CS 449 Final Project Report

Due: Fri Dec 8, 2023 11:59pm

1. Names and Net IDs

Ziwei Su, zsj3847; Wenhao Gu, wgb3242

2. Goals and Discussion

2.1 Essential Goals

> - We will train and test a diffusion model described in [1] on our CIFAR-10 data. (Complete)

> First, this project aims to recover the distribution of the original CIFAR-10 dataset for image synthesis with a diffusion probabilistic model. The model termed the Denoising Diffusion Probabilistic Models (DDPM) [1] has an encoder-decoder architecture. The encoder is a diffusion process that gradually adds Gaussian noise to the data according to a variance schedule. The decoder is the backward transition of the diffusion process represented by a neural network. The architecture of the neural network is the Unet [2]. This project first uses a simple Unet for the backward diffusion. The simple Unet is an architecture with only convolutional and linear layers, and trained by minimizing a modified variational lower bound that amounts to a MSE loss as suggested in [1].
>
> The first step is to train a simple DDPM with the simple Unet with the simpler MNIST dataset. One NVIDIA GeForce GTX 1080Ti and one NVIDIA GeForce GTX 3080Ti GPUs are used for model training. We got a lot of help from this github repository https://github.com/MarceloGennari/diffusion_mnist/
>
> It's interesting that DDPM models are easy to train compare to GAN, in the sense that the train process is robust to learning rate. Specifically, we found that choosing learning rate from 1e-4 to 1e-3 always guarantee the convergence. In contrast, in the GAN model in HW3, the training process is very sensitive to the learning rates.

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

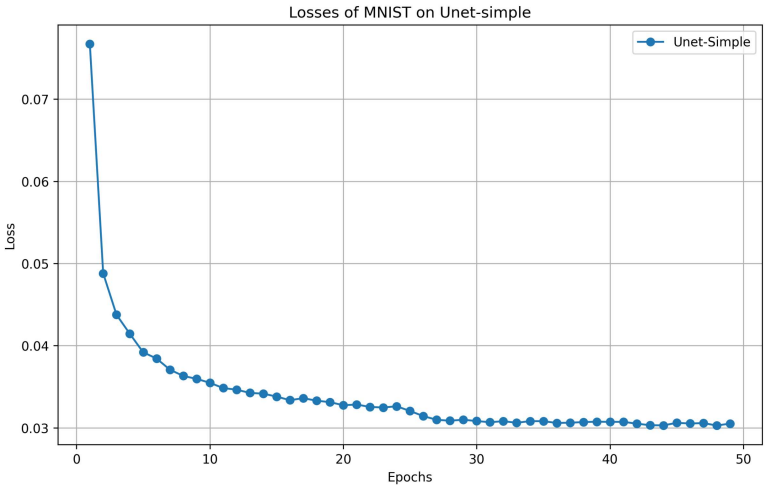
[2] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. Springer International Publishing, 2015.

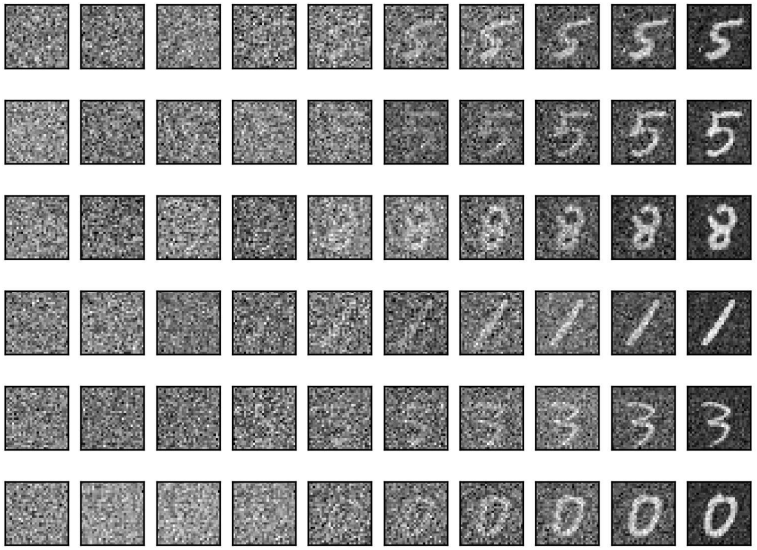
- We will visualize the results with the images in different time steps in the training process. (Complete)

MNIST

Training parameters are: batch_size = 128, learning rate = 2e-4, epochs = 50. The hidden dimensions of the Unet is [64, 128, 128, 64, 32].

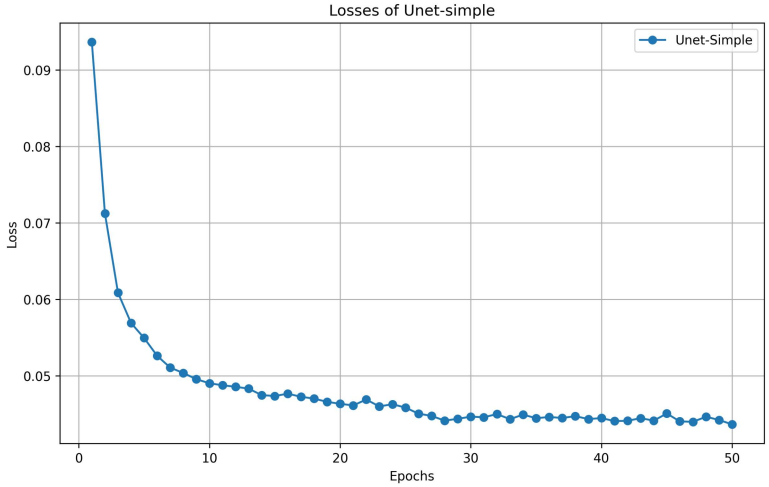
The loss versus epoch plot and the images in different timesteps are given as follows. The training of the simple DDPM on the MNIST dataset converges, and the model generates nice and recognizable digit images.

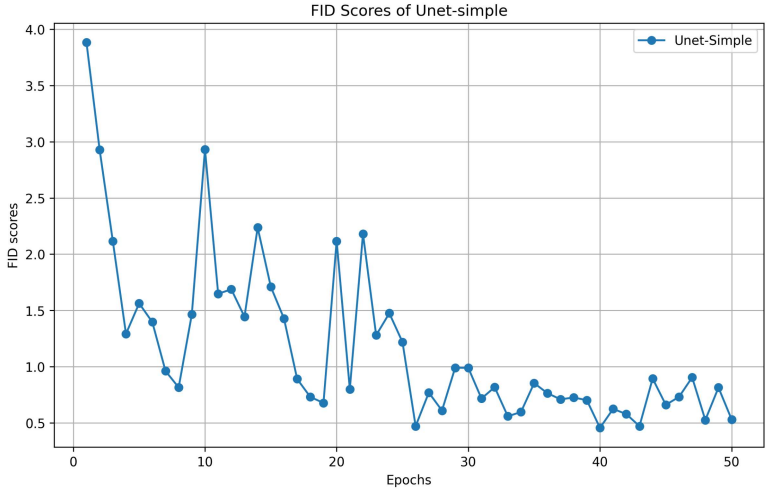




CIFAR-10

> After success on training the simple DDPM model on MNIST, this project trains the simple DDPM model on the CIFAR-10 dataset. Since CIFAR-10 is a more diverse dataset with more classes, larger size of images, and three RGB channels, we modify the simple DDPM model by increasing the hidden dimensions to [128, 256, 256, 128, 64].
>
> On top of the losses, we track the change of FID scores. The FID scores are computed on a batch of training data of size 128, due to the memory limit of our GPUs. We acknowledge the possible imprecisions resulting from computing FID on small size batches.
>
> The training parameters are: batch_size = 64, learning_rate = 1e-3, epochs = 50.
>
> As we can tell, the training of the simple DDPM on CIFAR-10 converges, and the FID score has a decreasing trend as well. The generated images, although not as perfect as the ones with MNIST, generates meaningful shapes that are clearly distinguishable from the noise.


Losses of Unet-simple
Unet-Simple
Loss
Epochs
0.09
0.08
0.07
0.06
0.05
0
10
20
30
40
50


FID Scores of Unet-simple
Unet-Simple
FID scores
Epochs
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
10
20
30
40
50

UNet-simple

> Generating the images for different time steps is clearly the most interesting part of this project. We feel tremendously accomplished when seeing the evolution of the image from pure noise to something that makes sense, it feels like parents watching their child growing.
>
> On the other hand, we can see from the generated images that there are still some noises that are not completely eliminated, and we are not able to easily tell what our generated images are. This might due to our simple model structure and limited training time, because we found that some other DDPM implementations on CIFAR-10 takes about 10hr to train the model and use more sophisticated models with group normalizations, Resnet blocks and attention layers, etc. For example, https://www.kaggle.com/code/binh234/denoising-diffusion-probabilistic-model.

2.2 Desired Goals

> - We will compare our fine-tuned diffusion model against the score-based model [3], and BigGAN [4] (Image quality is worse. Comparison not feasible in FID).
> - We will explore whether pre-training helps improve the model performance (Complete).
> - We will explore the effect of different loss functions, such as variational bound and L_2 loss of the noise, on the model performance (Discussed).
>
> [3] Song, Yang, et al. "Score-based generative modeling through stochastic differential equations." arXiv preprint arXiv:2011.13456 (2020).
>
> [4] Brock, Andrew, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis." International Conference on Learning Representations. 2018.

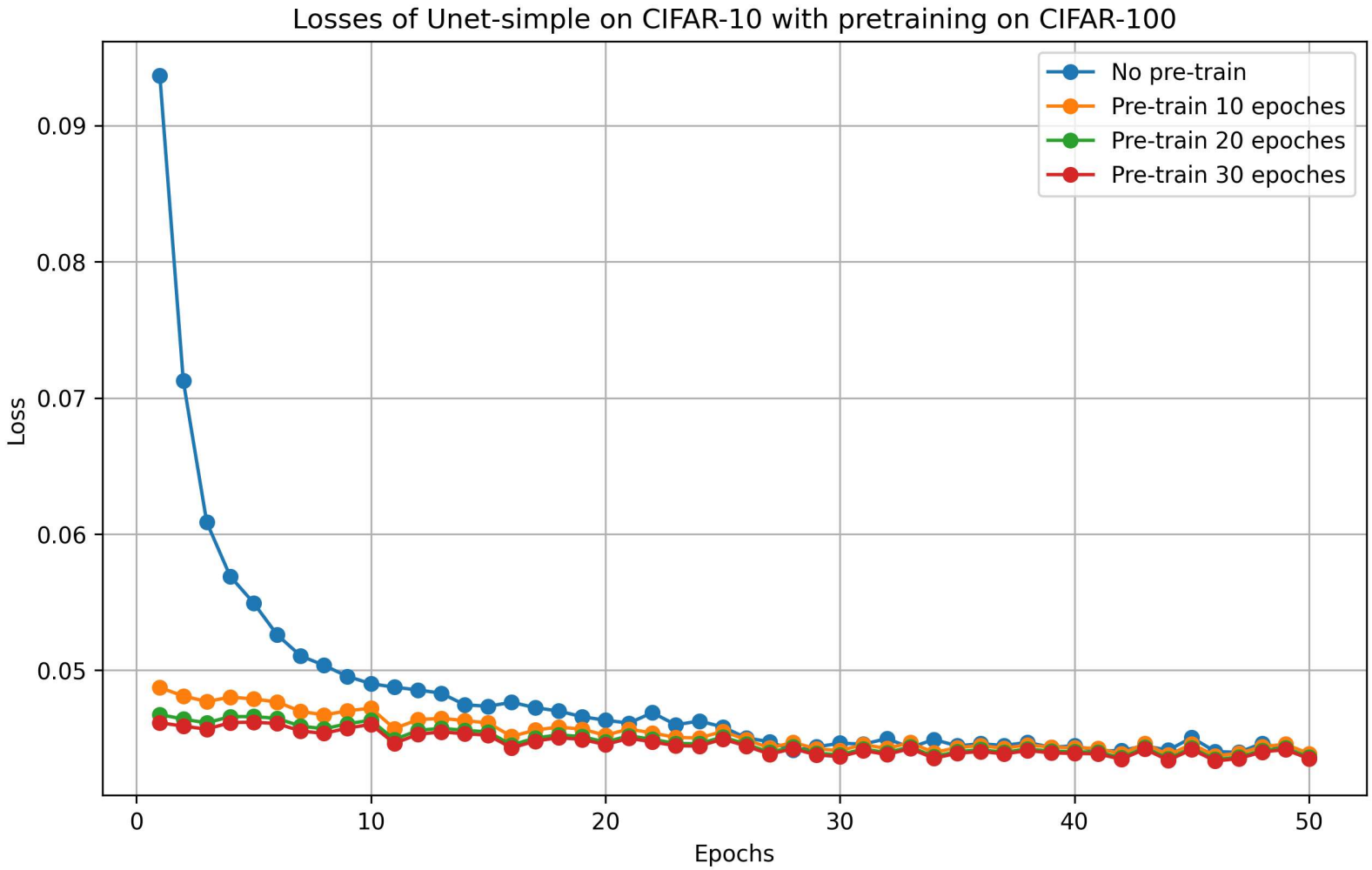
Compare with score based model and BigGAN

> This project seeks to compare the fine-tuned diffusion model against the score-based model [3] and BigGAN [4]. Due to the limits in both time and computational resources, we do not train [3] and [4] ourselves on the CIFAR-10 dataset. For example, as listed in the GitHub repo of the offical code for [3] [https://github.com/yang-song/score_sde], they use 4x Nvidia Tesla V100 GPUs (32GB) to train [3], while training the original model [3] takes more than a day with our GPUs.

> Therefore, this project seeks to compare the quality of images and the FID scores. Appearently, the generated images from [3] [https://github.com/yang-song/score_sde] and [4] [https://github.com/ajbrock/BigGAN-PyTorch] are of far superior qualities than our generated images. With respect to the FID scores, the final FID score and the smallest FID score during training for the simple DDPM is suspiciously very small. For example, the best FID score acheieved by [3] is 2.20, while the FID score of the simple DDPM is consistently below 1 after 30 epochs. The reason is that we only use a batch of size 128 to compute the FID score due to memory limit, while in [3] the whole MNIST dataset of 50000 training images is used to compute the FID score. Hence, comparing the FID score we get against the FID scores from [3] and [4] is not feasible.

Pre-training

> This project seeks to understand whether pre-training helps. Initially, this project proposes to use MNIST dataset for pre-training. It is not feasible as MNIST and CIFAR-10 images are of different sizes. Pre-training requires another image dataset with the same shape. Fortunately, CIFAR-100 has the same shape as CIFAR-10, and CIFAR-10 and CIFAR-100 are mutually exclusive with different classes.
>
> To explore whether pre-training helps, we train the model on CIFAR-100 for 10, 20, and 30 epoches, and then train the respective pre-trained models on CIFAR-10 for 50 epoches.
>
> The loss vs epoch plot is given as follows. From the plot, we can clearly tell that pre-training helps the model converge faster, and the performance of the model improves with more pre-training epochs.



Different loss functions

> This project seeks to explore the effect of different loss functions, such as variational bound and L_2 loss of the noise, on the model performance. A possible candidate for alternative loss is based on the empirical U-statistic approximation of the maximum mean discrepancy (MMD) [5]. The empirical U-statistic approximation to the MMD is unbiased and simple, and it has been applied in the likelihood-free intractable generative model context as in MMD-GAN [6]. However, due to the time limit, we abandon this goal.

[5] Gretton, Arthur, et al. "A kernel two-sample test." The Journal of Machine Learning Research 13.1 (2012): 723-773.

[6] Dziugaite, Gintare Karolina, Daniel M. Roy, and Zoubin Ghahramani. "Training generative neural networks via maximum mean discrepancy optimization." arXiv preprint arXiv:1505.03906 (2015).

2.3 Stretch Goals

- We will explore the effect of variance schedules on the model performance, and come up with novel model structures that modify the variance schedule in [1]. (Abandoned)
- We will explore how the network structure can affect the training outcome. Specifically, we are going to test the effectiveness of having group normalization layers and attention layers. (Complete)

Variance schedule

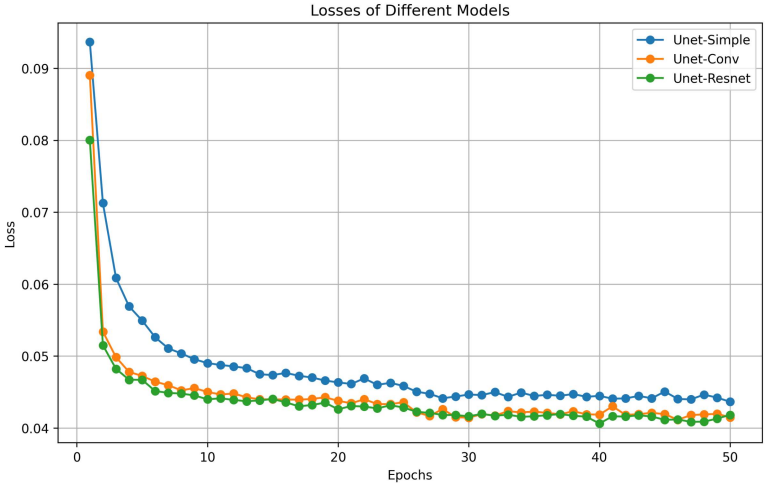
The goal is abandoned due to time limit.

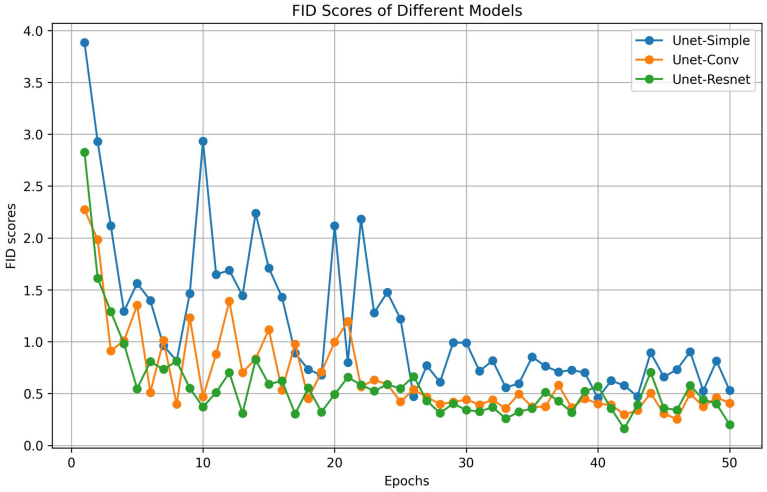
Different model performance

This project seeks to explore the effect of different network structure on the model performance. In particular, we compare our simple DDPM denoted Unet-Simple against another two models called Unet-Conv and Unet-Resnet. The main differences are that (1) Unet-Resnet uses the Resnet block with attention layes instead of the convolution blocks in Unet-Conv (2) Both Unet-Resnet and Unet-Conv are using group normalization layers in the blocks, while our Unet-simple uses batch normalizations.

One challenge is that it's not easy to directly add Resnet and attention layers to our Unet-simple model, therefore we seek help from the sophisticated model https://www.kaggle.com/code/binh234/denoising-diffusion-probabilistic-model, which we call it Unet-Kaggle. The Unet-Resnet is modified from Unet-Kaggle by replacing the downsample layers from the convolution layer to the maxpooling layer, for the purpose of reducing the number of parameters. We also reduces the hidden layer sizes. The Unet-Conv is modified from Uner-Resnet by replacing the Resnet block by

> the convolutional block. The losses and fid scores are shown below. Note that the fid score is calculated only on 128 samples, which should not be very accurate. However we do observe that the fid scores are decreasing.
>
> From the figures, we found that Unet-Conv and Unet-Resnet performs better than our Unet-simple. They not only converges faster, but also has lower limiting losses. When comparing Unet-Conv and Unet-Resnet. It is interesting that we observe that Uner-Resnet performs similar to the Unet-Conv. We expect than Unet-Resnet will perform better when we train the models on higher resolution images such as ImageNet.





2.4 Everything else

> We tried to compute the FID scores on larger samples. In [1], they used 50000 samples, but in the training process we only used 128 samples. However, we cannot do large samples with the function in torchmetric module directly due to the memory limit. In the paper [1], they manually calculate the feature columns (one of the step in calculating the FID scores) in batches. We wanted to follow their approach, but we do not have enough time to finish the code. On the other hand, calculating FID scores during the training process can make the training time very long. Therefore we didn't finish this goal.

3. Code and documentation

Models

> - "DDPM/diffusion_process.py" is the diffusion process and it's reverse process that serves as the encoder and decoder.
> - "DDPM/model/Unet_MNIST.py" is the Unet-simple model for MNIST dataset. It serves to approximate the inverse diffusion process (decoder).
> - "DDPM/model/Unet_CIFAR10.py" is the Unet-simple model for CIFAR-10 dataset.
> - "DDPM/model/Unet_CIFAR10_Conv.py" is the Unet-Conv model for CIFAR-10 dataset. It has a more sophisticated structure than Unet-simple model, with more layers and group normalization technique.
> - "DDPM/model/Unet_CIFAR10_Resnet.py" is the Unet-Resnet model for CIFAR-10 dataset. It uses Resnet blocks and attention layers.

Training files

> - "DDPM/main_MNIST.py" is the file that is training the DDPM models on MNIST dataset.
> - "DDPM/main_CIFAR10.py" is the file that is training the DDPM models on CIFAR-10 dataset.

Generate images

> - "DDPM/inference_denoising_MNIST.py" is the file that generate images from the trained DDPM models on MNIST dataset.
> - "DDPM/inference_denoising_CIFAR10.py" is the file that generate images from the trained DDPM models on CIFAR10 dataset.

4. Reflections

What was interesting?

> The most interesting and exciting part, as we mentioned above, is seeing how the image evolves from pure noise to something that makes sense. Besides, we learned how to use cuda, learned how to compute FID score, applied ResNet, and practiced using Git and debug with VSCode. We were quite well-trained in math and statistics, but the coding practice at class is really helpful for our coding and understanding other people's codes.

What was difficult?

> What was difficult is balancing our other schedules with the project work. Both of us are PhD students in Operations Research, and we are all busy on our respective papers. Ziwei has quite a few difficulties as he has quite a few internship interviews in the final weeks, and his (former) advisor left NU and he is still in a middle of a not-so-smooth transition to another group. We are glad it all works out because of our wonderful teamwork.

What's left to do?

> We will try to spend the money on some good GPUs such as the V100s so that we can train a deeper DDPM and compute the FID score with the whole CIFAR-10 dataset. The model is clearly still underfit as the generated image is still a bit noisy. We are definitely going to try incorporate the MMD loss.

Advice for future students

> Don't underestimate the scale and the computational resource required of generative models. Don't hesitate to use external computational resources; we underestimated the scale of the model so we can only train and evaluate our model on a much smaller scale.