Instructor Notes: Tutorial & Homework Overview

Mengyan Jing

2026-02-18

Instructor Notes: Tutorial & Homework Overview

For: Mengyan Jing (your own reference)
Course: STAT 9100 — DDPM Lecture

The Big Picture — What You're Delivering

```
LECTURE DAY:
  1. Slides (index.qmd)          → You present live (~25 min)
  2. Tutorial (tutorial.py)       → Students follow along or run later
  3. Homework (hw_starter.py)     → Students do on their own after class

GRADING:
  4. Solution key (hw_solution_KEY.py) → You use to grade
```

How Everything Connects

Key design: The tutorial teaches the exact same functions that appear in the homework starter code. Students don't need to learn new code — they just need to apply what the tutorial showed them.

What the Tutorial Covers (tutorial.py)

Part	Topic	What Students Do	Time
1	Background	Read the key equation	2 min
2	Test image	See how make_test_image() works	2 min
3	Schedules	See linear_schedule() and cosine_schedule() code	5 min
4	Forward process	See q_sample() — the core 3-line function	3 min
5	Demo 1: Forward	Run it, see image getting noisy over time	3 min
6	Demo 2: Reverse	Run it, see structure emerging from noise	3 min

Part	Topic	What Students Do	Time
7	Demo 3: SNR	Run it, see why cosine beats linear	5 min
8	Visual comparison	Run it, see linear vs cosine side by side	3 min
9	Summary	Reference table of all functions	1 min
10	Big picture	See where forward process fits in full DDPM	2 min

Total: ~30 minutes if students run through it themselves.

How students run it: - Option A: Open in VS Code → cells run as notebook (# %% markers) - Option B: Open in JupyterLab → same cell markers work - Option C: Just run `python tutorial.py` (plots pop up sequentially)

No GPU needed. No training. No PyTorch. Just numpy + matplotlib.

What the Homework Covers (hw_questions.md + hw_starter.py)

Question 1: Implement sqrt schedule (10 pts)

What they do: - Write ~3 lines of code: `alpha_bar = 1 - sqrt(t/T)` - Generate an SNR comparison plot with all 3 schedules - Write 2-3 sentences comparing sqrt to linear and cosine

What it tests: - Can they implement a schedule given a formula? (coding) - Can they read and interpret SNR curves? (understanding)

Expected answer: - Sqrt falls between linear and cosine - Drops faster than cosine early on, but slower than linear late - More aggressive than cosine but less wasteful than linear

Common mistakes to watch for: - Forgetting to clip alpha_bar (can get negative values) - Using t from 0 to T-1 instead of 1 to T (off-by-one) - Returning None (forgetting to remove the `pass` statement)

Question 2: Visualize forward process (10 pts)

What they do: - Write a plotting loop (~15 lines) to create a 2-row figure - Top row: linear schedule at t = 0, 200, 400, 600, 800, 999 - Bottom row: sqrt schedule at same timesteps, same noise - Write 2-3 sentences about when each schedule destroys the image

What it tests: - Can they use q_sample() correctly? (coding) - Do they understand the visual effect of different schedules? (understanding) - Do they know how to set random seeds for fair comparison? (methodology)

Expected answer: - Linear: unrecognizable around t=400-500 - Sqrt: unrecognizable around t=500-600 - Linear wastes more timesteps on pure noise (last ~25%)

Common mistakes: - Not resetting the random seed for each row (unfair comparison) - Forgetting axis("off") or row labels (minor style issue)

Question 3: Effect of T (10 pts)

What they do: - Plot SNR curves for cosine schedule with T=50, 200, 1000 - X-axis must be t/T (fraction), not raw t - Write 2-3 sentences about whether the shape changes with T

What it tests: - Can they loop over different T values and normalize the x-axis? (coding) - Do they understand why scale-invariance matters for fast sampling? (insight)

Expected answer: - The three curves nearly perfectly overlap - This means cosine is "scale-invariant" — same shape regardless of T - This is why you can train with T=1000 but sample with T=100: the noise levels at the subsampled steps still cover the same range

This is the most insightful question. Good students will connect this to the Nichol & Dhariwal fast sampling result (100 steps instead of 4000). The shape preservation is what makes it possible.

Common mistakes: - Using raw t on x-axis instead of t/T (curves won't overlap) - Forgetting log scale on y-axis (curves look compressed)

Question 4: Written/conceptual (10 pts)

No coding. Tests understanding of lecture content.

(a) Why predict noise instead of clean image x0? (4 pts)

Expected key points (any 2 of these gets full marks): - noise always comes from N(0,I) — consistent target regardless of image/timestep - Given predicted noise, can algebraically recover original x0 - Equivalent to denoising score matching - Ho et al. found empirically that it gives much better samples

(b) Key idea of consistency models (3 pts)

Expected key points: - Learn map any noisy point directly to clean data - Self-consistency property: same trajectory to same output - Enables 1-step generation

(c) One advantage + one disadvantage vs DDPM (3 pts)

Expected: - Advantage: much faster sampling (1-2 steps vs 1000) - Disadvantage: slightly lower quality (FID 3.55 vs 3.17), or: best results need a pre-trained diffusion model (distillation)

Grading Rubric Summary

Question	Coding	Written	Total
Q1a: sqrt schedule code	5 pts	—	5
Q1b: SNR plot + interpretation	2 pts (plot)	3 pts (written)	5
Q2a: visual comparison code	5 pts	—	5
Q2b: interpretation	—	5 pts	5
Q3a: T comparison code	5 pts	—	5
Q3b: interpretation	—	5 pts	5
Q4a: why noise-prediction	—	4 pts	4
Q4b: consistency model idea	—	3 pts	3
Q4c: advantage/disadvantage	—	3 pts	3
TOTAL	**17**	**23**	**40**

Files Checklist

File	Who gets it	Purpose
`index.qmd`	You present	Lecture slides
`references.bib`	Bundled with slides	Bibliography
`_quarto.yml`	Bundled with slides	Quarto config
`Speech_Notes_Complete.md`	You only	Your talk script
`tutorial.py`	Students + prof	Standalone tutorial
`hw_questions.md`	Students	Question sheet
`hw_starter.py`	Students	Code to fill in
`hw_solution_KEY.py`	You only	Solution key for grading

What to Send Your Prof Now

Your prof said: *"let me see your tutorial first, then I can evaluate the homework."*

Send him: 1. `tutorial.py` — the standalone tutorial 2. Optionally: `hw_questions.md` + `hw_starter.py` (so he can see the full plan)

Don't send: - `hw_solution_KEY.py` (unless he asks for it)

If Prof Has Questions

"Where's the neural network / training code?" → "The tutorial focuses on the forward process, which is the mathematical foundation. Training a real DDPM requires GPUs and hours of compute, so instead I demonstrate the core concepts that students need to understand before they could train one. The homework tests whether they understand the math well enough to implement new schedules and reason about the training loss."

"Is this too easy / too hard?" → Q1-Q3 are coding + interpretation (medium). Q4 is conceptual (requires understanding the lecture, not just running code). The mix means students who just blindly code won't get full marks — they need to demonstrate understanding in the written parts.

"Why not train a real model?" → "Training even a small DDPM takes hours on a GPU. I designed the homework so students can run everything on a laptop in under a minute, while still engaging with the real math. The tutorial + homework cover the same forward process equation used in production systems like Stable Diffusion."