Diffusion Project

EE367/CS448I Computational Imaging

Due: see poster, report, and code submission deadlines on website

Learning Goals

In this project, you will learn the basics of working with diffusion models. For this purpose, you will first review a few basic mathematical concepts, then learn to denoise an image using the provided pre-trained diffusion model and then implement the DDPM sampling method [Ho et al., 2020] to unconditionally generate images using this diffusion model with an iterative denoising procedure. Then, you will use the pretrained diffusion model as a prior to solve two inverse problems, namely inpainting and deconvolution, using a few different methods. The most intuitive method is SDEdit [Meng et al., 2022]; the more rigorous methods are ScoreALD [Jalal et al., 2021] and DPS [Chung et al., 2023], which are representative of modern posterior sampling approaches for solving inverse problems.

Notes

This is a project, not a homework, so you need to work more independently, refer to the lecture slides, and, more importantly, to the relevant literature for technical details. All tasks outlined in this document and in the starter code are guidelines – you do not need to solve exactly these tasks and can add additional baselines or variations of these algorithms. But do not make your project smaller in scope than the provided set of questions.

Our support will be limited because you're supposed to do this yourself. Work directly with your project mentor if you run into trouble but do not post code on Ed Discussion. Feel free to use Ed to ask general questions that may be of interest to other students, such as questions related to Colab, Jupyter Notebooks, Python, PyTorch, diffusion models, etc.

The starter code is provided as is. We tested it and it worked for us, but it's up to you to debug it if something isn't working out of the box for you. Other projects do not get any starter code, so please do not take this for granted or ask us to walk you through every line of code.

The starter code includes a pre-trained diffusion model. We downloaded this model from the DPS paper [Chung et al., 2023] repository at https://github.com/DPS2022/diffusion-posterior-sampling. The model is trained on the Flickr-Faces-HQ Dataset (FFHQ) dataset (https://github.com/NVlabs/ffhq-dataset).

In this project and the starter code, we are using the variance-preserving formulation of diffusion models introduced in the DDPM paper [Ho et al., 2020]. In the DDPM paper, authors describe their diffusion models as noise predictors, i.e., given a noisy image x_t, the model predicts the noise ϵ. The pre-trained model we are providing does not predict the noise ϵ but the score $\nabla_{x_t} \log p_t(x_t)$. As discussed in class, these formulations are interchangeable, but please keep this in mind.

Instructions

This project consists of six tasks. The first is a bit more theoretical and the remaining five are programming tasks.

For the programming part of the project, we recommend that you use Colab. Please refer to the README.md file provided as part of the starter code for detailed instructions on how to use Colab with your Stanford Google Drive account. You need to upload the starter code to your Stanford Drive and then fill in the required parts of the provided starter code to complete tasks 2–6.

How to Submit

For the **Final Project Poster session**, we would like you to prepare a poster with your results from tasks 2–6. Include relevant paper references, add a motivation, and include qualitative and quantitative results. Be able to explain what you did and any insights and findings. Please do **not** include task 1 in the poster!

You should structure your **Final Report** in the way discussed in class and on the course website. You need to write an abstract to summarize your work, an introduction to motivate this project, a related work section that discusses related work on diffusion models and solving inverse problems with diffusion models, a methods section that details both unconditional sampling with diffusion models and also posterior sampling with diffusion models for inverse problems in different subsection. Include your results in a separate "Experiments" section with both qualitative and quantitative results. Finally, include a discussion section. **Write the final report as if you came up with all the tasks and implemented them all, rather than treating it as another homework and discussing them task by task!**

To submit your **Code**, export your Jupyter notebook page into a pdf file. You can click "file" > "print" > "save as pdf". The exported pdf file will contain all the code as well as the outputs. Submit this pdf separately from your report as your code submission.

Submit your poster, code, and report to Gradescope.

Task 1 of 6: Some Derivations

Let's get started with a few derivations that will help you refine your mathematical background knowledge on concepts related to diffusion models.

Moreover, there are many important papers in this space and, unfortunately, not all of them use the same notations or formulations. Yet, they all discuss the same concepts. Part of this exercise is for you to understand some of these connections.

1. In Lecture 13 and in this project, we only consider the variance-preserving (VP) formulation of diffusion models introduced by the DDPM paper [Ho et al., 2020] rather than the variance-exploding formulation. In this VP formulation, the (forward) noise model from step $t-1$ to step t is given as

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathbf{z}_{t-1}, \quad t = 1, 2, \dots, T, \tag{1}$$

   where $\mathbf{z}_{t-1} \sim \mathcal{N}(0, I)$ are i.i.d. Gaussian random variables in each step and β_t is the noise schedule. In the lecture, we showed you that this forward Markov chain can be written as a conditional distribution that does

not depend on step $t-1$ but only on the very first step $t=0$:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\mathbf{z}, \tag{2}$$

where $\alpha_t = 1-\beta_t$, $\bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i$, and $\mathbf{z} \sim \mathcal{N}(0, I)$.

Your first task is to prove that this is indeed true – given equation 1, show how to derive equation 2 from it using the given definitions.

Hint: Consider how the variance terms accumulate through the chain of additions.

2. Refer back to Lecture 13, particularly the slide titled "DDPM vs. DPS". Here, we discussed the two equivalent forms of the DDPM reverse diffusion process. Prove that these two forms are equivalent. Specifically, prove that:

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t + (1-\bar{\alpha}_t)\mathbf{s}_\theta(\mathbf{x}_t, t)) \qquad \xrightarrow{\textit{is equivalent to}} \qquad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t + (1-\alpha_t)\mathbf{s}_\theta(\mathbf{x}_t, t))$$
$$\mathbf{x}_{t-1} = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\hat{\mathbf{x}}_0$$

Hint: Use the definition of $\bar{\alpha}_t$ to simplify your expressions. From its definition, we know $\bar{\alpha}_t = \alpha_t\bar{\alpha}_{t-1}$.

3. In the DDPM paper [Ho et al., 2020], particularly in Algorithm 2, the noise-predicting network $\boldsymbol{\epsilon}_\theta$ is used instead of a score-predicting network $\mathbf{s}_\theta$. Prove that the above one-step reverse diffusion step is equivalent to the formula in Algorithm 2. Specifically, prove that:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t + (1-\alpha_t)\mathbf{s}_\theta(\mathbf{x}_t, t)) \qquad \Leftrightarrow \qquad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)) \tag{3}$$

Hint: Refer to the cheat sheet for relevant formulas. In particular, use the VP forward diffusion process and Tweedie's formula.

Task 2 of 6: Image Denoising using a Pretrained Diffusion Model

In this task, you will implement single-step image denoising using a pretrained diffusion model. For this purpose, you should load a test image, add noise using the appropriate noise model, and then denoise it using the pretrained diffusion model. You'll learn how diffusion models can reconstruct clean images from noisy versions by acting as Gaussian denoisers.

Your tasks:

1. Understand how the forward (noising) process works and implement it
2. Implement the reverse (denoising) process using a single-step model prediction
3. Visualize and analyze the results at different noise levels

You can find starter code in the Jupyter notebook `task1.ipynb`. Fill in the missing parts of the code as indicated. Include qualitative results, i.e., the target image, the noisy image, and the denoised image, as well as quantitative results, i.e., PSNR and LPIPS numbers in your submission. Feel free to include additional test images, beyond the one listed in the starter code.

What to submit: Please include the target image, noisy images, and denoised images for at least three different noise levels in your submission. In addition, please report PSNR and LPIPS for each of the denoised images. Optionally, do this for multiple different test images.

Task 3 of 6: Image Generation using a Pretrained Diffusion Model

In this task, you will implement an unconditional image generation procedure using a pretrained diffusion model. For this purpose, you initialize your image as pure Gaussian noise and then iteratively denoise it using the DDPM procedure [Ho et al., 2020]. Implement this procedure using the provided parameter schedule in 1,000 steps. You'll learn how diffusion models can generate new images from pure noise through an iterative denoising process.

Your tasks:

1. Understand the reverse sampling process in diffusion models
2. Implement iterative denoising for image generation
3. Generate and visualize new images from random noise
4. Qualitatively evaluate the generated image results

You can find starter code in the Jupyter notebook `task2.ipynb`. Fill in the missing parts of the code as indicated and generate several images.

Note that image reconstruction metrics like PSNR and LPIPS do not apply to generated images because there is no reference image. The typical metric for quantitatively evaluating generated images is FID, but this is slightly beyond the scope of this exercise. Feel free to optionally implement this, but we are not asking you to quantify the quality of your results using FID.

What to submit: Please include at least three different unconditionally generated images in your submission.

Task 4 of 6: Solving Inverse Problems with SDEdit

In this task, you will implement SDEdit (Score-Distillation Editing) [Meng et al., 2022] using a pretrained diffusion model. SDEdit is an approach that can be used to solve inverse problems. You will test this algorithm on two inverse problems: image inpainting and deconvolution. SDEdit starts from a partially noised version of the observations/measurements (i.e., the masked input image or the blurry image) and we will apply the iterative DDPM denoising procedure to denoise it. You'll learn how to use this approach to implement a simple conditional generation procedure, i.e., it is conditioned on the measurements.

Your tasks:

1. Understand the partial noising process specific to SDEdit
2. Implement the modified sampling loop with some starting point for time
3. Apply controlled reverse diffusion from the partially noised state
4. Analyze the effect of the noise level parameter (default is time=500)

You can find starter code in the Jupyter notebook `task3.ipynb`. Fill in the missing parts of the code as indicated. Include qualitative results, i.e., the target images, the masked/blurry images, the noisy masked/blurry images, and

the inpainted/deconvolved images, as well as quantitative results, i.e., PSNR and LPIPS numbers in your submission. Feel free to include additional test images, beyond the one listed in the starter code.

What to submit: For each of the inpainting and deconvolution problems, include the masked/blurry measurements as well as reconstructed images for at least three different noise level parameters of SDEdit. So this is a total of 6 reconstructed images for the two problems. Report PSNR and LPIPS for each of these 6 reconstructions. Comment on how the choice of noise level parameter affects the reconstruction. Optionally, run this for multiple different target images.

Task 5 of 6: Solving Inverse Problems with ScoreALD

In this task, you will implement the ScoreALD algorithm [Jalal et al., 2021] using a pretrained diffusion model. ScoreALD is an approach that can be used to solve inverse problems. You will test this algorithm on two inverse problems: image inpainting and deconvolution. ScoreALD is a posterior sampling approach, so it will compute a different sample of the feasible set of solutions of the inverse problem every time you run it. ScoreALD incorporates measurement information during diffusion sampling through likelihood gradients and annealed dynamics.

Your tasks:

1. Implement likelihood gradient computation for different measurement operators (inpainting masks, blur kernels)
2. Implement ScoreALD conditioning by combining score function with likelihood gradients using proper scaling
3. Modify sampling loop to incorporate ScoreALD updates at each timestep
4. Test and analyze results with:
    - Different annealing factors
    - Different inverse problems (inpainting, deconvolution)

You can find starter code in the Jupyter notebook `task4.ipynb`. Fill in the missing parts of the code as indicated. Include qualitative results, i.e., the target images, the masked/blurry images, the noisy masked/blurry images, and the inpainted/deconvolved images, as well as quantitative results, i.e., PSNR and LPIPS numbers in your submission. Feel free to include additional test images, beyond the one listed in the starter code.

What to submit: For each of the inpainting and deconvolution problems, include the masked/blurry measurements as well as the reconstructions. Report PSNR and LPIPS for each of the reconstructions. Optionally test different mask patterns for the inpainting task and/or different blur levels for the deblurring task.

Task 6 of 6: Solving Inverse Problems with DPS

In this task, you will implement the DPS (Diffusion Posterior Sampling) algorithm [Chung et al., 2023] using a pretrained diffusion model. DPS is an approach that can be used to solve inverse problems. You will test this algorithm on two inverse problems: image inpainting and deconvolution. Similar to ScoreALD, DPS leverages the posterior sampling interpretation of the diffusion model to solve inverse problems. It normalizes likelihood gradients during the reverse diffusion process to ensure stable sampling. Although in this assignment we use DPS to solve linear inverse problems, DPS can handle more general inverse problems, including nonlinear problems, whereas Score

ALD cannot.

Your tasks:

1. Implement DPS conditioning with normalized gradient updates
2. Modify the sampling loop to incorporate DPS steps
3. Modify sampling loop to incorporate ScoreALD updates at each timestep
4. Test the method with:
    - Different scale parameters (0.1–1.0 range)
    - Different inverse problems (inpainting, deconvolution)
    - Compare results with those obtained by ScoreALD

You can find starter code in the Jupyter notebook `task5.ipynb`. Fill in the missing parts of the code as indicated. Include qualitative results, i.e., the target images, the masked/blurry images, the noisy masked/blurry images, and the inpainted/deconvolved images, as well as quantitative results, i.e., PSNR and LPIPS numbers in your submission. Feel free to include additional test images, beyond the one listed in the starter code.

What to submit: For each of the inpainting and deconvolution problems, include the masked/blurry measurements as well as the reconstructions. Report PSNR and LPIPS for each of the reconstructions. Comment on how the results of ScoreALD and DPS compare for the same measurements. Optionally test different mask patterns for the inpainting task and/or different blur levels for the deblurring task.

Questions?

First, review the lecture slides and reference because the answer to your question is likely in there. If you don't find it there, work directly with your project mentor, or email the course staff mailing list (in that order).

References

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. (2023). Diffusion posterior sampling for general noisy inverse problems. In *ICLR*.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In *NeurIPS*.

Jalal, A., Arvinte, M., Daras, G., Price, E., Dimakis, A. G., and Tamir, J. (2021). Robust compressed sensing mri with deep generative priors.

Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.-Y., and Ermon, S. (2022). Sdedit: Guided image synthesis and editing with stochastic differential equations.