Deep Learning Assignment #2 - GAN

2018062733 컴퓨터소프트웨어학부 윤동빈

1. 코드 설명

(1) GAN

```python
class Generator(nn.Module):
    def __init__(self, ):
        super(Generator, self).__init__()
        self.main = nn.Sequential(
            nn.ConvTranspose2d(100, 64 * 8, 4, 1, 0, bias=False),
            nn.BatchNorm2d(64 * 8),
            nn.ReLU(),
            nn.ConvTranspose2d(64 * 8, 64 * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(64 * 4),
            nn.ReLU(),
            nn.ConvTranspose2d(64 * 4, 64 * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(64 * 2),
            nn.ReLU(),
            nn.ConvTranspose2d(64 * 2, 64, 4, 2, 1, bias=False),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.ConvTranspose2d(64, 3, 4, 2, 1, bias=False),
            nn.Tanh()
        )

    def forward(self, input):
        # input data는 [batch size, 100, 1, 1]의 형태로 주어야합니다.
        return self.main(input)
```

```python
class Discriminator(nn.Module):
    # 모델의 코드는 여기서 작성해주세요

    def __init__(self):
        super(Discriminator, self).__init__()

        self.main = nn.Sequential(
            nn.Conv2d(3, 64, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2),

            nn.Conv2d(64, 64 * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(64*2),
            nn.LeakyReLU(0.2),

            nn.Conv2d(64 * 2, 64 * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(64*4),
            nn.LeakyReLU(0.2),

            nn.Conv2d(64 * 4, 64 * 8, 4, 2, 1, bias=False),
            nn.BatchNorm2d(64 * 8),
            nn.LeakyReLU(0.2),

            nn.Conv2d(64 * 8, 1, 4, 1, 0, bias=False),
            nn.Sigmoid()
        )

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

1) Generator

fake image를 만들어내는 Generator class를 정의하는 부분으로,
nn.ConvTranspose2d()는 input image에 대해 2D transposed convolution을 적용한다.
nn.ReLU()는 activation function인 ReLU 함수를 의미하며, BatchNorm2d()는 Batch Normalization을 적용하는 함수이다.

2) Discriminator

input이 실제 image인지 판별하는 Discriminator class를 정의하는 부분으로,
activation function으로는 leaky ReLU, Sigmoid 함수를 사용한다. BatchNorm2d()는 Batch Normalization을 적용하는 함수이다.

(2) 값 설정 및 학습 코드

```
if __name__ == "__main__":
    data_path = 'training_data/'
    #data_path = '/content/drive/MyDrive/training_data/'

    dataset = datasets.ImageFolder(root=data_path,
                                   transform=transforms.ToTensor()
                                   )

    batch_size = 128
    lr = 0.0002

    train_dataloader = torch.utils.data.DataLoader(dataset, batch_size=batch_size, shuffle=True)

    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

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

    criterion = torch.nn.BCELoss()
    g_optimizer = torch.optim.Adam(generator.parameters(), lr=lr, betas=(0.5, 0.999))
    d_optimizer = torch.optim.Adam(discriminator.parameters(), lr=lr, betas=(0.5, 0.999))
```

batch size=128, learning rate=0.0002, optimizer는 Adam optimizer로 설정한다.
Adam optimizer에 쓰이는 β_1, β_2는 각각 0.5, 0.999로 설정한다.

위에서 정의된 Generator, Discriminator class 인스턴스를 만들고 아래 코드에서 학습하는 데 활용한다.

```
epochs = 50

for epoch in range(epochs):
  for step, batch in enumerate(train_dataloader):
    d_optimizer.zero_grad()

    b_x = batch[0].to(device)
    num_img = b_x.size(0)
    real_label = torch.ones((num_img,)).to(device)
    fake_label = torch.zeros((num_img,)).to(device)

    real_logit = discriminator(b_x).view(-1)
    d_real_loss = criterion(real_logit, real_label)
    d_real_loss.backward()

    z = torch.randn(num_img, 100, 1, 1, requires_grad=False).to(device)
    fake_data = generator(z)
    fake_logit = discriminator(fake_data.detach()).view(-1)
    d_fake_loss = criterion(fake_logit, fake_label)
    d_fake_loss.backward()

    d_optimizer.step()

    g_optimizer.zero_grad()
    fake_logit = discriminator(fake_data).view(-1)
    g_loss = criterion(fake_logit, real_label)
    g_loss.backward()
    g_optimizer.step()
```

주어진 input data에 대해서 총 50(=epochs)회만큼 진행한다.
training data에 대해 Discriminator를 학습한 후, Generator로 fake images를 만든 뒤 이에 대한 Discriminator 판단 결과를 fake label을 기준으로 Discriminator를 학습한다. 그리고 fake images에 대한 Discriminator 판단 결과를 real label을 기준으로 Generator도 학습하면서 1회 학습을 마무리한다.

2. 실험결과

(1) 학습O, epochs=10

▶ FID score : 69.87

(2) 학습O, epochs=50

▶ FID score : 56.08