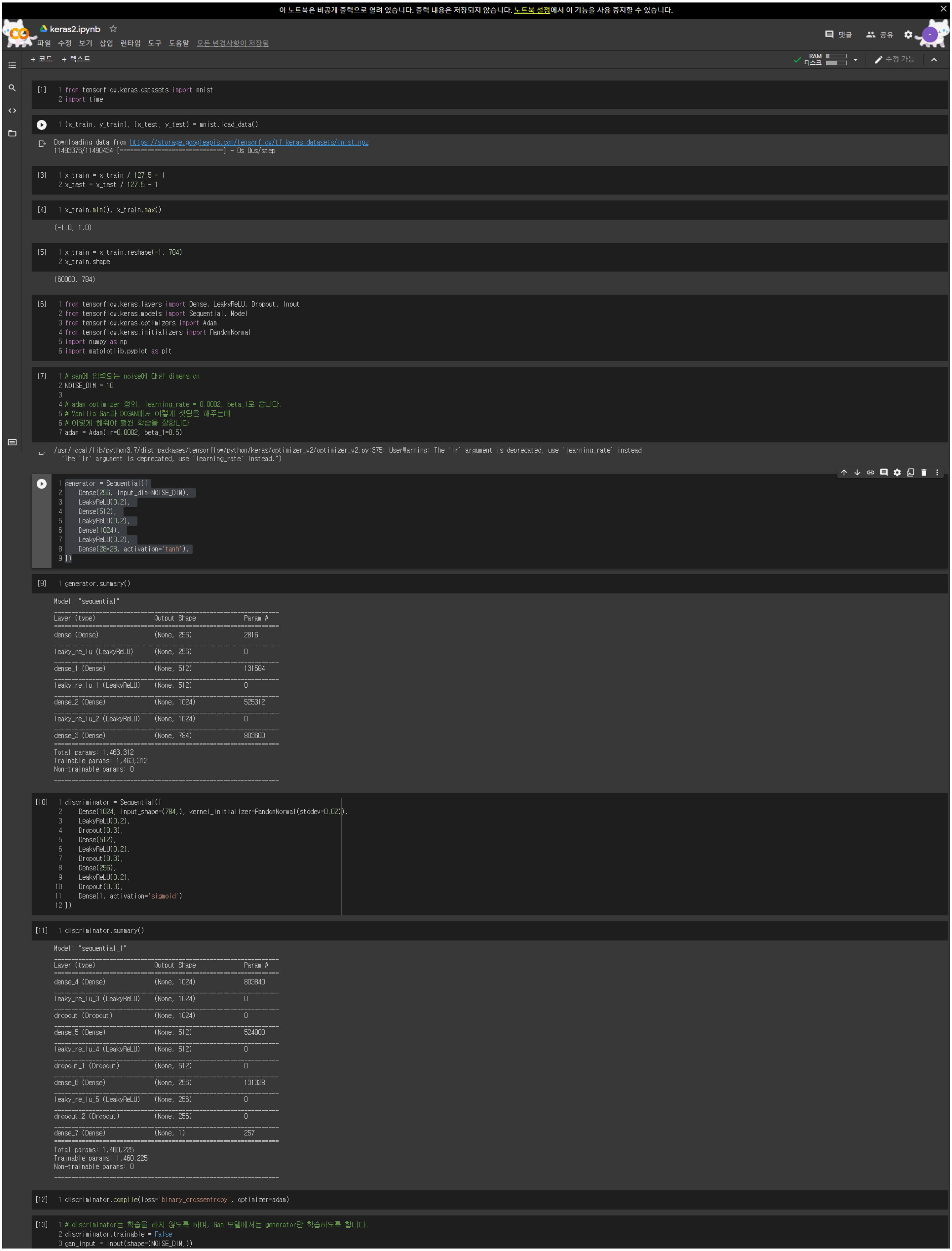

이 노트북은 비공개 출력으로 열려 있습니다. 출력 내용은 저장되지 않습니다. 노트북 설정에서 이 기능을 사용 중지할 수 있습니다.
keras2.ipynb ☆
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글
공유
+ 코드 + 텍스트
RAM
디스크
수정 가능

```
[1]  from tensorflow.keras.datasets import mnist
     import time
```

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

     Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
     11493376/11490434 [==============================] - 0s 0us/step
```

```
[3]  x_train = x_train / 127.5 - 1
     x_test = x_test / 127.5 - 1
```

```
[4]  x_train.min(), x_train.max()

     (-1.0, 1.0)
```

```
[5]  x_train = x_train.reshape(-1, 784)
     x_train.shape

     (60000, 784)
```

```
[6]  from tensorflow.keras.layers import Dense, LeakyReLU, Dropout, Input
     from tensorflow.keras.models import Sequential, Model
     from tensorflow.keras.optimizers import Adam
     from tensorflow.keras.initializers import RandomNormal
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[7]  # gan에 입력되는 noise에 대한 dimension
     NOISE_DIM = 10

     # adam optimizer 정의, learning_rate = 0.0002, beta_1로 줍니다.
     # Vanilla Gan과 DCGAN에서 이렇게 셋팅을 해주는데
     # 이렇게 해줘야 훨씬 학습을 잘합니다.
     adam = Adam(lr=0.0002, beta_1=0.5)

     /usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/optimizer_v2/optimizer_v2.py:375: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
       "The `lr` argument is deprecated, use `learning_rate` instead.")
```

```
     generator = Sequential([
         Dense(256, input_dim=NOISE_DIM),
         LeakyReLU(0.2),
         Dense(512),
         LeakyReLU(0.2),
         Dense(1024),
         LeakyReLU(0.2),
         Dense(28*28, activation='tanh'),
     ])
```

```
[9]  generator.summary()

     Model: "sequential"
     _________________________________________________________________
     Layer (type)                 Output Shape              Param #
     =================================================================
     dense (Dense)                (None, 256)               2816
     _________________________________________________________________
     leaky_re_lu (LeakyReLU)      (None, 256)               0
     _________________________________________________________________
     dense_1 (Dense)              (None, 512)               131584
     _________________________________________________________________
     leaky_re_lu_1 (LeakyReLU)    (None, 512)               0
     _________________________________________________________________
     dense_2 (Dense)              (None, 1024)              525312
     _________________________________________________________________
     leaky_re_lu_2 (LeakyReLU)    (None, 1024)              0
     _________________________________________________________________
     dense_3 (Dense)              (None, 784)               803600
     =================================================================
     Total params: 1,463,312
     Trainable params: 1,463,312
     Non-trainable params: 0
     _________________________________________________________________
```

```
[10] discriminator = Sequential([
         Dense(1024, input_shape=(784,), kernel_initializer=RandomNormal(stddev=0.02)),
         LeakyReLU(0.2),
         Dropout(0.3),
         Dense(512),
         LeakyReLU(0.2),
         Dropout(0.3),
         Dense(256),
         LeakyReLU(0.2),
         Dropout(0.3),
         Dense(1, activation='sigmoid')
     ])
```

```
[11] discriminator.summary()

     Model: "sequential_1"
     _________________________________________________________________
     Layer (type)                 Output Shape              Param #
     =================================================================
     dense_4 (Dense)              (None, 1024)              803840
     _________________________________________________________________
     leaky_re_lu_3 (LeakyReLU)    (None, 1024)              0
     _________________________________________________________________
     dropout (Dropout)            (None, 1024)              0
     _________________________________________________________________
     dense_5 (Dense)              (None, 512)               524800
     _________________________________________________________________
     leaky_re_lu_4 (LeakyReLU)    (None, 512)               0
     _________________________________________________________________
     dropout_1 (Dropout)          (None, 512)               0
     _________________________________________________________________
     dense_6 (Dense)              (None, 256)               131328
     _________________________________________________________________
     leaky_re_lu_5 (LeakyReLU)    (None, 256)               0
     _________________________________________________________________
     dropout_2 (Dropout)          (None, 256)               0
     _________________________________________________________________
     dense_7 (Dense)              (None, 1)                 257
     =================================================================
     Total params: 1,460,225
     Trainable params: 1,460,225
     Non-trainable params: 0
     _________________________________________________________________
```

```
[12] discriminator.compile(loss='binary_crossentropy', optimizer=adam)
```

```
[13] # discriminator는 학습을 하지 않도록 하며, Gan 모델에서는 generator만 학습하도록 합니다.
     discriminator.trainable = False
     gan_input = Input(shape=(NOISE_DIM,))
```

```python
x = generator(inputs=gan_input)
output = discriminator(x)
```

[14]

```python
gan = Model(gan_input, output)
```

[15]

```python
gan.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 10)]              0
_________________________________________________________________
sequential (Sequential)      (None, 784)               1463312
_________________________________________________________________
sequential_1 (Sequential)    (None, 1)                 1460225
=================================================================
Total params: 2,923,537
Trainable params: 1,463,312
Non-trainable params: 1,460,225
_________________________________________________________________
```

[16]

```python
gan.compile(loss='binary_crossentropy', optimizer=adam)
```

[17]

```python
def get_batches(data, batch_size):
    batches = []
    for i in range(int(data.shape[0] // batch_size)):
        batch = data[i * batch_size: (i + 1) * batch_size]
        batches.append(batch)
    return np.asarray(batches)
```

[18]

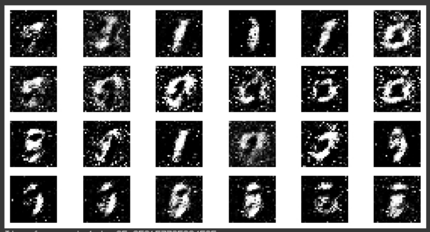
```python
def visualize_training(epoch, d_losses, g_losses):
    '''
    # 오차에 대한 시각화
    plt.figure(figsize=(8, 4))
    plt.plot(d_losses, label='Discriminator Loss')
    plt.plot(g_losses, label='Generatror Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    print('epoch: {}, Discriminator Loss: {}, Generator Loss: {}'.format(epoch, np.asarray(d_losses).mean(), np.asarray(g_losses).mean()))
    '''
    #샘플 데이터 생성 후 시각화
    noise = np.random.normal(0, 1, size=(24, NOISE_DIM))
    generated_images = generator.predict(noise)
    generated_images = generated_images.reshape(-1, 28, 28)

    plt.figure(figsize=(8, 4))
    for i in range(generated_images.shape[0]):
        plt.subplot(4, 6, i+1)
        plt.imshow(generated_images[i], interpolation='nearest', cmap='gray')
        plt.axis('off')
    plt.tight_layout()
    plt.show()
```

[19]

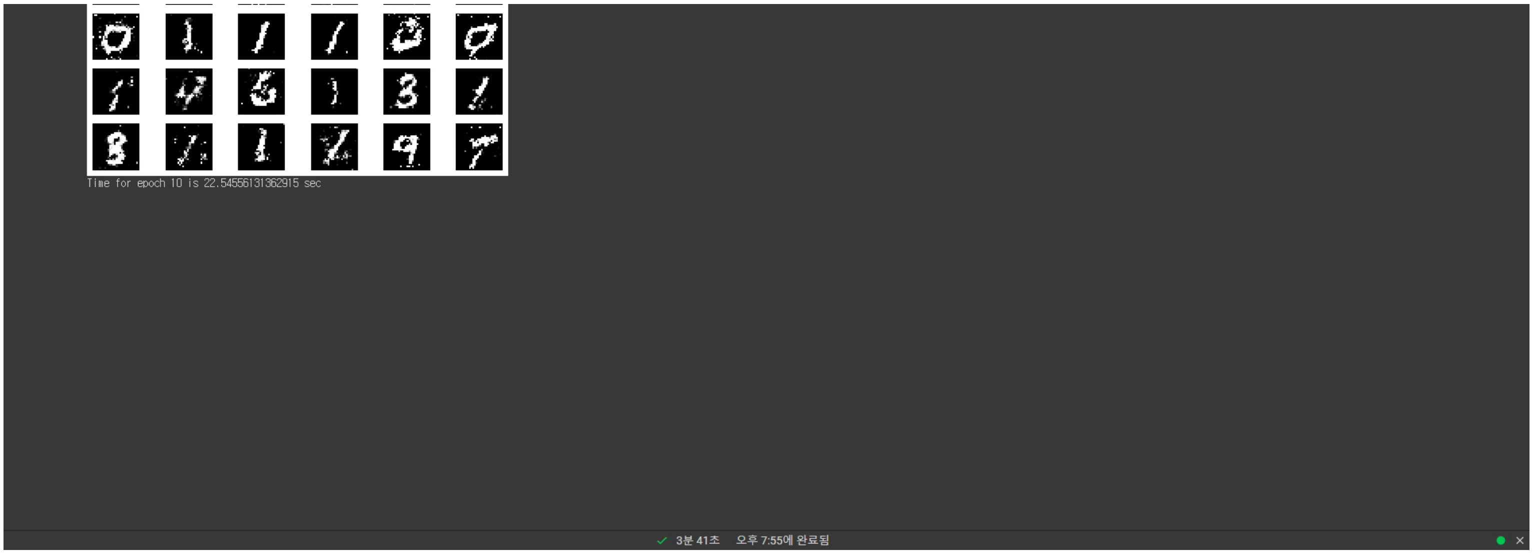
```python
BATCH_SIZE = 128
EPOCHS= 10
```

[20]

```python
# discriminator와 gan 모델의 loss 측정을 위한 list 입니다.
d_losses = []
g_losses = []

for epoch in range(1, EPOCHS + 1):
    start = time.time()

    # 각 배치별 학습
    for real_images in get_batches(x_train, BATCH_SIZE):
        # 랜덤 노이즈 생성
        input_noise = np.random.uniform(-1, 1, size=[BATCH_SIZE, NOISE_DIM])

        # 가짜 이미지 데이터 생성
        generated_images = generator.predict(input_noise)

        # Gan에 학습할 X 데이터 정의
        x_dis = np.concatenate([real_images, generated_images])

        # Gan에 학습할 Y 데이터 정의
        y_dis = np.zeros(2 * BATCH_SIZE)
        y_dis[:BATCH_SIZE] = 0.9

        # Discriminator 훈련
        discriminator.trainable = True
        d_loss = discriminator.train_on_batch(x_dis, y_dis)

        # Gan 훈련
        noise = np.random.uniform(-1, 1, size=[BATCH_SIZE, NOISE_DIM])
        y_gan = np.ones(BATCH_SIZE)

        # Discriminator의 판별 학습을 방지합니다
        discriminator.trainable = False
        g_loss = gan.train_on_batch(noise, y_gan)

    d_losses.append(d_loss)
    g_losses.append(g_loss)

    if epoch == 1 or epoch % 10 == 0:
        visualize_training(epoch, d_losses, g_losses)

    # print (' 에포크 {} 에서 걸린 시간은 {} 초 입니다'.format(epoch +1, time.time()-start))
    print ('Time for epoch {} is {} sec'.format(epoch, time.time()-start))
```



```
Time for epoch 1 is 25.952157735824585 sec
Time for epoch 2 is 21.33253788948059 sec
Time for epoch 3 is 21.51124858856201 sec
Time for epoch 4 is 21.60704803466797 sec
Time for epoch 5 is 21.698396682739258 sec
Time for epoch 6 is 21.824479818344116 sec
Time for epoch 7 is 21.795499563217163 sec
Time for epoch 8 is 21.749380350112915 sec
Time for epoch 9 is 21.799075841903687 sec
```



```
Time for epoch 10 is 22.54556131362915 sec
```