보고서 : Gan 실습

상태	완료
담당자	이시준

보고서: GAN 모델 실습

1. 서론

GAN(Generative Adversarial Network)은 Ian Goodfellow에 의해 제안된 모델로, 생성기(Generator)와 판별기(Discriminator)라는 두 개의 신경망이 서로 경쟁하면서 학습하는 구조를 갖습니다. 본 보고서는 MNIST 데이터셋을 활용한 GAN 모델의 구현과 학습 과정을 소개합니다.

2. 데이터 셋 준비

MNIST 데이터셋은 손글씨 숫자(0-9)를 포함하는 28x28 픽셀 크기의 이미지 데이터셋입니다. Keras 라이브러리를 통해 손쉽게 로드할 수 있으며, 실습에서는 학습용(train) 데이터셋만 사용됩니다.

```
from tensorflow.keras.datasets import mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train / 255.0
```

3. 모델 정의

3.1 생성기(Generator)

생성기는 랜덤한 노이즈를 입력받아 진짜와 유사한 이미지를 생성합니다. LeakyReLU 활성화 함수를 사용하여 네트워크가 보다 유연하게 학습할 수 있도록 돕습니다.

```
def create_generator():
    model = Sequential()
    model.add(Dense(256, input_dim=100, kernel_initializer=
RandomNormal(stddev=0.02)))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(512))
```

```
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(1024))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(28*28, activation='tanh'))
    model.compile(loss='binary_crossentropy', optimizer=Ada
m(0.0002, 0.5))
    return model
```

3.2 판별기(Discriminator)

판별기는 입력받은 이미지가 진짜인지 가짜인지 구분합니다. Binary Crossentropy 손실을 사용하여 진짜/가짜 여부를 학습합니다.

```
def create_discriminator():
    model = Sequential()
    model.add(Dense(1024, input_dim=28*28, kernel_initializ
er=RandomNormal(stddev=0.02)))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.3))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.3))
    model.add(Dense(256))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.3))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer=Ada
m(0.0002, 0.5))
    return model
```

4. GAN 구성 및 학습

생성기와 판별기를 결합하여 GAN 모델을 구성합니다. 판별기의 가중치 업데이트를 방지하기 위해 `trainable` 속성을 `False`로 설정하고 학습을 진행합니다.

```
def create_gan(generator, discriminator):
    discriminator.trainable = False
    gan_input = Input(shape=(100,))
    x = generator(gan_input)
```

```
    gan_output = discriminator(x)
    gan = Model(gan_input, gan_output)
    gan.compile(loss='binary_crossentropy', optimizer=Adam
(0.0002, 0.5))
    return gan
```

학습 루프

GAN 학습 루프에서는 진짜 이미지와 생성된 이미지를 번갈아 판별기에 입력하여 판별기의 정확도를 높이는 동시에, 생성기가 판별기를 속일 수 있도록 학습합니다.

```
def train_gan(gan, generator, discriminator, epochs=50, bat
ch_size=256):
    batch_count = x_train.shape[0] // batch_size
    for e in range(epochs):
        for _ in range(batch_count):
            noise = np.random.normal(0, 1, size=[batch_siz
e, 100])
            generated_images = generator.predict(noise)

            # Get a random batch of real images
            idx = np.random.randint(0, x_train.shape[0], ba
tch_size)
            real_images = x_train[idx]

            # Train discriminator
            d_loss_real = discriminator.train_on_batch(real
_images, np.ones((batch_size, 1)))
            d_loss_fake = discriminator.train_on_batch(gene
rated_images, np.zeros((batch_size, 1)))
            d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)

            # Train generator
            noise = np.random.normal(0, 1, size=[batch_siz
e, 100])
            g_loss = gan.train_on_batch(noise, np.ones((bat
ch_size, 1)))
```

```
        print(f"Epoch: {e}, D Loss: {d_loss}, G Loss: {g_lo
ss}")
```

5. 결과 및 시각화

학습이 완료된 후, 생성된 이미지를 시각화하여 모델의 성능을 확인할 수 있습니다.

```
def plot_generated_images(generator, n_ex=10, dim=(1, 10),
figsize=(10, 1)):
    noise = np.random.normal(0, 1, size=[n_ex, 100])
    generated_images = generator.predict(noise)
    generated_images = generated_images.reshape(n_ex, 28, 2
8)

    plt.figure(figsize=figsize)
    for i in range(n_ex):
        plt.subplot(dim[0], dim[1], i+1)
        plt.imshow(generated_images[i], interpolation='near
est', cmap='gray')
        plt.axis('off')
    plt.tight_layout()
    plt.show()

plot_generated_images(generator)
```

▼ 6. 배운점

1. GAN의 기본 개념

- **생성기(Generator)와 판별기(Discriminator)**: GAN은 두 개의 신경망 모델인 생성기와 판별기로 구성되어 있으며, 서로 경쟁하며 학습합니다. 생성기는 가짜 데이터를 생성하여 판별기를 속이는 역할을 하고, 판별기는 입력 데이터가 진짜인지 가짜인지 구분하는 역할을 합니다.

2. Keras와 TensorFlow를 활용한 모델 구현

- **모델 정의**: Keras의 `Sequential` 모델과 다양한 층(Layer)을 사용하여 생성기와 판별기를 정의하는 방법을 배웠습니다. 이를 통해 모델의 구조를 설계하고, 적절한 활성화 함수(예: LeakyReLU)를 선택하는 것이 중요함을 이해했습니다.

- **모델 컴파일**: 손실 함수(`binary_crossentropy`)와 옵티마이저(`Adam`)를 사용하여 모델을 컴파일하는 방법을 학습했습니다. 손실 함수는 모델의 학습 방향을 결정하고, 옵티마이저는 학습 속도를 조절합니다.

3. 학습 과정의 세부 사항

- **손실 계산 방법**: GAN 학습에서 판별기와 생성기의 손실을 각각 계산하는 방법을 이해했습니다. 판별기는 진짜 데이터를 진짜로, 가짜 데이터를 가짜로 분류하는 것을 목표로 하며, 생성기는 판별기를 속여 가짜 데이터를 진짜로 분류하게 만드는 것을 목표로 합니다.
- **학습 루프 구현**: GAN의 학습 루프를 구성하여, 각 배치마다 생성기와 판별기를 번갈아 학습시키는 방법을 배웠습니다. 이를 통해 모델이 점차적으로 성능을 개선해 나갈 수 있도록 합니다.

4. 문제 해결 및 최적화

- **그래프 모드와 `eager execution`**: TensorFlow의 그래프 모드와 `eager execution` 모드의 차이점을 이해하고, 각각의 모드에서 발생할 수 있는 문제를 해결하는 방법을 배웠습니다. 특히, `numpy()`를 사용할 때의 주의사항과 그래프 모드에서의 대안에 대해 학습했습니다.
- **콜백을 통한 학습 모니터링**: Keras의 `Callback` 기능을 사용하여 학습 과정을 모니터링하고, 손실 추적 및 시각화를 자동화하는 방법을 배웠습니다. 이를 통해 모델의 학습 과정을 보다 직관적으로 이해할 수 있었습니다.

5. 코드 분석 및 개선

- **코드 분석 및 디버깅**: GAN 모델 구현 과정에서 발생하는 다양한 오류를 디버깅하고, 올바른 해결 방법을 찾는 과정을 배웠습니다. 특히, 손실 값 저장 및 평가에 대한 문제를 해결하는 과정에서 많은 것을 배울 수 있었습니다.
- **리포트 작성**: Jupyter Notebook의 내용을 기반으로 보고서를 작성하는 방법을 이해했습니다. 코드 설명과 실습 내용을 체계적으로 정리하여 문서화하는 것이 중요하다는 것을 배웠습니다.