

Training-free approaches for image inversion and editing using latent diffusion and flow models

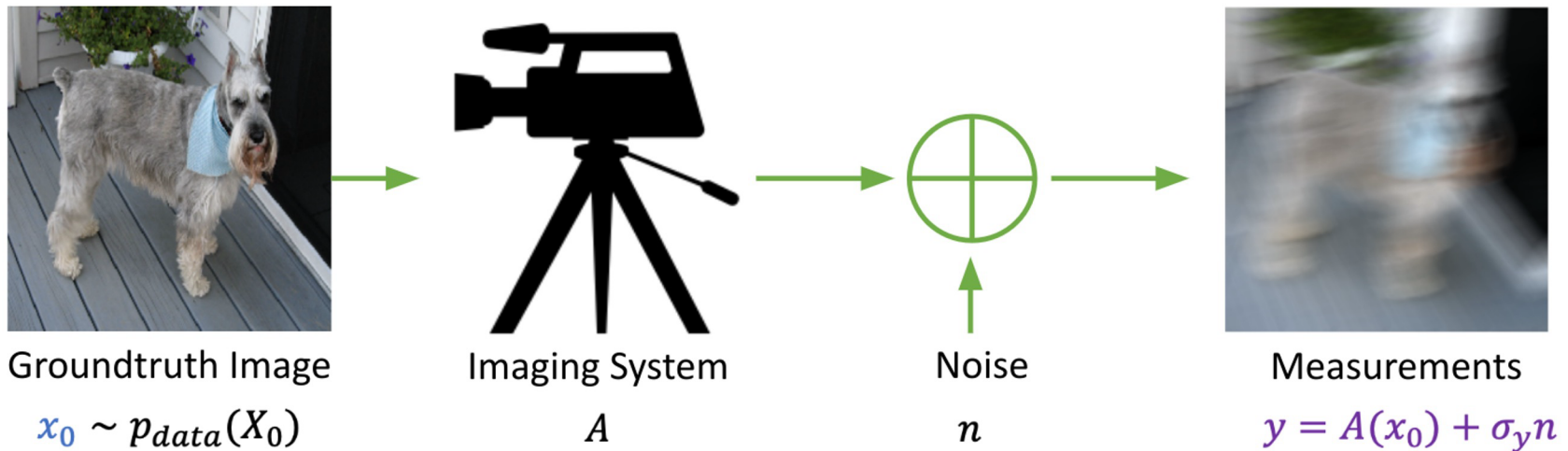
Sanjay Shakkottai

Based on joint work with: Litu Rout, Yujia Chen, Nataniel Ruiz, Abhishek Kumar*, Constantine Caramanis, and Wen-Sheng Chu

The University of Texas at Austin, Google Research, Google DeepMind



Inverse Problems Setting

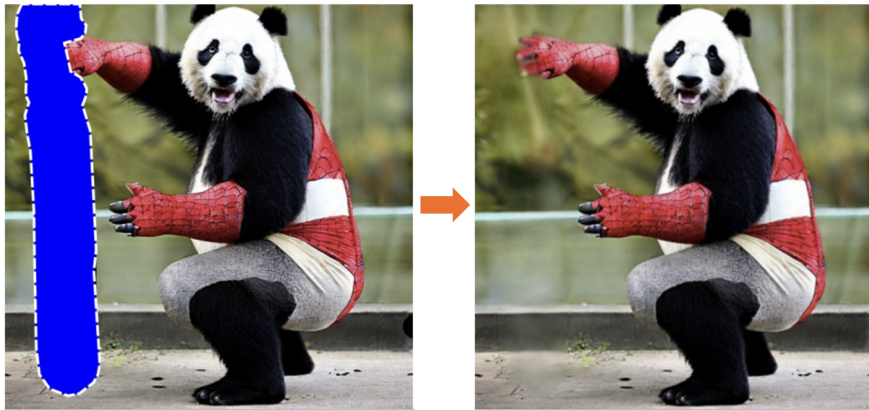


Problem: Reconstruct ground truth image x_0 from noisy measurements y

Challenge: Problem is **ill-posed**, that is infinitely many solutions x_0 exist

Approach: Use **prior** knowledge $p(x_0)$ of how the image should look like

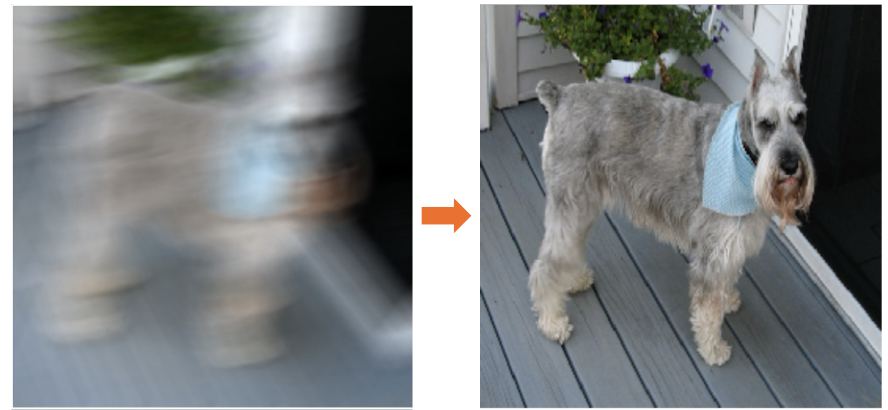
Examples of Inverse Problems



Free-form inpainting



Super-resolution (4X)



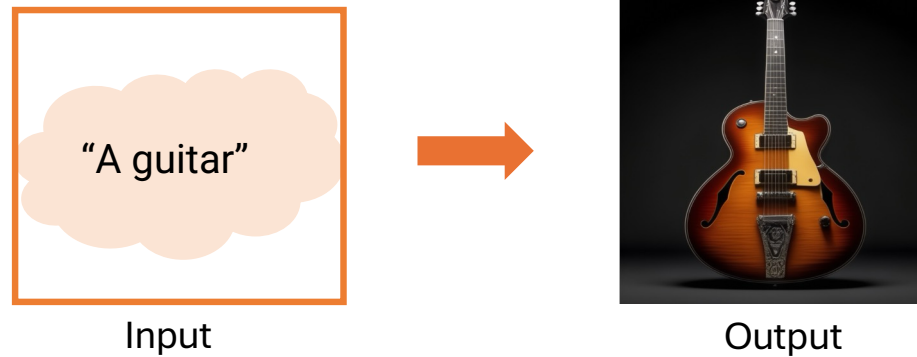
Motion Deblur



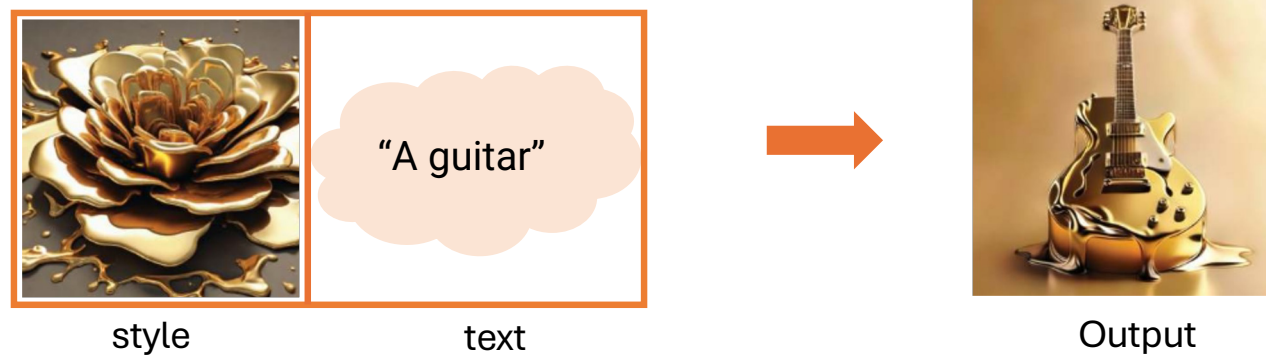
Gaussian Deblur

Stylization using Text and Image Prompts

Text-to-image generation



Personalized text-to-image generation: stylization



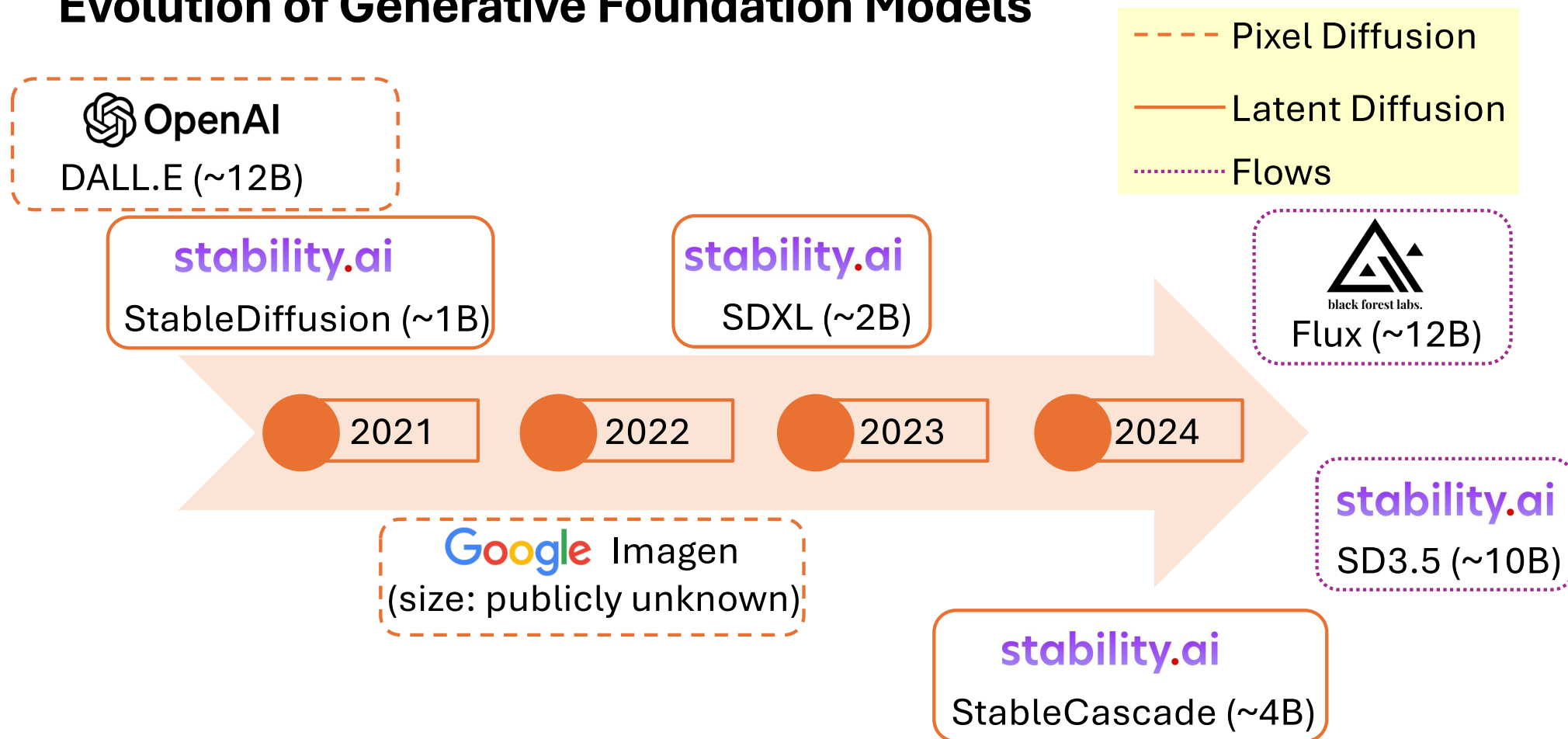
Content-Style Composition Using Text and Image Prompts

Personalized text-to-image generation: content-style composition

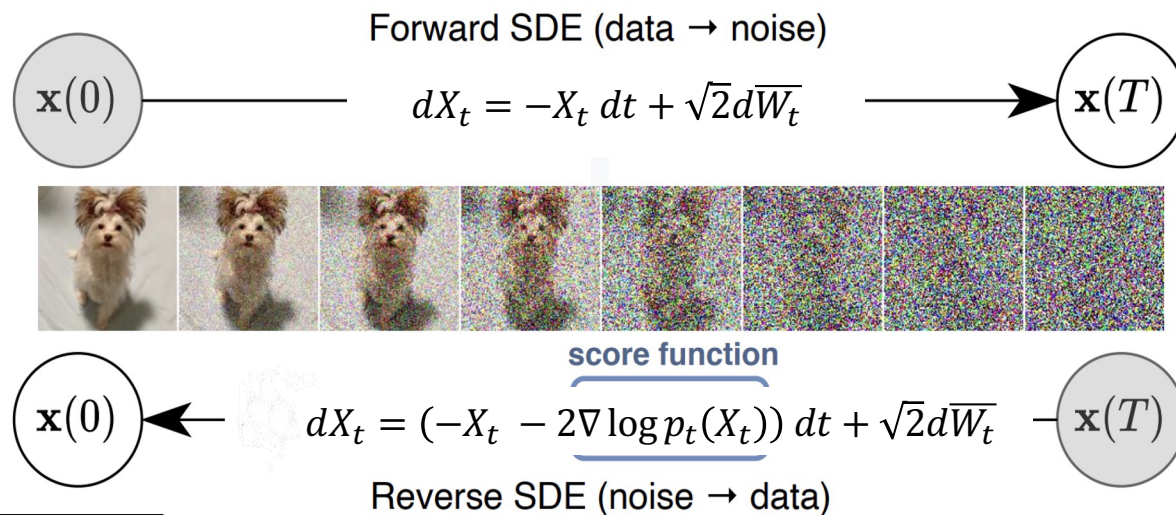


Diffusion models have recently emerged as powerful foundation models for solving such generalized inverse / composition problems

Evolution of Generative Foundation Models



Background on Diffusion Models



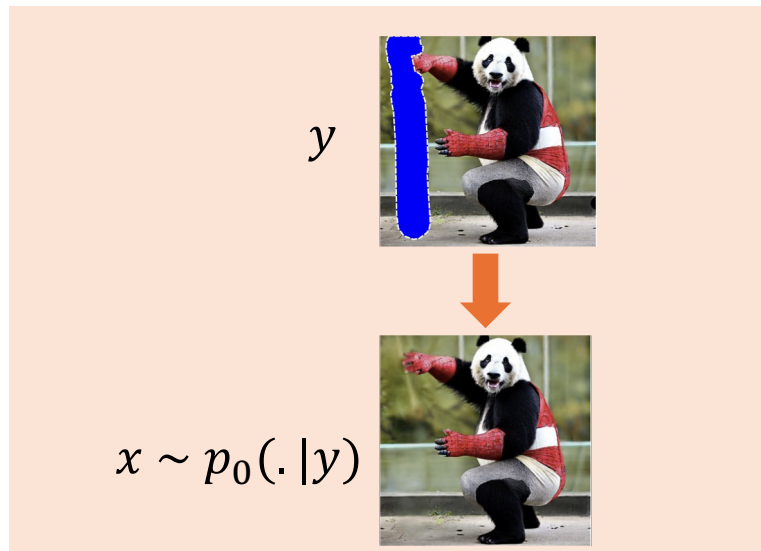
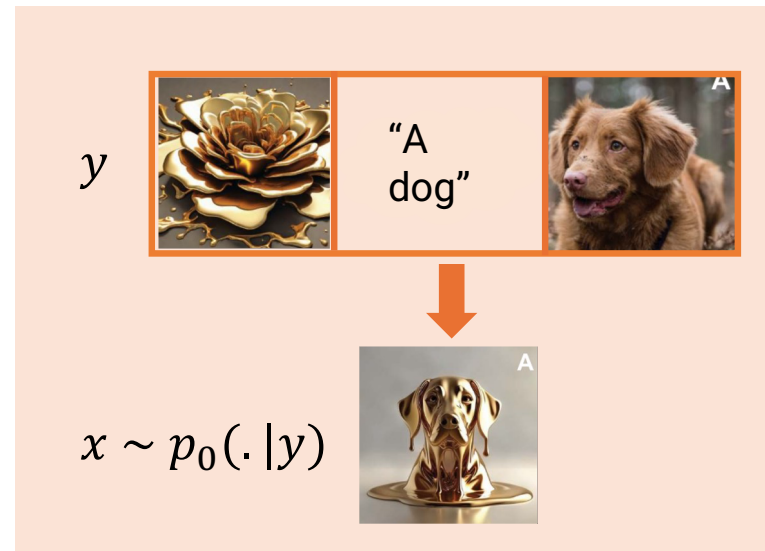
512 $\times$ 512 image $\approx$ 800K dim. vector

Image Credit: Song *et al.*, <https://arxiv.org/pdf/2011.13456.pdf>, ICLR'21

- **Goal:** Design a Markov process-based sampler (a transition kernel) such that stationary distribution samples images
- **Approach:** Learn annealed score that is affine in the conditional expectation of $X(0)$ (clean image) given $X(t)$ (noisy image) by **Tweedie's Formula**

Posterior Sampling with Diffusion

- Inverse problems such as infilling, super resolution, denoising, editing, stylization, etc. are all examples of **posterior sampling**
- **Goal:** Given “measurement / context” y , generate a sample x , where $x \sim p_0(\cdot | y)$



Posterior Sampling with Diffusion

Problem: Sample from $p_0(x_0|y)$ instead of $p(x_0)$

$$dX_t = (-X_t - 2 \nabla \log p_t(X_t|y)) dt + \sqrt{2} d\bar{W}_t, \quad t = T, \dots, 0$$

↓
Unknown

Bayes rule: $\log p_t(x_t|y) = \log p_t(y|x_t) + \log p_t(x_t) - \log p_t(y)$

$$dX_t = (-X_t - 2 \nabla \log p_t(y|X_t) - 2 \nabla \log p_t(X_t)) dt + \sqrt{2} d\bar{W}_t, \quad t = T, \dots, 0$$

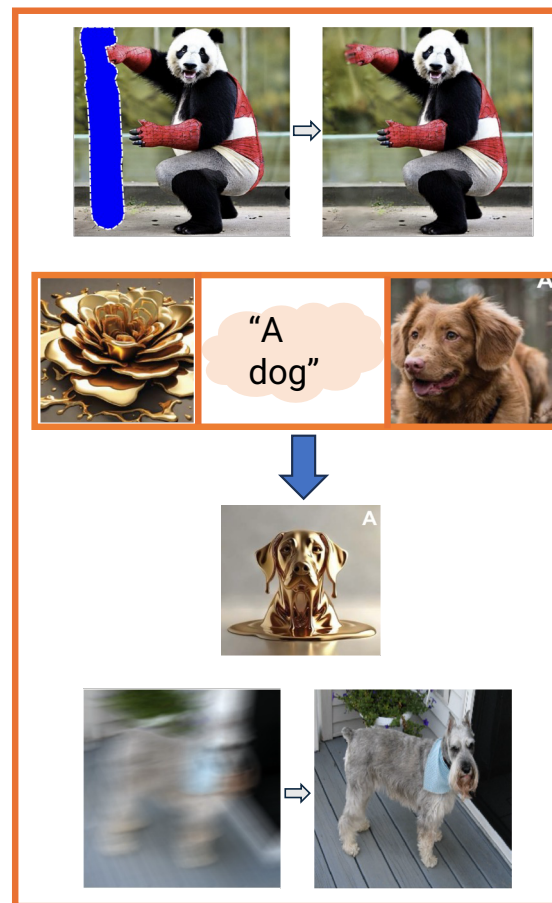
↓

Unknown

↓

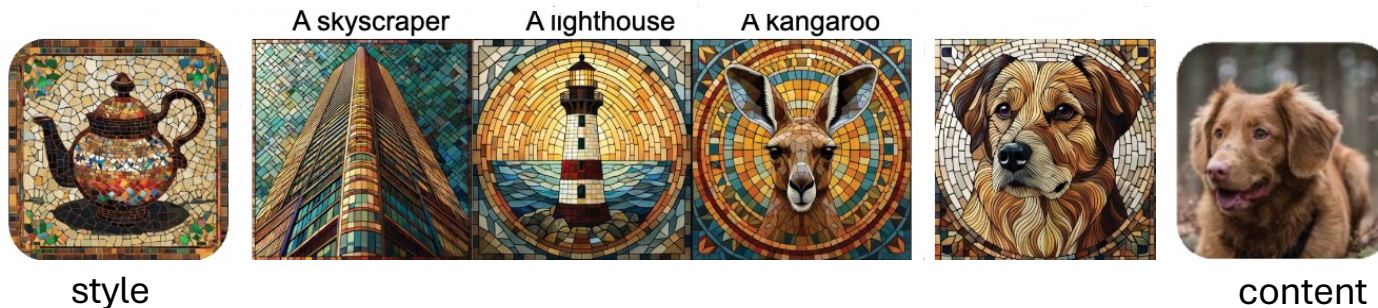
Known: $\nabla \log p_t(X_t) \approx s_\theta(X_t, t)$

How well can we approximate $\nabla \log p_t(y|x_t)$?



RB-Modulation for Content-Style Composition

- Algorithm for stylization and editing via [Test-time Optimization](#) using proximal methods
 - Training-free approach
 - **Avoids backpropagation through score network**
 - All previous training-free algorithms (e.g., DPS, PSLD, STSL) require backpropagating through the score network to address inverse problems



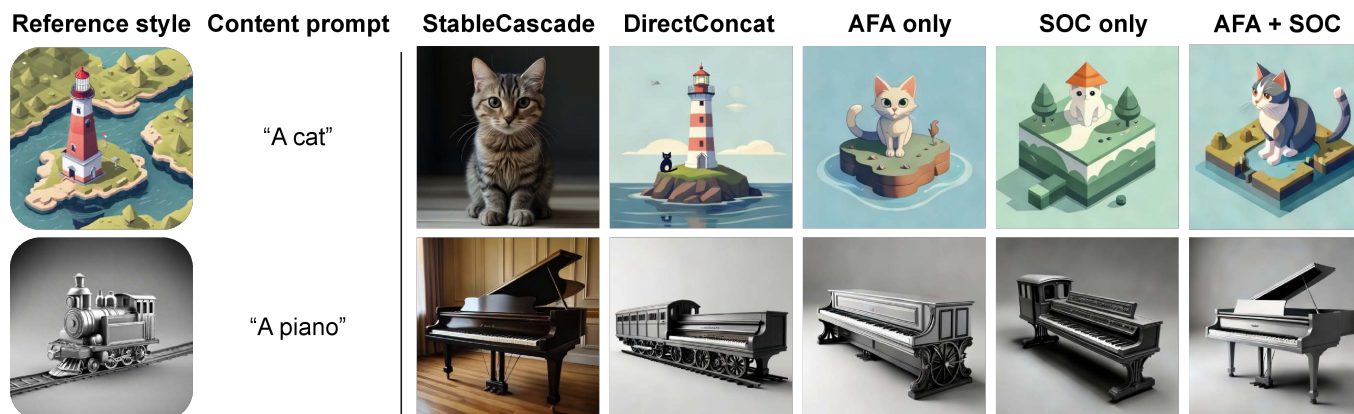
“RB-Modulation: Training-Free Stylization using Reference-Based Modulation”, Litu Rout, Yujia Chen, Nataniel Ruiz, Abhishek Kumar, Constantine Caramanis, Sanjay Shakkottai, Wen-Sheng Chu, *ICLR 2025 (Oral)*

Training vs Test-time Optimization

- Training-time optimization (DreamBooth, LoRA, IP-Adapter)
 - Approximately **10s of samples** per conditioning (style/content)
 - Single sample leads to **catastrophic** forgetting
 - Gradient computed with respect to **weights** of base model
 - LoRA finetuning takes **~20** min per style (40 min for content-style)
 - Full finetuning takes **hours**
- Test-time optimization (DPS, **PSLD**, P2L, **STSL**)
 - **Single** sample suffices (no catastrophic forgetting)
 - Gradient computed with respect to **input** to base model
 - Takes **~10** min for PSLD (1B), **~20** min for P2L(1B) (longer for Flux-12B)
- Proximal test-time optimization (**RB-Modulation**)
 - Takes **40 sec** using StableCascade (4B)

RB-Modulation: SOC and AFA

- RB Modulation has two key elements
 - **Stochastic Optimal Controller** (SOC) and **Attention Feature Aggregation** (AFA)
 - **SOC**: An optimal control formulation-based sampler, implemented as a **test-time proximal optimizer**
 - **SOC**: Incorporate desired attributes (e.g., style) in controller's **terminal cost**
 - **AFA**: Personalize the score and **disentangle content-style from the reference** images through an alternate cross-attention processor



Posterior Sampling using Diffusion and Optimal Control

Goal: Interpret posterior sampling as a control problem

Recall: Sample $p_X(\cdot | y)$ instead of $p_X(\cdot)$ using conditional reverse SDE

$$dX_t = (-X_t - 2\nabla \log p_t(X_t | y)) dt + \sqrt{2} d\bar{W}_t, \quad t = T, \dots, 0$$

Approach: (i) Using Bayes rule, $\log p_t(x_t | y) = \log p_t(y | x_t) + \log p_t(x_t) - \log p_t(y)$

$$dX_t = (-X_t - 2\nabla \log p_t(y | X_t) - 2\nabla \log p_t(X_t)) dt + \sqrt{2} d\bar{W}_t, \quad t = T, \dots, 0$$

(ii) Stochastic optimal control problem with terminal cost

$$\min_{u \in U} E \left[\int_0^1 [\|u(X_t^u, t)\|^2] dt + g(X_1^u, y) \right]$$

Terminal Cost

Style Features

$$dX_t^u = (-X_t + u(X_t^u, t) - 2\nabla \log p_t(X_t)) dt + \sigma(t) dW_t, \quad X_0^u \sim p_0.$$

Incorporating Style via Terminal Cost

Proposition A.4. Suppose $A \in \mathbb{R}^{k \times d}$ be a linear style extractor that operates on the terminal state $X_1^u \in \mathbb{R}^d$. Given reference style features y_1 , consider the control problem:

$$\min_{u \in \mathcal{U}} \int_{t_0}^1 \frac{1}{2} \|u(X_t^u, t)\|^2 dt + \frac{\gamma}{2} \|AX_1^u - y_1\|_2^2, \quad (9)$$

where $dX_t^u = u(X_t^u, t) dt$, $X_{t_0}^u = x_0$,   (10)

Then, in the limit when $\gamma \rightarrow \infty$, the optimal controller $u^* = \frac{(A^T A)^{-1} A^T (y_1 - AX_t^u)}{1-t}$, which yields the following controlled dynamic:

$$dX_t^u = \frac{(A^T A)^{-1} A^T (y_1 - AX_t^u)}{1-t} dt. \quad (11)$$

Implications:

- Given a linear style feature extractor, we derive a closed-form solution for the controller
- In practice, we use Contrastive Style Descriptor (CSD) for style feature extraction and solve the minimization problem (9) due to non-linearity of the descriptor

Incorporating Style via Additive Control

Proposition A.5. Suppose $A \in \mathbb{R}^{k \times d}$ be a linear style extractor that operates on the terminal state $X_1^u \in \mathbb{R}^d$. Let $\mathbf{p}_t$ denote $\nabla_{\mathbf{x}} V^*(\mathbf{x}, t)$ in HJB equation (A.1). Given reference style features y_1 , consider the control problem:

$$\min_{u \in \mathcal{U}} \int_{t_0}^1 \frac{1}{2} \|u(X_t^u, t)\|^2 dt + \frac{\gamma}{2} \|AX_1^u - y_1\|_2^2, \quad (21)$$

$$\text{where } dX_t^u = [X_t^u + u(X_t^u, t)] dt, \quad X_{t_0}^u = x_0, \quad (22)$$

Additive
Control

Then, the optimal controller becomes $u^*(t) = -\mathbf{p}_t$, where the instantaneous state $X_t^u = \mathbf{x}_t$ and $\mathbf{p}_t$ satisfy the following:

$$\begin{bmatrix} \mathbf{x}_t \\ \mathbf{p}_t \end{bmatrix} = \begin{bmatrix} x_0 e^t - \frac{\gamma}{2} A^T (A\mathbf{x}_1 - y_1) e^{1+t} + \frac{\gamma}{2} A^T (A\mathbf{x}_1 - y_1) e^{1-t} \\ \gamma A^T (A\mathbf{x}_1 - y_1) e^{1-t} \end{bmatrix}.$$

Implications:

- **General strategy:** Derive the optimal controller with known terminal state, and then replace the terminal state in the controller with its estimate using Tweedie's formula
- While the controllers so far have an explicit form, in practice, the data distribution is not Gaussian, and thus, we do not have a closed-form expression

Practical Implementation of SOC Module

Algorithm 1: Reference-Based Modulation

Input: Diffusion steps T , reference prompt $\mathbf{p}$, reference image $\mathbf{z}_0$, style descriptor $\Psi(\cdot)$, score network $s(\cdot, \cdot, \cdot; \theta)$

Tunable parameter: Stepsize η , optimization steps M

Output: Personalized latent X_0^u

```
1 Initialize  $\mathbf{x}_T \leftarrow \mathcal{N}(0, \mathbf{I}_d)$ 
2 for  $t = T$  to 1 do
3   Initialize controller  $u = 0$ 
4   for  $m = 1$  to  $M$  do
5      $\hat{\mathbf{x}}_t = \mathbf{x}_t + u$   $\triangleright$  controlled state
6      $\bar{X}_0^u = \frac{\hat{\mathbf{x}}_t}{\sqrt{\bar{\alpha}_t}} + \frac{(1-\bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}} s(\hat{\mathbf{x}}_t, t, \mathbf{p}; \theta)$ 
7      $h(\bar{X}_0^u) = \|\Psi(\mathbf{z}_0) - \Psi(\bar{X}_0^u)\|_2^2$  using Eq. (3)
8      $u = u - \eta \nabla_u h(\bar{X}_0^u)$   $\triangleright$  update controller
9   end
10   $\mathbf{x}_t^* = \mathbf{x}_t + u$   $\triangleright$  optimally controlled state
11   $\bar{X}_0^u = \frac{\mathbf{x}_t^*}{\sqrt{\bar{\alpha}_t}} + \frac{(1-\bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}} s(\mathbf{x}_t^*, t, \mathbf{p}; \theta)$   $\triangleright$  terminal state
12   $\mathbf{x}_{t-1} \leftarrow \text{DDIM}(\bar{X}_0^u, \mathbf{x}_t^*)$   $\triangleright$  one step reverse-SDE [20]
13 end
14 return  $X_0^u$ 
```

- Add initial controller to current state

Practical Implementation of SOC Module

Algorithm 1: Reference-Based Modulation

Input: Diffusion steps T , reference prompt $\mathbf{p}$, reference image

$\mathbf{z}_0$, style descriptor $\Psi(\cdot)$, score network $s(\cdot, \cdot, \cdot; \theta)$

Tunable parameter: Stepsize η , optimization steps M

Output: Personalized latent X_0^u

```
1 Initialize  $\mathbf{x}_T \leftarrow \mathcal{N}(0, \mathbf{I}_d)$ 
2 for  $t = T$  to 1 do
3   Initialize controller  $u = 0$ 
4   for  $m = 1$  to  $M$  do
5      $\hat{\mathbf{x}}_t = \mathbf{x}_t + u$   $\triangleright$  controlled state
6      $\bar{X}_0^u = \frac{\hat{\mathbf{x}}_t}{\sqrt{\bar{\alpha}_t}} + \frac{(1-\bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}} s(\hat{\mathbf{x}}_t, t, \mathbf{p}; \theta)$ 
7      $h(X_0^u) = \|\Psi(\mathbf{z}_0) - \Psi(X_0^u)\|_2^2$  using Eq. (3)
8      $u = u - \eta \nabla_u h(\bar{X}_0^u)$   $\triangleright$  update controller
9   end
10   $\mathbf{x}_t^* = \mathbf{x}_t + u$   $\triangleright$  optimally controlled state
11   $\bar{X}_0^u = \frac{\mathbf{x}_t^*}{\sqrt{\bar{\alpha}_t}} + \frac{(1-\bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}} s(\mathbf{x}_t^*, t, \mathbf{p}; \theta)$   $\triangleright$  terminal state
12   $\mathbf{x}_{t-1} \leftarrow \text{DDIM}(\bar{X}_0^u, \mathbf{x}_t^*)$   $\triangleright$  one step reverse-SDE [20]
13 end
14 return  $X_0^u$ 
```

- Add initial controller to current state
- Compute terminal state using Tweedie's formula and pre-trained score network

Practical Implementation of SOC Module

Algorithm 1: Reference-Based Modulation

Input: Diffusion steps T , reference prompt $\mathbf{p}$, reference image $\mathbf{z}_0$, style descriptor $\Psi(\cdot)$, score network $s(\cdot, \cdot, \cdot; \theta)$

Tunable parameter: Stepsize η , optimization steps M

Output: Personalized latent X_0^u

```
1 Initialize  $\mathbf{x}_T \leftarrow \mathcal{N}(0, \mathbf{I}_d)$ 
2 for  $t = T$  to 1 do
3   Initialize controller  $u = 0$ 
4   for  $m = 1$  to  $M$  do
5      $\hat{\mathbf{x}}_t = \mathbf{x}_t + u$   $\triangleright$  controlled state
6      $\bar{X}_0^u = \frac{\hat{\mathbf{x}}_t}{\sqrt{\bar{\alpha}_t}} + \frac{(1-\bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}} s(\hat{\mathbf{x}}_t, t, \mathbf{p}; \theta)$ 
7      $h(\bar{X}_0^u) = \|\Psi(\mathbf{z}_0) - \Psi(\bar{X}_0^u)\|_2^2$  using Eq. (3)
8      $u = u - \eta \nabla_u h(\bar{X}_0^u)$   $\triangleright$  update controller
9   end
10   $\mathbf{x}_t^* = \mathbf{x}_t + u$   $\triangleright$  optimally controlled state
11   $\bar{X}_0^u = \frac{\mathbf{x}_t^*}{\sqrt{\bar{\alpha}_t}} + \frac{(1-\bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}} s(\mathbf{x}_t^*, t, \mathbf{p}; \theta)$   $\triangleright$  terminal state
12   $\mathbf{x}_{t-1} \leftarrow \text{DDIM}(\bar{X}_0^u, \mathbf{x}_t^*)$   $\triangleright$  one step reverse-SDE [20]
13 end
14 return  $X_0^u$ 
```

- Add initial controller to current state
- Compute terminal state using Tweedie's formula and pre-trained score network
- Compute terminal constraint using style feature extractor network

Practical Implementation of SOC Module

Algorithm 1: Reference-Based Modulation

Input: Diffusion steps T , reference prompt $\mathbf{p}$, reference image

$\mathbf{z}_0$, style descriptor $\Psi(\cdot)$, score network $s(\cdot, \cdot, \cdot; \theta)$

Tunable parameter: Stepsize η , optimization steps M

Output: Personalized latent X_0^u

```
1 Initialize  $\mathbf{x}_T \leftarrow \mathcal{N}(0, \mathbf{I}_d)$ 
2 for  $t = T$  to 1 do
3   Initialize controller  $u = 0$ 
4   for  $m = 1$  to  $M$  do
5      $\hat{\mathbf{x}}_t = \mathbf{x}_t + u$   $\triangleright$  controlled state
6      $\bar{X}_0^u = \frac{\hat{\mathbf{x}}_t}{\sqrt{\bar{\alpha}_t}} + \frac{(1-\bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}} s(\hat{\mathbf{x}}_t, t, \mathbf{p}; \theta)$ 
7      $h(\bar{X}_0^u) = \|\Psi(\mathbf{z}_0) - \Psi(\bar{X}_0^u)\|_2^2$  using Eq. (3)
8      $u = u - \eta \nabla_u h(\bar{X}_0^u)$   $\triangleright$  update controller
9   end
10   $\mathbf{x}_t^* = \mathbf{x}_t + u$   $\triangleright$  optimally controlled state
11   $\bar{X}_0^u = \frac{\mathbf{x}_t^*}{\sqrt{\bar{\alpha}_t}} + \frac{(1-\bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}} s(\mathbf{x}_t^*, t, \mathbf{p}; \theta)$   $\triangleright$  terminal state
12   $\mathbf{x}_{t-1} \leftarrow \text{DDIM}(\bar{X}_0^u, \mathbf{x}_t^*)$   $\triangleright$  one step reverse-SDE [20]
13 end
14 return  $X_0^u$ 
```

- Add initial controller to current state
- Compute terminal state using Tweedie's formula and pre-trained score network
- Compute terminal constraint using style feature extractor network
- Update controller using one step of gradient descent on terminal cost

Practical Implementation of SOC Module

Algorithm 1: Reference-Based Modulation

Input: Diffusion steps T , reference prompt $\mathbf{p}$, reference image

$\mathbf{z}_0$, style descriptor $\Psi(\cdot)$, score network $s(\cdot, \cdot, \cdot; \theta)$

Tunable parameter: Stepsize η , optimization steps M

Output: Personalized latent X_0^u

```

1 Initialize  $\mathbf{x}_T \leftarrow \mathcal{N}(0, \mathbf{I}_d)$ 
2 for  $t = T$  to 1 do
3   Initialize controller  $u = 0$ 
4   for  $m = 1$  to  $M$  do
5      $\hat{\mathbf{x}}_t = \mathbf{x}_t + u$   $\triangleright$  controlled state
6      $\bar{X}_0^u = \frac{\hat{\mathbf{x}}_t}{\sqrt{\bar{\alpha}_t}} + \frac{(1-\bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}} s(\hat{\mathbf{x}}_t, t, \mathbf{p}; \theta)$ 
7      $h(\bar{X}_0^u) = \|\Psi(\mathbf{z}_0) - \Psi(\bar{X}_0^u)\|_2^2$  using Eq. (3)
8      $u = u - \eta \nabla_u h(\bar{X}_0^u)$   $\triangleright$  update controller
9   end
10   $\mathbf{x}_t^* = \mathbf{x}_t + u$   $\triangleright$  optimally controlled state
11   $\bar{X}_0^u = \frac{\mathbf{x}_t^*}{\sqrt{\bar{\alpha}_t}} + \frac{(1-\bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}} s(\mathbf{x}_t^*, t, \mathbf{p}; \theta)$   $\triangleright$  terminal state
12   $\mathbf{x}_{t-1} \leftarrow \text{DDIM}(\bar{X}_0^u, \mathbf{x}_t^*)$   $\triangleright$  one step reverse-SDE [20]
13 end
14 return  $X_0^u$ 

```

- Add initial controller to current state
- Compute terminal state using Tweedie's formula and pre-trained score network
- Compute terminal constraint using style feature extractor network
- Update controller using one step of gradient descent on terminal cost
- Repeat M times to get an updated controlled state

Practical Implementation of SOC Module

Algorithm 1: Reference-Based Modulation

Input: Diffusion steps T , reference prompt $\mathbf{p}$, reference image

$\mathbf{z}_0$, style descriptor $\Psi(\cdot)$, score network $s(\cdot, \cdot, \cdot; \theta)$

Tunable parameter: Stepsize η , optimization steps M

Output: Personalized latent X_0^u

```

1 Initialize  $\mathbf{x}_T \leftarrow \mathcal{N}(0, \mathbf{I}_d)$ 
2 for  $t = T$  to 1 do
3   Initialize controller  $u = 0$ 
4   for  $m = 1$  to  $M$  do
5      $\hat{\mathbf{x}}_t = \mathbf{x}_t + u$   $\triangleright$  controlled state
6      $\bar{X}_0^u = \frac{\hat{\mathbf{x}}_t}{\sqrt{\bar{\alpha}_t}} + \frac{(1-\bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}} s(\hat{\mathbf{x}}_t, t, \mathbf{p}; \theta)$ 
7      $h(\bar{X}_0^u) = \|\Psi(\mathbf{z}_0) - \Psi(\bar{X}_0^u)\|_2^2$  using Eq. (3)
8      $u = u - \eta \nabla_u h(\bar{X}_0^u)$   $\triangleright$  update controller
9   end
10   $\mathbf{x}_t^* = \mathbf{x}_t + u$   $\triangleright$  optimally controlled state
11   $\bar{X}_0^u = \frac{\mathbf{x}_t^*}{\sqrt{\bar{\alpha}_t}} + \frac{(1-\bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}} s(\mathbf{x}_t^*, t, \mathbf{p}; \theta)$   $\triangleright$  terminal state
12   $\mathbf{x}_{t-1} \leftarrow \text{DDIM}(\bar{X}_0^u, \mathbf{x}_t^*)$   $\triangleright$  one step reverse-SDE [20]
13 end
14 return  $X_0^u$ 

```

- Add initial controller to current state
- Compute terminal state using Tweedie's formula and pre-trained score network
- Compute terminal constraint using style feature extractor network
- Update controller using one step of gradient descent on terminal cost
- Repeat M times to get an updated controlled state
- Take one reverse diffusion step from the updated controlled state

Scalable Implementation of SOC Module (Proximal Algorithm)

Algorithm 2: RB-Modulation (Proximal)

Input: Diffusion time steps T , reference prompt $\mathbf{p}$, reference style image $\mathbf{z}_0$, style descriptor $\Psi(\cdot)$, score network $s(\cdot, \cdot, \cdot; \theta)$

Tunable parameters: Stepsize η , optimization steps M , proximal strength λ

Output: Personalized latent X_0^u

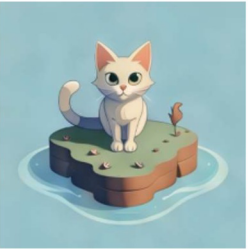
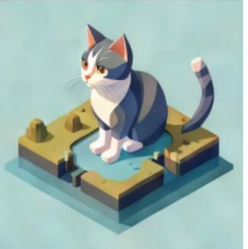
```
1 Initialize  $\mathbf{x}_T \leftarrow \mathcal{N}(0, \mathbf{I}_d)$ 
2 for  $t = T$  to 1 do
3   Compute posterior mean
       $\mathbb{E}[X_0^u | X_t^u = \mathbf{x}_t] = \frac{\mathbf{x}_t}{\sqrt{\bar{\alpha}_t}} + \frac{(1 - \bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}} s(\mathbf{x}_t, t, \mathbf{p}; \theta)$ 
4   Initialize opt. variable  $\mathbf{x}_0 = \mathbb{E}[X_0^u | X_t^u = \mathbf{x}_t]$ 
5   for  $m = 1$  to  $M$  do
6     Compute controller's cost  $\mathcal{L}(\mathbf{x}_0) := \|\Psi(\mathbf{z}_0) - \Psi(\mathbf{x}_0)\|_2^2 + \lambda \|\mathbf{x}_0 - \mathbb{E}[X_0^u | X_t^u = \mathbf{x}_t]\|_2^2$ 
7     Update optimization variable
         $\mathbf{x}_0 = \mathbf{x}_0 - \eta \nabla_{\mathbf{x}_0} \mathcal{L}(\mathbf{x}_0)$ 
8   end
9    $\mathbf{x}_{t-1} \leftarrow \text{DDIM}(\mathbf{x}_0, \mathbf{x}_t)$   $\triangleright$  one denoising step
10 end
11 return  $X_0^u$ 
```

- Proximal optimization that combines style consistency (aka measurement consistency) along with consistency with the image prior
- **Key point:** Loss gradient does not require backpropagation through the score network

Training vs Test-time Optimization

- Training-time optimization (DreamBooth, LoRA, IP-Adapter)
 - Approximately **10s of samples** per conditioning (style/content)
 - Single sample leads to **catastrophic** forgetting
 - Gradient computed with respect to **weights** of base model
 - LoRA finetuning takes **~20** min per style (40 min for content-style)
 - Full finetuning takes **hours**
- Test-time optimization (DPS, **PSLD**, P2L, **STSL**)
 - **Single** sample suffices (no catastrophic forgetting)
 - Gradient computed with respect to **input** to base model
 - Takes **~10** min for PSLD (1B), **~20** min for P2L(1B) (longer for Flux-12B)
- Proximal test-time optimization (**RB-Modulation**)
 - Takes **40 sec** using StableCascade (4B)

Stylization: Ablation Study

Reference style	Content prompt	StableCascade	DirectConcat	AFA only	SOC only	AFA + SOC
	"A cat"					
	"A piano"					

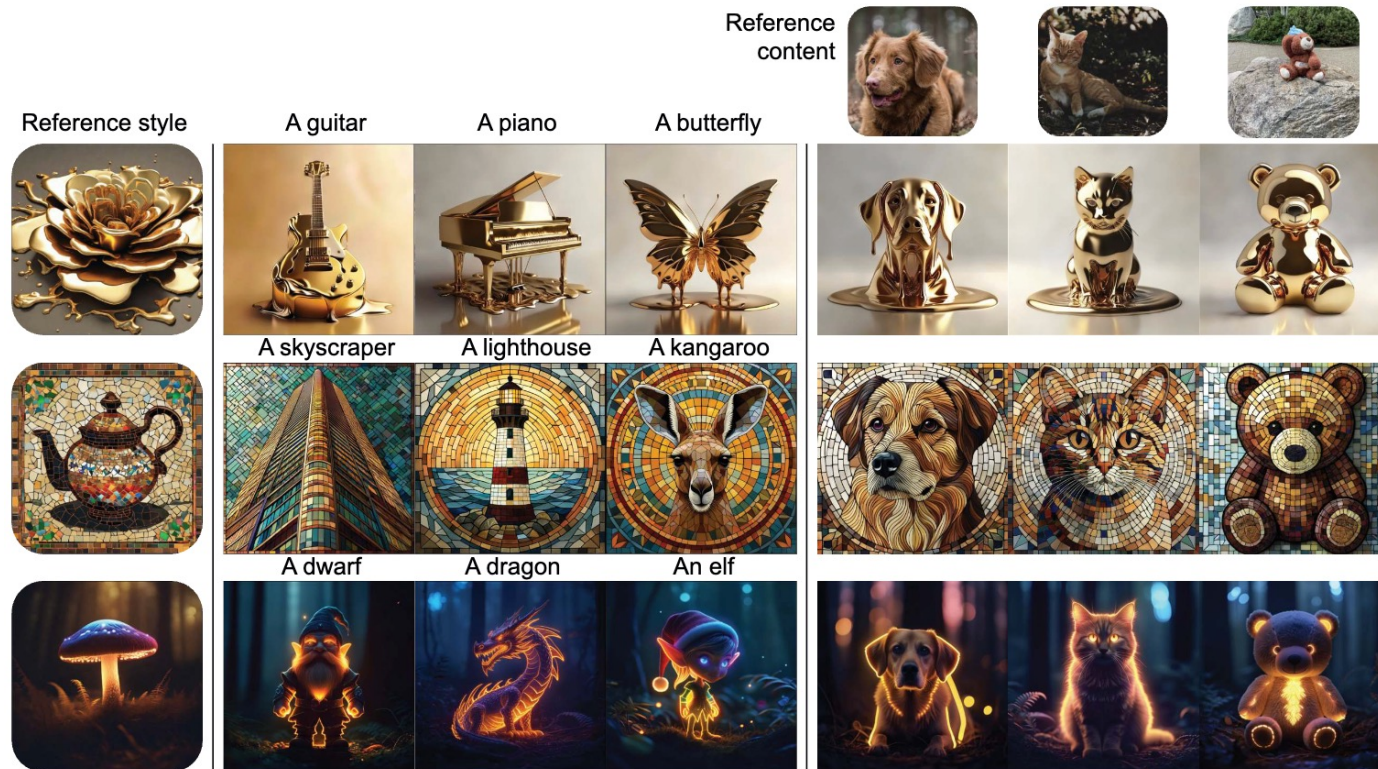
- DirectConcat mixes both embeddings, making it less effective in disentangling style from prompts (e.g., cat vs. lighthouse)
- AFA alone fails to capture the cat's style accurately, and SOC alone misplaces elements, like "a lighthouse hat on the cat" and "a railroad trunk on a piano"
- We observe consistent improvements with each module, with the best results when combined

Stylization: Human Evaluation

Human Preference (%)	Ours vs. InstantStyle [13]			Ours vs. StyleAligned [12]			Ours vs. IP-Adapter [21]		
	OQ ↑	SA ↑	PA ↑	OQ ↑	SA ↑	PA ↑	OQ ↑	SA ↑	PA ↑
Alternative	39.8	38.5	39.5	24.4	27.8	29.4	8.1	20.1	8.3
Tie	9.3	6.4	7.3	8.8	7.1	5.8	6.9	4.8	4.5
RB-Modulation (ours)	51.0	55.1	53.3	66.9	65.1	64.9	85.0	75.1	87.2

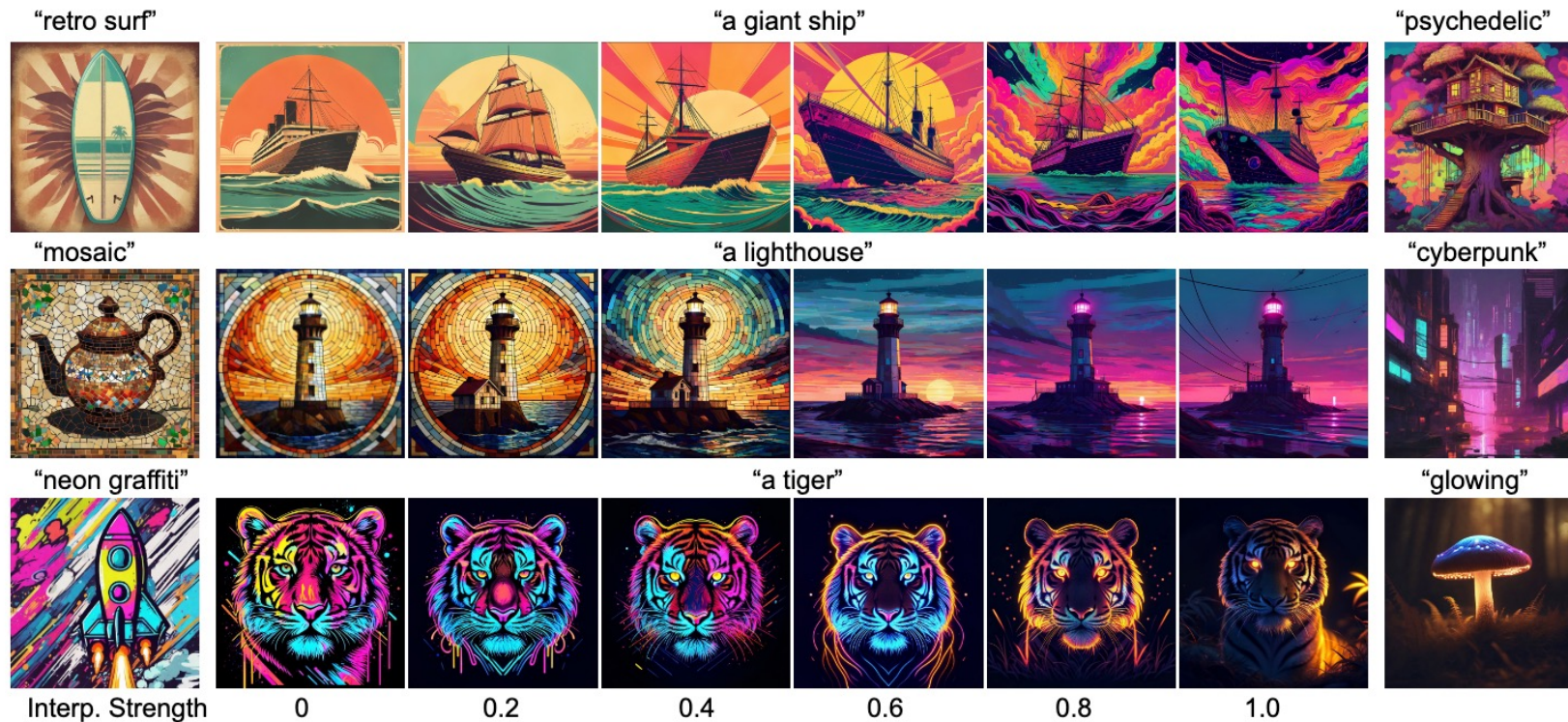
- Experiments covering stylization and content-style composition
 - User study (over 7000 responses) to capture human preference
- We report the % of human preference on ours vs. alternatives for overall quality (OQ), style alignment (SA), and prompt alignment (PA), including ties where users couldn't decide

Personalization using RB-Modulation



RB-Modulation as a plug-and-play solution for stylization and content-style composition with diverse layouts

Novel Style Synthesis: Interpolating Reference Styles



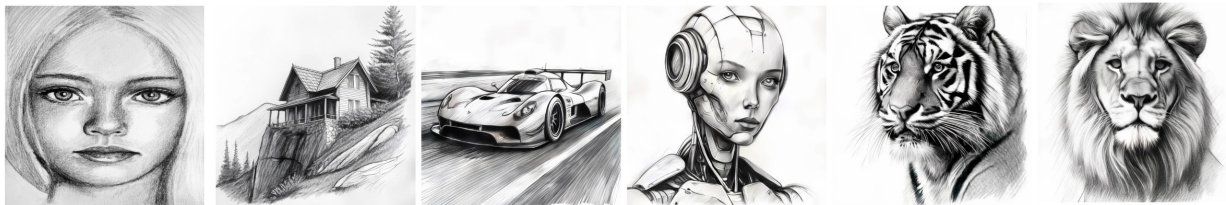
Training based methods cannot interpolate novel styles
due to lack of prior examples

Stylization: Hand Drawn Reference Images

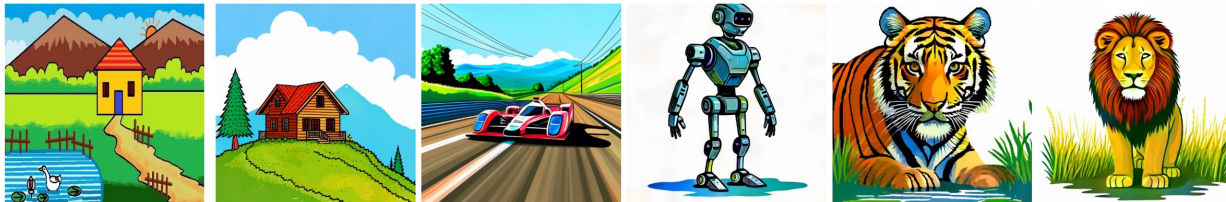
“plastic crayon”



“pencil sketch”



“comm. paint”



Reference Style “house on a mountain”

“racing car”

“futuristic robot”

“tiger”

“lion”

Text-guided creation of novel images based on real single-sample hand-drawn images

RB-Modulation: Production Status

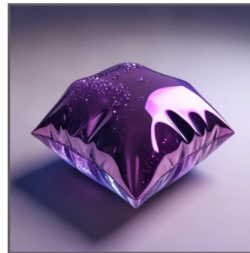
- Collaboration with Google and Google DeepMind researchers
 - Code available on github: <https://rb-modulation.github.io>
- Teams at Google are currently productionizing RB Modulation into several devices and production pipelines
 - Pixel, Chromebook, Tablet, and YouTube
 - Several application settings (e.g., on device personalization)
- Demo became #1 on HuggingFace in the week of its release



Reference Style



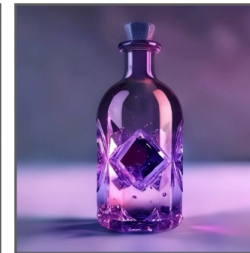
“mountain”



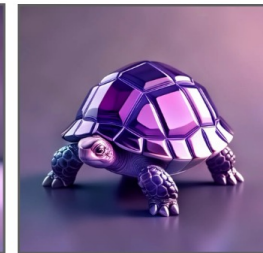
“pillow”



“building”



“bottle”

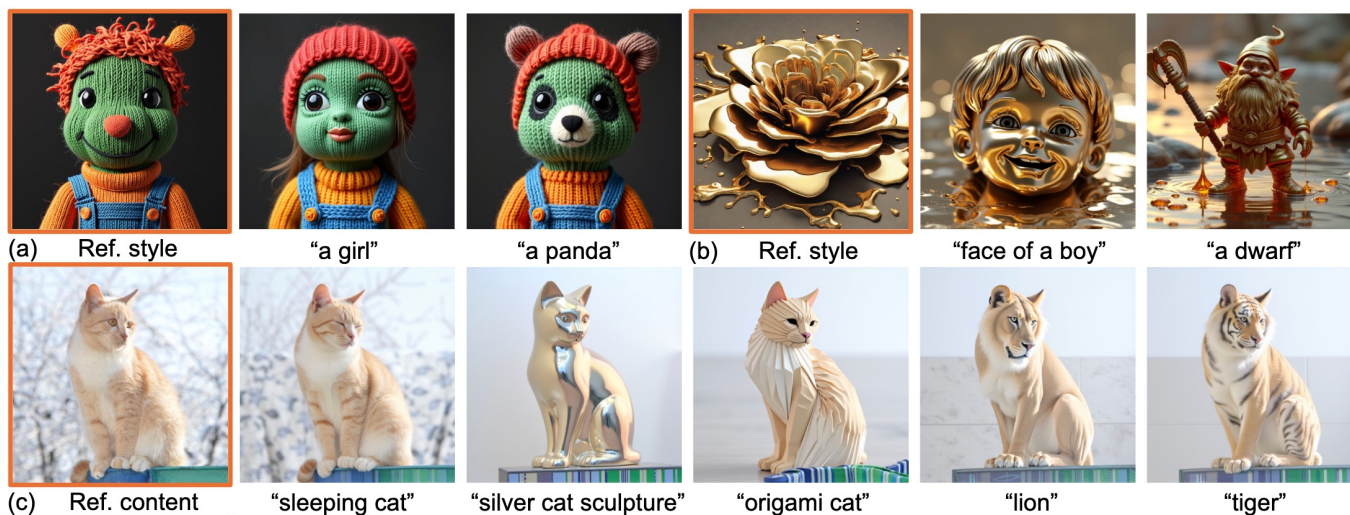


“turtle”

Emerging Foundation Models: Rectified Flows

RF-Inversion

- First Algorithm for Inversion and Editing with Rectified Flow ([ICLR 2025](#))
 - Rectified Flow models are current SOTA (Flux, SD3.5)
 - RF-Inversion **avoids any test-time optimization**
 - Can implement on **edge device** such as Pixel



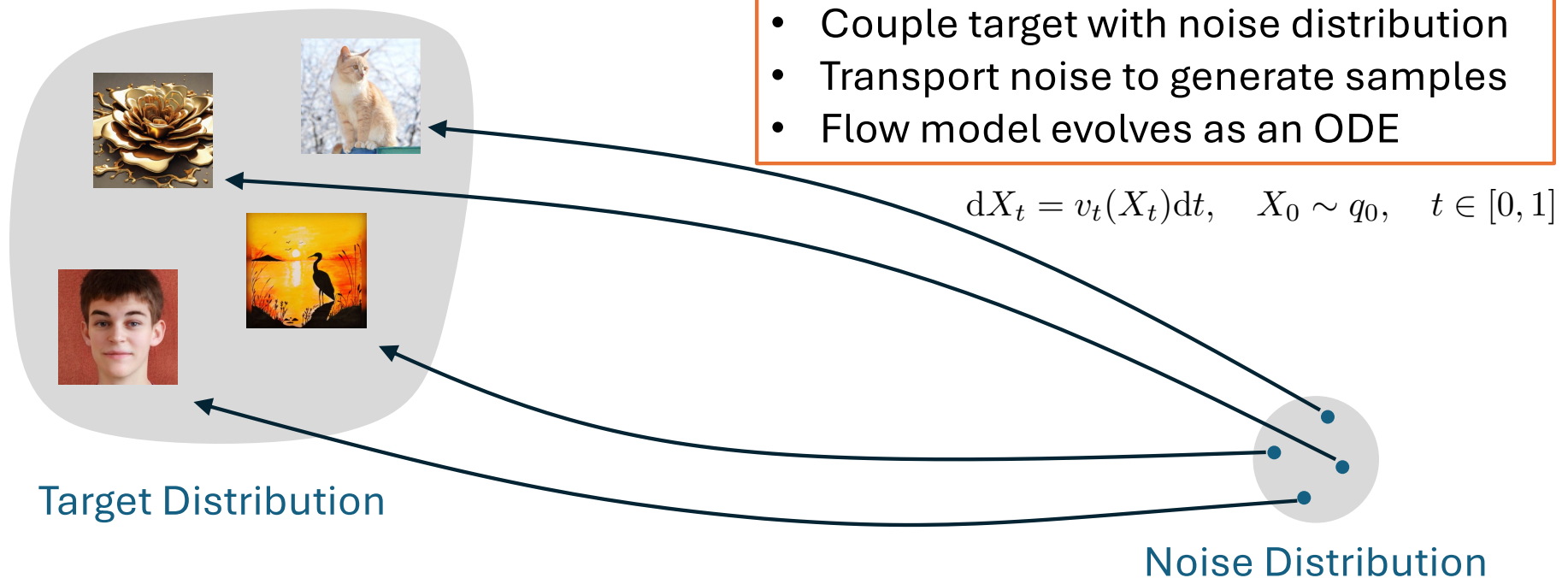
"Semantic Image Inversion and Editing using Stochastic Rectified Differential Equations", Litu Rout, Yujia Chen, Nataniel Ruiz, Constantine Caramanis, Sanjay Shakkottai, Wen-Sheng Chu, *ICLR 2025*

Goal of Rectified Flows

Generate samples from a target distribution given a (large) finite number of samples from that distribution

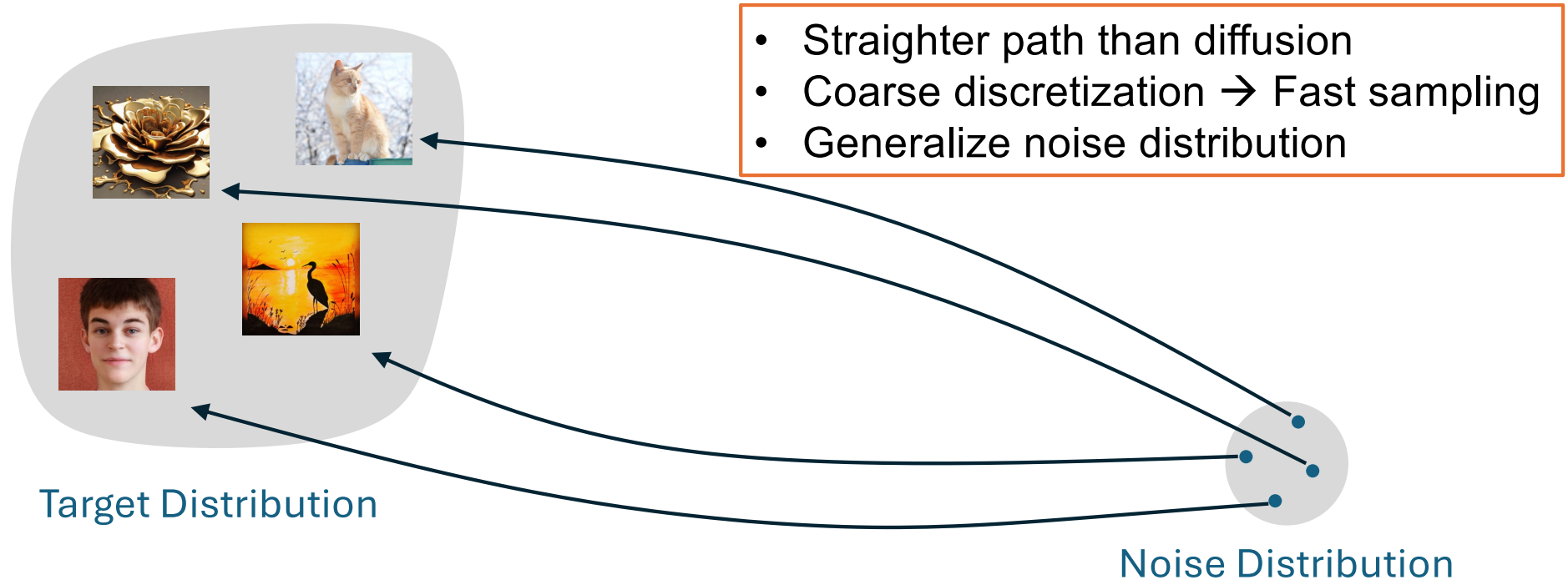
- Couple target with noise distribution
- Transport noise to generate samples
- Flow model evolves as an ODE

$$dX_t = v_t(X_t)dt, \quad X_0 \sim q_0, \quad t \in [0, 1]$$



Benefit of Rectified Flows

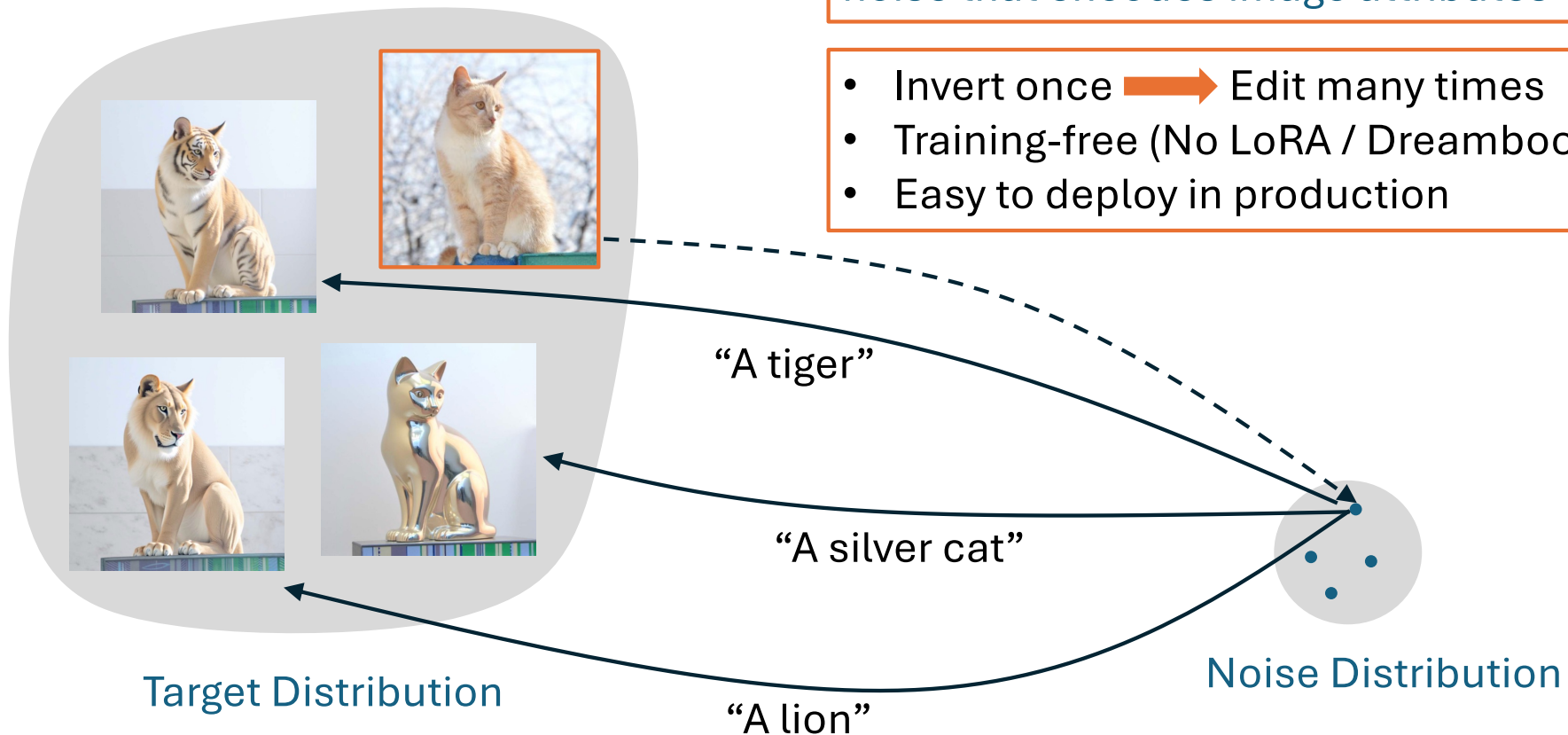
Generate samples from a target distribution given a (large) finite number of samples from that distribution



Inversion with RF (1/2)

Inversion: Transform image into structured noise that encodes image attributes

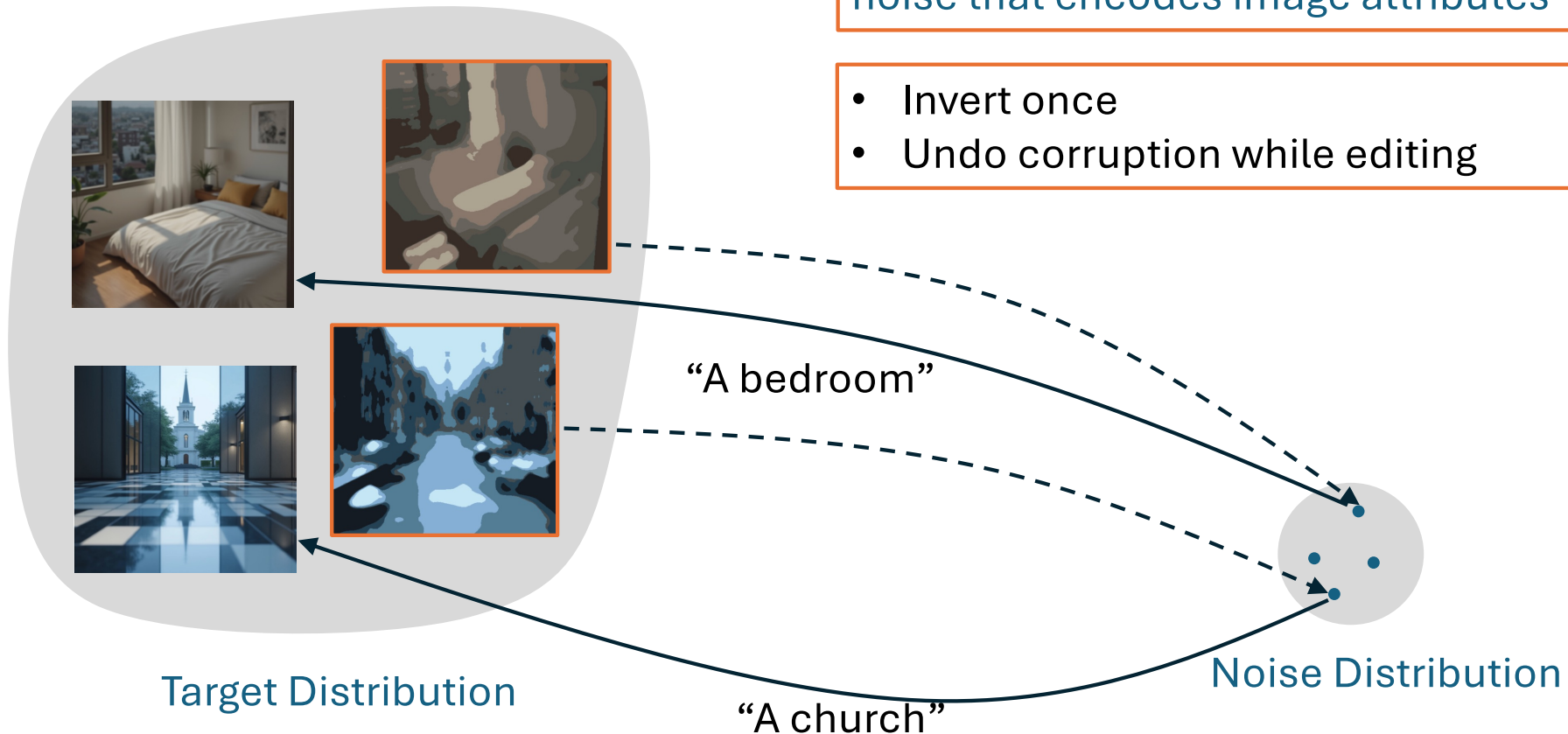
- Invert once → Edit many times
- Training-free (No LoRA / Dreambooth)
- Easy to deploy in production



Inversion with RF (2/2)

Inversion: Transform image into structured noise that encodes image attributes

- Invert once
- Undo corruption while editing



State-of-the-art Inversion

- No algorithm to directly invert and edit using rectified flows
- Other approaches available for **diffusion models**
 - Inversion possible through SDEdit and DDIM inversion (for diffusions) but ...
 - They lead to inconsistencies (preservation of conditioning structure/layout) due to highly non-linear sample paths
 - Alternate methods maintain consistency through expensive training (e.g., DreamBooth, LoRA), test-time optimization (RB Modulation), or complex attention processors (NTI, P2P)

Related Works: Inversion and Editing using Diffusion Models

Method	Training	Optimization	Attention Manipulation
SDEdit [MHS ⁺ 22]	X	X	X
DDIM [SME21]	X	X	X
NTI [MHA ⁺ 23]	X	✓	X
NTI+P2P [HMT ⁺ 22]	X	✓	✓
LEDIT++ [BFK ⁺ 24]	X	X	✓
InstructPix2Pix [BHE23]	✓	X	X
Ours	X	X	X

- Diffusion models are the mainstream approach for inversion and editing
- SoTA methods require training, optimization, or attention manipulation
- SDEdit, DDIM, NTI, NTI+P2P are leading training-free methods
- NTI and P2P require test-time optimization or complex attention processors

Goal: Inversion and Editing using Rectified Flows

- Diffusion models (DMs) traditionally outperformed Rectified Flows (RFs)
- SD3.5 and Flux show RFs can beat DMs
- RF-Inversion or editing remain **unexplored**
- DM inversion techniques face challenges in RFs
 - Training of additional parameters (DreamBooth, StyleDrop)
 - Optimization of latent variables (RB-Modulation)
 - No null conditioning in distilled Flux (NTI)
 - Complex cross-attention processors (P2P)

First efficient inversion and editing using rectified flows without training, optimization, complex attention processors

Introduction to Rectified Flows

Goal: Generate samples from a target distribution given a finite number of samples from that distribution

Approach: Simulate an ODE to generate samples

$$dX_t = v_t(X_t)dt, \quad X_0 \sim q_0, \quad t \in [0, 1]$$

vector field / drift

initialization

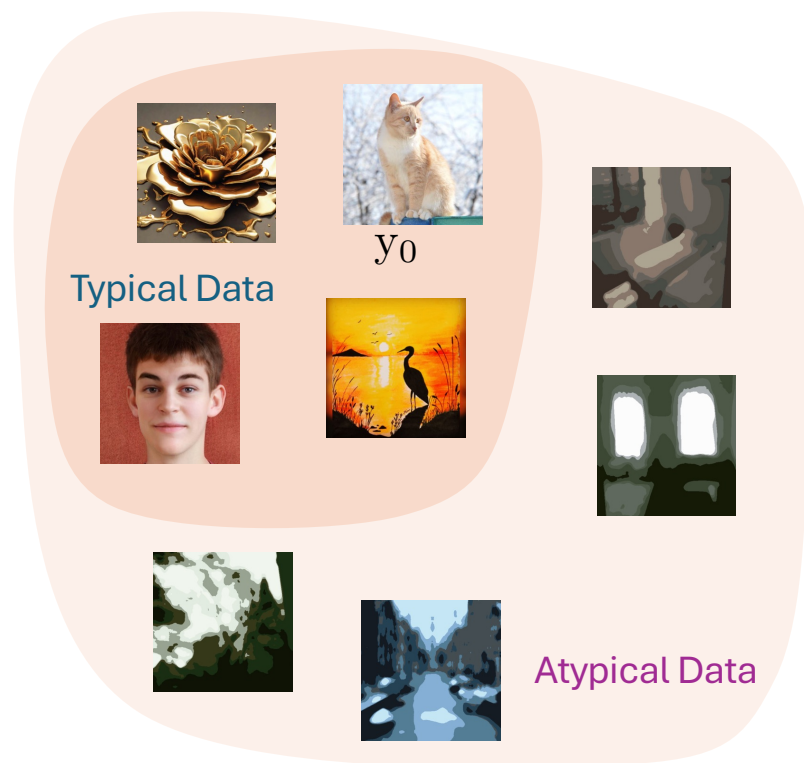
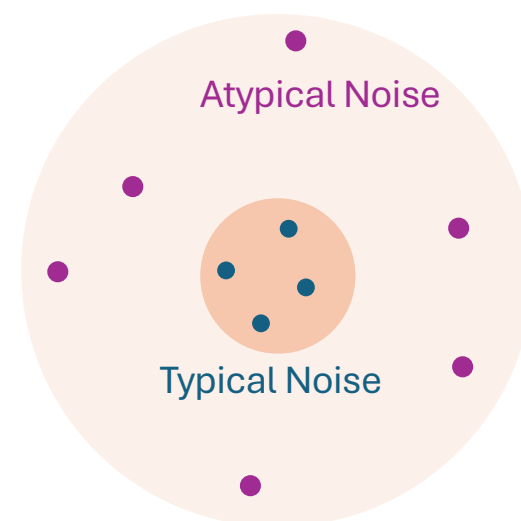
Normalized time

A common choice: $X_0 \sim \mathcal{N}(0, I)$ and $v_t(\cdot) = -u(\cdot, 1 - t; \phi)$

$u(\cdot, \cdot; \phi)$ is a Neural Network (NN) trained using Conditional Flow Matching (CFM)

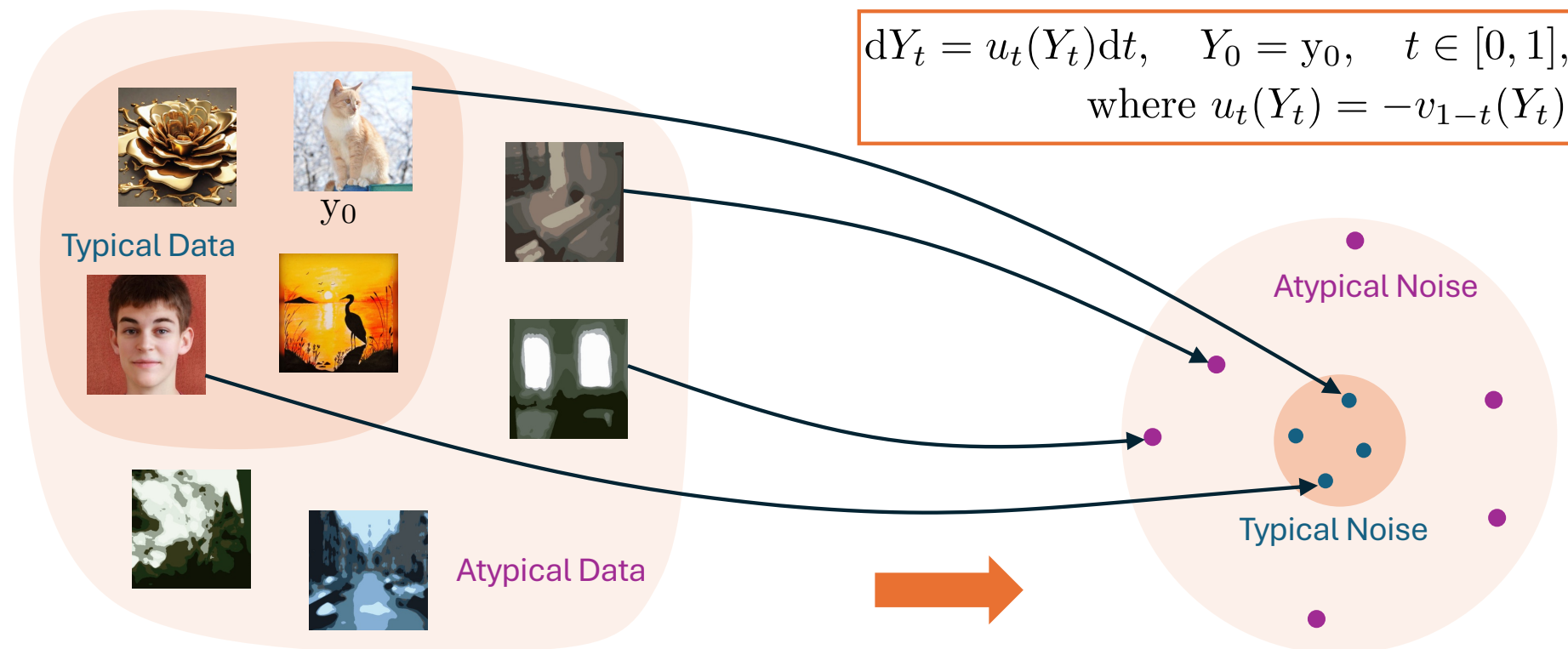
Our Approach: RF-Inversion

Inversion using Rectified Flows



Distributions can be (roughly) grouped into two types: typical and atypical

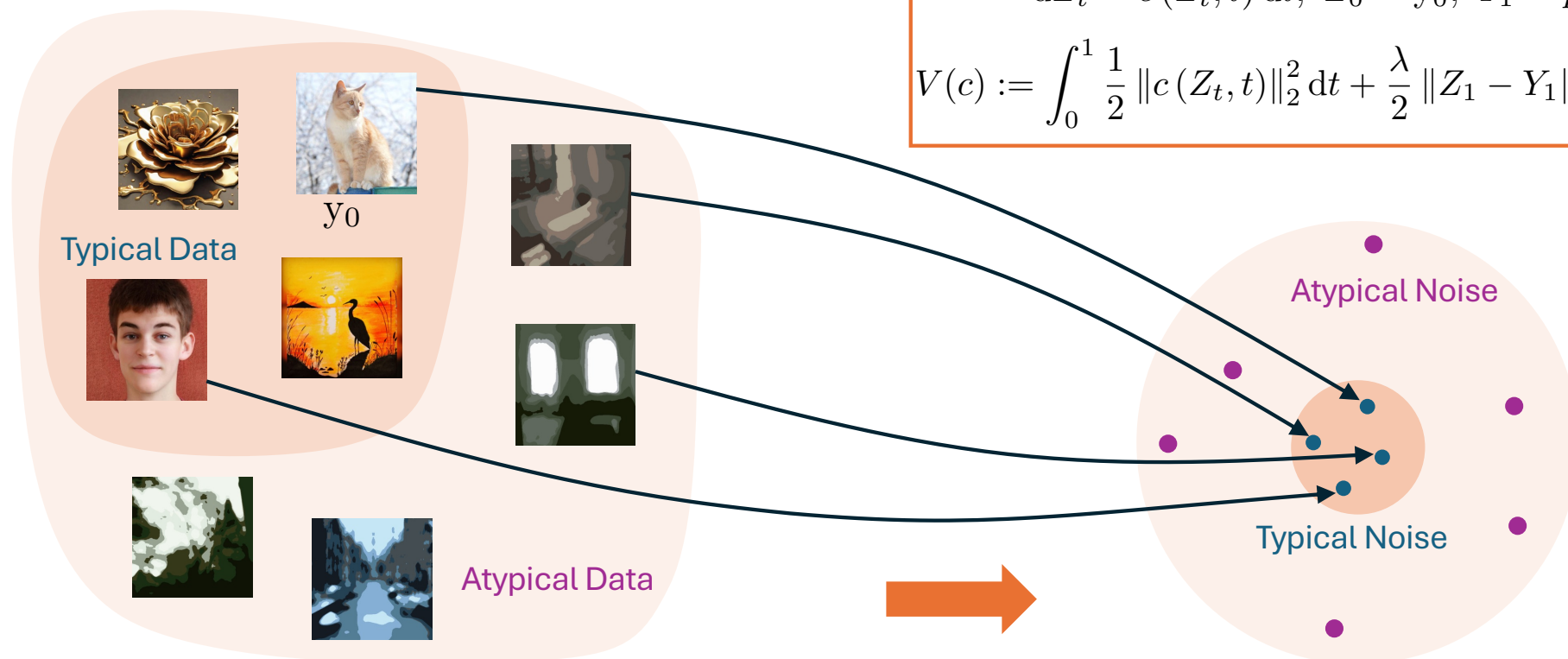
Inversion using Rectified Flows



RF transforms typical image to typical noise; atypical image to atypical noise

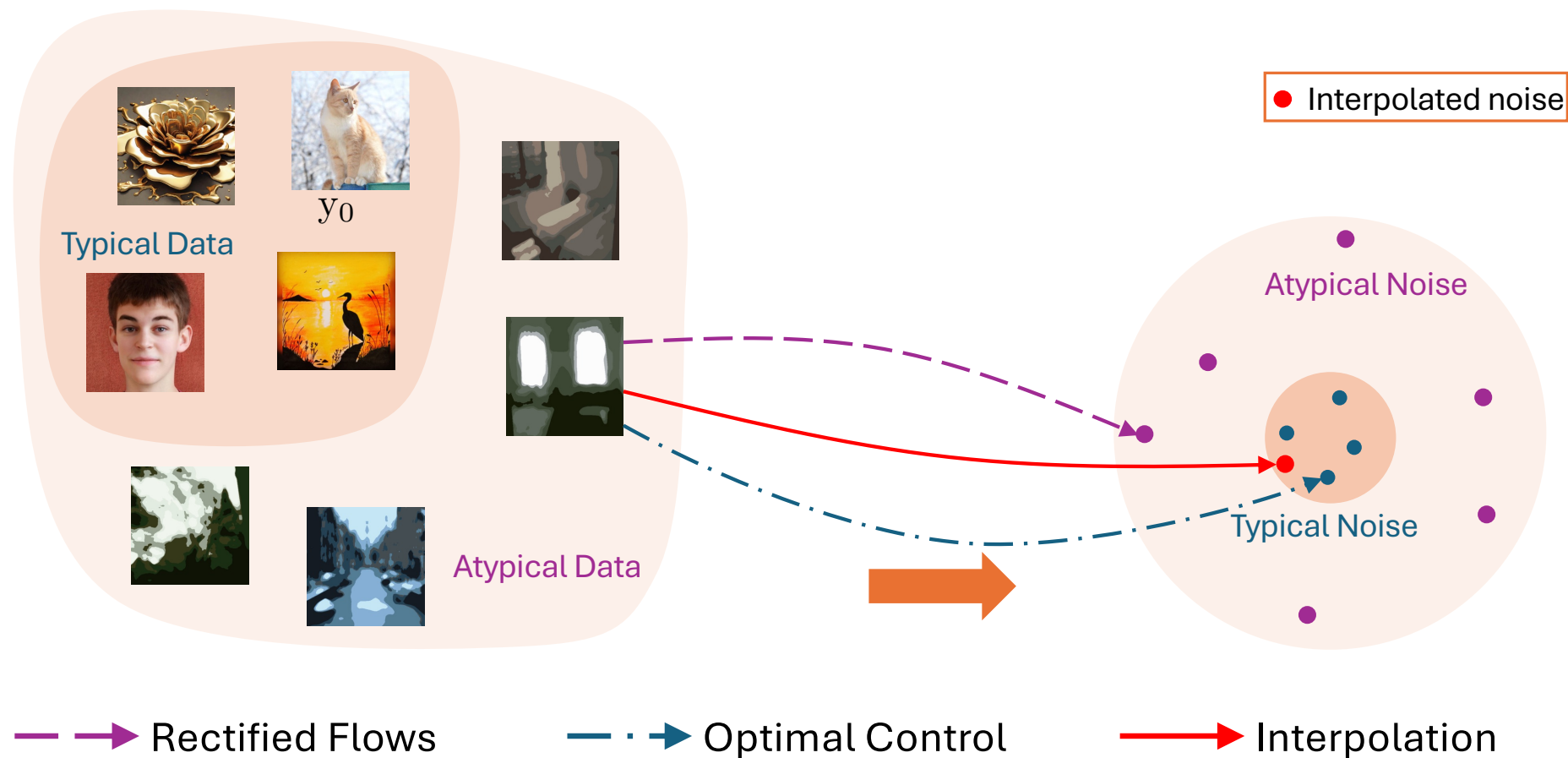
Inversion using Optimal Control

$$dZ_t = c(Z_t, t) dt, \quad Z_0 = y_0, \quad Y_1 \sim p_1$$
$$V(c) := \int_0^1 \frac{1}{2} \|c(Z_t, t)\|_2^2 dt + \frac{\lambda}{2} \|Z_1 - Y_1\|_2^2$$

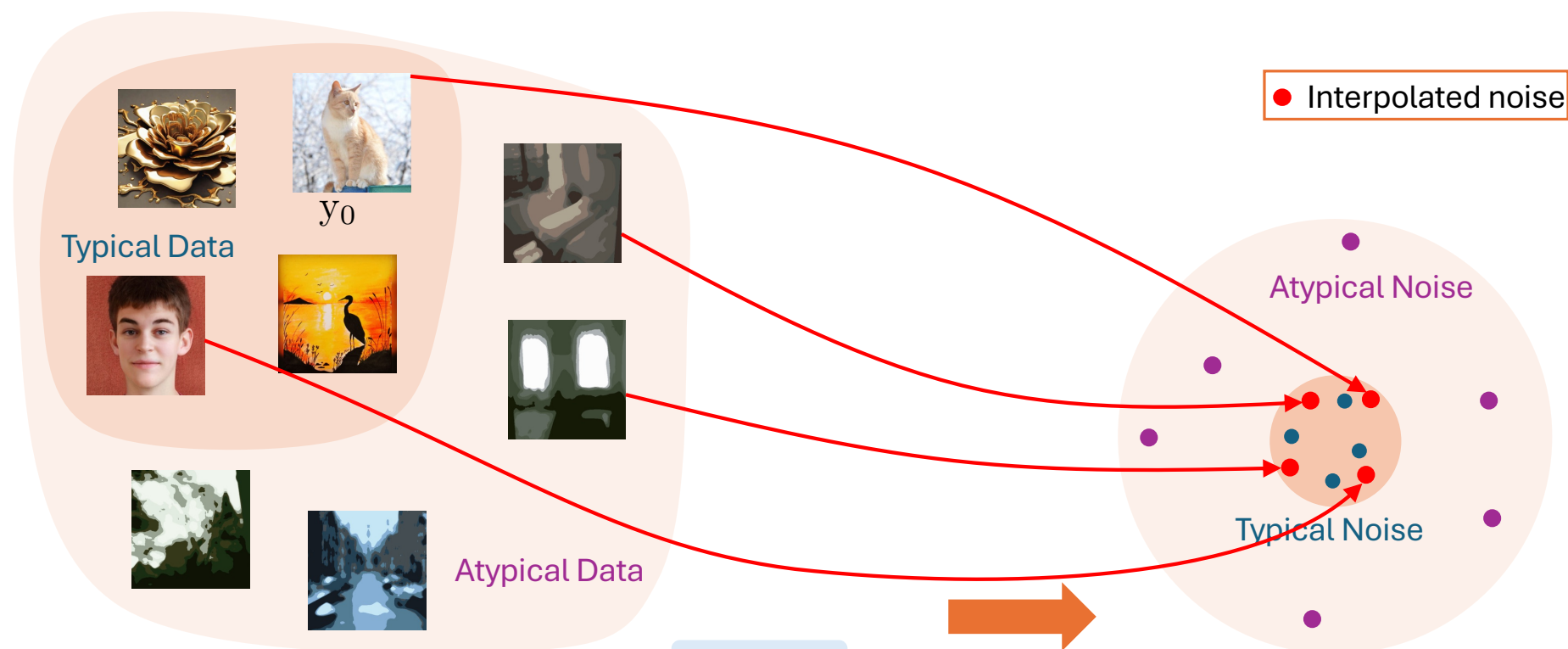


Optimal controller transforms any image to typical noise

Interpolation of the Two Fields

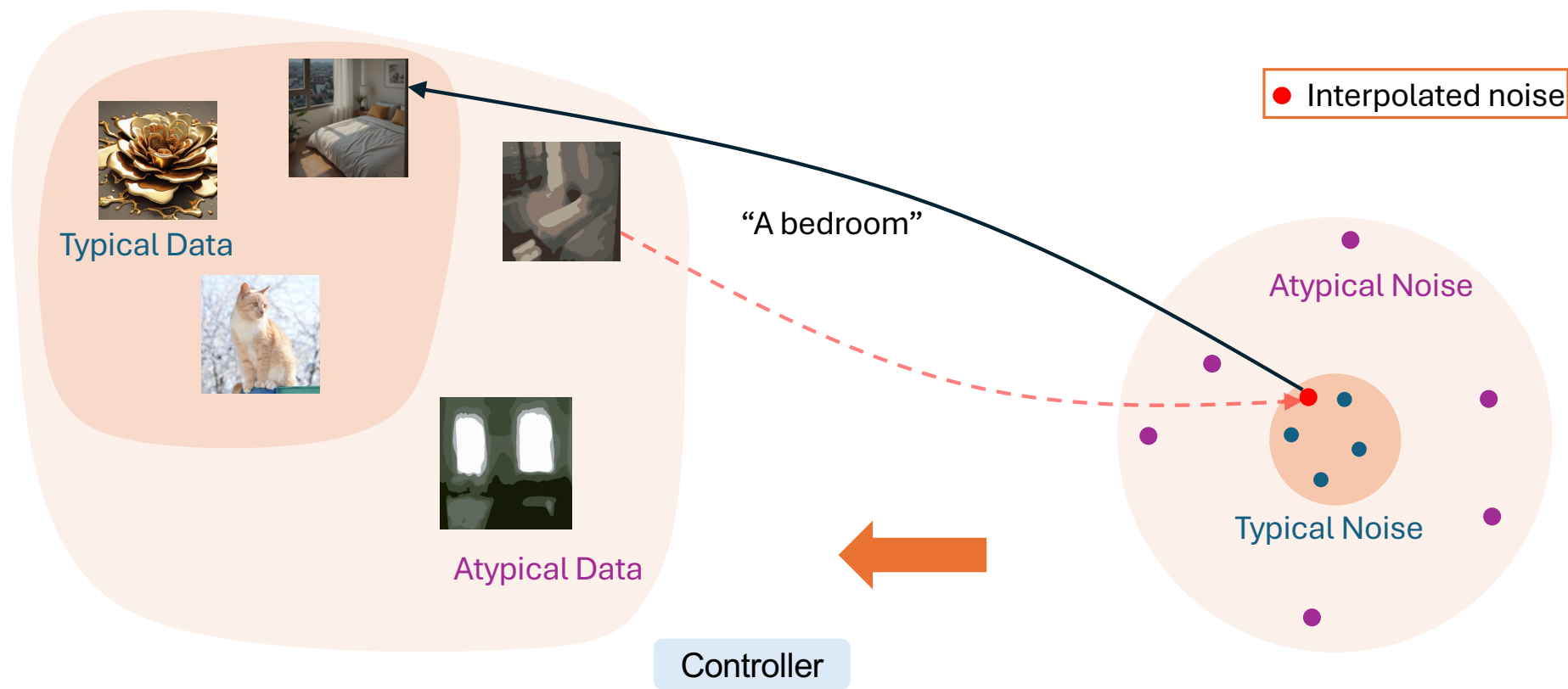


Inversion using Optimally Controlled Rectified Flow



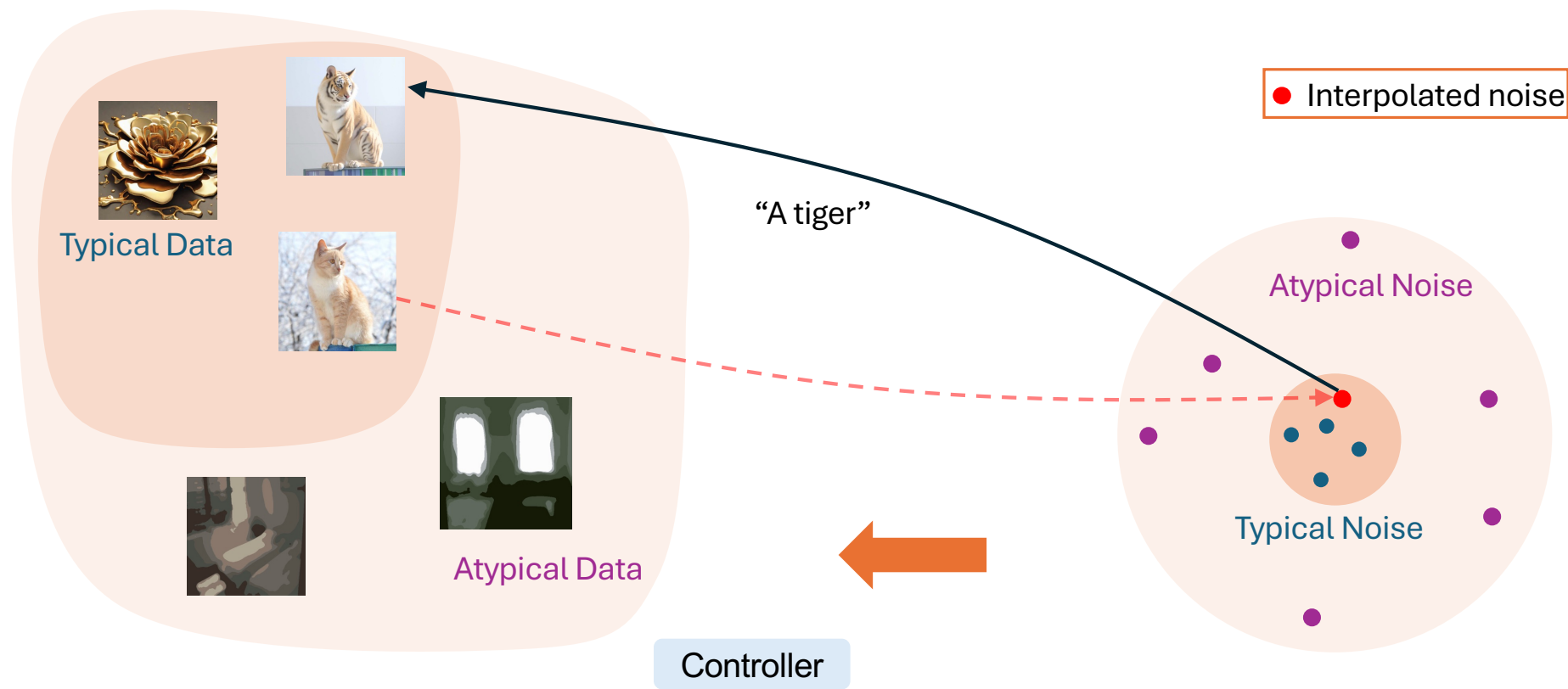
$$dY_t = \left[u_t(Y_t) + \gamma (u_t(Y_t|y_1) - u_t(Y_t)) \right] dt, \quad Y_0 = y_0$$

Generation using Optimally Controlled Rectified Flows



$$dX_t = \left[v_t(X_t, \text{prompt}) + \eta \left(v_t(X_t | y_0) - v_t(X_t, \text{prompt}) \right) \right] dt, \quad X_0 = y_1$$

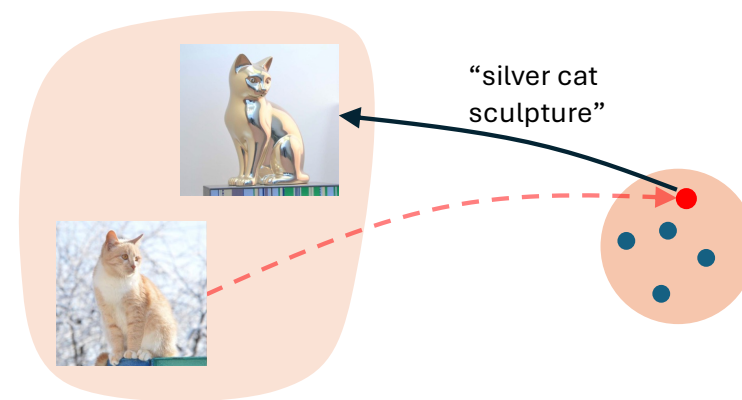
Generation using Optimally Controlled Rectified Flows



$$dX_t = \left[v_t(X_t, \text{prompt}) + \eta (v_t(X_t|y_0) - v_t(X_t, \text{prompt})) \right] dt, \quad X_0 = y_1$$

Counterfactual Sampling (1/2)

- The counterfactual question: “Imagine if this cat was a silver sculpture”
- Three step approach for counterfactual reasoning with an SCM (Pearl et. al. 2016)
 - Noise abduction
 - Action (‘do’)
 - Prediction
- RF-Inversion intuition
 - Going back to noise through **Inversion** $\Leftrightarrow$ **Noise abduction**
 - Doing through **text conditioning** $\Leftrightarrow$ **Action**
 - Generating through **reverse controlled flow** $\Leftrightarrow$ **Prediction**



RF-Inversion interpreted as a prototype of a counterfactual sampler

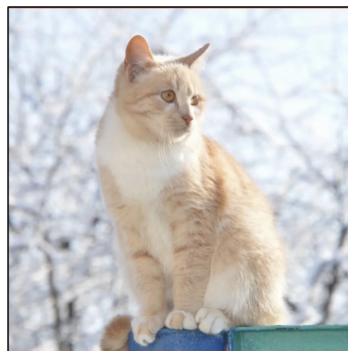
Counterfactual Sampling (2/2)



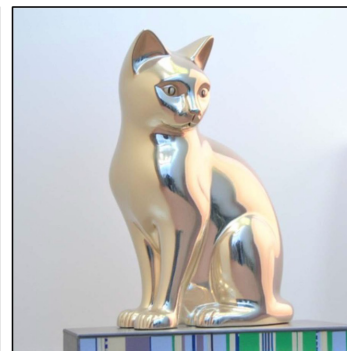
Ref. Image



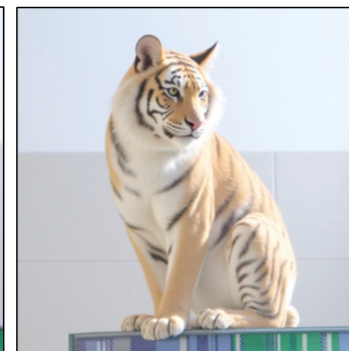
w/o controller +
null-text (“”)



w/ controller +
null-text (“”)



w/ controller +
“silver sculpture”



w/ controller +
“tiger”

- Reference image
- (Reference image → noise → generated image) **without** our controller
- (Reference image → noise → generated image) **with** our controller
 - Conditional vector field is **grounded** to the reference image
- Using text prompts of: ‘A silver cat sculpture’ and ‘A tiger’

Image Inversion and Editing using Rectified Flows



(a) Ref. style



"a girl"



"a panda"



(b) Ref. style



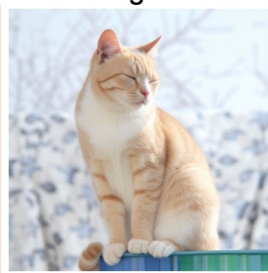
"face of a boy"



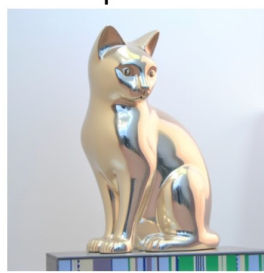
"a dwarf"



(c) Ref. content



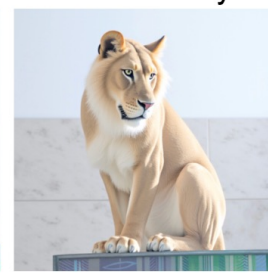
"sleeping cat"



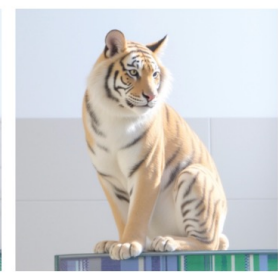
"silver cat sculpture"



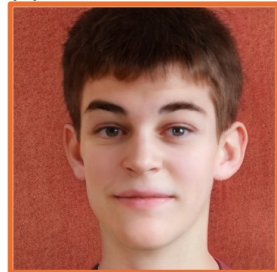
"origami cat"



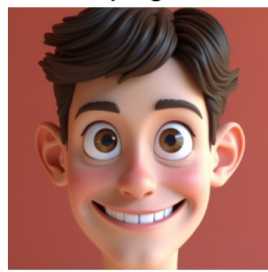
"lion"



"tiger"



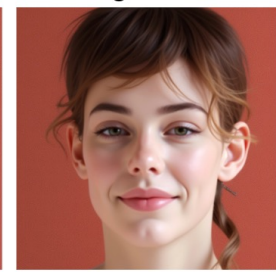
(d) Ref. content



"smiling cartoon"



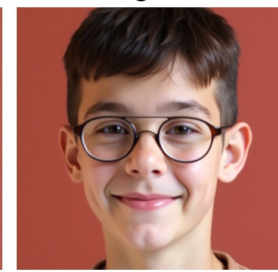
"angry cartoon"



"girl"



"old man"



"young boy+glasses"

A Stochastic Sampler for RF

- Benefits of a Stochastic Sampler for Rectified Flows
 - Many diffusion-based inversion and editing approaches rely on stochastic nature of the diffusion sampler
 - Higher-order solvers benefit from SDE interpretation of diffusion samplers
 - With finer discretization, SDE samplers outperform deterministic samplers in generative modeling, measured by Frechet Inception Distance (FID)
 - SDE samplers show **robustness to corruption** in the initial distribution, i.e., their invariant measure remains the same



Our Approach: Deterministic to Stochastic Sampler

- Closed-form expression for vector field with RF (using Tweedie's Formula):

$$u_t(y_t) = \mathbb{E}_{(Y_0, Y_1) \sim p_1 \times p_0} [Y_1 - Y_0 | Y_t = y_t] = \left[-\frac{1}{1-t} y_t - \frac{t}{1-t} \nabla \log p_t(y_t) \right]$$

- Closed form expression for optimal controller (using minimum principle):

$$u_t(y_t | Y_1) = \frac{Y_1 - y_t}{1-t}$$

- Interpolate between these drift fields to get structured noise:

$$dY_t = u_t(Y_t) + \gamma(u_t(Y_1 | y_1) - u_t(Y_t))dt, \quad Y_0 = y_0, \quad t \in [0, 1]$$

Our Approach: Deterministic to Stochastic Sampler

- Controlled Rectified Flow ODE:

$$dY_t = u_t(Y_t) + \gamma(u_t(Y_1|y_1) - u_t(Y_t))dt, \quad Y_0 = y_0, \quad t \in [0, 1]$$

- Density evolution by continuity equation:

$$\frac{\partial p_t(Y_t)}{\partial t} = \nabla \cdot \left[\left(\frac{1}{1-t} (Y_t - \gamma y_1) + \frac{(1-\gamma)t}{1-t} \nabla \log p_t(Y_t) \right) p_t(Y_t) \right]$$

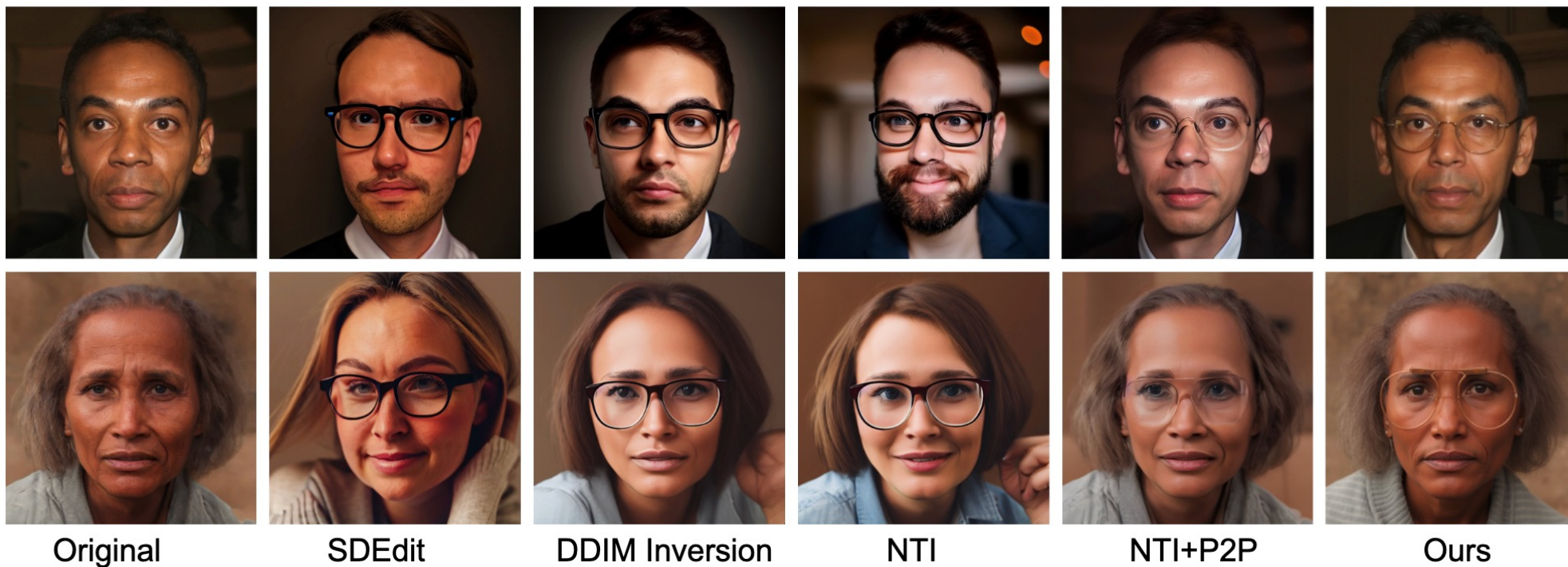
- Controlled SDE using Fokker-Planck equation:

$$dY_t = -\frac{1}{1-t} (Y_t - \gamma y_1) dt + \sqrt{\frac{2(1-\gamma)t}{1-t}} dW_t, \quad Y_0 \sim p_0$$

Analogous approach for deriving SDE for Generation

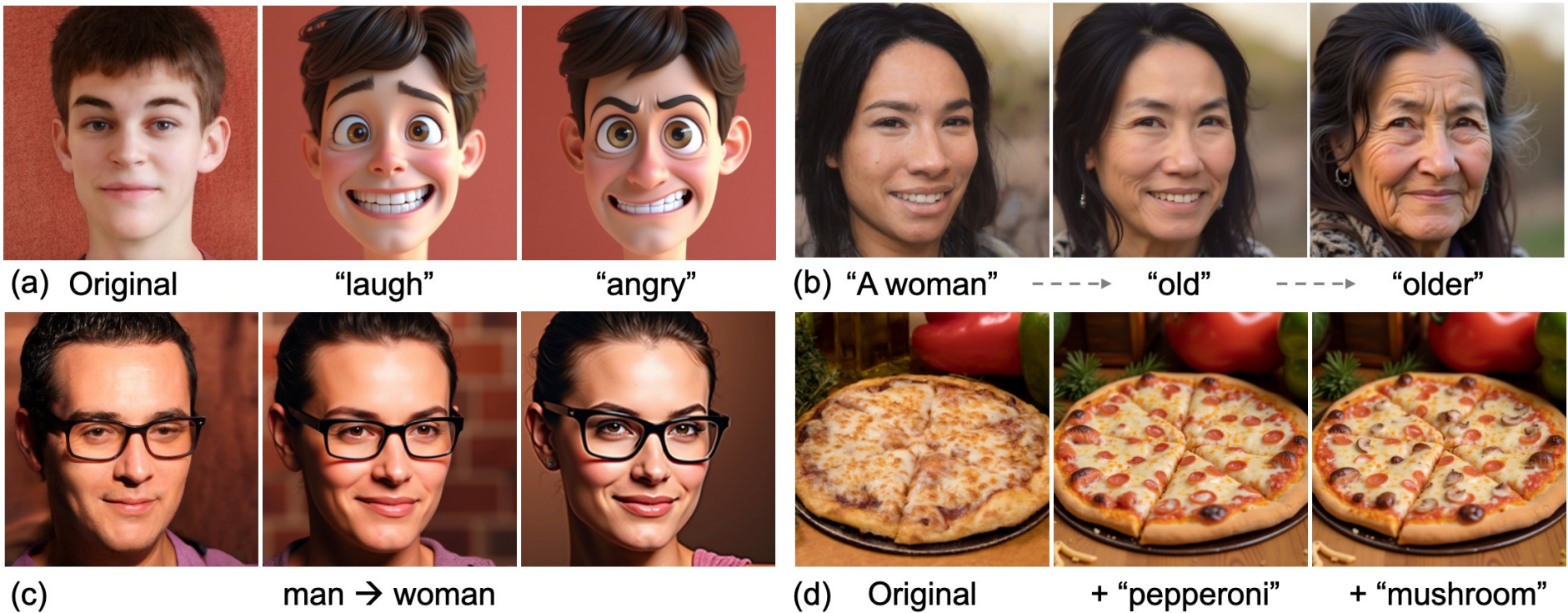
Experiments

Experiments: Identity Preservation in Face Editing



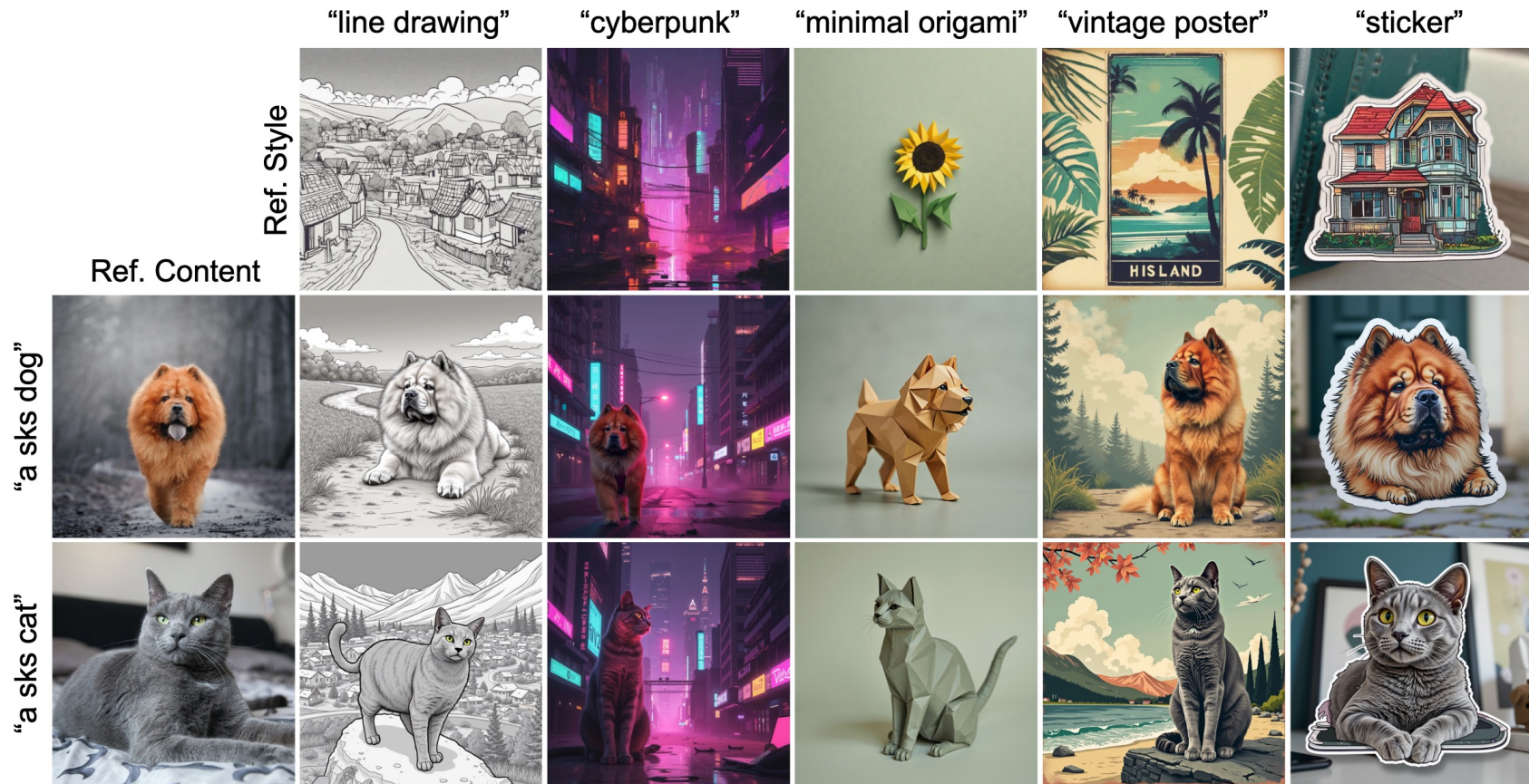
Prompt: "... + wearing glasses"

Experiments: Semantic Image Editing



Editing (a) stylized expression, (b) age, (c) gender, and (d) object insert

Experiments: Content-style composition



Experiments: Generalization to another flow model SD3.5



(a) Ref. style



"A boat"



"A car"



(b) Ref. Style



"A mad scientist"



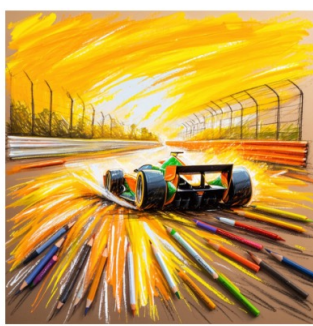
"A lion boy"



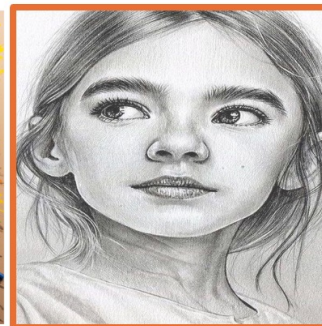
(c) Ref. style



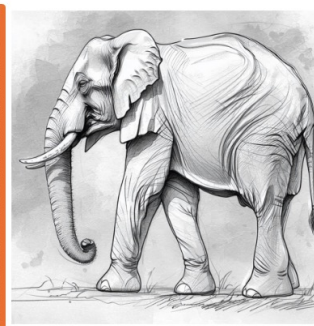
"A house on a hill"



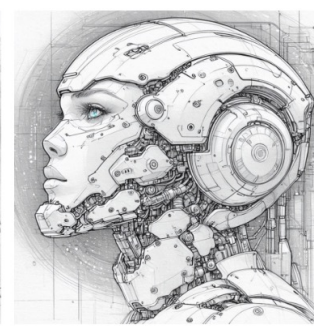
"A racing car"



(d) Ref. Style



"An elephant"



"A futuristic robot"

(a,b) Generated reference style (c,d) Hand drawn reference style

Please see: <https://openreview.net/forum?id=bniNPG5A32> for reference image credits

Experiments: Generative modeling using rectified flow SDE

Flux



FluxSDE (Ours)

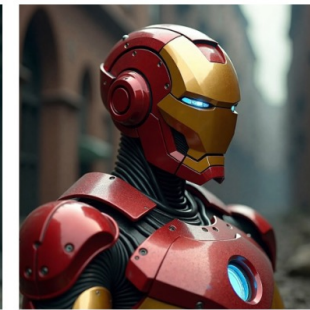


Prompt: "portrait, looking to one side of frame, lucid dream-like 3d model of an owl, video game character, forest, wonderland, photorealism, cinematic artistic style."

Flux



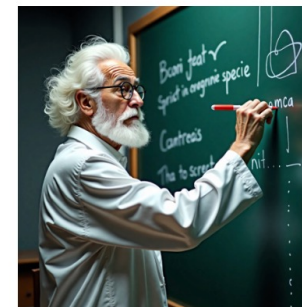
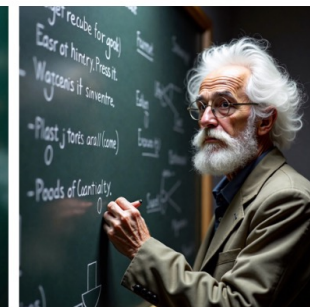
FluxSDE (Ours)



Prompt: "a robot with a reflective helmet, iron armor, photorealistic, in shades of red and golden brown, dark gloomy environment, epic scene."



Prompt: "a dragon soaring through the sky, battle ground, people fighting on the ground."



Prompt: "a genius scientist, in his 60s, standing, writing on the black board, white hair, white beard, round spectacles."

Community Developments

Day 1
(Oct 14)

- Paper released on ArXiv: <https://arxiv.org/pdf/2410.10792>
- Project page: <https://rf-inversion.github.io/>
- ComfyUI code reproduced results from RF-Inversion (<24hrs) <https://tinyurl.com/xwv24wbp>

Week 1-2

- 8 Steps Style and Face Transfer with Unsampling and RF Inversion
- YouTube Tutorial: https://www.youtube.com/watch?v=H_G2AaLWN2o
- Endless creative possibilities: <https://tinyurl.com/57b72ks4>
- Test of RF-Inversion on style transfer: <https://tinyurl.com/bdhs4vy3>
- Podcasts: <https://tinyurl.com/3x496jky>, <https://tinyurl.com/3djmevef>

Week 3-4

- Animate movies using RF-Inversion: <https://tinyurl.com/xwv24wbp>
- Mochi Video Editing with RF-Inversion: <https://tinyurl.com/yeyej7x8>
- Integration in diffusers from HuggingFace: <https://tinyurl.com/2avrfzh5>
- **Follow up works:** ReCapture (Zhang et al.), RF-Edit (Wang et al.), AnimateAnything (Lei et al.), EditAway (wang et al.), MyTimeMachine (Qi et al.), HeadRouter (Xu et al.)

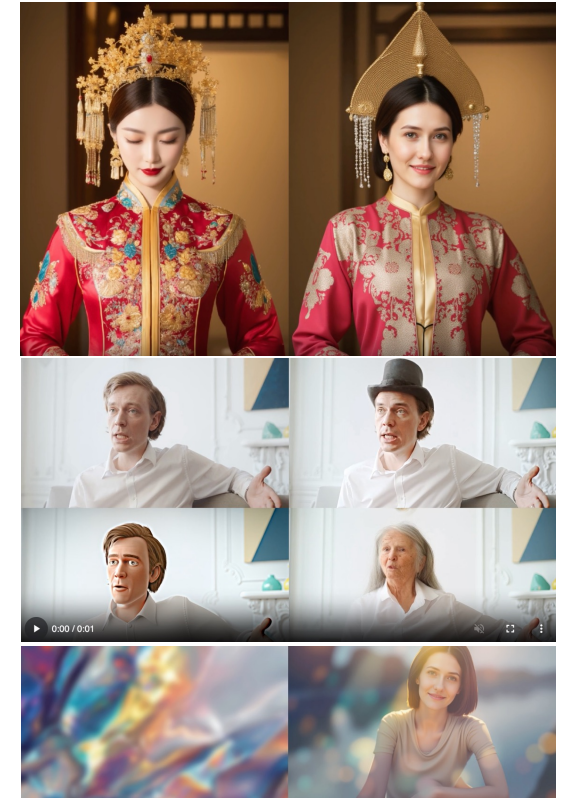
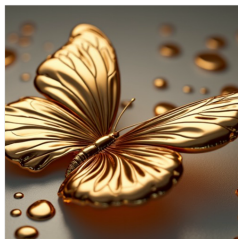


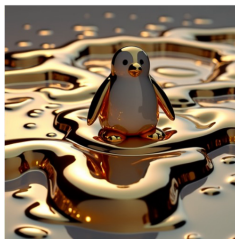
Image Credit: <https://tinyurl.com/57b72ks4>, <https://tinyurl.com/xwv24wbp>

Summary

- First efficient inversion and editing for rectified flows
 - Interpolates two vector fields
- Stochastic equivalence between rectified flow ODE and SDE
- State-of-the-art zero-shot performance w/o training, optimization, prompt tuning and complex attention processors
- Effectiveness in stroke-to-image synthesis, face editing, stylization, content-style composition, w/ large-scale human evaluations



"a butterfly"



"a baby penguin"



"a boat"



"a piano"