Report of Homework5

I have put all the experiment result into folder `result record` with the following structure:

```
result record
├── parameter 1
│   ├── submission
│   │   ├── 1.png
│   │   ├── 2.png
│   │   ├── ...
│   ├── samples
│   │   ├── sampling_step_0.png
│   │   ├── sampling_step_100.png
│   │   ├── ...
│   ├── parameters.png
│   ├── model-30.pt
│   ├── sample-30.png
│   ├── FID.png
│   └── ...
├── parameter 2
└── ...
├── source
│   ├── cat_10000.jpg
│   ├── cat_10001.jpg
│   ├── ...
├── fusion
│   ├── 10000_10001.jpg
│   ├── ...
├── cat_body_diffusion
│   ├── submission
│   │   ├── 1.jpg
│   │   ├── 2.png
│   │   ├── ...
│   ├── model-30.pt
│   ├── sample-30.png
│   ├── FID.png
├── cat_body_GAN
│   ├── model.pt
│   ├── images
│   │   ├── generated.jpg
│   │   ├── generated-ema.png
│   │   ├── generated-mr.png
```

Task1: The random generation with diffusion

1.1 The principles and ideas of diffusion Model

Diffusion model is a generative model designed to create data by iteratively transforming noise into a desired distribution.

Key Steps:

1. **Forward Process (Noise Addition)**:
   Starting with the original data, the model applies a series of Gaussian noise additions over multiple time steps. This gradually destroys the structure of the data, transforming it into pure noise.
2. **Reverse Process (Noise Removal)**:
   The reverse process involves learning to denoise the data step by step, moving backward from the noise to the original data distribution. This reverse transformation is achieved by training a neural network to approximate the denoising steps using the conditional probabilities.

Core Idea:
The model learns to approximate the reverse diffusion process by minimizing a loss function, typically derived from the Kullback-Leibler (KL) divergence between the true and estimated distributions. By training on many examples, it effectively learns how to recreate complex data distributions starting from random noise.

1.2 Analysis of Hyperparameters

1. `batch_size`
   The number of samples processed in a single forward and backward pass, Affecting memory usage and training speed.
2. `train_num_steps`
   Total number of training steps, defining how long the model trains.
3. `lr` (Learning Rate)
   The step size used during optimization(gradient descent). A value that is too large may lead to unstable training, while a value that is too small may result in slower convergence speed.
4. `grad_steps`
   Description: Gradient accumulation steps. When memory is limited, gradients are accumulated over multiple forward passes before an update. The effective batch size becomes `batch_size * grad_steps` .
5. `ema_decay`
   Decay rate for the exponential moving average of the model parameters. A higher value ensures smoother updates for the EMA parameters, often improving model stability.
6. `channels`
   Number of channels in the first layer(convolutional layer) of Unet model.

7. `dim_mults`
   Multipliers for the dimensions of subsequent layers. The model size grows as (channels, 2 * channels, 4 * channels) if `dim_mults = [1,2,4]` .
8. `timesteps`
   Number of timesteps in the diffusion process, which defines how many noise addition/removal steps are used in duffusion model.
9. `beta_schedule`
   The schedule for beta values (variance of the noise) over the diffusion process. Common schedules include linear, quadratic, or cosine.

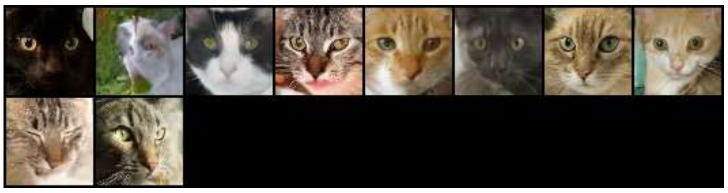
1.3 Parameter tuning process

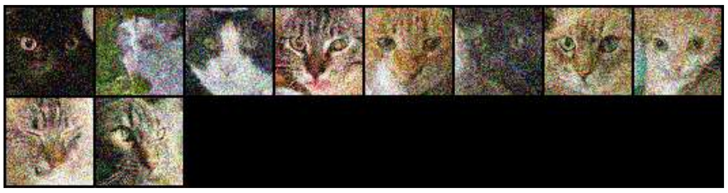
(Due to time limit, I just try 5 versions of parameters and let `grad_steps` `ema_decay` `timesteps` `beta_schedule` keep their original value.)

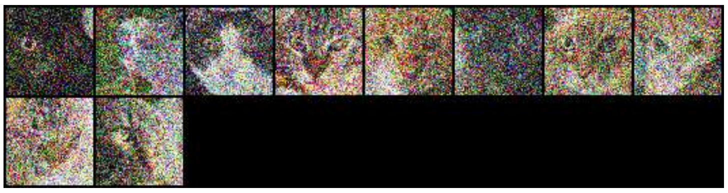
1.3.1 Result display

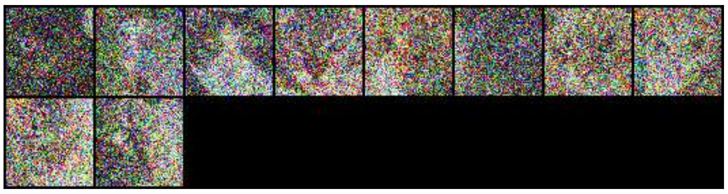
batch_size	lr	channels	dim_mults	FID
16	0.001	16	(1,2,4)	51.614
16	0.001	64	(1,2,4,8)	37.873
16	0.0005	64	(1,2,4,8)	29.404
16	0.0001	128	(1,2,4,8)	25.702
32	0.0001	128	(1,2,4,8)	21.730

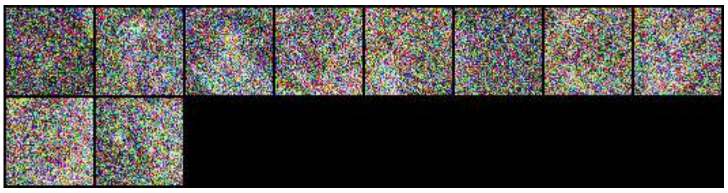
The following is the image display during the denoising process(with the best version of parameters):

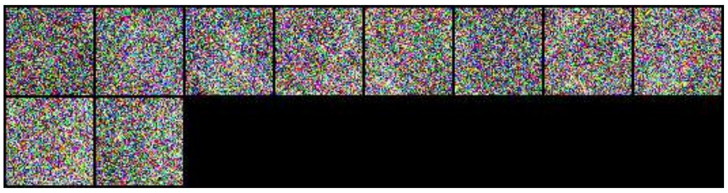


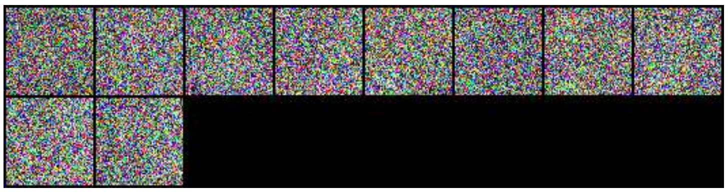


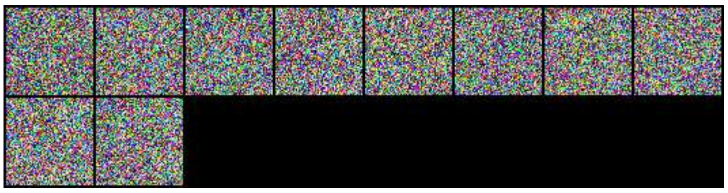


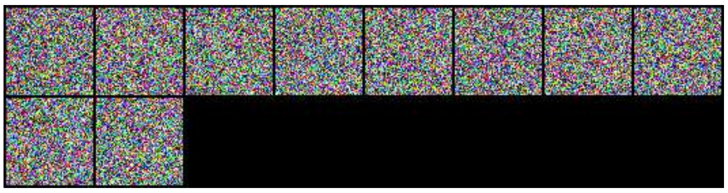


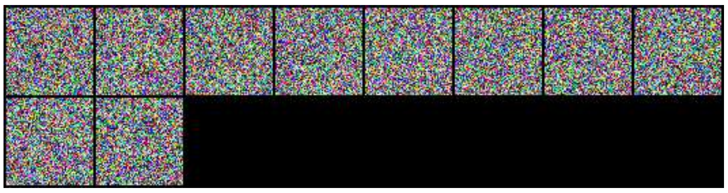












We can clearly see how the image transforms from random noise to a cat face by denoising.

1.3.2 The idea of parameter setting

Firstly, I believed that increasing channels and the number of layers in U-net model will make the model larger, which will improve the performance. So I added `channels` to 64 and change `dim_mults` into `(1,2,4,8)` , resulting in obvious decrease of FID.

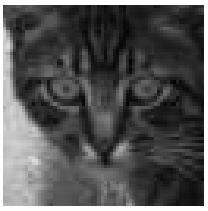
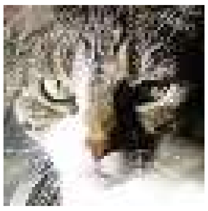
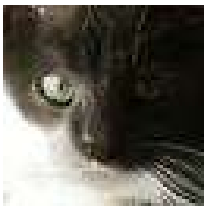
Then, I thought that `lr=1e-3` could be somehow large(resulting in unstable training), so I decreased its value to `le=5e-4` , resulting in decrease of FID, which validated my thought.

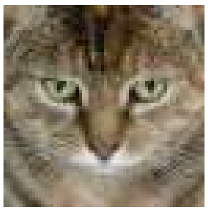
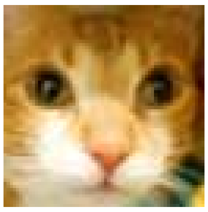
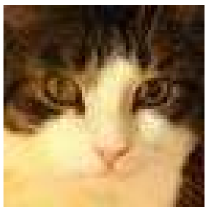
Then, I added the value of `channels` to 128 and decreased the value of `lr` to `1e-4` , resulting in decrease of FID.

Lastly, I added `batch_size` to `32` to check if the increase of `batch_size` can lead to better result. The fact prove that it works.

Task2: Fusion generation with diffusion

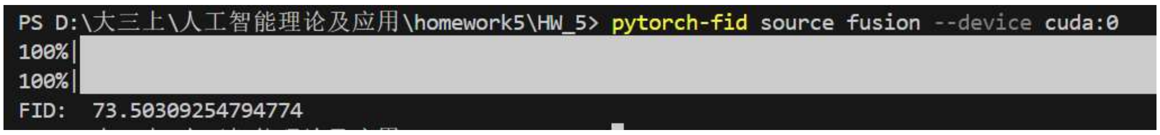
I used `trainer.inference2` for fusion generation. Here are two examples of source and fusion images.

We can see that the generated results have the characteristics of the source images such as color, shape of face and so on.

The FID between source images and fusion images is 73.503



```
PS D:\大三上\人工智能理论及应用\homework5\HW_5> pytorch-fid source fusion --device cuda:0
100%|
100%|
FID:  73.50309254794774
```

Task4: Use the styleGAN

4.1 The principles and ideas of styleGAN

StyleGAN (Style-Based Generative Adversarial Network) is a state-of-the-art generative model designed to produce high-quality, diverse, and controllable images. It introduces a novel architecture that decouples the latent space from the image synthesis process, offering unprecedented control over the generated content.

Key Principles

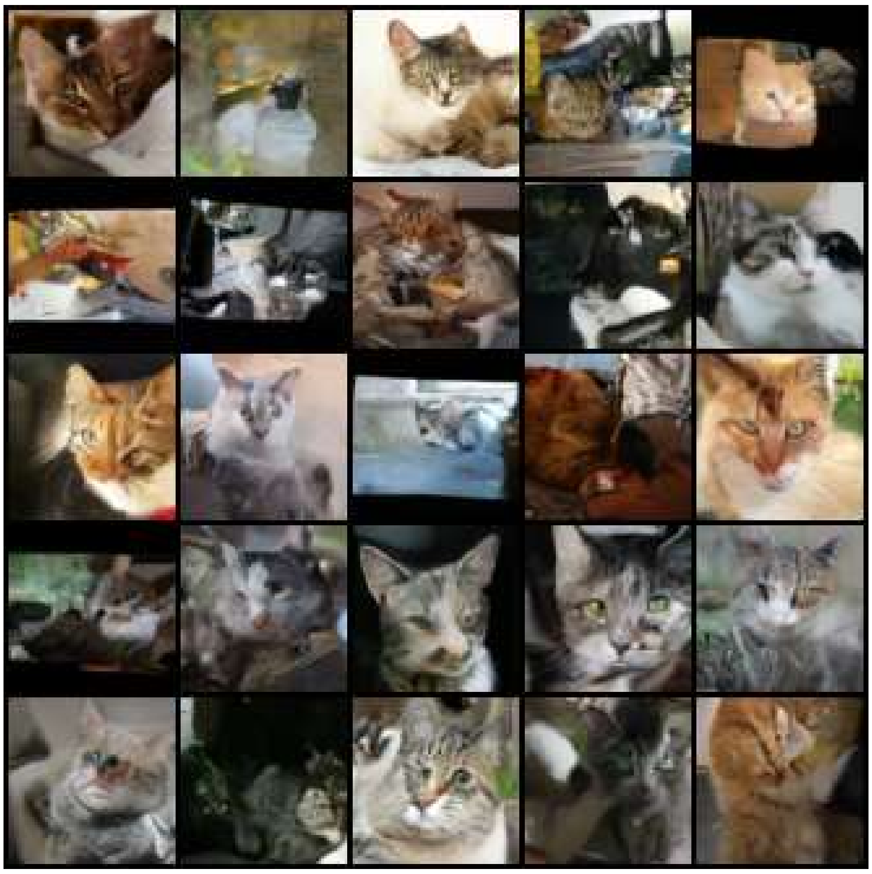
1. Style-Based Latent Space:
   Instead of directly using a latent vector z as input to the generator, StyleGAN transforms z into an intermediate latent space w through a mapping network. This intermediate space provides a disentangled representation, allowing better control over specific features.

2. Adaptive Instance Normalization (AdaIN):
   StyleGAN uses AdaIN layers to modulate feature maps at each level of the synthesis network. The style vector w controls the scaling and shifting of these feature maps, enabling fine-grained control over image attributes like texture, color, and structure.
3. Progressive Growing:
   The model generates images progressively, starting from a low resolution (like 4×4) and incrementally adding higher-resolution layers. This ensures stable training and improves image quality.
4. Noise Injection:
   Random noise is added at each layer of the generator, introducing stochastic variations and enhancing details like hair strands or freckles.
5. Disentanglement and Control:
   By separating the latent space and the synthesis process, StyleGAN allows users to manipulate specific aspects of an image (e.g., hair style, lighting) without affecting others.

Applications
StyleGAN is widely used in areas like art, fashion, gaming, and data augmentation. It has also inspired research into controllable generation, style transfer, and other creative AI applications.

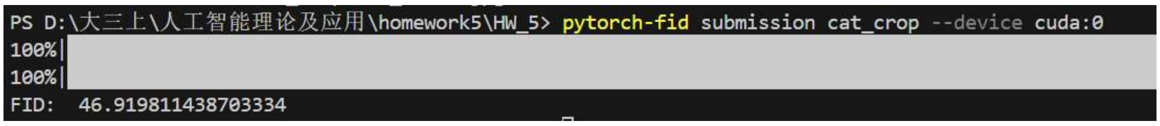
4.2 Train and test the diffusion model with cat body dataset

The result can be seen in `sample-30.png` :



We can see that the result of generated cat body images is of poor quality. In the generated image, the cats' bodies are incomplete or distorted.

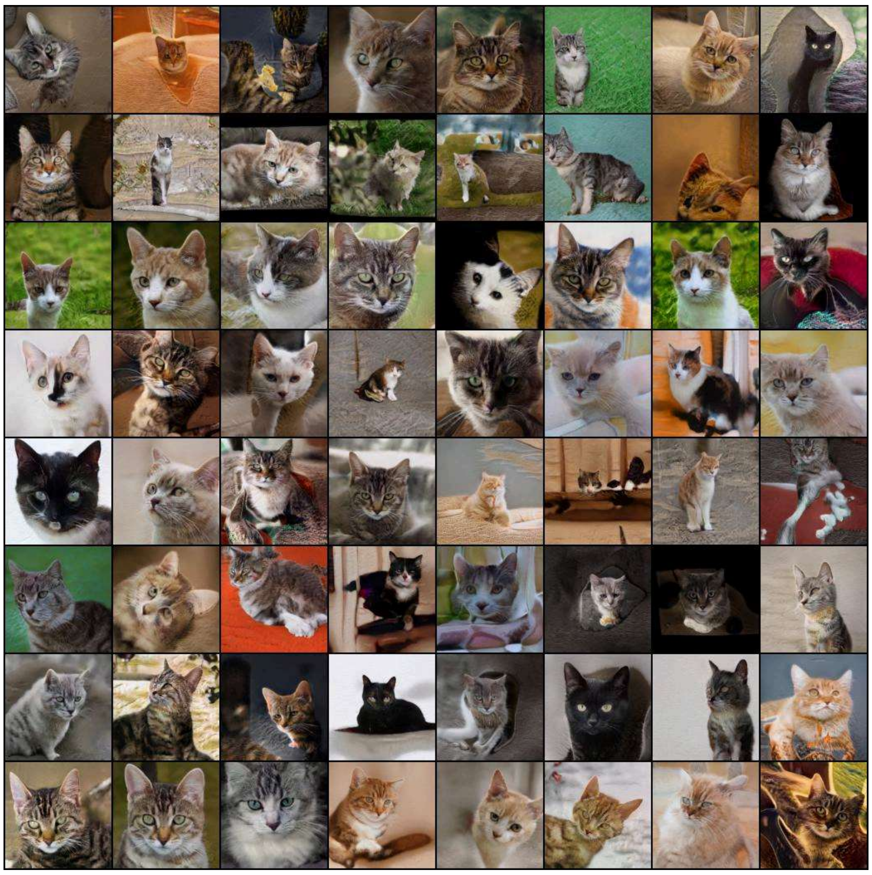
The FID between generated images and cat body dataset is calculated to be 46.9198.



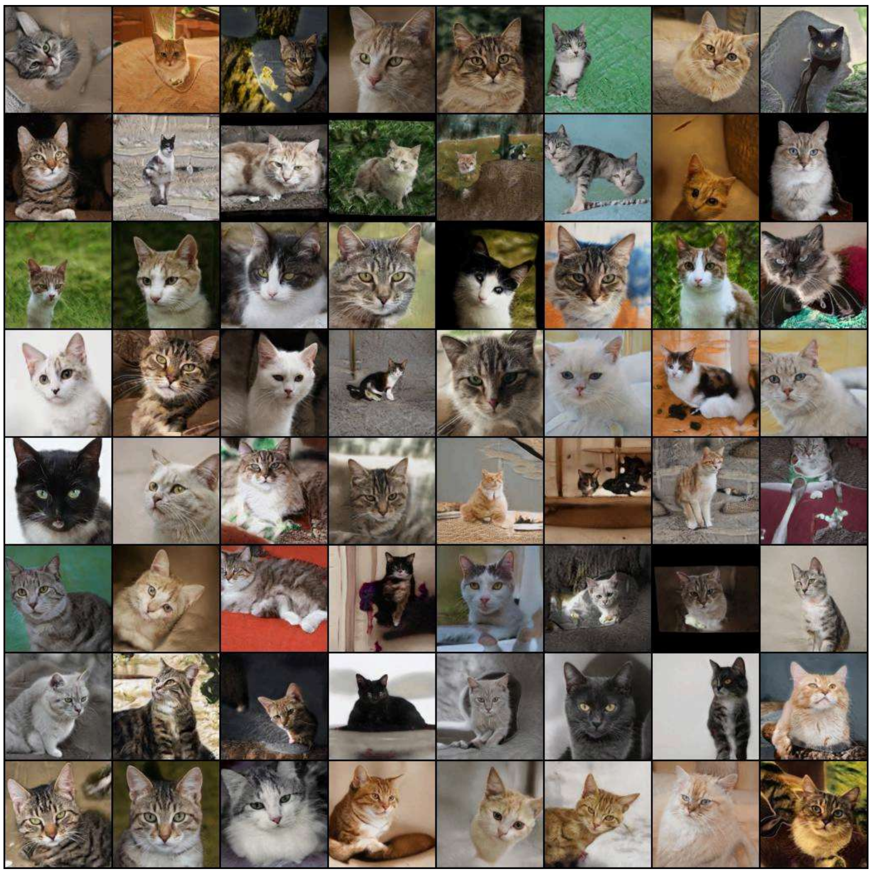
```
PS D:\大三上\人工智能理论及应用\homework5\HW_5> pytorch-fid submission cat_crop --device cuda:0
100%|
100%|
FID:  46.919811438703334
```

4.3 Train and test the model with cat body dataset

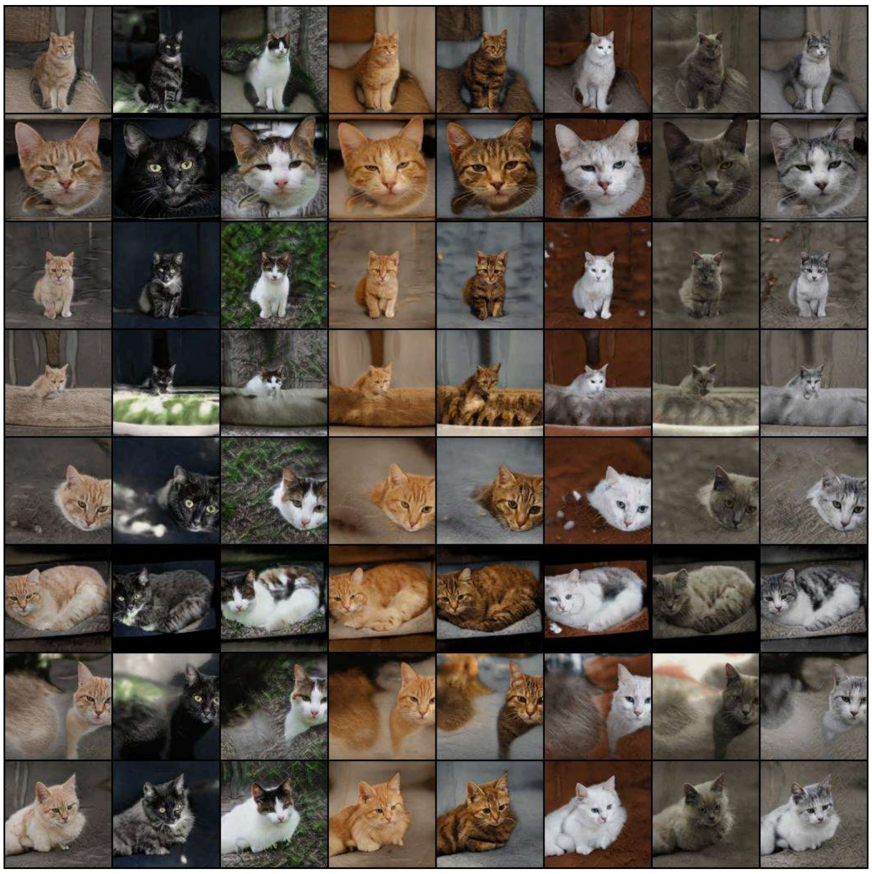
The result can be seen in `generated.jpg` :



And mixed style results can be seen in `generated-ema.jpg` :



Also, "gradual change" results can be seen in `generated-mr.jpg` :



We can see that the result of generated cat body images by styleGAN is of high quality and have different styles.

4.4 Comparison between diffusion model and styleGAN

In general, diffusion Models and styleGAN represent two distinct approaches to generative modeling. The differences between these two models are listed as follow:

1. Generation Mechanism:
   styleGAN: Based on the GAN framework, styleGAN employs a generator and a discriminator as its core components. The generator's role is to create new images, while the discriminator evaluates whether these images are real or fake, produced by the generator. Through adversarial training, where both networks compete against each other, the generator learns over time how to generate realistic-looking images.
   Diffusion Models: Utilize a step-by-step denoising process. They start by adding noise to the data and then learn to reverse this process, reconstructing the original data from pure noise. This is an iterative procedure where some noise is removed at each iteration until a clear image is eventually reconstructed.
2. Sampling Speed:
   styleGAN: Once trained, generating new samples is very fast, as it requires only a single forward pass through the network to produce results.
   Diffusion Models: Relatively slower when generating samples, as they require multiple steps to gradually remove noise and restore the image.
3. Flexibility and Control:
   styleGAN: Offers high controllability over the style of generated images, allowing users to finely tune specific aspects of the output.
   Diffusion Models: While also providing some level of control, such as conditional diffusion models that can generate specific types of images based on conditions, their control is not as fine-grained as styleGAN's.

In this cat body generation task, styleGAN outperformes diffusion model. The reasons why they have different perfermance on cat body generation are as follow:

1. The application of diffusion models in specific fields may require more adjustments and optimizations.
2. StyleGAN is specially designed with features such as multi-resolution input and adaptive instance normalization (AdaIN), which help capture and control the style features of images, potentially making it perform better on certain tasks.