CS-461

Foundation Models and Generative AI

Generative Models II:
Diffusion Models and Beyond

Karsten Kreis and Ruiqi Gao, Guest Lecture, Fall Semester 2025/26

Diffusion-based Generative Modeling:
Foundations and Applications

Karsten Kreis

Ruiqi Gao

NVIDIA

Google DeepMind





Black Forest Labs. https://bfl.ai

”A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She walks confidently and casually. The street is damp and reflective, creating a mirror effect of the colorful lights. Many pedestrians walk about.”

Brooks, Peebles, et al. https://openai.com/sora

Overview

I. Fundamentals (Karsten):

- Generation by Noising and Denoising
- A Differential Equation-based Perspective
- Flow Matching, Basic Architectures and Guidance
- Latent Diffusion Models

II. Applications (Ruiqi):

- What makes Diffusion Great?
- Video Diffusion Models
- Beyond Video Generation: World Models
- 3D/4D Generation

Tutorials on Diffusion Models

CVPR 2022: *Denoising Diffusion-based Generative Modeling: Foundations and Applications*

Website (~4 hours long, over 100,000 views on Youtube): https://cvpr2022-tutorial-diffusion-models.github.io/


Karsten Kreis
NVIDIA


Ruiqi Gao
Google Brain


Arash Vahdat
NVIDIA

CVPR 2023: *Denoising Diffusion Models: A Generative Learning Big Bang*

Website: https://cvpr2023-tutorial-diffusion-models.github.io/


Jiaming Song
NVIDIA


Chenlin Meng
Stanford


Arash Vahdat
NVIDIA

NeurIPS 2023: *Latent Diffusion Models: Is the Generative AI Revolution Happening in Latent Space?*

Website: https://neurips2023-ldm-tutorial.github.io/


Karsten Kreis
NVIDIA


Ruiqi Gao
Google Brain


Arash Vahdat
NVIDIA

Overview

I. Fundamentals (Karsten):

- Generation by Noising and Denoising
- A Differential Equation-based Perspective
- Flow Matching, Basic Architectures and Guidance
- Latent Diffusion Models

II. Applications (Ruiqi):

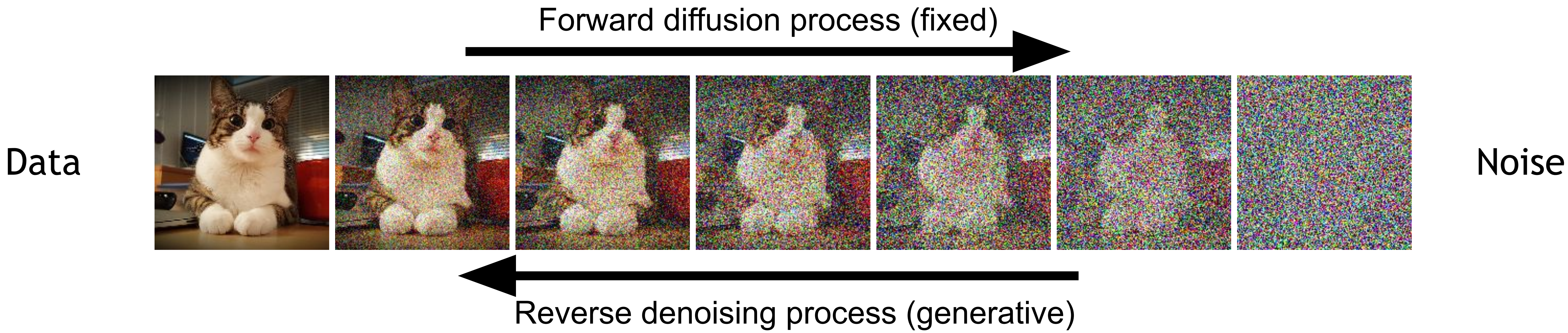
- What makes Diffusion Great?
- Video Diffusion Models
- Beyond Video Generation: World Models
- 3D/4D Generation

Diffusion Models

Learning to Generate by Denoising

Diffusion models consist of two processes:

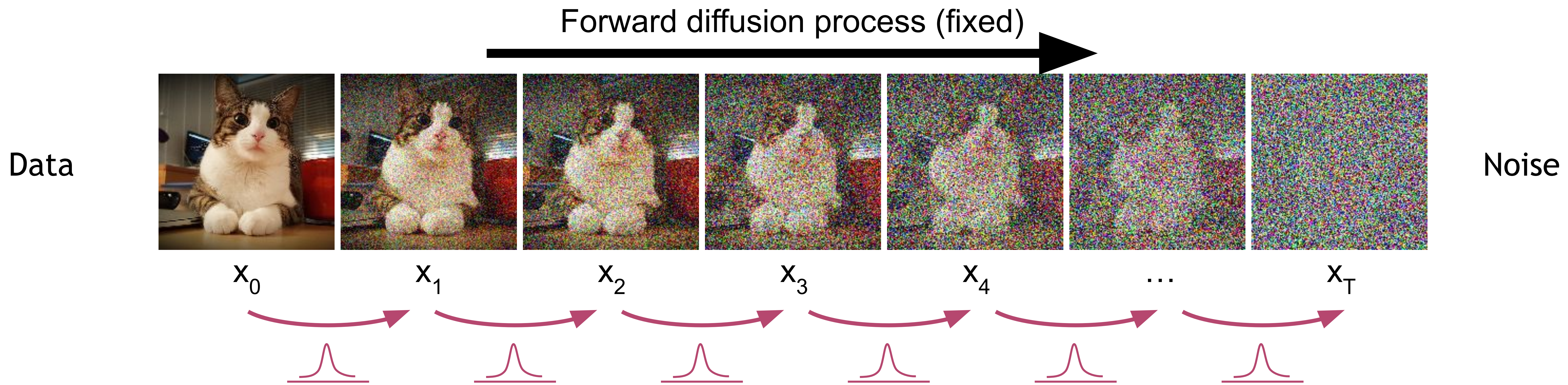
- (Fixed) forward diffusion process that gradually adds noise to input
- (Learnt) reverse denoising process that learns to generate data by denoising



Sohl-Dickstein et al., “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”, ICML 2015
Ho et al., “Denoising Diffusion Probabilistic Models”, *NeurIPS* 2020

Diffusion Models

The Fixed Forward Diffusion Process

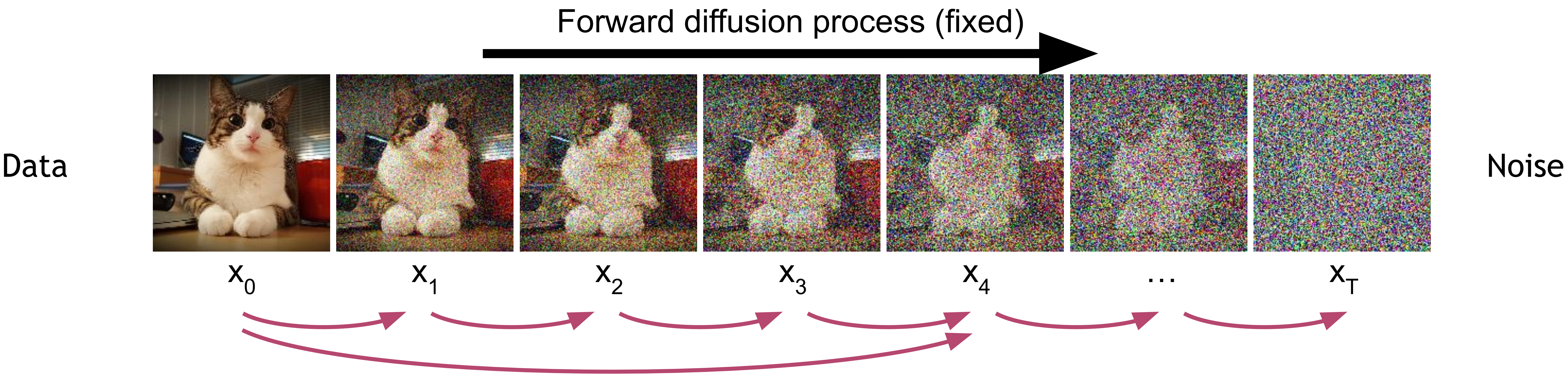


$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad \text{(joint)}$$

Ho et al., “Denoising Diffusion Probabilistic Models”, *NeurIPS* 2020

Diffusion Models

The Fixed Forward Diffusion Process



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 ("noise schedule") chosen such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

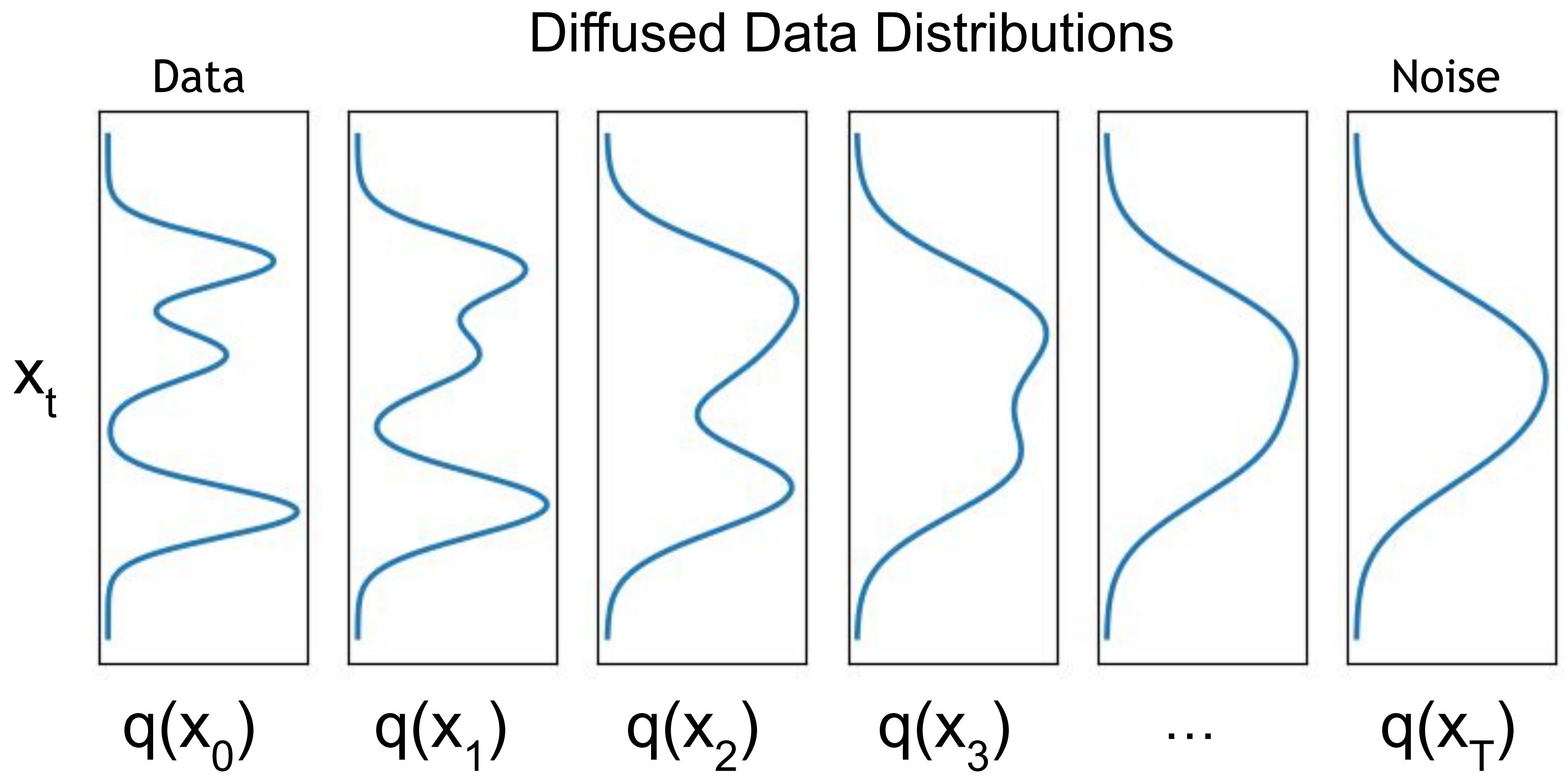
Ho et al., "Denoising Diffusion Probabilistic Models", *NeurIPS* 2020

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 $q(\mathbf{x}_t)$?

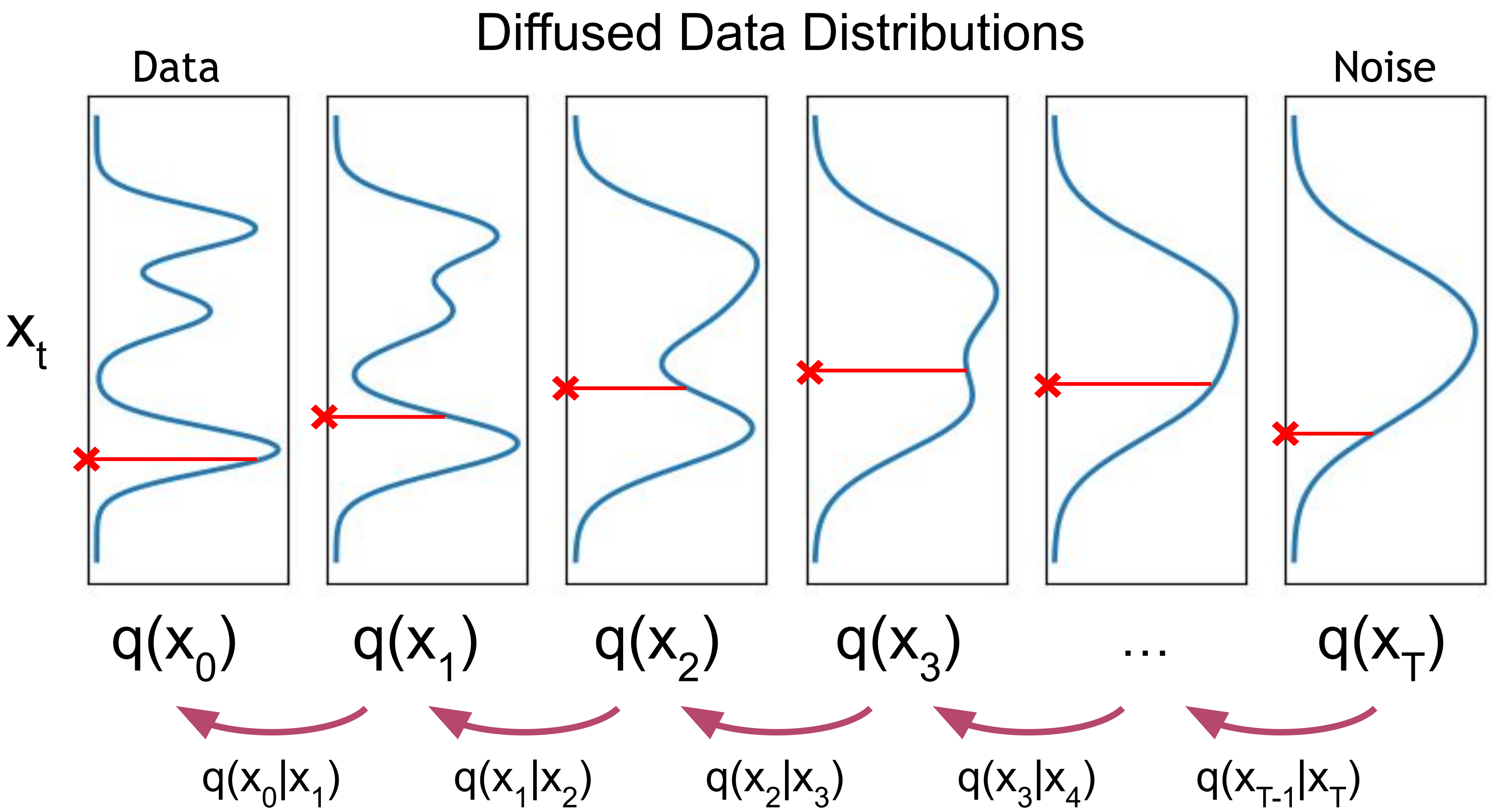
$$\underbrace{q(\mathbf{x}_t)}_{\substack{\text{Diffused} \\ \text{data dist.}}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\substack{\text{Joint} \\ \text{dist.}}} \, d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\substack{\text{Input} \\ \text{data dist.}}} \, \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\substack{\text{Diffusion}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

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



Generation:

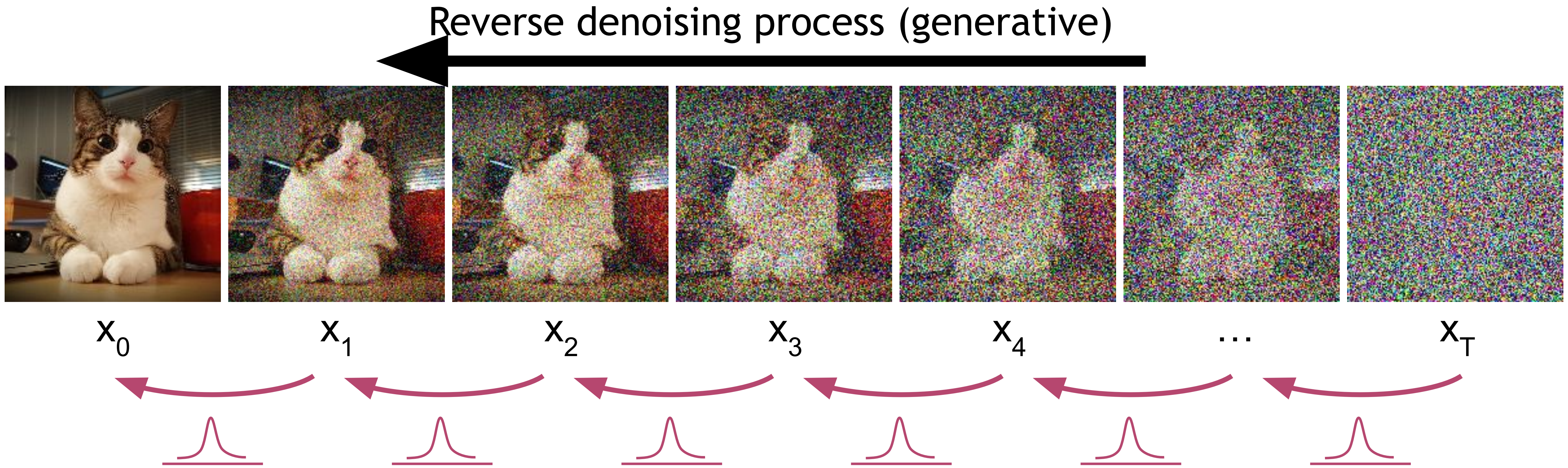
- Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T, \mathbf{0}, \mathbf{I})$
- Iteratively sample $\underbrace{\mathbf{x}_{t-1} \sim q(\mathbf{x}_{t-1}|\mathbf{x}_t)}_{\text{True Denoising Dist.}}$ ➡ **Intractable!** ➡ Approx. Gaussian for small steps. (β_t)!

➡ Learn Gaussian denoiser (and use many small steps)!

Diffusion Models

The Learnt Reverse Generative Process

Data



Noise

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

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

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

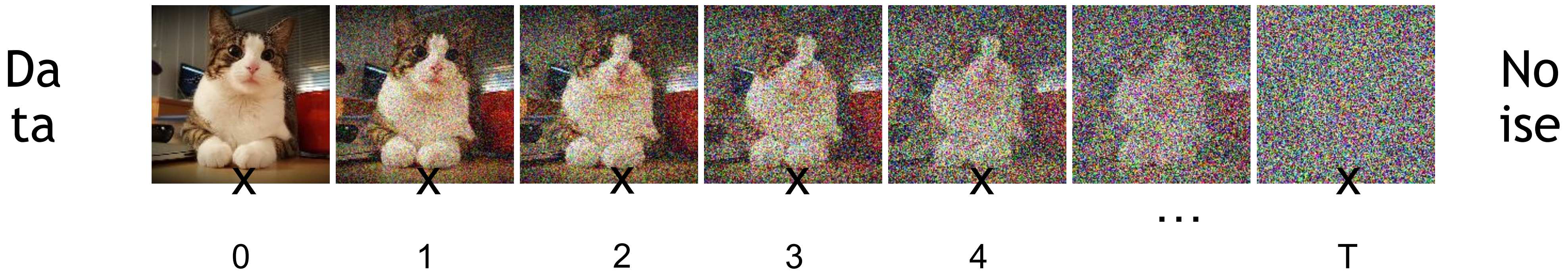
Trainable network
(U-Net, Denoising Autoencoder)

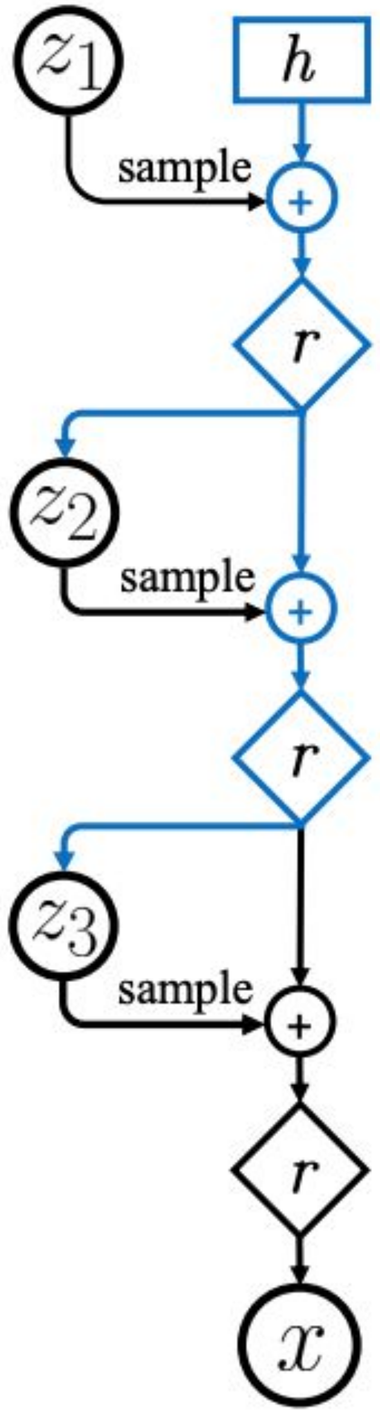
$\Rightarrow$

Trainable with maximum likelihood
(evidence lower bound/ELBO)

Ho et al., "Denoising Diffusion Probabilistic Models", *NeurIPS* 2020

Connection to Variational Autoencoders (VAEs)





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

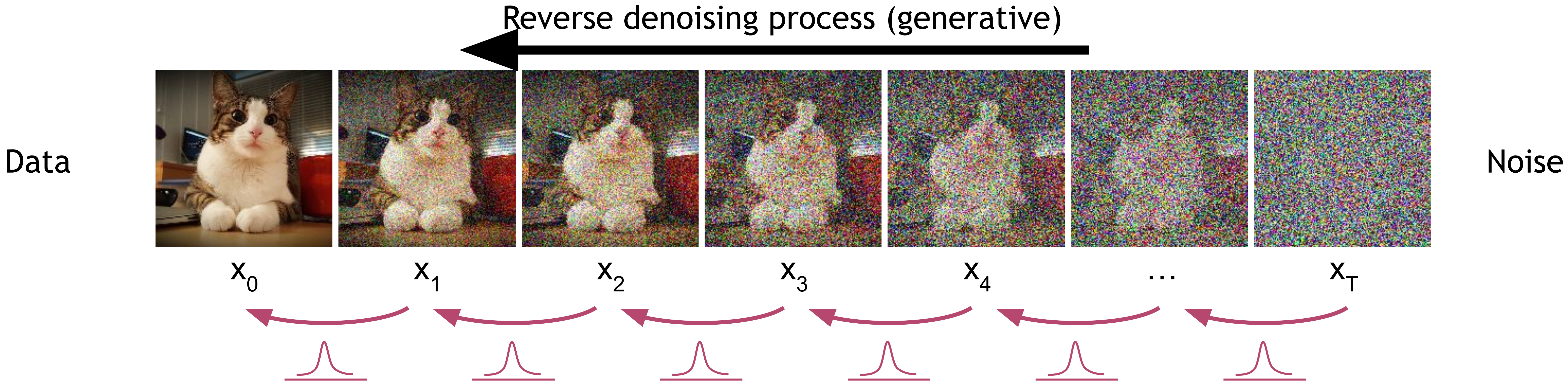
However, in diffusion models:

- “Encoder” (forward diffusion) is fixed
- Latent variables have same dimension as data
- Denoising model shared across timesteps
- Trained with reweighted variational bound
- No ”prior holes”, forward process converges to Normal distribution by construction

Ho et al., “Denoising Diffusion Probabilistic Models”, *NeurIPS* 2020

Diffusion Models

Evidence Lower Bound Objective

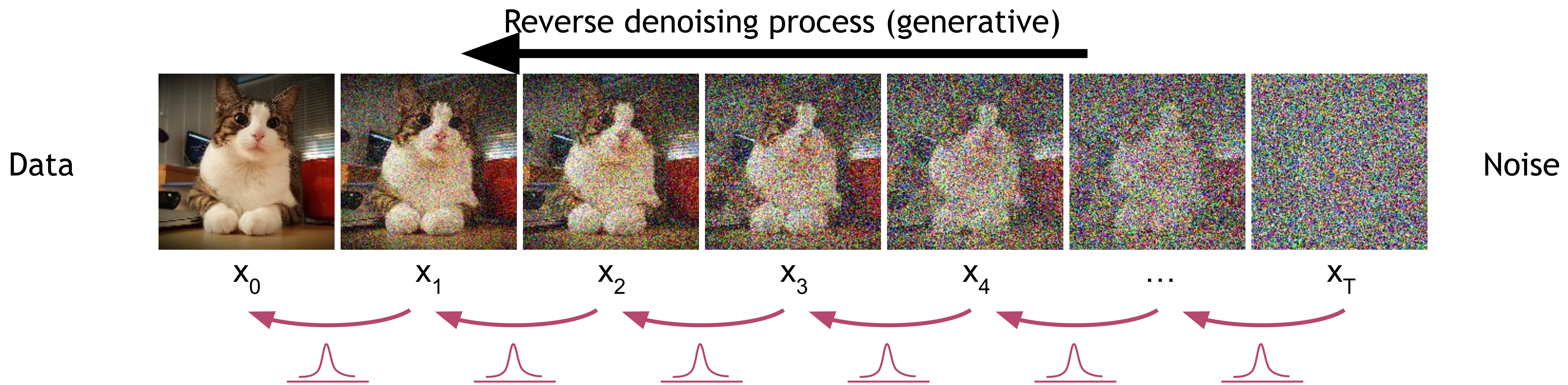


$$\mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\,\|\,p(\mathbf{x}_T))}_{\text{Constant}} + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\,\|\,p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right]$$

Ho et al., "Denoising Diffusion Probabilistic Models", *NeurIPS* 2020

Diffusion Models

Evidence Lower Bound Objective

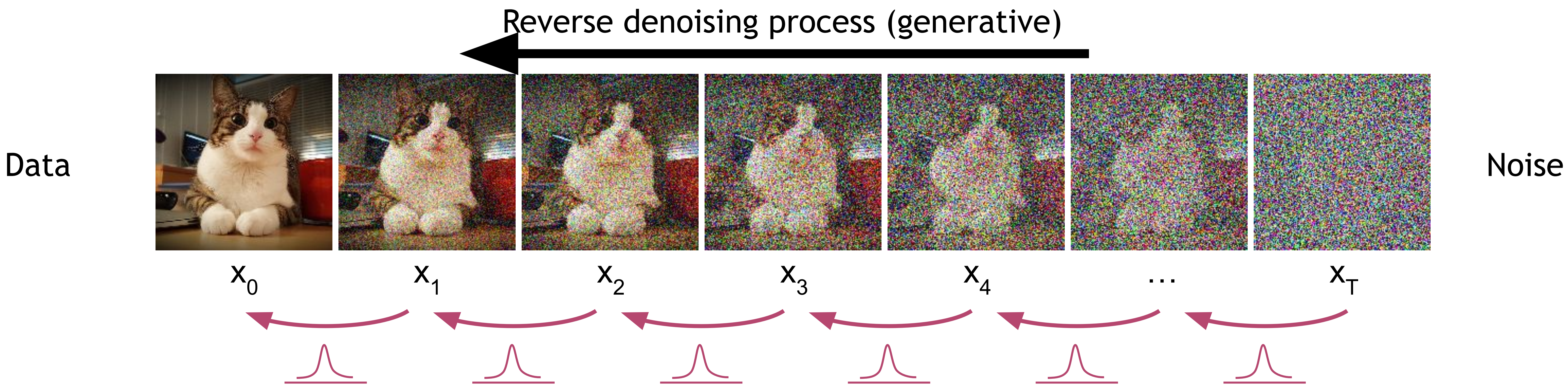


$$\mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\parallel p(\mathbf{x}_T))}_{\text{Constant}}+\underbrace{\sum_{t>1}D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\text{Matching Denoiser to Forward Diffusion}}-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right]$$

Ho et al., "Denoising Diffusion Probabilistic Models", *NeurIPS* 2020

Diffusion Models

Evidence Lower Bound Objective

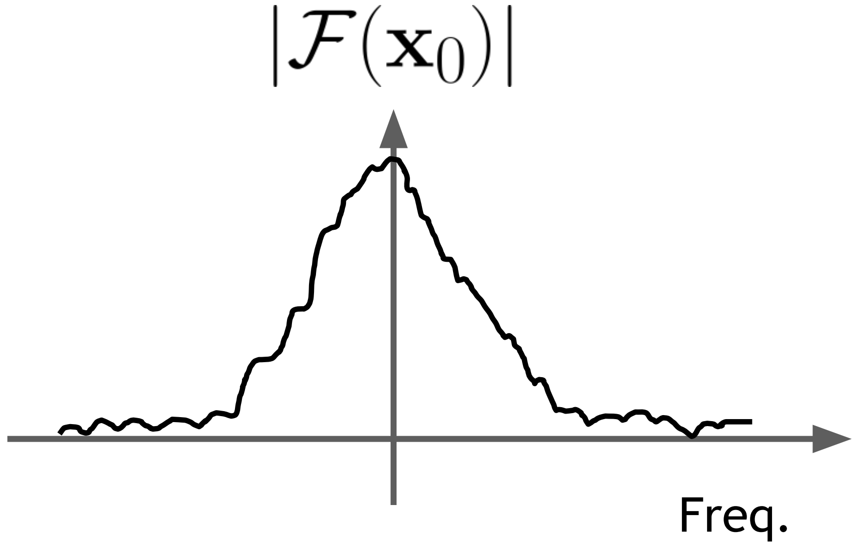


$$\mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\parallel p(\mathbf{x}_T))}_{\text{Constant}} + \underbrace{\sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\text{Matching Denoiser to Forward Diffusion}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{\text{Final step clean data prediction.}}\right]$$

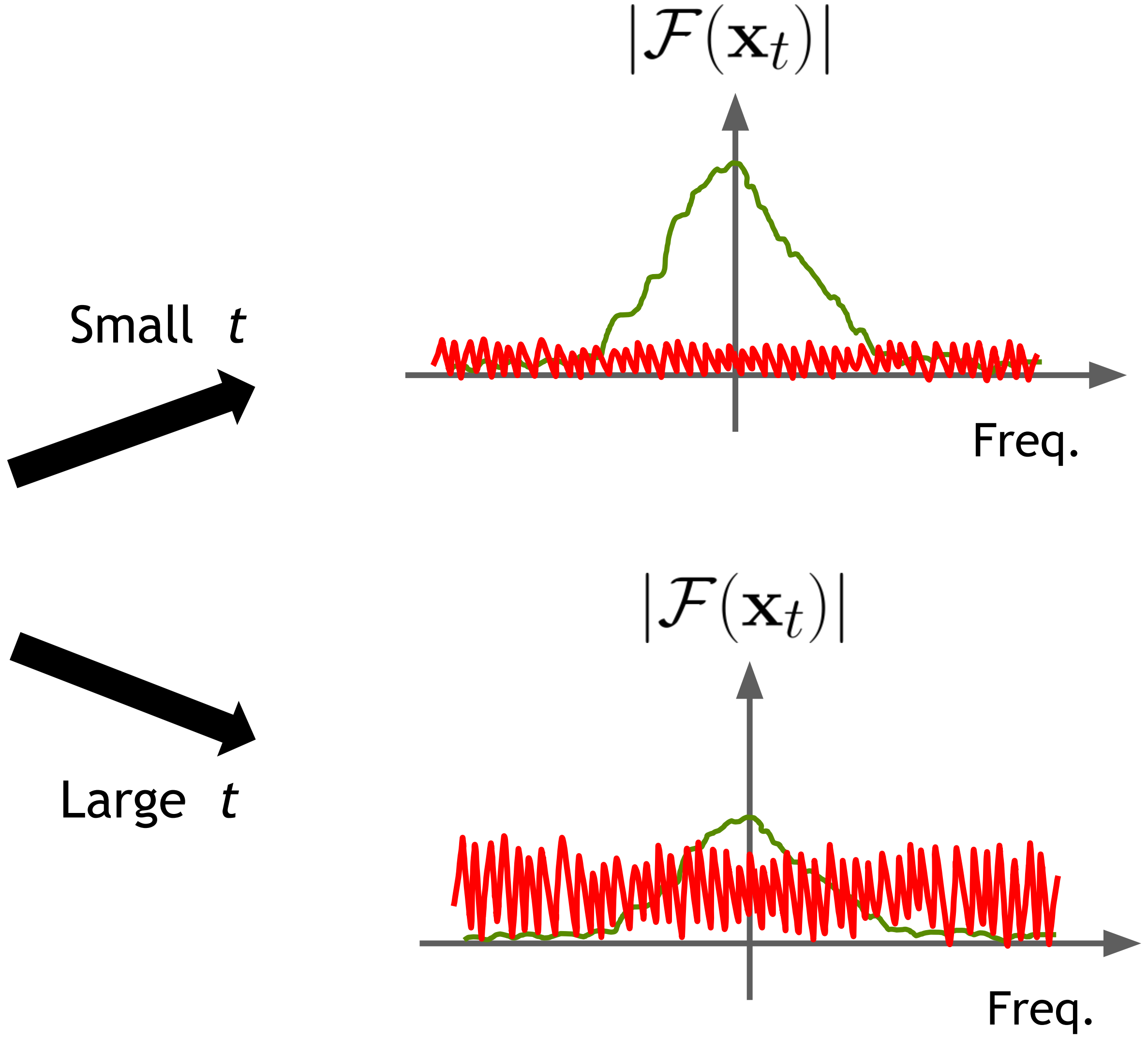
Ho et al., "Denoising Diffusion Probabilistic Models", *NeurIPS* 2020

What happens to an image in the forward diffusion process?

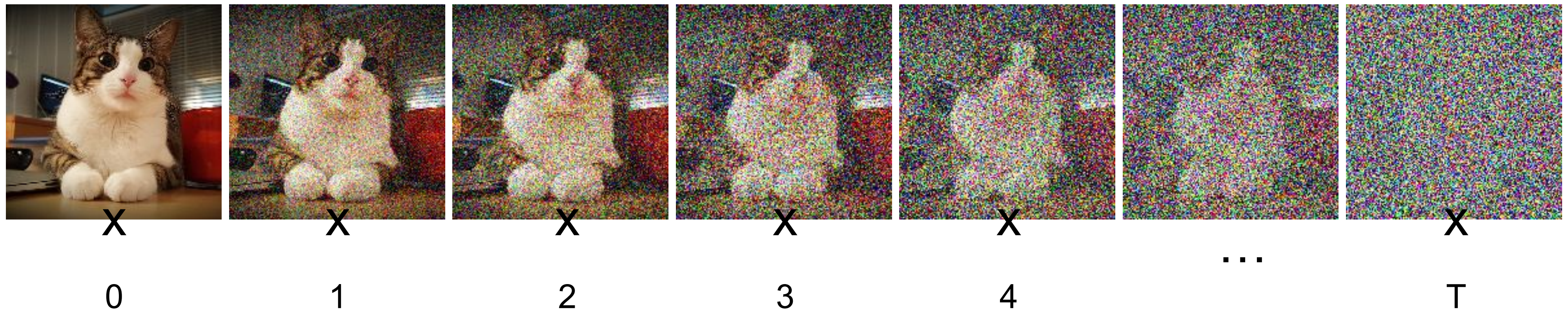
Fourier transform



Spatial frequency spectrum
of natural images.

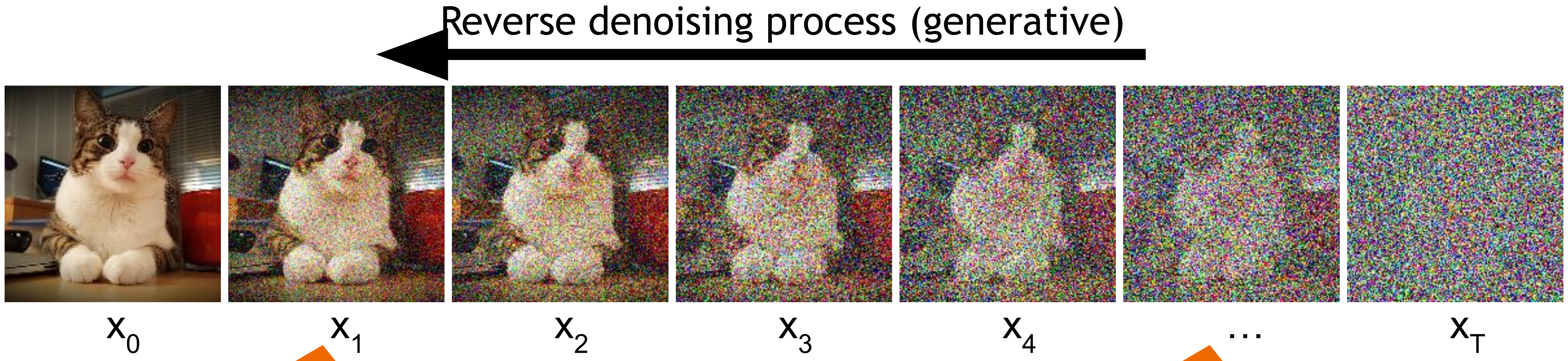


In forward diffusion, high frequency content is perturbed faster.



Content-Detail Tradeoff

Data



Noise

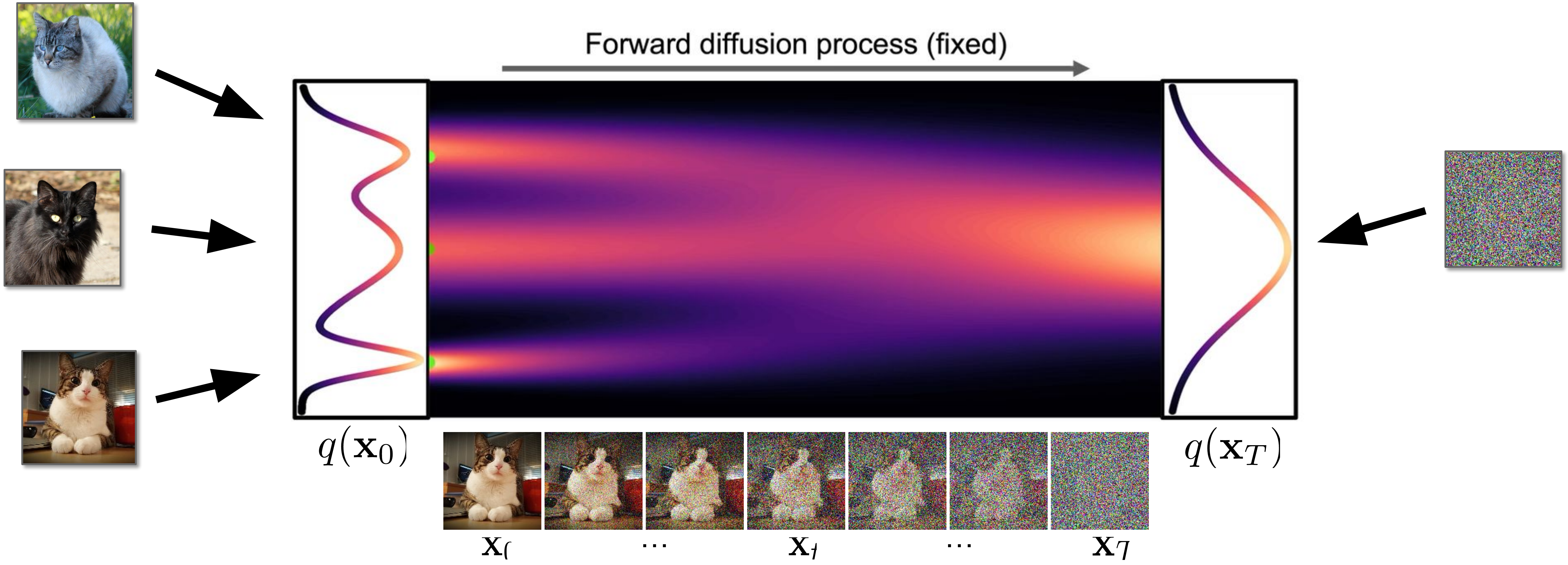
The denoising model is specialized for generating the high-frequency content (i.e., low-level details)

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!

Diffusion Models

A Stochastic Differential Equation-based Perspective

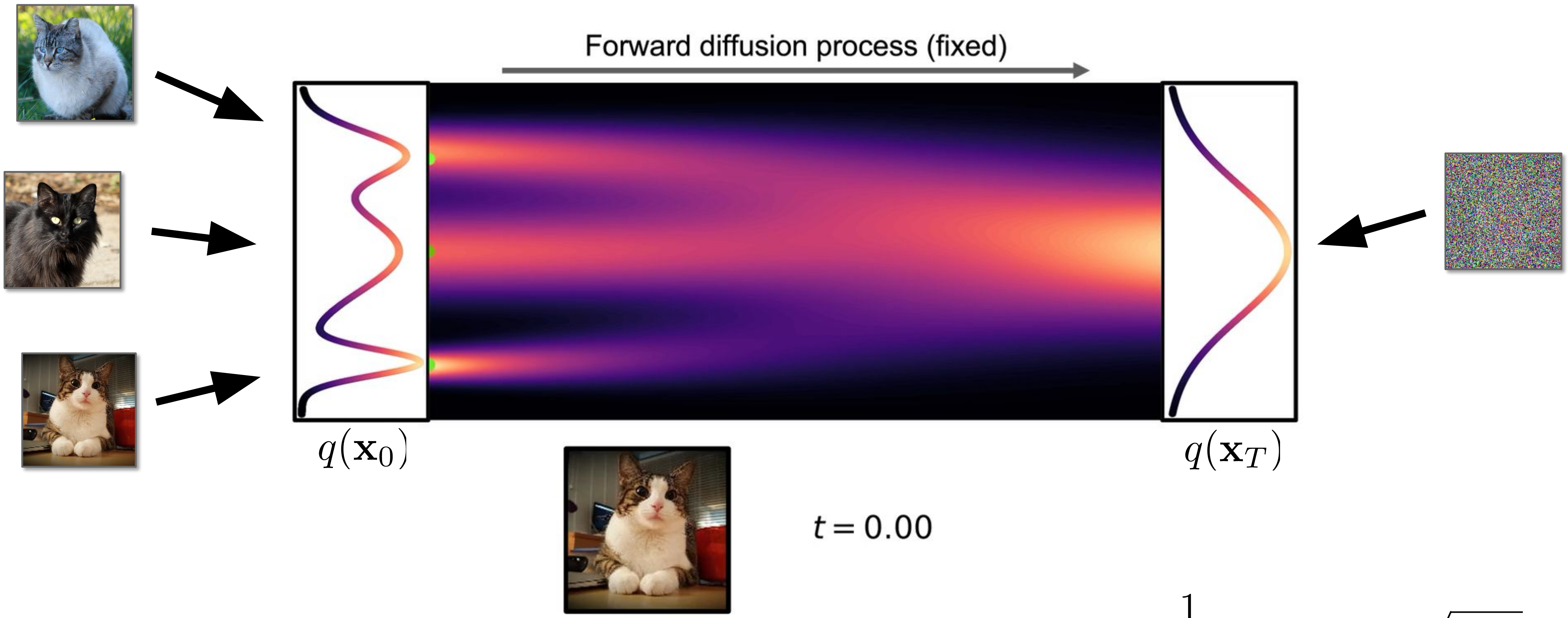


$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

Song et al., "Score-Based Generative Modeling through Stochastic Differential Equations", *ICLR*, 2021
Anderson, "Reverse-time diffusion equation models", *Stochastic Processes and their Applications*,1982

Diffusion Models

A Stochastic Differential Equation-based Perspective

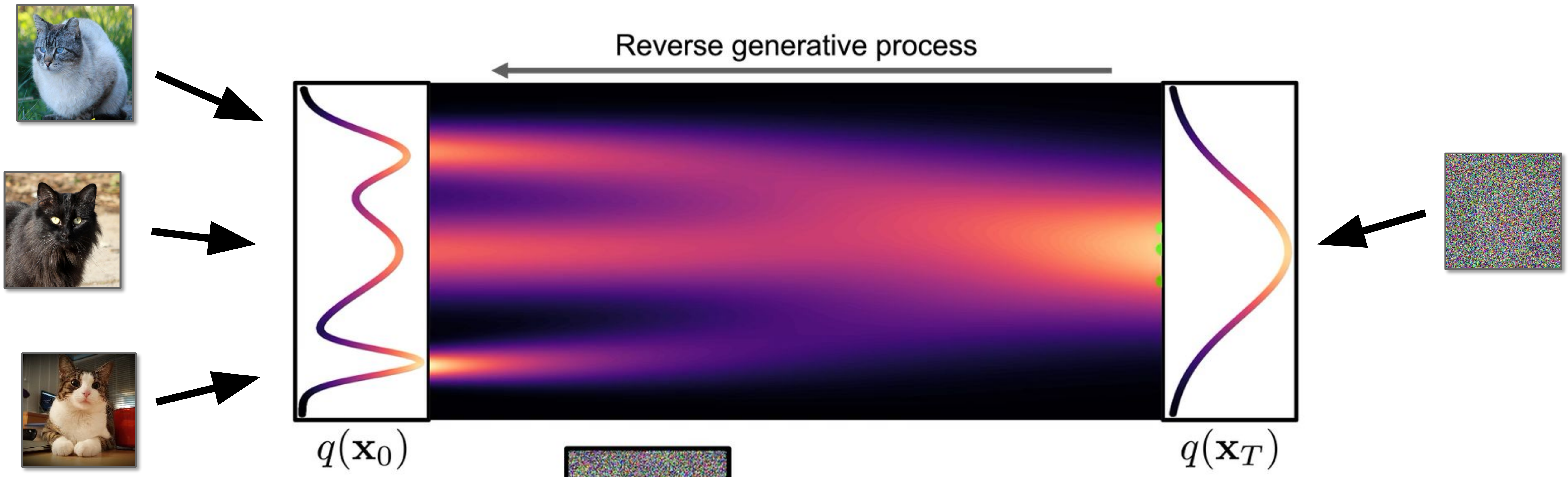


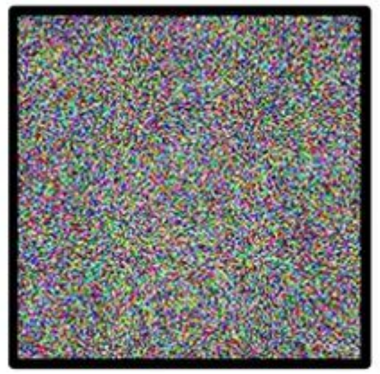
$$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,dt + \sqrt{\beta(t)}\,d\boldsymbol{\omega}_t$$

Song et al., “Score-Based Generative Modeling through Stochastic Differential Equations”, *ICLR*, 2021
Anderson, “Reverse-time diffusion equation models”, *Stochastic Processes and their Applications*,1982

Diffusion Models

A Stochastic Differential Equation-based Perspective

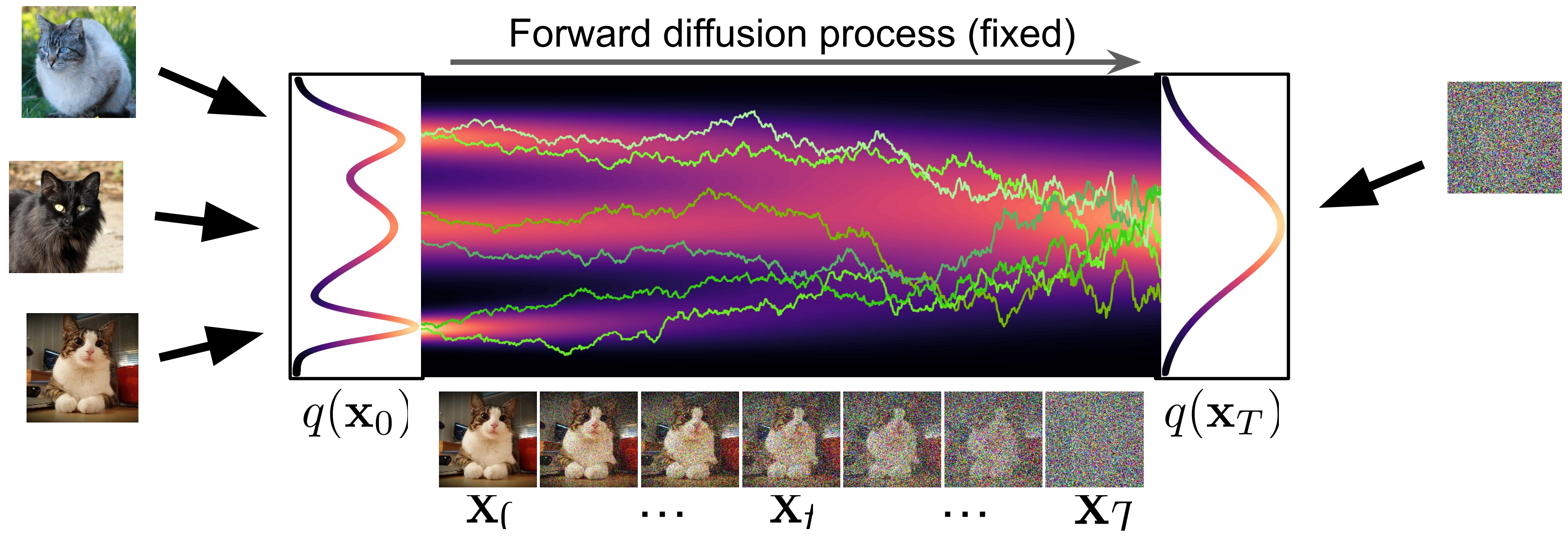




$$d\mathbf{x}_t = \left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\underbrace{\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)}_{\text{“Score Function” – modeled by neural network}}\right]dt + \sqrt{\beta(t)}\,d\bar{\boldsymbol{\omega}}_t$$

Song et al., “Score-Based Generative Modeling through Stochastic Differential Equations”, *ICLR*, 2021
Anderson, “Reverse-time diffusion equation models”, *Stochastic Processes and their Applications*, 1982

Score Matching



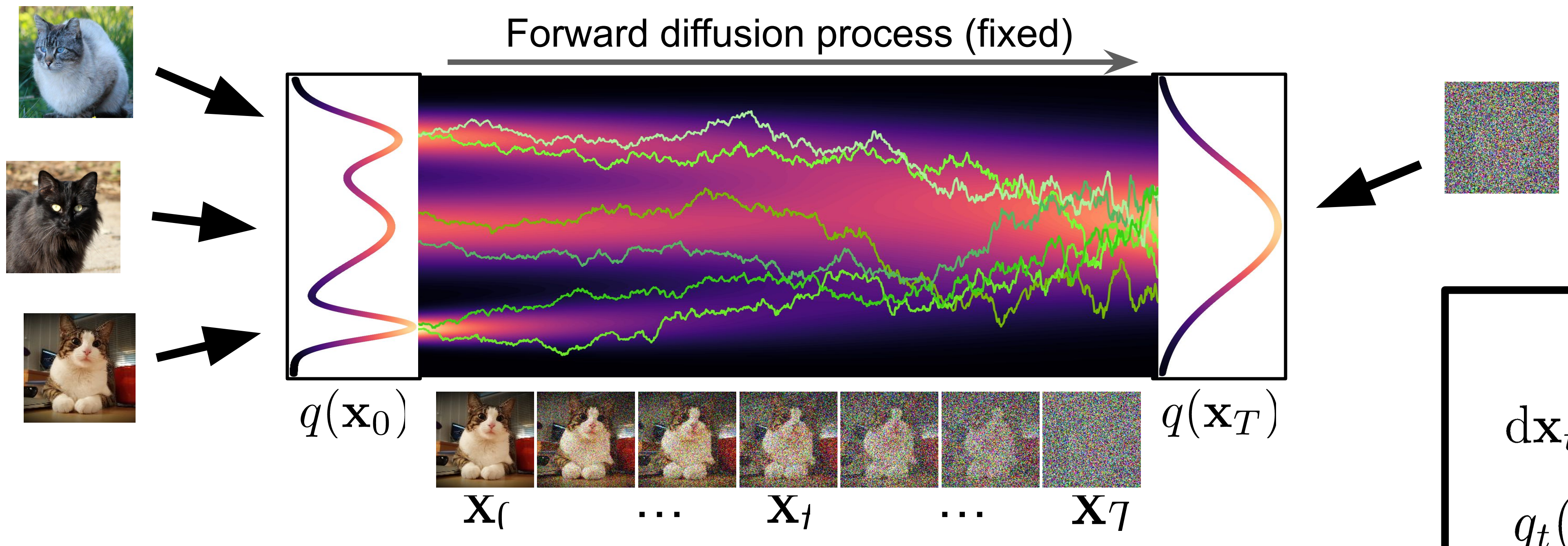
- Naïve idea, learn model for the score function by direct regression?

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

➡ **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:

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

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

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

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

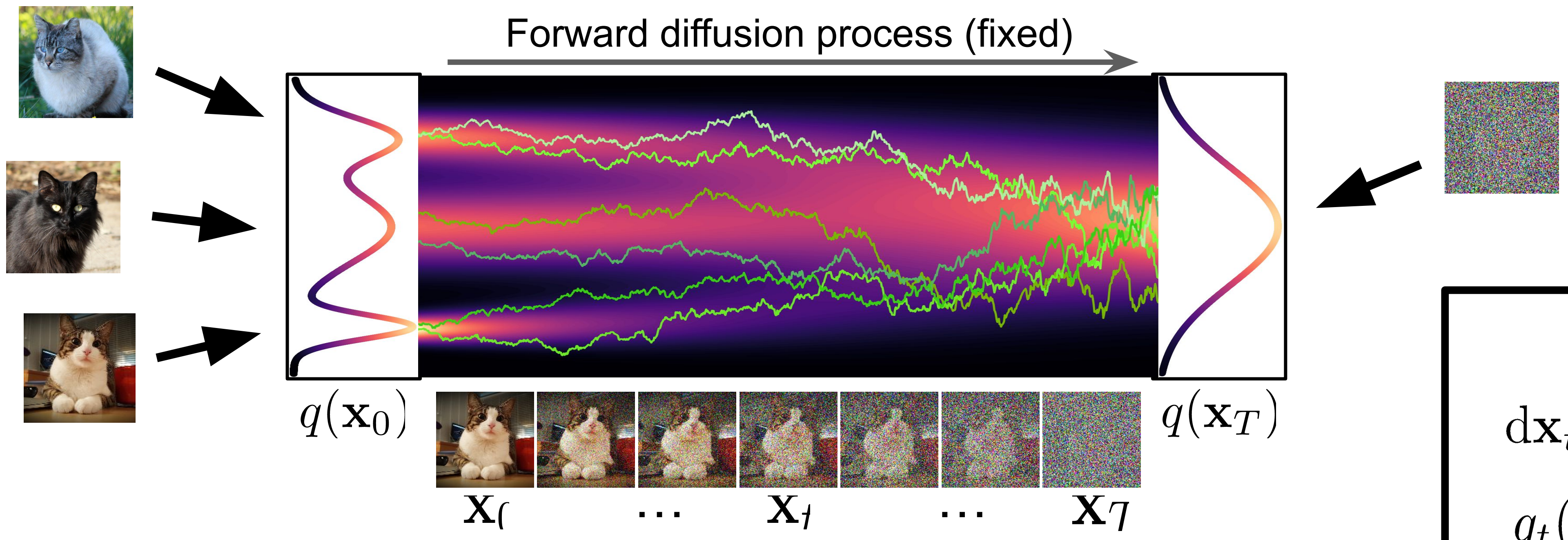
- Instead, diffuse individual data points $\mathbf{x}_0$. Diffused $q_t(\mathbf{x}_t|\mathbf{x}_0)$ is tractable!
- **Denoising Score Matching:**

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

➡ **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 with 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:**

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

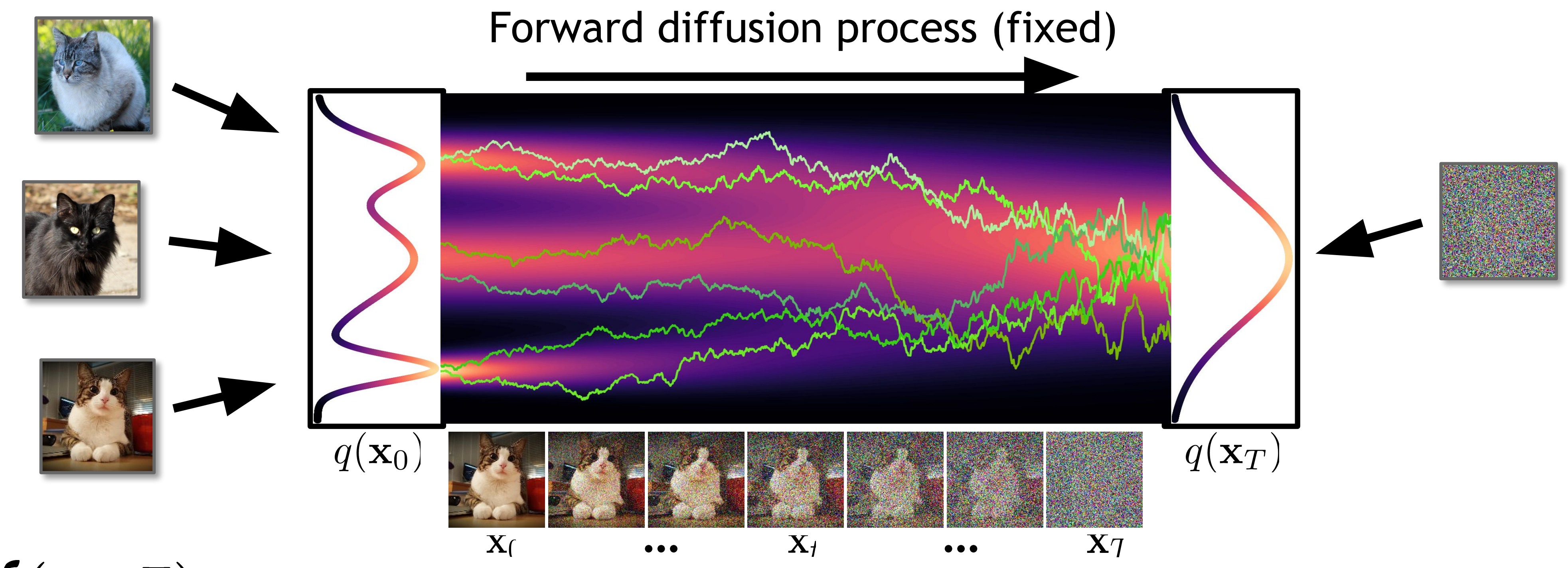
  - Re-parametrized sampling: $\mathbf{x}_t = \gamma_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - Score function: $\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t|\mathbf{x}_0) = -\nabla_{\mathbf{x}_t}\frac{(\mathbf{x}_t - \gamma_t\mathbf{x}_0)^2}{2\sigma_t^2}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

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

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

Denoising Score Matching

Training Objectives in Practice



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

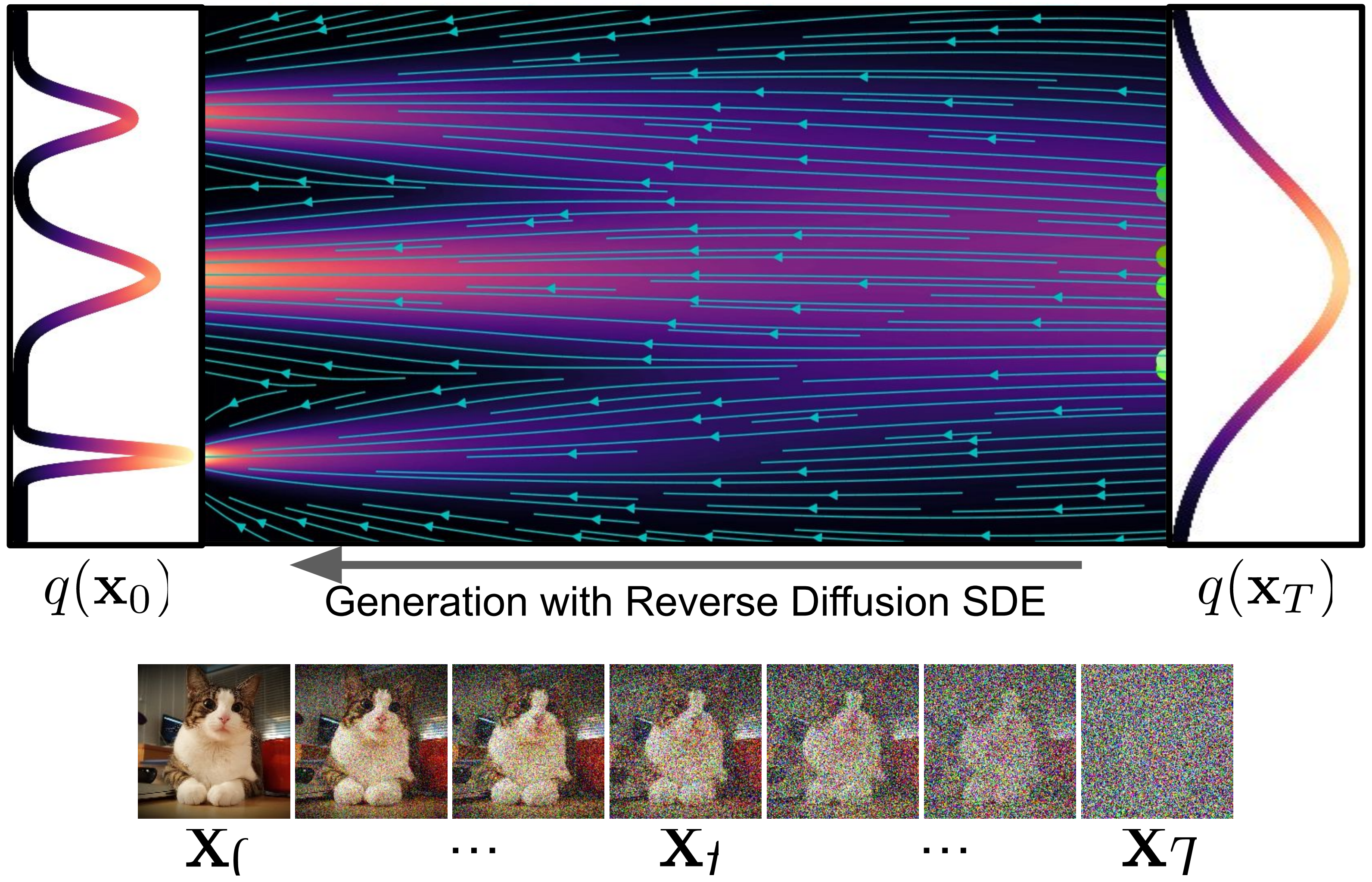
Data perturbation $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$ (α_t, σ_t define "noise schedule")

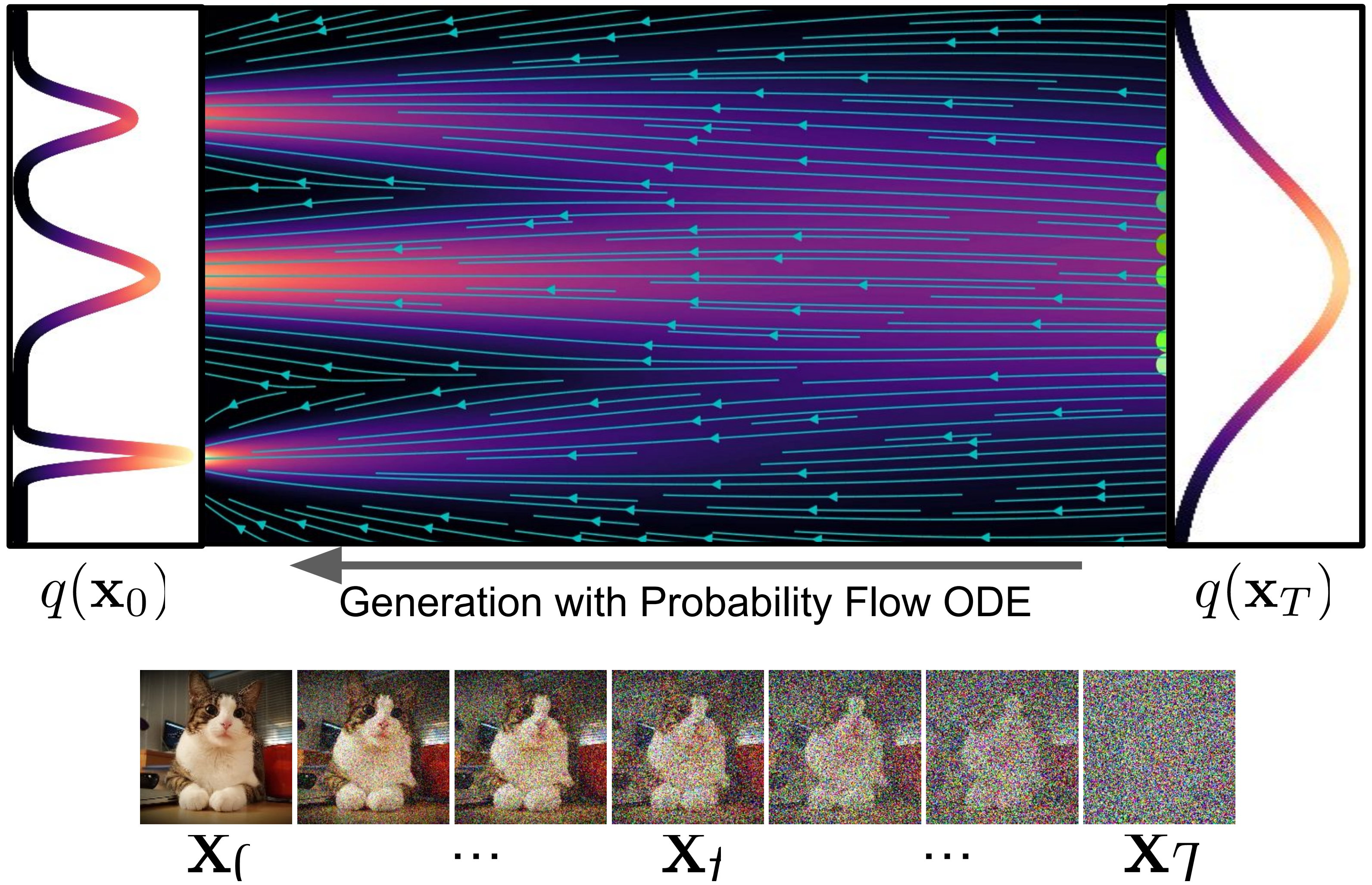
$$\min_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_{t \sim \mathcal{U}(0,T)}}_{\text{diffusion time } t} \underbrace{\mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0)}}_{\text{data sample } \mathbf{x}_0} \underbrace{\mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})}}_{\text{diffusion noise } \boldsymbol{\epsilon}} \underbrace{w(t)}_{\text{loss weighting function}} \underbrace{||\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \boldsymbol{\epsilon}||_2^2}_{\text{noise prediction objective}}$$

$$\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) = -\frac{\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\sigma_t}$$

Ho et al., "Denoising Diffusion Probabilistic Models", *NeurIPS* 2020
Kingma and Gao, "Understanding Diffusion Objectives as the ELBO with Simple Data Augmentation", NeurIPS, 2023

Synthesis with SDE vs. ODE





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

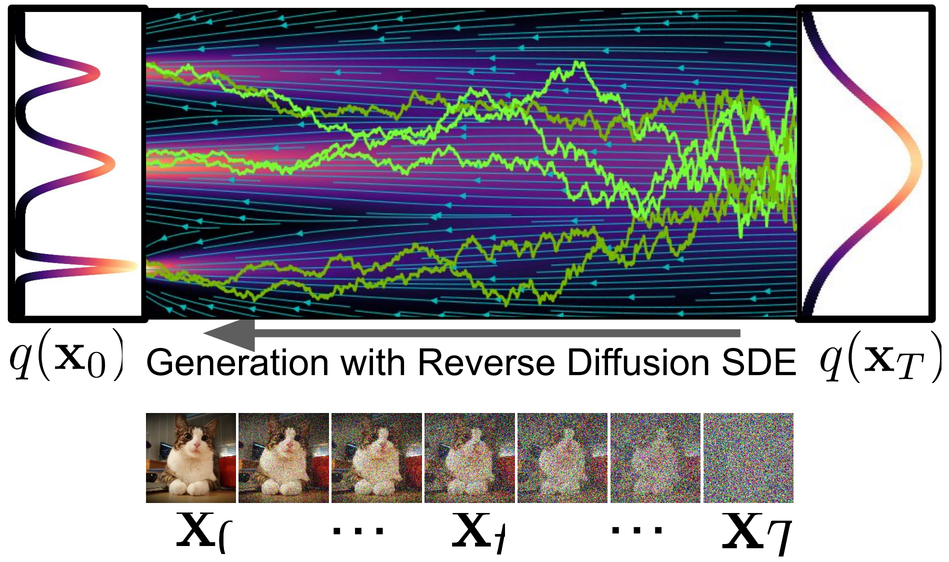
$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$$

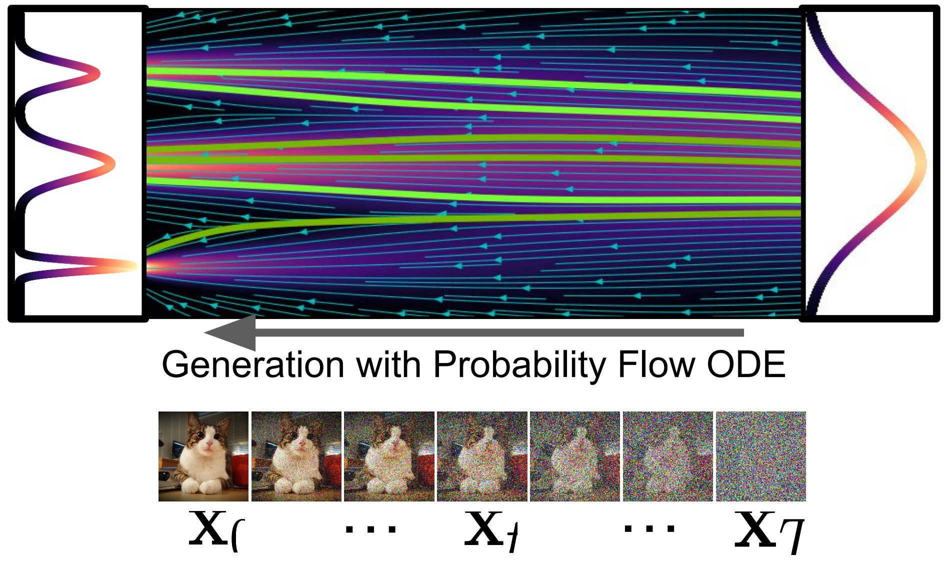
- **Generative Probability Flow ODE (deterministic):**

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

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

Synthesis with SDE vs. ODE – 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 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:

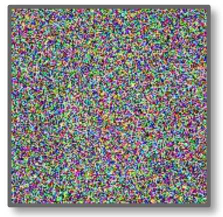
$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t$$

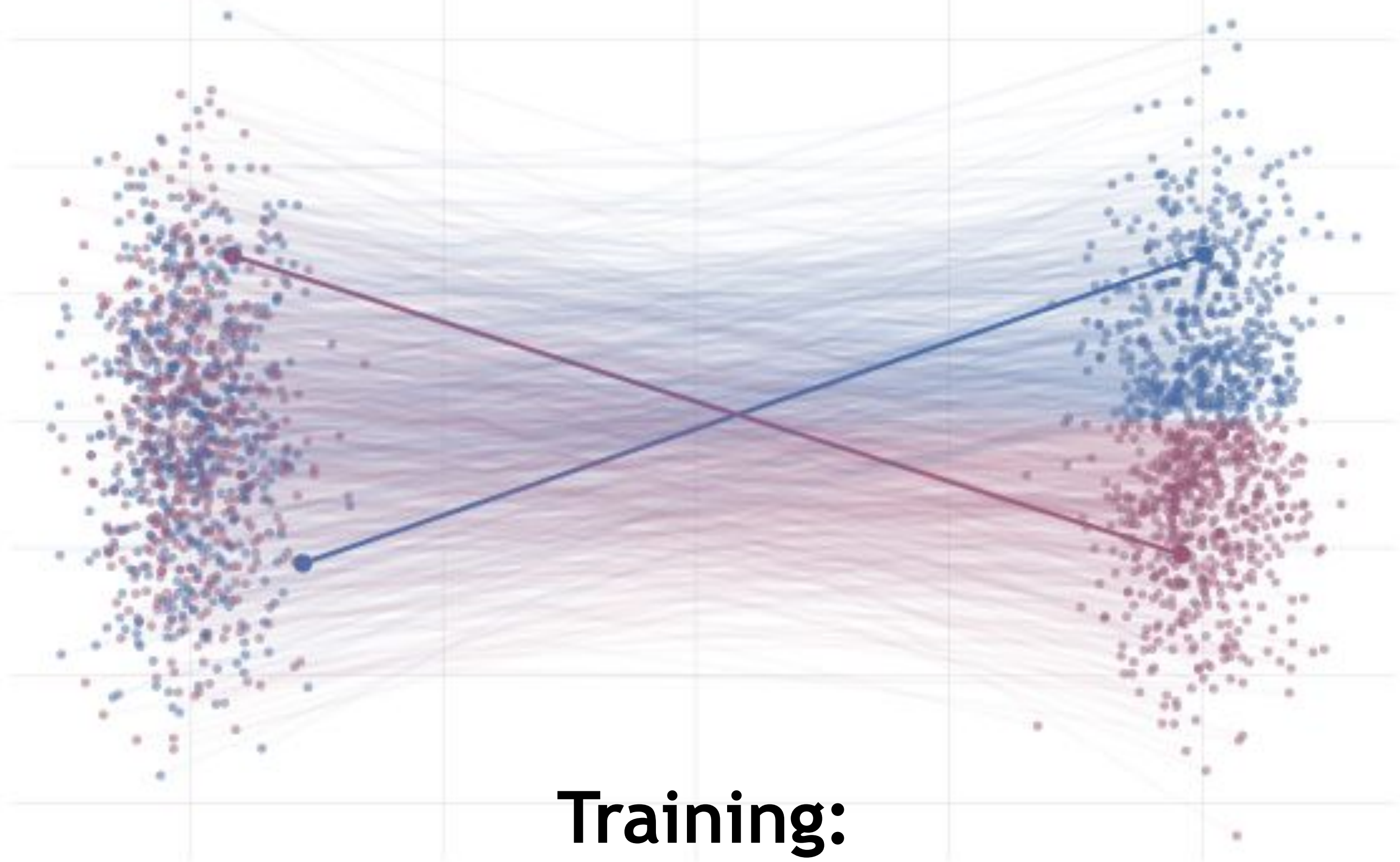
- *Pro:* Can leverage fast ODE solvers. Best when targeting very fast sampling.
- *Con:* No "stochastic" error correction, often slightly lower performance than stochastic sampling.

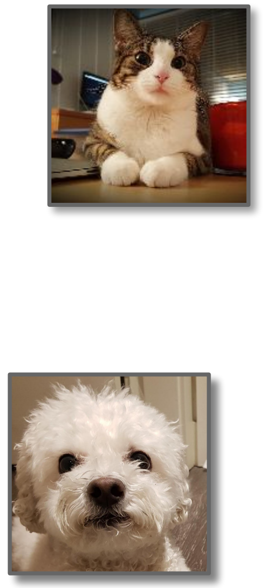
Karras et al., "Elucidating the Design Space of Diffusion-Based Generative Models", *NeurIPS*, 2022

Flow Matching

Learning the Vector Field between Noise and Data Directly







Training:

Interpolate between random pairs of noise and data distributions and learn vector field .

Lipman et al., "Flow Matching for Generative Modeling", *ICLR*, 2023
Albergo et al., "Stochastic Interpolants: A Unifying Framework for Flows and Diffusions", arXiv, 2023
Gao et al, "Diffusion Meets Flow Matching: Two Sides of the Same Coin", https://diffusionflow.github.io/, 2024
https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html

Flow Matching

Learning the Vector Field between Noise and Data Directly

After training, ... distributions.

Flow matching and diffusion models **equivalent** when coupling data to Gaussian noise distribution.

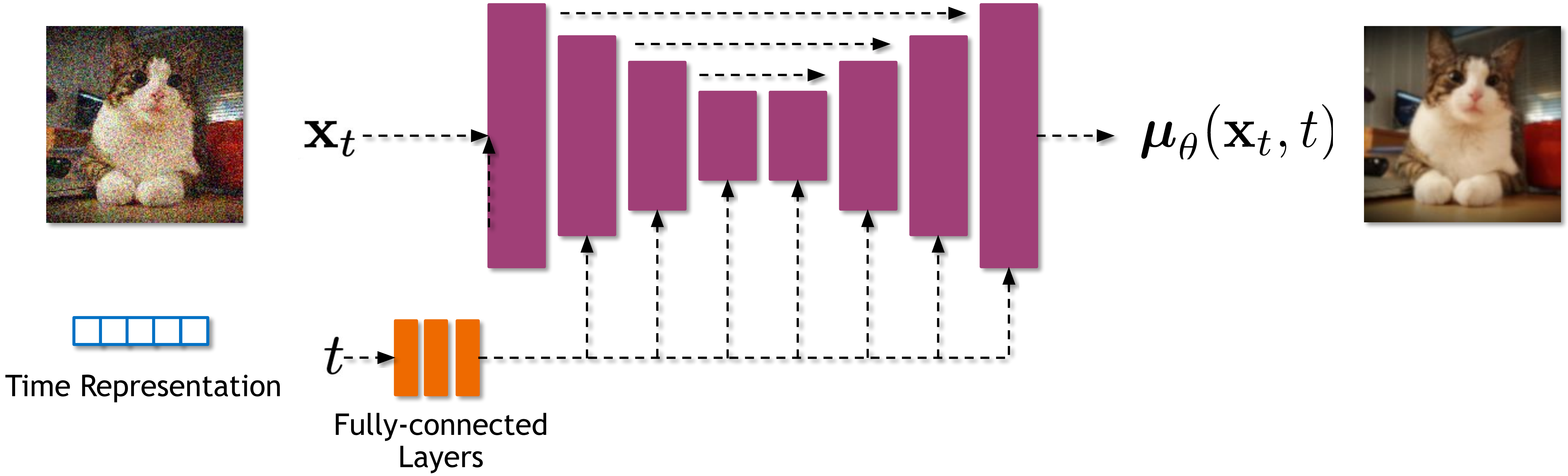
Flow matching popular due to **conceptual and implementation simplicity**:

Simply interpolate data and noise. Simulate ODE.
No diffusion or SDEs necessary.

Lipman et al., “Flow Matching for Generative Modeling”, *ICLR*, 2023
Albergo et al., “Stochastic Interpolants: A Unifying Framework for Flows and Diffusions”, arXiv, 2023
Gao et al, “Diffusion Meets Flow Matching: Two Sides of the Same Coin”, https://diffusionflow.github.io/, 2024
https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html

Diffusion Model Architectures

Diffusion models often use U-Net architectures with Conv. ResNet blocks, skip connections, and self-attention layers.



Time representation: sinusoidal positional embeddings or random Fourier features.

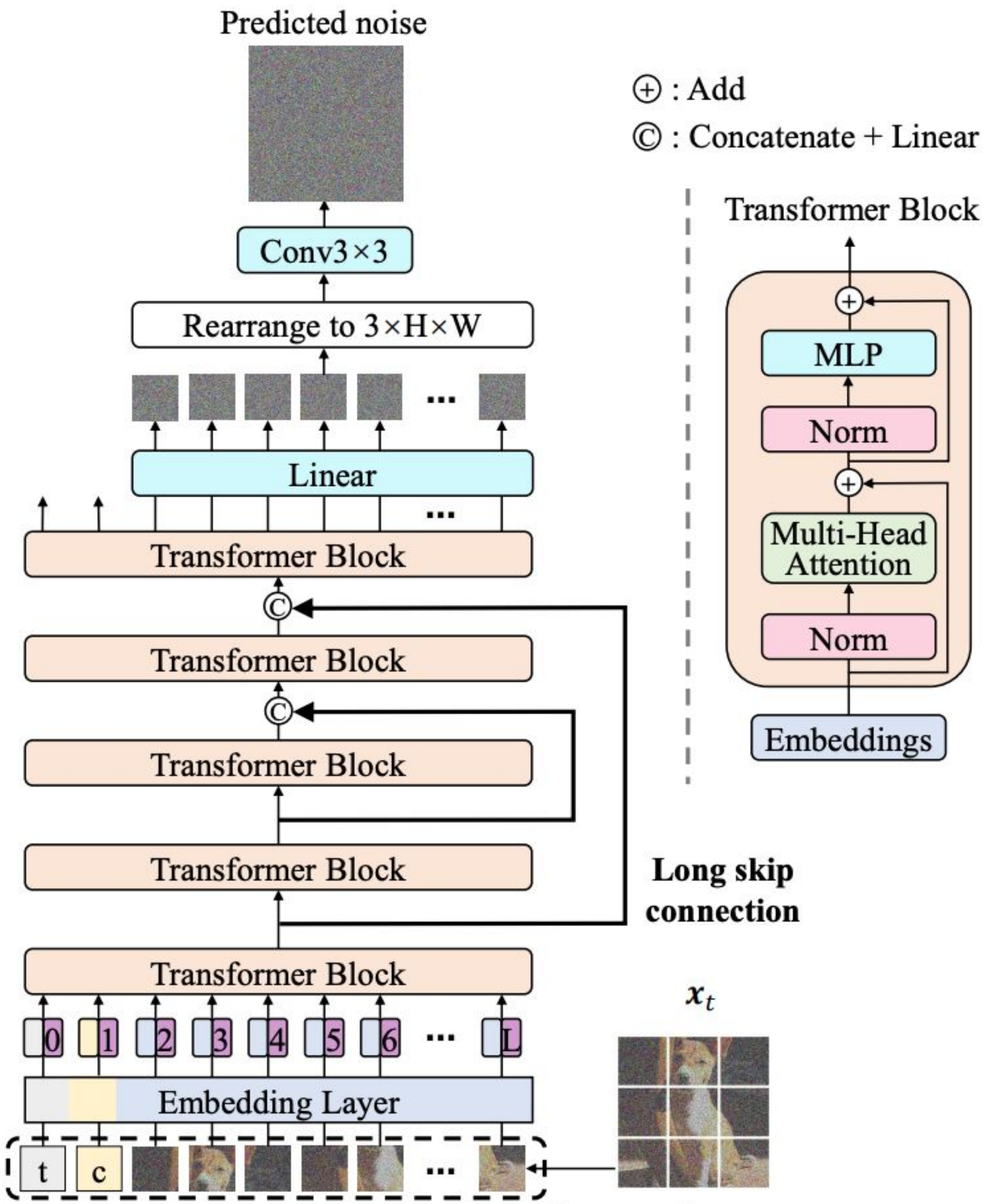
Time features are fed to residual blocks using either simple spatial addition or adaptive group normalization layers.

Ho et al., “Denoising Diffusion Probabilistic Models”, *NeurIPS*, 2020
Dhariwal and Nichol, “Diffusion Models Beat GANs on Image Synthesis”, *NeurIPS*, 2021
Karras et al., “Elucidating the Design Space of Diffusion-Based Generative Models”, *NeurIPS*, 2022
Hoogeboom et al., “simple diffusion: End-to-end diffusion for high resolution images”, *ICML*, 2023
Karras et al., “Analyzing and Improving the Training Dynamics of Diffusion Models”, *CVPR*, 2024

Diffusion Transformers

Vision transformer backbone for diffusion models:

- Noise and denoise tokenized image patches.
- Similar architectures as in LLMs.



Bao et al., “All are Worth Words: A ViT Backbone for Diffusion Models”, *CVPR*, 2023
Peebles and Xie, “Scalable Diffusion Models with Transformers”, *ICCV*, 2023
Hatamizadeh et al., “DiffiT: Diffusion Vision Transformers for Image Generation”, 2023

A Crucial Trick: Classifier(-free) Guidance

- **Unconditional generation:** $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$ used score during iterative generation.
- **Conditional generation:** $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|c)$
- **Classifier guidance:** Train classifier $p(c|\mathbf{x}_t)$ and use $\nabla_{\mathbf{x}_t} [\log p(\mathbf{x}_t|c) + \omega \log p(c|\mathbf{x}_t)]$

➡ approximately samples from $\tilde{p}(\mathbf{x}_t|c) \propto p(\mathbf{x}_t|c)\, p(c|\mathbf{x}_t)^{\omega}$

- **Classifier-free guidance:** Construct implicit classifier $p(c|\mathbf{x}_t) \propto \frac{p(\mathbf{x}_t|c)}{p(\mathbf{x}_t)}$

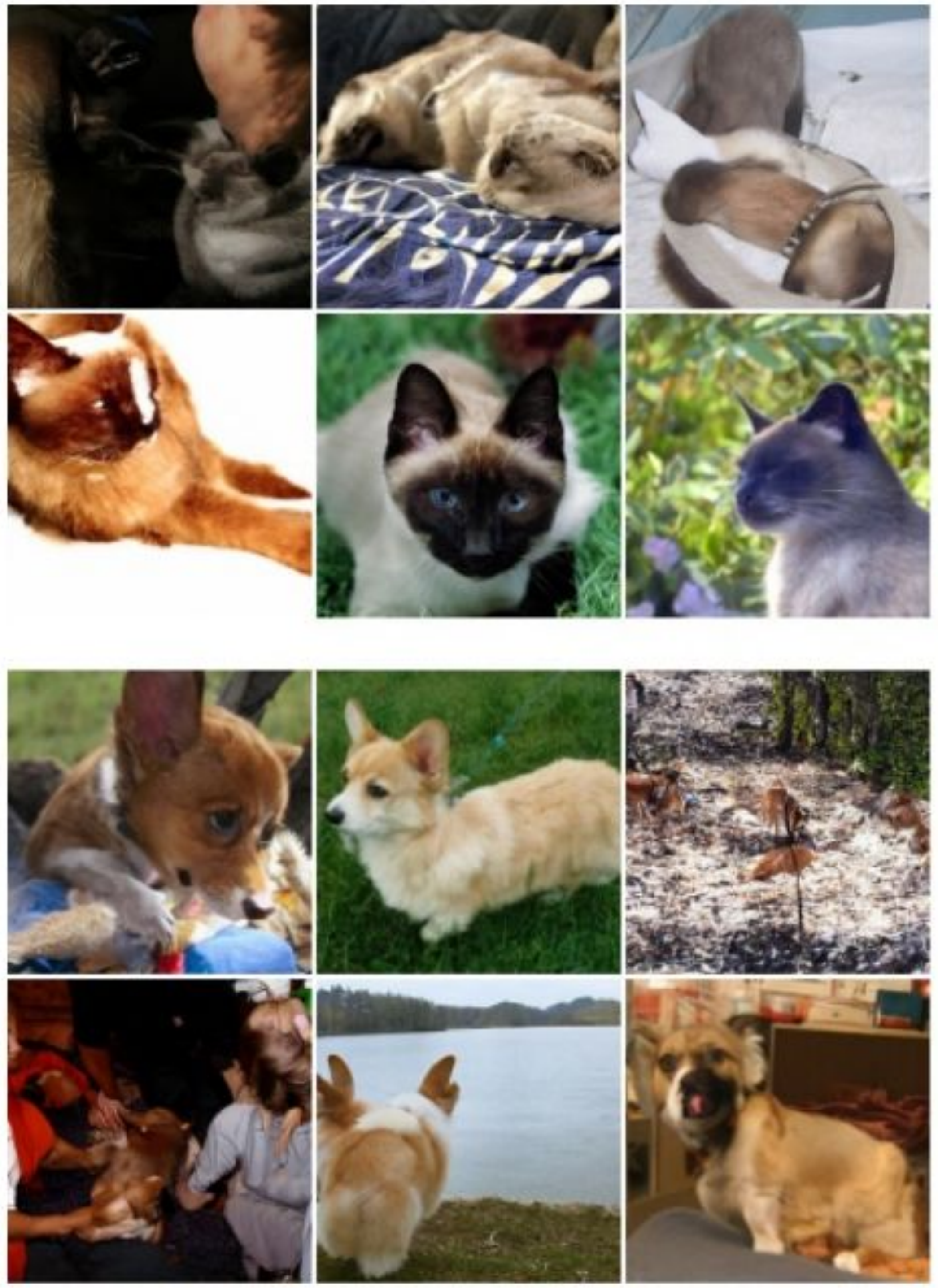
Conditional diffusion model

Unconditional diffusion model

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

Dhariwal and Nichol, "Diffusion Models Beat GANs on Image Synthesis", *NeurIPS*, 2021
Ho and Salimans, "Classifier-Free Diffusion Guidance", *arXiv*, 2021
Karras et al., "Guiding a Diffusion Model with a Bad Version of Itself", *NeurIPS*, 2024

A Crucial Trick: Classifier(-free) Guidance



Non-guided samples

Guided samples $(\omega = 3.0)$

Dhariwal and Nichol, “Diffusion Models Beat GANs on Image Synthesis”, *NeurIPS*, 2021
Ho and Salimans, “Classifier-Free Diffusion Guidance”, *arXiv*, 2021
Karras et al., “Guiding a Diffusion Model with a Bad Version of Itself”, *NeurIPS*, 2024

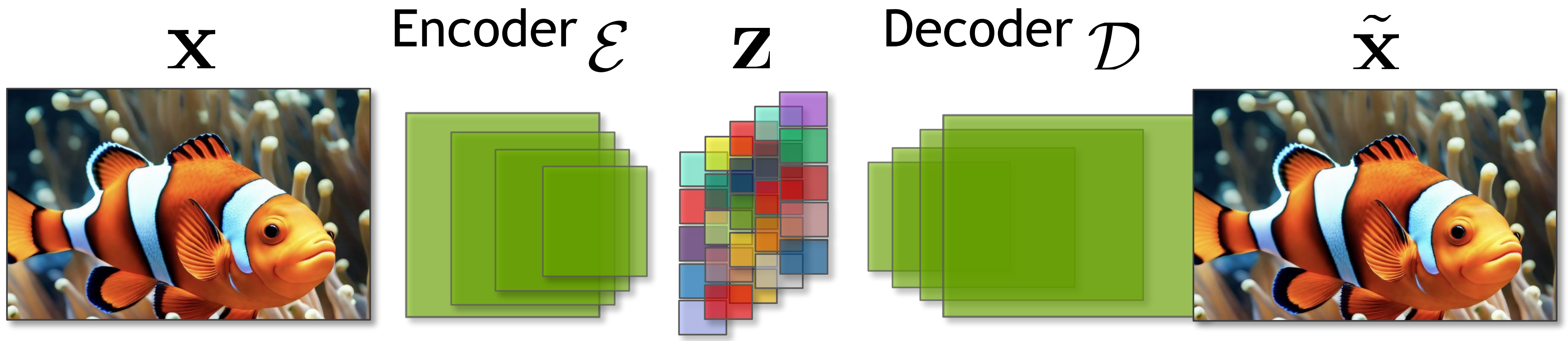
Latent Diffusion Models

Map Data into Compressed Latent Space. Train Diffusion Model efficiently in Latent Space.

- Stage 1:

Train Autoencoder

$$\tilde{\mathbf{x}} = \mathcal{D}(\mathcal{E}(\mathbf{x}))$$

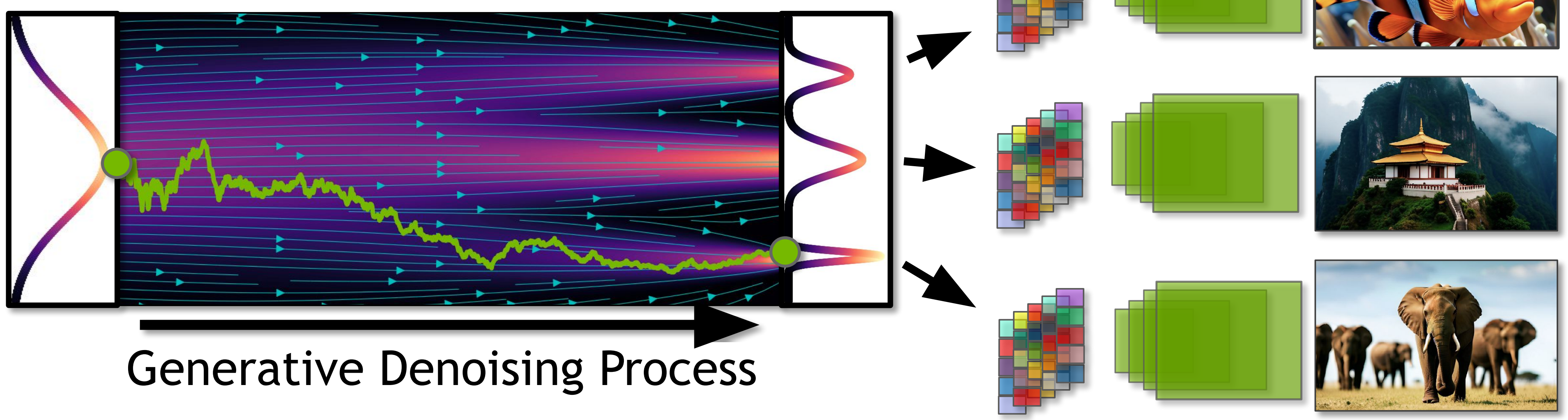


Pixelwise and/or Visual Feature Space (LPIPS) Reconstruction Objective

- Stage 2:

Latent embedding distribution modeled with Diffusion Model

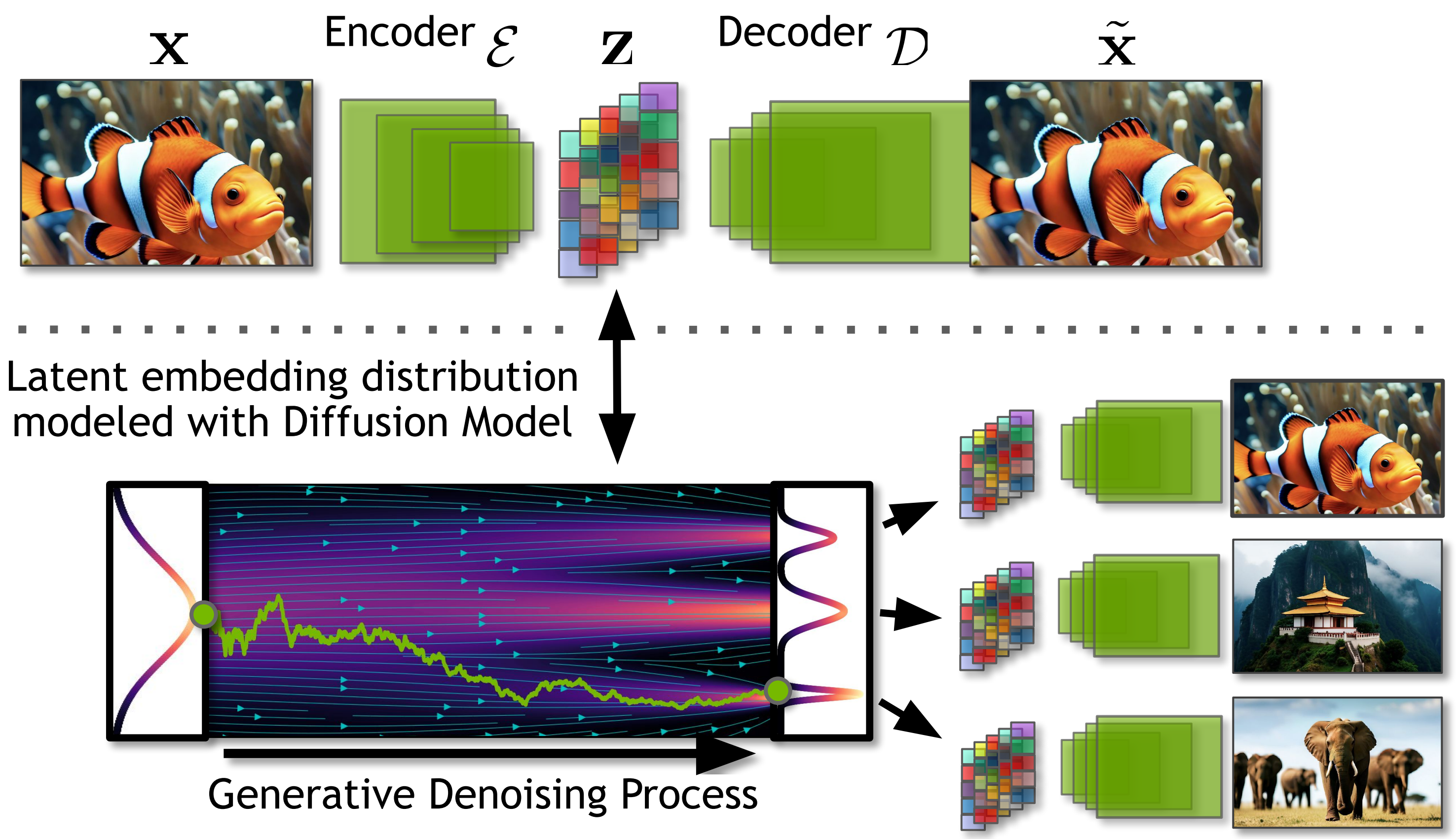
Train **Latent** Diffusion Model



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
Sinha et al., "D2C: Diffusion-Denoising Models for Few-shot Conditional Generation", *NeurIPS*, 2021
Mittal et al., "Symbolic Music Generation with Diffusion Models", *ISMIR*, 2021

Latent Diffusion Models

Map Data into Compressed Latent Space. Train Diffusion Model efficiently in Latent Space.



Advantages:

1. *Compressed latent space:* Train diffusion model in **lower resolution** latent space ➡ **computationally more efficiently**
2. *Regularized smooth/compressed latent space:* **Easier task** for diffusion model and **faster sampling**
3. *Flexibility*: **Autoencoder can be tailored to data** (images, video, text, graphs, 3D point clouds, meshes, 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
Sinha et al., "D2C: Diffusion-Denoising Models for Few-shot Conditional Generation", *NeurIPS*, 2021
Mittal et al., "Symbolic Music Generation with Diffusion Models", *ISMIR*, 2021

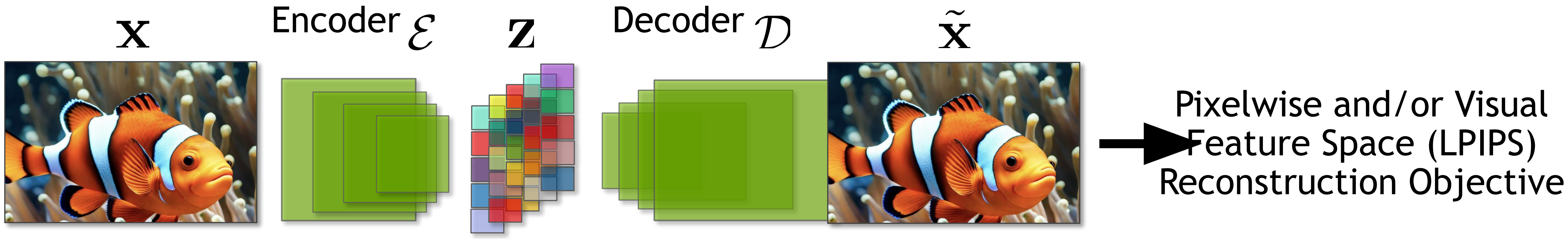
Latent Diffusion Models

Add Adversarial Patch-based Discriminator on top of Reconstruction Loss for Perceptual Compression

- Stage 1:

Train Autoencoder

$$\tilde{\mathbf{x}} = \mathcal{D}(\mathcal{E}(\mathbf{x}))$$



- Stage 2:

Train Latent Diffusion Model

Rombach et al., “High-Resolution Image Synthesis with Latent Diffusion Models”, *CVPR*, 2022

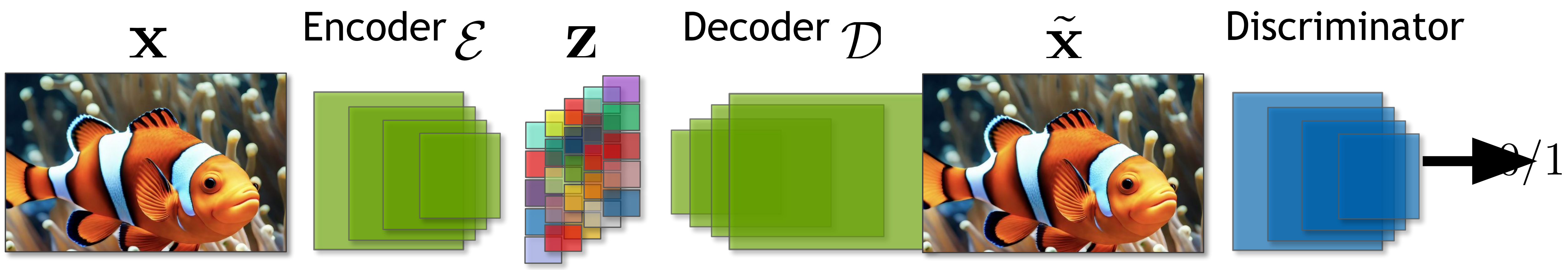
Latent Diffusion Models

Add Adversarial Patch-based Discriminator on top of Reconstruction Loss for Perceptual Compression

- Stage 1:

Train Autoencoder

$$\tilde{\mathbf{x}} = \mathcal{D}(\mathcal{E}(\mathbf{x}))$$



- Stage 2:

Train Latent Diffusion Model

Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models", *CVPR*, 2022

Input

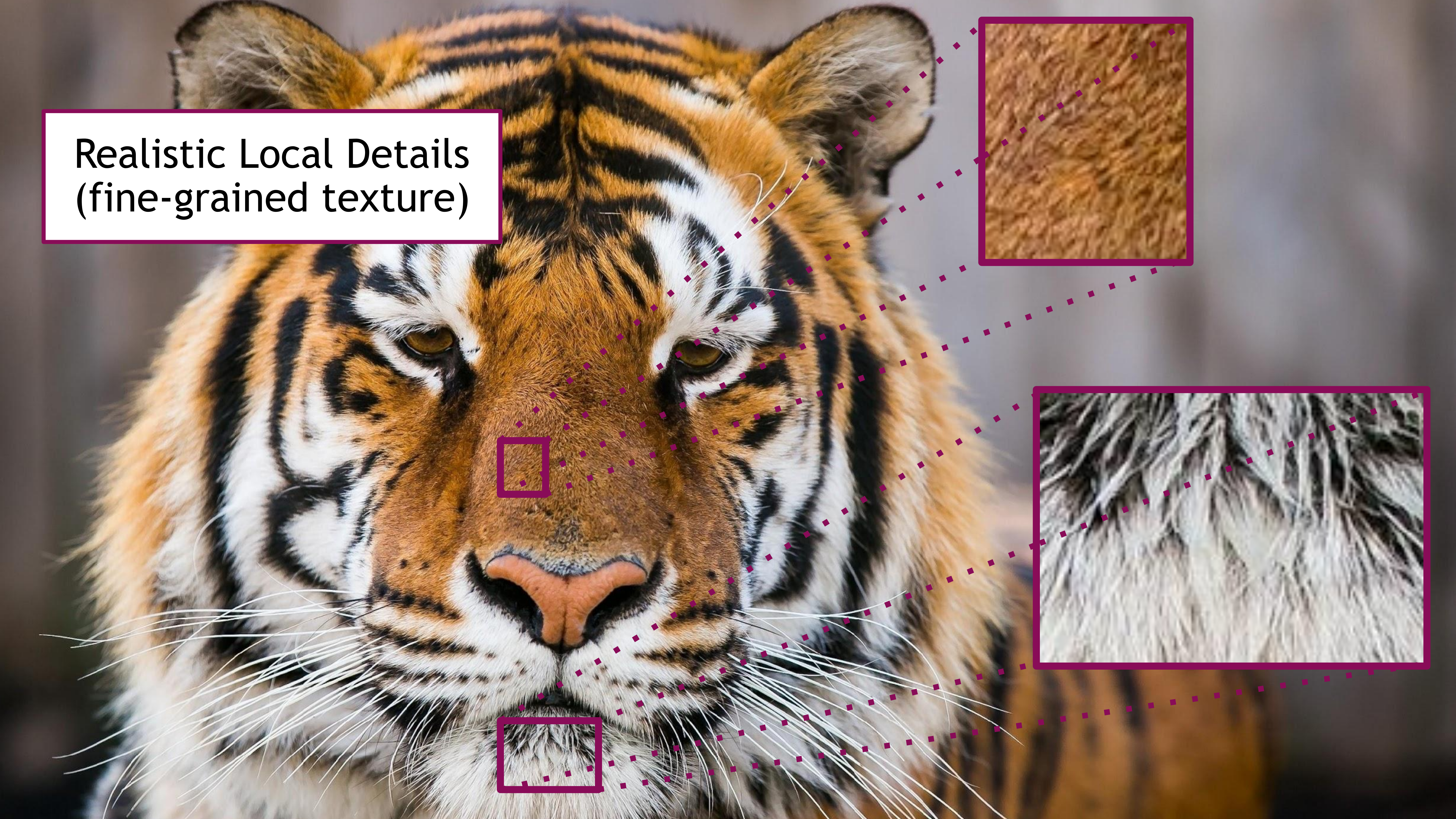
Reconstruction without Discriminator

Reconstruction with Discriminator

Rombach et al., “High-Resolution Image Synthesis with Latent Diffusion Models”, *CVPR*, 2022

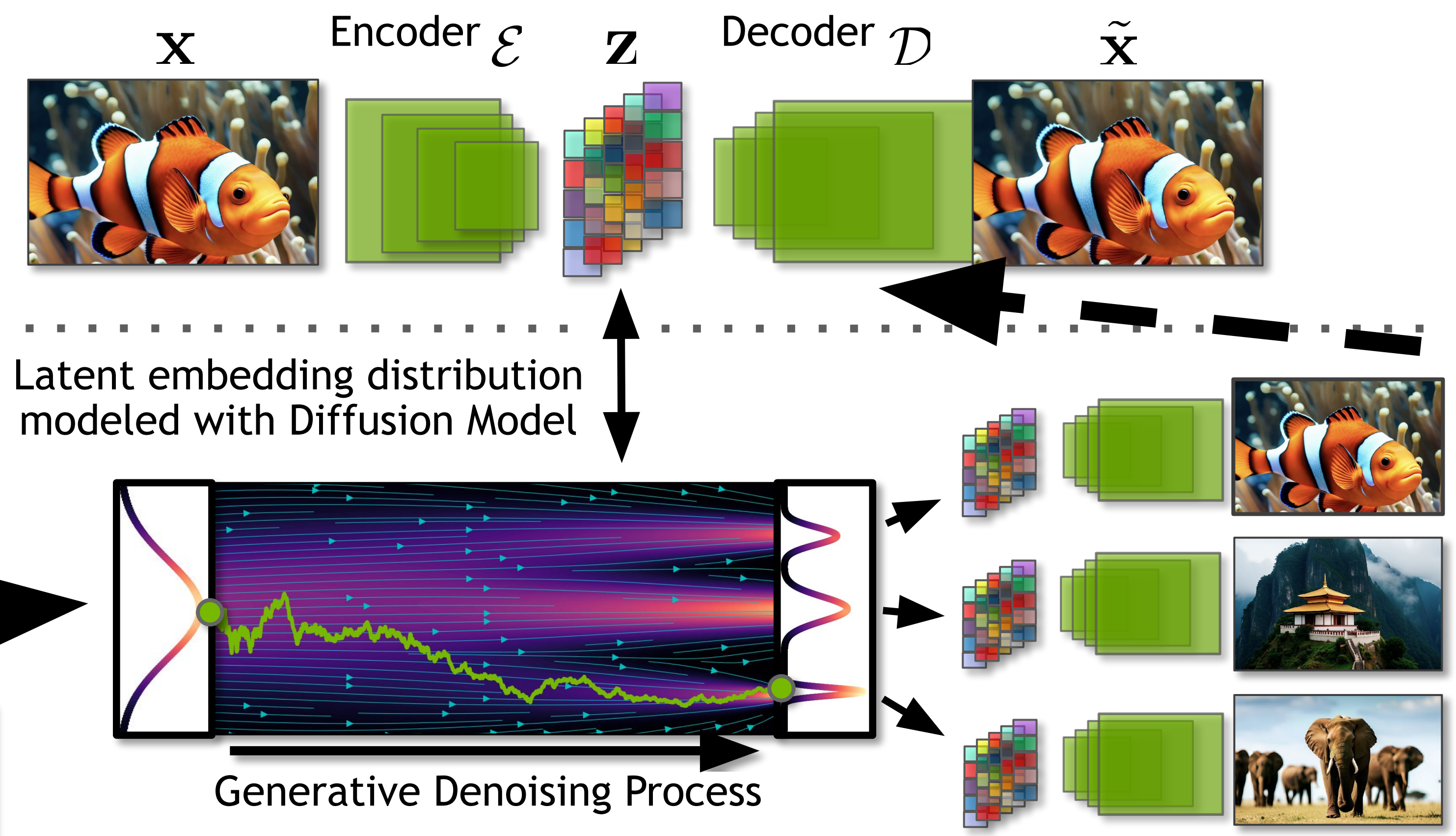
What makes an image look realistic and high-quality?


Realistic Global Structure
(correct placement of ears,
eyes, fur pattern, etc.)


Realistic Local Details
(fine-grained texture)

Latent Diffusion Models

Map Data into Compressed Latent Space. Train Diffusion Model efficiently in Latent Space.



Local "imperceptible" details generated by decoder (with adversarial objective).

Global semantic structure modeled by latent diffusion model.

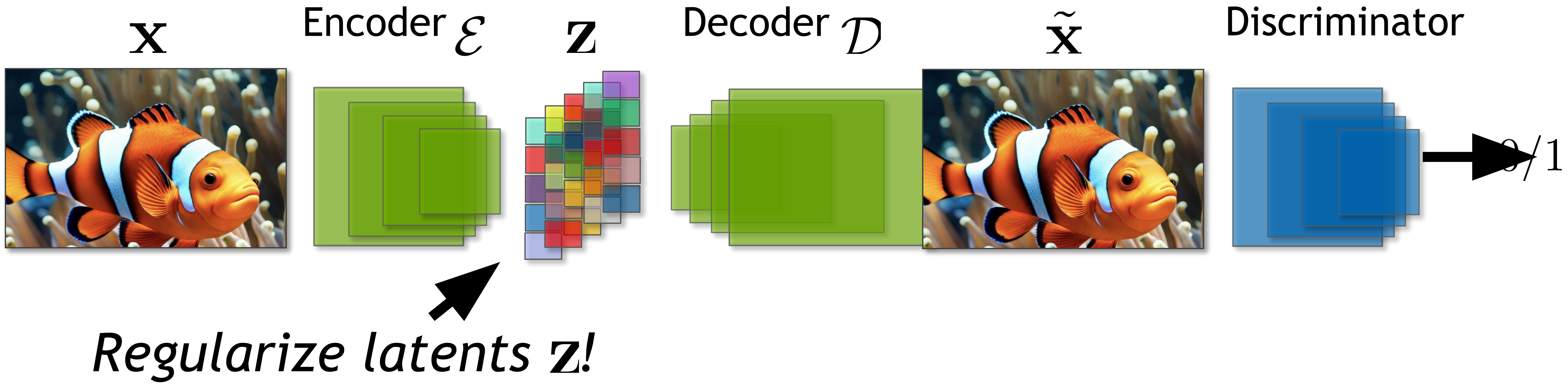
Latent space compression needs to be tuned carefully!

Balance between what content the latent DM needs to model and what is generated by decoder!

Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models", *CVPR*, 2022
Ho et al., "Denoising Diffusion Probabilistic Models", *NeurIPS* 2020

Latent Space Regularization

Regularize Latent Space for better Compression and easier Training of Latent Space Diffusion Models



- **Option 1: Kullback-Leibler (KL) regularization**

  Parametrize encoder by diagonal Gaussian, regularize towards standard normal distribution (as in regular VAEs).

  Use very small weight for KL regularization term (weak regularization).

Encoder distribution:

$$q_{\mathcal{E}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mathcal{E}_\mu, \mathcal{E}_\sigma^2)$$

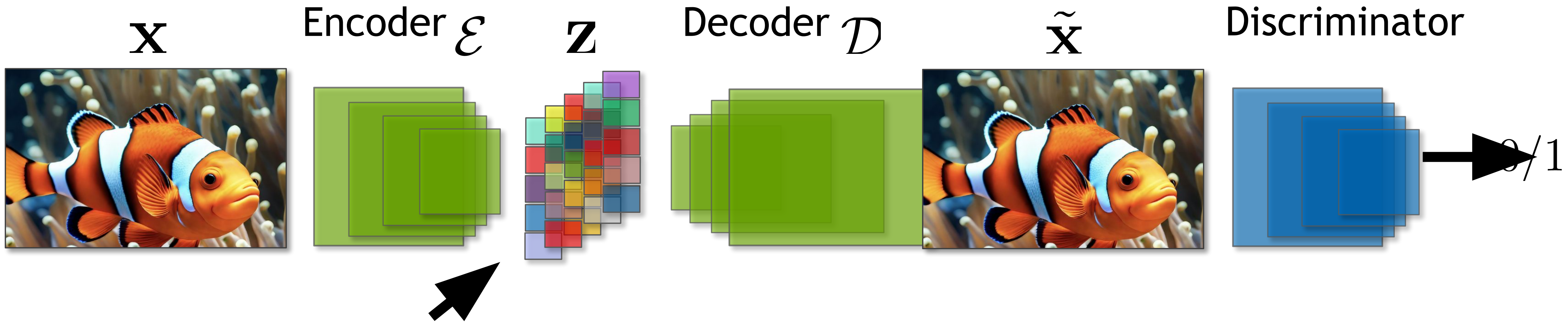
KL regularization in latent space:

$$\mathrm{KL}\left(q_{\mathcal{E}}(\mathbf{z}|\mathbf{x}) || \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})\right)$$

Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models", *CVPR*, 2022
Esser et al., "*Taming Transformers for High-Resolution Image Synthesis*", *CVPR*, 2021

Latent Space Regularization

Regularize Latent Space for better Compression and easier Training of Latent Space Diffusion Models

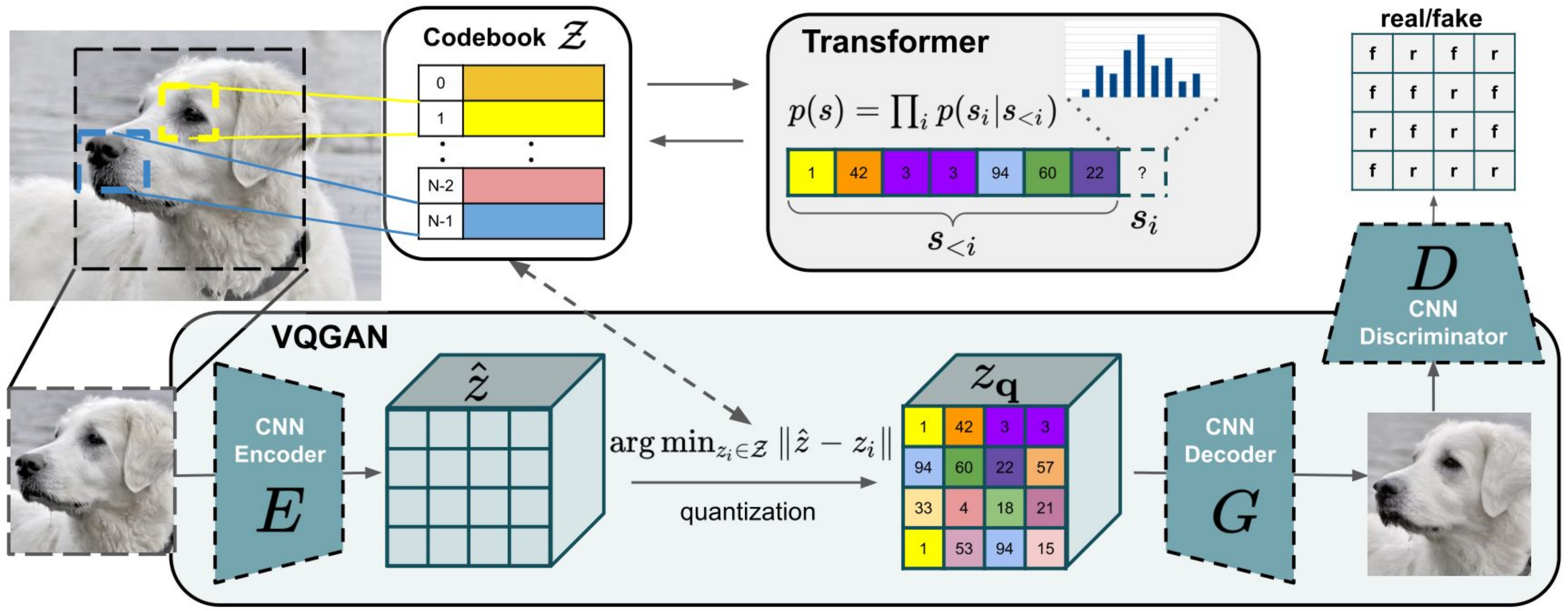


Regularize latents $\mathbf{z}$*!*

- **Option 2: Vector Quantization (VQ) regularization**

  Discretize latent encodings using finite-sized learnable codebook as in VQ-VAEs (implemented by vector-quantization layer in decoder).

  Use large codebook size (weak regularization).



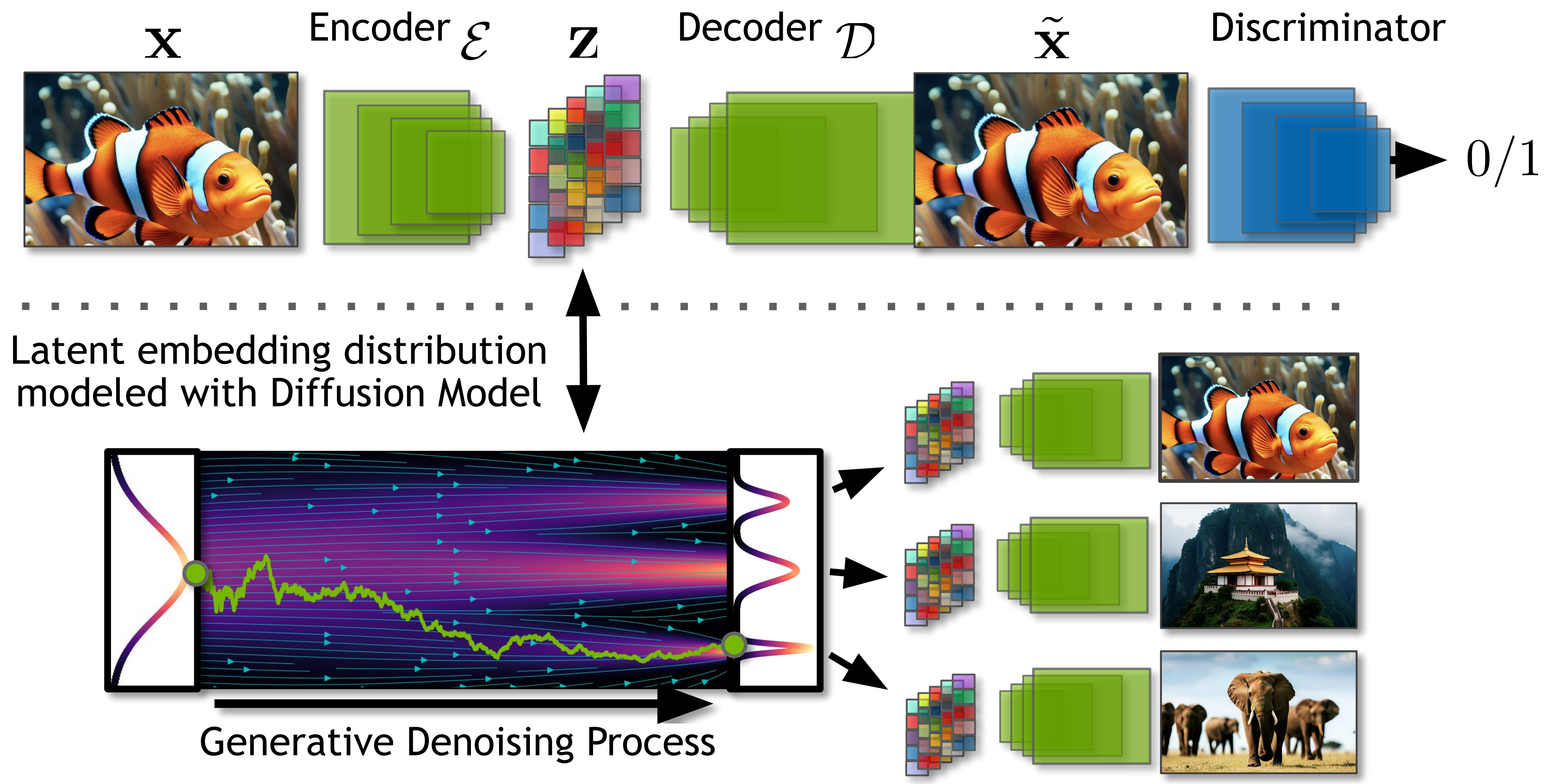
Rombach et al., “High-Resolution Image Synthesis with Latent Diffusion Models”, *CVPR*, 2022
Esser et al., “*Taming Transformers for High-Resolution Image Synthesis*”, *CVPR*, 2021

Latent Diffusion Models

Latent Diffusion Models offer Excellent Trade-off between Performance and Compute Demands

LDM "Recipe":

1. Train strong autoencoder
    - Compress…
      (downsampling factor / latent space regularization)
    - …while ensuring high visual quality on reconstructions
      ("upper bound" on synthesis quality)



Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models", *CVPR*, 2022

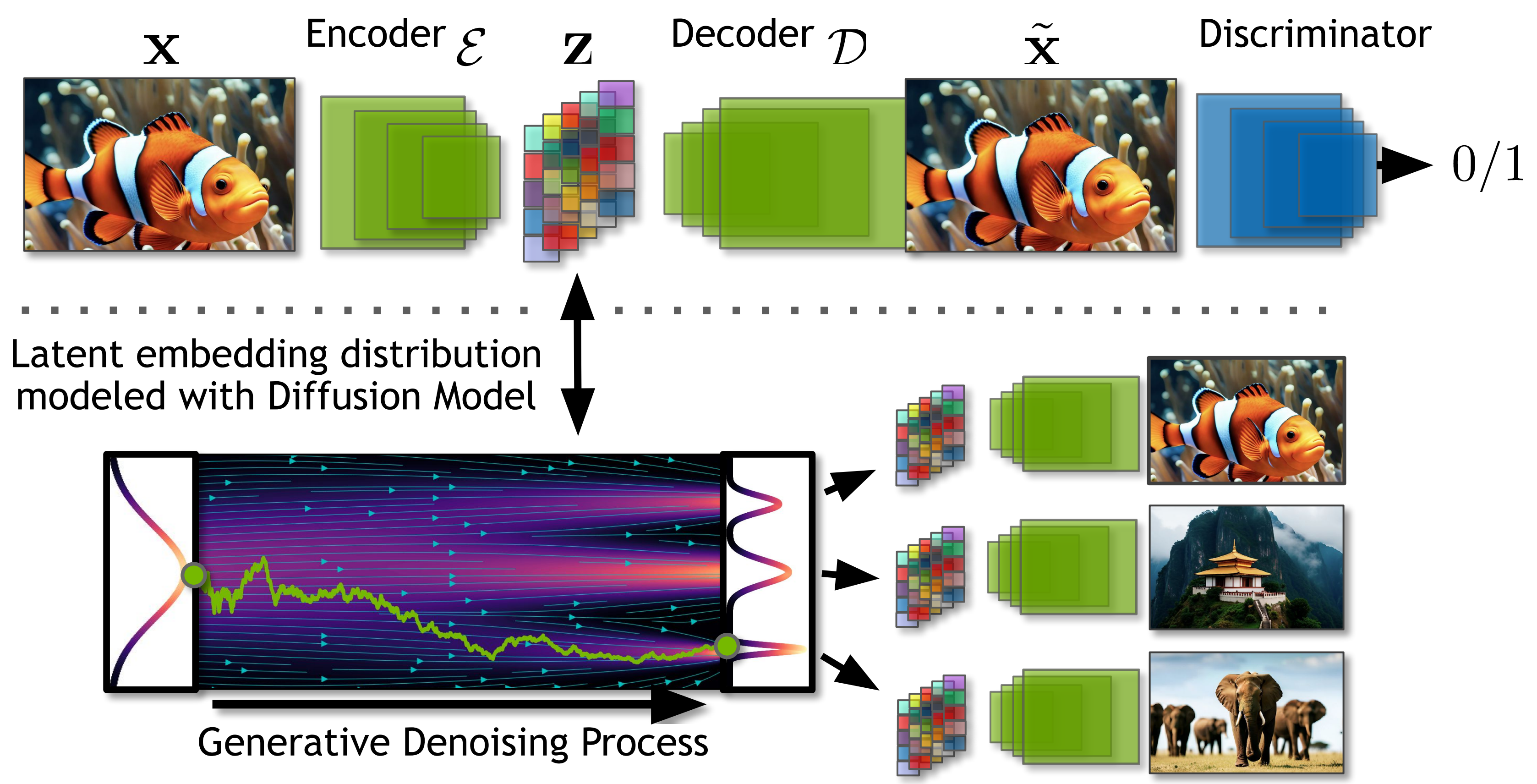
Latent Diffusion Models

Latent Diffusion Models offer Excellent Trade-off between Performance and Compute Demands

LDM "Recipe":

1. Train strong autoencoder
    - Compress…
      (downsampling factor / latent space regularization)
    - …while ensuring high visual quality on reconstructions
      ("upper bound" on synthesis quality)

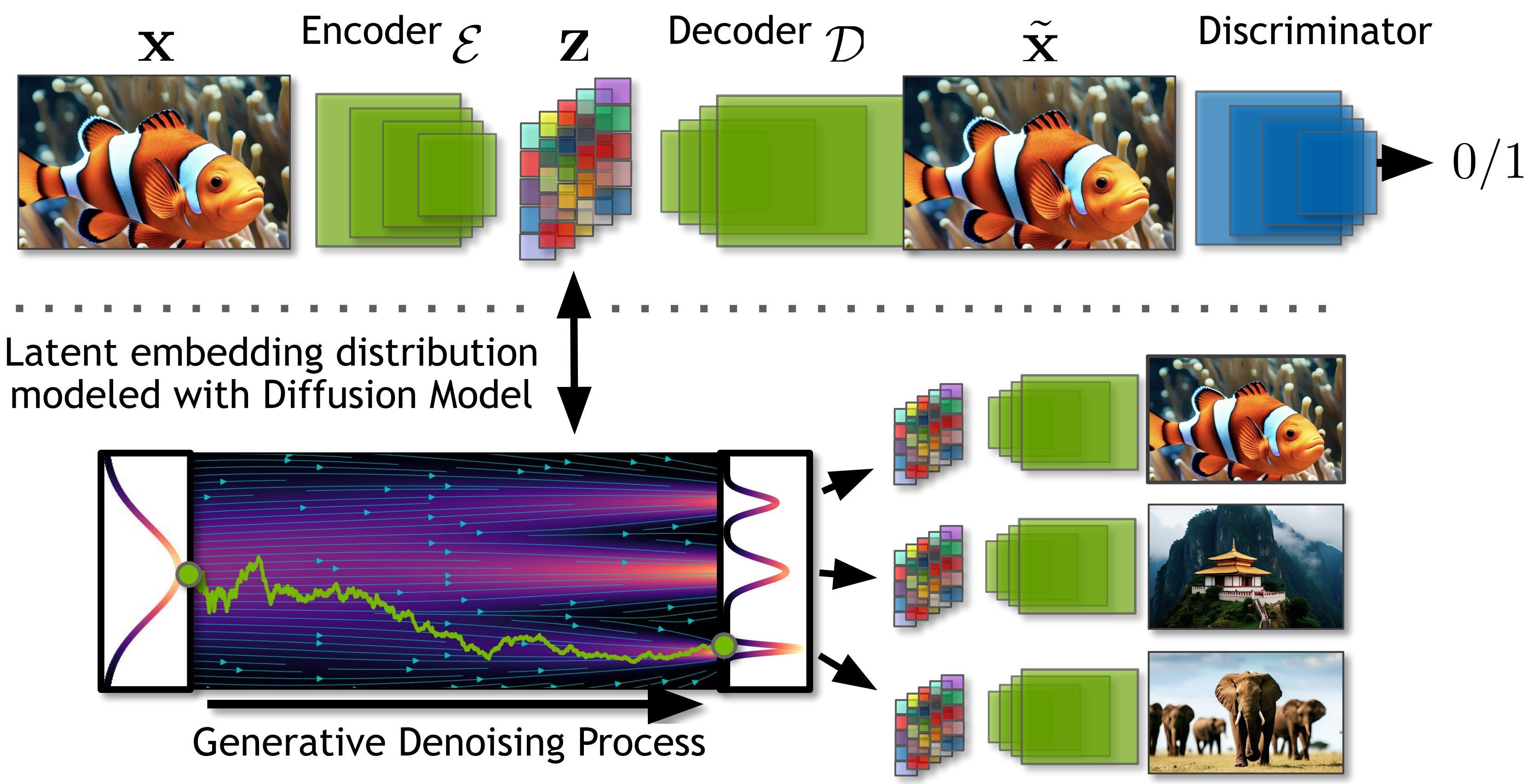
2. Train efficient latent diffusion model
    - Latent space compression/regularization makes diffusion model training easier ➡ but trade-off with respect quality? Not really…
    - …because discriminator ➡ high quality despite compression (re-generate details, not encode)!



Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models", *CVPR*, 2022

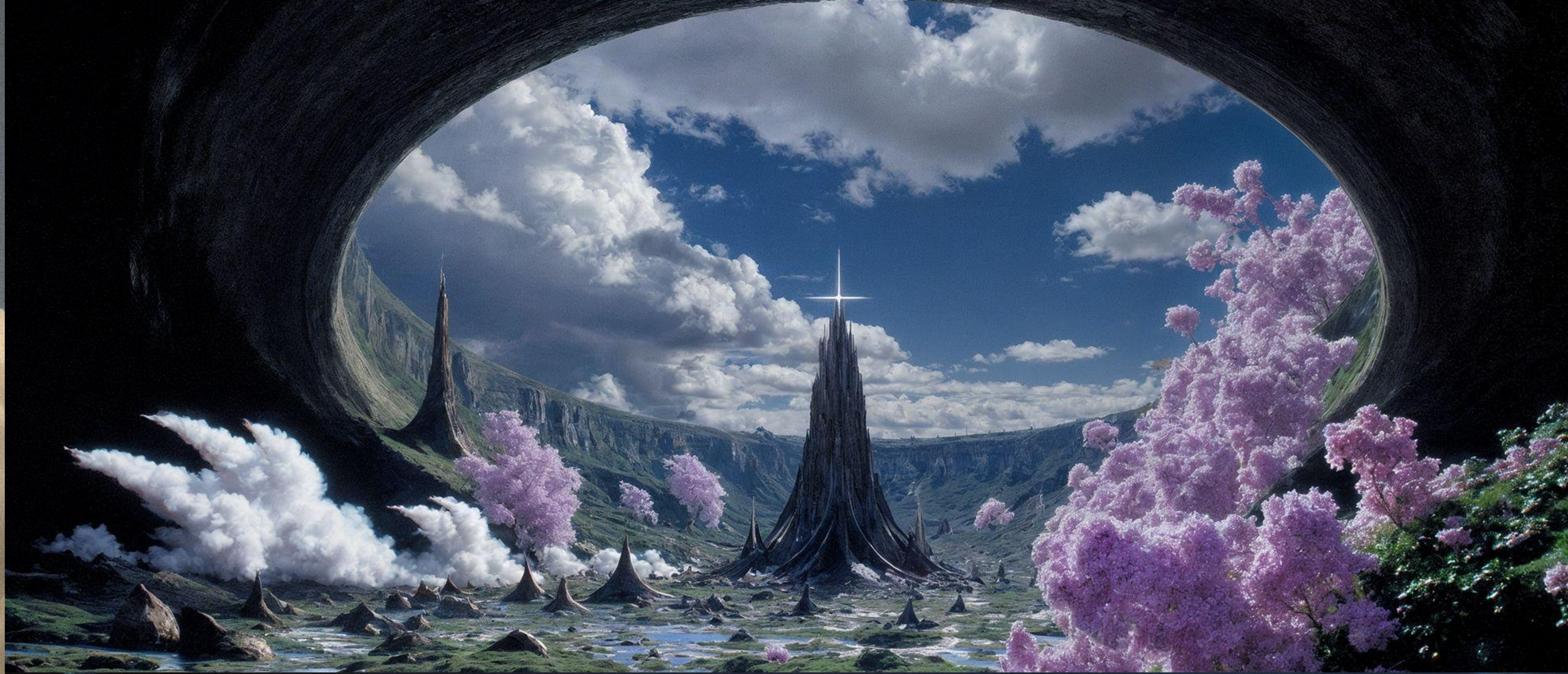
Latent Diffusion Models

Latent Diffusion Models offer Excellent Trade-off between Performance and Compute Demands

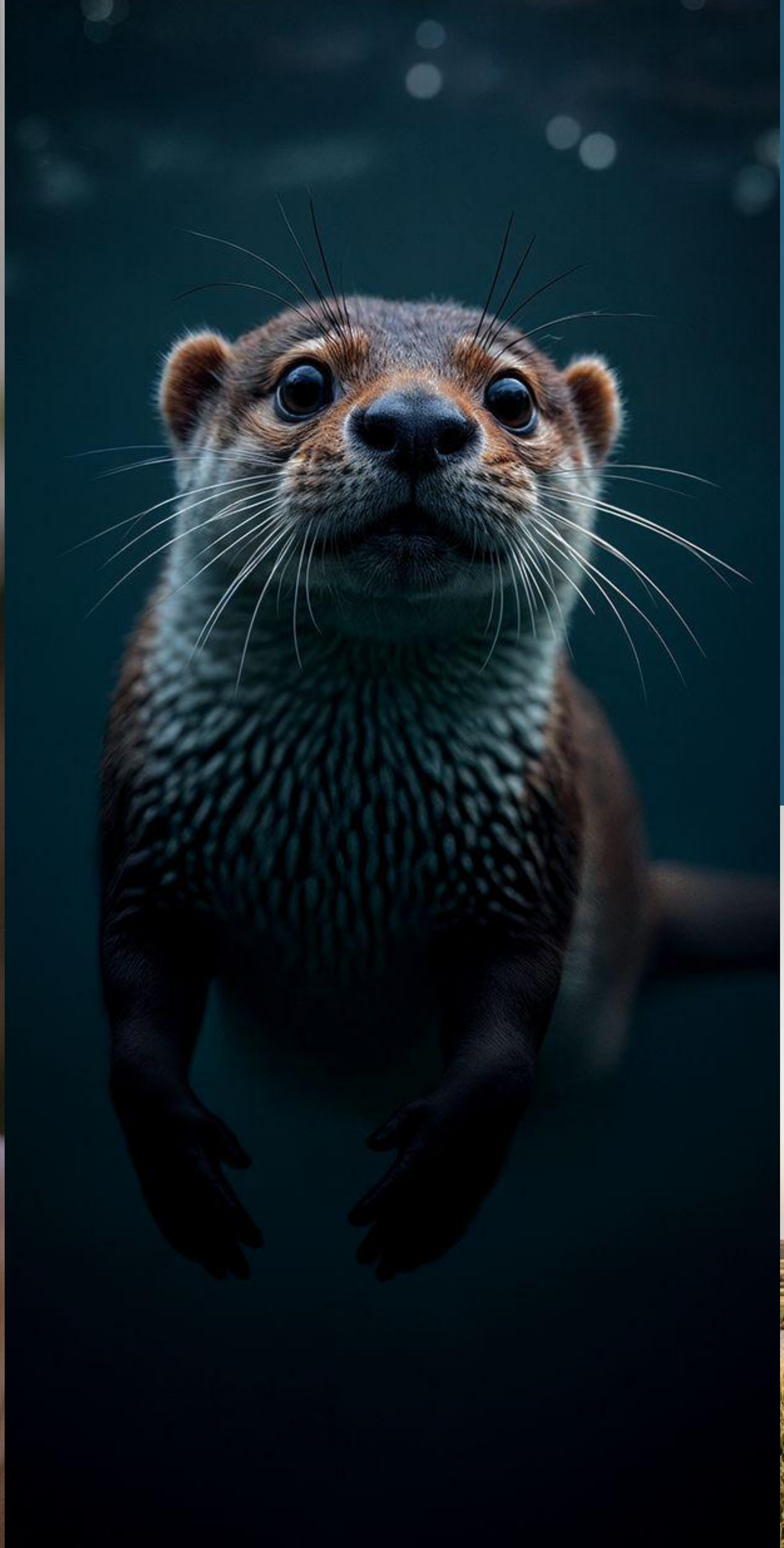


➡ LDM with appropriate regularization, compression, downsampling ratio and strong autoencoder reconstruction:

- Computationally efficient diffusion model in latent space (compression & lower resolution).
- Yet very high-performance (latent diffusion + autoencoder + discriminator = ❤️).
- Highly flexible (can adjust autoencoder for different tasks and data).


Black Forest Labs. https://bfl.ai






Black Forest Labs. https://bfl.ai

Overview

I. **Fundamentals (Karsten):**
   - Generation by Noising and Denoising
   - A Differential Equation-based Perspective
   - Flow Matching, Basic Architectures and Guidance
   - Latent Diffusion Models

II. **Applications (Ruiqi):**
   - What makes Diffusion Great?
   - Video Diffusion Models
   - Beyond Video Generation: World Models
   - 3D/4D Generation