A4 Style GAN

Part 1: Sampling images with the StyleGAN2 model

```
[227] 1  G = Generator_Wrapper()
      2  images, latents = G.generate_image(num_images = 8)
```

```
1  G.tensor2image(images)
```

Part 2: Linear interpolation of latent codes

```
1  #TODO: generate two images and visualize
2
3  images, interpolation_latents = G.generate_image(num_images = 2)
4  G.tensor2image(images)
```

```
[234] 1  #interpolate two latents
      2  def interpolate_latents(latents,interpolate_scale):
      3      #TODO: interpolate between two latents generated above. interpolate_scale (x) is the interpolation coefficient
      4      new_latent = latents[0] * interpolate_scale + latents[1] * (1 - interpolate_scale)
      5      new_latent = new_latent.unsqueeze(0)
      6      image_new, latent_new = G.generate_image(input=new_latent,input_is_latent=True)
      7      image_new = G.tensor2image(image_new)
      8      return image_new
```

```
1  #make slider and visualize
2  interact(interpolate_latents, latents=fixed(interpolation_latents),interpolate_scale=widgets.FloatSlider(0.5, min=0, max=1., step=0.05))
```

interpolate... 0.45

<function __main__.interpolate_latents(latents, interpolate_scale)>

```
#make slider and visualize
interact(interpolate_latents, latents=fixed(interpolation_latents),interpolate_scale=widgets.FloatSlider(0.5, min=0, max=1., step=0.05))
```

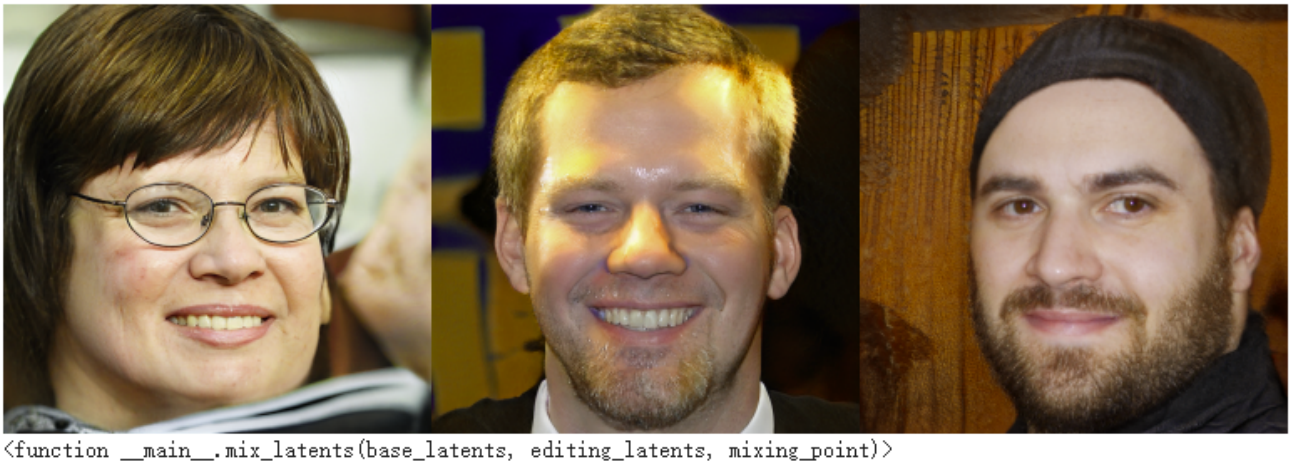
interpolate... 0.65

<function __main__.interpolate_latents(latents, interpolate_scale)>

Part 3:Latent Mixing

```
interact(mix_latents,
         base_latents = fixed(base_latents),
         editing_latents = fixed(editing_latents),
         mixing_point=widgets.IntSlider(5, min=0, max=18, step=1))
```

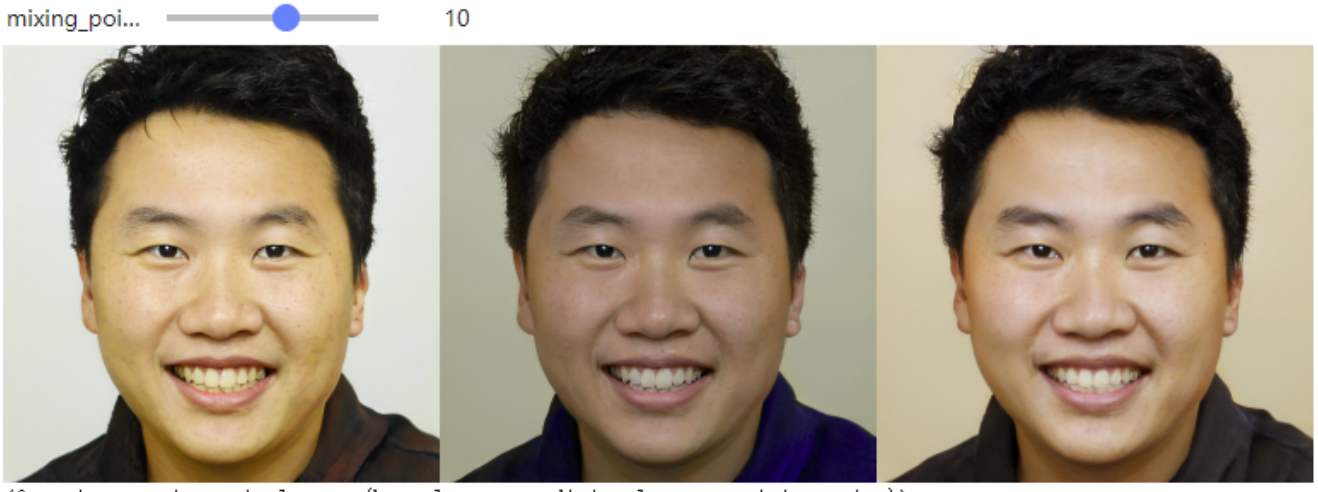
mixing_poi... 2



<function __main__.mix_latents(base_latents, editing_latents, mixing_point)>

mixing_poi... 5

<function __main__.mix_latents(base_latents, editing_latents, mixing_point)>



<function __main__.mix_latents(base_latents, editing_latents, mixing_point)>

If the blend point is lower, the overall structure of the base image is preserved, while only the more subtle features (such as texture and color) are replaced by the edited image. Conversely, if the blend point is higher, even more underlying image features, including overall structure, will be replaced by the edited image

The architecture of StyleGAN2 is layered, with different layers responsible for generating different aspects of the image. At lower levels, the network learns to generate coarse features of the image, such as the overall structure and shape, while at higher levels, it learns finer features, such as texture and color details.

Part 4

Only use MSE

```
1  # Use MSE
2  generated_image,_ = styleGAN_inverse(aligned_tensor)
```

```
Step 0, Loss: 0.1276504248380661
Step 100, Loss: 0.05611291900277138
Step 200, Loss: 0.050302714109420776
Step 300, Loss: 0.0475178137421608
Step 400, Loss: 0.04612504690885544
Step 500, Loss: 0.039183083921670914
Step 600, Loss: 0.03759707883000374
Step 700, Loss: 0.03689045086503029
Step 800, Loss: 0.03592357784509659
Step 900, Loss: 0.03619703650474548
Step 999, Loss: 0.03578568994998932
```

```
[30]  1  G.tensor2image(generated_image.detach().cpu())
```

Use W+ space

```
[31]  1  # Use w+
      2  num_steps = 1000
      3  use_W_plus = True
      4  generated_image_w_plus,_ = styleGAN_inverse(aligned_tensor)
```

```
Step 0, Loss: 0.12800434231758118
Step 100, Loss: 0.044230058789253235
Step 200, Loss: 0.03837491571903229
Step 300, Loss: 0.03624989837408066
Step 400, Loss: 0.03406738489866257
Step 500, Loss: 0.03387259691953659
Step 600, Loss: 0.033543046563863754
Step 700, Loss: 0.03266698867082596
Step 800, Loss: 0.03158900886774063
Step 900, Loss: 0.030259646475315094
Step 999, Loss: 0.03029073029756546
```

```
1  G.tensor2image(generated_image_w_plus.detach().cpu())
```

Using LPIPS

```
# Use MSE + LPIPS
num_steps = 1500
lambda_lpips = 2
use_W_plus = True
use_LPIPS = True
generated_image_LPIPS,generated_latents_LPIPS = styleGAN_inverse(aligned_tensor,lr=0.01)
```

```
Step 0, Loss: 1.6527119874954224
Step 100, Loss: 0.6051868796348572
Step 200, Loss: 0.558060884475708
Step 300, Loss: 0.5591106414794922
Step 400, Loss: 0.5792225003242493
Step 500, Loss: 0.590905487537384
Step 600, Loss: 0.5930681824684143
Step 700, Loss: 0.6002585291862488
Step 800, Loss: 0.6041116118431091
Step 900, Loss: 0.5978984236717224
Step 1000, Loss: 0.5983676910400391
Step 1100, Loss: 0.5942931175231934
Step 1200, Loss: 0.5845506191253662
Step 1300, Loss: 0.5767844915390015
Step 1400, Loss: 0.5673852562904358
Step 1499, Loss: 0.5656790137290955
```

[45]
```
G.tensor2image(generated_image_LPIPS.detach().cpu())
```

Bonus

```
[61]  1  # Bonus
      2  base_image, base_latents = G.generate_image(num_images = 1)
      3  latents = torch.cat([generated_latents_LPIPS,base_latents],dim=0).detach()
      4  assert latents.shape != ([2, 18, 512])
      5  interact(interpolate_latents, latents=fixed(latents),interpolate_scale=widgets.FloatSlider(0.5, min=0, max=1., step=0.05))
```

interpolate... 0.75

<function __main__.interpolate_latents(latents, interpolate_scale)>

```
1  editing_images, editing_latents = G.generate_image(num_images = 3)
2  interact(mix_latents,
3           base_latents = fixed(generated_latents_LPIPS),
4           editing_latents = fixed(editing_latents),
5           mixing_point=widgets.IntSlider(5, min=0, max=18, step=1))
```

mixing_poi... 10

<function __main__.mix_latents(base_latents, editing_latents, mixing_point)>