CS 4331/5331 – Generative AI
Homework 4

Department of Computer Science
Whitacre College of Engineering
Texas Tech University

Due by Monday, Nov. 17, 11:59pm via Canvas

You are encouraged to work in pairs. Each team member must understand the entire solution, and clearly indicate their individual contributions (who did what) at the top of the first page of the submitted assignment.

The use of generative AI to solve this assignment, especially for writing code, is strictly prohibited. Any violation will result in failing the course. Please feel free to reach out to the course staff with any questions. Gaining hands-on experience is an essential part of the learning process.

1 Denoising Diffusion Probabilistic Models (100 pts)

This rubric evaluates your assignment on Denoising Diffusion Probabilistic Models (DDPM). You are given (incomplete) template code for training the DDPM in `train.py`. In order for the code to run you need to make modifications in three places: `q_sample`, `p_sample`, and the calculation of loss in the main method. Each place you need to modify in `train.py` is marked. Once your model is trained it will automatically save the weights to a separate file. If you run the `test.py` script it will generate some images showing the denoising process on a distribution of data.

The default dataset that is used in the `train.py` and `test.py` file is very simple, its just the classic spiral dataset from `scikit-learn`. You must adapt the existing code to work with images (e.g. MNIST) and to use your 2D UNet implementation from the first assignment. Once you have successfully trained your DDPM on image data plot the **reverse sampling process** at a fixed number of timesteps. An example of the **forward sampling process** is given in the `forward_process.py` file. You will create a new file `reverse_process.py` which calls your trained model to denoise images and plot the intermediate steps.

1.1 Complete Training Code (30 points)

To-Do: Complete the missing parts in `train.py` marked as "To-Do". Take a screenshot of each change.

- `q_sample`: See Algorithm (1) and Equation (4) in [1].
- `p_sample`: See Algorithm (2) and Equation (7) in [1]
- `loss`: See Algorithm 1 in [1]

1.2 Train DDPM on Spiral Dataset (5 points)

To-Do: Run `train.py` and collect the output images and model. Ensure that your model trains successfully and the loss decreases.

- Compare two different beta schedules.
- Compare two different numbers of diffusion steps.

1.3 Evaluate trained model on Spiral Dataset (5 points)

To-Do: Run `test.py` and collect the output image.

1.4 Adapt Code to Train on Images (30 points)

To-Do: Adapt your DDPM to run on an image dataset with a UNet architecture.

- Start with your UNet implementation from the first assignment.
- Modify your UNet to accept an integer timestep as an input. The UNet should take in a noisy image x_t and an integer timestep t and output a noise prediction ϵ.

$$\epsilon = \text{UNet}_\theta(x_t, t) \tag{1}$$

- Train your DDPM with UNet on MNIST.
- For inspiration on how to add time conditioning see:
  `https://github.com/tristan-shah/ddpm/blob/main/ddpm/model.py`
- This video on UNet is also very helpful:
  `https://youtu.be/ZBKpAp_6TGI?si=-zVuIhuDWMBKCrld&t=7752`

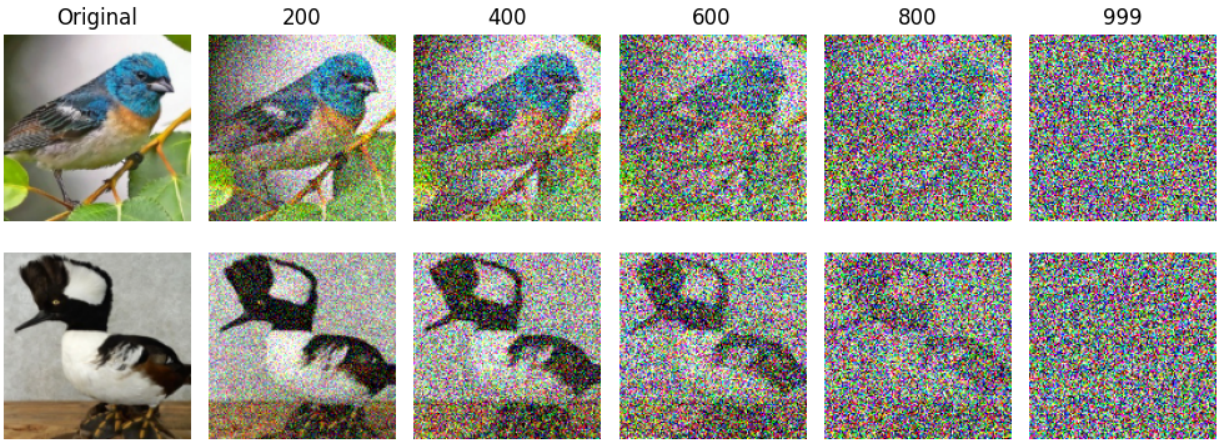

Figure 1: Your **forward process** should look something like the above figure. But for MNIST digits. The **reverse process** should go in the opposite direction starting from noise and ending up as generated digits.

1.5 Plot Forward and Reverse Process For Images (20 points)

To-Do: Create plots of the forward and reverse process on an individual image.

Forward Process:

- Select a real image from the MNIST dataset, corrupt it using the forward process until it is pure noise.
- Plot the image at intermediate noisy steps (e.g. T = 100, 200, ..., 500).
- See `forward_process.py` for a demonstration on how to do this.
- Compare two different beta schedules.
- Compare two different numbers of diffusion steps.
- Qualitatively, how do different schedules / diffusion steps affect how quickly the image is corrupted?

Reverse Process:

- Using your trained UNet DDPM demonstrate the reverse sampling process of a single image starting from pure noise.
- Plot images of the reverse sampling process similar to how the figure generated by `forward_process.py` does.

- Plot the image at intermediate noisy steps (e.g. T = 100, 200, ..., 500).
- You will need to re-train your model for this:
  - Compare two different beta schedules.
  - Compare two different numbers of diffusion steps.
- Qualitatively, how do different schedules /diffusion steps affect the rate at which the image is generated?

1.6 Short Write Up (10 points)

To-Do: Write a few sentences about what you learned in this assignment.

- What did changing beta schedules do?
- What did increasing / decreasing the number of time steps do?
- Any challenges?

Deliverables

Please only submit screenshots and plots (.png, .jpeg, etc...) no need to submit code.

- Screenshots of modifications to `train.py`.
- Loss plots from the spiral dataset for different beta schedules / diffusion steps.
- Screenshot of your modifications to UNet to condition on time.
- Loss plots from training UNET on MNIST for different beta schedules / diffusion steps.
- Forward process images for different beta schedules / diffusion steps.
- Reverse process images from different beta schedules / diffusion steps.

Files

1. `train.py` - Contains all code for implementing and training DDPM on the spiral dataset.
2. `test.py` - Contains code to test DDPM on spiral dataset.
3. `forward_process.py` - A demonstration of how to apply the forward process to an image.
4. `dog.jpeg` - An image of a dog.
5. `GenAI_diffusion_rubric.pdf` - Grading rubric.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: `2006.11239 [cs.LG]`. URL: `https://arxiv.org/abs/2006.11239`.