نگین طالبی

یه دونه gan که عدد تولید کنه روی mnist

نسبت به آخرین شماره دانشجویی

(2)

```
!pip install tensorflow
!pip install keras
!pip install numpy
from keras.datasets import mnist

(train_images, train_labels), (_, _) = mnist.load_data()

#  reshape and normalize images
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255.
import keras
from keras import layers

latent_dim = 100

# define generator
generator_input = keras.Input(shape=(latent_dim,))
x = layers.Dense(128 * 7 * 7)(generator_input)
x = layers.LeakyReLU()(x)
x = layers.Reshape((7, 7, 128))(x)
x = layers.Conv2DTranspose(128, 4, strides=2, padding='same')(x)
x = layers.LeakyReLU()(x)
x = layers.Conv2DTranspose(128, 4, strides=2, padding='same')(x)
x = layers.LeakyReLU()(x)
x = layers.Conv2D(1, 7, activation='sigmoid', padding='same')(x)
generator = keras.models.Model(generator_input, x)

# define discriminator
discriminator_input = layers.Input(shape=(28, 28, 1))
x = layers.Conv2D(64, 3)(discriminator_input)
x = layers.LeakyReLU()(x)
x = layers.Conv2D(128, 3, strides=2)(x)
x = layers.LeakyReLU()(x)
x = layers.Conv2D(128, 3, strides=2)(x)
x = layers.LeakyReLU()(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.4)(x)
x = layers.Dense(1, activation='sigmoid')(x)
discriminator = keras.models.Model(discriminator_input, x)

# combined model
gan_input = keras.Input(shape=(latent_dim,))
gan_output = discriminator(generator(gan_input))
gan = keras.models.Model(gan_input, gan_output)

# compile discriminator
discriminator.compile(optimizer='rmsprop',
                      loss='binary_crossentropy')

# compile GAN
discriminator.trainable = False
```

```
gan.compile(optimizer='rmsprop',
            loss='binary_crossentropy')
def generate_2():
    # prepare labels for generated images
    y_gen = np.ones((1, 1))

    # generate image with GAN
    noise = np.random.normal(size=(1, latent_dim))
    generated_images = generator.predict(noise)

    # filter for 2s only
    generated_2s = []
    for img in generated_images:
        if np.argmax(train_labels) == 2:
            generated_2s.append(img)

    # train discriminator on generated 2s
    X = np.array(generated_2s)
    y = np.zeros((len(X), 1))
    discriminator.train_on_batch(X, y)

    # train GAN on generated 2s
    noise = np.random.normal(size=(1, latent_dim))
    y_gan = np.ones((1, 1))
    gan.train_on_batch(noise, y_gan)

# train generator and discriminator alternately
for i in range(10000):
    generate_2()
```