

Diffusion Models

for Conditional Generation & Visual Restoration

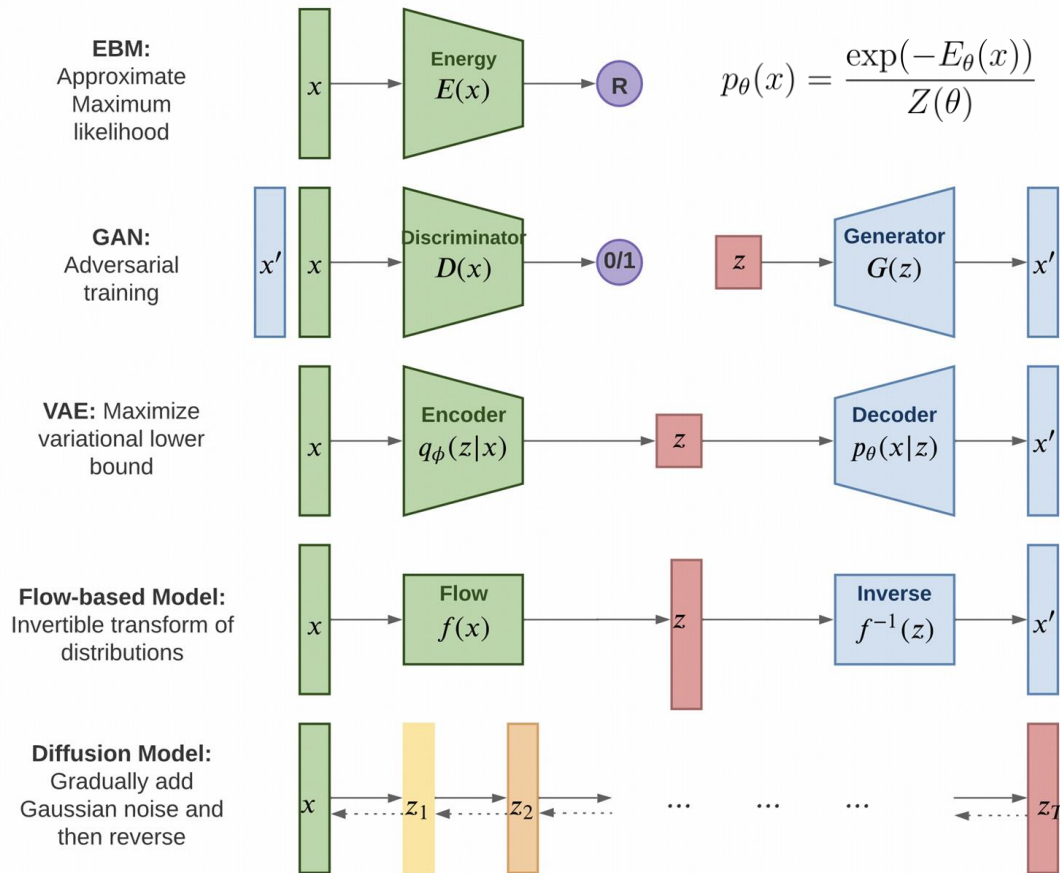
Yuanzhi Zhu

Supervisors: Prof. Kai Zhang

Content

- Diffusion Models
- Conditional Generation
- Image Restoration

A Tale of Generative Models as Mapping Connecting Distributions



Model the distribution?

→ hard to draw new samples

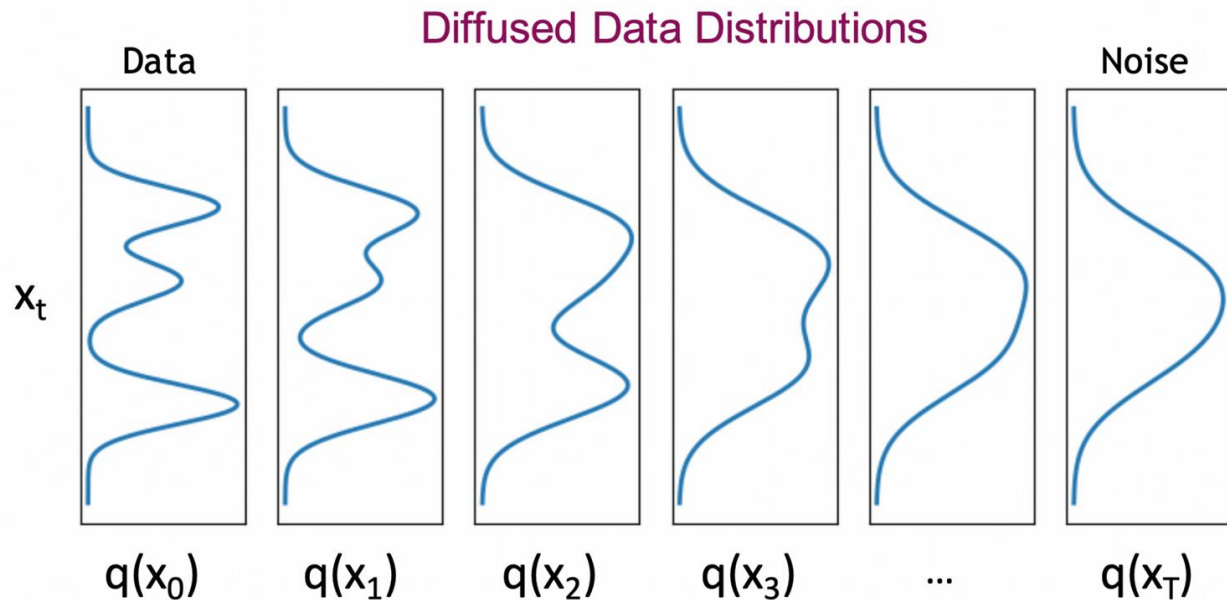
Model the mapping!

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 $q(x_T)$



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

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

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

Interpolation!

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

$x_0 \sim q(x_0)$

$z_t \sim q(x_T)$

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 \mathbf{x}_0 or \mathbf{z}_t to perform a reverse step

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

$$\text{😊} \left\{ \begin{array}{l} D_{\theta}(x_t, t) \\ \mathbf{z}_{\theta}(x_t, t) \end{array} \right.$$

Diffusion Models: Divide and Conquer → Equivalent Objectives

The training objective of different diffusion / score methods are equivalent

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

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

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: A Set of *Denoisers* at Different Noise Level

The training objective of different diffusion / score methods are equivalent

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

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

define *score* as the

$$\begin{aligned} & \frac{\partial \log p(x_t | x_0)}{\partial x_t} \\ &= - \frac{x_t - \sqrt{\bar{\alpha}_t}x_0}{1 - \bar{\alpha}_t} \end{aligned}$$

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 are Trained as Denoisers!!

Diffusion Models: Training & Sampling



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

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:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \mathbf{z}_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$   
5: end for perform a reverse step  
6: return  $\mathbf{x}_0$ 
```

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

Diffusion Models: Training & Sampling

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  
    $\nabla_{\theta} \left\| \epsilon - \mathbf{z}_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t) \right\|^2$   
6: until converged  $\mathbf{x}_t$ 
```

```
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 * x_0  
    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()
```

Algorithm 2 Sampling

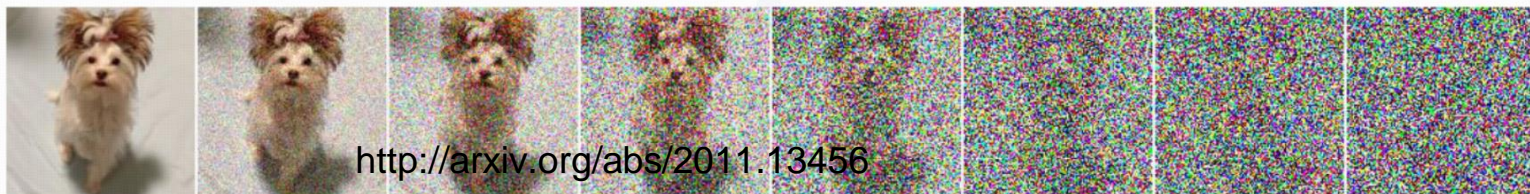
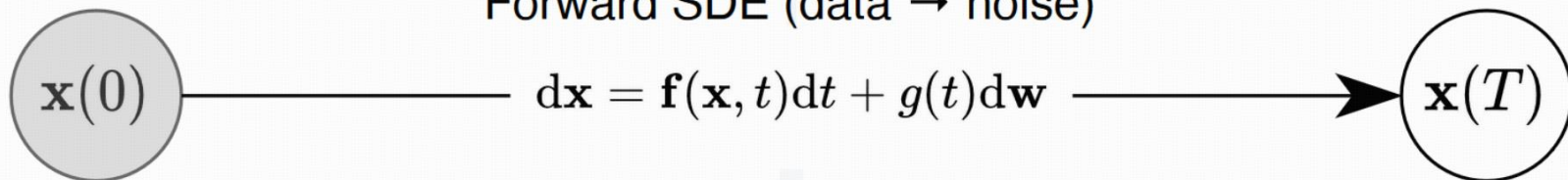
```
1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$   
2: for  $t = T, \dots, 1$  do  
3:  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$   
4:  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1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

```
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)
```

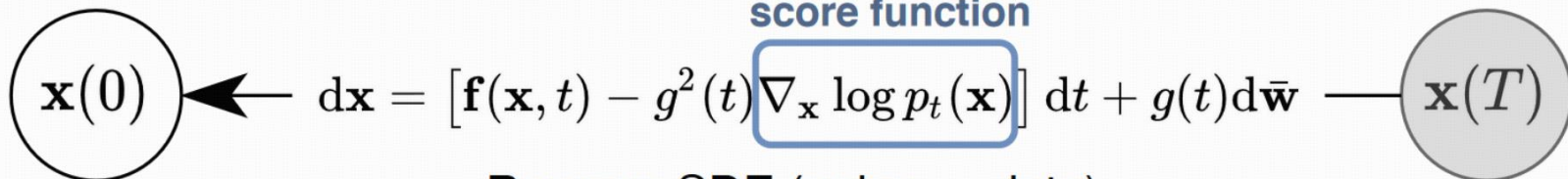
Diffusion Models: From Discrete to Continuous

f and g are known,
related to schedule

Forward SDE (data \rightarrow noise)



score function

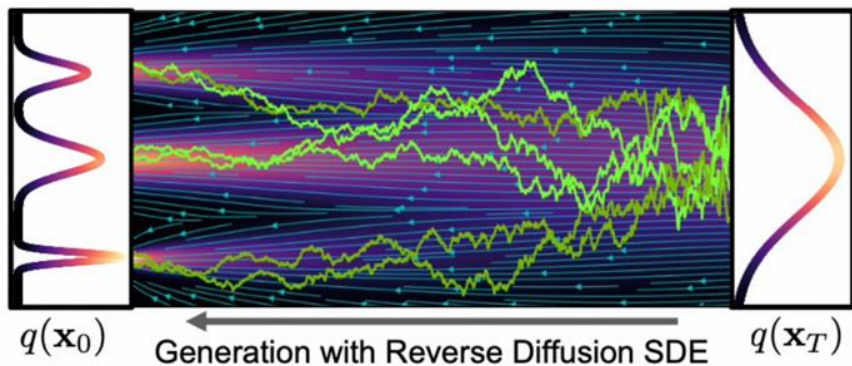


Reverse SDE (noise \rightarrow data)

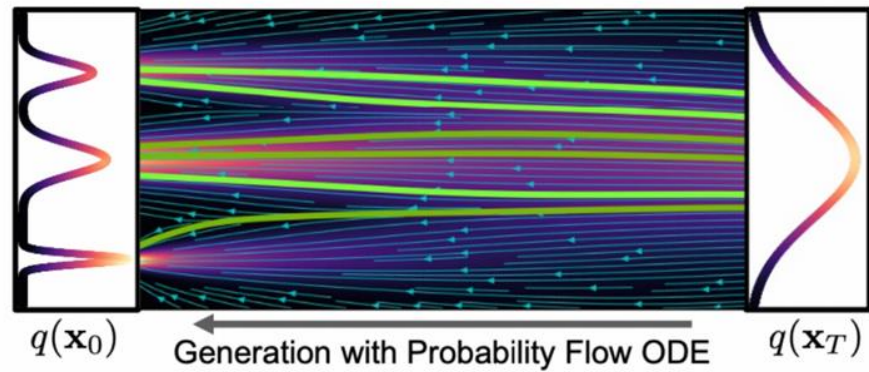
Diffusion Models: New Sampling Perspective

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

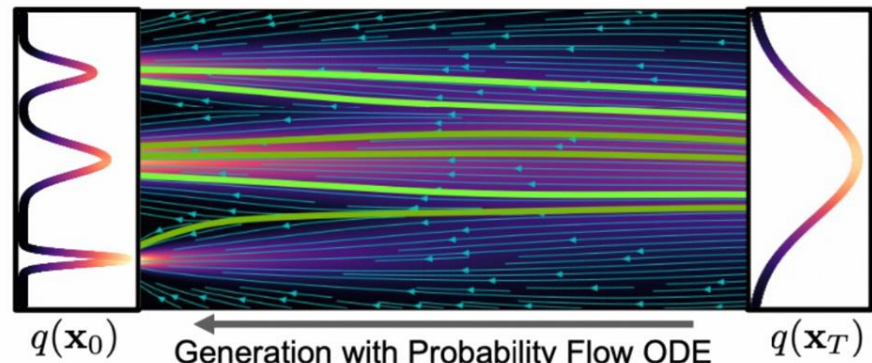


$$dx = [f(x, t) - g^2(t)s_\theta(x, t)]dt + g(t)dw$$

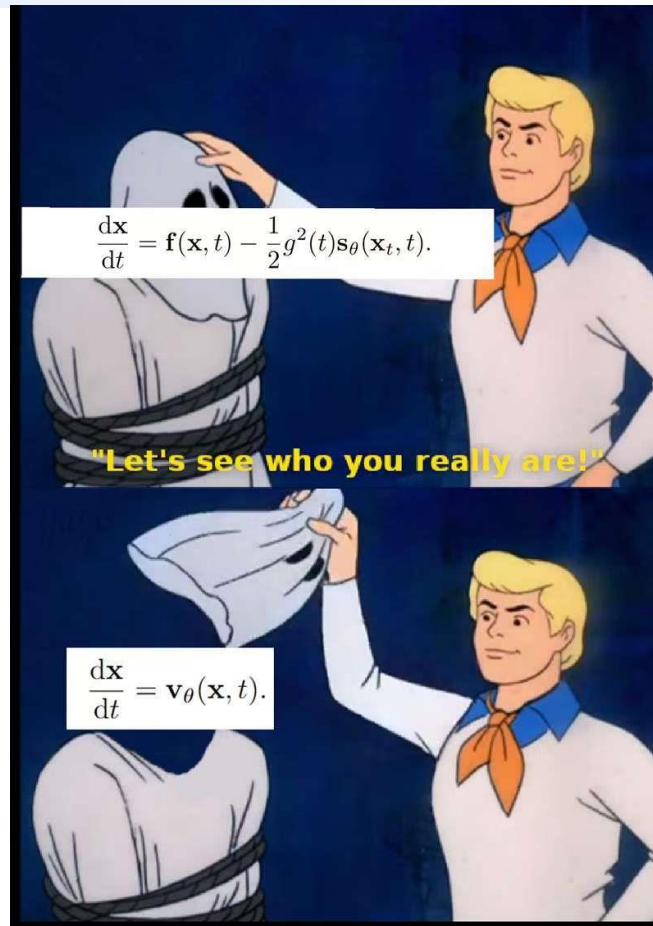


$$dx = \left[f(x, t) - \frac{1}{2}g^2(t)s_\theta(x, t) \right] dt$$

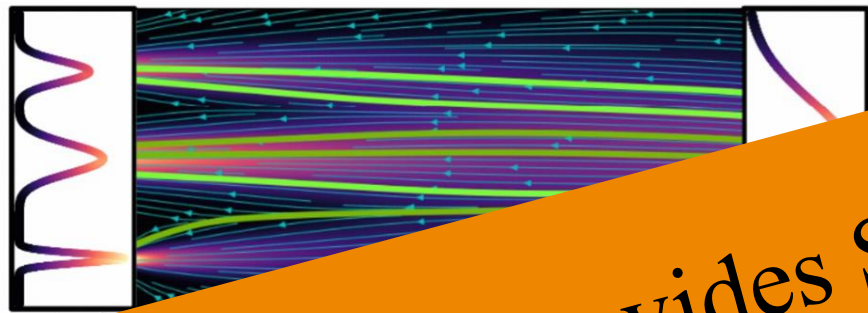
Diffusion Models: New Sampling Perspective



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



Diffusion Models: New Sampling Perspective



Denoisers Provides Sampling Directions!!

$$\left[\frac{d\mathbf{x}}{dt} = \mathbf{v}_\theta(\mathbf{x}, t) \right] dt$$

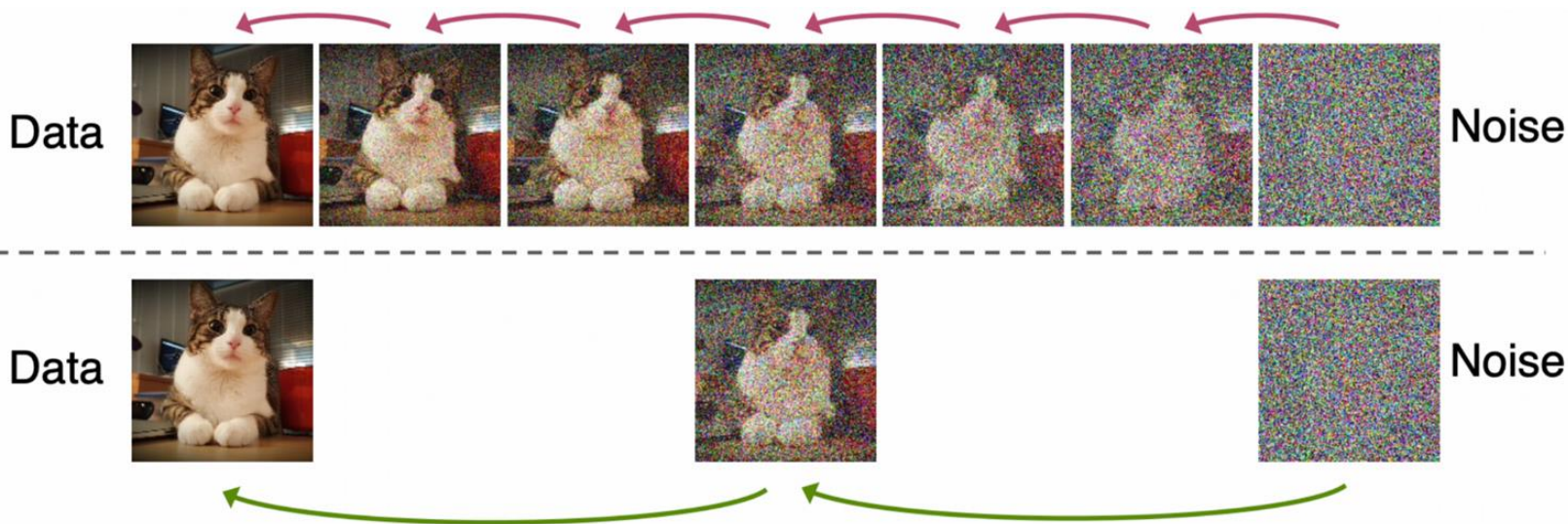
Together with Sampling Schedule



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: Classifier Guidance

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

Linear decomposition of score

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

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**

$\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1 - \bar{\alpha}_t} \nabla_{x_t} \log f_\phi(y|x_t)$

//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

Conditional Generation: Classifier Guidance

Off-the-shelf ImageNet classifiers



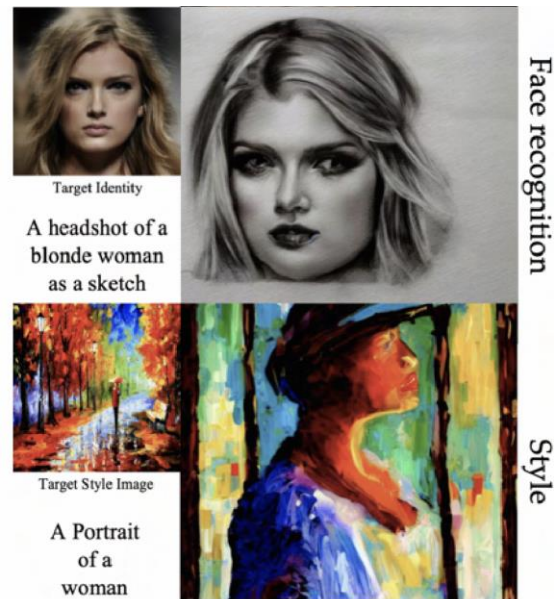
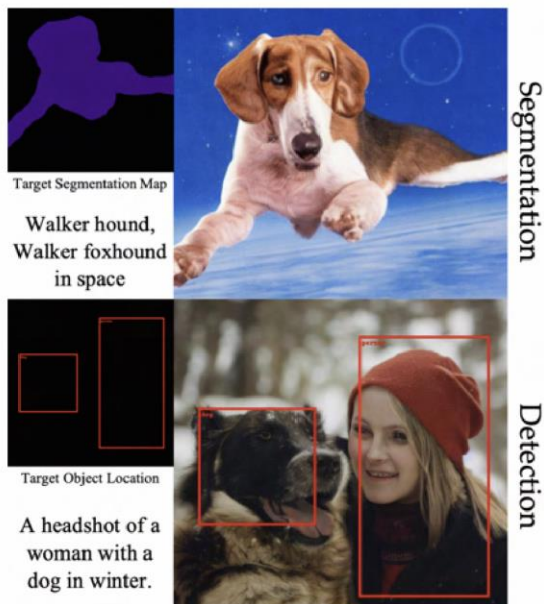
Figure 3: Samples from an unconditional diffusion model with classifier guidance to condition on the class "Pembroke Welsh corgi". Using classifier scale 1.0 (left; FID: 33.0) does not produce convincing samples in this class, whereas classifier scale 10.0 (right; FID: 12.0) produces much more class-consistent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

Conditional Generation: Likelihood Constraints

Generalize classifier guidance to constraints / loss applied on x_t and condition \mathcal{C}

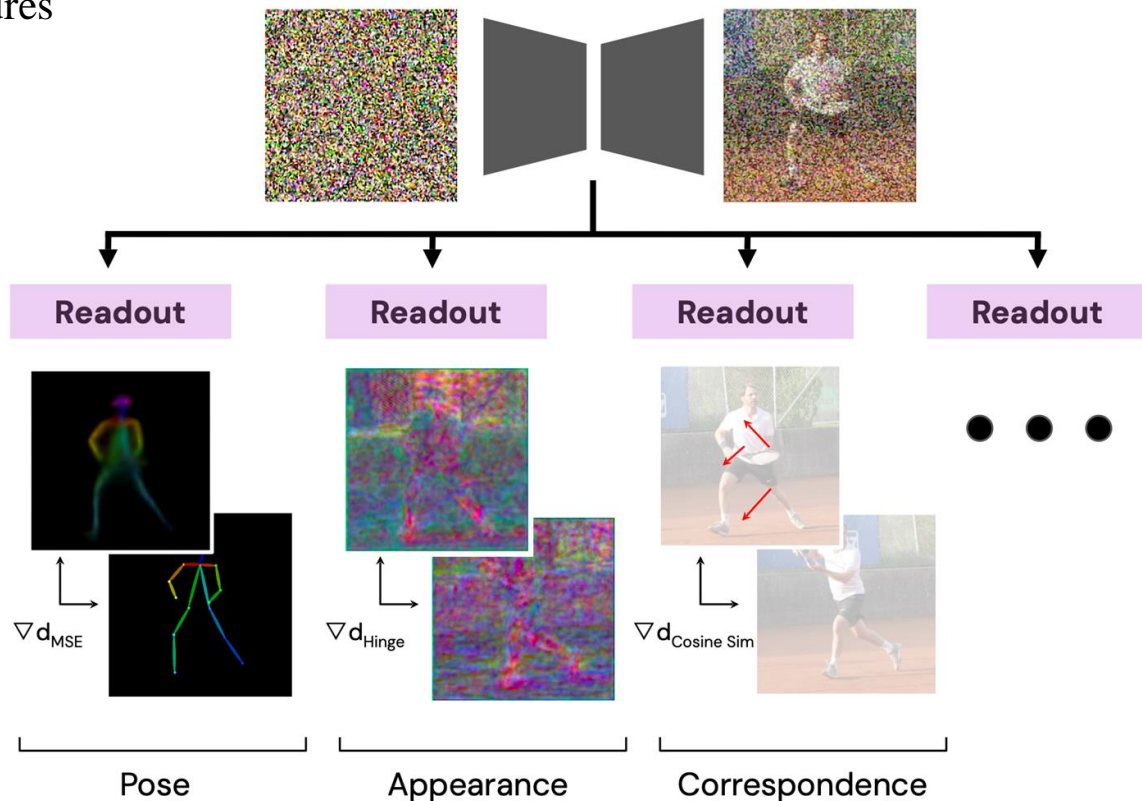
external guidance function f and the loss function l

$$s_{\theta}(x_t, c) = s_{\theta}(x_t) + \gamma \nabla_{x_t} \ell(c, f(x_t))$$



Conditional Generation: More General Constraints

Learning Control from Diffusion Features



our guidance function as a distance between the reference r and our predicted readout $\hat{r} = f_{\psi}(x_t)$.

$$\hat{\epsilon}_t \leftarrow \epsilon_{\theta}(x_t) + w \cdot \nabla_{x_t} d(r, f_{\psi}(x_t))$$

<https://arxiv.org/abs/2312.02150>

<https://readout-guidance.github.io/>

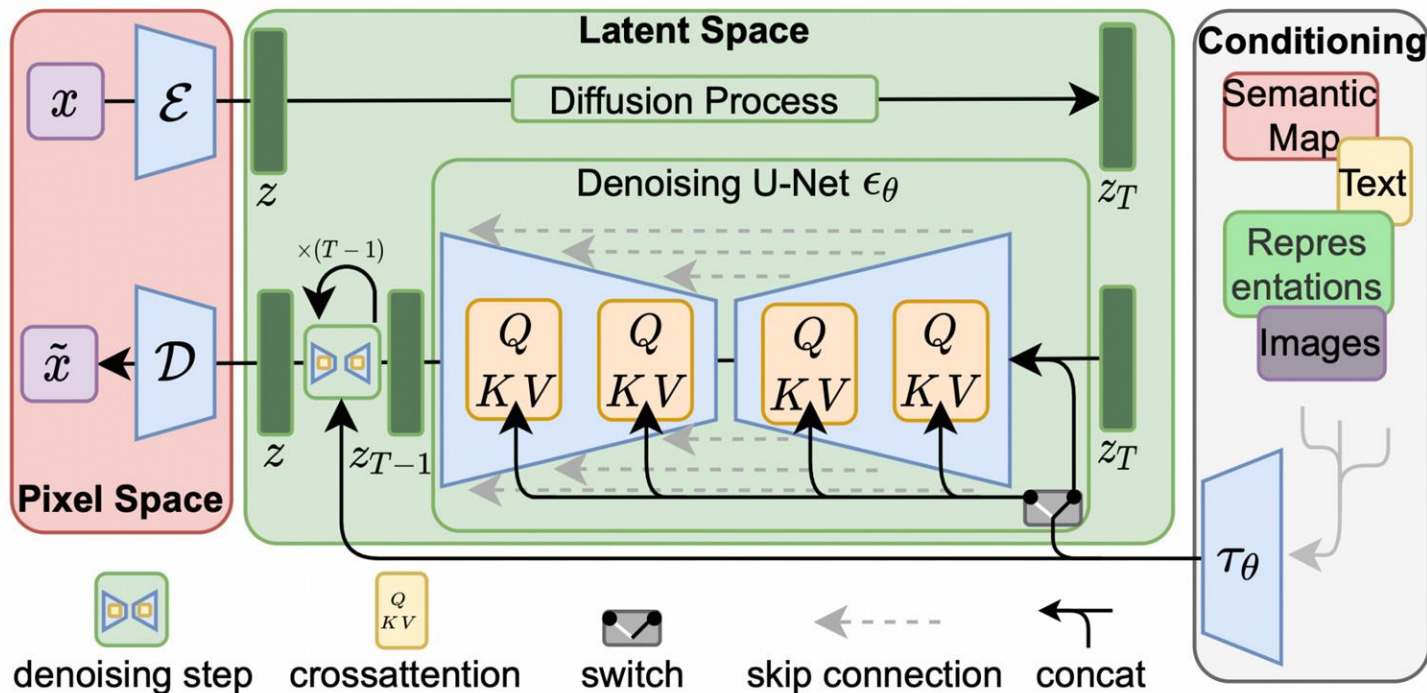
<https://arxiv.org/abs/2306.00986>

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 probability directly

$$s_{\theta}(x_t, c)$$



Conditional Generation: NN Handle Everything → Instruct Pix2Pix


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

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** → 

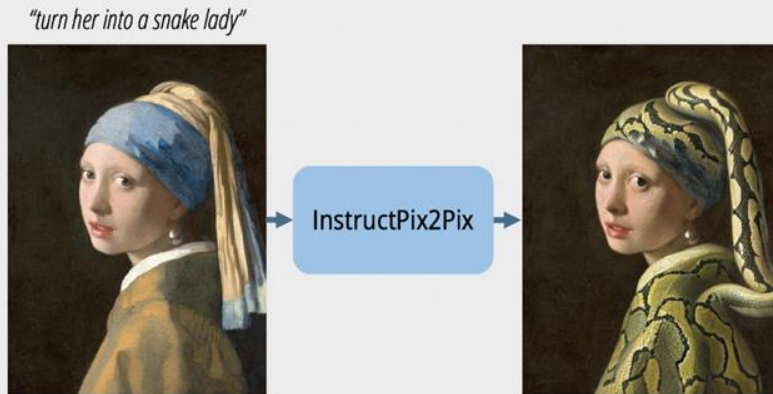
(c) Generated training examples:



450,000 training examples to finetune the whole model

Instruction-following Diffusion Model

(d) Inference on real images:



output: $\mathbf{Z}_{\theta}(x_t, c_I, c_t)$

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

Tim Brooks leads Sora in OpenAI!

Conditional Generation: NN Handle Everything \rightarrow ControlNet

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



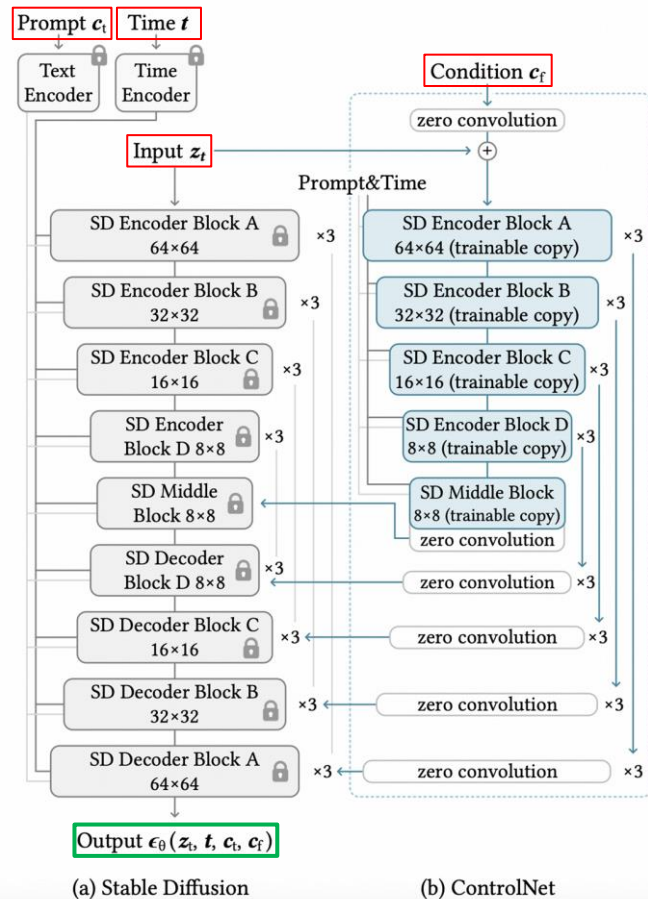
"Lion"

1k images

50k images

3m images

Figure 10: The influence of different training dataset sizes.



(a) Stable Diffusion

(b) ControlNet

Content

- Diffusion Models
- Conditional Generation
- Image Restoration

Image Restoration: A Special Kind Conditional Generation

For image restoration task, the measurement y is known

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

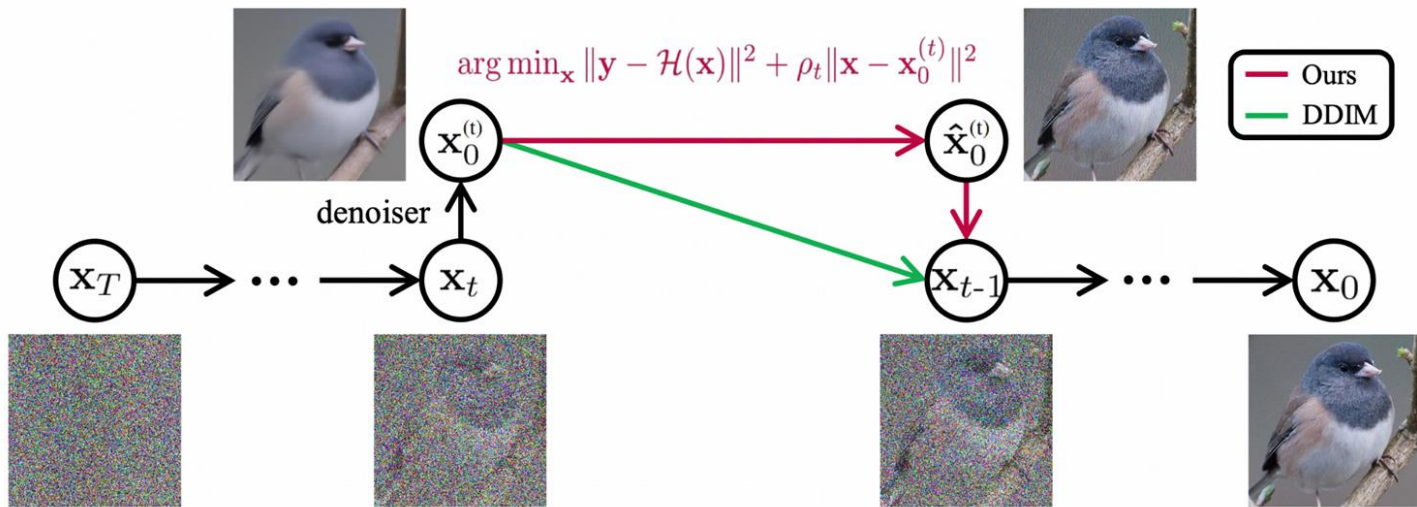
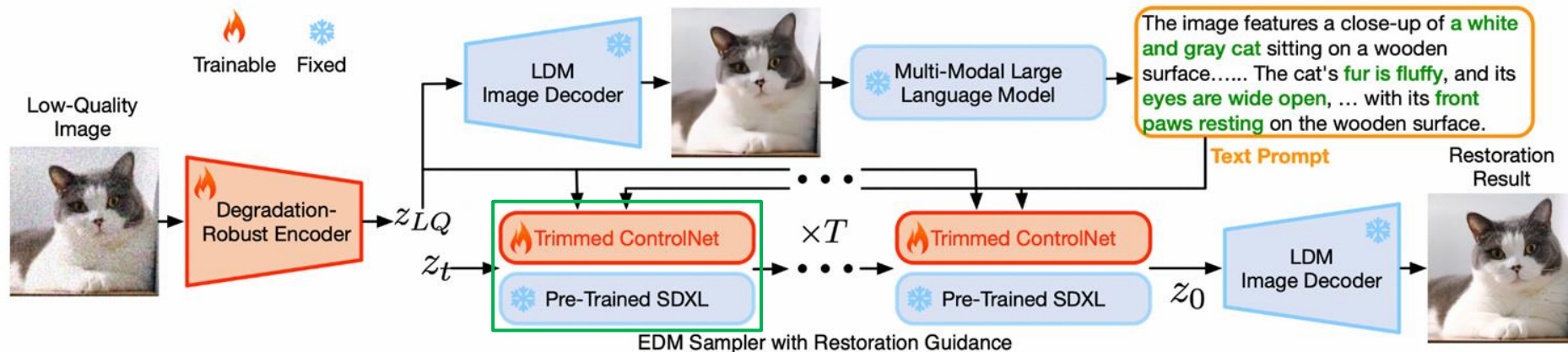


Image Restoration: SUPIR → ControlNet Alike Method

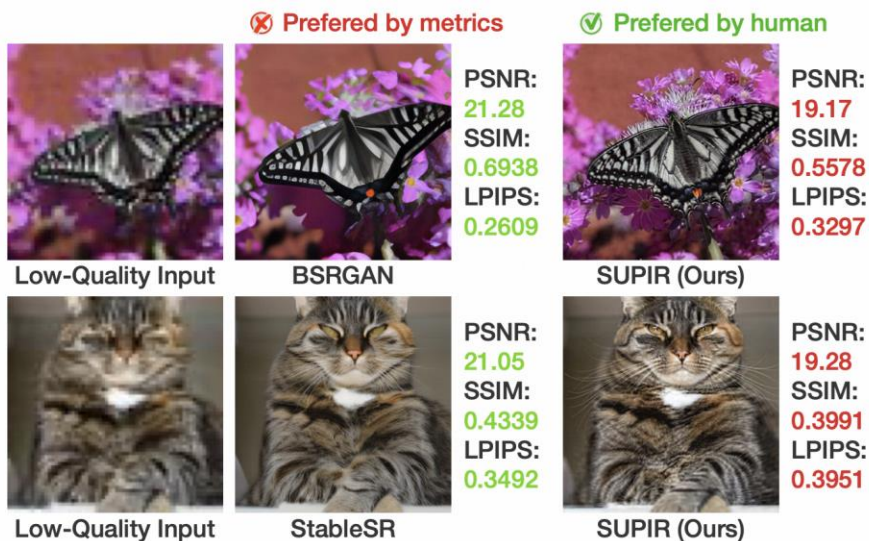
Provide diffusion UNet+ControlNet with LQ image and I2T caption



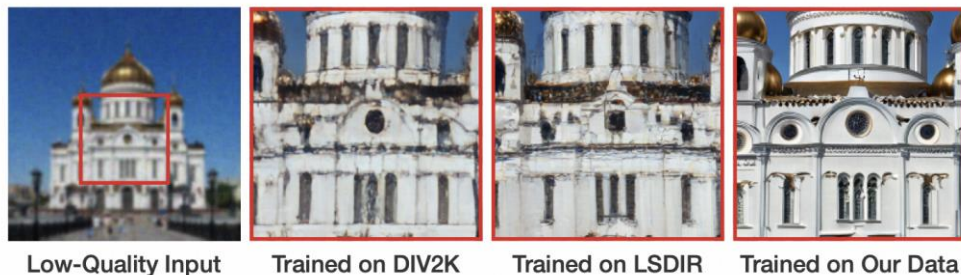
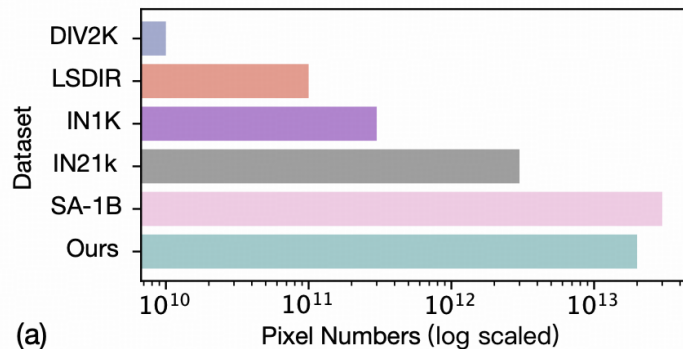
 model input: z_t z_{LQ} c_{text}
model target: z_0

Image Restoration: SUPIR → ControlNet Alike Method

Misalignment between human and metric



Dataset size matters



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

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