Diffusion Models

for Conditional Generation & Visual Restoration

Yuanzhi Zhu

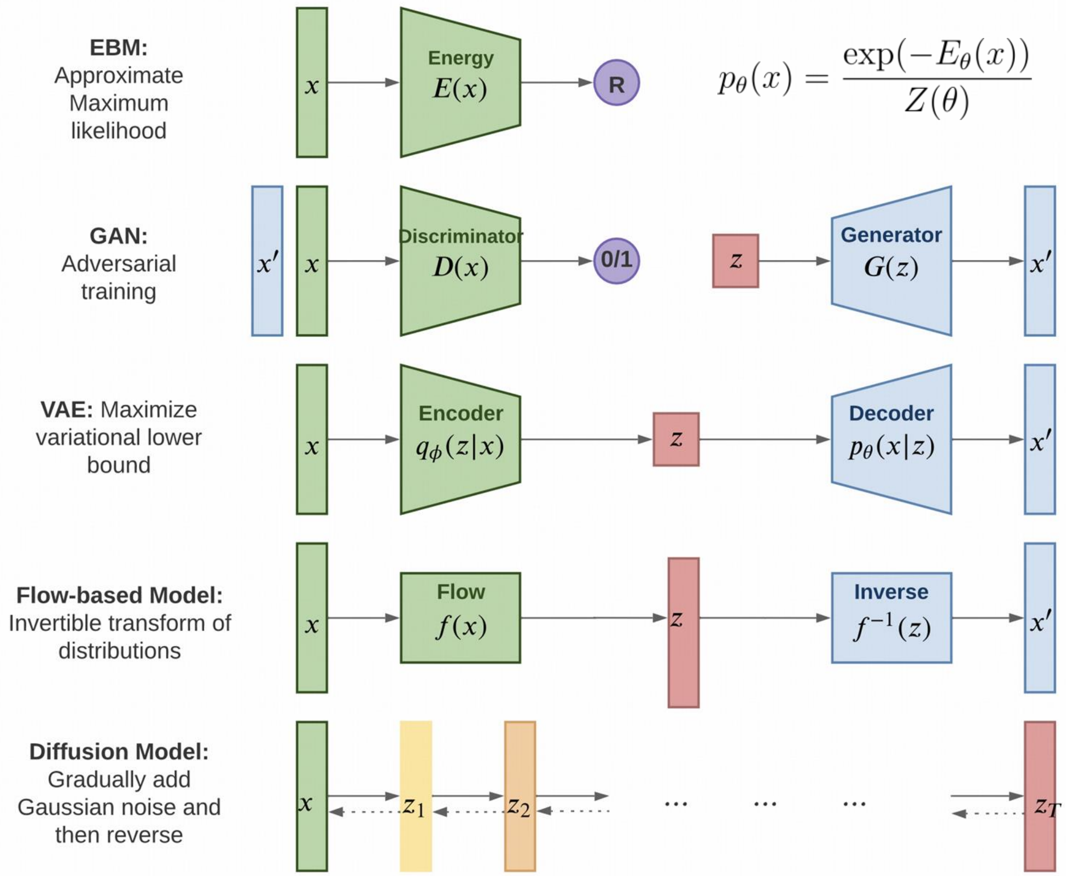
Supervisors: *Prof*. Kai Zhang

27/Sep/2024

Content

- Diffusion Models
- Conditional Generation
- Image Restoration

A Tale of Generative Models as Mapping Connecting Distributions



Model the distribution?

→ hard to draw new samples

Model the mapping!

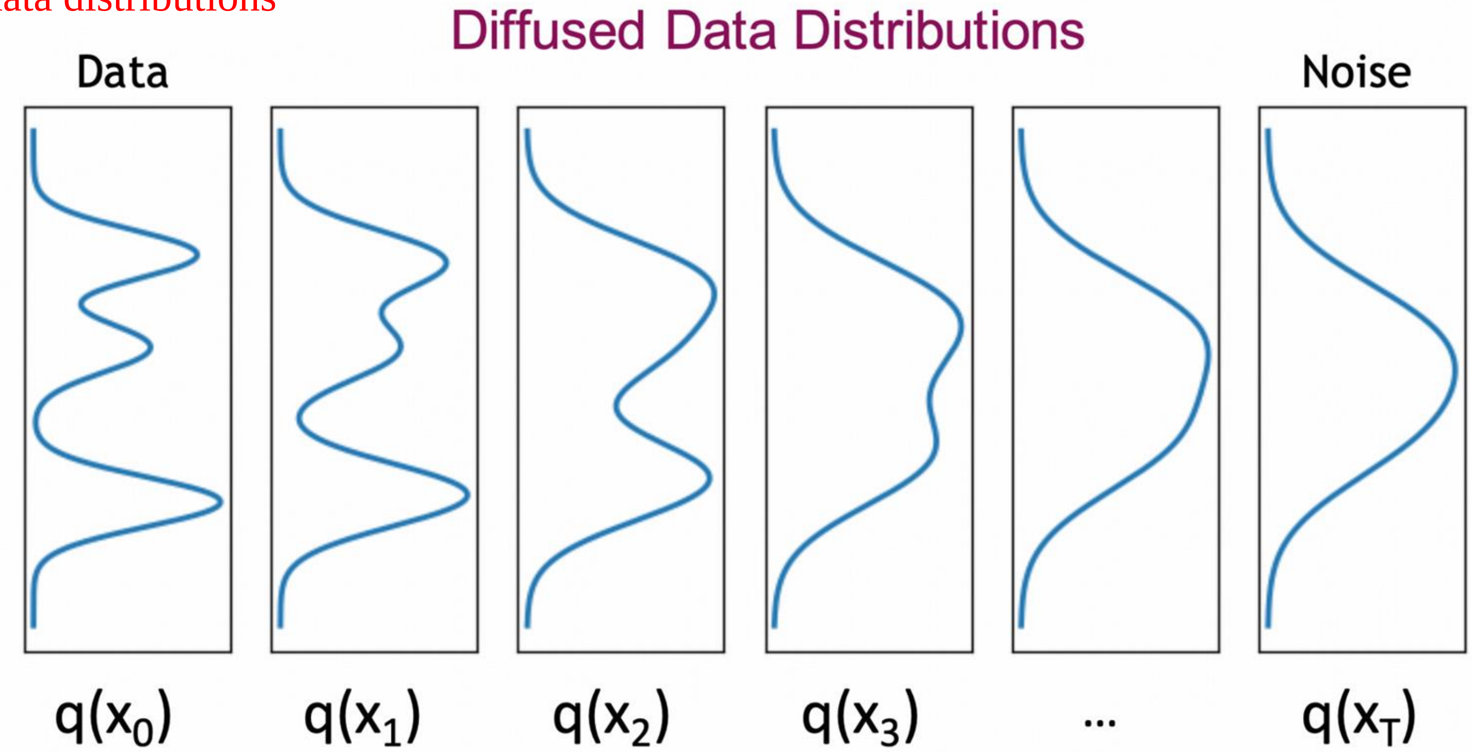
Input: samples from initial distribution $q(x_T)$ and target distribution $q(x_0)$

Output: mapping from easy to sample dist. p(z) to target dist. p(x)

Diffusion Models: Divide and Conquer → Build Intermediate Process

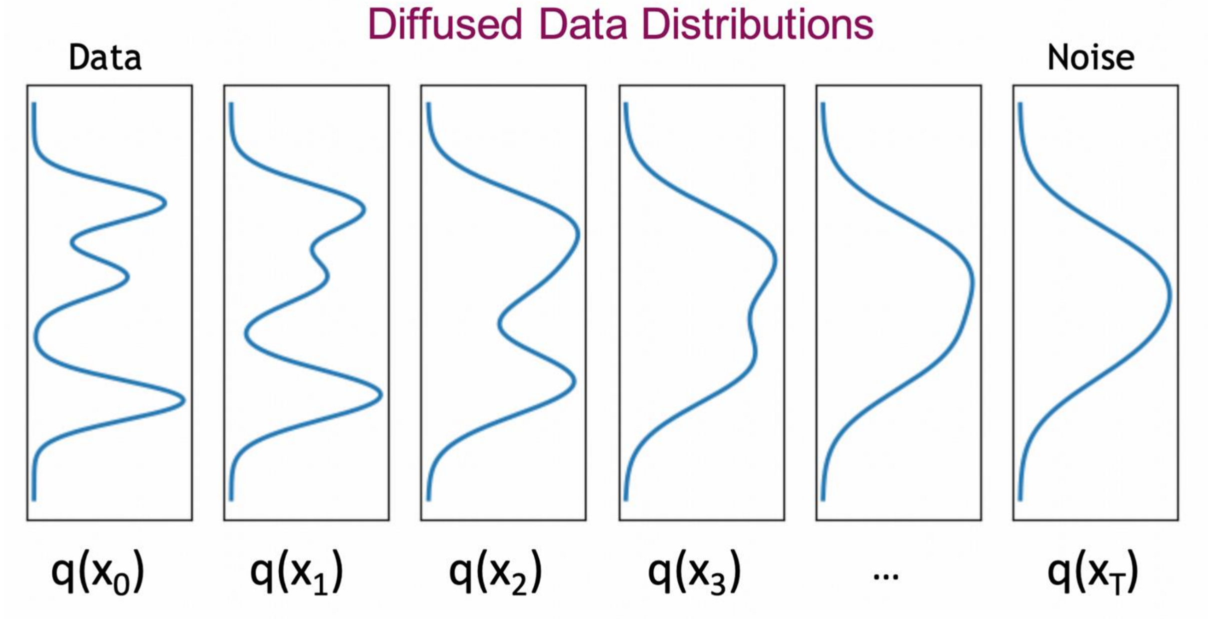
Construct *T-1* intermediate diffused distributions using $q(x_0)$ and $q(x_T)$

Noisy data distributions



Diffusion Models: Divide and Conquer → Build Intermediate Process

Construct *T-1* intermediate diffused distributions using $q(x_0)$ and $q(x_T)$



Noisy data distributions through
Linear Interpolation!

```
x_0 = next(iter(data_loader))
eps = torch.randn_like(x_0)
mean = alpha_bar[t] ** 0.5 * x0
var = 1 - alpha_bar[t]
x_t = mean + (var ** 0.5) * eps
```

How to construct sample $x_t \sim q(x_t)$:

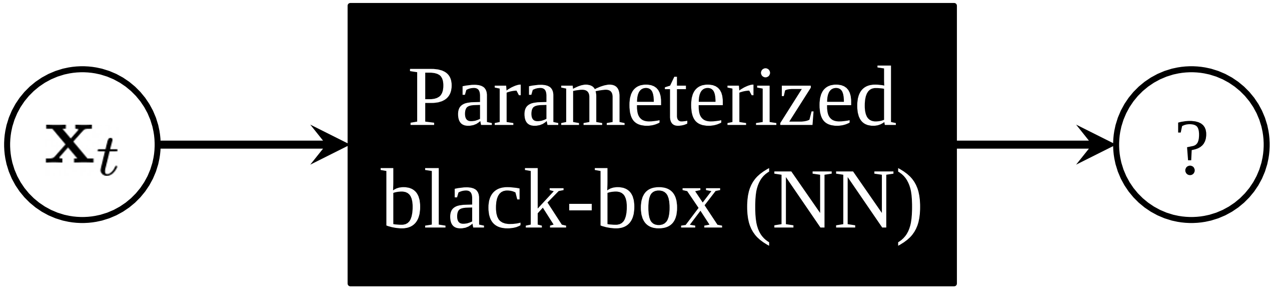
$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} z_t$$

$\bar{\alpha}_t$: predefined schedule

$x_0 \sim q(x_0)$ $z_t \sim q(x_T)$

Diffusion Models: Divide and Conquer → Model Output

Imaging the ***iterative*** generation process of diffusion



$\mathbf{x}_{t\text{-}1}$?

We do not have $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to sample this target at all ??!!

Diffusion Models: Divide and Conquer → Mapping from t to t-1

What do we need to perform the mini-step mapping, or one reverse diffusion step

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}z_t$$

Reverse when condition on x_0

$$q(x_{t-1}|x_t, x_0) = \boxed{q(x_t|x_{t-1}, x_0)\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)}}$$

all three are ***known*** forward processes

$$\longrightarrow q(x_{t-1}|x_t, x_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I})$$

$$\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t$$

$$\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boxed{x_0} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boxed{\mathbf{z}_t}\right)$$

Need either $\boldsymbol{x}_0$ or $\mathbf{z}_t$ to perform a reverse step

model $\boldsymbol{x}_0$ or $\mathbf{z}_t$ with NN

☺ $\begin{cases} D_\theta(x_t, t) \\ \mathbf{z}_\theta(x_t, t) \end{cases}$

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion Models: Divide and Conquer → Equivalent Objectives

The training objective of different diffusion / score methods are equivalent

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} z_t$$

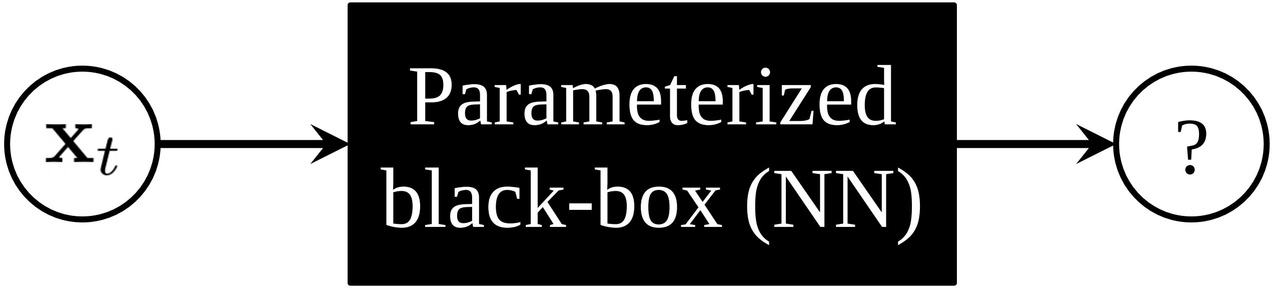
$$D_\theta(x_t, t) = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(x_t - \sqrt{1 - \bar{\alpha}_t} \mathbf{z}_\theta(x_t, t) \right)$$

define ***score*** as the gradient of log probability

$$\begin{aligned} \mathbf{s}_\theta(x_t, t) &\approx \nabla_{x_t} \log p(x_t | x_0) \\ &= -\frac{x_t - \sqrt{\bar{\alpha}_t} x_0}{1 - \bar{\alpha}_t} \\ &\approx -\frac{x_t - \sqrt{\bar{\alpha}_t} D_\theta(x_t, t)}{1 - \bar{\alpha}_t} = -\frac{\mathbf{z}_\theta(x_t, t)}{\sqrt{1 - \bar{\alpha}_t}} \end{aligned}$$

Diffusion Models: Divide and Conquer → Model Output

Imaging the ***iterative*** generation process of diffusion



$\mathbf{x}_0$!

The benefits:
Connection to ***Tweedie's Formula*** (easier for human)
Same target for a single model (easier for NN)

[[2409.06219] Denoising: A Powerful Building-Block for Imaging, Inverse Problems, and Machine Learning]()

Diffusion Models: A Set of *Denoisers* at Different Noise Level

The training objective of different diffusion / score methods are equivalent

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} z_t$$

$$D_\theta(x_t, t) = \frac{1}{\cdots}\big(\cdots$$

define ***score*** as the g…

Diffusion Models are Trained as Denoisers!!

equivalently

$$\cdots p(x_t | x_0)\big)$$

$$= -\frac{x_t - \sqrt{\bar{\alpha}_t} x_0}{1 - \bar{\alpha}_t}$$

at each noise level

$$\approx -\frac{x_t - \sqrt{\bar{\alpha}_t} D_\theta(x_t, t)}{1 - \bar{\alpha}_t} = -\frac{\mathbf{z}_\theta(x_t, t)}{\sqrt{1 - \bar{\alpha}_t}}$$

Diffusion Models: Training & Sampling

Algorithm 1 Training

repeat
$\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
$\quad t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
$\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
$\quad$ Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \mathbf{z}_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}}_{x_t}, t) \right\|^2$$

until converged

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
$\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$

$$\boxed{\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}}$$

perform a reverse step

end for
return $\mathbf{x}_0$

$\boxed{\text{output: } \mathbf{z}_\theta(x_t, t)}$

https://arxiv.org/abs/2006.11239

Diffusion Models: Training & Sampling

Algorithm 1 Training

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

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\boxed{\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}}$
end for
return $\mathbf{x}_0$

perform a reverse step

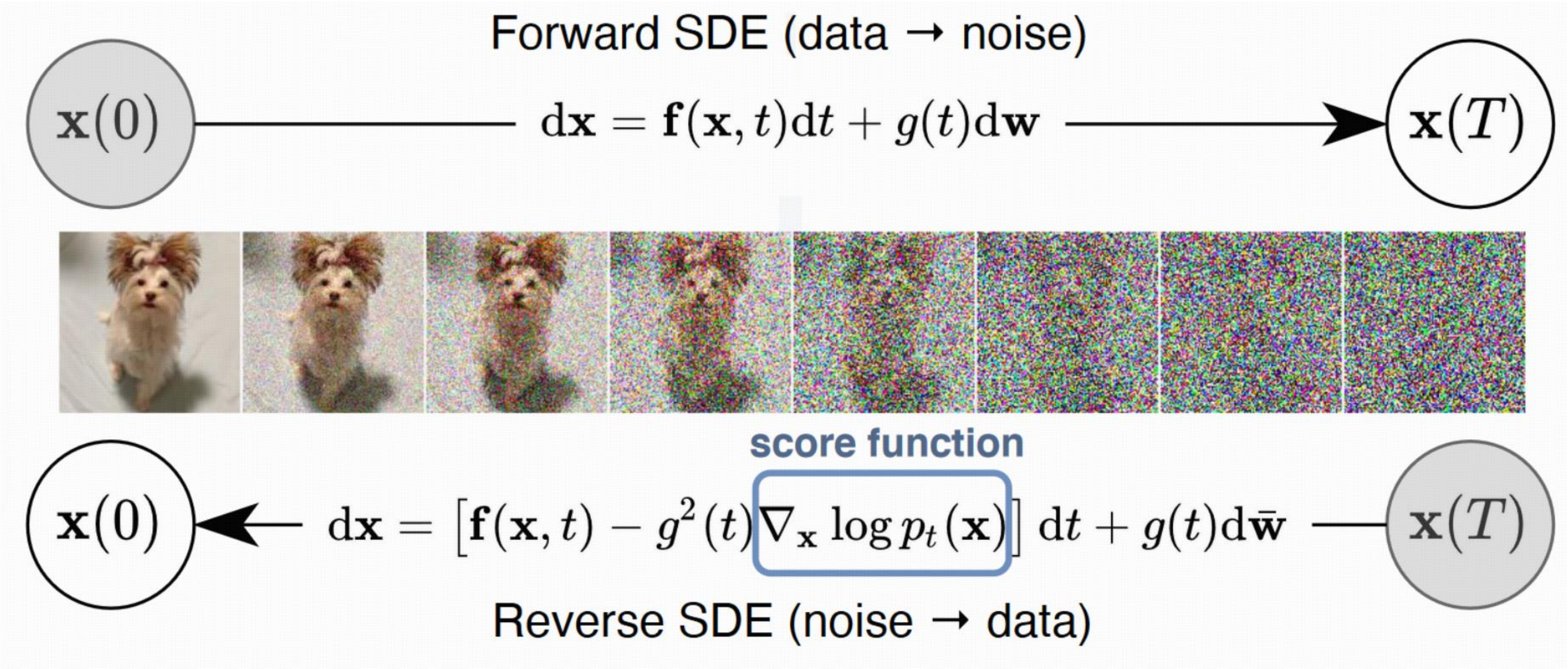
```python
while True:
    x_0 = next(iter(data_loader))
    eps = torch.randn_like(x_0)
    t = torch.randint(0, T, (bs,))
    mean = alpha_bar[t] ** 0.5 * x0
    var = 1 - alpha_bar[t]
    x_t = mean + (var ** 0.5) * eps
    loss = F.mse_loss(model(x_t, t), x_0)
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
```

```python
x_0 = next(iter(data_loader))
# start the sampling from x_T
x = x_T = torch.randn_like(x_0)
for t in range(T, 0, -1):
    # get schedule alpha_bar_t
    a_bar_t = alpha_bar[t]
    # get x_0_pred based on x_t and t
    x_0_pred = model(x, t)
    # sample x_{t-1}, one reverse step
    x = schedule.step(x, x_0_pred, a_bar_t)
```

https://arxiv.org/abs/2006.11239

Diffusion Models: From Discrete to Continuous

f and g are known, related to schedule

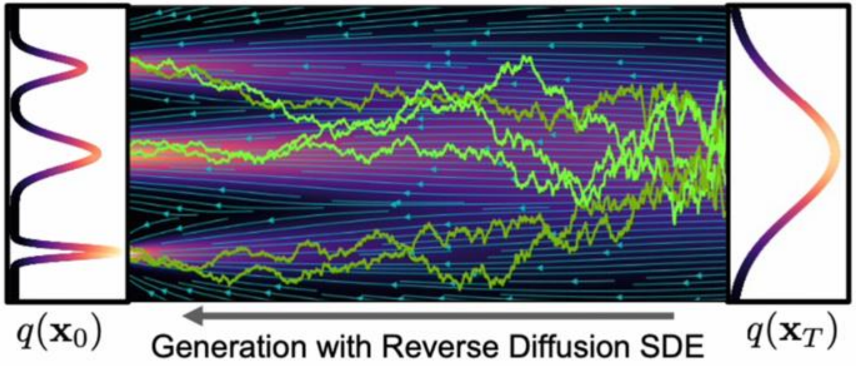


https://yang-song.net/blog/2021/score/ [2011.13456] Score-Based Generative Modeling through Stochastic Differential Equations (arxiv.org)

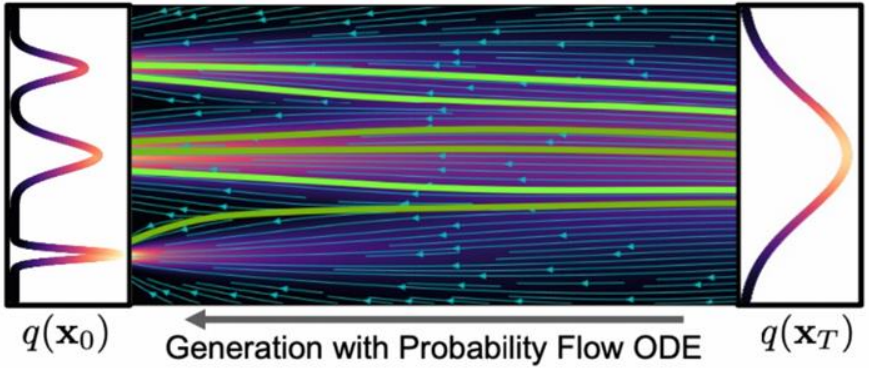
Diffusion Models: New Sampling Perspective

What are we doing when we perform a sampling step?

- OLD: Denoise a bit; change noise level from index *t* to *t-1*
- NEW: Solve the SDE / ODE for one step



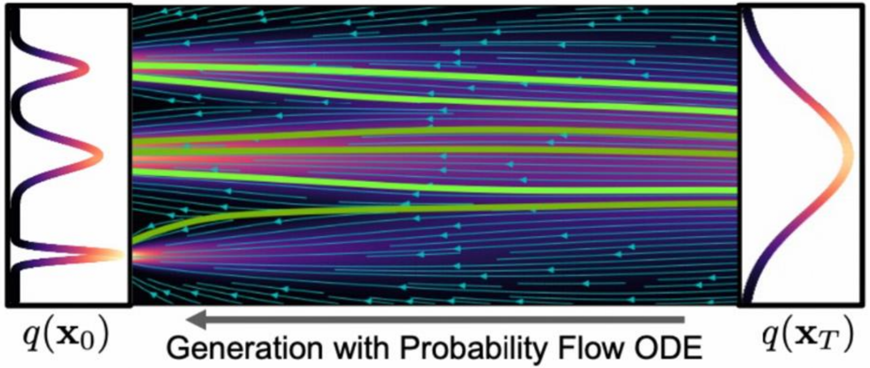
$$\mathrm{d}x = [f(x,t) - g^2(t)s_\theta(x,t)]\mathrm{d}t + g(t)\mathrm{d}w$$



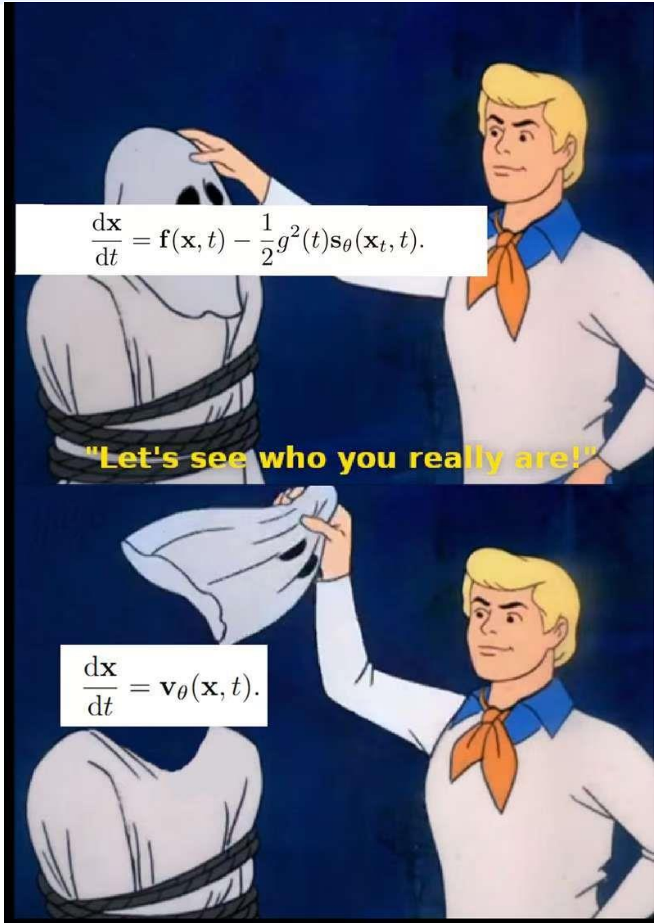
$$\mathrm{d}x = \left[f(x,t) - \frac{1}{2}g^2(t)s_\theta(x,t)\right]\mathrm{d}t$$

https://probml.github.io/pml-book/book2.html
Home | Let us Flow Together 🌊🚀

Diffusion Models: New Sampling Perspective

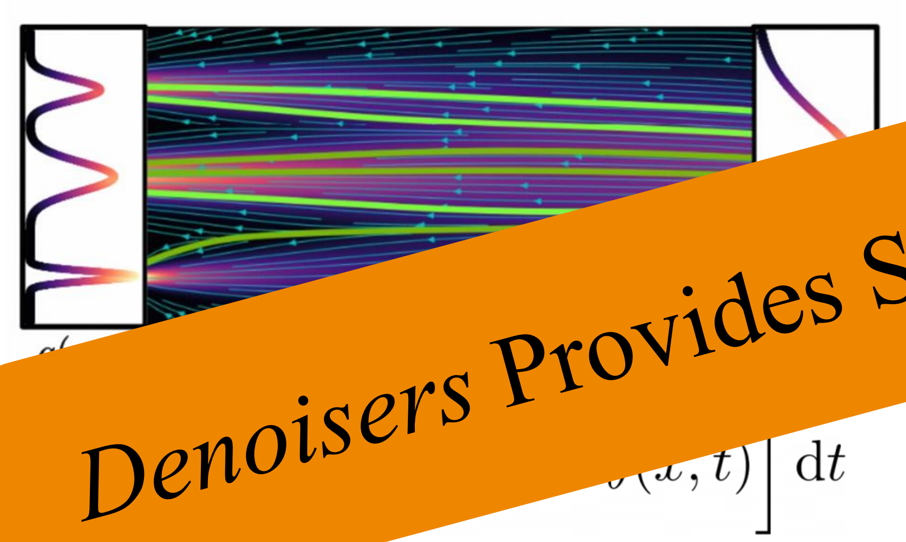


$$\mathrm{d}x = \left[f(x,t) - \frac{1}{2} g^2(t) s_\theta(x,t) \right] \mathrm{d}t$$



https://probml.github.io/pml-book/book2.html
Home | Let us Flow Together 🌊🚀

Diffusion Models: New Sampling Perspective



Denoisers Provides Sampling Directions!!

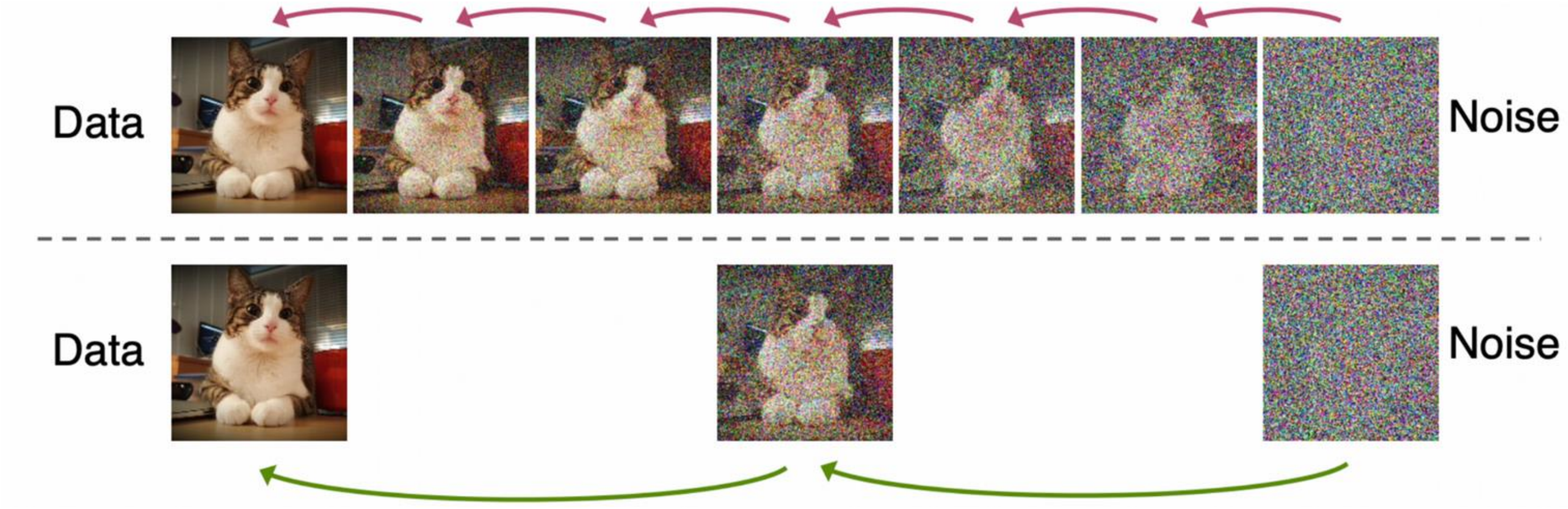
Together with Sampling Schedule

https://probml.github.io/pml-book/book2.html
[Home | Let us Flow Together 🌊🚀]()

Diffusion Models: Key Components

What do we need to sample images with diffusion models?

1. NN ***denoisers*** cover *sufficient* number of noise levels ← Necessary → direction
2. Human defined ***scheduler*** tell us how to reduce noise ← Flexible → step size



Solve ODE with different step size → numerical error

Content

- Diffusion Models
- Conditional Generation
- Image Restoration

Conditional Generation: Decomposition of the Conditional Score

$$\nabla_{x_t} \log q(x_t) \rightarrow \nabla_{x_t} \log q(x_t|c)$$

Linear decomposition of score

$$\nabla_{x_t} \log q(x_t|c) = \underbrace{\nabla_{x_t} \log q(x_t)}_{\text{unconditional score } s_\theta} + \gamma \nabla_{x_t} \log \underbrace{q(c|x_t)}_{\text{classifier likelihood } f_\phi}$$

- Utilize existing unconditional and plug in our likelihood models
- Model the conditional model directly (construct the dataset!)

https://arxiv.org/abs/2105.05233 https://sander.ai/2022/05/26/guidance.html

Conditional Generation: Classifier Guidance

$$\nabla_{x_t} \log q(x_t) \rightarrow \nabla_{x_t} \log q(x_t|c)$$

Linear decomposition of score

$$\nabla_{x_t} \log q(x_t|c) = \underbrace{\nabla_{x_t} \log q(x_t)}_{\text{unconditional score } s_\theta} + \gamma \nabla_{x_t} \log \underbrace{q(c|x_t)}_{\text{classifier likelihood } f_\phi}$$

Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $f_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \boxed{\sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log f_\phi(y|x_t)}$ //From unconditional to conditional
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$ //Perform sampling step with new direction
end for
return x_0

https://arxiv.org/abs/2105.05233 https://sander.ai/2022/05/26/guidance.html

Conditional Generation: Classifier Guidance

Off-the-shelf ImageNet classifiers

Figure 3: Samples from an unconditional diffusion model with classifier guidance to condition on the class "Pembroke Welsh corgi". Using classifier scale 1.0 (left; FID: 33.0) does not produce convincing samples in this class, whereas classifier scale 10.0 (right; FID: 12.0) produces much more class-consistent images.

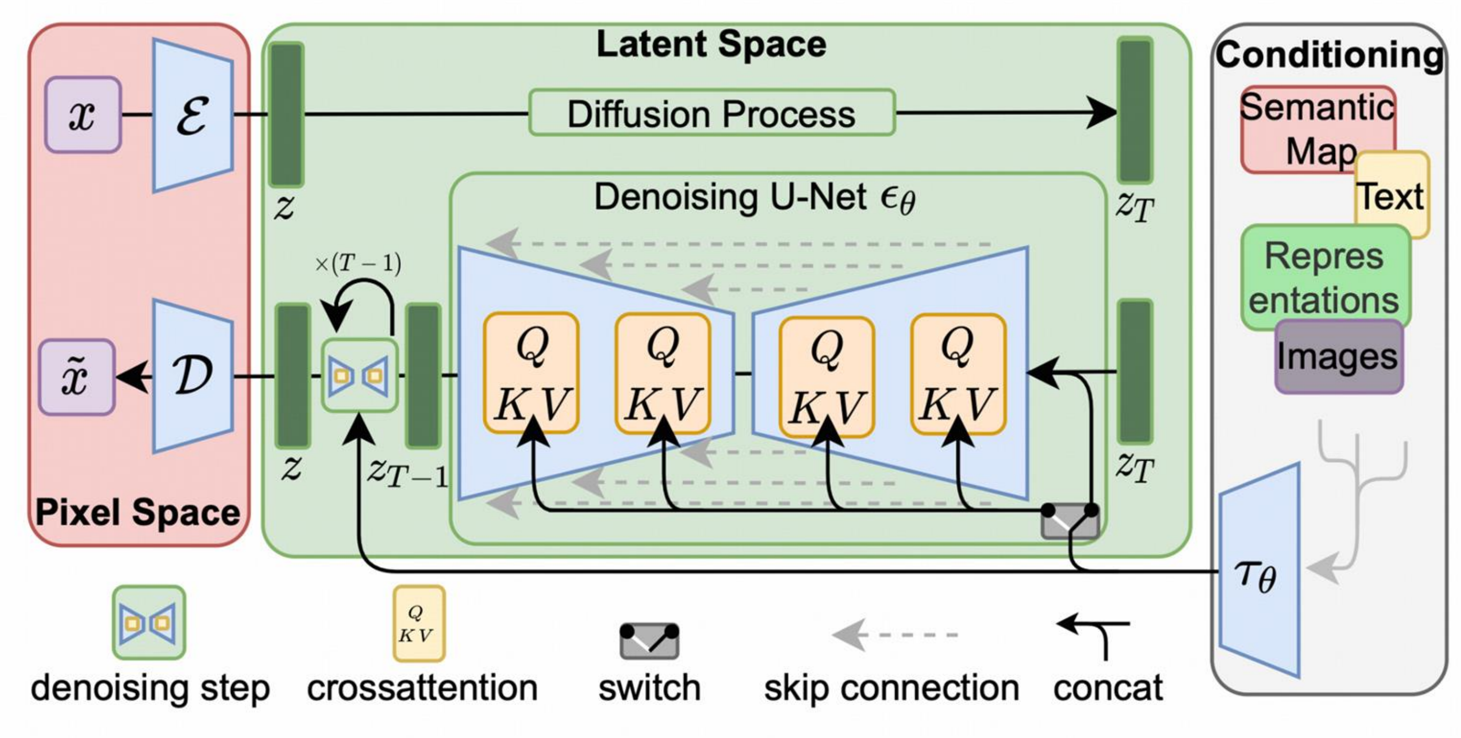
https://arxiv.org/abs/2105.05233

Conditional Generation: Latent Diffusion Models → Stable Diffusion

VAE is mainly used for compressing the data to save computation

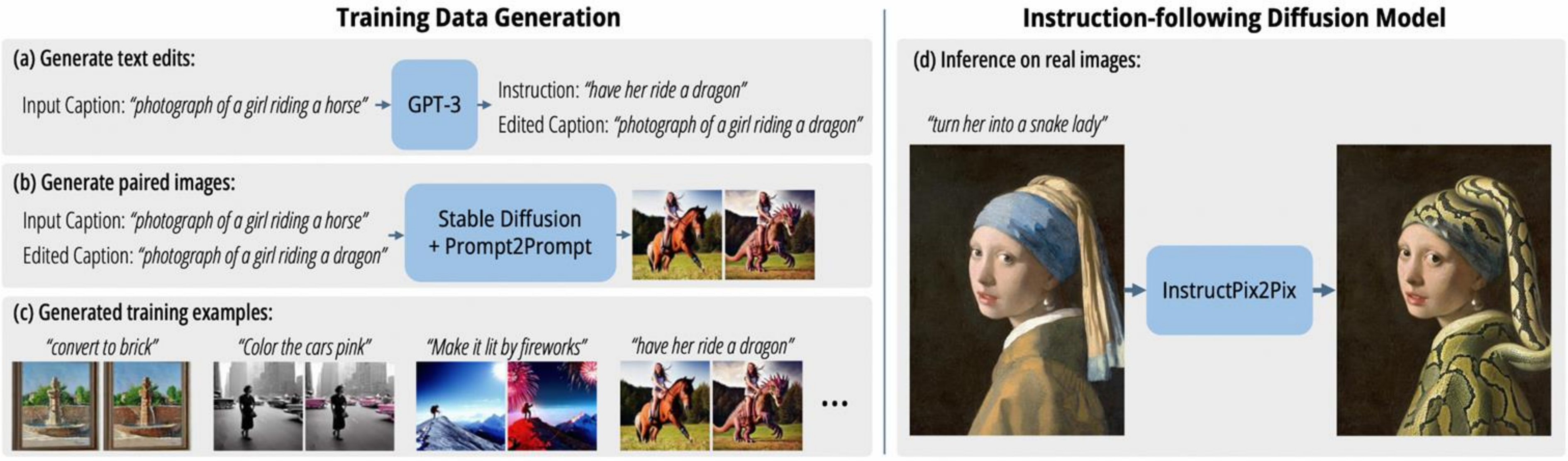
Cross Attention is used to achieve conditional generation → Model conditional score directly

$s_\theta(x_t, c)$



Conditional Generation: NN Handle Everything → Instruct Pix2Pix

Create the right dataset (supervision), NN can handle it for you!



450,000 training examples to finetune the whole model

$$\boxed{\text{output:} \quad \mathbf{z}_\theta(x_t, c_I, c_t)}$$

https://www.timothybrooks.com/instruct-pix2pix

Tim Brooks leads Sora in OpenAI!

Conditional Generation: NN Handle Everything → ControlNet

Create the right dataset (supervision), NN can handle it for you!

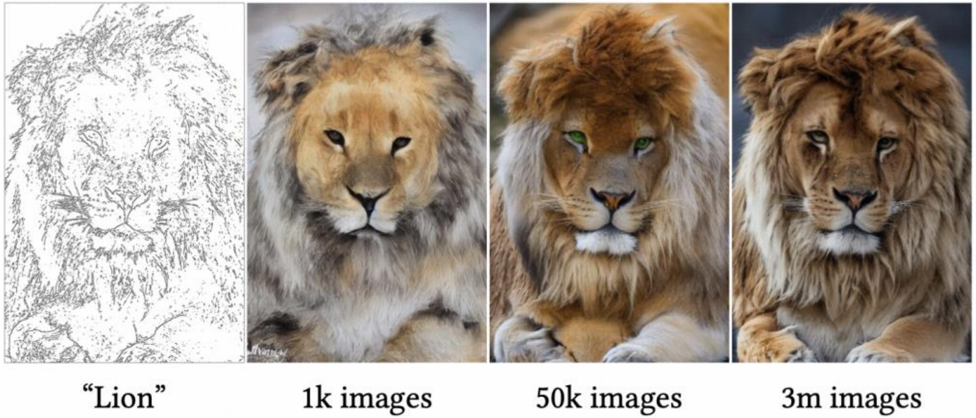
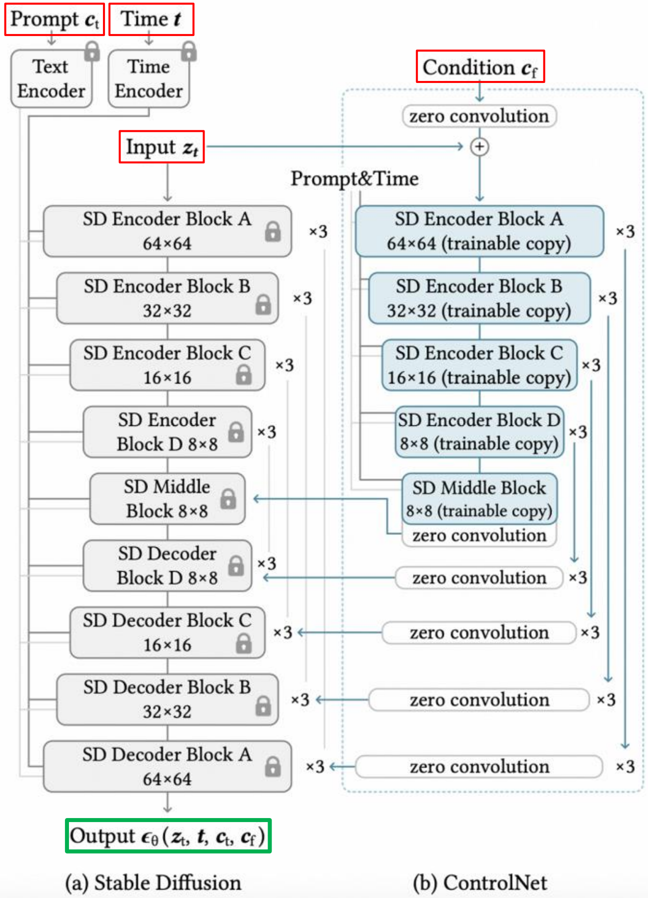


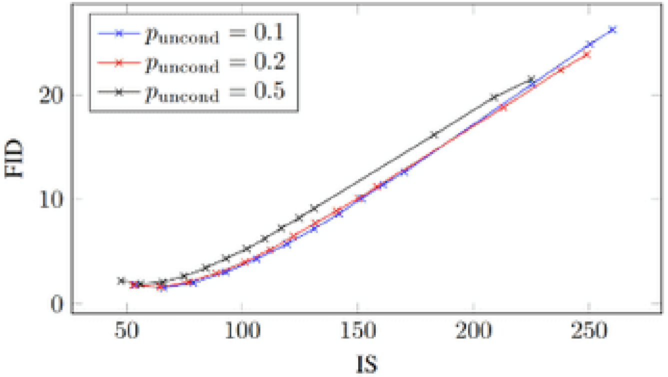
Figure 10: The influence of different training dataset sizes.



https://arxiv.org/abs/2302.05543

Classifier Free Guidance: Ultimate Hack to Diffusion Models

CLASSIFIER-FREE DIFFUSION GUIDANCE



$$\nabla_x \log p'(x_t \mid c) = \nabla_x \log p(x_t) + w\Big(\underbrace{\nabla_x \log p(x_t \mid c)}_{\text{conditional score}} - \underbrace{\nabla_x \log p(x_t)}_{\text{unconditional score}}\Big).$$

[2207.12598] Classifier-Free Diffusion Guidance
Guidance: a cheat code for diffusion models – Sander Dieleman
[2406.02507] Guiding a Diffusion Model with a Bad Version of Itself

Content

- Diffusion Models
- Conditional Generation
- Image Restoration

Image Restoration: A Special Kind Conditional Generation

For image restoration task, the measurement y is known

$$p_\theta(x_{t-1}|x_t)\begin{cases}\mathbf{x}_0^{(t)} = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{(1-\bar{\alpha}_t)}\mathbf{z}_\theta(\mathbf{x}_t, t)) \\ \hat{\mathbf{x}}_0^{(t)} = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathcal{H}(\mathbf{x})\|^2 + \rho_t\|\mathbf{x} - \mathbf{x}_0^{(t)}\|^2 \\ \mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{\mathbf{x}}_0^{(t)}(\mathbf{x}_t, \mathbf{y}) + \sqrt{1-\bar{\alpha}_{t-1}}(\sqrt{1-\zeta}\hat{\mathbf{z}}(\mathbf{x}_t, \mathbf{y}) + \sqrt{\zeta}\epsilon_t)\end{cases}$$

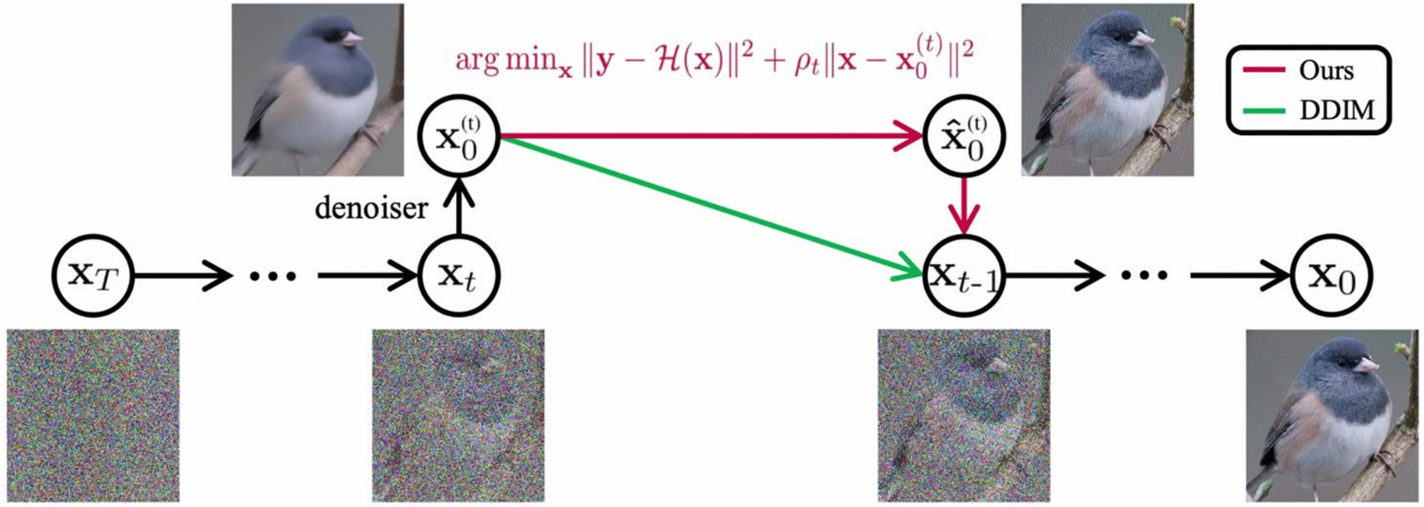
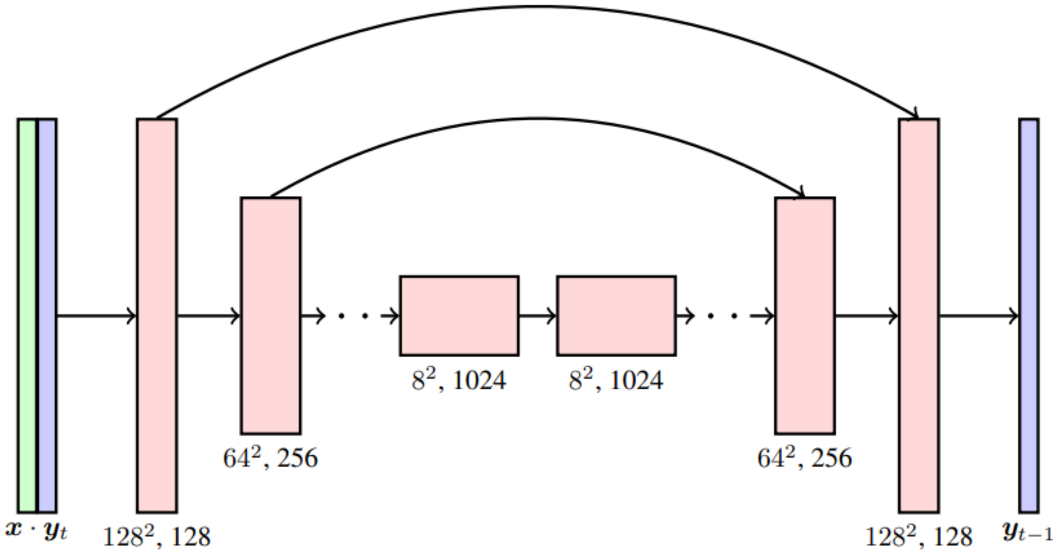


Image Super-Resolution via Iterative Refinement*

The condition is concatenated with y_t along the channel dimension (cascaded)

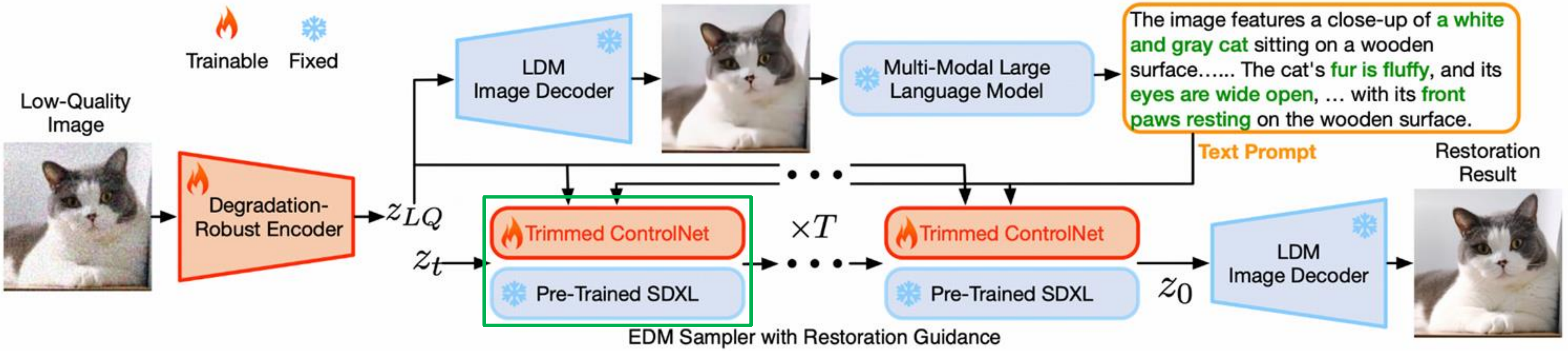


Same author also proposed palette for multi-tasks†, same architecture used for cascaded diffusion‡

*[2104.07636] Image Super-Resolution via Iterative Refinement (arxiv.org)
†[2111.05826] Palette: Image-to-Image Diffusion Models (arxiv.org)
‡[2106.15282] Cascaded Diffusion Models for High Fidelity Image Generation (arxiv.org)

Image Restoration: SUPIR → ControlNet Alike Method

Provide diffusion UNet+ControlNet with LQ image and I2T caption



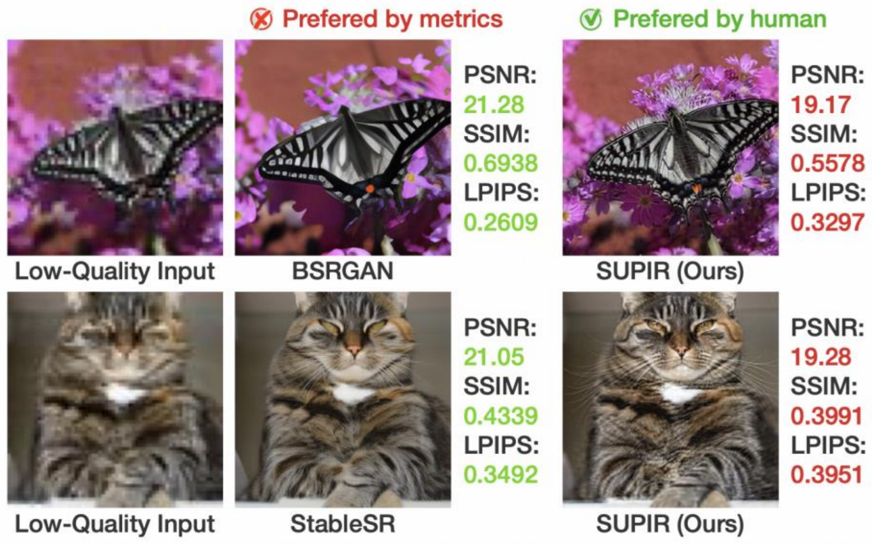
model input: $z_t \quad z_{LQ} \quad c_{\text{text}}$

model target: z_0

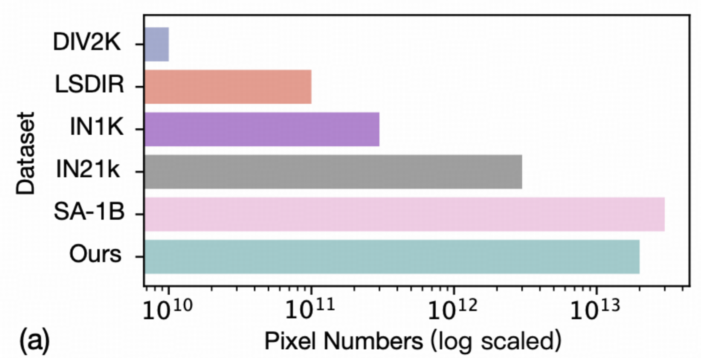
http://supir.xpixel.group/

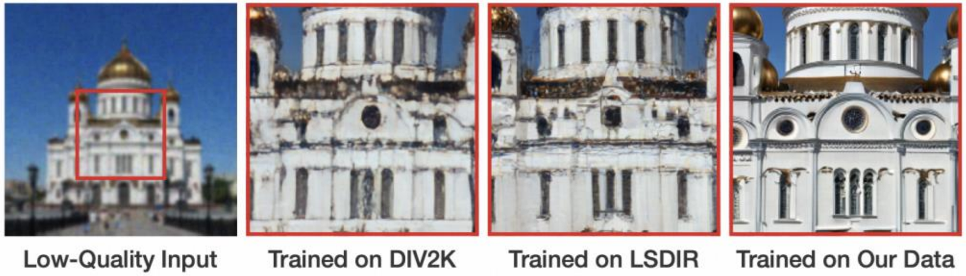
Image Restoration: SUPIR → ControlNet Alike Method

Misalignment between human and metric



Dataset size matters





http://supir.xpixel.group/

https://arxiv.org/pdf/2401.13627

The End

Diffusion Models: Euler ODE Solver

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} z_t$$

From noise level *t* to *s*

$$\begin{aligned} x_t &= \alpha_t x_0 + \sigma_t \epsilon \\ x_s &= \alpha_s x_0 + \sigma_s \epsilon' \end{aligned} \implies x_s = \frac{\sigma_s}{\sigma_t} x_t + \left(\alpha_s - \alpha_t \frac{\sigma_s}{\sigma_t} \right) D_\theta$$

Can verify this is the same as deterministic DDIM

$$x_s = \alpha_s D_\theta + \sigma_s \mathbf{z}_\theta$$

https://arxiv.org/abs/2010.02502