RIZKI_MUHAMMAD_SYAMSI_50422047_4IA21_PRAKTIKUM_ROBOT

October 1, 2025

Nama: RIZKI MUHAMMAD SYAMSI **Kelas**: 4IA21 **NPM**: 50422047 **Mata Kuliah**: PRAKTIKUM ROBOTIKA CERDAS **Pertemuan**: 1

1 IMPORT LIBRARY

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

Pertama-tama saya mengimpor library yang dibutuhkan. Saya menggunakan keras dan tensorflow.keras untuk membangun model Generator dan Discriminator. Kemudian saya mengimpor BatchNormalization, Dense, Reshape, Flatten, dan LeakyReLU yang berfungsi sebagai lapisan dalam jaringan saraf. Optimizer Adam juga saya gunakan karena umum dipakai pada pelatihan model GAN. Selain itu, saya mengimpor numpy untuk melakukan operasi numerik.

2 INISIALISASI PARAMETER DASAR

[ ]:
```
img_width = 28
img_height = 28
channels = 1
img_shape = (img_width, img_height, channels)
latent_dim = 100
adam = Adam(learning_rate=0.0001)
```

Setelah itu, saya mendefinisikan parameter dasar. Dataset MNIST memiliki ukuran gambar 28x28 piksel dengan 1 channel (grayscale). Oleh karena itu saya menyimpan bentuk input gambar ke dalam variabel img_shape. Kemudian saya menentukan dimensi noise (latent_dim) sebesar 100, yang akan menjadi input Generator untuk menghasilkan gambar palsu. Saya juga menginisialisasi optimizer Adam dengan learning rate 0.0001 untuk digunakan pada Generator dan Discriminator.

#MEMBANGUN GENERATOR (Versi Awal / Debug)

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

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

  model.summary()
  return model
```

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

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.12/dist-
packages/keras/src/layers/activations/leaky_relu.py:41: UserWarning: Argument
`alpha` is deprecated. Use `negative_slope` instead.
  warnings.warn(
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 256)	25,856
leaky_re_lu (LeakyReLU)	(None, 256)	0
batch_normalization (BatchNormalization)	(None, 256)	1,024

Total params: 26,880 (105.00 KB)

Trainable params: 26,368 (103.00 KB)

Non-trainable params: 512 (2.00 KB)

Pada bagian ini saya membuat fungsi build_generator() versi awal untuk keperluan pengecekan arsitektur. Generator menerima input berupa vektor noise berdimensi 100, kemudian melewati lapisan Dense(256) dengan aktivasi LeakyReLU. Agar distribusi data lebih stabil saat training, saya menambahkan BatchNormalization. Model ini saya cetak dengan model.summary() untuk

memastikan struktur lapisannya sudah sesuai.

#MEMBANGUN GENERATOR (Versi Lengkap)

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

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

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

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

  model.add(Dense(np.prod(img_shape), activation='tanh'))
  model.add(Reshape(img_shape))

  model.summary()
  return model
```

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

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 256)	25,856
leaky_re_lu_1 (LeakyReLU)	(None, 256)	0
batch_normalization_1 (BatchNormalization)	(None, 256)	1,024
dense_2 (Dense)	(None, 512)	131,584
leaky_re_lu_2 (LeakyReLU)	(None, 512)	0
batch_normalization_2 (BatchNormalization)	(None, 512)	2,048
dense_3 (Dense)	(None, 1024)	525,312

```
 leaky_re_lu_3 (LeakyReLU)        (None, 1024)                   0

 batch_normalization_3            (None, 1024)               4,096
 (BatchNormalization)

 dense_4 (Dense)                  (None, 784)              803,600

 reshape (Reshape)                (None, 28, 28, 1)              0


Total params: 1,493,520 (5.70 MB)

Trainable params: 1,489,936 (5.68 MB)

Non-trainable params: 3,584 (14.00 KB)
```

Setelah berhasil membangun versi sederhana, saya memperluas arsitektur Generator dengan beberapa lapisan tambahan. Saya menambahkan lapisan Dense(512) dan Dense(1024) dengan aktivasi LeakyReLU dan BatchNormalization agar jaringan lebih dalam dan mampu belajar pola yang lebih kompleks. Output Generator berupa 784 neuron (28x28x1) dengan aktivasi tanh, lalu saya Reshape menjadi gambar 28x28x1. Fungsi tanh dipilih karena output gambar akan berada pada rentang [-1, 1], sehingga sesuai dengan preprocessing data MNIST yang biasanya dinormalisasi ke dalam range tersebut.

#MEMBANGUN DISCRIMINATOR

[ ]:
```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()
```

```
/usr/local/lib/python3.12/dist-
packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

```
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense_5 (Dense)	(None, 512)	401,920
leaky_re_lu_4 (LeakyReLU)	(None, 512)	0
dense_6 (Dense)	(None, 256)	131,328
leaky_re_lu_5 (LeakyReLU)	(None, 256)	0

```
Total params: 533,248 (2.03 MB)

Trainable params: 533,248 (2.03 MB)

Non-trainable params: 0 (0.00 B)
```

Selanjutnya saya membuat Discriminator dengan fungsi build_discriminator(). Discriminator bertugas membedakan apakah gambar yang diberikan merupakan gambar asli dari dataset MNIST atau gambar palsu hasil Generator. Pertama, gambar input 28x28x1 saya Flatten menjadi vektor 784 elemen. Kemudian vektor ini diproses melalui lapisan Dense(512) dan Dense(256) dengan aktivasi LeakyReLU. Dengan arsitektur ini, Discriminator mampu mendeteksi ciri-ciri penting dari gambar untuk menentukan label "asli" atau "palsu". Terakhir, model ini saya kompilasi menggunakan binary crossentropy loss karena tugasnya berupa klasifikasi biner.

#MEMBANGUN GABUNGAN GENERATOR + DISCRIMINATOR (GAN)

[ ]:
```
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')
```

Pada bagian terakhir saya membangun model GAN dengan cara menggabungkan Generator dan Discriminator. Sebelum digabungkan, saya set discriminator.trainable = False supaya saat melatih GAN, hanya Generator yang diperbarui parameternya. Dengan cara ini Generator belajar menghasilkan gambar yang semakin mirip dengan data asli, agar bisa "menipu" Discriminator. Akhirnya

saya menyusun GAN sebagai Sequential model, menambahkan Generator dan Discriminator secara berurutan, lalu mengompilasinya menggunakan binary crossentropy loss.