# InstaFlow:
## One Step is Enough for High-Quality Diffusion-Based Text-to-Image Generation

Authors:
Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, Qiang Liu

Presenter:
James Soole


# Text-to-Image
DALL-E 2 (OpenAI)


# Unconditional Generation
Latent Diffusion


# Customizing Images
DreamBooth (Google)

# Inpainting
RePaint


# Stylized Images
LoRA on SDXL (UIUC,Google)

- Denoising
- Super Resolution
- ….

# Background: Diffusion Summary

## Diffusion

- Diffusion Probabilistic Models (2015, Sohl-Dickstein et al.)
- Denoising Diffusion Probabilistic Models (2019, Ho et al.)

## Latent Diffusion

- Latent Diffusion (Rombach et al., 2021)
    - → Stable Diffusion

## Even More Diffusion

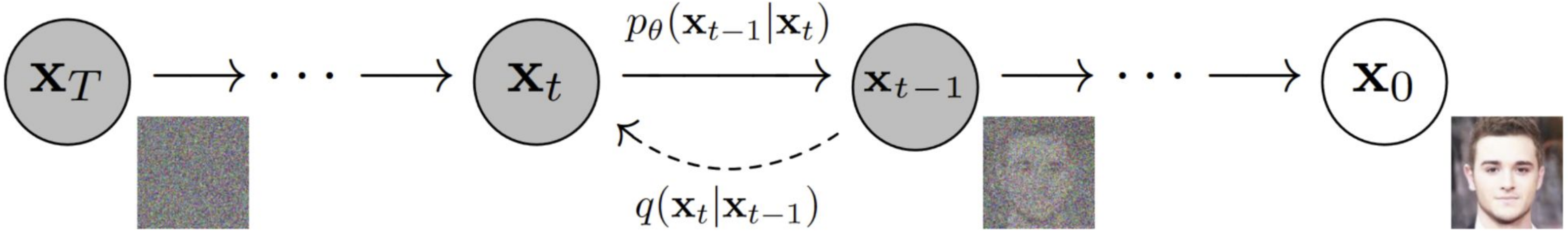
- InstaFlow (this)
- and many more …

Connections To:
Nonequilibrium Thermodynamics
Langevin dynamics
Score Matching

# Background: **Diffusion**

**Reverse process *p***

turns noise into images

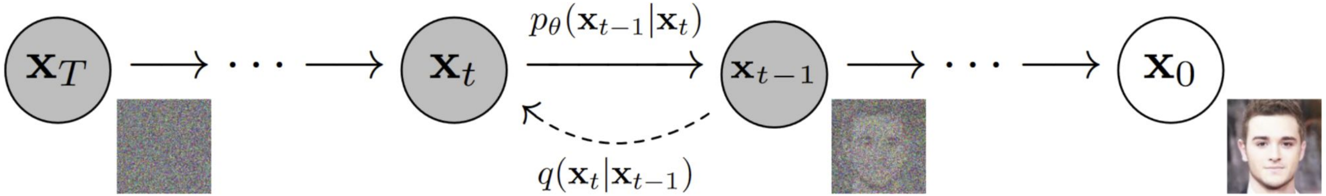



**Forward process *q***

turns images into Gaussian noise

- Provided transitions t are small enough, $p_\theta(x_{t-1}| x_t)$ is gaussian
- we can train a neural network to estimate $x_{t-1}$ from $x_t$

# Background: Diffusion




## Forward Process (←)

- Gradually add gaussian noise
  - B varies by timestep (from $10^{-4}$ to 0.02)
  - until the image approaches gaussian noise
- Can sample in closed form

**constant**

$$q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)=\mathcal{N}\left(\mathbf{x}_t ; \sqrt{1-\beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$$

## Reverse Process (→)

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

**predicted**

- Gradually predict the added noise
  - our trained model predicts noise **mean** and **variance**
- Reconstruct image by subtracting predicted noise

# Background: Diffusion

**Algorithm 1** Training

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

add some amount of noise

predict noise

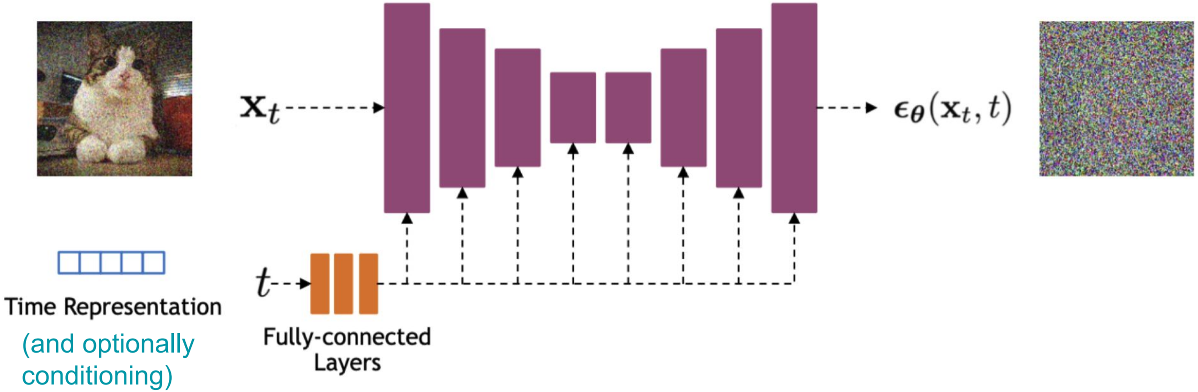
**Algorithm 2** Sampling

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

predict noise

subtract noise

# Background: Diffusion

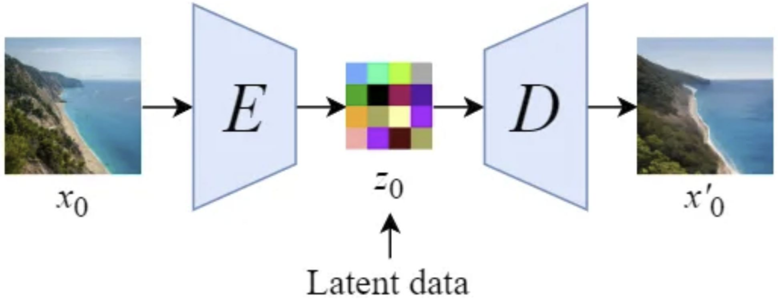


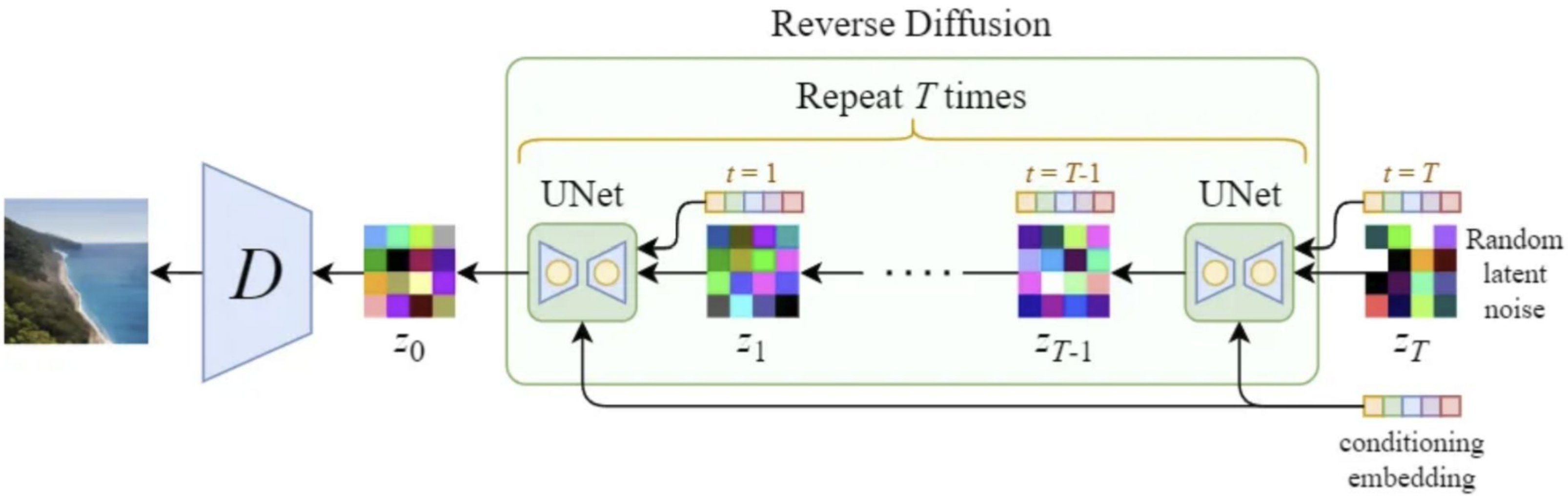

Hindrances:

- Iterative
- Image-sized inputs + latents

# Background: **Latent Diffusion**

- Encode image into smaller latent space
- Run diffusion in latent space

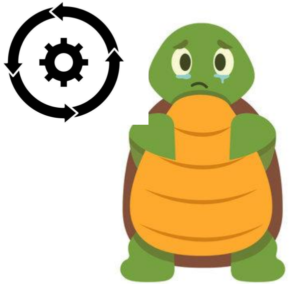


# Motivation

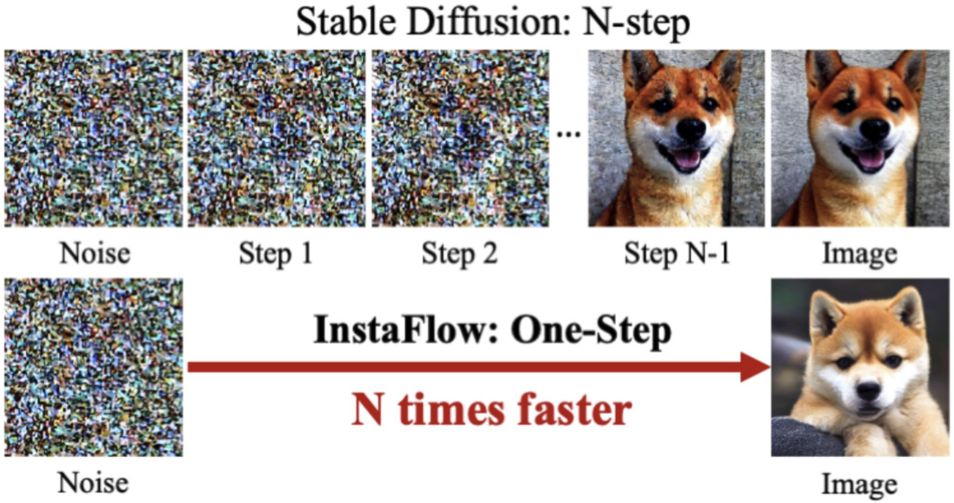
Sampling is still **iterative**

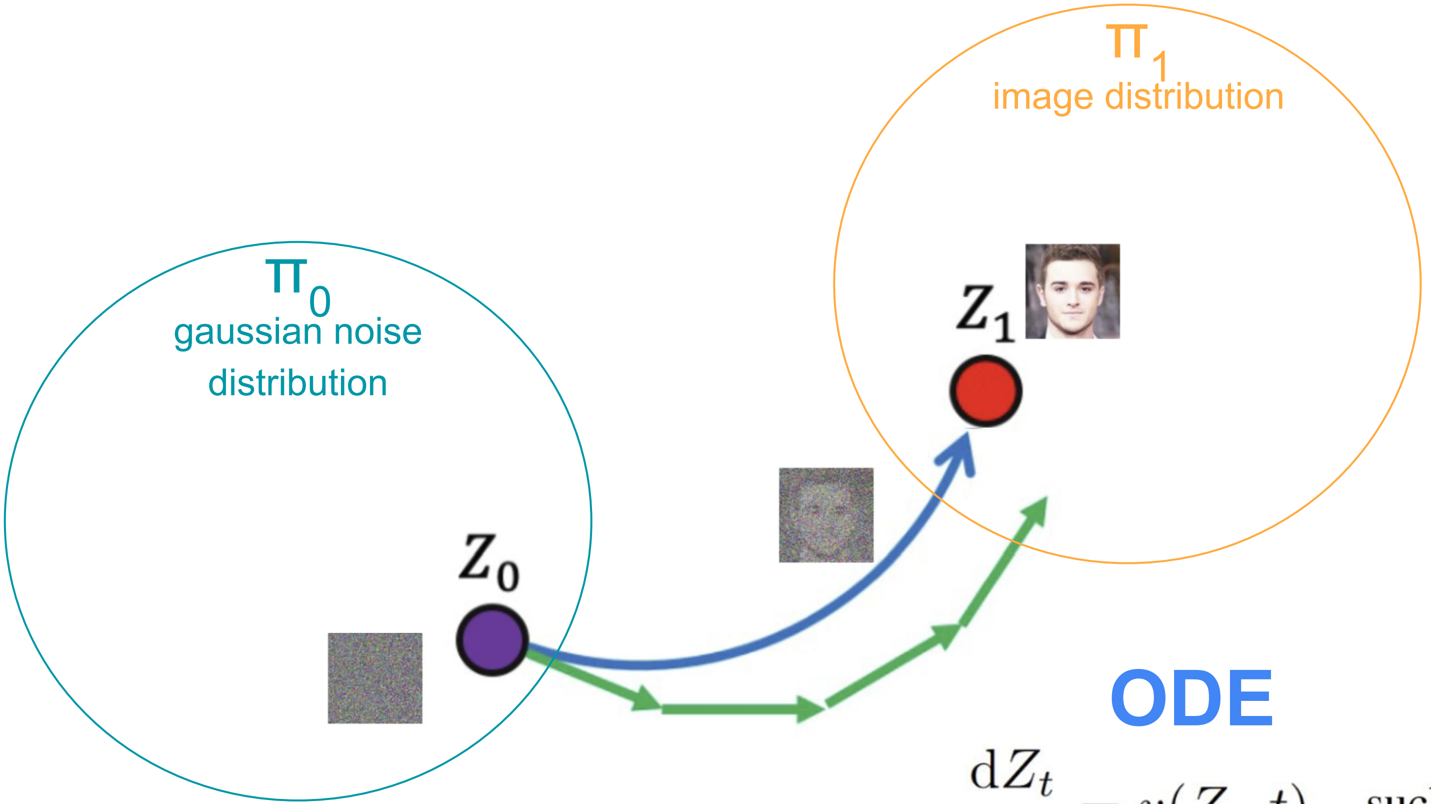
That makes it **slow**



**InstaFlow** claims

- “an ultra-fast one-step model”
- “with SD-level image quality”

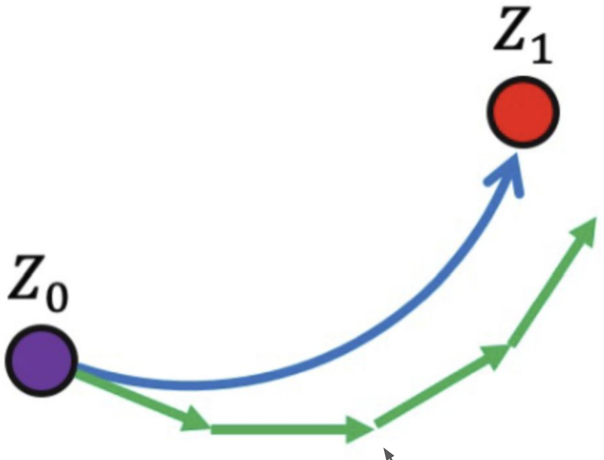




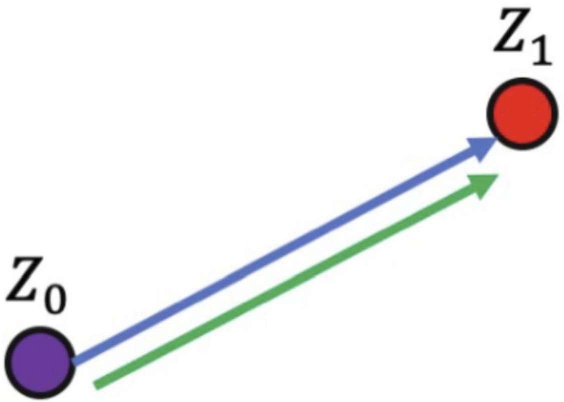

**ODE**

$$\frac{\mathrm{d}Z_t}{\mathrm{d}t} = v(Z_t, t), \quad \text{such that } Z_1 \sim \pi_1$$
$$Z_0 \sim \pi_0$$

**v is the ODE's velocity field**




$$Z_{t+\frac{1}{N}} = Z_t + \frac{1}{N} v(Z_t, t),$$




**Perfect Simulation in just one step**

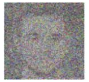
Rectified Flow: an **ODE** Framework

- Learns to transfer $\pi_0$ to $\pi_1$
- approximated by numerical solvers (e.g. Euler Method)
- choice of N yields a cost-accuracy trade-off
- **want a straight flow**

Neural Net

$$\min_{v} \mathbb{E}_{(X_0, X_1)\sim\gamma}\left[\int_0^1 \|\, \frac{\mathrm{d}}{\mathrm{d}t} X_t - v(X_t, t)\, \|^2 \,\mathrm{d}t\right]$$

Can choose different interpolations for $X_t$



$$X_t = \phi(X_0, X_1, t)$$

$$\boxed{X_t = tX_1 + (1-t)X_0}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} X_t = X_1 - X_0$$

can use any time-differentiable interpolation

DDIM uses this one

$$= \min_{v} \mathbb{E}_{(X_0, X_1)\sim\gamma}\left[\int_0^1 \|\, (X_1 - X_0) - v(X_t, t)\, \|^2 \,\mathrm{d}t\right]$$

# Rectified Flow

**Now we have v,**
approximated by a neural net

**We can now solve this ODE**
starting from $Z_0 \sim \pi_0$ to transfer $\pi_0$ to $\pi_1$

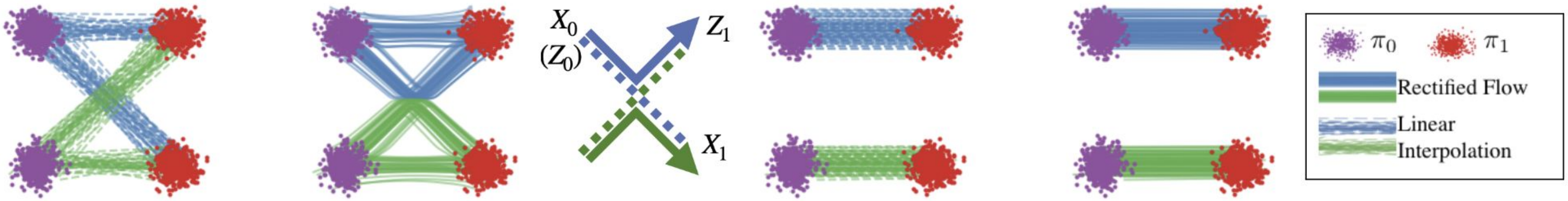
$$\frac{\mathrm{d}Z_t}{\mathrm{d}t} = v(Z_t, t),$$

$$(Z_0, Z_1) = \texttt{Rectify}((X_0, X_1))$$

can be an arbitrary coupling of $\pi_0, \pi_1$

the rectified coupling

why is this any better?




(a) Linear interpolation $X_t = tX_1 + (1-t)X_0$

(b) Rectified flow $Z_t$ induced by $(X_0, X_1)$

(c) Linear interpolation $Z_t = tZ_1 + (1-t)Z_0$

(d) Rectified flow $Z'_t$ induced by $(Z_0, Z_1)$

Figure 2: (a) Linear interpolation of data input $(X_0, X_1) \sim \pi_0 \times \pi_1$. (b) The rectified flow $Z_t$ induced by $(X_0, X_1)$; the trajectories are "rewired" at the intersection points to avoid the crossing. (c) The linear interpolation of the end points $(Z_0, Z_1)$ of flow $Z_t$. (d) The rectified flow induced from $(Z_0, Z_1)$, which follows straight paths.

- Paths for a well-defined ODE cannot cross each other
  - That would mean the ODE solution is not unique
- **Rectified Flow rewires individual trajectories to avoid crossing**
  - Still keeps the same density map

**Flows avoid crossing** A key to understanding the method is the non-crossing property of flows: the different paths following a well defined ODE $\mathrm{d}Z_t = v(Z_t, t)\mathrm{d}t$, whose solution exists and is unique, *cannot cross each other* at any time $t \in [0, 1)$. Specifically, there exists no location $z \in \mathbb{R}^d$ and time $t \in [0, 1)$, such that two paths go across $z$ at time $t$ along different directions, because otherwise the solution of the ODE would be non-unique. On the other hand, the paths of the interpolation process $X_t$ may intersect with each other (Figure 2a), which makes it non-causal. Hence, as shown in Figure 2b, the rectified flow *rewires* the individual trajectories passing through the intersection points to avoid crossing, while tracing out the same density map as the linear interpolation paths due to the optimization of (1). We can view the linear interpolation $X_t$ as building roads (or tunnels) to connect $\pi_0$ and $\pi_1$, and the rectified flow as traffics of particles passing through the roads in a myopic, memoryless, non-crossing way, which allows them to ignore the global path information of how $X_0$ and $X_1$ are paired, and rebuild a more deterministic pairing of $(Z_0, Z_1)$.

**Algorithm 1** Rectified Flow: Main Algorithm

**Procedure**: $\boldsymbol{Z} = \texttt{RectFlow}((X_0, X_1))$:

*Inputs*: Draws from a coupling $(X_0, X_1)$ of $\pi_0$ and $\pi_1$; velocity model $v_\theta \colon \mathbb{R}^d \to \mathbb{R}^d$ with parameter $\theta$.

*Training*: $\hat{\theta} = \arg\min_{\theta} \mathbb{E}\left[\|X_1 - X_0 - v(tX_1 + (1-t)X_0,\ t)\|^2\right]$, with $t \sim \text{Uniform}([0,1])$.

*Sampling*: Draw $(Z_0, Z_1)$ following $\mathrm{d}Z_t = v_{\hat{\theta}}(Z_t, t)\mathrm{d}t$ starting from $Z_0 \sim \pi_0$ (or backwardly $Z_1 \sim \pi_1$).

*Return*: $\boldsymbol{Z} = \{Z_t \colon t \in [0,1]\}$.

**Reflow** (optional): $\boldsymbol{Z}^{k+1} = \texttt{RectFlow}((Z_0^k, Z_1^k))$, starting from $(Z_0^0, Z_1^0) = (X_0, X_1)$.

1) **Draw $(X_0, X_1)$**
2) **Train *v***
3) **Follow *v* to get new mapping $(Z_0, Z_1)$**

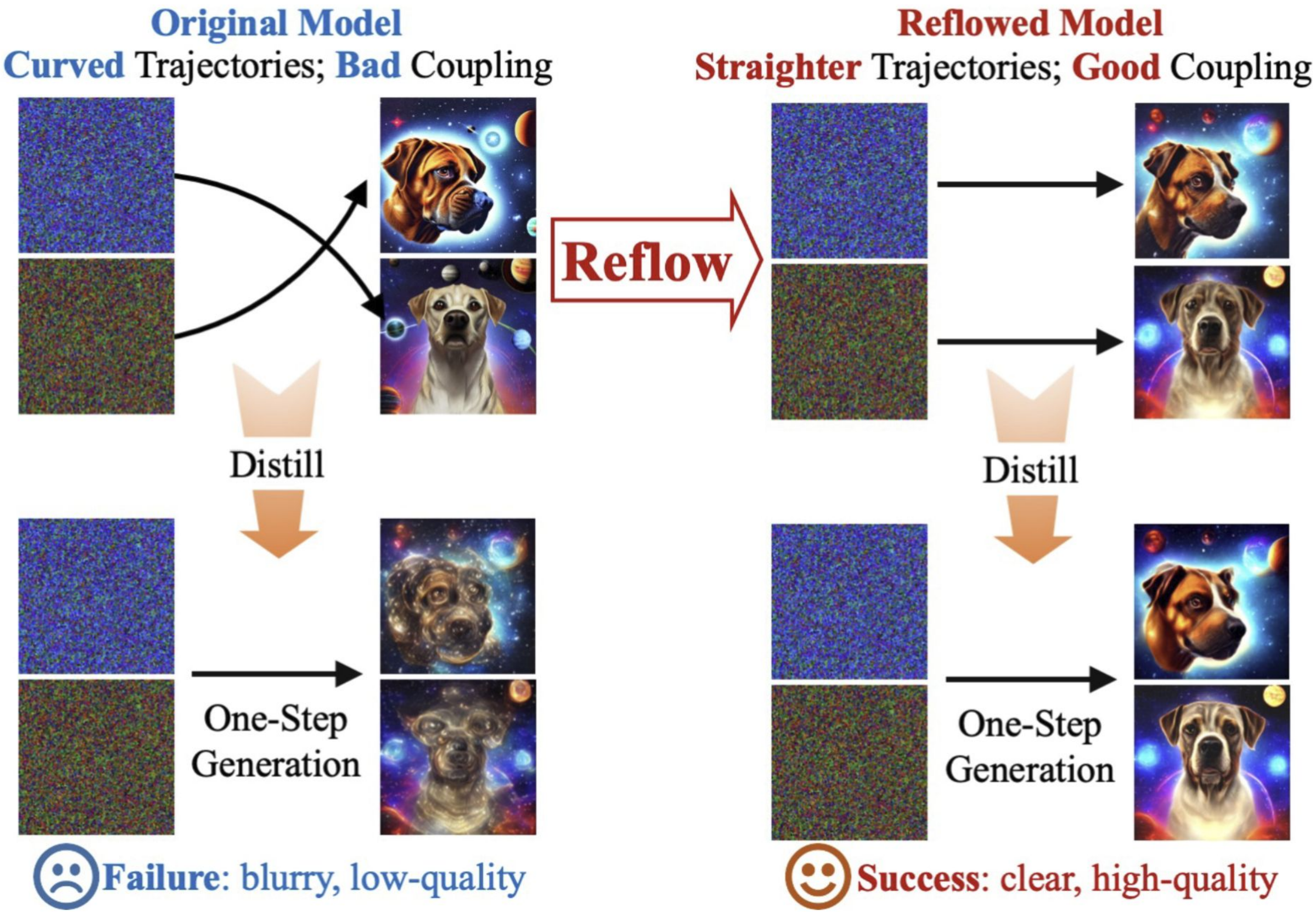
4) **Do it again.**
   - **Prev $(Z_0, Z_1)$ is now the input $(X_0, X_1)$**
   - **"k-flow"**

bad coupling

$$X_1 = \texttt{ODE}[v_k](X_0)$$

$$X_1^{\text{new}} = \texttt{ODE}[v_{k+1}](X_0)$$

better coupling


Original Model
Curved Trajectories; Bad Coupling
Reflowed Model
Straighter Trajectories; Good Coupling
Reflow
Distill
Distill
One-Step
Generation
One-Step
Generation
Failure: blurry, low-quality
Success: clear, high-quality

# Text-Conditioned

$$\min_{v} \mathbb{E}_{(X_0,X_1)\sim\gamma}\left[\int_0^1 \|\ (X_1 - X_0) - v(X_t,t)\ \|^2 \,\mathrm{d}t\right]$$

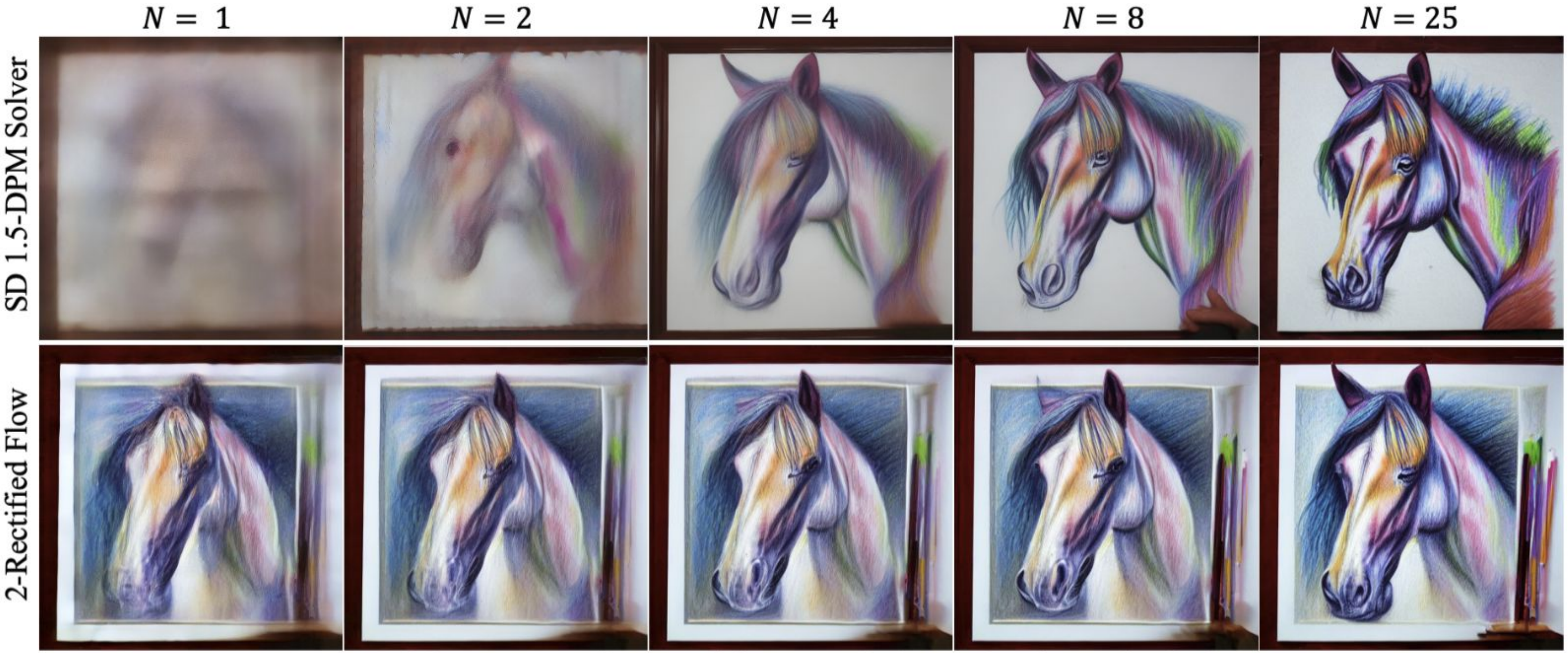
$$v_{k+1} = \arg\min_{v}\ \mathbb{E}_{X_0\sim\pi_0,\mathcal{T}\sim D_{\mathcal{T}}}\left[\int_0^1 \|\ (X_1 - X_0) - v(X_t,t \mid \mathcal{T})\ \|^2 \,\mathrm{d}t\right]$$

$D_{\mathcal{T}}$ is a dataset of text prompts

$$X_1 = \mathtt{ODE}[v_k](X_0 \mid \mathcal{T}) = X_0 + \int_0^1 v_k(X_t, t \mid \mathcal{T})$$

# Evaluation


Stable Diffusion
2-Rectified Flow
3-Rectified Flow
Stable Diffusion
2-Rectified Flow
3-Rectified Flow
N=25
N=1
+Distill




*'Masterpiece color pencil drawing of a horse; bright vivid color'*

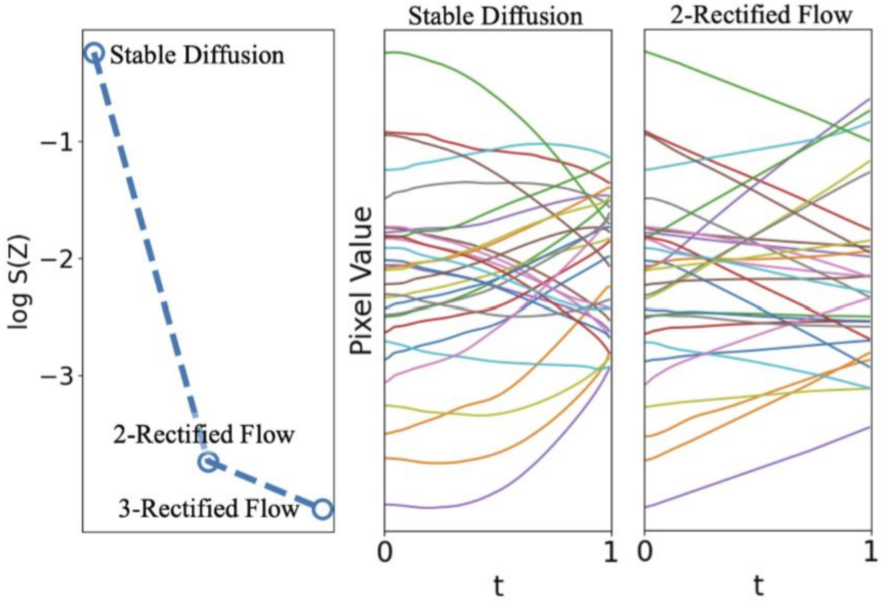
# Evaluation: Flow

Flow 'straightness'

$$S(Z) = \int_{t=0}^{1} \mathbb{E}\left[ ||\ (Z_1 - Z_0) - v(Z_t, t)\ ||^2 \right] \mathrm{d}t$$

How pixel values change over time

Straighter trajectories

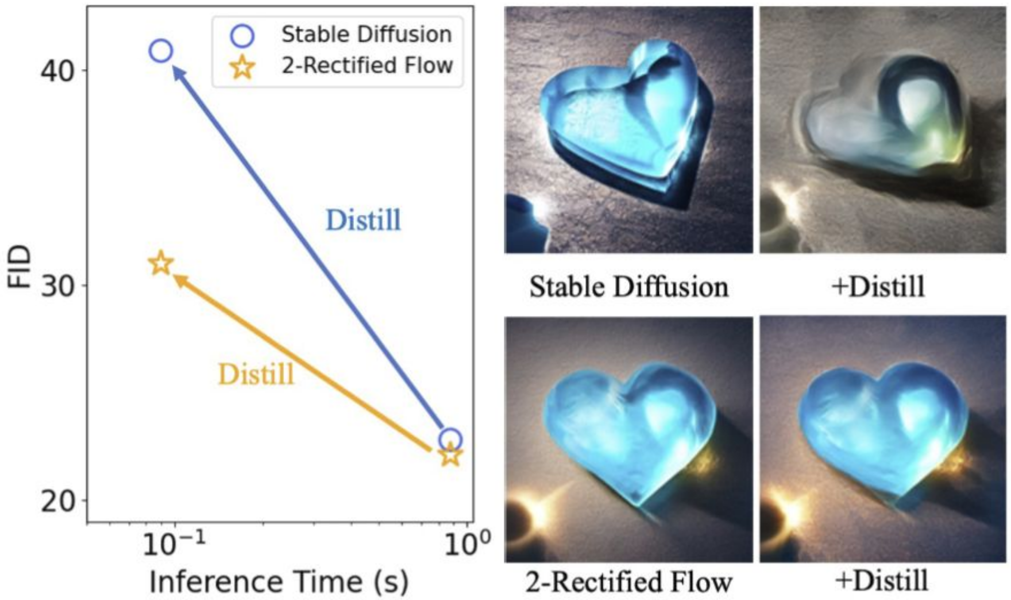


# Evaluation: Inference Time

SD cannot effectively distill

k-Rectified can

Distilled version of k-Flow has a smaller gap with teacher



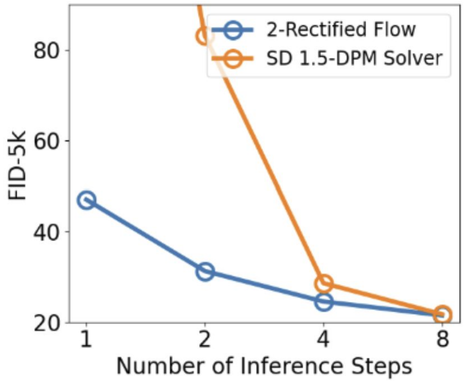
# Evaluation: FID + CLIP

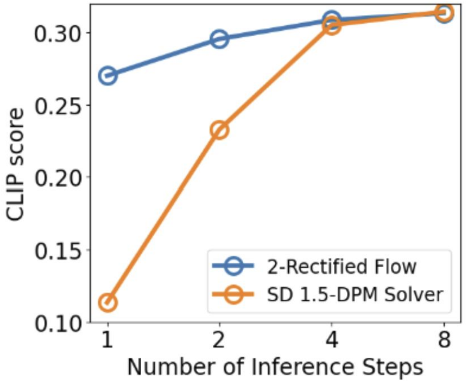
| Method | Inf. Time | FID-5k | CLIP |
|---|---|---|---|
| SD 1.4 (25 step)[70] | 0.88s | 22.8 | **0.315** |
| **(Pre) 2-RF (25 step)** | 0.88s | **22.1** | 0.313 |
| PD (1 step)[58] | 0.09s | 37.2 | 0.275 |
| SD 1.4+Distill | 0.09s | 40.9 | 0.255 |
| **(Pre) 2-RF (1 step)** | 0.09s | 68.3 | 0.252 |
| **(Pre) 2-RF+Distill** | 0.09s | **31.0** | **0.285** |

(a) MS COCO 2017

| Method | Inf. Time | FID-30k |
|---|---|---|
| SD* [70] | 2.9s | **9.62** |
| **(Pre) 2-RF (25 step)** | 0.88s | 13.4 |
| SD 1.4+Distill | 0.09s | 34.6 |
| **(Pre) 2-RF+Distill** | 0.09s | **20.0** |

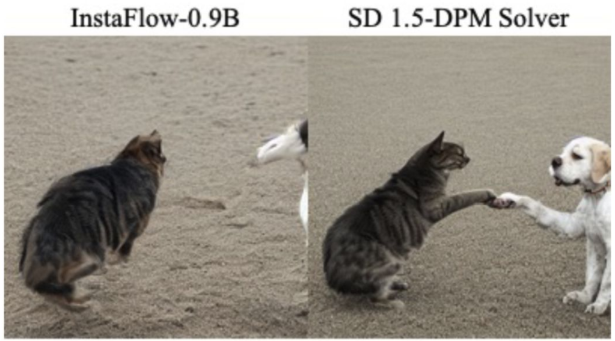
(b) MS COCO 2014



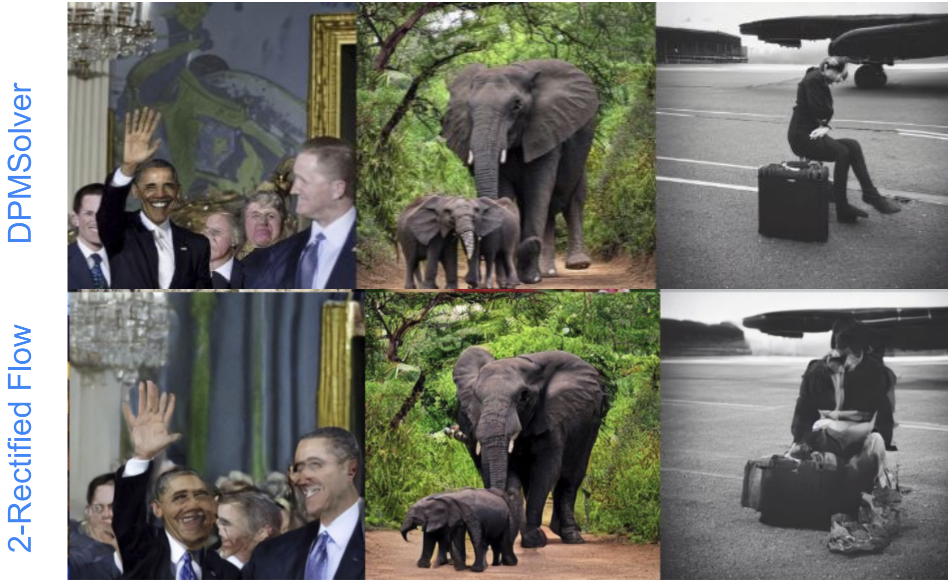



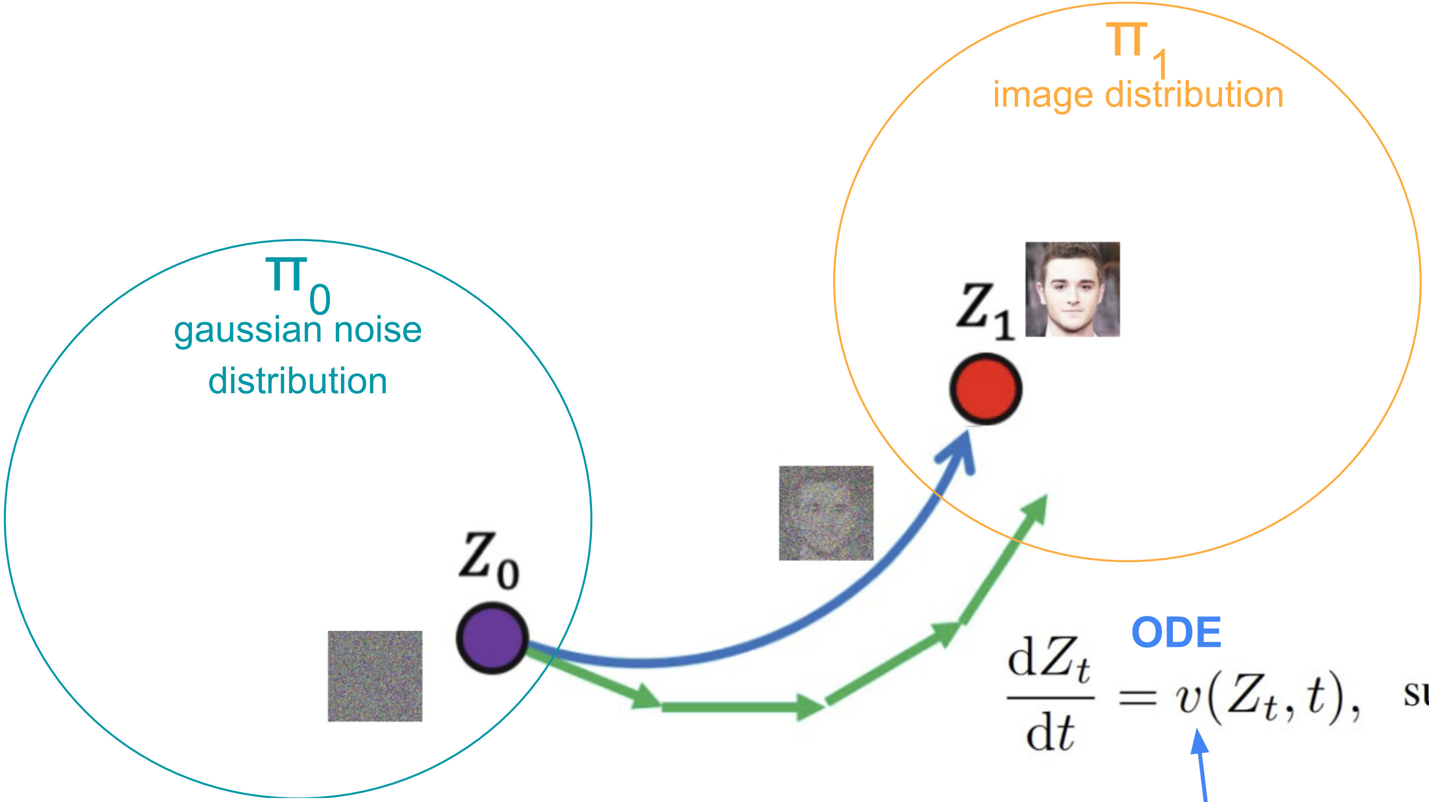
# Thoughts

- It’s all based on Rectified Flow
- New text-conditioning
- boasts strong improvement (FID + Results)
- failures on complicated prompts
- usefulness of FID ?
- better reflow?
- 1-step ?



*A dog and a cat shake hands with each other*






ODE

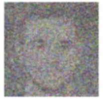
$$\frac{\mathrm{d}Z_t}{\mathrm{d}t} = v(Z_t, t), \quad \text{such that } Z_1 \sim \pi_1, \; Z_0 \sim \pi_0$$

**Approx w Neural Net**

$$\min_v \mathbb{E}_{(X_0, X_1)\sim\gamma}\left[\int_0^1 \| \frac{\mathrm{d}}{\mathrm{d}t} X_t - v(X_t, t) \|^2 \, \mathrm{d}t\right]$$

$$\phi(X_0, X_1, t),$$

Can choose different interpolations for $X_t$



$$X_t = \phi(X_0, X_1, t)$$

$$\boxed{X_t = tX_1 + (1-t)X_0}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} X_t = X_1 - X_0$$

can use any time-differentiable interpolation

DDIM uses this

Neural Net

$$\min_v \mathbb{E}_{(X_0, X_1)\sim\gamma}\left[\int_0^1 \|\, \frac{\mathrm{d}}{\mathrm{d}t} X_t - v(X_t, t)\, \|^2 \,\mathrm{d}t\right]$$

$$= \min_v \mathbb{E}_{(X_0, X_1)\sim\gamma}\left[\int_0^1 \|\, (X_1 - X_0) - v(X_t, t)\, \|^2 \,\mathrm{d}t\right]$$