Aldi Dharmawan

50420108

4IA10

Praktikum Robotika Cerdas Minggu Ke-3

1) Importing Python Packages for GAN

```
from keras.datasets import cifar10, mnist
from keras.models import Sequential
from keras.layers import Reshape
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Conv2D
from keras.layers import Conv2DTranspose
from keras.layers import Dropout
from tensorflow.keras.layers import LeakyReLU
from tensorflow.keras.optimizers import Adam
import numpy as np
!mkdir generated_images
```

```
mkdir: cannot create directory ‘generated_images’: File exists
```

Pertama-tama mengimport library yang digunakan dalam program dari implementasi convolutional autoencoder mengunakan modul keras dan tensorflow untuk melatih model jaringan saraf tiruan. Kemudian mengimpot sebuah dataset CIFAR-10 dan MNIST yang diimport langsung dari library keras.

2) Parameters for Neural Networks & Data

```
[ ] img_width = 32
    img_height = 32
    channels = 3
    img_shape = (img_width, img_height, channels)
    latent_dim = 100
    adam = Adam(lr=0.0002)
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
```

Langkah kedua, baris selanjutnya menginisialisasi beberapa variable yang diperlukan untuk membangun dan melatih autoencoder. Variabel-variabel

ini nantinya akan digunakan dalam proses pembuatan, pelatihan, dan penggunaan model autoencoder yang telah dibangun sebelumnya.

3) Building Generator

```
def build_generator():
    model = Sequential()

    # Create first layer, to receive the input
    model.add(Dense(256 * 4 * 4, input_dim = latent_dim))
    # 256 * 8 * 8; for upscaling the layers,
    # initial shape to construct into final shape

    # Create default activation function
    model.add(LeakyReLU(alpha = 0.2))

    # Create reshape layer
    model.add(Reshape((4, 4,256)))
    # 8,8,256 ; reffers to first layer

    # Adding more layers for neurons and better result
    model.add(Conv2DTranspose(128, (4,4), strides = (2,2), padding = 'same'))
    model.add(LeakyReLU(alpha= 0.2))
    model.add(Conv2DTranspose(128, (4,4), strides = (2,2), padding = 'same'))
    model.add(LeakyReLU(alpha= 0.2))
    model.add(Conv2DTranspose(128, (4,4), strides = (2,2), padding = 'same'))
    model.add(LeakyReLU(alpha= 0.2))
    # (4,4) >> filter size
    # strides = (2,2) >> Convolutional layers, that how NN understand images

    # Create Final output layer and forming image shape
    # the shape (3, (3,3)) reffers to image shape :
    #    >>>  img_shape = (img_width, img_height, channels)
    model.add(Conv2D(3, (3,3), activation= 'tanh', padding = 'same'))

    #
    model.summary()
    return model
```

Output:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 4096)              413696

 leaky_re_lu (LeakyReLU)     (None, 4096)              0

 reshape (Reshape)           (None, 4, 4, 256)         0

 conv2d_transpose (Conv2DTr  (None, 8, 8, 128)         524416
 anspose)

 leaky_re_lu_1 (LeakyReLU)   (None, 8, 8, 128)         0

 conv2d_transpose_1 (Conv2D  (None, 16, 16, 128)       262272
 Transpose)

 leaky_re_lu_2 (LeakyReLU)   (None, 16, 16, 128)       0

 conv2d_transpose_2 (Conv2D  (None, 32, 32, 128)       262272
 Transpose)

 leaky_re_lu_3 (LeakyReLU)   (None, 32, 32, 128)       0

 conv2d (Conv2D)             (None, 32, 32, 3)         3459

=================================================================
Total params: 1466115 (5.59 MB)
Trainable params: 1466115 (5.59 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Langkah Ketiga, membangun generator, diawali dengan membuat fungsi build_generator(), di dalam bariini digunakan untuk membangun generator dalam model GAN. Generator berguna untuk menghasilkan gambar-gambar baru yang akan meniru gambar dari dataset asli.

- **4) Building Discriminator**

```
def build_discriminator():
    model = Sequential()

    # Create input layer and filter and stride layer. That makes NN understand image
    model.add(Conv2D(64, (3,3), padding = 'same', input_shape = img_shape))

    # Adding activation function
    model.add(LeakyReLU(alpha = 0.2))
    model.add(Conv2D(128, (3,3), padding = 'same'))
    model.add(LeakyReLU(alpha = 0.2))
    model.add(Conv2D(128, (3,3), padding = 'same'))
    model.add(LeakyReLU(alpha = 0.2))
    model.add(Conv2D(256, (3,3), padding = 'same'))
    model.add(LeakyReLU(alpha = 0.2))
    model.add(Flatten())

    model.add(Dropout(0.4))

    # Create output layer
    model.add(Dense(1, activation = 'sigmoid'))

    model.summary()
    return model

discriminator = build_discriminator()
discriminator.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
```

Output:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 32, 32, 64)        1792

 leaky_re_lu_4 (LeakyReLU)   (None, 32, 32, 64)        0

 conv2d_2 (Conv2D)           (None, 32, 32, 128)       73856

 leaky_re_lu_5 (LeakyReLU)   (None, 32, 32, 128)       0

 conv2d_3 (Conv2D)           (None, 32, 32, 128)       147584

 leaky_re_lu_6 (LeakyReLU)   (None, 32, 32, 128)       0

 conv2d_4 (Conv2D)           (None, 32, 32, 256)       295168

 leaky_re_lu_7 (LeakyReLU)   (None, 32, 32, 256)       0

 flatten (Flatten)           (None, 262144)            0

 dropout (Dropout)           (None, 262144)            0

 dense_1 (Dense)             (None, 1)                 262145

=================================================================
Total params: 780545 (2.98 MB)
Trainable params: 780545 (2.98 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Langkah keempat, membangunsebuah discriminator. Diawali dengan membuat sebuah fungsi build_discriminator, yang digunakan untuk membedakan antara gambar-gambar asli dari dataset dan gambar-gambar yang dihasilkan oleh generator. Kemudian, model discriminator yang telah dibangun nantinya disimpan dalam variable ‘discriminator’, model tersebut siap untuk digunakan dalam GAN. Model ini nanti akan bersaing dengan generator dalam pelatihan GAN untuk menghasilkan gambar-gambar baru yang semirip mungkin dengan data asli.

5) Connecting Neural Networks to build GAN

```
[6]  GAN = Sequential()
     discriminator.trainable = False
     GAN.add(generator)
     GAN.add(discriminator)

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

Langkah kelima yaitu menghubungkan jaringan saraf tiruan untuk membangun GAN. Kode program ini akan menggabungkan model generator dan model discriminator. Model GAN adalah model utama dalam GAN yang mengkoordinasikan pelatihan generator dan diskriminator. Selama training GAN nanti, model ini akan digunakan untuk melatih generator dengan mengoptimalkan discriminator, sehingga generator dapat menghasilkan gambar-gambar yang semakin mirip dengan gambar asli.