

CERDAS MENGUASAI GIT

CERDAS MENGUASAI GIT

Dalam 24 Jam

Rolly M. Awangga
Informatics Research Center



Kreatif Industri Nusantara

Penulis:

Rolly Maulana Awangga

ISBN : 978-602-53897-0-2

Editor:

M. Yusril Helmi Setyawan

Penyunting:

Syafrial Fachrie Pane

Khaera Tunnisa

Diana Asri Wijayanti

Desain sampul dan Tata letak:

Deza Martha Akbar

Penerbit:

Kreatif Industri Nusantara

Redaksi:

Jl. Ligar Nyawang No. 2

Bandung 40191

Tel. 022 2045-8529

Email : awangga@kreatif.co.id

Distributor:

Informatics Research Center

Jl. Sariasih No. 54

Bandung 40151

Email : irc@poltekpos.ac.id

Cetakan Pertama, 2019

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara
apapun tanpa ijin tertulis dari penerbit

*‘Jika Kamu tidak dapat
menahan lelahnya
belajar, Maka kamu harus
sanggup menahan
perihnya Kebodohan.’
Imam Syafi’i*

CONTRIBUTORS

ROLLY MAULANA AWANGGA, Informatics Research Center., Politeknik Pos Indonesia, Bandung, Indonesia

CONTENTS IN BRIEF

1 Chapter 1	1
2 Chapter 2	3
3 Chapter 3	5
4 Chapter 4	7
5 Chapter 5	9
6 Chapter 6	11
7 Chapter 7	13
8 Chapter 8	15
9 Chapter 9	31
10 Chapter 10	57
11 Chapter 11	59
12 Chapter 12	61
13 Chapter 13	63
14 Chapter 14	65

DAFTAR ISI

Foreword	xi
Kata Pengantar	xiii
Acknowledgments	xv
Acronyms	xvii
Glossary	xix
List of Symbols	xxi
Introduction	xxiii
<i>Rolly Maulana Awangga, S.T., M.T.</i>	
1 Chapter 1	1
2 Chapter 2	3
3 Chapter 3	5
4 Chapter 4	7
	ix

5	Chapter 5	9
6	Chapter 6	11
7	Chapter 7	13
8	Chapter 8	15
8.1	1164013 - Ikrima Ningrum	15
8.1.1	Soal Teori	15
8.1.2	Praktek Program	19
8.1.3	Penanganan Error	29
8.1.4	Bukti Tidak Plagiat	30
9	Chapter 9	31
9.1	1174096 - Nico Ekklesia Sembiring	31
9.1.1	Soal Teori	31
9.1.2	Praktek Program	35
9.1.3	Penanganan Error	40
9.1.4	Bukti Tidak Plagiat	41
9.2	1164013 - Ikrima Ningrumsari Mulyana	41
9.2.1	Teori	41
9.2.2	Praktek Program	45
9.2.3	Penanganan Error	55
9.2.4	Bukti Tidak Plagiat	55
10	Chapter 10	57
11	Chapter 11	59
12	Chapter 12	61
13	Chapter 13	63
14	Chapter 14	65
	Daftar Pustaka	67
	Index	69

FOREWORD

Sepatah kata dari Kaprodi, Kabag Kemahasiswaan dan Mahasiswa

KATA PENGANTAR

Buku ini diciptakan bagi yang awam dengan git sekalipun.

R. M. AWANGGA

*Bandung, Jawa Barat
Februari, 2019*

ACKNOWLEDGMENTS

Terima kasih atas semua masukan dari para mahasiswa agar bisa membuat buku ini lebih baik dan lebih mudah dimengerti.

Terima kasih ini juga ditujukan khusus untuk team IRC yang telah fokus untuk belajar dan memahami bagaimana buku ini mendampingi proses Intership.

R. M. A.

ACRONYMS

ACGIH	American Conference of Governmental Industrial Hygienists
AEC	Atomic Energy Commission
OSHA	Occupational Health and Safety Commission
SAMA	Scientific Apparatus Makers Association

GLOSSARY

git	Merupakan manajemen sumber kode yang dibuat oleh linus torvald.
bash	Merupakan bahasa sistem operasi berbasiskan *NIX.
linux	Sistem operasi berbasis sumber kode terbuka yang dibuat oleh Linus Torvald

SYMBOLS

A Amplitude

$\&$ Propositional logic symbol

a Filter Coefficient

\mathcal{B} Number of Beats

INTRODUCTION

ROLLY MAULANA AWANGGA, S.T., M.T.

Informatics Research Center
Bandung, Jawa Barat, Indonesia

Pada era disruptif saat ini. git merupakan sebuah kebutuhan dalam sebuah organisasi pengembangan perangkat lunak. Buku ini diharapkan bisa menjadi penghantar para programmer, analis, IT Operation dan Project Manajer. Dalam melakukan implementasi git pada diri dan organisasinya.

Rumusnya cuman sebagai contoh aja biar keren[1].

$$ABCDEF\alpha\beta\Gamma\Delta\sum_{def}^{abc} \tag{I.1}$$

BAB 1

CHAPTER 1

BAB 2

CHAPTER 2

BAB 3

CHAPTER 3

BAB 4

CHAPTER 4

BAB 5

CHAPTER 5

BAB 6

CHAPTER 6

BAB 7

CHAPTER 7

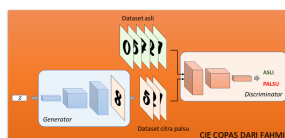
BAB 8

CHAPTER 8

8.1 1164013 - Ikrima Ningrum

8.1.1 Soal Teori

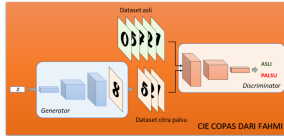
1. Jelaskan dengan ilustrasi gambar sendiri apa itu generator dengan perumpamaan anda sebagai mahasiswa sebagai generatornya.
Tugas Generator sekarang sedang dibuat untuk membuat koleksi gambar palsu, yang saat ini dilihat oleh Diskriminator. Diskriminator tidak dapat membedakan antara yang asli dan yang palsu. Untuk ilustrasi, lihat gambar berikut:



Gambar 8.1 Teori 1

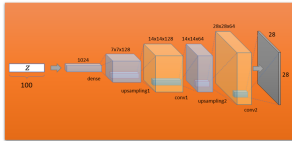
2. Jelaskan dengan ilustrasi gambar sendiri apa itu diskriminator dengan perumpamaan dosen anda sebagai diskriminatornya.

Diskriminator adalah CNN yang menerima input gambar yang dimiliki dan menghasilkan angka biner yang meminta input gambar, lalu menghasilkan gambar dari dataset asli atau menghasilkan gambar palsu. Untuk ilustrasi, lihat gambar berikut:



Gambar 8.2 Teori 2

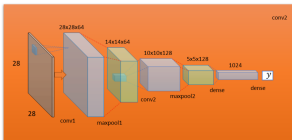
3. Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur generator dibuat. Aksitektur generator dibuat bisa dijelaskan pada gambar berikut :



Gambar 8.3 Teori 3

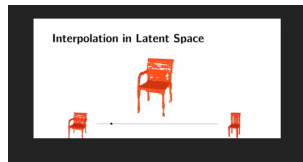
4. Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur diskriminator dibuat.

Aksitektur diskriminator dibuat bisa dijelaskan pada gambar berikut :



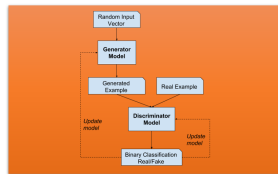
Gambar 8.4 Teori 4

5. Jelaskan dengan ilustrasi gambar apa itu latent space. Latent space dijelaskan pada gambar berikut :



Gambar 8.5 Teori 5

6. Jelaskan dengan ilustrasi gambar apa itu adversarial play.
Adversarial play dijelaskan pada gambar berikut :



Gambar 8.6 Teori 6

7. Jelaskan dengan ilustrasi gambar apa itu Nash equilibrium.
Nash equilibrium adalah Teori permainan adalah studi tentang interaksi strategis antara agen rasional. Sederhananya itu berarti itu adalah studi interaksi ketika pihak-pihak yang terlibat mencoba dan melakukan yang terbaik dari sudut pandang mereka, detailnya dapat dijelaskan pada gambar berikut :

```

In [4]: runfile('D:/FOLDER KHUSUS NGAMPUS/SEMESTER 4/
8/src/1174021/tugas8/teori1.py', wdir='D:/FOLDER KHUSUS
PERTENLOAN KE 8/AI PULL 8/src/1174021/tugas8')
Bi matrix game with payoff matrices:

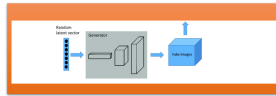
Row player:
[[3 0]
 [4 1]]

Column player:
[[3 4]
 [0 1]]

```

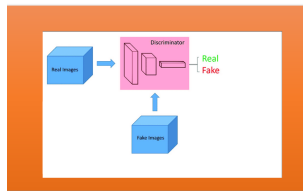
Gambar 8.7 Teori 7

8. Sebutkan dan jelaskan contoh-contoh implementasi dari GAN.
Menurut saya implementasi 3DGAN yaitu pada MAPS dan juga IKEA. Karena pada maps dan juga ikea sudah menerapkan bentuk 3 dimensi yang bisa lebih menarik perhatian pengguna.
9. Berikan contoh dengan penjelasan kode program beserta gambar arsitektur untuk membuat generator(neural network) dengan sebuah input layer, tiga hidden layer(dense layer), dan satu output layer(reshape layer).
Untuk penjelasan tersebut dijelaskan pada gambar dibawah ini :



Gambar 8.8 Teori 9

10. Berikan contoh dengan ilustrasi dari arsitektur diskriminator dengan sebuah input layer, 3 buah hidden layer, dan satu output layer.
Untuk penjelasan tersebut dijelaskan pada gambar dibawah ini :



Gambar 8.9 Teori 10

11. Jelaskan bagaimana kaitan output dan input antara generator dan diskriminator tersebut. Jelaskan kenapa inputan dan outputan seperti itu.
Gambar dari Generator yang berhasil di deteksi oleh Diskriminator sebagai gambar fake, akan dikembalikan dengan feedback pke generator. Kini Generator bertugas untuk bisa membuat sekumpulan gambar palsu, yang nantinya dapat dilihat oleh Diskriminator, lalu, Diskriminator tidak bisa membedakan fake dan real.
12. Jelaskan apa perbedaan antara Kullback-Leibler divergence (KL divergence)/relative entropy / Jensen-Shannon(JS) divergence / information radius(iRaD) / total divergence to the average dalam mengukur kualitas dari model.
Perbedaan nya yaitu memiliki model dari rumus yang berbeda-beda sehingga mempengaruhi hasil train dan test
13. Jelaskan apa itu fungsi objektif yang berfungsi untuk mengukur kesamaan antara gambar yang dibuat dengan yang asli.
Ukuran penting untuk menilai kualitas model. lalu kemudian akan melihat di keseimbangan Nash.
14. Jelaskan apa itu scoring algoritma selain mean square error atau cross entropy seperti The Inception Score dan The Frechet Inception distance.
The inception score adalah algoritma penilaian yang paling banyak digunakan untuk GAN. The Frechet Inception distance adalah Untuk mengatasi berbagai kekurangan Skor awal

15. Jelaskan kelebihan dan kekurangan GAN.

Kelebihan : GAN dapat memvisualisasikan bentuk model menjadi plot. Kemudian pada Kelemahan : model susah untuk implementasikan yang membuat data training menjadi lemah.

8.1.2 Praktek Program

1. Soal 1

Konvolusi 3D. Konvolusi 3D menerapkan filter 3 dimensi ke kumpulan data dan filter 3 arah (x, y, z) untuk menghitung representasi fitur tingkat rendah. Bentuk outputnya adalah ruang volume 3 seperti kubus atau berbentuk kubus. 3D sangat membantu dalam pendeteksian peristiwa dalam video, gambar medis 3D, dll. Generative Adversarial Network adalah arsitektur jaringan saraf tiruan yang dimaksudkan untuk membuat atau membuat data yang benar-benar baru, dari nol hingga tidak ada sama sekali. Melihat target utama GAN adalah data gambar. Singkatnya, jaringan GAN berfungsi untuk memberikan gambar baru berdasarkan koleksi gambar yang telah ada sebelumnya selama proses training.

2. Soal 2

```

1 def build_generator():
2     """
3     Create a Generator Model with hyperparameters values defined
4     as follows
5     """
6     z_size = 200
7     gen_filters = [512, 256, 128, 64, 1]
8     gen_kernel_sizes = [4, 4, 4, 4, 4]
9     gen_strides = [1, 2, 2, 2, 2]
10    gen_input_shape = (1, 1, 1, z_size)
11    gen_activations = ['relu', 'relu', 'relu', 'relu', 'sigmoid']
12    gen_convolutional_blocks = 5
13
14    input_layer = Input(shape=gen_input_shape)
15
16    # First 3D transpose convolution(or 3D deconvolution) block
17    a = Deconv3D(filters=gen_filters[0],
18                kernel_size=gen_kernel_sizes[0],
19                strides=gen_strides[0])(input_layer)
20    a = BatchNormalization()(a, training=True)
21    a = Activation(activation='relu')(a)
22
23    # Next 4 3D transpose convolution(or 3D deconvolution) blocks
24    for i in range(gen_convolutional_blocks - 1):
25        a = Deconv3D(filters=gen_filters[i + 1],
26                    kernel_size=gen_kernel_sizes[i + 1],
27                    strides=gen_strides[i + 1], padding='same')(
28            a)
29        a = BatchNormalization()(a, training=True)
30        a = Activation(activation=gen_activations[i + 1])(a)

```

```

30     gen_model = Model(inputs=[input_layer], outputs=[a])
31     return gen_model

```

Kode di atas akan melakukan create generator ialah gloss, Bentuk jaringan Generator dapat dilihat berkebalikan dengan struktur jaringan saraf pada umumnya. Generator biasanya menerima input sebuah vektor z , yang kemudian mengubahnya menjadi sebuah output 3D atau 3 dimensi.

3. Soal 3

```

1  def build_discriminator():
2      """
3      Create a Discriminator Model using hyperparameters values
4      defined as follows
5      """
6
7      dis_input_shape = (64, 64, 64, 1)
8      dis_filters = [64, 128, 256, 512, 1]
9      dis_kernel_sizes = [4, 4, 4, 4, 4]
10     dis_strides = [2, 2, 2, 2, 1]
11     dis_paddings = ['same', 'same', 'same', 'same', 'valid']
12     dis_alphas = [0.2, 0.2, 0.2, 0.2, 0.2]
13     dis_activations = ['leaky_relu', 'leaky_relu', 'leaky_relu',
14                        'leaky_relu', 'sigmoid']
15
16     dis_convolutional_blocks = 5
17
18     dis_input_layer = Input(shape=dis_input_shape)
19
20     # The first 3D Convolutional block
21     a = Conv3D(filters=dis_filters[0],
22               kernel_size=dis_kernel_sizes[0],
23               strides=dis_strides[0],
24               padding=dis_paddings[0])(dis_input_layer)
25
26     # a = BatchNormalization(a, training=True)
27     a = LeakyReLU(dis_alphas[0])(a)
28
29     # Next 4 3D Convolutional Blocks
30     for i in range(dis_convolutional_blocks - 1):
31         a = Conv3D(filters=dis_filters[i + 1],
32                   kernel_size=dis_kernel_sizes[i + 1],
33                   strides=dis_strides[i + 1],
34                   padding=dis_paddings[i + 1])(a)
35         a = BatchNormalization(a, training=True)
36         if dis_activations[i + 1] == 'leaky_relu':
37             a = LeakyReLU(dis_alphas[i + 1])(a)
38         elif dis_activations[i + 1] == 'sigmoid':
39             a = Activation(activation='sigmoid')(a)
40
41     dis_model = Model(inputs=[dis_input_layer], outputs=[a])
42     return dis_model

```

Diskriminator adalah d_{loss} , Jaringan Discriminator merupakan jaringan klasifikasi biner yang menerima input gambar tiga dimensi dan mengeluarkan klasifikasi menyatakan input gambar adalah gambar asli dari dataset atau merupakan

gambar buatan Generator. Diskriminator dilatih dengan dataset yang diambil dari Generator, lalu di training untuk membedakan keduanya. Gambar dari Generator yang berhasil di deteksi oleh Diskriminator sebagai gambar fake, akan dikembalikan dengan feedback pke generator. Kini Generator bertugas untuk bisa membuat sekumpulan gambar palsu, yang nantinya dapat dilihat oleh Diskriminator, lalu, Diskriminator tidak bisa membedakan fake dan real.

4. Soal 4

Proses training 3D GAN yaitu dengan melakukan epoch sebanyak yang ditentukan.

5. Soal 5

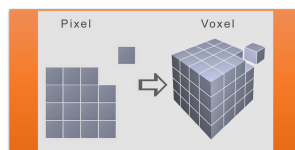
- Clone github
- Download dataset
- Buat folder baru logs dan results

6. Soal 6

Dataset yang digunakan yaitu 3DShapeNets yang berisi model model bentuk benda dll, folder train berisi train dan folder test berisi data testing. dan semua data tersebut di simpan didalam folder volumetric_data.

7. Soal 7

Volume pixel atau voxel adalah titik dalam ruang tiga dimensi. Sebuah voxel mendefinisikan posisi dengan tiga koordinat dalam arah x, y, dan z. Sebuah voxel adalah unit dasar untuk mewakili gambar 3D. Untuk gambar nya ialah sebagai berikut :



Gambar 8.10 Praktek 7

8. Soal 8

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun May 10 18:10:55 2020
4
5 @author: Ikrimaa
6
7
8 """

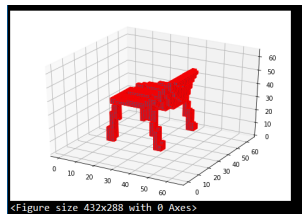
```

```

9
10 # In []
11
12 import scipy.io as io
13 voxels = io.loadmat("data/3DShapeNets/volumetric_data/chair/30/
    test/chair_000000000_1.mat")['instance']
14 #%%
15 import numpy as np
16 voxels = np.pad(voxels, (1, 1), 'constant', constant_values=(0,
    0))
17 #%%
18 import scipy.ndimage as nd
19 voxels = nd.zoom(voxels, (2, 2, 2), mode='constant', order=0)
20 #%%
21 import matplotlib.pyplot as plt
22 from mpl_toolkits.mplot3d import Axes3D
23 fig = plt.figure()
24 ax = Axes3D(fig)
25 ax.voxels(voxels, edgecolor="red")
26
27 plt.show()
28 plt.savefig('data')

```

Kode di atas berfungsi untuk visualisasi dataset dalam tampilan plot. langkah-langkah seperti ini : import library, load data file .mat dan lakukan read memakai matplotlib, Hasilnya adalah sebagai berikut :



Gambar 8.11 Praktek 8

9. Soal 9

```

1 #%% soal9
2
3 def build_generator():
4     """
5     Create a Generator Model with hyperparameters values defined
6     as follows
7     """
8     z_size = 200
9     gen_filters = [512, 256, 128, 64, 1]
10    gen_kernel_sizes = [4, 4, 4, 4, 4]
11    gen_strides = [1, 2, 2, 2, 2]
12    gen_input_shape = (1, 1, 1, z_size)
13    gen_activations = ['relu', 'relu', 'relu', 'relu', 'sigmoid']

```

```

13 gen_convolutional_blocks = 5
14
15 input_layer = Input(shape=gen_input_shape)
16
17 # First 3D transpose convolution(or 3D deconvolution) block
18 a = Deconv3D( filters=gen_filters[0],
19               kernel_size=gen_kernel_sizes[0],
20               strides=gen_strides[0])(input_layer)
21 a = BatchNormalization()(a, training=True)
22 a = Activation(activation='relu')(a)
23
24 # Next 4 3D transpose convolution(or 3D deconvolution) blocks
25 for i in range(gen_convolutional_blocks - 1):
26     a = Deconv3D( filters=gen_filters[i + 1],
27                   kernel_size=gen_kernel_sizes[i + 1],
28                   strides=gen_strides[i + 1], padding='same')(
29     a)
30     a = BatchNormalization()(a, training=True)
31     a = Activation(activation=gen_activations[i + 1])(a)
32
33 gen_model = Model(inputs=[input_layer], outputs=[a])
34 return gen_model

```

Kode di atas berfungsi untuk membuat generator yaitu dengan ketentuan gen sebagai variabel dan membuat fungsi atau variabel genmodel lalu dilakukan return.

10. Soal 10

```

1 #%% soal10
2
3
4  def build_discriminator():
5      """
6      Create a Discriminator Model using hyperparameters values
7      defined as follows
8      """
9
10     dis_input_shape = (64, 64, 64, 1)
11     dis_filters = [64, 128, 256, 512, 1]
12     dis_kernel_sizes = [4, 4, 4, 4, 4]
13     dis_strides = [2, 2, 2, 2, 1]
14     dis_paddings = ['same', 'same', 'same', 'same', 'valid']
15     dis_alphas = [0.2, 0.2, 0.2, 0.2, 0.2]
16     dis_activations = ['leaky_relu', 'leaky_relu', 'leaky_relu',
17                        'leaky_relu', 'sigmoid']
18     dis_convolutional_blocks = 5
19
20     dis_input_layer = Input(shape=dis_input_shape)
21
22     # The first 3D Convolutional block
23     a = Conv3D( filters=dis_filters[0],
24                 kernel_size=dis_kernel_sizes[0],
25                 strides=dis_strides[0],
26                 padding=dis_paddings[0])(dis_input_layer)

```



```

26 # a = BatchNormalization()(a, training=True)
27 a = LeakyReLU(dis_alphas[0])(a)
28
29 # Next 4 3D Convolutional Blocks
30 for i in range(dis_convolutional_blocks - 1):
31     a = Conv3D(filters=dis_filters[i + 1],
32               kernel_size=dis_kernel_sizes[i + 1],
33               strides=dis_strides[i + 1],
34               padding=dis_paddings[i + 1])(a)
35     a = BatchNormalization()(a, training=True)
36     if dis_activations[i + 1] == 'leaky_relu':
37         a = LeakyReLU(dis_alphas[i + 1])(a)
38     elif dis_activations[i + 1] == 'sigmoid':
39         a = Activation(activation='sigmoid')(a)
40
41     dis_model = Model(inputs=[dis_input_layer], outputs=[a])
42     return dis_model

```

Kode di atas berfungsi untuk membangun diskriminator berfungsi untuk mendefinisikan seluruh gambar yang sudah di load generator sebagai gambar fake dan real.

11. Soal 11

Jika interpreter python menjalankan if name == main sebagai program utama, itu ialah menetapkan variabel name untuk memiliki nilai main. Jika file ini sedang di impor dari modul lain, name akan ditetapkan ke nama modul. Nama modul tersedia sebagai nilai untuk name variabel global.

12. Soal 12

```

1  ##### soal 12
2  object_name = "airplane"
3  data_dir = "data/3DShapeNets/volumetric_data/" \
4            "{}/30/train/*.mat".format(object_name)
5  gen_learning_rate = 0.0025
6  dis_learning_rate = 10e-5
7  beta = 0.5
8  batch_size = 1
9  z_size = 200
10 epochs = 1
11 MODE = "train"

```

Kode di atas berfungsi untuk melakukan load dataset dengan ketentuan data yang hanya dalam folder chair pada data train.

13. Soal 13

```

1  ##### soal 13
2  """
3  Create models
4  """
5  gen_optimizer = Adam(lr=gen_learning_rate, beta_1=beta)

```

```

6     dis_optimizer = Adam(lr=dis_learning_rate , beta_1=beta)
7
8     discriminator = build_discriminator()
9     discriminator.compile(loss='binary_crossentropy', optimizer=
10    dis_optimizer)
11
12    generator = build_generator()
13    generator.compile(loss='binary_crossentropy', optimizer=
14    gen_optimizer)

```

Kode di atas menggunakan Adam sebagai algoritma pengoptimalan dan binary_crossentropy sebagai kerugian loss.

14. Soal 14

```

1  ### soal14
2  discriminator.trainable = False
3
4  input_layer = Input(shape=(1, 1, 1, z_size))
5  generated_volumes = generator(input_layer)
6  validity = discriminator(generated_volumes)
7  adversarial_model = Model(inputs=[input_layer], outputs=[
8  validity])
9  adversarial_model.compile(loss='binary_crossentropy',
10  optimizer=gen_optimizer)

```

Kode di atas artinya ialah kita memasukkan random vector kedalam generator model lalu membagi 2 yaitu generated example dan real example, dan meneruskan ke diskriminator model sebagai real atau fake

15. Soal 15

```

1  ### soal15
2  print("Loading data...")
3  volumes = get3DImages(data_dir=data_dir)
4  volumes = volumes[..., np.newaxis].astype(np.float)
5  print("Data loaded...")

```

Kode di atas berfungsi untuk melakukan load data pada dataset.

16. Soal 16

```

1  ### soal16
2  tensorboard = TensorBoard(log_dir="logs/{}".format(time.time
3  ()))
4  tensorboard.set_model(generator)
5  tensorboard.set_model(discriminator)

```

Kode di atas berfungsi untuk membuat tensorboard yang nantinya bisa diakses melalui localhost.

17. Soal 17

```

1  ##### soal17
2      labels_real = np.reshape(np.ones((batch_size,)), (-1, 1, 1,
3      labels_fake = np.reshape(np.zeros((batch_size,)), (-1, 1, 1,
      1, 1))

```

Kode di atas berfungsi untuk melakukan reshape agar shape yang dihasilkan tidak terlalu besar. Dengan membuat variabel real dan fake.

18. Soal 18

```

1  ##### soal18
2      if MODE == 'train':
3          for epoch in range(epochs):
4              print("Epoch:", epoch)
5
6              gen_losses = []
7              dis_losses = []

```

Kode di atas berfungsi untuk melakukan training epoch, karena jika epoch semakin banyak maka kualitas training yang dihasilkan akan semakin baik.

19. Soal 19

```

1  ##### soal19
2      number_of_batches = int(volumes.shape[0] / batch_size
3      )
4      print("Number of batches:", number_of_batches)
5      for index in range(number_of_batches):
6          print("Batch:", index + 1)

```

Batch adalah jumlah file yang akan di training.

20. Soal 20

```

1  ##### soal20
2      z_sample = np.random.normal(0, 0.33, size=[
3      batch_size, 1, 1, 1, z_size]).astype(np.float32)
4      volumes_batch = volumes[index * batch_size:(index
5      + 1) * batch_size, :, :, :]

```

Kode di atas berfungsi untuk untuk membuat gambar bersih dari noise dan juga menyesuaikan shape.

21. Soal 21

```

1  ##### soal21
2      # Next, generate volumes using the generate
3      network
4      gen_volumes = generator.predict_on_batch(z_sample
5      )

```

Kode di atas berfungsi untuk membuat sample gambar palsu yang akan diteruskan ke diskriminator.

22. Soal 22

```

1  ##### soal22
2      discriminator.trainable = True
3      if index % 2 == 0:
4          loss_real = discriminator.train_on_batch(
volumes_batch, labels_real)
5          loss_fake = discriminator.train_on_batch(
gen_volumes, labels_fake)
6
7          d_loss = 0.5 * np.add(loss_real, loss_fake)
8          print("d_loss:{}".format(d_loss))
9
10         else:
11             d_loss = 0.0

```

Kode di atas berfungsi untuk membuat diskriminator bisa load gambar fake dan real dari generator, oleh karena itu ada generator loss dan diskriminator loss untuk melihat seberapa baik kualitas yang dihasilkan.

23. Soal 23

```

1  ##### soal23
2      discriminator.trainable = False
3      """
4      Train the generator network
5      """
6      z = np.random.normal(0, 0.33, size=[batch_size,
1, 1, 1, z_size]).astype(np.float32)
7      g_loss = adversarial_model.train_on_batch(z,
labels_real)
8      print("g_loss:{}".format(g_loss))
9
10     gen_losses.append(g_loss)
11     dis_losses.append(d_loss)

```

Kode di atas berfungsi untuk melakukan print gloss untuk generator dan juga dloss untuk diskriminator.

24. Soal 24

```

1  ##### soal24
2      # Every 10th mini-batch, generate volumes and
save them
3      if index % 10 == 0:
4          z_sample2 = np.random.normal(0, 0.33, size=[
batch_size, 1, 1, 1, z_size]).astype(np.float32)
5          generated_volumes = generator.predict(
z_sample2, verbose=3)

```

```

6         for i, generated_volume in enumerate(
7             generated_volumes[:5]):
8             voxels = np.squeeze(generated_volume)
9             voxels[voxels < 0.5] = 0.
10            voxels[voxels >= 0.5] = 1.
            saveFromVoxels(voxels, "results/img-{}-{}
            -{}".format(epoch, index, i))

```

Mengapa ada perulangan ? karena untuk melakukan perbandingan dari hasil yang sudah didapat.

25. Soal 25

```

1  ##### soal25
2      # Write losses to Tensorboard
3      write_log(tensorboard, 'g_loss', np.mean(gen.losses),
4      epoch)
5      write_log(tensorboard, 'd_loss', np.mean(dis.losses),
6      epoch)

```

TensorBoard adalah sebuah aplikasi web localhost untuk memeriksa dan menyelesaikan grafik dari hasil TensorFlow.

26. Soal 26

```

1  ##### soal26
2      generator.save_weights(os.path.join("models", "
3      generator_weights.h5"))
4      discriminator.save_weights(os.path.join("models", "
5      discriminator_weights.h5"))

```

File H5 adalah file data yang disimpan dalam Format Data Hirarki (HDF). Ini berisi array multidimensi data ilmiah.

27. Soal 27

```

1  ##### soal27
2      if MODE == 'predict':
3          # Create models
4          generator = build_generator()
5          discriminator = build_discriminator()
6
7          # Load model weights
8          generator.load_weights(os.path.join("models", "
9          generator_weights.h5"), True)
10         discriminator.load_weights(os.path.join("models", "
11         discriminator_weights.h5"), True)
12
13         # Generate 3D models
14         z_sample = np.random.normal(0, 1, size=[batch_size, 1, 1,
15         1, z_size]).astype(np.float32)

```

```

13         generated_volumes = generator.predict(z_sample, verbose
14         =3)
15         for i, generated_volume in enumerate(generated_volumes
16         [:2]):
17             voxels = np.squeeze(generated_volume)
18             voxels[voxels < 0.5] = 0.
19             voxels[voxels >= 0.5] = 1.
20             saveFromVoxels(voxels, "results/gen-{}".format(i))

```

Ini adalah tahap akhir untuk melakukan testing dari model yang telah dibuat dan buat model dari yang sudah di create sebelumnya yaitu generator dan diskriminator. Untuk ilustrasi gambar sebagai berikut :



Gambar 8.12 Praktek 27

8.1.3 Penanganan Error

1. ValueError

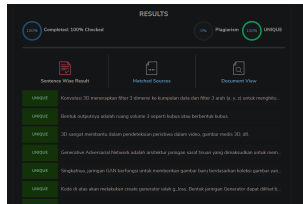
[illegible]

Gambar 8.13 Value error

2. Cara Penanganan Error

- **ValueError**
Error tersebut karena disebabkan gagal load dataset karena salah penamaan direktori.

8.1.4 Bukti Tidak Plagiat



Gambar 8.14 Bukti Tidak Melakukan Plagiat Chapter 8

BAB 9

CHAPTER 9

9.1 1174096 - Nico Ekklesia Sembiring

9.1.1 Soal Teori

1. Jelaskan dengan ilustrasi gambar sendiri apa perbedaan antara vanilla GAN dan cGAN.

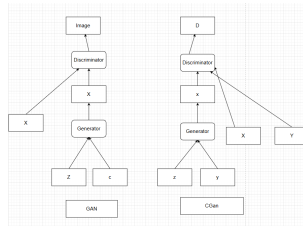
Perbedaan Vanilla GANs dan cGAN adalah sebagai berikut :

Vanilla GANs biasanya tidak memiliki convolutional Neural Jaringan (CNNs) di jaringan mereka.

Conditional GANs (cGANs) adalah perpanjangan dari model GAN. Mereka memungkinkan untuk generasi gambar yang memiliki kondisi tertentu atau atribut dan telah terbukti menjadi lebih baik dari Vanilla GANs sebagai hasilnya.

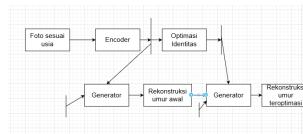
cGANs adalah jenis GAN yang dikondisikan pada beberapa informasi tambahan. informasi tambahan ke Generator sebagai lapisan input tambahan. Dalam Vanilla GANs, tidak ada kontrol atas Kategori gambar yang dihasilkan. Ketika kita menambahkan kondisi y ke Generator, kita dapat menghasilkan gambar dari kategori tertentu, menggunakan y , yang mungkin jenis data, seperti label kelas atau data integer. Vanilla GANs bisa belajar hanya satu kategori dan sangat

sulit untuk arsitek GANs untuk beberapa kategori. Sebuah cGAN, bagaimanapun, dapat digunakan untuk menghasilkan model multi-modal dengan kondisi yang berbeda untuk kategori yang berbeda.



Gambar 9.1 Ilustrasi Vanilla GANs dan cGAN

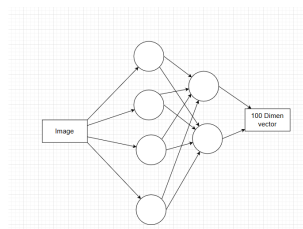
2. Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN. Age cGan melakukan input sesuai dengan wajah beserta umur. Lalu akan dilakukan proses encoding dan selanjutnya melakukan Optimasi identitas. Lalu dihasilkan rekonstruksi awal yang nanti dilanjutkan dengan generator yang mengoptimasi rekonstruksi.



Gambar 9.2 ilustrasi Age-cGAN

3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari AgecGAN.

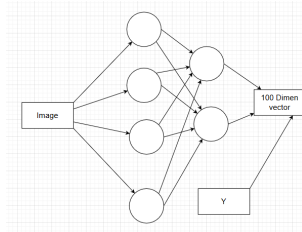
Encoder network pada Age cGAN menghasilkan vektor dari gambar yang dimasukkan. Encoder network sendiri merupakan sebuah CNN yang mengambil gambar dengan dimensi sekitar 64,64,3 dan mengkonversikannya menjadi vektor dengan dimensi 100



Gambar 9.3 Ilustrasi arsitektur encoder AgecGAN

4. Jelaskan dengan ilustrasi gambar sendiri arsitektur generator network dari AgecGAN

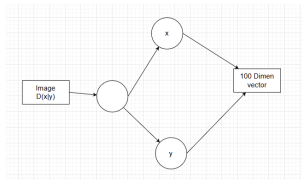
Generator network digunakan mengambil representasi yang tersembunyi dari fotowajah dan kondisi vektor sesuai dengan kondisi yang ada. Generator network sendiri memiliki vektor 100 dimensi dan kondisi vektor Y.



Gambar 9.4 ilustrasi arsitektur generator network AgecGAN

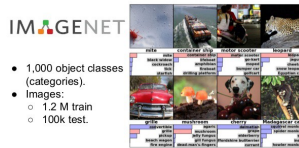
5. Jelaskan dengan ilustrasi gambar sendiri arsitektur discriminator network dari AgecGAN.

Jaringan diskriminator digunakan untuk mengidentifikasi apakah gambar yang disediakan adalah palsu atau nyata. Hal ini dilakukan dengan melewati gambar melalui serangkaian lapisan sampling bawah dan beberapa lapisan klasifikasi. Dengan kata lain, ini memprediksi Apakah gambar itu nyata atau palsu. Seperti jaringan lain, Jaringan diskriminator lain dalam jaringan convolutional. Ini berisi beberapa blok convolutional. Setiap blok convolutional berisi lapisan convolutional, lapisan normalisasi batch, dan fungsi aktivasi, selain blok convolutional pertama, yang tidak memiliki lapisan normalisasi batch.



Gambar 9.5 Ilustrasi discriminator network AgecGAN

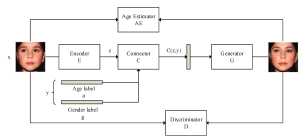
6. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model. Inception Resnet 2 model pre trained merupakan sebuah package untuk mendefinisikan, melatih, dan mengevaluasi sesuai dengan checkpoint dan definisi model untuk beberapa network pada klasifikasi gambar.



Gambar 9.6 Ilustrasi pretrained Inception-ResNet-2 Model

7. Jelaskan dengan ilustrasi gambar sendiri arsitektur Face recognition network Age-cGAN.

Face Recognition network Age-cGAN digunakan untuk dapat mengenali identitas seseorang dalam gambar yang diberikan



Gambar 9.7 ilustrasi arsitektur Face recognition network Age-cGAN

8. Sebutkan dan jelaskan serta di sertai contoh-contoh tahapan dari Age-cGAN
Age-cGAN memiliki beberapa tahapan pelatihan. Age-cGAN memiliki empat jaringan, yang dilatih dalam tiga tahap. Pelatihan AgecGAN terdiri dari tiga tahap:

- pelatihan GAN bersyarat: pada tahap ini, kita melatih jaringan Generator dan jaringan diskriminator.
- awal pendekatan vektor laten: pada tahap ini, kami melatih jaringan Encoder.
- optimasi vektor laten: pada tahap ini, kami mengoptimalkan kedua encoder dan jaringan generator.

9. Berikan contoh perhitungan fungsi training objektif

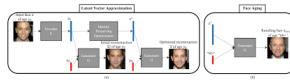
Objektif Training ialah untuk meminimalkan loss function sebagai log likelihood function yang diberikan pada persamaan dimana D melambangkan training data.

$$L(\theta) = - \sum_{\{\mathbf{x}, \mathbf{y}\} \in \mathbf{D}} \log p(\mathbf{y} | \mathbf{x}, \theta)$$

Gambar 9.8 ilustrasi perhitungan fungsi training objektif

10. Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation

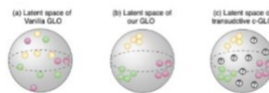
Latent Vector Aproximation adalah metode untuk memperkirakan vektor laten untuk mengoptimalkan rekonstruksi gambar wajah. Untuk memperkirakan vektor laten, kami memiliki jaringan Encoder. Kami melatih jaringan Encoder pada gambar yang dihasilkan dan gambar nyata. Setelah dilatih, Jaringan Encoder akan mulai menghasilkan vektor laten dari Distribusi. Tujuan pelatihan fungsi untuk pelatihan jaringan Encoder adalah kehilangan jarak Euclidean.



Gambar 9.9 ilustrasi Initial latent vector approximation

11. Berikan contoh perhitungan latent vector optimization

Perhitungan latent optimization menggunakan metode yang relatif sederhana, tergantung pada jumlah kecil parameter yang diperlukan, sehingga pada latent optimization dapat memetakan setiap gambar x dari dataset ke vektor acak dimensi rendah z dalam ruang laten z .



Gambar 9.10 ilustrasi perhitungan latent vector optimization

9.1.2 Praktek Program

1. Jelaskan bagaimana cara ekstrak file dataset Age-cGAN menggunakan google colab

Cara yang dilakukan untuk ekstraksi file melalui google colab yaitu dengan mengopy path dari file yang telah ada di drive kedalam code ekstraksi

```
1 # In [1. Ekstrak File]:
2 import tarfile
3 tf = tarfile.open("/content/drive/My Drive/Colab Notebooks /
4   wiki_crop.tar")
5 tf.extractall(path="/content/drive/My Drive/Colab Notebooks")
```

2. Jelaskan bagaimana kode program bekerja untuk melakukan load terhadap dataset yang sudah di ekstrak, termasuk bagaimana penjelasan kode program perhitungan usia

Dalam melakukan load dataset yang telah diekstrak, terdapat beberapa proses, yaitu melakukan load file wiki.mat. Setelah itu melakukan load list file. Kemudian list serial number matlab, tahun pengambilan foto. Lalu mengkalkulasikan umur. setelah itu membuat list tuple, lalu mereturn list semua gambar dan perkiraan umur.

```

1 # In[2. Load Data]:
2 def load_data(wiki_dir, dataset='wiki'):
3     # Load the wiki.mat file
4     meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset)))
5
6     # Load the list of all files
7     full_path = meta[dataset][0, 0]["full_path"][0]
8
9     # List of Matlab serial date numbers
10    dob = meta[dataset][0, 0]["dob"][0]
11
12    # List of years when photo was taken
13    photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
14
15    # Calculate age for all dobs
16    age = [calculate_age(photo_taken[i], dob[i]) for i in range(
17        len(dob))]
18
19    # Create a list of tuples containing a pair of an image path
20    # and age
21    images = []
22    age_list = []
23    for index, image_path in enumerate(full_path):
24        images.append(image_path[0])
25        age_list.append(age[index])
26
27    # Return a list of all images and respective age
28    return images, age_list

```

3. Jelaskan bagaimana kode program The Encoder Network bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana
Proses Encoder berfungsi untuk mempelajari pemetaan terbalik dari gambar wajah dan kondisi usia dengan vector latent Z.

```

1 # In[3. Encoder Bekerja]:
2 def build_encoder():
3     """
4     Encoder Network
5     """
6     input_layer = Input(shape=(64, 64, 3))
7
8     # 1st Convolutional Block
9     enc = Conv2D(filters=32, kernel_size=5, strides=2, padding='
10    same')(input_layer)
11    # enc = BatchNormalization()(enc)
12    enc = LeakyReLU(alpha=0.2)(enc)
13
14    # 2nd Convolutional Block
15    enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='
16    same')(enc)
17    enc = BatchNormalization()(enc)
18    enc = LeakyReLU(alpha=0.2)(enc)
19
20    # 3rd Convolutional Block

```

```

19     enc = Conv2D(filters=128, kernel_size=5, strides=2, padding='
    same')(enc)
20     enc = BatchNormalization()(enc)
21     enc = LeakyReLU(alpha=0.2)(enc)
22
23     # 4th Convolutional Block
24     enc = Conv2D(filters=256, kernel_size=5, strides=2, padding='
    same')(enc)
25     enc = BatchNormalization()(enc)
26     enc = LeakyReLU(alpha=0.2)(enc)
27
28     # Flatten layer
29     enc = Flatten()(enc)
30
31     # 1st Fully Connected Layer
32     enc = Dense(4096)(enc)
33     enc = BatchNormalization()(enc)
34     enc = LeakyReLU(alpha=0.2)(enc)
35
36     # Second Fully Connected Layer
37     enc = Dense(100)(enc)
38
39     # Create a model
40     model = Model(inputs=[input_layer], outputs=[enc])
41     return model

```

4. Jelaskan bagaimana kode program The Generator Network bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana

Proses Generator agar bekerja dengan baik dibutuhkan representasi dari gambar wajah dan vector kondisi sebagai inputan yang menghasilkan sebuah gambar.

```

1 # In[4. Generator Network Bekerja]:
2 def build_generator():
3     """
4     Create a Generator Model with hyperparameters values defined
5     as follows
6     """
7     latent_dims = 100
8     num_classes = 6
9
10    input_z_noise = Input(shape=(latent_dims,))
11    input_label = Input(shape=(num_classes,))
12
13    x = concatenate([input_z_noise, input_label])
14
15    x = Dense(2048, input_dim=latent_dims + num_classes)(x)
16    x = LeakyReLU(alpha=0.2)(x)
17    x = Dropout(0.2)(x)
18
19    x = Dense(256 * 8 * 8)(x)
20    x = BatchNormalization()(x)
21    x = LeakyReLU(alpha=0.2)(x)
22    x = Dropout(0.2)(x)
23
24    x = Reshape((8, 8, 256))(x)

```

```

25     x = UpSampling2D(size=(2, 2))(x)
26     x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
27     x = BatchNormalization(momentum=0.8)(x)
28     x = LeakyReLU(alpha=0.2)(x)
29
30     x = UpSampling2D(size=(2, 2))(x)
31     x = Conv2D(filters=64, kernel_size=5, padding='same')(x)
32     x = BatchNormalization(momentum=0.8)(x)
33     x = LeakyReLU(alpha=0.2)(x)
34
35     x = UpSampling2D(size=(2, 2))(x)
36     x = Conv2D(filters=3, kernel_size=5, padding='same')(x)
37     x = Activation('tanh')(x)
38
39     model = Model(inputs=[input_z_noise, input_label], outputs=[x
40 ])
41     return model

```

5. Jelaskan bagaimana kode program The Discriminator Network bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana
Proses Discriminator untuk membedakan antara gambar asli dan gambar palsu.

```

1  # In[5.Discriminator Network Bekerja]:
2  def build_discriminator():
3      """
4      Create a Discriminator Model with hyperparameters values
5      defined as follows
6      """
7      input_shape = (64, 64, 3)
8      label_shape = (6,)
9      image_input = Input(shape=input_shape)
10     label_input = Input(shape=label_shape)
11
12     x = Conv2D(64, kernel_size=3, strides=2, padding='same')(
13         image_input)
14     x = LeakyReLU(alpha=0.2)(x)
15
16     label_input1 = Lambda(expand_label_input)(label_input)
17     x = concatenate([x, label_input1], axis=3)
18
19     x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
20     x = BatchNormalization()(x)
21     x = LeakyReLU(alpha=0.2)(x)
22
23     x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
24     x = BatchNormalization()(x)
25     x = LeakyReLU(alpha=0.2)(x)
26
27     x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
28     x = BatchNormalization()(x)
29     x = LeakyReLU(alpha=0.2)(x)
30
31     x = Flatten()(x)
32     x = Dense(1, activation='sigmoid')(x)
33
34     model = Model(inputs=[image_input, label_input], outputs=[x])

```

```
33     return model
```

6. Jelaskan bagaimana kode program Training cGAN bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana
Proses Training cGAN ini dengan load file .mat pada dataset lalu epoch sebanyak 500 kali.

```
1 # In[6. Training cGAN]:
2     if __name__ == '__main__':
3         # Define hyperparameters
4         data_dir = "data"
5         wiki_dir = os.path.join(data_dir, "wiki_crop1")
6         epochs = 500
7         batch_size = 2
8         image_shape = (64, 64, 3)
9         z_shape = 100
10        TRAIN_GAN = True
11        TRAIN_ENCODER = False
12        TRAIN_GAN_WITH_FR = False
13        fr_image_shape = (192, 192, 3)
14
15        # Define optimizers
16        dis_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
17                               epsilon=10e-8)
18        gen_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
19                               epsilon=10e-8)
20        adversarial_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=
21                                     =0.999, epsilon=10e-8)
```

7. Jelaskan bagaimana kode program Initial dan latent vector approximation bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana
Initial dan Latent Vector Approximation bekerja melakukan prediksi epoch yang telah di buat sebanyak 500 kali, dan nanti hasilnya ada di folder result.

```
1 # In[7. Laten Vector]:
2     """
3     Train encoder
4     """
5
6     if TRAIN_ENCODER:
7         # Build and compile encoder
8         encoder = build_encoder()
9         encoder.compile(loss=euclidean_distance_loss, optimizer='
10                        adam')
11
12        # Load the generator network's weights
13        try:
14            generator.load_weights("generator.h5")
15        except Exception as e:
16            print("Error:", e)
17
18        z_i = np.random.normal(0, 1, size=(5000, z_shape))
19
20        y = np.random.randint(low=0, high=6, size=(5000,), dtype=
21                                np.int64)
```



```

20     num_classes = len(set(y))
21     y = np.reshape(np.array(y), [len(y), 1])
22     y = to_categorical(y, num_classes=num_classes)
23
24     for epoch in range(epochs):
25         print("Epoch:", epoch)
26
27         encoder_losses = []
28
29         number_of_batches = int(z_i.shape[0] / batch_size)
30         print("Number of batches:", number_of_batches)
31         for index in range(number_of_batches):
32             print("Batch:", index + 1)
33
34             z_batch = z_i[index * batch_size:(index + 1) *
35 batch_size]
36             y_batch = y[index * batch_size:(index + 1) *
37 batch_size]
38
39             generated_images = generator.predict_on_batch([
40 z_batch, y_batch])
41
42             # Train the encoder model
43             encoder_loss = encoder.train_on_batch(
44 generated_images, z_batch)
45             print("Encoder loss:", encoder_loss)
46
47             encoder_losses.append(encoder_loss)
48
49             # Write the encoder loss to Tensorboard
50             write_log(tensorboard, "encoder_loss", np.mean(
51 encoder_losses), epoch)
52
53             # Save the encoder model
54             encoder.save_weights("encoder.h5")

```

9.1.3 Penanganan Error

1. IndentationError: unexpected indent



```

D:\test\test_indent.py, line 2
if __name__ == '__main__':
    print('Hello, World!')
IndentationError: unexpected indent

```

Gambar 9.11 IndentationError

2. Tuliskan Kode Error dan Jenis Error

- IndentationError

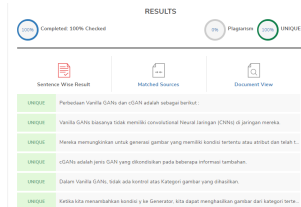
3. Cara Penangan Error

- IndentationError

Error terjadi karena indentasi atau letak kode berada di posisi yang tidak se-

harusnya. cara untuk mengatasi error tersebut adalah dengan memperbaiki posisi indentasi kode sesuai dengan penulisan kode python yang benar.

9.1.4 Bukti Tidak Plagiat



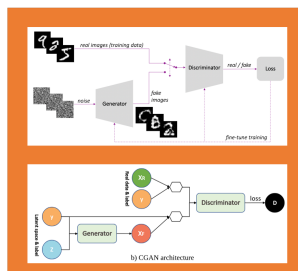
Gambar 9.12 Bukti Tidak Melakukan Plagiat Chapter 9

9.2 1164013 - Ikrima Ningrumsari Mulyana

9.2.1 Teori

1. Jelaskan dengan ilustrasi gambar sendiri apa perbedaan antara vanilla GAN dan cGAN.

Perbedaan antara vanilla GAN dan cGAN terdapat pada saat input proses generator, vanilla GAN memakai data noise yang di proses menjadi data fake, kalau cGAN memakai latent space atau label untuk generator. Untuk ilustrasi, lihat gambar berikut:



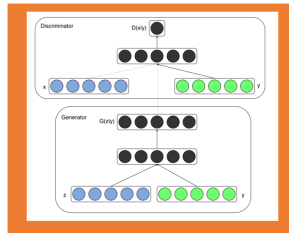
Gambar 9.13 Teori 1

2. Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN. Untuk arsitektur dari Age cGAN mempunyai 4 yaitu :

- Encoder
- FaceNet
- Generator

- Discriminator

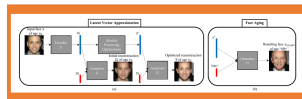
Untuk ilustrasi, lihat gambar berikut:



Gambar 9.14 Teori 2

3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari AgecGAN.

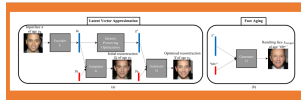
Encoder mempelajari pemetaan terbalik dari gambar wajah input dan kondisi usia dengan vektor laten Z . Jaringan encoder menghasilkan vektor laten dari gambar input. Jaringan Encoder adalah CNN yang mengambil gambar dari dimensi (64, 64, 3) dan mengubahnya menjadi vektor 100 dimensi. Ada empat blok konvolusional dan dua lapisan padat. Setiap blok konvolusional memiliki lapisan konvolusional, diikuti oleh lapisan normalisasi batch, dan fungsi aktivasi kecuali lapisan konvolusional pertama.



Gambar 9.15 Teori 3

4. Jelaskan dengan ilustrasi gambar sendiri arsitektur generator network dari AgecGAN.

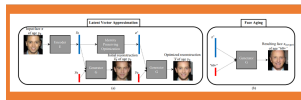
Generator dibutuhkan representasi tersembunyi dari gambar wajah dan vektor kondisi sebagai input dan menghasilkan gambar. Generator adalah CNN dan dibutuhkan vektor laten 100 dimensi dan vektor kondisi y , dan mencoba menghasilkan gambar realistis dari dimensi (64, 64, 3). Generator memiliki lapisan padat, membingungkan, dan konvolusional. Dibutuhkan dua input satu adalah vektor noise dan yang kedua adalah vektor kondisi. Vektor kondisi adalah informasi tambahan yang disediakan untuk jaringan. Untuk Age-cGAN, ini akan menjadi age.



Gambar 9.16 Teori 4

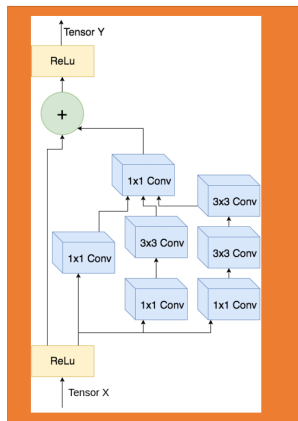
5. Jelaskan dengan ilustrasi gambar sendiri arsitektur discriminator network dari Age-cGAN.

Diskriminator mencoba membedakan antara gambar asli dan gambar palsu. Diskriminator adalah CNN dan memprediksi gambar yang diberikan adalah nyata atau palsu. Ada beberapa blok konvolusional. Setiap blok konvolusional berisi lapisan konvolusional yang diikuti oleh lapisan normalisasi batch, dan fungsi aktivasi, kecuali blok konvolusional pertama, yang tidak memiliki lapisan normalisasi batch.



Gambar 9.17 Teori 5

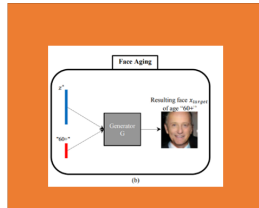
6. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model. Model Inception-ResNet v2 adalah model yang ciptakan untuk keperluan klasifikasi image dengan bobot di ImageNet.



Gambar 9.18 Teori 6

7. Jelaskan dengan ilustrasi gambar sendiri arsitektur Face recognition network Age-cGAN.

FaceNet: Ini adalah jaringan pengenalan wajah yang mempelajari perbedaan antara gambar input x dan gambar yang direkonstruksi x . FaceNet mengenali identitas seseorang dalam gambar yang diberikan. Model Inception, ResNet-50 atau Inception-ResNet-2 yang telah dilatih sebelumnya tanpa lapisan yang terhubung sepenuhnya dapat digunakan. Embedding yang diekstraksi untuk gambar asli dan gambar yang direkonstruksi dapat dihitung dengan menghitung jarak Euclidean dari embeddings.



Gambar 9.19 Teori 7

8. Sebutkan dan jelaskan serta di sertai contoh-contoh tahapan dari Age-cGAN. Untuk tahapan dari Age cGAN yaitu :

- Input
- Training
- Testing

9. Berikan contoh perhitungan fungsi training objektif

Untuk penjelasan tersebut dijelaskan pada gambar dibawah ini : Fungsi obyektif training untuk training cGAN Dimana $\log D(x, y)$ adalah kerugian untuk model Diskriminator. $\log (1 - D(G(x, y'), y'))$ adalah kerugian untuk model Generator. $P(\text{data})$ adalah distribusi dari semua gambar yang mungkin.

$$\min_{\theta_G} \max_{\theta_D} v(\theta_G, \theta_D) = \mathbb{E}_{x, y \sim p_{\text{data}}} [\log D(x, y)] + \mathbb{E}_{z \sim p_z(z)} \mathbb{E}_{\tilde{y} \sim p_y} [\log (1 - D(G(z, \tilde{y}), \tilde{y}))]$$

Gambar 9.20 Teori 9

10. Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation.

Initial latent vector approximation: Encoder network training adakah sebuah metode perkiraan awal vektor laten digunakan untuk memperkirakan vektor laten untuk mengoptimalkan rekonstruksi gambar wajah. Encoder adalah jaringan saraf yang mendekati vektor laten. Kami melatih jaringan encoder pada gambar yang dihasilkan dan gambar nyata. Setelah dilatih, jaringan encoder akan

mulai menghasilkan vektor laten dari distribusi yang dipelajari. Fungsi tujuan pelatihan untuk melatih jaringan encoder adalah kehilangan jarak Euclidean.

11. Berikan contoh perhitungan latent vector optimization

$$z^*_{IP} = \underset{z}{\operatorname{argmin}} \|FR(x) - FR(\hat{x})\|_{L_2}$$

Gambar 9.21 Teori 11

9.2.2 Praktek Program

1. Jelaskan bagaimana cara ekstrak file dataset Age-cGAN menggunakan google colab.

```
1 from google.colab import drive
2 drive.mount('/content/drive')
3
4 import tarfile
5 tf = tarfile.open("/content/drive/My Drive/Colab Notebooks/GIT CH
6 tf.extractall(path="/content/drive/My Drive/Colab Notebooks/GIT
   9/data/wiki_crop.tar")
   CH 9/data")
```

Kode di atas akan melakukan mount dan extract dataset.

- Login ke google colab menggunakan akun google
 - Mount google drive
 - Lakukan proses unzip melalui notebook python di google colab, unzip pakai codingan
 - Selesai
2. Jelaskan bagaimana kode program bekerja untuk melakukan load terhadap dataset yang sudah di ekstrak, termasuk bagaimana penjelasan kode program perhitungan usia.

```
1 def calculate_age(taken, dob):
2     birth = datetime.fromordinal(max(int(dob) - 366, 1))
3
4     if birth.month < 7:
5         return taken - birth.year
6     else:
7         return taken - birth.year - 1
8
9     """
10 def load_data(wiki_dir, dataset='wiki'):
```

```

11 # Load the wiki.mat file
12 meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset
13 )))
14 # Load the list of all files
15 full_path = meta[dataset][0, 0]["full_path"][0]
16
17 # List of Matlab serial date numbers
18 dob = meta[dataset][0, 0]["dob"][0]
19
20 # List of years when photo was taken
21 photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
22
23 # Calculate age for all dobs
24 age = [calculate_age(photo_taken[i], dob[i]) for i in range(
25 len(dob))]
26
27 # Create a list of tuples containing a pair of an image path
28 # and age
29 images = []
30 age_list = []
31 for index, image_path in enumerate(full_path):
32     images.append(image_path[0])
33     age_list.append(age[index])

```

Kode di atas untuk load data dan melakukan fungsi perhitungan usia.

3. Jelaskan bagaimana kode program The Encoder Network bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana.

```

1 def build_encoder():
2     """
3     Encoder Network
4     """
5     input_layer = Input(shape=(64, 64, 3))
6
7     # 1st Convolutional Block
8     enc = Conv2D(filters=32, kernel_size=5, strides=2, padding='
9     same')(input_layer)
10    # enc = BatchNormalization()(enc)
11    enc = LeakyReLU(alpha=0.2)(enc)
12
13    # 2nd Convolutional Block
14    enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='
15    same')(enc)
16    enc = BatchNormalization()(enc)
17    enc = LeakyReLU(alpha=0.2)(enc)
18
19    # 3rd Convolutional Block
20    enc = Conv2D(filters=128, kernel_size=5, strides=2, padding='
21    same')(enc)
22    enc = BatchNormalization()(enc)
23    enc = LeakyReLU(alpha=0.2)(enc)
24
25    # 4th Convolutional Block

```

```

23     enc = Conv2D(filters=256, kernel_size=5, strides=2, padding='
    same')(enc)
24     enc = BatchNormalization()(enc)
25     enc = LeakyReLU(alpha=0.2)(enc)
26
27     # Flatten layer
28     enc = Flatten()(enc)
29
30     # 1st Fully Connected Layer
31     enc = Dense(4096)(enc)
32     enc = BatchNormalization()(enc)
33     enc = LeakyReLU(alpha=0.2)(enc)
34
35     # Second Fully Connected Layer
36     enc = Dense(100)(enc)
37
38     # Create a model
39     model = Model(inputs=[input_layer], outputs=[enc])
40     return model

```

Encoder berfungsi untuk mempelajari pemetaan terbalik dari gambar wajah input dan kondisi usia dengan vektor laten Z.

4. Jelaskan bagaimana kode program The Generator Network bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana.

```

1 def build_generator():
2     """
3     Create a Generator Model with hyperparameters values defined
4     as follows
5     """
6     latent_dims = 100
7     num_classes = 6
8
9     input_z_noise = Input(shape=(latent_dims,))
10    input_label = Input(shape=(num_classes,))
11
12    x = concatenate([input_z_noise, input_label])
13
14    x = Dense(2048, input_dim=latent_dims + num_classes)(x)
15    x = LeakyReLU(alpha=0.2)(x)
16    x = Dropout(0.2)(x)
17
18    x = Dense(256 * 8 * 8)(x)
19    x = BatchNormalization()(x)
20    x = LeakyReLU(alpha=0.2)(x)
21    x = Dropout(0.2)(x)
22
23    x = Reshape((8, 8, 256))(x)
24
25    x = UpSampling2D(size=(2, 2))(x)
26    x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
27    x = BatchNormalization(momentum=0.8)(x)
28    x = LeakyReLU(alpha=0.2)(x)

```



```

29     x = UpSampling2D(size=(2, 2))(x)
30     x = Conv2D(filters=64, kernel_size=5, padding='same')(x)
31     x = BatchNormalization(momentum=0.8)(x)
32     x = LeakyReLU(alpha=0.2)(x)
33
34     x = UpSampling2D(size=(2, 2))(x)
35     x = Conv2D(filters=3, kernel_size=5, padding='same')(x)
36     x = Activation('tanh')(x)
37
38     model = Model(inputs=[input_z_noise, input_label], outputs=[x])
39     return model

```

Generator network agar bekerja dengan baik dibutuhkan representasi tersembunyi dari gambar wajah dan vektor kondisi sebagai input dan menghasilkan gambar.

5. Jelaskan bagaimana kode program The Discriminator Network bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana.

```

1 def build_discriminator():
2     """
3     Create a Discriminator Model with hyperparameters values
4     defined as follows
5     """
6     input_shape = (64, 64, 3)
7     label_shape = (6,)
8     image_input = Input(shape=input_shape)
9     label_input = Input(shape=label_shape)
10
11     x = Conv2D(64, kernel_size=3, strides=2, padding='same')(
12         image_input)
13     x = LeakyReLU(alpha=0.2)(x)
14
15     label_input1 = Lambda(expand_label_input)(label_input)
16     x = concatenate([x, label_input1], axis=3)
17
18     x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
19     x = BatchNormalization()(x)
20     x = LeakyReLU(alpha=0.2)(x)
21
22     x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
23     x = BatchNormalization()(x)
24     x = LeakyReLU(alpha=0.2)(x)
25
26     x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
27     x = BatchNormalization()(x)
28     x = LeakyReLU(alpha=0.2)(x)
29
30     x = Flatten()(x)
31     x = Dense(1, activation='sigmoid')(x)
32
33     model = Model(inputs=[image_input, label_input], outputs=[x])
34     return model

```

Diskriminator mencoba untuk membedakan antara gambar asli dan gambar palsu.

6. Jelaskan bagaimana kode program Training cGAN bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana.

```

1  if __name__ == '__main__':
2      # Define hyperparameters
3      data_dir = "data"
4      wiki_dir = os.path.join(data_dir, "wiki_cropl")
5      epochs = 500
6      batch_size = 2
7      image_shape = (64, 64, 3)
8      z_shape = 100
9      TRAIN_GAN = True
10     TRAIN_ENCODER = False
11     TRAIN_GAN_WITH_FR = False
12     fr_image_shape = (192, 192, 3)
13
14     # Define optimizers
15     dis_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
16         epsilon=10e-8)
17     gen_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
18         epsilon=10e-8)
19     adversarial_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2
20         =0.999, epsilon=10e-8)
21
22     """
23     Build and compile networks
24     """
25
26     # Build and compile the discriminator network
27     discriminator = build_discriminator()
28     discriminator.compile(loss=['binary_crossentropy'], optimizer
29         =dis_optimizer)
30
31     # Build and compile the generator network
32     generator = build_generator()
33     generator.compile(loss=['binary_crossentropy'], optimizer=
34         gen_optimizer)
35
36     # Build and compile the adversarial model
37     discriminator.trainable = False
38     input_z_noise = Input(shape=(100,))
39     input_label = Input(shape=(6,))
40     recons_images = generator([input_z_noise, input_label])
41     valid = discriminator([recons_images, input_label])
42     adversarial_model = Model(inputs=[input_z_noise, input_label
43         ], outputs=[valid])
44     adversarial_model.compile(loss=['binary_crossentropy'],
45         optimizer=gen_optimizer)
46
47     tensorboard = TensorBoard(log_dir="logs/{}".format(time.time
48         ()))
49     tensorboard.set_model(generator)
50     tensorboard.set_model(discriminator)

```

```

42     """
43
44     Load the dataset
45     """
46     images, age_list = load_data(wiki_dir=wiki_dir, dataset="wiki")
47     age_cat = age_to_category(age_list)
48     final_age_cat = np.reshape(np.array(age_cat), [len(age_cat),
49                                     1])
49     classes = len(set(age_cat))
50     y = to_categorical(final_age_cat, num_classes=len(set(age_cat)))
51
52     loaded_images = load_images(wiki_dir, images, (image_shape
53                                         [0], image_shape[1]))
54
55     # Implement label smoothing
56     real_labels = np.ones((batch_size, 1), dtype=np.float32) *
57         0.9
58     fake_labels = np.zeros((batch_size, 1), dtype=np.float32) *
59         0.1
60
61     """
62     Train the generator and the discriminator network
63     """
64     if TRAIN_GAN:
65         for epoch in range(epochs):
66             print("Epoch:{}".format(epoch))
67
68             gen_losses = []
69             dis_losses = []
70
71             number_of_batches = int(len(loaded_images) /
72                                     batch_size)
73             print("Number of batches:", number_of_batches)
74             for index in range(number_of_batches):
75                 print("Batch:{}".format(index + 1))
76
77                 images_batch = loaded_images[index * batch_size:(
78                     index + 1) * batch_size]
79                 images_batch = images_batch / 127.5 - 1.0
80                 images_batch = images_batch.astype(np.float32)
81
82                 y_batch = y[index * batch_size:(index + 1) *
83                     batch_size]
84                 z_noise = np.random.normal(0, 1, size=(batch_size
85                     , z_shape))
86
87                 """
88                 Train the discriminator network
89                 """
90
91                 # Generate fake images
92                 initial_recon_images = generator.predict_on_batch
93                     ([z_noise, y_batch])

```

```

87         d_loss_real = discriminator.train_on_batch([
images_batch, y_batch], real_labels)
88         d_loss_fake = discriminator.train_on_batch([
initial_recon_images, y_batch], fake_labels)
89
90         d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)
91         print("d_loss:{}".format(d_loss))
92
93         """
94         Train the generator network
95         """
96
97         z_noise2 = np.random.normal(0, 1, size=(
batch_size, z_shape))
98         random_labels = np.random.randint(0, 6,
batch_size).reshape(-1, 1)
99         random_labels = to_categorical(random_labels, 6)
100
101         g_loss = adversarial_model.train_on_batch([
z_noise2, random_labels], [1] * batch_size)
102
103         print("g_loss:{}".format(g_loss))
104
105         gen_losses.append(g_loss)
106         dis_losses.append(d_loss)
107
108         # Write losses to Tensorboard
109         write_log(tensorboard, 'g_loss', np.mean(gen_losses),
epoch)
110         write_log(tensorboard, 'd_loss', np.mean(dis_losses),
epoch)
111
112         """
113         Generate images after every 10th epoch
114         """
115         if epoch % 10 == 0:
116             images_batch = loaded_images[0:batch_size]
117             images_batch = images_batch / 127.5 - 1.0
118             images_batch = images_batch.astype(np.float32)
119
120             y_batch = y[0:batch_size]
121             z_noise = np.random.normal(0, 1, size=(batch_size
, z_shape))
122
123             gen_images = generator.predict_on_batch([z_noise,
y_batch])
124
125             for i, img in enumerate(gen_images[:5]):
126                 save_rgb_img(img, path="results/img-{}-{}.png
".format(epoch, i))
127
128         # Save networks
129         try:
130             generator.save_weights("generator.h5")
131             discriminator.save_weights("discriminator.h5")
132         except Exception as e:

```

```
133         print("Error:", e)
```

Proses training dengan load file .mat pada dataset, lalu epoch sebanyak 500 kali.

7. Jelaskan bagaimana kode program Initial dan latent vector approximation bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana.

```

1  if TRAIN_ENCODER:
2      # Build and compile encoder
3      encoder = build_encoder()
4      encoder.compile(loss=euclidean_distance_loss, optimizer='
adam')
5
6      # Load the generator network's weights
7      try:
8          generator.load_weights("generator.h5")
9      except Exception as e:
10         print("Error:", e)
11
12         z_i = np.random.normal(0, 1, size=(5000, z_shape))
13
14         y = np.random.randint(low=0, high=6, size=(5000,), dtype=
np.int64)
15         num_classes = len(set(y))
16         y = np.reshape(np.array(y), [len(y), 1])
17         y = to_categorical(y, num_classes=num_classes)
18
19         for epoch in range(epochs):
20             print("Epoch:", epoch)
21
22             encoder_losses = []
23
24             number_of_batches = int(z_i.shape[0] / batch_size)
25             print("Number of batches:", number_of_batches)
26             for index in range(number_of_batches):
27                 print("Batch:", index + 1)
28
29                 z_batch = z_i[index * batch_size:(index + 1) *
batch_size]
30                 y_batch = y[index * batch_size:(index + 1) *
batch_size]
31
32                 generated_images = generator.predict_on_batch([
z_batch, y_batch])
33
34                 # Train the encoder model
35                 encoder_loss = encoder.train_on_batch(
generated_images, z_batch)
36                 print("Encoder loss:", encoder_loss)
37
38                 encoder_losses.append(encoder_loss)
39
40                 # Write the encoder loss to Tensorboard
41                 write_log(tensorboard, "encoder_loss", np.mean(
encoder_losses), epoch)

```

```

42     # Save the encoder model
43     encoder.save_weights("encoder.h5")
44
45     """
46
47     Optimize the encoder and the generator network
48     """
49     if TRAIN_GAN_WITH_FR:
50
51         # Load the encoder network
52         encoder = build_encoder()
53         encoder.load_weights("encoder.h5")
54
55         # Load the generator network
56         generator.load_weights("generator.h5")
57
58         image_resizer = build_image_resizer()
59         image_resizer.compile(loss=['binary_crossentropy'],
60                               optimizer='adam')
61
62         # Face recognition model
63         fr_model = build_fr_model(input_shape=fr_image_shape)
64         fr_model.compile(loss=['binary_crossentropy'], optimizer=
65                           "adam")
66
67         # Make the face recognition network as non-trainable
68         fr_model.trainable = False
69
70         # Input layers
71         input_image = Input(shape=(64, 64, 3))
72         input_label = Input(shape=(6,))
73
74         # Use the encoder and the generator network
75         latent0 = encoder(input_image)
76         gen_images = generator([latent0, input_label])
77
78         # Resize images to the desired shape
79         resized_images = Lambda(lambda x: K.resize_images(
80                               gen_images, height_factor=3, width_factor=3,
81                               data_format='channels_last'))(gen_images)
82         embeddings = fr_model(resized_images)
83
84         # Create a Keras model and specify the inputs and outputs
85         # for the network
86         fr_adversarial_model = Model(inputs=[input_image,
87                                               input_label], outputs=[embeddings])
88
89         # Compile the model
90         fr_adversarial_model.compile(loss=euclidean_distance_loss,
91                                       optimizer=adversarial_optimizer)
92
93         for epoch in range(epochs):
94             print("Epoch:", epoch)
95
96             reconstruction_losses = []

```

```

91         number_of_batches = int(len(loaded_images) /
92         batch_size)
93         print("Number of batches:", number_of_batches)
94         for index in range(number_of_batches):
95             print("Batch:", index + 1)
96
97             images_batch = loaded_images[index * batch_size:(
98             index + 1) * batch_size]
99             images_batch = images_batch / 127.5 - 1.0
100             images_batch = images_batch.astype(np.float32)
101
102             y_batch = y[index * batch_size:(index + 1) *
103             batch_size]
104
105             images_batch_resized = image_resizer.
106             predict_on_batch(images_batch)
107
108             real_embeddings = fr_model.predict_on_batch(
109             images_batch_resized)
110
111             reconstruction_loss = fr_adversarial_model.
112             train_on_batch([images_batch, y_batch], real_embeddings)
113
114             print("Reconstruction loss:", reconstruction_loss
115             )
116
117             reconstruction_losses.append(reconstruction_loss)
118
119             # Write the reconstruction loss to Tensorboard
120             write_log(tensorboard, "reconstruction_loss", np.mean
121             (reconstruction_losses), epoch)
122
123             """
124             Generate images
125             """
126             if epoch % 10 == 0:
127                 images_batch = loaded_images[0:batch_size]
128                 images_batch = images_batch / 127.5 - 1.0
129                 images_batch = images_batch.astype(np.float32)
130
131                 y_batch = y[0:batch_size]
132                 z_noise = np.random.normal(0, 1, size=(batch_size
133                 , z_shape))
134
135                 gen_images = generator.predict_on_batch([z_noise,
136                 y_batch])
137
138                 for i, img in enumerate(gen_images[:5]):
139                     save_rgb_img(img, path="results/img_opt-{}-
140                     {}.png".format(epoch, i))
141
142             # Save improved weights for both of the networks
143             generator.save_weights("generator_optimized.h5")
144             encoder.save_weights("encoder_optimized.h5")

```

Proses kerja nya dengan membuat model .h5, lalu load data dengan menghasilkan result.

9.2.3 Penanganan Error

1. ValueError

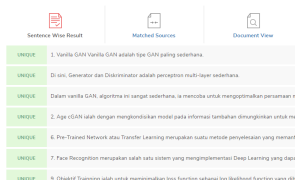


Gambar 9.22 FileNotFoundError

2. Cara Penanganan Error

- `FileNotFoundError`
Error tersebut karena disebabkan gagal load dataset karena salah penamaan type file.

9.2.4 Bukti Tidak Plagiat



Gambar 9.23 Bukti Tidak Melakukan Plagiat Chapter 9

BAB 10

CHAPTER 10

BAB 11

CHAPTER 11

BAB 12

CHAPTER 12

BAB 13

CHAPTER 13

BAB 14

CHAPTER 14

DAFTAR PUSTAKA

- [1] R. Awangga, "Sampeu: Servicing web map tile service over web map service to increase computation performance," in *IOP Conference Series: Earth and Environmental Science*, vol. 145, no. 1. IOP Publishing, 2018, p. 012057.

Index

disruptif, **xxiii**
modern, **xxiii**