CS C280 Computer Vision
Spring 2026 Alyosha Efros, Angjoo Kanazawa

This homework is due on Tuesday, Mar 17, 2025, at 11:59PM.

Diffusion [1] and flow-based [3] models are powerful approaches to generative modeling, achieving state-of-the-art performance in numerous domains. In this homework, you will implement a flow matching model as introduced in lecture, and systematically explore its behavior—building from denoising networks to class-conditioned generation and idealized flow machines to better understand the principles underlying generative dynamics and creativity of such models.

You should first download the files here which contain starter code and instructions for the exercises below.

1 Flow Matching

Follow instructions in `flow_train.ipynb` notebook and fill in the code.

1.1 Denoising UNet

Tasks.

- **Implement the UNet.** Load MNIST (resize to 32×32, normalize to $[-1, 1]$) and implement the attention-based UNet components as shown by the different diagrams.
  - `ResBlock` (skip-connected convolutional block).
  - `AttnBlock` (self-attention refinement with residual connection).
  - Full `UNet` with down/up-sampling and skip connections.
- **Noise visualization.** For a clean digit x and $\epsilon \sim \mathcal{N}(0, I)$, form $z = x + \sigma\epsilon$ and visualize the noised images for
$$\sigma \in \{0.0,\ 0.2,\ 0.4,\ 0.5,\ 0.6,\ 0.8,\ 1.0\}.$$
- **Train one-step denoiser.** Train the UNet denoiser D_θ to map $z \mapsto x$ with $\sigma = 0.5$.
- **OOD noise levels.** Evaluate the trained denoiser on MNIST test digits under
$$\sigma \in \{0.0,\ 0.2,\ 0.4,\ 0.5,\ 0.6,\ 0.8,\ 1.0\}$$
and visualize results.
- **Denoising pure noise.** Retrain the network to denoise pure Gaussian noise inputs $\epsilon \sim \mathcal{N}(0, I)$ into digits (one-step). Compute the *average image* of the MNIST training set and compare it to the model outputs when denoising pure noise.

Deliverables.

- *Figure*. Grid of noised examples for $\sigma \in \{0.0, 0.2, 0.4, 0.5, 0.6, 0.8, 1.0\}$.

- *Plot*. Training loss curve (iterations) for the $\sigma = 0.5$ denoiser.
- *Visuals*. Show noisy inputs z and outputs $\hat{x}$ on MNIST test set after epoch 1 and 5.
- Visuals: denoiser outputs on MNIST test digits for the OOD noise sweep $\sigma \in \{0.0, 0.2, 0.4, 0.5, 0.6, 0.8, 1.0\}$.
- *Visuals*. Results of denoising pure noise after epochs 1 and 5.
- *Short written response*. What you notice comparing the MNIST average image and one-step denoising-from-pure-noise outputs, and *why*.

1.2 Time-Conditioned Velocity UNet

Tasks.

- **Implement the time-conditioned UNet.**
  - Implement the time-conditioned `ResBlock`.
  - Implement sinusoidal positional embeddings for scalar $t \in [0, 1]$ with log-spaced frequencies.
  - Implement the time-conditioned `UNet`.
- **Implement** `FlowMatching.loss`. Sample $x_1 \sim \mathcal{N}(0, I)$ and $t \sim U[0, 1]$, form

  $$x_t = (1-t)x_0 + tx_1, \quad v = x_1 - x_0,$$

  and minimize MSE between v and $v_\theta(x_t, t)$.
- **Implement** `FlowMatching.sample`. Initialize x_t as noise (or provided x_1), then iteratively update for T steps:

  $$x \leftarrow x - \tfrac{1}{T}\,\hat{v}, \quad \hat{v} = v_\theta(x, t),$$

  and return the final sample.
- **Train your velocity model.**

Deliverables.

- *Plot*. Training loss curve for flow matching.
- *Visuals*. Sampled generations after epochs 1, 5 and 10 (starting from noise).

1.3 Class-Conditioned UNet

Tasks.

- **Implement the class-conditioned UNet.**
  - Extend `ResBlock` to condition on both time and class.
  - Extend `UNet` to accept class conditioning.
- **Implement classifier-free guidance (CFG).**
  - For training, drop class conditioning with probability $p_{\text{uncond}} = 0.1$ by replacing with a null conditioning signal.

- For sampling and guidance scale γ,

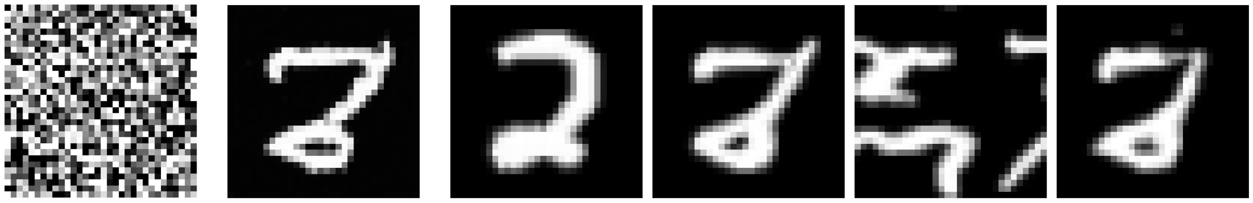
$$\hat{v} = v_\theta(x_t, t, \varnothing) + \gamma\big(v_\theta(x_t, t, c) - v_\theta(x_t, t, \varnothing)\big).$$

- **Train your class-conditioned velocity model.**

Deliverables.

- *Plot.* Training loss curve for class-conditioned flow matching.
- *Visuals.* Sampling results after epochs 1, 5 and 10 with with $\gamma = 5.0$. Produce 4 images per digit class (0–9) and display as a 10×4 grid.

2 Ideal Flow Machines



Read and understand the paper *An Analytic Theory of Creativity in Convolutional Diffusion Models*[1] [2]. Then, follow instructions in `ideal_flow.ipynb` notebook and fill in the code.

Tasks.

- **UNet.** Import your UNet velocity model with trained weights.
- **IS.** The Ideal Score (IS) machine only reproduces the training set. Every denoising step, it moves closer to a weighted average of the training images. It would mimic the behavior of the velocity model if it only were able to memorize the data distribution, which we know to not be true.
- **LS.** The Local Score (LS) machine reflects the locality bias of convolutions in the UNet. At each pixel, it compares co-located patches from the training set and moves the pixel toward a weighted average of their center values.
- **ELS.** The Equivariant Local Score (ELS) machine further recognizes the equivariant inductive bias from convolutions and attempts to model it with any other patch not neceesarly centered at the same pixel location.
- **bbELS.** The Boundary-Broken Equivariant Local Score (bbELS) machine accounts for the fact that patches near image borders contain location information. Border patches are compared only with training patches at similar relative positions to the boundary, while interior patches remain fully equivariant and are compared with all patches of the same shape.

[1] arXiv:2412.20292

Deliverables.

- *Visuals*. For a minimum of 5 starting noise images, show the initial noise and outputs for the UNet, IS, LS, ELS and bbELS machines. The starting noise must be the same for different methods.

Some initial noise samples may result in results than others. For the deliverables, try multiple options and pick examples where at least one machine's output closely resembles the UNet's. We provide `x_1.pt`, a starting noise tensor of shape `[1, 1, 32, 32]`, which should produce good results.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[2] Mason Kamb and Surya Ganguli. An analytic theory of creativity in convolutional diffusion models. *arXiv preprint arXiv:2412.20292*, 2024.

[3] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.