ณสิต ผลัญชัย 65010273

วิวัตร เตชะโกศล 65011001

Lab 9

```python
class Generator(nn.Module):
    def __init__(self, z_dim=100, im_ch=3, hidden_dim=1024):
        super(Generator, self).__init__()
        self.gen = nn.Sequential(
            self.conv_block(z_dim, hidden_dim, kernel_size=4, stride=1, padding=0),
            self.conv_block(hidden_dim, hidden_dim//2, kernel_size=4, stride=2, padding=1),
            self.conv_block(hidden_dim//2, hidden_dim//4, kernel_size=4, stride=2, padding=1),
            self.conv_block(hidden_dim//4, hidden_dim//8, kernel_size=4, stride=2, padding=1),
            self.conv_block(hidden_dim//8, hidden_dim//16, kernel_size=4, stride=2, padding=1),
            self.conv_block(hidden_dim//16, im_ch, kernel_size=4, stride=2, padding=1, final_layer=True)
        )

    def conv_block(self, in_ch, out_ch, kernel_size=4, stride=2, padding=1, final_layer=False):
        if not final_layer:
            return nn.Sequential(
                nn.ConvTranspose2d(in_channels=in_ch, out_channels=out_ch, kernel_size=kernel_size, stride=stride, padding=padding),
                nn.BatchNorm2d(out_ch),
                nn.ReLU(0.2)
            )
        else:
            return nn.Sequential(
                nn.ConvTranspose2d(in_channels=in_ch, out_channels=out_ch, kernel_size=kernel_size, stride=stride, padding=padding),
                nn.Tanh()
            )
    def forward(self, noise):
        noise = noise.view(noise.size(0), noise.size(1), 1, 1)
        return self.gen(noise)
```

สร้าง Class Generator() ที่จะใช้สร้างรูปภาพจาก noise โดยเราจะทำการ up sampling input จาก 1 x 1 จนกลายเป็น output ขนาด 128 x 128 ที่มี 3 channels โดยใน layer ปกติเราจะทำการ batch norm และ apply ReLU แต่ใน final layer เราจะใช้ tanh เพื่อให้ค่าแต่ละ pixel อยู่ใน range (-1, 1)

```python
class Discriminator(nn.Module):
    def __init__(self,im_ch=3, hidden_dim=32):
        super(Discriminator, self).__init__()
        self.disc = nn.Sequential(
            self.conv_block(im_ch, hidden_dim, kernel_size=4, stride=2, padding=1),
            self.conv_block(hidden_dim, hidden_dim*2, kernel_size=4, stride=2, padding=1),
            self.conv_block(hidden_dim*2, hidden_dim*4, kernel_size=4, stride=2, padding=1),
            self.conv_block(hidden_dim*4, hidden_dim*8, kernel_size=4, stride=2, padding=1),
            self.conv_block(hidden_dim*8, 1, kernel_size=4, stride=2, padding=0, final_layer=True)
        )

    def conv_block(self, in_ch, out_ch, kernel_size=4, stride=2, padding=1, final_layer=False):
        if not final_layer:
            return nn.Sequential(
                nn.Conv2d(in_channels=in_ch, out_channels=out_ch, kernel_size=kernel_size, stride=stride, padding=padding),
                nn.BatchNorm2d(out_ch),
                nn.LeakyReLU(0.2)
            )
        else:
            return nn.Sequential(
                nn.Conv2d(in_channels=in_ch, out_channels=out_ch, kernel_size=kernel_size, stride=stride, padding=padding),
                nn.Sigmoid()
            )
```

```python
def forward(self, image):
    return self.disc(image).view(-1)
```

Class Discriminator() คือตัวที่จะคอยตรวจว่าภาพที่ Generator สร้างมานั้นเป็นภาพจริงหรือปลอม โดยจะทำการรับ input เข้ามาเป็นภาพขนาด 128 x 128 และทำการ Down sampling จนเหลือขนาด 1 x 1 โดยในแต่ละครั้งที่ทำการ Down sampling ก็จะมีการ Batch norm และ apply Leaky ReLU แต่ในส่วนของ Final layer นั้นจะใช้เป็น Sigmoid เนื่องจากผลลัพธ์ที่ต้องการคือค่าความน่าจะเป็นว่าภาพที่ input เข้ามานั้นเป็นภาพจริงหรือภาพปลอม ซึ่ง output นี้จะถูกใช้ในการปรับ parameter ในการ train ของทั้ง Generator และ Discriminator ไปพร้อมๆกัน

```python
def get_noise(n_sample, z_dim, device='cuda'):
    return torch.randn(n_sample, z_dim, device=device)
```

Function get_noise() จะทำการ return noise tensor ออกมาตามจำนวน n_sample ที่ใส่เข้าไปโดยแต่ละ sample จะมีมิติตาม z_dim

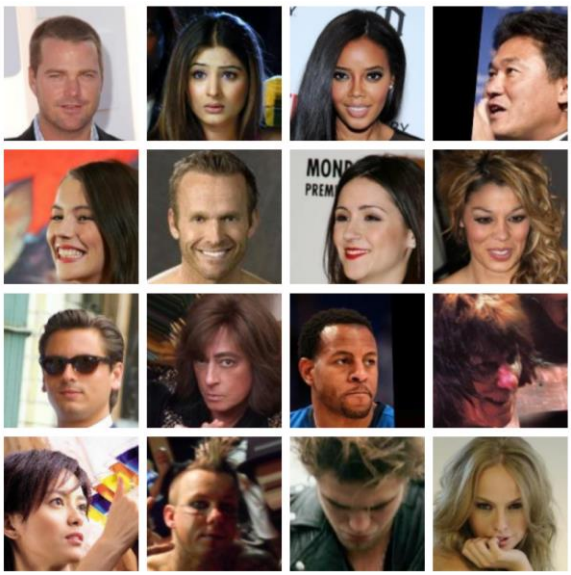
```python
n_sample = 25
z_dim = 100
noise = get_noise(n_sample, z_dim)
assert noise.shape == (n_sample, z_dim), f"Expected shape {(n_sample, z_dim)}, but got {noise.shape}"

noise_cpu = get_noise(n_sample, z_dim, device='cpu')
assert noise_cpu.device.type == 'cpu', f"Expected tensor to be on 'cpu', but got {noise_cpu.device.type}"
assert noise.dtype == torch.float32, f"Expected dtype to be torch.float32, but got {noise.dtype}"
```

✓ 0.1s

```
device = 'cuda' if torch.cuda.is_available() else 'cpu'

path = r'C:\Users\Nickv\Documents\ImageProcessing\Week5\images'

transform = transforms.Compose([
    transforms.Resize(128),
    transforms.CenterCrop(128),
    transforms.ToTensor(),
    transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5]),
])

generator = Generator().to(device)
discriminator = Discriminator().to(device)

dataset = datasets.ImageFolder(root=path, transform=transform)
dataloader = DataLoader(dataset, batch_size=16, shuffle=True)

gen_opt = torch.optim.Adam(generator.parameters(), lr=0.001, betas=(0.5, 0.999))
disc_opt = torch.optim.Adam(discriminator.parameters(), lr=0.001, betas=(0.5, 0.999))
```

ทำการเตรียมข้อมูลเพื่อการ train โดยจะมีการ transforms ภาพด้วยการ resize เป็น 128 และเปลี่ยนภาพเป็น tensor ก่อนจะทำการ normalize ทำการโหลด dataset โดยใช้ batch size 16

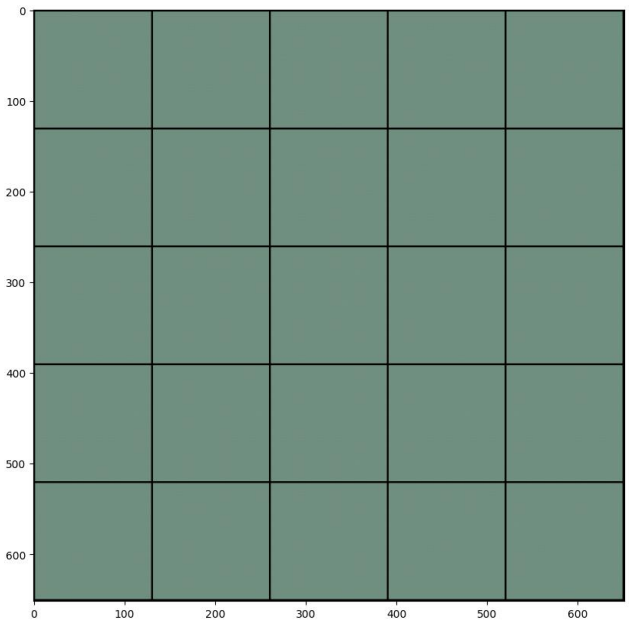
```
images, labels = next(iter(dataloader))
imshow_grid(images)
```



ทำการแสดงผลรูปภาพ

```python
with torch.no_grad():
    generator.eval()
    fake_images = generator(noise).cuda()

def show_images(images, nrow=5):
    images = images.cpu()
    images = (images + 1) / 2
    grid_img = torchvision.utils.make_grid(images, nrow=nrow)
    plt.figure(figsize=(10, 10))
    plt.imshow(np.transpose(grid_img.numpy(), (1, 2, 0)))
    plt.show()

show_images(fake_images)
```

ทำการลองให้ Generator generate ภาพจาก noise โดย



```python
### START CODE HERE ###
def train(generator, discriminator, gen_opt, disc_opt, criterion, dataloader, test_noise, z_dim, epochs=10, writer=None,
    print("🤖 Training on", device)
    generator.to(device)
    discriminator.to(device)

    for epoch in range(epochs):
        gen_loss_avg = 0
        disc_loss_avg = 0
        num_batches = len(dataloader)
        train_bar = tqdm(dataloader, desc=f'🚀 Training Epoch [{epoch+1}/{epochs}]', unit='batch')

        for real_images, _ in train_bar:
            real_images = real_images.to(device)
            batch_size = real_images.size(0)

            disc_opt.zero_grad()

            noise = torch.randn(batch_size, z_dim, device=device)
            fake_images = generator(noise)

            real_pred = discriminator(real_images)
            fake_pred = discriminator(fake_images.detach())
            real_loss = criterion(real_pred, torch.ones_like(real_pred))
            fake_loss = criterion(fake_pred, torch.zeros_like(fake_pred))
            disc_loss = (real_loss + fake_loss) / 2

            disc_loss.backward()
            disc_opt.step()

            gen_opt.zero_grad()

            fake_pred = discriminator(fake_images)
            gen_loss = criterion(fake_pred, torch.ones_like(fake_pred))

            gen_loss.backward()
            gen_opt.step()

            gen_loss_avg += gen_loss.item() / num_batches
            disc_loss_avg += disc_loss.item() / num_batches

        if writer:
            writer.add_scalar('Generator Loss', gen_loss_avg, epoch)
            writer.add_scalar('Discriminator Loss', disc_loss_avg, epoch)
```

Function train() จะ train ทั้ง Discriminator และ Generator พร้อมๆกัน โดย Discriminator จะทำการ predict ภาพจาก dataset และ ภาพปลอมจาก Generator จากนั้นนำเข้า BCE Loss เพื่อนำมาปรับ weights ต่อไป ส่วนของ Generator จะใช้แค่ BCE Loss ที่ Discriminator predict ภาพปลอมที่ Generator สร้างขึ้น

```
    with torch.no_grad():
        generator.eval()
        fake_images_test = generator(test_noise.to(device))

        img_grid = torchvision.utils.make_grid(fake_images_test, nrow=5)
        if writer:
            writer.add_image('Generated Images', img_grid, epoch)

        generator.train()

    if checkpoint_path:
        torch.save({
            'generator_state_dict': generator.state_dict(),
            'discriminator_state_dict': discriminator.state_dict(),
            'gen_opt_state_dict': gen_opt.state_dict(),
            'disc_opt_state_dict': disc_opt.state_dict(),
            'epoch': epoch
        }, f"{checkpoint_path}/gan_checkpoint_epoch_{epoch}.pth")

    print(f'Epoch {epoch+1}, Generator loss: {gen_loss_avg:.4f}, Discriminator loss: {disc_loss_avg:.4f}')
```

โดยจะมีการเก็บข้อมูลกราฟ Generator Loss และ Discriminator Loss รวมถึงพัฒนาการของภาพที่ถูก Generator สร้างขึ้นใน Tensor Board และมีการ Save checkpoint ทุกๆ epoch

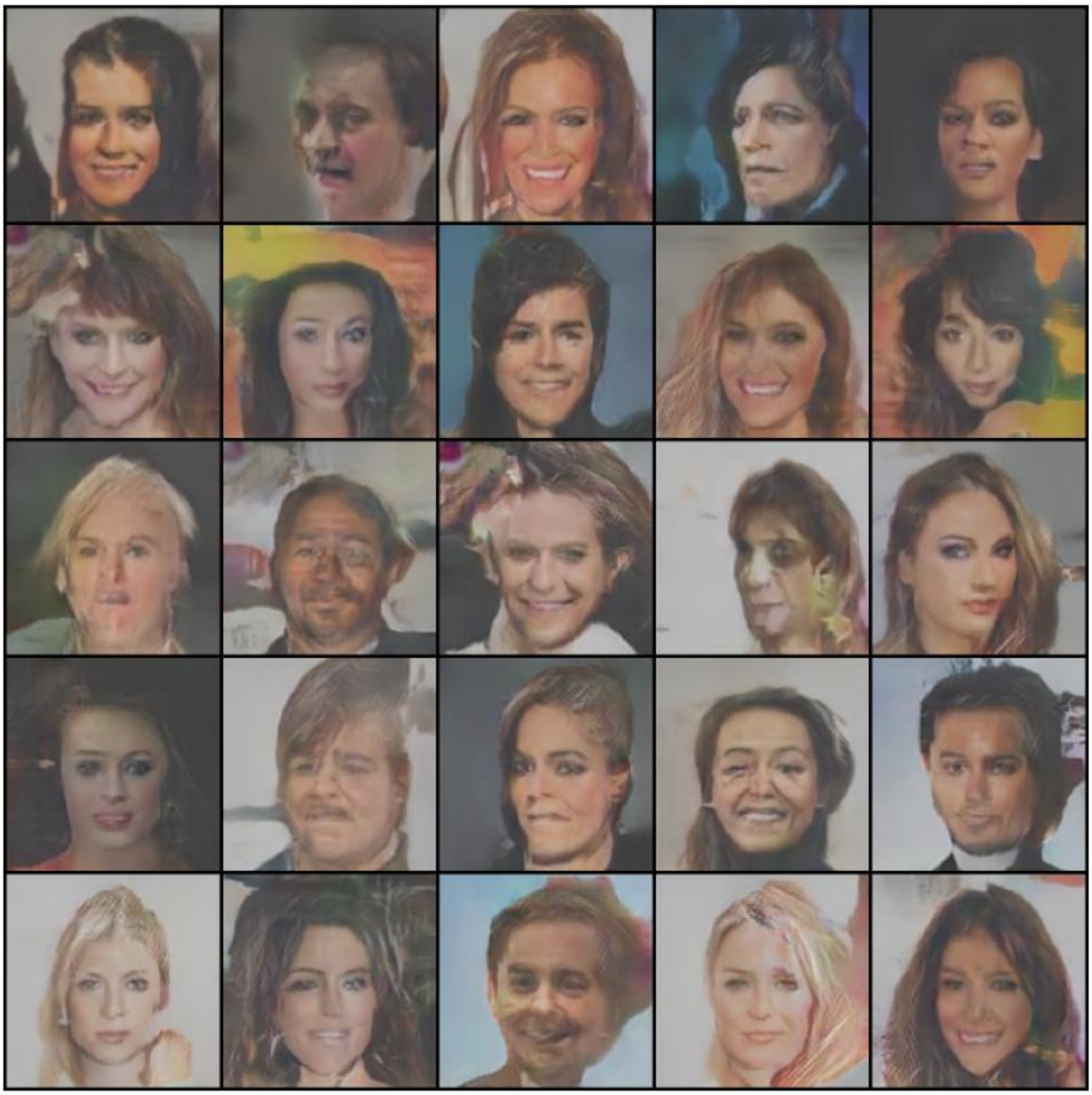
```
device = 'cuda' if torch.cuda.is_available() else 'cpu'

generator = Generator(z_dim=100).to(device)
discriminator = Discriminator(im_ch=3).to(device)

gen_opt = torch.optim.Adam(generator.parameters(), lr=0.0001, betas=(0.5, 0.999))
disc_opt = torch.optim.Adam(discriminator.parameters(), lr=0.0001, betas=(0.5, 0.999))

criterion = nn.BCELoss()

dataloader = DataLoader(dataset, batch_size=16, shuffle=True)

test_noise = torch.randn(25, 100).to(device)

writer = SummaryWriter(log_dir='runs/gan_experiments')

train(generator, discriminator, gen_opt, disc_opt, criterion, dataloader, test_noise, z_di

writer.close()
```

ทำการ Train GAN model โดยสร้าง Generator และ Discriminator ซึ่งเราเลือกใช้ optimizer เป็น Adam โดยตั้งค่า Learning rate ที่ 0.0001 beta1=0.5, beta2=0.999 และใช้ criterion เป็น BCE Loss โดย train ทั้งหมด 30 epochs

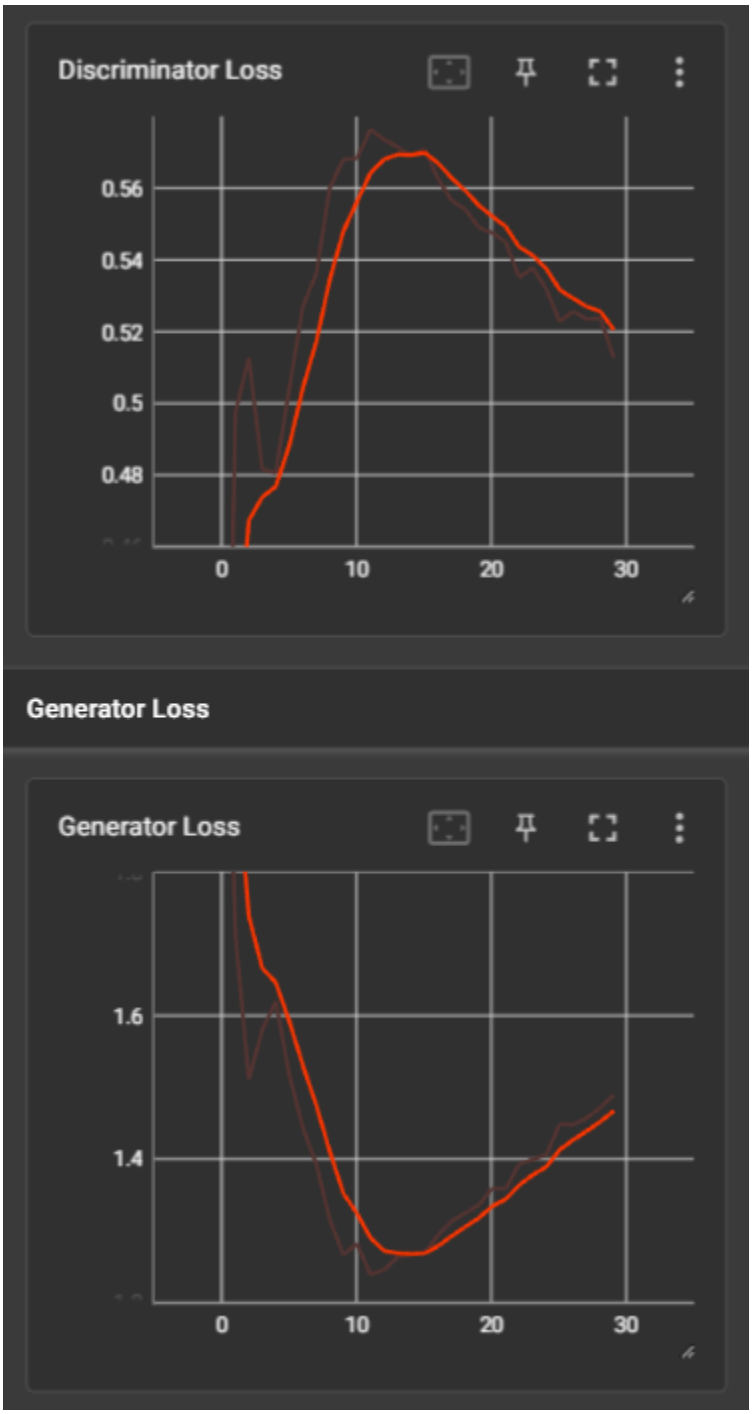
```
Training on cuda
Training Epoch [1/30]: 100%|██████████| 1875/1875 [04:04<00:00,  7.68batch/s]
Epoch 1, Generator loss: 2.4098, Discriminator loss: 0.2920
Training Epoch [2/30]: 100%|██████████| 1875/1875 [01:26<00:00, 21.67batch/s]
Epoch 2, Generator loss: 1.7146, Discriminator loss: 0.4972
Training Epoch [3/30]: 100%|██████████| 1875/1875 [01:26<00:00, 21.65batch/s]
Epoch 3, Generator loss: 1.5112, Discriminator loss: 0.5127
Training Epoch [4/30]: 100%|██████████| 1875/1875 [01:25<00:00, 21.99batch/s]
Epoch 4, Generator loss: 1.5805, Discriminator loss: 0.4816
Training Epoch [5/30]: 100%|██████████| 1875/1875 [01:27<00:00, 21.50batch/s]
Epoch 5, Generator loss: 1.6198, Discriminator loss: 0.4807
Training Epoch [6/30]: 100%|██████████| 1875/1875 [01:28<00:00, 21.17batch/s]
Epoch 6, Generator loss: 1.5165, Discriminator loss: 0.5043
Training Epoch [7/30]: 100%|██████████| 1875/1875 [01:24<00:00, 22.27batch/s]
Epoch 7, Generator loss: 1.4435, Discriminator loss: 0.5270
Training Epoch [8/30]: 100%|██████████| 1875/1875 [01:25<00:00, 21.90batch/s]
Epoch 8, Generator loss: 1.3924, Discriminator loss: 0.5364
Training Epoch [9/30]: 100%|██████████| 1875/1875 [01:26<00:00, 21.60batch/s]
Epoch 9, Generator loss: 1.3152, Discriminator loss: 0.5602
Training Epoch [10/30]: 100%|██████████| 1875/1875 [01:26<00:00, 21.77batch/s]
```

```python
def generate_and_display_grid(generator, z_dim, grid_size=(5, 5), device='cuda'):
    generator.eval()

    noise = torch.randn(grid_size[0] * grid_size[1], z_dim, device=device)

    with torch.no_grad():
        fake_images = generator(noise).cpu()

    fake_images = (fake_images + 1) / 2

    img_grid = torchvision.utils.make_grid(fake_images, nrow=grid_size[1])

    plt.figure(figsize=(10, 10))
    plt.imshow(img_grid.permute(1, 2, 0))
    plt.axis('off')
    plt.show()

generate_and_display_grid(generator, z_dim=100)
```

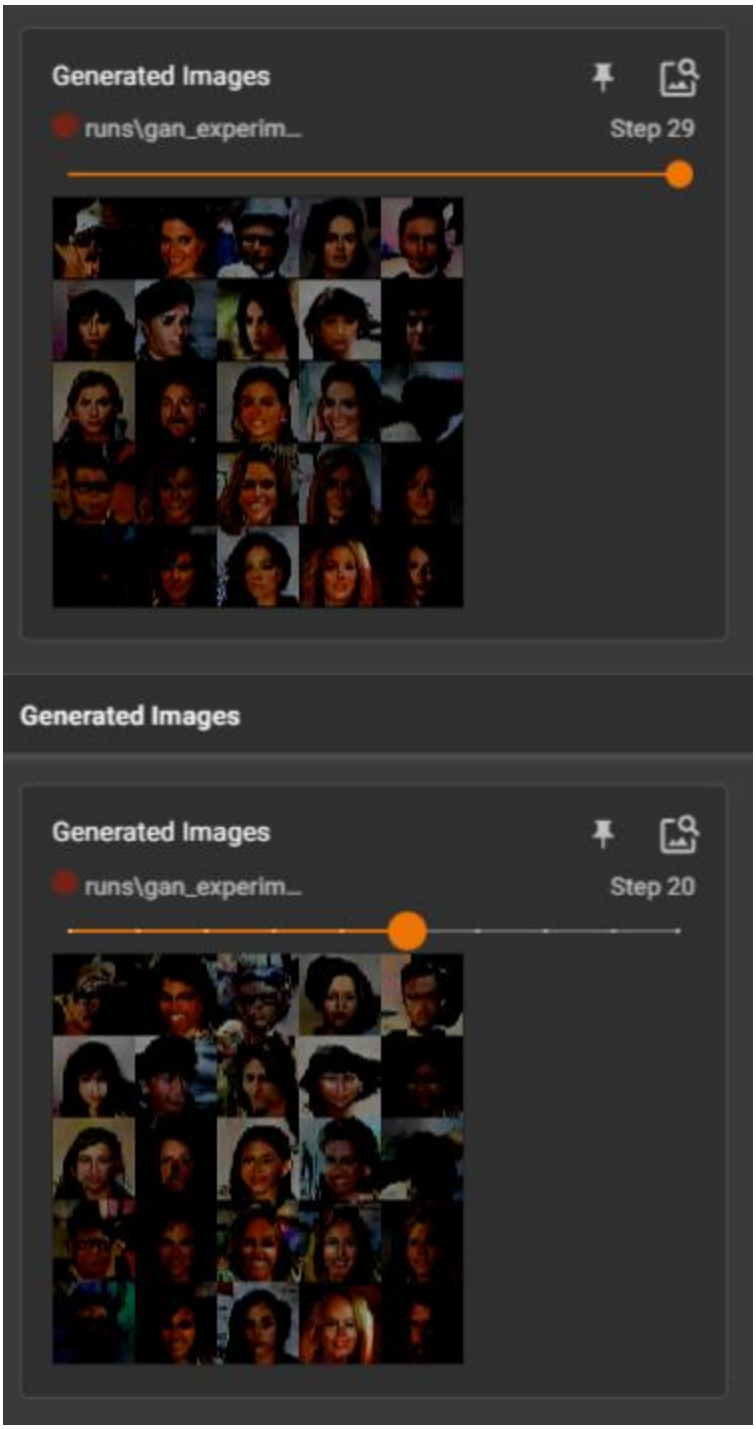
Function generate_and_display_grid() ใช้เพื่อแสดงผลภาพที่ generator สร้างขึ้นโดยทำการสุ่ม noise ขึ้นมาและนำมา input ผ่าน generator model แล้วนำภาพที่ได้มา denormalize



ภาพผลลัพธ์จาก Tensor board




จะเห็นได้ว่า model มีการ overfitting เล็กน้อยหลังผ่าน epoch ที่ 15




Fake images ในแต่ละ epoch โดยมีความชัดเจนมากขึ้นแต่ภาพมีความมืดเนื่องจากไม่ได้ทำการ denormalize ใน training loop