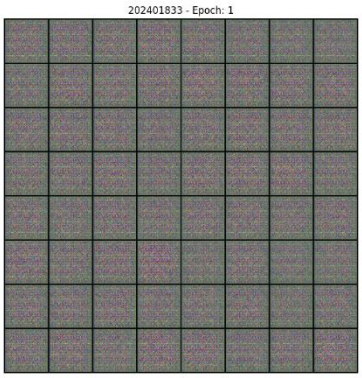
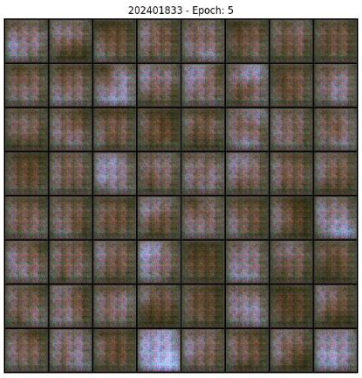
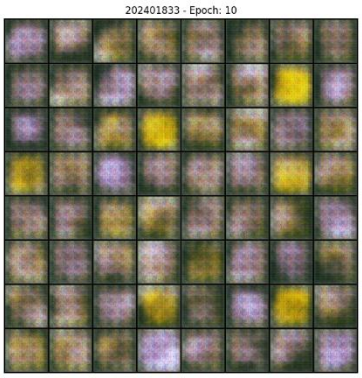
HW07

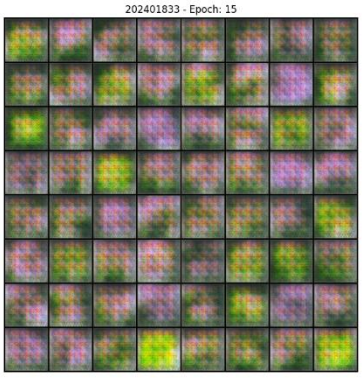
202401833 신해솔

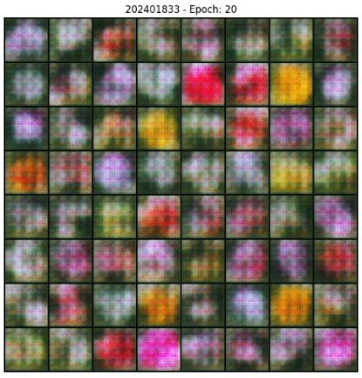
0. 수행 결과

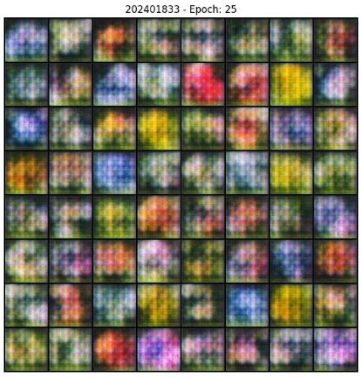


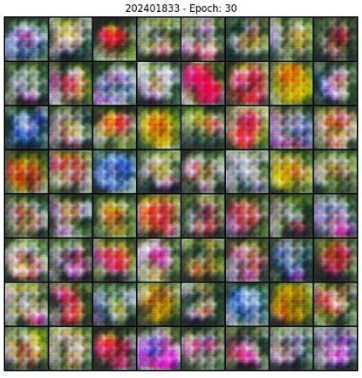


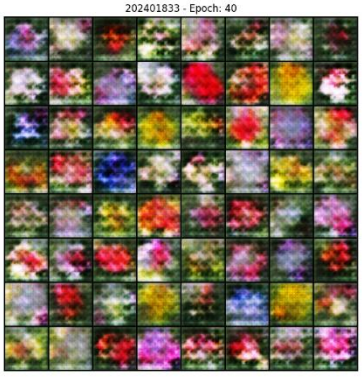


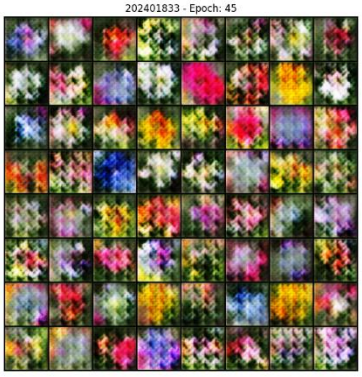


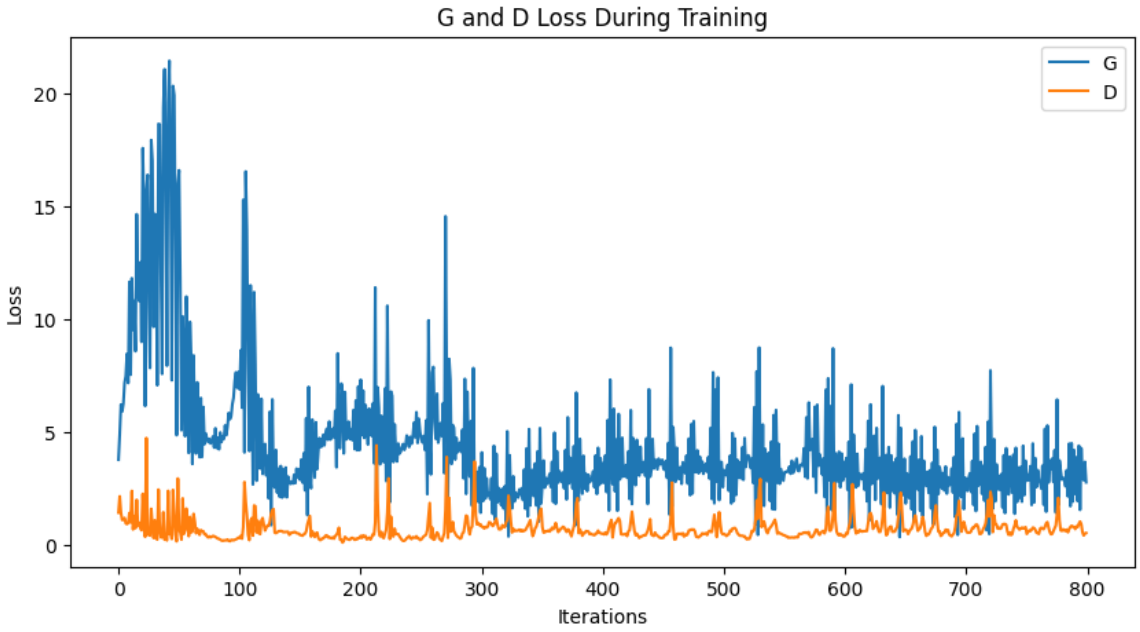












1. 드라이브 마운트 및 파일 압축 해제

```
from google.colab import drive
drive.mount('/content/drive')
```

```
ive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
!tar -xvzf /content/drive/MyDrive/102flowers.tgz -C ./data
```

2. 필요 라이브러리 import 및 기초 설정

```
import numpy as np
import torch
import torch.nn as nn
import torchvision.utils as vutils
import torchvision.transforms as transforms
import torchvision.datasets as datasets
from torch.utils.data import DataLoader

import matplotlib.pyplot as plt
```

```
IMG_SIZE = 64
nc = 3
nz = 100
ngf = 64
ndf = 64
```

3. 데이터셋 로드 및 데이터로더 설정

```
data_root = "./data"

dataset = datasets.ImageFolder(root=data_root, transform=transforms.Compose([
    transforms.Resize(IMG_SIZE),
    transforms.CenterCrop(IMG_SIZE),
    transforms.ToTensor(),
    transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5])
]))

BATCH_SIZE = 512
dataloader = DataLoader(dataset, batch_size=BATCH_SIZE, shuffle=True)
```

4. 이미지 출력 과정을 간소화하는 함수 정의

```
def show_images(images, title):
    plt.figure(figsize=(8, 8))
    plt.axis("off")
    plt.title(title)
    plt.imshow(
        np.transpose(
            vutils.make_grid(
                images[:64],
                padding=2,
                normalize=True).cpu(),
            (1, 2, 0)
        ))
    plt.show()


sample_batch = next(iter(dataloader))
show_images(sample_batch[0], title="Training Images")
```

5. 생성자 정의

```
class Generator(nn.Module):
    def __init__(self):
        super().__init__()
        self.main = nn.Sequential(
            nn.ConvTranspose2d(nz, ngf*8, 4, 1, 0, bias=False),
            nn.BatchNorm2d(ngf*8),
            nn.ReLU(True),

            nn.ConvTranspose2d(ngf*8, ngf*4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 4),
            nn.ReLU(True),

            nn.ConvTranspose2d(ngf*4, ngf*2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 2),
            nn.ReLU(True),

            nn.ConvTranspose2d(ngf*2, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(True),
            nn.ConvTranspose2d(ngf, nc, 4, 2, 1, bias=False),
            nn.Tanh()
        )

    def forward(self, x):
        return self.main(x)
```

6. 식별자 정의

```
class Discriminator(nn.Module):
    def __init__(self):
        super().__init__()
        self.main = nn.Sequential(
            nn.Conv2d(nc, ndf, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(ndf, ndf*2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 2),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(ndf*2, ndf*4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 4),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(ndf*4, ndf*8, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 8),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(ndf*8, 1, 4, 1, 0, bias=False),
            nn.Sigmoid()
        )

    def forward(self, x):
        return self.main(x)
```

7. 가중치 초기화 함수 정의 및 gpu 사용 설정

```
def weights_init(m):
    class_name = m.__class__.__name__
    if class_name.find("Conv") != -1:
        m.weight.data.normal_(0.0, 0.02)
    elif class_name.find("BatchNorm") != -1:
        m.weight.data.normal_(1.0, 0.02)
        m.bias.data.fill_(0)
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")
```

8. 모델 gpu로 이동 및 모델 확인

```
G = Generator().to(device)
G.apply(weights_init)
print(G)

D = Discriminator().to(device)
D.apply(weights_init)
print(D)
```

하이퍼 파라미터 설정 및 학습 기초 설정

```
criterion = nn.BCELoss()

d_optimizer = torch.optim.Adam(D.parameters(), lr=0.0002, betas=(0.5, 0.999))
g_optimizer = torch.optim.Adam(G.parameters(), lr=0.0002, betas=(0.5, 0.999))
```

```
G_losses = []
D_losses = []

fixed_noise = torch.randn(64, nz, 1, 1).to(device)
```

9. 모델 학습

```
NUM_EPOCHS = 50
for epoch in range(NUM_EPOCHS):
    d_loss = 0
    g_loss = 0
    for real_images, _ in dataloader:
        real_images = real_images.to(device)

        D.zero_grad()
        b_size = real_images.size(0)

        real_labels = torch.ones((b_size,)).to(device)

        output = D(real_images).view(-1)
        d_loss_real = criterion(output, real_labels)

        z = torch.randn(b_size, nz, 1, 1).to(device)
        fake_labels = torch.zeros((b_size,)).to(device)
        fake_images = G(z)

        output = D(fake_images.detach()).view(-1)
        d_loss_fake = criterion(output, fake_labels)

        d_loss = d_loss_real + d_loss_fake
        d_loss.backward()
        d_optimizer.step()

        G.zero_grad()

        output = D(fake_images).view(-1)
        g_loss = criterion(output, real_labels)
        g_loss.backward()
        g_optimizer.step()

        G_losses.append(g_loss.item())
        D_losses.append(d_loss.item())

    print(f"[{epoch + 1}/{NUM_EPOCHS}], d_loss: {d_loss.item():.4f}, g_loss: {g_loss.item():.4f}")

    if (epoch + 1) % 5 == 0 or epoch == 0:
        with torch.no_grad():
            generated_images = G(fixed_noise).detach().cpu()
        show_images(generated_images, title=f"202401833 - Epoch: {epoch + 1}")
```

10. 학습 과정에서의 손실 출력

```
plt.figure(figsize=(10, 5))
plt.title("G and D Loss During Training")
plt.plot(G_losses, label="G")
plt.plot(D_losses, label="D")
plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.legend()
plt.show()
```