Praktik_1_-_GAN

March 7, 2022

0.1 Elmo Allistair - 12118220 - 4KA17

0.2 1. Import Libary

Pada tahapan ini Anda mendefinisikan terlebih dahulu library yang akan Anda gunakan

[1]:
```
from keras.datasets import mnist
```

[2]:
```
from tensorflow.keras import Sequential
from keras.layers import BatchNormalization, Dense, Reshape, Flatten
from keras.layers.advanced_activations import LeakyReLU
from tensorflow.keras.optimizers import Adam
import numpy as np
```

0.3 2. Mendefinisikan Variabel untuk Neural Network dan Data

Pada tahapan ini kita akan mendefinisikan beberapa variabel yang diperlukan diantaranya

- Ukuran gambar yang akan di generate oleh algoritma GAN
- Channel warna yang akan digunakan
- Variable noise / latent yang akan membentuk gambar
- Optimizer - Variable yang menentukan algoritma optimisasi pembelajaran yang akan dilakukan. Disini kita menggunakan algoritma Stochastic Gradient Decent dengan learning rate 0.0001. Learning rate mendefinisikan seberapa cepat algoritma mempelajari data yang diberikan, dan nilai 0.0001 adalah nilai yang direkomendasikan digunakan untuk pembentukan model GAN.

[3]:
```
## mendefinisikan variable gambar
## ukuran disesai
img_width = 28
img_height = 28
channels = 1
img_shape = (img_width, img_height, channels)
latent_dim = 100
adam = Adam(learning_rate=0.0001)
```

0.4 3. Membentuk Generator

Generator adalah bagian dari GAN yang bertugas untuk belajar membuat data palsu dengan memasukkan umpan balik dari diskriminator. Ia belajar membuat diskriminator mengklasifikasikan outputnya sebagai yang sebenarnya atau nyata.

[8]:
```
def build_generator():
    model = Sequential()

    model.add(Dense(256, input_dim=latent_dim))
    ## add activation function
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))

    model.add(Dense(512))
    ## add activation function
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))

    model.add(Dense(1024))
    ## add activation function
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))

    ##membuat model menjadi ukuran 28x28x1
    model.add(Dense(np.prod(img_shape), activation='tanh'))
    model.add(Reshape(img_shape))

    model.summary()
    return model
```

[9]:
```
generator = build_generator()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 256)               25856
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 256)               0
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024
_________________________________________________________________
dense_5 (Dense)              (None, 512)               131584
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 512)               0
_________________________________________________________________
batch_normalization_5 (Batch (None, 512)               2048
_________________________________________________________________
```

```
dense_6 (Dense)              (None, 1024)              525312
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 1024)              0
_________________________________________________________________
batch_normalization_6 (Batch (None, 1024)              4096
_________________________________________________________________
dense_7 (Dense)              (None, 784)               803600
_________________________________________________________________
reshape (Reshape)            (None, 28, 28, 1)         0
=================================================================
Total params: 1,493,520
Trainable params: 1,489,936
Non-trainable params: 3,584
_________________________________________________________________
```

0.5 4. Mendefinisikan Discriminator

Diskriminator dalam GAN hanyalah sebuah fungsi klasifikasi. Ia mencoba untuk membedakan data nyata dari data yang dibuat oleh generator. Diskriminator bisa menggunakan arsitektur jaringan apa pun yang sesuai dengan jenis data yang diklasifikasikan.

[10]:
```python
def build_discriminator():
    model = Sequential()

    model.add(Flatten(input_shape=img_shape))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))

    model.add(Dense(256))
    model.add(LeakyReLU(alpha=0.2))

    model.summary()
    return model

discriminator = build_discriminator()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense_8 (Dense)              (None, 512)               401920
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 512)               0
_________________________________________________________________
dense_9 (Dense)              (None, 256)               131328
_________________________________________________________________
```

```
leaky_re_lu_8 (LeakyReLU)    (None, 256)               0
=================================================================
Total params: 533,248
Trainable params: 533,248
Non-trainable params: 0
_________________________________________________________________
```

0.6 5. Menghubungkan Discriminator dan Generator untuk membentuk GAN

Pada tahapan ini kita akan menghubungkan discriminator dan generator yang telah dibuat untuk membentuk sebuah GAN

[11]:
```
discriminator.compile(loss='binary_crossentropy', optimizer='adam')

GAN = Sequential()
discriminator.trainable = False
GAN.add(generator)
GAN.add(discriminator)

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