
Diffusion Models

— Yutong (Kelly) He 11/20/23 —



About Me



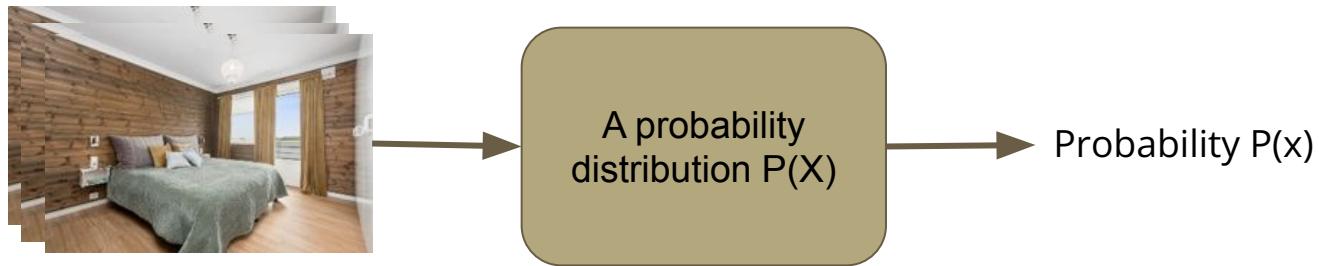
- Yutong (Kelly) He, 2nd year PhD in MLD advised by Zico & Russ
- Things I have done in the past
 - H-Divergence: Comparing Distributions by Measuring Differences that Affect Decision Making [[paper](#)]
 - Generative Modeling:
 - Image-to-Image Translation [[paper](#)]
 - Super-Resolution [[paper](#)]
 - ✓ SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations [[paper](#)]

Generative Modeling

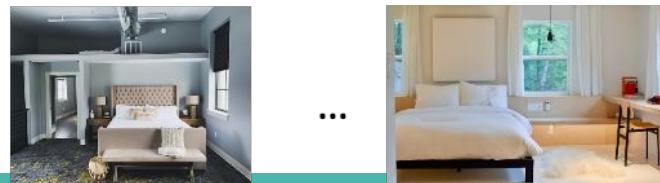


A (statistical) generative model is a **probability distribution $P(X)$** (as opposed to $P(Y | X)$ in discriminative models)

- **Data:** samples (e.g., images of bedrooms)
- **Prior knowledge:** parametric form, loss function, optimization, etc.

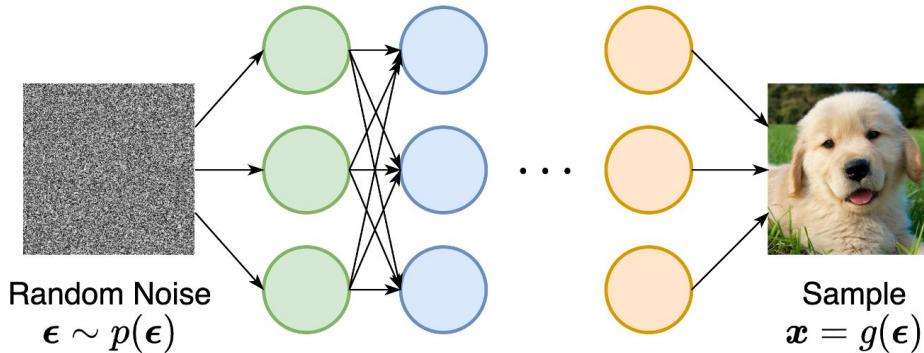


It is generative because sampling from $p(x)$ generates new images



Generative Models

- **Likelihood Based:** Autoregressive models, variational autoencoders (VAE), normalizing flow, energy-based models (EBM)
- **Likelihood Free:** Generative adversarial networks (GAN)

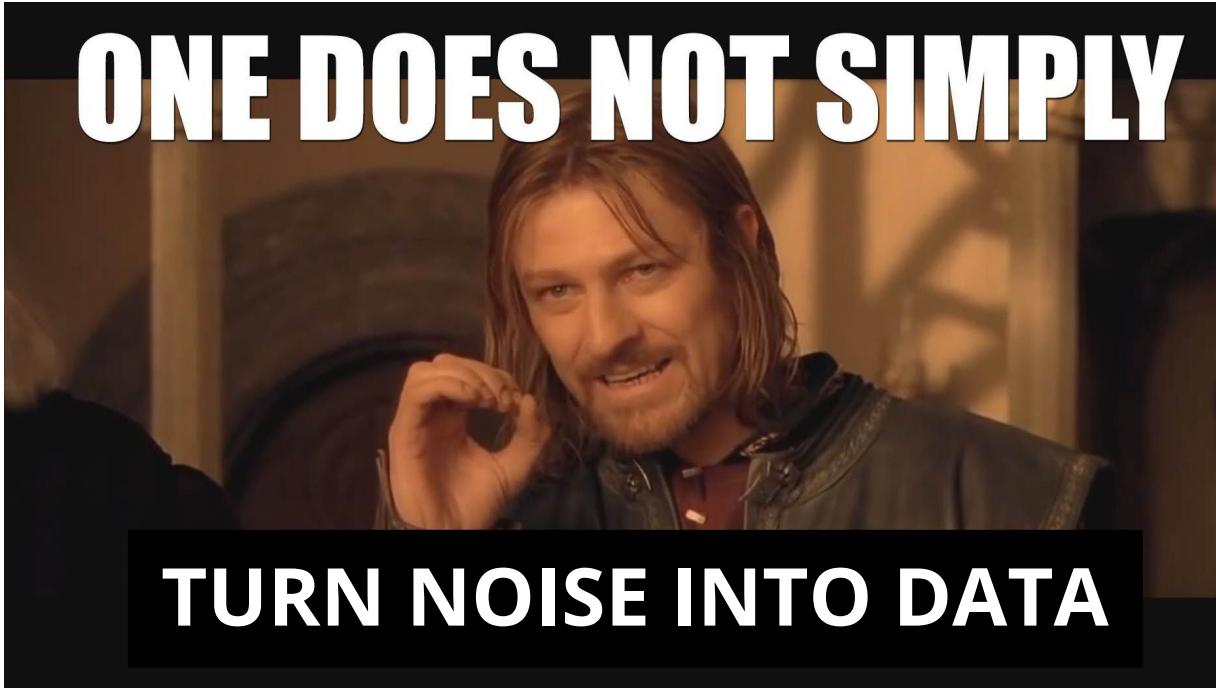


Directly sampling from $P(X)$ is usually hard because they are usually complicated! But **sampling from a simpler distribution** (eg. a Gaussian) is easy!

From Noise to Data

ONE DOES NOT SIMPLY

TURN NOISE INTO DATA

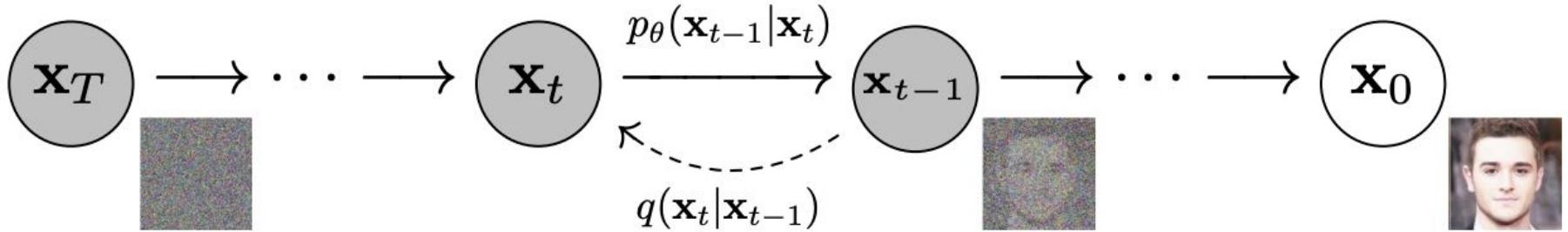


From Noise to Data



Now you get **Diffusion Model!**

Diffusion Model (Sohl-Dickstein, et al. 2015)



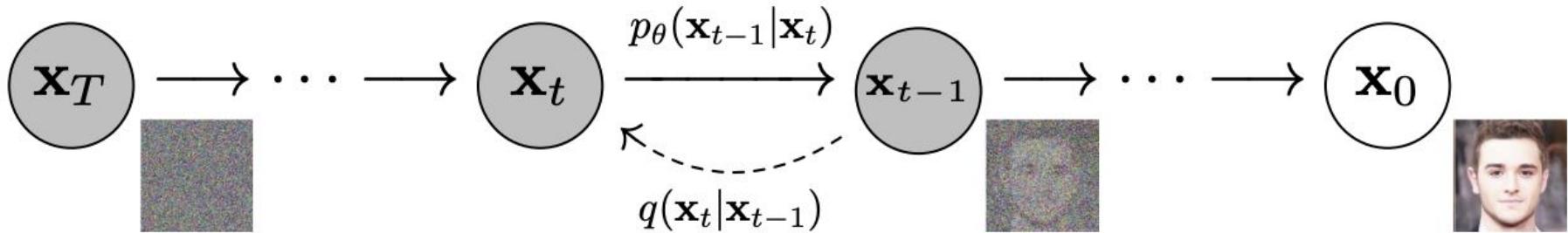
Forward (Adding Noise):

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

Backward (Denoising):

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Diffusion Model (Sohl-Dickstein, et al. 2015)



Training Objective:

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

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

Diffusion Model (Sohl-Dickstein, et al. 2015)

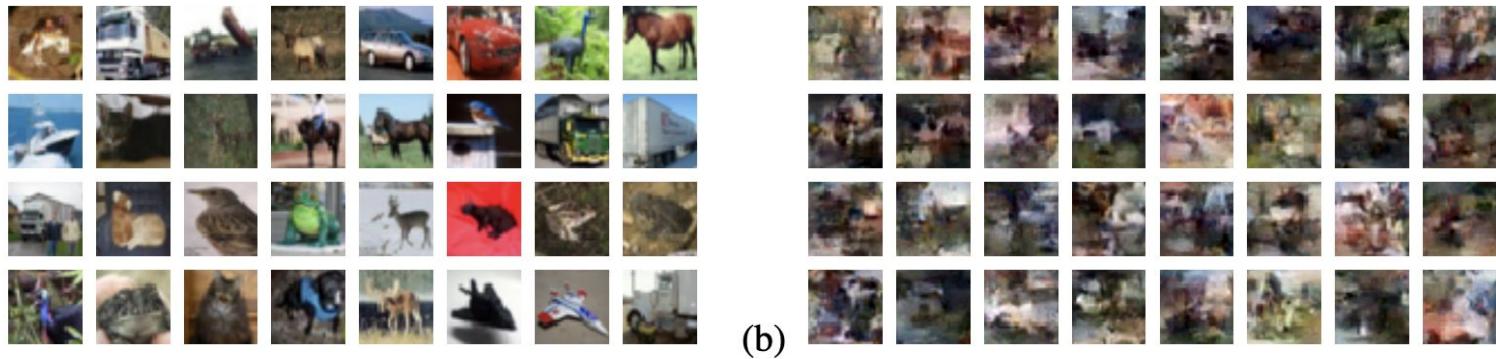
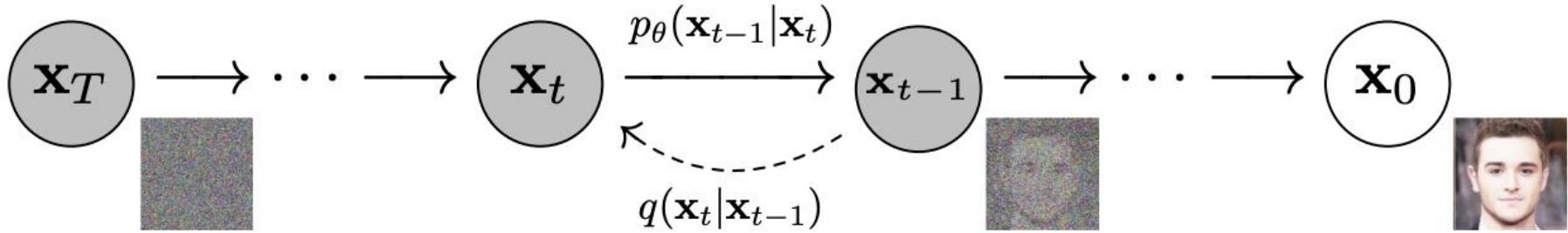


Figure 3. The proposed framework trained on the CIFAR-10 (Krizhevsky & Hinton, 2009) dataset. (a) Example training data. (b) Random samples generated by the diffusion model.

Denoising Diffusion Probabilistic Model (DDPM)

(Ho, et al. 2020)



So we know this thing is Markov – it just adds a small amount of noise at every time step. Then why don't we just learn a noise predictor to predict the noise at each time step and then gradually reduce the noise?

Denoising Diffusion Probabilistic Model (DDPM)

(Ho, et al. 2020)

Algorithm 1 Training

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

Algorithm 2 Sampling

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

Denoising Diffusion Probabilistic Model (DDPM)

(Ho, et al. 2020)

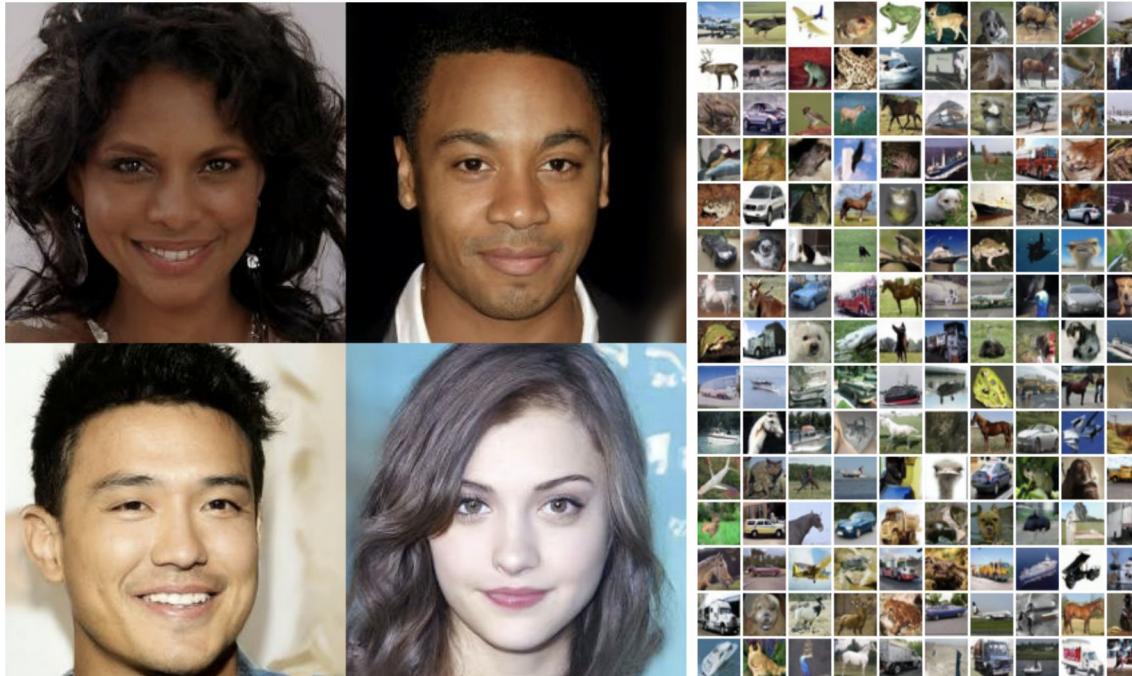
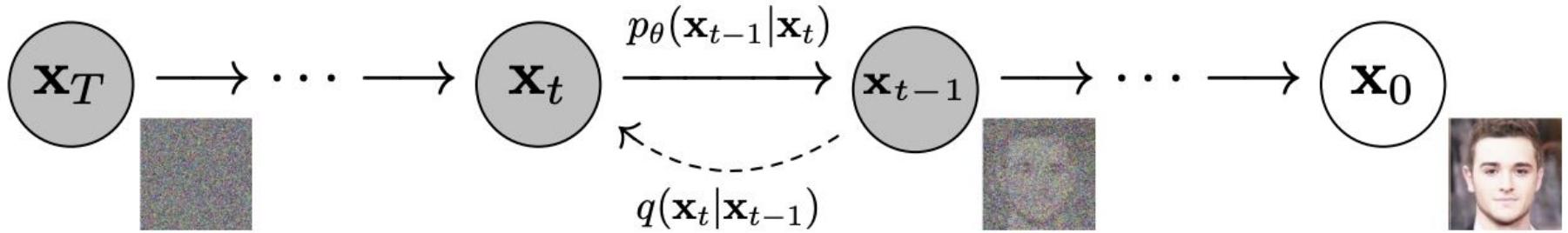


Figure 1: Generated samples on CelebA-HQ 256×256 (left) and unconditional CIFAR10 (right)

Denoising Diffusion Probabilistic Model (DDPM) (Ho, et al. 2020)



But how is this not dark magic? How do we know where/which direction the noise should go?



To explain this, we can take a probabilistic approach

A (statistical) generative model is a **probability distribution $P(X)$** (as opposed to $P(Y | X)$ in discriminative models)

- **Likelihood Based:** Autoregressive models, variational autoencoders (VAE), normalizing flow, energy-based models (EBM)

Maximizes the likelihood of the data under the model $\max_{\theta} \sum_{i=1}^N \log p_{\theta}(\mathbf{x}_i)$

- However, modeling $p_{\theta}(\mathbf{x})$ is hard
 - VAE: Surrogate loss
 - Normalizing Flow: Weird architecture
 - EBM: Intractable partition function
 - Autoregressive Models: Break it up with chain rule, works for some, doesn't make sense for others, can also take a long time and can't generate everything all at once



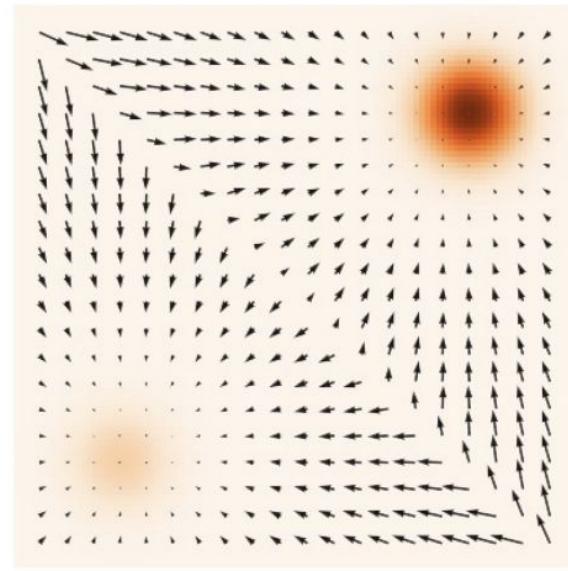
But ...

Do we really need to estimate $p_\theta(\mathbf{x})$ in order to train the model?

- How we usually train a model: Stochastic **gradient** descent (or ascent) with maximum **log likelihood**
- So we just need $\nabla_{\mathbf{x}} \log p(\mathbf{x})$



Score function



Score-based Model

Now we just need to train a model to estimate the score function

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$$

by minimizing

$$\mathbb{E}_{p(\mathbf{x})} [\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2]$$

- No more intractable partition function
- No more adversarial training
- No more weird architecture
- No breaking up with chain rule => can generate everything all at once

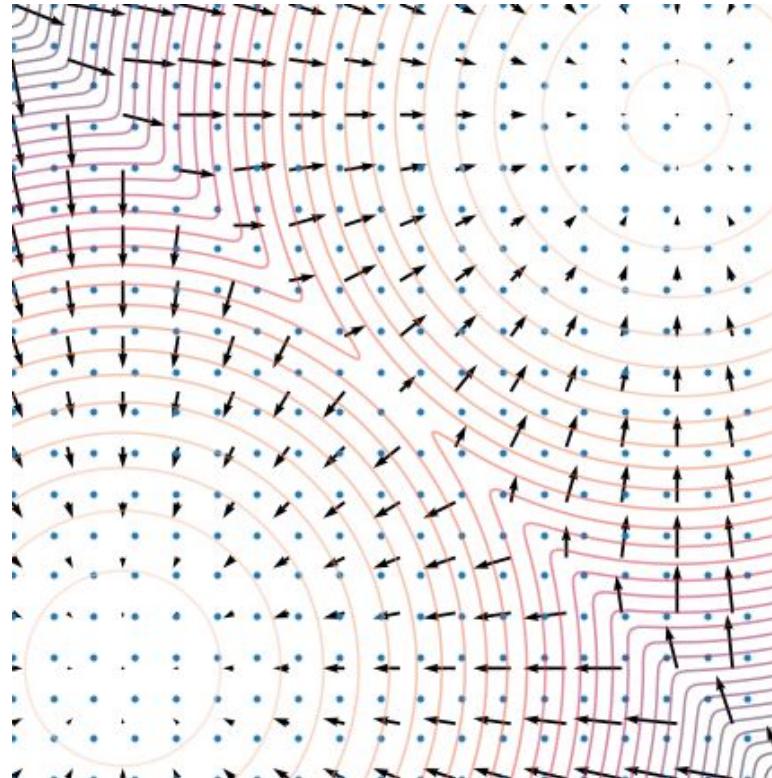
Sampling with Langevin Dynamics

Once we have a trained score model, we can do “gradient ascent” in the data space to get a sample.

- First draw from a easier-to-sample prior distribution $\mathbf{x}_0 \sim \pi(\mathbf{x})$.
- Then with small ϵ and large K and $\mathbf{z}_i \sim \mathcal{N}(0, I)$, iterate

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon} \mathbf{z}_i, \quad i = 0, 1, \dots, K,$$

Sampling with Langevin Dynamics



However ...

- The score is only defined in the whole data space, if data resides in a lower dimensional manifold (i.e. some area of the data space will have no support), then the score estimation is not consistent any more

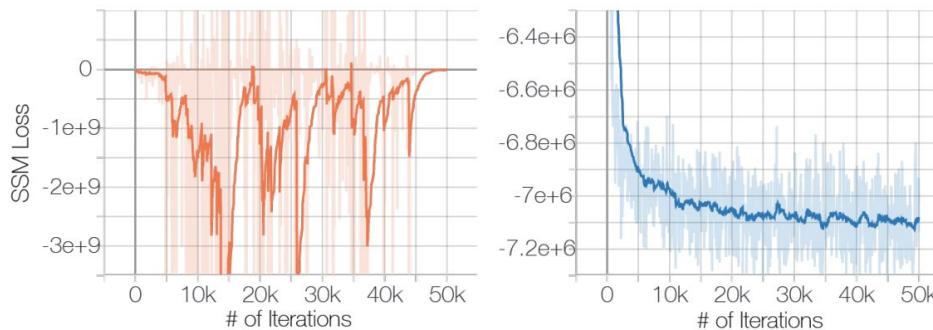
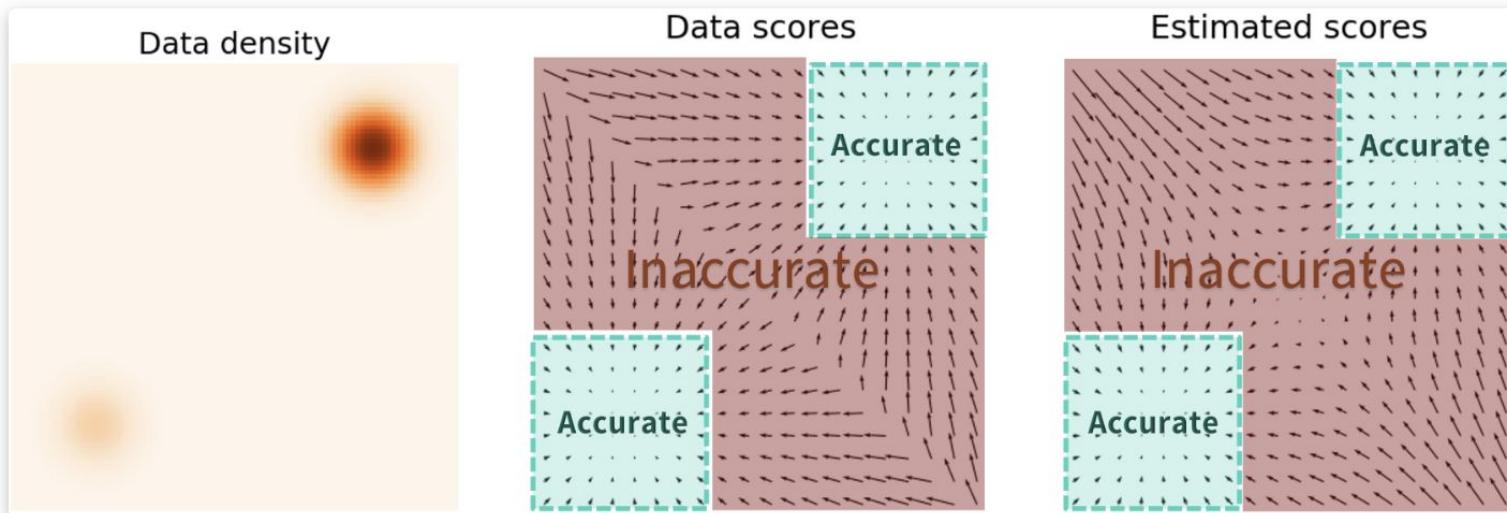


Figure 1: **Left:** Sliced score matching (SSM) loss w.r.t. iterations. No noise is added to data. **Right:** Same but data are perturbed with $\mathcal{N}(0, 0.0001)$.

However (2) ...

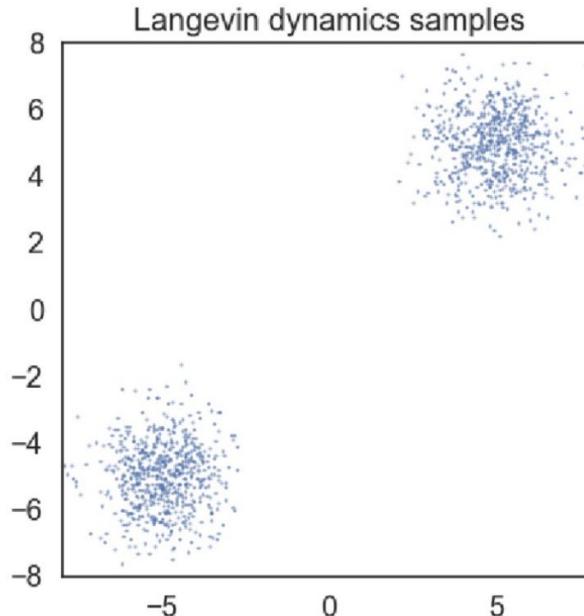
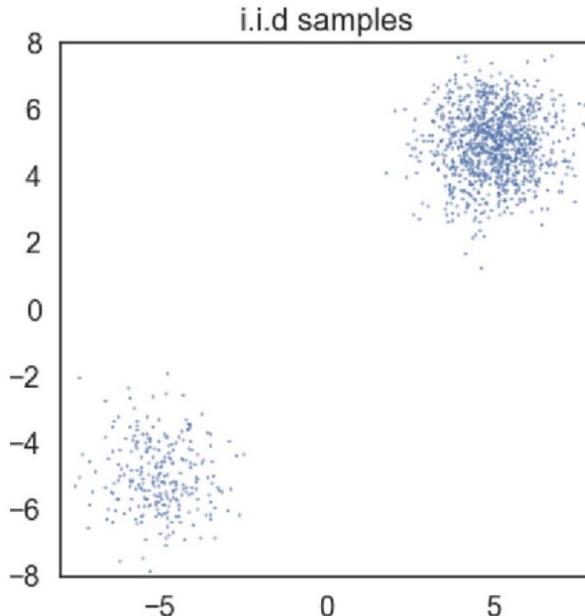
- Even when we do have full support data space, the score estimation is inaccurate in the low density regions

And our initial sample is very likely to be in those low density regions!!!



However (3) ...

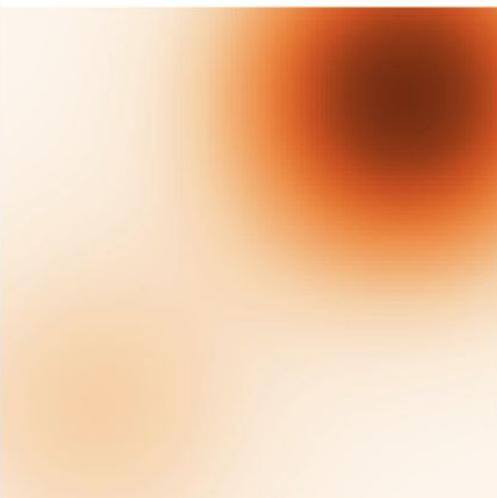
- Even in high density region, we can still get inaccurate sample distributions



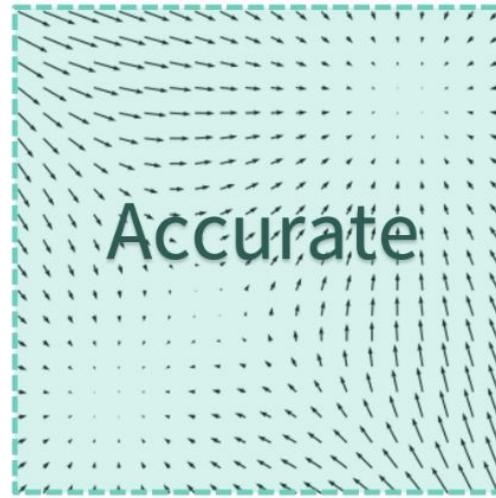
To solve all these problems

How about just **add noise** to the data?

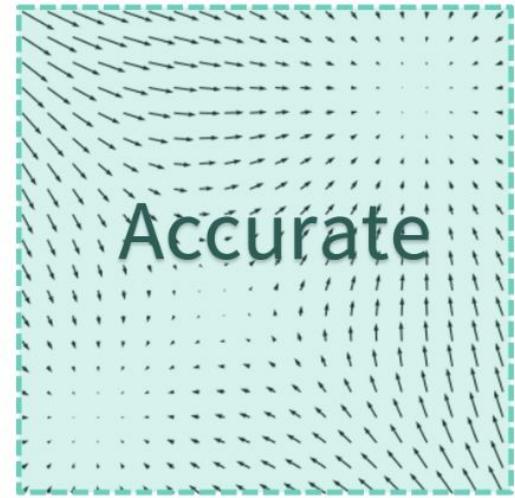
Perturbed density



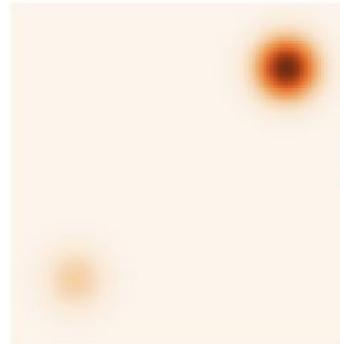
Perturbed scores



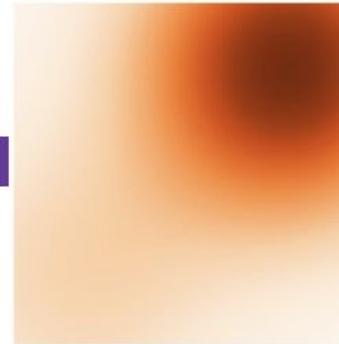
Estimated scores



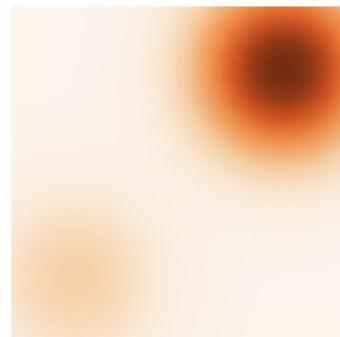
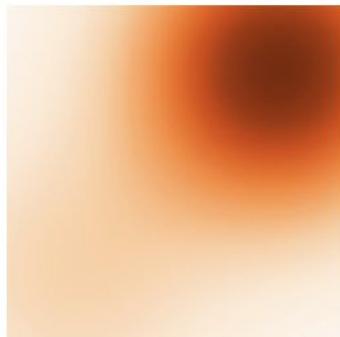
But how much noise we should add?



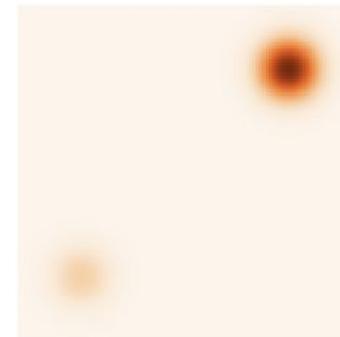
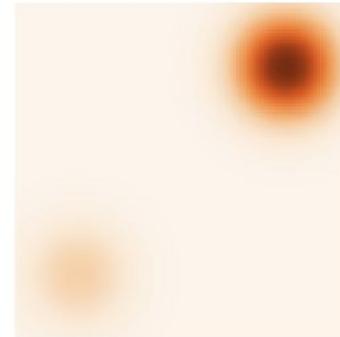
Too small



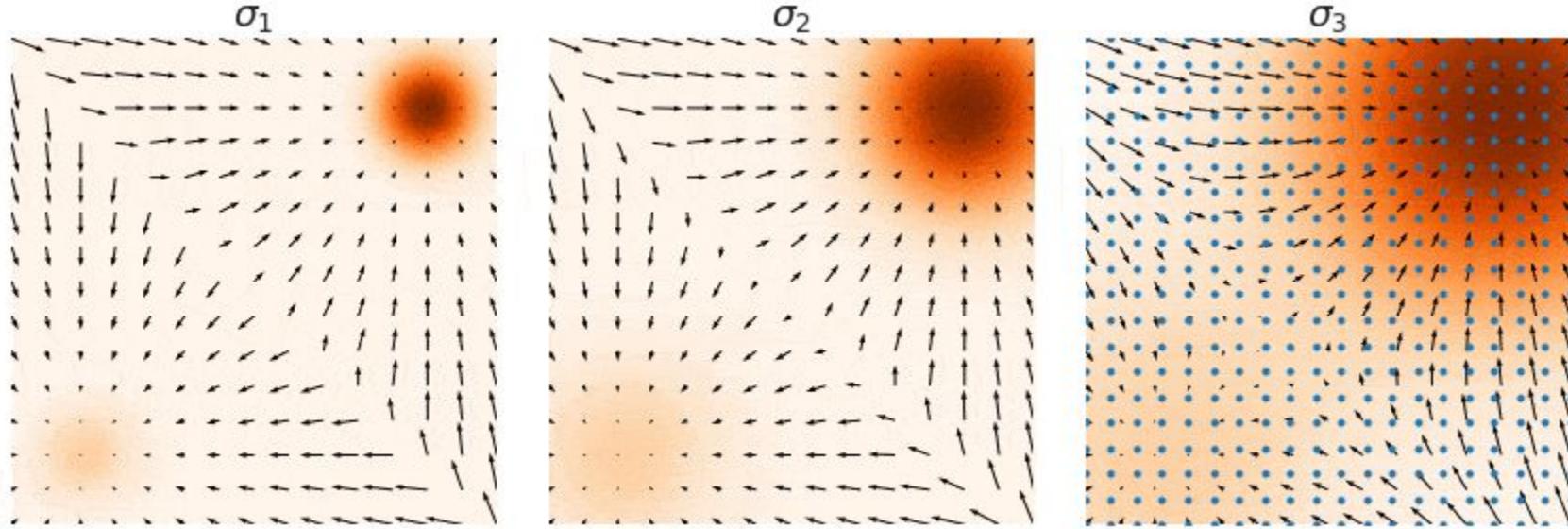
Too large



...



Annealed Langevin Dynamics



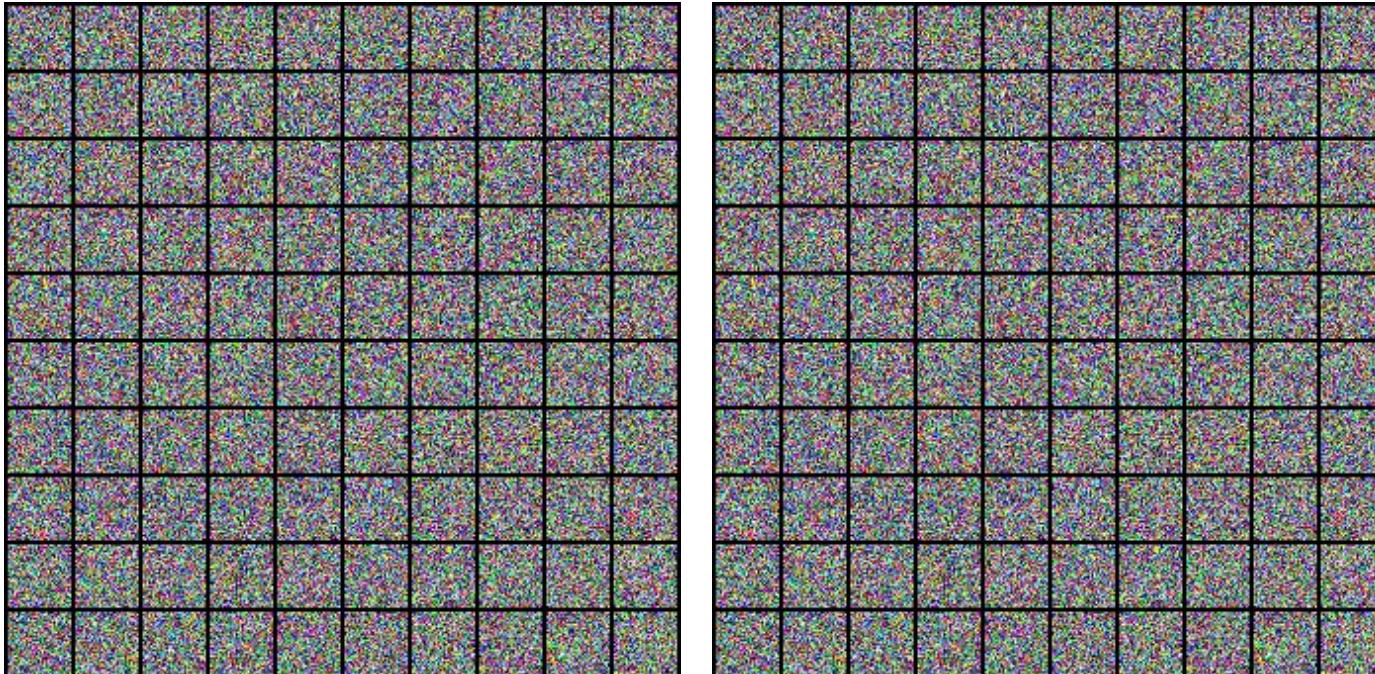
Annealed Langevin Dynamics

Algorithm 1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^L, \epsilon, T$.

```
1: Initialize  $\tilde{\mathbf{x}}_0$ 
2: for  $i \leftarrow 1$  to  $L$  do
3:    $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$        $\triangleright \alpha_i$  is the step size.
4:   for  $t \leftarrow 1$  to  $T$  do
5:     Draw  $\mathbf{z}_t \sim \mathcal{N}(0, I)$ 
6:      $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i} \mathbf{z}_t$ 
7:   end for
8:    $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$ 
9: end for
return  $\tilde{\mathbf{x}}_T$ 
```

Noise Conditional Score Network (NCSC) (Song and Ermon 2019)



Diffusion Model (Sohl-Dickstein, et al. 2015)

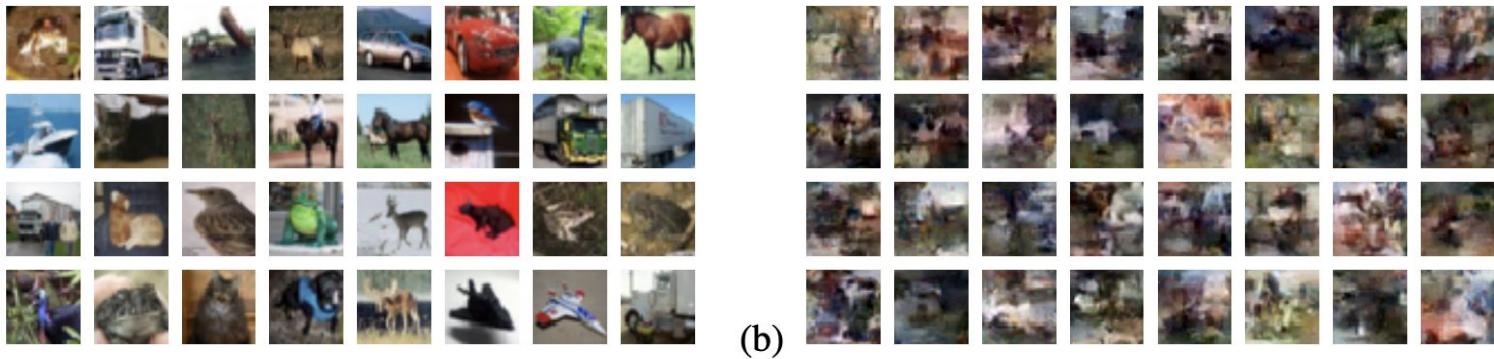
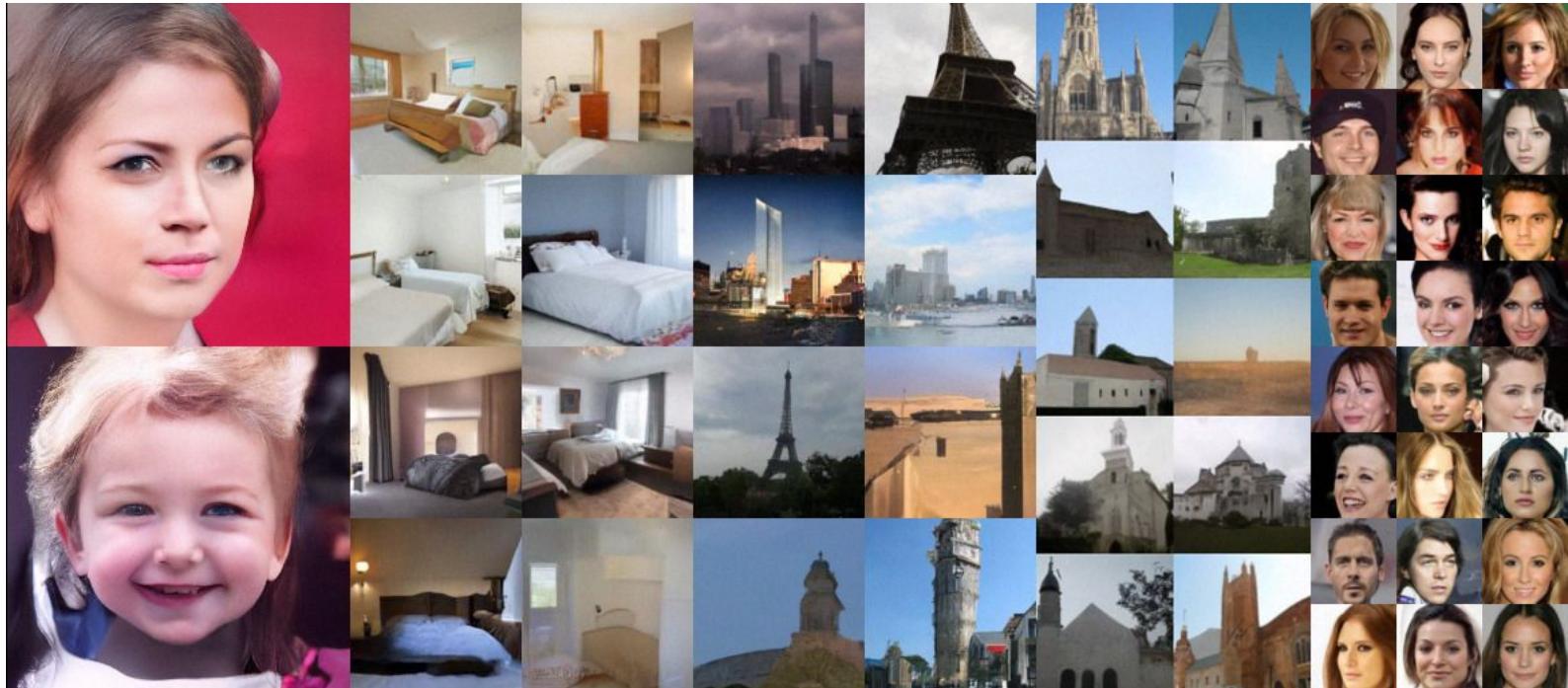


Figure 3. The proposed framework trained on the CIFAR-10 (Krizhevsky & Hinton, 2009) dataset. (a) Example training data. (b) Random samples generated by the diffusion model.

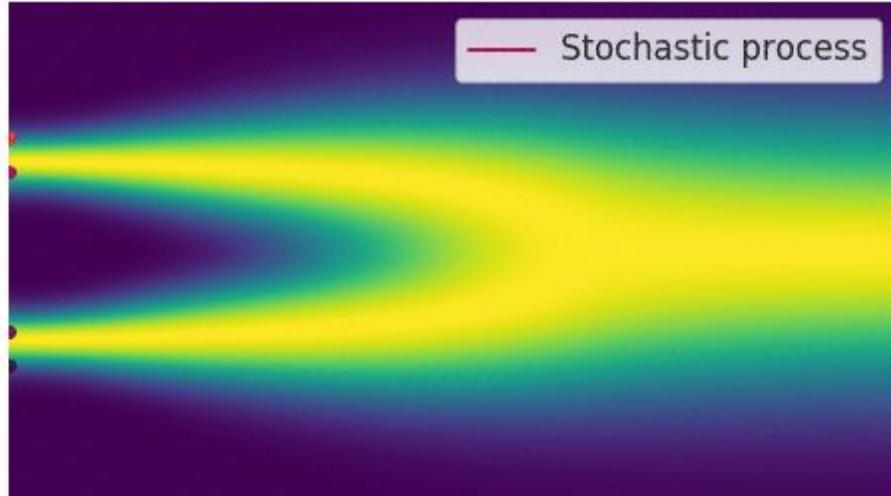
NCSC v2 (Song and Ermon 2020)



When # of noise scales goes to infinity

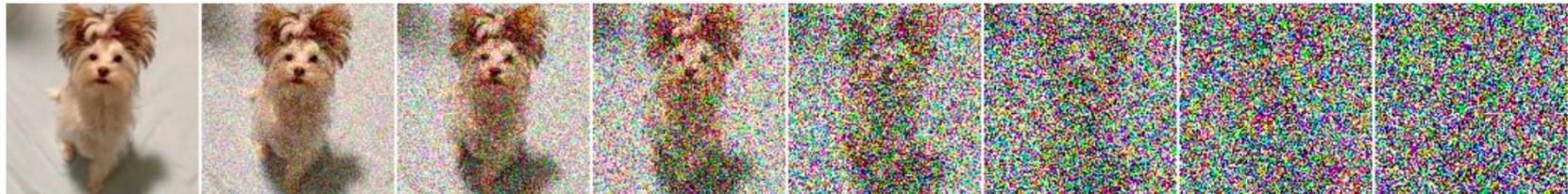
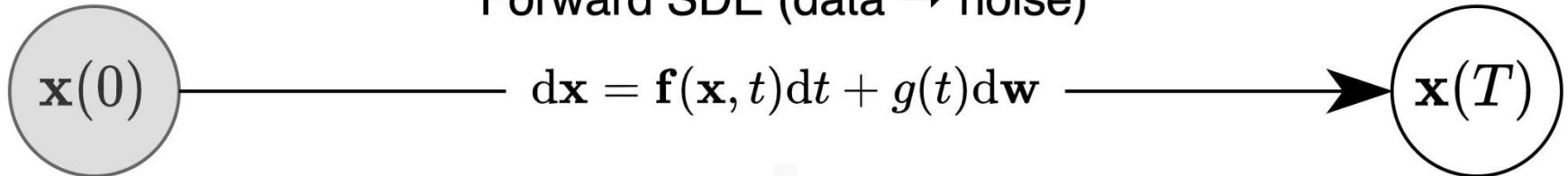
It becomes a continuous-time stochastic process, many of which can be solved by stochastic differential equations (SDEs) (Diffusion is no exception)

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

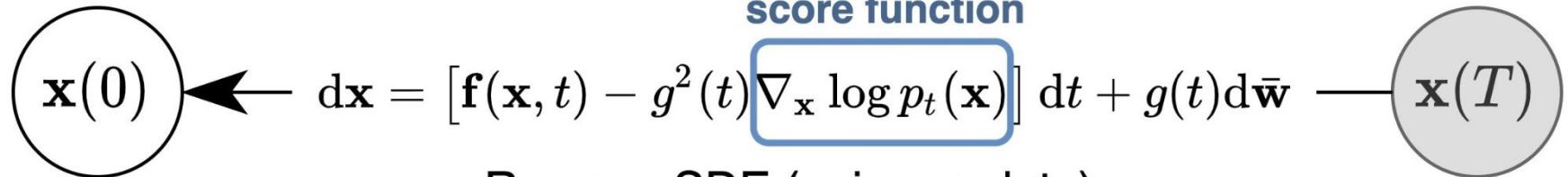


Score-SDE (Song et al. 2021)

Forward SDE (data → noise)



score function

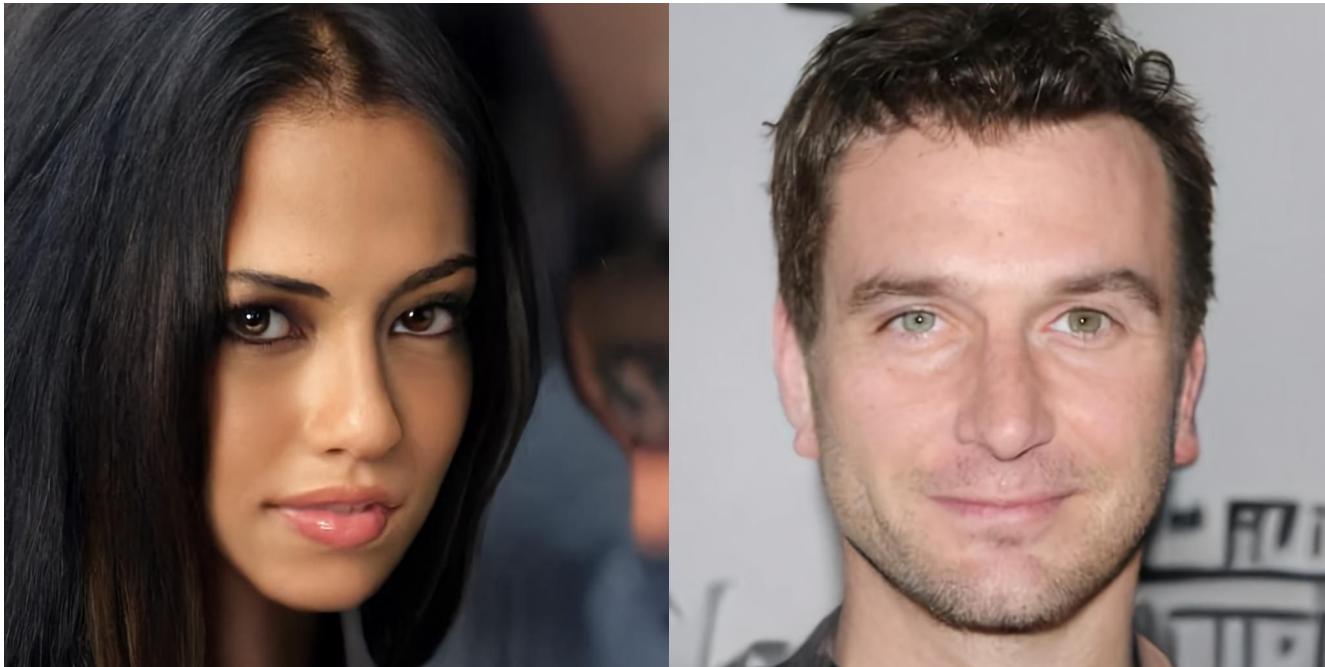


Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Score-Based Generative Modeling through Stochastic Differential Equations". ICLR 2021. <https://openreview.net/pdf?id=PxTIG12RRHS>

How to sample from the reverse SDE

- First sample from the prior distribution $\mathbf{x}(T) \sim \pi$
- Then apply numerical SDE solver to solve for $\mathbf{x}(0)$
 - Eg. Euler-Maruyama method
- Since we have the **score model** and we **only care about the final sample** (and not the trajectory), we can even improve the numerical solution by applying a MCMC-based approach called **Predictor-Corrector** samplers to fine-tune the trajectories
 - **Predictor:** Choose a proper step size, and then predict the next sample based on the current sample and trajectory (eg. any numerical SDE solver)
 - **Corrector:** improve the sample prediction with MCMC according to our score-based model so that becomes a higher-quality sample from the probability distribution of the next noise level (eg. Langevin dynamics)

Score-SDE (Song et al. 2021)



Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Score-Based Generative Modeling through Stochastic Differential Equations". ICLR 2021. <https://openreview.net/pdf?id=PxTIG12RRHS>

Score-SDE (Song et al. 2021)



Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Score-Based Generative Modeling through Stochastic Differential Equations". ICLR 2021. <https://openreview.net/pdf?id=PxTIG12RRHS>

Conditional Generation w/ Score-SDE (Song et al. 2021)

By Bayes' Rule we know that

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

Hence

$$d\mathbf{x} = \{\mathbf{f}(\mathbf{x}, t) - g(t)^2 [\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{y} \mid \mathbf{x})]\} dt + g(t) d\bar{\mathbf{w}}$$

↑
Time-dependent

Conditional Generation w/ Score-SDE (Song et al. 2021)

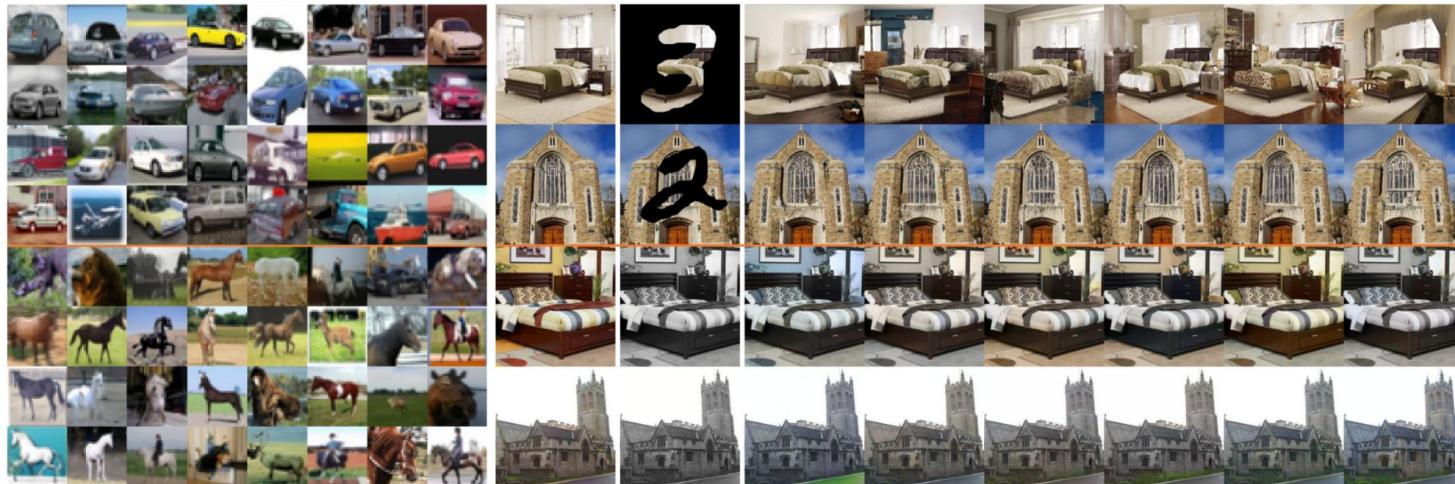
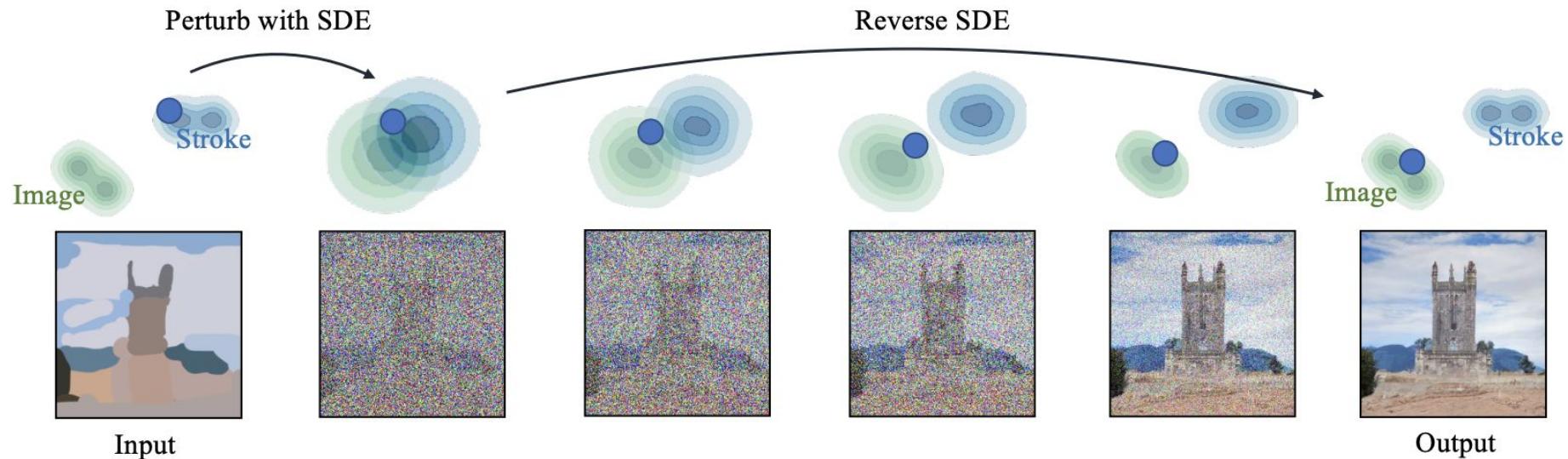
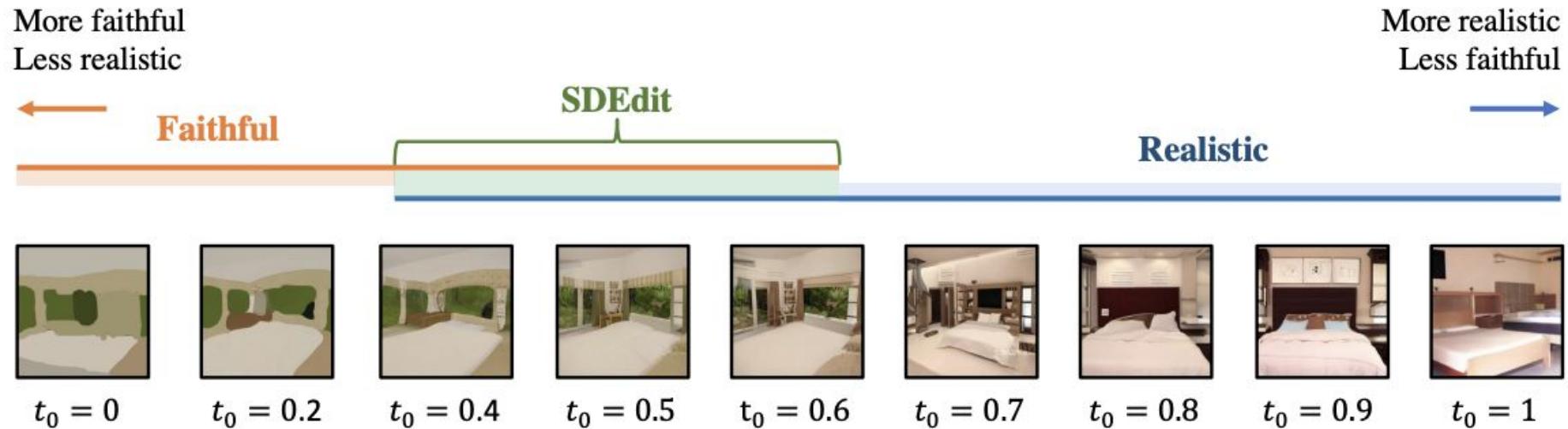


Figure 4: *Left:* Class-conditional samples on 32×32 CIFAR-10. Top four rows are automobiles and bottom four rows are horses. *Right:* Inpainting (top two rows) and colorization (bottom two rows) results on 256×256 LSUN. First column is the original image, second column is the masked/grayscale image, remaining columns are sampled image completions or colorizations.

SDEdit: Conditional Generation w/o Training (Meng et al. 2022)



SDEdit: Conditional Generation w/o Training (Meng et al. 2022)

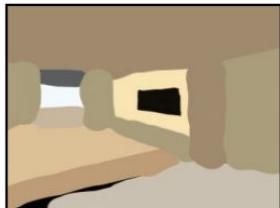


SDEdit: Conditional Generation w/o Training (Meng et al. 2022)

Stroke Painting to Image



Stroke-based Editing



Input (guide)

Output

Image Compositing



Source

Input (guide)

Output

Source

Input (guide)

Output

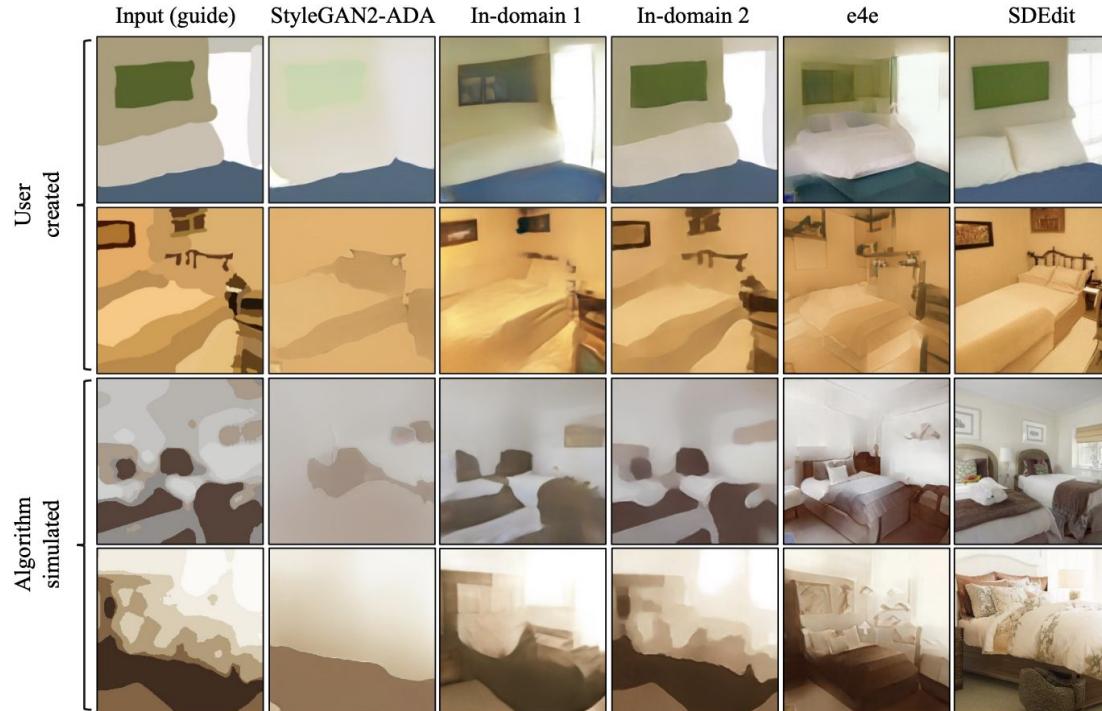


Source

Input (guide)

Output

SDEdit: Conditional Generation w/o Training (Meng et al. 2022)

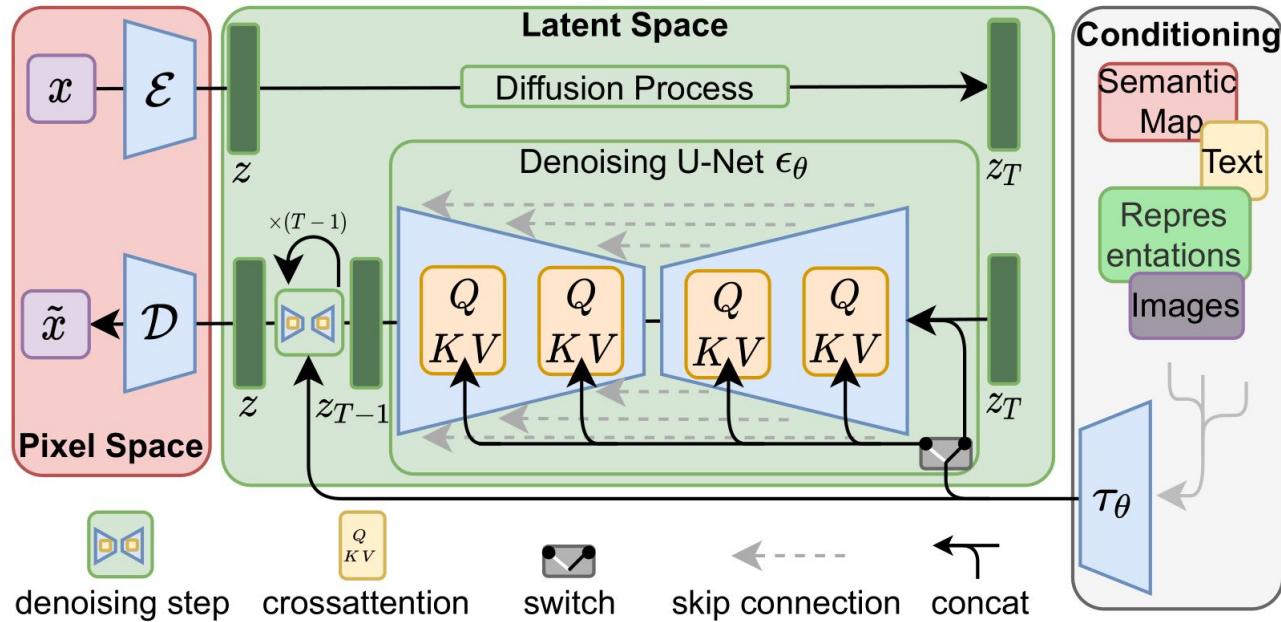


SDEdit: Conditional Generation w/o Training (Meng et al. 2022)



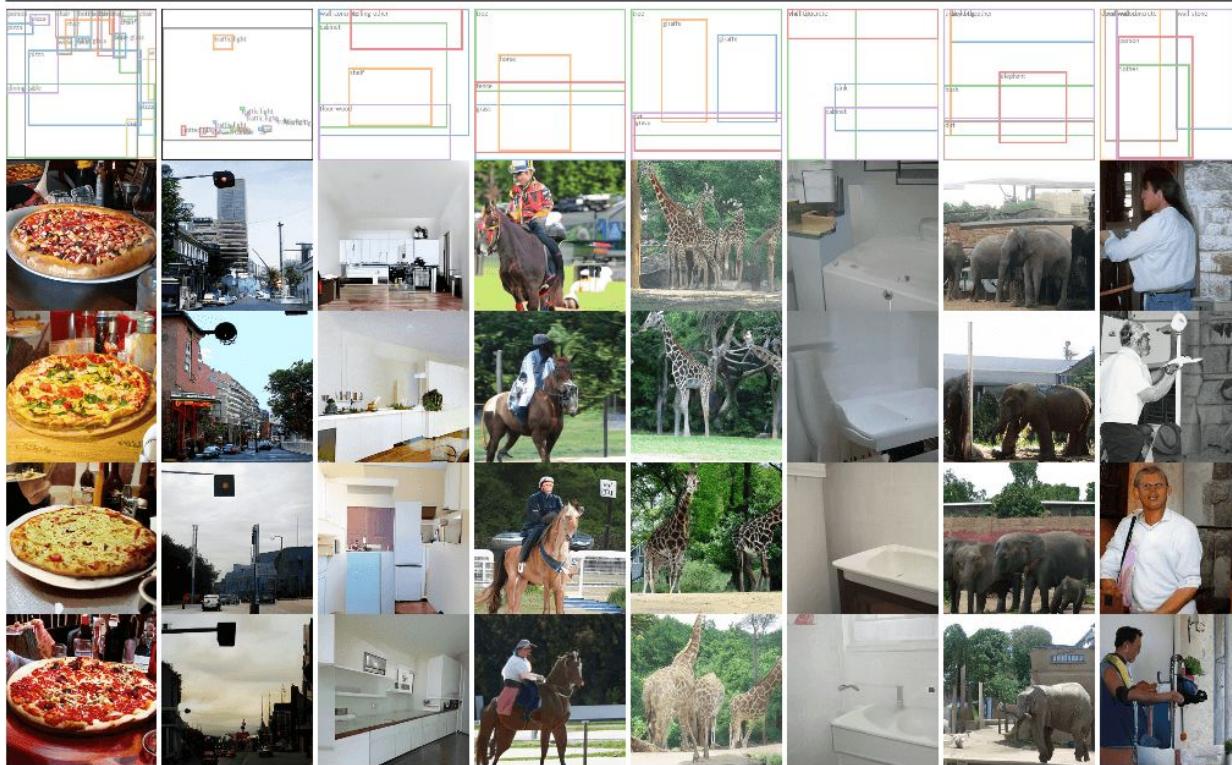
Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. "SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations". ICLR 2022. <https://arxiv.org/pdf/2108.01073.pdf>

Stable Diffusion: Conditional Generation w/ diffusion in the latent space (Rombach et al. 2022)



Stable Diffusion (Rombach et al. 2022)

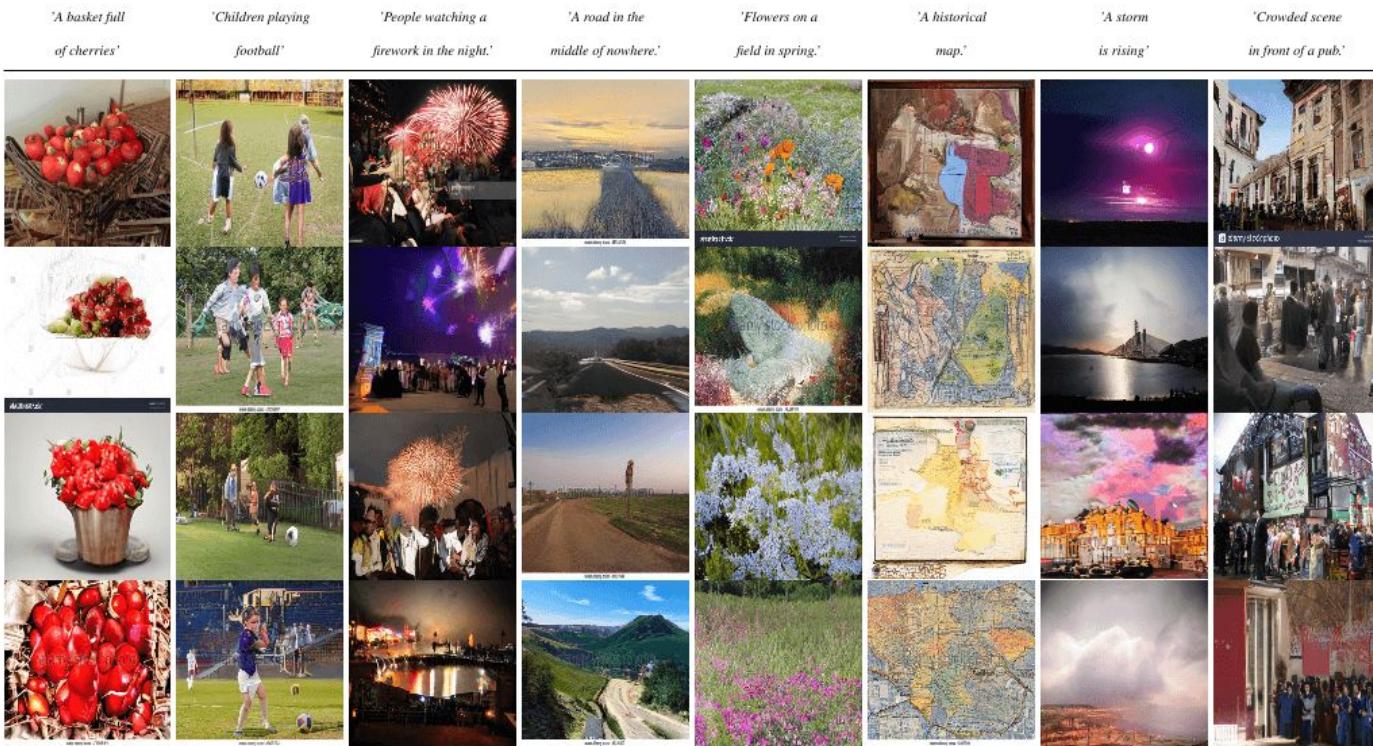
layout-to-image synthesis on the COCO dataset



Robin Rombach*, Andreas Blattmann*, Dominik Lorenz, Patrick Esser, Björn Ommer. "High-Resolution Image Synthesis with Latent Diffusion Models". CVPR 2022.
<https://arxiv.org/pdf/2112.10752.pdf>

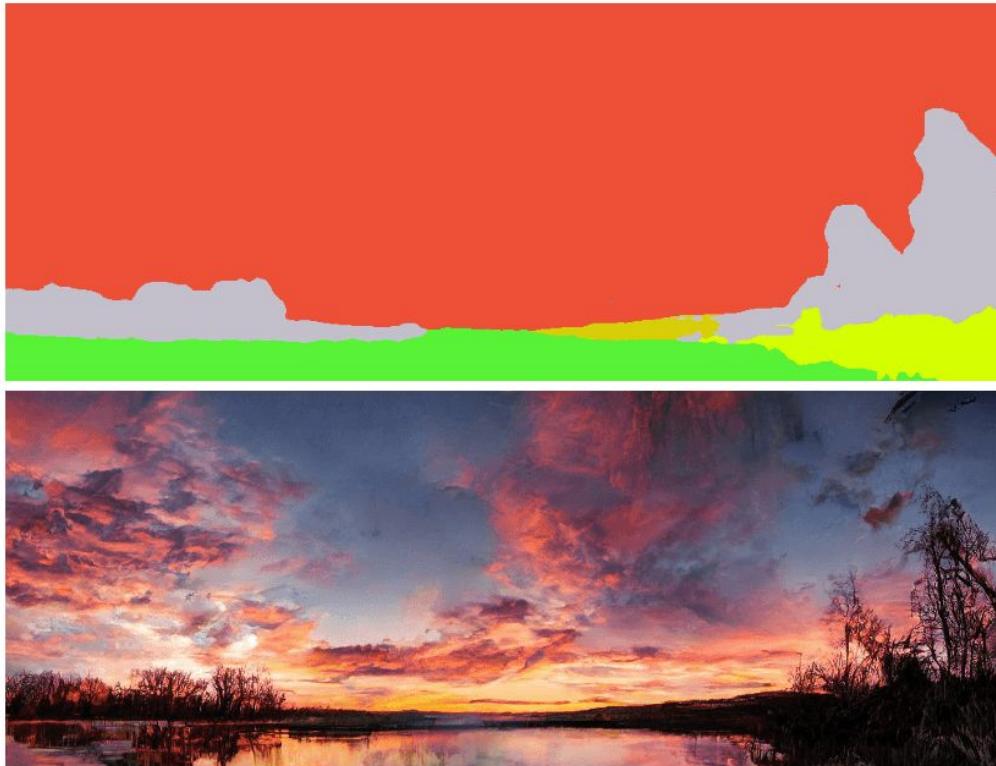
Stable Diffusion (Rombach et al. 2022)

Text-to-Image Synthesis on the Conceptual Captions dataset



Stable Diffusion (Rombach et al. 2022)

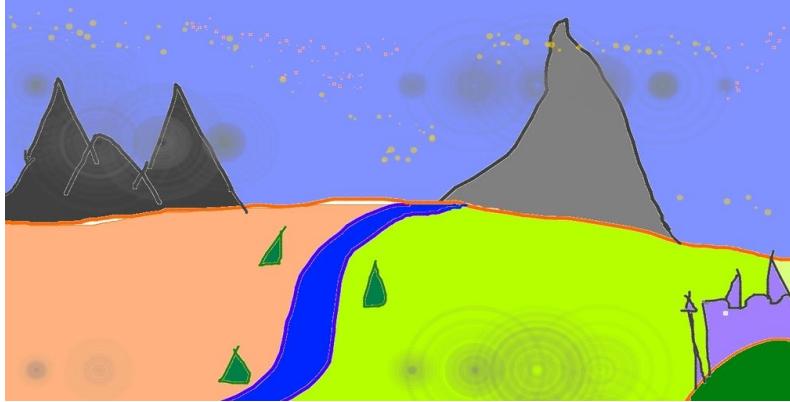
Semantic Synthesis on Flickr-Landscapes [21]



Robin Rombach*, Andreas Blattmann*, Dominik Lorenz, Patrick Esser, Björn Ommer. "High-Resolution Image Synthesis with Latent Diffusion Models". CVPR 2022.
<https://arxiv.org/pdf/2112.10752.pdf>

Stable Diffusion (Rombach et al. 2022)

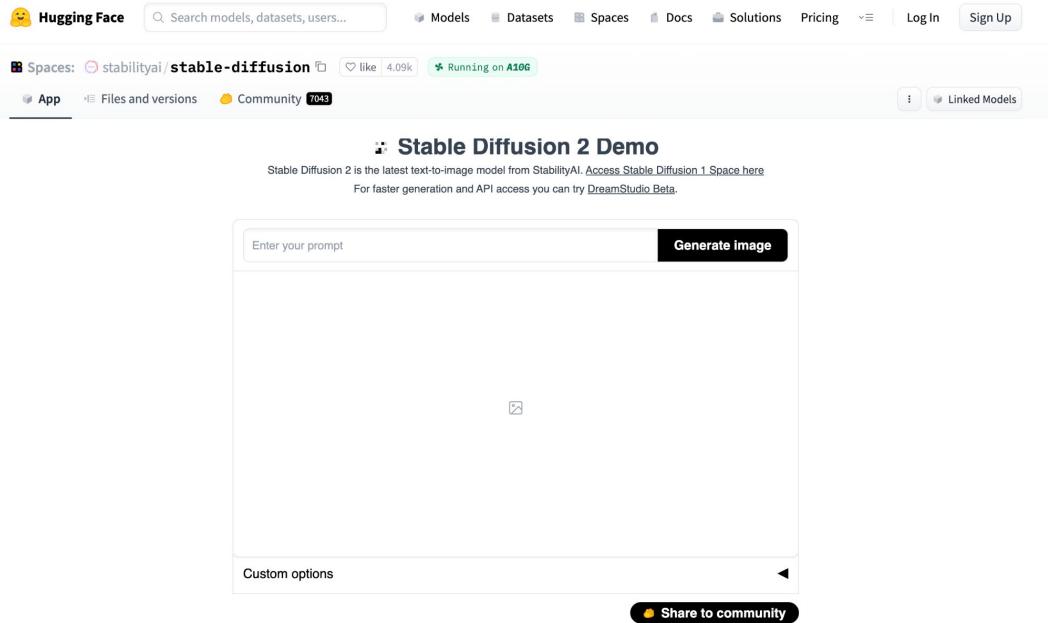
"A fantasy landscape, trending on artstation"



Uses SDEdit!

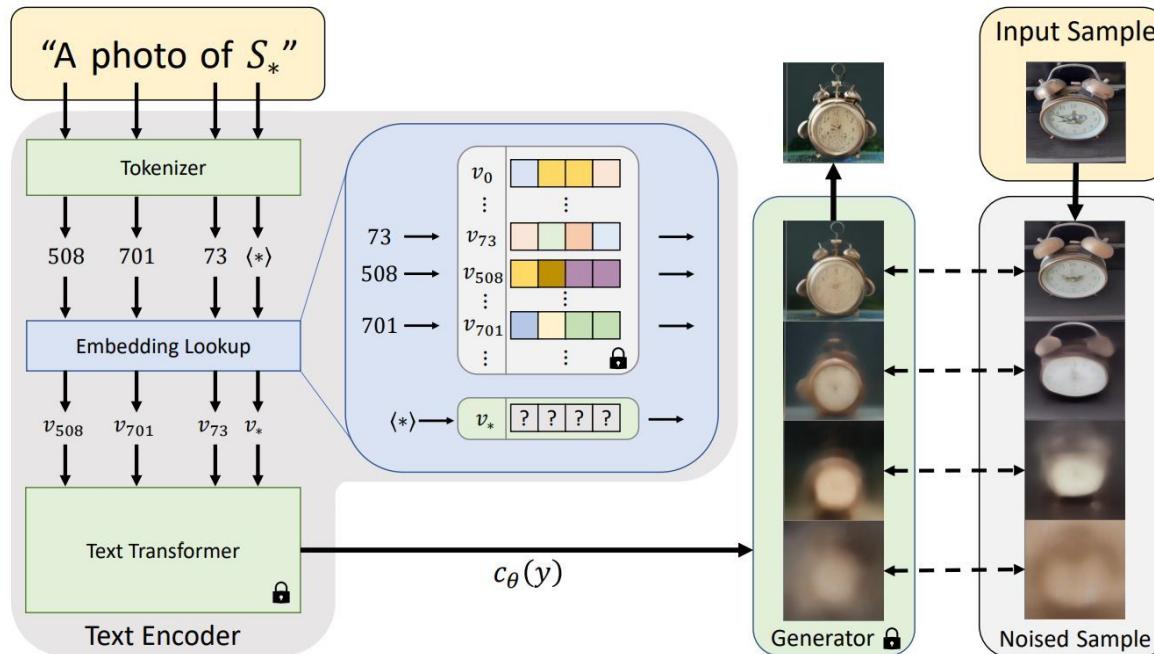
Stable Diffusion (Rombach et al. 2022)

Demo: <https://huggingface.co/spaces/stabilityai/stable-diffusion> (v2)
<https://huggingface.co/spaces/stabilityai/stable-diffusion-1> (v1)



Robin Rombach*, Andreas Blattmann*, Dominik Lorenz, Patrick Esser, Björn Ommer. "High-Resolution Image Synthesis with Latent Diffusion Models". CVPR 2022.
<https://arxiv.org/pdf/2112.10752.pdf>

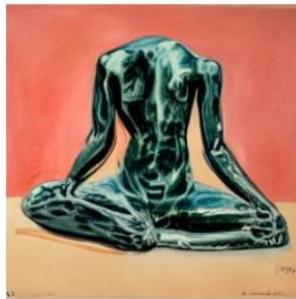
Textual Inversion (Gal et al. 2022)



Textual Inversion (Gal et al. 2022)



Input samples $\xrightarrow{\text{invert}}$ “ S_* ”



“An oil painting of S_* ”



“App icon of S_* ”



“Elmo sitting in
the same pose as S_* ”



“Crochet S_* ”



Input samples $\xrightarrow{\text{invert}}$ “ S_* ”



“Painting of two S_*
fishing on a boat”



“A S_* backpack”

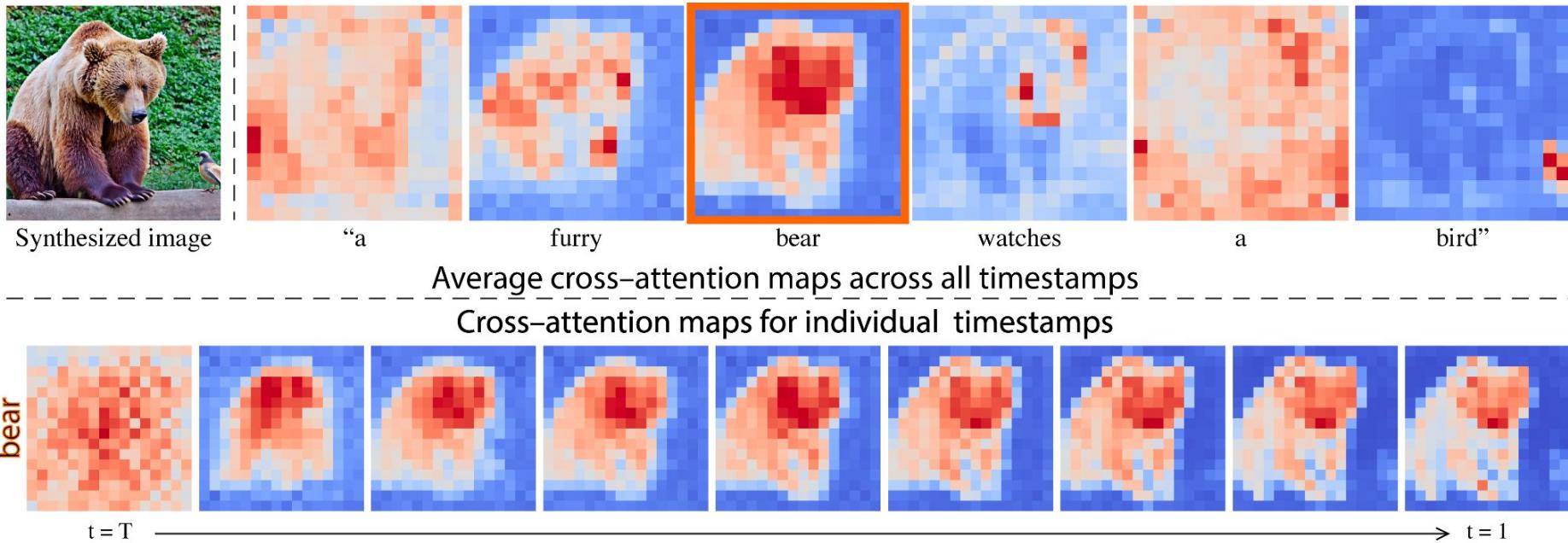


“Banksy art of S_* ”

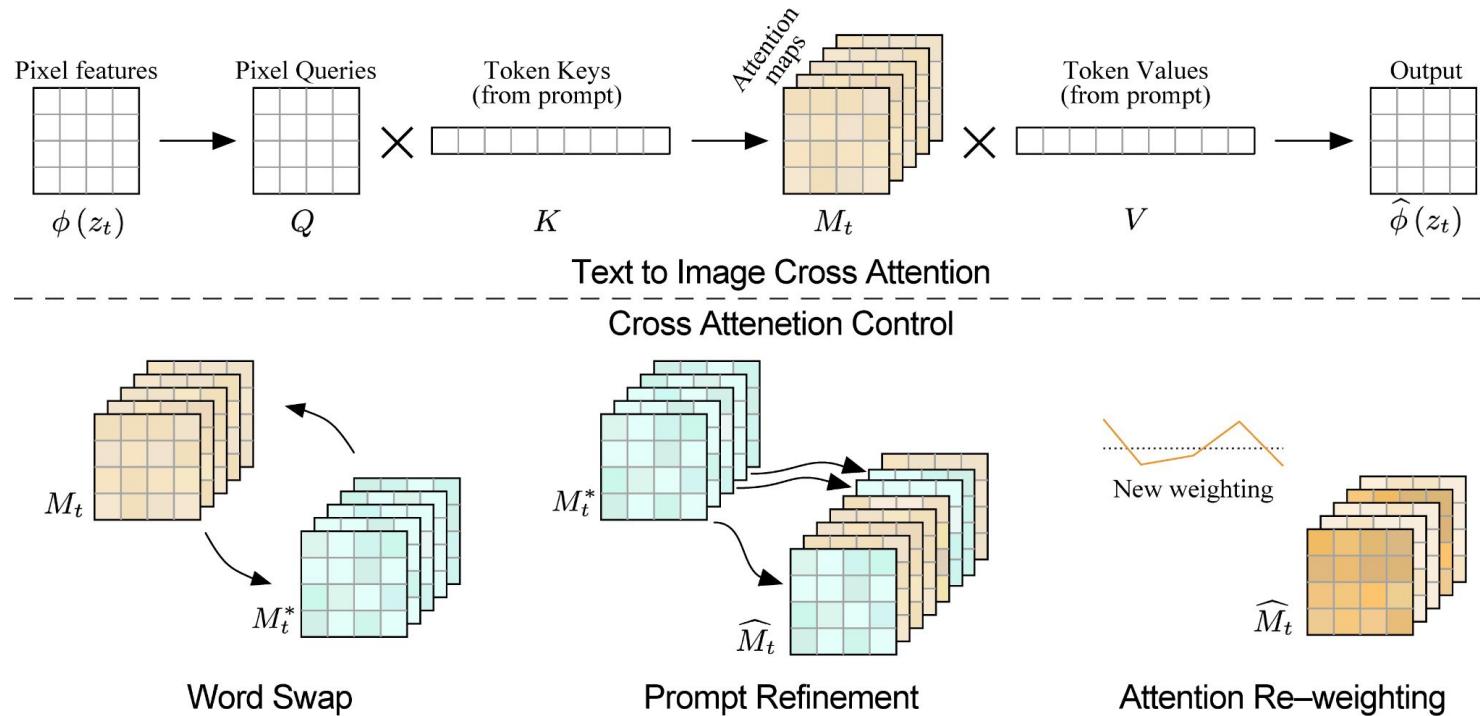


“A S_* themed lunchbox”

Prompt-to-Prompt Image Editing ([Hertz et al. 2022](#))



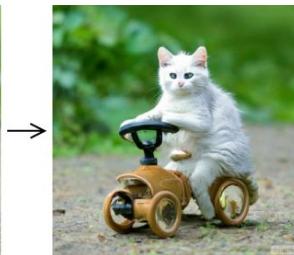
Prompt-to-Prompt Image Editing ([Hertz et al. 2022](#))

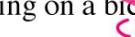


Prompt-to-Prompt Image Editing (Hertz et al. 2022)



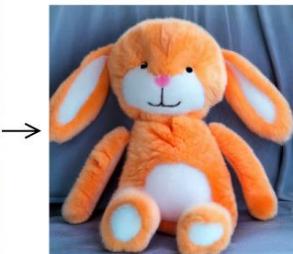
"The boulevards are crowded today."

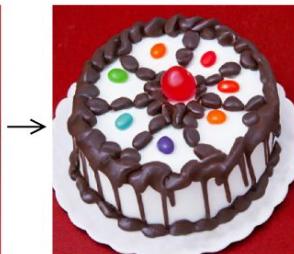
"Photo of a cat riding on a bicycle."
~~car~~




"Landscape with a house near a river
and a rainbow in the background!"

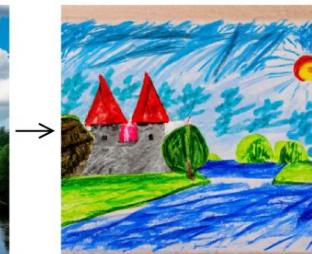



"My fluffy bunny doll."

"a cake with decorations."

jelly beans



"Children drawing of a castle next to a river."


InstructPix2Pix (Brooks et al. 2023)

Training Data Generation

(a) Generate text edits:

Input Caption: "photograph of a girl riding a horse" → GPT-3 → Instruction: "have her ride a dragon"
Edited Caption: "photograph of a girl riding a dragon"

(b) Generate paired images:

Input Caption: "photograph of a girl riding a horse"
Edited Caption: "photograph of a girl riding a dragon" → Stable Diffusion + Prompt2Prompt → 

(c) Generated training examples:



InstructPix2Pix (Brooks et al. 2023)

Training Data Generation

(a) Generate text edits:

Input Caption: "photograph of a girl riding a horse" → GPT-3 → Instruction: "have her ride a dragon"
Edited Caption: "photograph of a girl riding a dragon"

(b) Generate paired images:

Input Caption: "photograph of a girl riding a horse" → Stable Diffusion + Prompt2Prompt → Edited Caption: "photograph of a girl riding a dragon"

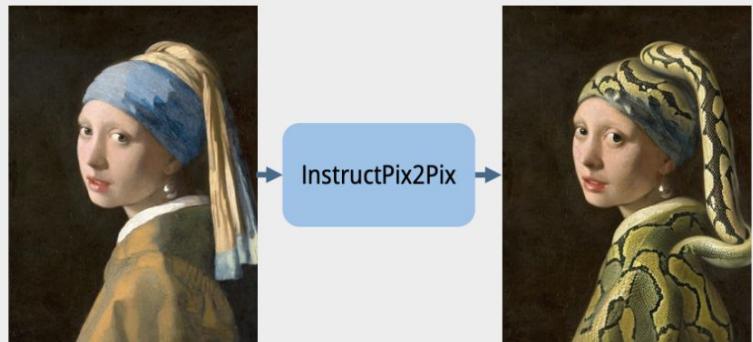
(c) Generated training examples:



Instruction-following Diffusion Model

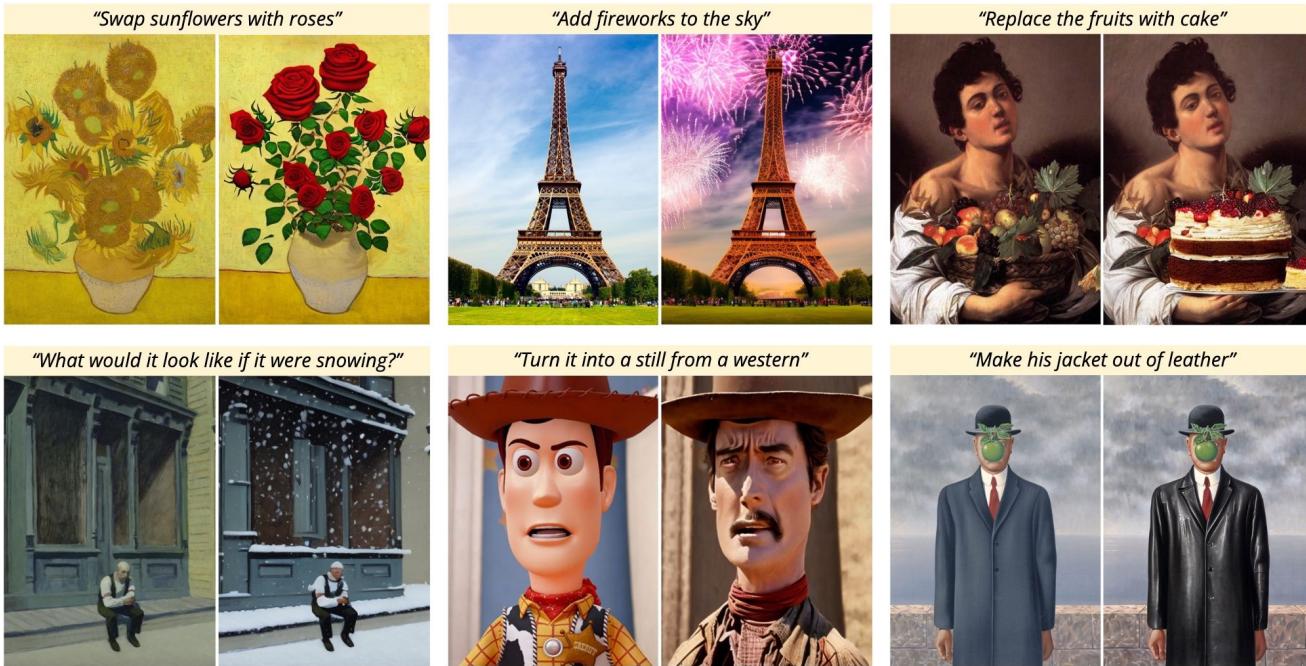
(d) Inference on real images:

"turn her into a snake lady"



InstructPix2Pix (Brooks et al. 2023)

Demo



Tim Brooks, Aleksander Holynski, Alexei A. Efros. "InstructPix2Pix: Learning to Follow Image Editing Instructions". CVPR 2023.
<https://www.timothybrooks.com/instruct-pix2pix>

Does Stable Diffusion solve it all? Are there any other applications of diffusion models we can think of?



Questions I have been thinking about

- How to best incorporate auxiliary information in conditional generation of diffusion models so that it minimizes the additional training
 - How can we incorporate pretrained non-time-dependent models in conditional generation of diffusion models (in the data space)?
- Instead of the latent space, can we learn a diffusion model in the function space?
 - And since we already know how to represent data in the function space, can we in turn (better) sample data from this space?
 - Also denoising diffusion really looks like SGD, what is their relationship?

Other Fun Papers

- Faster Sampling
 - [DDIM](#)
 - [Diffusion Distillation](#)
- Guided Diffusion
 - [Classifier-guided diffusion](#)
 - [Classifier-free diffusion guidance](#)
 - [Prompt-to-Prompt for real Image Editing](#)
- Diffusion in the latent space
 - [D2C](#)
- Other Modalities
 - [Video](#)
 - [Point cloud](#)
 - [Music](#)

Resources

- Lil' Log. "What are Diffusion Models?"
<https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>
- Yang Song. "Generative Modeling by Estimating Gradients of the Data Distribution". <https://yang-song.net/blog/2021/score/>
- Niels Rogge and Kashif Rasul. "The Annotated Diffusion Model".
<https://huggingface.co/blog/annotated-diffusion>
- My email: yutonghe@andrew.cmu.edu

My website: <https://kellyyutonghe.github.io/>