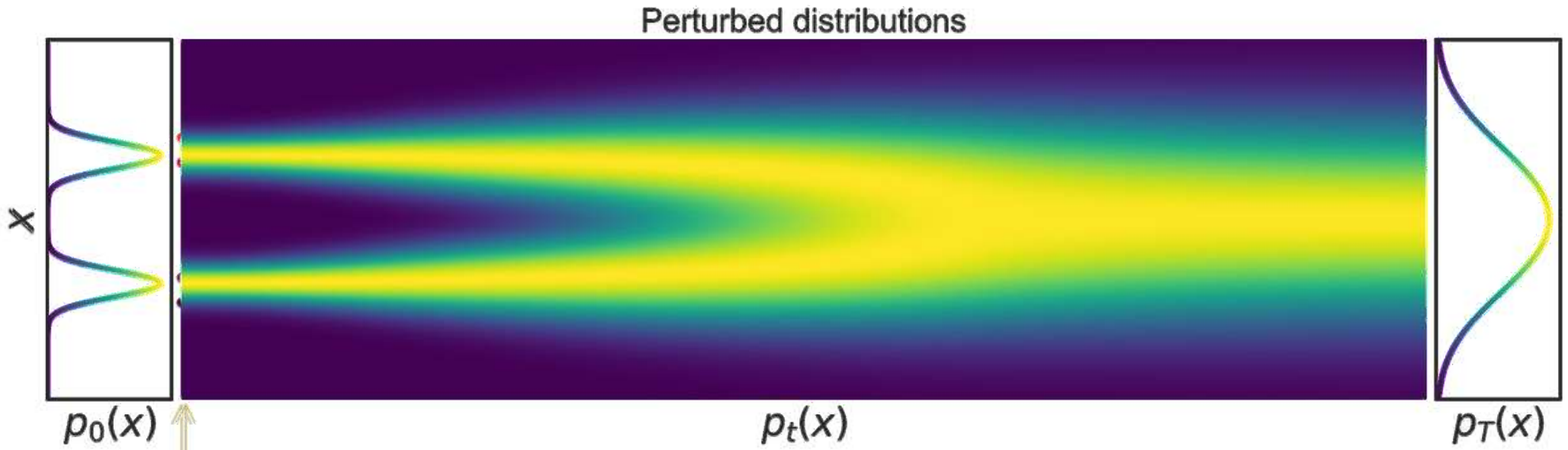
Recap.

Perturbing data with stochastic processes



Stochastic process

$\{\mathbf{x}_t\}_{t\in[0,T]}$

Probability densities

$\{p_t(\mathbf{x})\}_{t\in[0,T]}$

Stochastic differential equation (SDE)

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

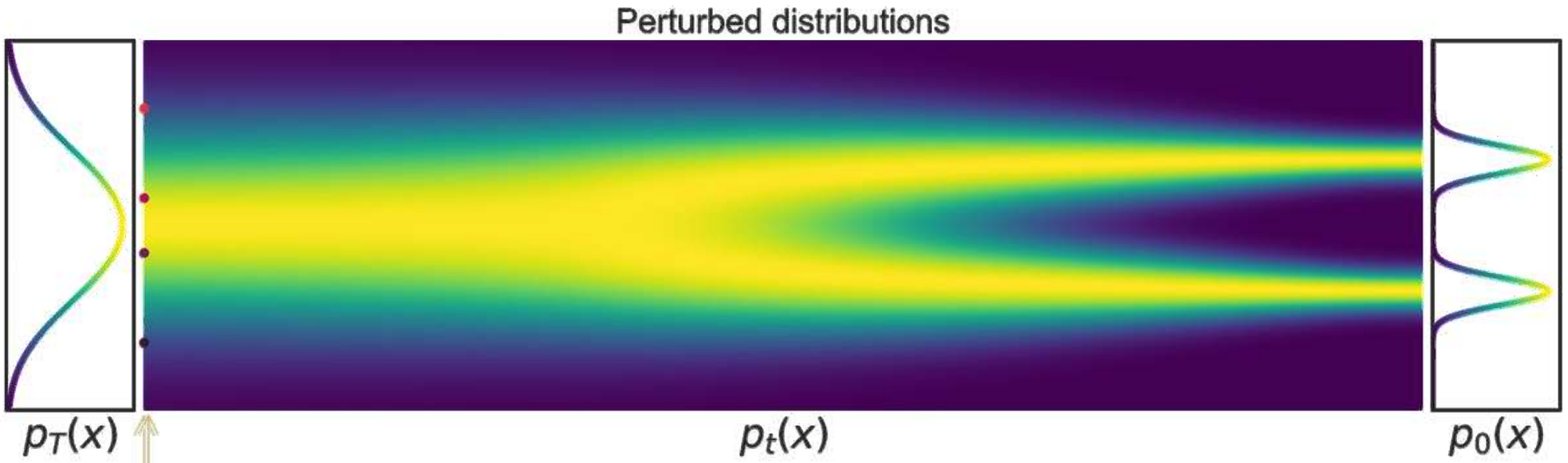
Deterministic drift — Infinitesimal noise

WLOG: Toy SDE

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

$$p_T(\mathbf{x}) \approx \pi(\mathbf{x})$$

Generation via reverse stochastic processes



$\pi(\mathbf{x}) \approx p_T(\mathbf{x})$

Forward SDE (t: 0→T)

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

Reverse SDE (t: T→0)

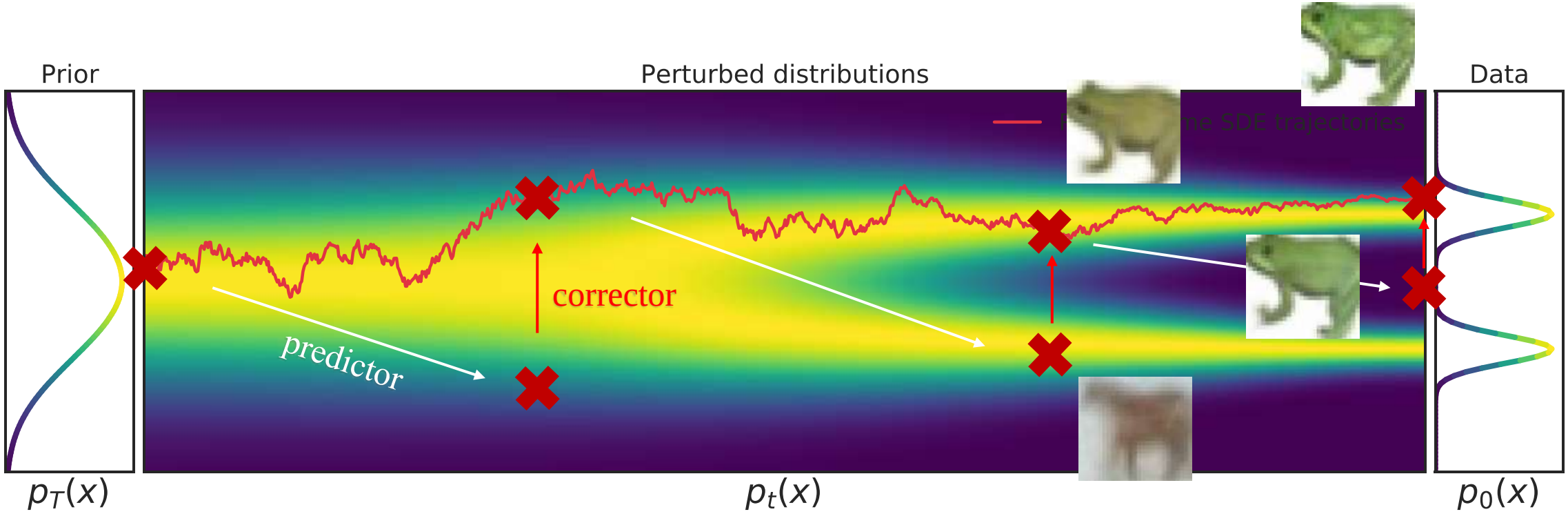
$$\mathrm{d}\mathbf{x}_t = -\sigma(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\bar{\mathbf{w}}_t$$

Infinitesimal noise in the reverse time direction

Score function!

Predictor-Corrector sampling methods

- Predictor-Corrector sampling.
  - **Predictor:** Numerical SDE solver
  - **Corrector:** Score-based MCMC



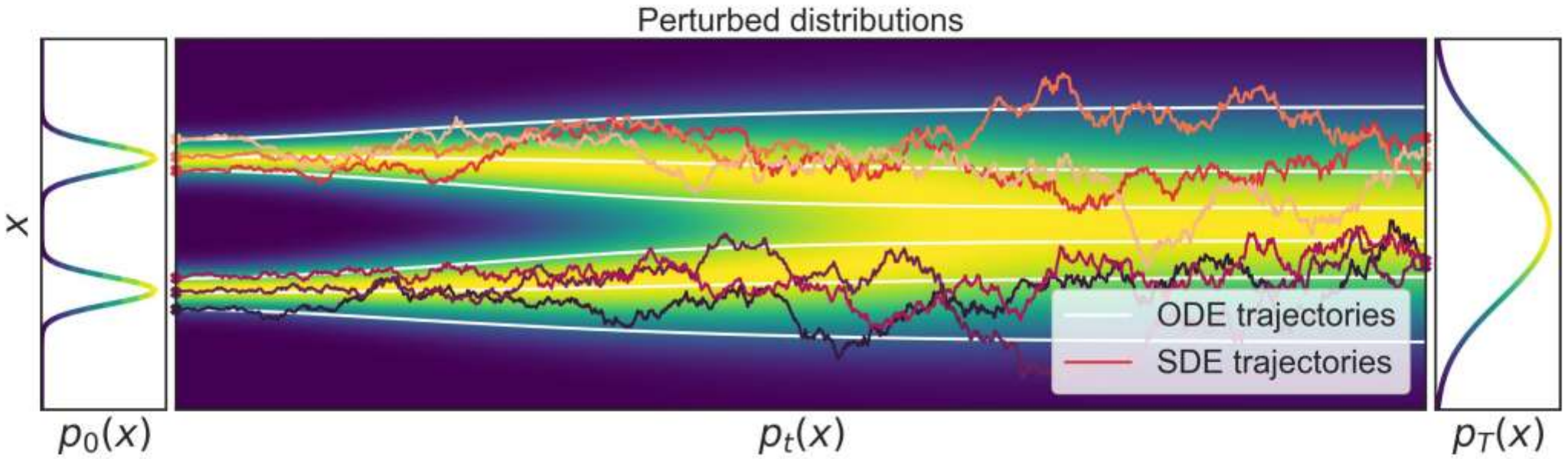
Predictor-Corrector sampling methods

- Predictor-Corrector sampling.
  - **Predictor:** Numerical SDE solver
  - **Corrector:** Score-based MCMC

Algorithm 3 PC sampling (VP SDE)

$\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $i = N - 1$ **to** 0 **do**
  $\mathbf{x}'_i \leftarrow (2 - \sqrt{1 - \beta_{i+1}})\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_{i+1}, i + 1)$
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\beta_{i+1}}\mathbf{z}$ — Predictor
  for $j = 1$ **to** M **do** — Corrector
    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_i, i) + \sqrt{2\epsilon_i}\mathbf{z}$
return $\mathbf{x}_0$

Converting the SDE to an ODE

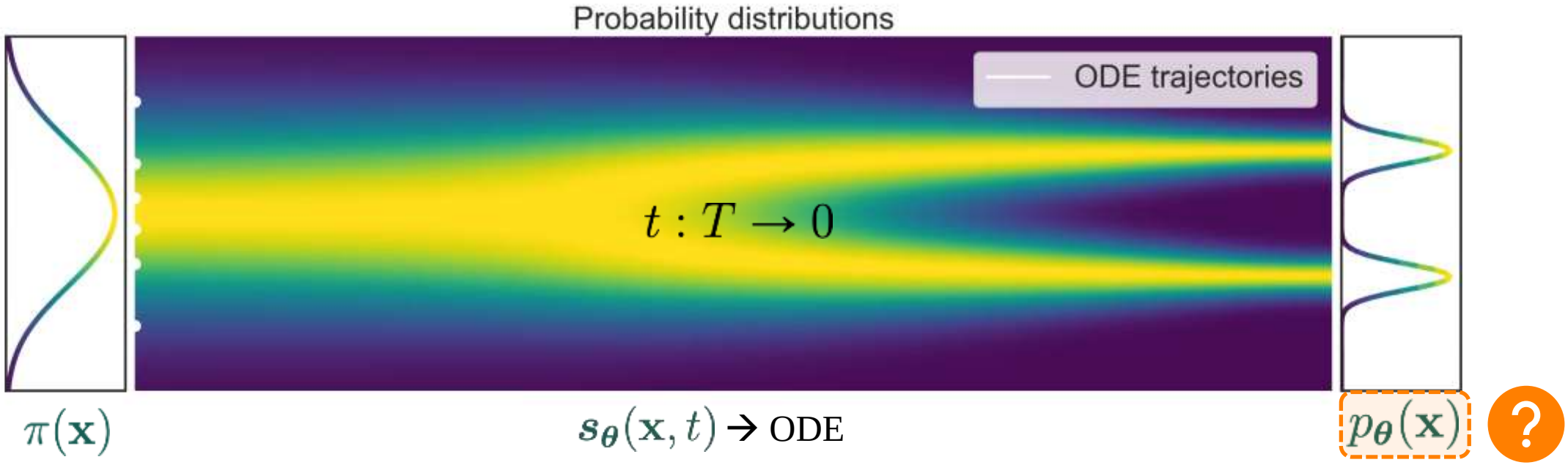


SDE **Ordinary differential equation (ODE)**

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t \quad \Longleftrightarrow \quad \frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -\frac{1}{2}\sigma(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$$

Score function
$\approx s_\theta(\mathbf{x}, t)$

Evaluating the probabilities with ODEs



Computing the probability density function (change of variables formula)

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_0) = \log \pi(\mathbf{x}_T) - \frac{1}{2}\int_0^T \sigma(t)^2 \operatorname{trace}\left(\nabla_{\mathbf{x}} \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\right) \mathrm{d}t$$

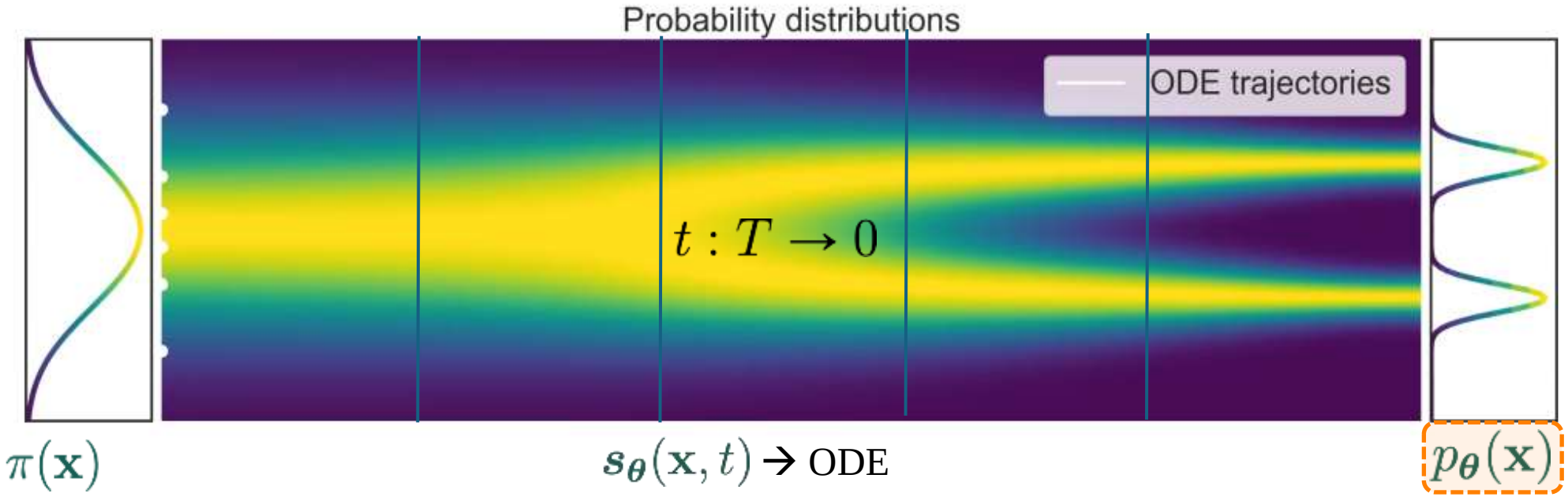
ODE solver

Computable in polynomial time

Unbiased estimator

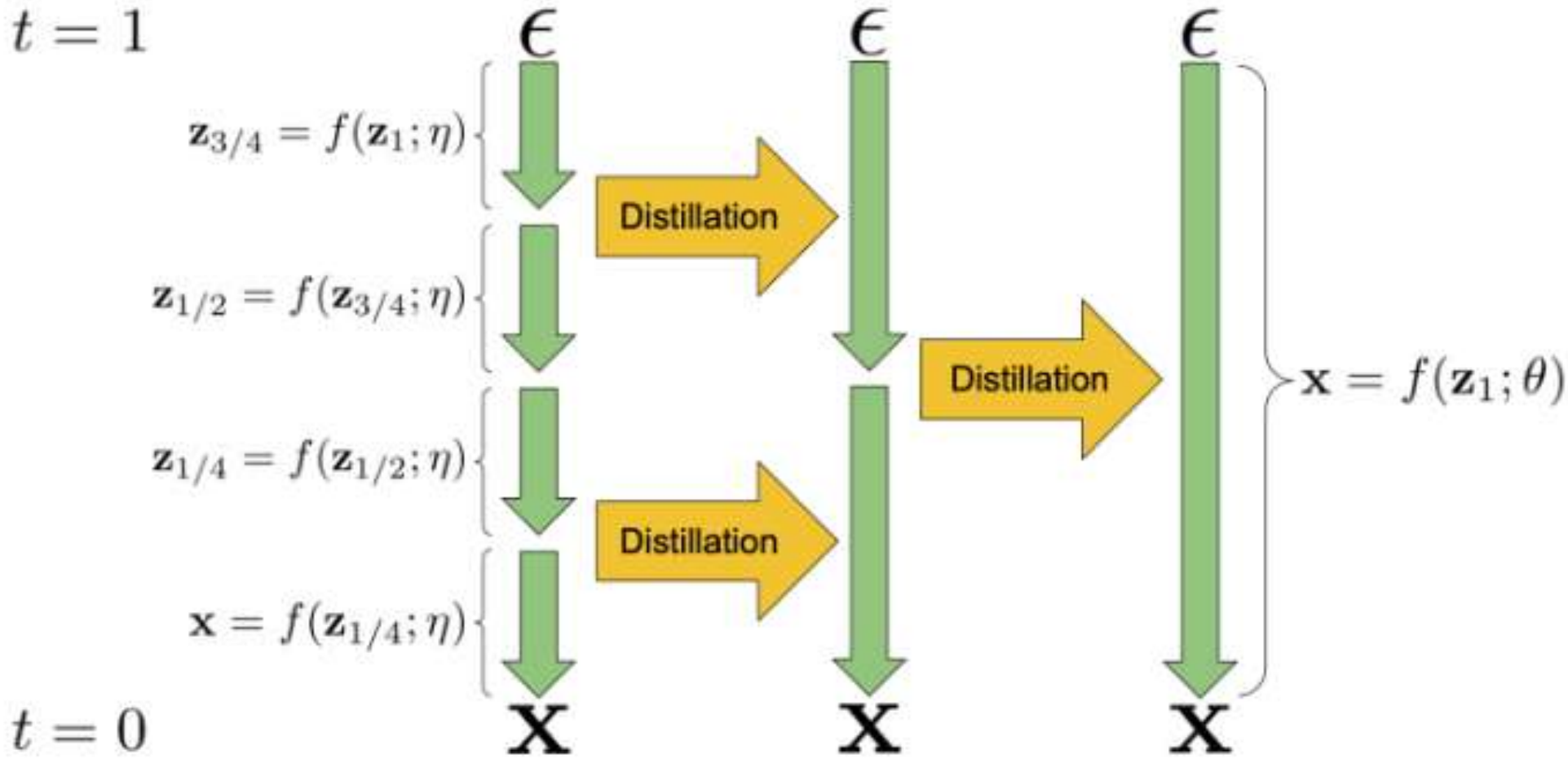
- **It is a (continuous-time) normalizing flow model!**

Accelerated sampling

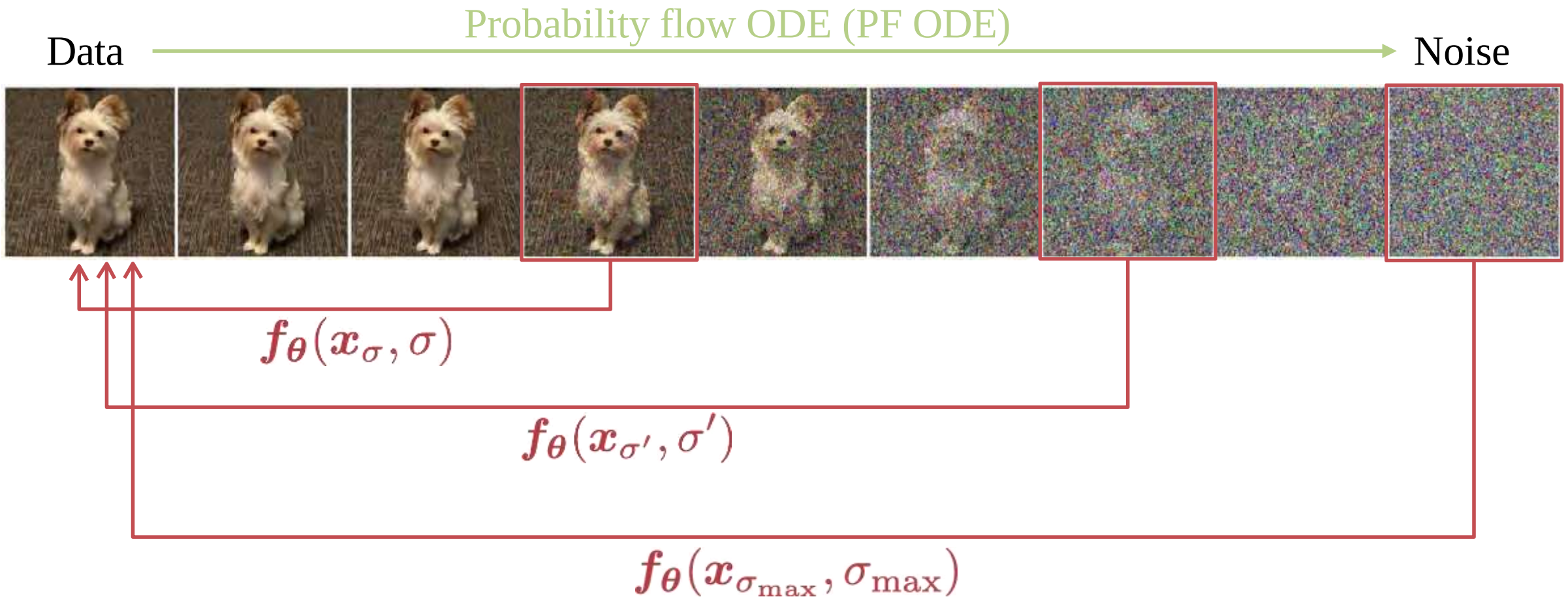


- Numerical methods + ODE formulation to accelerate sampling
- DDIM [Song and Ermon, 2021]:
  - Coarsely discretize the time axis, take big steps
  - Corresponds to exponential integrator (semi-linear ODE) [Lu et al, 2022; Zhang and Chen, 2022]
  - 10x-50x speedups, comparable sample quality

Distillation



Consistency models are designed for one-step generation

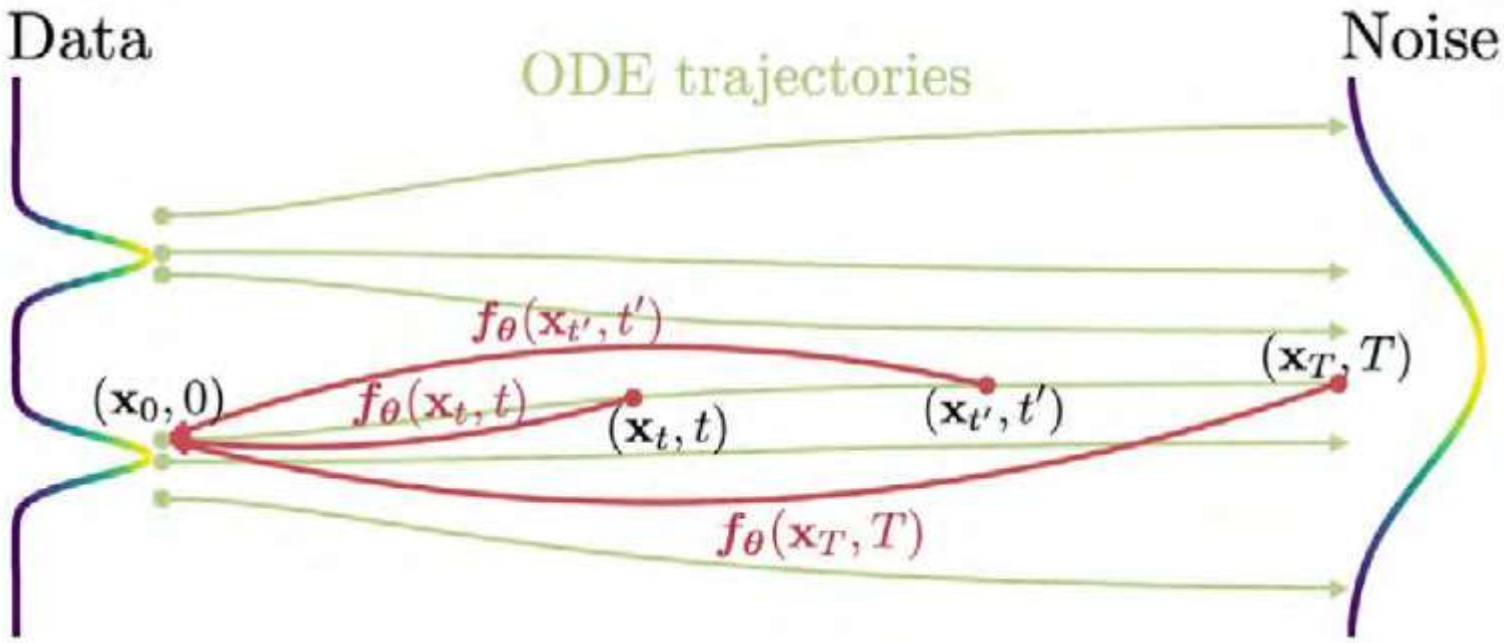


?

How does this differ from a denoiser?

Song, Dhariwal, Chen, Sutskever. Consistency Models. ICML 2023

Enforcing self-consistency via distillation



$$\min_{\theta} \|\boldsymbol{f}_{\theta}(\mathbf{x}_{t'}, t') - \boldsymbol{f}_{\theta}(\mathbf{x}_t, t)\|_2^2$$

$\mathbf{x}_0$ ← $\mathbf{x}_t$ ← $\mathbf{x}_{t'}$ ← $\mathbf{x}_T$

ODE solver + pretrained score function

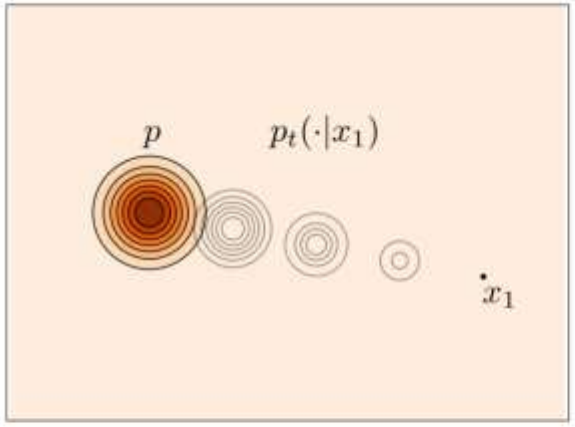
ODE solver + pretrained score function

ODE solver + pretrained score function

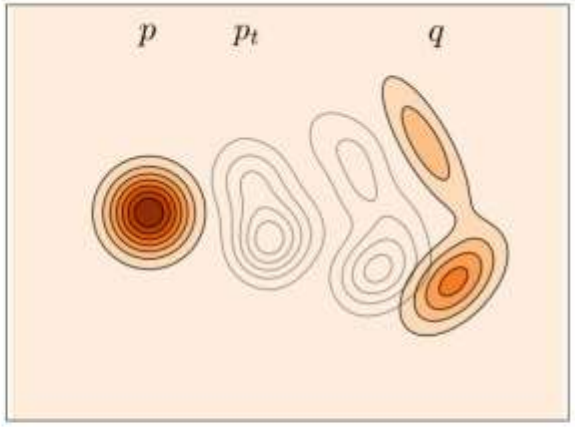
$$\min_{\theta} \|\boldsymbol{f}_{\theta}(\mathbf{x}_t, t) - \boldsymbol{f}_{\theta}(\mathbf{x}_0, 0)\|_2^2$$

$$\min_{\theta} \|\boldsymbol{f}_{\theta}(\mathbf{x}_T, T) - \boldsymbol{f}_{\theta}(\mathbf{x}_{t'}, t')\|_2^2$$

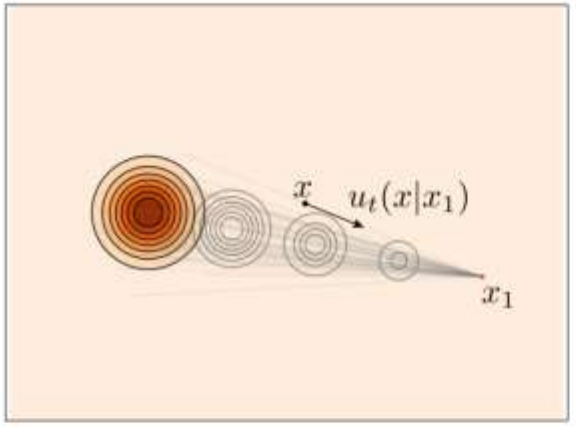
Flow Matching



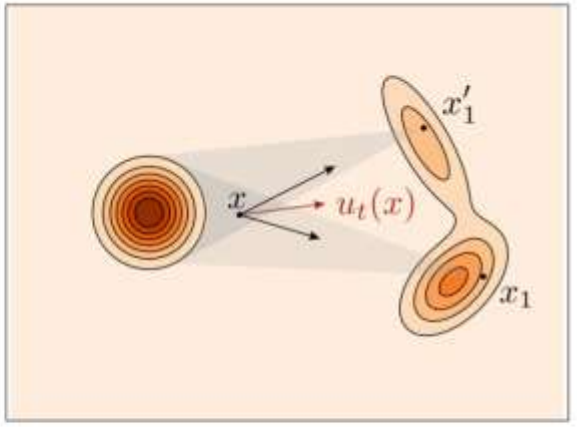
(a) Conditional probability path $p_t(x|x_1)$.



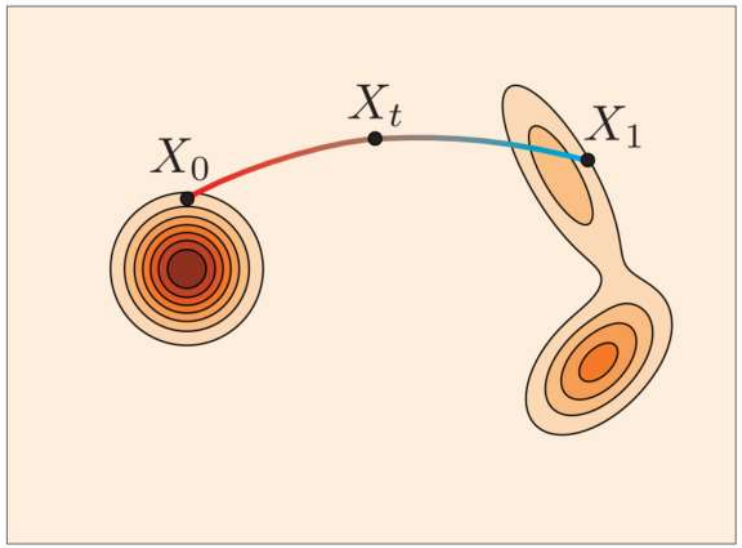
(b) (Marginal) Probability path $p_t(x)$.



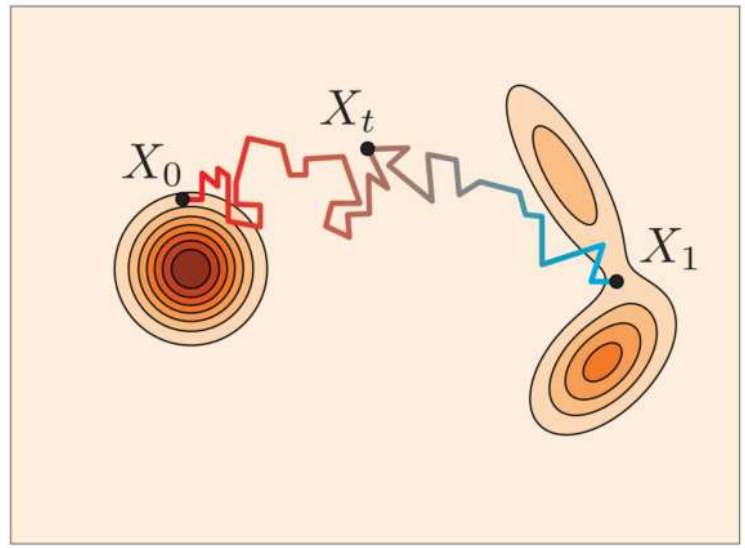
(c) Conditional velocity field $u_t(x|x_1)$.



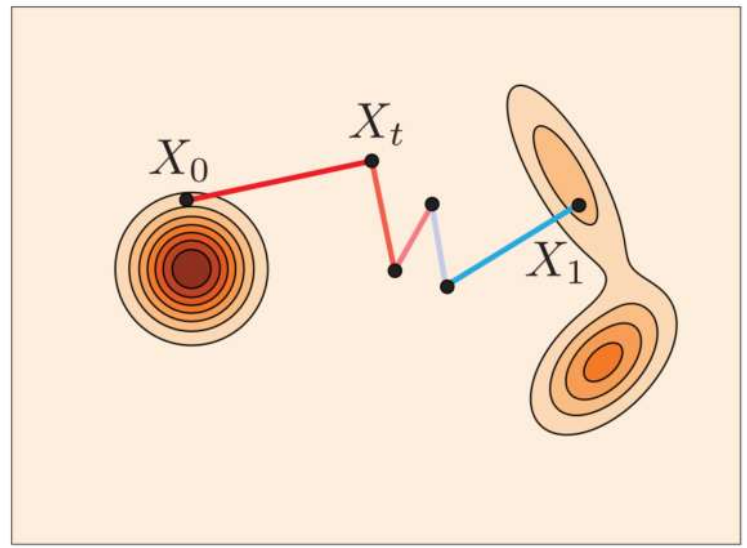
(d) (Marginal) Velocity field $u_t(x)$.



(a) Flow

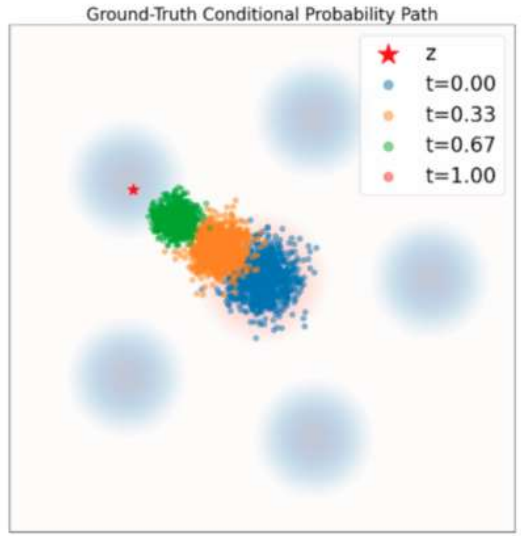


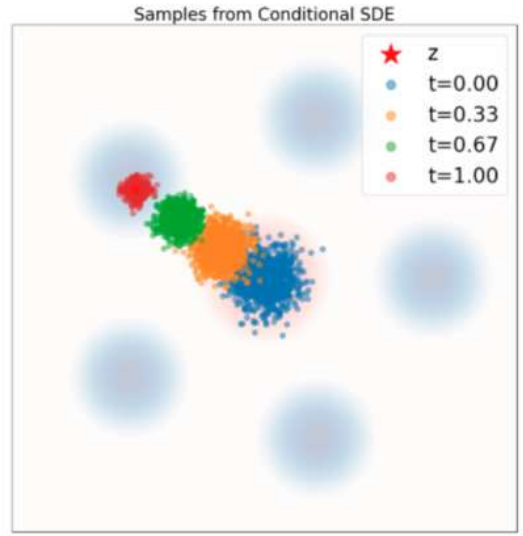
(b) Diffusion

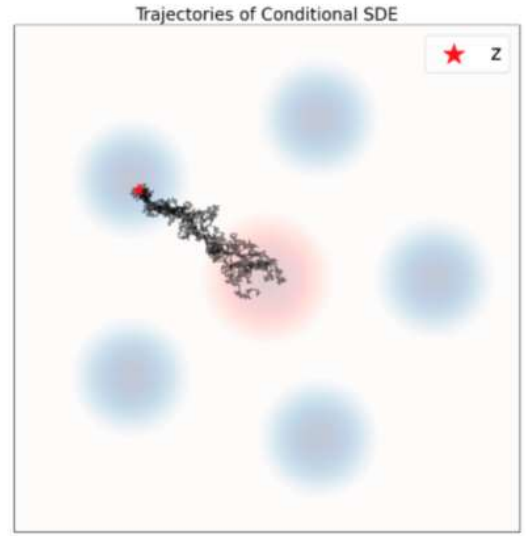


(c) Jump

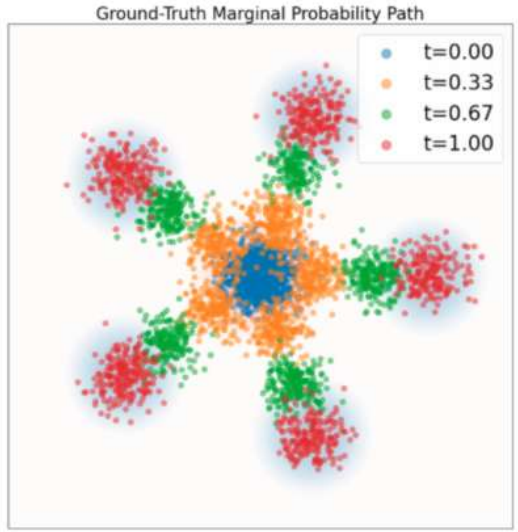
$p_t(\cdot|z)$

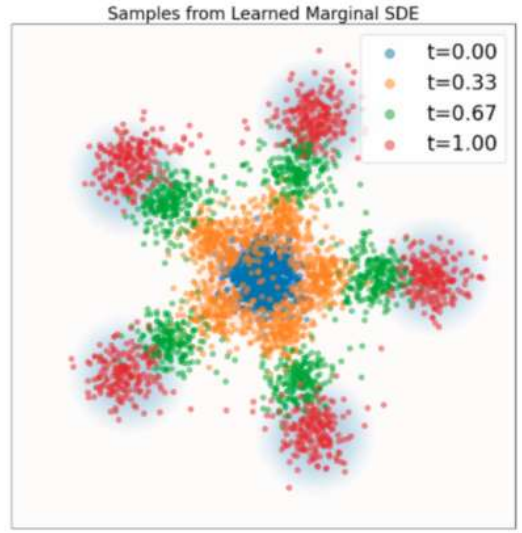

Ground-Truth Conditional Probability Path
z
t=0.00
t=0.33
t=0.67
t=1.00

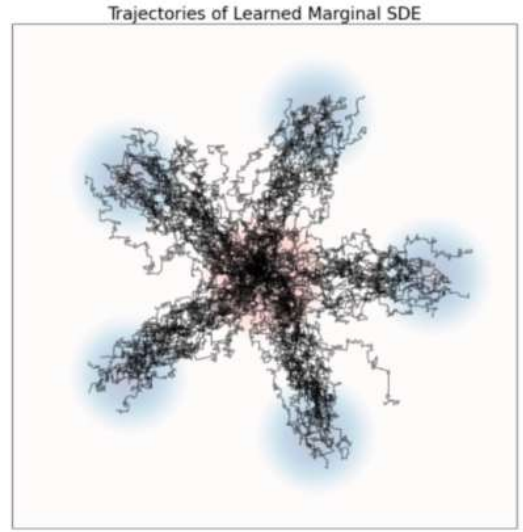

Samples from Conditional SDE
z
t=0.00
t=0.33
t=0.67
t=1.00


Trajectories of Conditional SDE
z

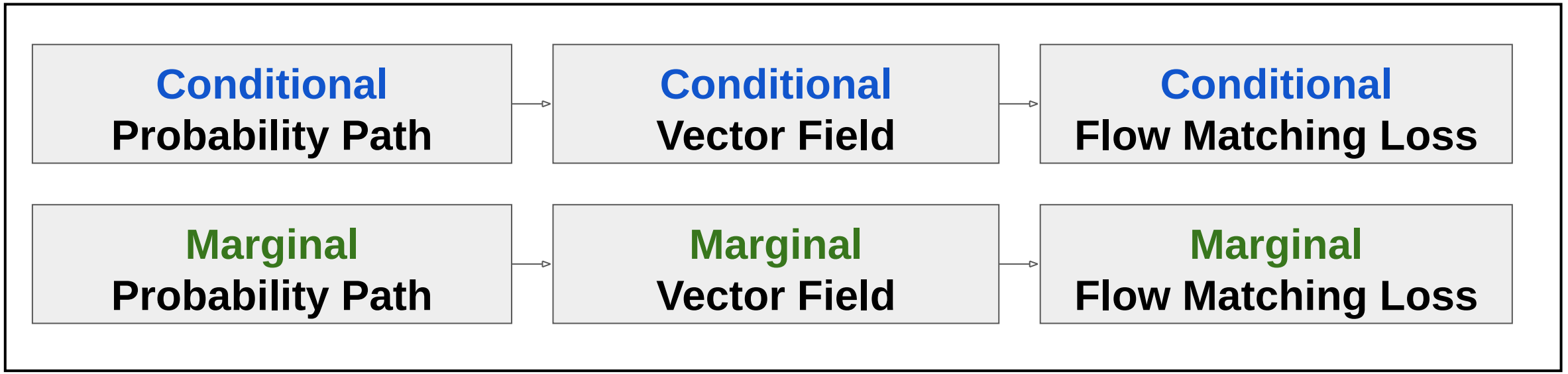
p_t


Ground-Truth Marginal Probability Path
t=0.00
t=0.33
t=0.67
t=1.00


Samples from Learned Marginal SDE
t=0.00
t=0.33
t=0.67
t=1.00


Trajectories of Learned Marginal SDE

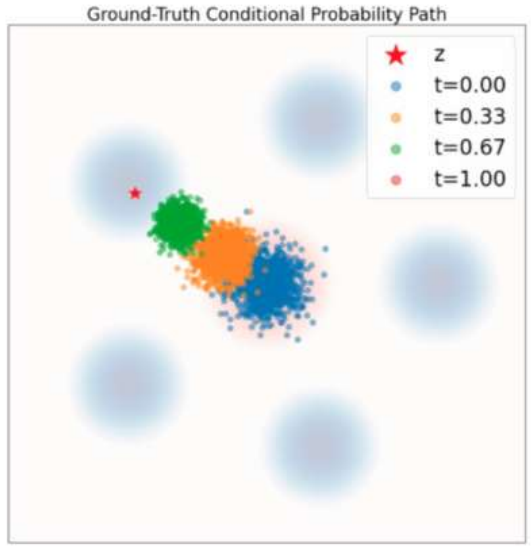
The Flow Matching Matrix

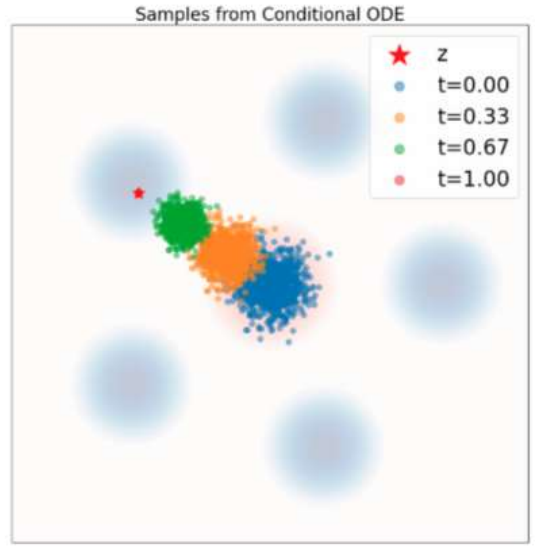


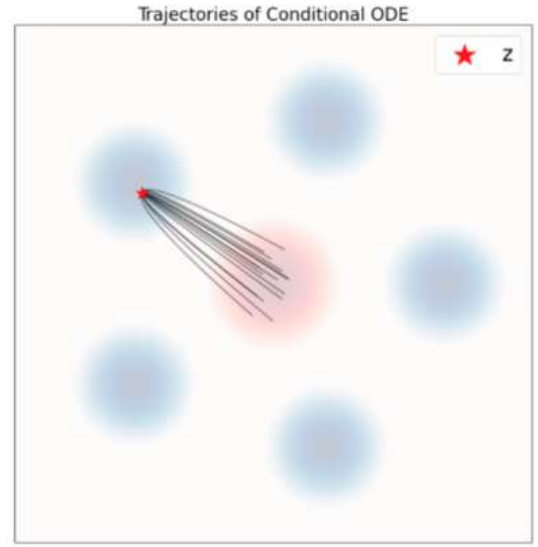
“Conditional” = “Per single data point”

“Marginal” = “Across distribution of data points”

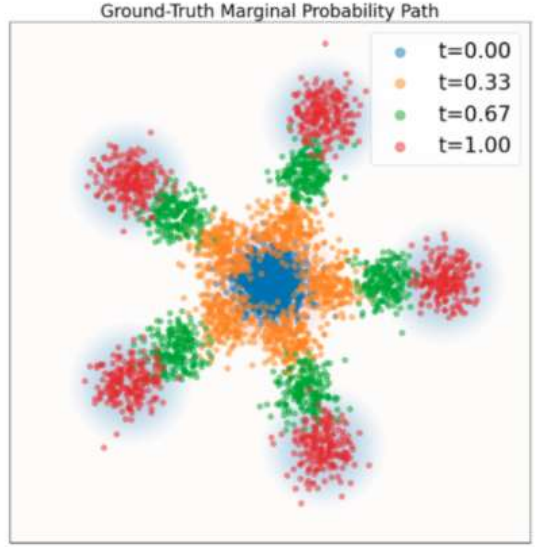
$p_t(\cdot|z)$

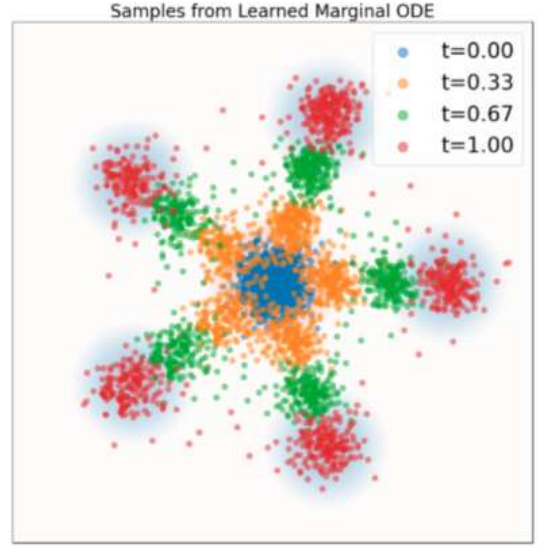

Ground-Truth Conditional Probability Path
z
t=0.00
t=0.33
t=0.67
t=1.00

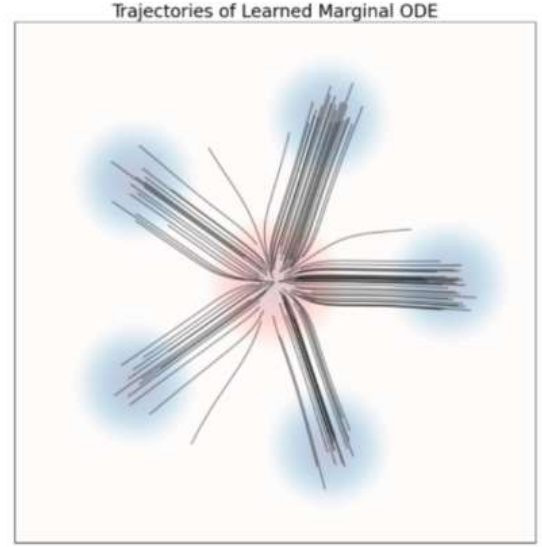

Samples from Conditional ODE
z
t=0.00
t=0.33
t=0.67
t=1.00


Trajectories of Conditional ODE
z

p_t


Ground-Truth Marginal Probability Path
t=0.00
t=0.33
t=0.67
t=1.00


Samples from Learned Marginal ODE
t=0.00
t=0.33
t=0.67
t=1.00


Trajectories of Learned Marginal ODE

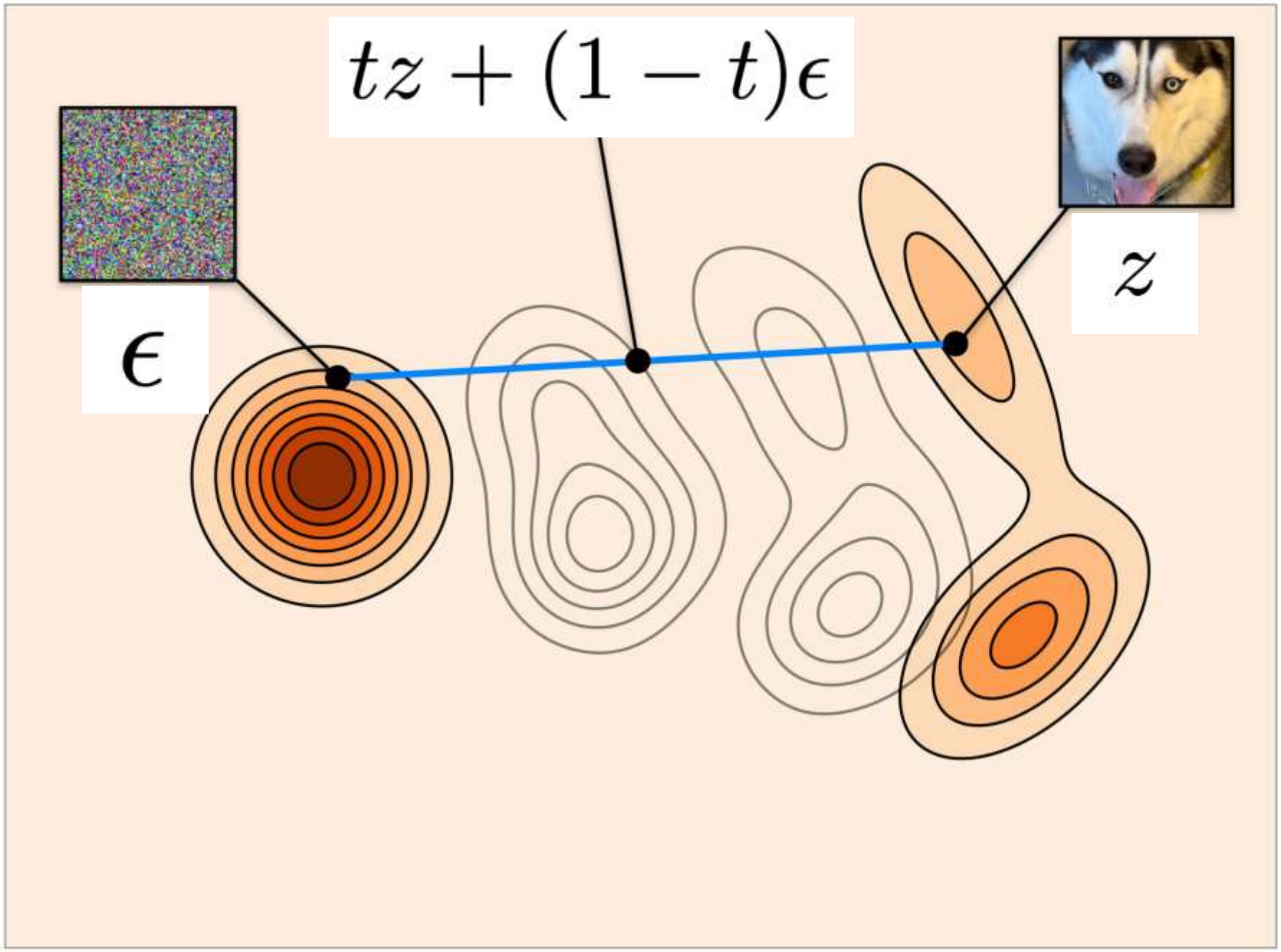
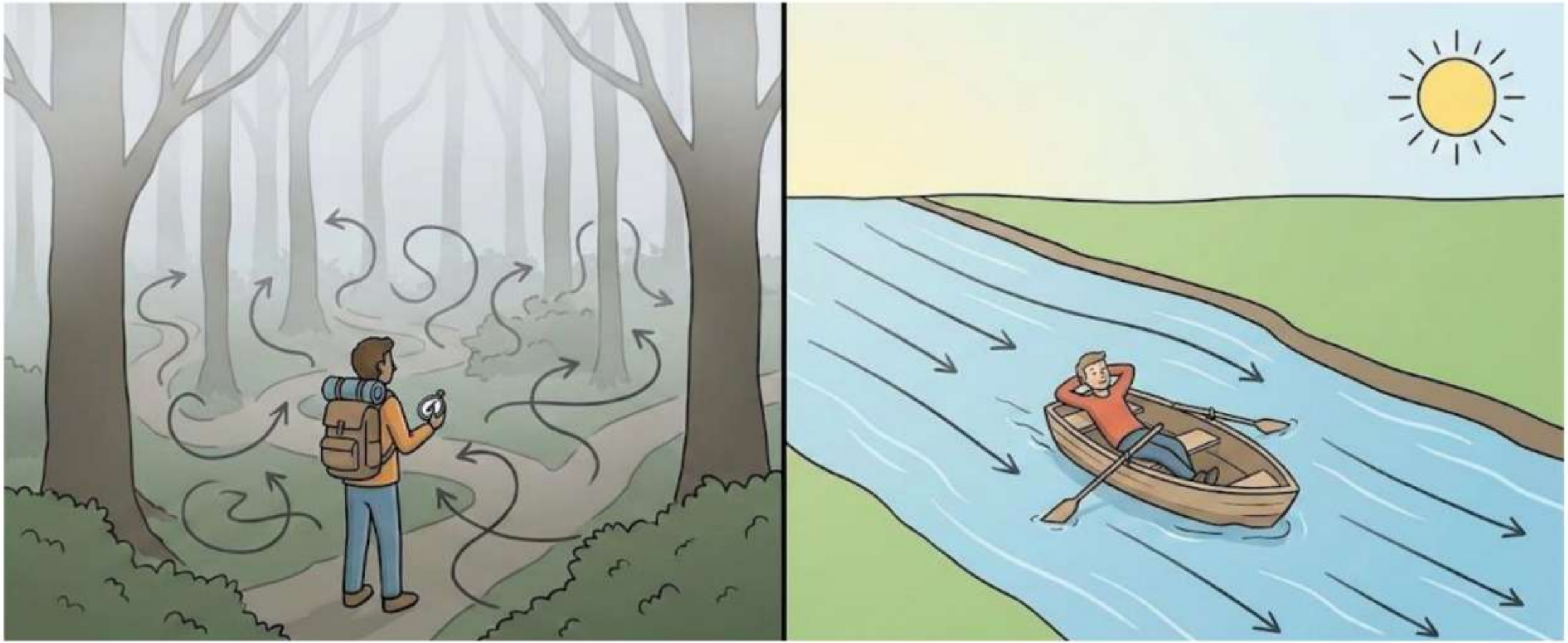


Figure credit: Yaron Lipman

Diffusion Models

Flow Matching



Conditional Generation of Diffusion Models

Conditional generation

User input:

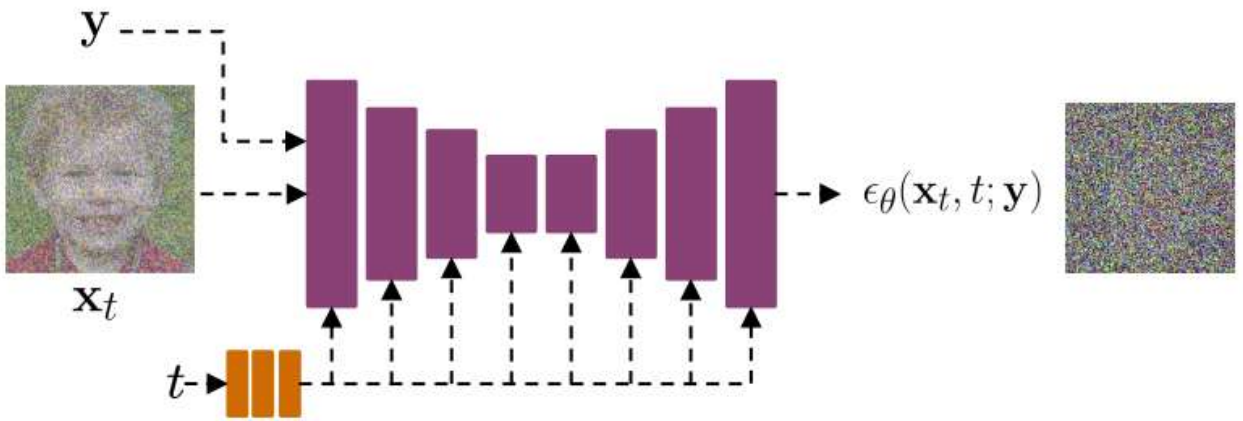
An astronaut riding a horse

Conditional generation

- Let (x,y) denote (image,caption) pairs
- Training a conditional generative model involves learning p(x | y)

- Train score model for the image x conditional on caption y

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_{\text{data}}(\mathbf{x},\mathbf{y})}\mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\mathbb{E}_{t\sim\mathcal{U}[0,T]}\ ||\epsilon_\theta(\mathbf{x}_t, t; \mathbf{y}) - \epsilon||_2^2$$



Control the generation process

$p(\mathbf{x})$

Forward model $p(\mathbf{y} \mid \mathbf{x})$

Control signal $\mathbf{y}$ = "dog"

$p(\mathbf{x} \mid \mathbf{y})$

Inverse distribution

Bayes' rule:

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x})\,p(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y})}$$

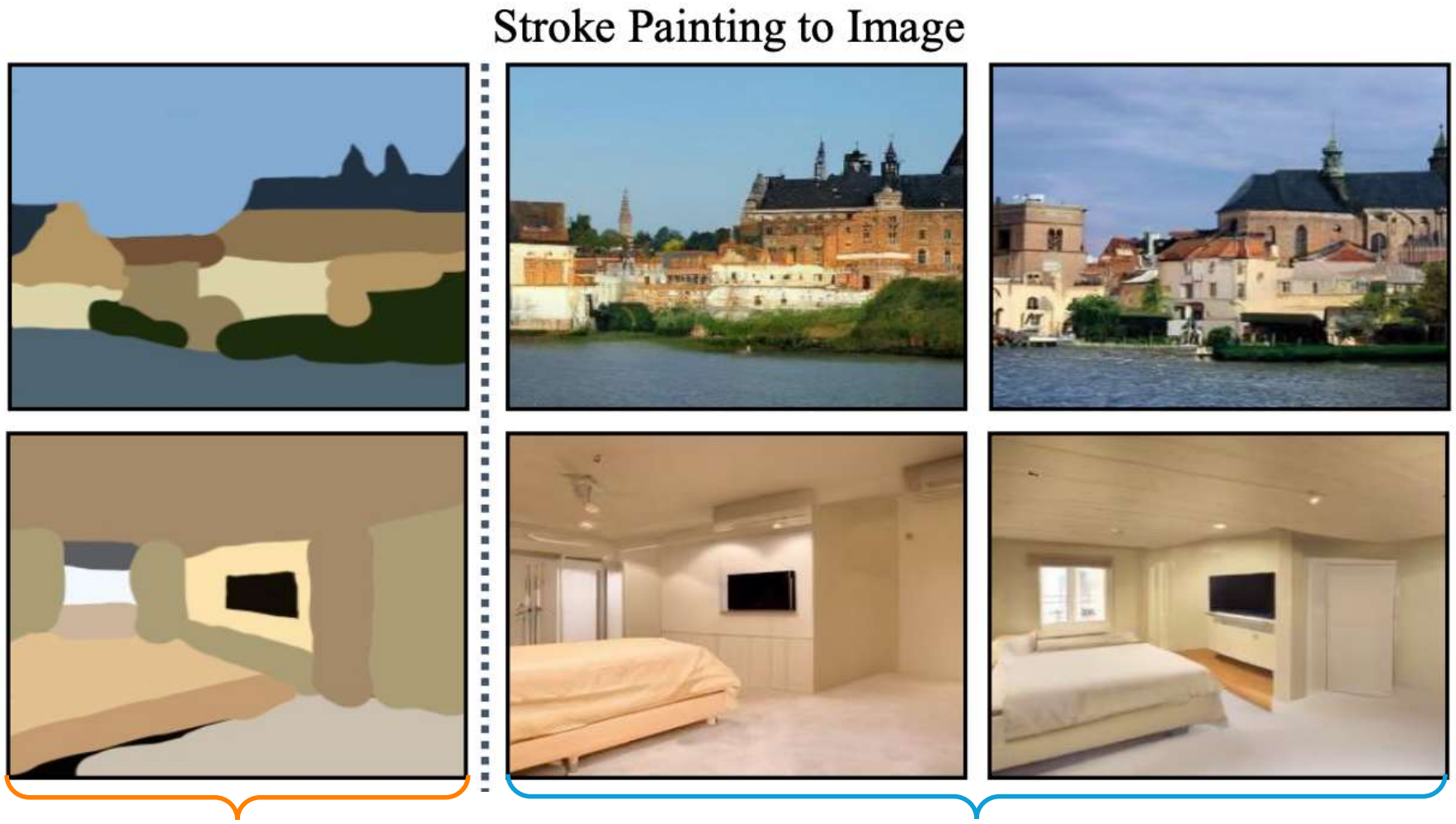
Bayes' rule for score functions:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{y})$$

$$= \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$$

Plug in different forward models for the same score model

Stroke to image synthesis



Forward model
$p(\mathbf{y} \mid \mathbf{x})$
can be specified.

Stroke paintings
$\mathbf{y}$

Sampled images
$\mathbf{x} \mid \mathbf{y}$

[Meng, He, Song, Song, Wu, Zhu, Ermon. ICLR 2022]

Language-guided image generation

$\mathbf{y}$ $\mathbf{x} \mid \mathbf{y}$

(Prompt)
Treehouse in the style of Studio Ghibli animation

Forward model

$p(\mathbf{y} \mid \mathbf{x})$

is an image captioning neural network.

[Work by @danielrussruss]

Classifier Guidance

Bayes' rule:

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x})\,p(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y})}$$

Bayes' rule for score functions:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{y})$$

$\nabla_{\mathbf{x}} \log p(\mathbf{y})$ → **0**

$$= \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$$

$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y})$: Conditional score

$\nabla_{\mathbf{x}} \log p(\mathbf{x})$: Unconditional score

$\nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$: Classifier obtained as the difference

Classifier-Guided Diffusion

To explicit incorporate class information into the diffusion process, trained a classifier $f_\phi(y|\mathbf{x}_t, t)$ on noisy image the use gradients to guide diffusion sampling process toward the condition

$$\begin{aligned}
\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t, y) &= \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(y|\mathbf{x}_t) \\
&\approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t) \\
&= -\frac{1}{\sqrt{1-\bar{\alpha}_t}} (\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t} \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t))
\end{aligned}$$

A new classifier-guided predictor $\bar{\epsilon}_\theta$ would take the form as following,

$$\bar{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t) = \boldsymbol{\epsilon}_\theta(x_t, t) - \sqrt{1-\bar{\alpha}_t} \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)$$

To control the strength of the classifier guidance, we can add a weight w to the delta part,

$$\bar{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t) = \boldsymbol{\epsilon}_\theta(x_t, t) - \sqrt{1-\bar{\alpha}_t}\, w \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)$$

Classifier-Guided Diffusion

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $f_\phi(y|x_t)$, 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 $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log f_\phi(y|x_t), \Sigma)$
end for
return x_0

Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $f_\phi(y|x_t)$, 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**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log f_\phi(y|x_t)$
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$
end for
return x_0

https://arxiv.org/pdf/2105.05233

Classifier-Guided Diffusion

https://arxiv.org/pdf/2105.05233

Classifier-Free Guidance

- Train both a conditional and an unconditional score model (by randomly dropping the caption during training)

- Combine the two models as follows

$$(1+w)\nabla_{\mathbf{x}_t}\log p(\mathbf{y}|\mathbf{x}_t)+\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t)=(1+w)\big(\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{y})-\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t)\big)+\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t)$$

$$=(1+w)\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{y})-w\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t)$$

- w is the classifier-guidance strength

Effect of classifier guidance

Increased classifier guidance strength (w)

Text-to-Image

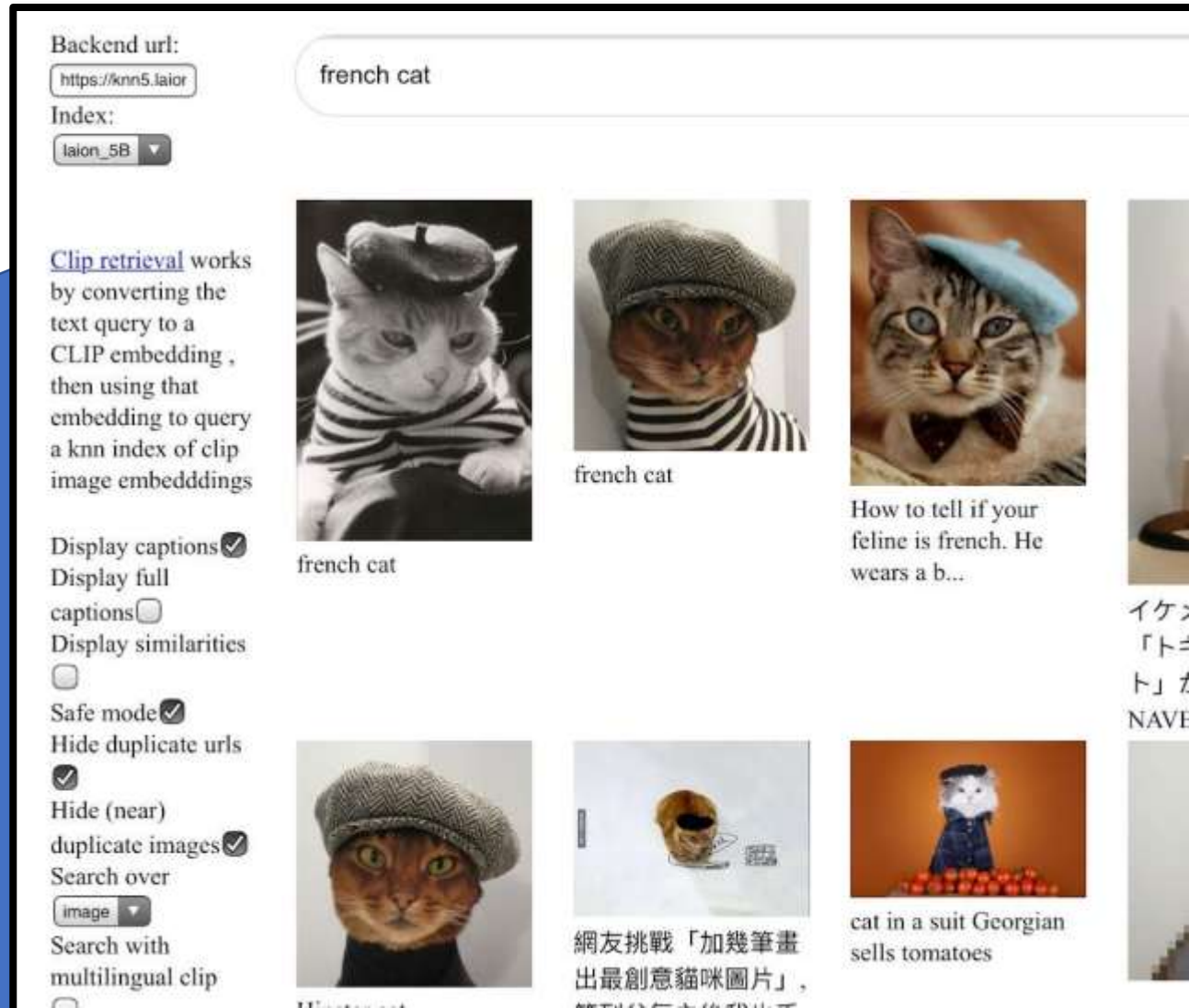
https://laion.ai/blog/laion-5b/

A cat in the snow → Text-to-image Generator →

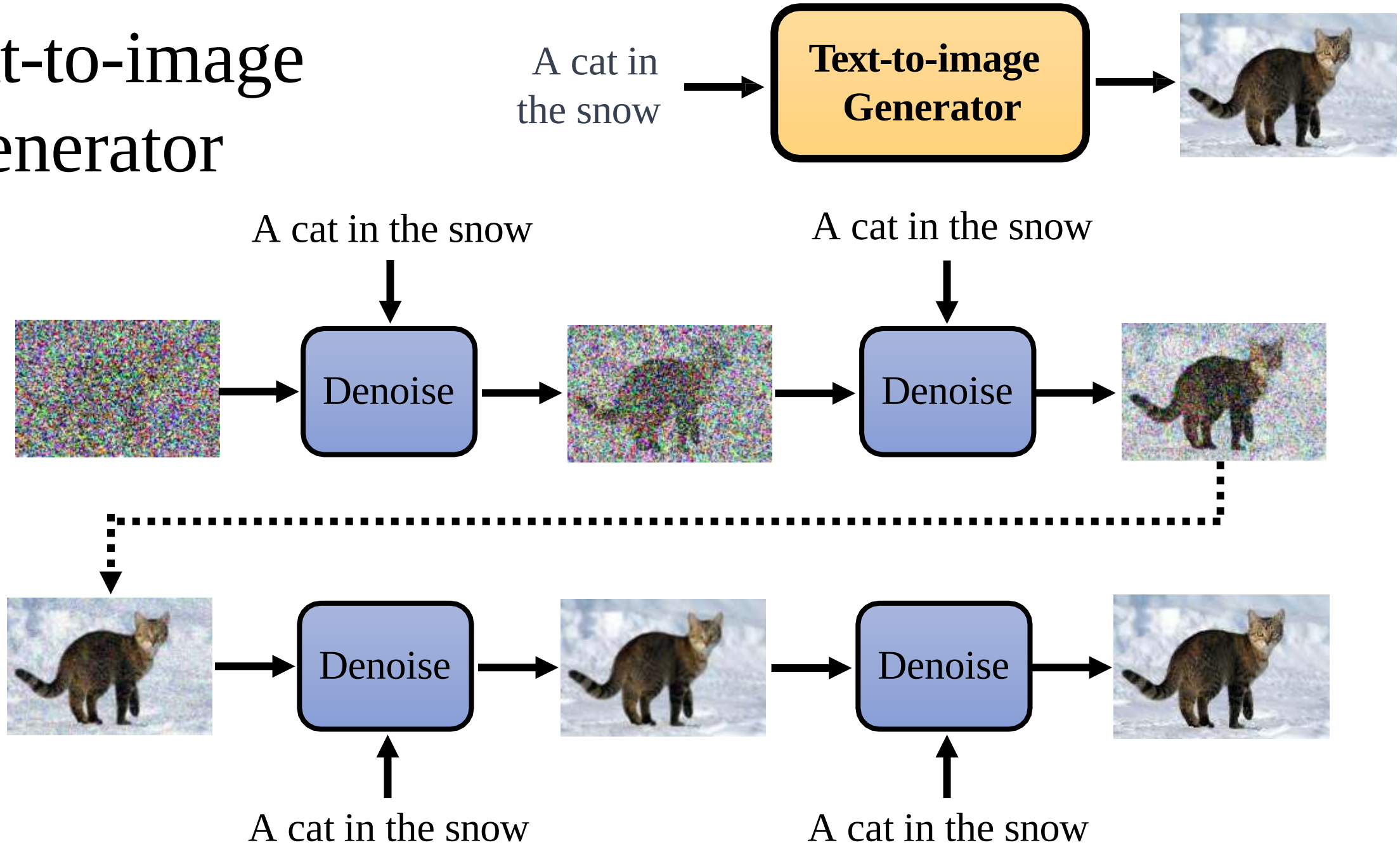
HW6 70k

ImageNet 1M

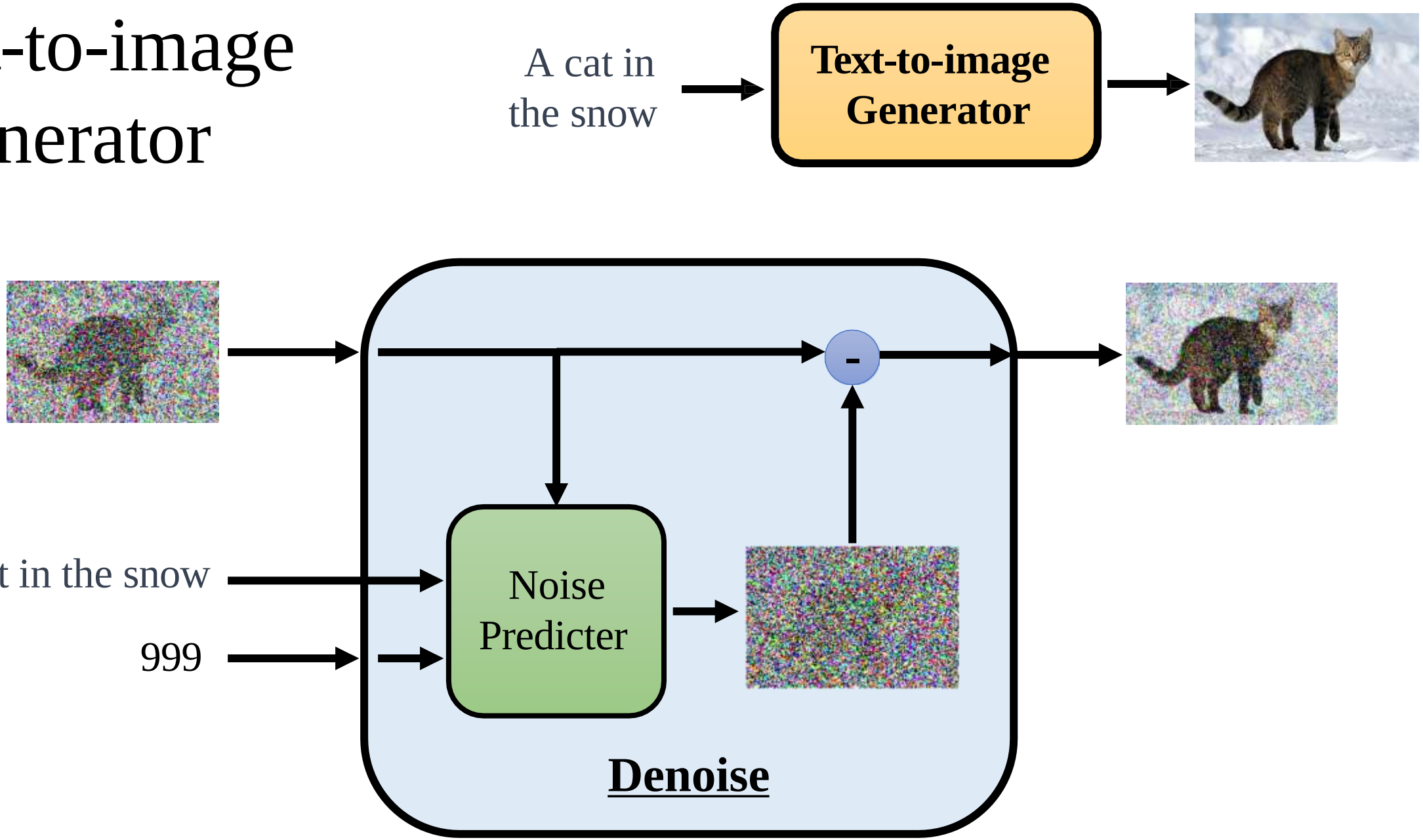
LAION 5.85B

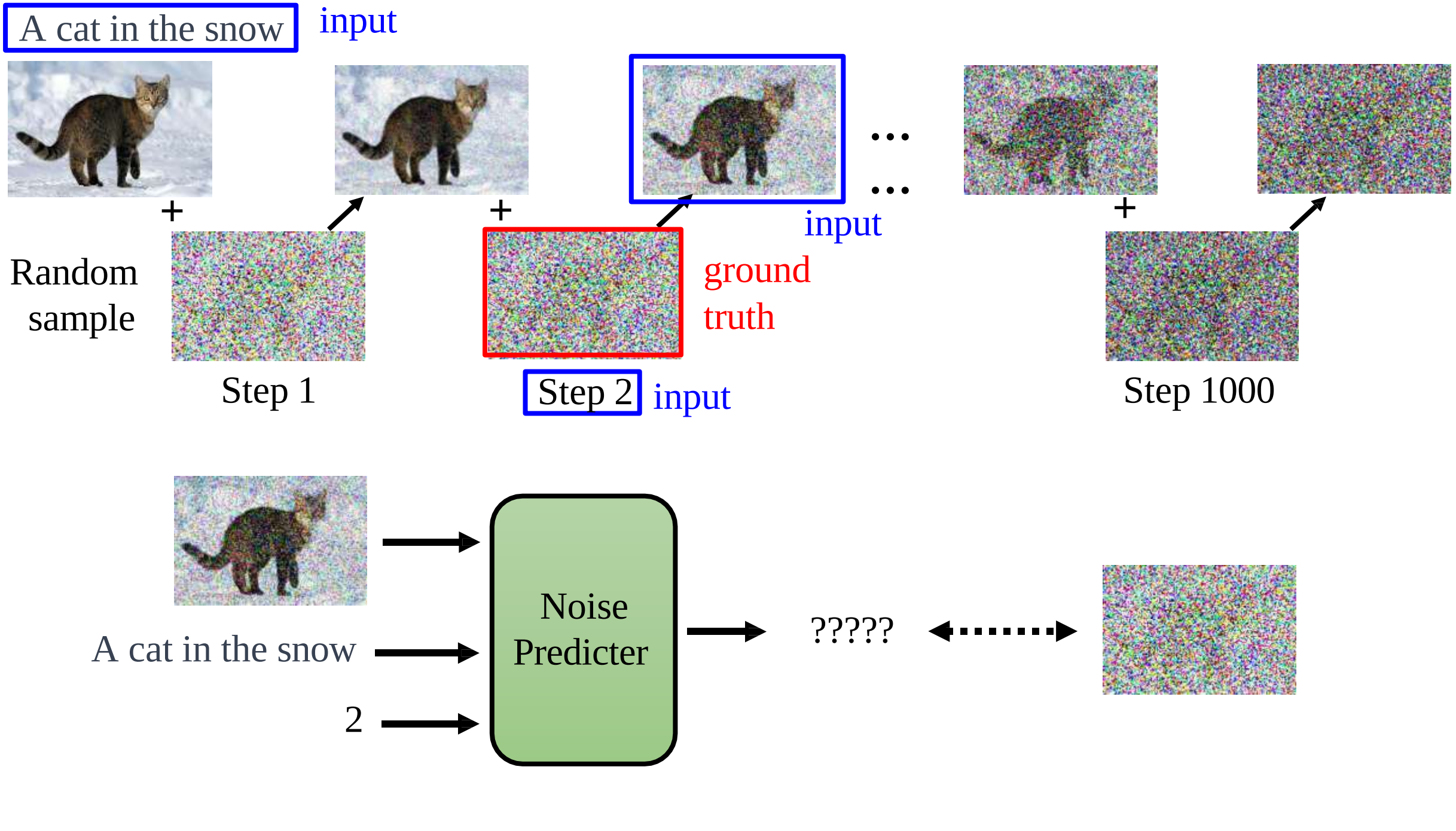


Text-to-image Generator



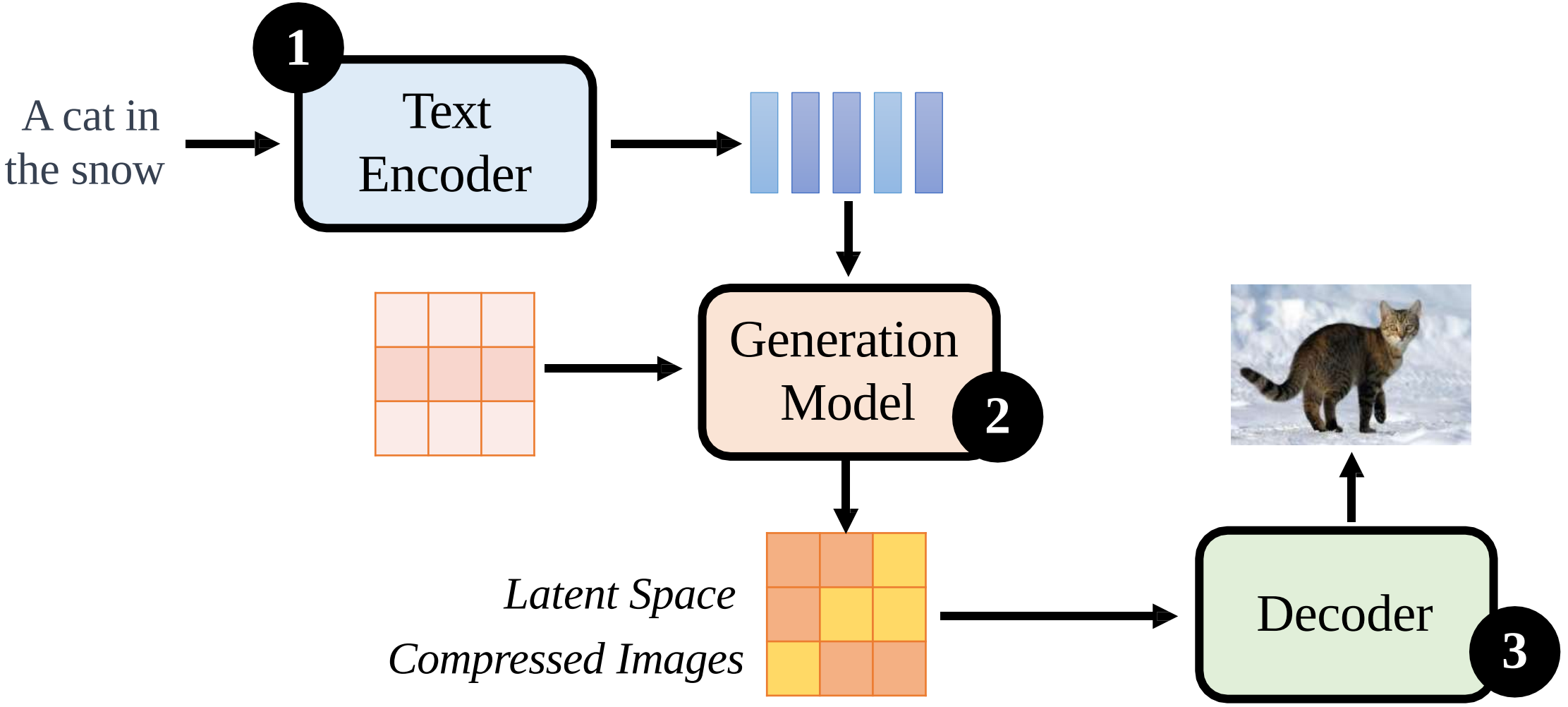
Text-to-image Generator




A cat in the snow
input
Random sample
Step 1
Step 2
input
ground truth
input
Step 1000
A cat in the snow
2
Noise Predicter
?????

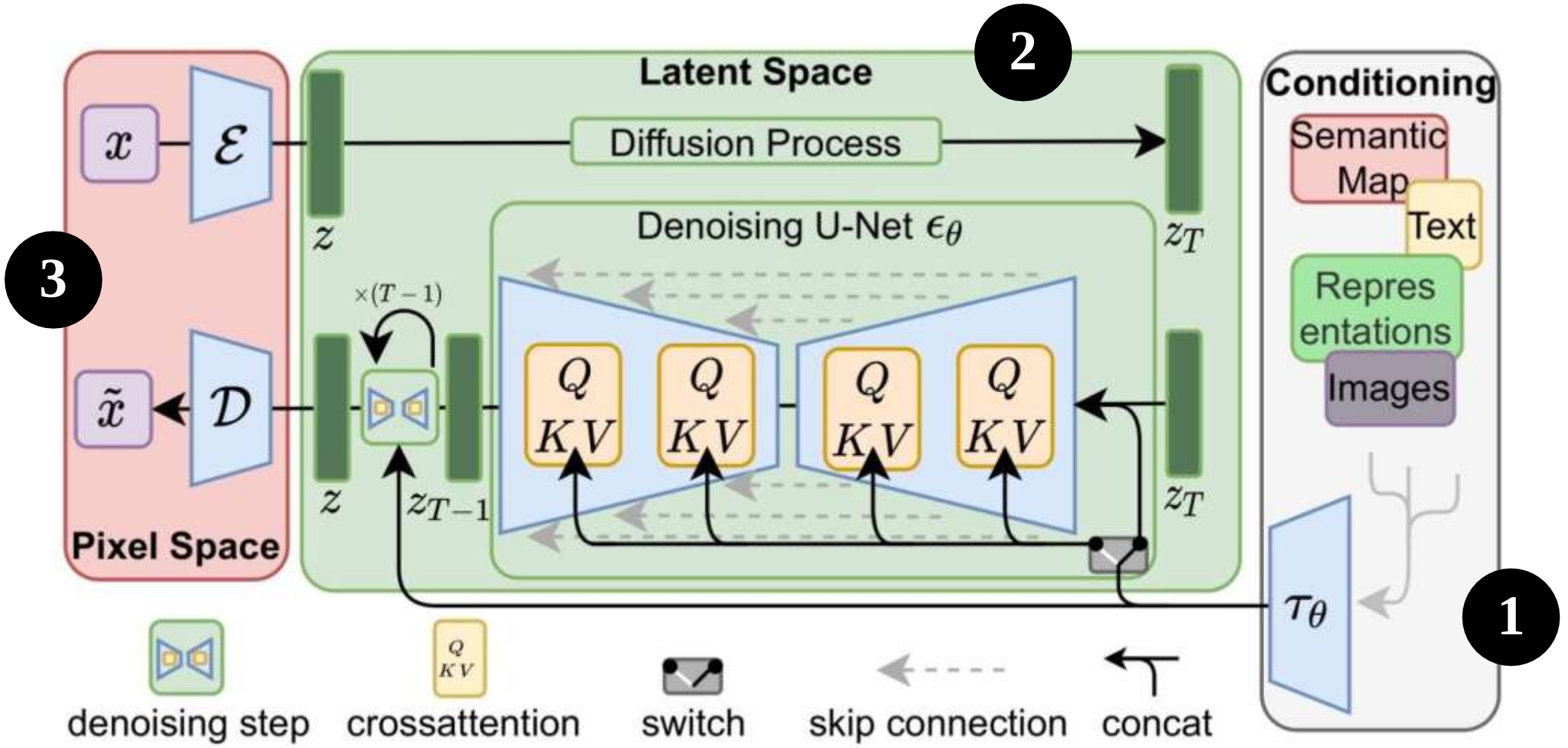
Framework

A cat in the snow → Text-to-image Generator →



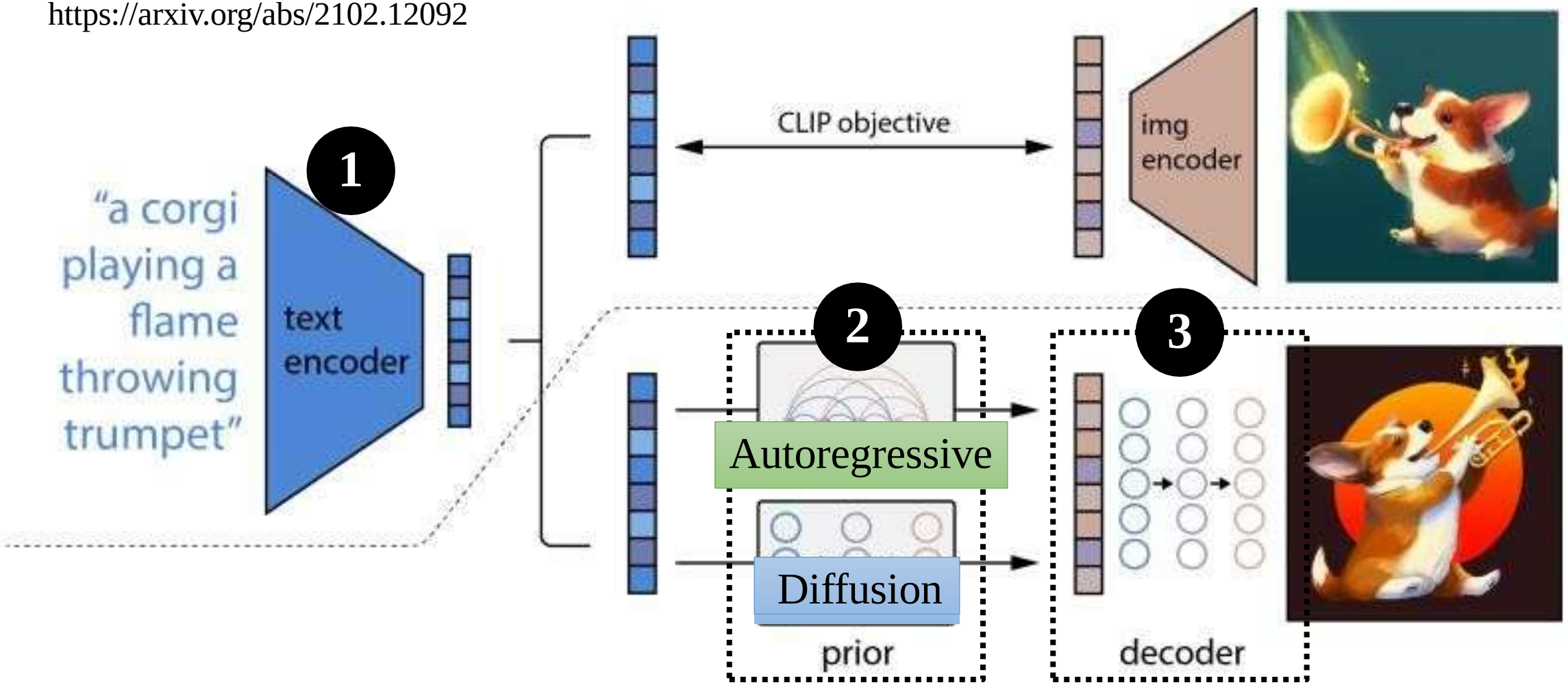
Stable Diffusion

https://arxiv.org/abs/2112.10752



DALL-E series

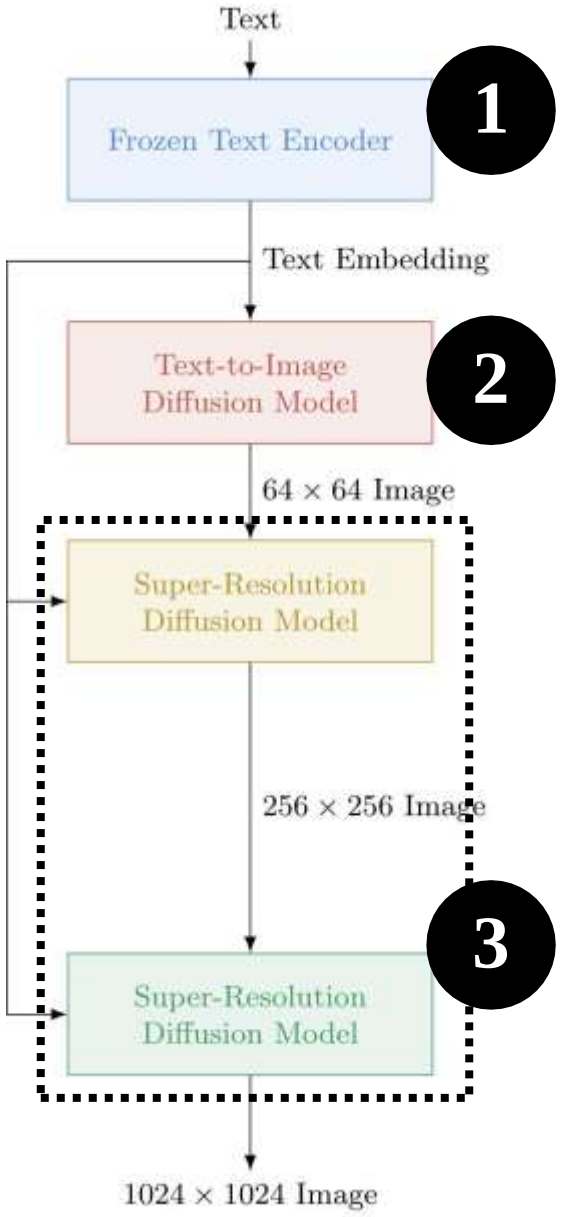
https://arxiv.org/abs/2204.06125
https://arxiv.org/abs/2102.12092

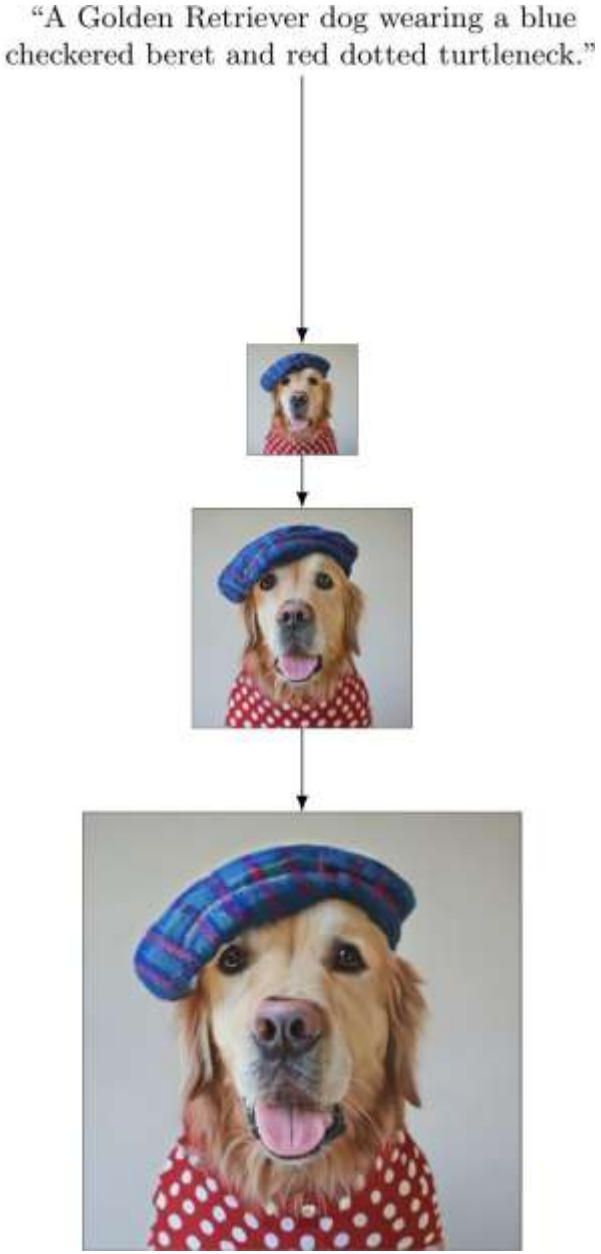


Imagen

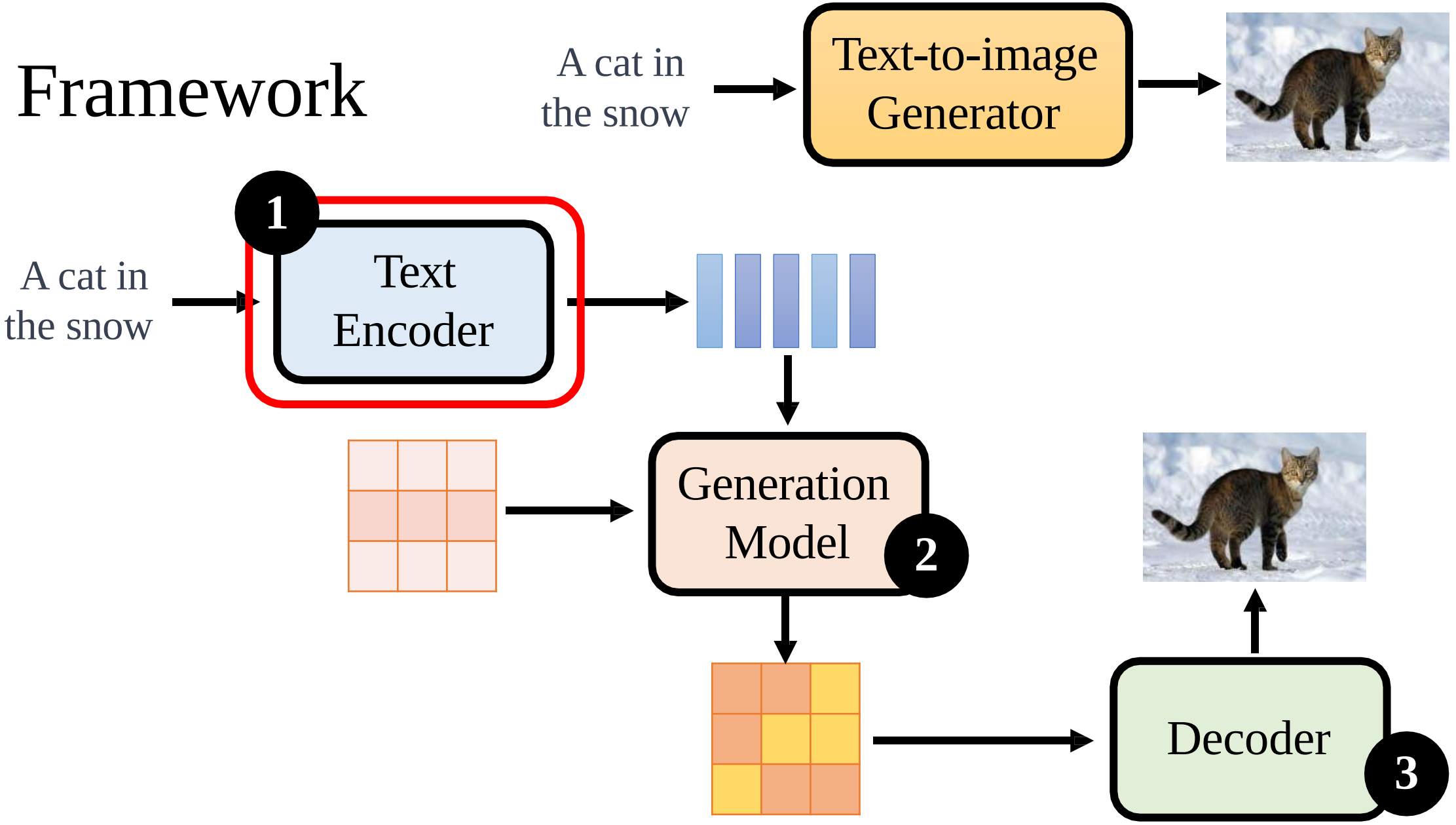
https://imagen.research.google/

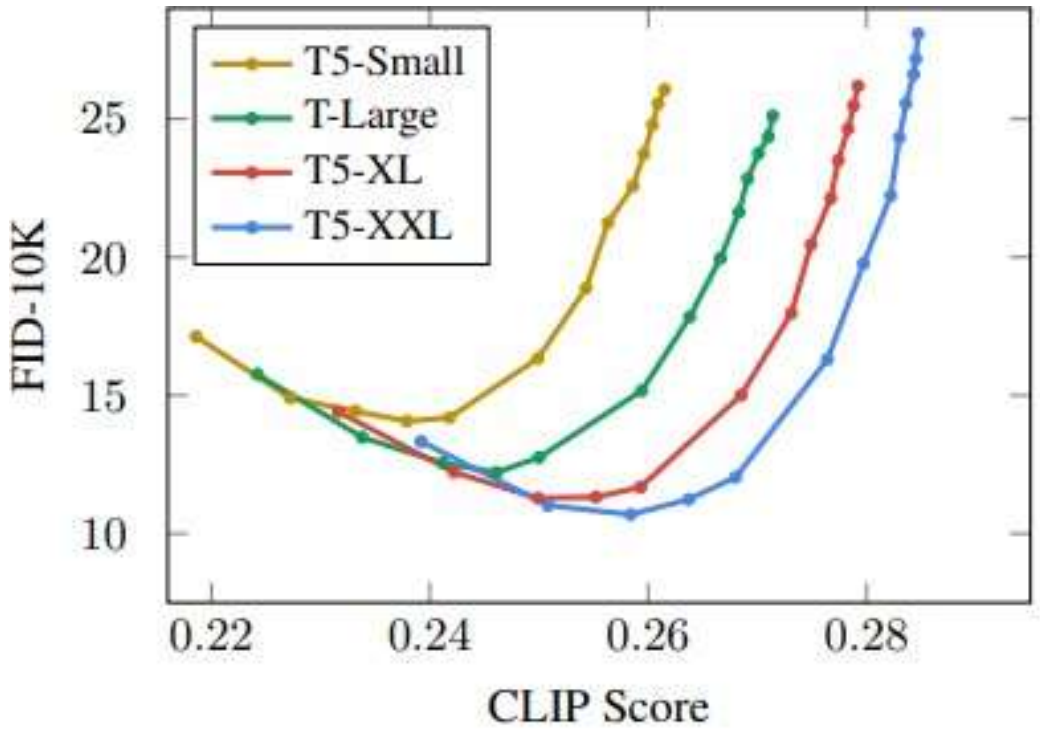
https://arxiv.org/abs/2205.11487



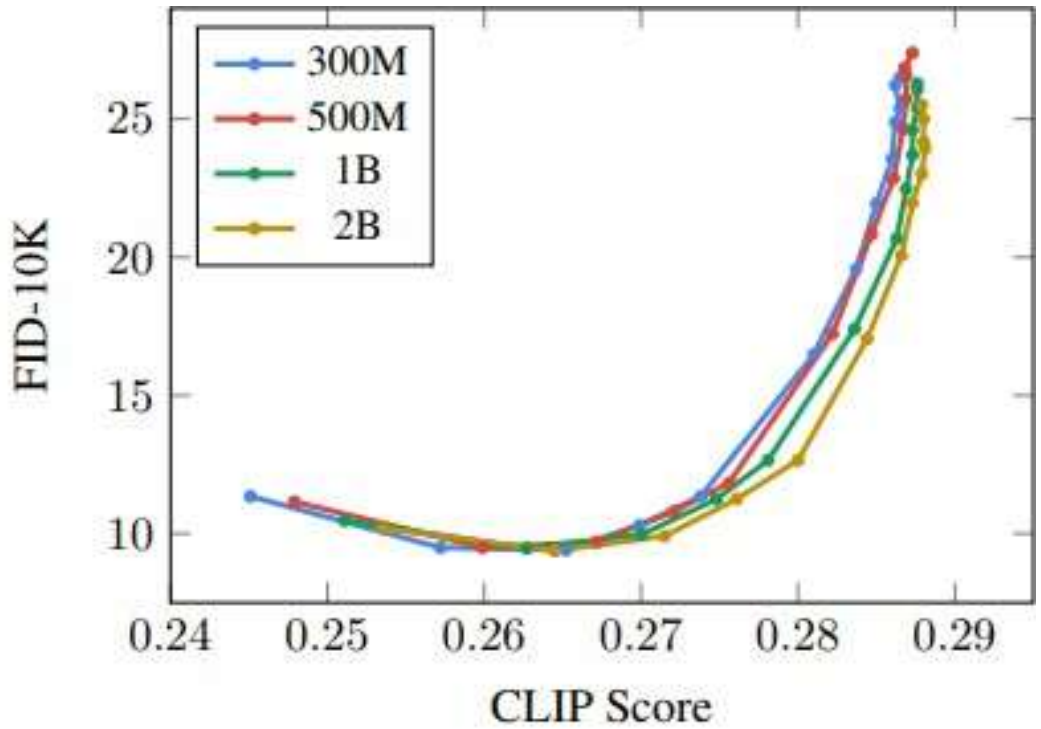


Framework


A cat in the snow
Text-to-image Generator
A cat in the snow
1
Text Encoder
Generation Model
2
Decoder
3



(a) Impact of encoder size.



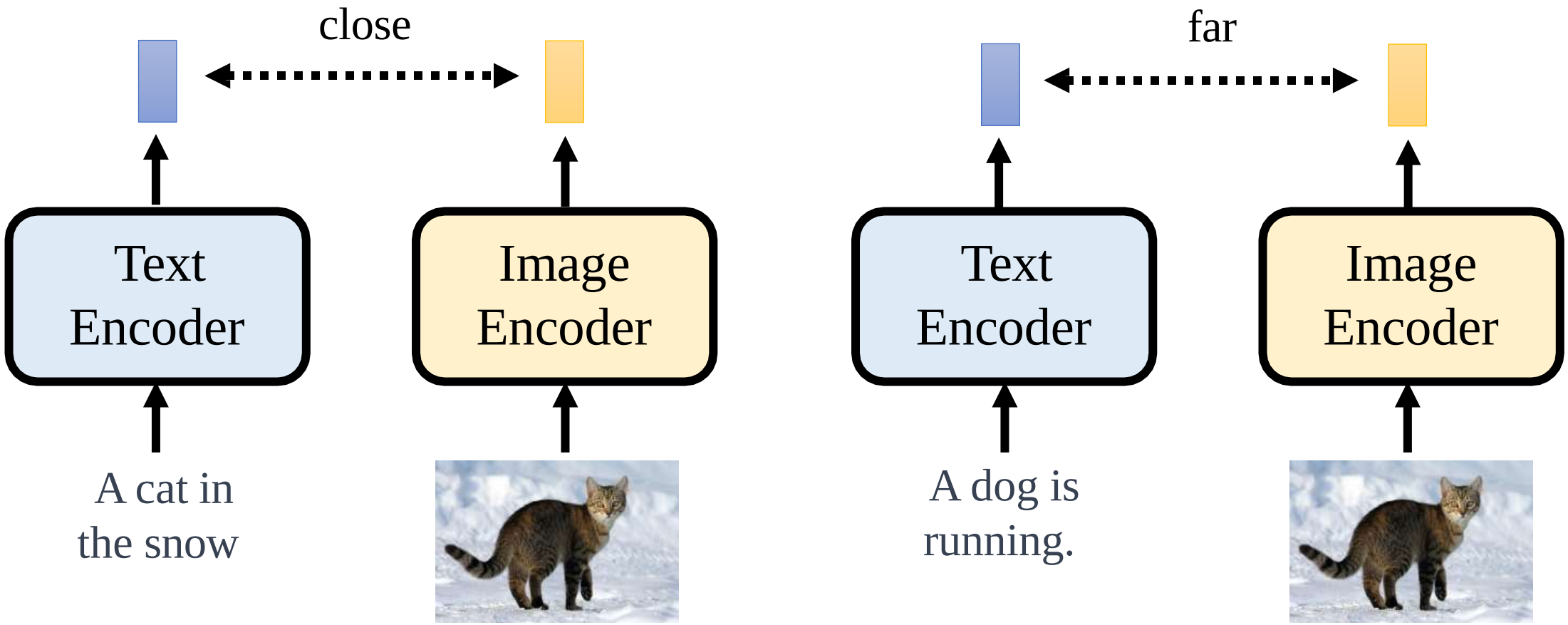
(b) Impact of U-Net size.

https://arxiv.org/abs/2205.11487

Contrastive Language-Image Pre-Training (CLIP)

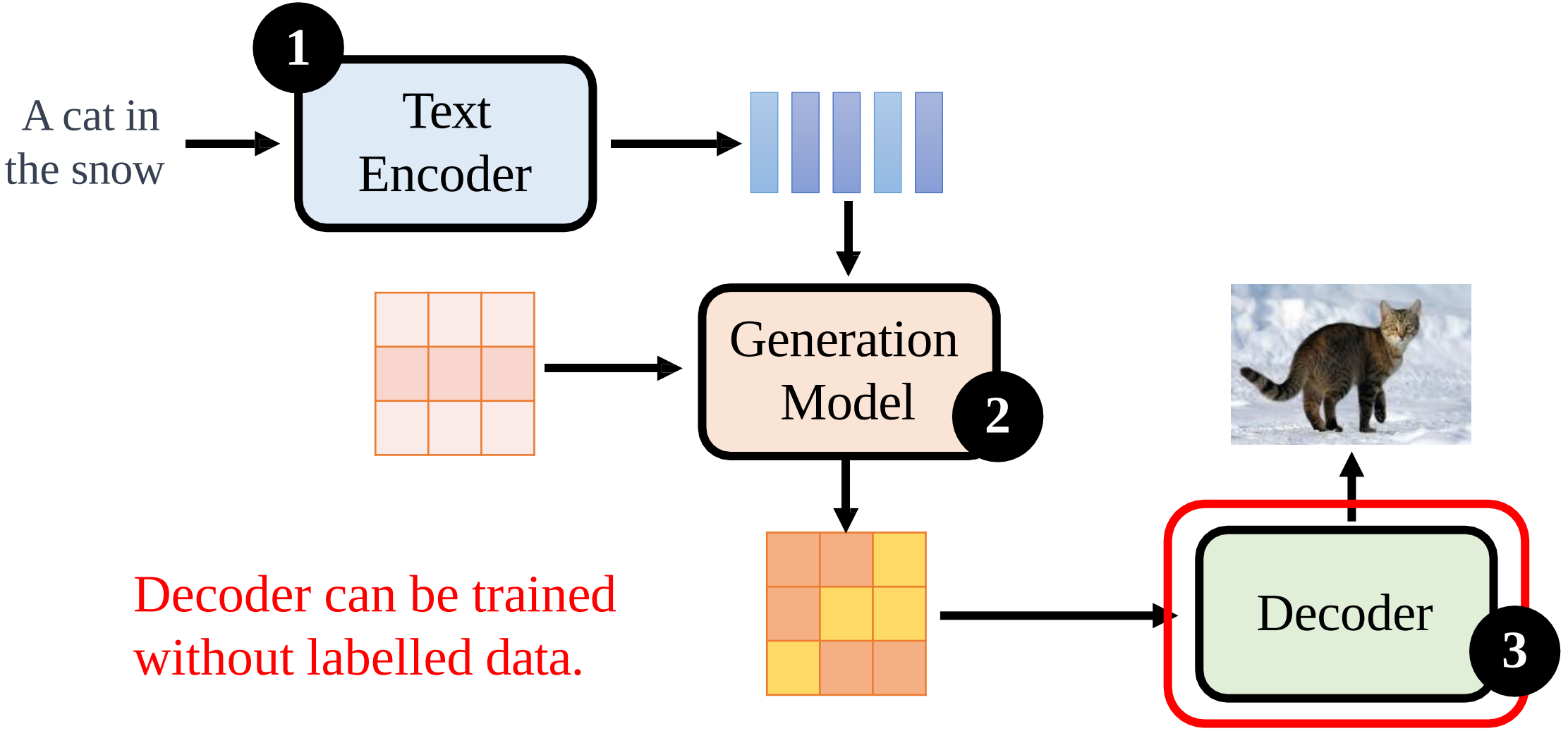
https://arxiv.org/abs/2103.00020

400 million image-text pairs

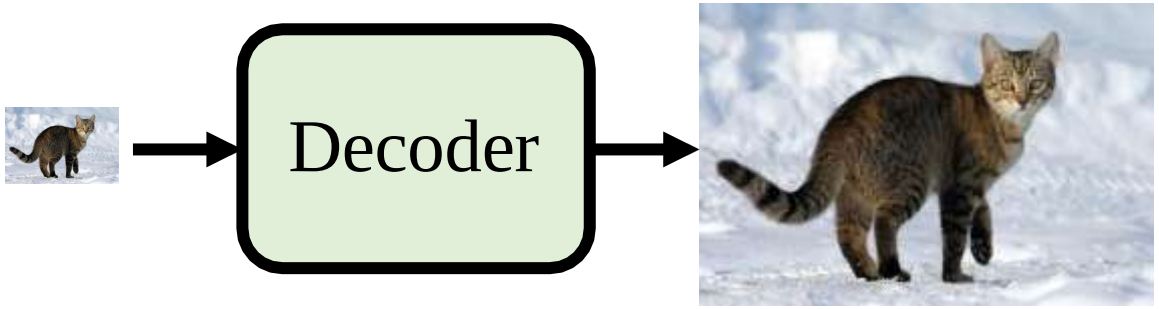


Framework

A cat in the snow → Text-to-image Generator →



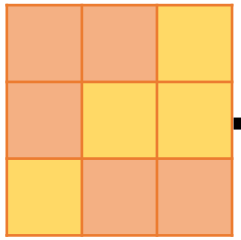
Progressive/Multi-Scale



(Images are generated by Midjourney)

Latent Diffusion

Latent Representation



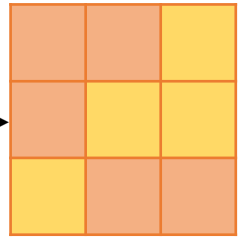
Decoder

Auto-encoder

$H \times W \times 3$

Encoder

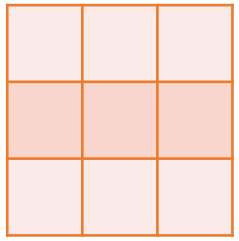
$h \times w \times c$

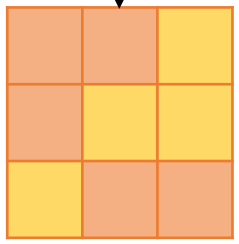


Decoder

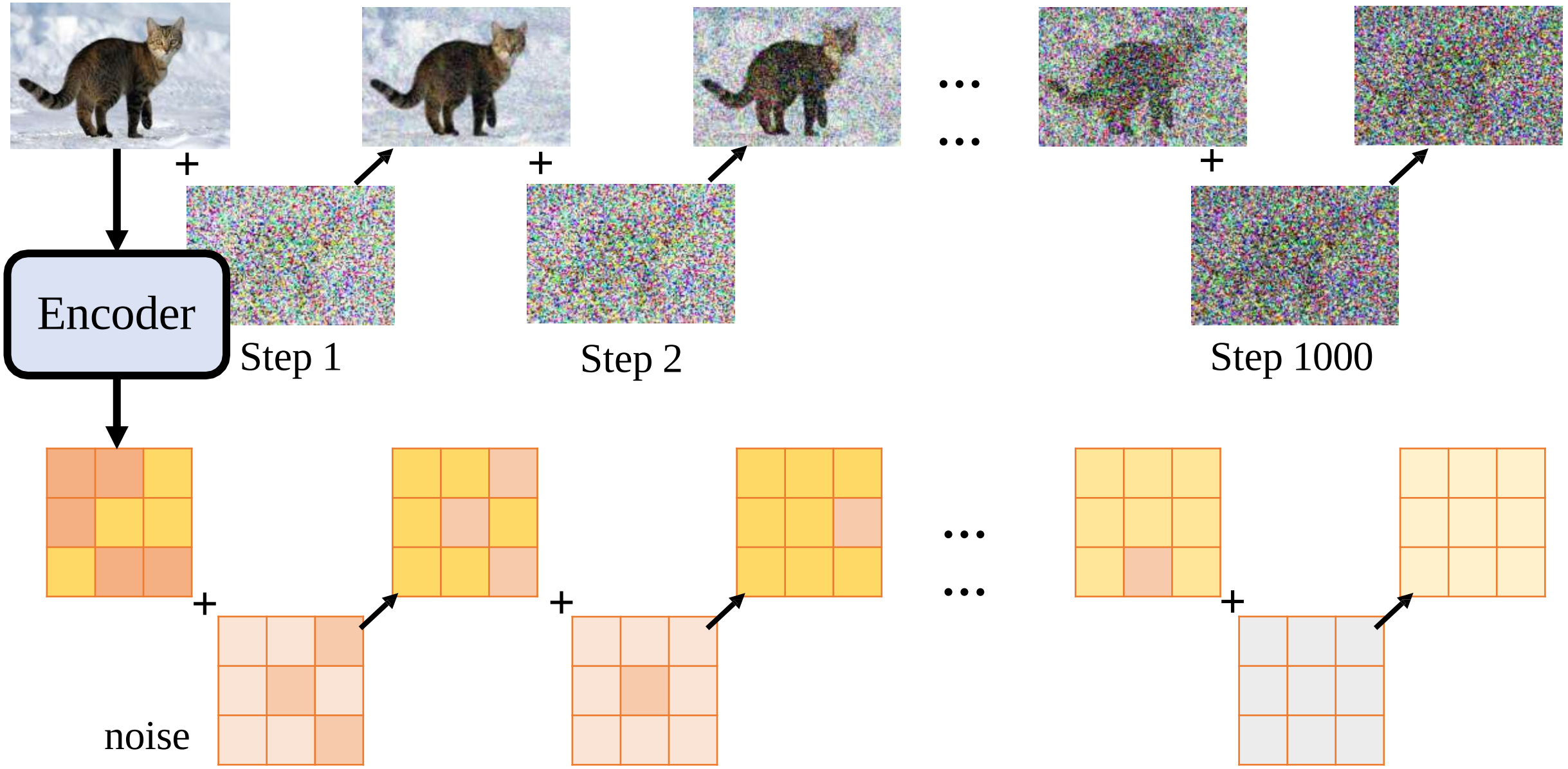
Framework

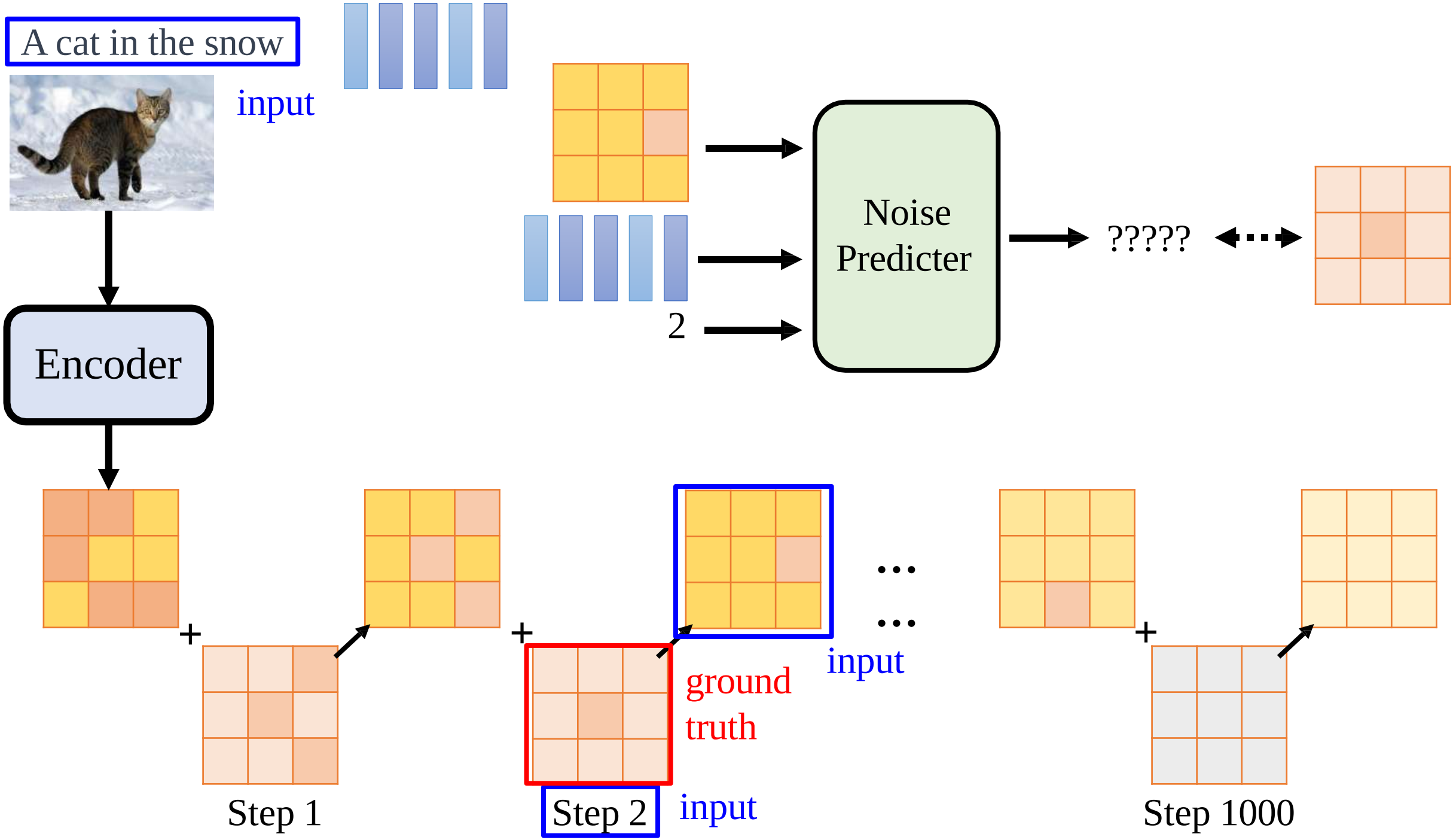
A cat in the snow → Text-to-image Generator →

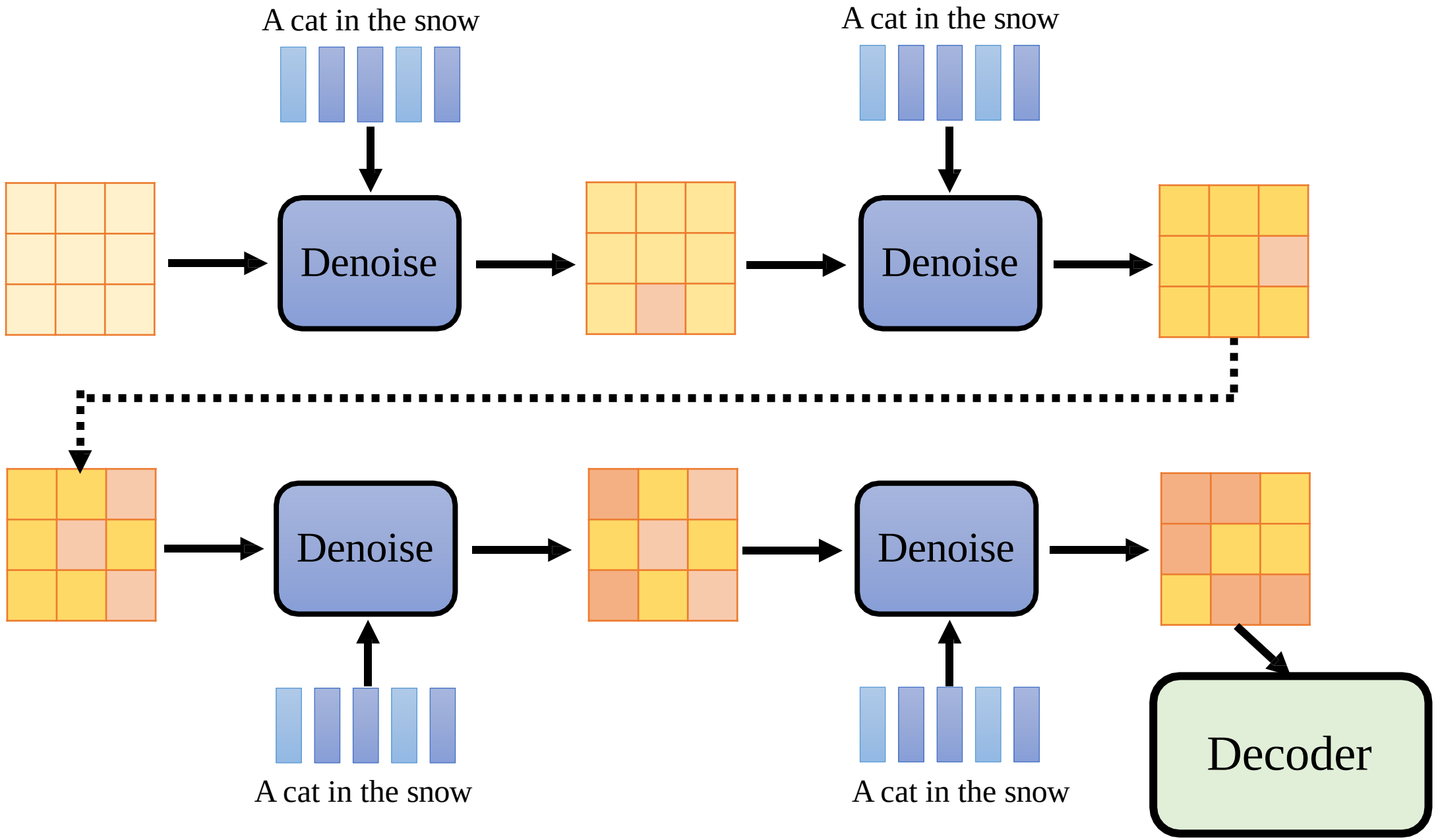
A cat in the snow → 1 Text Encoder → → Generation Model 2 ← 

Generation Model 2 →  → Decoder 3 →

A cat in the snow

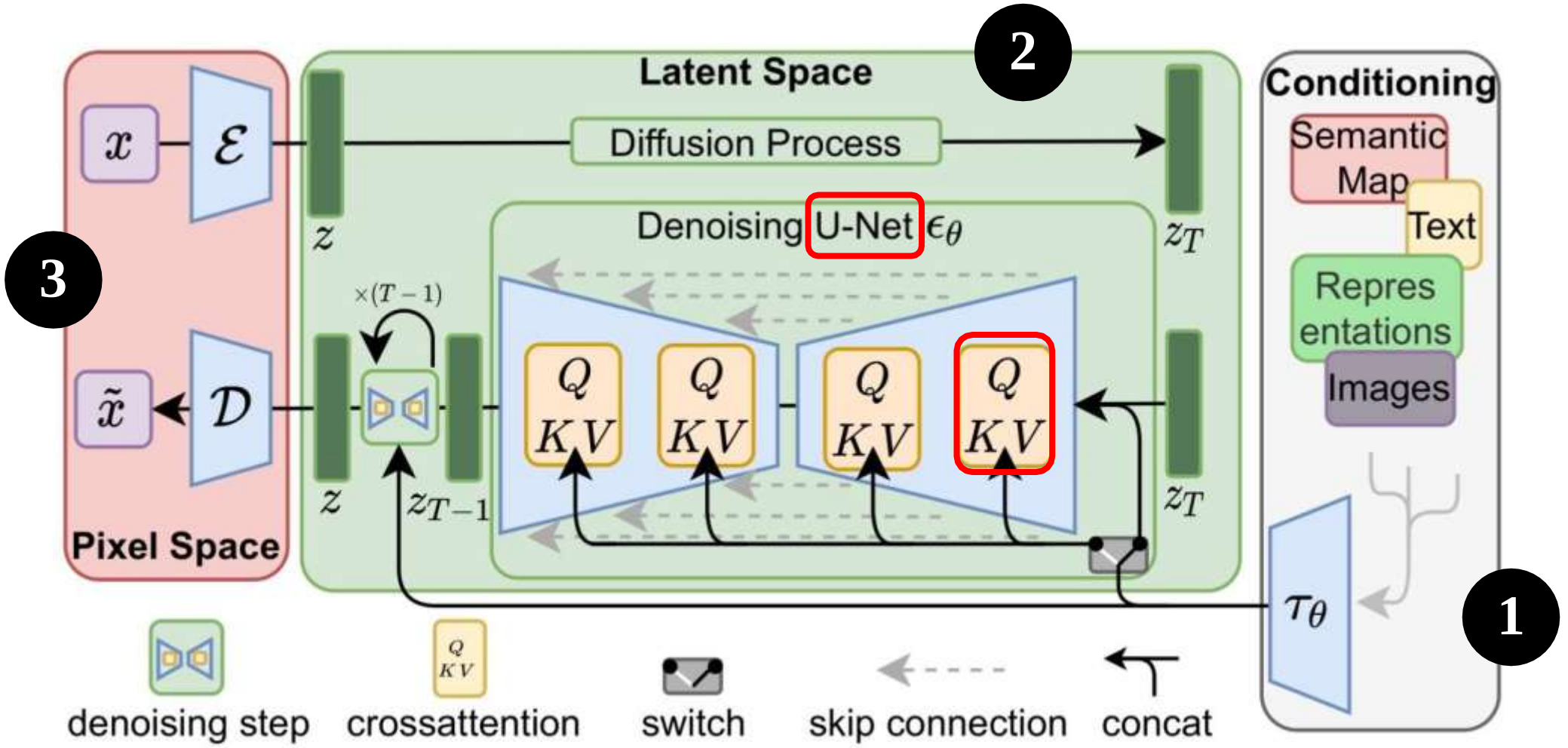



A cat in the snow
input
Encoder
2
Noise Predicter
?????
+
Step 1
+
ground truth
input
Step 2
input
...
...
+
Step 1000

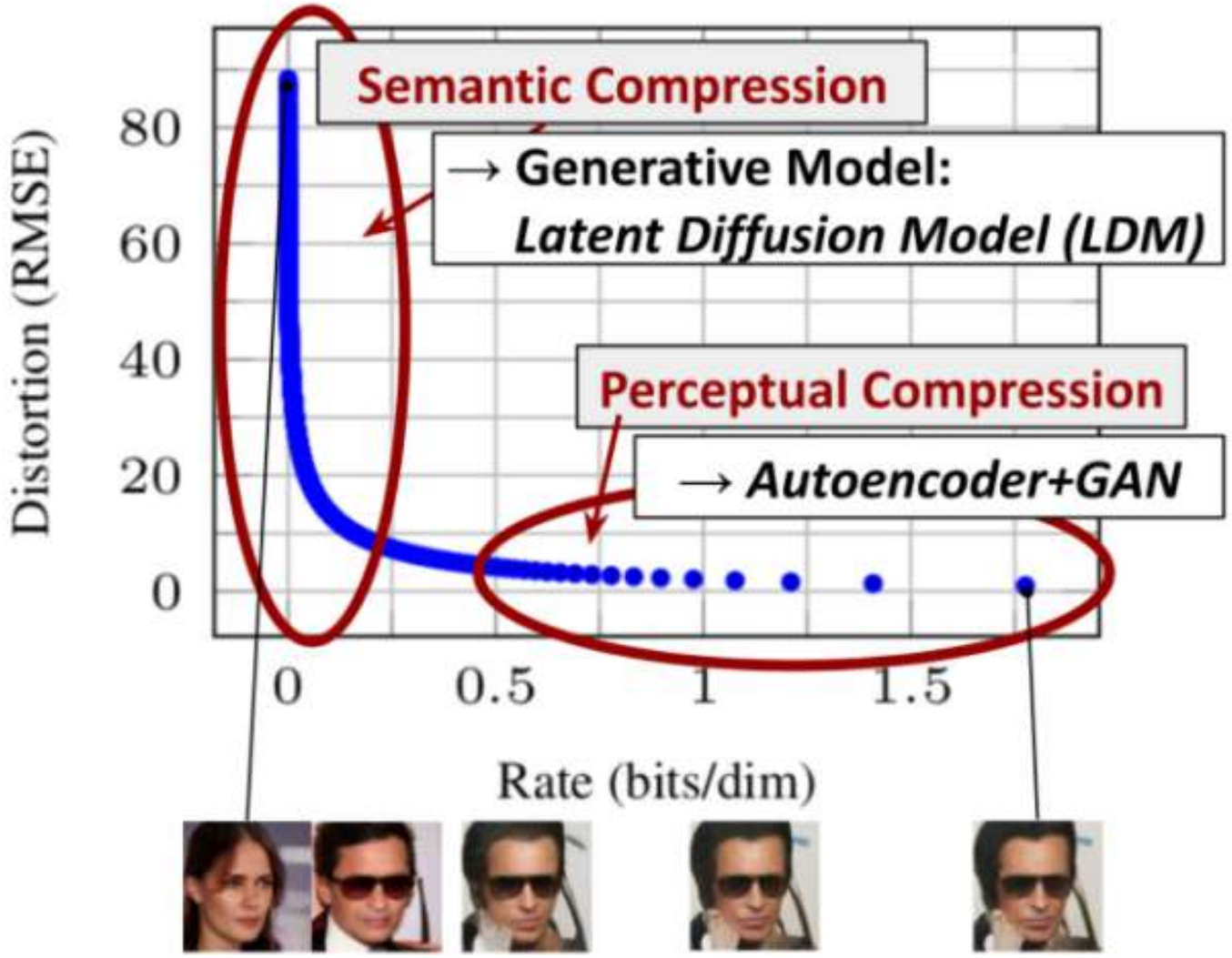

A cat in the snow
A cat in the snow
Denoise
Denoise
Denoise
Denoise
A cat in the snow
A cat in the snow
Decoder

Stable Diffusion

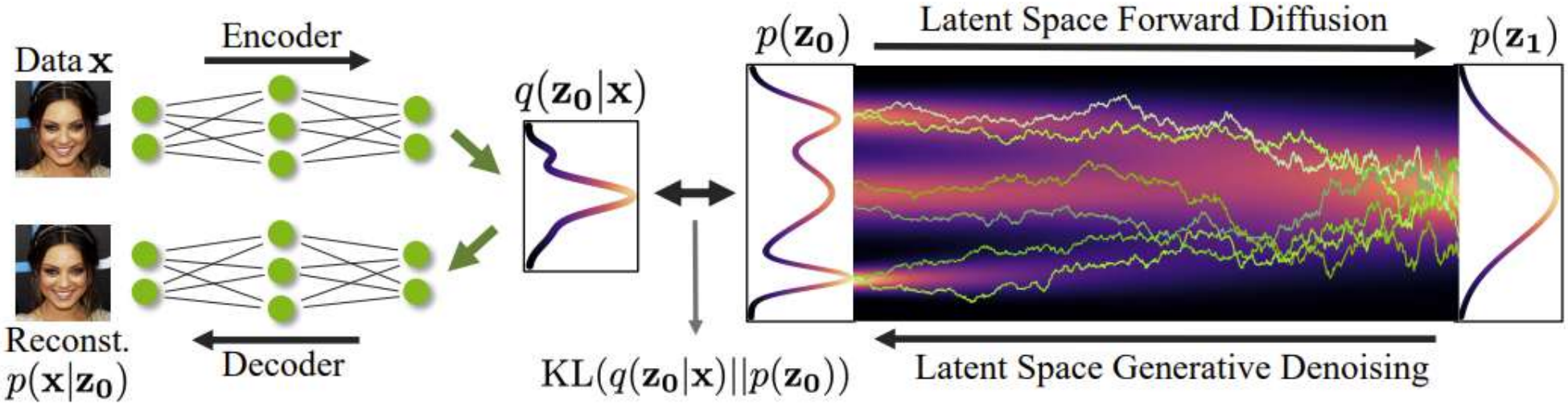
https://arxiv.org/abs/2112.10752



Latent diffusion model



Latent diffusion model

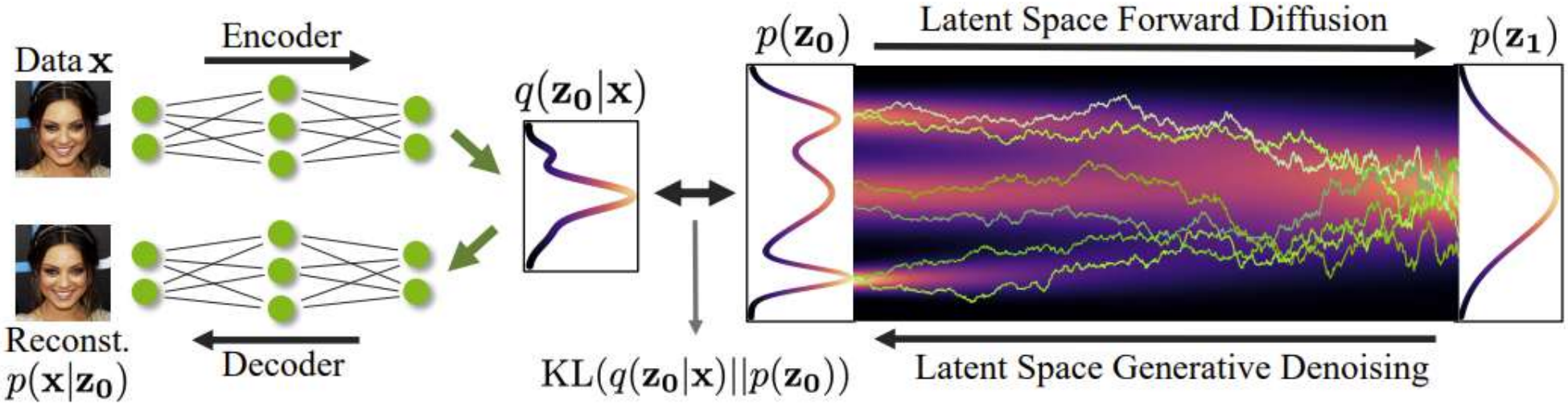


VAE mapping data to lower dimensional space

1. **Faster**
2. **Can be applied to any data type (e.g. discrete)**

Diffusion model prior over the latent space of the autoencoder

Stable diffusion text2image model



VAE mapping data to lower dimensional space

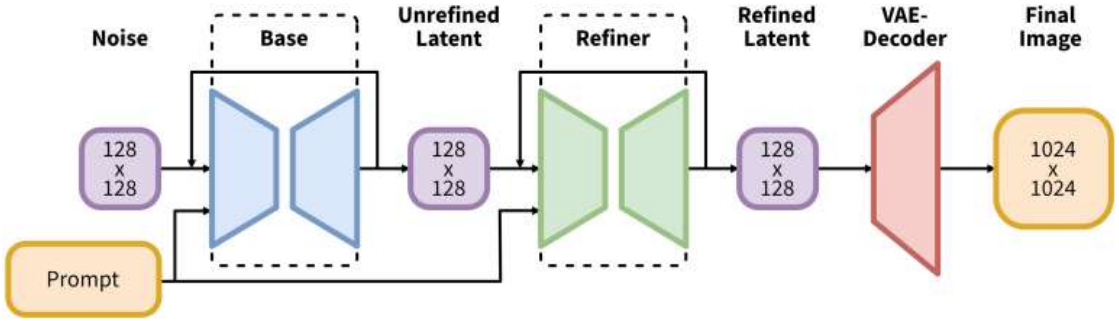
1. Pre-trained, focus on reconstruction (autoencoder)

Diffusion model prior over the latent space of the autoencoder

2. Trained in the second stage, keeping initial autoencoder fixed

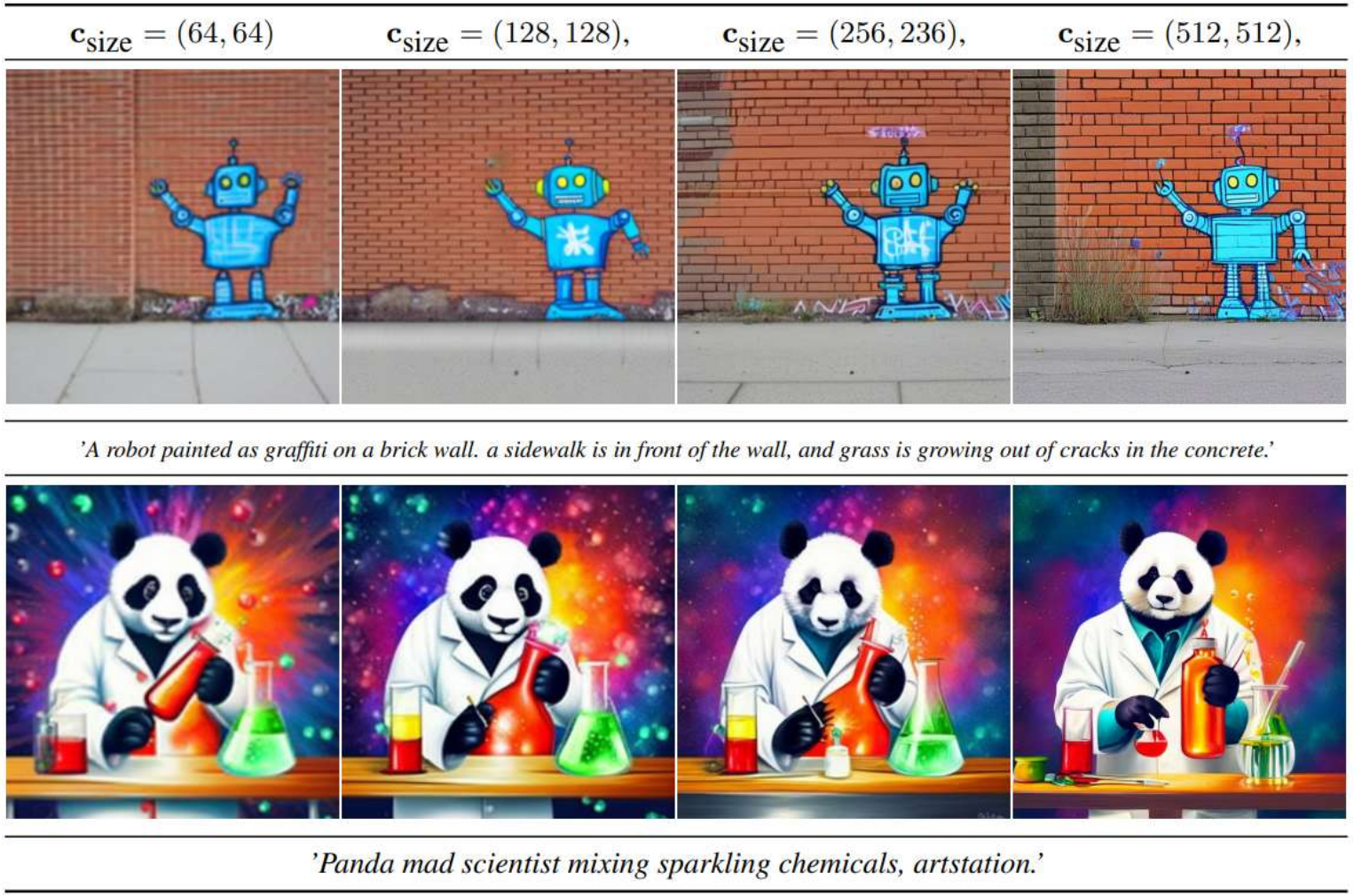
Large scale, open source model, widely adopted

SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis



https://arxiv.org/pdf/2307.01952

SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis



https://arxiv.org/pdf/2307.01952

SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis

$c_{\text{crop}} = (0, 0)$	$c_{\text{crop}} = (0, 256)$,	$c_{\text{crop}} = (256, 0)$,	$c_{\text{crop}} = (512, 512)$,
'An astronaut riding a pig, highly realistic dslr photo, cinematic shot.'			
'A capybara made of lego sitting in a realistic, natural field.'			

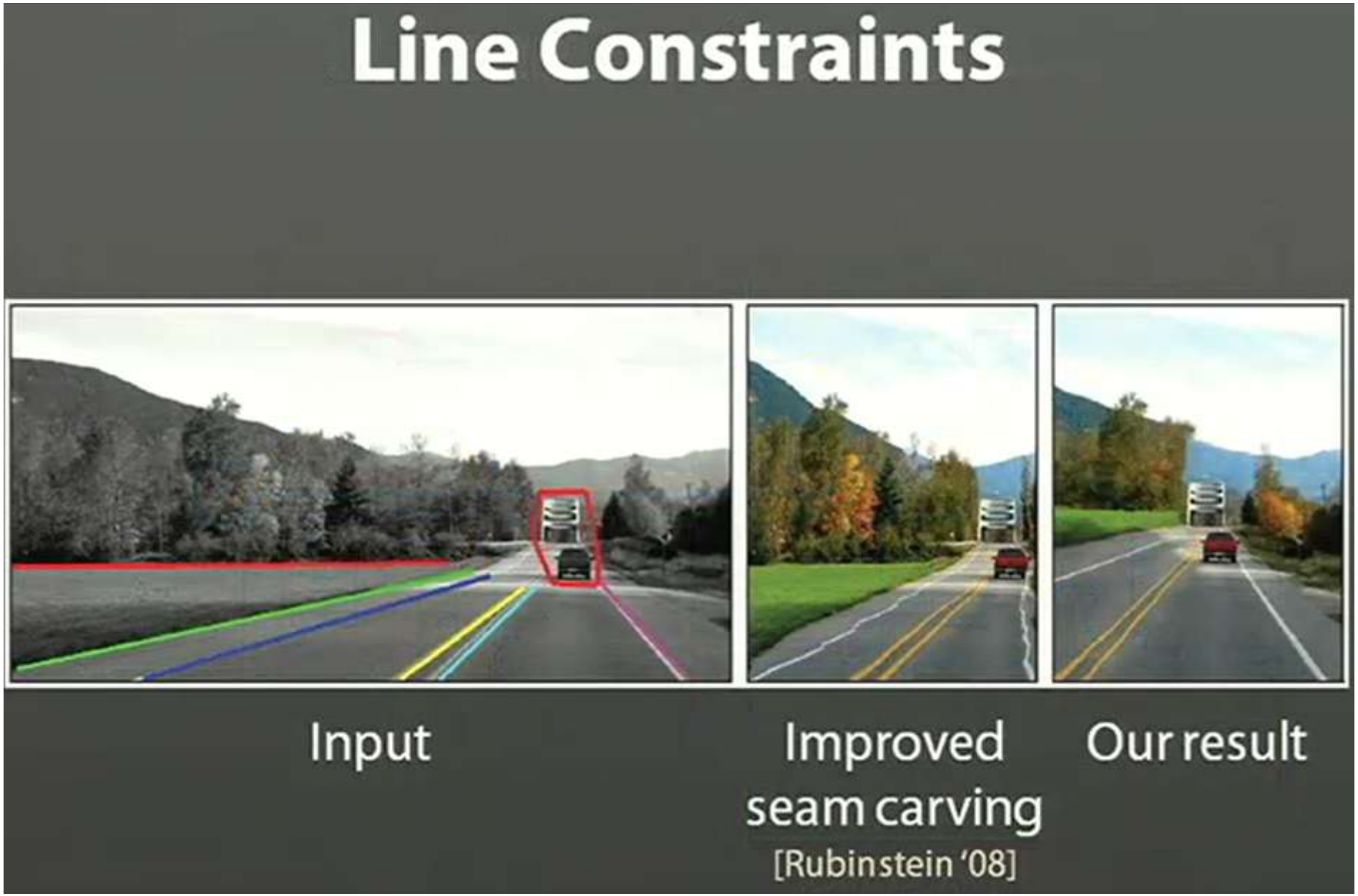
https://arxiv.org/pdf/2307.01952

SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis

https://arxiv.org/pdf/2307.01952

Recap: PatchMatch

* C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. TOG, 2009.



ControlNet: Adding Conditional Control to Text-to-Image Diffusion Models

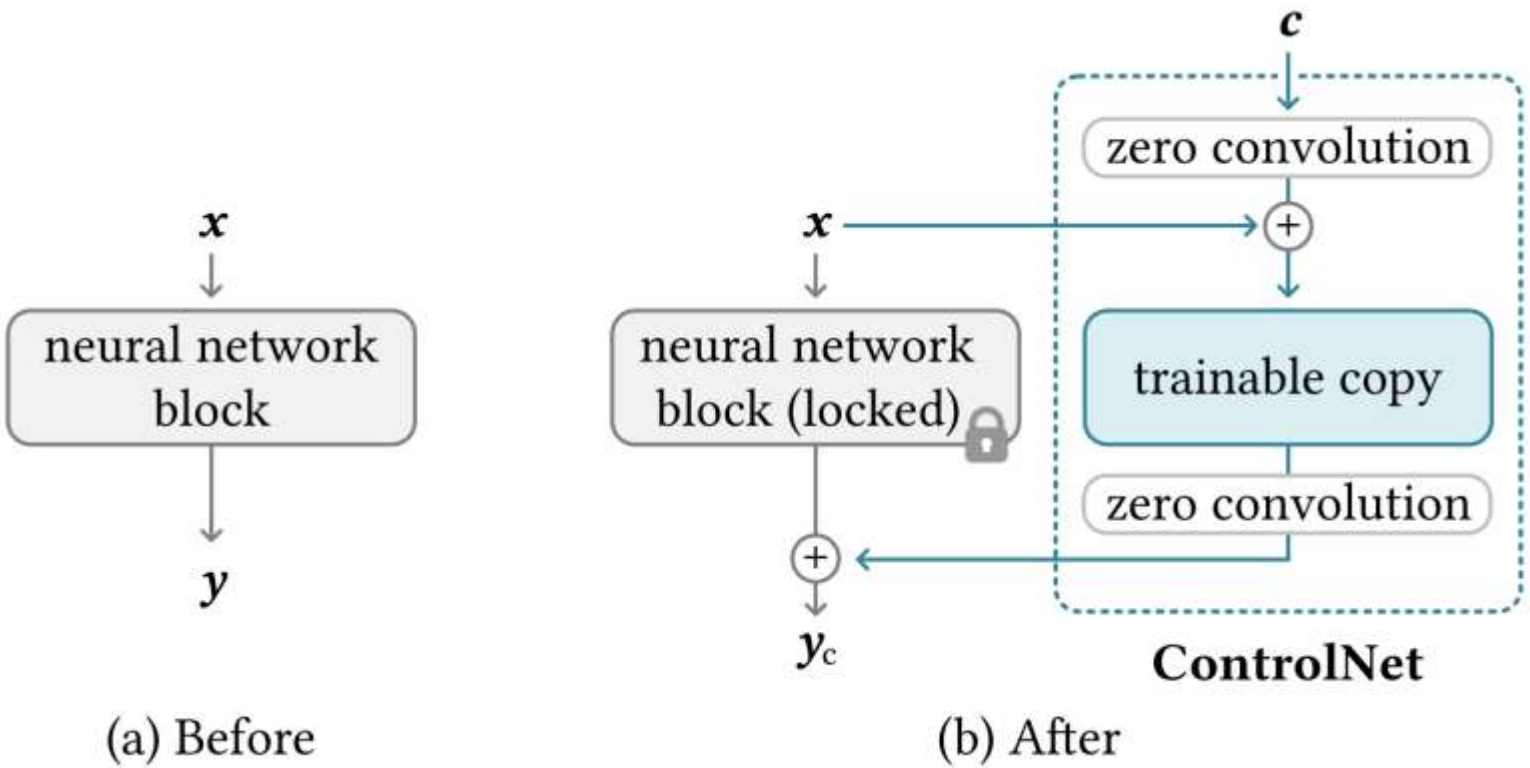
https://arxiv.org/abs/2302.05543

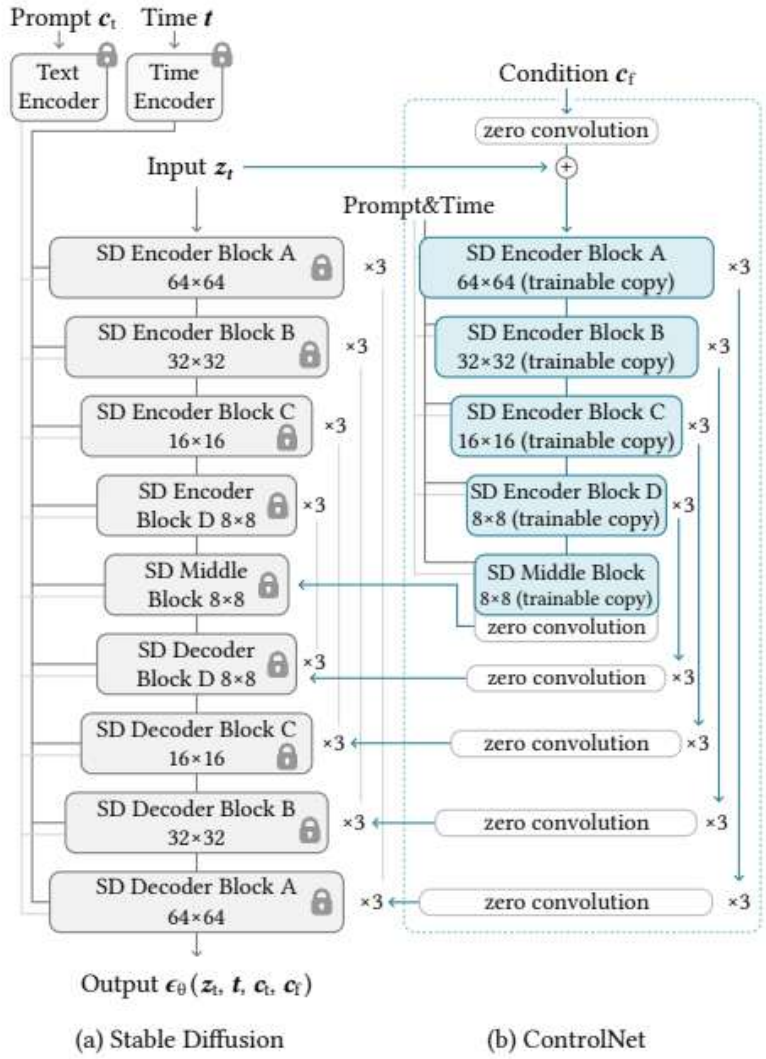
ControlNet: Adding Conditional Control to Text-to-Image Diffusion Models

Bilibili BV1Lg4y1t797 with ControlNet

https://arxiv.org/abs/2302.05543

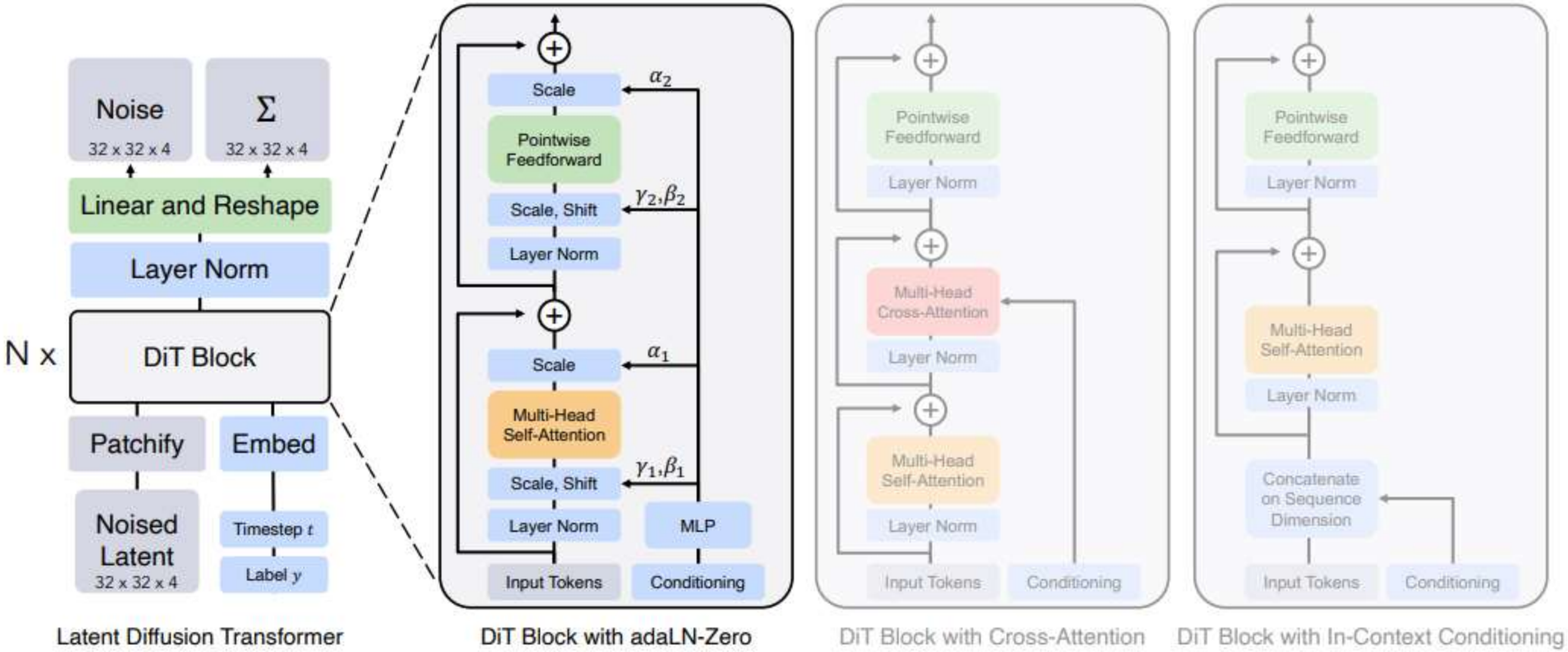
ControlNet: Adding Conditional Control to Text-to-Image Diffusion Models





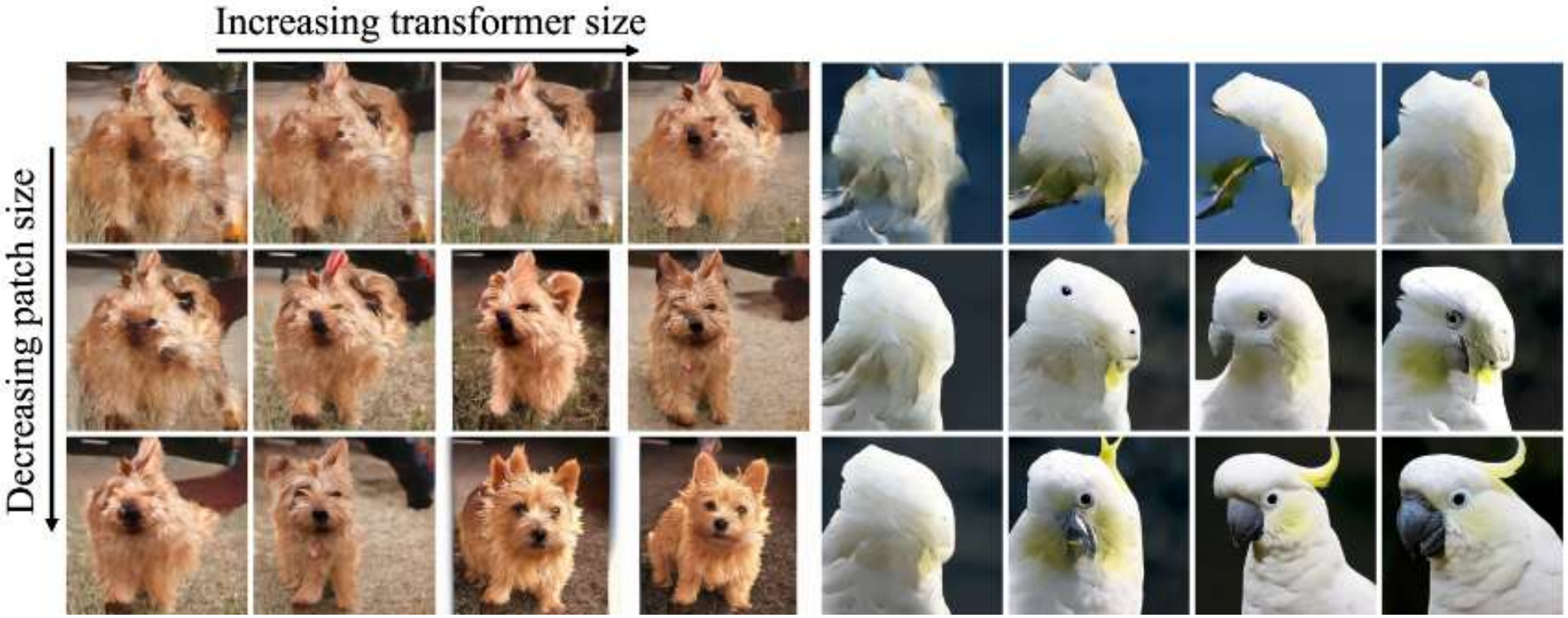
https://arxiv.org/abs/2302.05543

DiT: Scalable Diffusion Models with Transformers



https://arxiv.org/pdf/2212.09748

DiT: Scalable Diffusion Models with Transformers



https://arxiv.org/pdf/2212.09748

DiT: Scalable Diffusion Models with Transformers

DiT “Dog-Ball”

BigGAN “Dog-Ball”

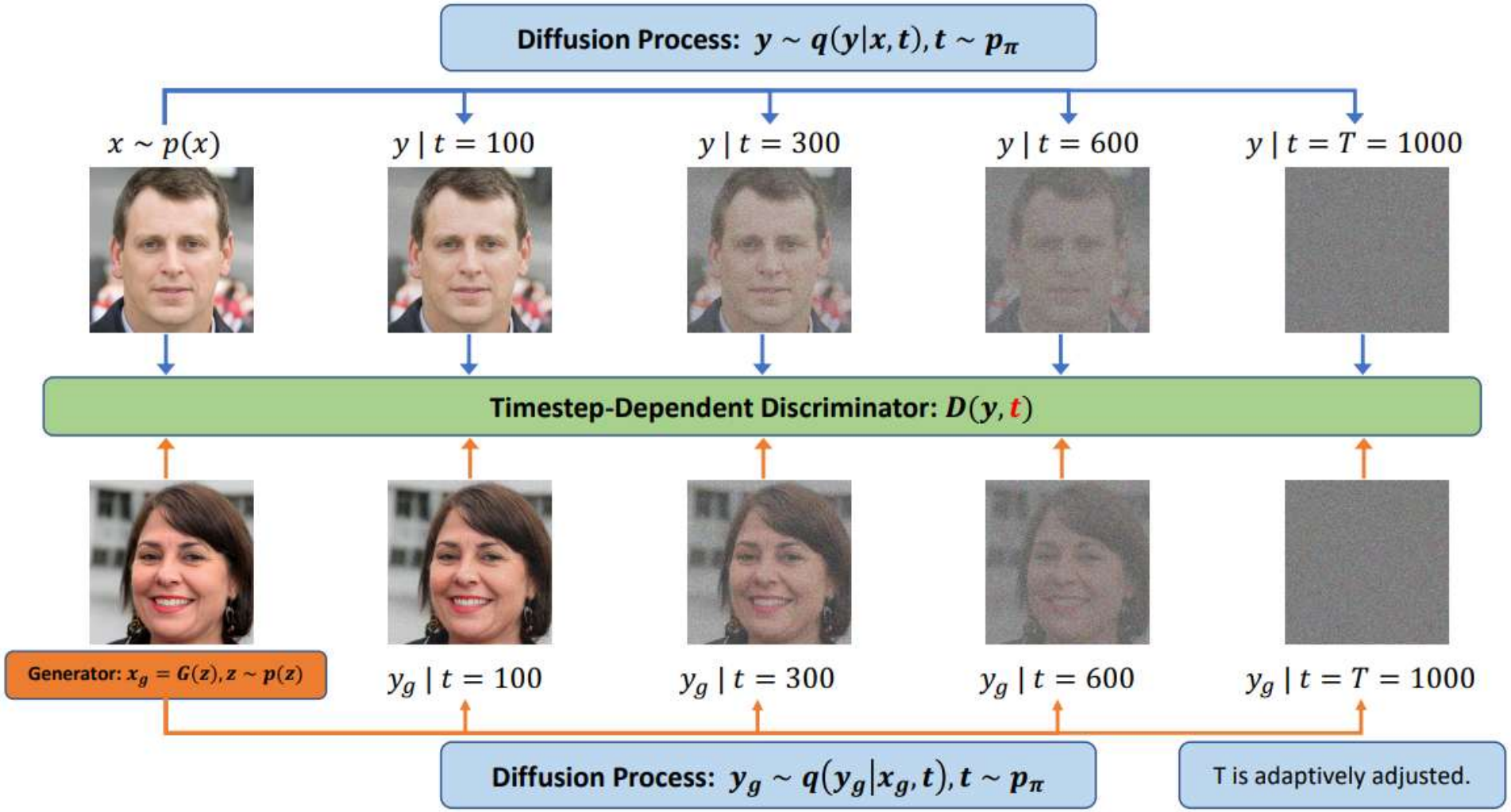
https://arxiv.org/pdf/2212.09748

DiT: Scalable Diffusion Models with Transformers

https://arxiv.org/pdf/2212.09748

When Diffusion Models Meet …

DIFFUSION-GAN: TRAINING GANS WITH DIFFUSION



Still GAN, but different strategy.
Gradually increase the difficulty by feeding it samples from larger t.

https://arxiv.org/pdf/2206.02262

Adversarial Diffusion Distillation

$s \in T_{\text{student}} = \{\tau_1, \ldots \tau_n\}$

$t \in T_{\text{teacher}} = \{1, \ldots 1000\}$

$\epsilon, \epsilon' \sim \mathcal{N}(0, [I])$

$d(x, y)$ distance metric e.g. $||x - y||_2^2$

adversarial loss

x_0

forward diffusion process

ϕ

real /fake

Discriminator

θ

$x_s = \alpha_s x_0 + \sigma_s \epsilon$

ADD-student

$\hat{x}_\theta(x_s, s)$

distillation loss

$c(t)d(\hat{x}_\theta, \hat{x}_\psi)$

stop grad

ψ🔒

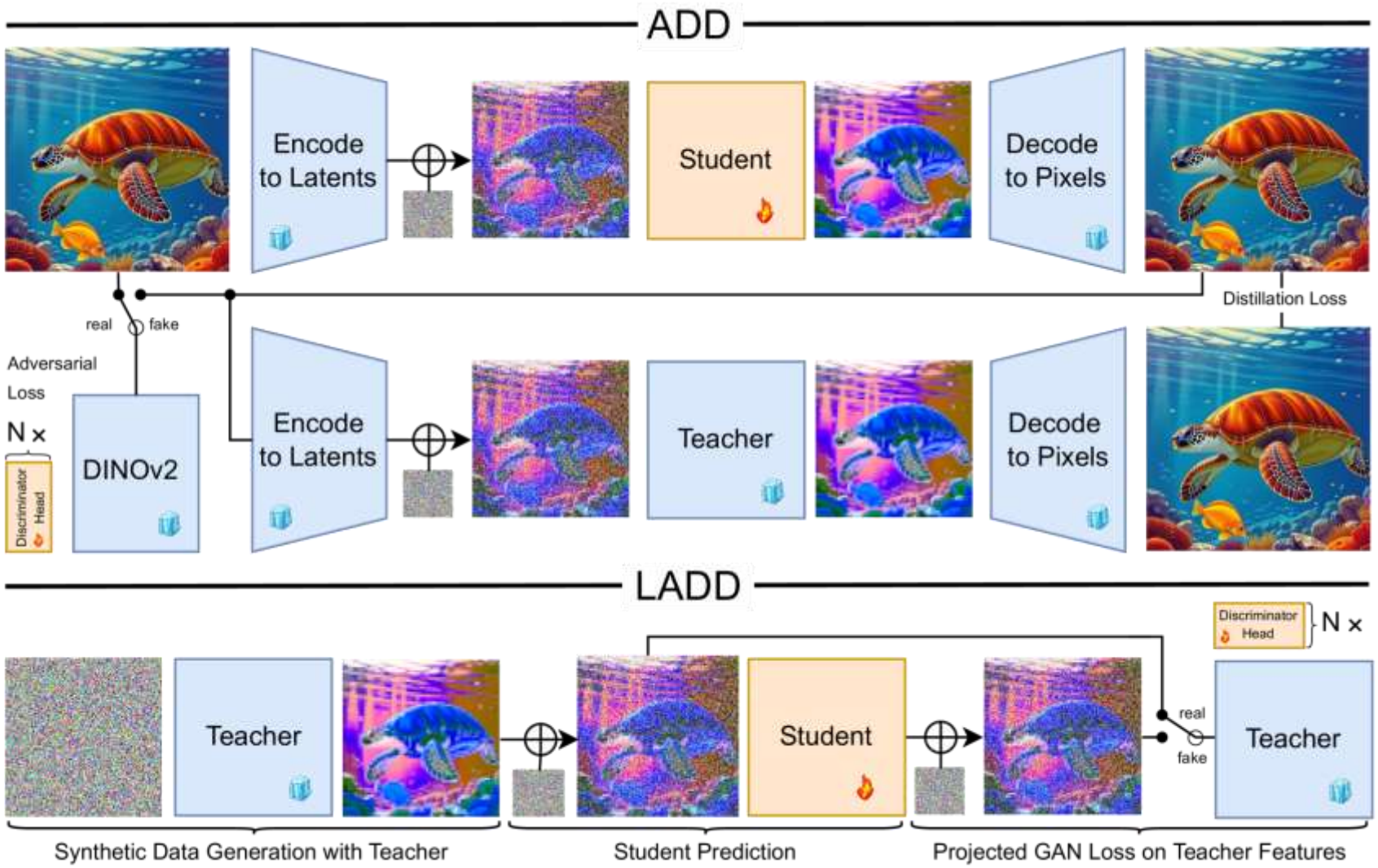
$\hat{x}_{\theta,t} = \alpha_t \hat{x}_\theta + \sigma_t \epsilon'$

DM-teacher

$\hat{x}_\psi(\hat{x}_{\theta,t}, t)$

https://arxiv.org/pdf/2311.17042

Latent Adversarial Diffusion Distillation



https://arxiv.org/html/2403.12015

DIFFUSION-GAN: TRAINING GANS WITH DIFFUSION

Methods	CIFAR-10 (32 × 32)		CelebA (64 × 64)		STL-10 (64 × 64)		LSUN-Bedroom (256 × 256)		LSUN-Church (256 × 256)		FFHQ (1024 × 1024)	
	FID	Recall	FID	Recall	FID	Recall	FID	Recall	FID	Recall	FID	Recall
StyleGAN2 (Karras et al., 2020a)	8.32*	0.41*	2.32	0.55	11.70	0.44	3.98	0.32	3.93	0.39	4.41	0.42
StyleGAN2 + DiffAug (Zhao et al., 2020)	5.79*	0.42*	2.75	0.52	12.97	0.39	4.25	0.19	4.66	0.33	4.46	0.41
StyleGAN2 + ADA (Karras et al., 2020a)	**2.92***	0.49*	2.49	0.53	13.72	0.36	7.89	0.05	4.12	0.18	4.47	0.41
Diffusion StyleGAN2	3.19	**0.58**	**1.69**	**0.67**	**11.43**	**0.45**	**3.65**	**0.32**	**3.17**	**0.42**	**2.83**	**0.49**

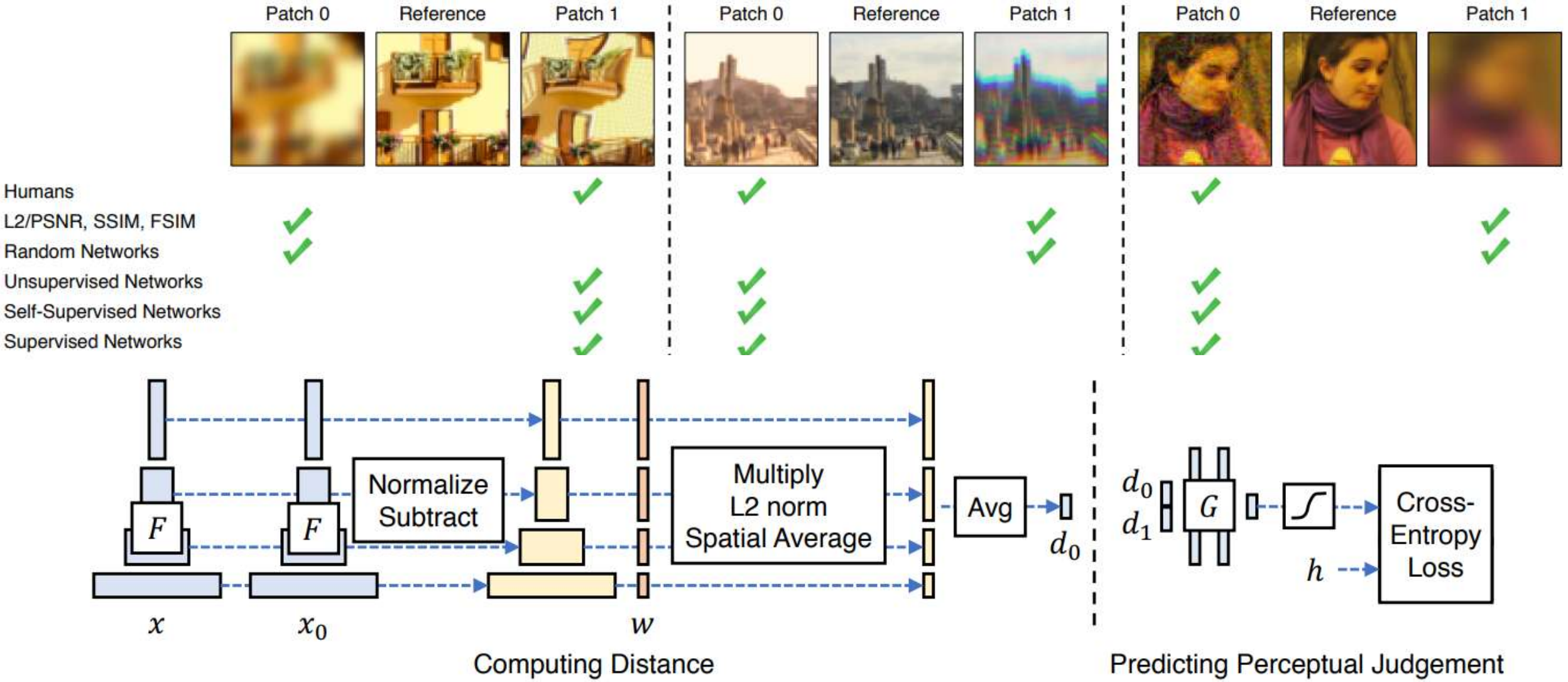
(a) CIFAR-10 (b) CelebA (c) STL-10

(d) LSUN-Bedroom (e) LSUN-Church

(f) FFHQ

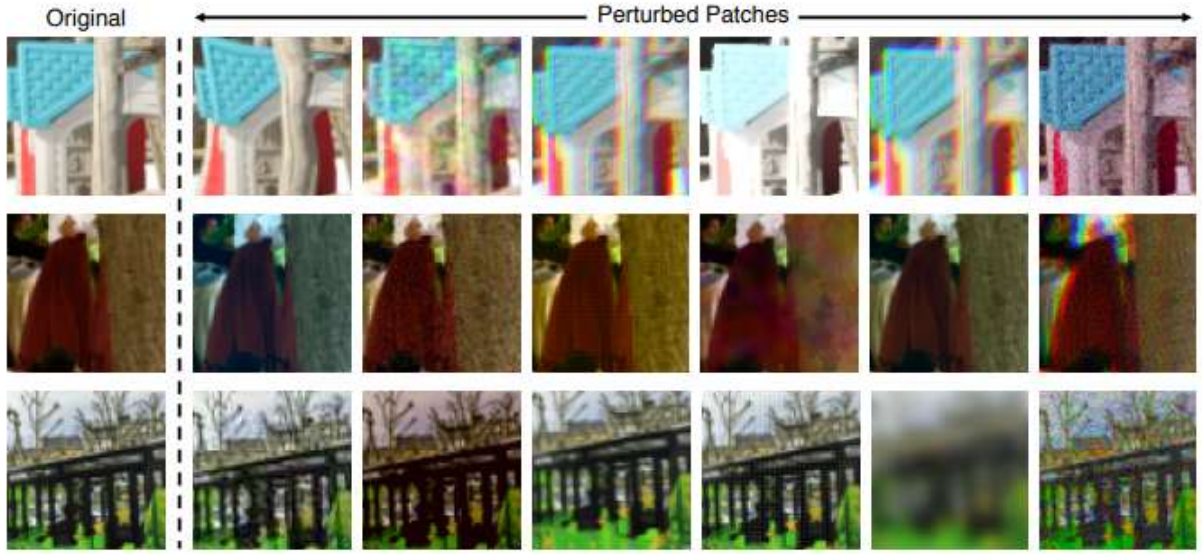
https://arxiv.org/pdf/2206.02262

LPIPS：The Unreasonable Effectiveness of Deep Features as a Perceptual Metric

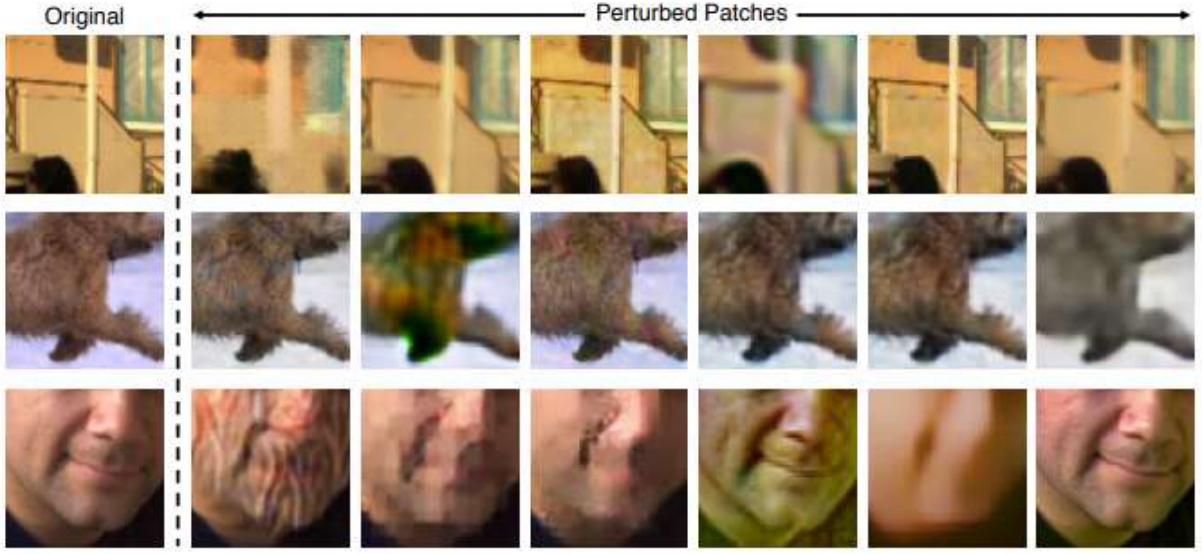


https://arxiv.org/abs/1801.03924

LPIPS：The Unreasonable Effectiveness of Deep Features as a Perceptual Metric



(a) **Traditional**



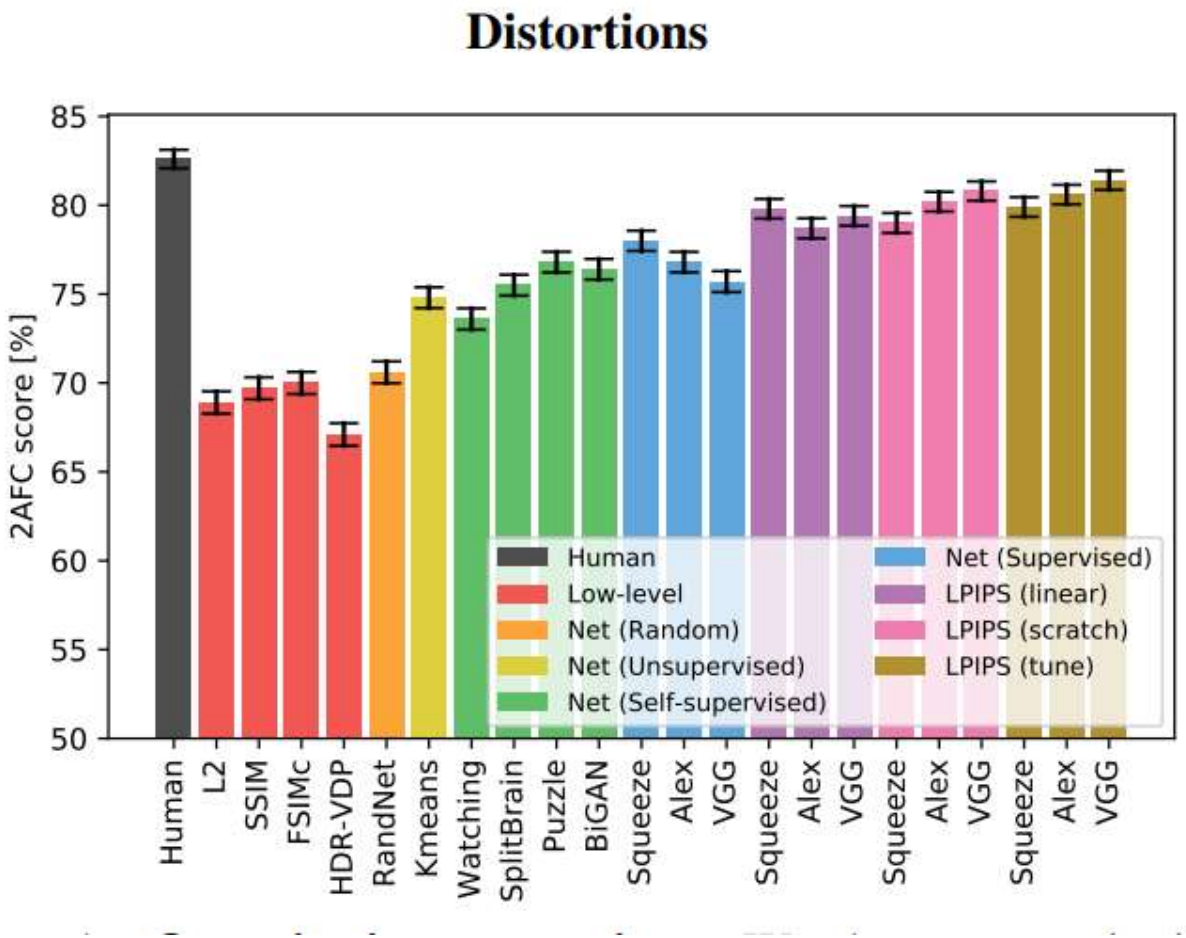
(b) **CNN-based**

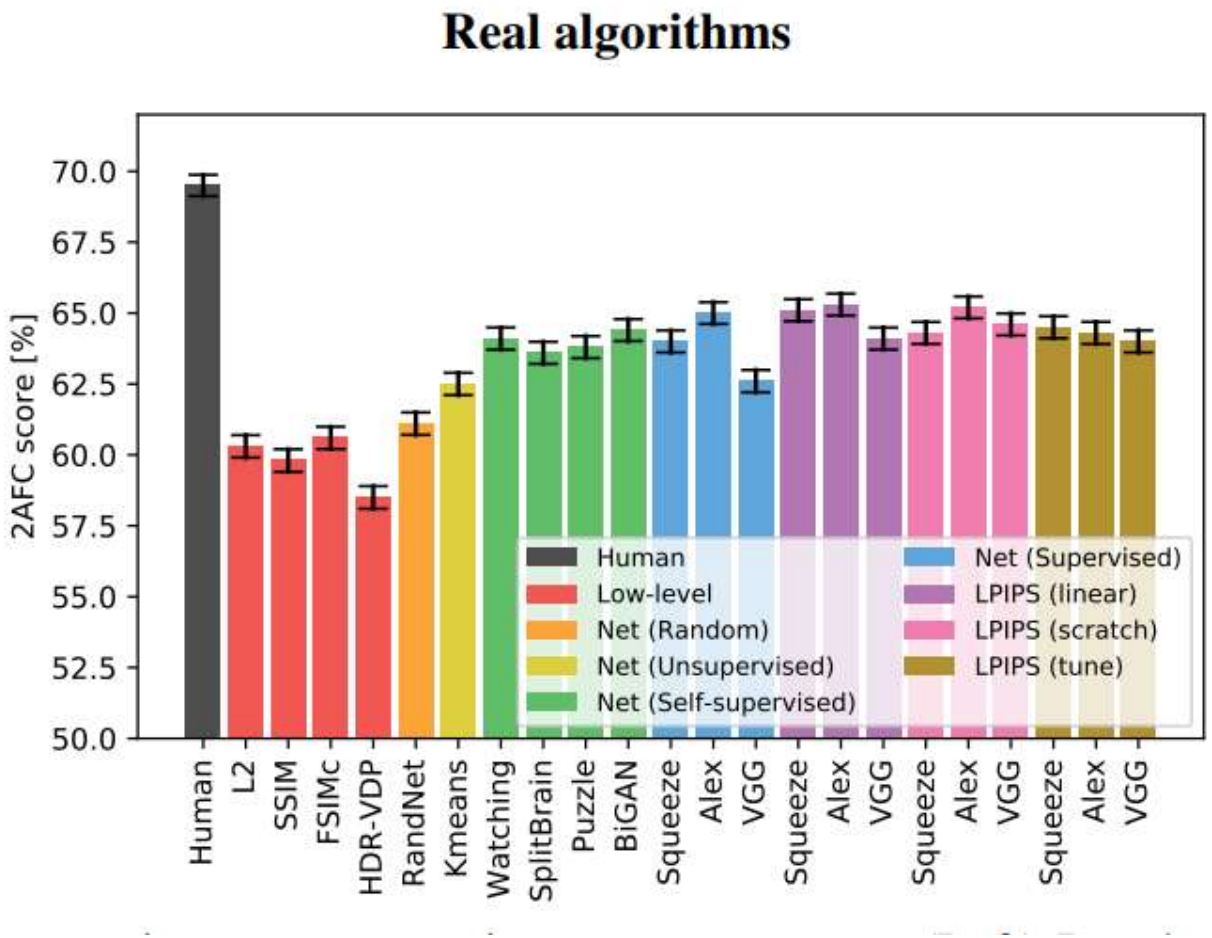
Sub-type	Distortion type
Photometric	lightness shift, color shift, contrast, saturation
Noise	uniform white noise, Gaussian white, pink, & blue noise, Gaussian colored (between violet and brown) noise, checkerboard artifact
Blur	Gaussian, bilateral filtering
Spatial	shifting, affine warp, homography, linear warping, cubic warping, ghosting, chromatic aberration,
Compression	jpeg

Parameter type	Parameters
Input corruption	null, pink noise, white noise, color removal, downsampling
Generator network architecture	# layers, # skip connections, # layers with dropout, force skip connection at highest layer, upsampling method, normalization method, first layer stride # channels in 1^{st} layer, max # channels
Discriminator	number of layers
Loss/Learning	weighting on oixel-wise (ℓ_1), VGG, discriminator losses, learning rate

https://arxiv.org/abs/1801.03924

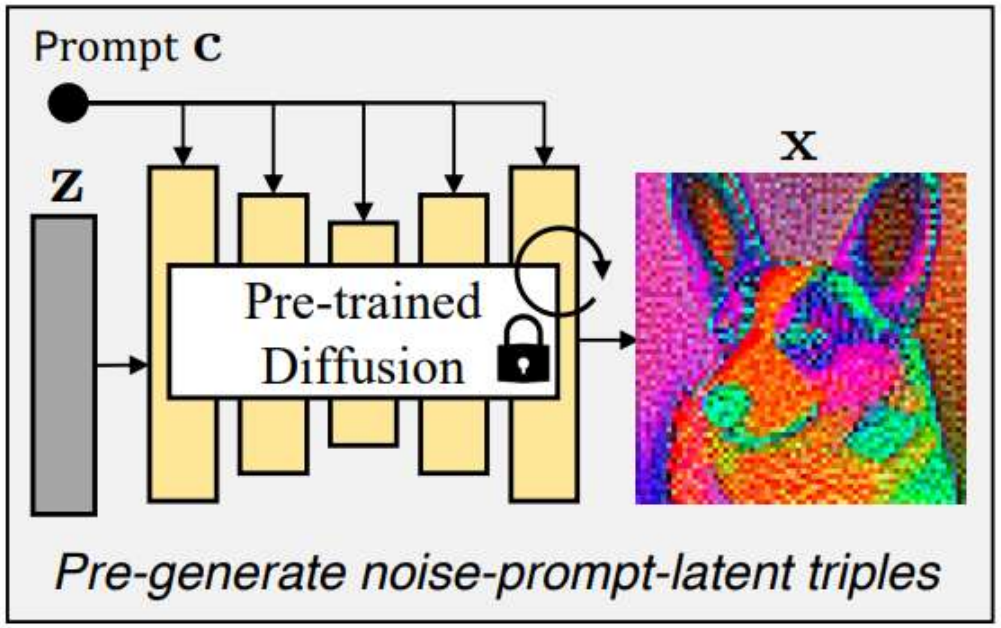
LPIPS：The Unreasonable Effectiveness of Deep Features as a Perceptual Metric

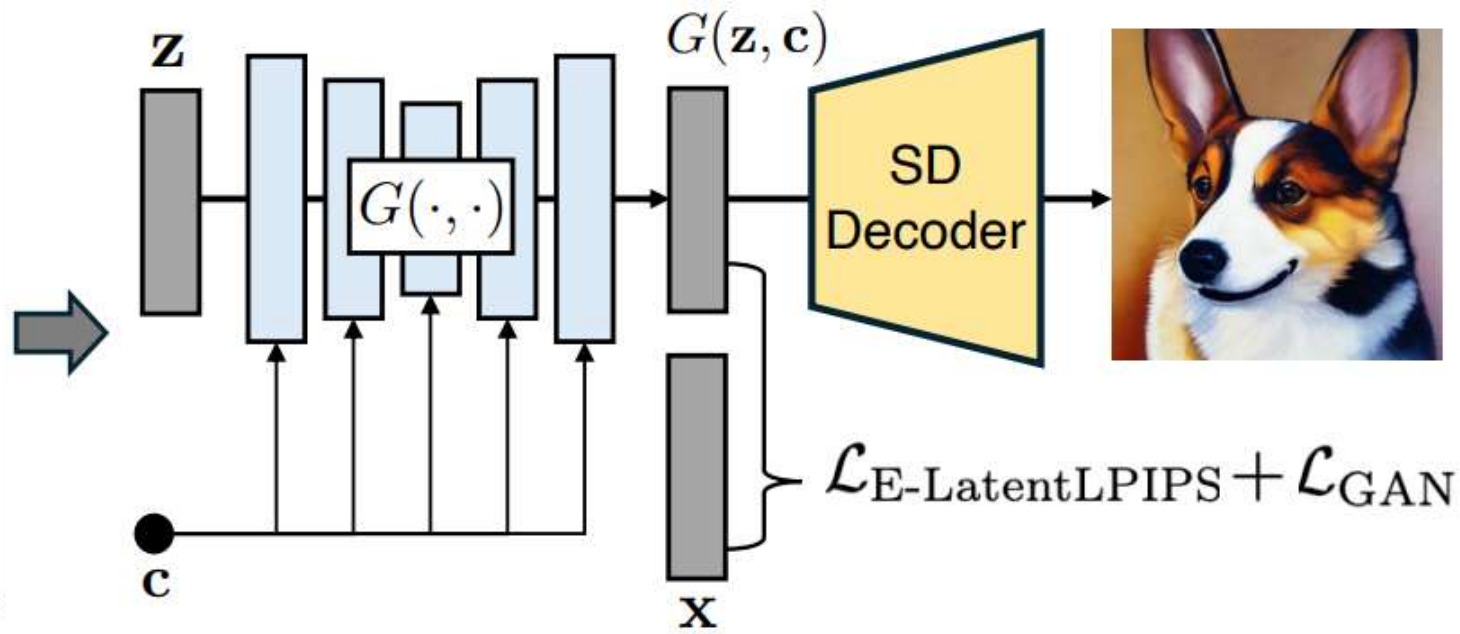




https://arxiv.org/abs/1801.03924

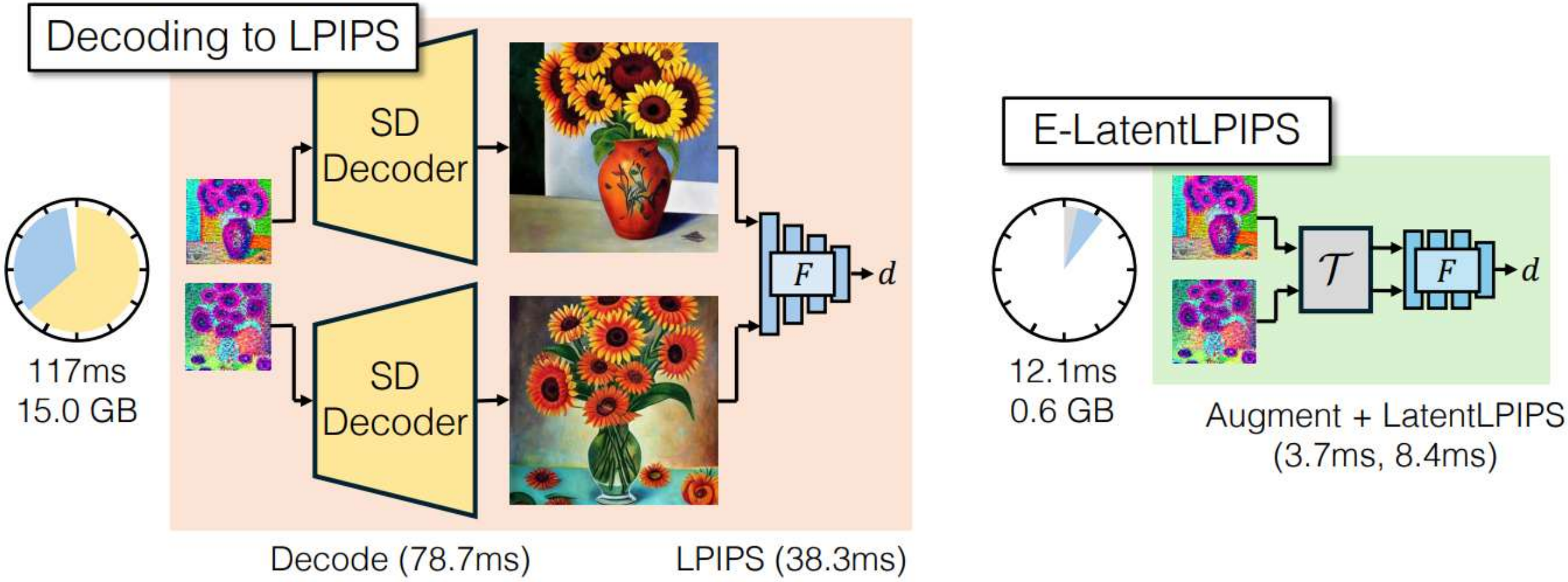
Distilling Diffusion Models into Conditional GANs





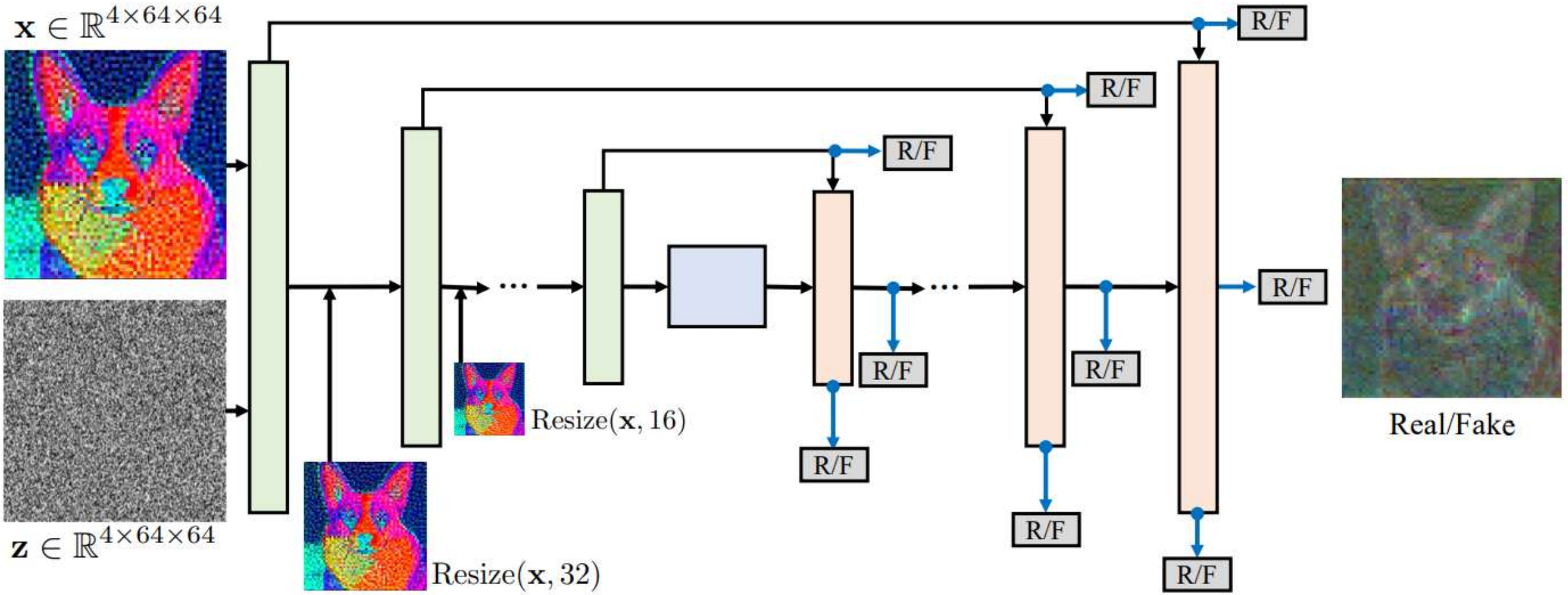
https://arxiv.org/pdf/2405.05967

Distilling Diffusion Models into Conditional GANs



https://arxiv.org/pdf/2405.05967

Distilling Diffusion Models into Conditional GANs

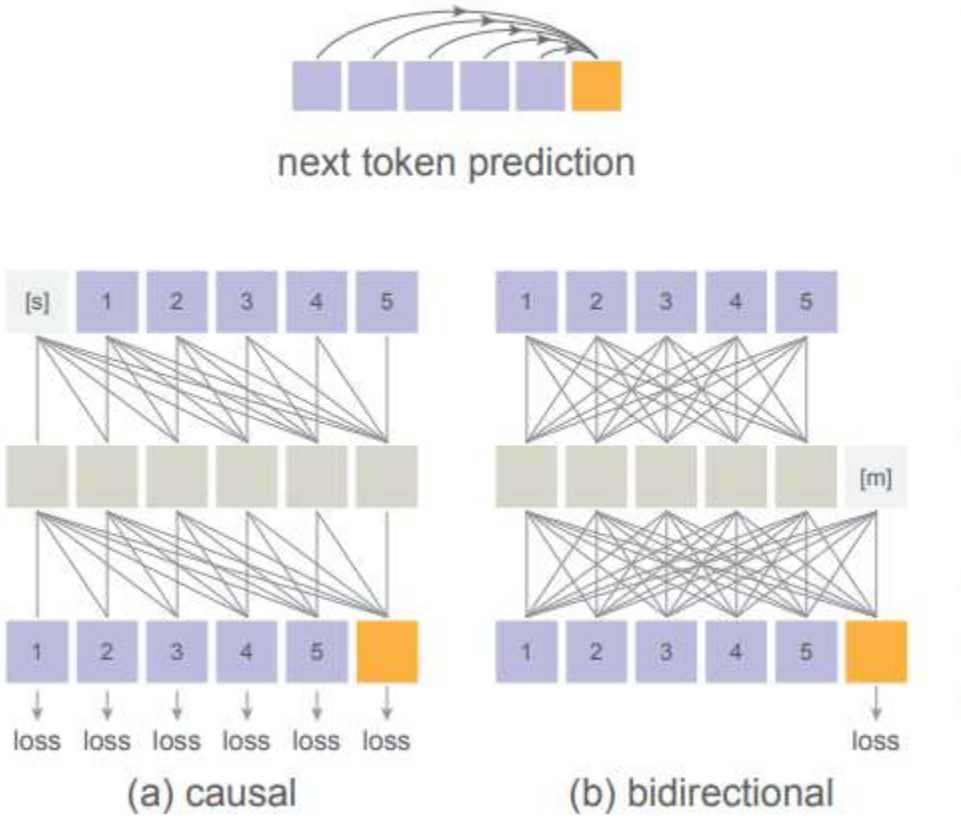


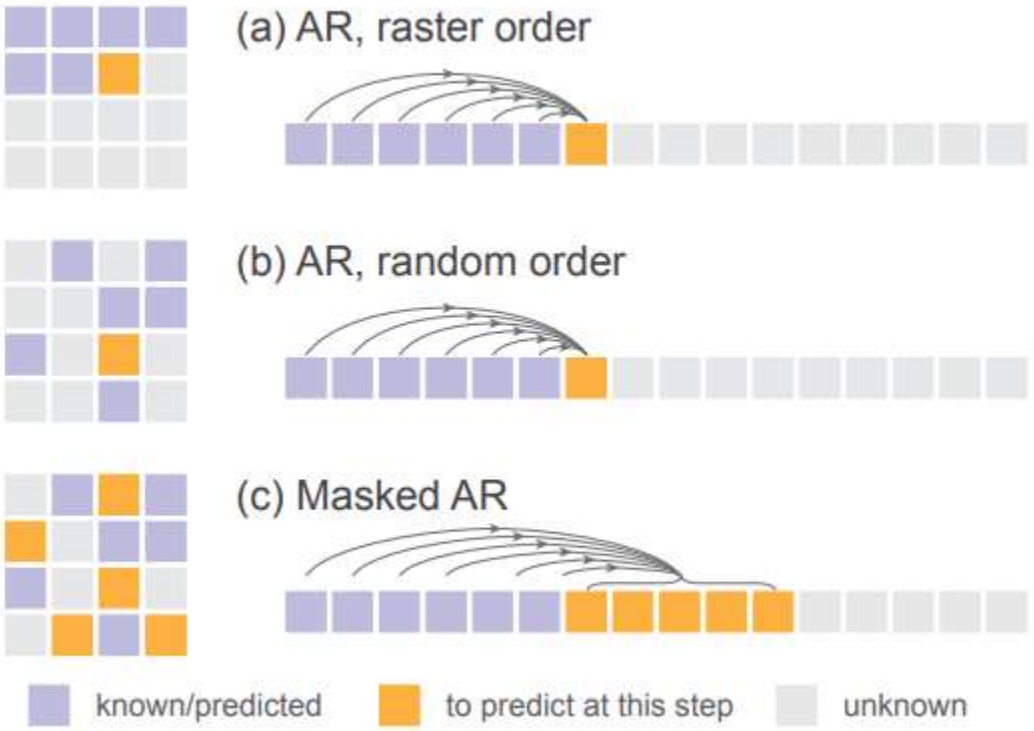
https://arxiv.org/pdf/2405.05967

Distilling Diffusion Models into Conditional GANs

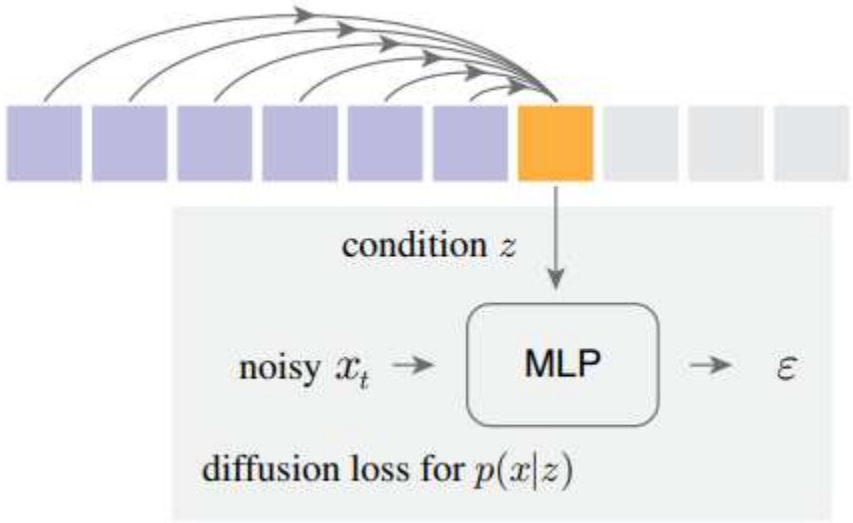
https://arxiv.org/pdf/2405.05967

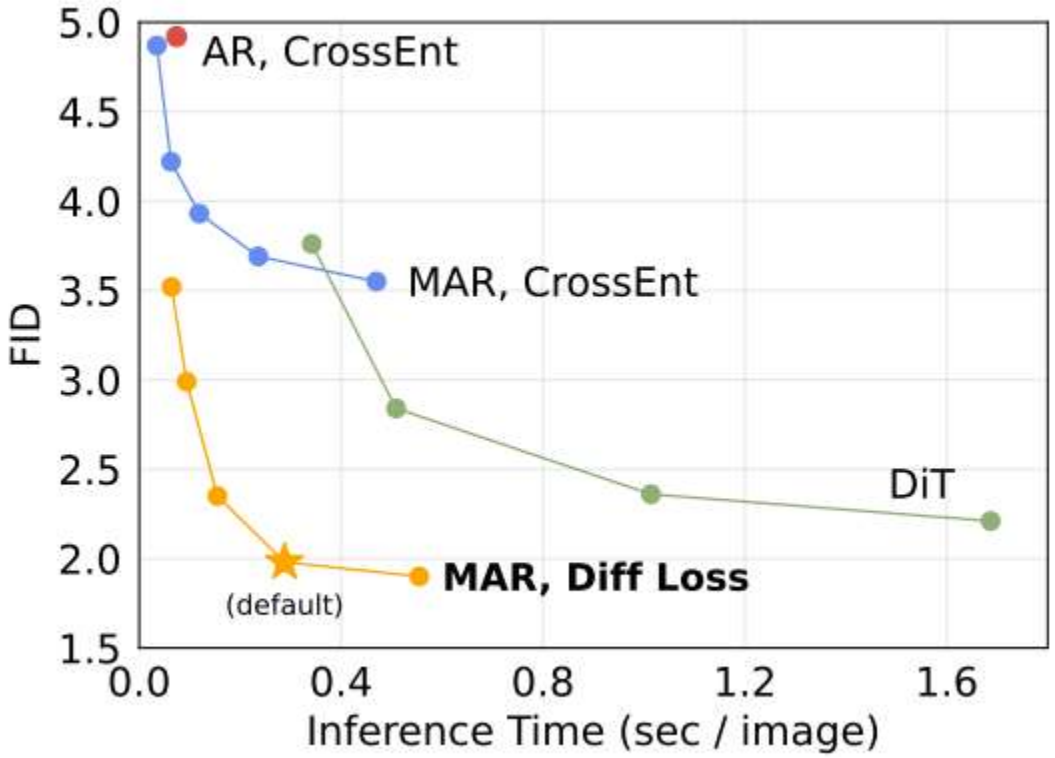
Recap: Auto-Regressive Models





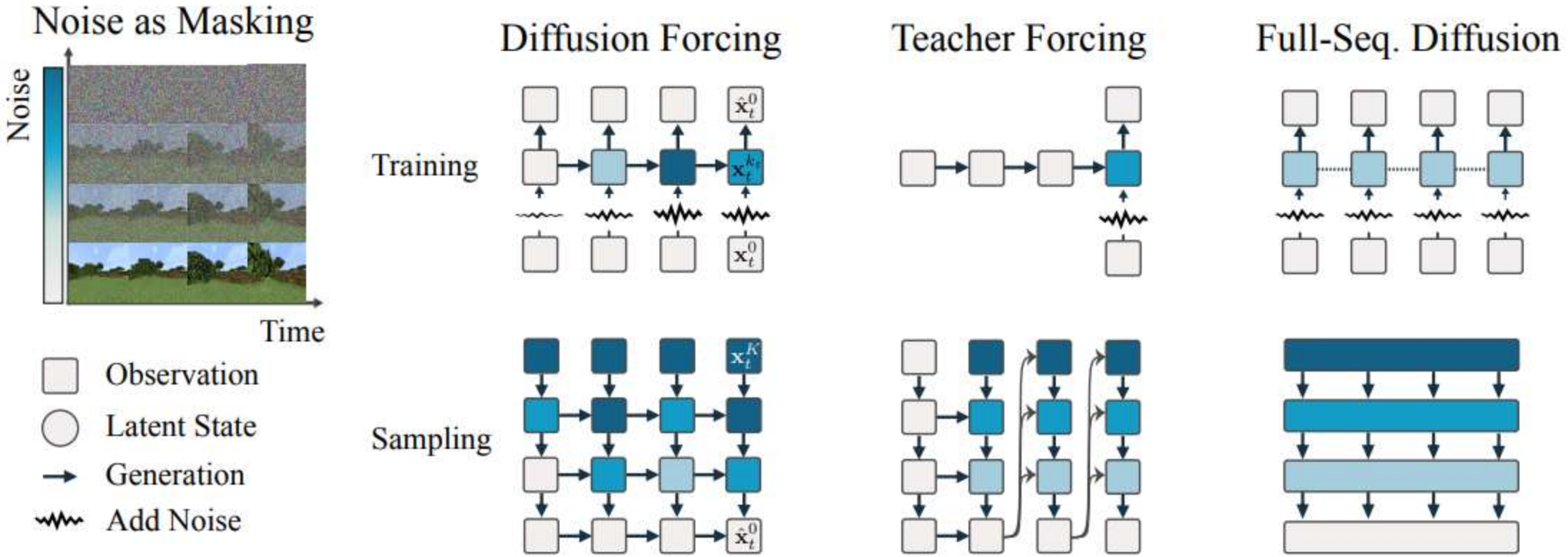
Autoregressive Image Generation without Vector Quantization





https://arxiv.org/pdf/2406.11838

Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion



https://arxiv.org/pdf/2407.01392