

Generative Models for Visual Signals – Assignment

資訊113 E14096407 范芳瑜

Github link: [github_project4](#)

Theoretical Justification

Denoising Diffusion Probabilistic Models (DDPM) has the following advantages:

1. DDPMs can typically generate high-quality and high-fidelity images, as they gradually transform noise into an image in a relatively stable and controllable process.
2. Since DDPM relies on step-by-step denoising, each step in the generation process maintains high stability and controllability, better avoiding instabilities in the generation process.

Deep Image Prior (DIP) has the following advantages:

1. One notable advantage of DIP is that it doesn't require a pre-trained dataset. It processes only using the input image itself, which is particularly useful when training data is scarce or unavailable.
2. Since it does not require extensive data for training, DIP is very simple to apply. Optimization is performed on a single image, making it highly convenient for various practical applications.

Considering DIP's excellent performance in image denoising and its easier application, I plan to try "Example 1: Accelerating DDPM with DIP-based Initial Priors" by integrating the DIP model into the training process of DDPM.

This combination may bring several potential advantages:

1. Utilizing the initial priors generated by DIP can help DDPM converge to high-quality solutions more rapidly, as DIP provides a good estimate of natural image statistics.
2. DIP demonstrates excellent performance in image denoising. Integrating it into DDPM can improve the handling of noise during generation, thereby enhancing the quality and fidelity of generated images.
3. DDPM and DIP each possess distinct strengths that can complement one another. Combining DDPM's diffusion capabilities with DIP's efficient generation may lead to overall better results.

Also, I consider some potential challenges that may arise from combining DIP with DDPM:

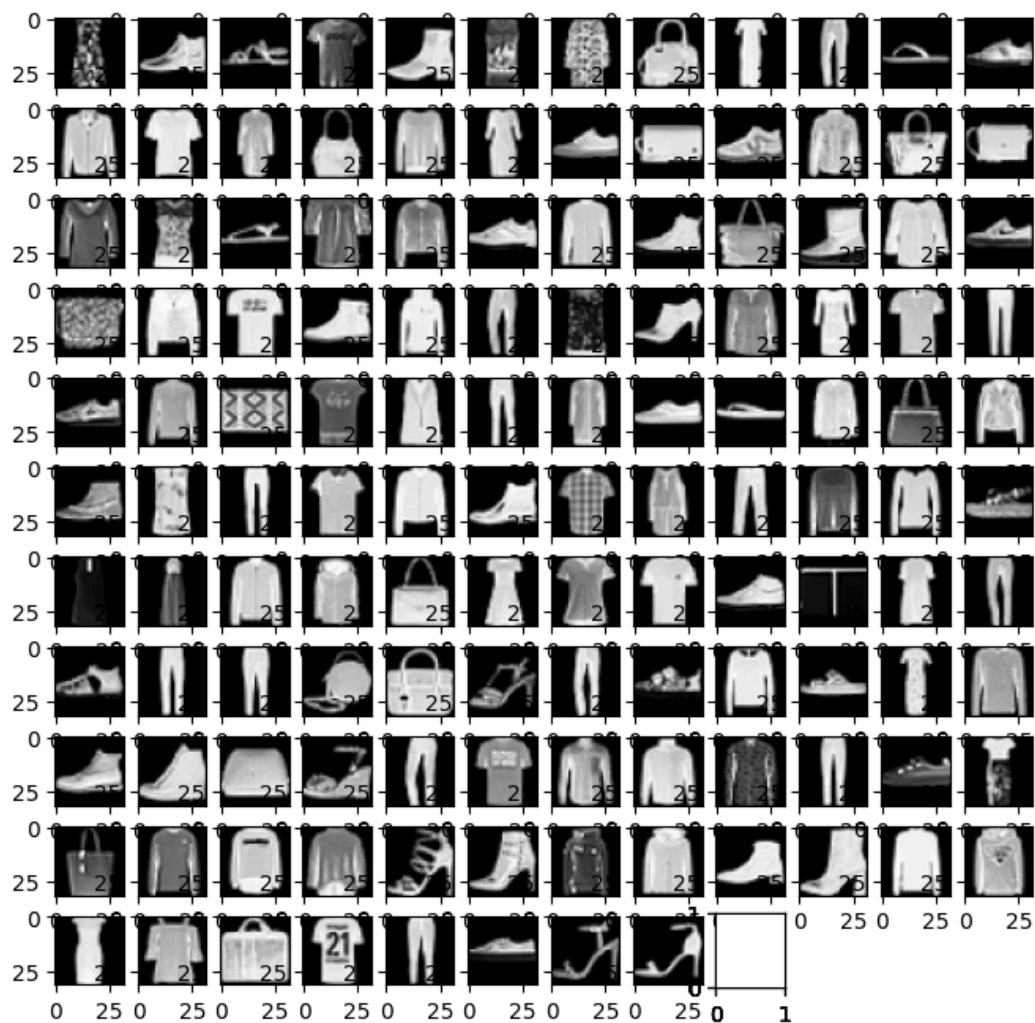
1. Integrating DIP into DDPM's training process could introduce additional complexity in model architecture and training procedures.
2. Both DIP and DDPM rely on intricate optimization processes.
3. The combined model may demand significant computational resources due to the computational intensity of both DDPM's diffusion process and DIP's iterative optimization.
4. Combining two powerful models could increase the risk of overfitting to specific datasets or tasks.

Experimental Verification

Dataset: FashionMNIST (A fashion classification dataset from Kaggle.)

Since the DDPM tutorial article and sample code use this dataset, I decided to use it as well.

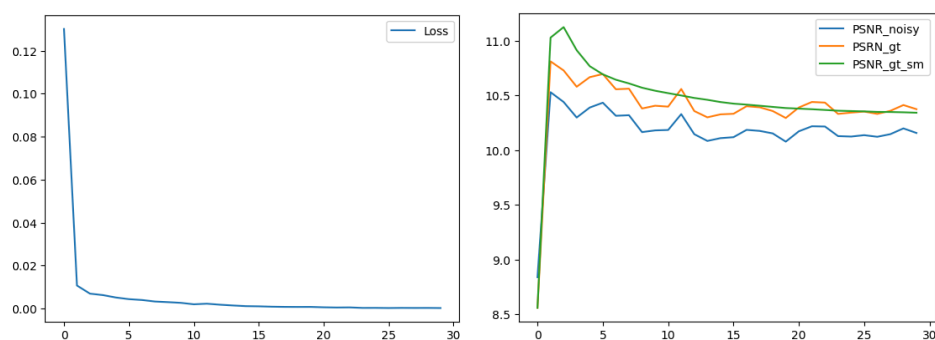
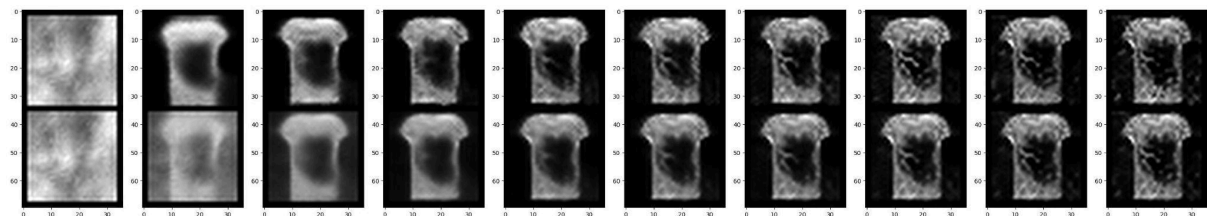
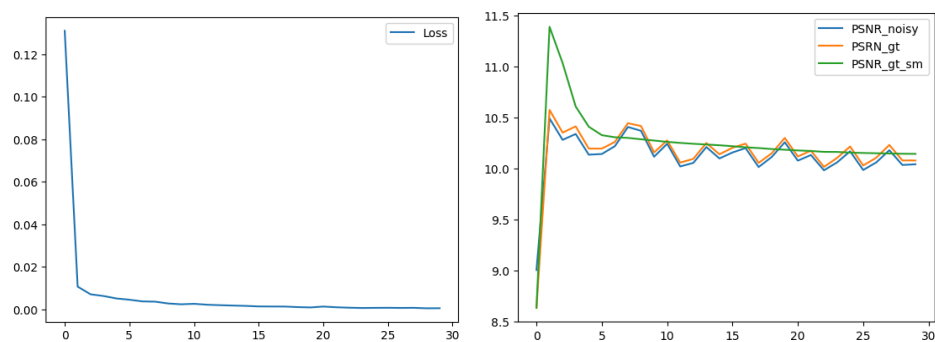
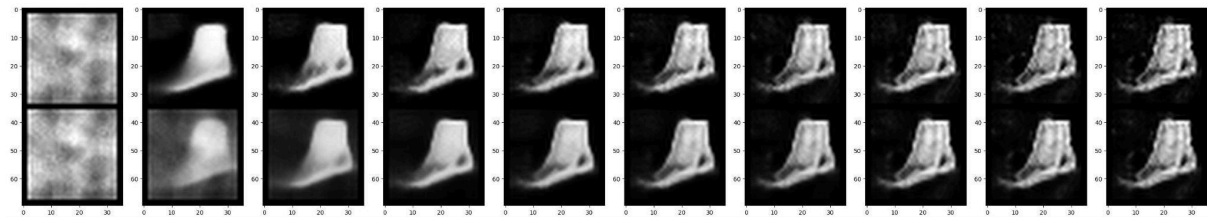
Images in the first batch

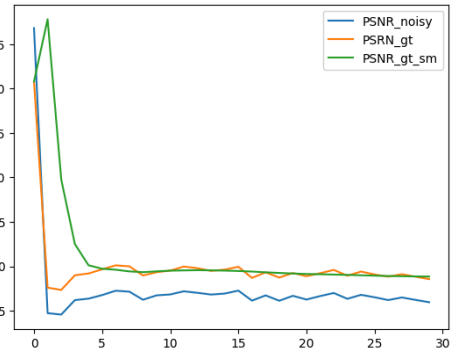
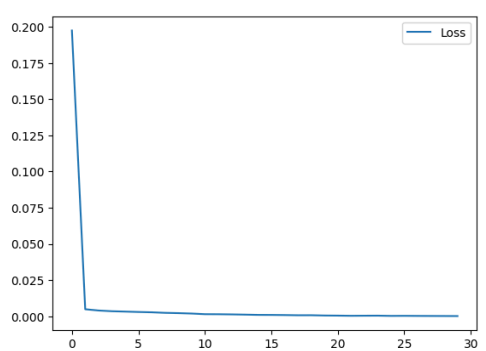
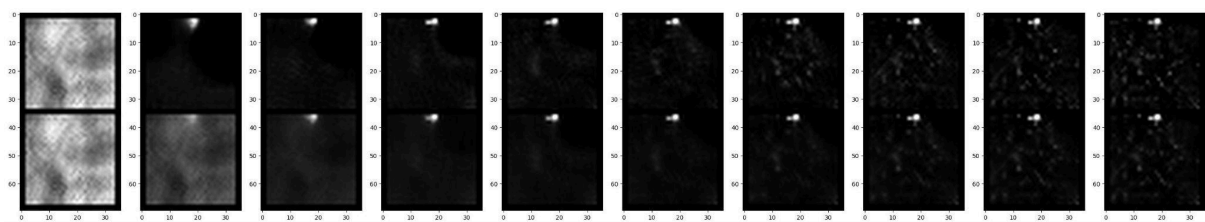
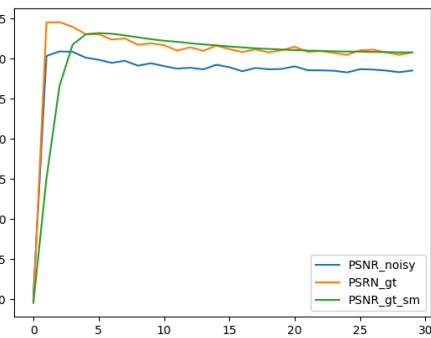
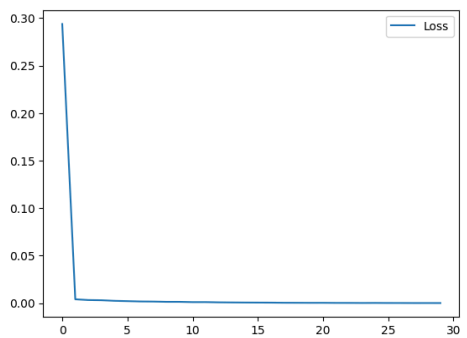
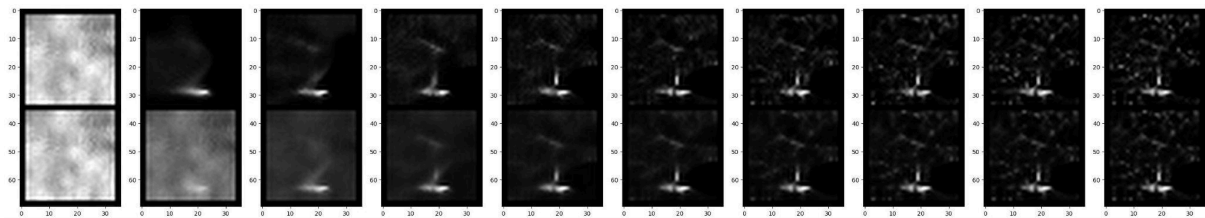
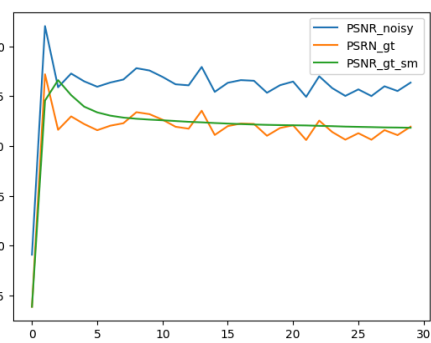
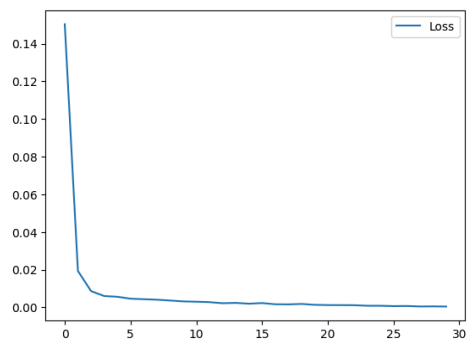
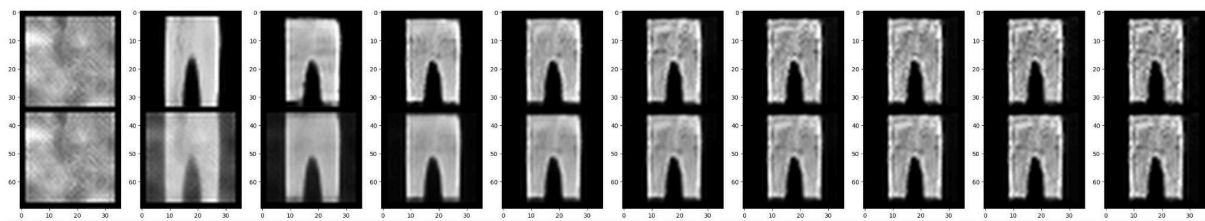


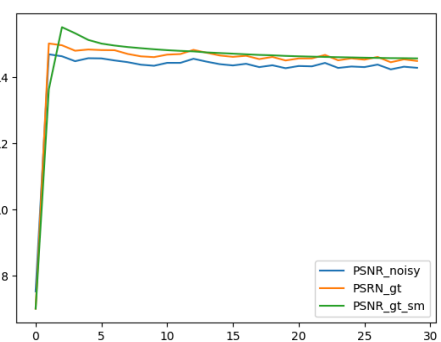
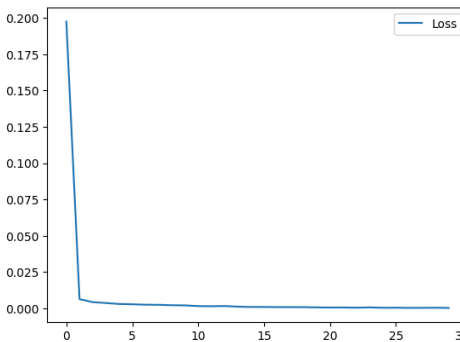
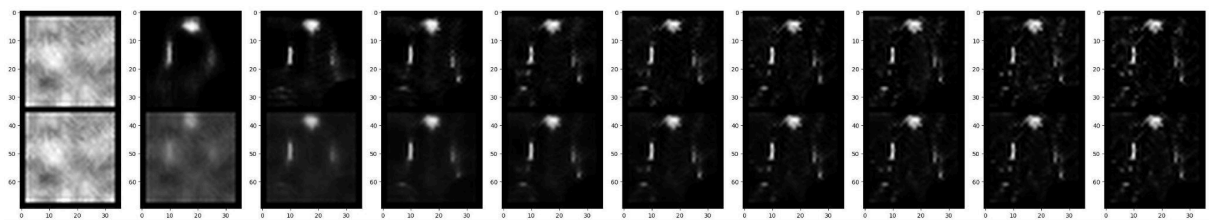
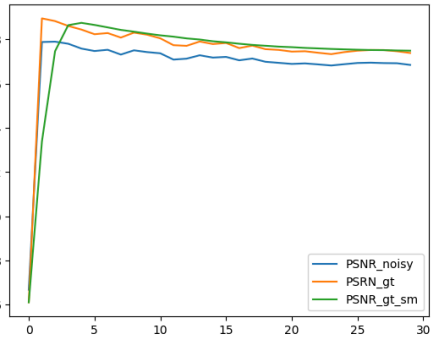
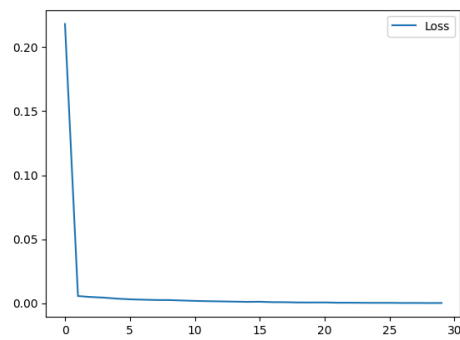
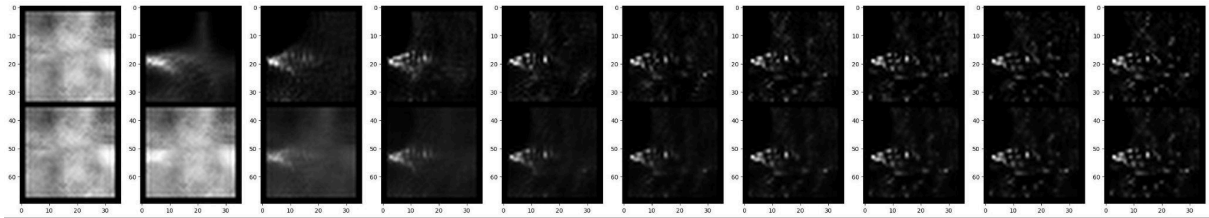
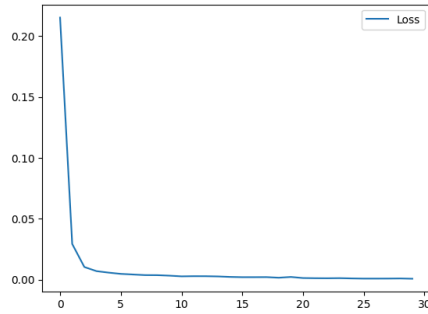
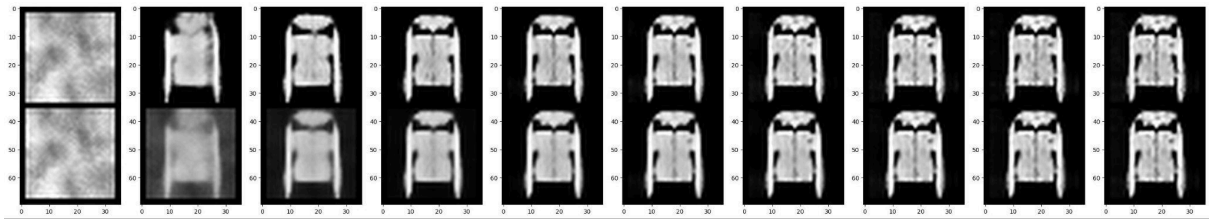
Deep Image Prior (DIP) [DIP Github](#)

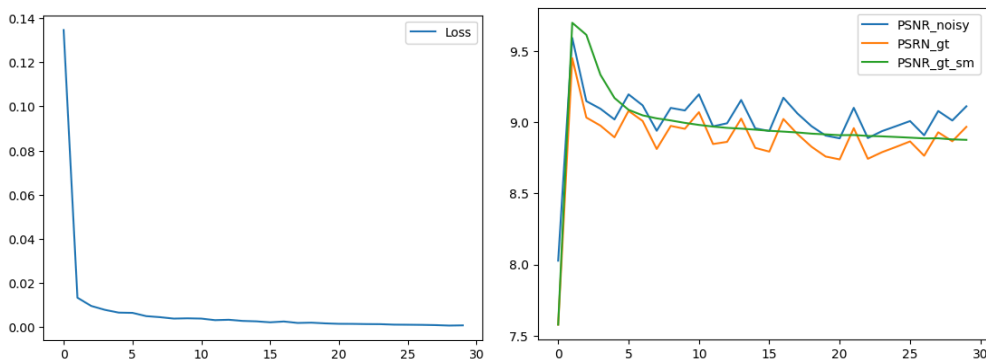
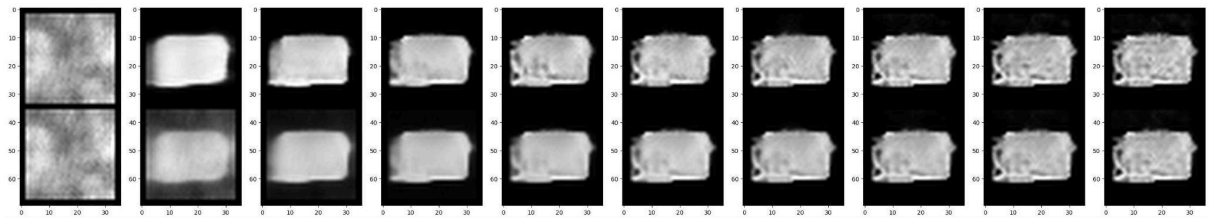
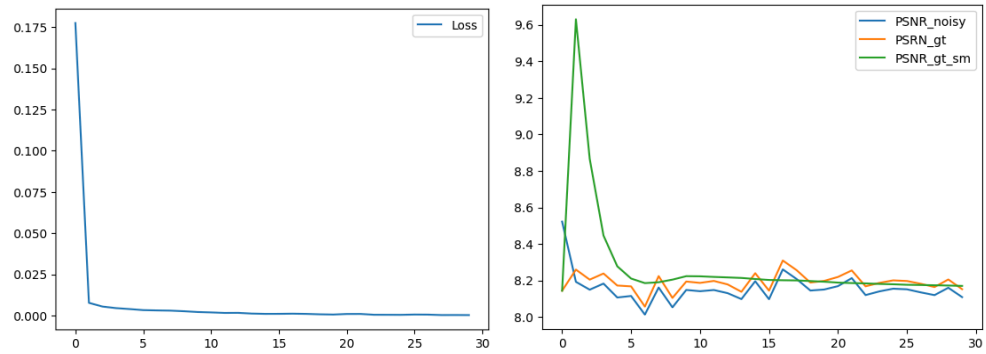
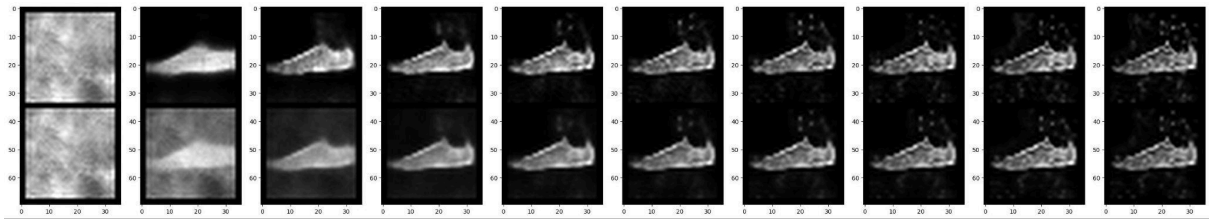
In the implementation of DIP, I cloned a copy of the author's original code and modified it to use different input images. I selected the first ten images from the batch to demonstrate the denoising process with DIP. The optimization process was relatively fast, completing in about 8 minutes. Throughout the process, PSNR was used to assess the denoising effectiveness.

The results are as follows:









From the above charts, it can be observed that DIP achieves a rapid decrease in loss within a short period. The improvement in PSNR is also notable. However, after continued optimization, there is a slight decline in PSNR, which stabilizes afterwards. It is speculated that after multiple rounds of training, overfitting may become a concern. Nonetheless, this confirms that DIP can effectively and simply address denoising problems.

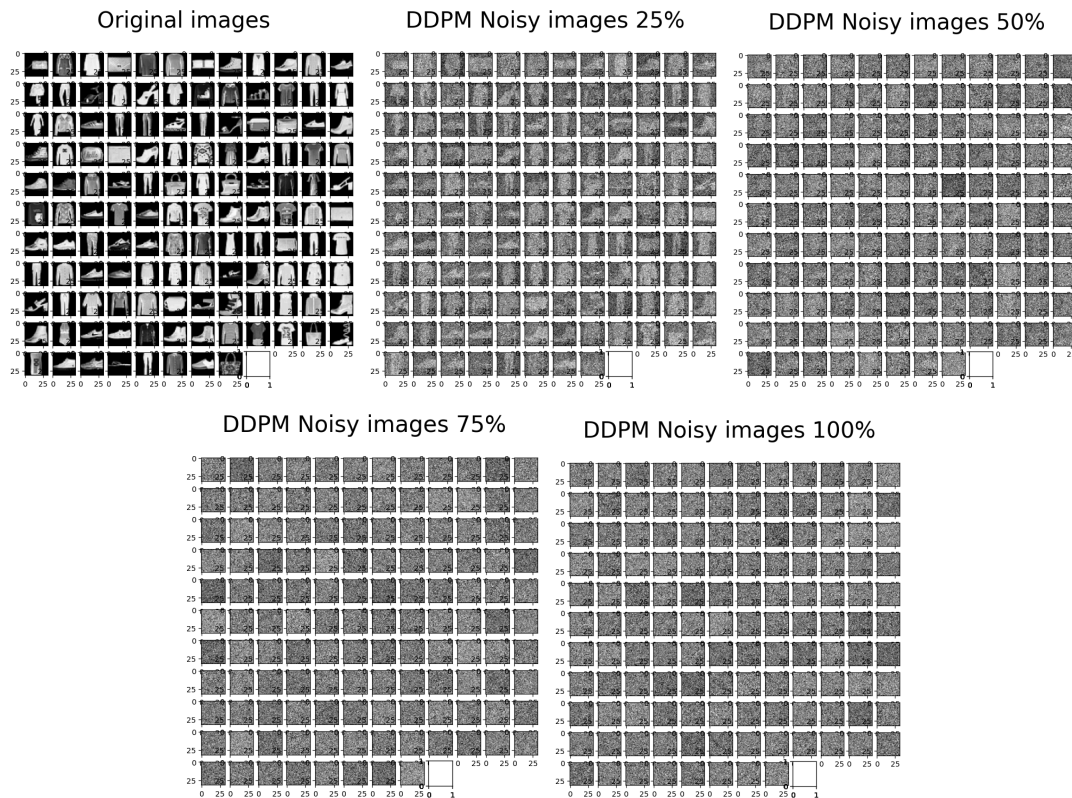
Denoising Diffusion Probabilistic Models (DDPM)

[DDPM code reference](#)

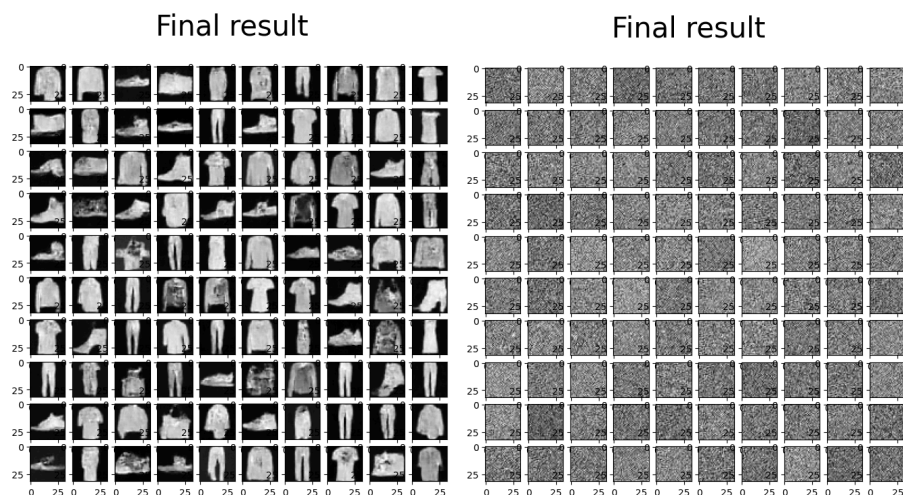
For implementing the DDPM model, following a Medium article and example code, I utilized the FashionMNIST dataset to build a denoising model. DDPM can be

straightforwardly divided into forward and backward steps. I plan to integrate the DIP model between these two steps, where the output of DIP serves as the input for the backward step.

The results of the DDPM forward pass are as follows:

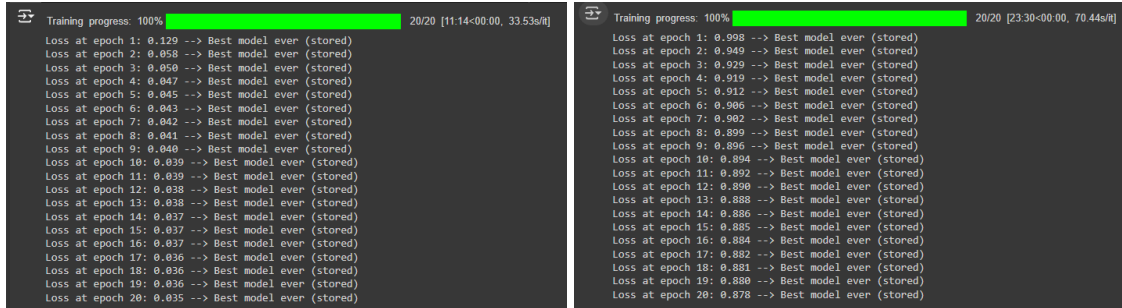


When using DDPM backward for denoising, the training time is approximately 11 minutes, and the obtained results are as follows (Left). However, when using DIP as the initial prior for DDPM, the training time increases to 23 minutes. It is observed that there is no acceleration in training. Furthermore, the denoising results become worse (Right)



Ablation Studies and Analysis

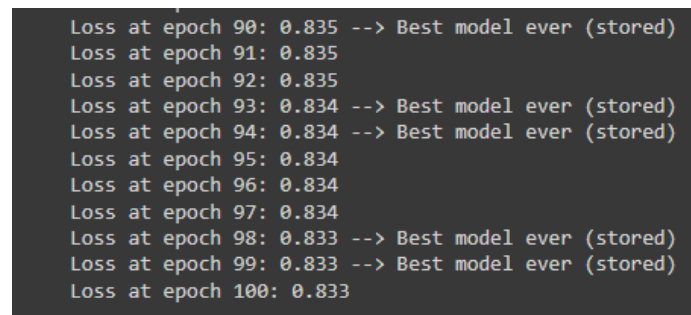
I observed the convergence of loss in both models and found that they are indeed converging with each epoch. However, in the version where DIP is integrated, the loss decreases very slowly.



```
Training progress: 100% 20/20 [11:14<00:00, 33.53s/it]
Loss at epoch 1: 0.129 --> Best model ever (stored)
Loss at epoch 2: 0.058 --> Best model ever (stored)
Loss at epoch 3: 0.050 --> Best model ever (stored)
Loss at epoch 4: 0.047 --> Best model ever (stored)
Loss at epoch 5: 0.045 --> Best model ever (stored)
Loss at epoch 6: 0.043 --> Best model ever (stored)
Loss at epoch 7: 0.042 --> Best model ever (stored)
Loss at epoch 8: 0.041 --> Best model ever (stored)
Loss at epoch 9: 0.040 --> Best model ever (stored)
Loss at epoch 10: 0.039 --> Best model ever (stored)
Loss at epoch 11: 0.039 --> Best model ever (stored)
Loss at epoch 12: 0.038 --> Best model ever (stored)
Loss at epoch 13: 0.038 --> Best model ever (stored)
Loss at epoch 14: 0.037 --> Best model ever (stored)
Loss at epoch 15: 0.037 --> Best model ever (stored)
Loss at epoch 16: 0.037 --> Best model ever (stored)
Loss at epoch 17: 0.036 --> Best model ever (stored)
Loss at epoch 18: 0.036 --> Best model ever (stored)
Loss at epoch 19: 0.036 --> Best model ever (stored)
Loss at epoch 20: 0.035 --> Best model ever (stored)

Training progress: 100% 20/20 [23:30<00:00, 70.44s/it]
Loss at epoch 1: 0.998 --> Best model ever (stored)
Loss at epoch 2: 0.949 --> Best model ever (stored)
Loss at epoch 3: 0.929 --> Best model ever (stored)
Loss at epoch 4: 0.919 --> Best model ever (stored)
Loss at epoch 5: 0.912 --> Best model ever (stored)
Loss at epoch 6: 0.906 --> Best model ever (stored)
Loss at epoch 7: 0.902 --> Best model ever (stored)
Loss at epoch 8: 0.899 --> Best model ever (stored)
Loss at epoch 9: 0.896 --> Best model ever (stored)
Loss at epoch 10: 0.894 --> Best model ever (stored)
Loss at epoch 11: 0.892 --> Best model ever (stored)
Loss at epoch 12: 0.890 --> Best model ever (stored)
Loss at epoch 13: 0.888 --> Best model ever (stored)
Loss at epoch 14: 0.886 --> Best model ever (stored)
Loss at epoch 15: 0.885 --> Best model ever (stored)
Loss at epoch 16: 0.884 --> Best model ever (stored)
Loss at epoch 17: 0.882 --> Best model ever (stored)
Loss at epoch 18: 0.881 --> Best model ever (stored)
Loss at epoch 19: 0.880 --> Best model ever (stored)
Loss at epoch 20: 0.878 --> Best model ever (stored)
```

Therefore, I increased the number of epochs in an attempt to bring the loss convergence closer to the level of DDPM alone.



```
Loss at epoch 90: 0.835 --> Best model ever (stored)
Loss at epoch 91: 0.835
Loss at epoch 92: 0.835
Loss at epoch 93: 0.834 --> Best model ever (stored)
Loss at epoch 94: 0.834 --> Best model ever (stored)
Loss at epoch 95: 0.834
Loss at epoch 96: 0.834
Loss at epoch 97: 0.834
Loss at epoch 98: 0.833 --> Best model ever (stored)
Loss at epoch 99: 0.833 --> Best model ever (stored)
Loss at epoch 100: 0.833
```

Based on the information in the picture, we can see that the epoch has been increased to 100. The loss is indeed continuing to decrease, but very slowly, and there is no significant improvement in denoising results.

As a result, based on the observation that integrating DIP into DDPM did not improve performance compared to DDPM alone, possible reasons include:

1. DIP may have disrupted the high stability of DDPM.
2. I did not adjust the algorithm of the DIP model according to the timestep of DDPM, which may have caused the DIP outputs to lack necessary data for DDPM during the backward process.
3. Both DDPM and DIP are complex models, and directly combining them might lead to excessive complexity. It may be necessary to delve deeper and modify the internal algorithms of both models to achieve the desired results.