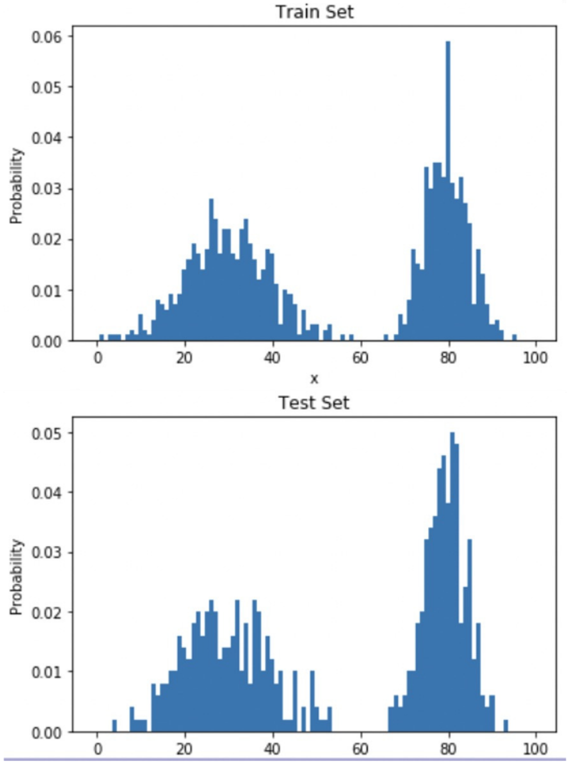
Diffusion

EECS 442 Fall 2023

Presenters: Sarah Jabbour, Yiming Dou, Daniel Geng

Recall: Data Synthesis

Lets say we have some training set of data following some distribution $p_{data}(x)$:



The goal of generative machine learning models is to *learn* this distribution to the best of their ability

We generate new data by *sampling* from the learned distribution

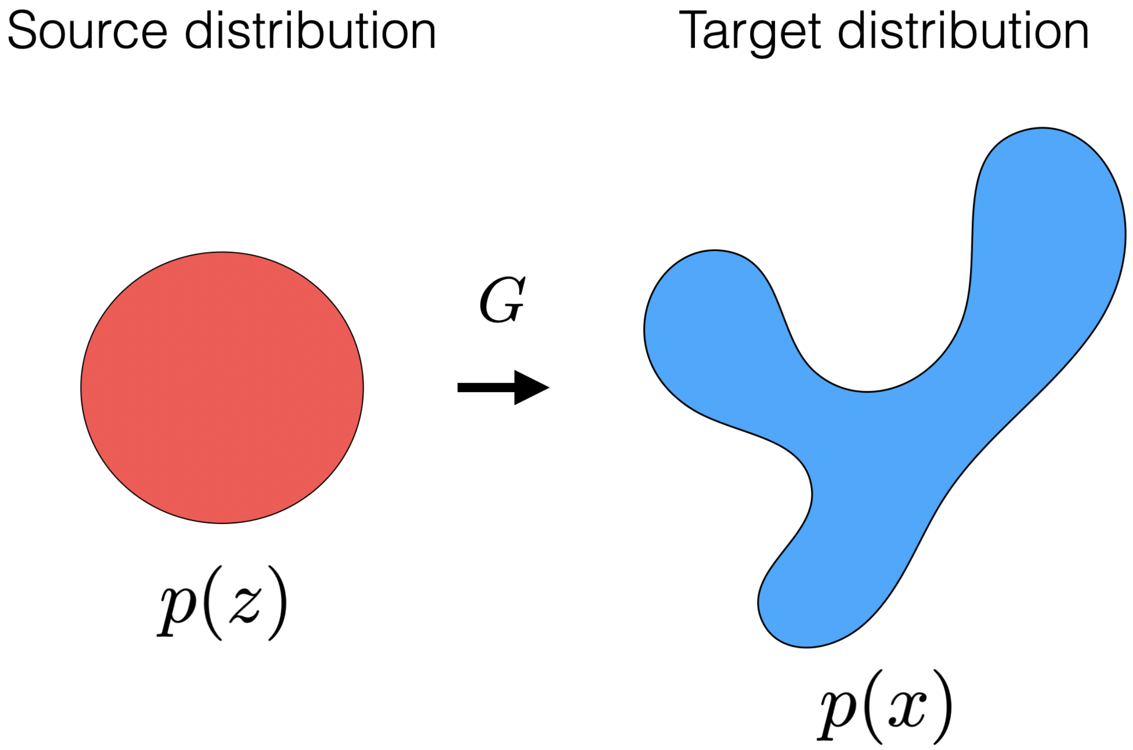
2 Methods for Learning Data Distribution

(1) We want to train models that maximize the expected log likelihood of $p_\theta(x)$. I.e., If I sample from the distribution and get a

- high likelihood → likely the sample came from the training distribution
- low likelihood → the sample probably didn't come from the training distribution

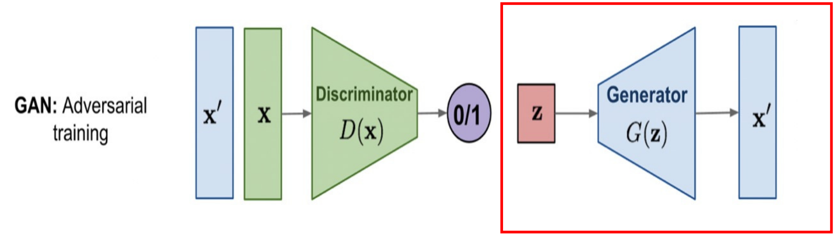
(2) We want to minimize some divergence metrics between the training data distribution, and the distribution that the model learns

Sampling from Noise



GANs

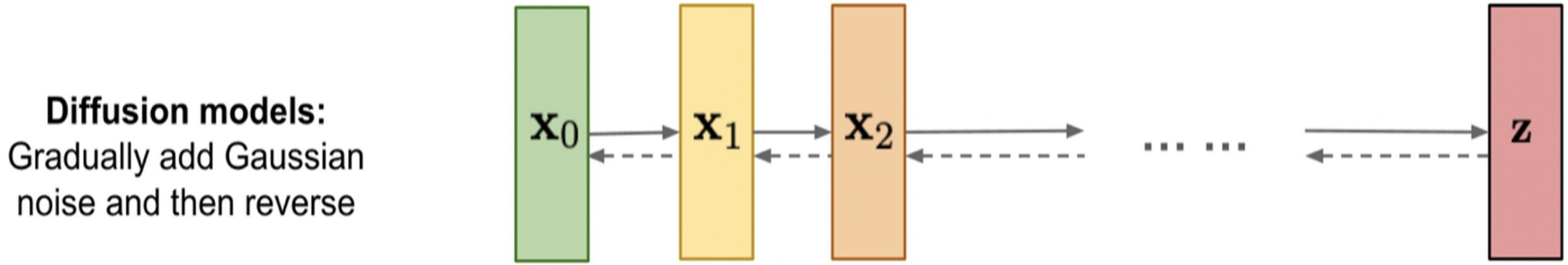
- Convert latent noise vector to target distribution in one step



- Lots of issues with training:
  - Vanishing gradients: if discriminator is too good, gradients go to zero
  - Mode collapse: if the generator learns to generate an exceptionally plausible output, it will just continue generating it. Then the discriminator will learn to always reject it, and then the generator will produce the same outputs, which the discriminator will then reject… bad loop!

Diffusion

- Idea: Estimating and analyzing small step sizes is more tractable/easier than a single step from random noise to the learned distribution
- Convert a well-known and simple *base distribution* (like a Gaussian) to the *target (data) distribution* iteratively, with small step sizes, via a Markov chain:



- Markov chain: outlines the probability associated with a sequence of events occurring based on the state in the previous event.

Forward Process

- Noise added can be parameterized by:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad \{\beta_t \in (0,1)\}_{t=1}^{T}$$

Vary the parameters of the Gaussian according to a *noise schedule*

- You can prove with some math that as T approaches infinity, you eventually end up with an Isotropic Gaussian (i.e. pure random noise)
- Note: forward process is fixed

Reparameterization trick

Do you *have* to add noise *iteratively* to get to some timestep t? Nope!

Reverse process can be written in one step:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0,\ (1 - \bar{\alpha}_t)\mathbf{I}\right)$$

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

This will be useful during training!

Implementing Forward Process

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0,\ (1 - \bar{\alpha}_t)\mathbf{I}\right)$$

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

1. Sample an image from the dataset: 
2. Sample noise $\epsilon \sim N(0, \mathbf{I})$ (from a **standard** normal distribution)
3. Scale the image by $\sqrt{\overline{\alpha_t}}$: $\sqrt{\overline{\alpha_t}}\, x_0$

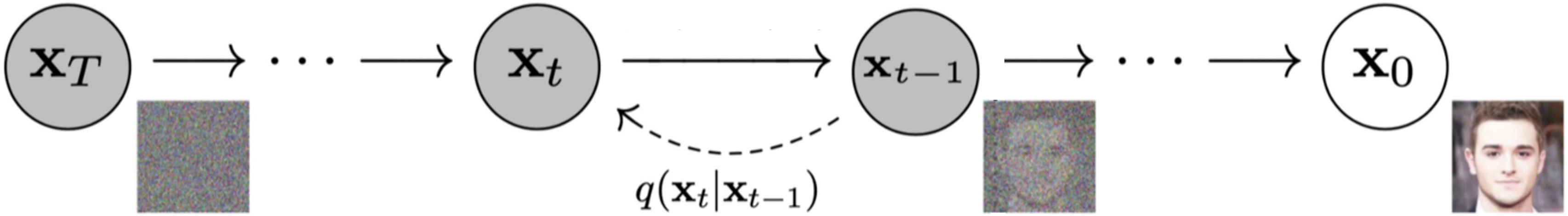
where
$$\alpha_t = 1 - \beta_t$$
$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

4. Add $\sqrt{1 - \overline{\alpha_t}}\, \epsilon$: $\sqrt{\overline{\alpha_t}}\, x_0 + \sqrt{1 - \overline{\alpha_t}}\, \epsilon$ →

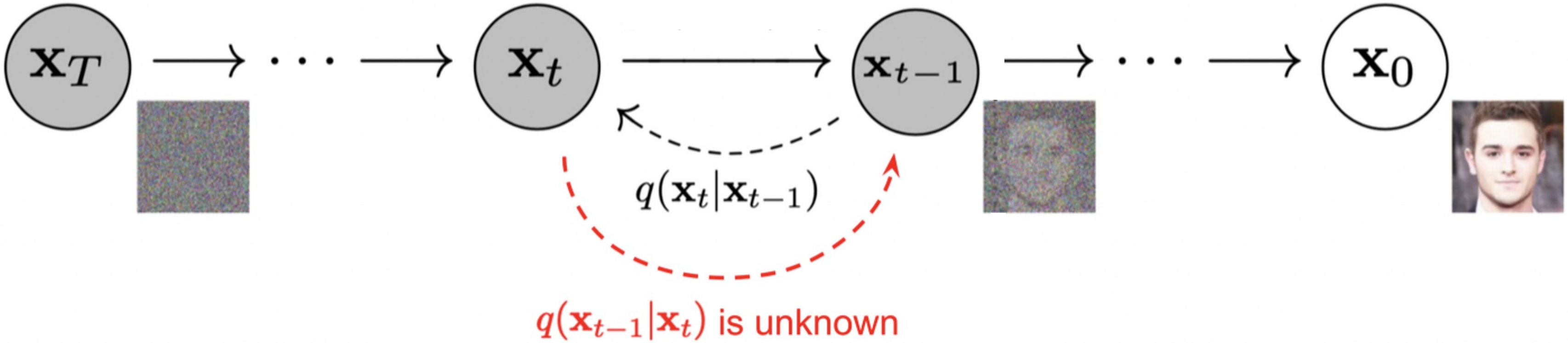
x_t

Reverse Process

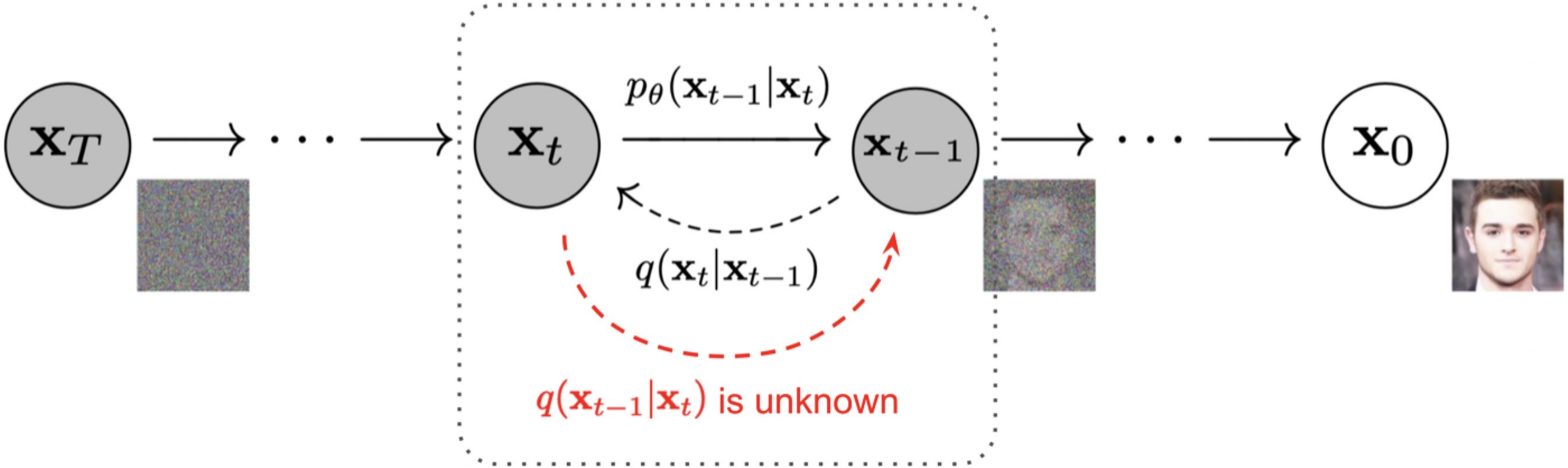
Reverse Process



Reverse Process



Reverse Process

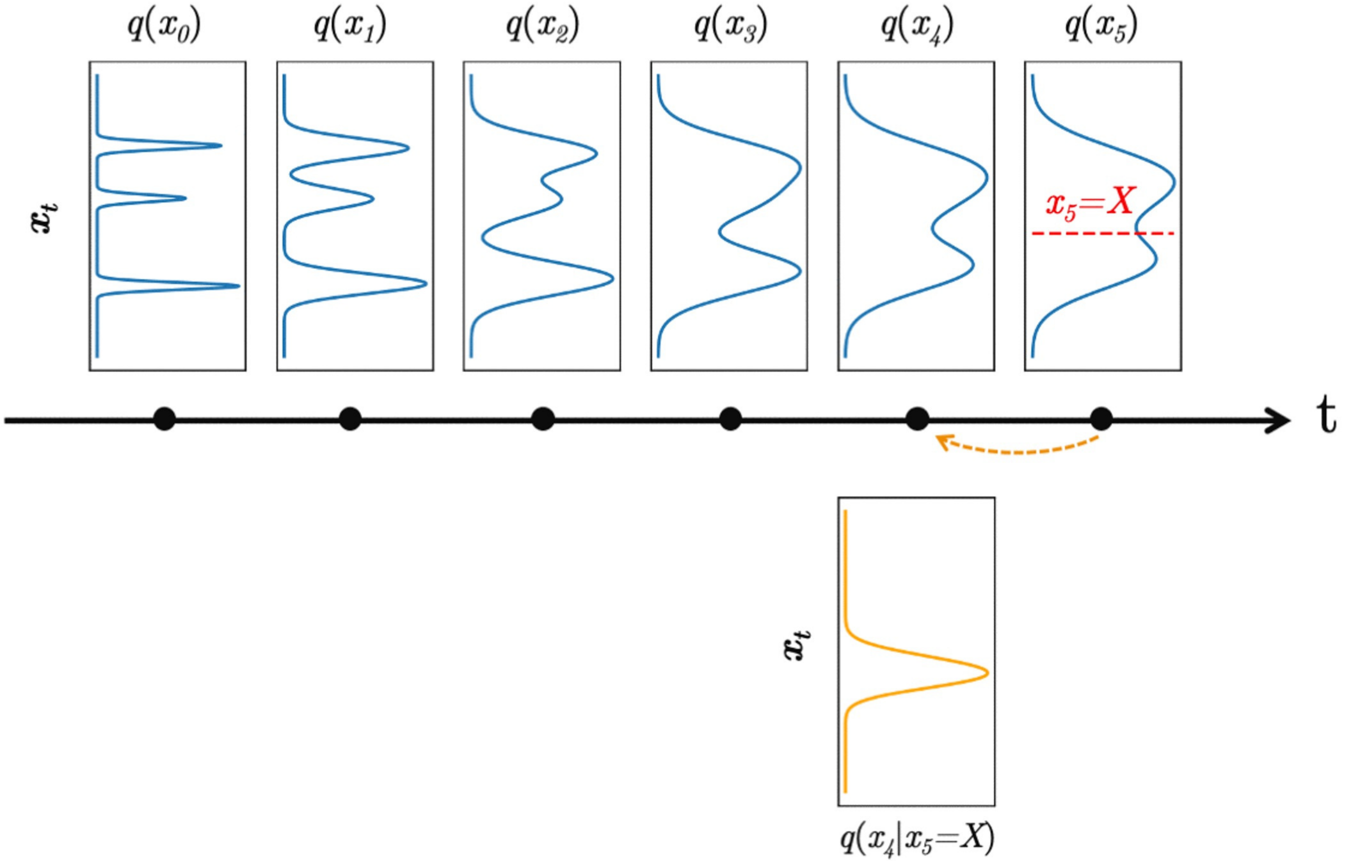


- The goal of a diffusion model is to **learn** the reverse *denoising* process to iteratively **undo** the forward process
- In this way, the reverse process appears as if it is generating new data from random noise!

Diffused
Data Distribution

Diffused
Data Distribution

True Denoising
Distribution



What should the distribution look like?

Turns out that for small enough forward steps, i.e. $\{\beta_t \in (0,1)\}_{t=1}^{T}$

the reverse process step $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ can be estimated as a Gaussian distribution too

Therefore, we can parametrize the *learned* reverse process as

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

In practice, Σ is just the identify matrix, so we only need to learn the mean of the distribution

Preliminary objective

When we write out the loss function, we get something that looks like this:

$$
\begin{aligned}
L_{\mathrm{VLB}} &= L_T + L_{T-1} + \cdots + L_0 \\
\text{where } L_T &= D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T)) \\
\boxed{L_t = D_{\mathrm{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1} \\
L_0 &= -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)
\end{aligned}
$$

Middle Loss Term - Intuition

$$L_t = D_{\mathrm{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1$$

KL Divergence: measures distance between two distributions

→ If high, very dissimilar distributions

→ If low, very similar distributions

Goal: drive this very low

Final Loss

Recall, our goal was to learn the following μ_θ (network that parameterizes the mean of the data distribution):

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

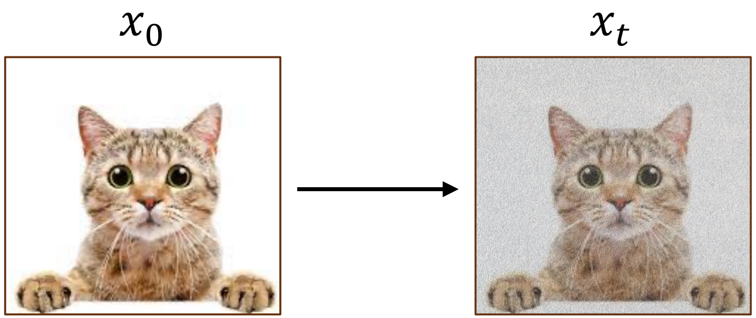
So we minimize:

$$\text{MSE}(\mu_\theta(x_t, t), x_{t-1})$$

How do we do this in practice?

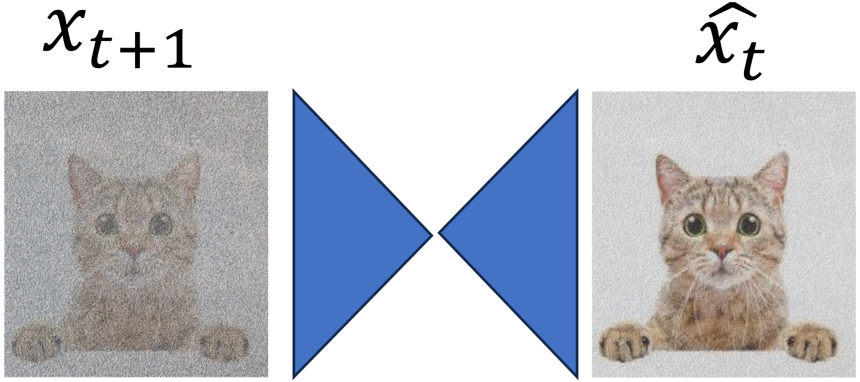
Step 1: Sample image from the dataset, generate noisy image using forward process

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0,\, (1-\bar{\alpha}_t)\mathbf{I}\right)$$



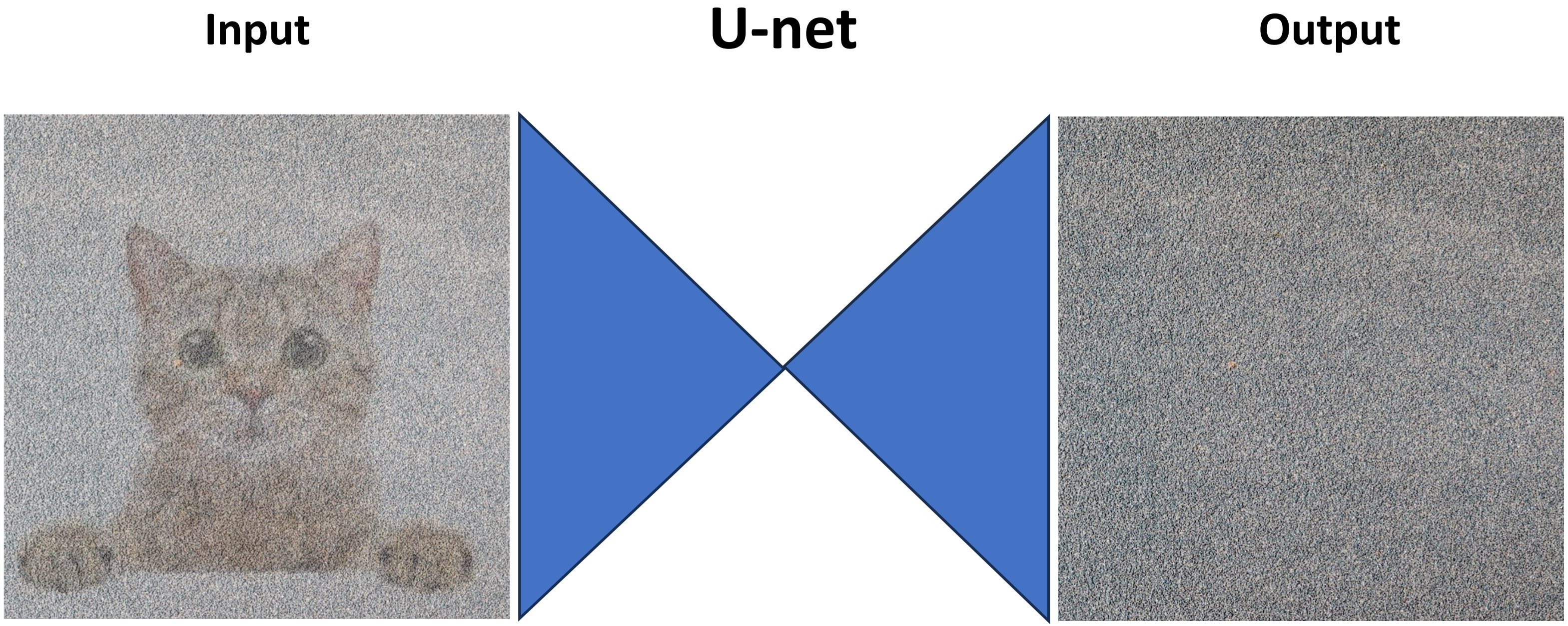
Step 2: Given noisy image, generate slightly noisier image

How do we do this in practice?



Loss: MSE($x_t, \widehat{x_t}$)

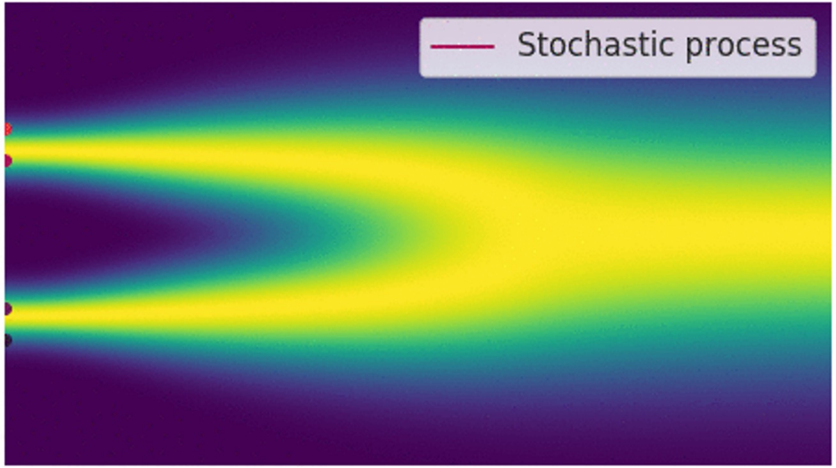
Neural Network that predicts noise



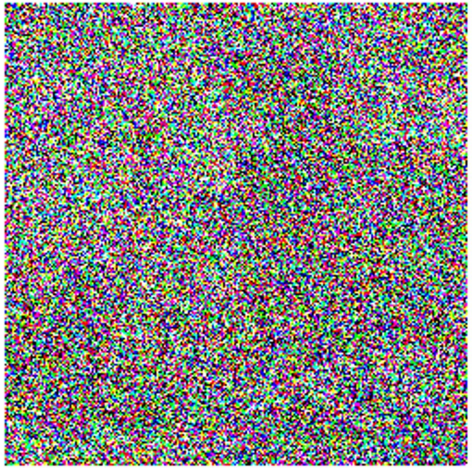
Training

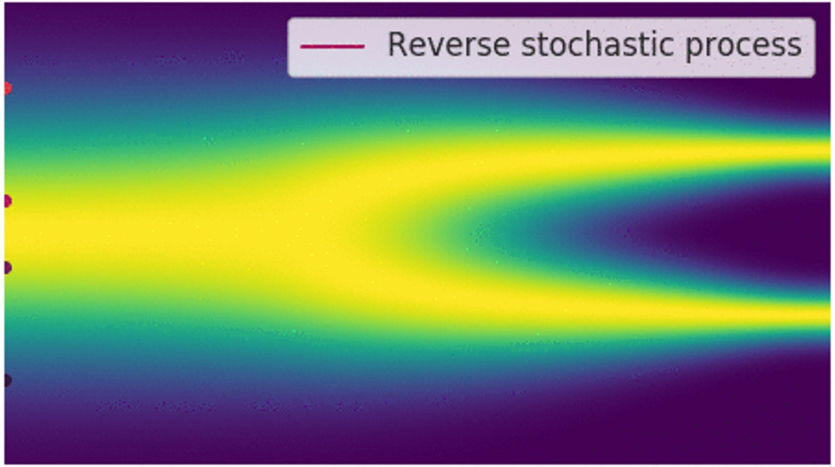
Algorithm 1 Training

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



Forward process: converting the image distribution to pure noise





Reverse process: sampling from the image distribution, starting with pure noise

Diffusion Models Beats GANs

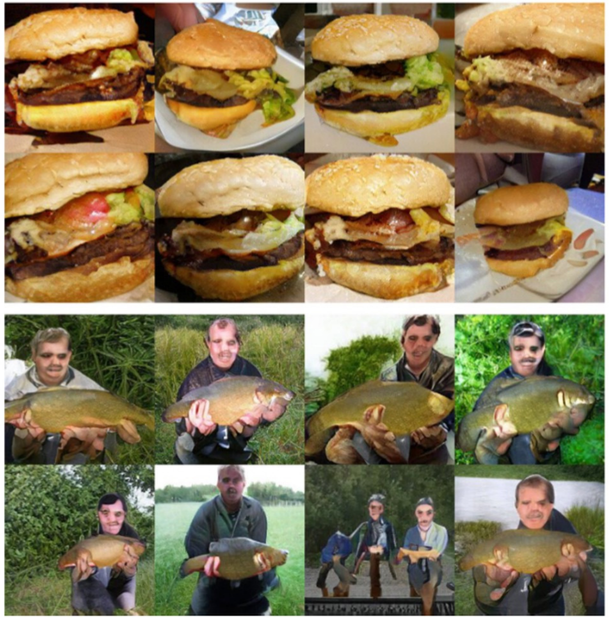
BigGAN

Diffusion

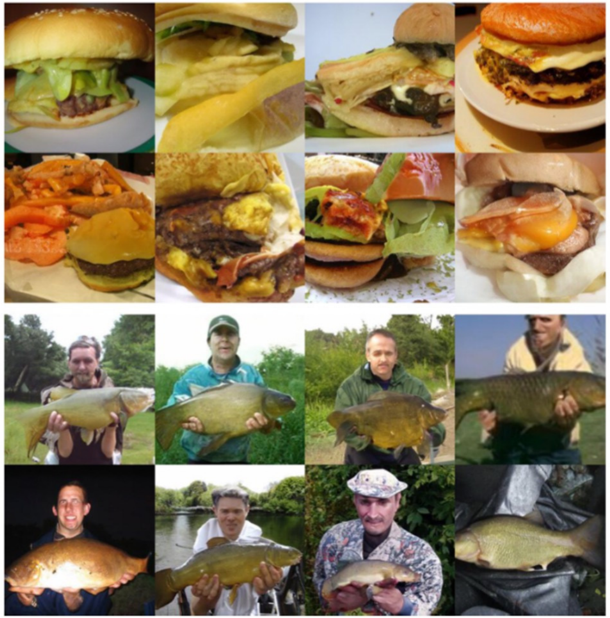
Training Set

Diffusion Models Beats GANs

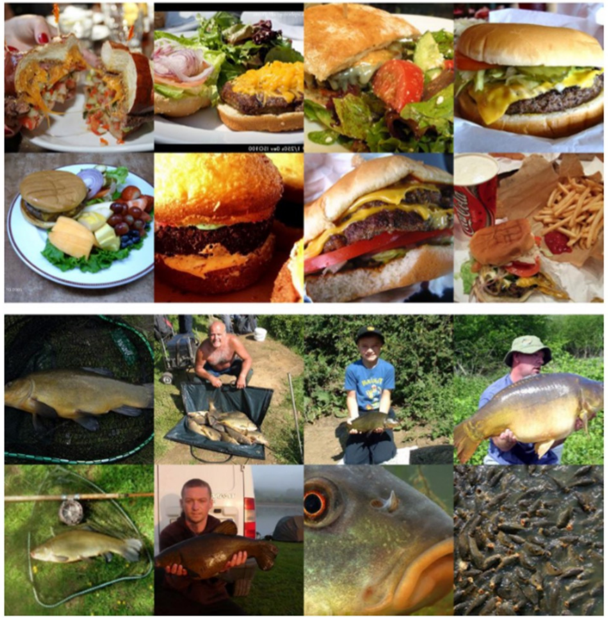
BigGAN



Diffusion

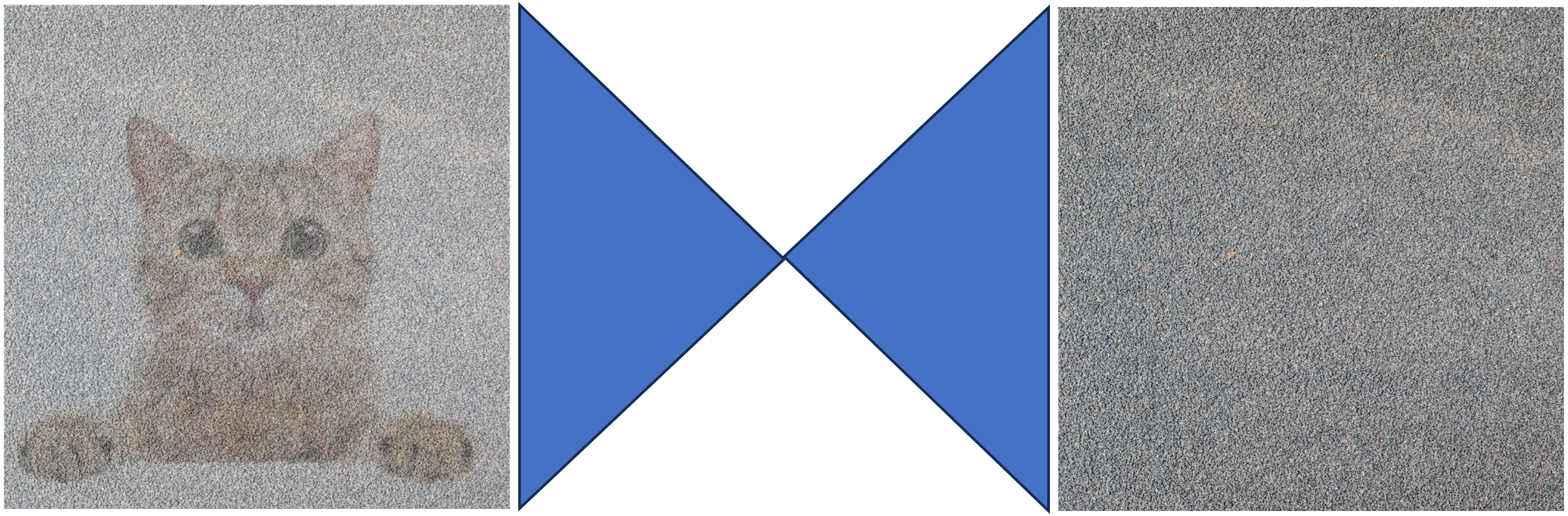


Training Set



U-net Problem

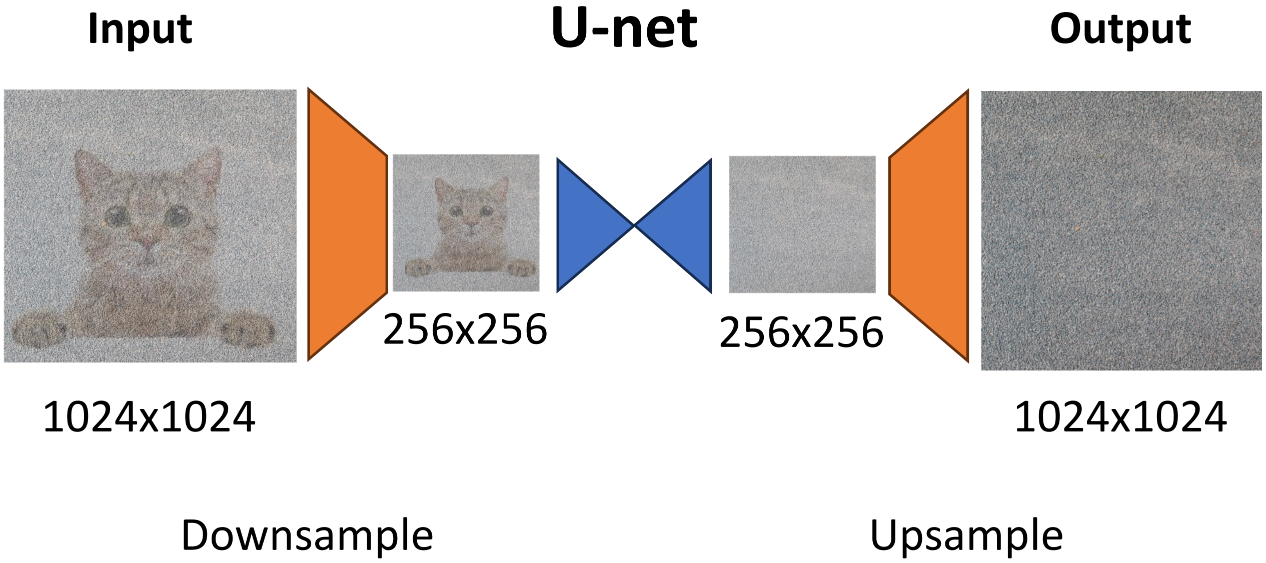
Input **U-net** **Output**



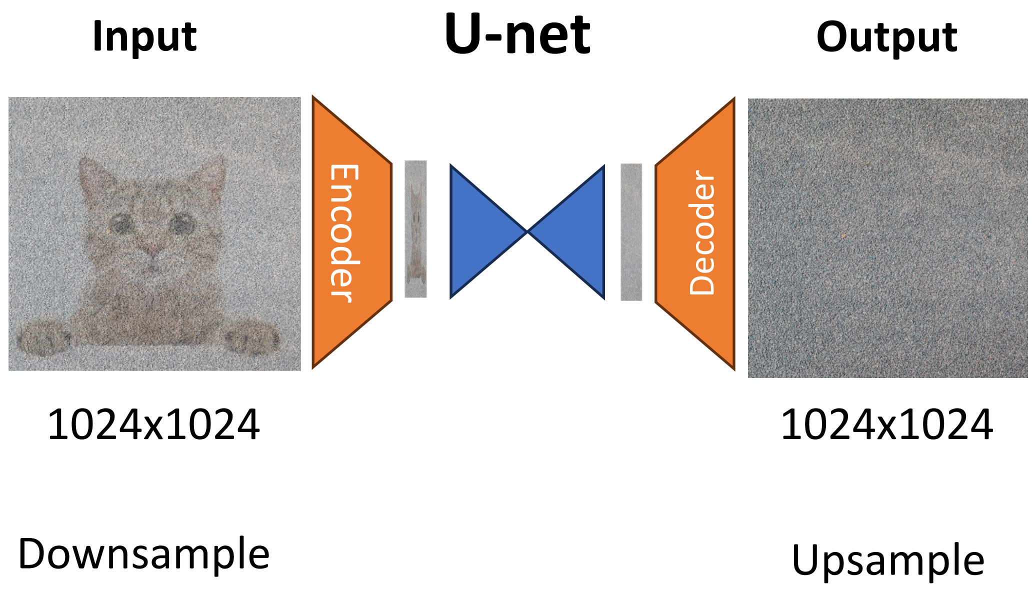
1024x1024 1024x1024

Problem: operating in the input space is very computationally expensive!

Option #1: Generate Low-Resolution + Upsample

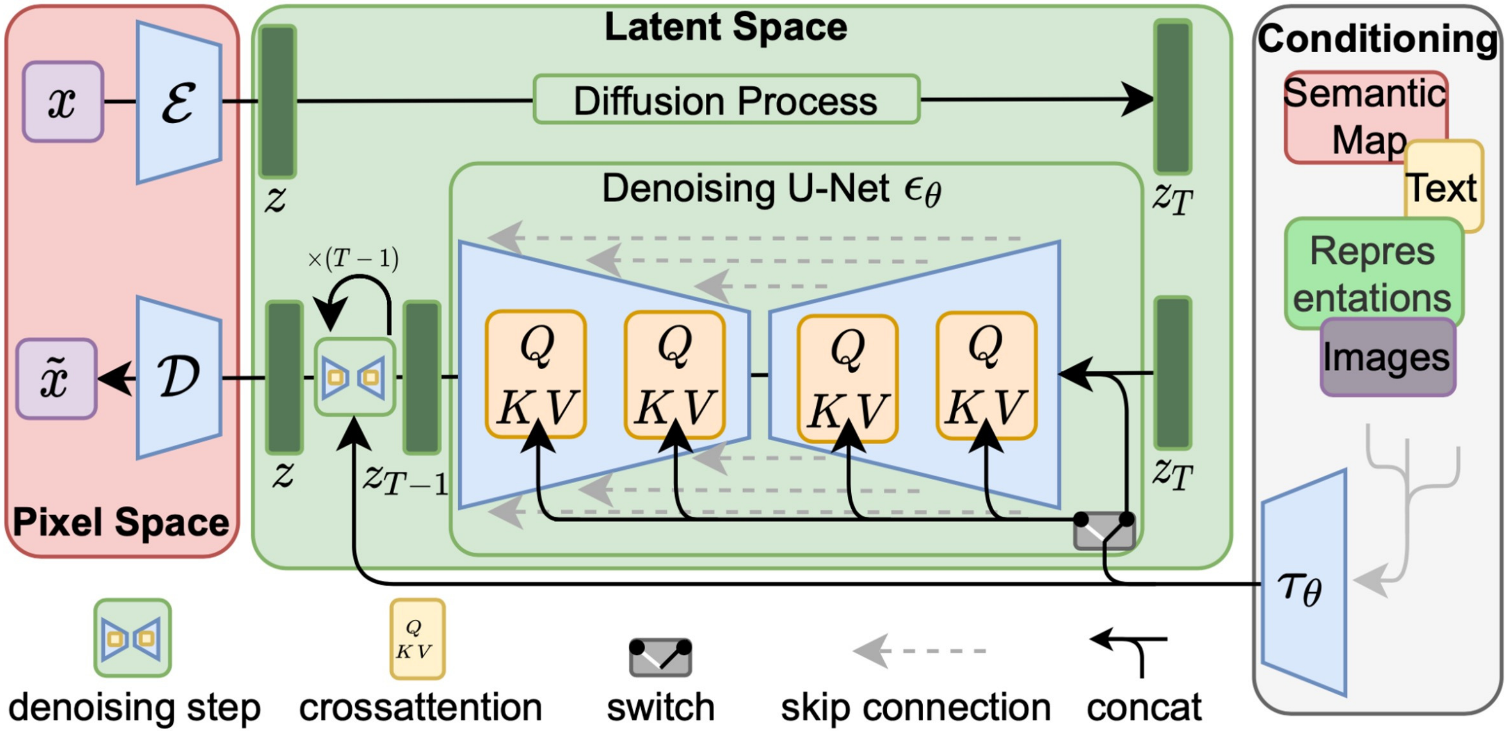


Option #2: Generate in Latent Space



Stable Diffusion

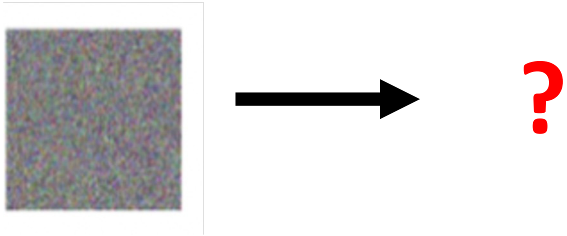
What's going on here?



Guided/Conditioned Diffusion

Lets say we train a diffusion model on images of cats and dogs:

If we start from random noise, and generate a new image, what will the model generate?



Leveraging Diffusion Models for Visual-Tactile Cross Generation

EECS 442 Team

Contents

1 Background: Tactile Sensor & Touch Images

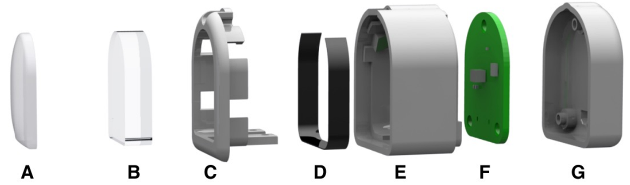
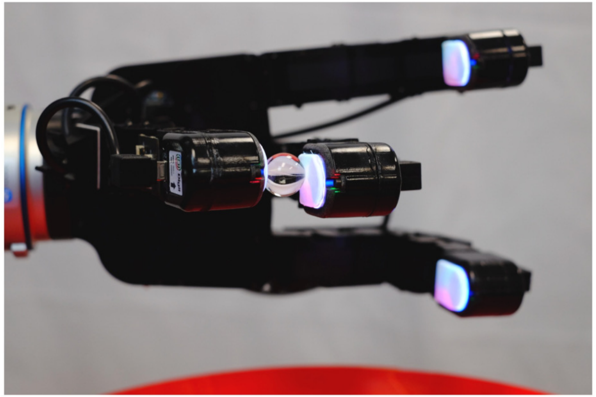
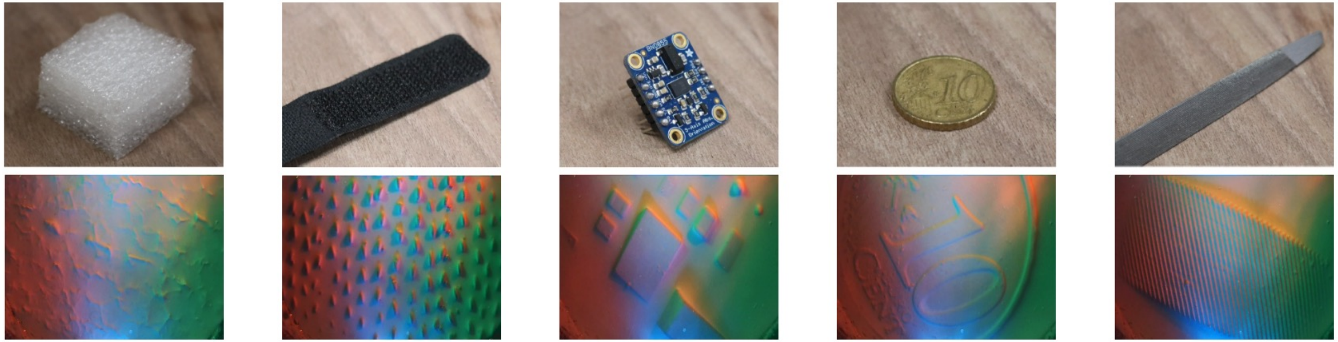
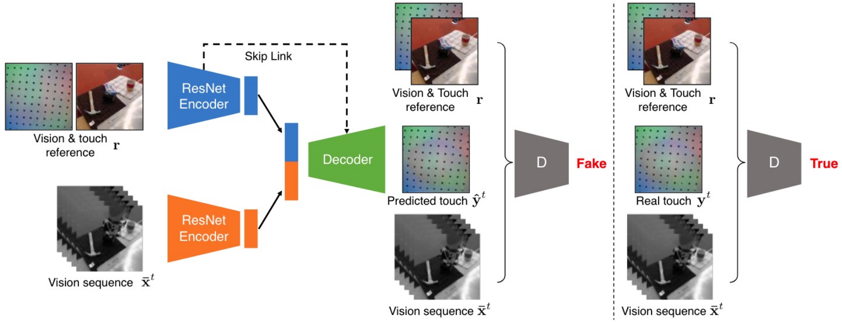


Figure 2: Exploded view of a single DIGIT sensor. A) elastomer, B) acrylic window, C) snap-fit holder, D) lighting PCB, E) plastic housing, F) camera PCB, G) back housing.

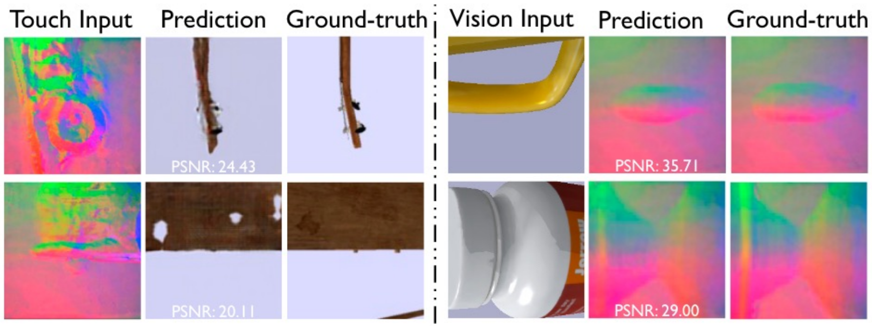




2 Previous Methods



VisGel [1]: GAN-based exocentric generation

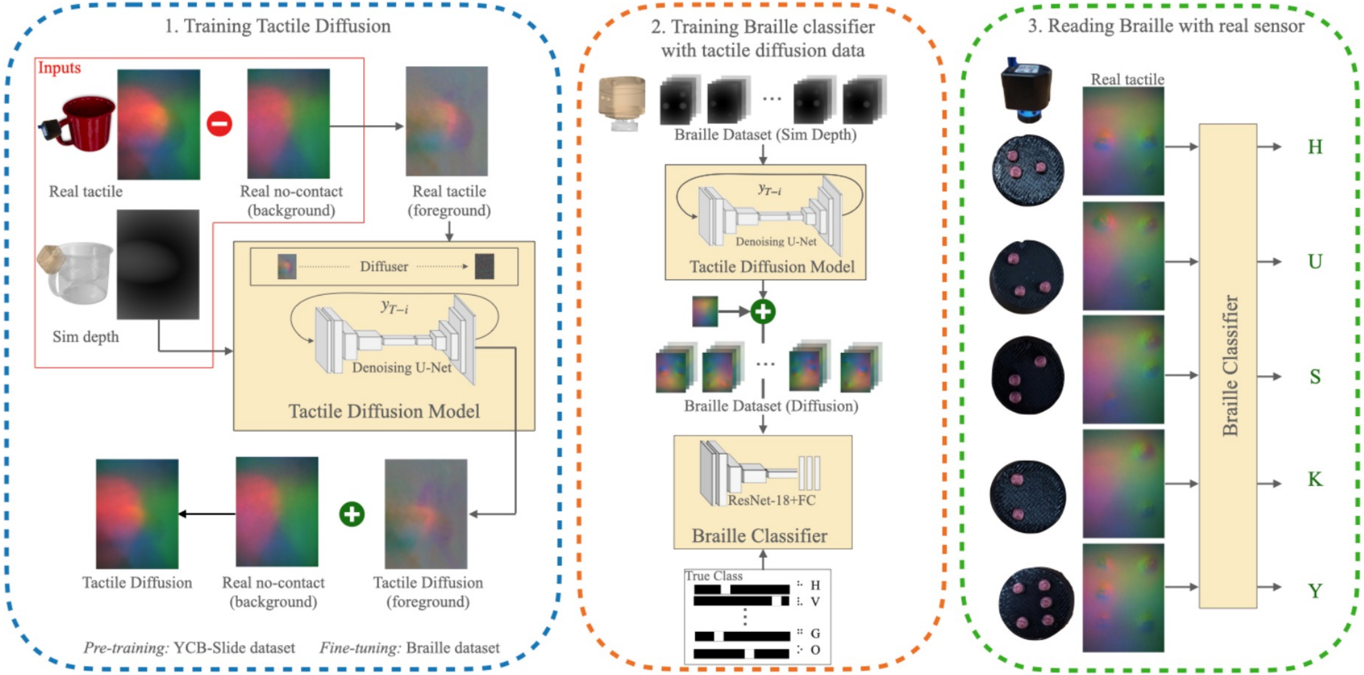


ObjectFolder [2]: GAN-based egocentric generation

[1]: Li, Yunzhu, Jun-Yan Zhu, Russ Tedrake, and Antonio Torralba. “Connecting Touch and Vision via Cross-Modal Prediction.” In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10609–18, 2019.

[2]: Gao, Ruohan, Yiming Dou, Hao Li, Tanmay Agarwal, Jeannette Bohg, Yunzhu Li, Li Fei-Fei, and Jiajun Wu. “The ObjectFolder Benchmark: Multisensory Learning with Neural and Real Objects.” arXiv, June 1, 2023. https://doi.org/10.48550/arXiv.2306.00956.

3 Learning to Read Braille (Vision-to-Touch)



Higuera, Carolina, Byron Boots, and Mustafa Mukadam. "Learning to Read Braille: Bridging the Tactile Reality Gap with Diffusion Models." arXiv, April 3, 2023. http://arxiv.org/abs/2304.01182.

3 Learning to Read Braille (Vision-to-Touch)

Data: Simulated local depth map & Real tactile images collected on YCB dataset

3.1 Training Tactile Diffusion

- Data:

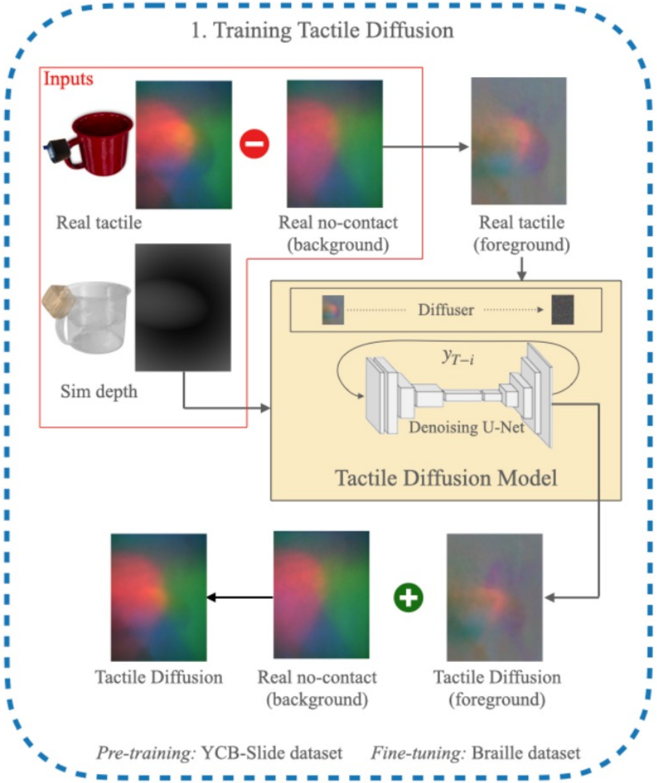
  Simulated local depth map & Real tactile images collected on YCB dataset

- Diffusion decoder:

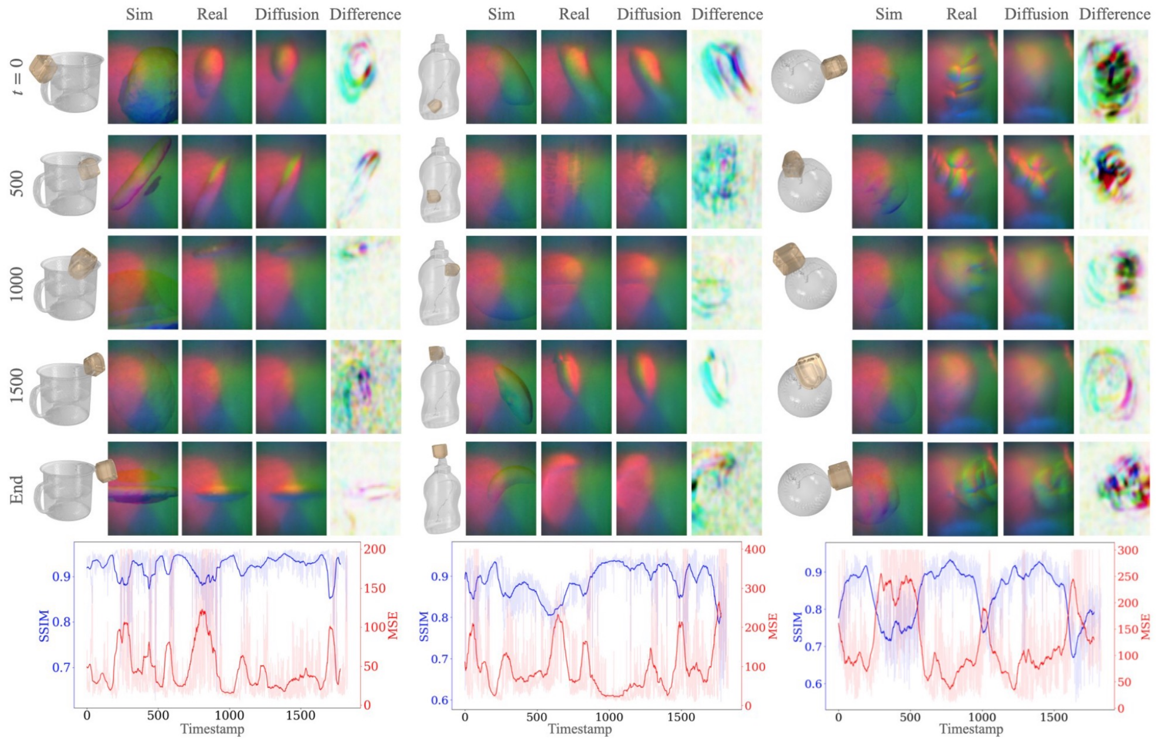
  Conditional U-Net backbone that takes depth map as input and renders colorful tactile images

- Evaluation:

  SSIM (structural similarity) & MSE (mean squared error)



3.1 Training Tactile Diffusion



Simulation, real, tactile diffusion results
(SSIM is generally above 0.80)

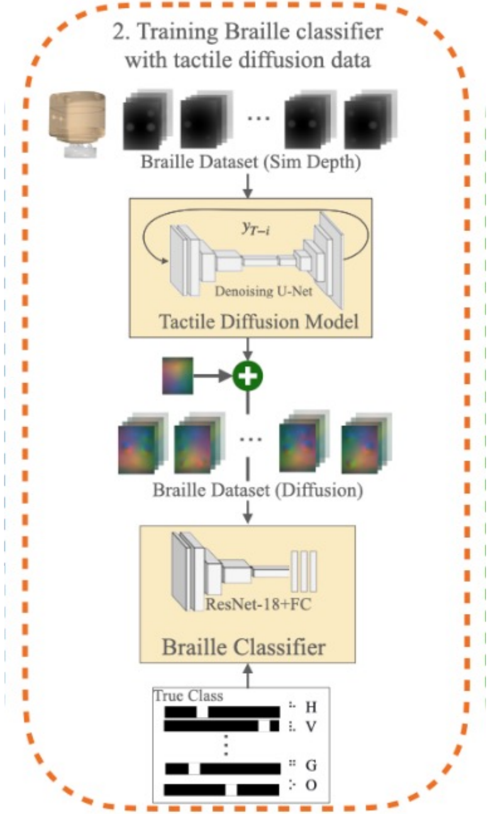
3.2 Training Braille Classifier in Simulator

- Sim2Real Transfer:

  Train a classifier to detect real-world braille letters with DIGIT sensor

- Comparison:

  Compare results from Sim / cGAN / Diffusion / Real data.

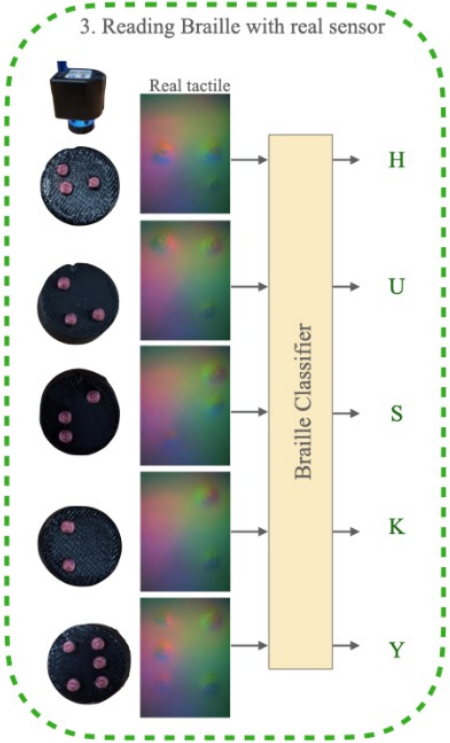


3.3 Reading Braille with Real-World Sensor

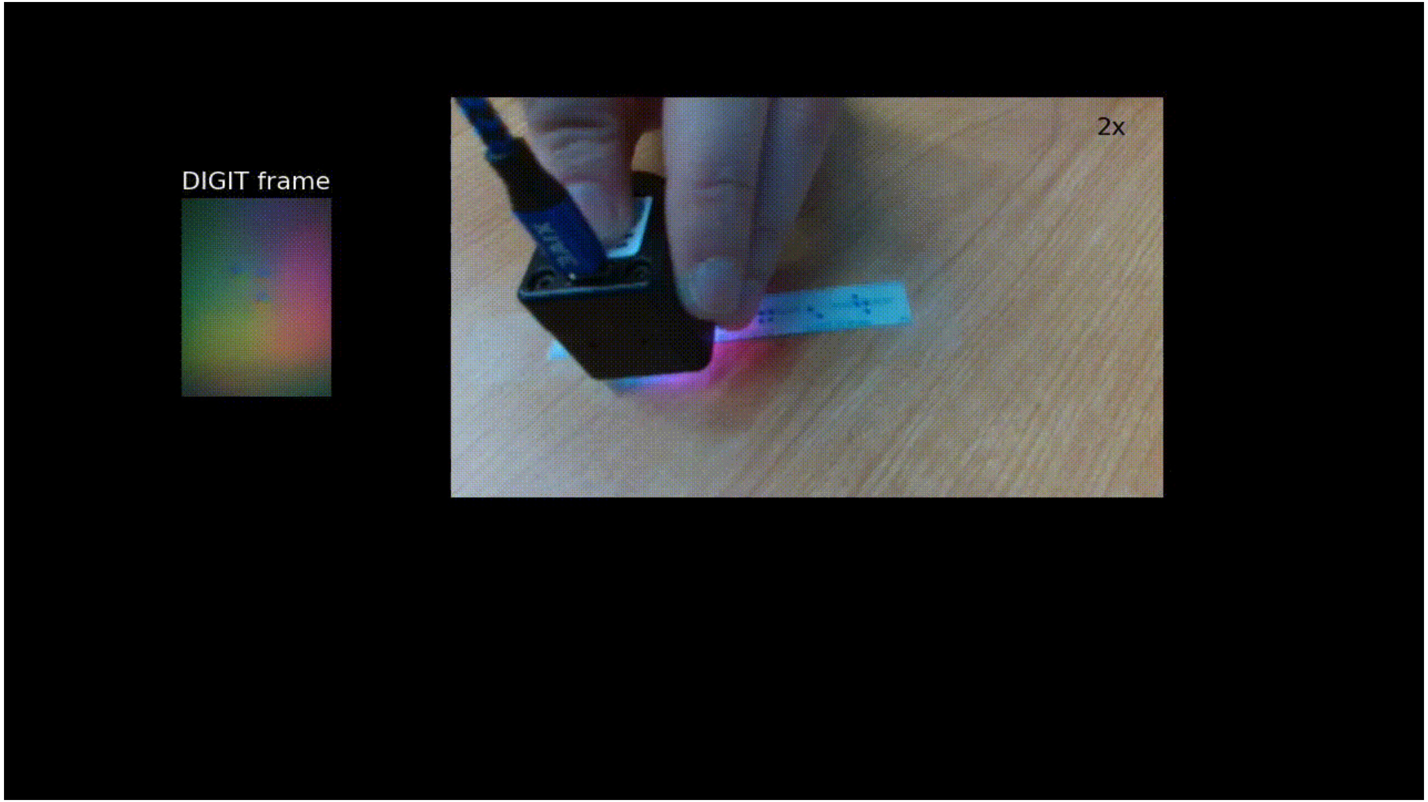
TABLE I: Metrics on braille classification task.

Training data source	% real data fine-tuning	Accuracy %	Precision	Recall
Sim	-	30.23	0.34	0.30
	20	64.99	0.71	0.65
	80	73.11	0.80	0.73
	100	73.95	**0.81**	0.74
Sim + data aug.	-	43.48	0.61	0.43
	100	73.23	0.76	0.73
cGAN	-	31.18	0.40	0.31
Tactile diffusion	-	**75.74**	0.79	**0.76**
Real	-	100.0	1.00	1.00

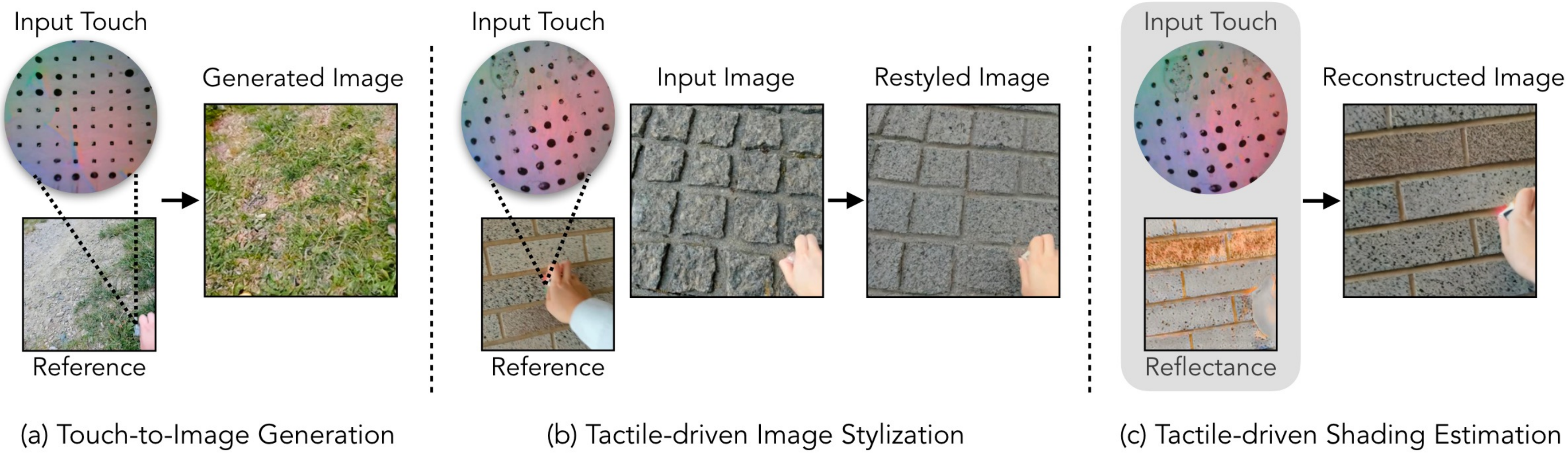
Training cGAN on 100% real, tactile diffusion on YCB-Slide + 20% real



3.3 Reading Braille with Real-World Sensor



4 Generating Visual Scenes from Touch (Touch-to-Vision)

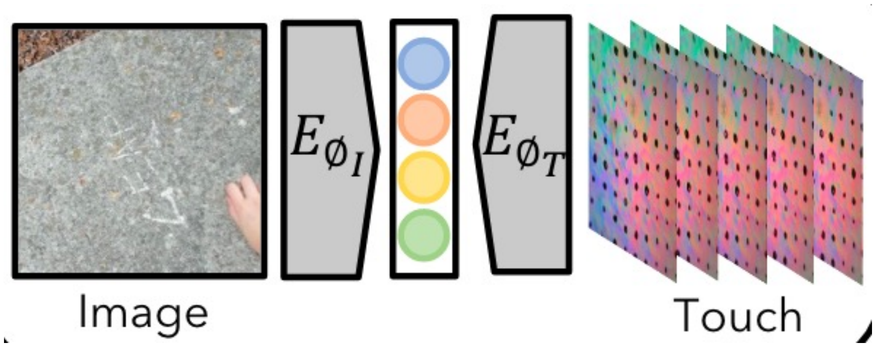


Yang, Fengyu, Jiacheng Zhang, and Andrew Owens. "Generating Visual Scenes from Touch." arXiv, September 26, 2023. https://doi.org/10.48550/arXiv.2309.15117.

4.1 Contrastive Visuo-tactile Pretraining (CVTP)

Given N visual-tactile image pairs, sample K from them and perform contrastive learning (mapping them into the uniform hidden space) using InfoNCE Loss:

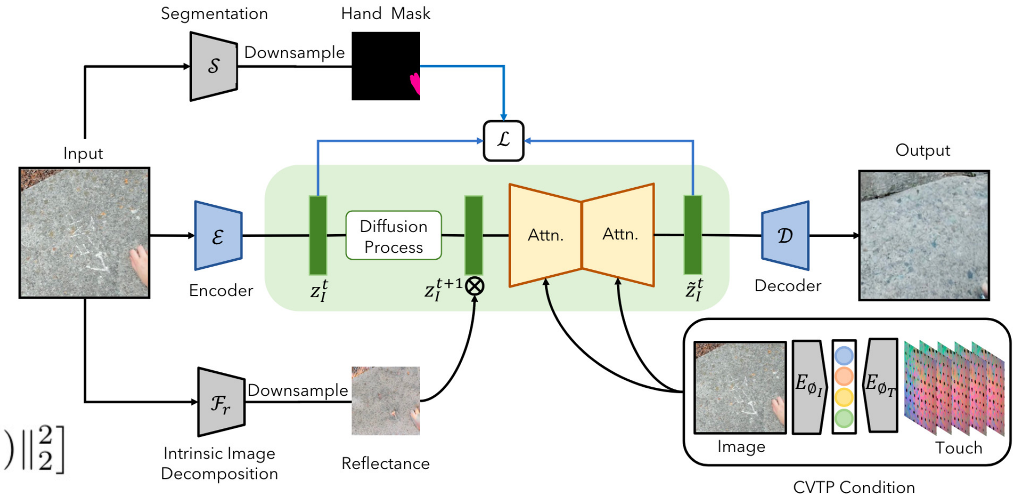
$$\mathcal{L}_i^{V_I,V_T} = -\log \frac{\exp(E_{\phi_I}(v_I^i) \cdot E_{\phi_T}(v_T^i)/\tau)}{\sum_{j=1}^{K} \exp(E_{\phi_I}(v_I^i) \cdot E_{\phi_T}(v_T^j)/\tau)} \quad (1)$$



4.2 Touch-conditioned Image Generation

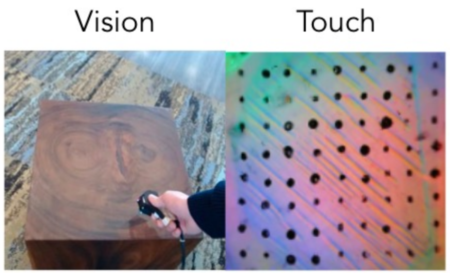
- Touch signal is represented by multi-frames from tactile sensor
- The diffusion process is conditioned on the touch signal. The loss function is:

$$L(\theta, \phi) = \mathbb{E}_{\mathbf{z}_I, \mathbf{c}, \epsilon, t}\left[\|\epsilon_t - \epsilon_\theta(\mathbf{z}_I^t, t, E_{\phi_T}(\mathbf{v}_T))\|_2^2\right]$$

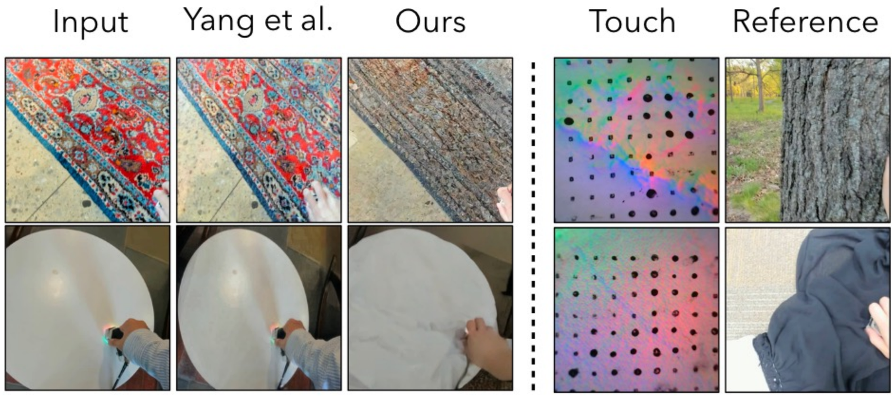


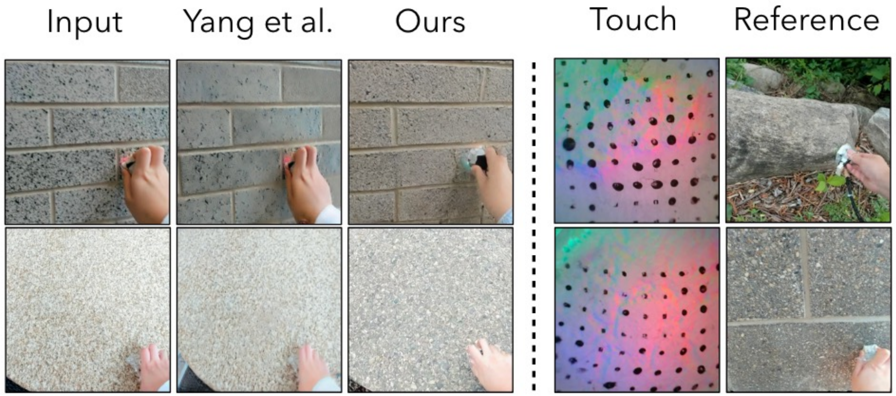
4.3 Qualitative Results: Tactile-driven Image Stylization

Touch and Go: An indoor-outdoor dataset with humans holding DIGIT sensor to touch objects



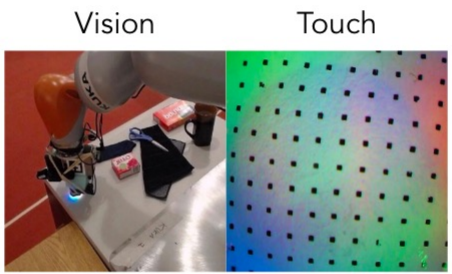
Touch and Go [64]



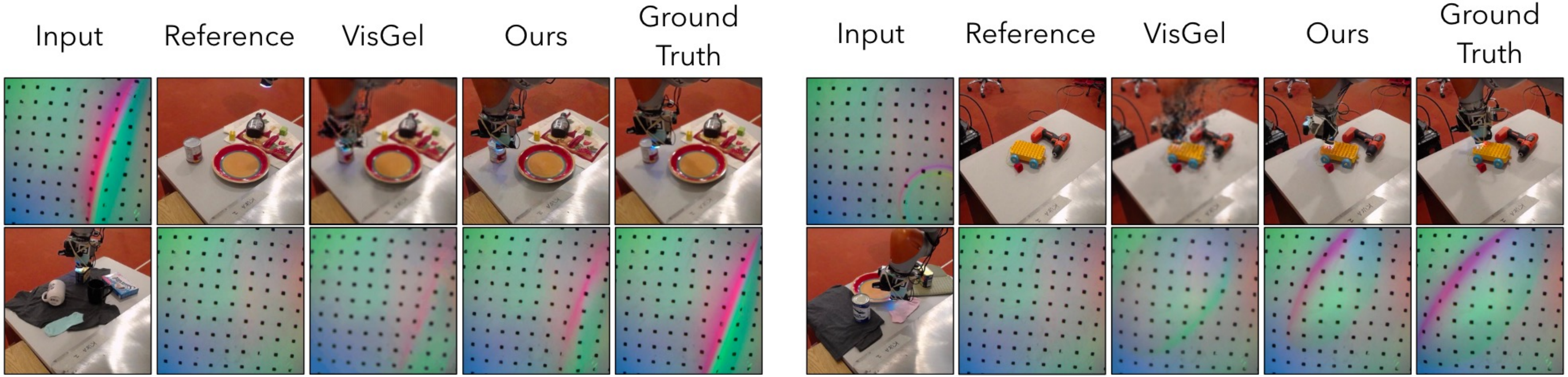


4.4 Qualitative Results: Visual-Tactile Cross Generation

VisGel: A dataset that collects paired touch videos and third-view robot arms.



VisGel [38]



Impressive Results

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.

Slide Credit: CVPR 2023 Diffusion Models Tutorial

How to Control Diffusion Models

Explicit Conditioning

Classifier Guidance

Classifier-Free Guidance

How to Control Diffusion Models

Explicit Conditioning

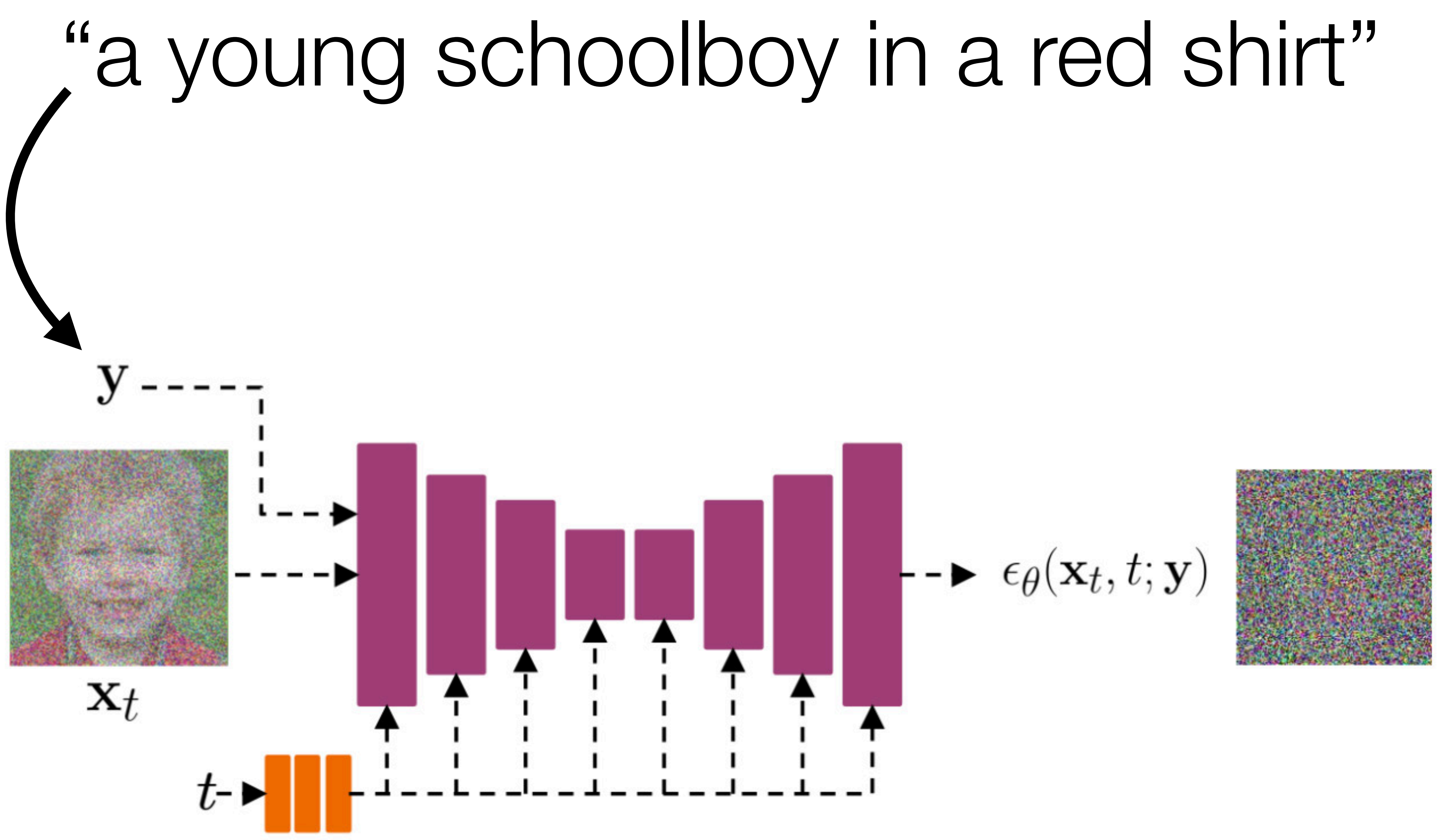
Classifier Guidance

Classifier-Free Guidance

Explicit Conditioning

“a young schoolboy in a red shirt”

Explicit Conditioning



Slide Credit: CVPR 2023 Diffusion Models Tutorial

Explicit Conditioning

How do we train this?

Explicit Conditioning

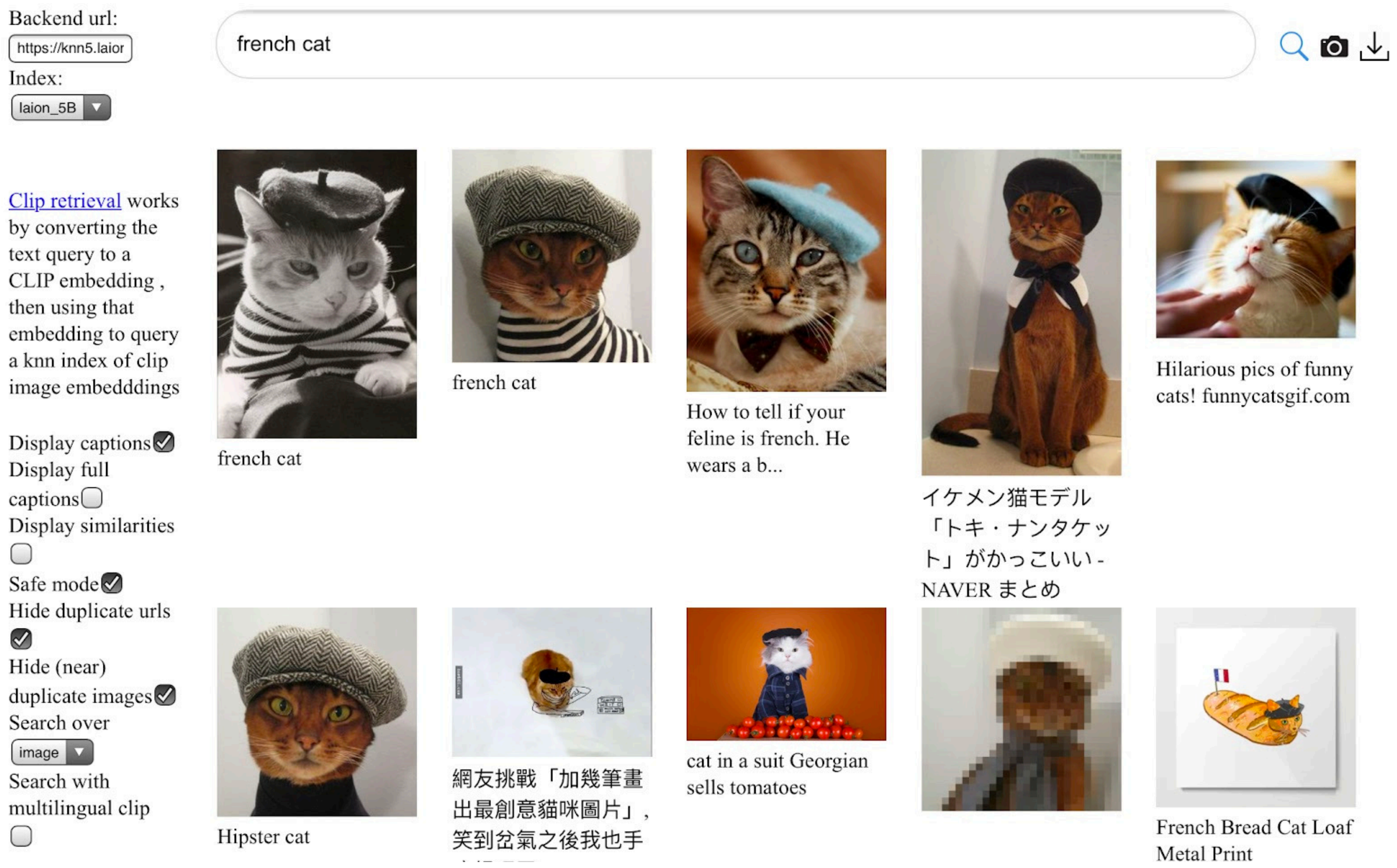
How do we train this?

Use an Image-Text dataset (for example, LAION 5B)

Explicit Conditioning

How do we train this?

Use an Image-Text dataset (for example, LAION 5B)



How to Control Diffusion Models

Explicit Conditioning

Classifier Guidance

Classifier-Free Guidance

Classifier Guidance

Diffusion goes from noise to real images step-by-step

Image Credit: CVPR 2023 Diffusion Models Tutorial

Classifier Guidance

Diffusion goes from noise to real images step-by-step

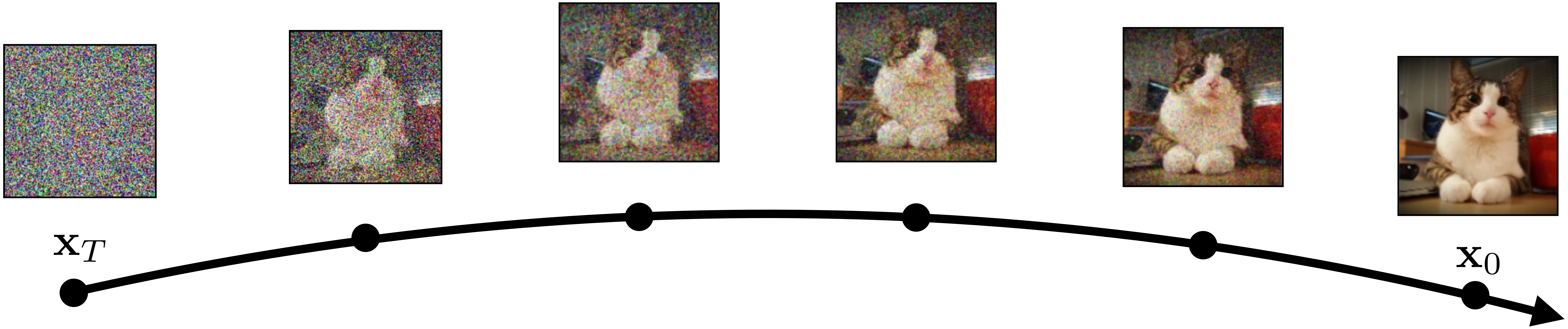
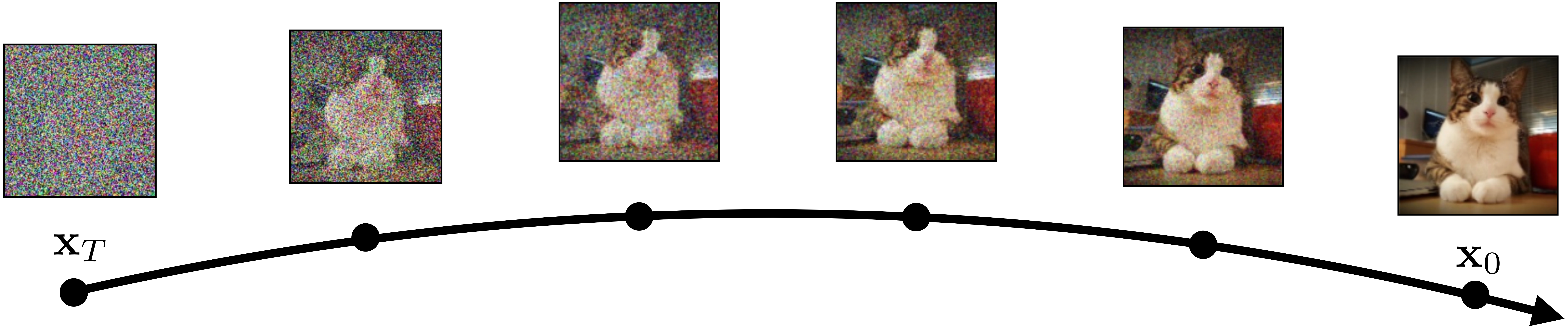


Image Credit: CVPR 2023 Diffusion Models Tutorial

Classifier Guidance

Diffusion goes from noise to real images step-by-step



Idea: Perturb the Denoising Trajectory

Image Credit: CVPR 2023 Diffusion Models Tutorial

Classifier Guidance

How do we get this perturbation?

Classifier Guidance

How do we get this perturbation?

Let's take an image classifier $p(x|y)$

Classifier Guidance

How do we get this perturbation?

Let's take an image classifier $p(x|y)$

And look at it's gradients (w.r.t. x) $\nabla_x \log p(x|y)$

Classifier Guidance

How do we get this perturbation?

Let’s take an image classifier $p(x|y)$

And look at it’s gradients (w.r.t. x) $\nabla_x \log p(x|y)$

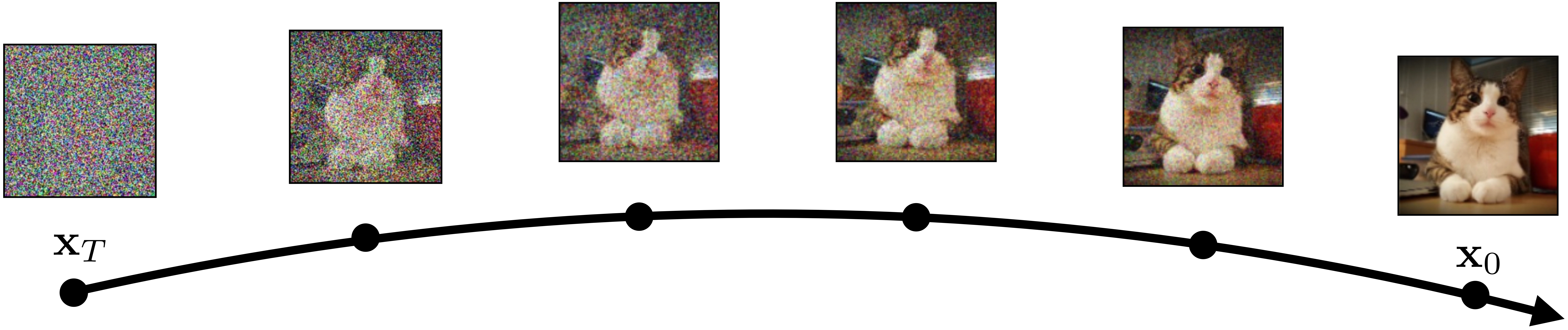
Intuitively: how to change x so it looks like a “y”

Classifier Guidance

Perturb using gradients of a classifier:

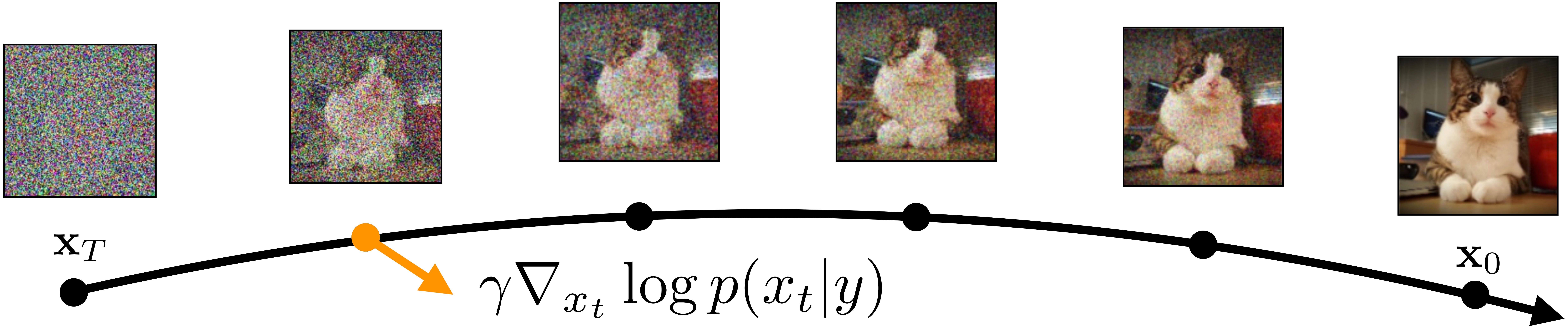
Classifier Guidance

Perturb using gradients of a classifier:



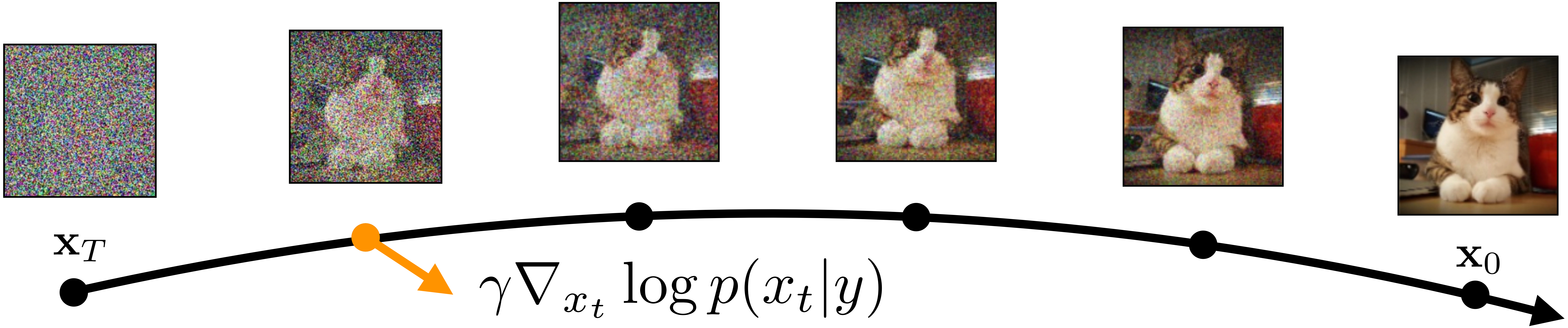
Classifier Guidance

Perturb using gradients of a classifier:



Classifier Guidance

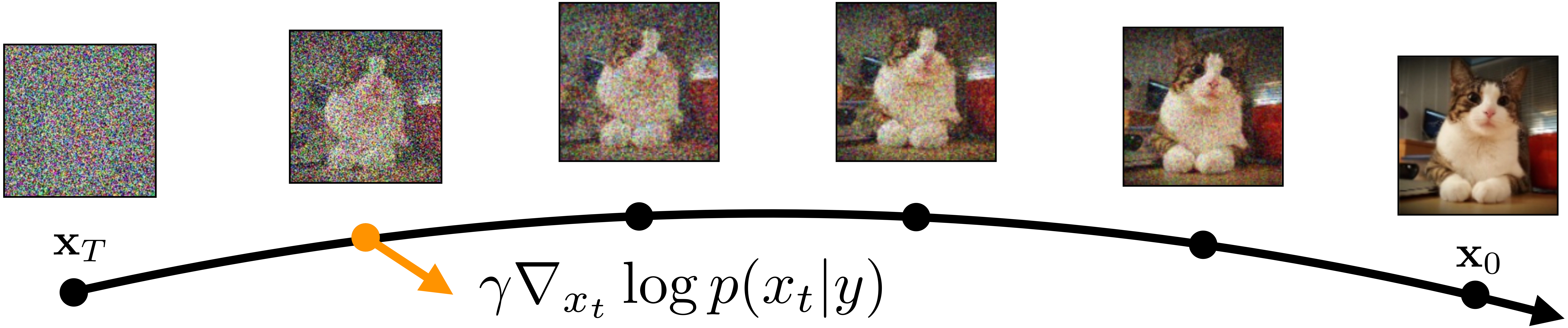
Perturb using gradients of a classifier:



$$\boxed{\tilde{\epsilon}_t(x_t, t, y) = \epsilon_\theta(x_t, t) - \gamma\nabla_{x_t}\log p(x_t|y)}$$

Classifier Guidance

Perturb using gradients of a classifier:



$$\boxed{\tilde{\epsilon}_t(x_t, t, y) = \epsilon_\theta(x_t, t) - \gamma \nabla_{x_t} \log p(x_t|y)}$$

There’s a small problem though…

Classifier Guidance

Problem: Classifier isn’t trained on noisy images!

Classifier Guidance

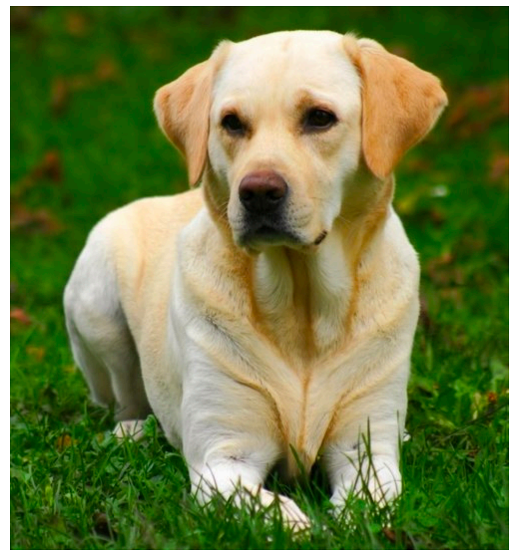
Problem: Classifier isn't trained on noisy images!

Solution: Finetune the classifier on noisy images

Classifier Guidance

Problem: Classifier isn't trained on noisy images!

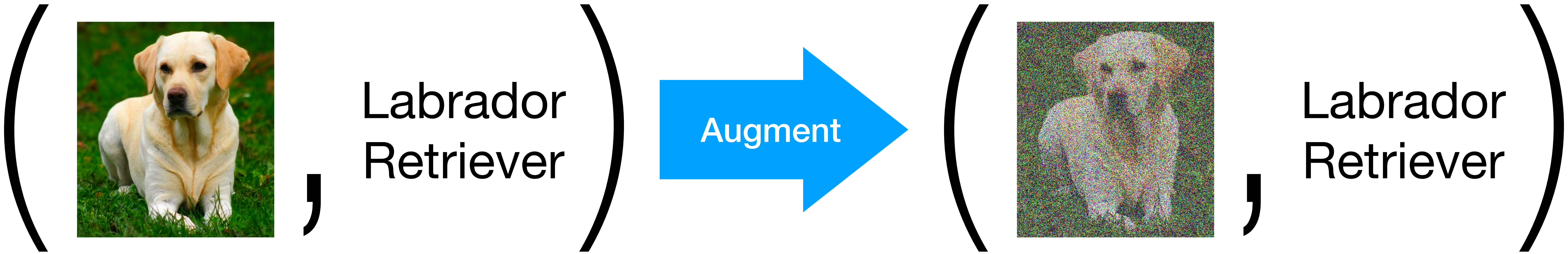
Solution: Finetune the classifier on noisy images

( , Labrador Retriever)

Classifier Guidance

Problem: Classifier isn’t trained on noisy images!

Solution: Finetune the classifier on noisy images



Classifier Guidance

Guidance Weight 1.0

Guidance Weight 10.0

Dhariwal and Nichol, “Diffusion Models Beat GANs on Image Synthesis”

Problems with Classifier Guidance

Problems with Classifier Guidance

- Need to fine-tune or re-train a classifier on **noisy** data

Problems with Classifier Guidance

- Need to fine-tune or re-train a classifier on **noisy** data
- Need a pre-trained classification model

Problems with Classifier Guidance

- Need to fine-tune or re-train a classifier on **noisy** data
- Need a pre-trained classification model
    - What if we want to use *any* text prompt as input?

Problems with Classifier Guidance

- Need to fine-tune or re-train a classifier on **noisy** data
- Need a pre-trained classification model
  - What if we want to use *any* text prompt as input?
- Classifier gradients are poor. They can suffer from “shortcuts”

How to Control Diffusion Models

Explicit Conditioning

Classifier Guidance

Classifier-Free Guidance

Classifier Free Guidance

Idea: Use the diffusion model itself to get perturbations for guidance

Classifier Free Guidance

Idea: Use the diffusion model itself to get perturbations for guidance

Train an explicitly conditioned diffusion model: $\epsilon_\theta(x_t, t, y)$

Classifier Free Guidance

Idea: Use the diffusion model itself to get perturbations for guidance

Train an explicitly conditioned diffusion model: $\epsilon_\theta(x_t, t, y)$

But also train it to be **unconditional**

Classifier Free Guidance

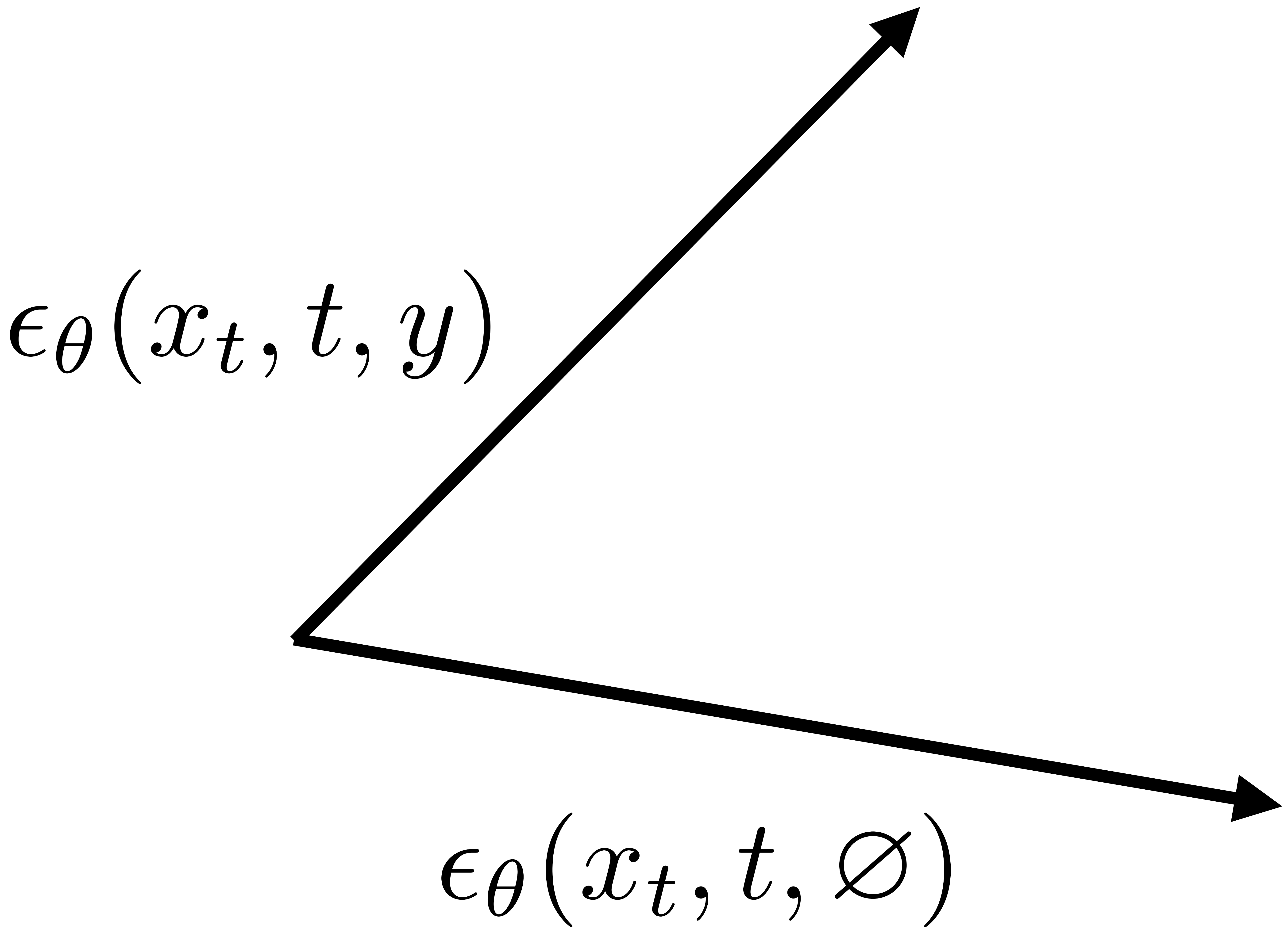
Idea: Use the diffusion model itself to get perturbations for guidance

Train an explicitly conditioned diffusion model: $\epsilon_\theta(x_t, t, y)$

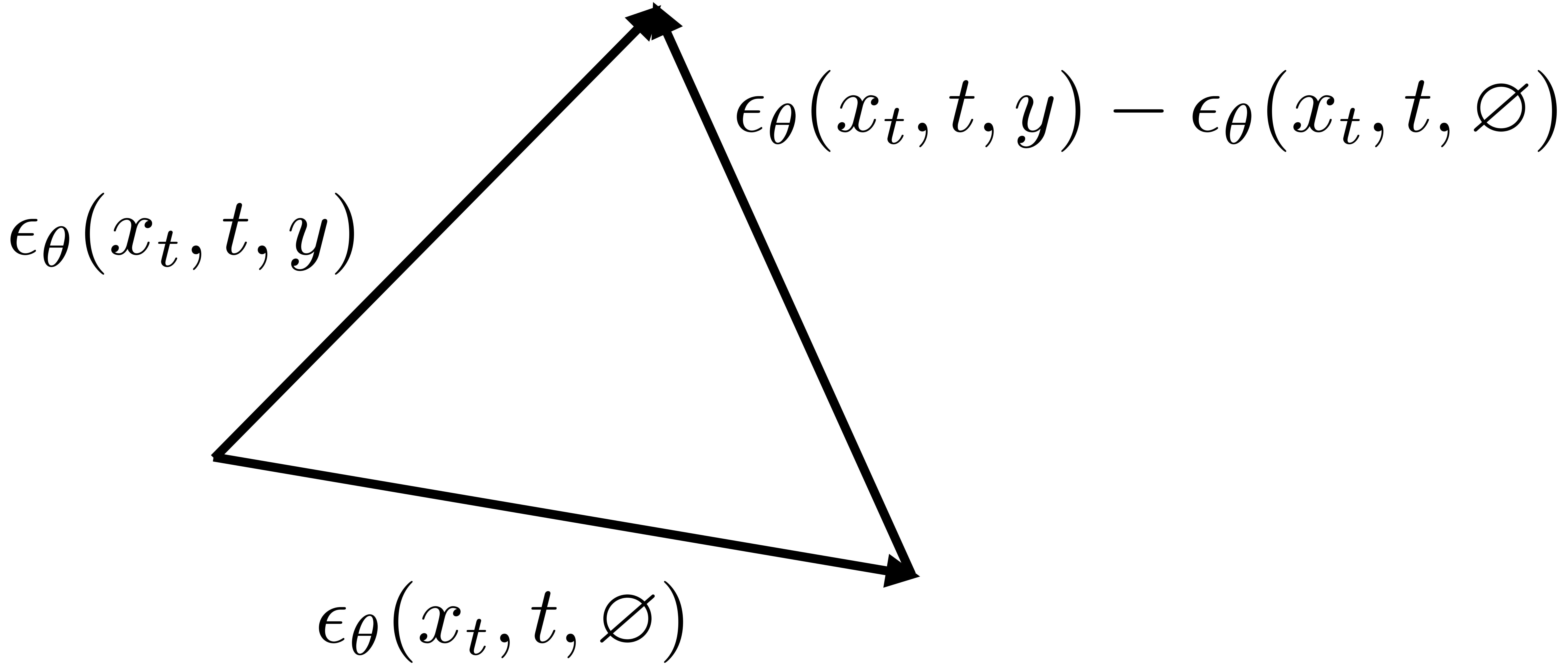
But also train it to be **unconditional**

We can do this with *conditioning dropout:* $\epsilon_\theta(x_t, t, \varnothing)$

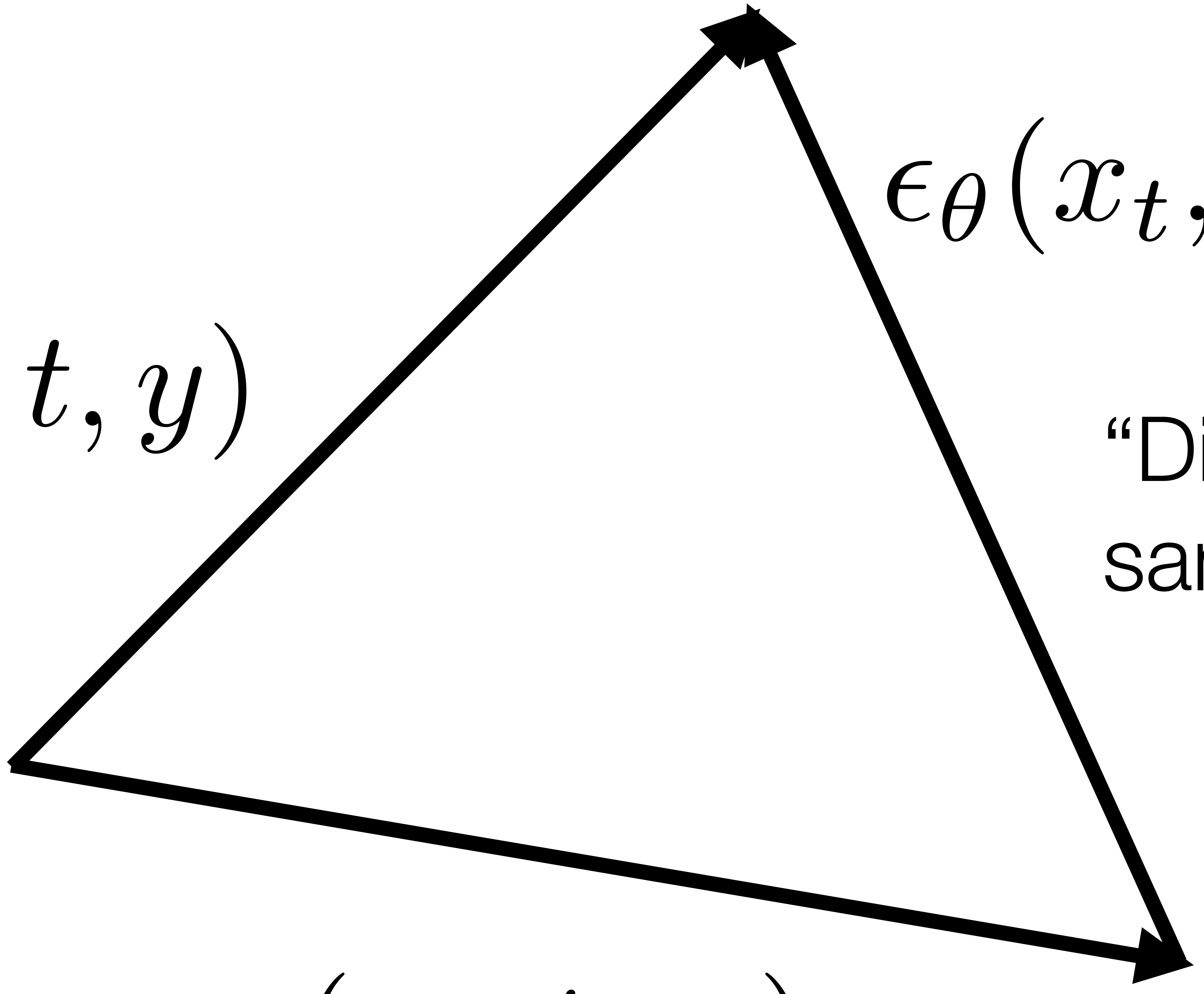
Classifier Free Guidance



Classifier Free Guidance

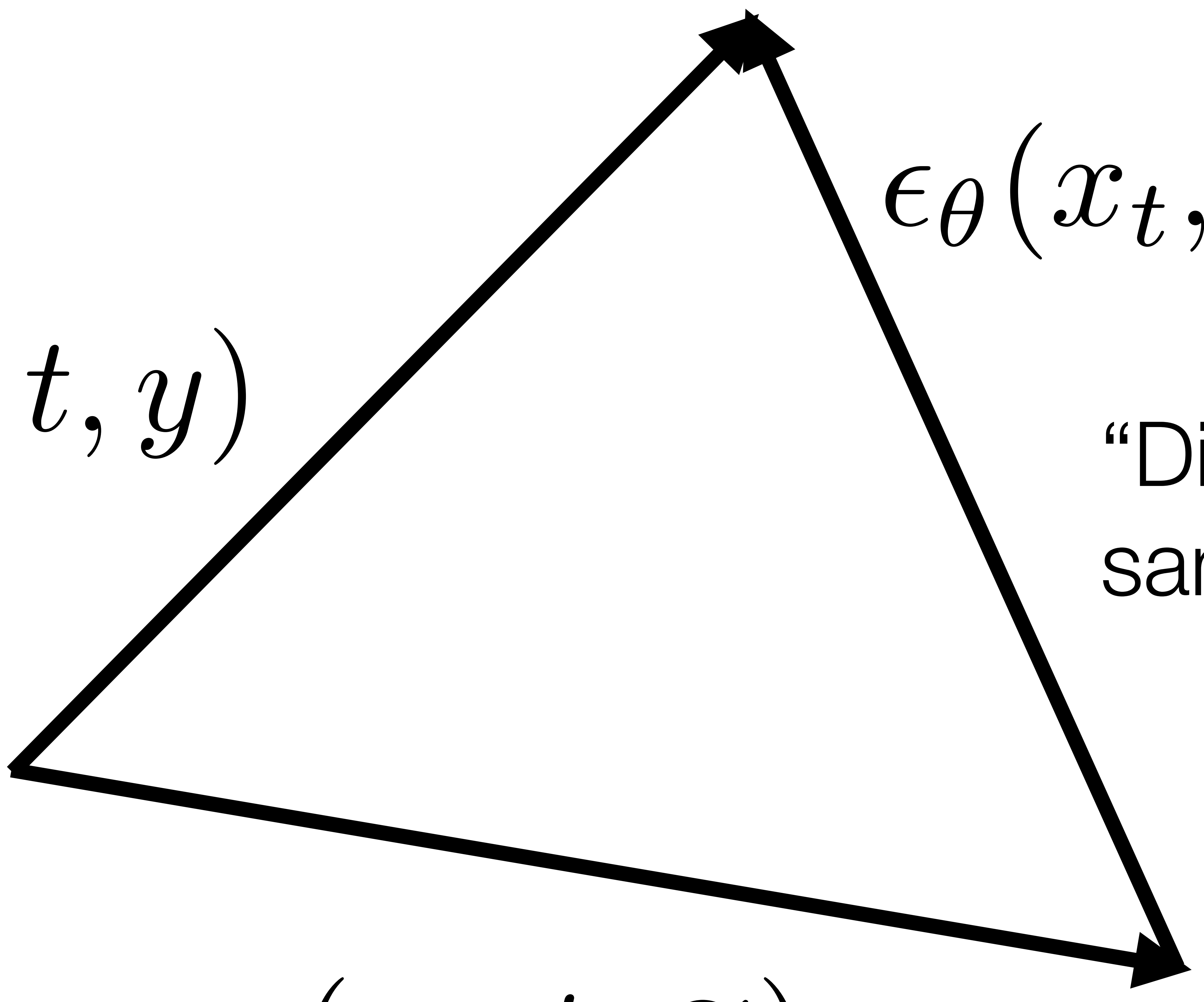


Classifier Free Guidance



“Direction” from unconditional sample to conditional sample

Classifier Free Guidance



“Direction” from unconditional sample to conditional sample

Use this as our guidance perturbation

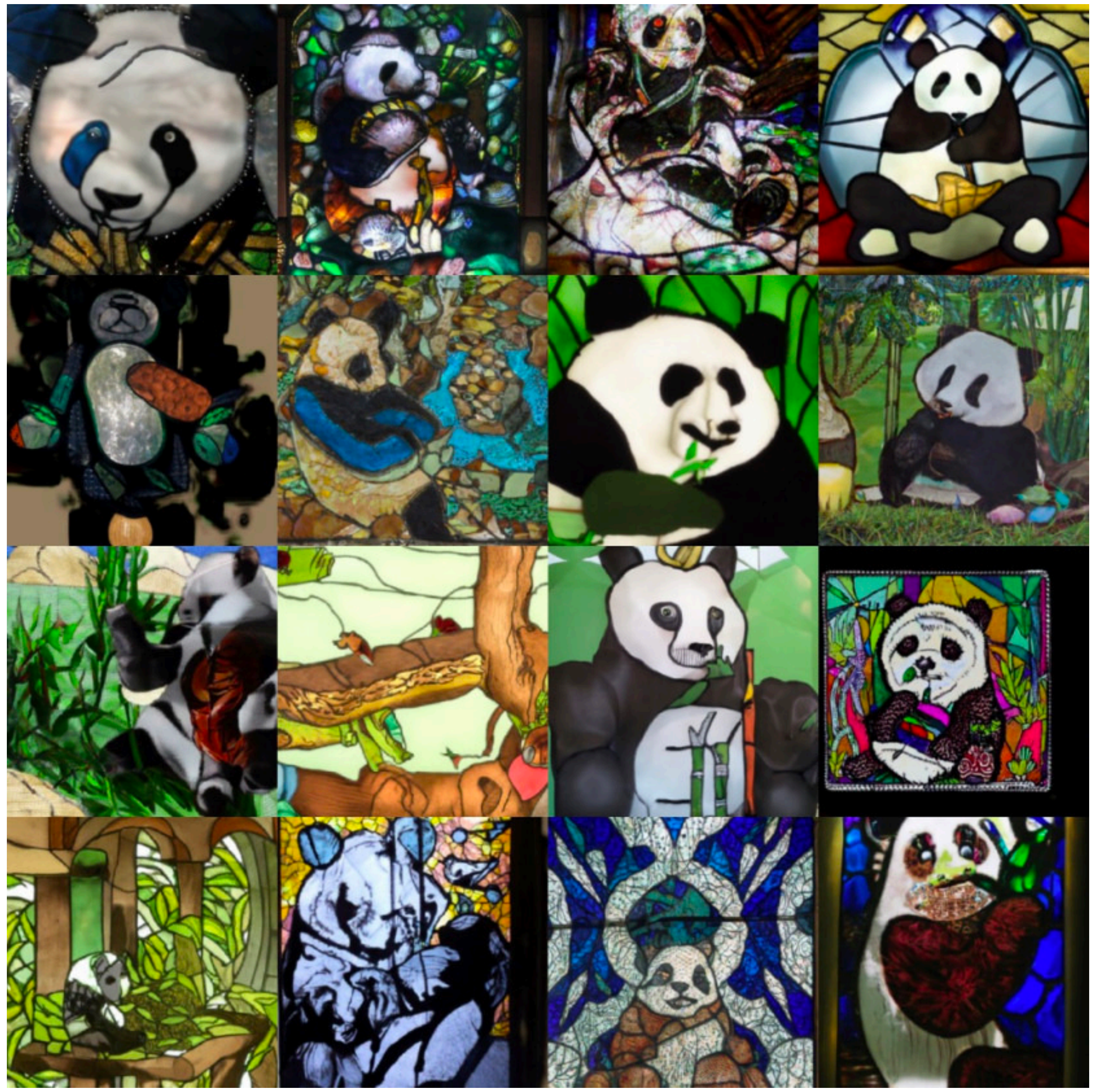
Classifier Free Guidance

Our new noise estimate will then be:

$$\tilde{\epsilon}(x_t, t, y) = \epsilon_\theta(x_t, t, \varnothing) + \gamma\underbrace{(\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t, \varnothing))}_{\text{“Direction” from unconditional to conditional}}$$

Classifier Free Guidance

"A stained glass window of a panda eating bamboo"



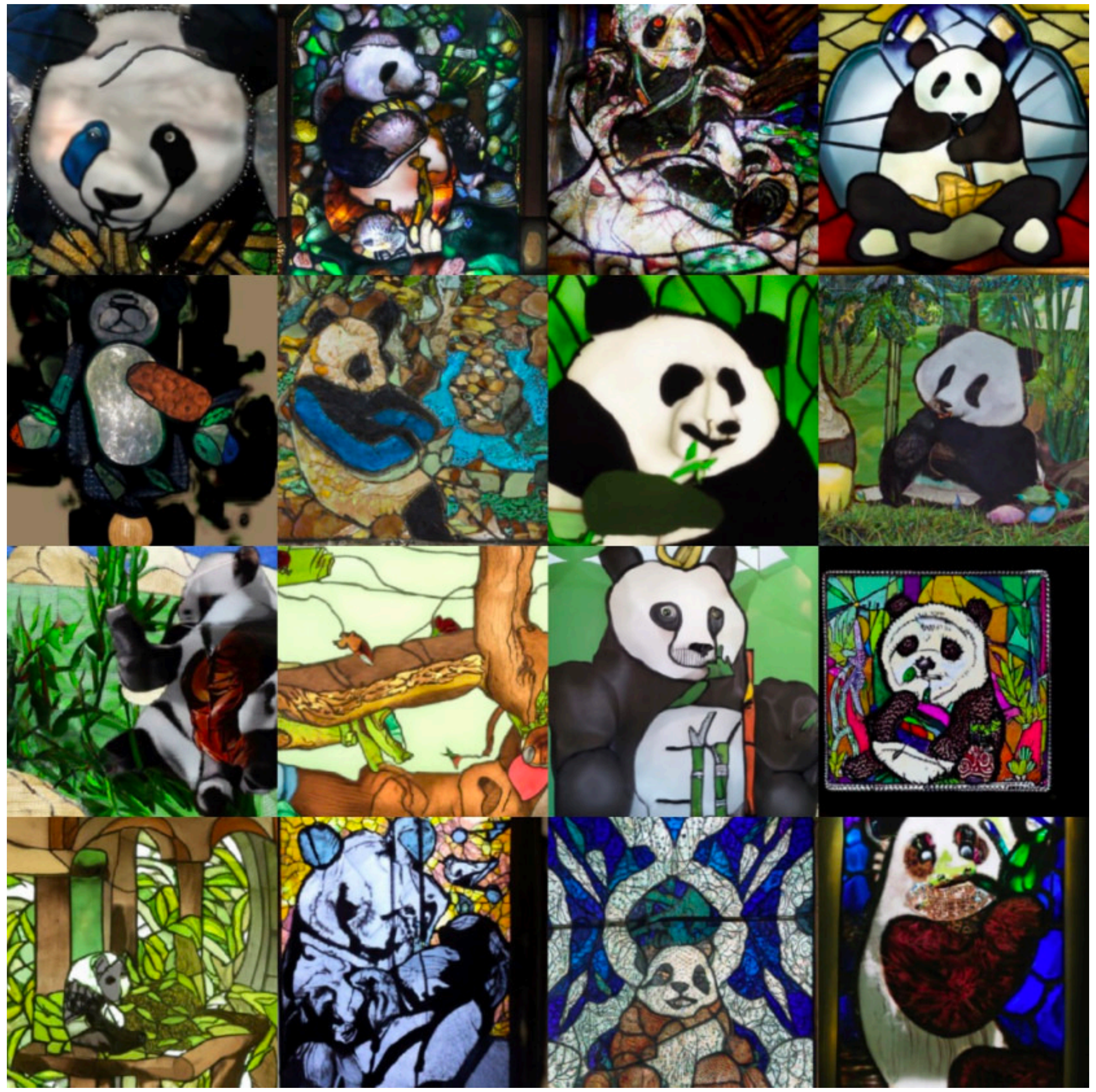
$\gamma = 1$

Equivalent to explicit conditioning.
No guidance

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

Classifier Free Guidance

“A stained glass window of a panda eating bamboo”



$\gamma = 1$

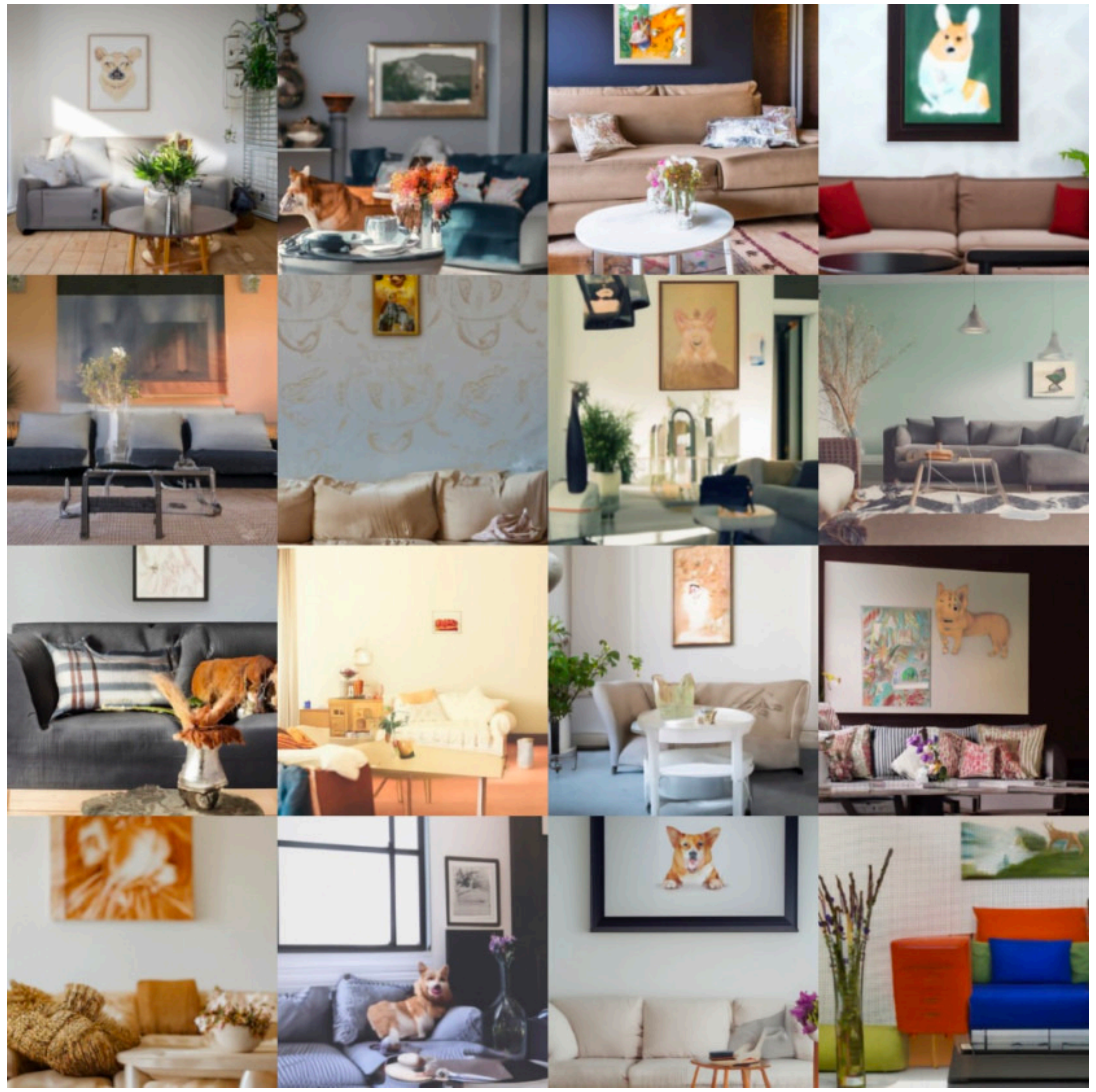
Equivalent to explicit conditioning.
No guidance

$\gamma = 3$

Nichol et al. “GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models”

Classifier Free Guidance

“A cozy living room with a painting of a corgi on the wall above a couch and a round coffee table in front of a couch and a vase of flowers on a coffee table.”



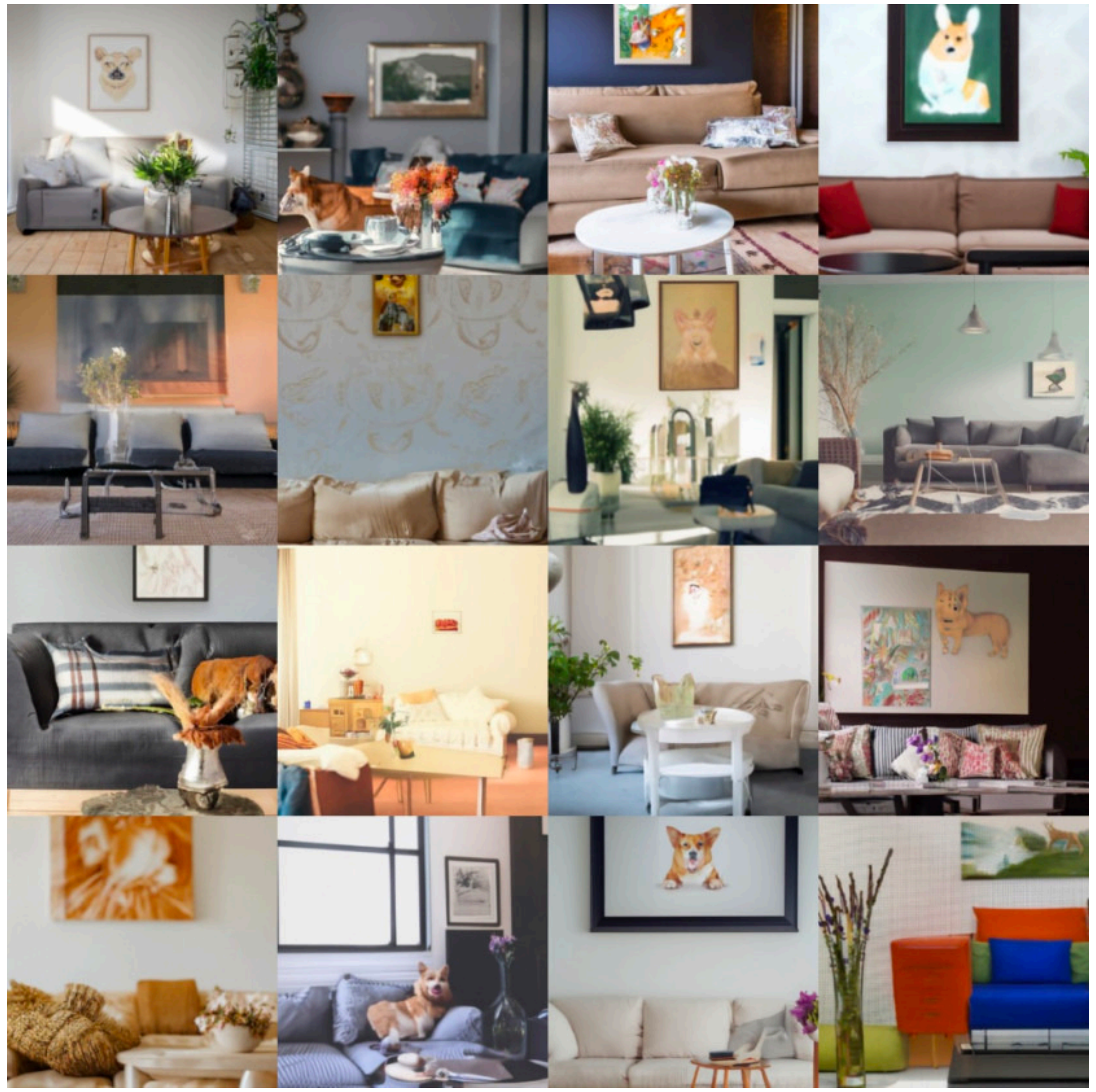
$\gamma = 1$

Equivalent to explicit conditioning.
No guidance

Nichol et al. “GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models”

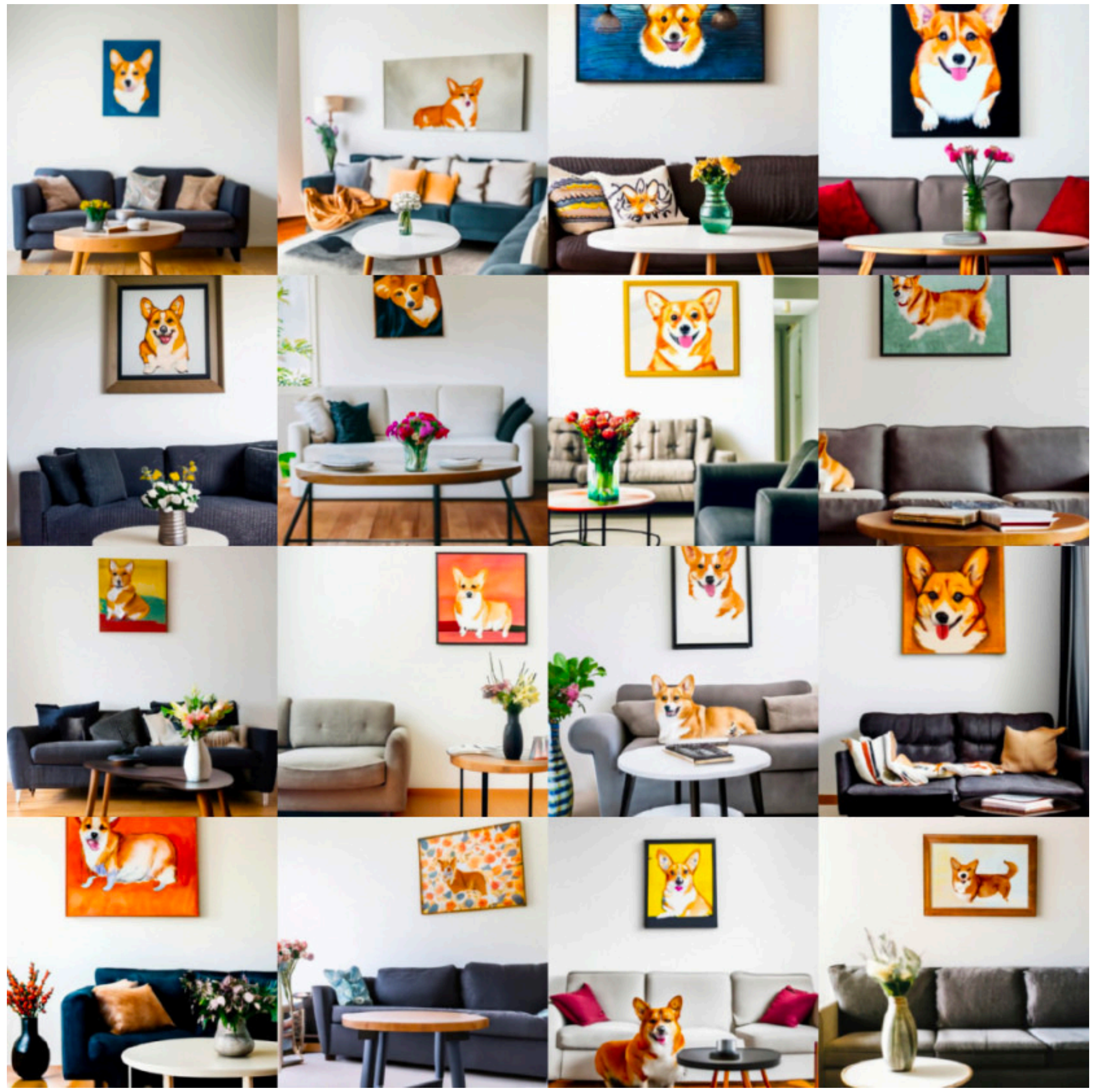
Classifier Free Guidance

"A cozy living room with a painting of a corgi on the wall above a couch and a round coffee table in front of a couch and a vase of flowers on a coffee table."



$\gamma = 1$

Equivalent to explicit conditioning.
No guidance



$\gamma = 3$

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

More Resources

- Lilian Weng Tutorial: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
- Guidance Tutorial by Sander Dieleman: https://sander.ai/2022/05/26/guidance.html

Image Editing with Diffusion Models

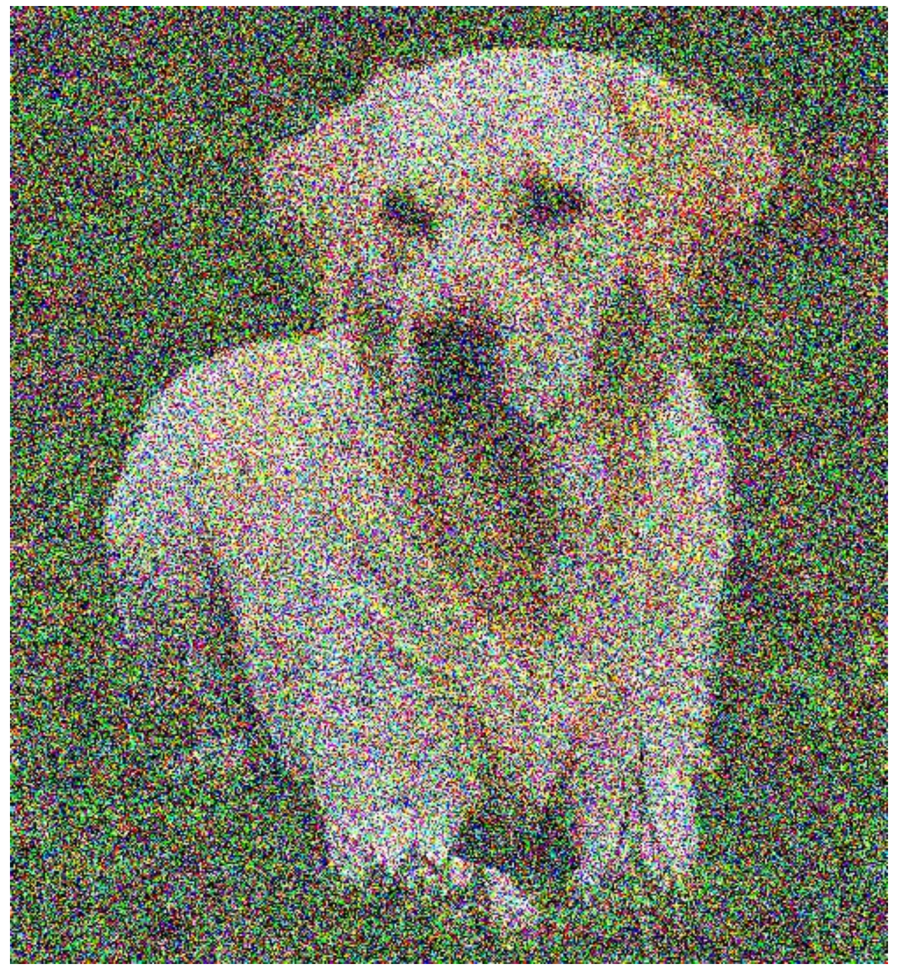
SDEdit

Idea: Add noise to an image, and then remove it with a diffusion model

Meng et al. “SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations”

SDEdit

Idea: Add noise to an image, and then remove it with a diffusion model



Add noise by running the forward process $q(x_t|x_0)$

Meng et al. "SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations"

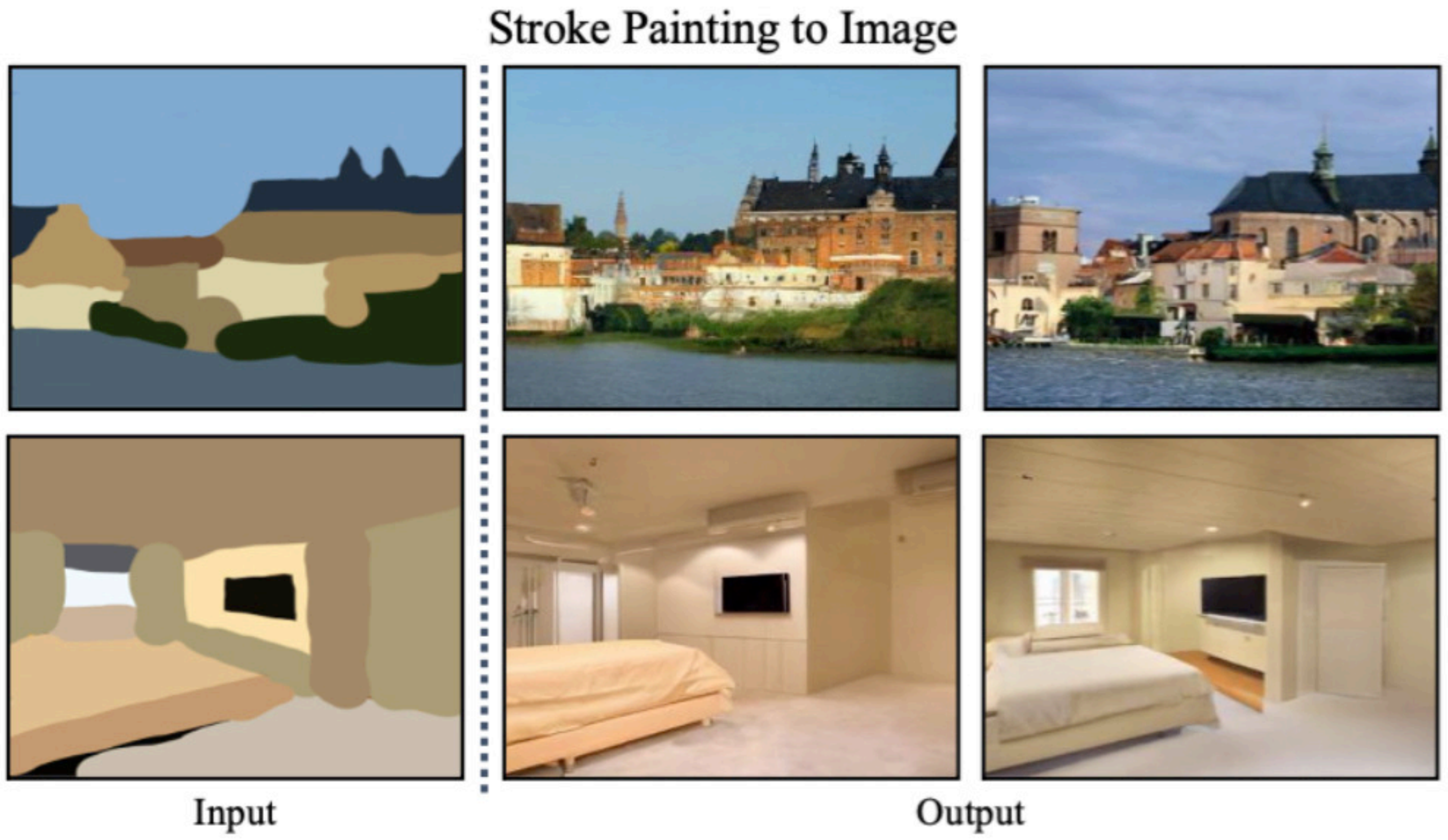
SDEdit

Stroke Painting to Image

Input

Meng et al. “SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations”

SDEdit



Meng et al. “SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations”

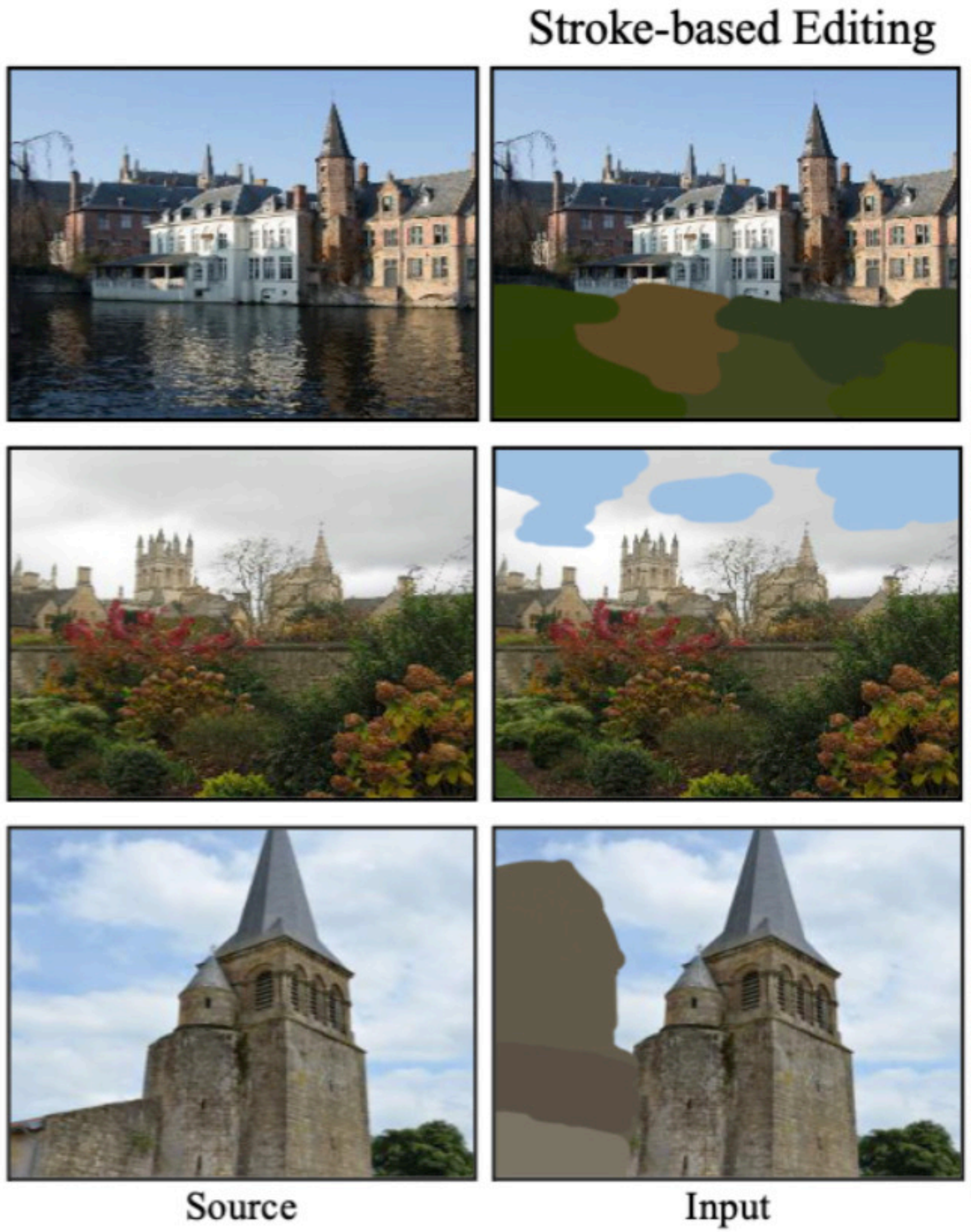
SDEdit

Stroke-based Editing

Source

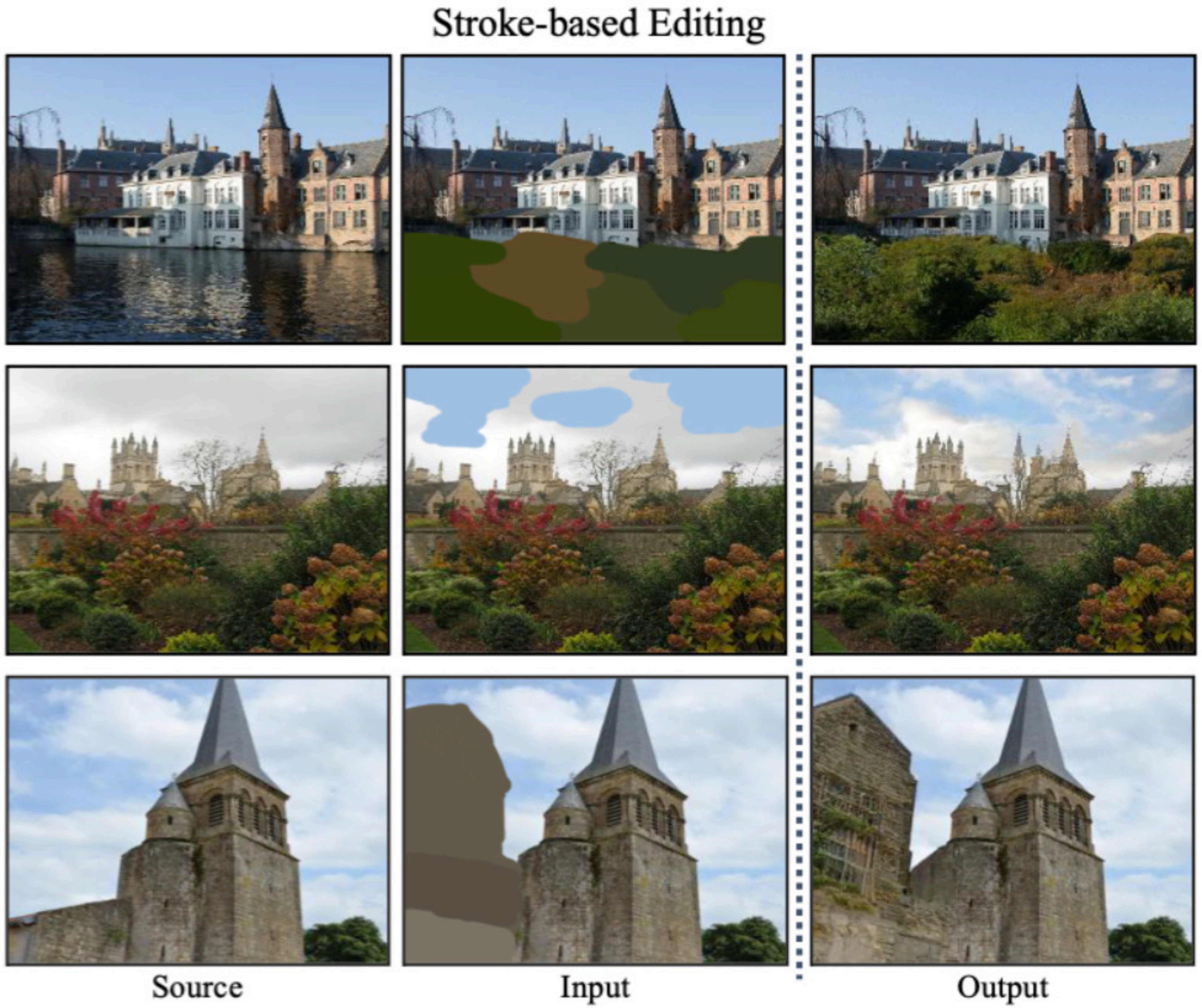
Meng et al. “SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations”

SDEdit



Meng et al. “SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations”

SDEdit



Meng et al. "SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations"

Prompt-to-Prompt

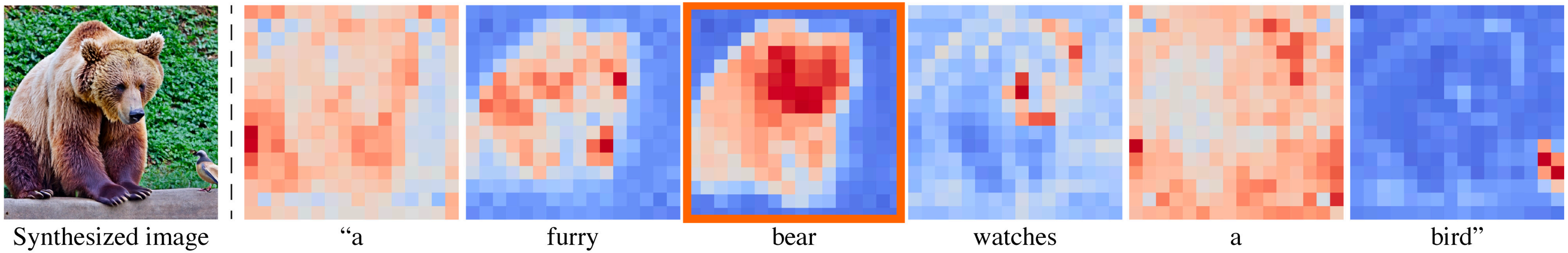
Hertz et al. "Prompt-to-Prompt Image Editing with Cross-Attention Control"

Prompt-to-Prompt

Hertz et al. "Prompt-to-Prompt Image Editing with Cross-Attention Control"

Prompt-to-Prompt

(High Level) Idea: Features inside diffusion models encode very high level information such as: style, content, and structure

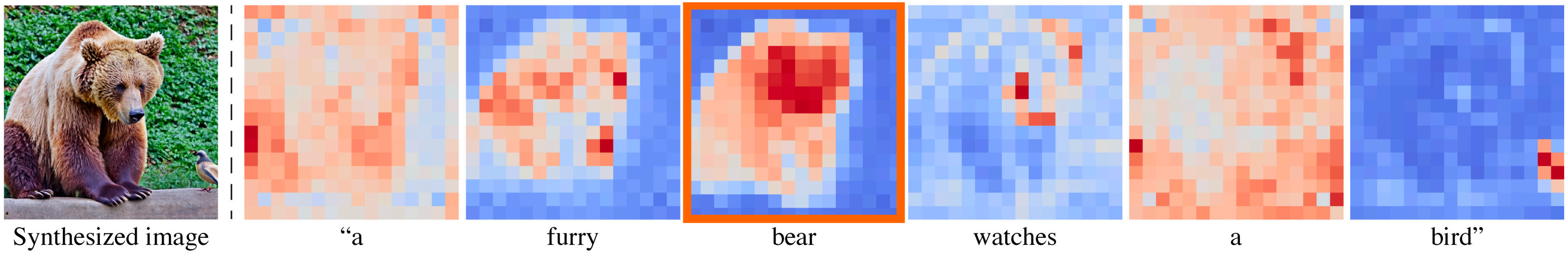


Average cross–attention maps across all timestamps

Hertz et al. “Prompt-to-Prompt Image Editing with Cross-Attention Control”

Prompt-to-Prompt

(High Level) Idea: Features inside diffusion models encode very high level information such as: style, content, and structure



Average cross–attention maps across all timestamps

Reuse (copy and paste) the features from the previous prompt

Hertz et al. “Prompt-to-Prompt Image Editing with Cross-Attention Control”

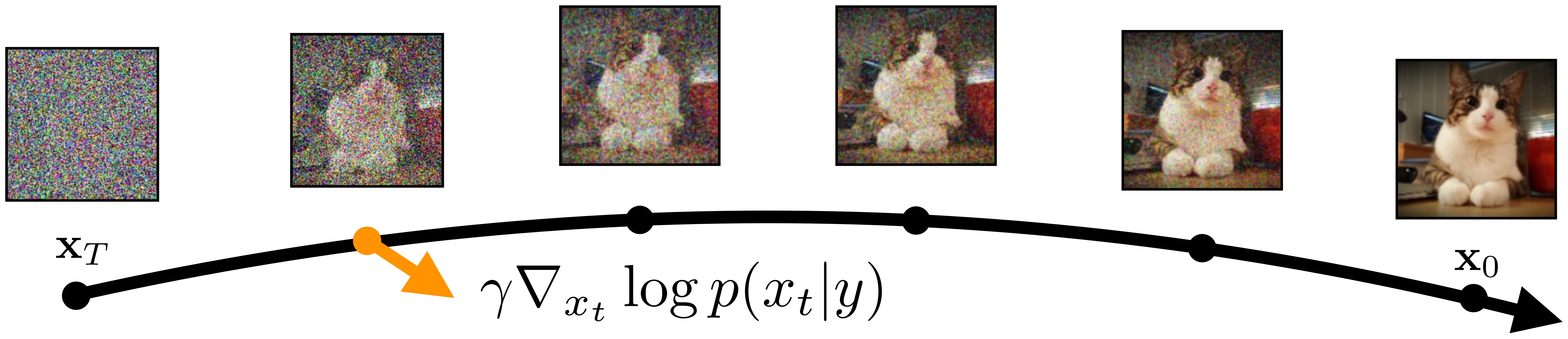
Motion Guidance

Motion Guidance

Earlier: We saw we can do classifier guidance with an ImageNet classifier

Motion Guidance

Earlier: We saw we can do classifier guidance with an ImageNet classifier



$$\boxed{\tilde{\epsilon}_t(x_t, t, y) = \epsilon_\theta(x_t, t) - \gamma \nabla_{x_t} \log p(x_t|y)}$$

Motion Guidance

We can use other models besides
image classifiers for classifier guidance

Teed and Deng. "RAFT: Recurrent All
Pairs Field Transforms for Optical Flow"

Motion Guidance

We can use other models besides
image classifiers for classifier guidance

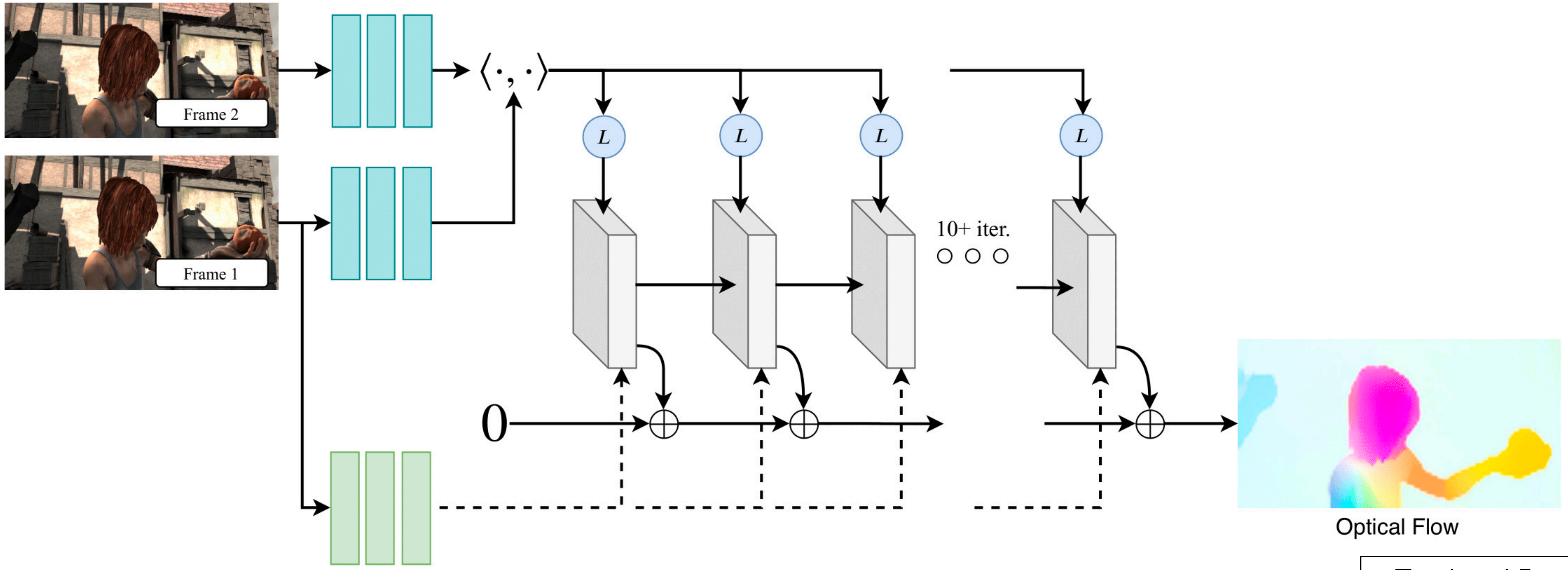
Idea: Let's do classifier guidance with a
"motion estimator" (optical flow network)

Teed and Deng. "RAFT: Recurrent All Pairs Field Transforms for Optical Flow"

Motion Guidance

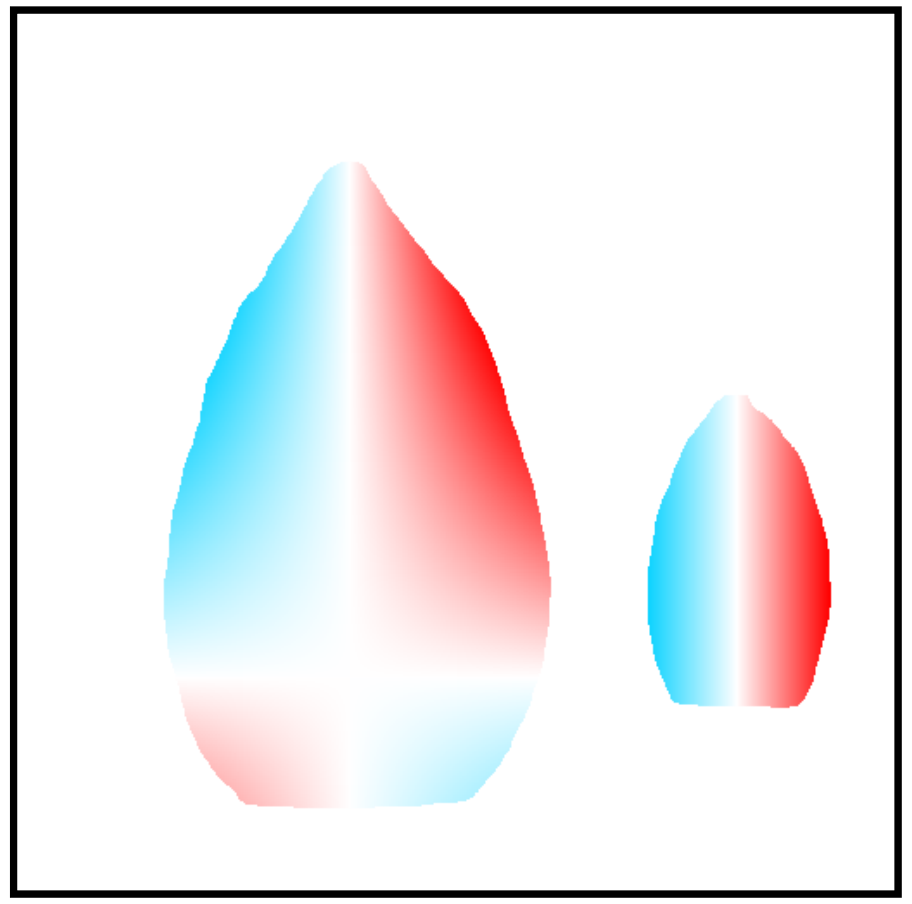
We can use other models besides image classifiers for classifier guidance

Idea: Let's do classifier guidance with a "motion estimator" (optical flow network)



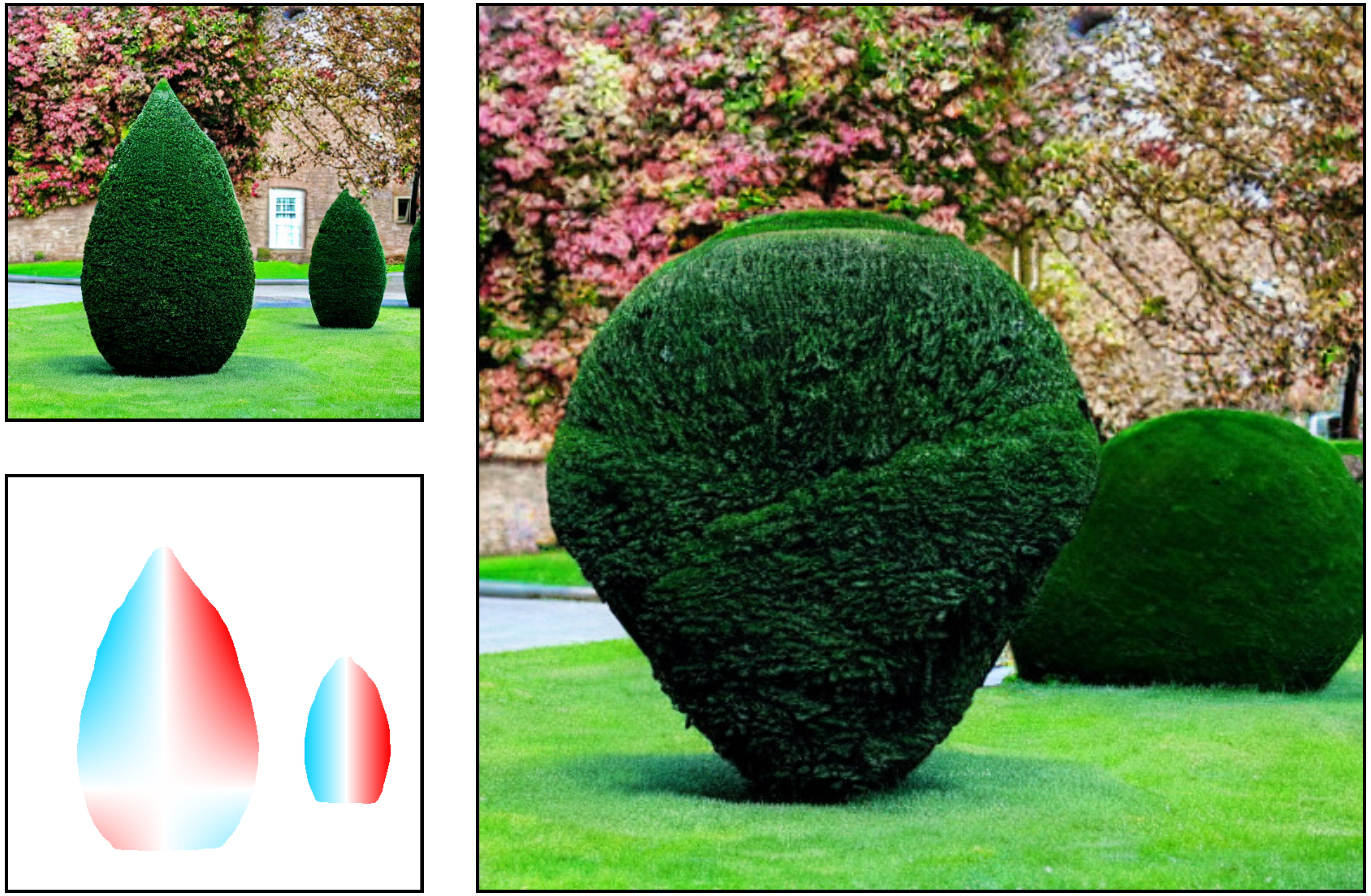
Teed and Deng. "RAFT: Recurrent All Pairs Field Transforms for Optical Flow"

Motion Guidance



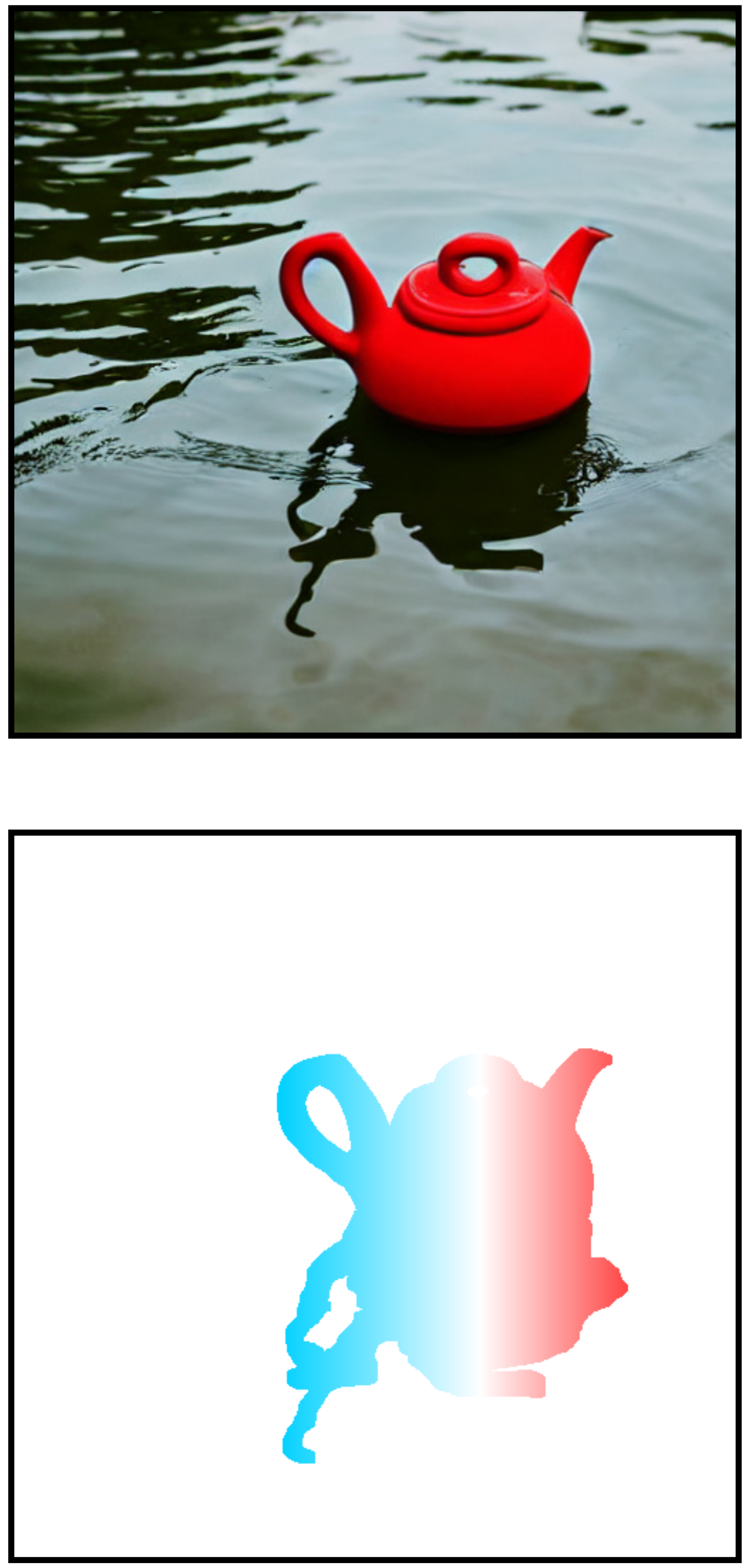
"a photo of topiary"

Motion Guidance



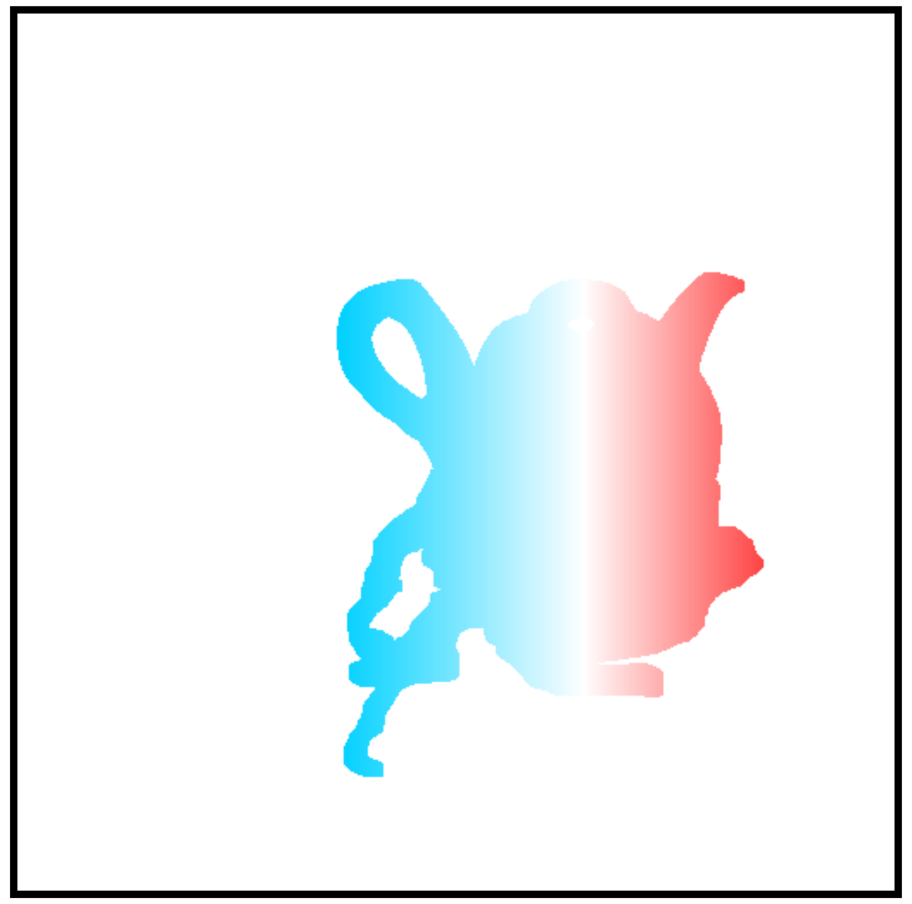
“a photo of topiary”

Motion Guidance



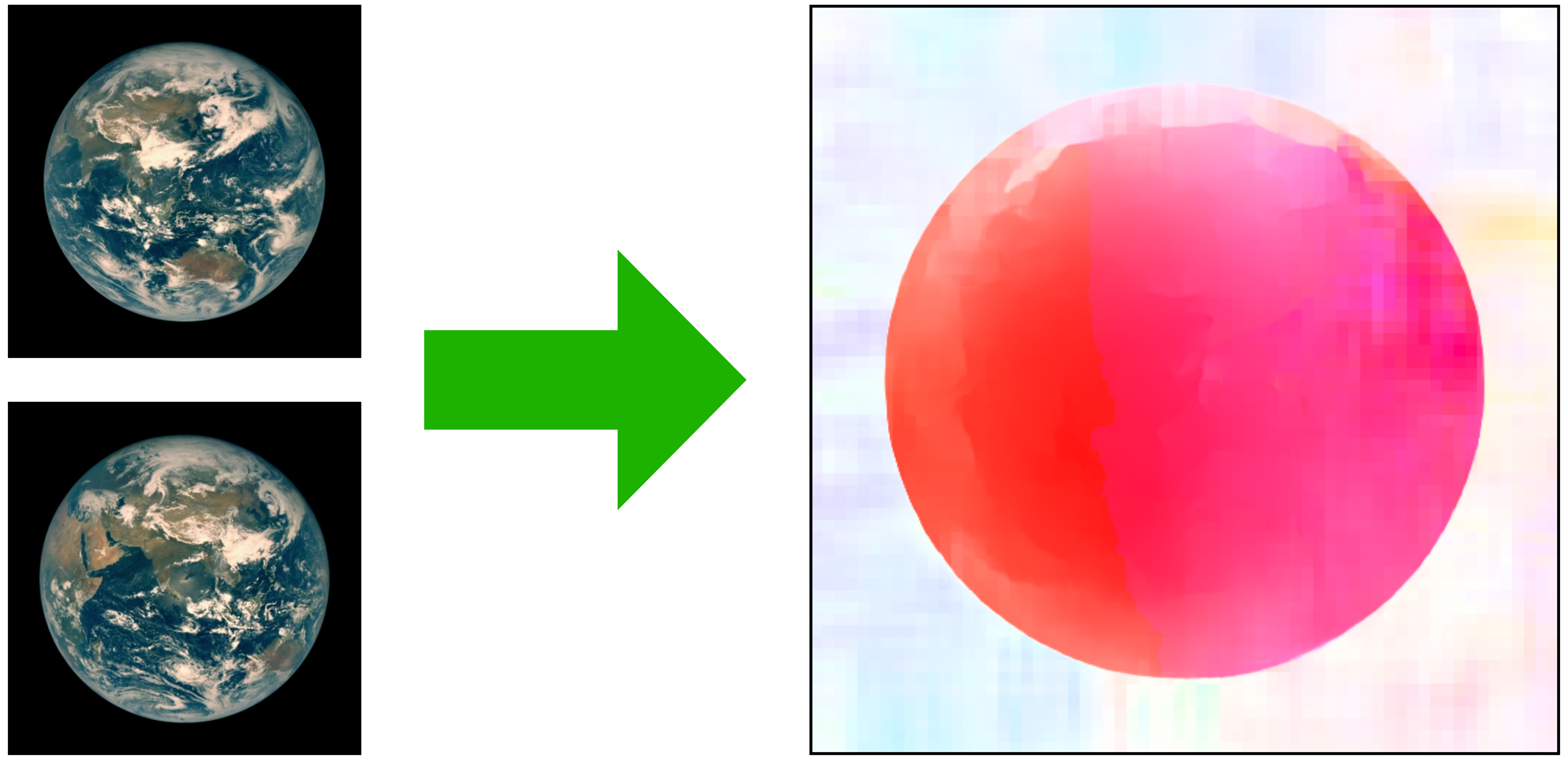
a) *"a teapot floating in water"*

Motion Guidance



a) *"a teapot floating in water"*

Motion Guidance

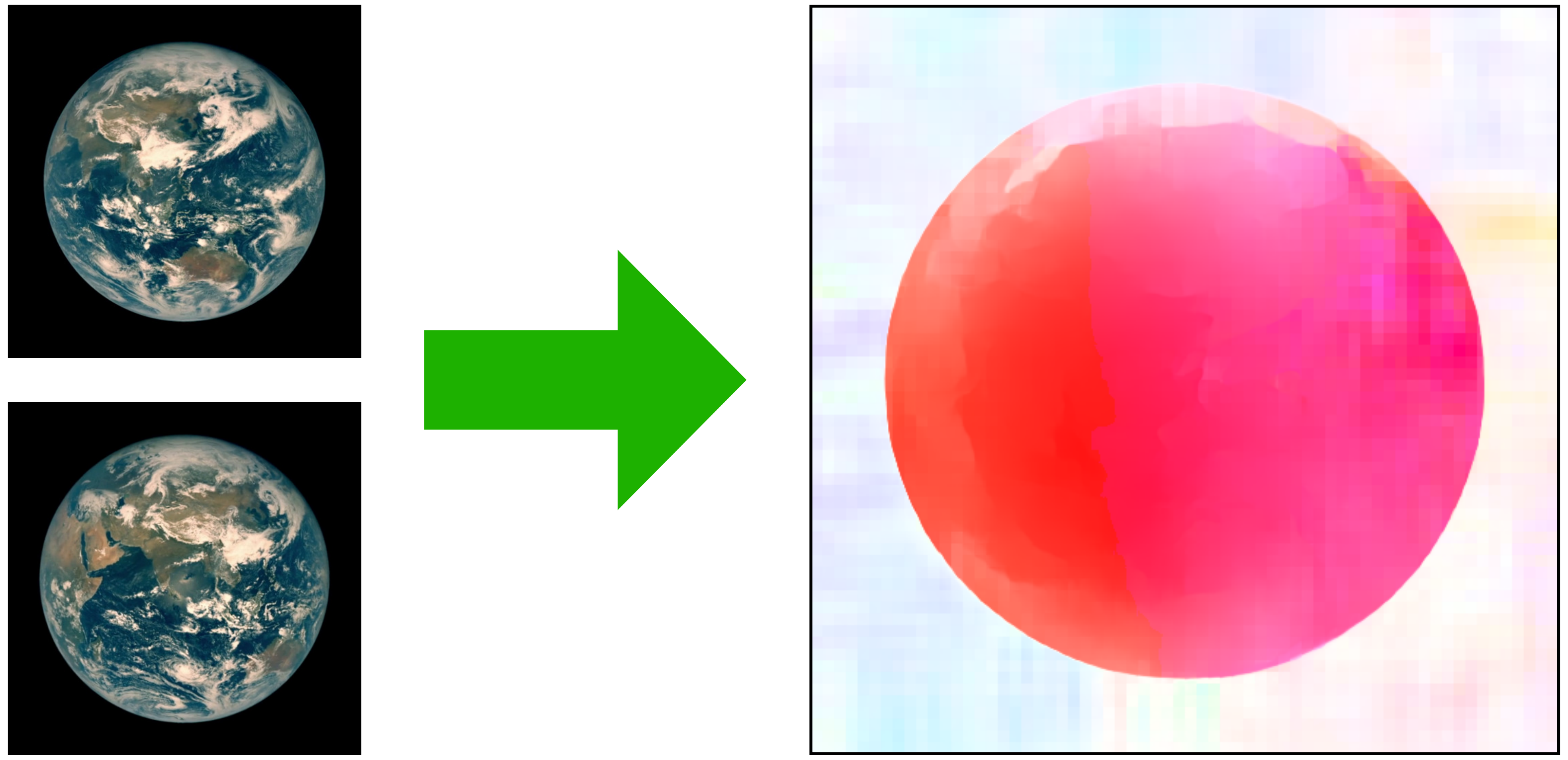


[real image]

[real image]

[real image]

Motion Guidance



[real image]

[real image]

[real image]