CS 449 Final Project Report

Shreya Sridhar, Siddharth Saha

1. Goals and Discussion (8 points)

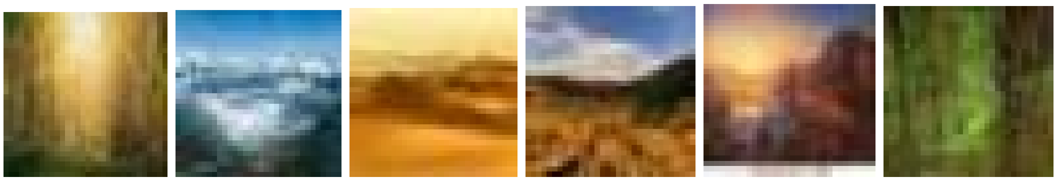
List all goals that you proposed either in your revised proposal or project update. For each, (a) include one or two sentences on how you completed it, what progress you made, or why you abandoned it, and (b) include one to two sentences on what was the most interesting or difficult part.

1.1 Essential Goals:

- *Goal 1: We want to train a diffusion model unconditionally (without information about image classes) with the images from the Kaggle Landscape Recognition dataset without their labels. (https://www.kaggle.com/datasets/utkarshsaxenadn/landscape-recognition-image-dataset-12k-images)*

We completed this goal by starting with an existing GitHub repository [2] that had already implemented an unconditional diffusion model. We used the same denoising and noising framework provided in [2] as we were not focused on implementing the mathematical concepts of diffusion. We then modified the code to use our dataset, and we trained on a virtual machine. The most difficult part of this goal was figuring out how to train the model, since training diffusion models is computationally expensive and takes a long time (see "Everything Else" section).

Below are some of the images generated for this task:



These images can be further explored in FinalResults/Unconditional.

- *Goal 2: We will use an existing library for computing FID (Frechet Inception Distance) scores to assess the performance of our model.*

We used a pre-existing FID implementation [5]. There were initially two options: the FID implementation from the torchmetrics module, or Pytorch's own FID implementation. The Pytorch implementation, however, required file paths to the dataset as an input. This would not suit our needs since the images required pre-processing that occurred in the notebook, which would then not be visible to that implementation. Thus, we elected to use the torchmetrics implementation.

After giving the FID instance 150 images from the testing set, and 150 images from the generated set, we got an FID score of 3.44 on the unconditional mode, and 1.45 on the conditional (see FID.ipynb). These are very exciting results, as it indicates that the diffusion models are able to generate very convincing 32x32 images. At a higher resolution, our model would likely not be able to attain such a good score (in FID, lower is better). The low FID scores make sense- it is especially hard to distinguish generated and real photos at a lower resolution (32x32). The scores also demonstrated to us that diffusion models tend to do much better when they have more guidance (i.e. the class) when they're generating images.

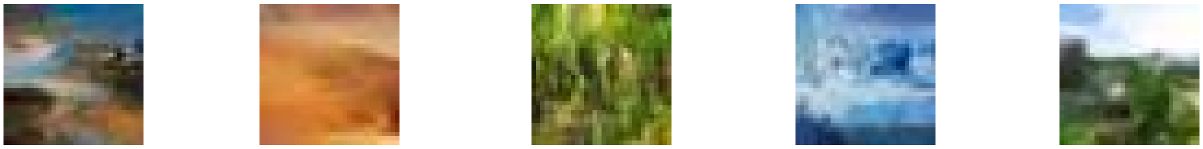
Model used	**Training Dataset**	**FID score**
Unconditional model	4000 images, undistinguished by class	3.44
Conditional model	4000 images, 800 per class	1.45

1.2 Desired Goals:

- *Goal 1: We will next train our diffusion model conditionally on a dataset of images labeled with landscape classes (specifically, coast, desert, glacier, forest, mountain).*

To accomplish this goal, we started with a conditional diffusion model from [2]. We used the same denoising and noising framework provided in [2] as we were not focused on implementing the mathematical concepts of diffusion. We chose our own parameters for training and wrote functions for loading the model and sampling the generated images. Our function for inference (generate_n) was written such that the final result was compatible with our GAN discriminator (further details below).

Below are some of the generated images for each class (Coast, Desert, Forest, Glacier, Mountain):



These images can be further explored in FinalResults/Conditional.

- *Goal 2: We will train a conditional GAN-style discriminator to distinguish the different classes of landscape images in our training data with (real pictures). We will then produce images using our conditional diffusion model and use our trained discriminator to identify the class of the generated pictures. We can then use this as a metric to gauge performance- the more of our images classified as the class they were supposed to belong to, the better.*

Initially, the purpose of this goal was to use a classifier to distinguish real and fake generated images. However, this goal seemed to be testing the performance of the discriminator itself rather than the diffusion model. Thus, we updated this goal to instead classify our generated images into their respective landscape classes. To build our discriminator, we used [4] as a guide for how to preprocess the images, and also for the identification of key hyperparameters that would likely have the most impact on the performance.

We tried to play around with hyperparameters of the model, to prevent overfitting, but ultimately were not able to increase testing performance beyond 67.6%.

Below are the results for this classifier: (from GANClassifier.ipynb)

Model Used	**Dataset**	**Accuracy**
SVC (baseline)	Training (4000 images, 800 per class)	81.8%
	Testing (500 images, 100 per class)	66%
SVC (Regularization = 10)	Training Dataset	99%
	Testing Dataset	66%
SVC (Regularization = 100)	Training	99.95%
	Testing	67%
SVC (Regularization = 100, gamma = 0.001)	Training	99.15%
	Testing	65%
SVC (Regularization = 100, gamma = 0.01)	Training	99.95%
	Testing	67.6%
SVC (Regularization = 100, gamma = 0.1)	Training	99.99%
	Testing	36%
SVC (Regularization = 100, gamma = 0.01)	***Generated Images with optimal hyperparameters (only tested this on best performing model from above)***	***83.9%***
SVC (Regularization = 100, gamma = 0.01)	***Generated Images with modified hyperparameters (only tested this on best performing model from above)***	***24.9%***

Our best performing classifier gave a 99% training accuracy, and 67.6% testing accuracy (on a dataset of real landscape images). When using our generated images, the GAN gave an 83.9% accuracy. It was surprising to us that the model generalized better to the generated images than the real landscape images, but our theory is that the diffusion model produced images that had the most common features of that class, whereas real landscape images could have unique features that confused the model. Additionally, we generated images with modified hyperparameters and tested those with the GAN; more details are included in the following “stretch goals” section.

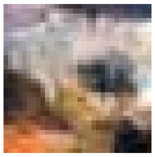
We also did further analysis on the performance of the classifier to determine if it performed better for some classes over others. However, we were not able to find any significant differences in the classifier's performance based on image class.

1.3 Stretch Goals:

- *Goal 1: We will try to find the optimal values for the following hyperparameters:*
  - *noise schedule hyperparameters (for example: max noise, linear v. more complex function, number of time steps in noise schedule)*
  - *activation functions used (eg. Gelu v. Relu)*

We first assessed how the variance schedule for a diffusion model affects the images produced, as explained in the " Denoising Diffusion Probabilistic Models" paper [3]. We first found that the linear noise schedule appears to be better suited to diffusion and decided to leave it unchanged. We next modified the number of timesteps. We had used a value of 1000 timesteps as our baseline (the same as [3]), and next modified it to a value of 500. This is because the images for 500 steps looked noticeably different in the DDPM paper [3], and we wanted to assess the difference in quality.

Below are some of the generated images for each class (Coast, Desert, Forest, Glacier, Mountain). The performance of these images with our GAN discriminator and FID are detailed in the next goal.



These images can be further explored in FinalResults/HyperparamImg.

We were not able to experiment further with different values for activation functions as we had mentioned in our goal, given the model's large training time. However, our consequent stretch goal analyzes the performance of the model with a modified noise schedule.

- *Goal 2: We will assess the change in performance with our new hyperparameters using the FID and discriminator results mentioned above.*

Our changes resulted in an average FID score of 0.6136 but an accuracy of only 24.9% from our GAN discriminator.

Indeed, our result images do not represent their classes well, hence the low value from the GAN. Our FID score is characteristically low, as we have found for our low resolution picture (see Essential Goal 2). It appears that because of the changes we made to the noise schedule, the image had more variance and thus more accurately represented the distribution of the training images. However, as a result of this variance, the model had a tougher time staying within the confines of the class it was trying to generate the image for. Hence, 1000 diffusion timesteps with a linear schedule produced the best results in the experiments we ran.

1.4 Everything Else:

In our initial setup for our first essential goal (unconditional training), it took us quite a while to figure out how to set up a virtual machine that would continue training the model even if our computers had gone to sleep overnight.

Another difficult part of that was determining how many images to use in training. The landscape dataset originally had 12000 images, but if we used too many images, our GPU would run out of memory in training. Thus, we had to experiment several different times, trying to find a balance between this memory constraint and having enough images that our model would actually learn something. We finally settled on using 4000 images (800 from each class). We also had to downsize our images from 64x64to 32x32. The difficulty of this was coupled by the fact that Google Drive has trouble handling that many images, so a bulk of our time was spent just uploading images to Google Drive.

2. Code and Documentation (4 points)

While you should not include code in this report, all your code should be uploaded to your GitHub repository. In this part of your report, you should list all .py or .ipynb files that contain your code. If you wrote more than ten such files, just **list the ten most important files. For each file you list here, include a one- or two-sentence description of what that file contains and why it's relevant to your project.**

1. run_model.py
    a. Contains the script to load in all dependencies, import models, and begin training either the conditional or unconditional model.
2. ddpm.py
    a. Contains the noising and denoising functions from [2] for the unconditional model. Also contains our code for saving and loading the model, and doing inference after the diffusion model has trained.
    b. Note: the code for the *noising and denoising* process of Diffusion is entirely from [2]. This is because our goal was not to start from the mathematical concepts of Diffusion from scratch, but to implement both a conditionally and unconditionally trained Diffusion Model. We focused on comparing the results from the 2 models and observing the results of changing hyperparameters for the noise schedule and UNet architecture.
3. ddpm_conditional.py
    a. Contains the noising and denoising functions from [2] for the conditional model.
    b. Contains our code for saving and loading the model, and doing inference after the Diffusion model has trained.
    c. Contains our function for saving images for each class in a separate folder (for our GAN's evaluation).
4. FID.ipynb
    a. Contains the code that calculates the FID scores of images generated from both our conditional and unconditional diffusion models. This gives us an industry standard metric with which to judge the performance of our models.

5. GANClassifier.ipynb
    a. Contains the code that trains and tests the classifier model from our 2nd desired goal, as well as further analysis on why we might be getting those particular results. It gives us another metric, besides FID score, with which to judge the performance of our conditional diffusion model.
6. utils
   A file from [2] with functions that:
    a. Save images produced by the diffusion model
    b. Normalize input images from the training and testing dataset
7. modules
    a. Includes the UNet architecture from [2].
8. download_images.ipynb
    a. Unzipping our dataset into the Jupyter environment
    b. Deleting directories once our model is done, to save credits for computation
    c. Zipping up our results from the model

Folders with our result images have additionally been uploaded to GitHub.

3. Reflections (4 points)

This section is your chance to reflect on this project. You should write at least a few sentences for each of these questions. However, try to be concise; a longer answer is not necessarily a better answer. If you want to write several paragraphs about something you're excited about, that's great! On the other hand, don't just write several paragraphs listing all the hyperparameter values you tried; if we fall asleep reading your answers, you might lose points.

3.1 What was interesting?

What did you learn from this project that wasn't covered by other parts of the class? What concepts from the lectures and readings were most relevant to your project?

We learned:

1. how to work with pre-existing repositories on GitHub, how to read other people's code and understand what it's doing even with just a quick skim,
2. how to modify complex models to achieve our end goal (producing landscape images, comparing a conditional model with an unconditional model).

Both of the above are especially important skills, especially since deciphering large codebases is going to be a frequent concern when we start working. We are very glad that we could use this project as an opportunity to hone these skills.

In terms of relevant readings, Reading 4 (Diffusion models beats GANs on image synthesis) was extremely useful. Reading that paper gave us a good foundation on what diffusion models are / how they work. We were then able to apply that in determining how to set our goals (for instance, what are the various hyperparameters we might play around with?).

3.2 What was difficult?

What were the hardest or most frustrating parts of this project? If someone were about to start on a similar project from scratch, what would you encourage them to do differently?

The most challenging parts of this project were:

1. Figuring out how to handle the data:
   - It was difficult to gauge how many images we should train our diffusion model on. Our original dataset had 12000 images which was impractically large given our constraints. We finally found the optimal number to be 4000 with trial and error.
   - Choosing the right image resolution was crucial. Our model was unable to train with a 64x64 resolution, which demanded too much memory. We kept trying with different dataset sizes, to no avail. We finally had to compromise and produce 32x32 pixel images- while these were not nearly as defined, our model could finally train with a reasonable speed.
2. Figuring out how to train the model systematically and manage credits:
   - Our model would train for around 5 hours overnight, so we had to ensure our setup had no flaws before beginning. Testing on lower epochs first and then starting the model was helpful. Saving and loading models correctly helped streamline the process a lot as well.
   - Leaving data on the virtual machine after we were done training proved to be very expensive, draining away our credits. As such, we would store away the saved model, training data, and results every time we trained and then delete them from the virtual environment. Zipping and unzipping files came very handy for this- we found it far easier than trying with *scp*.
   - When handling image datasets on Google Drive and tweaking the number of images to train on, uploading a fresh dataset might be wiser than cutting down a large dataset. We tried to cut a 12000 image dataset to one with 4000 and this proved to be very involved (deletion was much slower than we expected). Uploading a fresh dataset every time was far more efficient, as we could work on other things while the data was being uploaded.

3.3 What's left to do?

If you were going to spend another month working full-time on this project, what would you try to accomplish? Why? If Northwestern gave your group 1,000,000 USD to use for data collection or compute resources, what would spend it on? Why?

There are several key things we would like to do if we were given more time and/or resources:

1. We would train on higher resolution images and possibly use more images for training. We were able to achieve a high FID score for our 32x32 pixel images, as generated images can look very similar to real ones at a low resolution. The generation task becomes more challenging and worthwhile at even higher resolutions.
2. The money would help with training for more high-resolution images, but beyond a point, we'd like to shift our focus to the architecture of our model. We happened to choose a model that utilized a UNet architecture, which is particularly well-suited to image classification tasks in the medical setting. The UNet architecture is useful in identifying localized features in photos – for

example, detecting cancerous moles. In our case, it helped in building a more lightweight model that can train on fewer images. We would like to look into making changes to the architecture that are particularly useful for generating landscape images. Landscape photos have some distinctive features that we might suit our model towards- often being zoomed out and looking at a distant feature, being very color dependent (desert is yellow, glacier is blue), and the subject being diffused throughout the photograph (eg: trees).

Citations

[1]https://github.com/dome272/Diffusion-Models-pytorch
[2]https://github.com/tcapelle/Diffusion-Models-pytorch
[3]Jonathan Ho, Ajay Jain, Pieter Abbeel: “Denoising Diffusion Probabilistic Models”, 2020; [http://arxiv.org/abs/2006.11239 arXiv:2006.11239].
[4] [Computer Vision Engineer]. (2022, November 21). *Image classification with Python and Scikit learn | Computer vision tutorial* [Video]. Youtube. https://www.youtube.com/watch?v=il8dMDlXrIE
[5] *FRECHET INCEPTION DISTANCE (FID)*. TorchMetrics. https://torchmetrics.readthedocs.io/en/stable/image/frechet_inception_distance.html#frechet-inception-distance-fid