
Diffusion Models

— Yutong (Kelly) He 10/29/25 —

Many figures adopted from Yang Song's ["Generative Modeling by Estimating Gradients of the Data Distribution"](#) tutorial & the lecture slides of CMU 10708 by Prof. Andrej Risteski, Prof. Albert Gu & Stanford CS 236 by Prof. Stefano Ermon



About Me



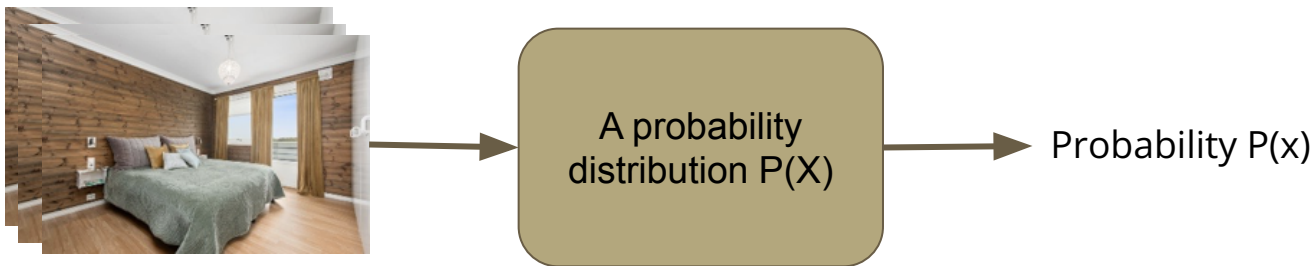
- Yutong (Kelly) He, 4th year PhD in MLD advised by Zico & Russ
- Main things I have done in the past
 - H-Divergence: Comparing Distributions by Measuring Differences that Affect Decision Making [[paper](#)]
 - Generative Modeling:
 - Image-to-Image Translation [[paper](#)]
 - Super-Resolution [[paper](#)]
 - ✓ SDEdit: Guided Image Synthesis and Editing with *Stochastic Differential Equations* [[paper](#)]
 - ✓ CAC: Localized Text-to-Image Generation for Free via Cross Attention Control [[paper](#)]
 - ✓ MPGD: Manifold Preserving Guided *Diffusion* [[paper](#)]
 - Black-box Prompt Engineering for Personalized Text-to-Image Generation [[paper](#)]
 - ✓ LADiBI: Blind Inverse Problem Solving Made Easy by Text-to-Image Latent *Diffusion* [[paper](#)]
 - Diffusion applications in reinforcement learning [[paper](#)]
 - Diffusion/flow-based distillation methods [[paper](#)]

Generative Modeling

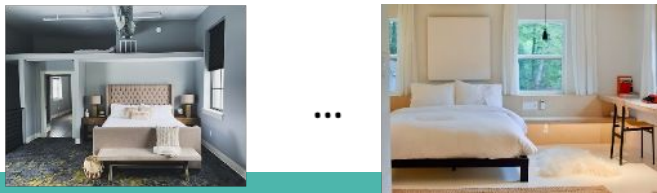


A (statistical) generative model is a **probability distribution $P(X)$** (as opposed to $P(Y|X)$ in discriminative models)

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

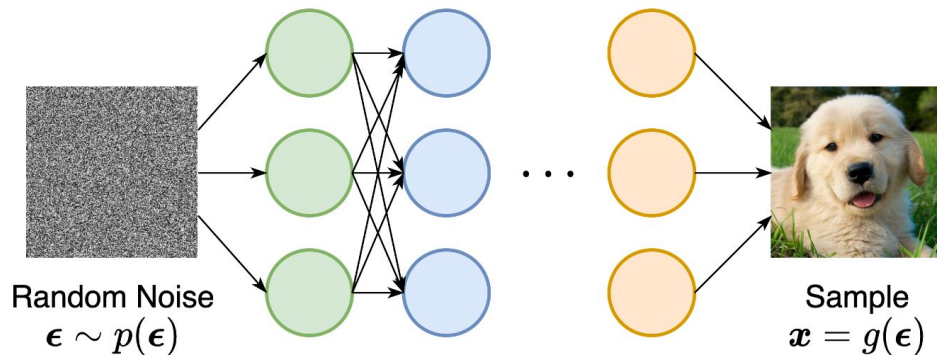


It is generative because sampling from $p(x)$ generates new images



Generative Models

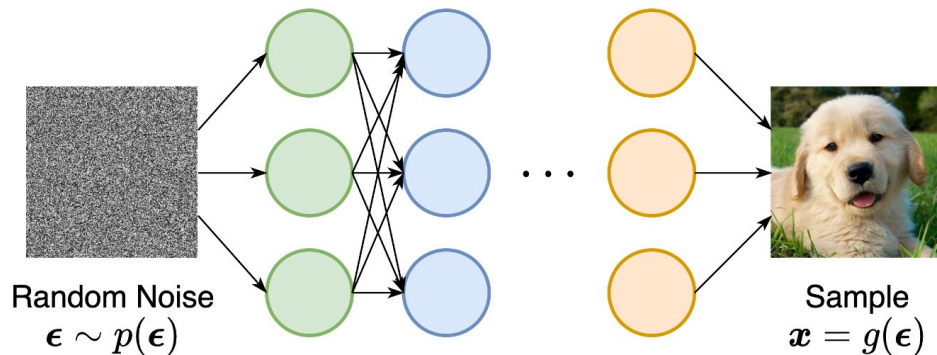
- **Likelihood Based:** Autoregressive models, variational autoencoders (VAE), normalizing flow, energy-based models (EBM)
- **Likelihood Free:** Generative adversarial networks (GAN)



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

Generative Models

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



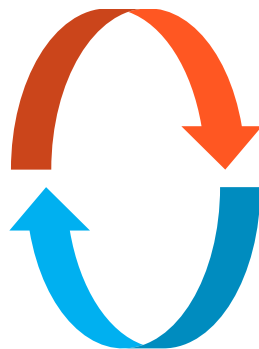
Directly sampling from $P(X)$ is usually hard because they are usually complicated!
But **sampling from a simpler distribution** (eg. a Gaussian) is easy!

GAN (Goodfellow et. al 2014) Overview

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

Generator

Tries to make the fake samples more and more realistic so that it can fool the discriminator

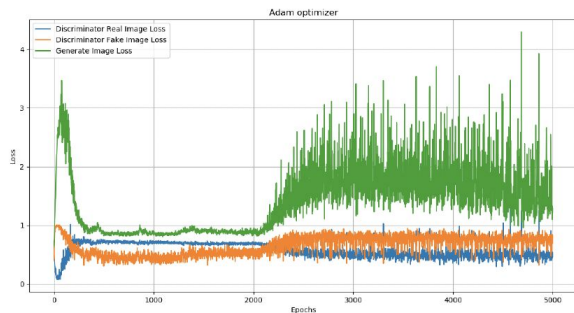


Discriminator

Tries to be better and better at distinguishing fake samples from real ones

However...

- **Theorem (informal):** If the generator updates are made in function space and discriminator is optimal at every step, then the generator is guaranteed to converge to the data distribution
- **Unrealistic assumptions!**
- In practice, the generator and discriminator loss keeps oscillating during GAN training

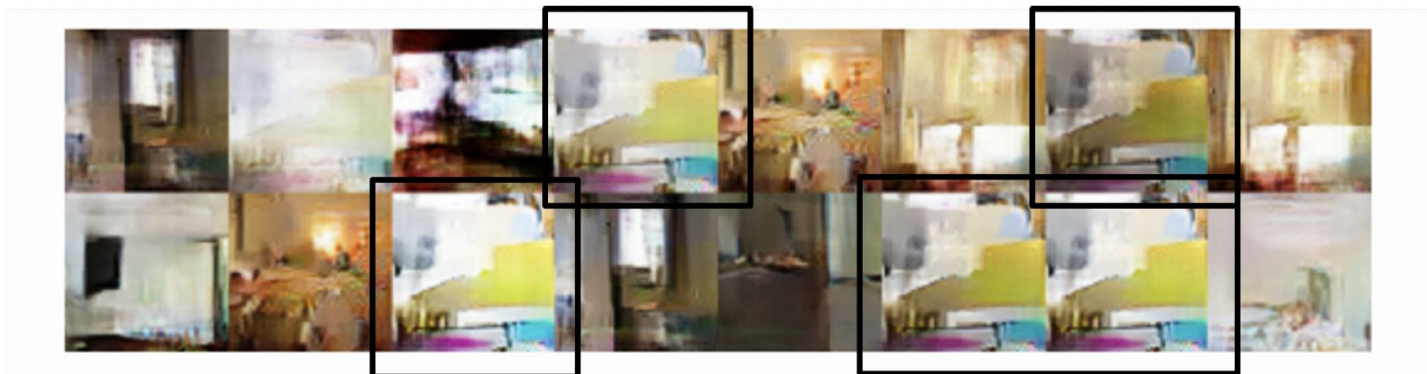


Source: [Mirantha Jayathilaka](#)

- No robust stopping criteria in practice (unlike MLE)

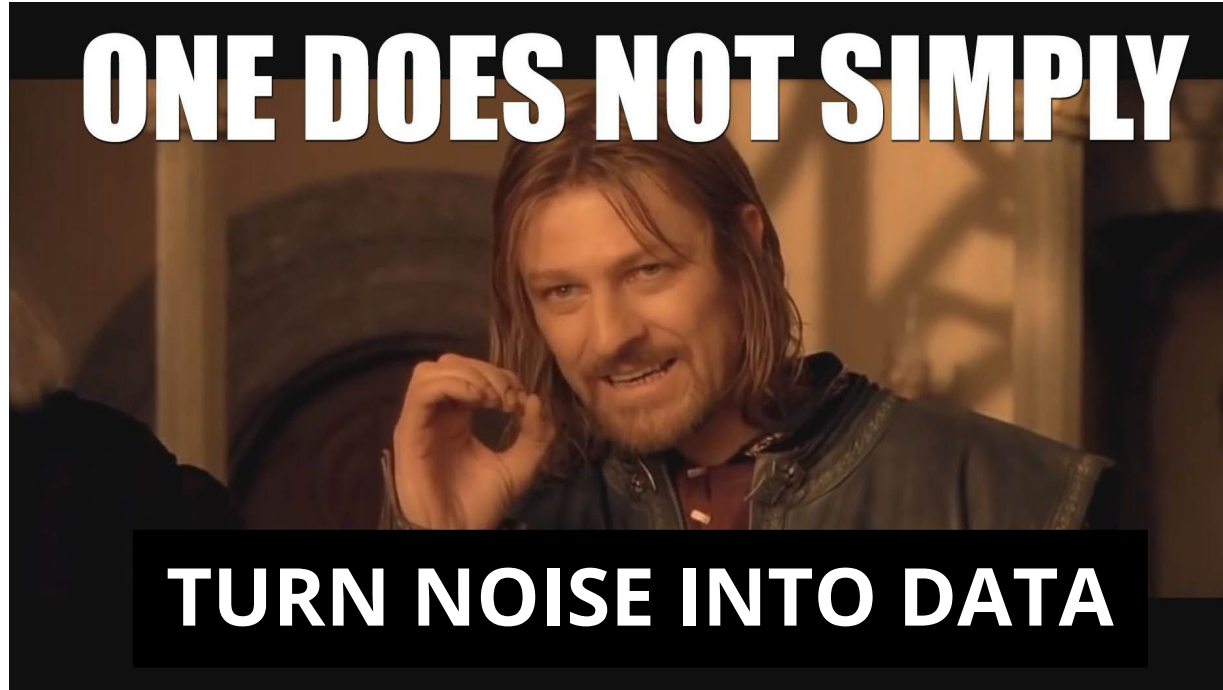
However...

- GANs are notorious for suffering from **mode collapse**
- Intuitively, this refers to the phenomena where the generator of a GAN collapses to one or few samples (dubbed as “modes”)



Arjovsky et al., 2017

From Noise to Data

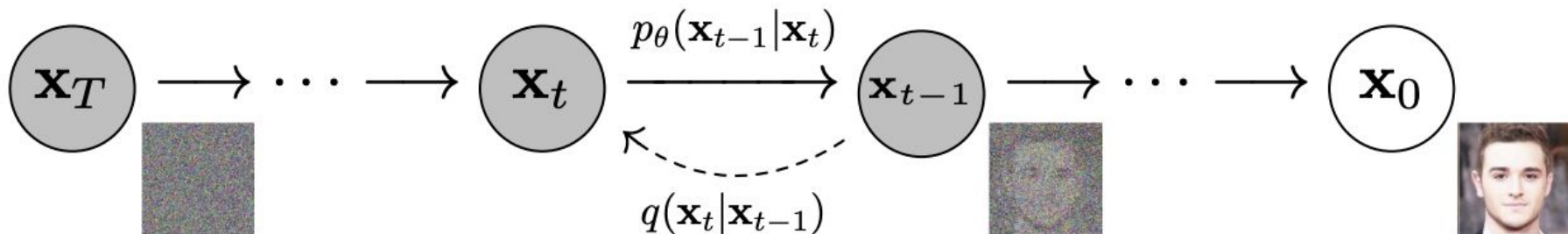


From Noise to Data



Now you get **Diffusion Model!**

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



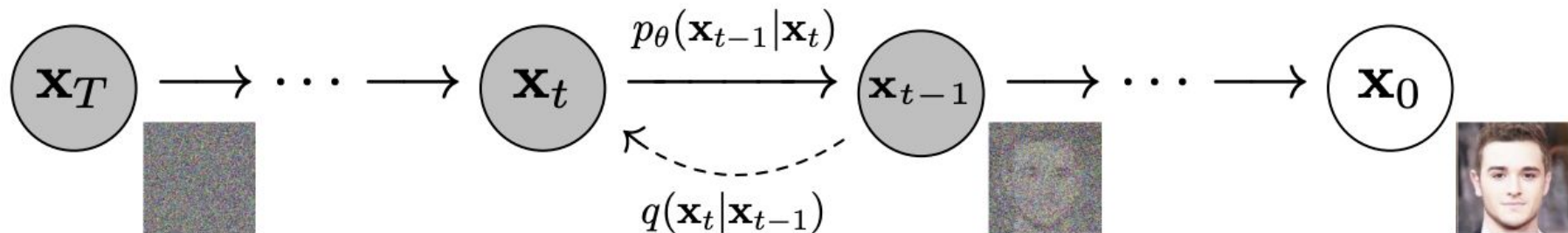
Forward (Adding Noise):

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

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:

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

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

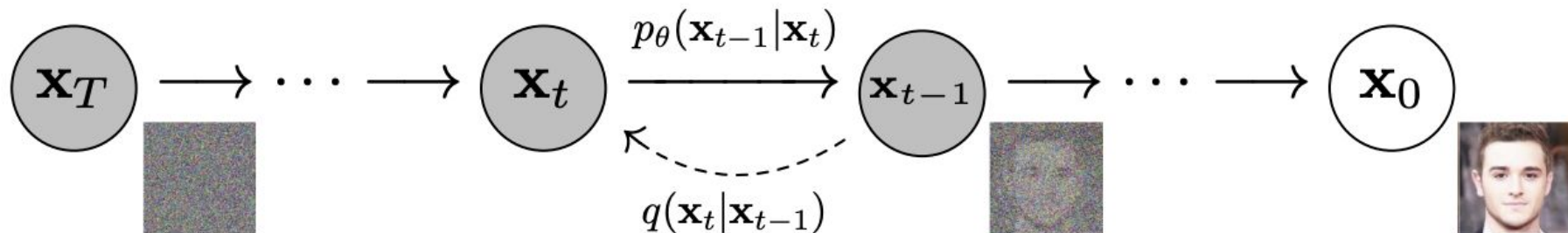
Diffusion Model (Sohl-Dickstein, et al. 2015)



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: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 5: Take gradient descent step on
$$\nabla_{\theta} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2$$
 - 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: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
 - 5: **end for**
 - 6: **return** \mathbf{x}_0
-

Denoising Diffusion Probabilistic Model (DDPM)

(Ho, et al. 2020)

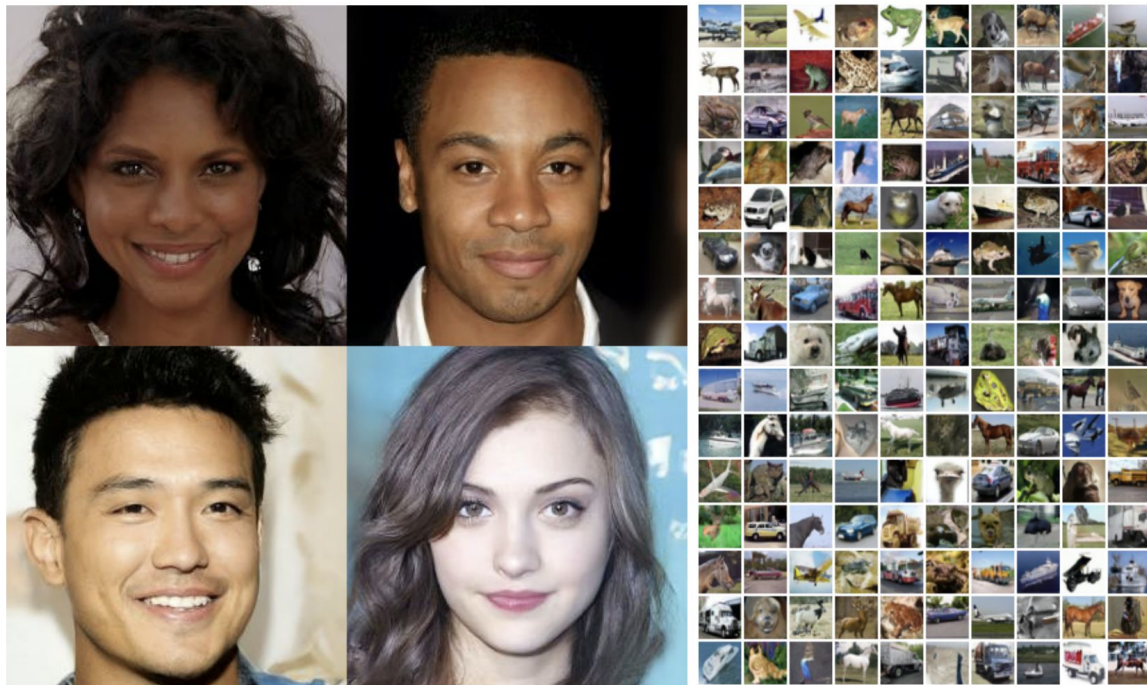
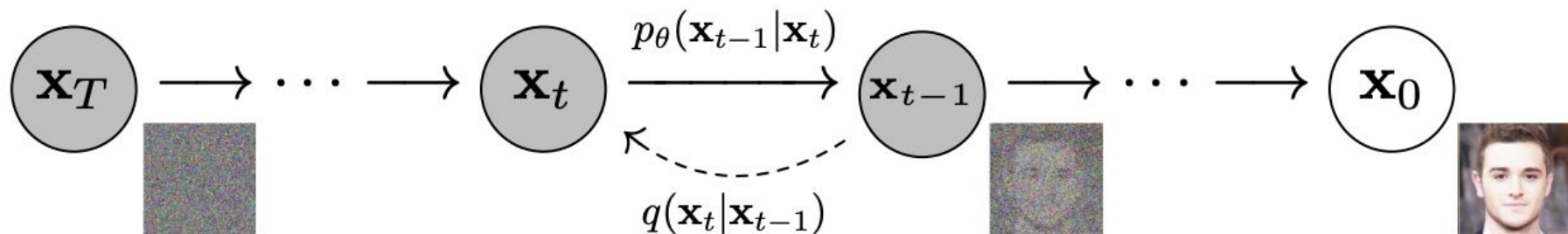


Figure 1: Generated samples on CelebA-HQ 256×256 (left) and unconditional CIFAR10 (right)

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 $P(\mathbf{X})$** (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 other, can also take a long time and can't generate everything all at once



But ...

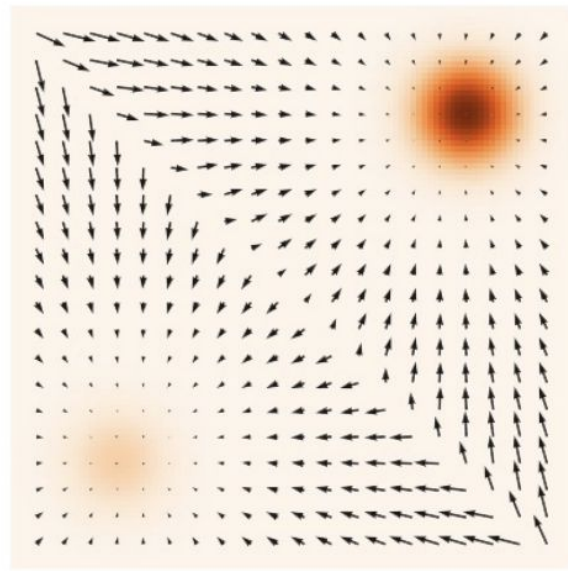
Do we really need to estimate $p_{\theta}(\mathbf{x})$ in order to sample from the model?

- Idea: Stochastic **gradient** descent (or ascent) with maximum **log likelihood** in the data space

- 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

How to Train with Score Matching

But how do we get this ground truth $\nabla_{\mathbf{x}} \log p(\mathbf{x})$?

Turns out you don't have to!

We will rewrite the loss slightly:

$$\begin{aligned} & \mathbb{E}_{p_{data}} \|\nabla_x \log p_{\theta}(x) - \nabla_x \log p_{data}(x)\|^2 \\ &= \mathbb{E}_{p_{data}} \|\nabla_x \log p_{\theta}(x)\|^2 + \mathbb{E}_{p_{data}} \|\nabla_x \log p_{data}(x)\|^2 \\ & \quad - 2\mathbb{E}_{p_{data}} \langle \nabla_x \log p_{\theta}(x), \nabla_x \log p_{data}(x) \rangle \end{aligned}$$

$$\|x - y\|^2 = \langle x - y, x - y \rangle = \langle x, x \rangle + \langle y, y \rangle - 2\langle x, y \rangle$$

How to Train with Score Matching

We will rewrite the loss slightly:

$$\begin{aligned} & \mathbb{E}_{p_{data}} \|\nabla_x \log p_{\theta}(x) - \nabla_x \log p_{data}(x)\|^2 \\ = & \mathbb{E}_{p_{data}} \|\nabla_x \log p_{\theta}(x)\|^2 + \cancel{\mathbb{E}_{p_{data}} \|\nabla_x \log p_{data}(x)\|^2} \\ & - 2\mathbb{E}_{p_{data}} \langle \nabla_x \log p_{\theta}(x), \nabla_x \log p_{data}(x) \rangle \end{aligned}$$

Main trick: integration by parts (vector version):

$$\mathbb{E}_p \langle f(x), \nabla_x \log p(x) \rangle = -\mathbb{E}_p [\operatorname{div} f(x)], \text{ where}$$

$$\operatorname{div} f(x) = \sum_i \frac{\partial}{\partial x_i} f_i(x)$$

How to Train with Score Matching

We will rewrite the loss slightly:

$$\begin{aligned} & \mathbb{E}_{p_{data}} \|\nabla_x \log p_{\theta}(x) - \nabla_x \log p_{data}(x)\|^2 \\ &= \mathbb{E}_{p_{data}} \|\nabla_x \log p_{\theta}(x)\|^2 + \mathbb{E}_{p_{data}} \|\nabla_x \log p_{data}(x)\|^2 \\ &\quad - 2\mathbb{E}_{p_{data}} \langle \nabla_x \log p_{\theta}(x), \nabla_x \log p_{data}(x) \rangle \end{aligned}$$

Integrating by parts, the third term becomes,

$$-2\mathbb{E}_{p_{data}} [\operatorname{div} \nabla_x \log p_{\theta}(x)] = 2\mathbb{E}_{p_{data}} [\operatorname{Tr}(\nabla_x^2 \log p_{\theta}(x))]$$


So, the loss can be equivalently written as:

$$\mathbb{E}_{p_{data}} \|\nabla_x \log p_{\theta}(x)\|^2 + 2\mathbb{E}_{p_{data}} [\operatorname{Tr}(\nabla_x^2 \log p_{\theta}(x))]$$

Score Matching Loss

The training loss then becomes:

Jacobian of s_θ


$$\frac{1}{N} \sum_{\text{training data } x_i} \|s_\theta(x_i)\|^2 + 2 \left[\text{Tr}(Ds_\theta(x_i)) \right]$$


However ...

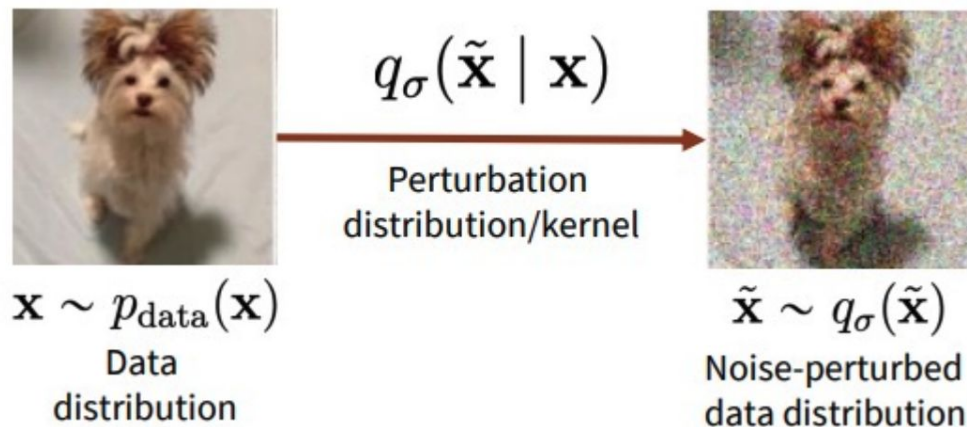
This Jacobian is super computationally expensive to calculate!

The training loss then becomes:

Jacobian of s_θ

$$\frac{1}{N} \sum_{\text{training data } x_i} \|s_\theta(x_i)\|^2 + 2 [\text{Tr}(Ds_\theta(x_i))]$$


Denoising Score Matching



Now the objective has become

$$\frac{1}{2} \mathbb{E}_{q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x})p_{\text{data}}(\mathbf{x})} [\|\mathbf{s}_{\theta}(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_{\sigma}(\tilde{\mathbf{x}} | \mathbf{x})\|_2^2].$$

Denoising Score Matching

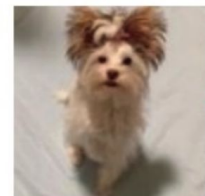
Now the objective has become

$$\frac{1}{2} \mathbb{E}_{q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x})p_{\text{data}}(\mathbf{x})} [\|\mathbf{s}_{\theta}(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_{\sigma}(\tilde{\mathbf{x}} | \mathbf{x})\|_2^2].$$

If $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$, then

$$q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{(2\pi)^{d/2} \sigma^d} e^{-\frac{1}{2\sigma^2} \|\tilde{\mathbf{x}} - \mathbf{x}\|^2}$$

$$\nabla_{\tilde{\mathbf{x}}} \log q_{\sigma}(\tilde{\mathbf{x}} | \mathbf{x})\|_2^2 = \frac{1}{\sigma^2} (\mathbf{x} - \tilde{\mathbf{x}})$$



$\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$
Data
distribution

$$q_{\sigma}(\tilde{\mathbf{x}} | \mathbf{x})$$

Perturbation
distribution/kernel



$\tilde{\mathbf{x}} \sim q_{\sigma}(\tilde{\mathbf{x}})$
Noise-perturbed
data distribution

Sampling with Langevin Dynamics

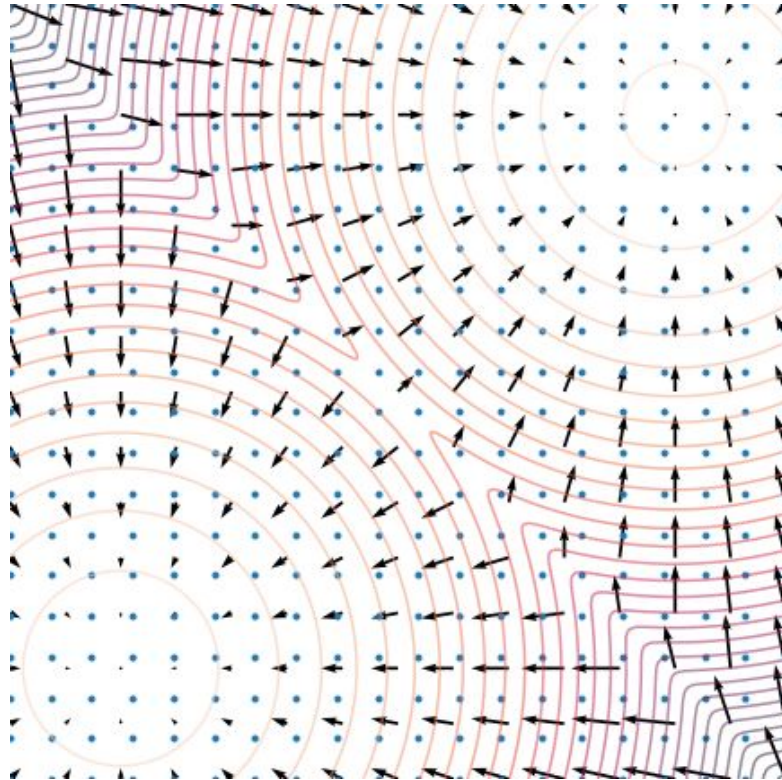
Once we have a trained score model, we can do “gradient ascent” in the data space to get a sample.

- 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

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

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

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

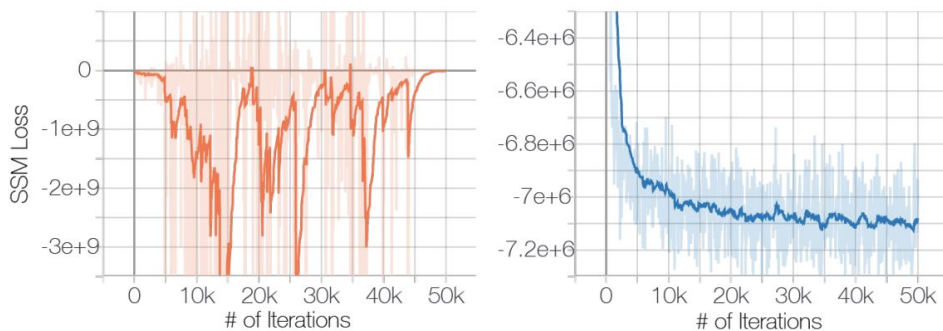
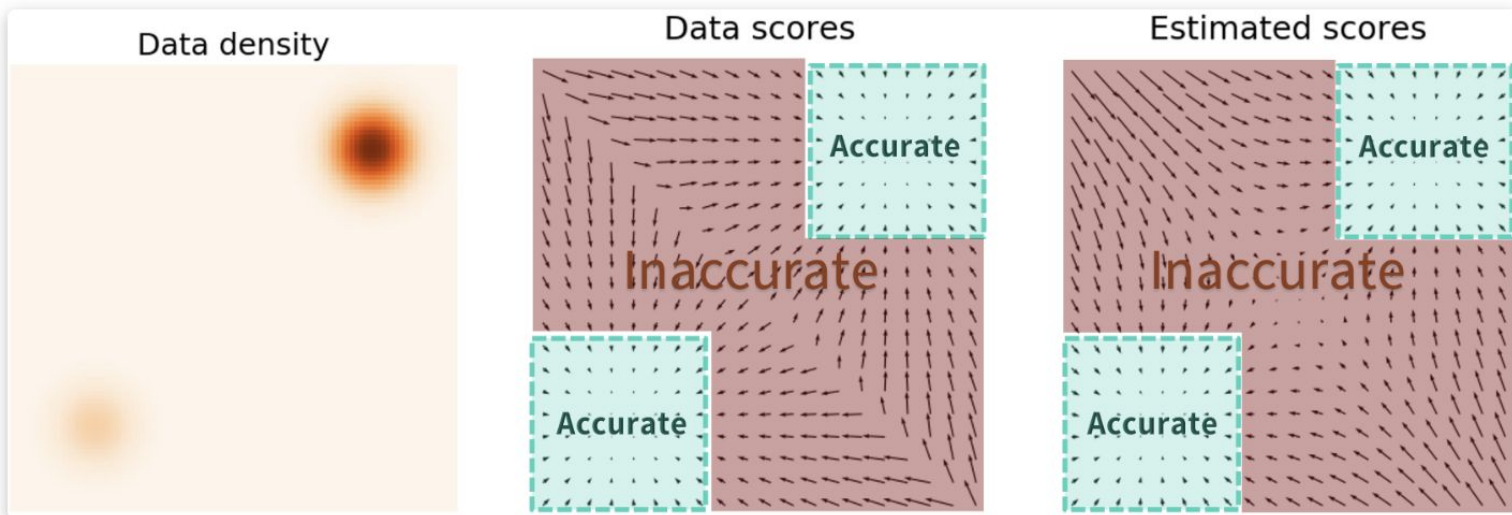


Figure 1: Left: Sliced score matching (SSM) loss w.r.t. iterations. No noise is added to data. **Right:** Same but data are perturbed with $\mathcal{N}(0, 0.0001)$.

However (2) ...

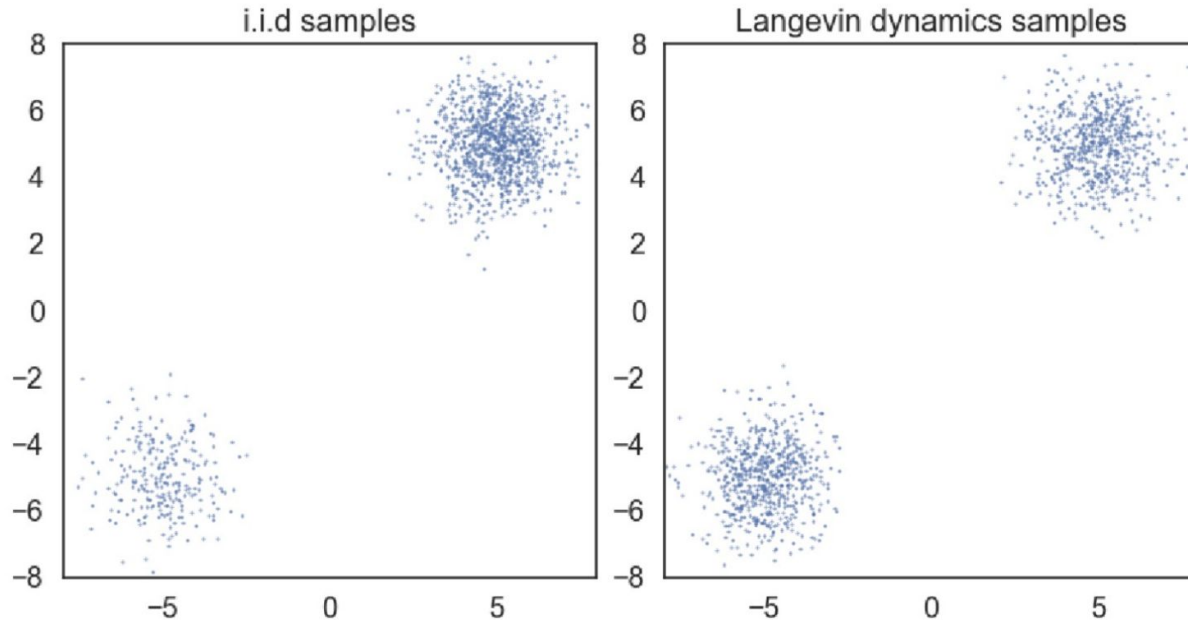
- Even when we do have full support data space, the score estimation is inaccurate in the low density regions

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



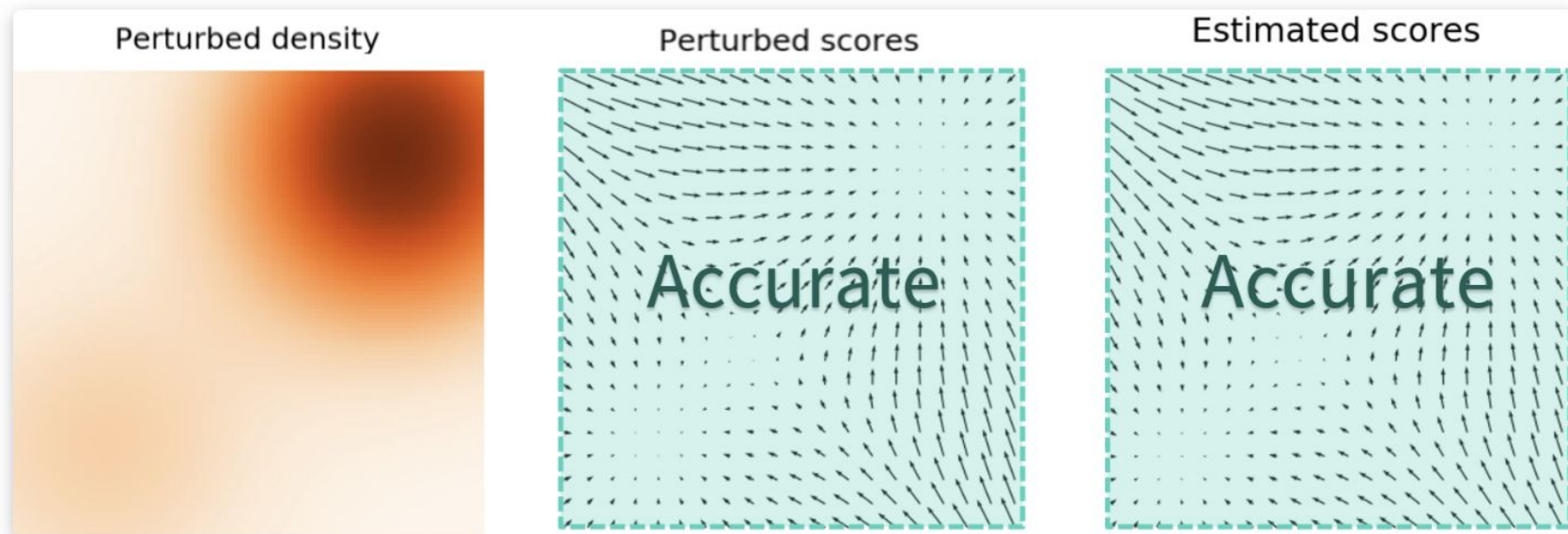
However (3) ...

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

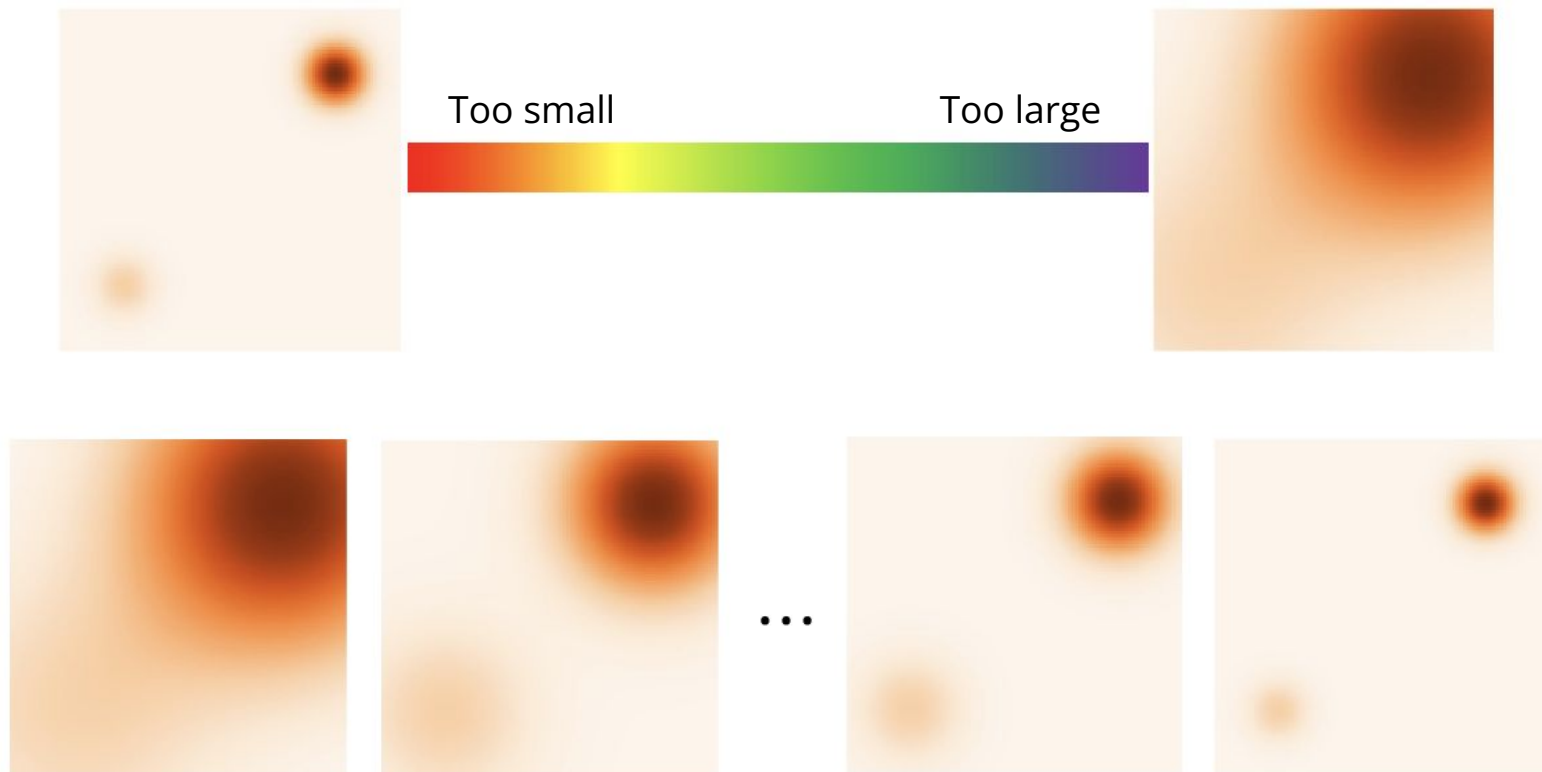


To solve all these problems

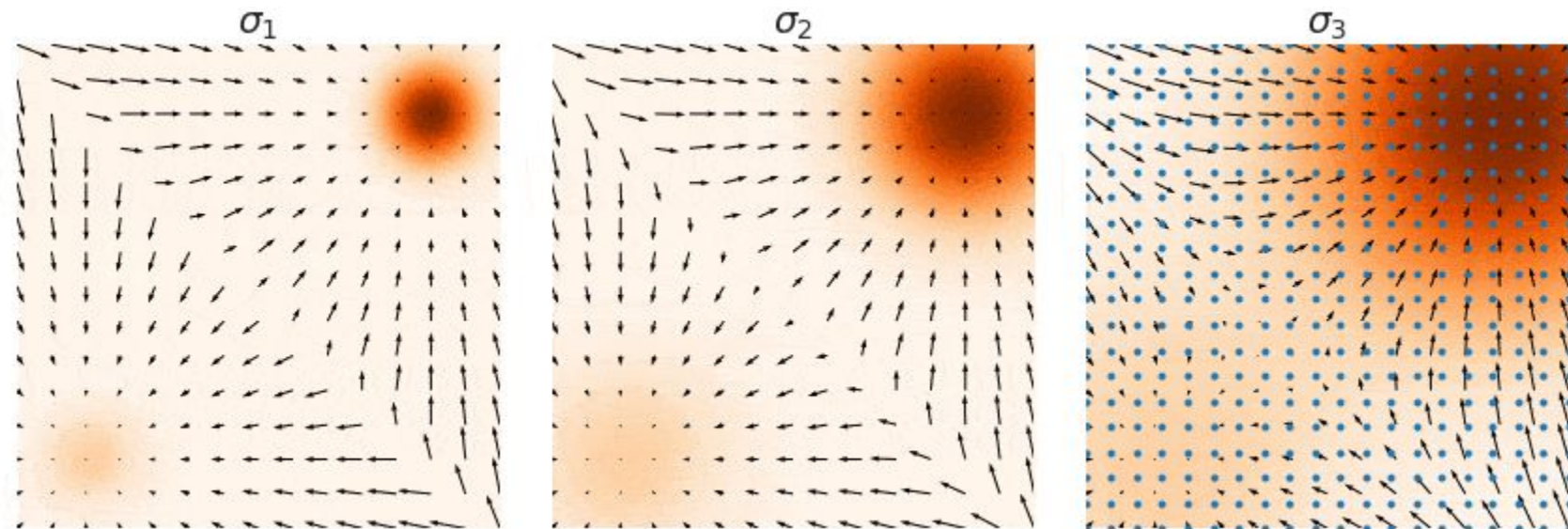
How about just **add noise** to the data?



But how much noise we should add?



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: $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i} \mathbf{z}_t$

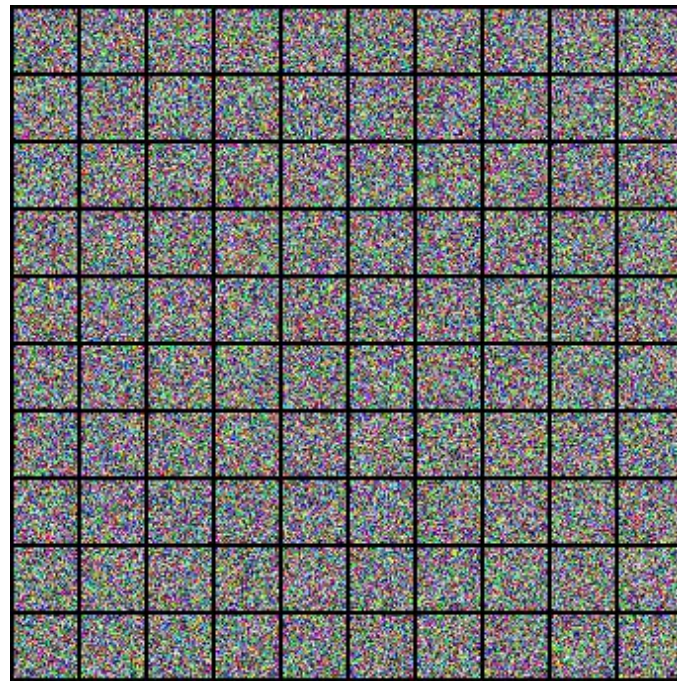
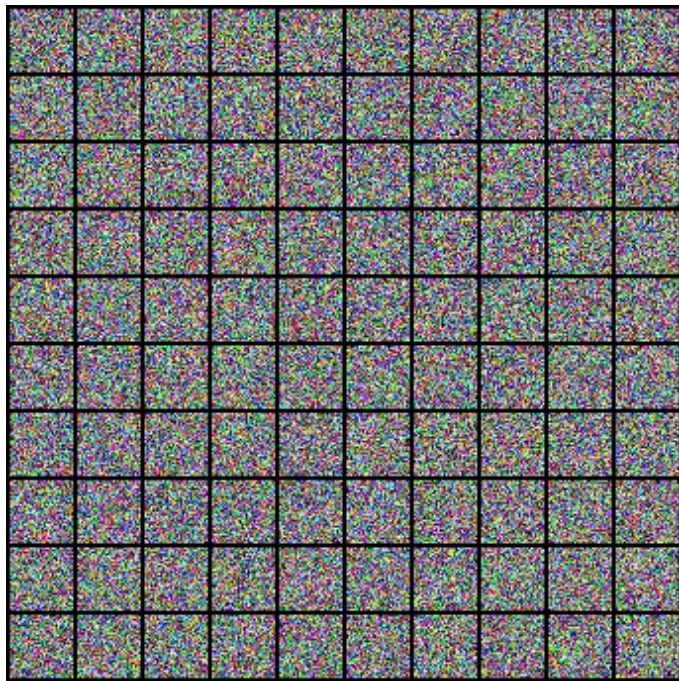
7: **end for**

8: $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$

9: **end for**

return $\tilde{\mathbf{x}}_T$

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

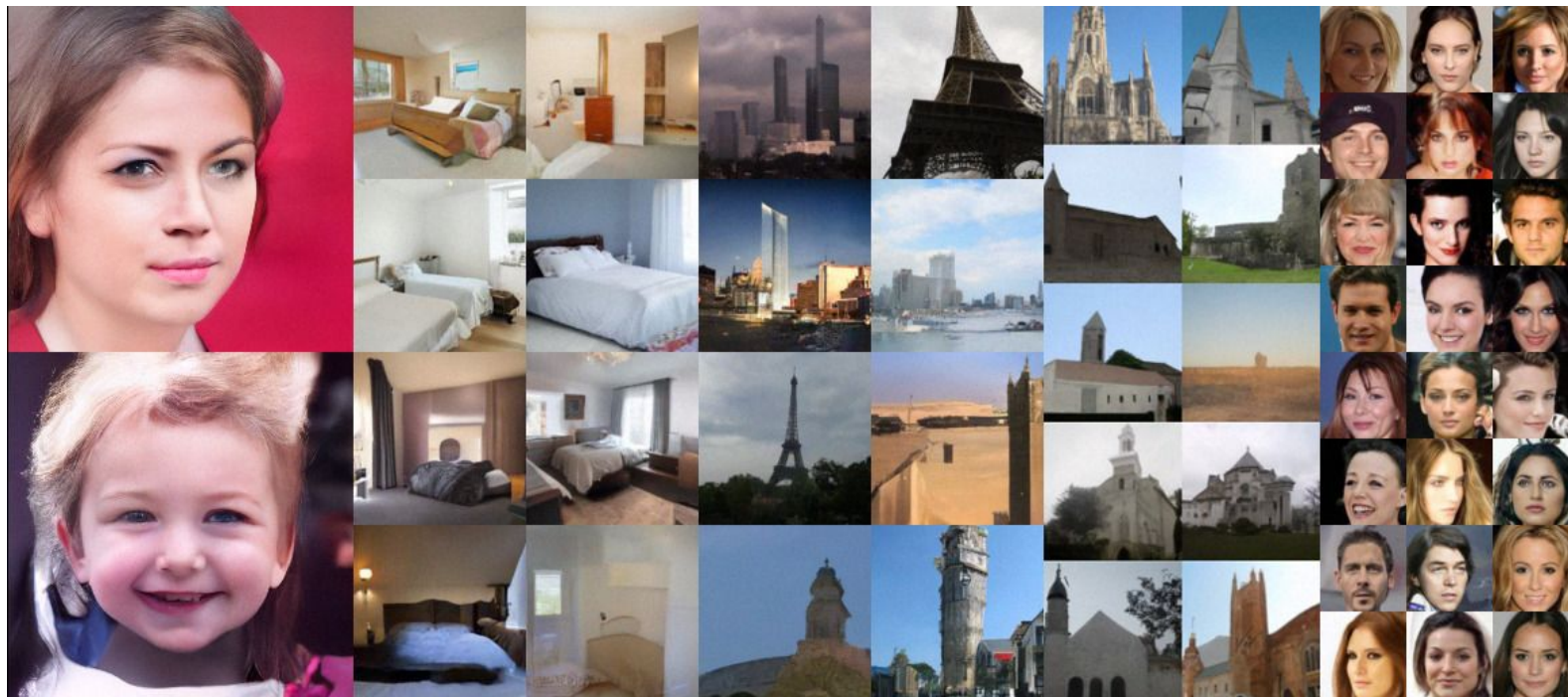


Diffusion Model (Sohl-Dickstein, et al. 2015)



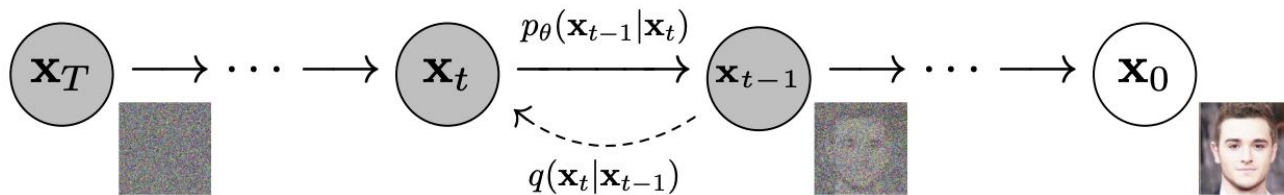
Figure 3. The proposed framework trained on the CIFAR-10 (Krizhevsky & Hinton, 2009) dataset. (a) Example training data. (b) Random samples generated by the diffusion model.

NCSN v2 (Song and Ermon 2020)

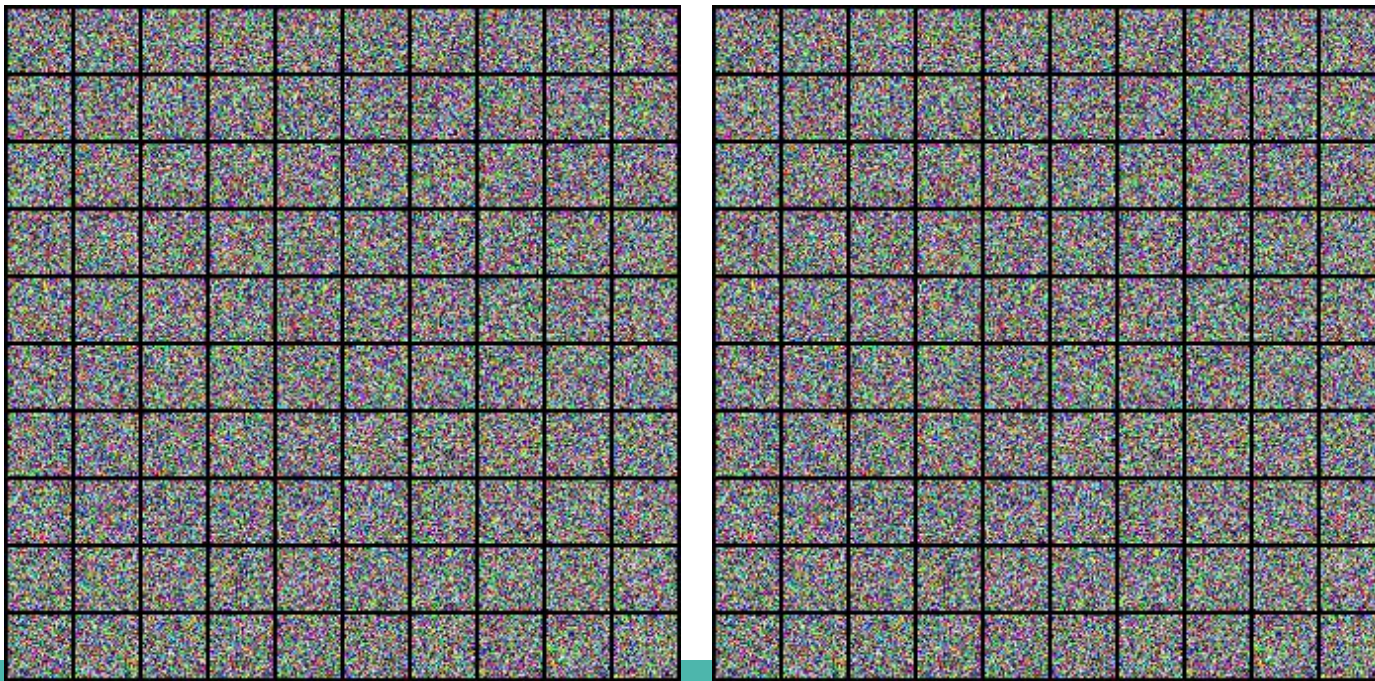


Does this look familiar ...

DDPM



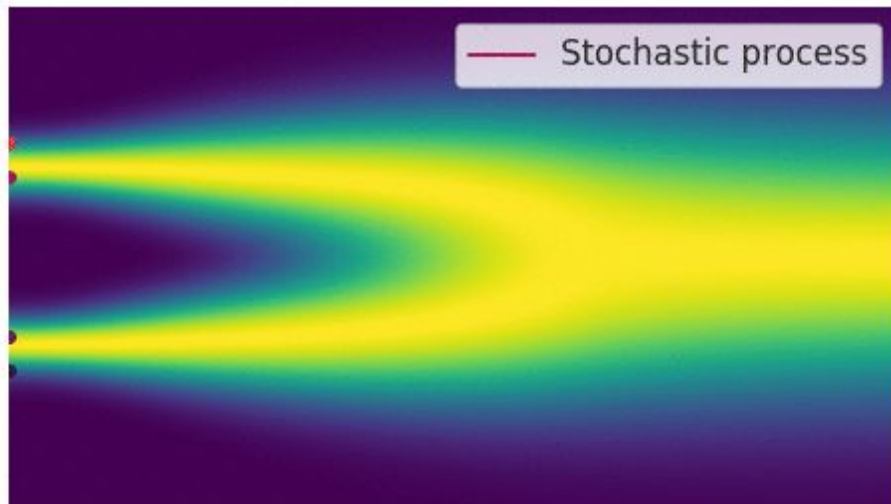
NCSN



When # of noise scales goes to infinity

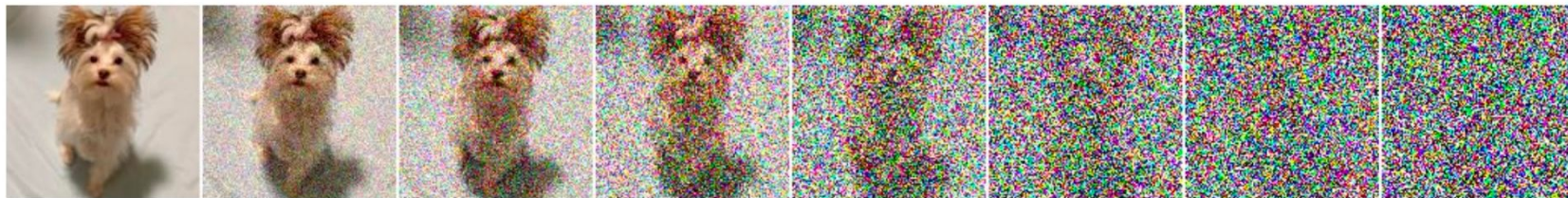
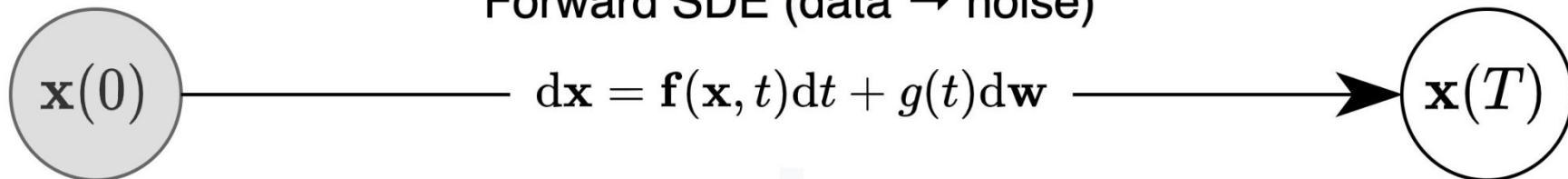
It becomes a continuous-time stochastic process, many of which can be solved by stochastic differential equations (SDEs) (Diffusion is no exception)

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

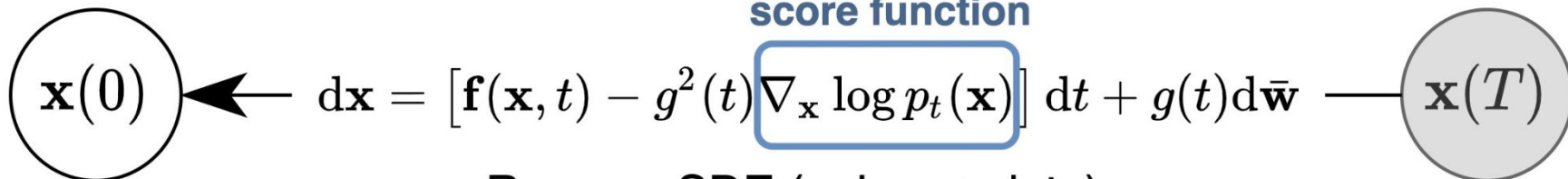


Score-SDE (Song et al. 2021)

Forward SDE (data \rightarrow noise)

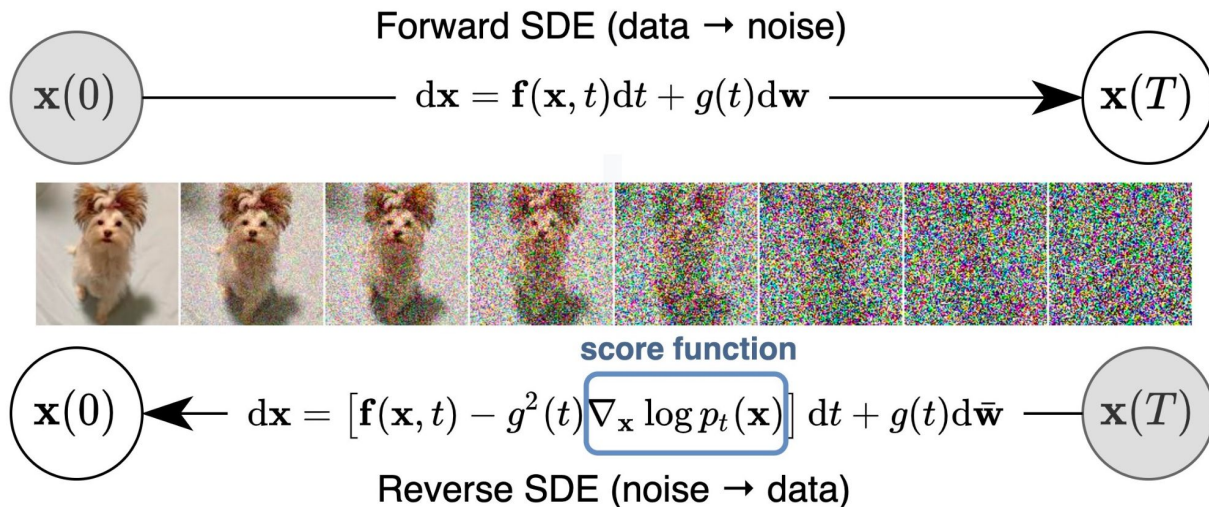


score function



Reverse SDE (noise \rightarrow data)

Diffusion Model is Modeling a Score-SDE!



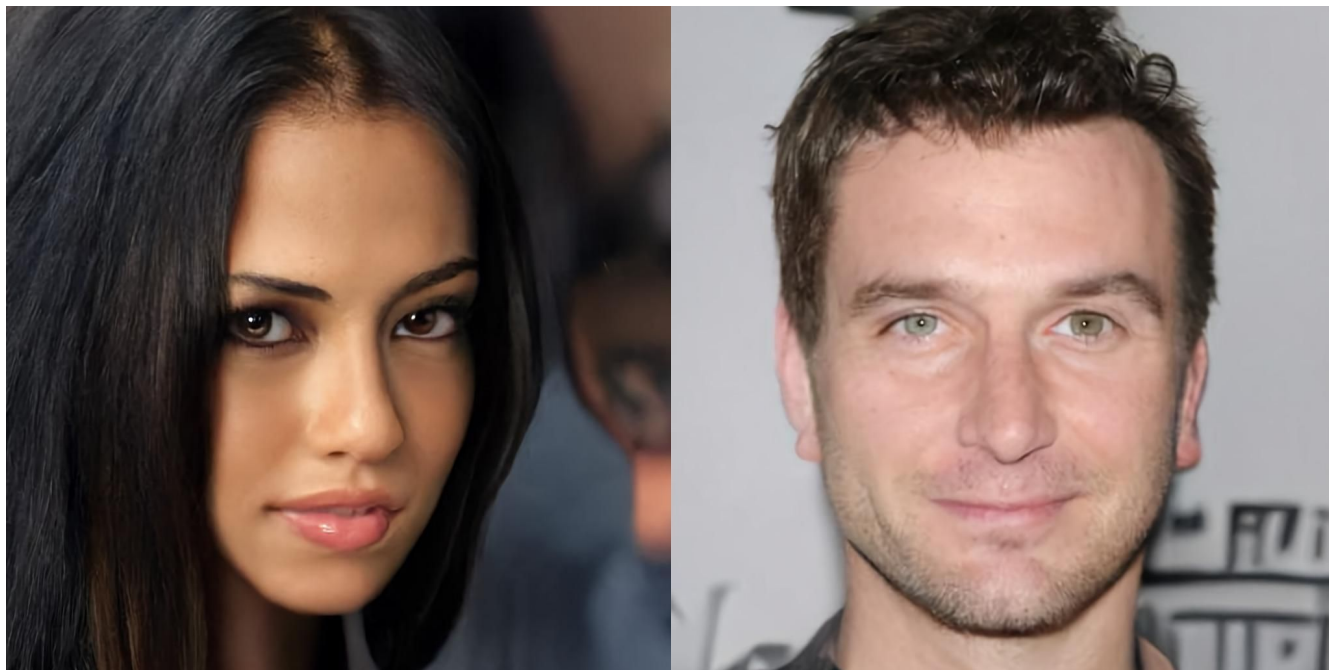
$$\mathbf{x}_i = \sqrt{1 - \beta_i} \mathbf{x}_{i-1} + \sqrt{\beta_i} \mathbf{z}_{i-1}, \quad i = 1, \dots, N.$$

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

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

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

Hence

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



Time-dependent

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

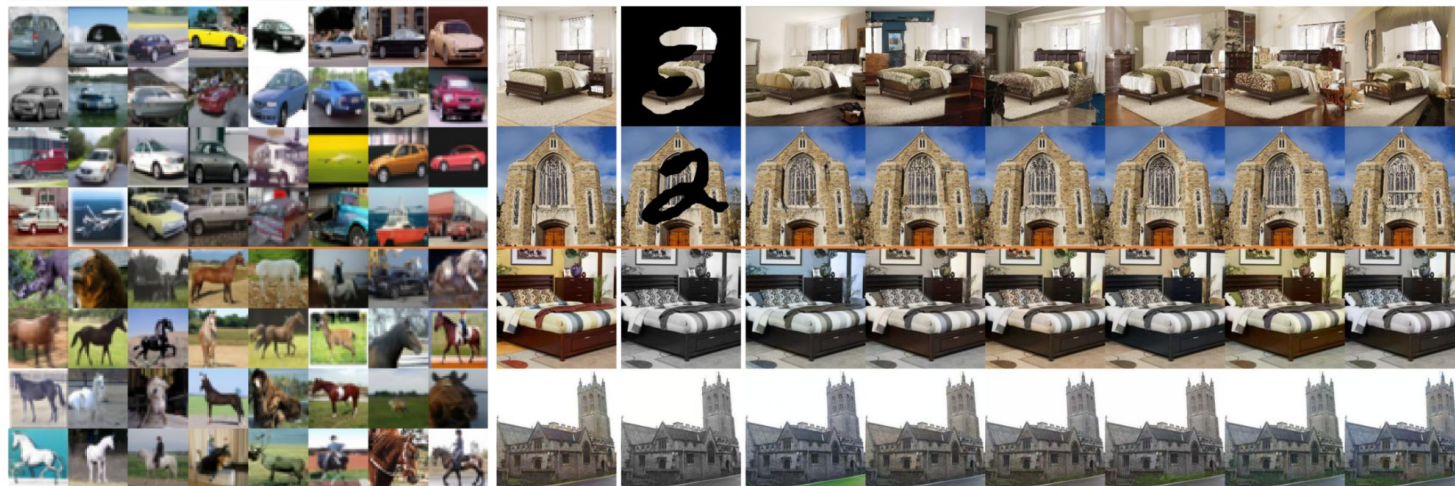


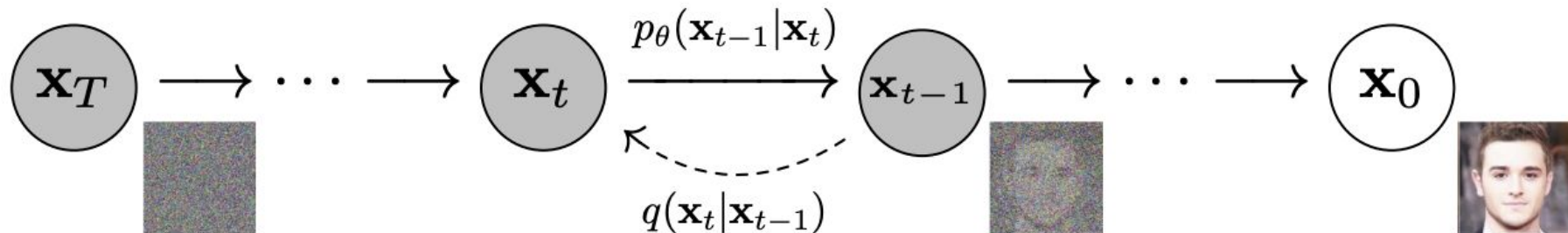
Figure 4: *Left*: Class-conditional samples on 32×32 CIFAR-10. Top four rows are automobiles and bottom four rows are horses. *Right*: Inpainting (top two rows) and colorization (bottom two rows) results on 256×256 LSUN. First column is the original image, second column is the masked/gray-scale image,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

Is diffusion model perfect?

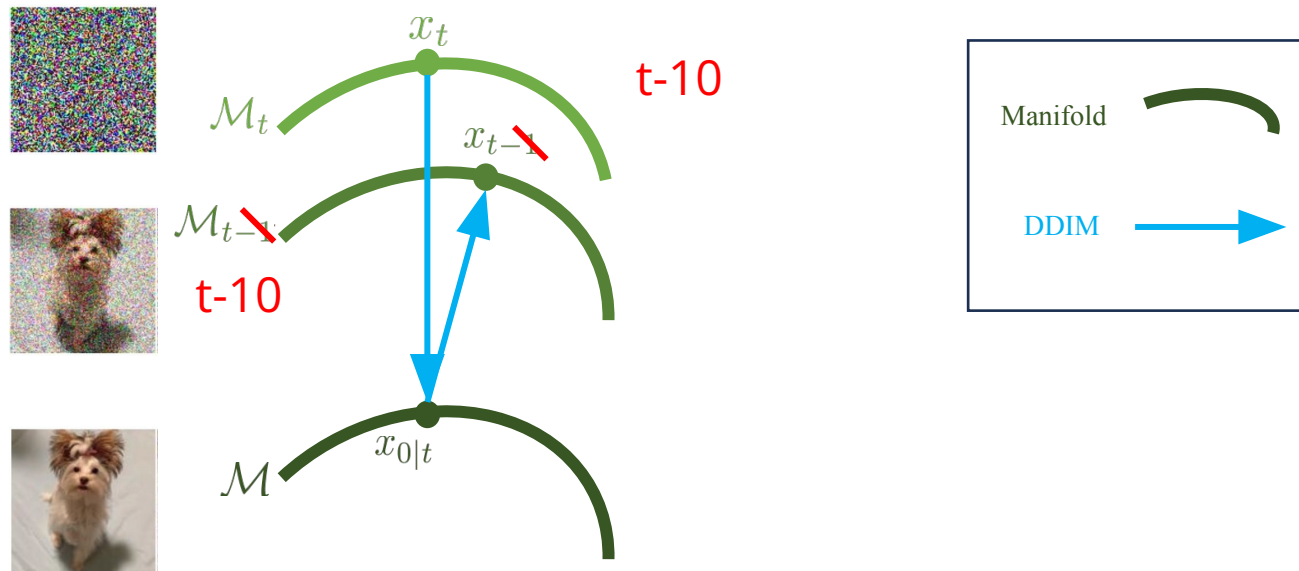


Big Problem: It's very very extremely slow...

Vanilla diffusion model can take 10~30 minutes to generate a single 256X256 image!



DDIM (Song et. al 2021): Fast Sampling by Skipping Time Steps



Diffusion Distillation

Learning another model such that 1 step in this new model is equivalent to N steps in the teacher diffusion model

1. [Progressive Distillation](#)
2. [Consistency Models](#)
3. [Consistency Trajectory Models](#)
4. [Adversarial Distillation](#)
5. [Flow maps](#)

Is diffusion model perfect? How about conditional sampling?



Can we use off-the-shelf classifiers?

By Bayes' Rule we know that

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

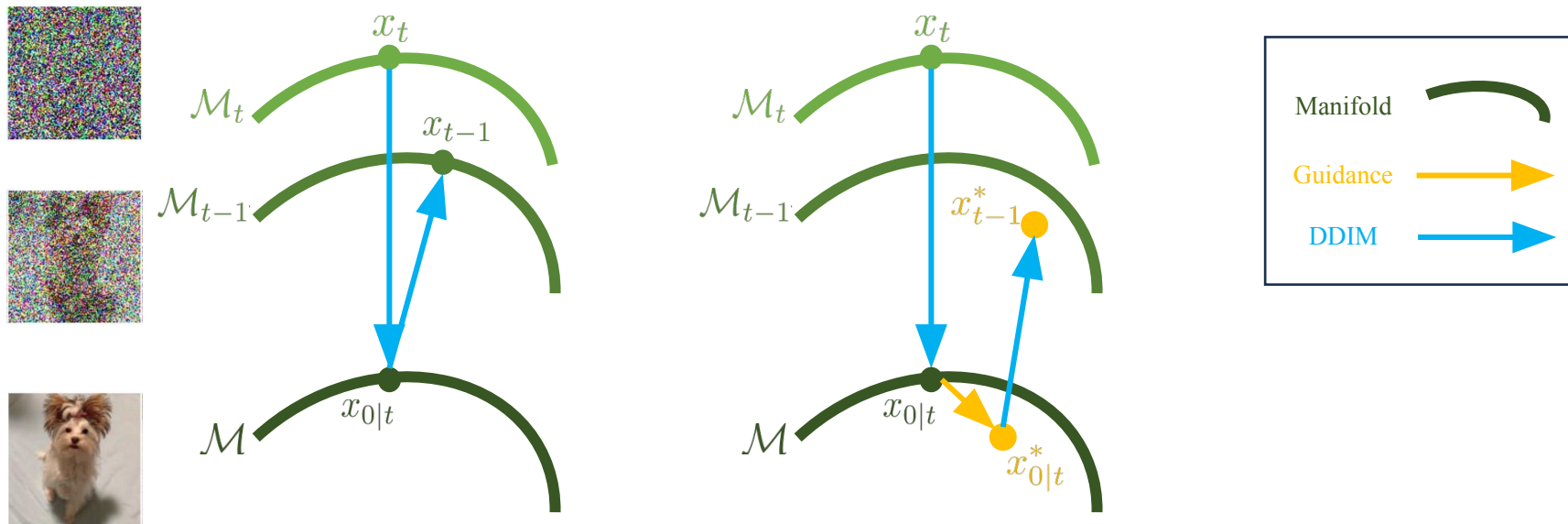
Hence

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

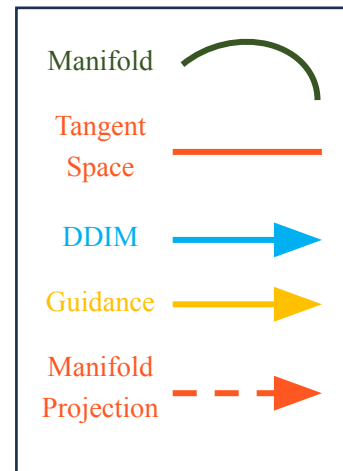
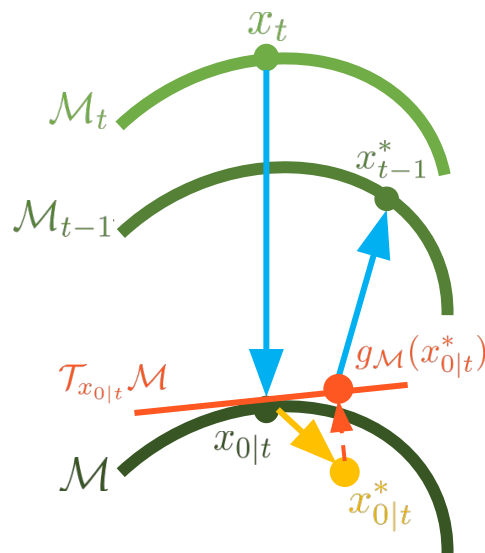
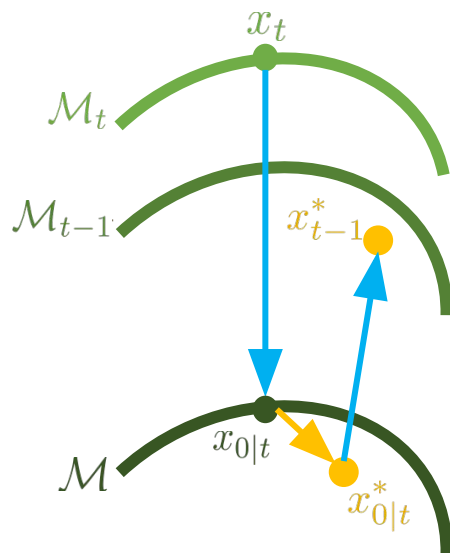
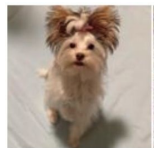
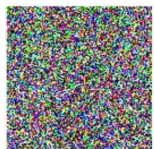
NO! Because this needs to be
time-dependent!



Diffusion Posterior Sampling (Chung et. al 2023): Enable Diffusion Guidance from Off-the-Shelf Classifiers



Manifold Preserving Guided Diffusion (He et. al 2024): Making sure that the guidance gradients are well behaved



MPGD (He et. al 2024)

The same diffusion model can now be used for many applications without additional training!

Noisy Linear Inverse Problems

Measurement



Ground Truth

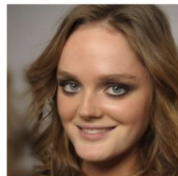


MPGD (~1s/img)



FaceID Guidance Generation

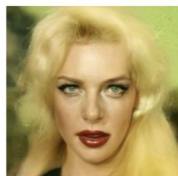
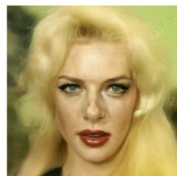
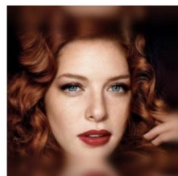
Input Reference



MPGD w/o Proj.



MPGD



CLIP Guidance Generation

Unconditional



MPGD



Prompt: "a headshot of a person wearing red lipstick"

Style Guidance + Stable Diffusion

Input Reference



MPGD (~10s/img)



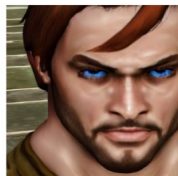
Prompt: "a canal in Venice"

FaceID Guidance + Stable Diffusion

Input Reference



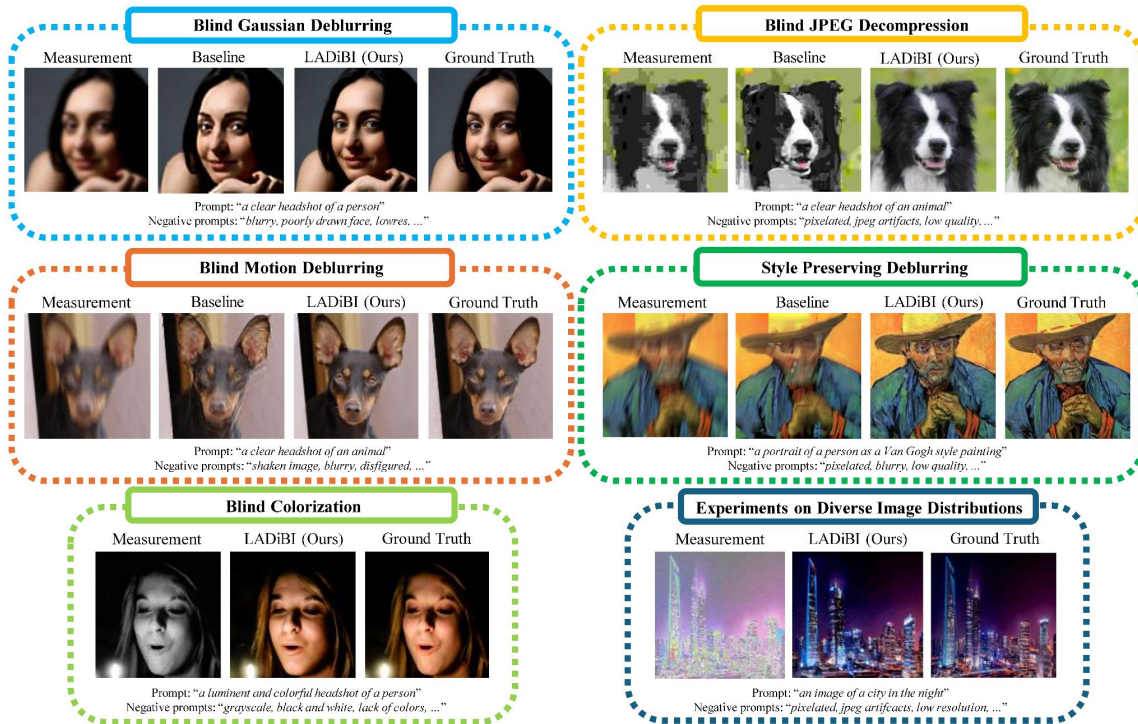
MPGD



Prompt: "a headshot of a male game character"

LaDiBI (Dontas et. al 2024)

The same diffusion model can now be used for many applications without additional training!

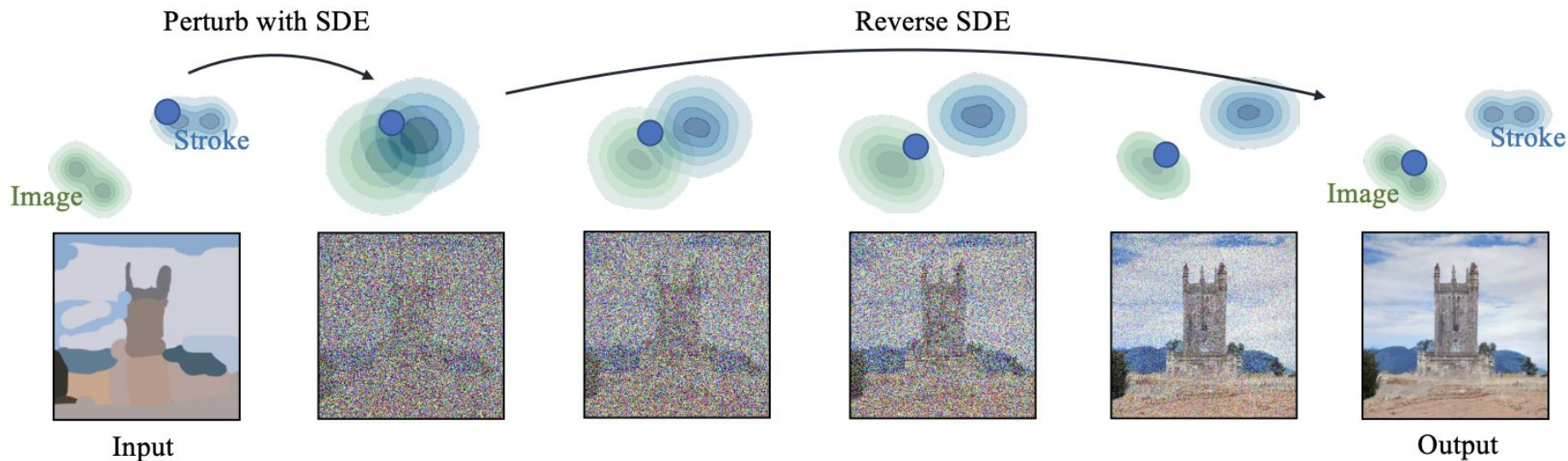


Any other ways to apply condition to diffusion models?



SDEdit: Conditional Generation w/o Training

(Meng et al. 2022)

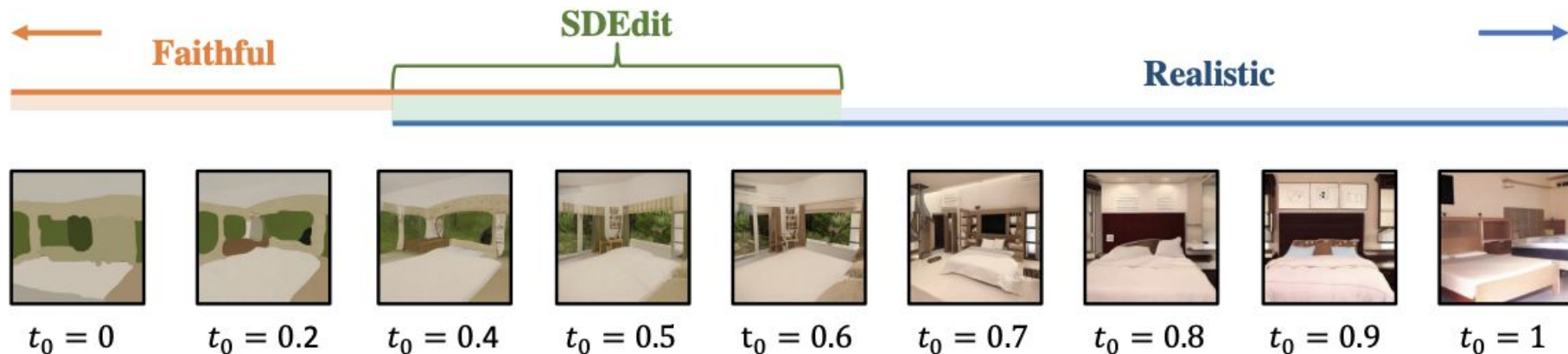


SDEdit: Conditional Generation w/o Training

(Meng et al. 2022)

More faithful
Less realistic

More realistic
Less faithful



SDEdit: Conditional Generation w/o Training

(Meng et al. 2022)

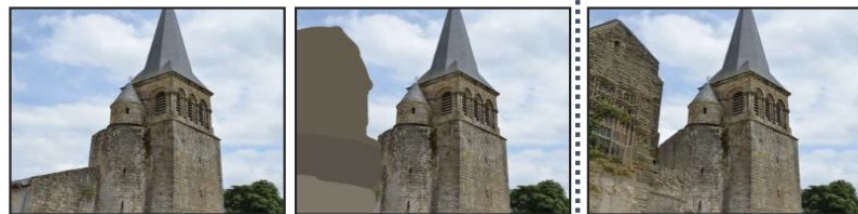
Stroke Painting to Image



Input (guide)

Output

Stroke-based Editing



Source

Input (guide)

Output

Image Compositing



Source

Input (guide)

Output

Source

Input (guide)

Output

SDEdit: Conditional Generation w/o Training

(Meng et al. 2022)



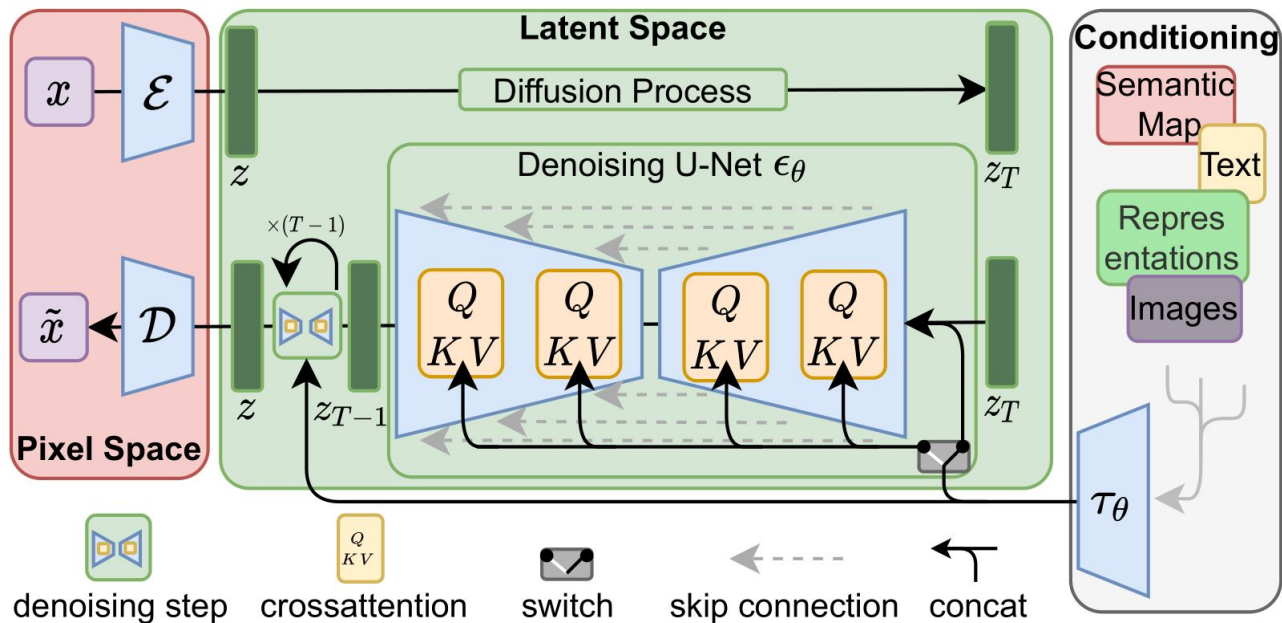
SDEdit: Conditional Generation w/o Training

(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)



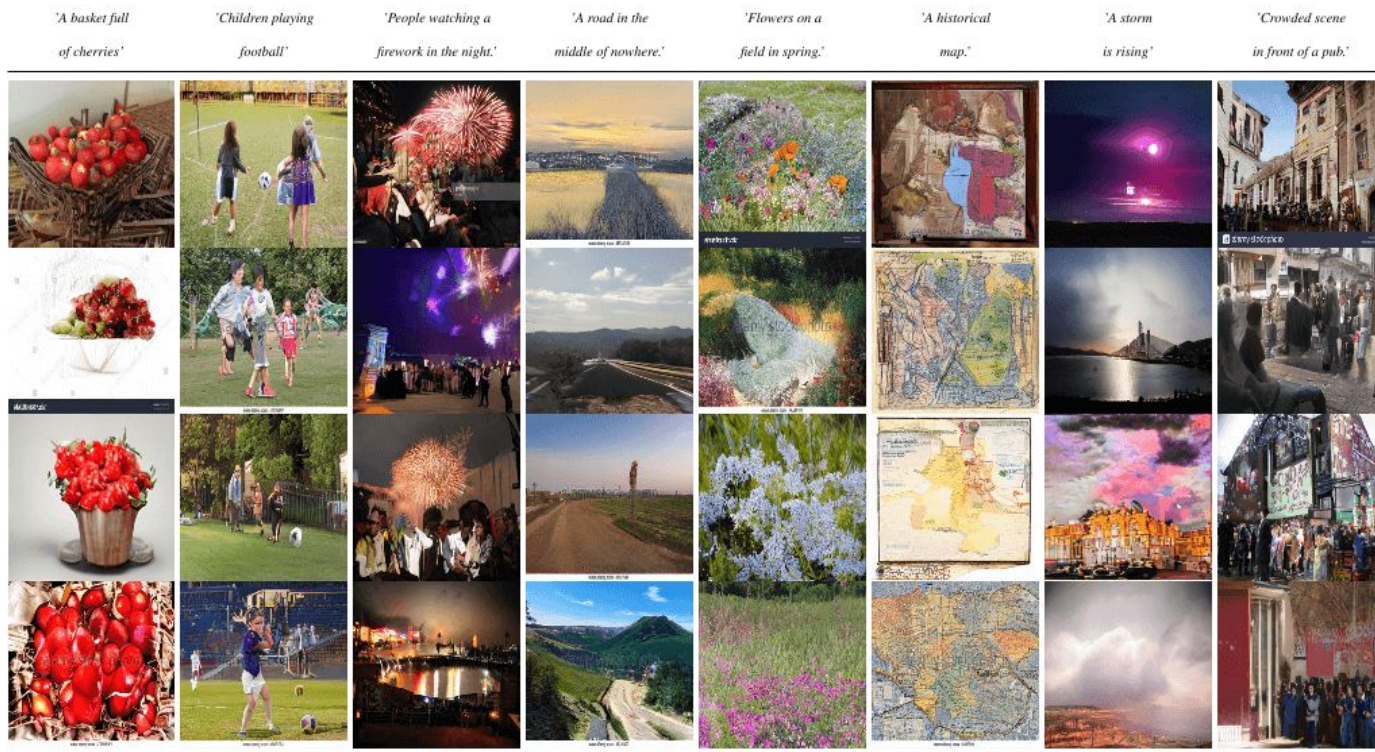
Stable Diffusion (Rombach et al. 2022)

layout-to-image synthesis on the COCO dataset



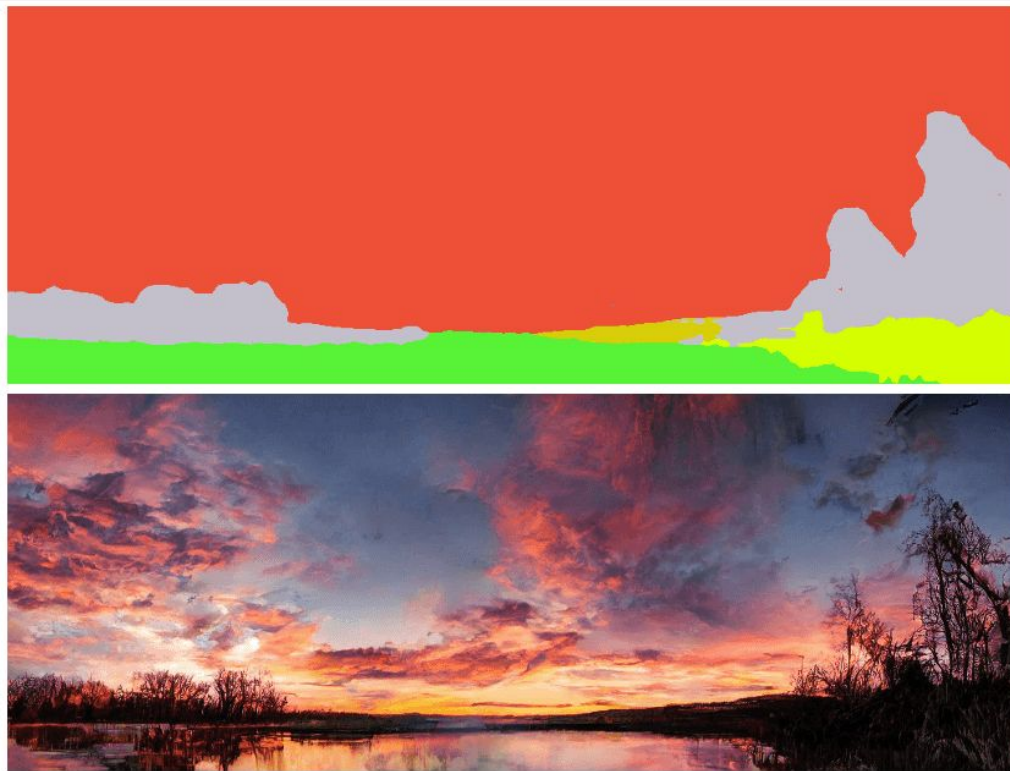
Stable Diffusion (Rombach et al. 2022)

Text-to-Image Synthesis on the Conceptual Captions dataset



Stable Diffusion (Rombach et al. 2022)

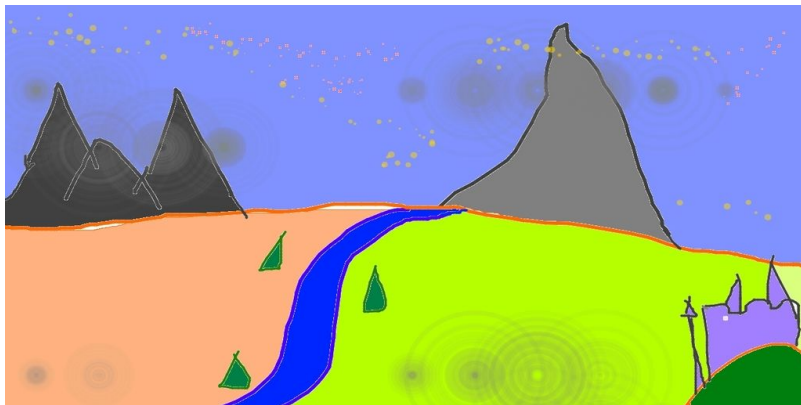
Semantic Synthesis on Flickr-Landscapes [21]



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)

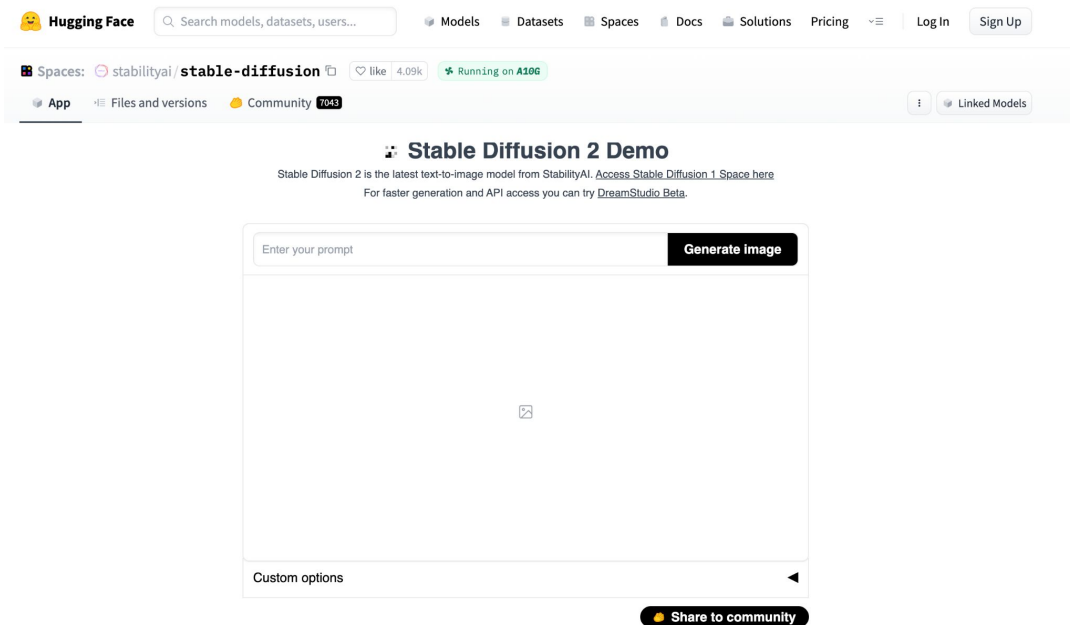
“A fantasy landscape, trending on artstation”



Uses SDEdit!

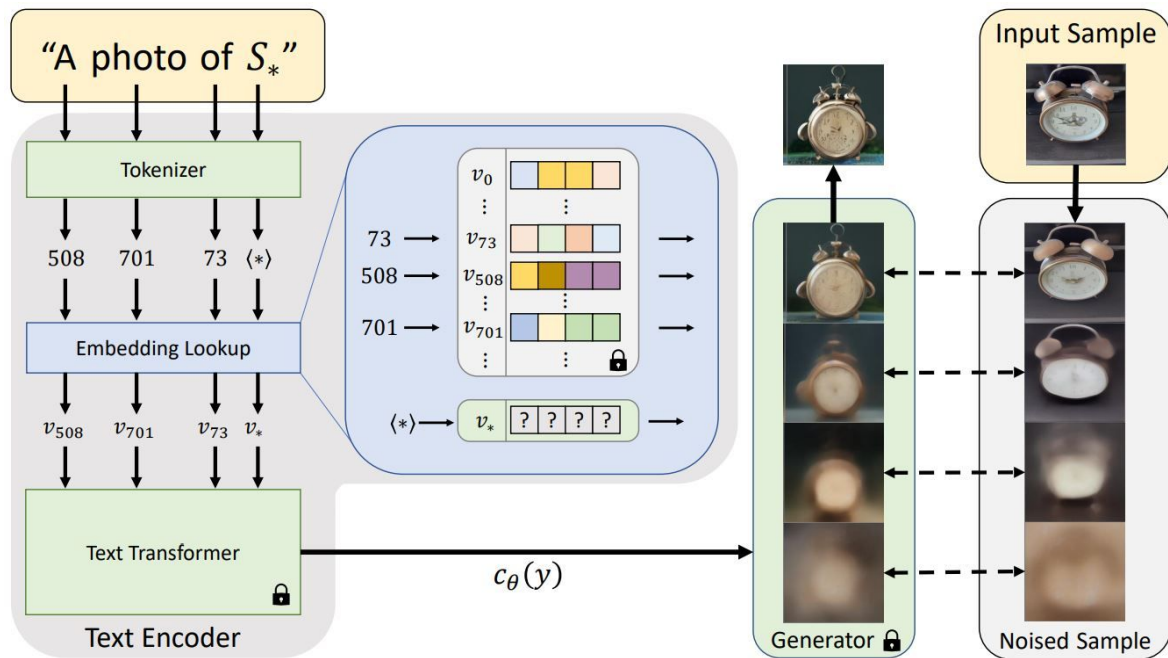
Stable Diffusion (Rombach et al. 2022)

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>

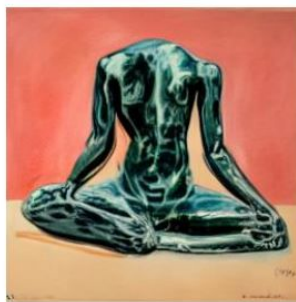
Textual Inversion (Gal et al. 2022)



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”

DreamBooth (Ruiz et. al 2022)



Input images



in the Acropolis



swimming



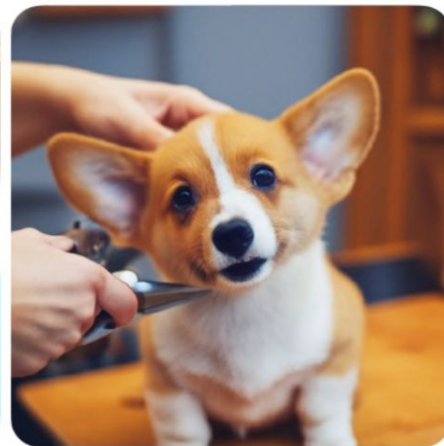
sleeping



in a doghouse

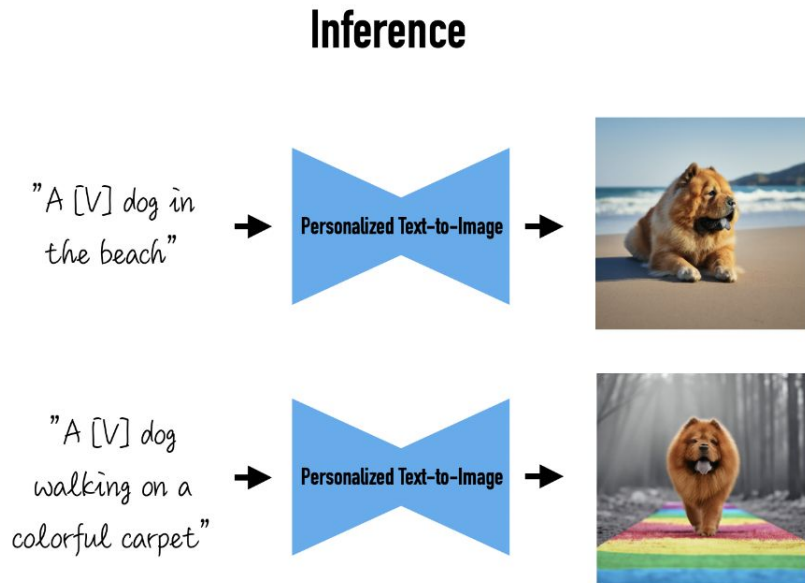
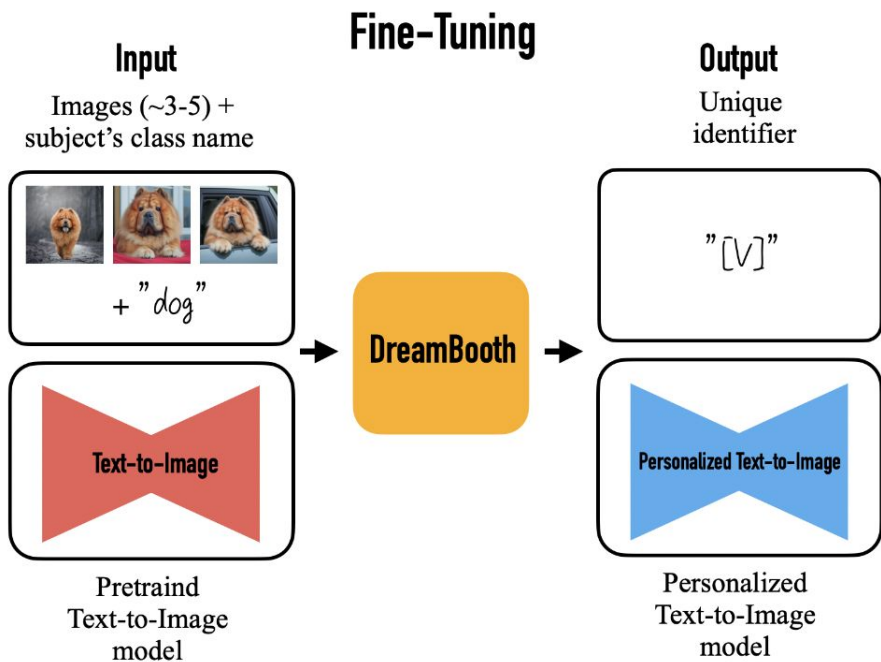


in a bucket

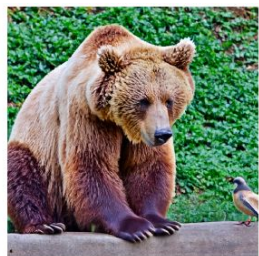


getting a haircut

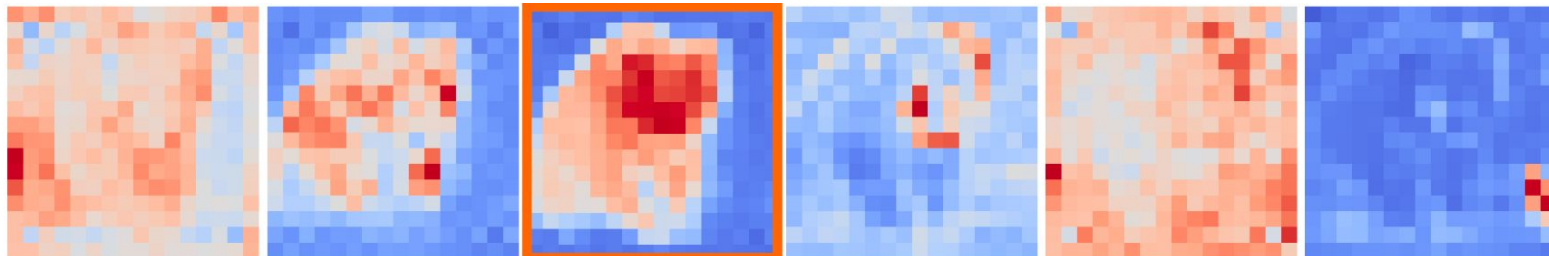
DreamBooth (Ruiz et. al 2022)



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



Synthesized image



“a

furry

bear

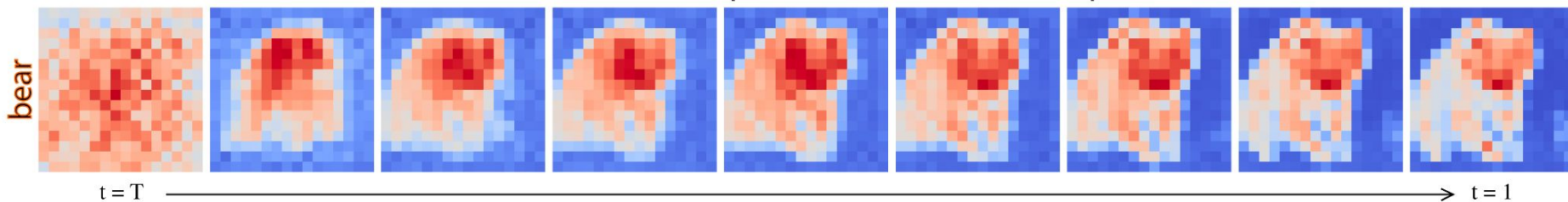
watches

a

bird”

Average cross-attention maps across all timestamps

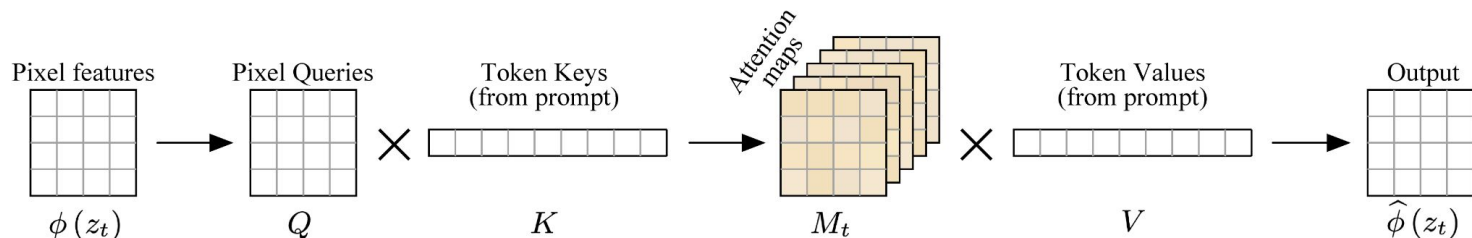
Cross-attention maps for individual timestamps



t = T

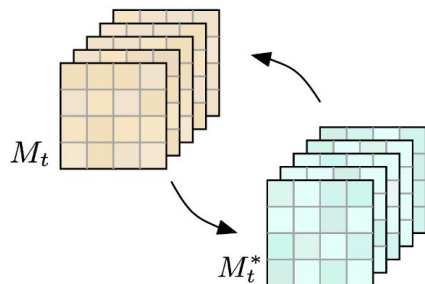
→ t = 1

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

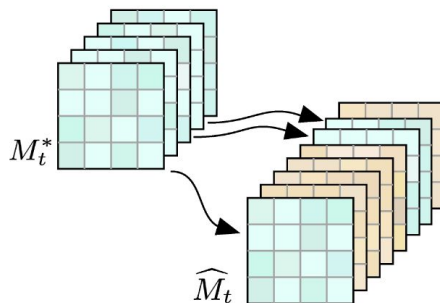


Text to Image Cross Attention

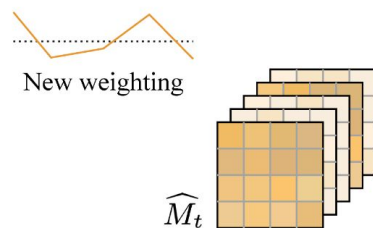
Cross Attention Control



Word Swap

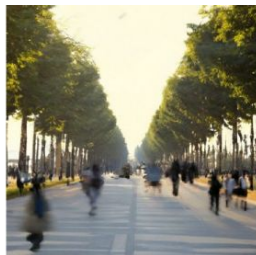


Prompt Refinement



Attention Re-weighting

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



“The boulevards are crowded today.”



“My fluffy bunny doll.”



“Photo of a cat riding on a ~~bicycle.~~”

car



“a cake with decorations.”

jelly beans

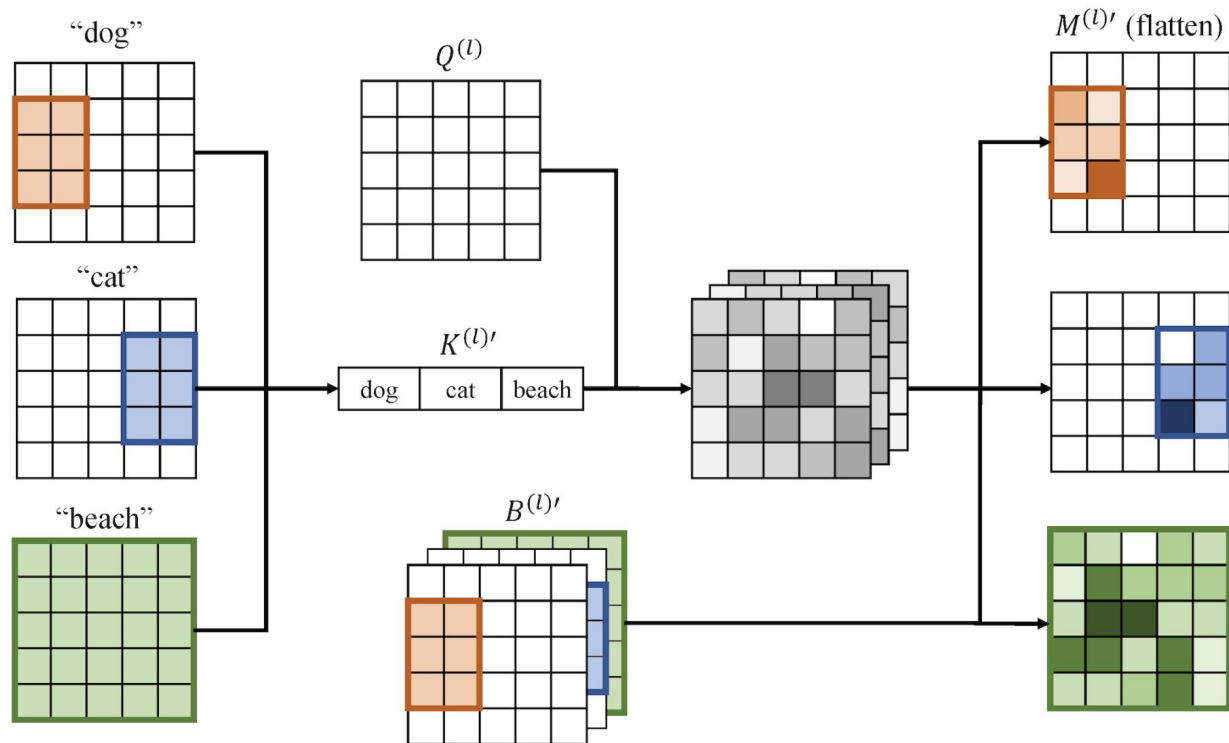


“Landscape with a house near a river
and a rainbow in the background.”



“Children drawing of a castle next to a river.”

CAC (He et. al 2023)



CAC (He et. al 2023)

Generation w/ Bounding Boxes



Caption: Martian landscape
Localized Prompts: an astronaut riding a horse, a pink flag



Caption: a photo of a beach
Localized Prompts: a palm tree, an air balloon, a beach umbrella

Generation w/ GLIGEN



Caption: a blue cup and a green cell phone
Localized Prompts: a blue cup, a green cell phone

Generation w/ Semantic Segmentation Maps



Caption: a cute cat
Localized Prompts: blue eye, green eye



Caption: a photo of a peaceful landscape
Localized Prompts: a unicorn, snow mountain, lake, pink cloud, dusk sky

Localized Style Generation



Caption: Mona Lisa
Localized Prompts: the style of Monet, Van Gogh, Picasso, da Vinci


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"

Edited Caption: "photograph of a girl riding a dragon"

Stable Diffusion
+ Prompt2Prompt



(c) Generated training examples:

"convert to brick"



"Color the cars pink"



"Make it lit by fireworks"



"have her ride a dragon"



...

Instruction-following Diffusion Model

(d) Inference on real images:

"turn her into a snake lady"



InstructPix2Pix



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>

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
 - [EDM](#)
 - [Distillation Papers](#)
- Guided Diffusion
 - [Classifier-guided diffusion](#)
 - [Classifier-free diffusion guidance](#)
 - [ControlNet](#)
- Other Modalities
 - [Video](#)
 - [Point cloud](#)
 - [Music](#)
 - [Discrete](#)
 - [Reinforcement Learning](#)

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>
 - Calvin Luo. "Understanding Diffusion Models: A Unified Perspective".
<https://calvinluo.com/2022/08/26/diffusion-tutorial.html>
 - My email: yutonghe@andrew.cmu.edu
- My website: <https://kellyyutonghe.github.io/>

New class in Spring 2026!

10-799: Diffusion & Flow Matching

