CS726: Programming Assignment 2

March 12, 2025

General Instructions

1. Plagiarism will be strictly penalized including but not limited to reporting to DADAC and zero in assignments. If you use tools such as ChatGPT, Copilot, you must explicitly acknowledge their usage in your report. If you use external sources (e.g., tutorials, papers, or open-source code), you must cite them properly in your report and comments in the code.
2. Submission: Submit a report explaining your approach, implementation details, and results. Clearly mention the contributions of each team member in the report. Submit your code and report as a compressed `<TeamName>_<student1rollno>_<student2rollno>_<student3rollno>.zip` file. Fill a student roll number as `NOPE` if less than 3 members.
3. Start well ahead of the deadline. Submissions up to two days late will be capped at 80% of the total marks, and no marks will be awarded beyond that.
4. A notebook `explore.ipynb` is included to help you understand the starter code and provided datasets. We have also provided a `utils.py` file to provide some convenient functions to plot, visualize and evaluate. You are encouraged to include plots and visualizations wherever possible.
5. Do not modify the environment provided. Any runtime errors during evaluations will result in zero marks. `README.md` provides instructions and tips to set up the environment and run the code.
6. For most of the assignment, you have to fill in your code in already existing files. Apart from report, do not submit any additional models and files unless explicity asked. Any additional files should be placed in the top-level directory.

1 Problem Statement

1.1 Denoising Diffusion Probabilistic Models

In this part, you will implement a "Denoising Diffusion Probabilistic Model" (DDPM)[2] on various datasets. It is strongly recommended that you go through the paper before you start the assignment since the implementation will use notation very similar to that of the paper.

1. Implement an unconditional DDPM[2] in `DDPM` class, assuming that your data is n-dimensional. Since the model will be unconditional, you can discard the label information in the dataset.
2. Study the effects of hyper parameters:
   (a) **Number of diffusion steps (T)**:*Are the gains marginal when increasing T?* – Train at least 5 DDPMs for T = 10, 50, 100, 150, 200 respectively, keeping all other hyperparameters fixed.
   (b) **Noise schedule**: *What settings in noise schedule works the best?* The authors in [2] use a linear noise which has two hyperparameters `lbeta` and `ubeta`. Study the effect over atleast 5 values and report. Extra-credit will be awarded if other noise schedules such cosine and sigmoid [1] are explored.

   `utils.py` provides metrics such as Earth Mover Distance (EMD) and (approximate)Negative Log-Likelihood (NLL) which you can use to evaluate the quality of your samples.

3. Using your best hyperparameter setting, train a model on `albatross` dataset. An additional file `data/albatross_prior_samples.npy` is also provided. This file contains 32561 vectors sampled from $\mathcal{N}(0, I_d)$. Use these vectors to initialise x_T at step 1 of the sampling algorithm and set $z = 0$ in step 4 (Check Algorithm 2 in [2]). Note that this makes the entire run of this algorithm deterministic. Submit the model and the generated samples(`albatross_samples.npy`). Include a script `reproduce.py` to reproduce these samples from this model. This script should load your model and dump the samples as `albatross_samples_reproduce.npy` in the top-level directory.

   Quality of this sampled data will be used to assign a component of marks.

1.2 Classifier-Free Guidance

In this part you'll implement "Classifier-Free Guidance" (CFG) [3]. Note that this will require you to train an conditional DDPM which also takes label information as input, so you'll need to modify your implementation in 1.1 accordingly. Implement this in a new class `ConditionalDDPM` in ddpm.py. Implement the training and sampling code in different functions `trainConditional` and `sampleConditional`.

1. Discuss the difference between guided sampling and conditional sampling.
2. Study the effect of guidance scale(atleast over 5 values) on quality of generated data. Discuss the metric you will use assess quality for samples from a particular class. Are the generated samples classified as the correct class? Note that you'll need to train a classifier to answer this question. Report the average results over all possible class labels, also report the accuracy of this classifier.
3. Design a training-free method to classify a given input using the conditional diffusion model trained for CFG. Include any derivations and motivation for your method in your report. Your method must be implemented within `ClassifierDDPM` class which accepts a `ConditionalDDPM` during initialization. Compare `ClassifierDDPM` with the classifier you trained on the data for answering 1.2.2.

   This classifier will be evaluated on hidden datasets.

1.3 Bonus: Reward Guidance

CFG requires a conditional model. Newer algorithms such as "Soft Value-Based Decoding" (SVDD) [4] allows you to achieve similar objective from an unconditional model. These algorithms generate samples from a pre-trained models which can achieve high "rewards".

1. Implement SVDD-PM algorithm in `sampleSVDD` function which can accept an arbitrary reward function. Discuss how the posterior mean approximation is achieved for DDPM.

   samplerSVDD function will be checked using hidden reward functions and datasets.
2. Discuss the reward function which is required to sample from the objective in CFG. Compare the results with the results obtained in 1.2. Hint: You might need to use the classifier you trained in 1.2.

Note that bonus marks will be considered only if the entire main part of the assignment is completed.

References

[1] Ting Chen. On the importance of noise scheduling for diffusion models. *ArXiv*, abs/2301.10972, 2023.

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

[3] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[4] Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gökcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, and Masatoshi Uehara. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding, 2025.