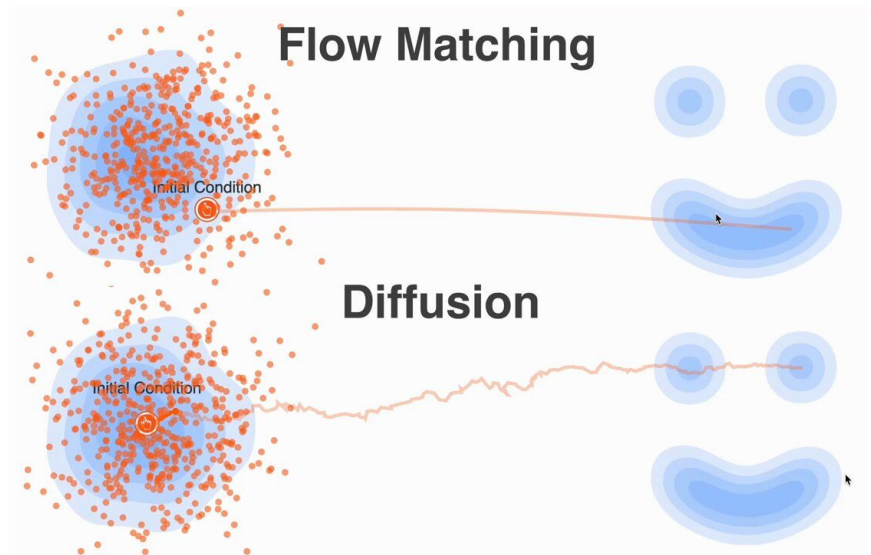


Lab 4 – Flow Matching



TA: Po-Fan Yu

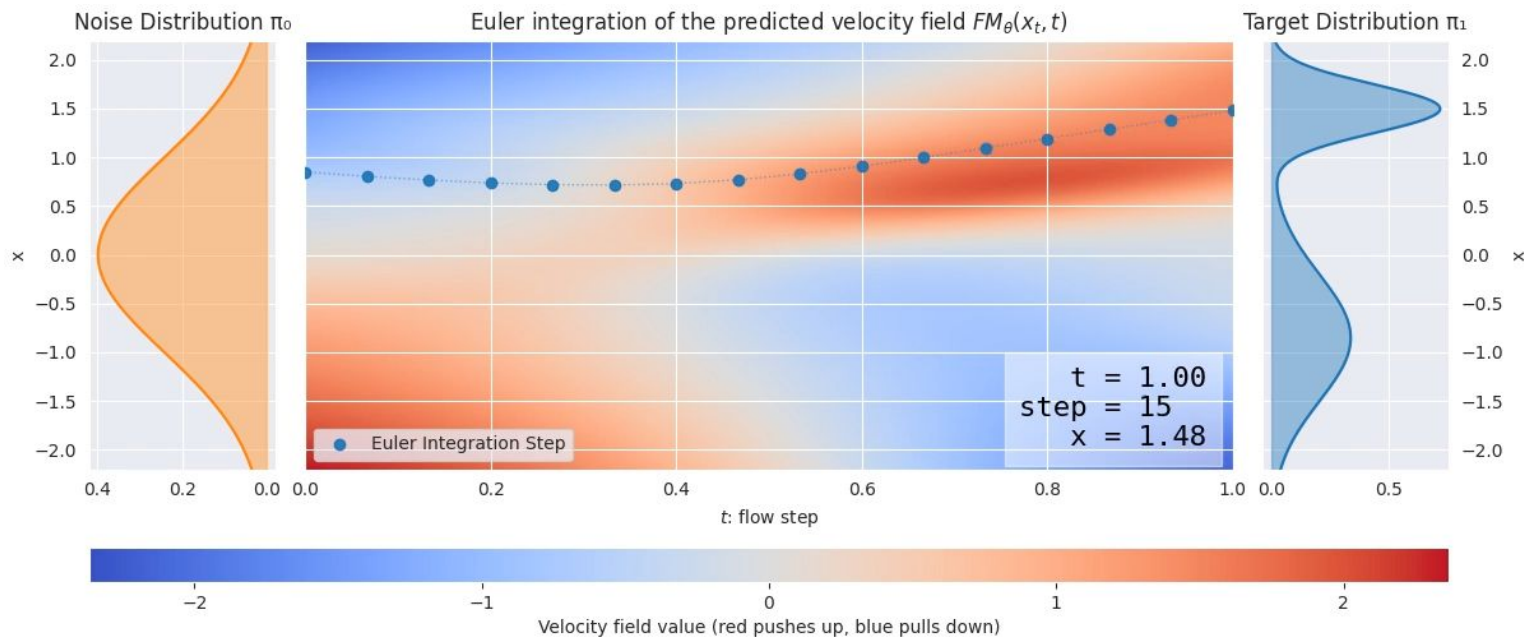
Due: 2026/11/12 (Thu.) 23:59

code : <https://github.com/ben13579/Lab4-FlowMatching>

Outline

- Introduction
 - Flow Matching
 - Rectified Flow & Reflow
 - InstaFlow
- Implementation
 - Task 1: 2D Flow Matching
 - Task 2: Image Flow Matching with CFG
 - Task 3: Rectified Flow
 - (Bonus) Task 4: InstaFlow One-Step Generation
- Evaluation & Grading
- Report
- Submission

Flow Matching



$$\mathcal{L} = \mathbb{E}_{t \sim \mathcal{U}[0,1]} \mathbb{E}_{x_t \sim p_t} [\|v_\theta(x_t, t) - u_t(x_t)\|^2]$$

Flow Matching

$$\mathcal{L} = \mathbb{E}_{t \sim \mathcal{U}[0,1]} \mathbb{E}_{\mathbf{x}_t \sim p_t} [\|v_\theta(\mathbf{x}_t, t) - u_t(\mathbf{x}_t)\|^2]$$

is equivalent to

$$\mathcal{L} = \mathbb{E}_{t \sim \mathcal{U}[0,1]} \mathbb{E}_{\mathbf{x}_1 \sim p_1, \mathbf{x}_t \sim p_{t|1}(\mathbf{x}_t|\mathbf{x}_1)} [\|v_\theta(\mathbf{x}_t, t) - u_t(\mathbf{x}_t|\mathbf{x}_1)\|^2]$$

Flow Matching – Optimal Transfer

Conditional probability paths $p_{t|1}(\mathbf{x}_t|\mathbf{x}_1)$:

$$p_{t|1}(\mathbf{x}_t|\mathbf{x}_1) = \mathcal{N}(t\mathbf{x}_1, (1-t)\mathbf{I})$$

Conditional flow map $\phi_t(\mathbf{x}_t)$:

$$\phi_t(\mathbf{x}_t|\mathbf{x}_1) = (1-t)\mathbf{x}_t + t\mathbf{x}_1 \quad \text{Linear Interpolation!}$$

Conditional vector field $u_t(\mathbf{x}_t|\mathbf{x}_1)$:

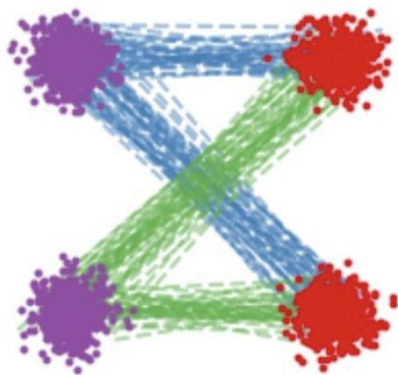
$$u_t(\mathbf{x}_t|\mathbf{x}_1) = \frac{\mathbf{x}_1 - \mathbf{x}_t}{1-t}$$

Flow Matching – Optimal Transfer

$$\mathcal{L} = \mathbb{E}_{t \sim u[0,1]} \mathbb{E}_{\mathbf{x}_1 \sim p_1, \mathbf{x}_0 \sim p_0} [\|v_\theta(\phi_t(\mathbf{x}_0|\mathbf{x}_1), t) - u_t(\phi_t(\mathbf{x}_0|\mathbf{x}_1)|\mathbf{x}_1)\|^2]$$

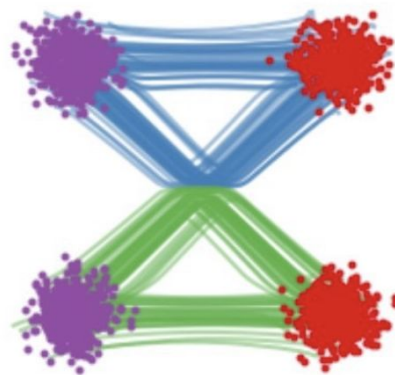
1. Sample $\mathbf{x}_0 \sim p_0$ and $\mathbf{x}_1 \sim p_1$.
2. Compute $\mathbf{x}_t = \phi_t(\mathbf{x}_0|\mathbf{x}_1) = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$ as a linear interpolation between $\mathbf{x}_0$ and $\mathbf{x}_1$.
3. Compute $u_t(\mathbf{x}_t|\mathbf{x}_1) = \frac{\mathbf{x}_1 - \mathbf{x}_t}{1-t} = \mathbf{x}_1 - \mathbf{x}_0$.
4. Backpropagate through the regression loss.

Rectified Flow



(a) Linear interpolation

$$X_t = tX_1 + (1 - t)X_0$$



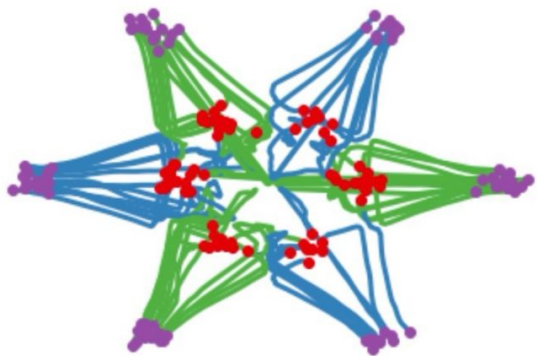
(b) Rectified flow Z_t

induced by (X_0, X_1)

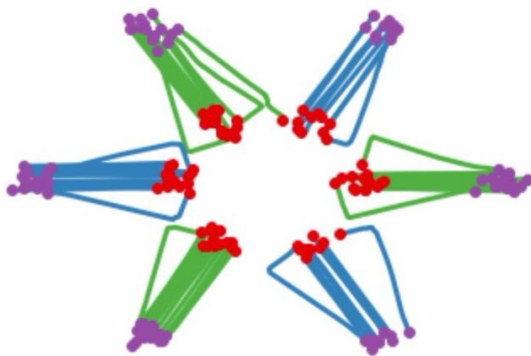
Rectified Flow

- **Goal:** We want a more straight trajectory
- **Why?** Because in this way we can sample fewer steps during inference
- Improve inference speed!
- Faster inference but similar quality

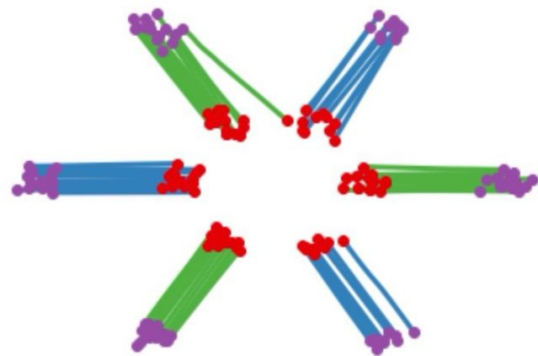
Rectified Flow



(a) The 1st rectified flow Z^1



(b) Reflow Z^2

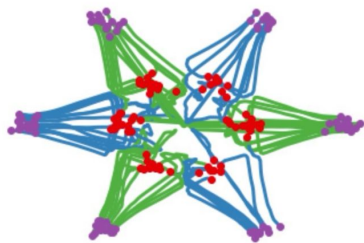


(c) Reflow Z^3

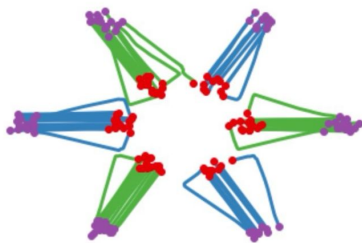
Collect (X_0, X_1) by sampling

Rectified Flow – *Reflow* Recipe

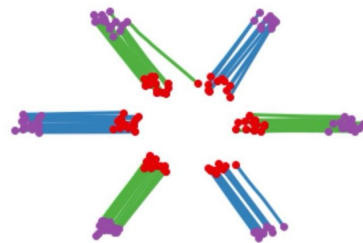
1. Train a Flow Matching model Z^0 using real-world datasets
2. Sample noise X_0 from normal distribution, use Z^0 to generate corresponding X_1
3. Collect N pairs of (X_0, X_1)
4. Use these N pairs of data to train another Flow Matching model Z^1 from scratch
5. Sample noise X_0' from normal distribution, use Z^1 to generate corresponding X_1'
6. Collect N pairs of (X_0', X_1')
7. Use these N pairs of data to train another Flow Matching model Z^2 from scratch
8. Repeat...



(a) The 1st rectified flow Z^1



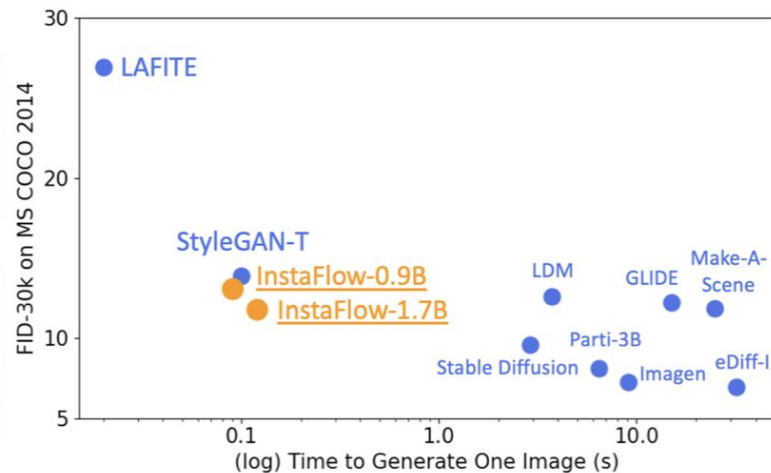
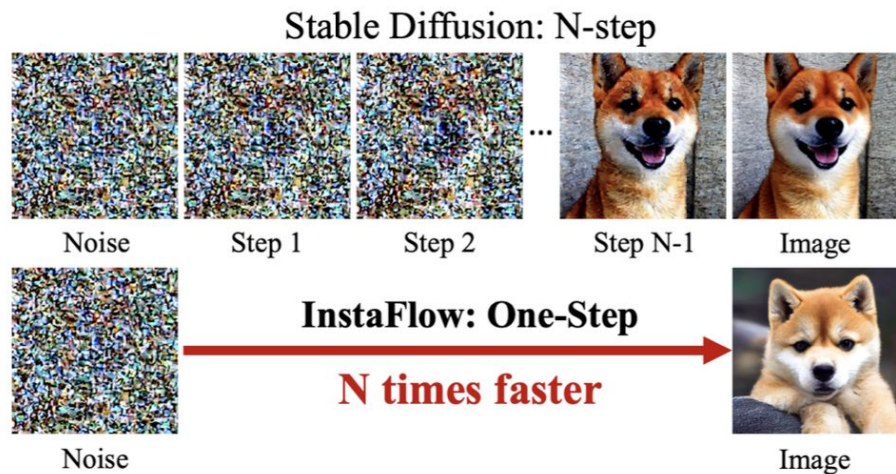
(b) Reflow Z^2



(c) Reflow Z^3

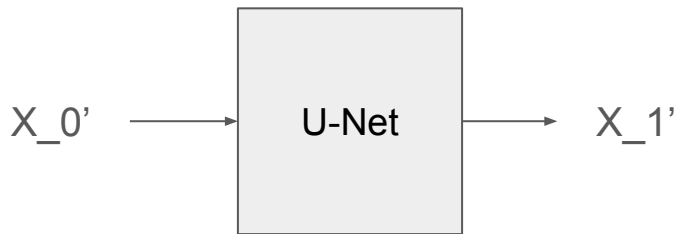
InstaFlow

- **Goal:** Distill DDPM / DDIM model (e.g. Stable Diffusion) into one-step generative model



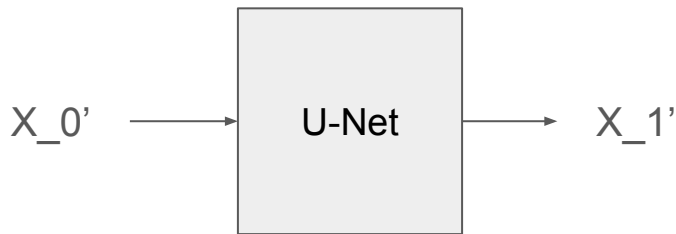
InstaFlow

- **Goal:** Distill DDPM / DDIM model (e.g. Stable Diffusion) into one-step generative model
- **How?**
 - a. Sample from Stable Diffusion, collecting N pairs of (X_0, X_1)
 - b. Train a flow matching model Z^0 using these pairs of data
 - c. Reflow & Sample training data (X_0', X_1') for final distillation
 - d. **Distill:** Force a simple network (e.g. U-Net) to learn the mapping between (X_0', X_1')



InstaFlow

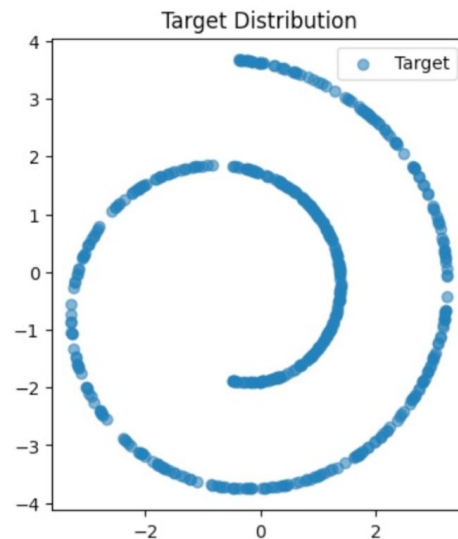
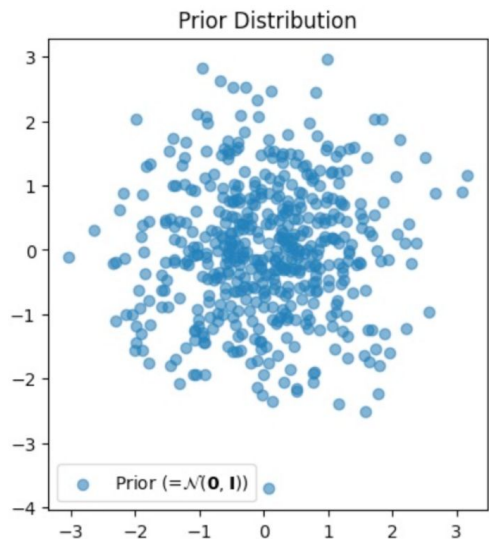
- **Goal:** Distill DDPM / DDIM model (e.g. Stable Diffusion) into one-step generative model
- **How?**
 - a. Sample from Stable Diffusion, collecting N pairs of (X_0, X_1)
 - b. Train a flow matching model Z^0 using these pairs of data
 - c. Reflow & Sample training data (X_0', X_1') for final distillation
 - d. **Distill:** Force a simple network (e.g. U-Net) to learn the mapping between (X_0', X_1')



One-step Generation!

Task 1 – 2D Flow Matching

- Similar to Lab 1, but we use Flow Matching objective instead of DDPM
- Same network architecture



Task 1 – 2D Flow Matching

Complete the following in `task1_2d_flow_matching/`:

`fm.py`:

-  `FMScheduler.compute_psi_t()` - Conditional flow $\psi_t(x|x_1)$
-  `FMScheduler.step()` - Euler ODE solver
-  `FlowMatching.get_loss()` - CFM training objective
-  `FlowMatching.sample()` - Sampling with CFG support

`network.py`:

-  `SimpleNet` - MLP velocity network (reuse from Assignment 1)

Task 2 – Image Flow Matching with CFG



Task 2 – Image Flow Matching with CFG

Complete the following in `image_common/fm.py` (same TODOs as Task 1, adapted for images):

-  `FMScheduler.compute_psi_t()` - Conditional flow for images
-  `FMScheduler.step()` - Euler ODE solver
-  `FlowMatching.get_loss()` - CFM training loss
-  `FlowMatching.sample()` - Sampling with Classifier-Free Guidance

Task 3 – Rectified Flow

Complete in `task3_rectified_flow/generate_reflow_data.py`:

-  Generate synthetic (x_0, x_1) pairs by following the ODE trajectory of the Task 2 model

Task 4 – InstaFlow (Bonus)

Complete the following in `image_common/instafLOW.py`:

InstaFlowModel.get_loss() - One-step distillation objective (Eq. 6):

-  Create $t=0$ tensor for batch
-  Predict velocity $v(x_0, 0 \mid \text{class_label})$
-  Compute prediction: $x1_pred = x_0 + v_pred$
-  Calculate L2 loss: $\|x1_pred - x1\|^2$
-  Add optional LPIPS perceptual loss

InstaFlowModel.sample() - One-step inference:

-  Initialize x_0 with noise
-  Create $t=0$ tensor
-  Predict velocity (CFG is "baked in" during training)
-  Generate: $x_1 = x_0 + v_pred$

Evaluation & Grading

Component	Points	Description
Task 1	20 pts	2D Flow Matching Implementation
Task 2	25 pts	Image Flow Matching with Classifier-Free Guidance
Task 3	25 pts	Rectified Flow for Faster Sampling
Task 4	+10 pts	(Bonus) InstaFlow One-Step Generation
Report	30 pts	Analysis, Visualizations, and Discussion
Total	100 pts	+ 10 pts Bonus

Evaluation & Grading – Task 1

Chamfer Distance	Points
< 40	20 pts
$40 \leq CD < 60$	10 pts
≥ 60	0 pts

Evaluation & Grading – Task 2

FID Score (CFG=7.5)	Points
< 30	25 pts
$30 \leq \text{FID} < 50$	15 pts
≥ 50	0 pts

Evaluation & Grading – Task 3

Performance (CFG=7.5)	Points
FID < 30 with 5 steps	25 pts
$30 \leq \text{FID} < 50$ with 5 or 10 steps	15 pts
Otherwise	0 pts

Evaluation & Grading – Task 4

FID Score (1 step)	Points
< 30	+10 pts
$30 \leq \text{FID} < 50$	+5 pts
≥ 50	0 pts

Evaluation & Grading – Report

- Format: PDF
- Length: 6-10 pages (excluding code/appendices)
- Includes:
 - Introduction (2 pts)
 - Methodology (8 pts)
 - Experiments & Results (12 pts)
 - Discussion (6 pts)
 - Conclusion (2 pts)

Submission – Format

- Create `{STUDENT_ID}_lab4.zip` with the following structure:

```
./submission/  
├── task1_2d_flow_matching/  
│   ├── fm.py                # TODO implementations  
│   └── network.py           # SimpleNet  
├── task3_rectified_flow/  
│   └── generate_reflow_data.py # Data generation logic  
├── image_common/  
│   ├── fm.py                # Image FM implementation  
│   └── instafLOW.py          # (Optional) InstaFlow  
└── report.pdf                # Complete analysis report
```

Submission – Additional Files

- **Additional Files:** If you modified any scripts beyond the required TODO files (e.g., training scripts, data loaders, network architectures), include them in your submission following the original directory structure. In your report, provide a section explaining:
 - Which additional files were modified
 - Rationale for each modification
 - How the changes improve performance or functionality

Submission – Policy

- Deadline: 2026/11/12 (Thu.) 23:59
- Platform: E3
- Late Policy: No late submissions accepted