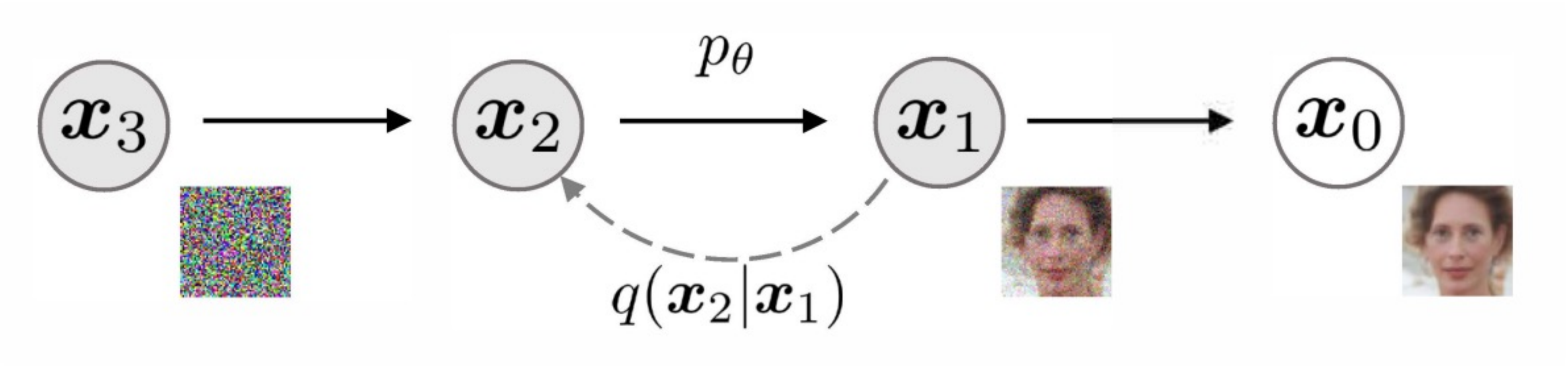
(assume $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\ \hat{\mathbf{x}}_0 + \sqrt{(1-\bar{\alpha}_t)}\ \epsilon_\theta(\mathbf{x}_t, t)$)

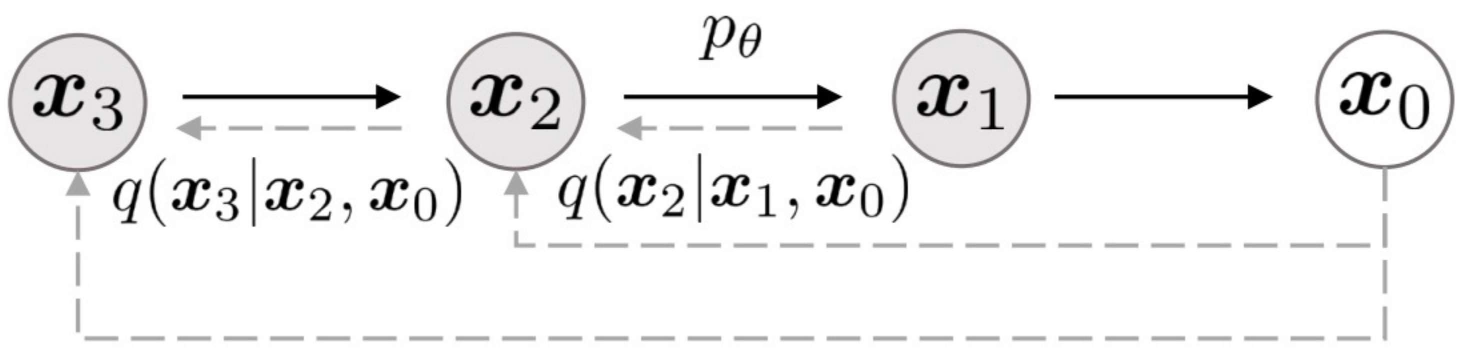
No need to specify $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ to be a Markovian process!

Song et al., "Denoising Diffusion Implicit Models", ICLR 2021.

Denoising diffusion implicit models (DDIM)

Non-Markovian diffusion process





For the forward process $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\sigma}_t \mathbf{I})$, $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = a\mathbf{x}_t + b\epsilon = a\mathbf{x}_t + b\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}$, need to choose a, b such that $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$

Define a family of forward processes that meets the above requirement:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\tilde{\sigma}_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}, \tilde{\sigma}_t^2\mathbf{I}\right)$$

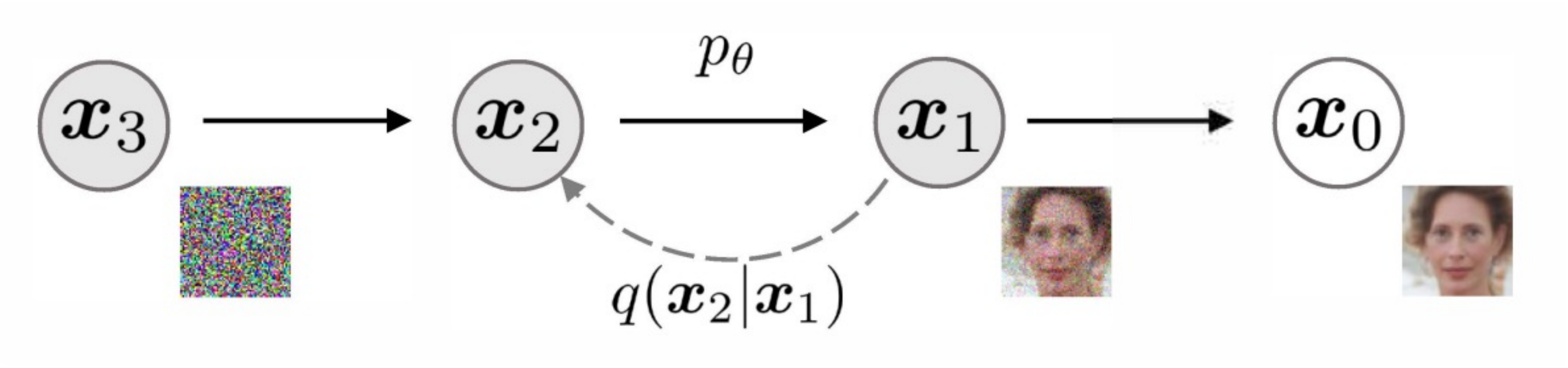
The corresponding reverse process is

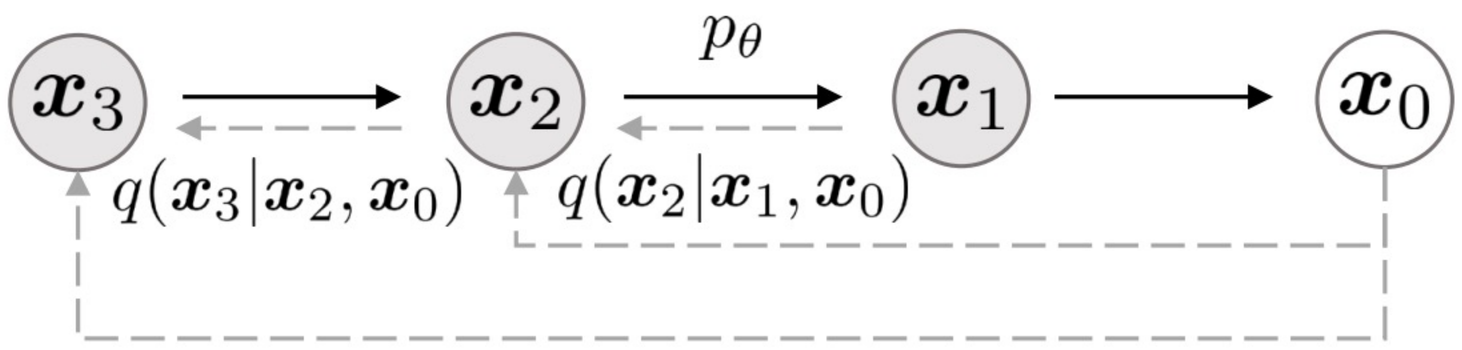
$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\hat{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\tilde{\sigma}_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\hat{\mathbf{x}}_0}{\sqrt{1-\bar{\alpha}_t}}, \tilde{\sigma}_t^2\mathbf{I}\right)$$

Song et al., "Denoising Diffusion Implicit Models", ICLR 2021.

DDIM sampler

Deterministic generative process





$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\hat{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\tilde{\sigma}_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\hat{\mathbf{x}}_0}{\sqrt{1-\bar{\alpha}_t}}, \tilde{\sigma}_t^2\mathbf{I}\right)$$

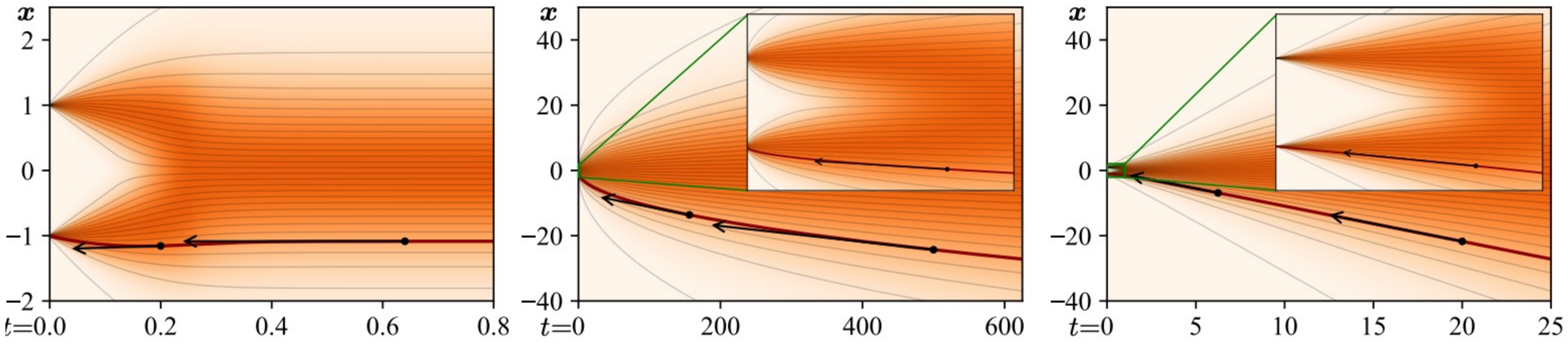
- DDIM sampler – $\tilde{\sigma}_t^2 = 0, \forall t$

  - a deterministic generative process, with randomness from only t=T.

Song et al., "Denoising Diffusion Implicit Models", ICLR 2021.

ODE interpretation

Deterministic generative process



- DDIM sampler can be considered as an integration rule of the following ODE:

$$\mathrm{d}\bar{\mathbf{x}}(t) = \epsilon_\theta^{(t)}\left(\frac{\bar{\mathbf{x}}(t)}{\sqrt{\eta^2+1}}\right)\mathrm{d}\eta(t); \quad \bar{\mathbf{x}} = \mathbf{x}/\sqrt{\bar{\alpha}}, \eta = \sqrt{1-\bar{\alpha}}/\sqrt{\bar{\alpha}}$$

- With the optimal model, the ODE is equivalent to a probability flow ODE of a "variance-exploding" SDE:

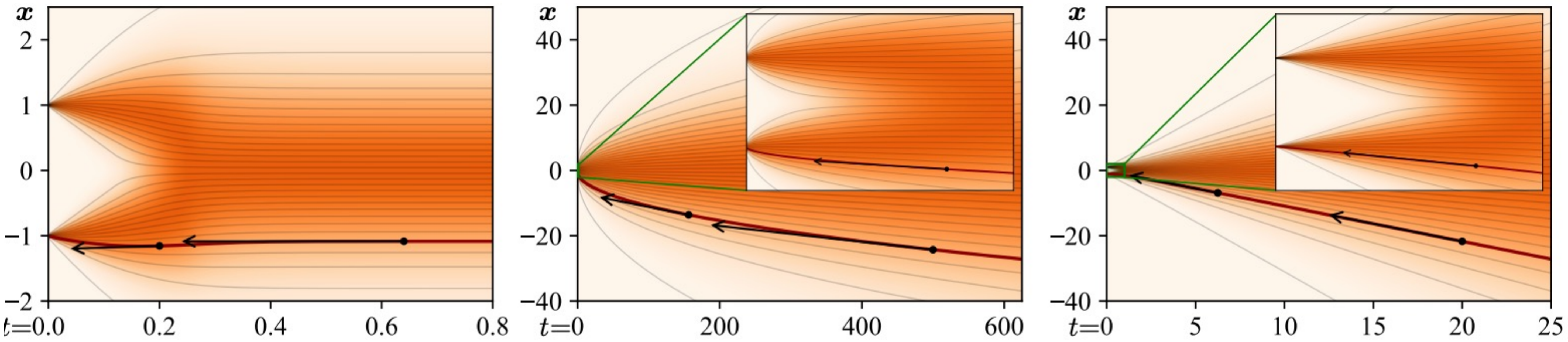
$$\mathrm{d}\bar{\mathbf{x}} = -\frac{1}{2}g(t)^2\nabla_{\bar{\mathbf{x}}}\log p_t(\bar{\mathbf{x}})\mathrm{d}t, \quad g(t) = \sqrt{\frac{\mathrm{d}\eta^2(t)}{\mathrm{d}t}}$$

- Sampling procedure can be different from standard Euler's method: wrt. $\mathrm{d}\eta(t)$ vs wrt. $\mathrm{d}t$

Song et al., "Denoising Diffusion Implicit Models", ICLR 2021.
Karras et al., "Elucidating the Design Space of Diffusion-Based Generative Models", arXiv 2022.
Salimans & Ho, "Progressive distillation for fast sampling of diffusion models", ICLR 2022.

DDIM sampler

Faster & low curvature



(a) Variance preserving ODE [42] (b) Variance exploding ODE [42] (c) DDIM [40] / Our ODE

(Karras et al.) argues that the ODE of DDIM is favored, as the tangent of the solution trajectory always points towards the denoiser output.

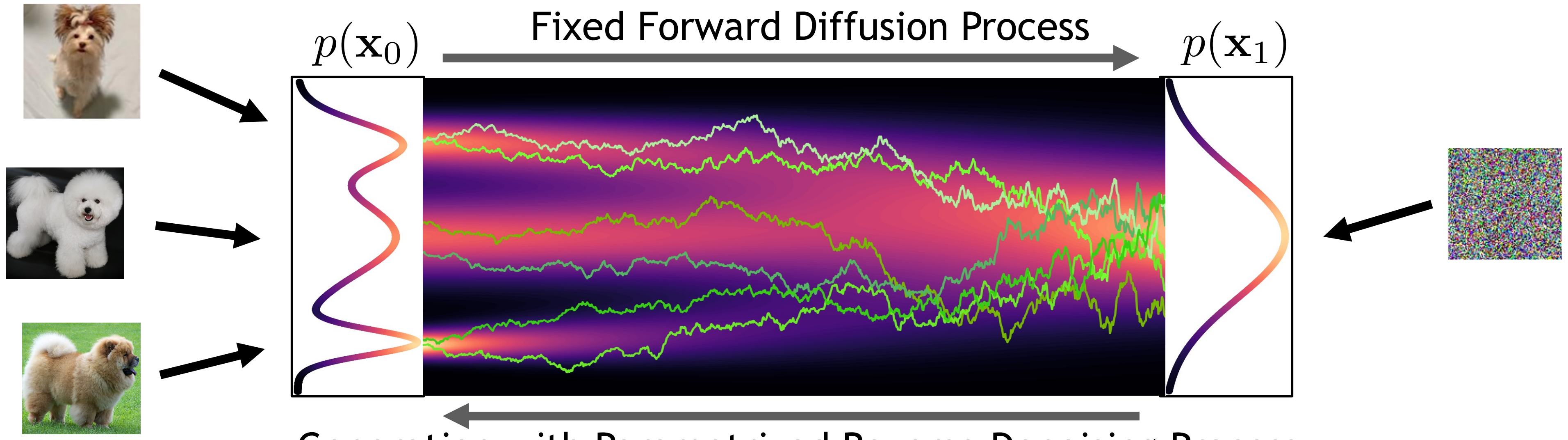
Leads to largely linear solution trajectories with low curvature.

Low curvature means less truncation errors accumulated over the trajectories.

Song et al., "Denoising Diffusion Implicit Models", ICLR 2021.
Karras et al., "Elucidating the Design Space of Diffusion-Based Generative Models", arXiv 2022.
Salimans & Ho, "Progressive distillation for fast sampling of diffusion models", ICLR 2022.

Critically-damped Langevin diffusion

"fast mixing" diffusion process



- Regular forward Diffusion Process:

$$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t dt + \sqrt{\beta(t)}d\mathbf{w}_t$$

- It is a special case of (overdamped) Langevin dynamics:

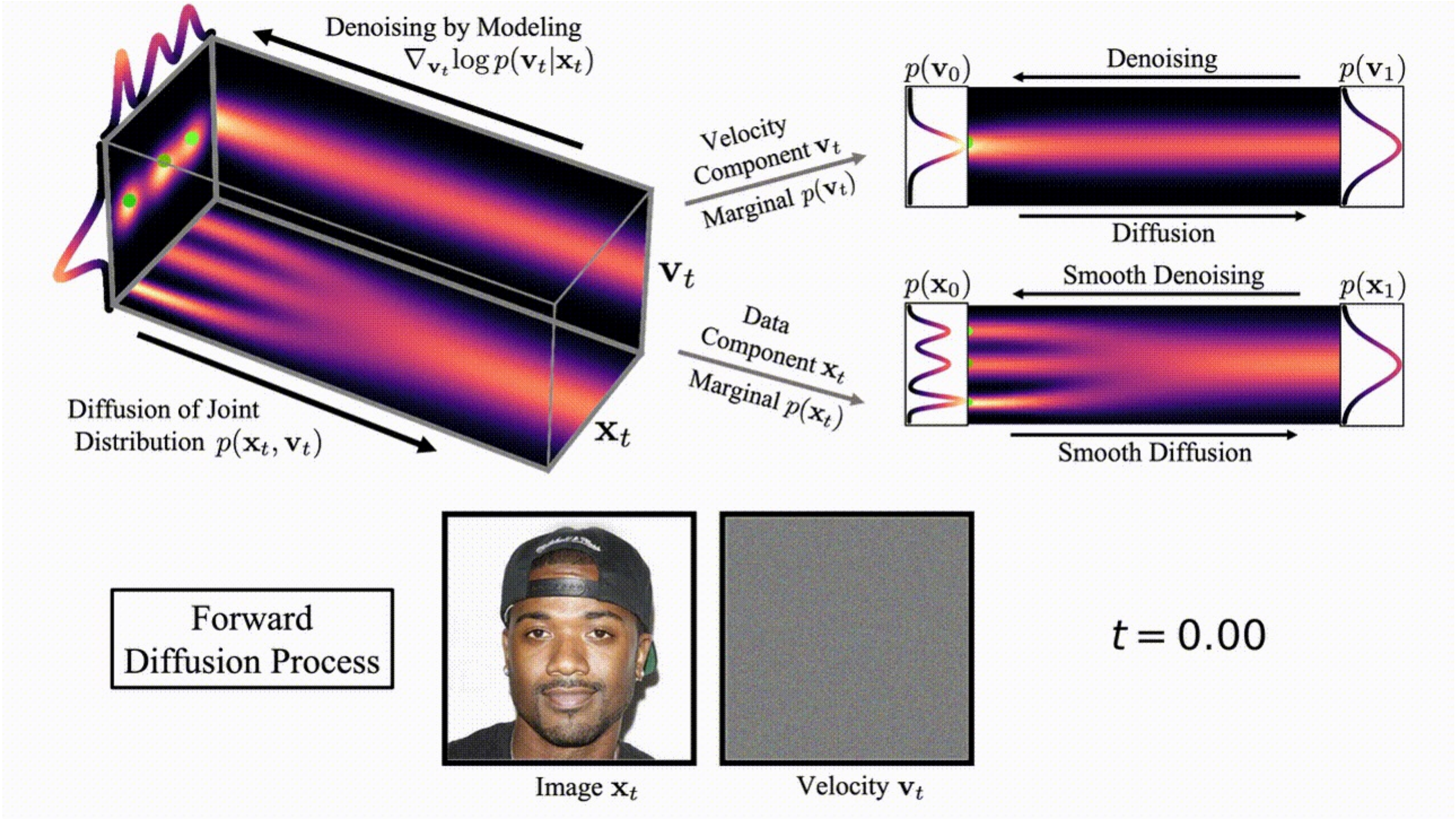
$$d\mathbf{x}_t = \frac{1}{2}\beta(t)\nabla_{\mathbf{x}_t}\log p_{\mathrm{EQ}}(\mathbf{x}_t)dt + \sqrt{\beta(t)}d\mathbf{w}_t$$

$$p_{\mathrm{EQ}}(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; \mathbf{0}, \mathbf{I}) \sim e^{-\frac{1}{2}\mathbf{x}_t^2}$$

Dockhorn et al., "Score-Based Generative Modeling with Critically-Damped Langevin Diffusion", ICLR 2022.

"Momentum-based" diffusion

Introduce a velocity variable and run diffusion in extended space

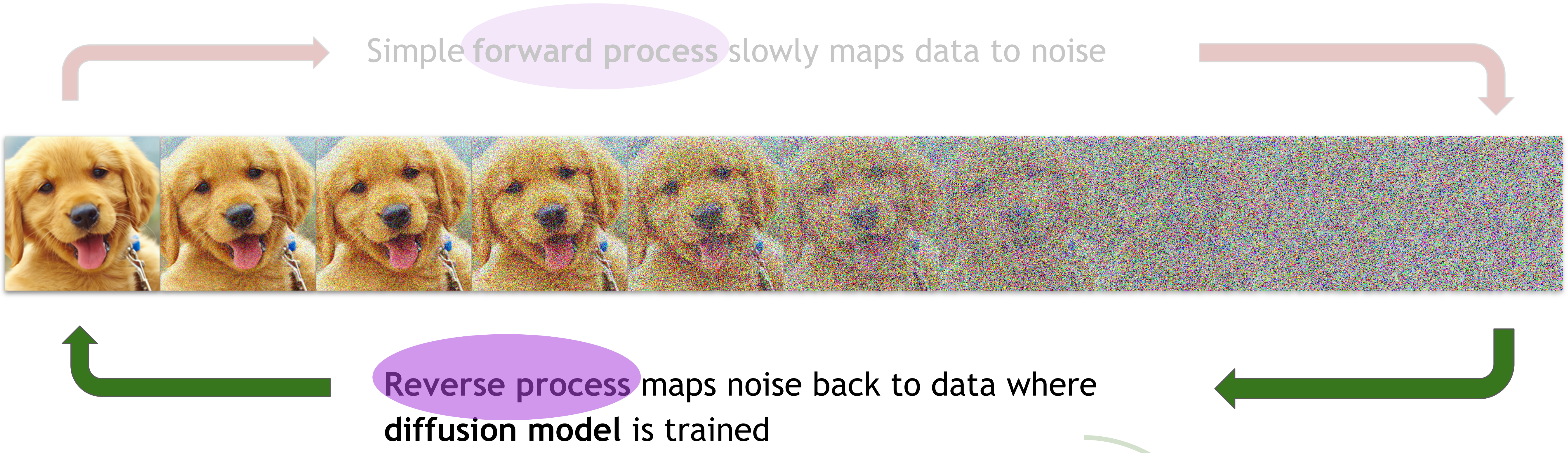


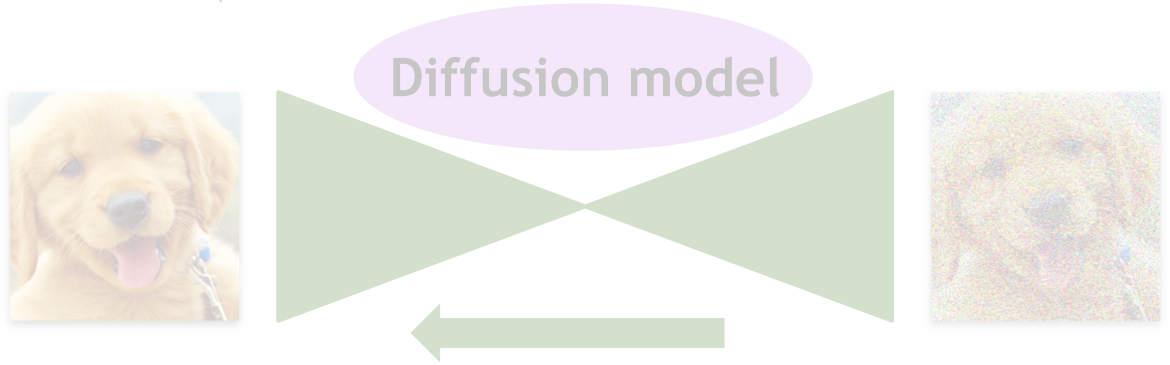
Main idea: Inject noise only into $\mathbf{v}_t$, faster mixing through Hamiltonian component!

Dockhorn et al., "Score-Based Generative Modeling with Critically-Damped Langevin Diffusion", ICLR 2022.

Advanced reverse process

Approximate reverse process with more complicated distributions

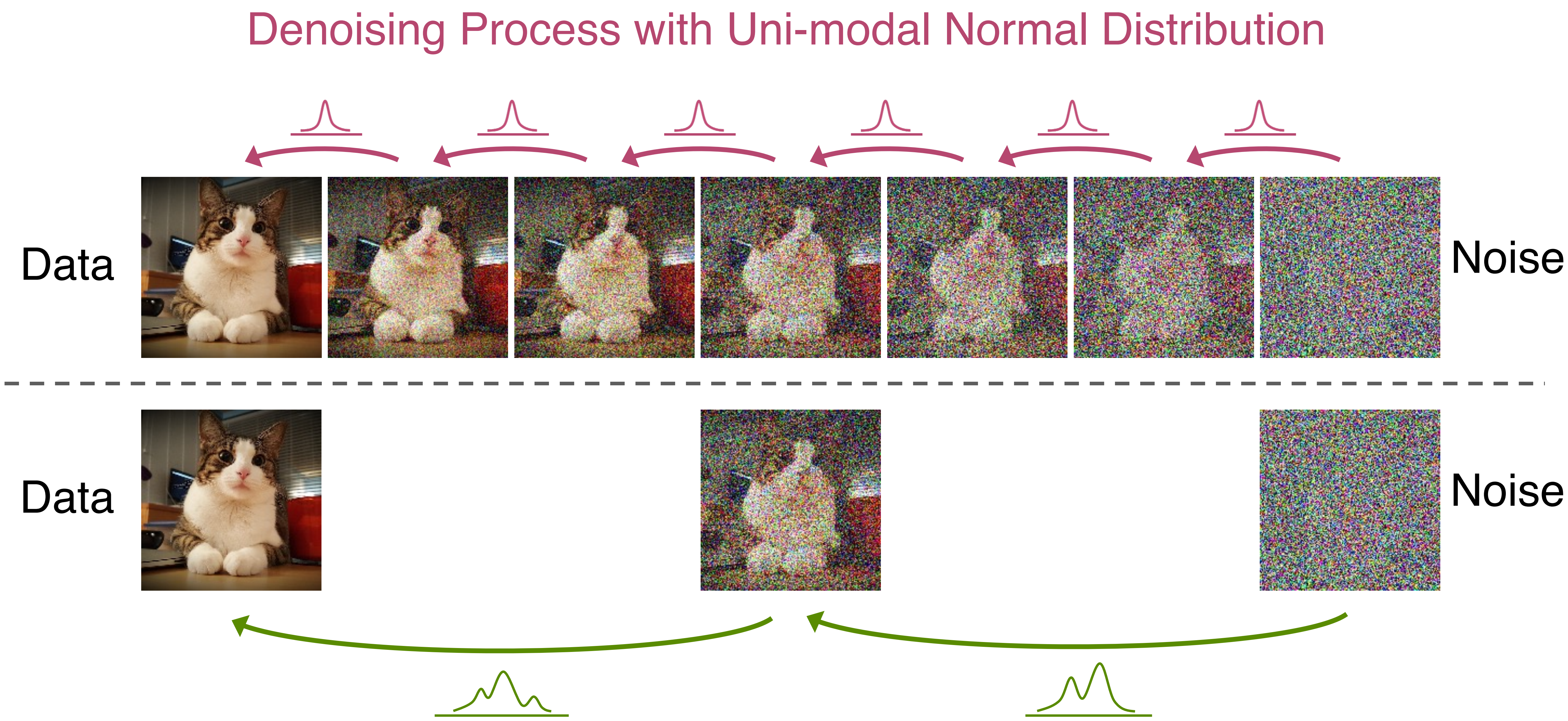




- Q: is normal approximation of the reverse process still accurate when there're less diffusion time steps?

Advanced approximation of reverse process

Normal assumption in denoising distribution holds only for small step



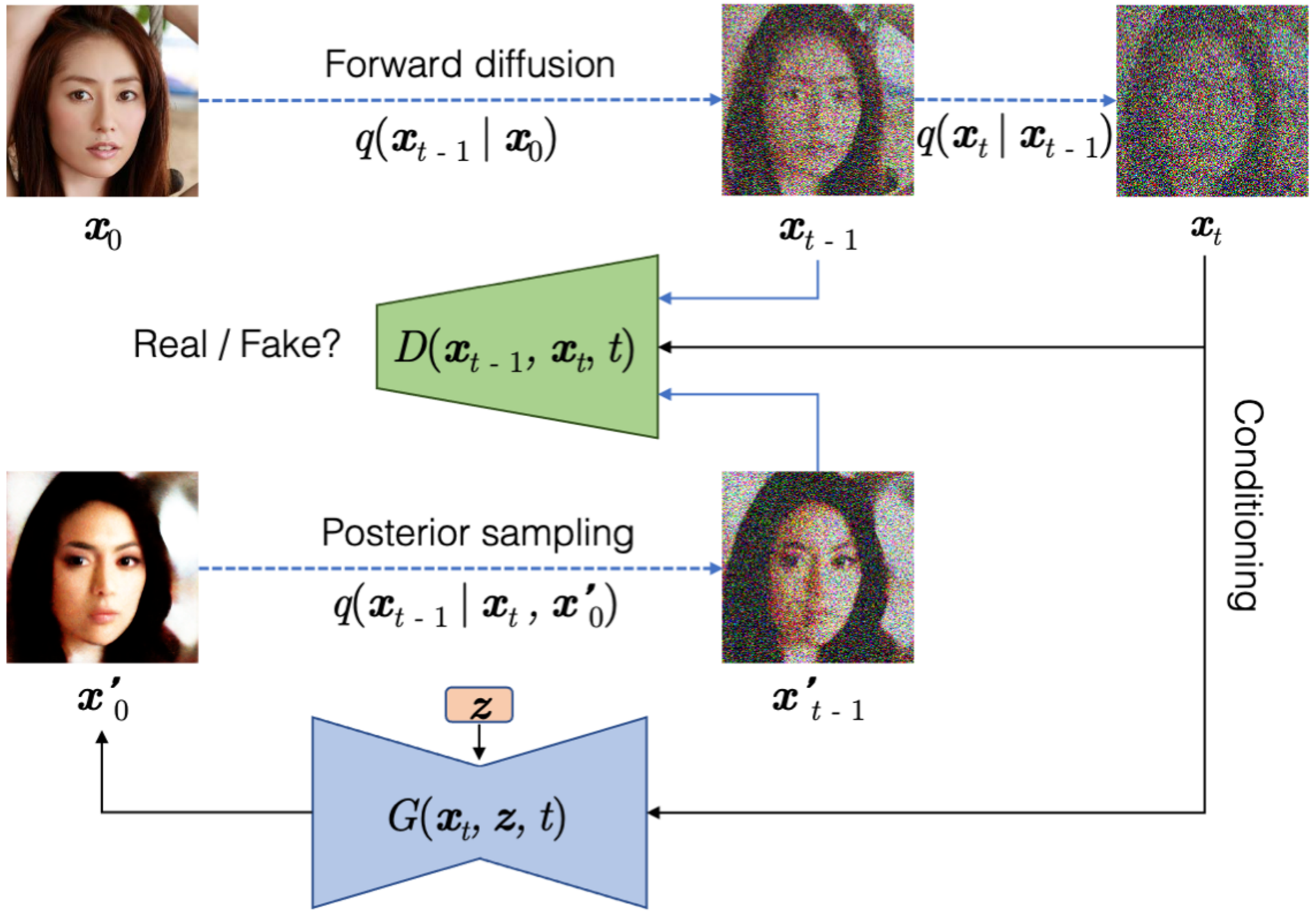
Requires more complicated functional approximators!

[Xiao et al., “Tackling the Generative Learning Trilemma with Denoising Diffusion GANs”, ICLR 2022.](#)
[Gao et al., “Learning energy-based models by diffusion recovery likelihood”, ICLR 2021.](#)

Denoising diffusion GANs

Approximating reverse process by conditional GANs

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



Compared to a one-shot GAN generator:

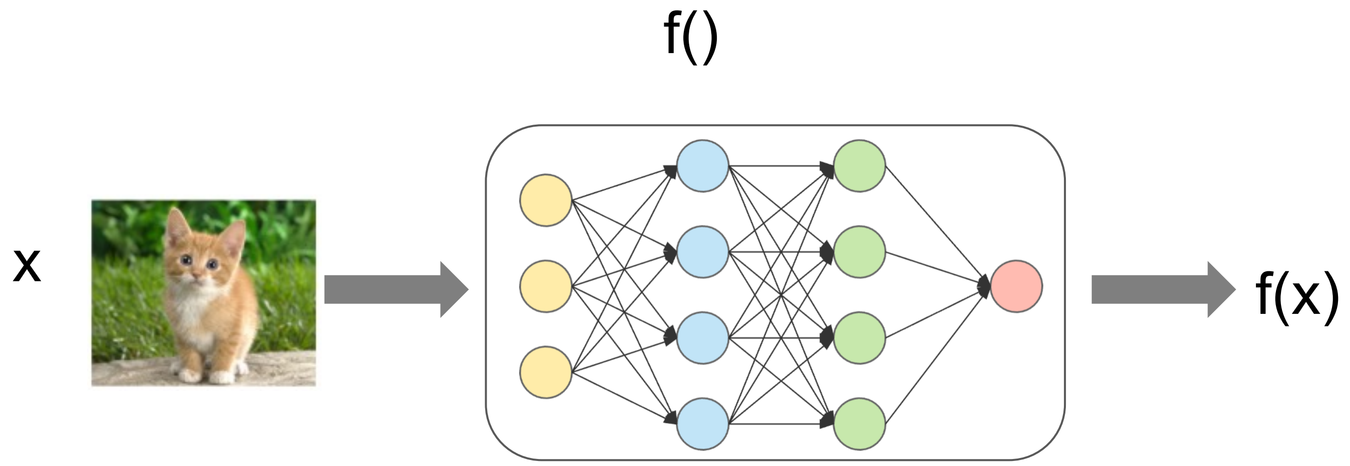
- Both generator and discriminator are solving a much simpler problem.
- Stronger mode coverage
- Better training stability

Diffusion energy-based models

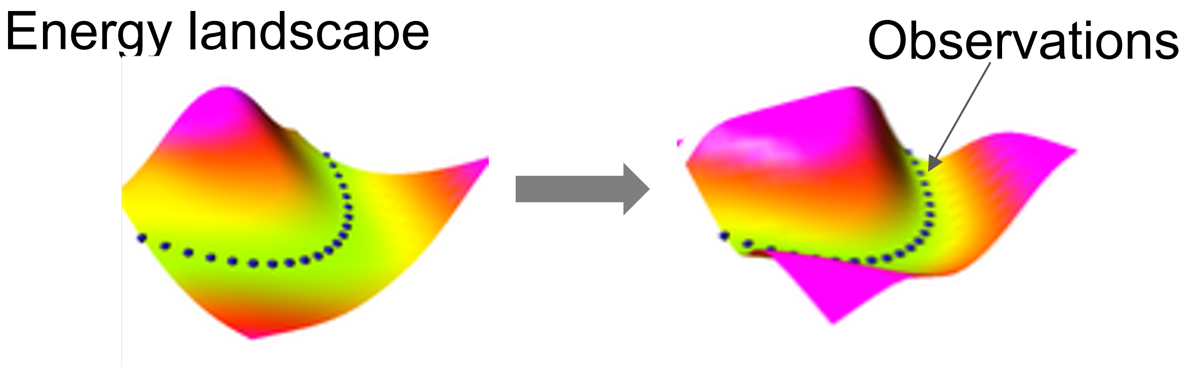
Approximating reverse process by conditional energy-based models

- An energy-based model (EBM) is in the form

$$p_\theta(\mathbf{x}) = \frac{1}{Z_\theta}\exp(f_\theta(\mathbf{x})) = \frac{1}{\underbrace{Z_\theta}_{\text{Partition function, Analytically intractble}}}\exp(-\underbrace{E_\theta(x)}_{\text{Energy function}})$$



Parametrization



Learning energy-based models

- Optimizing energy-based models requires MCMC from the current model $p_\theta(\mathbf{x})$

$$\nabla_\theta \log p_\theta(\mathbf{x}) = \nabla_\theta f(\mathbf{x}) - \mathbb{E}_{p_\theta(x')}[\nabla_\theta f(x')]$$

Gao et al., “Learning energy-based models by diffusion recovery likelihood”, ICLR 2021.

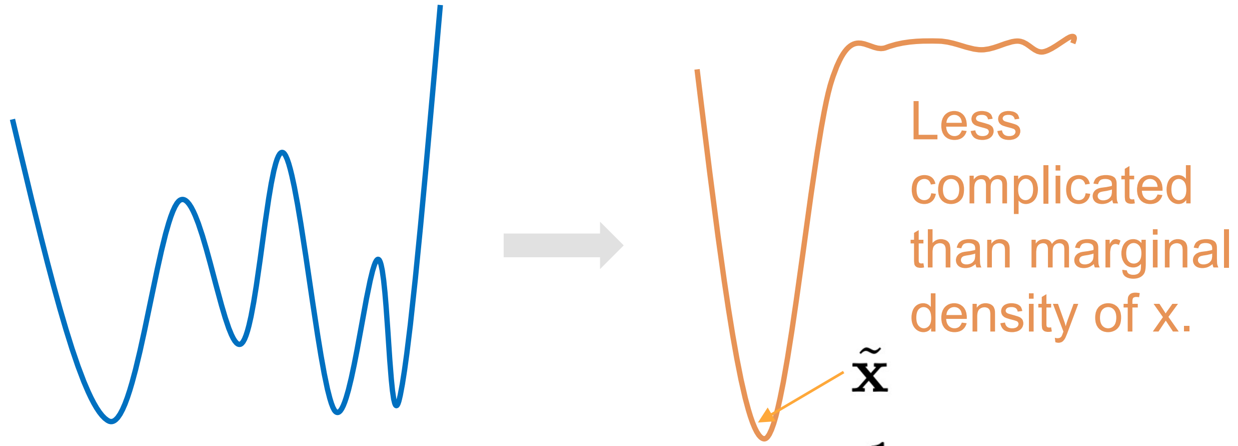
Diffusion energy-based models

Conditional energy-based models

- Assume at each diffusion step marginally $p_\theta(\mathbf{x}) = \frac{1}{Z_\theta}\exp(f_\theta(\mathbf{x}))$. Let $\tilde{\mathbf{x}} = \mathbf{x} + \sigma\boldsymbol{\epsilon}$ (data at a higher noise level).
- The conditional energy-based models can be derived by Bayes' rule:

$$p_\theta(\mathbf{x}|\tilde{\mathbf{x}}) = \frac{1}{\tilde{Z}_\theta(\tilde{\mathbf{x}})}\exp\left(f_\theta(\mathbf{x}) - \boxed{\frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - \mathbf{x}\|^2}\right)$$

Localize the energy landscape



$-f_\theta(\mathbf{x})$ $\quad$ $-(f_\theta(\mathbf{x}) - \frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - \mathbf{x}\|^2)$

Compared to a single EBM:

- Sampling is more friendly and easier to converge
- Training is more efficient
- Well-formed energy potential

Compared to diffusion models:

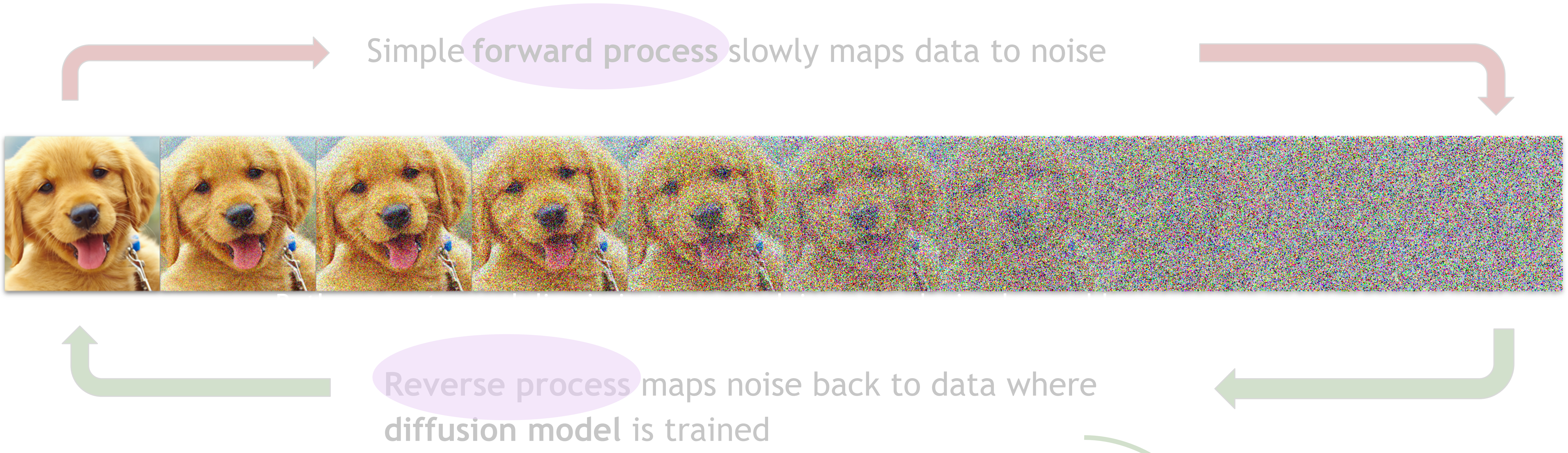
- Much less diffusion steps (6 steps)

- Learn the sequence of EBMs by maximizing conditional log-likelihoods: $\mathcal{J}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\log p_\theta(\mathbf{x}_i|\tilde{\mathbf{x}}_i)$
- Get samples by progressive sampling from EBMs from high-noise levels to low-noise levels.

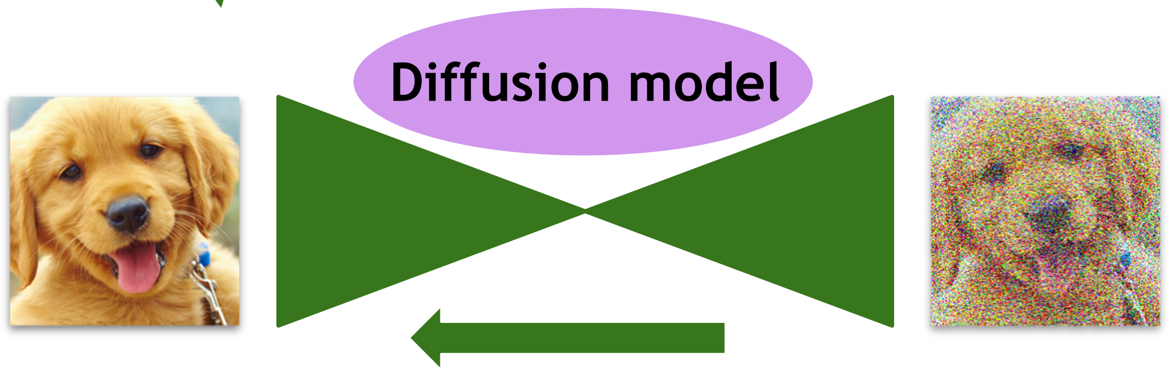
Gao et al., "Learning energy-based models by diffusion recovery likelihood", ICLR 2021.

Advanced modeling

Latent space modeling & model distillation

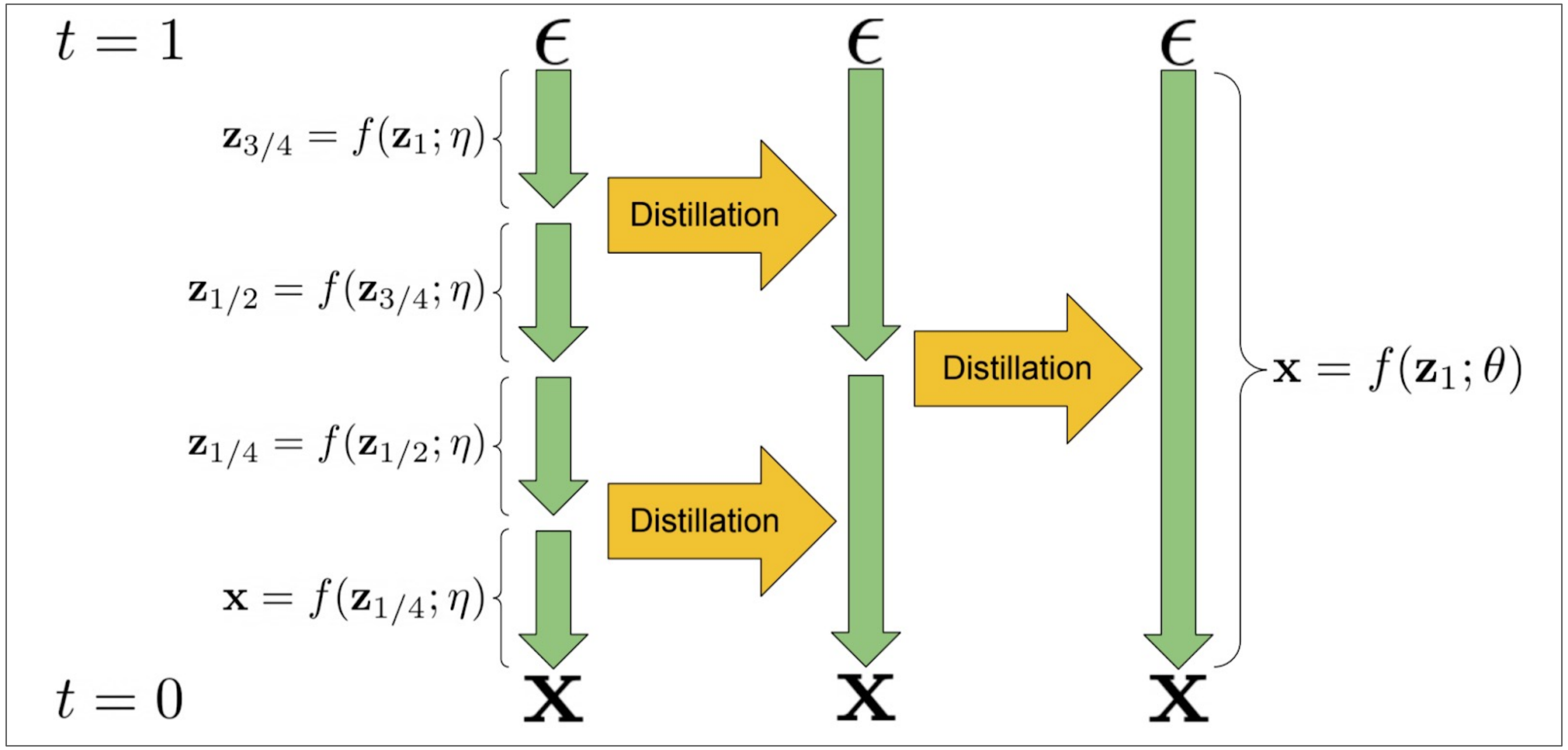


- Can we do model distillation for fast sampling?
- Can we lift the diffusion model to a latent space that is faster to diffuse?



Progressive distillation

- Distill a deterministic 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.



Distillation stage $\longrightarrow$

[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

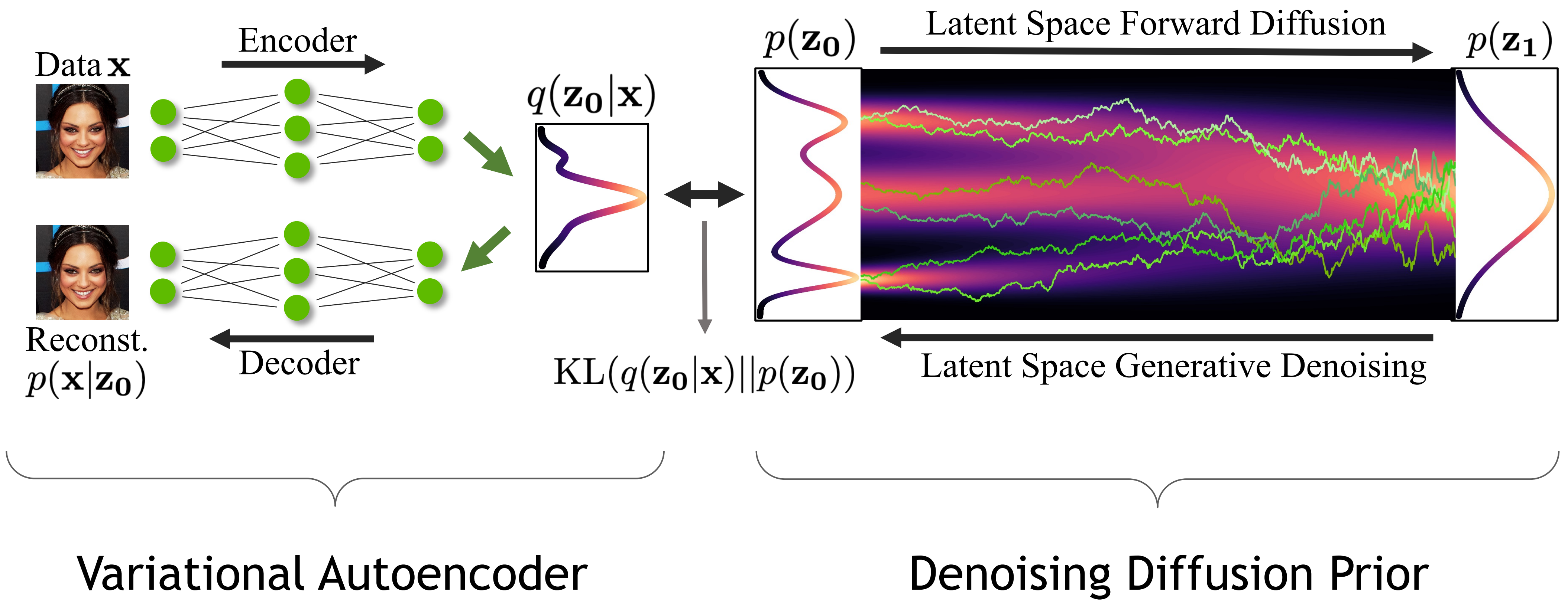
Algorithm 2 Progressive distillation

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**
    $\mathbf{x} \sim \mathcal{D}$
    $t = i/N, \quad i \sim Cat[1, 2, \ldots, N]$
    $\epsilon \sim N(0, I)$
    $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$
    `# 2 steps of DDIM with teacher`
    $t' = t - 0.5/N, \quad t'' = t - 1/N$
    $$\boxed{\begin{aligned}\mathbf{z}_{t'} &= \alpha_{t'}\hat{\mathbf{x}}_\eta(\mathbf{z}_t) + \frac{\sigma_{t'}}{\sigma_t}(\mathbf{z}_t - \alpha_t \hat{\mathbf{x}}_\eta(\mathbf{z}_t))\\ \mathbf{z}_{t''} &= \alpha_{t''}\hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}) + \frac{\sigma_{t''}}{\sigma_{t'}}(\mathbf{z}_{t'} - \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}))\end{aligned}}$$
    $\boxed{\tilde{\mathbf{x}} = \frac{\mathbf{z}_{t''} - (\sigma_{t''}/\sigma_t)\mathbf{z}_t}{\alpha_{t''} - (\sigma_{t''}/\sigma_t)\alpha_t}}$ ▷ Teacher $\hat{\mathbf{x}}$ target
    $\lambda_t = \log[\alpha_t^2/\sigma_t^2]$
    $\boxed{L_\theta = w(\lambda_t)\|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_\theta(\mathbf{z}_t)\|_2^2}$
    $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$
  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.

Latent-space diffusion models

Variational autoencoder + score-based prior



Main Idea

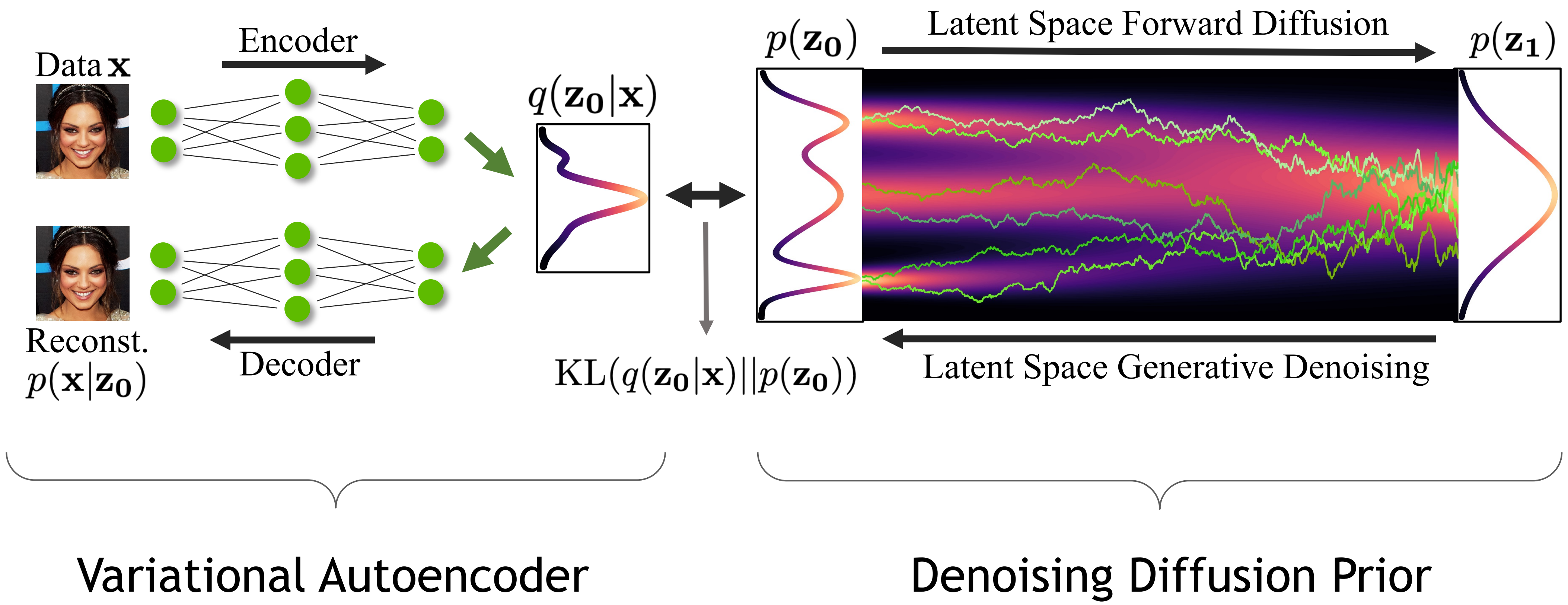
Encoder maps the input data to an embedding space

Denoising diffusion models are applied in the latent space

Vahdat et al., "Score-based generative modeling in latent space", NeurIPS 2021.
Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR 2022.

Latent-space diffusion models

Variational autoencoder + score-based prior



Advantages:

(1) The distribution of latent embeddings close to Normal distribution → ***Simpler denoising, Faster Synthesis!***

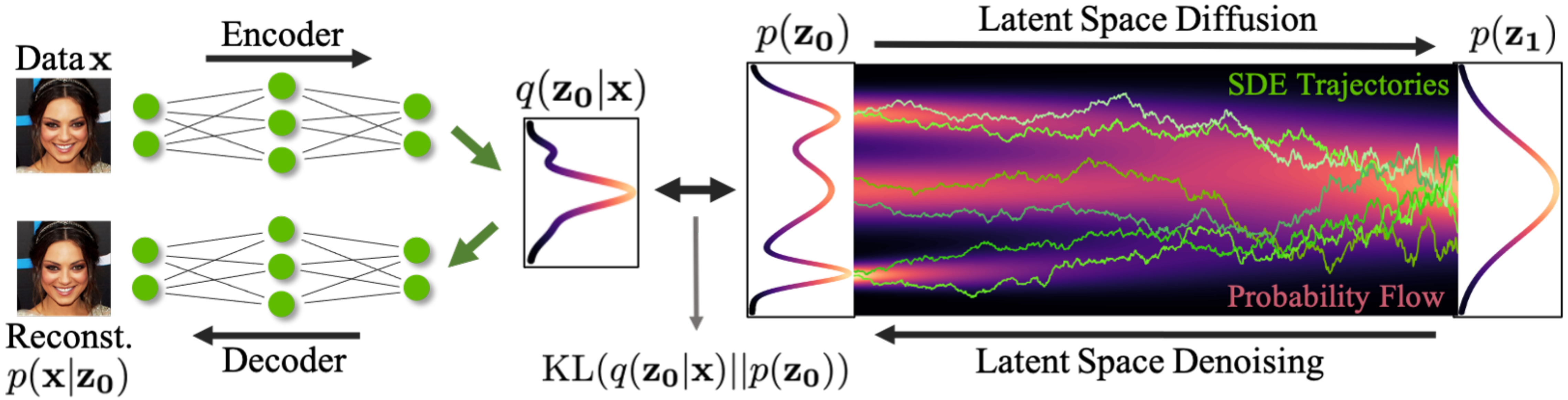
(2) Augmented latent space → ***More expressivity!***

(3) Tailored Autoencoders → ***More expressivity, Application to any data type (graphs, text, 3D data, etc.) !***

Vahdat et al., "Score-based generative modeling in latent space", NeurIPS 2021.
Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR 2022.

Latent-space diffusion models

Training objective: score-matching for cross entropy



$$\mathcal{L}(\mathbf{x},\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{\psi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[-\log p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z}_0)\right] + \mathrm{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}_0)\right)$$

$$= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[-\log p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z}_0)\right]}_{\text{reconstruction term}} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})\right]}_{\text{negative encoder entropy}} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[-\log p_{\boldsymbol{\theta}}(\mathbf{z}_0)\right]}_{\text{cross entropy}}$$

$$CE(q(\mathbf{z}_0|\mathbf{x})||p(\mathbf{z}_0)) = \underbrace{\mathbb{E}_{t\sim\mathcal{U}[0,1]}}_{\text{time sampling}}\left[\frac{g(t)^2}{2}\underbrace{\mathbb{E}_{q(\mathbf{z}_t,\mathbf{z}_0|\mathbf{x})}}_{\text{Forward diffusion}}\left[||\underbrace{\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{z}_0)}_{\text{Diffusion kernel}} - \underbrace{\nabla_{\mathbf{z}_t}\log p(\mathbf{z}_t)}_{\text{Trainable score function}}||_2^2\right]\right] + \underbrace{\frac{D}{2}\log\left(2\pi e\sigma_0^2\right)}_{\text{Constant}}$$

Vahdat et al., "Score-based generative modeling in latent space", NeurIPS 2021.

Part 3-2:
Q: How to do high-resolution conditional generation?

Impressive conditional diffusion models

Text-to-image generation

DALL·E 2

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

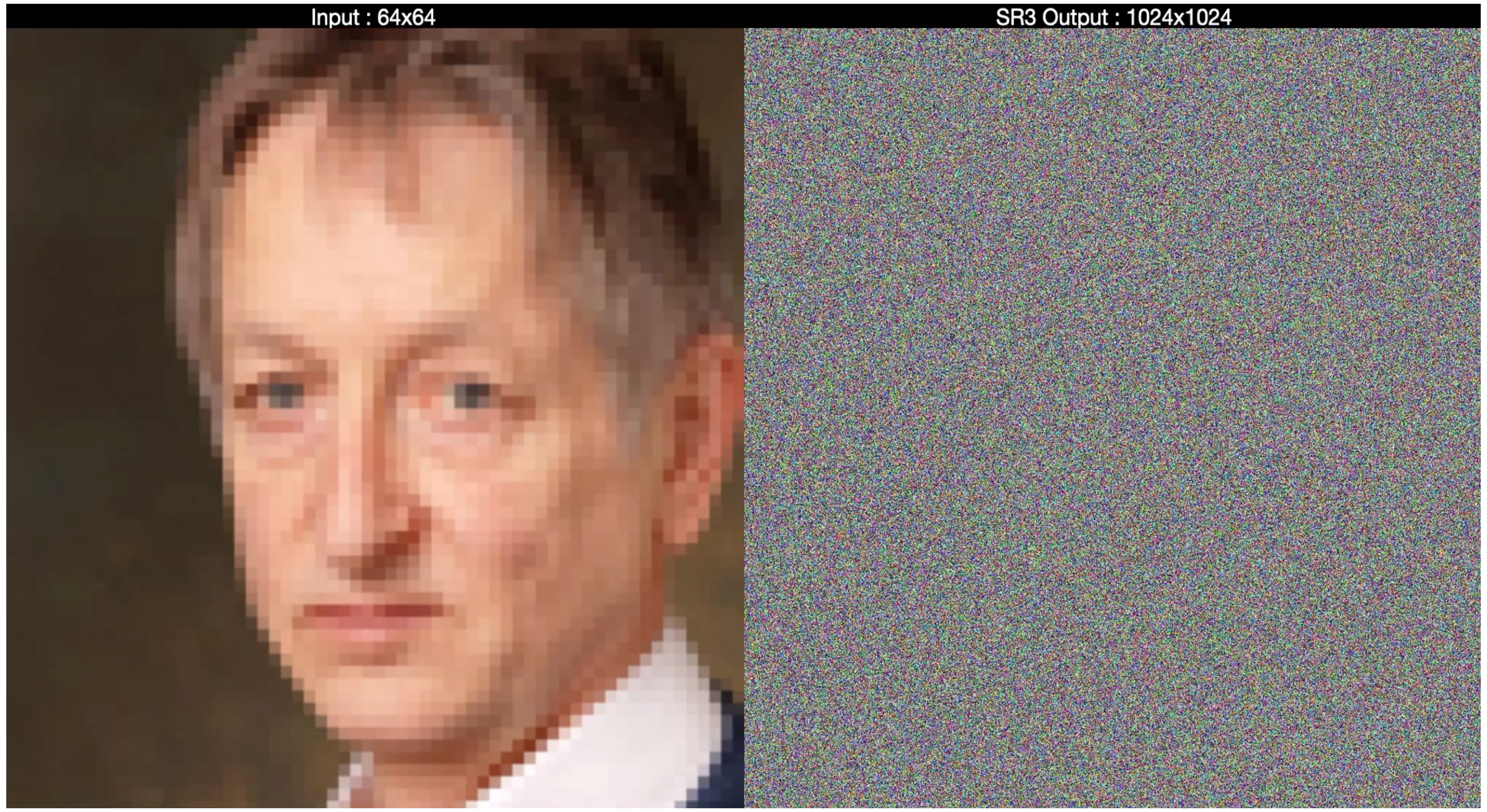
IMAGEN

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

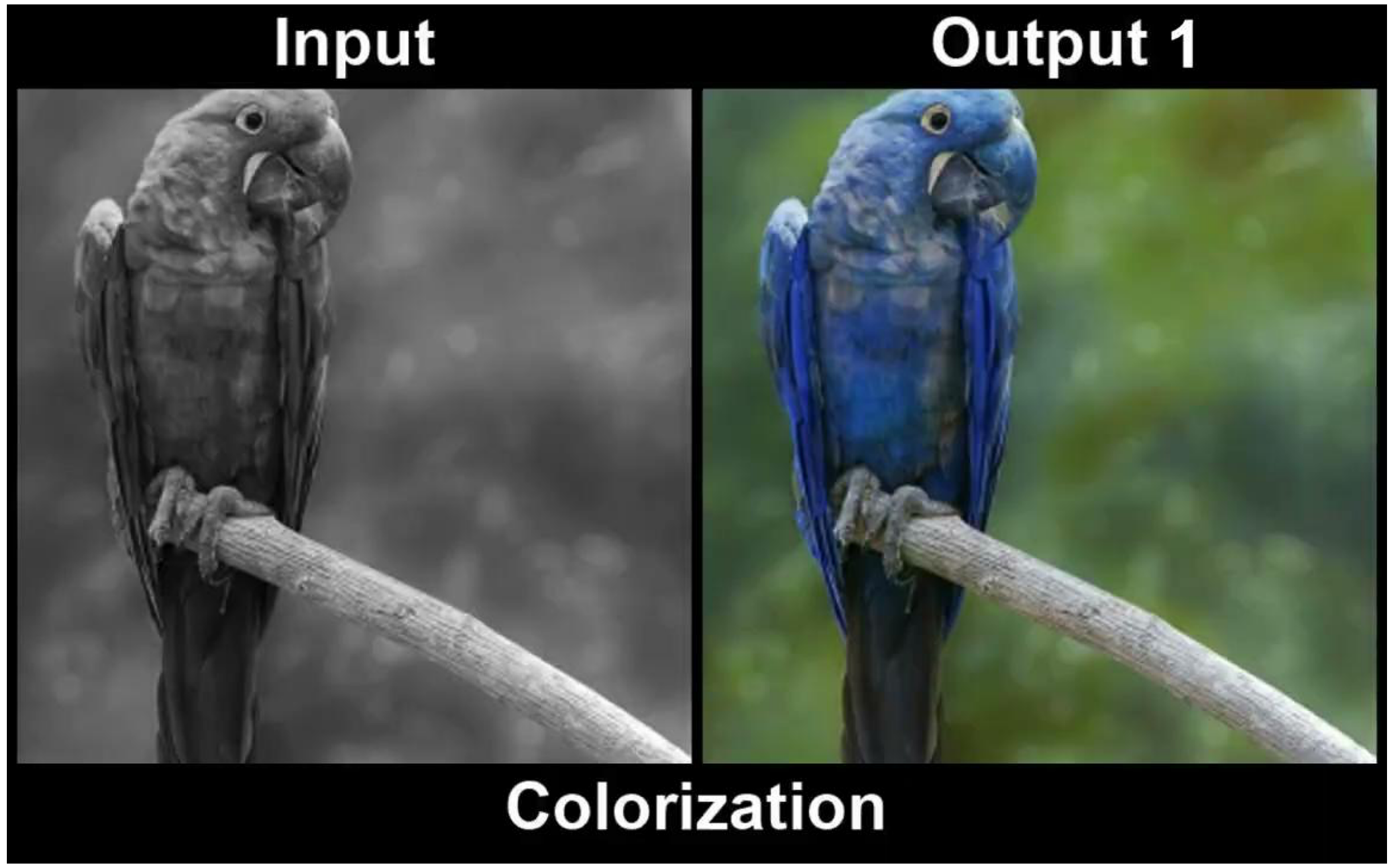
[Ramesh et al., “Hierarchical Text-Conditional Image Generation with CLIP Latents”, arXiv 2022.](#)
[Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.](#)

Impressive conditional diffusion models

Super-resolution & colorization



Super-resolution

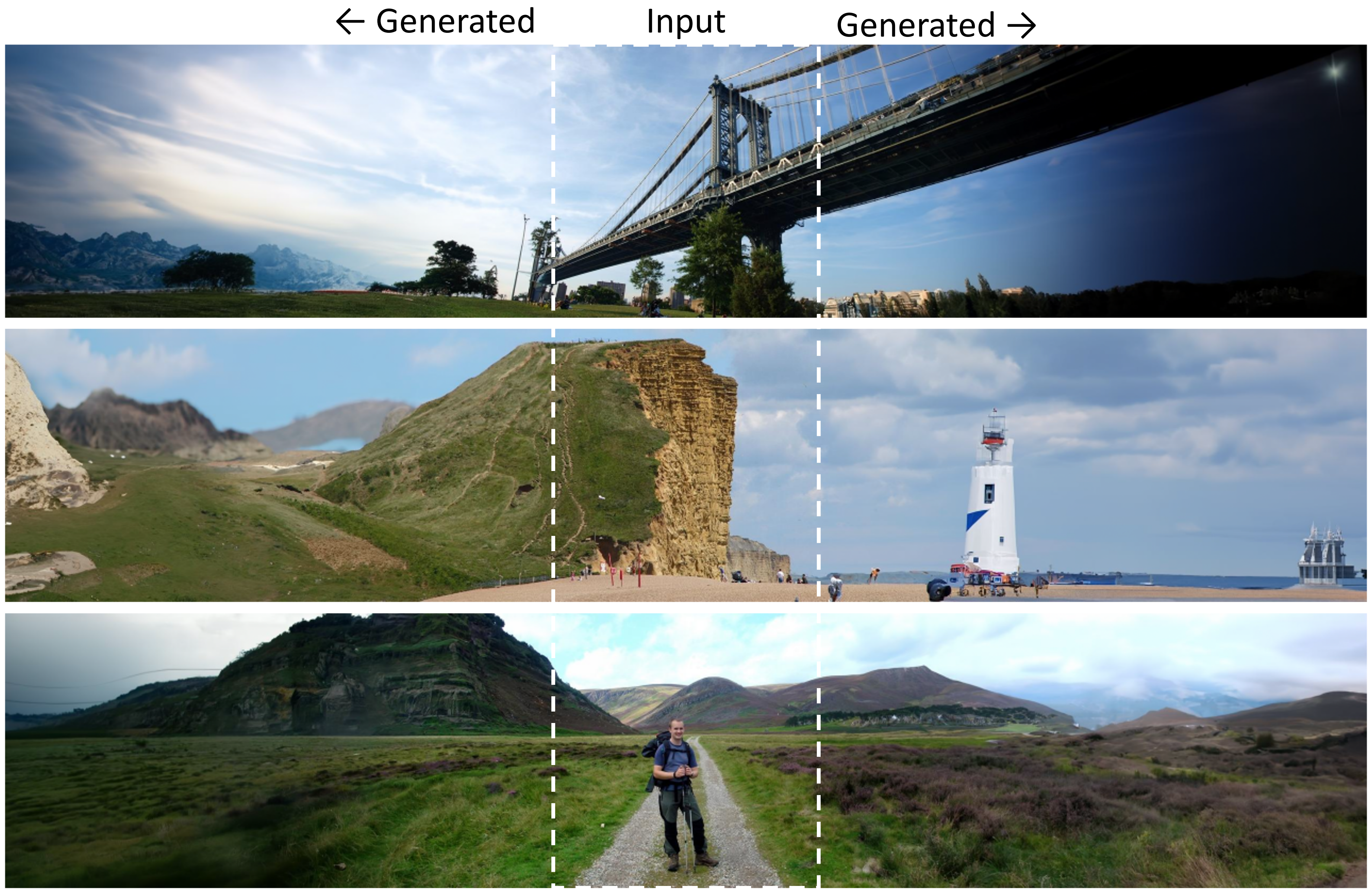


Colorization

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

Impressive conditional diffusion models

Panorama generation



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

Conditional diffusion models

Include condition as input to reverse process

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

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

Score model

Classifier gradient

$x_T \leftarrow$ 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

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

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

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 guidance

Get 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:

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

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 guidance

Trade-off for sample quality and sample diversity

Non-guidance

$\omega = 1$

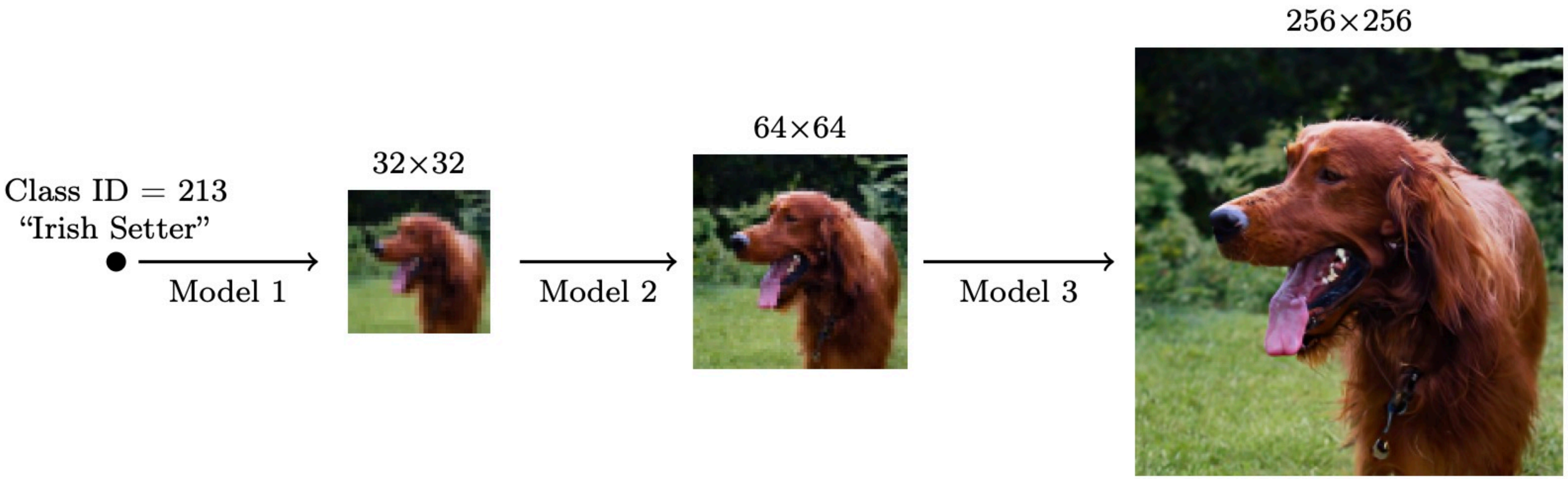
$\omega = 3$

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

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

Cascaded generation

Pipeline



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.]

Noise conditioning augmentation

Reduce compounding error

- 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.

Summary

Questions to address with advanced techniques

- Q1: How to accelerate the sampling process?
    - Advanced forward diffusion process
    - Advanced reverse process
    - Hybrid models & model distillation
- Q2: How to do high-resolution (conditional) generation?
    - Conditional diffusion models
    - Classifier(-free) guidance
    - Cascaded generation

Today’s Program

Title	Speaker	Time
Introduction	Arash	10 min
Part (1): Denoising Diffusion Probabilistic Models	Arash	35 min
Part (2): Score-based Generative Modeling with Differential Equations	Karsten	45 min
Part (3): Advanced Techniques: Accelerated Sampling, Conditional Generation, and Beyond	Ruiqi	45 min
Applications (1): Image Synthesis, Text-to-Image, Controllable Generation	Ruiqi	15 min
Applications (2): Image Editing, Image-to-Image, Super-resolution, Segmentation	Arash	15 min
Applications (3): Video Synthesis, Medical Imaging, 3D Generation, Discrete State Models	Karsten	15 min
Conclusions, Open Problems and Final Remarks	Arash	10 min

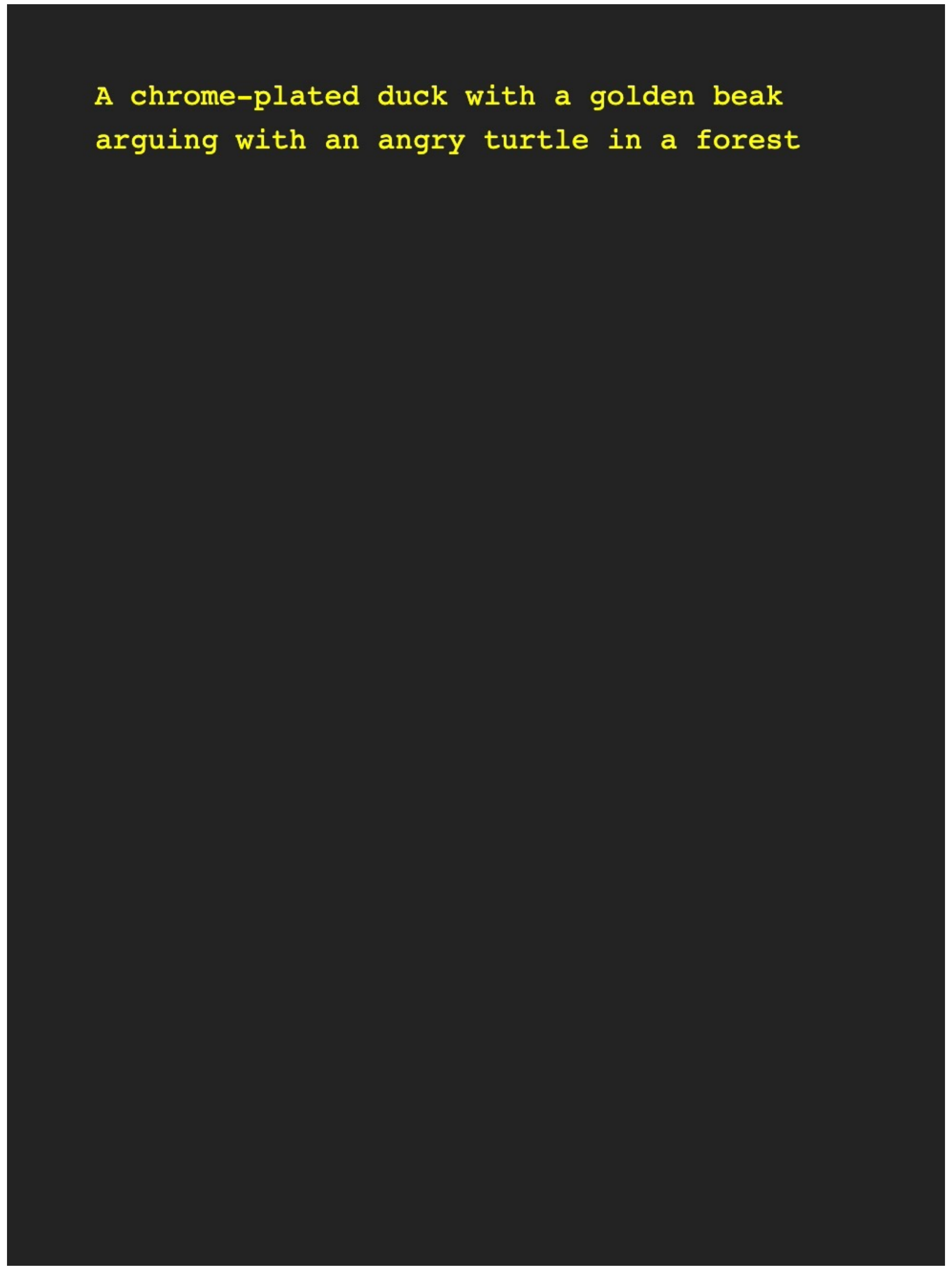
cvpr2022-tutorial-diffusion-models.github.io

Applications (1):
Image Synthesis, Controllable Generation, Text-to-Image

Text-to-image generation

Inverse of image captioning

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



Video source: [Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.]

GLIDE

OpenAI

- 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)

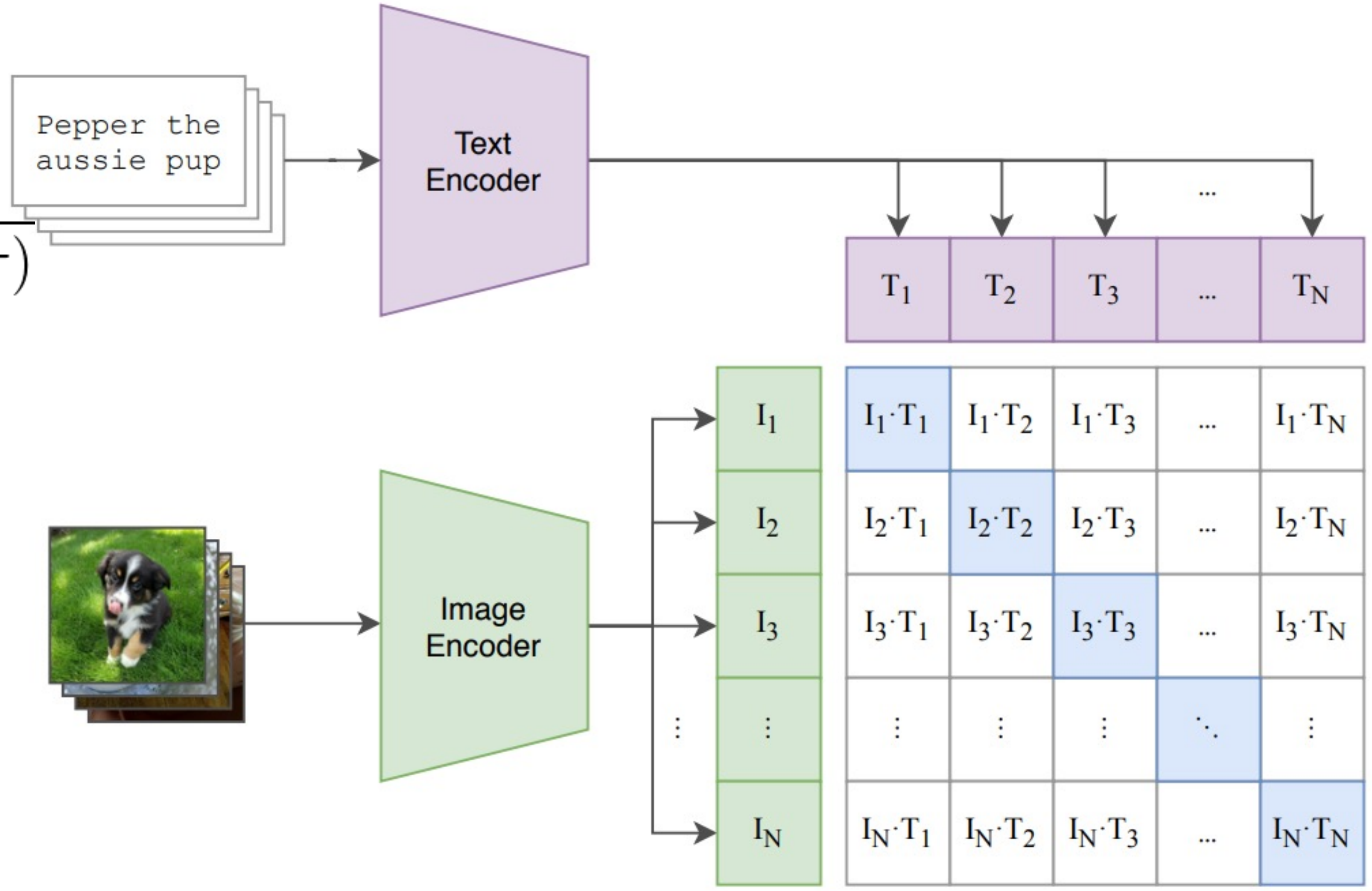
[Nichol et al., "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models", 2021.]

CLIP guidance

What is a CLIP model?

- 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.]

CLIP guidance

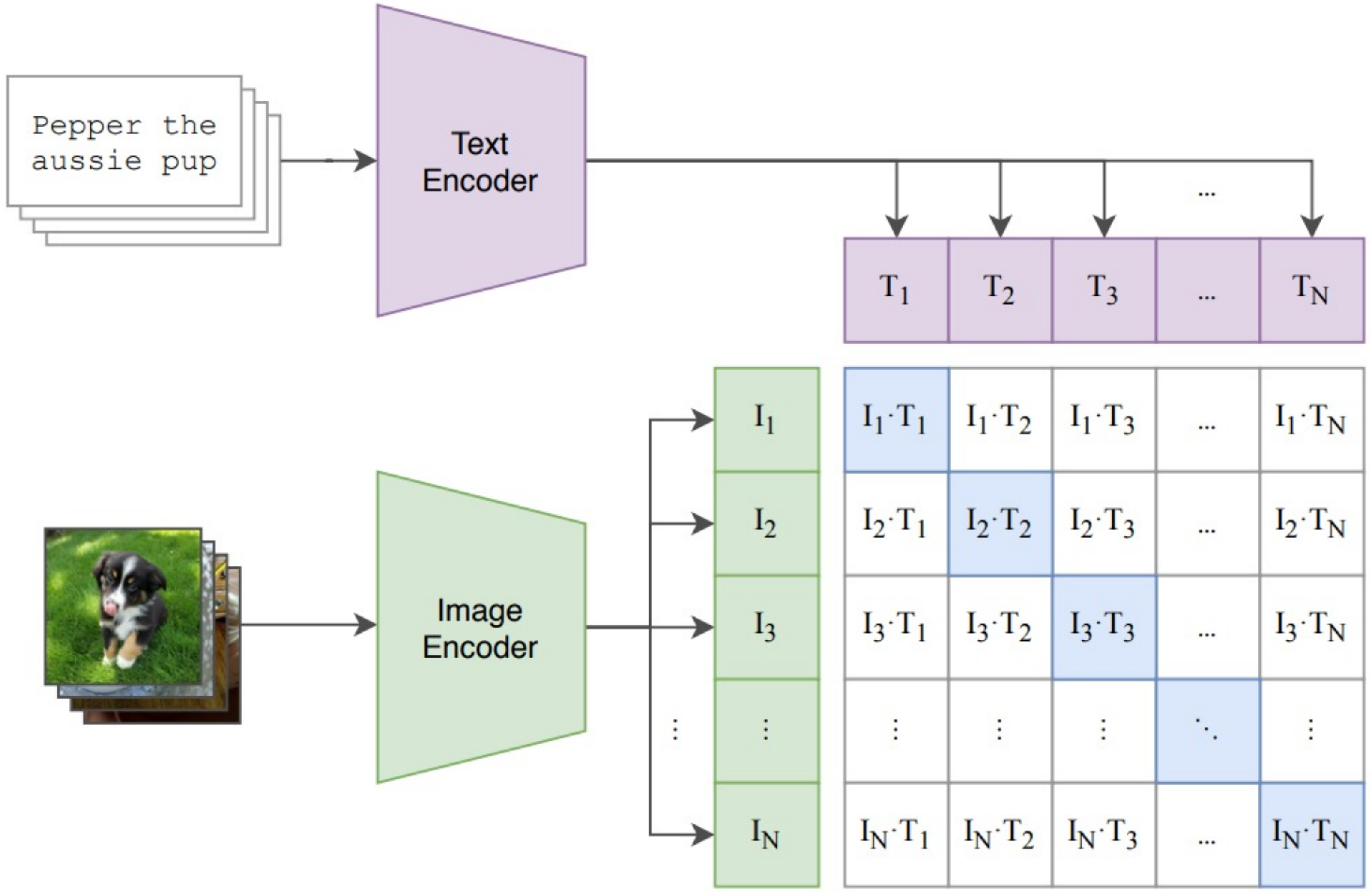
Replace the classifier in classifier guidance with a CLIP model

- 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

OpenAI

- 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.](#)

DALL·E 2

OpenAI


a shiba inu wearing a beret and black turtleneck


a close up of a handpalm with leaves growing from it

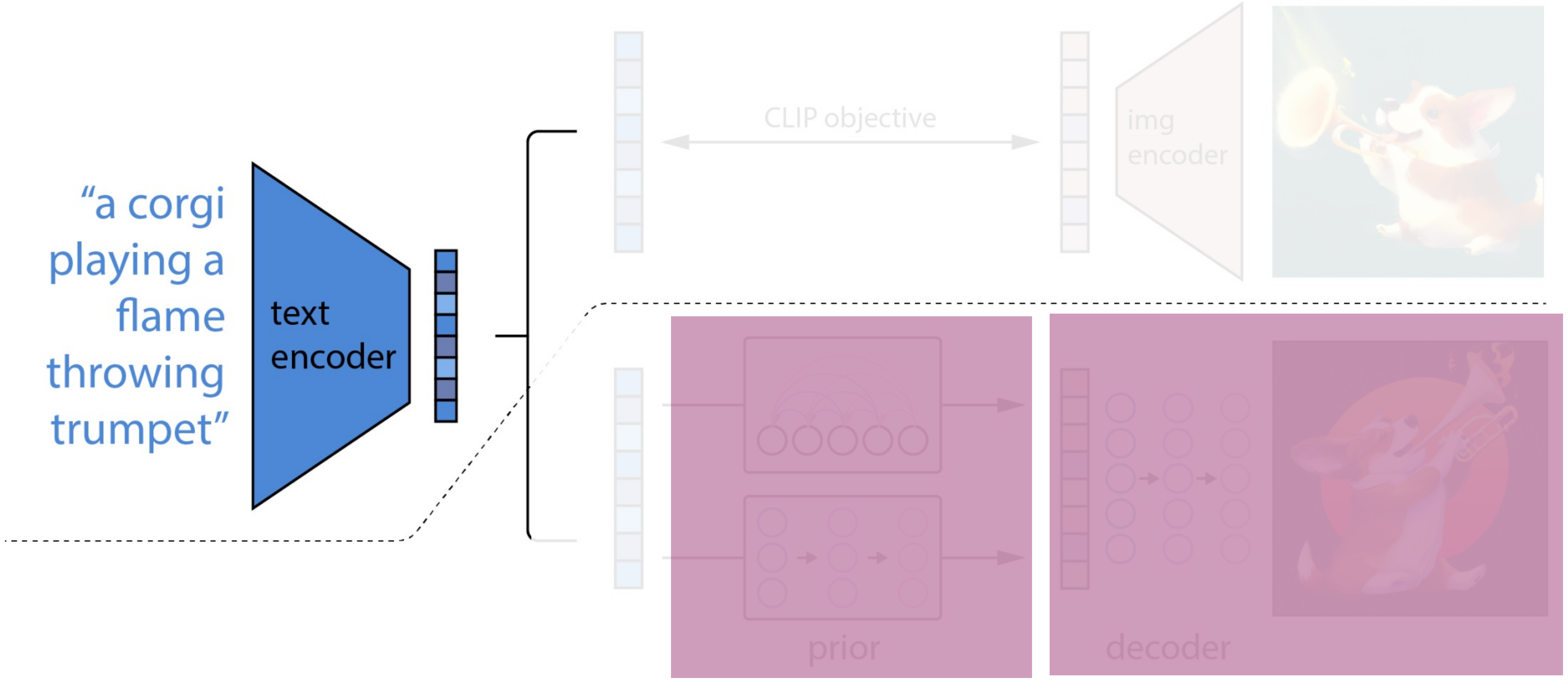
1kx1k Text-to-image generation.

Outperform DALL-E (autoregressive transformer).

Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

DALL·E 2

Model components



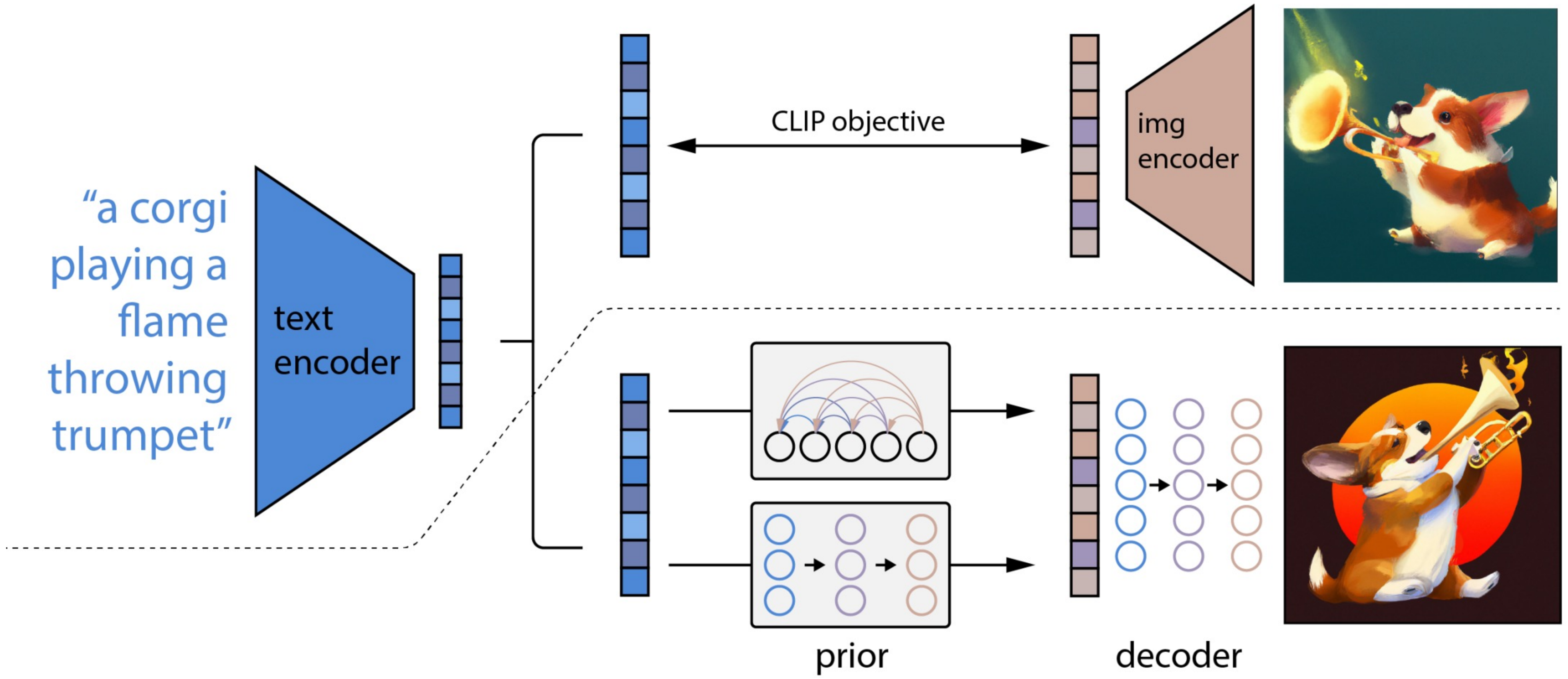
Prior: produces CLIP image embeddings conditioned on the caption.

Decoder: produces images conditioned on CLIP image embeddings and text.

[Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.]

DALL·E 2

Model components



Why conditional on CLIP image embeddings?

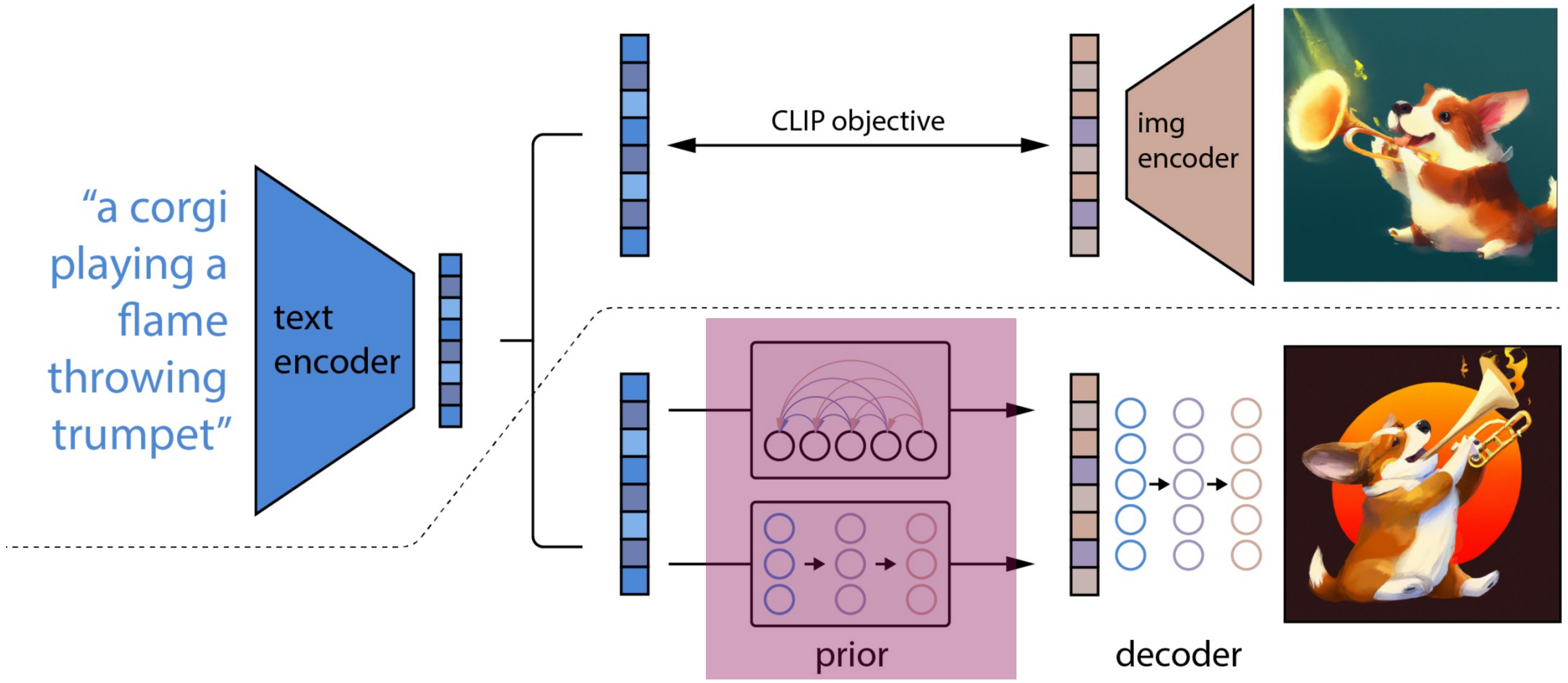
CLIP image embeddings capture high-level semantic meaning; latents in the decoder model take care of the rest.

The bipartite latent representation` enables several text-guided image manipulation tasks.

Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

DALL·E 2

Model components (1/2): prior model



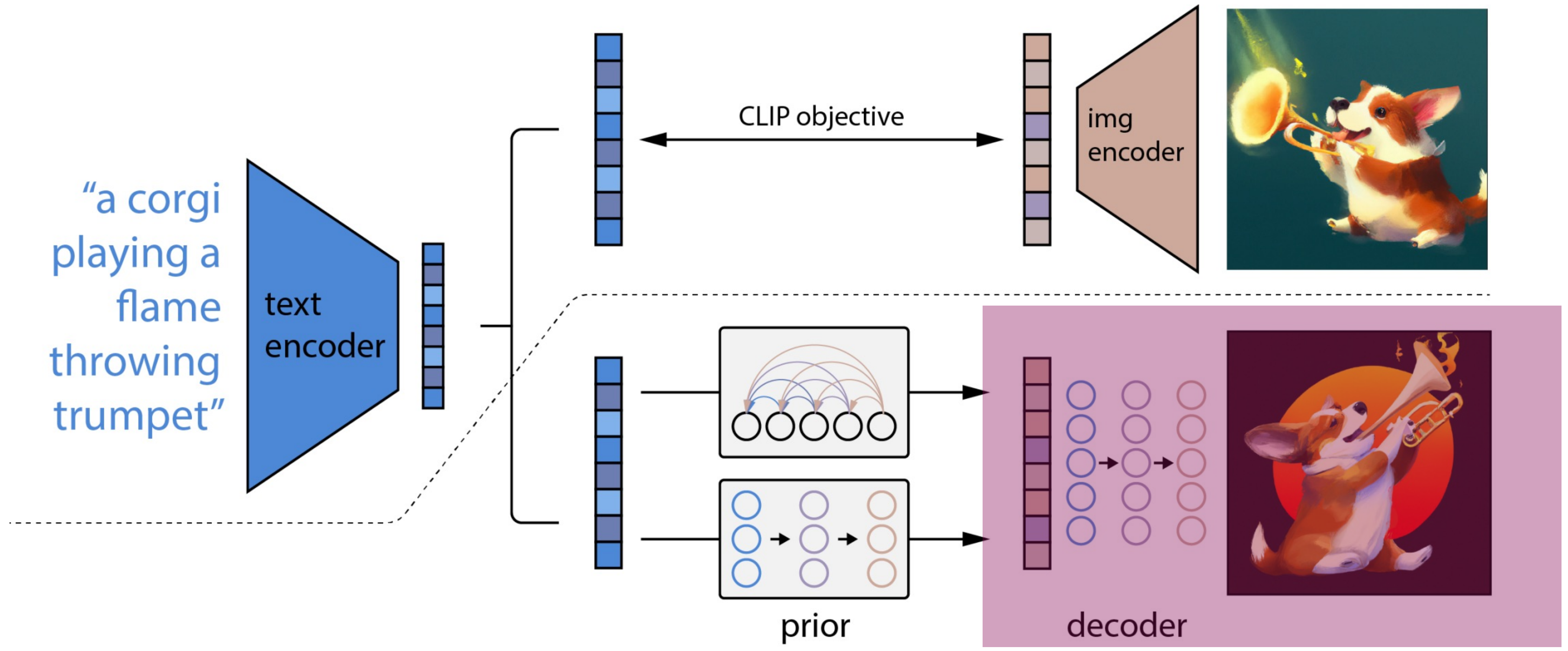
Prior: produces CLIP image embeddings conditioned on the caption.

- Option 1. autoregressive prior: quantize image embedding to a seq. of discrete codes and predict them autoregressively.
- Option 2. diffusion prior: model the continuous image embedding by diffusion models conditioned on caption.

[Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.](#)

DALL·E 2

Model components (2/2): decoder model



Decoder: produces images conditioned on CLIP image embeddings (and text).

- Cascaded diffusion models: 1 base model (64x64), 2 super-resolution models (64x64 → 256x256, 256x256 → 1024x1024).
- Largest super-resolution model is trained on patches and takes full-res inputs at inference time.
- Classifier-free guidance & noise conditioning augmentation are important.

Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

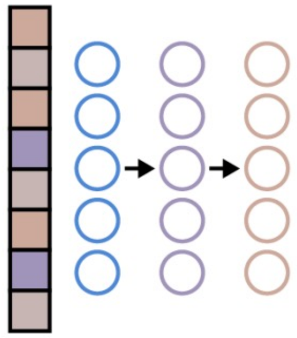
DALL·E 2

Bipartite latent representations

Bipartite latent representations $(\mathbf{z}, \mathbf{x}_T)$

$\mathbf{z}$: CLIP image embeddings

$\mathbf{x}_T$: inversion of DDIM sampler
(latents in the decoder model)



decoder

Near exact
reconstruction

DALL·E 2

Image variations

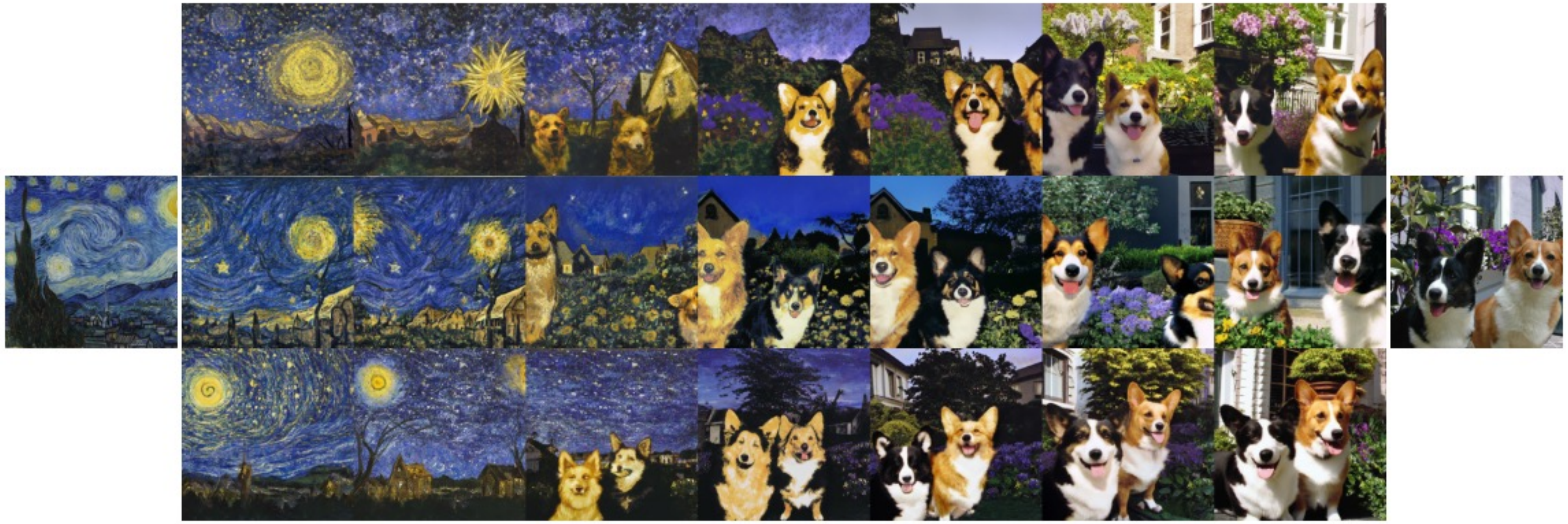
Fix the CLIP embedding $\mathbf{z}_i$

Decode using different decoder latents $\mathbf{x}_T$

Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

DALL·E 2

Image interpolation



Interpolate image CLIP embeddings $\mathbf{z}$.

Use different $\mathbf{x}_T$ to get different interpolation trajectories.

Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

DALL·E 2

Text Diffs

a photo of a cat → an anime drawing of a super saiyan cat, artstation

a photo of a victorian house → a photo of a modern house

a photo of an adult lion → a photo of lion cub

Change the image CLIP embedding towards the difference of the text CLIP embeddings of two prompts.

Decoder latent is kept as a constant.

Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

Imagen

Google Research, Brain team

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

Google Research, Brain team

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

Google Research, Brain team

A dragon fruit wearing karate belt in the snow.

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

Imagen

Google Research, Brain team

A relaxed garlic with a blindfold reading a newspaper while floating in a pool of tomato soup.

[Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.]

Imagen

Google Research, Brain team

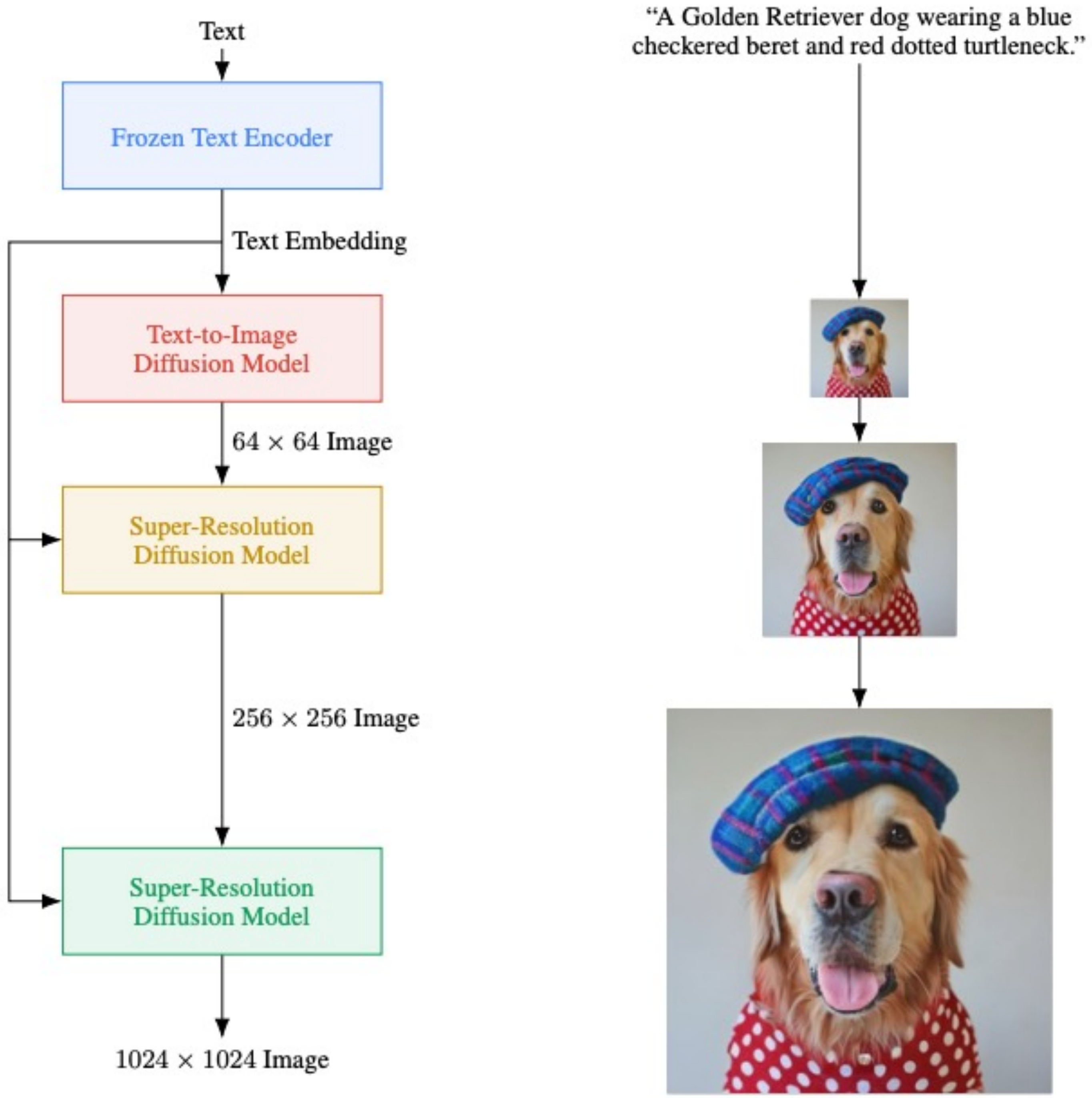
A cute hand-knitted koala wearing a sweater with 'CVPR' written on it.

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

Imagen

Key modeling components:

- Cascaded diffusion models
- Classifier-free guidance and dynamic thresholding.
- Frozen large pretrained language models as text encoders. (T5-XXL)

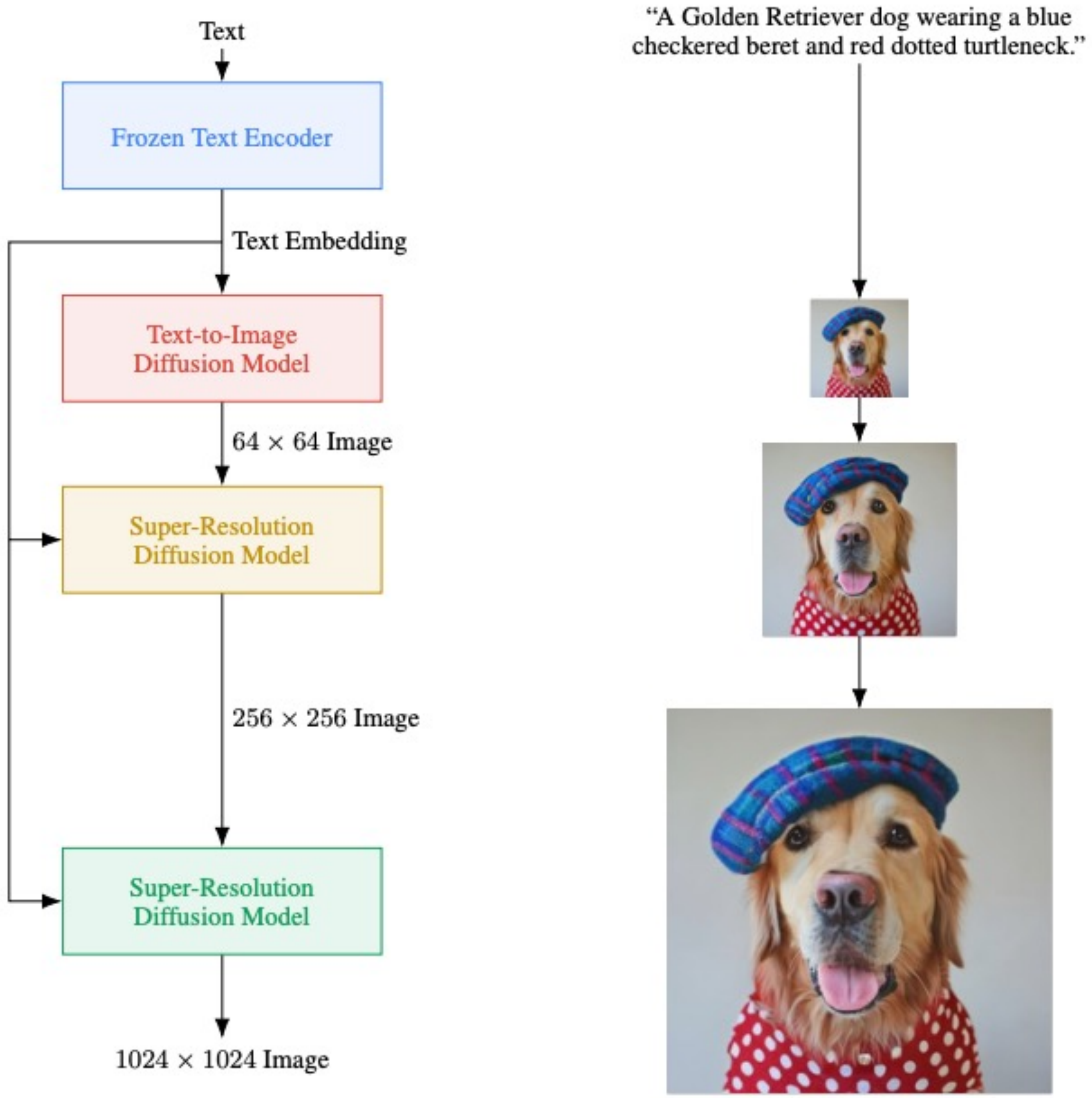


Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

Imagen

Key observations:

- Beneficial to use text conditioning for all super-res models.
    - Noise conditioning augmentation weakens information from low-res models, thus needs text conditioning as extra information input.
- Scaling text encoder is extremely efficient.
    - More important than scaling diffusion model size.
- Human raters prefer T5-XXL as the text encoder over CLIP encoder on DrawBench.

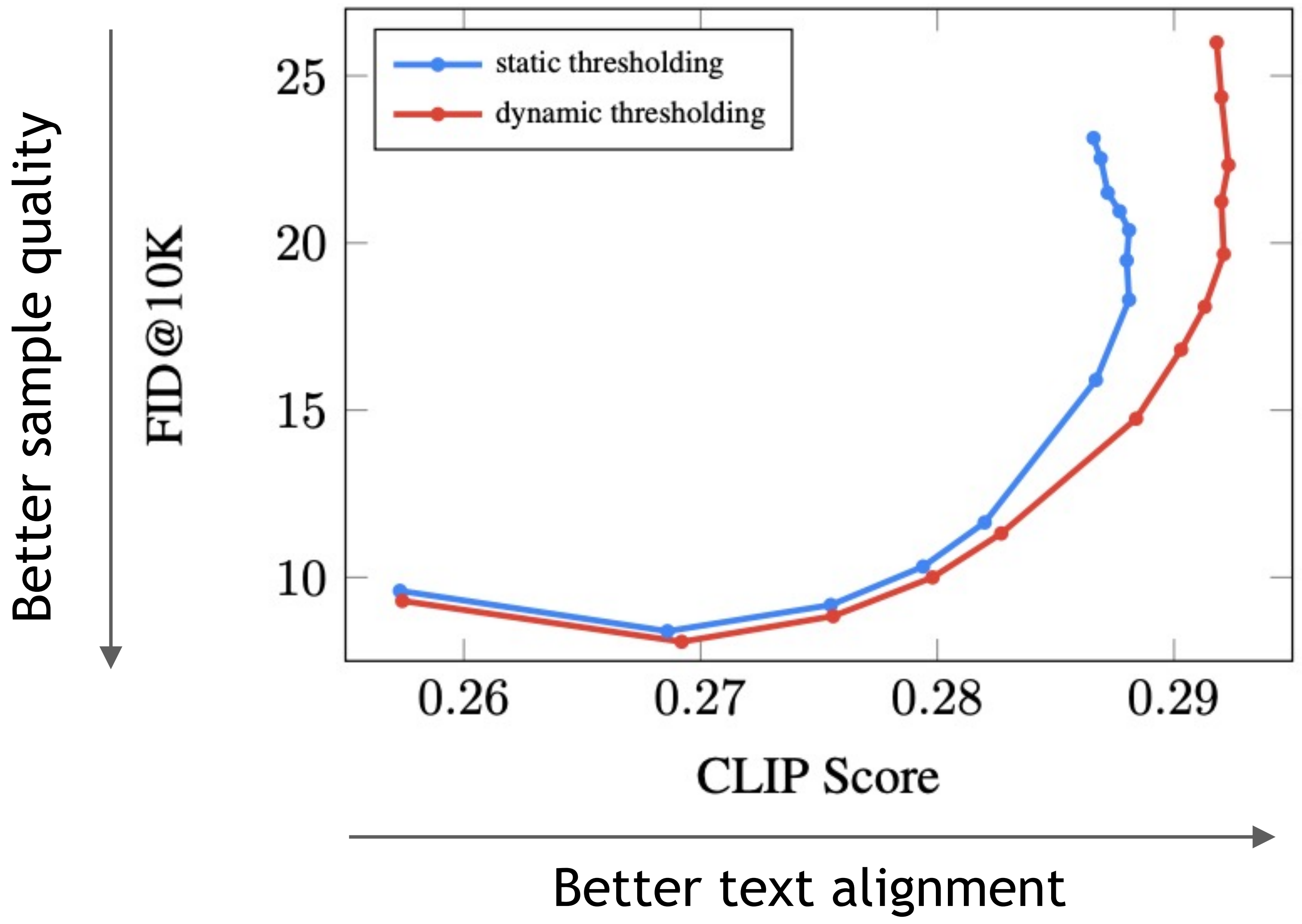


[Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.]()

Imagen

Dynamic thresholding

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



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

Imagen

Dynamic thresholding

- Large classifier-free guidance weights → better text alignment, worse image quality
- Hypothesis : at large guidance weight, the generated images are saturated due to the very large gradient updates during sampling
- Solution – dynamic thresholding: adjusts the pixel values of samples at each sampling step to be within a dynamic range computed over the statistics of the current samples.

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

Imagen

Dynamic thresholding

- Large clas
- Hypothesis dates during sampling
- Solution – dynamic range com

Static thresholding

Dynamic thresholding

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

Imagen

DrawBench: new benchmark for text-to-image evaluations

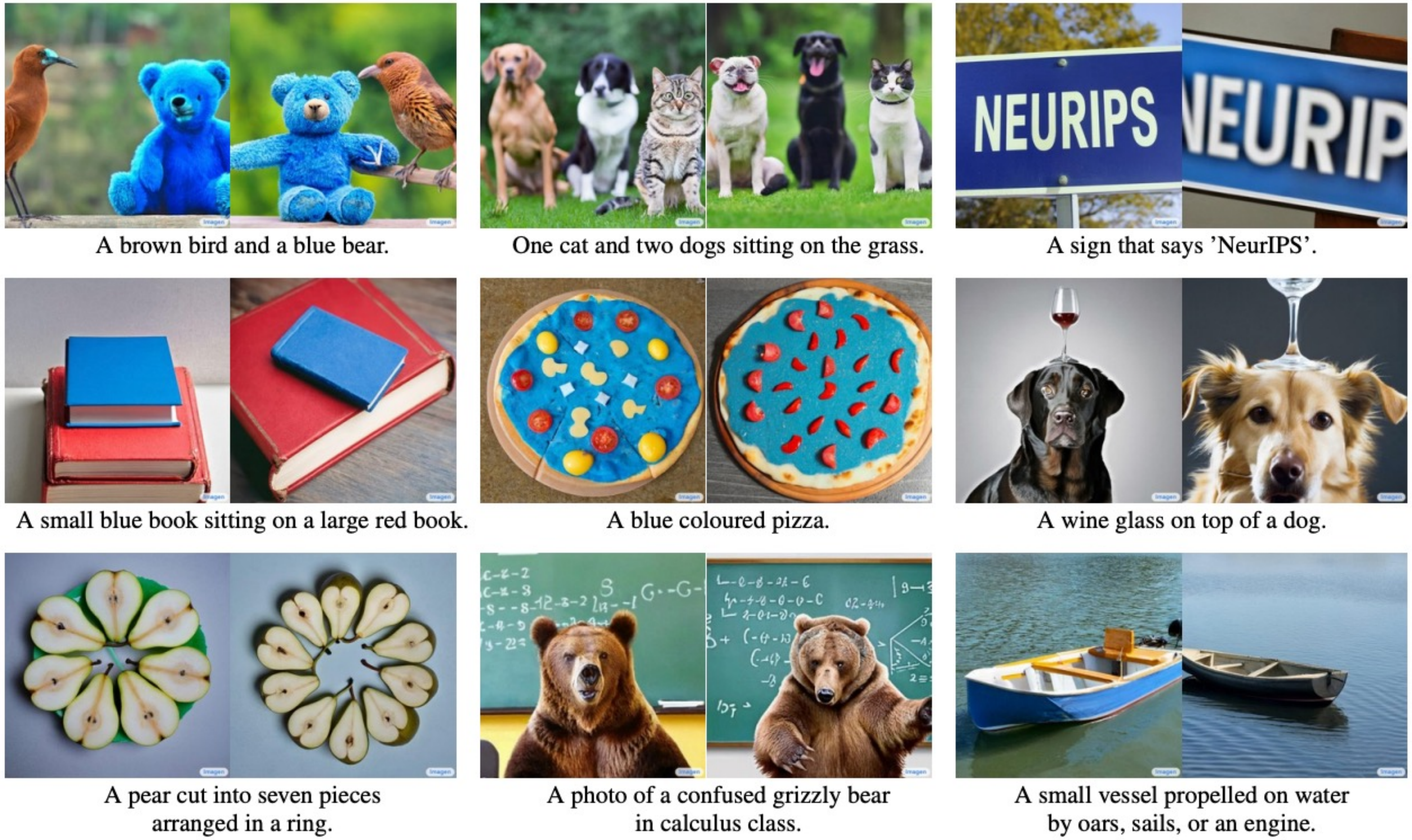
- A set of 200 prompts to evaluate text-to-image models across multiple dimensions.
    - E.g., the ability to faithfully render different colors, numbers of objects, spatial relations, text in the scene, unusual interactions between objects.
    - Contains complex prompts, e.g, long and intricate descriptions, rare words, misspelled prompts.

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

Imagen

DrawBench: new benchmark for text-to-image evaluations

- A set of 200 pro
  - E.g., the cene, unusual interactio
  - Contains (



A brown bird and a blue bear. One cat and two dogs sitting on the grass. A sign that says 'NeurIPS'.
A small blue book sitting on a large red book. A blue coloured pizza. A wine glass on top of a dog.
A pear cut into seven pieces arranged in a ring. A photo of a confused grizzly bear in calculus class. A small vessel propelled on water by oars, sails, or an engine.

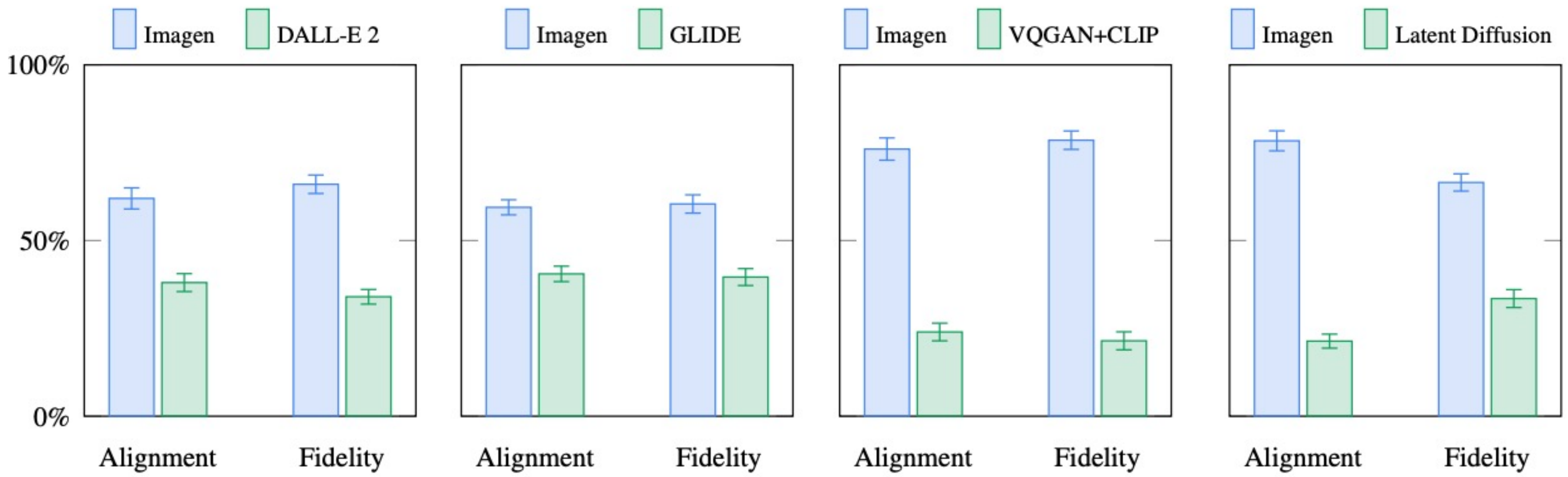
Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

Imagen
Evaluations

Imagen got SOTA automatic evaluation scores on COCO dataset

Model	FID-30K	Zero-shot FID-30K
AttnGAN [76]	35.49	
DM-GAN [83]	32.64	
DF-GAN [69]	21.42	
DM-GAN + CL [78]	20.79	
XMC-GAN [81]	9.33	
LAFITE [82]	8.12	
Make-A-Scene [22]	7.55	
DALL-E [53]		17.89
LAFITE [82]		26.94
GLIDE [41]		12.24
DALL-E 2 [54]		10.39
Imagen (Our Work)		**7.27**

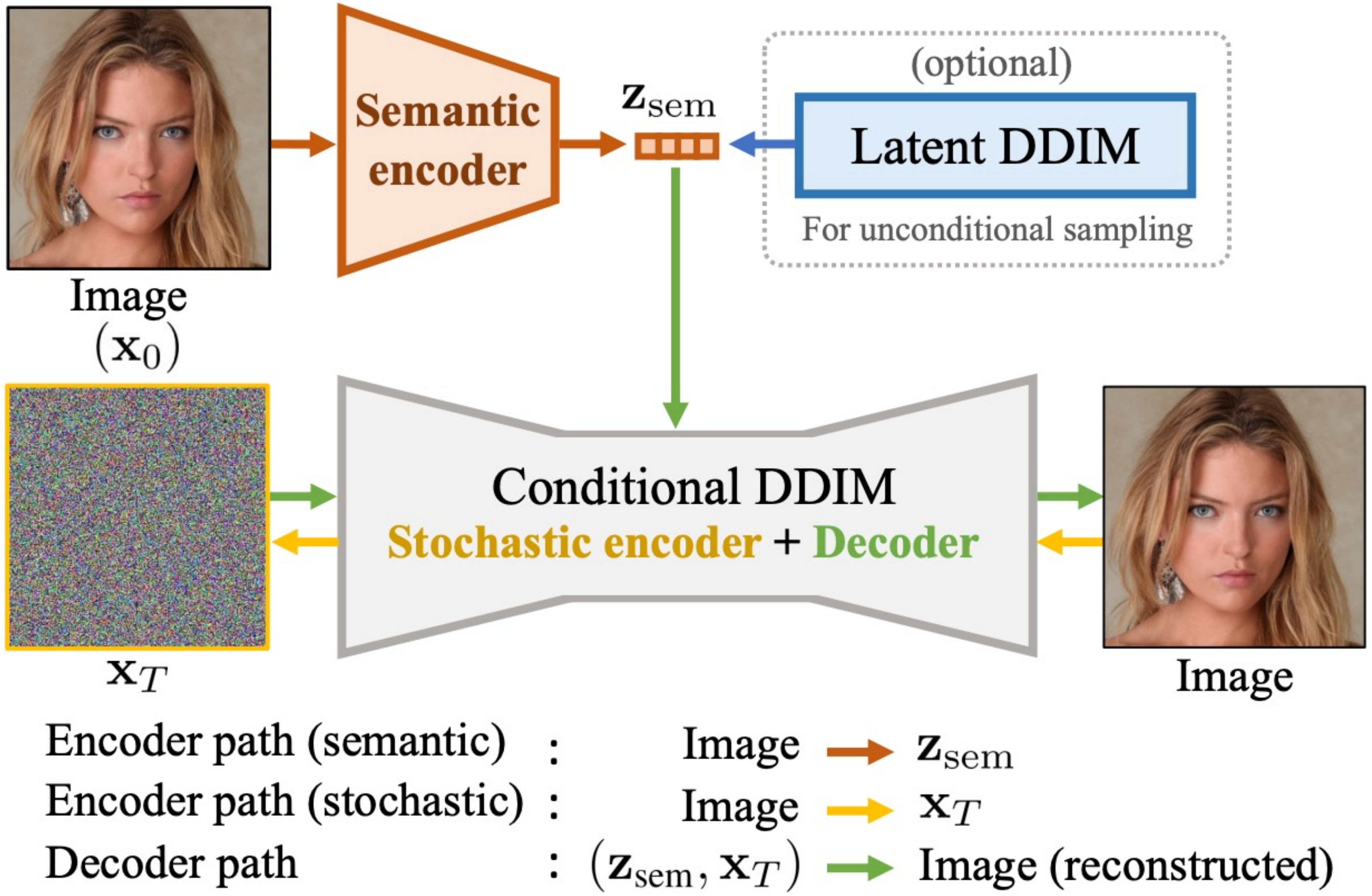
Imagen is preferred over recent work by human raters in sample quality & image-text alignment on DrawBench.



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

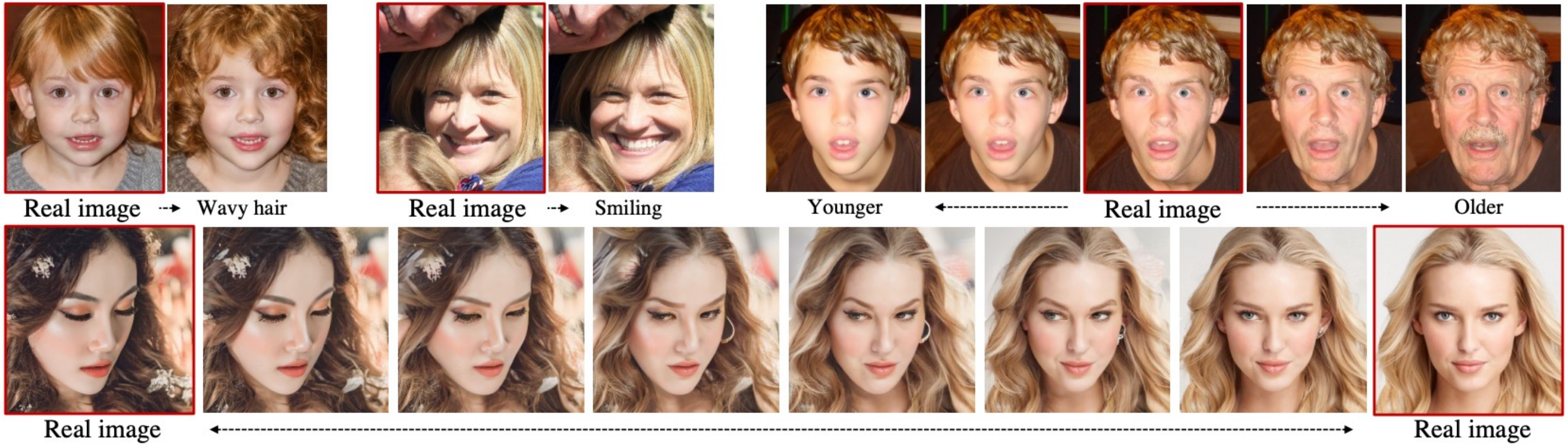
Diffusion Autoencoders

Learning semantic meaningful latent representations in diffusion models



Diffusion Autoencoders

Learning semantic meaningful latent representations in diffusion models

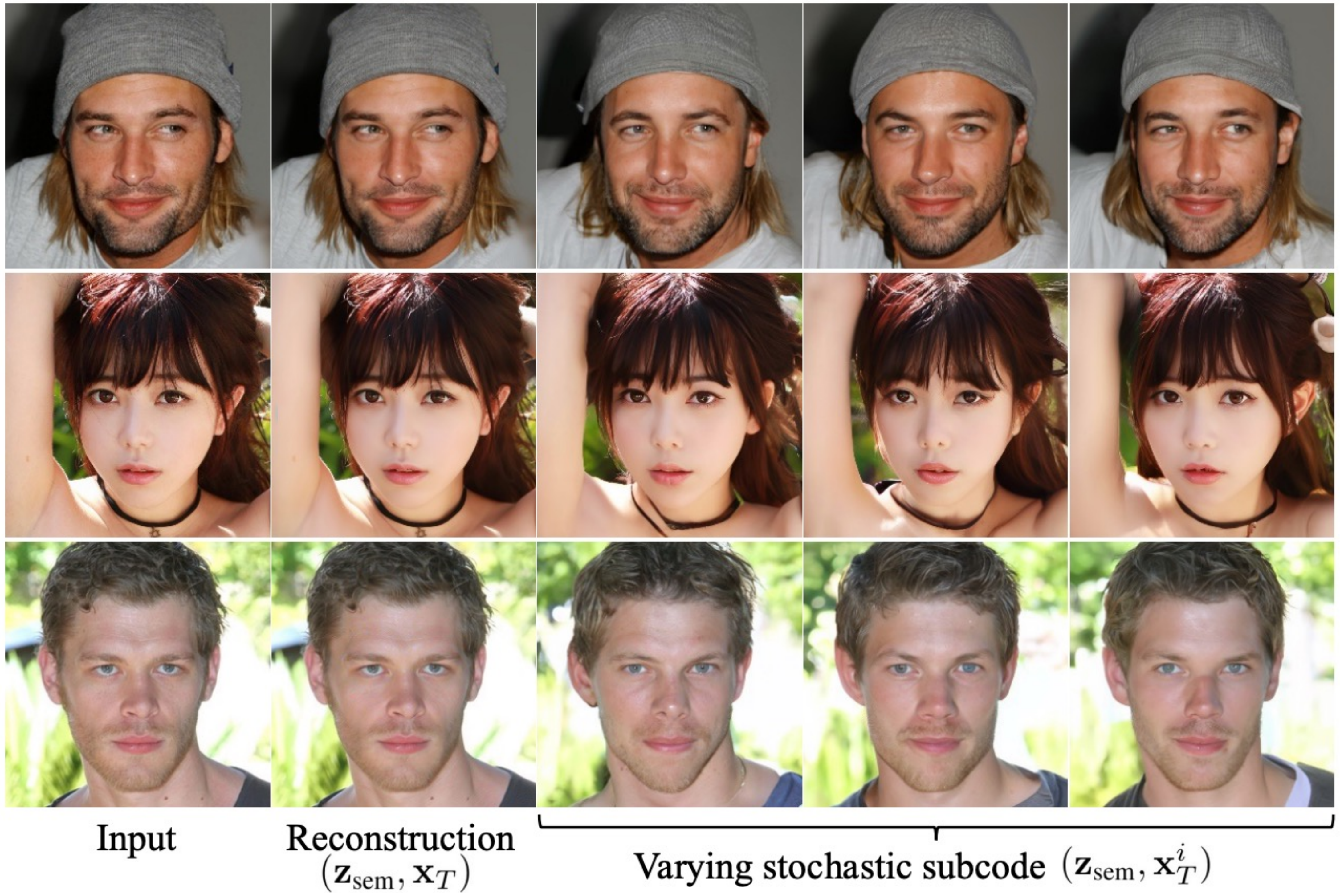


Changing the semantic latent $\mathbf{z}_{\text{sem}}$

Preechakul et al., "Diffusion Autoencoders: Toward a Meaningful and Decodable Representation", CVPR 2022.

Diffusion Autoencoders

Learning semantic meaningful latent representations in diffusion models



[Preechakul et al., “Diffusion Autoencoders: Toward a Meaningful and Decodable Representation”, CVPR 2022.]

Today's Program

Title	Speaker	Time
Introduction	Arash	10 min
Part (1): Denoising Diffusion Probabilistic Models	Arash	35 min
Part (2): Score-based Generative Modeling with Differential Equations	Karsten	45 min
Part (3): Advanced Techniques: Accelerated Sampling, Conditional Generation, and Beyond	Ruiqi	45 min
Applications (1): Image Synthesis, Text-to-Image, Controllable Generation	Ruiqi	15 min
Applications (2): Image Editing, Image-to-Image, Super-resolution, Segmentation	Arash	15 min
Applications (3): Video Synthesis, Medical Imaging, 3D Generation, Discrete State Models	Karsten	15 min
Conclusions, Open Problems and Final Remarks	Arash	10 min

cvpr2022-tutorial-diffusion-models.github.io

Applications (2):
Image Editing, Image-to-Image, Super-resolution, Segmentation

Super-Resolution

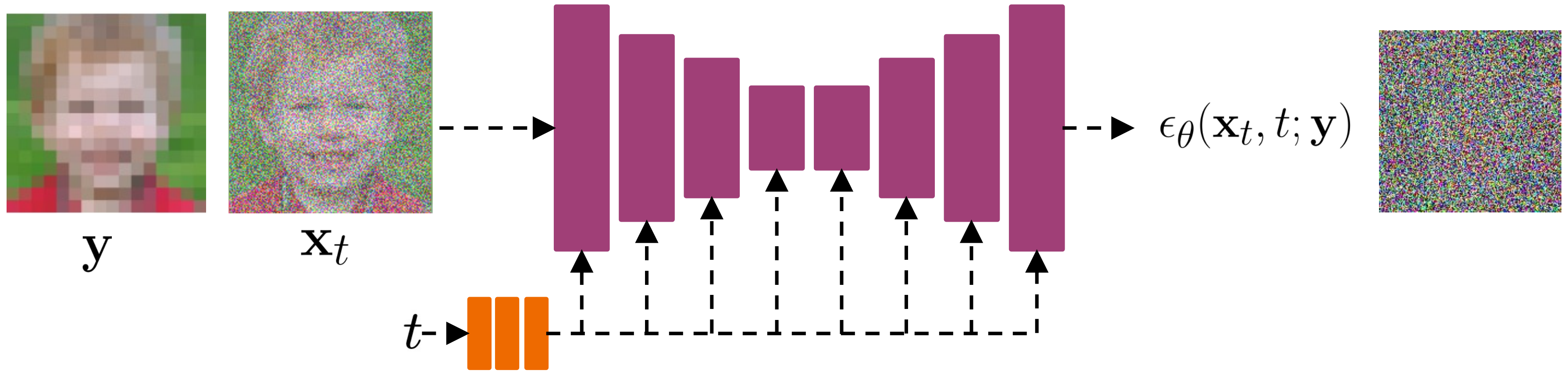
Super-Resolution via Repeated Refinement (SR3)

Image super-resolution can be considered as training $p(\mathbf{x}|\mathbf{y})$ where $\mathbf{y}$ is a low-resolution image and $\mathbf{x}$ is the corresponding high-resolution image

Train a score model for $\mathbf{x}$ conditioned on $\mathbf{y}$ using:

$$\mathbb{E}_{\mathbf{x},\mathbf{y}}\, \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\, \mathbb{E}_t\, ||\epsilon_\theta(\mathbf{x}_t, t; \mathbf{y}) - \epsilon||_p^p$$

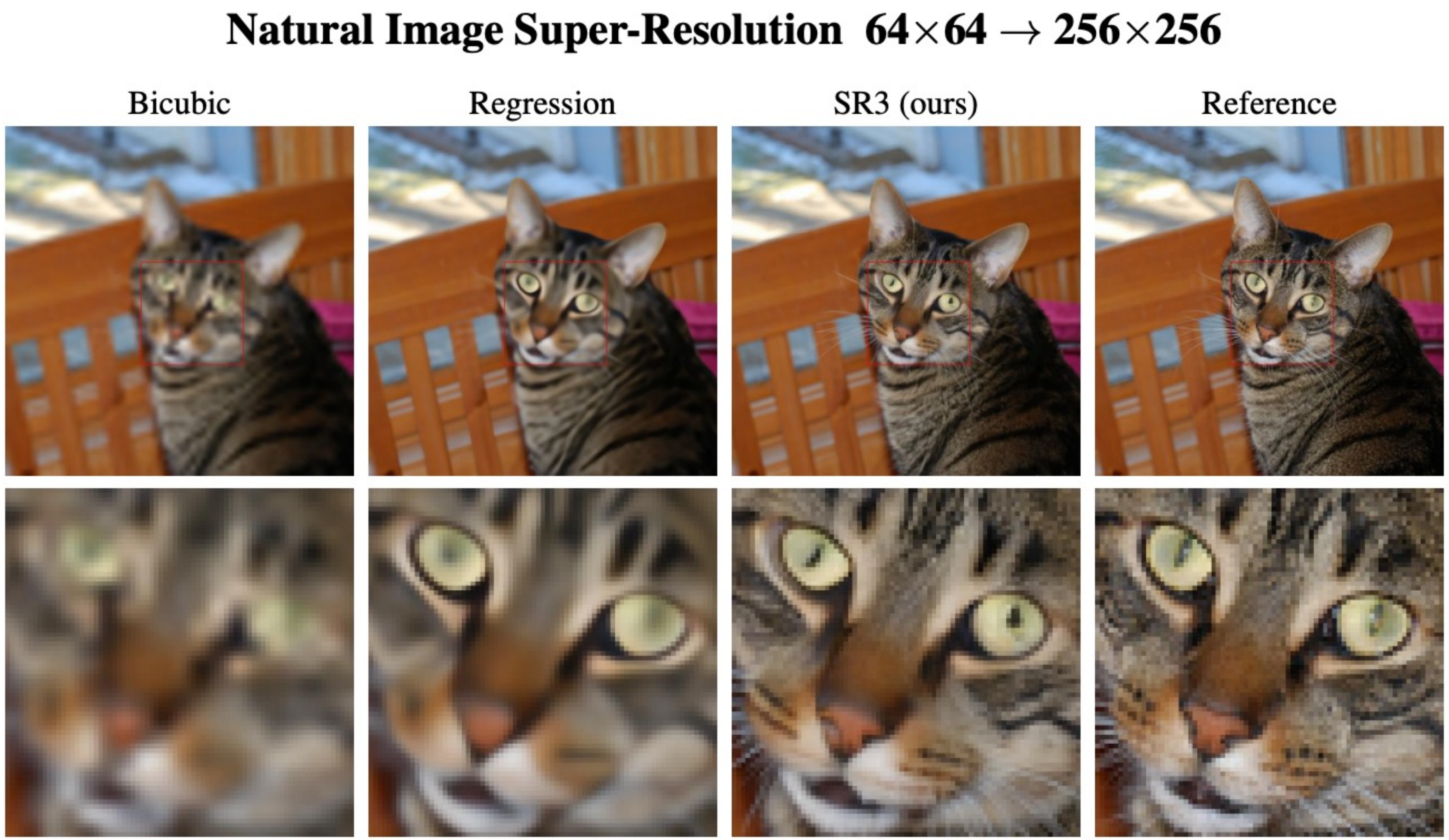
The conditional score is simply a U-Net with $\mathbf{x}_t$ and $\mathbf{y}$ (resolution image) concatenated.



Saharia et al., Image Super-Resolution via Iterative Refinement, 2021

Super-Resolution

Super-Resolution via Repeated Refinement (SR3)



[Saharia et al., Image Super-Resolution via Iterative Refinement, 2021]

Image-to-Image Translation

Palette: Image-to-Image Diffusion Models

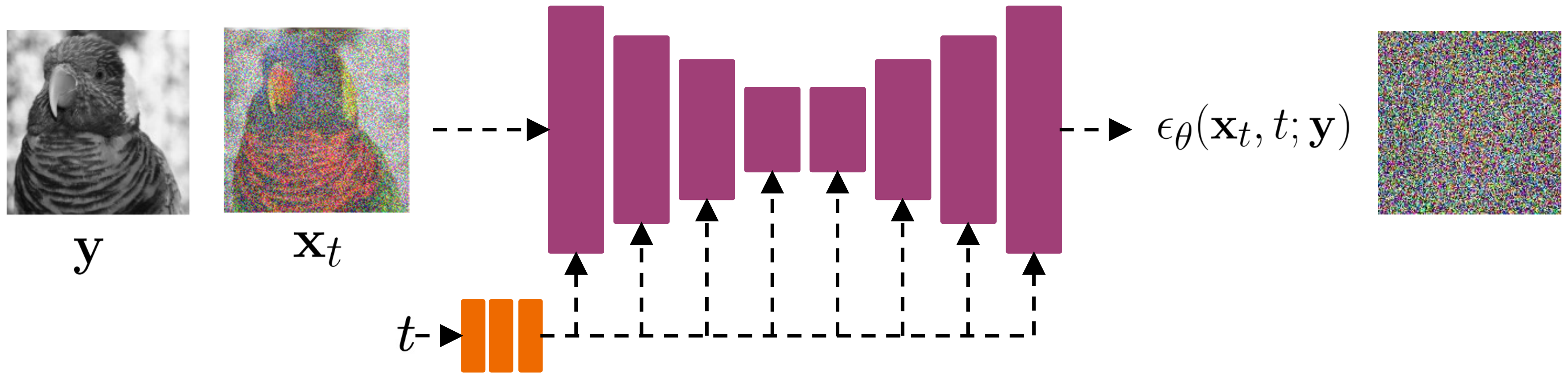
Many image-to-image translation applications can be considered as training $p(\mathbf{x}|\mathbf{y})$ where $\mathbf{y}$ is the input image.

For example, for colorization, $\mathbf{x}$ is a colored image and $\mathbf{y}$ is a gray-level image.

Train a score model for $\mathbf{x}$ conditioned on $\mathbf{y}$ using:

$$\mathbb{E}_{\mathbf{x},\mathbf{y}}\, \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\, \mathbb{E}_t\, ||\epsilon_\theta(\mathbf{x}_t, t; \mathbf{y}) - \epsilon||_p^p$$

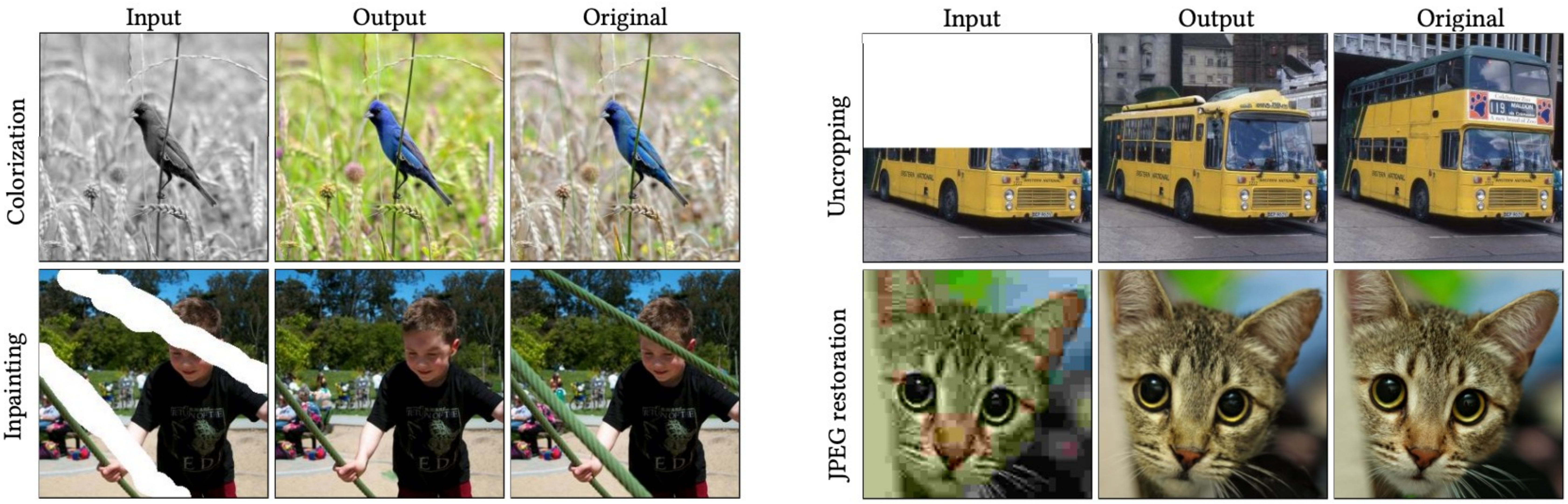
The conditional score is simply a U-Net with $\mathbf{x}_t$ and $\mathbf{y}$ concatenated.



Saharia et al., Palette: Image-to-Image Diffusion Models, 2022

Image-to-Image Translation

Palette: Image-to-Image Diffusion Models



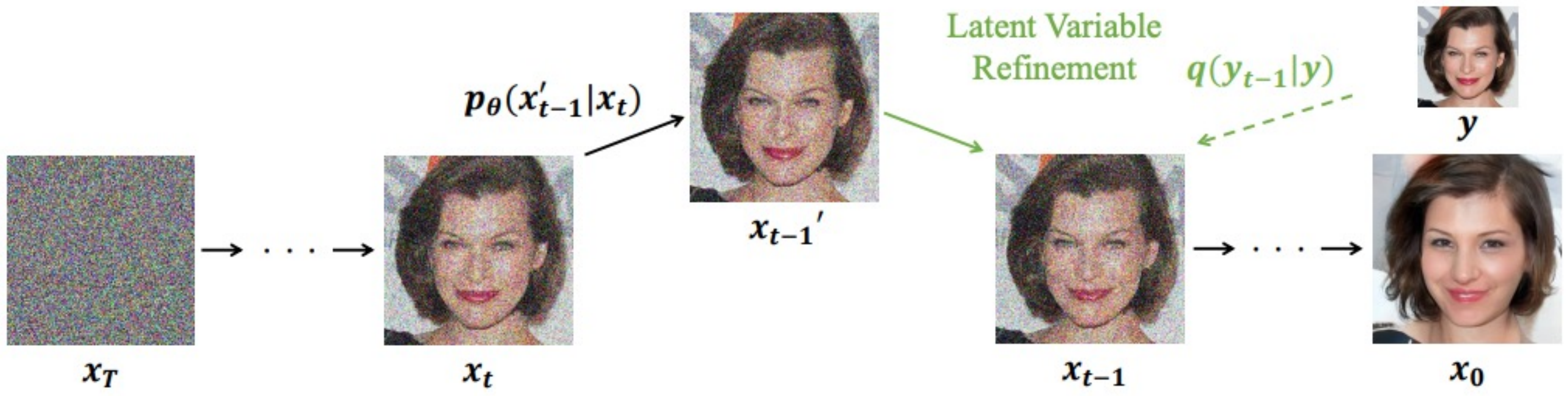
Saharia et al., Palette: Image-to-Image Diffusion Models, 2022

Conditional Generation

Iterative Latent Variable Refinement (ILVR)

A simple technique to guide the generation process of an unconditional diffusion model using a reference image.

Given the conditioning (reference) image y the generation process is modified to pull the samples towards the reference image.

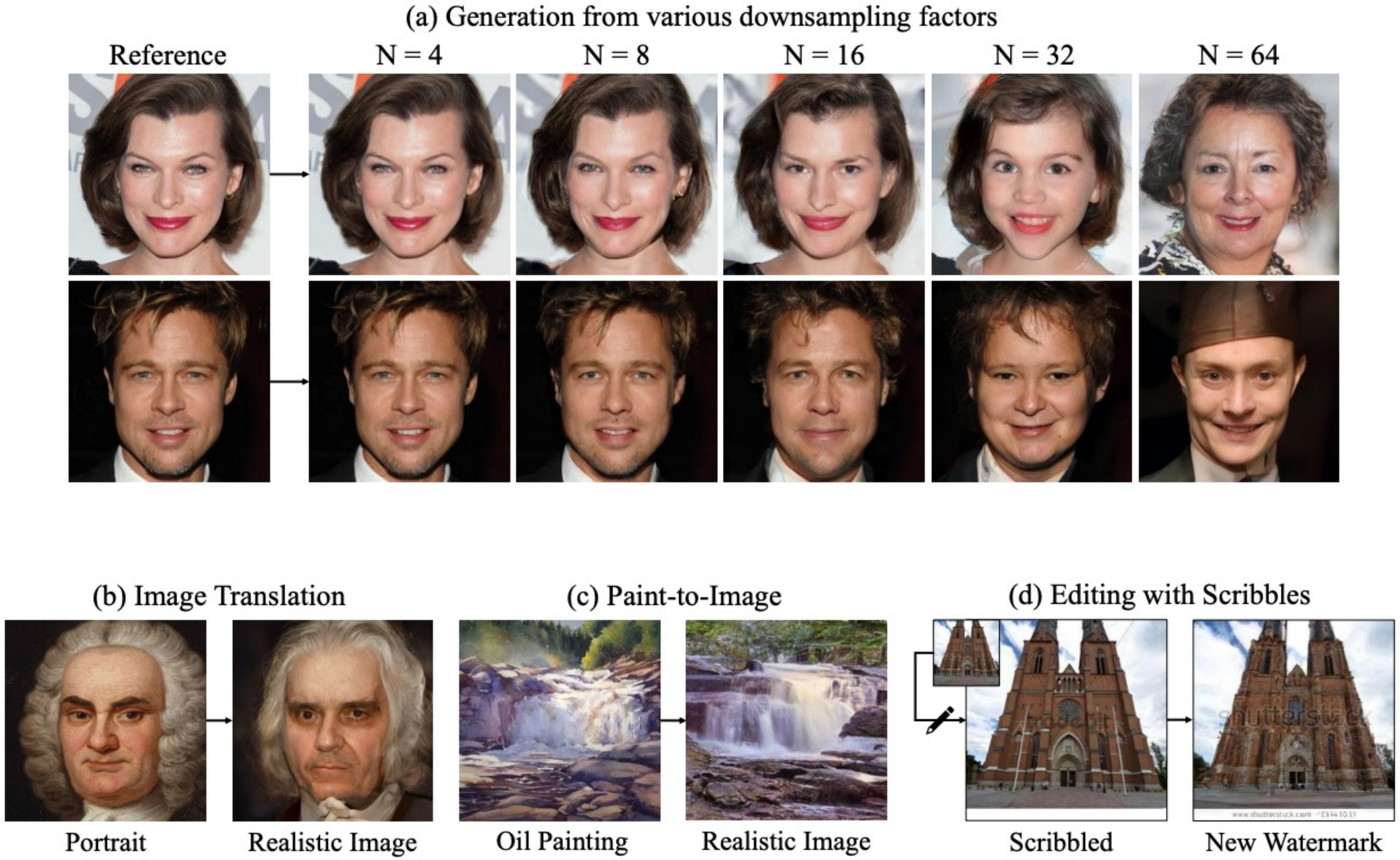


$$
\begin{aligned}
&\textbf{for } t = T, ..., 1 \textbf{ do} \\
&\quad \mathbf{z} \sim N(\mathbf{0}, \mathbf{I}) \\
&\quad x'_{t-1} \sim p_\theta(x'_{t-1}|x_t) && \triangleright \text{ unconditional proposal} \\
&\quad y_{t-1} \sim q(y_{t-1}|y) && \triangleright \text{ condition encoding} \\
&\quad x_{t-1} \leftarrow \phi_N(y_{t-1}) + x'_{t-1} - \phi_N(x'_{t-1}) \\
&\textbf{end for}
\end{aligned}
$$

Choi et al., ILVR: Conditioning Method for Denoising Diffusion Probabilistic Models, ICCV 2021

Conditional Generation

Iterative Latent Variable Refinement (ILVR)

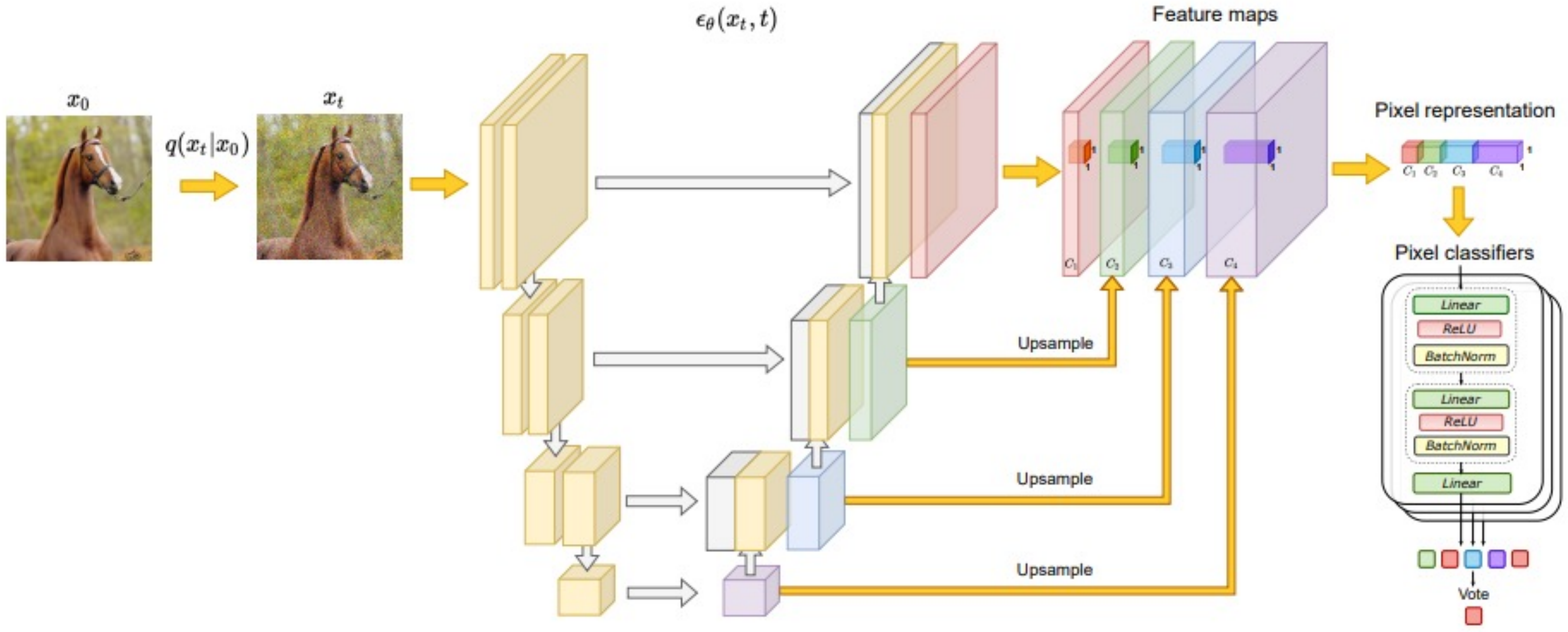


[Choi et al., ILVR: Conditioning Method for Denoising Diffusion Probabilistic Models, ICCV 2021](#)

Semantic Segmentation

Label-efficient semantic segmentation with diffusion models

Can we use representation learned from diffusion models for downstream applications such as semantic segmentation?

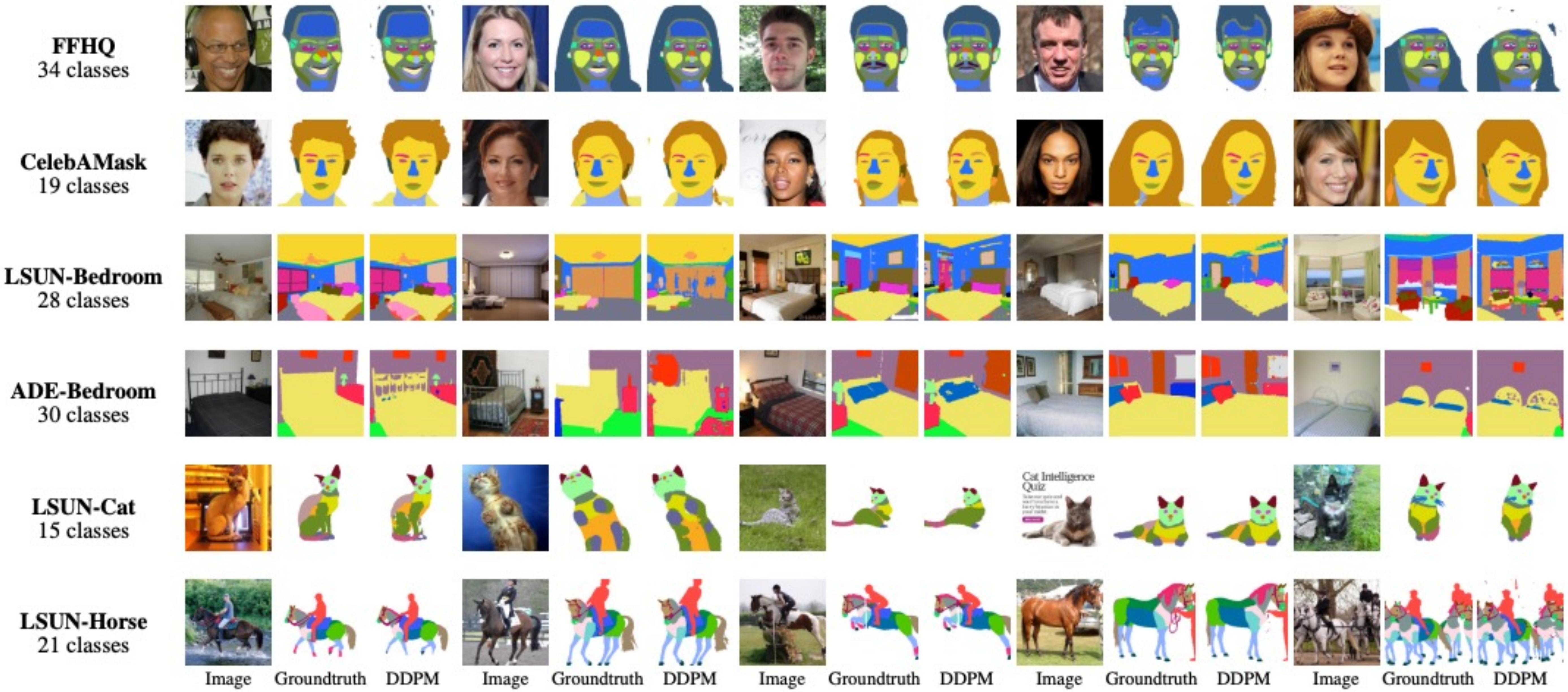


[Baranchuk et al., Label-Efficient Semantic Segmentation with Diffusion Models, ICLR 2022]

Semantic Segmentation

Label-efficient semantic segmentation with diffusion models

The experimental results show that the proposed method outperforms Masked Autoencoders, GAN and VAE-based models.

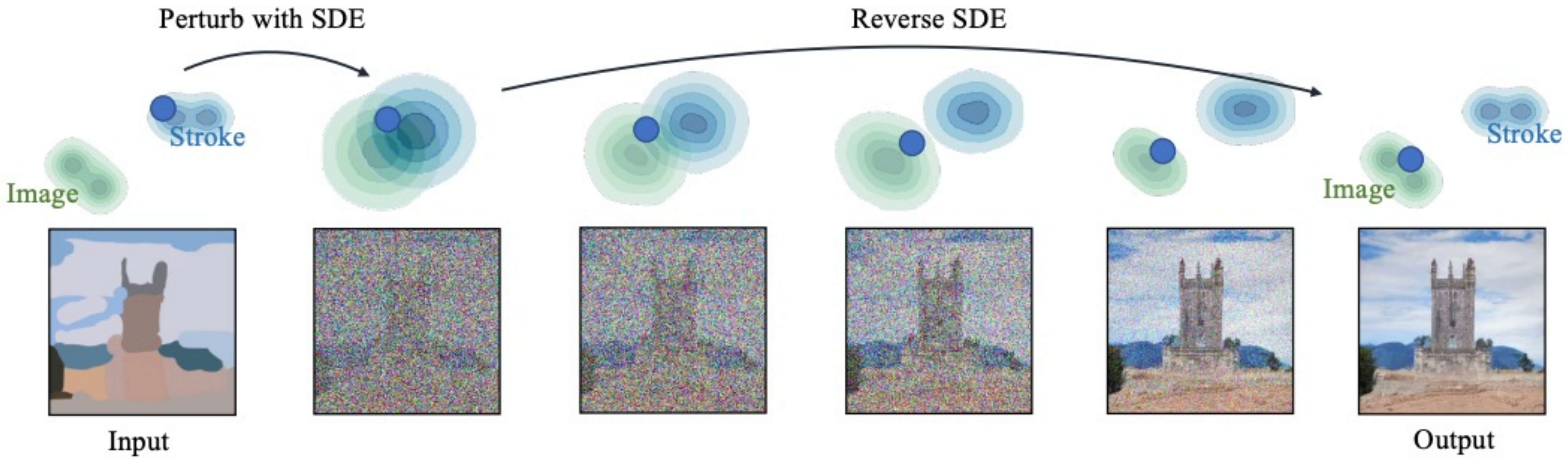


Baranchuk et al., Label-Efficient Semantic Segmentation with Diffusion Models, ICLR 2022

Image Editing

SDEdit

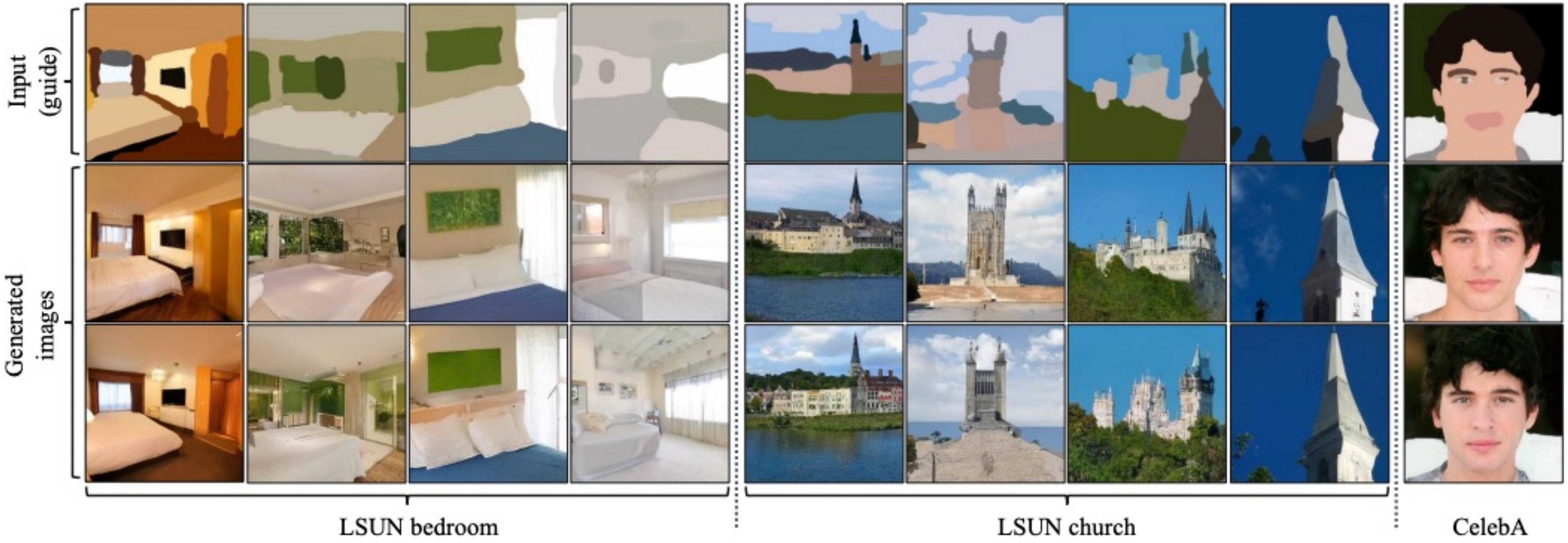
Forward diffusion brings two distributions close to each other



Meng et al., SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations, ICLR 2022

Image Editing

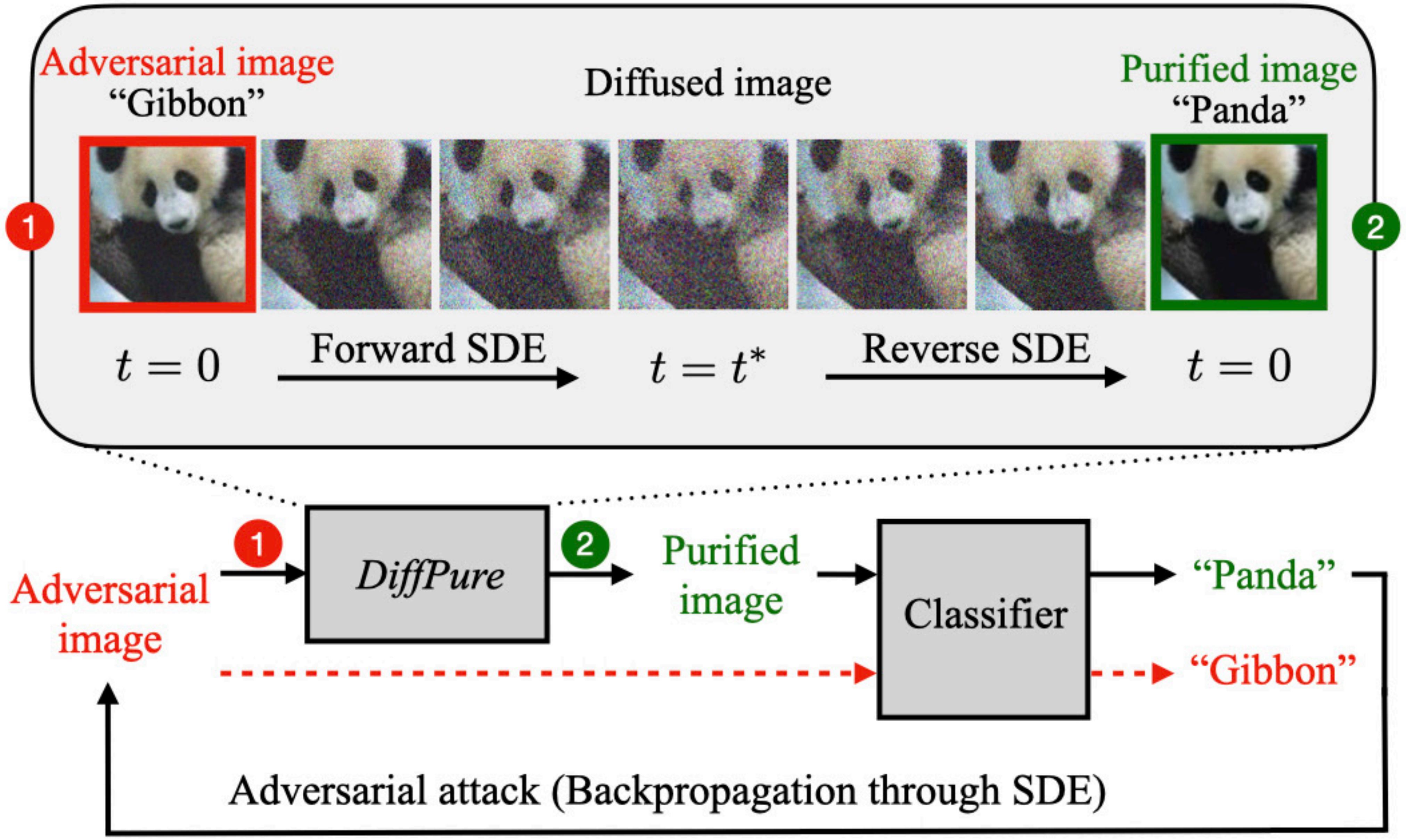
SDEdit



[Meng et al., SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations, ICLR 2022]

Adversarial Robustness

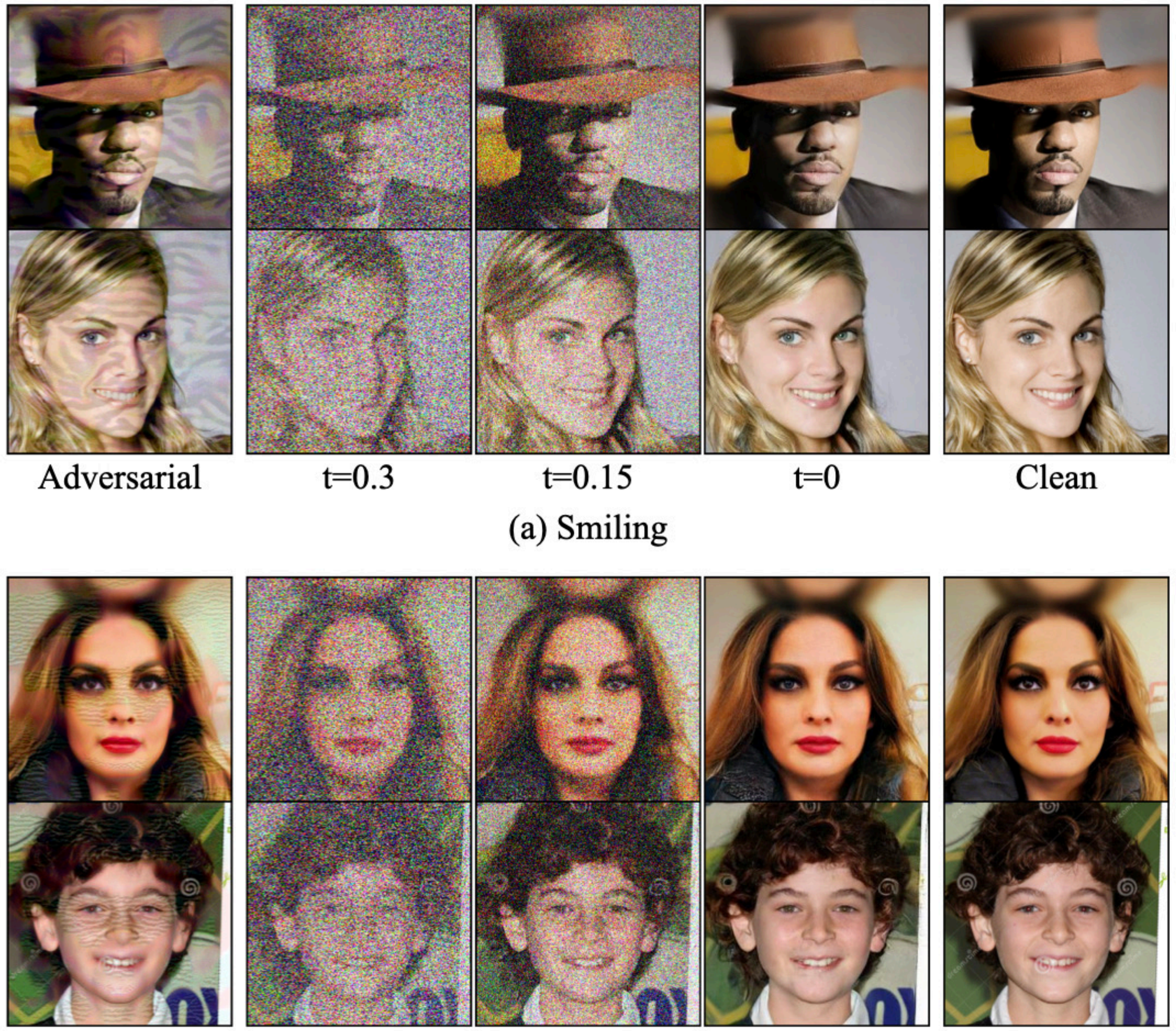
Diffusion Models for Adversarial Purification



Nie et al., Diffusion Models for Adversarial Purification, ICML 2022

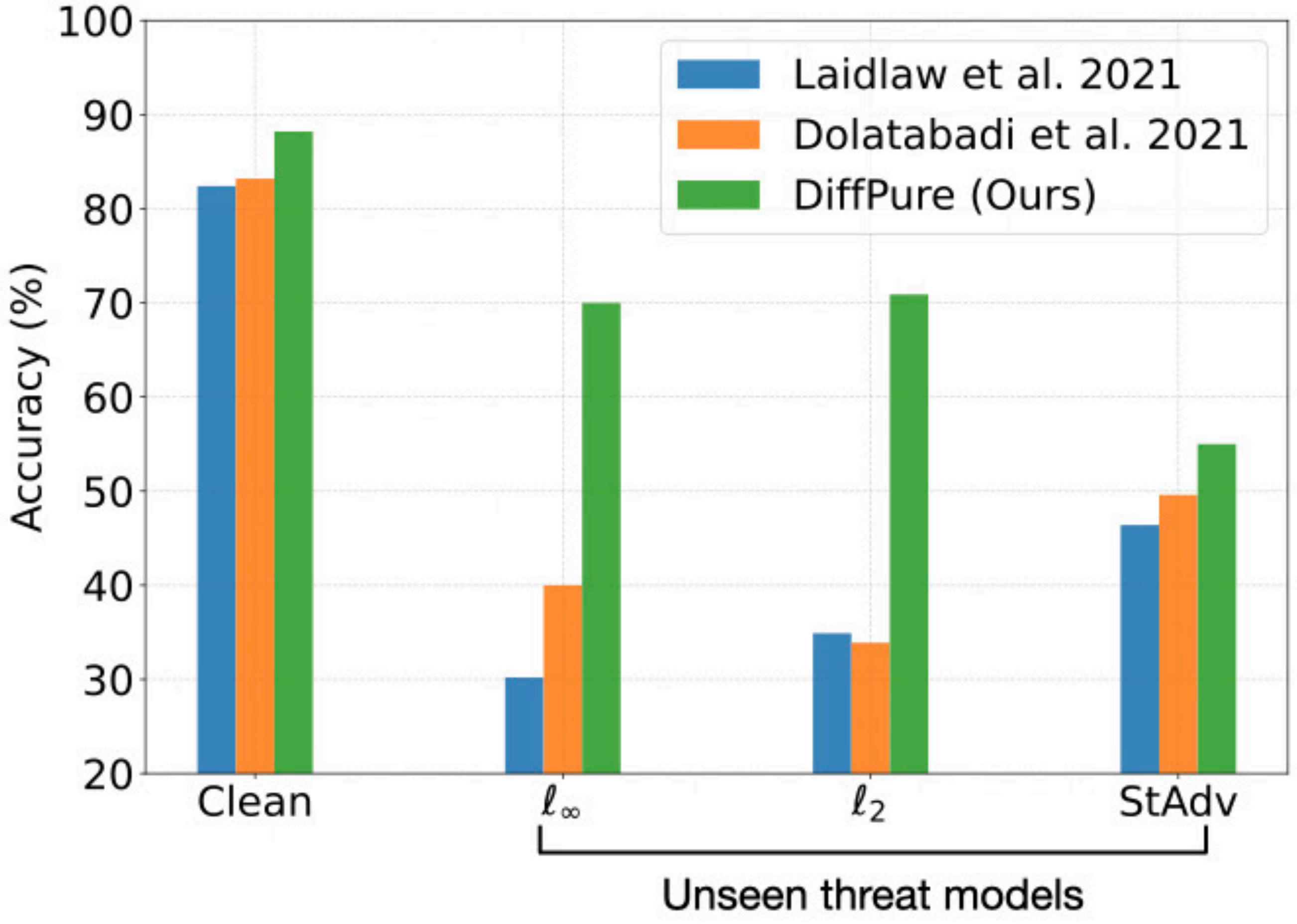
Adversarial Robustness

Diffusion Models for Adversarial Purification



(a) Smiling

(b) Eyeglasses



Comparison with state-of-the-art defense methods against unseen threat models (including AutoAttack ℓ_∞, AutoAttack ℓ_2, and StdAdv) on ResNet-50 for CIFAR-10.

Nie et al., Diffusion Models for Adversarial Purification, ICML 2022

Today's Program

Title	Speaker	Time
Introduction	Arash	10 min
Part (1): Denoising Diffusion Probabilistic Models	Arash	35 min
Part (2): Score-based Generative Modeling with Differential Equations	Karsten	45 min
Part (3): Advanced Techniques: Accelerated Sampling, Conditional Generation, and Beyond	Ruiqi	45 min
Applications (1): Image Synthesis, Text-to-Image, Controllable Generation	Ruiqi	15 min
Applications (2): Image Editing, Image-to-Image, Super-resolution, Segmentation	Arash	15 min
Applications (3): Video Synthesis, Medical Imaging, 3D Generation, Discrete State Models	Karsten	15 min
Conclusions, Open Problems and Final Remarks	Arash	10 min

cvpr2022-tutorial-diffusion-models.github.io

Applications (3):
Video Synthesis, Medical Imaging, 3D Generation, Discrete State Models

Video Generation

Samples from a text-conditioned video diffusion model, conditioned on the string *fireworks*.

(video from: Ho et al., “Video Diffusion Models”, *arXiv*, 2022, https://video-diffusion.github.io/)

Ho et al., “Video Diffusion Models”, *arXiv*, 2022
Harvey et al., “Flexible Diffusion Modeling of Long Videos”, arXiv, 2022
Yang et al., “Diffusion Probabilistic Modeling for Video Generation”, *arXiv*, 2022
Höppe et al., “Diffusion Models for Video Prediction and Infilling”, arXiv, 2022
Voleti et al., “MCVD: Masked Conditional Video Diffusion for Prediction, Generation, and Interpolation”, *arXiv*, 2022

Video Generation

Video Generation Tasks:

- Unconditional Generation (Generate all frames)
- Future Prediction (Generate future from past fames)
- Past Prediction (Generate past from future fames)
- Interpolation (Generate intermediate frames)

➡ Learn a model of the form:

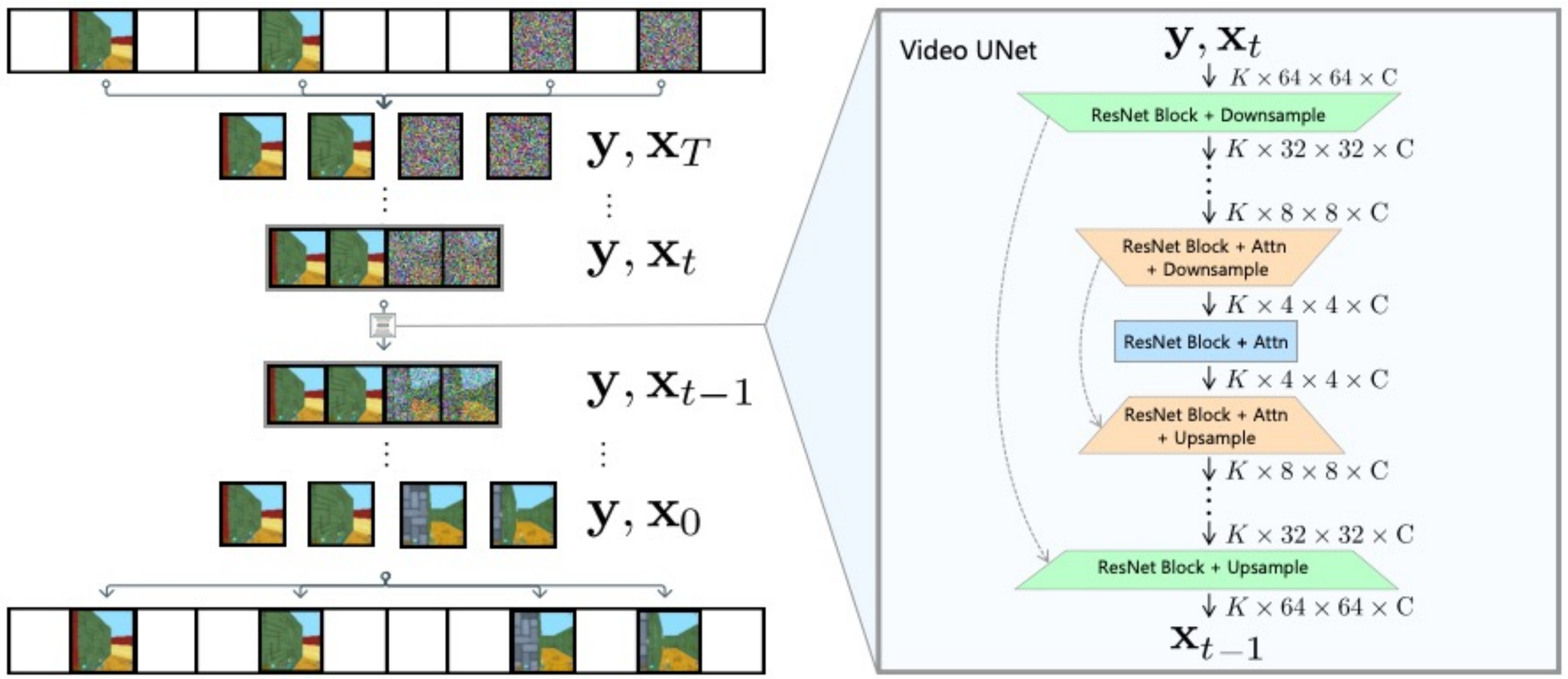
$$p_{\boldsymbol{\theta}}(\mathbf{x}^{t_1}, \cdots, \mathbf{x}^{t_K} | \mathbf{x}^{\tau_1}, \cdots, \mathbf{x}^{\tau_M})$$

Given frames: $\mathbf{x}^{\tau_1}, \cdots, \mathbf{x}^{\tau_M}$

Frames to be predicted: $\mathbf{x}^{t_1}, \cdots, \mathbf{x}^{t_K}$

➡ Learn one model for everything:

- Architecture as **one diffusion model** over **all frames concatenated.**
- Mask frames to be predicted; provide conditioning frames; vary applied masking/conditioning for different tasks during training.
- Use **time position encodings** to encode times.



(image from: Harvey et al., “Flexible Diffusion Modeling of Long Videos”, *arXiv*, 2022)

Ho et al., “Video Diffusion Models”, *arXiv*, 2022
Harvey et al., “Flexible Diffusion Modeling of Long Videos”, arXiv, 2022
Yang et al., “Diffusion Probabilistic Modeling for Video Generation”, *arXiv*, 2022
Höppe et al., “Diffusion Models for Video Prediction and Infilling”, arXiv, 2022
Voleti et al., “MCVD: Masked Conditional Video Diffusion for Prediction, Generation, and Interpolation”, *arXiv*, 2022

Video Generation

Architecture Details

Architecture Details:

Data is 4D (image height, image width, #frames, channels)

- Option (1): 3D Convolutions. Can be computationally expensive.

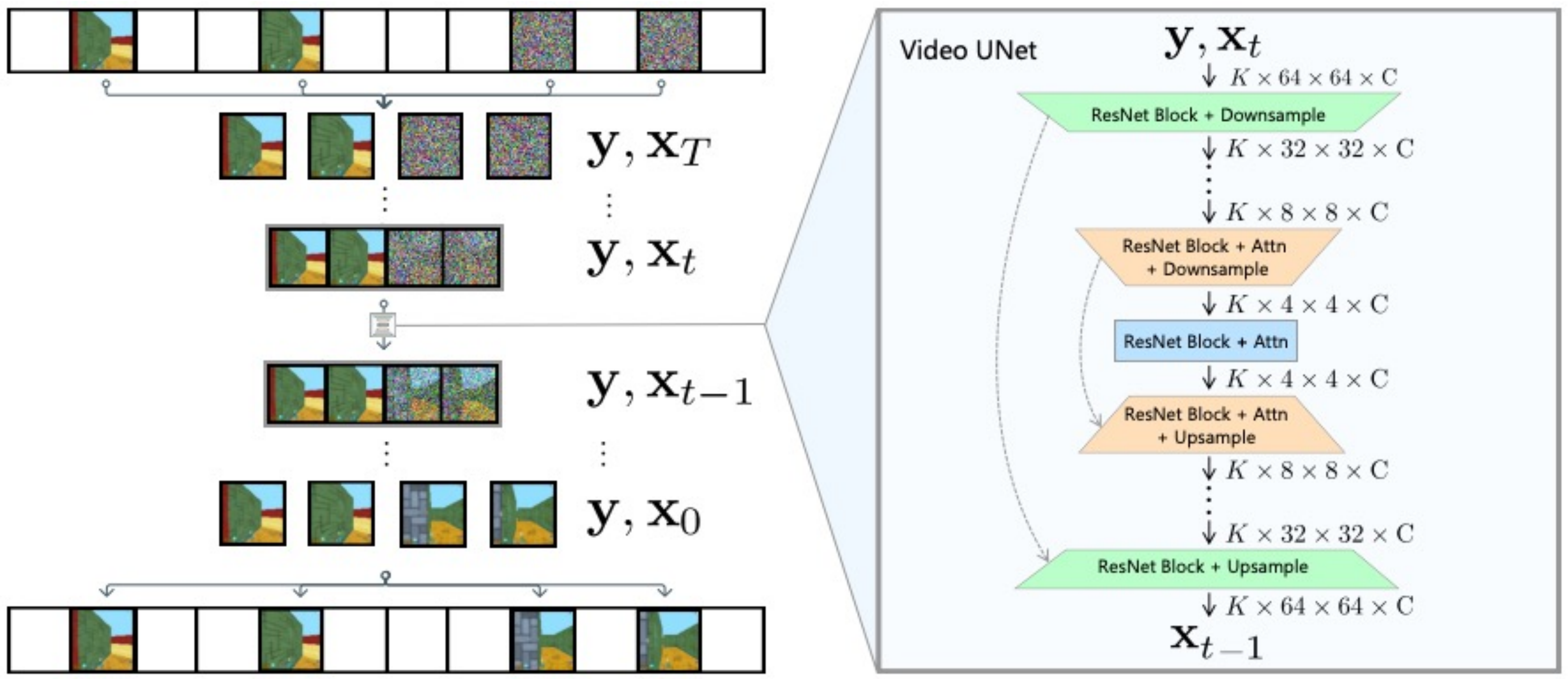
- Option (2): Spatial 2D Convolutions + Attention Layers along frame axis.

➡ Additional Advantage:

Ignoring the attention layers, the model can be trained additionally on pure image data!

➡ Learn one model for everything:

- Architecture as **one diffusion model** over **all frames concatenated.**
- Mask frames to be predicted; provide conditioning frames; vary applied masking/conditioning for different tasks during training.
- Use **time position encodings** to encode times.



(image from: Harvey et al., "Flexible Diffusion Modeling of Long Videos", *arXiv*, 2022)

Ho et al., "Video Diffusion Models", *arXiv*, 2022
Harvey et al., "Flexible Diffusion Modeling of Long Videos", arXiv, 2022
Yang et al., "Diffusion Probabilistic Modeling for Video Generation", *arXiv*, 2022
Höppe et al., "Diffusion Models for Video Prediction and Infilling", arXiv, 2022
Voleti et al., "MCVD: Masked Conditional Video Diffusion for Prediction, Generation, and Interpolation", *arXiv*, 2022

Video Generation

Results

Long term video generation in hierarchical manner:

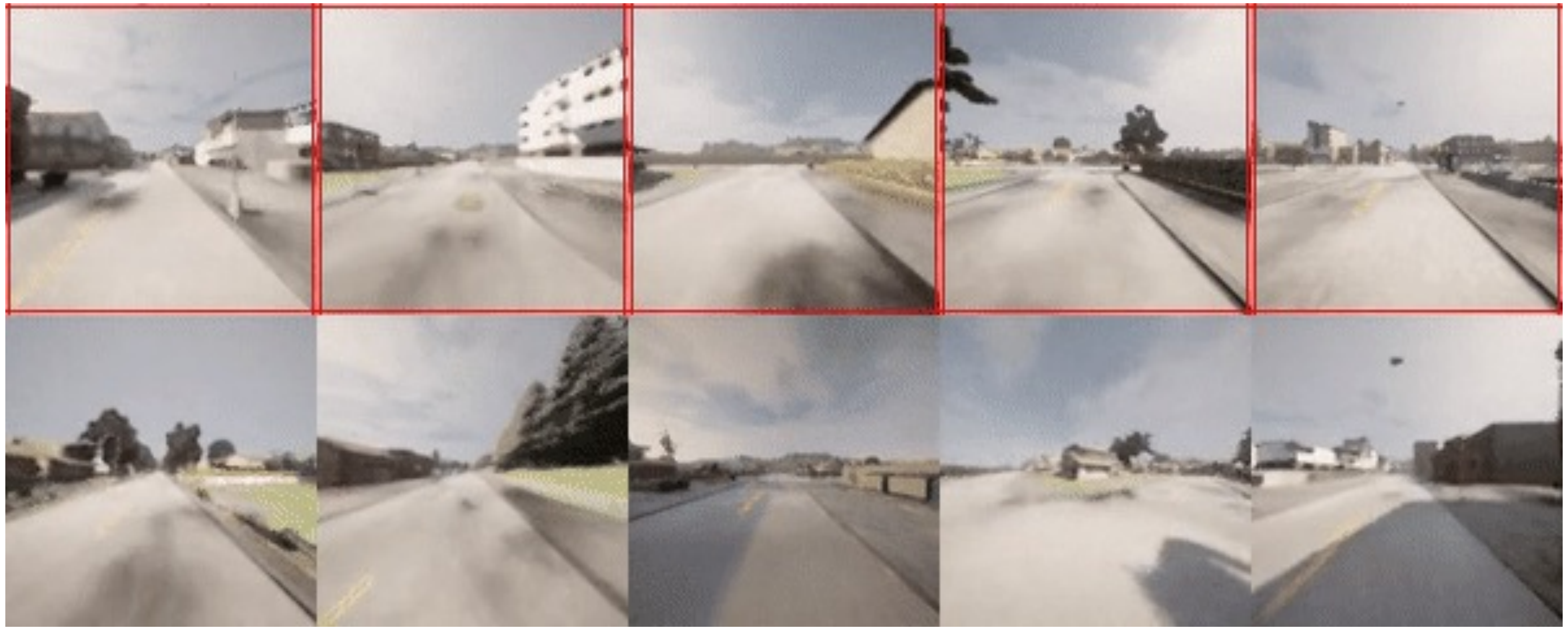
- 1. Generate future frames in sparse manner, conditioning on frames far back
- 2. Interpolate in-between frames

⬇

1+ hour coherent video generation possible!

Test Data:

Generated:

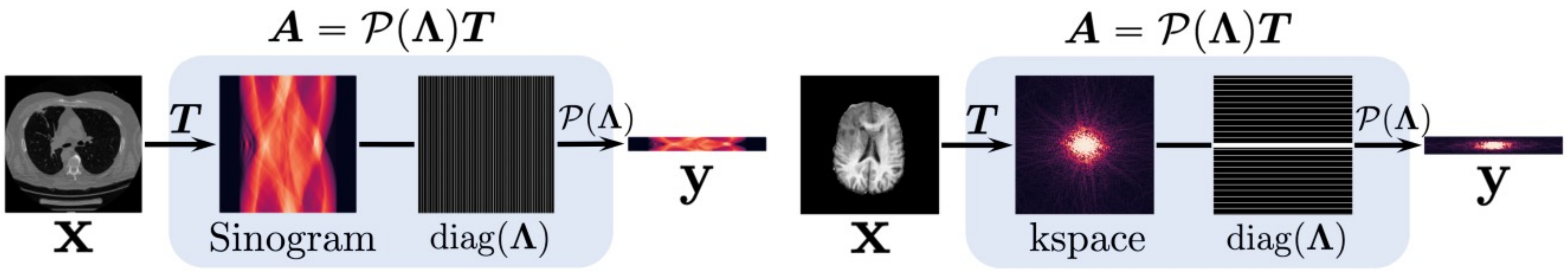


(video from: Harvey et al., “Flexible Diffusion Modeling of Long Videos”, *arXiv*, 2022, https://plai.cs.ubc.ca/2022/05/20/flexible-diffusion-modeling-of-long-videos/)

Ho et al., “Video Diffusion Models”, *arXiv*, 2022
Harvey et al., “Flexible Diffusion Modeling of Long Videos”, arXiv, 2022
Yang et al., “Diffusion Probabilistic Modeling for Video Generation”, *arXiv*, 2022
Höppe et al., “Diffusion Models for Video Prediction and Infilling”, arXiv, 2022
Voleti et al., “MCVD: Masked Conditional Video Diffusion for Prediction, Generation, and Interpolation”, *arXiv*, 2022

Solving Inverse Problems in Medical Imaging

Forward CT or MRI imaging process (simplified):



sparse-view CT

undersampled MRI

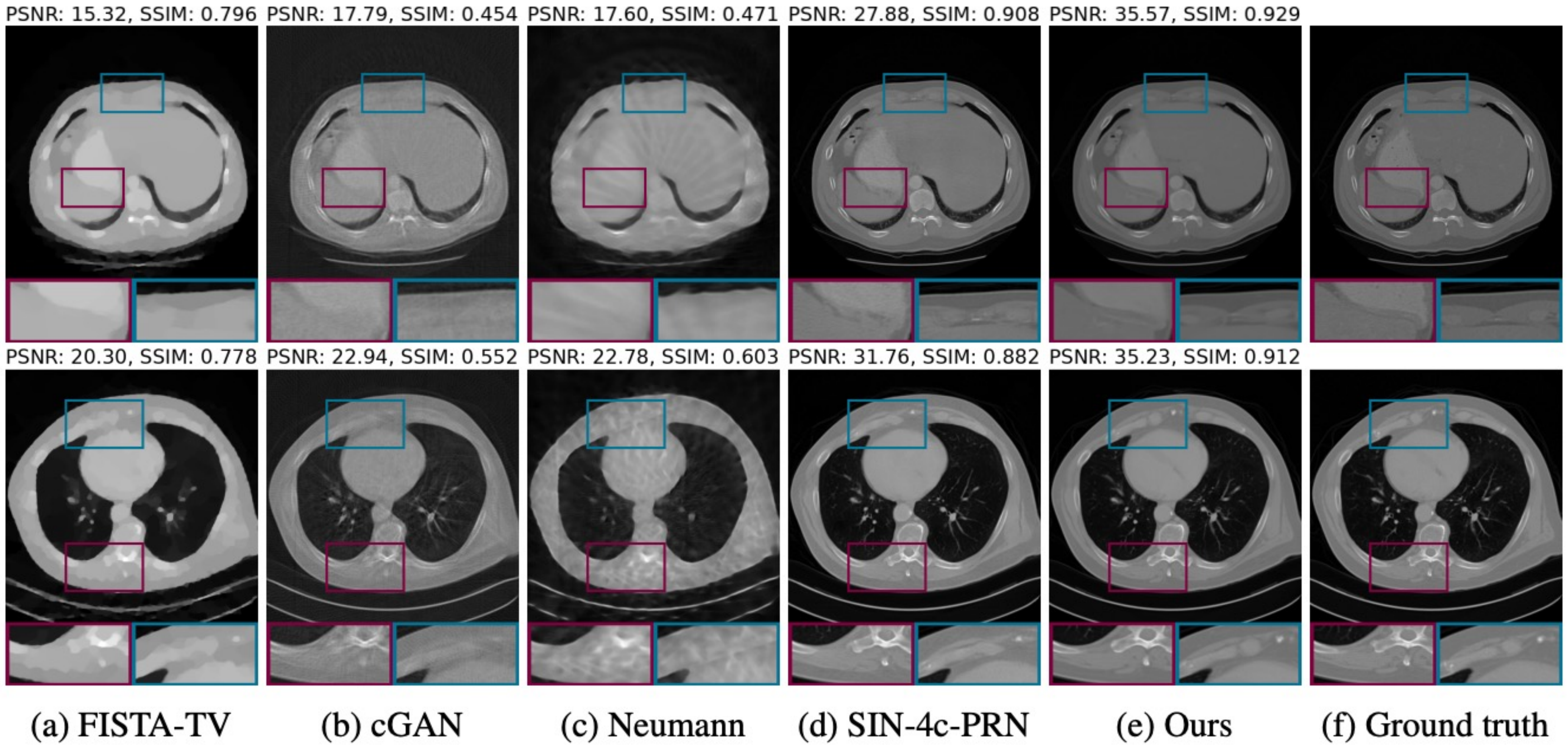
(image from: Song et al., "Solving Inverse Problems in Medical Imaging with Score-Based Generative Models", *ICLR*, 2022)

➡ *Inverse Problem:*
Reconstruct original image from sparse measurements.

Song et al., "Solving Inverse Problems in Medical Imaging with Score-Based Generative Models", *ICLR*, 2022

Solving Inverse Problems in Medical Imaging

High-level idea: Learn Generative Diffusion Model as "prior"; then guide synthesis conditioned on sparse observations:



(image from: Song et al., "Solving Inverse Problems in Medical Imaging with Score-Based Generative Models", *ICLR*, 2022)

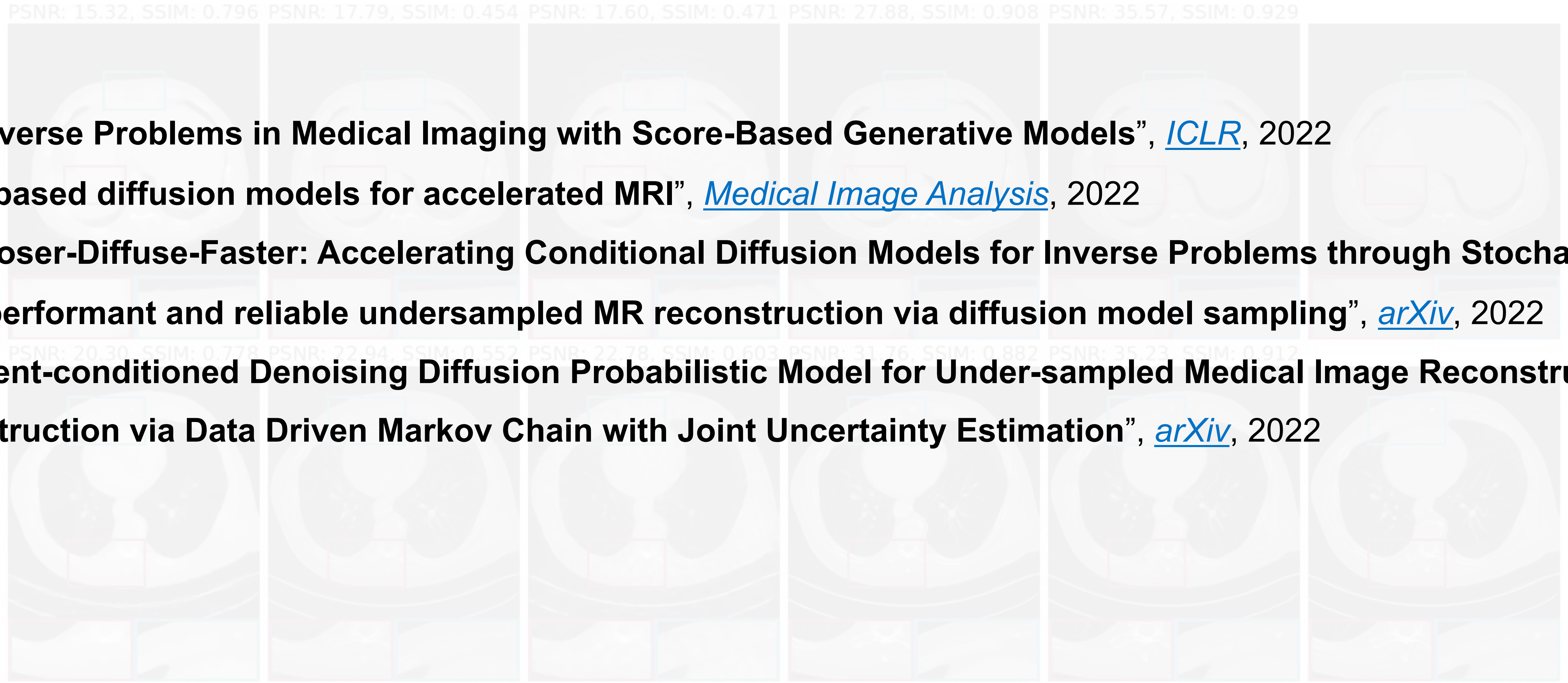
➡ *Outperforms even fully-supervised methods.*

Song et al., "Solving Inverse Problems in Medical Imaging with Score-Based Generative Models", *ICLR*, 2022

Solving Inverse Problems in Medical Imaging

Lots of Literature

High-level idea: Learn Generative Diffusion Model as "prior"; then guide synthesis conditioned on sparse observations:

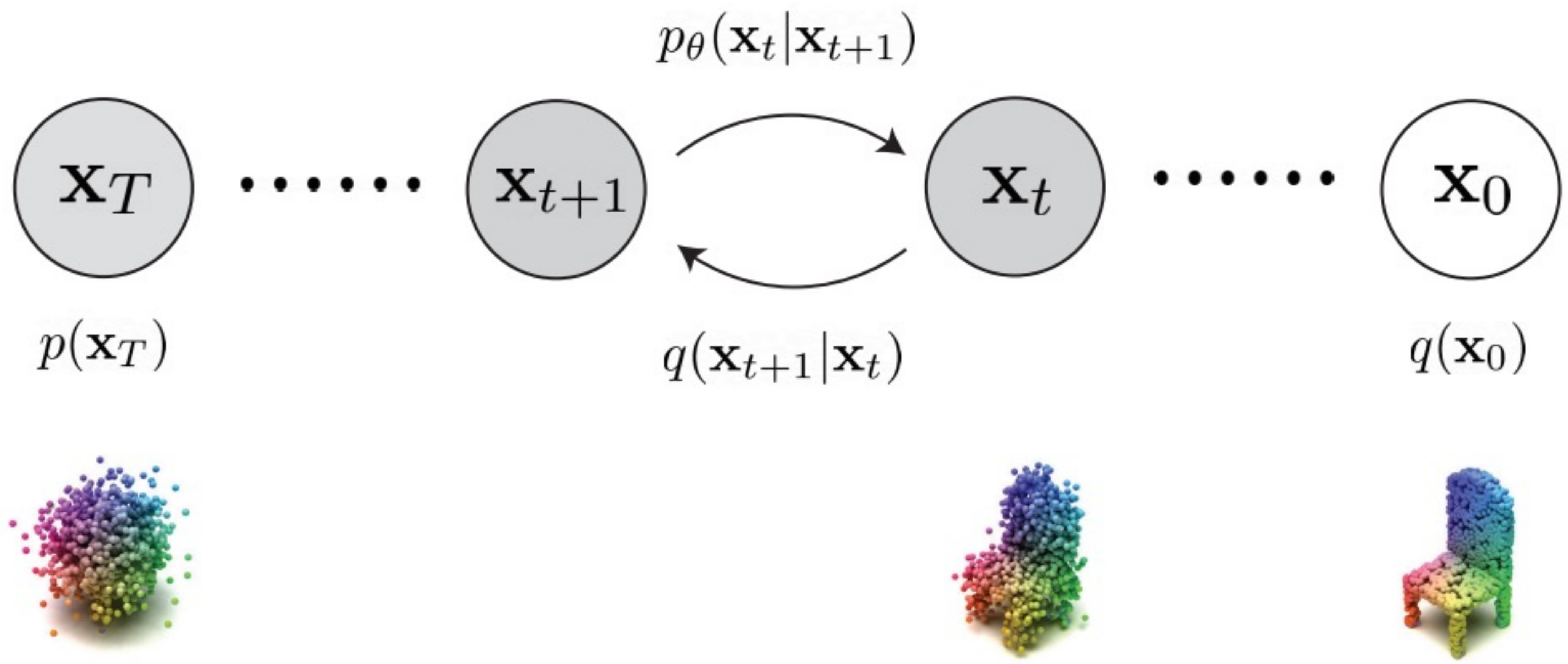


- Song et al., "**Solving Inverse Problems in Medical Imaging with Score-Based Generative Models**", *ICLR*, 2022
- Chung and Ye, "**Score-based diffusion models for accelerated MRI**", *Medical Image Analysis*, 2022
- Chung et al., "**Come-Closer-Diffuse-Faster: Accelerating Conditional Diffusion Models for Inverse Problems through Stochastic Contraction**", *CVPR*, 2022
- Peng et al., "**Towards performant and reliable undersampled MR reconstruction via diffusion model sampling**", *arXiv*, 2022
- Xie and Li, "**Measurement-conditioned Denoising Diffusion Probabilistic Model for Under-sampled Medical Image Reconstruction**", *arXiv*, 2022
- Luo et al, "**MRI Reconstruction via Data Driven Markov Chain with Joint Uncertainty Estimation**", *arXiv*, 2022
- …

(Song et al., "Solving Inverse Problems in Medical Imaging with Score-Based Generative Models", *ICLR*, 2022)

3D Shape Generation

- Point clouds as 3D shape representation can be diffused easily and intuitively
- Denoiser implemented based on modern point cloud-processing networks (PointNets & Point-VoxelCNNs)

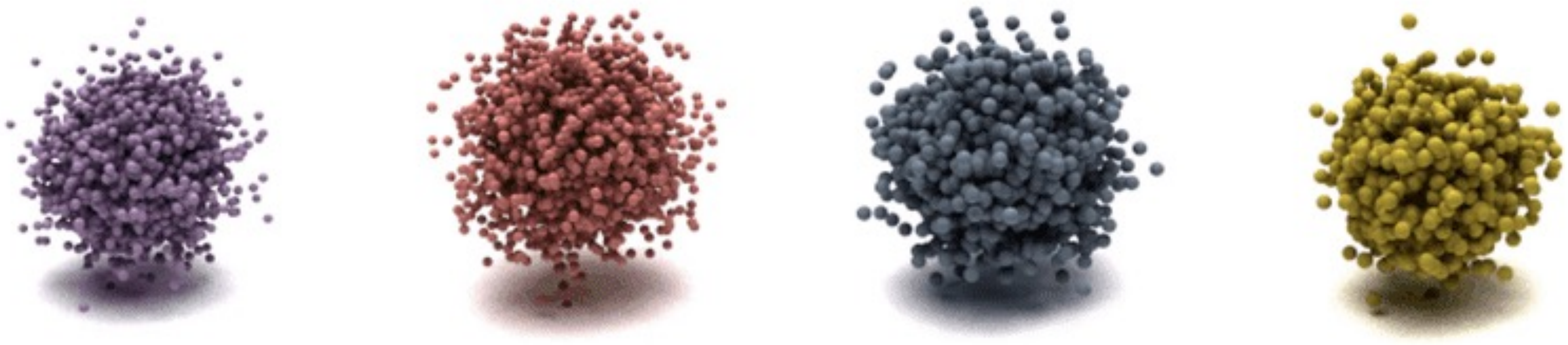


(image from: Zhou et al., "3D Shape Generation and Completion through Point-Voxel Diffusion", *ICCV*, 2021)

[Zhou et al., "3D Shape Generation and Completion through Point-Voxel Diffusion", *ICCV*, 2021]
[Luo and Hu, "Diffusion Probabilistic Models for 3D Point Cloud Generation", *CVPR*, 2021]

3D Shape Generation

- Point clouds as 3D shape representation can be diffused easily and intuitively
- Denoiser implemented based on modern point cloud-processing networks (PointNets & Point-VoxelCNNs)



(video from: Zhou et al., "3D Shape Generation and Completion through Point-Voxel Diffusion", *ICCV*, 2021, https://alexzhou907.github.io/pvd)

[Zhou et al., "3D Shape Generation and Completion through Point-Voxel Diffusion", *ICCV*, 2021]

3D Shape Generation

Shape Completion

- Can train conditional shape completion diffusion model (subset of points fixed to given conditioning points):

(video from: Zhou et al., “3D Shape Generation and Completion through Point-Voxel Diffusion”, *ICCV*, 2021, https://alexzhou907.github.io/pvd)

Zhou et al., “3D Shape Generation and Completion through Point-Voxel Diffusion”, *ICCV*, 2021

3D Shape Generation

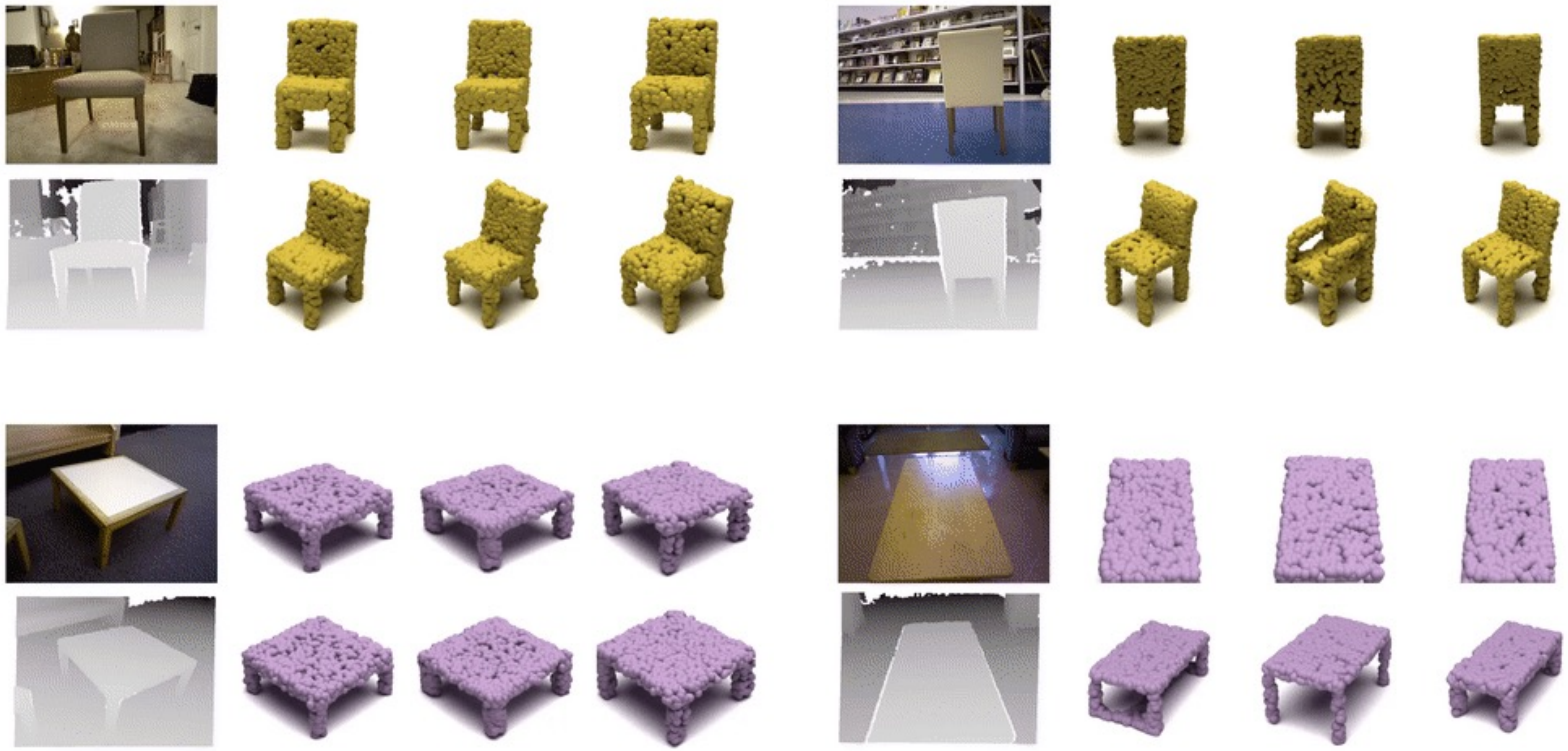
Shape Completion – Multimodality

(video from: Zhou et al., "3D Shape Generation and Completion through Point-Voxel Diffusion", *ICCV*, 2021, https://alexzhou907.github.io/pvd)

[Zhou et al., "3D Shape Generation and Completion through Point-Voxel Diffusion", *ICCV*, 2021]

3D Shape Generation

Shape Completion – Multimodality – On Real Data



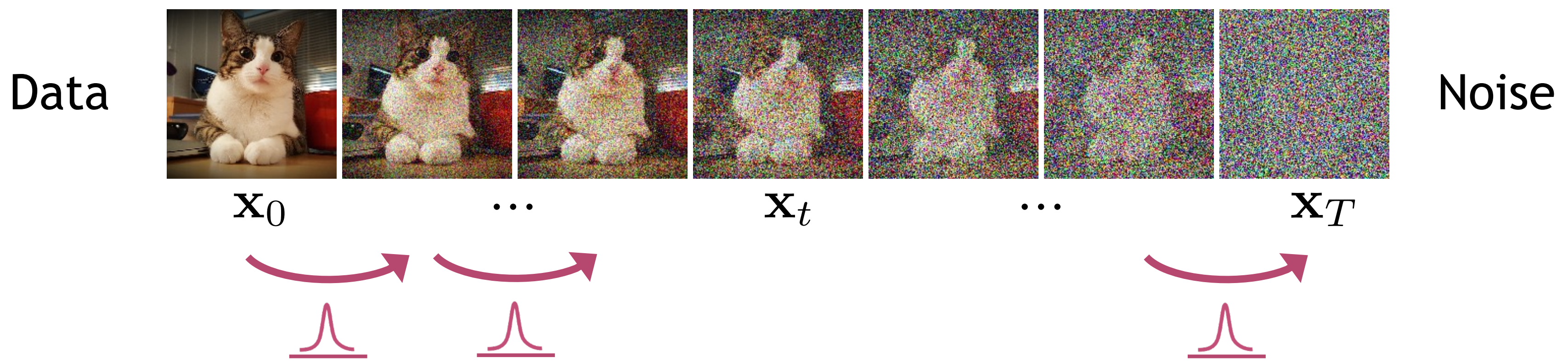
(video from: Zhou et al., “3D Shape Generation and Completion through Point-Voxel Diffusion”, *ICCV*, 2021, https://alexzhou907.github.io/pvd)

Zhou et al., “3D Shape Generation and Completion through Point-Voxel Diffusion”, *ICCV*, 2021

Towards Discrete State Diffusion Models

- So far:

  Continuous diffusion and denoising processes.



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

Reverse generative process: $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

➡ *But what if data is discrete? Categorical?*
Continuous perturbations are not possible!

(Text, Pixel-wise Segmentation Labels,
Discrete Image Encodings, etc.)

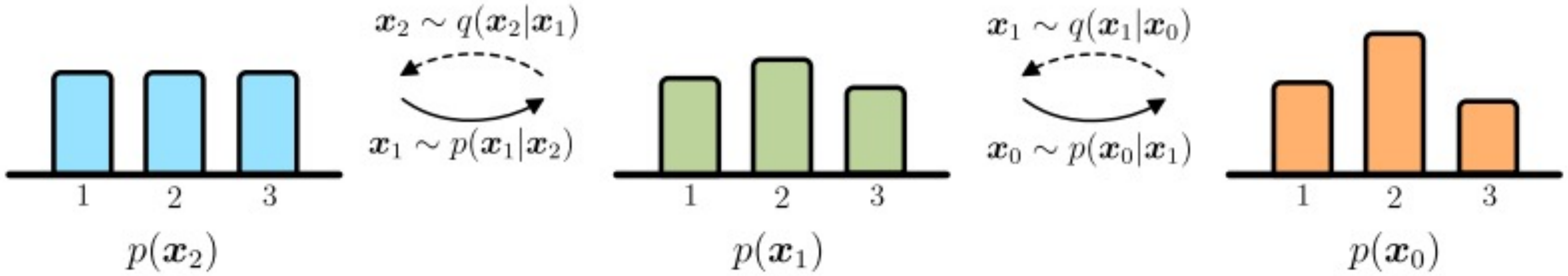
Discrete State Diffusion Models

- **Categorical diffusion:** $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{Cat}(\mathbf{x}_t; \mathbf{p} = \mathbf{x}_{t-1}\mathbf{Q}_t)$

  $\mathbf{x}_t$: one-hot state vector

  $\mathbf{Q}_t$: transition matrix $[\mathbf{Q}_t]_{ij} = q(x_t = j|x_{t-1} = i)$

- Reverse process can be parametrized categorical distribution.



(image from: Hoogeboom et al., “Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions”, *NeurIPS*, 2022)

Austin et al., “Structured Denoising Diffusion Models in Discrete State-Spaces”, *NeurIPS*, 2021
Hoogeboom et al., “Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions”, *NeurIPS*, 2022

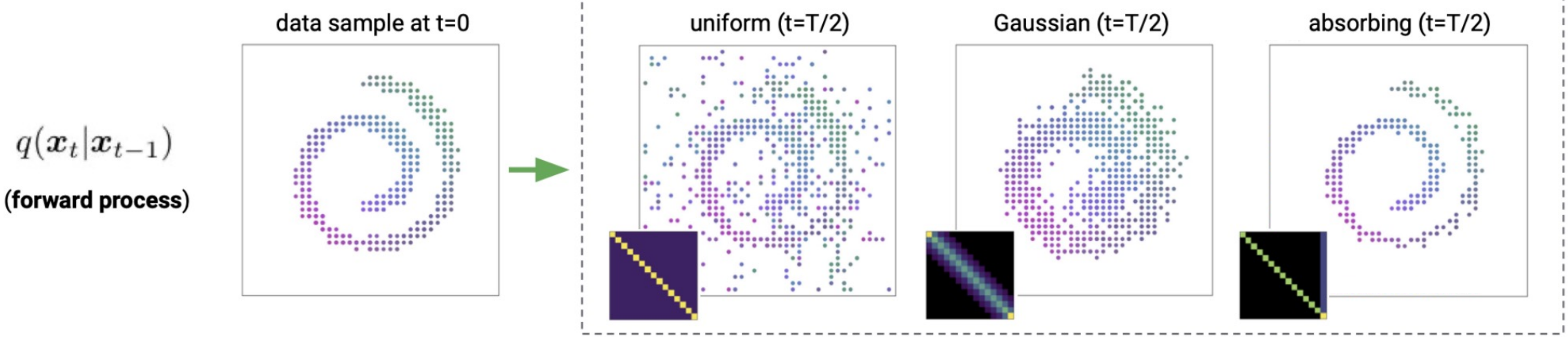
Discrete State Diffusion Models

Options for forward process:

- Uniform categorical diffusion:

$$\mathbf{Q}_t = (1-\beta_t)\mathbf{I} + \frac{\beta_t}{K}\mathbb{1}\mathbb{1}^\top$$

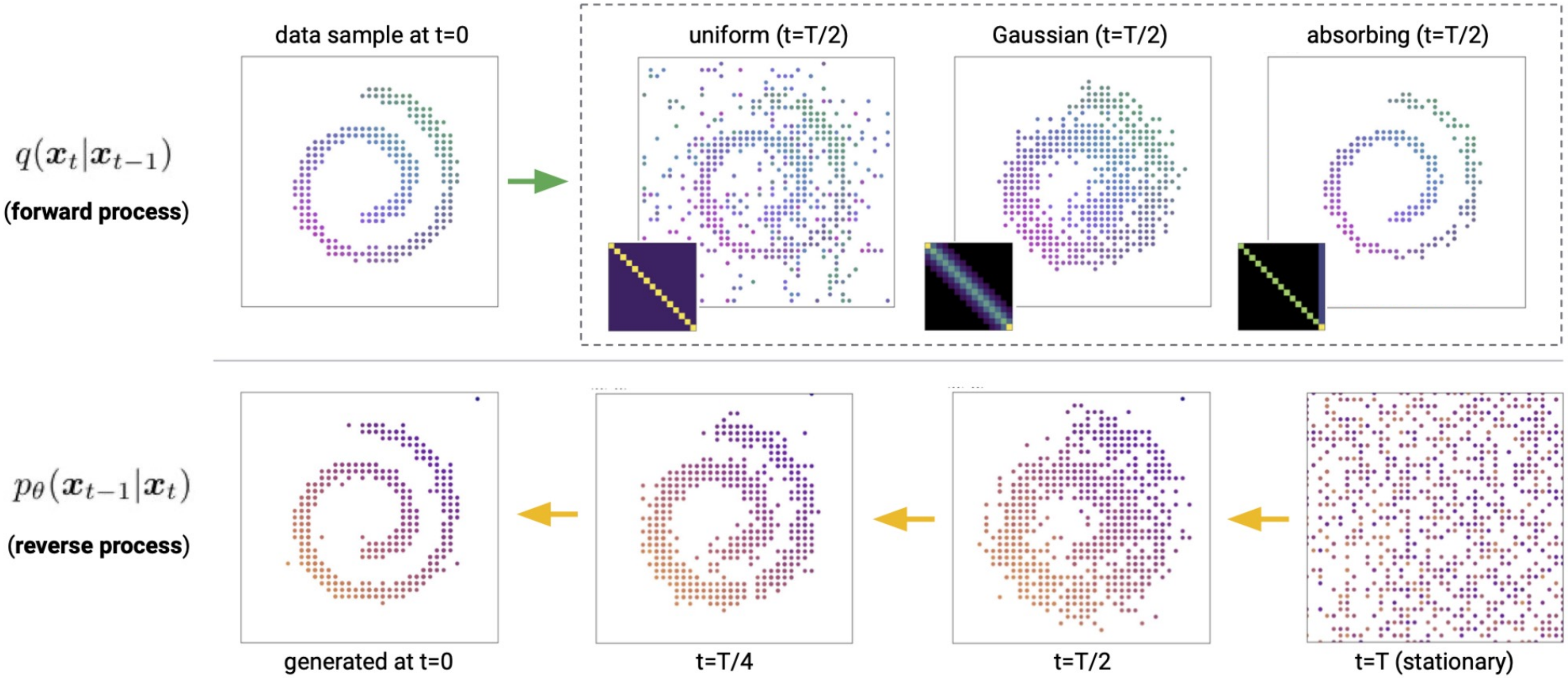
- Progressive masking out of data (generation is "de-masking")
- Tailored to ordinal data (e.g. discretized Gaussian)



(image from: Austin et al., "Structured Denoising Diffusion Models in Discrete State-Spaces", *NeurIPS*, 2021)

Austin et al., "Structured Denoising Diffusion Models in Discrete State-Spaces", *NeurIPS*, 2021

Discrete State Diffusion Models



(image from: Austin et al., "Structured Denoising Diffusion Models in Discrete State-Spaces", *NeurIPS*, 2021)

Austin et al., "Structured Denoising Diffusion Models in Discrete State-Spaces", *NeurIPS*, 2021

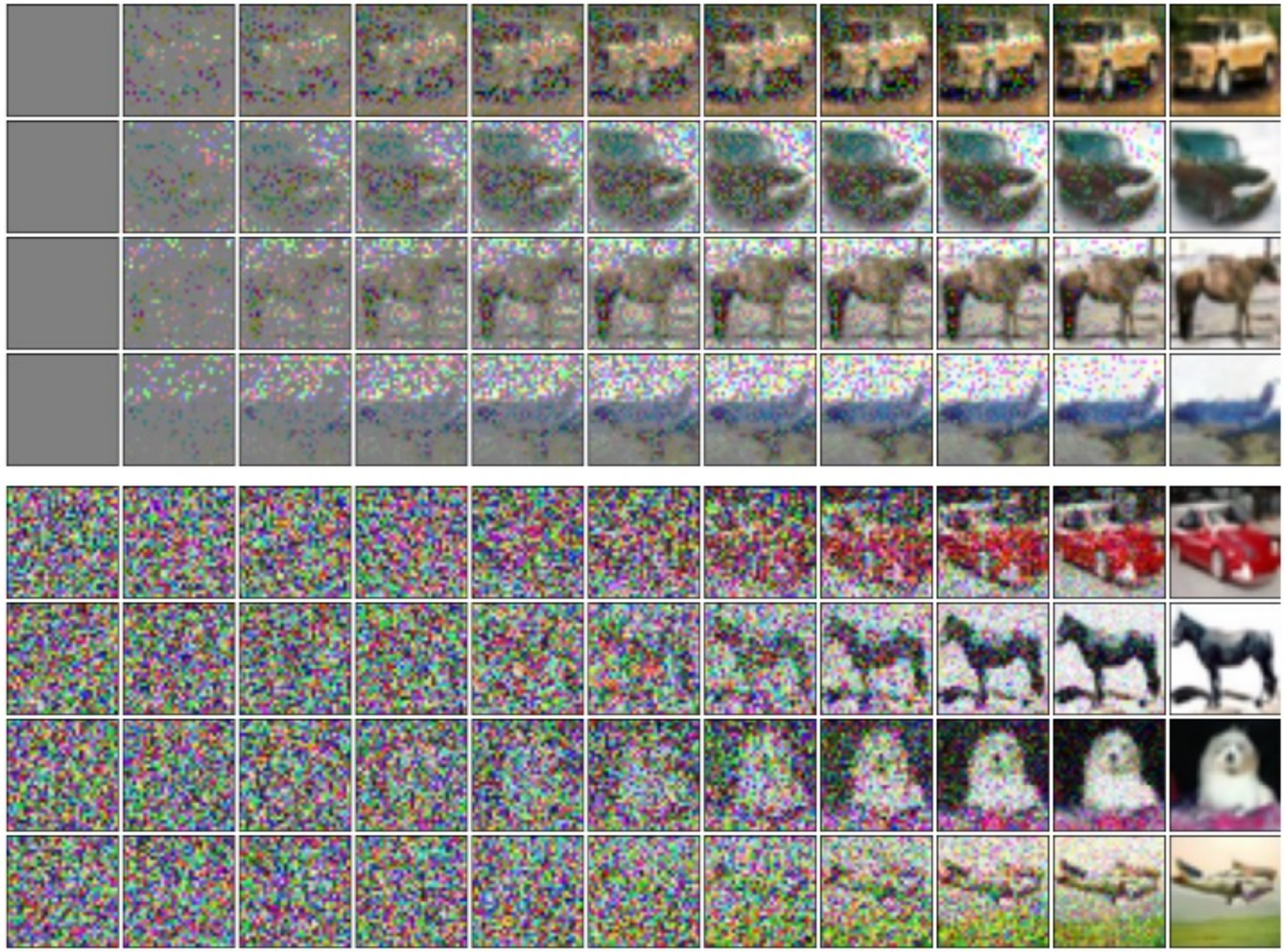
Discrete State Diffusion Models

Modeling Categorical Image Pixel Values

Progressive denoising starting from all-masked state.

Progressive denoising starting from random uniform state.

(with discretized Gaussian denoising model)

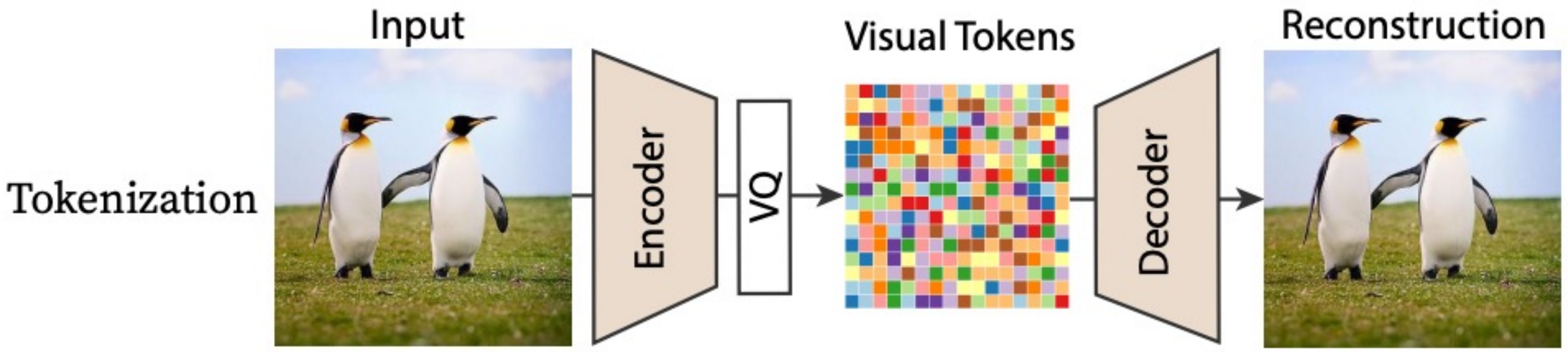


(image from: Austin et al., “Structured Denoising Diffusion Models in Discrete State-Spaces”, *NeurIPS*, 2021)

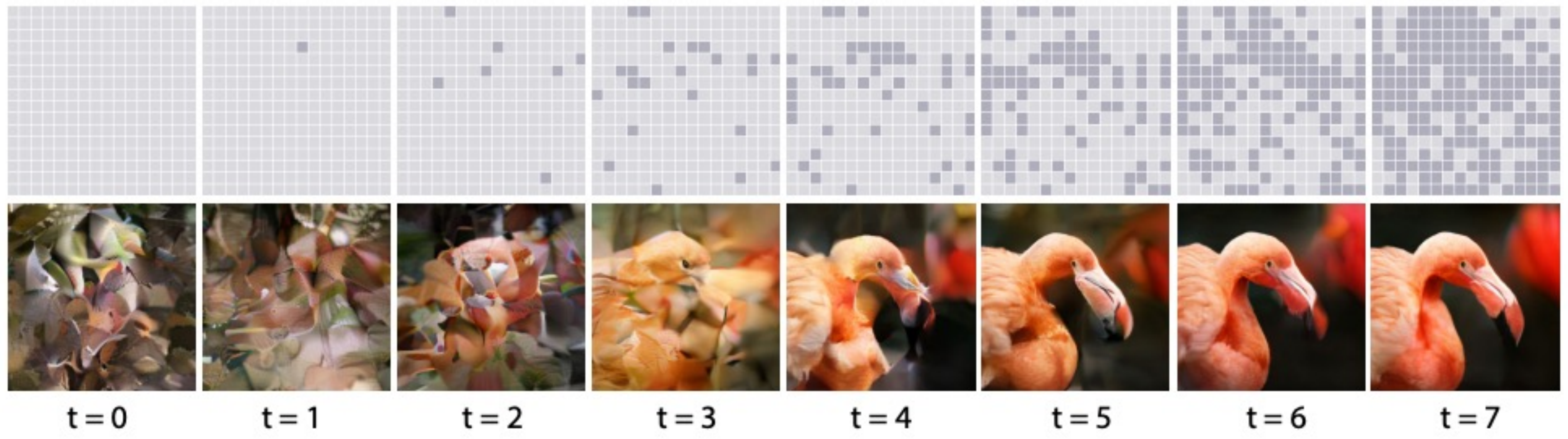
Austin et al., “Structured Denoising Diffusion Models in Discrete State-Spaces”, *NeurIPS*, 2021

Discrete State Diffusion Models

Modeling Discrete Image Encodings



Encoding images into latent space with discrete tokens, and modeling discrete token distribution



Iterative generation

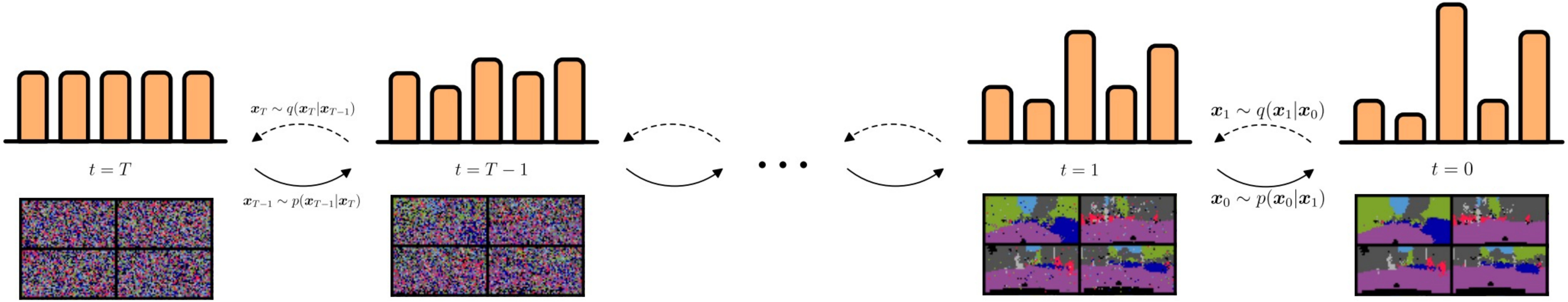
Class-conditional model samples

(images from: Chang et al., "MaskGIT: Masked Generative Image Transformer", *CVPR*, 2022)

[Chang et al., "MaskGIT: Masked Generative Image Transformer", *CVPR*, 2022]
[Esser et al., "ImageBART: Bidirectional Context with Multinomial Diffusion for Autoregressive Image Synthesis", *NeurIPS*, 2021]

Discrete State Diffusion Models

Modeling Pixel-wise Segmentations



(image from: Hoogeboom et al., "Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions", *NeurIPS*, 2022)

Hoogeboom et al., "Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions", *NeurIPS*, 2022

Today's Program

Title	Speaker	Time
Introduction	Arash	10 min
Part (1): Denoising Diffusion Probabilistic Models	Arash	35 min
Part (2): Score-based Generative Modeling with Differential Equations	Karsten	45 min
Part (3): Advanced Techniques: Accelerated Sampling, Conditional Generation, and Beyond	Ruiqi	45 min
Applications (1): Image Synthesis, Text-to-Image, Controllable Generation	Ruiqi	15 min
Applications (2): Image Editing, Image-to-Image, Super-resolution, Segmentation	Arash	15 min
Applications (3): Video Synthesis, Medical Imaging, 3D Generation, Discrete State Models	Karsten	15 min
Conclusions, Open Problems and Final Remarks	Arash	10 min

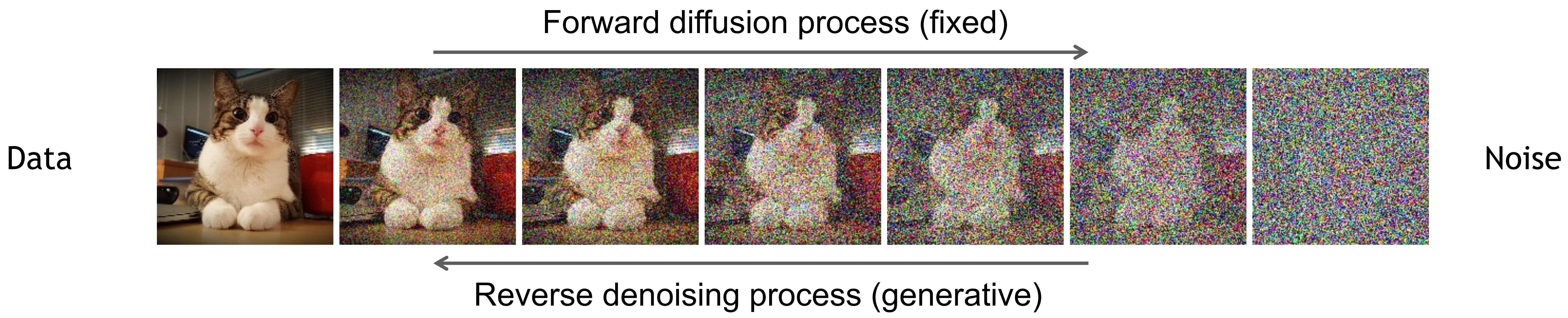
cvpr2022-tutorial-diffusion-models.github.io

Conclusions, Open Problems and Final Remarks

Summary: Denoising Diffusion Probabilistic Models

"Discrete-time" Diffusion Models

We started with denoising diffusion probabilistic models:



We showed how the denoising model can be trained by predicting noise injected in each diffused image:

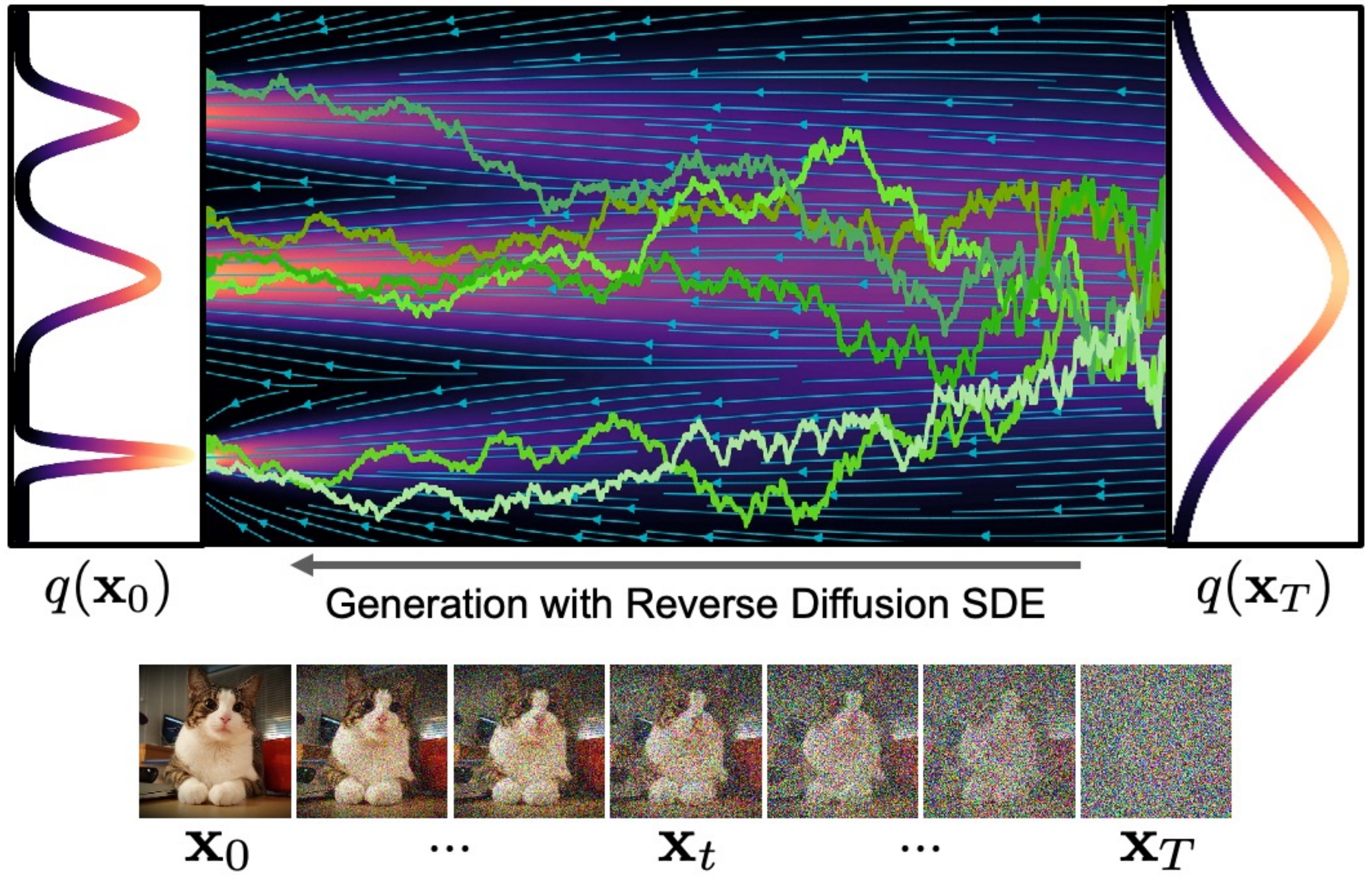
$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1,T)} \left[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, t)||^2 \right]$$

Summary: Score-based Generative Models with Differential Eqn.

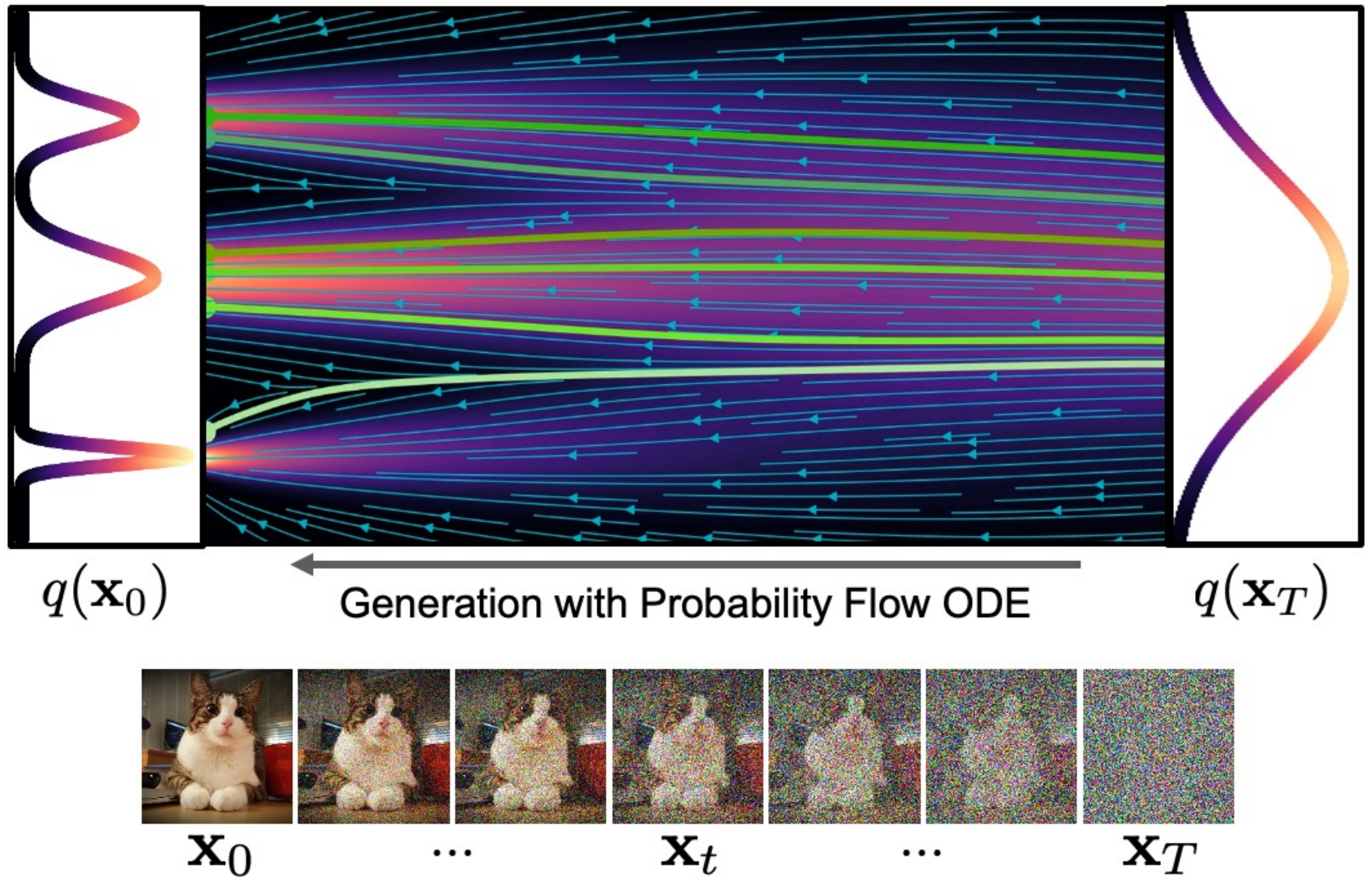
“Continuous-time” Diffusion Models

In the second part, we considered the limit of an infinite number of steps with an infinitesimal noise

Generative Reverse Diffusion SDE (stochastic)



Generative Probability Flow ODE (deterministic):



These continuous-time diffusion models allow us to choose discretization and ODE/SDE solvers at test time.

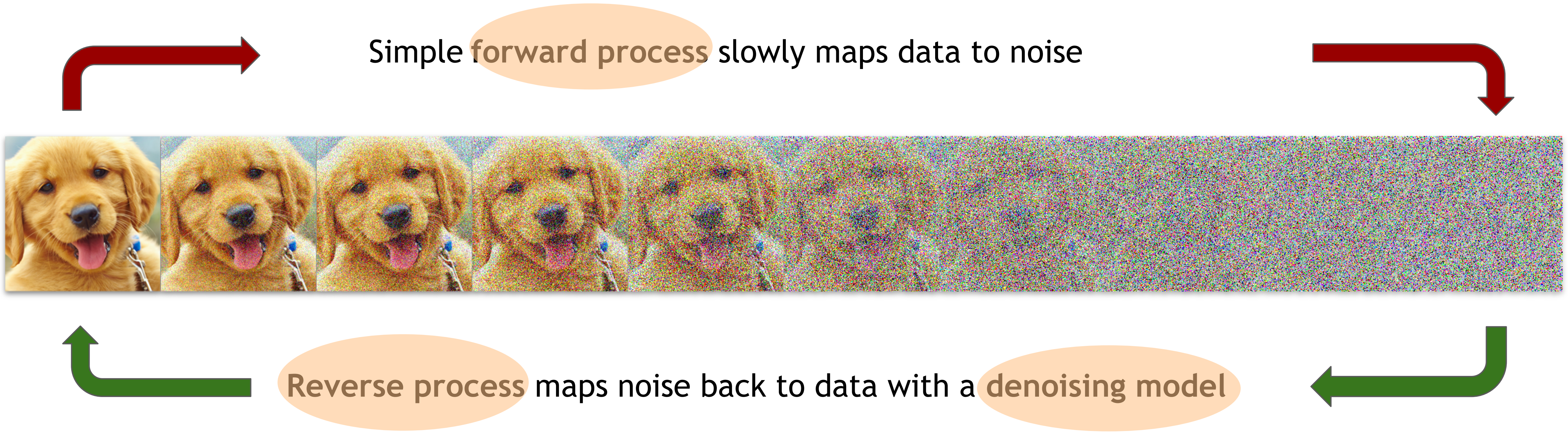
Summary: Advanced Techniques

Acceleration, Guidance and beyond

In the third part, we discussed several advanced topics in diffusion models.

How can we accelerate the sample generation?

[Image credit: Ben Poole, Mohammad Norouzi]



How to scale up diffusion models to high-resolution (conditional) generation?

- Cascaded models
- Guided diffusion models

Summary: Applications

We covered many successful applications of diffusion models:

- 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

Open Problems (1)

- Diffusion models are a special form of VAEs and continuous normalizing flows
    - Why do diffusion models perform so much better than these models?
    - How can we improve VAEs and normalizing flows with lessons learned from diffusion models?

- Sampling from diffusion models is still slow especially for interactive applications
    - The best we could reach is 4-10 steps. How can we have one step samplers?
    - Do we need new diffusion processes?

- Diffusion models can be considered as latent variable models, but their latent space lacks semantics
    - How can we do latent-space semantic manipulations in diffusion models

Open Problems (2)

- How can diffusion models help with discriminative applications?
    - Representation learning (high-level vs low-level)
    - Uncertainty estimation
    - Joint discriminator-generator training

- What are the best network architectures for diffusion models?
    - Can we go beyond existing U-Nets?
    - How can we feed the time input and other conditioning?
    - How can we improve the sampling efficiency using better network designs?

Open Problems (3)

- How can we apply diffusion models to other data types?
  - 3D data (e.g., distance functions, meshes, voxels, volumetric representations), video, text, graphs, etc.
  - How should we change diffusion models for these modalities?

- Compositional and controllable generation
  - How can we go beyond images and generate scenes?
  - How can we have more fine-grained control in generation?

- Diffusion models for X
  - Can we better solve applications that were previously addressed by GANs and other generative models?
  - Which applications will benefit most from diffusion models?

Thanks!

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

@karsten_kreis

@RuiqiGao

@ArashVahdat