LIUZHIXIAN
3036373609

Report of DASC7606 A2

Introduction: Denoising Diffusion Probabilistic Models (DDPMs) have garnered widespread attention for their exceptional performance in generating high-quality images, which introduce noise step by step and train the model to reverse this process, producing samples that are rich in detail and diverse. Meanwhile the MNIST dataset is one of the most famous datasets in CV and there are many works and models related to it. This assignment is aiming for training a DDPM to generate handwriting number pictures of MNIST dataset and furthermore training a model that can generate the number pictures under given labels.

Method: In this program, I construct the diffusion model based on the U-Net backbone construction. The basic U-Net module is formed by a decoder and an encoder, and both are similar to the construction mentioned in the paper[1] but add a hidden state called time MLP to their inner blocks as the information of the step/time of the diffusion process. As to the model dealing with the given label part, I add a hidden state called label MLP after updating the time information in each block to convey the label information into it. In training sessions, I utilize the AdamW as the optimizer and MSE as the loss function to minimize the difference between predicted noise and the true noise.

Experiment:

1. Origin model with cos variance and 1000 timesteps:


10 epochs


30 epochs


60 epochs


100 epochs

2. Origin model with cos variance and 10 timesteps:


10 epochs


30 epochs


60 epochs

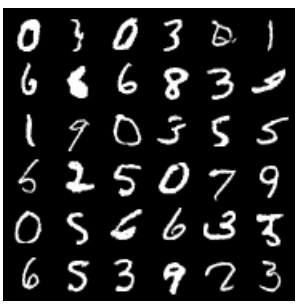

100 epochs

3. Origin model with linear variance and 1000 timesteps:


10 epochs


30 epochs


60 epochs


100 epochs

1. https://arxiv.org/abs/1505.04597

4. Model with given labels with cos variance and 1000 timesteps:

10 epochs

30 epochs

60 epochs

100 epochs

5. 0 – 9 outputs with labels:

Analysis:

1. Diffusion can handle the generative task ideally and can generate high-quality images; however, every image generated by the diffusion model is somehow different from each other, which may lead to instability.
2. Compared to the original model, the model trained to generate numbers under given labels converges to the final state in a relatively early epoch. I think this is because the hidden states of the label provide more information than the original one, hence the faster training speed.
3. The model with fewer timesteps converges to the final state faster than the original one, but after finishing all the training procedures, the quality of numbers generated by the model with fewer timesteps is much lower than the original one or we can say the outputs of the 10 timesteps model is more unstable. I think this is because fewer timesteps means fewer states between the noise and the true picture hence the model cannot learn as much as the model with larger timesteps.
4. The linear variance schedule is harder to use than the cos one because the range of the x_0 to x_T needs to be set properly (0.0001 to 0.02). For example, if I set the linear range from 0.001 to 0.999, the training process will not work. By studying from the Internet, I have learned that linear variance will make the noise grow much faster than the cos variance which will lead to a large amount of the samples being totally noised and cannot learn anything from them.
5. The output of the model with linear variance schedule is less quantitative than the original one, I think the reason is similar to the case when decreasing the timesteps which is that the linear variance schedule will produce less valid noised samples than the cos variance schedule, hence the training process is less effective.

1. https://arxiv.org/abs/1505.04597