实现功能

- 图像生成功能：集成到WebUI中，输入seed，输出一张图像
- 图像编辑功能：未集成到WebUI中，可以生成编辑前后的图像

图像生成功能

代码实现

按照tutorial给出的代码定义Generator并进行训练，将参数保存下来。

将图像生成功能集成到WebUI中主要有以下几个步骤：

1. 导入定义好的Generator类并创建Generator对象
2. 将torch种子设定为传入的种子
3. 随机采样一个长度为100的向量z
4. 将向量z传入Generator类并得到图像
5. 对图像进行处理，并展示

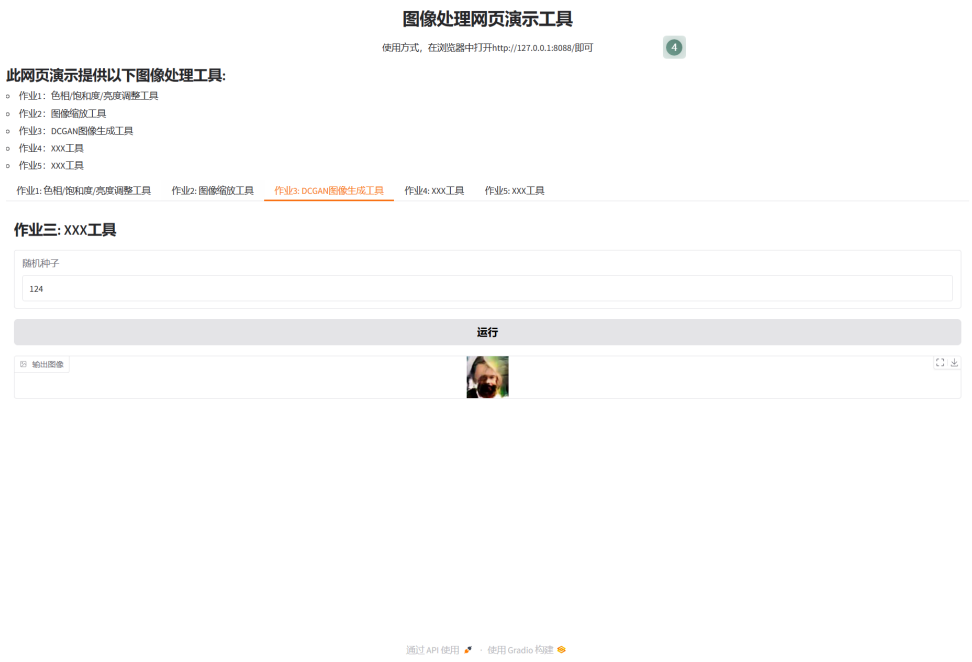
functions.py中的代码如下：

```
    try:
        seed = int(seed)
    except:
        raise gr.Error('输入错误：种子必须为整数', duration=5)

    # 设置种子
    torch.manual_seed(seed)
    # 创建generator
    netG = Generator(0).to("cpu")
    # 加载权重
    netG.load_state_dict(torch.load("./checkpoint/netG.pth", map_location="cpu",
weights_only=False))
    # 生成图像
    with torch.no_grad():
        image = netG(torch.randn(1, 100, 1, 1))
    # 转为numpy
    image = image.cpu().detach().numpy() # [1, 3, 64, 64]
    # 转为8bit图像
    image = (image.squeeze().transpose(1, 2, 0) * 127.5 + 127.5).astype(np.uint8)

    return image
```

界面及效果



图像编辑功能

按照作业描述中给出的提示，我实现了图像编辑功能。

首先仍然需要导入Generate，创建对象，并加载模型参数。

接下来先生成64个随机噪声向量，并生成对应的图像：

```
# 第一步：生成64个随机噪声向量
num_samples = 64
nz = 100  # 潜在向量z的维度大小
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
noise = torch.randn(num_samples, nz, 1, 1, device=device)

# 使用生成器生成对应的图像
with torch.no_grad():
    fake_images = netG(noise).detach().cpu()
grid = vutils.make_grid(fake_images, padding=2, normalize=True)
plt.figure(figsize=(10, 10))
plt.imshow(np.transpose(grid, (1, 2, 0)))
plt.axis("off")
plt.show()
```

接下来我手动选择了这64个图像中具有相反笑容程度的图像：

```
# 第二步：选择具有相反属性的图像，并获取其噪声向量
# 例如，indices_pos是具有属性+A的图像索引
# indices_neg是具有属性-A的图像索引
indices_pos = [36, 10, 49, 32]  # 替换为实际的索引
indices_neg = [27, 28, 29, 30]  # 替换为实际的索引

noise_pos = noise[indices_pos]
noise_neg = noise[indices_neg]
```

求平均偏移向量并保存：

```
vA = noise_pos.mean(dim = 0) - noise_neg.mean(dim = 0)
torch.save(vA, 'smile.pt')
```

最后查看图像编辑的效果：

```
# 生成原始向量和编辑后向量
z = torch.randn(1, nz, 1, 1, device=device)
edited_z = z + vA.unsqueeze(0)

with torch.no_grad():
    image_z = netG(z).detach().cpu()
    image_edited_z = netG(edited_z).detach().cpu()

plt.figure(figsize=(10, 10))
plt.subplot(1, 2, 1)
plt.imshow(np.transpose(image_z[0], (1, 2, 0)))
plt.axis("off")
plt.title("Original Image")

plt.subplot(1, 2, 2)
plt.imshow(np.transpose(image_edited_z[0], (1, 2, 0)))
plt.axis("off")
plt.title("Edited Image")
```

最终效果如下，比较明显地改变了笑的程度。