Homework 2: Flow Matching & DDIM Implementation

CMU 10-799: Diffusion & Flow Matching

Spring 2026

Due: Tuesday, February 3, 2026 at 11:59 PM ET **Total Points:** 100
Late Due Date: Thursday, February 5, 2026 at 11:59 PM ET
Submission: Gradescope `https://www.gradescope.com/courses/1207241`
Starter Code: `https://github.com/KellyYutongHe/cmu-10799-diffusion/`

Introduction

Welcome to Homework 2! In HW1, you built a DDPM [Ho et al., 2020] model that learns to denoise images step by step. Now you'll **implement Flow Matching** [Lipman et al., 2023], a simpler and often more efficient framework that learns a velocity field to transport noise to data along straight paths.

In addition to flow matching, you will also **implement DDIM** [Song et al., 2020], a deterministic sampler that lets your existing DDPM model generate images in far fewer steps. Explore and compare flow matching and DDIM with DDPM sampling from HW1!

By the end of this homework, you'll have two working generative models (DDPM and Flow Matching) and will **choose your specialization track** for the rest of the course.

This homework is AI-friendly. Same rules as HW1. You may use any AI coding assistants, chatbots, or reference implementations. You may also use any other resources that you can find on the Internet. At the end of the homework, you'll document what resources you used.

Part 0: Setup your codebase (0 point)

You'll continue using the same codebase from HW1. Your DDPM implementation should still work and you'll need it for this homework. **What You'll Implement:**

File	**What to implement**
`src/methods/flow_matching.py`	Create your own flow matching implementation
`src/methods/ddpm.py`	Add DDIM sampling
`configs/`	Add configs for flow matching & DDIM
`scripts/`	Add scripts for flow matching & DDIM
`train.py`	Add flow matching training options
`sample.py`	Add flow matching & DDIM sampling options

Hint for Flow Matching:

Your `flow_matching.py` should follow the same structure as your `ddpm.py`:

- A `FlowMatching` class that inherits from `BaseMethod`
- `compute_loss(x_0, ...)` — returns training loss given a batch of real images
- `sample(batch_size, image_shape, ...)` — generates new images starting from noise

The core algorithm is different (and simpler!) but the interface should feel familiar.

Compute Budget:

You have **$500 in Modal credits** for the entire course (all 4 homework). Budget wisely!

Part I: Implement Flow Matching (25 points)

Now it's time to build your flow matching model! Take a good look at what you have built so far and start building again! Remember, you are free to add, delete and modify any and all parts of the starter code to fit your preference.

Q1. Building Intuition [optional, 0 pts]

You can still leverage the same strategies from HW1 to develop some intuitions of the algorithm with some toy experiments before diving into the full scale training.

(a) [0 pts] You can still use the Jupyter notebook `01_1d_playground.ipynb` to experiment with Flow Matching on 1D toy data. This notebook contains some options of mixture of Gaussians and their visualizations, you can try out your algorithm first in this playground.

(b) [0 pts] As with HW1, you can use the `--overfit-single-batch` flag to sanity check your implementation before full-scale training.

Q2. Implementation [25 pts]

Now let's implement flow matching! Use the same codebase you have from HW1 as the skeleton, and your job is to fill in the core algorithm. Remember: this is your codebase. Feel free to modify any part of the starter code to fit your preferences. Add helper functions, reorganize files, change the config structure... whatever helps. The only requirement is that your final code runs and produces good results.

(a) [5 pts] Train your flow matching model to convergence. You may use either the provided configs or your own. **You should use roughly the same model architecture, training iterations and batch size as your DDPM model.** Report:

1. Model size
2. Batch size and total training iterations
3. Training loss curve
4. Compute cost (GPU hours)
5. Number of sampling steps

(b) [10 pts] Generate a grid of 16 samples from your trained model. Include this grid below.

(c) [10 pts] Evaluate your model KID with 1k samples and report your KID score (both mean and std). You should be able to obtain KID < 0.005 with single L40S GPU training for a few hours.

(d) [0 pts] Provide a zip file of your code through Gradescope "Homework 2 Code" assignment. Make sure your code is runnable because we may run it to verify your results, and do not include any large files such as model checkpoints or dataset files. If you do not provide a zip file for your code, you will receive 0 point on Part I, Part III and Part IV of this homework.

Part III: Help Your "Classmate" Debug (15 points)

Kale is back! She's trying Flow Matching now but running into new issues, yet again. In each of the following scenario, you may be provided a loss curve, a grid of samples or a description of the situation. Try your best to help Kale with her implementation!

Q3. Pseudo Debugging Scenarios [15 pts]

(a) [5 pts] Kale trained both DDPM (from HW1) and flow matching on the same dataset with the same model architecture. She notices that her converged Flow Matching loss is significantly higher than her converged DDPM loss. Worried that something is wrong, she asks you: is this expected? Should she be concerned?

(b) [5 pts] Kale's training loss starts reasonable but occasionally spikes to very large values or NaN, especially early in training. Her DDPM implementation worked fine with the same hyperparameters and learning rate. What might be the cause and how to fix it? Identify the issue and provide two potential strategies to fix the bug.

(c) [5 pts] After the model fully converged, Kale tried to sample from her model and obtained the images below. She's using 200 Euler steps and her training loss looks normal. What are the debugging steps that you would recommend her to take?

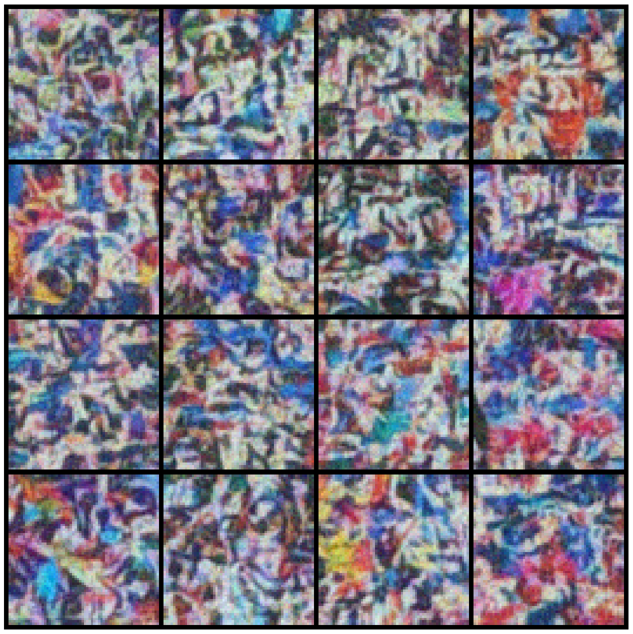
Figure 1: The samples that Kale obtained.

Part III: Implement DDIM (20 points)

So far you've trained two generative models: DDPM (HW1) and Flow Matching (this homework). You may have noticed that DDPM requires many more sampling steps (∼1000) compared to Flow Matching (∼100) to produce good results. Can we make DDPM faster without retraining?

DDIM (Denoising Diffusion Implicit Models) [Song et al., 2020] answers this question. The key insight is that DDPM's reverse process can be generalized to a family of non-Markovian processes that share the same training objective. By setting the stochasticity to zero, we get a deterministic ODE that can be solved with far fewer steps — using the exact same trained model.

The DDIM update rule is:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \underbrace{\left(\frac{x_t - \sqrt{1-\bar{\alpha}_t} \cdot \epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}} \right)}_{\hat{x}_0} + \sqrt{1-\bar{\alpha}_{t-1}} \cdot \epsilon_\theta(x_t, t) \tag{1}$$

Or in pseudocode:

Algorithm 1 DDIM Sampling

```
Require: trained noise predictor $\epsilon_\theta$, number of steps $S$, noise schedules $\bar{\alpha}$
 1: Sample $x_T \sim \mathcal{N}(0, I)$
 2: Create timestep subsequence $[\tau_S, \tau_{S-1}, ..., \tau_1]$ from $[T, ..., 1]$    ▷ e.g., $[1000, 900, 800, ...]$
 3: for $i = S, S-1, ..., 1$ do
 4:     $t \leftarrow \tau_i$
 5:     $t_{\text{prev}} \leftarrow \tau_{i-1}$ (or 0 if $i = 1$)
 6:     $\epsilon \leftarrow \epsilon_\theta(x_t, t)$    ▷ Predict noise using your trained DDPM
 7:     $\hat{x}_0 \leftarrow \frac{x_t - \sqrt{1-\bar{\alpha}_t} \cdot \epsilon}{\sqrt{\bar{\alpha}_t}}$    ▷ Predict clean image
 8:     $x_{t_{\text{prev}}} \leftarrow \sqrt{\bar{\alpha}_{t_{\text{prev}}}} \cdot \hat{x}_0 + \sqrt{1-\bar{\alpha}_{t_{\text{prev}}}} \cdot \epsilon$    ▷ DDIM step
 9: end for
10: return $x_0$
```

Q4. DDIM Implementation [20 pts]

Implement DDIM sampling for DDPM model that you trained for HW1. No retraining needed! DDIM works with your existing trained model!

(a) [10 pts] Generate a grid of 16 samples using 100 DDIM steps and include it below.

(b) [10 pts] Evaluate KID with 1k samples using 100 DDIM steps. Report your KID score (mean and std). How does this compare to your original DDPM with 1000 steps and your Flow Matching model?

Part IV: Ablation Study (25 points)

In this part of the homework, we will explore how the number of sampling steps affects sample quality for both Flow Matching and DDIM.

Q5. Sampling Steps Ablation [25 pts]

You now have three ways to generate images: DDPM (1000 steps from HW1), DDIM (fewer steps, same trained DDPM model), and Flow Matching (Euler integration). Let's see how they compare.

(a) [15 pts] Report KID scores (mean and std) for Flow Matching and DDIM at different step counts. Fill in the table below:

Steps	**Flow Matching KID**	**DDIM KID**
1		
5		
10		
50		
100		
200		
1000		

(b) [10 pts] Based on your results, compare flow matching, DDIM, and your original DDPM from HW1:

1. At what step count does each method start producing reasonable samples?
2. For each method, what is the minimum number of steps needed to achieve similar quality to your DDPM with 1000 steps from HW1?
3. How do you compare flow matching and DDIM with DDPM at 100 steps? What about at 1000 steps?
4. If you wanted the best sample quality regardless of speed, which method would you choose? If you wanted fast generation with acceptable quality, which would you choose?

Part V: Track Selection (10 points)

For the rest of the course, you'll specialize in one of three tracks. Your choice determines what you'll implement in HW3 and HW4.

The Three Tracks:

- **Fidelity**: Best image quality

  Your goal is to generate the highest quality images possible. This track is all about pushing the limits of sample quality through architecture improvements, better training recipes, advanced samplers, or any other techniques you can find. You'll measure success primarily through KID/-FID scores, but visual quality matters too. Example directions include: trying different model architectures (e.g., DiT, U-ViT), experimenting with training techniques (e.g., noise schedule tuning), or implementing advanced samplers.

  Evaluation Metric: KID/FID ↓

- **Controllability**: User control over generation

  Your goal is to control what the model generates. This is the most open-ended track: you have creative freedom to define what "control" means to you. Some ideas: conditioning on attributes (e.g., generate smiling faces, faces with glasses), text-to-image generation, image editing, interpolation in latent space, style transfer, or anything else you can imagine. Be creative! But remember: with great power comes great responsibility. Since this track is open-ended, you'll also need to define your own evaluation metrics that make sense for your chosen form of control.

  Evaluation Metric: You define it! (e.g., CLIP score, attribute classifier accuracy, user study, or custom metrics that fit your approach)

- **Speed**: Fast inference

  Your goal is to generate good images in as few sampling steps as possible. This track explores the tradeoff between speed and quality: can you match your baseline quality with 10x fewer steps?

100x fewer? Or even with single step sampling? You'll explore techniques like distillation, consistency models, better ODE solvers, or learned step-size schedules. Success is measured by how few steps you need to reach a target KID threshold.

Evaluation Metric: Number of steps to reach KID $<$ threshold (e.g., KID < 0.005)

Q6. Track Selection [10 pts]

(a) [2 pts] Which track do you choose?

(b) [3 pts] Why does this track interest you? What draws you to this particular challenge?

(c) [3 pts] What's one thing you're uncertain about or curious to explore in your chosen track?

(d) [2 pts] What resources (papers, repos, tutorials) have you found that might help with your chosen track? List at least 2.

Part VI: Reflection (5 points)

Q7. Reflection [5 pts]

(a) [2 pts] Looking back at HW1 and HW2, what's the most surprising or interesting thing you learned about diffusion models and flow matching? What intuition did you build that you didn't have before?

(b) [2 pts] Based on your experience with DDPM, DDIM, and Flow Matching: if you were starting a new generative modeling project today, which method would you choose as your starting point? Why?

(c) [1 pts] List all the resources you used for this homework: AI tools, open source code, tutorials, papers, classmates, etc.

That's it! You have completed HW 1! Congrats on your freshly baked diffusion models!

References

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=PqvMRDCJT9t`.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.