

**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 Tunnisia

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. Sariasisih 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**

---



## DAFTAR ISI

---



# **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

$B$  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[?].

$$ABC\mathcal{DEF}\alpha\beta\Gamma\Delta \sum_{def}^{abc} \quad (I.1)$$



# **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 1174083 - Bakti Qilan Mufid

Chapter 8 - Perkenalan Generative Adversarial Network

#### 8.1.1 Teori

##### 8.1.1.1 *Jelaskan dengan ilustrasi gambar sendiri apa itu generator dengan perumpamaan anda sebagai mahasiswa sebagai generatornya.*

Menurut KBBI, Generator adalah pembangkit. Arti lainnya dari generator adalah alat pembangkit tenaga listrik. atau jika pada kecerdasan buatan bisa diartikan sebagai pemubuat. misalnya pembuat gambar, pembuat musik, dll. jika perumpamaan mahasiswa sebagai generator maka dapat digambarkan seperti berikut:



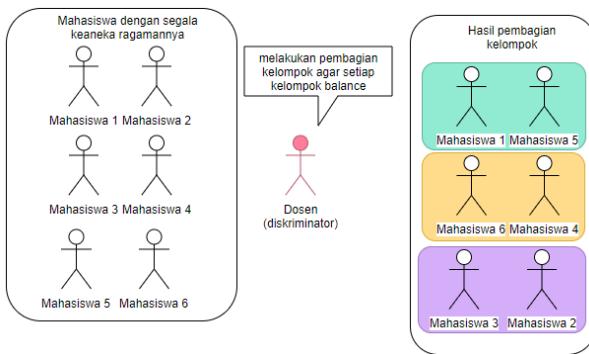
**Gambar 8.1** gambaran penjelasan no. 1

#### **8.1.1.2 Jelaskan dengan ilustrasi gambar sendiri apa itu diskriminator dengan perumpamaan dosen anda sebagai diskriminatorya.**

arti kata diskriminator adalah rangkaian dalam berbagai alat. berasal dari kata diskriminasi yang merujuk pada pelayanan yang tidak adil terhadap individu tertentu, di mana layanan ini dibuat berdasarkan karakteristik yang diwakili oleh individu tersebut. dalam machine learning, Discrimination itu bisa ada dua, yaitu

1. Disparate Treatment(perawatan berbeda) atau membedakan/ mengklasifikasikan sesuatu atau seseorang
2. Disparate Impact (dampak berbeda) ialah melihat konsekuensi klasifikasi/ pengambilan keputusan pada kelompok tertentu.

berikut ilustrasi jika dosen sebagai diskriminator:



**Gambar 8.2** gambaran penjelasan no. 2

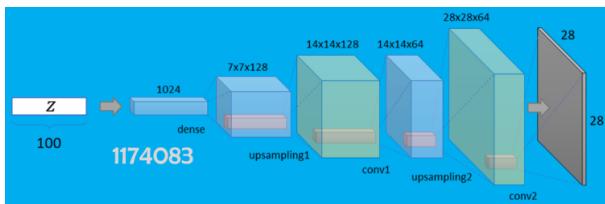
#### **8.1.1.3 Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur generator dibuat**

arsitektur generator didalam GAN terdiri dari lima lapisan, yaitu: satu input layer, tiga dense layer dan satu output layer. bisa dilihat arsitektur geerator pada GAN-Project seperti gambar berikut:

	13970549349376	
dense_1:Dense	Input:	(None, None, 100)
	Output:	(None, None, 500)
dense_2:Dense	Input:	(None, None, 500)
	Output:	(None, None, 500)
dense_3:Dense	Input:	(None, None, 500)
	Output:	(None, None, 784)
reshape_1:Reshape	Input:	(None, None, 784)
	Output:	(None, 28, 28)

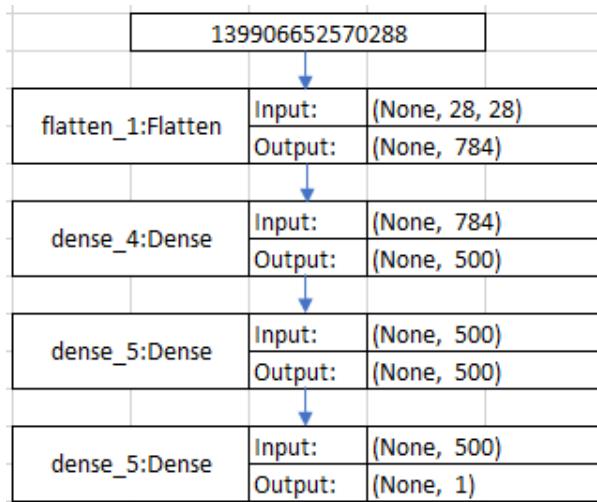
**Gambar 8.3** gambar arsitektur generator pada GAN-Project

untuk ilustrasinya seperti berikut:



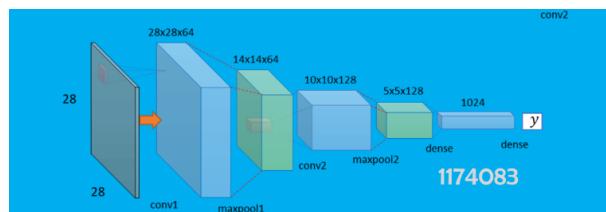
**Gambar 8.4** gambaran penjelasan no. 3

**8.1.1.4 Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur diskriminatore dibuat** arsitektur diskriminator didalam GAN terdiri dari lima lapisan, yaitu: satu input layer, tiga dense layer dan satu output layer. bisa dilihat arsitektur diskriminator pada GAN-Project seperti gambar berikut:



**Gambar 8.5** gambar arsitektur generator pada GAN-Project

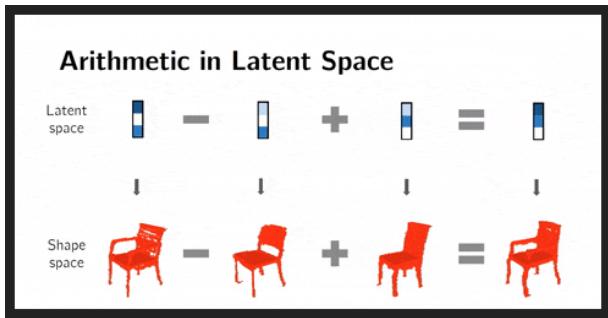
untuk ilustrasinya seperti berikut:



**Gambar 8.6** gambaran penjelasan no. 4

#### 8.1.1.5 Jelaskan dengan ilustrasi gambar apa itu latent space.

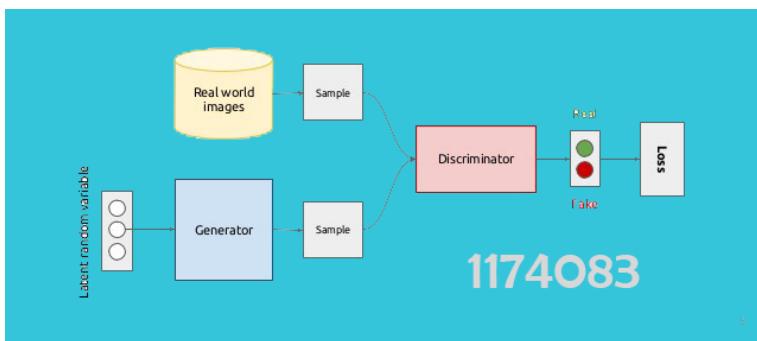
latent space adalah space atau spot yang tersembunyi, sehingga akan mengambil titik data yang dekat dan yang mirip. ilustrasinya seperti pada gambar berikut:



**Gambar 8.7** gambaran penjelasan no. 5

#### 8.1.1.6 Jelaskan dengan ilustrasi gambar apa itu adversarial play

Dalam GAN adversarial play ialah persaingan antara generator dan diskriminator. bisa dilihat pada ilustrasi gambar berikut:



**Gambar 8.8** gambaran penjelasan no. 6

#### 8.1.1.7 Jelaskan dengan ilustrasi gambar apa itu Nash equilibrium

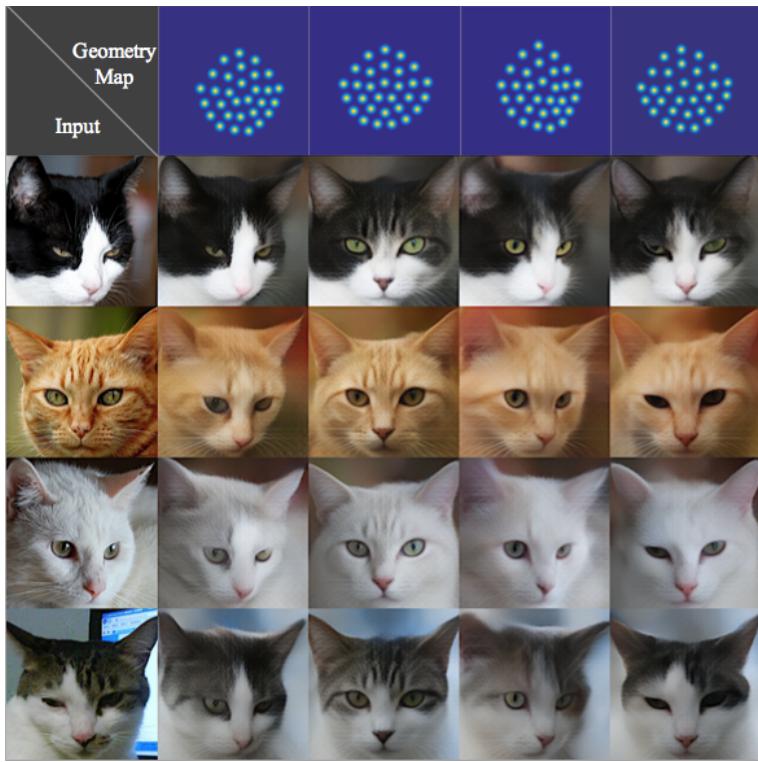
nash equilibrium adalah strategi yang dipilih oleh masing-masing pemain, dengan strategi pemain lain tertentu. Strategi ini merupakan strategi terbaik dari masing-masing pemain dengan mempertimbangkan strategi tertentu yang diambil oleh pemain lainnya, atau yang disebut dengan Best Respons.

		TOM	
		A	B
SAM	A	1,1	1,-1
	B	-1,1	0,0

**Gambar 8.9** gambaran penjelasan no. 7

#### 8.1.1.8 **Sebutkan dan jelaskan contoh-contoh implementasi dari GAN**

Contoh implementasi dari GAN salah satunya adalah meng-generate wajah kucing, seperti pada gambar berikut:



**Gambar 8.10** gambaran penjelasan no. 8

**8.1.1.9 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 gambar arsitektur nya ada pada gambar no. 3 sedangkan untuk kode programnya seperti berikut:

```

1 def generator_model():
2     model = Sequential([
3         Dense(1024, input_dim=100, activation='tanh'),
4         Dense(128*7*7),
5         Reshape((7, 7, 128)),
6         UpSampling2D(size=(2, 2)),
7         Conv2D(64, (5, 5), padding='same', activation='tanh'),
8         UpSampling2D(size=(2, 2)),
9         Conv2D(1, (5, 5), padding='same', activation='tanh')
10    ])
11    return model
12
13 generator_model().summary()

```

**Listing 8.1** Kode program arsitektur generator

### **8.1.1.10 Berikan contoh dengan ilustrasi dari arsitektur dikriminator dengan sebuah input layer, 3 buah hidden layer, dan satu output layer.**

untuk gambar arsitektur nya ada pada gambar no. 4 sedangkan untuk kode programnya seperti berikut:

```

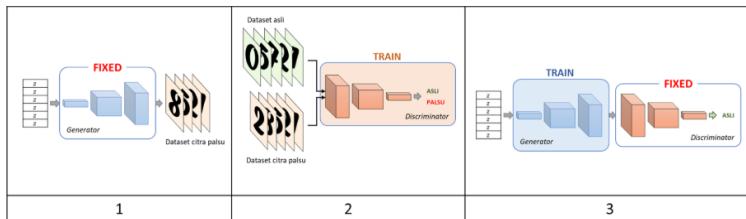
1 def discriminator_model():
2     model = Sequential([
3         Conv2D(64, (5, 5), padding='same', input_shape=(28, 28, 1),
4         activation='tanh'),
5         MaxPooling2D(pool_size=(2, 2)),
6         Conv2D(128, (5, 5), activation='tanh'),
7         MaxPooling2D(pool_size=(2, 2)),
8         Flatten(),
9         Dense(1024, activation='tanh'),
10        Dense(1, activation='sigmoid')
11    ])
12    return model
13
14 discriminator_model().summary()

```

**Listing 8.2** Kode program arsitektur diskriminator

### **8.1.1.11 Jelaskan bagaimana kaitan output dan input antara generator dan diskriminatot tersebut. Jelaskan kenapa inputan dan outputan seperti itu.**

kaitan antara output pada generator dan inputan pada diskriminatot adalah suatu keterhubungan yang akan menghasilkan data yang benar, karena output dari generator merupakan inoutan pada diskriminatot, berikut gambarannya:



**Gambar 8.11** gambaran penjelasan no. 9

### **8.1.1.12 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.**

diantara keduanya yang paling terkenal adalah KL divergence, karena KL divergence memiliki beberapa sifat/rumus yang bagus .salah satunya adalah  $KL[q; p]$  jenis daerah yang di mana  $q(x)$  memiliki massa non-null dan  $p(x)$  memiliki massa null. Ini mungkin terlihat seperti bug, tetapi sebenarnya ini merupakan fitur dalam situasi tertentu. sedangkan JS tidak memiliki

### **8.1.1.13 Jelaskan apa itu fungsi objektif yang berfungsi untuk mengukur kesamaan antara gambar yang dibuat dengan yang asli.**

fungsi objektif merupakan fungsi yang akan mengambil parameter data dan model sebagai argumen, dan dapat dievaluasi untuk mengembalikan angka sehingga mampu membedakan gambar yang asli dan yang di buat/fake.

#### ***8.1.1.14 Jelaskan apa itu scoring algoritma selain mean square error atau cross entropy seperti The Inception Score dan The Frechet Inception distance.***

Inception Score digunakan untuk mengukur seberapa realistik output dari GAN, dimana ada dua parameter, yaitu : gambarnya punya variasi dan setiap gambar jelas terlihat seperti sesuatu. Frechet Inception Distance adalah ukuran kesamaan antara dua dataset gambar. Itu terbukti berkorelasi baik dengan penilaian manusia terhadap kualitas visual dan paling sering digunakan untuk mengevaluasi kualitas sampel Generative Adversarial Networks. FID dihitung dengan menghitung jarak Fréchet antara dua Gaussians dipasang ke representasi fitur dari jaringan Inception.

#### ***8.1.1.15 Jelaskan kelebihan dan kekurangan GAN***

- Kelebihan

1. GAN Menghasilkan data baru yang bisa hampir mirip dengan data asli. Karena hasil pelatihannya, GAN dapat menghasilkan data gambar, teks, audio, dan video yang dapat dibilang hampir mirip dengan yang aslinya. Berkat hal tersebut, GAN dapat digunakan dalam sistem marketing, e-commerce, games, iklan, dan industri lainnya
2. GAN mempelajari representasi data secara internal sehingga beberapa masalah pada machine learning dapat diatasi dengan mudah
3. Discriminator yang sudah dilatih dapat menjadi sebuah classifier atau pendekripsi jika data sudah sesuai. Karena Discriminator yang akan menjadi tidak efisien berkat seringnya dilatih
4. GAN dapat dilatih menggunakan data yang belum dilabeled

- Kekurangan

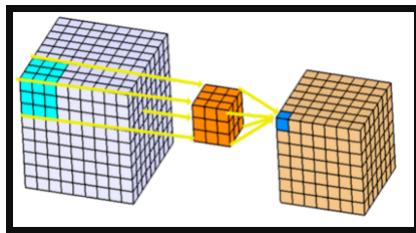
1. Data saat diproses oleh metode gan tidak konvergensi
2. Jenis sampel yang dihasilkan oleh generator terbatas karena modenya terbatas
3. Ketidak seimbangnya antara generator dan discriminator dapat menyebabkan overfitting atau terlalu dekat dengan hasil sampel
4. Sangat sensitif dengan data yang sudah diinisiasi sebelumnya

#### **8.1.2 Praktek**

##### ***8.1.2.1 Jelaskan apa itu 3D convolutions***

3D convolutions merupakan operasi konvolusi 3D yang menerapkan filter 3D ke data

input dengan tiga arah, yaitu x,y, dan z. fitur ini menciptakan sebuah daftar peta yang ditumpuk. agar lebih mudah saya menggunakan sebuah ilustrasi seperti berikut:



**Gambar 8.12** gambaran 3D Convolutions

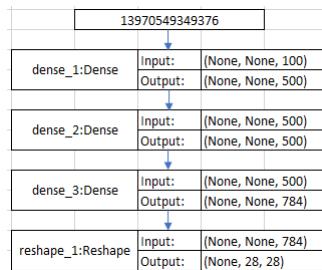
#### 8.1.2.2 Jelaskan dengan kode program arsitektur dari generator networknya, beserta penjelasan input dan output dari generator network.

```

1 def build_generator():#nama kelasnya
2     """
3         Create a Generator Model with hyperparameters values defined as
4         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')(a)
28        a = BatchNormalization()(a, training=True)
29        a = Activation(activation=gen_activations[i + 1])(a)
30
31    gen_model = Model(inputs=[input_layer], outputs=[a])
32    return gen_model

```

kode diatas merupakan generator network yang terdiri dari lima layer, satu input layer, tiga hidden layer, dan satu output layer.Dan prosesnya dapat digambarkan seperti berikut:



**Gambar 8.13** gambaran proses generator network

penjelasan

- Input layer mengambil 100-dimensi sampel dari distribusi Gaussian(normal) dan meneruskan tensor ke. hidden layer pertama tanpa ada modifikasi
- ketiga hidden layer adalah dense layer dengan unit masing-masing 500, 500, dan 784. dimana dense layer pertama mengkonversi bentuk tensor (batch\_size, 100) ke bentuk tensor(batch\_size, 500). sedangkan pada layer dense kedua menghasilkan bentuk tensor (batch\_size, 500), dan layer dense ketiga menghasilkan (batch\_size, 784)
- Output layer, tensor akan dibentuk kembali dari bentuk tensor (batch\_size, 784) menjadi (batch\_size, 28, 28) artinya hasil dari generator ini akan menghasilkan banyak gambar, dimana setiap gambarnya memiliki ukuran (28, 28).

#### **8.1.2.3 Jelaskan dengan kode program arsitektur dari diskriminator network, beserta penjelasan input dan outputnya.**

```

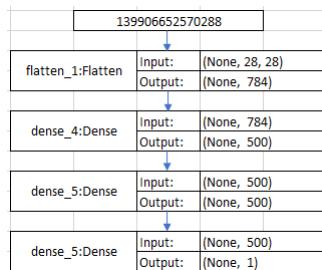
1 def build_discriminator():
2     """
3         Create a Discriminator Model using hyperparameters values defined
4             as follows
5         """
6     dis_input_shape = (64, 64, 64, 1)
7     dis_filters = [64, 128, 256, 512, 1]
8     dis_kernel_sizes = [4, 4, 4, 4, 4]
9     dis_strides = [2, 2, 2, 2, 1]
10    dis_paddings = ['same', 'same', 'same', 'same', 'valid']
11    dis_alphas = [0.2, 0.2, 0.2, 0.2, 0.2]
12    dis_activations = ['leaky_relu', 'leaky_relu', 'leaky_relu',
13                      'leaky_relu', 'sigmoid']
14    dis_convolutional_blocks = 5
15
16    dis_input_layer = Input(shape=dis_input_shape)
17
18    # The first 3D Convolutional block
19    a = Conv3D(filters=dis_filters[0],
20               kernel_size=dis_kernel_sizes[0],
21               strides=dis_strides[0],
22               padding=dis_paddings[0],
23               activation=dis_activations[0])
24    a = a(dis_input_layer)
25
26    # The remaining 4 blocks
27    for i in range(1, dis_convolutional_blocks):
28        a = Conv3D(filters=dis_filters[i],
29                   kernel_size=dis_kernel_sizes[i],
30                   strides=dis_strides[i],
31                   padding=dis_paddings[i],
32                   activation=dis_activations[i])(a)
33
34    dis_output_layer = a
  
```

```

20         strides=dis_strides[0],
21         padding=dis_paddings[0])(dis_input_layer)
22 # a = BatchNormalization()(a, training=True)
23 a = LeakyReLU(dis_alphas[0])(a)
24
25 # Next 4 3D Convolutional Blocks
26 for i in range(dis_convolutional_blocks - 1):
27     a = Conv3D(filters=dis_filters[i + 1],
28                 kernel_size=dis_kernel_sizes[i + 1],
29                 strides=dis_strides[i + 1],
30                 padding=dis_paddings[i + 1])(a)
31     a = BatchNormalization()(a, training=True)
32     if dis_activations[i + 1] == 'leaky_relu':
33         a = LeakyReLU(dis_alphas[i + 1])(a)
34     elif dis_activations[i + 1] == 'sigmoid':
35         a = Activation(activation='sigmoid')(a)
36
37 dis_model = Model(inputs=[dis_input_layer], outputs=[a])
38 return dis_model

```

kode diatas merupakan diskriminatior network, dimana prosesnya dapat digambarkan seperti berikut:



**Gambar 8.14** gambaran proses diskriminatior network

penjelasan

- awalnya diskriminatior menerima input dengan bentuk 28x28.
- Input layer mengambil tensor input dan meneruskannya ke hidden layer pertama tanpa modifikasi apapun.
- lalu flattens layer akan meratakan tensor menjadi 784-dimensi vektor, yang akan diteruskan ke hidden layer (dense layer) pertama. hidden layer pertama dan kedua aka memodifikasinya menjadi vektor 500-dimensi.
- Output layer mesuk kedalam dense layer, dengan satu unit neuron dan sigmoid sebagai fungsi aktivasi. sigmoid akan menghasilkan nilai tunggal, 0 atau 1. Nilai 0 akan menunjukan bahwa gambar yang diberikan palsu. sebaliknya jika nilai 1 maka gambar asli.

#### **8.1.2.4 Jelaskan proses training 3D-GANs**

Proses training 3D-GANs dilakukan seperti langkah-langkah berikut:

- Terdapat sebuah vektor noise dengan dimensi 200 dari distribusi Gaussian(normal).
- Meng-generate gambar palsu menggunakan model generator.
- Melatih jaringan generator dengan gambar yang asli(sampel dari data yang real) dan dengan gambar palsu yang dihasilkan oleh generator.
- Gunakan adversarial model untuk melatih generator model, jangan melatih diskriminatore model.
- Ulangi langkah ini dengan jumlah epoch tertentu.

#### **8.1.2.5 Jelaskan bagaimana melakukan settingan awal chapter 02 untuk memenuhi semua kebutuhan sebelum melanjutkan ke tahapan persiapan data.**

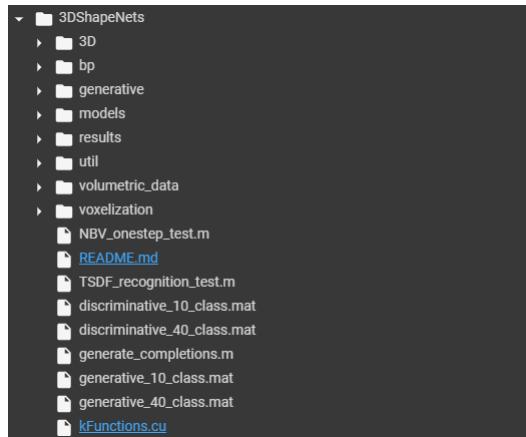
Sebelum melakukan tahapan persiapan data, kita harus melakukan beberapa hal seperti berikut:

- membaca buku panduan atau keterangan dari Generative-Adversarial-Network project ini. bisa didownload di portal kampus keren atau di github <https://github.com/PacktPublishing/Generative-Adversarial-Networks-Projects>
- persiapkan laptop/pc dengan spek yang cukup bagus(tidak kentang). jika tidak ada, bisa menggunakan google colab(cara penggunaanya akan dijelaskan di video).
- usahakan versi yang terinstall sama dengan versi yang ada pada requirement.txt agar terhindar dari error.

#### **8.1.2.6 Jelaskan tentang dataset yang digunakan, dari mulai tempat unduh, cara membuka dan melihat data. sampai deskripsi dari isi dataset dengan detail penjelasan setiap folder/file yang membuat orang awam paham.**

Penjelasan dataset

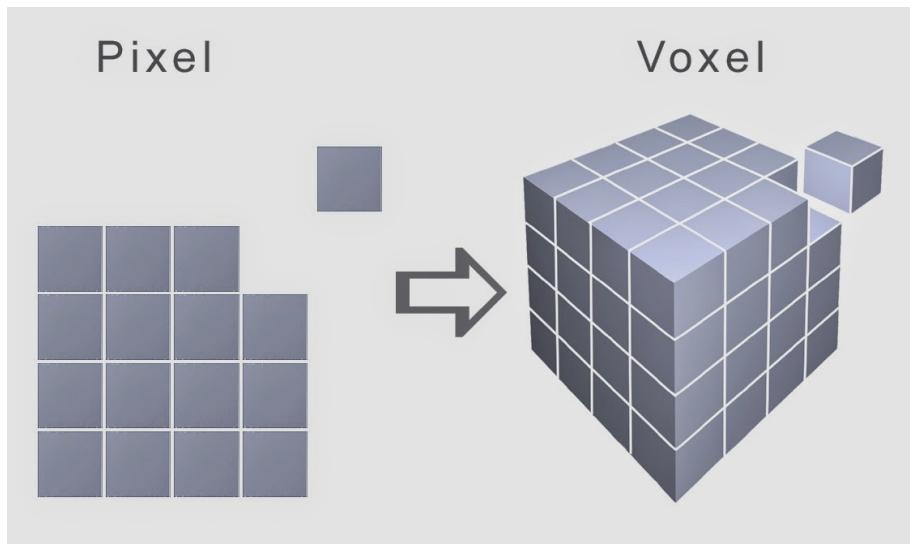
- dataset bisa didownload pada halaman: <http://3dshapenets.cs.princeton.edu/3DShapeNets>
- Untuk membuka data, setelah didownload cukup di eksrak saja menggunakan winrar/ 7zip
- Untuk mengetahui lebih jelasnya tentang dataset ini bisa dibuka saja file README.md, disana akan dijelaskan semuanya.



**Gambar 8.15** Isi dari dataset.zip

#### **8.1.2.7 Jelaskan apa itu voxel dengan ilustrasi dan bahasa paling awam**

Secara singkat, voxel dapat diartikan sebagai 3D pixel atau pixel dalam bentuk 3D. seperti pada ilustrasi berikut:



**Gambar 8.16** Pixel VS Voxel

#### **8.1.2.8 Visualisasikan dataset tersebut dalam tampilan visual plot, jelaskan cara melakukan visualisasinya**

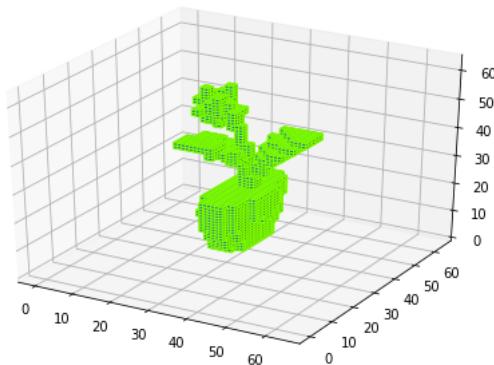
```
1 import scipy.io as io
```

```

2 import matplotlib.pyplot as plt
3 import scipy.ndimage as nd
4 import numpy as np
5 from mpl_toolkits.mplot3d import Axes3D
6
7 voxels = io.loadmat("/content/3DShapeNets/volumetric_data/flower_pot
8 /30/test/flower_pot_000000010_1.mat")['instance']
9 voxels = np.pad(voxels, (1, 1), 'constant', constant_values=(0, 0))
9 voxels = nd.zoom(voxels, (2, 2, 2), mode='constant', order=0)
10
11 fig = plt.figure()
12 ax = Axes3D(fig)
13 ax.voxels(voxels, edgecolor="chartreuse")
14
15 plt.show()
16 plt.savefig('flower_pot')

```

Kode diatas berfungsi untuk melakukan visualisasi dataset dalam tampilan plot. dengan tahapan import library, load data file .mat, dan lakukan read memakai matplotlib. dan hasilnya seperti berikut



**Gambar 8.17** Visualisasi dataset(contoh)

#### 8.1.2.9 Buka file run.py jelaskan perbaris kode pada fungsi untuk membuat generator yaitu build generator

```

1 z_size = 200
2 gen_filters = [512, 256, 128, 64, 1]
3 gen_kernel_sizes = [4, 4, 4, 4, 4]
4 gen_strides = [1, 2, 2, 2, 2]
5 gen_input_shape = (1, 1, 1, z_size)
6 gen_activations = ['relu', 'relu', 'relu', 'relu', 'sigmoid']
7 gen_convolutional_blocks = 5

```

potongan kode diatas berfungsi sebagai penentuan nilai untuk hyperparameters yang berbeda.

```
1 input_layer = Input(shape=gen_input_shape)
```

potongan kode diatas berfungsi untuk membuat input layer

```
1 # First 3D transpose convolution(or 3D deconvolution) block
2 a = Deconv3D(filters=gen_filters[0],
3               kernel_size=gen_kernel_sizes[0],
4               strides=gen_strides[0])(input_layer)
5 a = BatchNormalization()(a, training=True)
6 a = Activation(activation='relu')(a)
```

potongan kode diatas berfungsi untuk menambahkan 3D transpose pertama.

```
1 # Next 4 3D transpose convolution(or 3D deconvolution) blocks
2 for i in range(gen_convolutional_blocks - 1):
3     a = Deconv3D(filters=gen_filters[i + 1],
4                   kernel_size=gen_kernel_sizes[i + 1],
5                   strides=gen_strides[i + 1], padding='same')(a)
6     a = BatchNormalization()(a, training=True)
7     a = Activation(activation=gen_activations[i + 1])(a)
```

potongan kode diatas berfungsi untuk menambahkan empat 3D transpose lainnya.

```
1 gen_model = Model(inputs=[input_layer], outputs=[a])
```

potongan kode diatas berfungsi untuk membuat model keras dan menentukan input dan output untuk network generator.

#### **8.1.2.10 jelaskan juga fungsi untuk membangun diskriminator pada fungsi build discriminator.**

```
1 dis_input_shape = (64, 64, 64, 1)
2 dis_filters = [64, 128, 256, 512, 1]
3 dis_kernel_sizes = [4, 4, 4, 4, 4]
4 dis_strides = [2, 2, 2, 2, 1]
5 dis_paddings = ['same', 'same', 'same', 'same', 'valid']
6 dis_alphas = [0.2, 0.2, 0.2, 0.2, 0.2]
7 dis_activations = ['leaky_relu', 'leaky_relu', 'leaky_relu',
```

potongan kode diatas berfungsi sebagai penentuan nilai untuk hyperparameters yang berbeda.

```
1 dis_input_layer = Input(shape=dis_input_shape)
```

potongan kode diatas berfungsi untuk membuat input layer, berupa gambar 3D dengan dimensi 64x64x64x1

```
1 # The first 3D Convolutional block
2 a = Conv3D(filters=dis_filters[0],
3             kernel_size=dis_kernel_sizes[0],
4             strides=dis_strides[0],
5             padding=dis_paddings[0])(dis_input_layer)
6 # a = BatchNormalization()(a, training=True)
7 a = LeakyReLU(dis_alphas[0])(a)
```

potongan kode diatas berfungsi untuk menambahkan 3D transpose pertama.

```

1 # Next 4 3D Convolutional Blocks
2 for i in range(dis_convolutional_blocks - 1):
3     a = Conv3D(filters=dis_filters[i + 1],
4                 kernel_size=dis_kernel_sizes[i + 1],
5                 strides=dis_strides[i + 1],
6                 padding=dis_paddings[i + 1])(a)
7     a = BatchNormalization()(a, training=True)
8     if dis_activations[i + 1] == 'leaky_relu':
9         a = LeakyReLU(dis_alphas[i + 1])(a)
10    elif dis_activations[i + 1] == 'sigmoid':
11        a = Activation(activation='sigmoid')(a)

```

potongan kode diatas berfungsi untuk menambahkan empat 3D transpose lainnya.

```
1 dis_model = Model(inputs=[dis_input_layer], outputs=[a])
```

potongan kode diatas berfungsi untuk membuat model keras dan menentukan input dan output untuk network generator.

#### **8.1.2.11 jelaskan apa maksud dari kode program name == 'main'**

```
1 if __name__ == '__main__':
```

**Listing 8.3** Kode program run.py

maksudnya kode tersebut memiliki fungsi jika interpreter python menjalankan kode tersebut, maka akan menetapkan variable name untuk memiliki nilai main. jika file ini di import dari modul lain, maka name akan ditetapkan ke nama modul tersebut.

#### **8.1.2.12 jelaskan secara detil perbaris dan per parameter apa arti dari kode program :**

Penjelasan kode:

```

1 # In[ soal_12]
2     object_name = "airplane" #membuat variabel
3     data_dir = "data/3DShapeNets/volumetric_data/" \
4                 "{}/30/train/*.mat".format(object_name) #menunjukkan
5     direktori dari variable sebelumnya
6     gen_learning_rate = 0.0025 #membuat rate pada generator
7     dis_learning_rate = 1e-5 #membuat rate pada diskriminatot
8     beta = 0.5 #
9     batch_size = 1 #menentukan ukuran kumpulan gambar
10    z_size = 200 #
11    epochs = 1 #menetukkan masa periode. setiap epoch memiliki semua
12      batch
13    MODE = "train" #membuat traon mode

```

**Listing 8.4** Kode program run.py

#### **8.1.2.13 Jelaskan secara detil dari kode program pembuatan dan kompilasi arsitektur berikut :**

```

1 gen_optimizer = Adam(lr=gen_learning_rate, beta_1=beta)
2 dis_optimizer = Adam(lr=dis_learning_rate, beta_1=beta)
3
4 discriminator = build_discriminator()
5 discriminator.compile(loss='binary_crossentropy', optimizer=
6 dis_optimizer)
7
8 generator = build_generator()
9 generator.compile(loss='binary_crossentropy', optimizer=
10 gen_optimizer)

```

**Listing 8.5** Kode program run.py

penjelasanya ialah membuat model dari generator model dan diskriminatormodel,

#### **8.1.2.14 Jelaskan secara detil kode program untuk membuat dan melakukan kompilasi model adversarial berikut:**

```

1 # In[ soal 14]
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=[validity])
8 adversarial_model.compile(loss='binary_crossentropy', optimizer=
9 gen_optimizer)

```

**Listing 8.6** Kode program run.py

penjelasanya ialah membuat model diskriminatormodel tidak di training dan membuat model adversarial.

#### **8.1.2.15 Jelaskan Ekstrak dan load data kursi dengan menggunakan fungsi getVoxelsFormat dan get3DImages yang digunakan pada kode program berikut :**

```

1 # In[ soal 15]
2 print("Loading data...")
3 volumes = get3DImages(data_dir=data_dir)
4 volumes = volumes [..., np.newaxis].astype(np.float)
5 print("Data loaded...")

```

**Listing 8.7** Kode program run.py

penjelasanya ialah melakukan print loading data, membuat variabel volumes untuk mendapatkan data gamabr 3D.

#### **8.1.2.16 Jelaskan maksud dari kode program instansiasi TensorBoard yang menambahkan generator dan diskriminatormodel pada program berikut:**

```

1 # In[ soal_16]
2     tensorboard = TensorBoard(log_dir="logs/{}".format(time.time()))
3     tensorboard.set_model(generator)
4     tensorboard.set_model(discriminator)

```

**Listing 8.8** Kode program run.py

penjelasanya ialah membuat variabel tensorboard untuk melakukan pencatatan log, dan men-set model generator dan model diskriminatot

#### **8.1.2.17 Jelaskan apa fungsi dari np reshape ones zeros pada kode program berikut dengan parameternya:**

Penjelasan kode:

```

1 # In[ soal_17]
2     labels_real = np.reshape(np.ones((batch_size,)), (-1, 1, 1, 1, 1))
3     labels_fake = np.reshape(np.zeros((batch_size,)), (-1, 1, 1, 1, 1))

```

**Listing 8.9** Kode program run.py

membuat label\_real untuk menetapkan bahwa jika output 1 itu menunjukan gambar real, dan label\_fake untuk menunjukan bahwa output 0 untuk data fake/palsu.

#### **8.1.2.18 Jelaskan kenapa harus ada perulangan dalam meraih epoch. Dan jelaskan apa itu epoch terkait kode program berikut:**

```

1 # In[ soal_18]
2     if MODE == 'train':
3         for epoch in range(epochs):
4             print("Epoch:", epoch)
5
6             gen_losses = []
7             dis_losses = []

```

**Listing 8.10** Kode program run.py

melakukan perulangan jika MODE = train maka akan diulangi terus menerus sampai epoch(periode) terpenuhi dengan disertai keterangan generator dan diskriminatot losses.

#### **8.1.2.19 Jelaskan apa itu batches dan kaitannya dengan kode program berikut, dan kenapa berada di dalam epoch:**

```

1 # In[ soal_19]
2     number_of_batches = int(volumes.shape[0] / batch_size)
3     print("Number of batches:", number_of_batches)
4     for index in range(number_of_batches):
5         print("Batch:", index + 1)

```

**Listing 8.11** Kode program run.py

menghitung jumlah batch yang telah ditrain

**8.1.2.20 Berikut adalah kode program pengambilan gambar dan noise. Jelaskan apa fungsi `np.random.normal` serta `astype`, serta jelaskan apa arti parameter titik dua dan jelaskan isi dari `z sample` dan `volumes batch`:**

```

1 # In[ soal 20]
2             z_sample = np.random.normal(0, 0.33, size=[batch_size
3 , 1, 1, 1, z_size]).astype(np.float32)
4             volumes_batch = volumes[index * batch_size:(index +
5 1) * batch_size, :, :, :]

```

**Listing 8.12** Kode program run.py

untuk membersihkan gambar dari noise dan menyesuaikan shape.

**8.1.2.21 Berikut adalah kode program generator gambar palsu. Jelaskan apa fungsi `generator.predict_on_batch`, serta jelaskan apa arti parameter `z sample`:**

```

1 # In[ soal 21]
2             # Next, generate volumes using the generate network
3             gen_volumes = generator.predict_on_batch(z_sample)

```

**Listing 8.13** Kode program run.py

untuk meng-generate volume menggunakan model jaringan generator.

**8.1.2.22 Berikut adalah kode program training diskriminator dengan gambar palsu dari generator dan gambar asli. Jelaskan apa maksudnya harus dilakukan training diskriminator secara demikian dan jelaskan apa isi `loss fake` dan `loss real` serta `d loss` dan fungsi `train on batch`.**

```

1 # In[ soal 22]
2             discriminator.trainable = True
3             if index % 2 == 0:
4                 loss_real = discriminator.train_on_batch(
5                     volumes_batch, labels_real)
6                 loss_fake = discriminator.train_on_batch(
7                     gen_volumes, labels_fake)
8
9                 d_loss = 0.5 * np.add(loss_real, loss_fake)
10                print("d_loss:{}".format(d_loss))
11
12            else:
13                d_loss = 0.0

```

**Listing 8.14** Kode program run.py

untuk membuat diskriminator bisa meload gambar fake dan real dari model generator. dengan disertai generatot dan diskriminatot loss, agar terlihat seberapa baik kualitas yang dihasilkan.

**8.1.2.23 Berikut adalah kode program training model adversarial yang terdapat generator dan diskriminator. Jelaskan apa bagaimana proses terbentuknya**

**parameter z dan g loss:**

```

1 # In[ soal_23]
2         discriminator.trainable = False
3         """
4             Train the generator network
5             """
6         z = np.random.normal(0, 0.33, size=[batch_size, 1, 1,
7             1, z_size]).astype(np.float32)
8         g_loss = adversarial_model.train_on_batch(z,
9             labels_real)
10        print("g_loss:{}".format(g_loss))
11
12        gen_losses.append(g_loss)
13        dis_losses.append(d_loss)

```

**Listing 8.15** Kode program run.py

untuk mentrain model generator dan variabel g\_loss untuk melakukan perbandingan antara label gambar yang asli.

**8.1.2.24 Berikut adalah kode program generate dan menyimpan gambar 3D setelah beberapa saat setiap epoch. Jelaskan mengapa ada perulangan dengan parameter tersebut, serta jelaskan arti setiap variabel beserta perlihatkan isinya dan artikan isinya :**

```

1 # In[ soal_24]
2         # Every 10th mini-batch, generate volumes and save
3         them
4         if index % 10 == 0:
5             z_sample2 = np.random.normal(0, 0.33, size=[batch_size, 1, 1, 1, z_size]).astype(np.float32)
6             generated_volumes = generator.predict(z_sample2,
7                 verbose=3)
8             for i, generated_volume in enumerate(
9                 generated_volumes[:5]):
10                voxels = np.squeeze(generated_volume)
11                voxels[voxels < 0.5] = 0.
12                voxels[voxels >= 0.5] = 1.
13                saveFromVoxels(voxels, "results/img_{:}-{:}-{:}".
14 .format(epoch, index, i))

```

**Listing 8.16** Kode program run.py

untuk melakukan perulangan dimana setiap terdapat 10 mini-batch akan meng-generate volumenya dan akan menyimpannya.

**8.1.2.25 Berikut adalah kode program menyimpan average losses setiap epoch. Jelaskan apa itu tensorboard dan setiap parameter yang digunakan pada kode program ini :**

```

1 # In[ soal_25]

```

```

2         # Write losses to Tensorboard
3         write_log(tensorboard , 'g_loss' , np.mean(gen_losses) ,
4         epoch)           write_log(tensorboard , 'd_loss' , np.mean(dis_losses) ,
epoch)

```

**Listing 8.17** Kode program run.py

untuk menuliskan log losses ke Tensorboard

**8.1.2.26 Berikut adalah kode program menyimpan model. Jelaskan apa itu format h5 dan penjelasan dari kode program berikut :**

```

1 # In[ soal_26]
2     generator.save_weights(os.path.join("models", "generator_weights.h5"))
3     discriminator.save_weights(os.path.join("models", "discriminator_weights.h5"))

```

**Listing 8.18** Kode program run.py

untuk menyimpan berat model generator dan model diskriminasi kedalam file h5.

H5 adalah Hierarchical Data Format 5 File. File H5 adalah file data yang disimpan dalam Format Data Hirarki (HDF). Ini berisi array multidimensi data ilmiah. File H5 biasanya digunakan di luar angkasa, fisika, teknik, keuangan, penelitian akademis, genomik, astronomi, instrumen elektronik, dan bidang medis.

**8.1.2.27 Berikut adalah kode program testing model. Jelaskan dengan ilustrasi gambar dari mulai meload hingga membuat gambar 3D dengan menggunakan z sample, bisakah parameter z sample tersebut diubah? :**

```

1     generator.load_weights(os.path.join("models", "generator_weights.h5"), True)
2     discriminator.load_weights(os.path.join("models", "discriminator_weights.h5"), True)
3
4     # Generate 3D models
5     z_sample = np.random.normal(0, 1, size=[batch_size, 1, 1, 1,
z_size]).astype(np.float32)
6     generated_volumes = generator.predict(z_sample, verbose=3)
7
8     for i, generated_volume in enumerate(generated_volumes[:2]):
9         voxels = np.squeeze(generated_volume)
10        voxels[voxels < 0.5] = 0.
11        voxels[voxels >= 0.5] = 1.
12        saveFromVoxels(voxels, "results/gen_{}".format(i))

```

**Listing 8.19** Kode program run.py

untuk membuat mode predict/prediksi. dimana dia melakukan pembuatan model generator, model diskriminasi dan meload data h5 yang sudah dibuat tadi dan menggenerate nya kedalam model 3D, lalu menyimpan model voxel nya ke folde results.

### 8.1.3 Penanganan Error

#### 8.1.3.1 Terjadi error

1. terjadi error Summary has no attribute, seperti pada gambar berikut:

```
AttributeError                                Traceback (most recent call last)
<ipython-input-4-d9ec96389e6a> in <module>()
    233
    234      # Write losses to Tensorboard
--> 235      write_log(tensorboard, 'g_loss', np.mean(gen_losses), epoch)
    236      write_log(tensorboard, 'd_loss', np.mean(dis_losses), epoch)
    237

<ipython-input-4-d9ec96389e6a> in write_log(callback, name, value, batch_no)
    93
    94 def write_log(callback, name, value, batch_no):
---> 95     summary = tf.Summary()
    96     summary_value = summary.value.add()
    97     summary_value.simple_value = value

AttributeError: module 'tensorflow' has no attribute 'Summary'
```

Gambar 8.18 terjadi Error 1

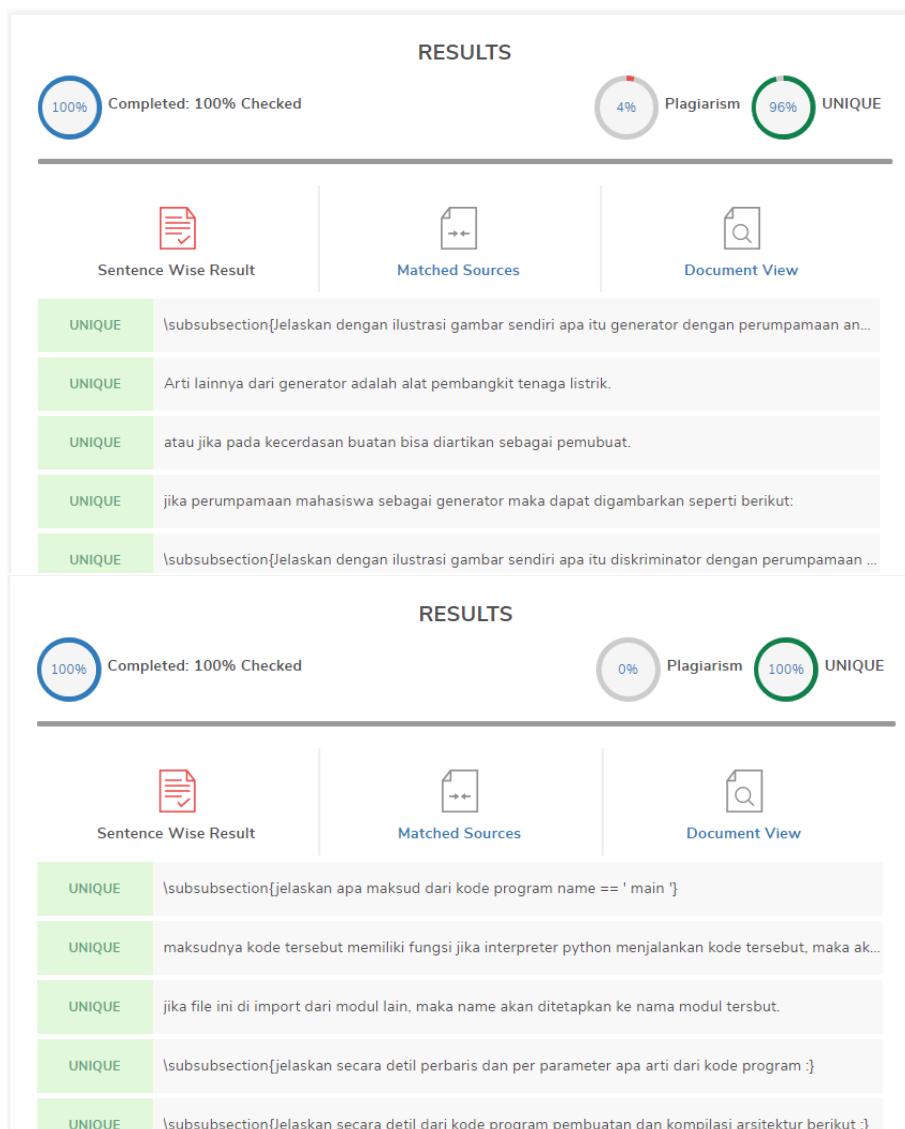
#### 8.1.3.2 Solusi

1. solusi dari error 1 ialah: dengan menginstall versi tensorflow 1.xx

```
(base) C:\Users\Bakti Qilan>pip install tensorflow==1.2...
```

Gambar 8.19 solusi Error 1

### 8.1.4 Bukti Tidak Plagiat



**Gambar 8.20** Bukti tidak plagiat

### 8.1.5 Link Youtube

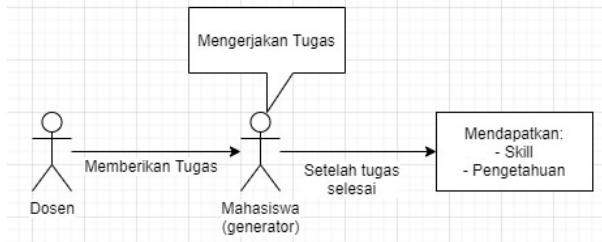
<https://bit.ly/baktiListVideo>

## 8.2 1174066 - D.Irga B. Naufal Fakhri

### 8.2.1 Teori

#### 8.2.1.1 Apa itu generator dengan perumpamaan anda sebagai mahasiswa sebagai generatornya.

Generator adalah pembuat dan didalam program, generator adalah pembuat sesuatu. misalnya:

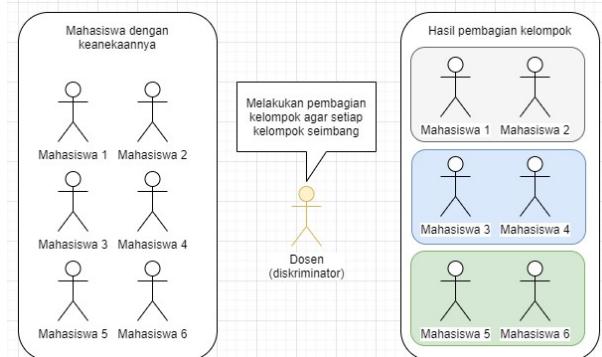


**Gambar 8.21** Penjelasan no.1

#### 8.2.1.2 Jelaskan dengan ilustrasi gambar sendiri apa itu diskriminator dengan perumpamaan dosen anda sebagai diskriminatorya.

Arti kata diskriminator adalah rangkaian dalam berbagai alat. berasal dari kata diskriminasi yang merujuk pada pelayanan yang tidak adil terhadap individu tertentu, di mana layanan ini dibuat berdasarkan karakteristik yang diwakili oleh individu tersebut. dalam machine learning, Discrimination itu bisa ada dua, yaitu:

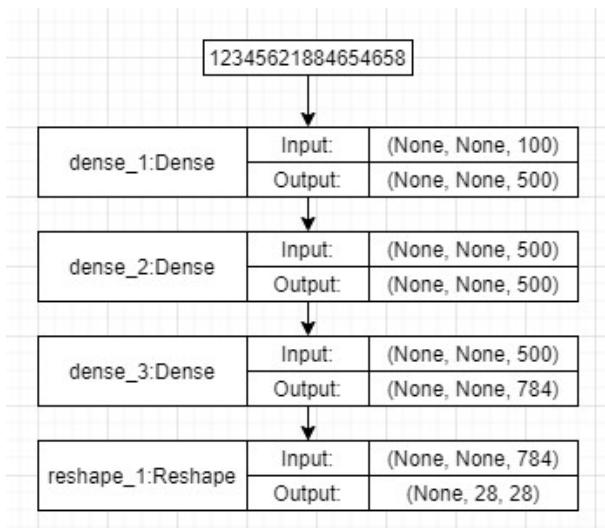
1. Disparate Treatment(perawatan berbeda) atau membedakan/ mengklasifikasikan sesuatu atau seseorang
2. Disparate Impact (dampak berbeda) ialah melihat konsekuensi klasifikasi/ pengambilan keputusan pada kelompok tertentu.



**Gambar 8.22** Penjelasan no.2

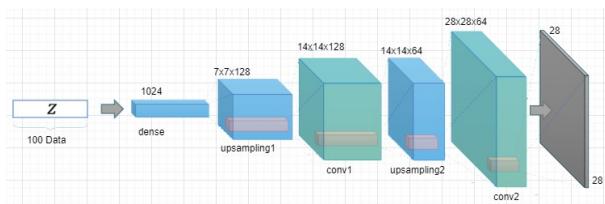
### 8.2.1.3 Bagaimana arsitektur generator dibuat

Aristekturn generator didalam GAN terdiri dari lima lapisan, yaitu: satu input layer, tiga dense layer dan satu output layer.



**Gambar 8.23** Arsitektur generator pada GAN-Project

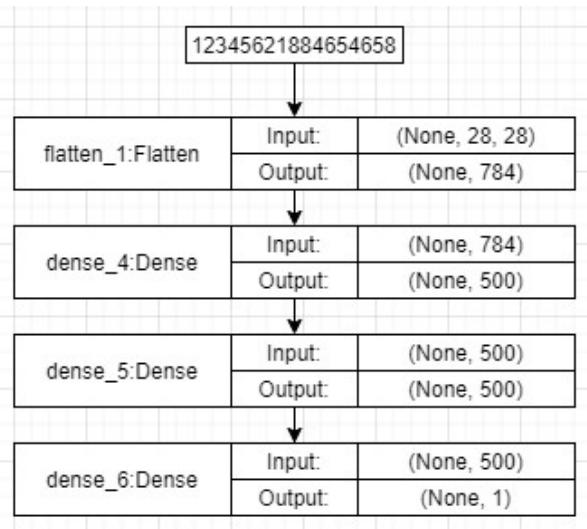
Dengan ilustrasi sebagai berikut:



**Gambar 8.24** Penjelasan no.3

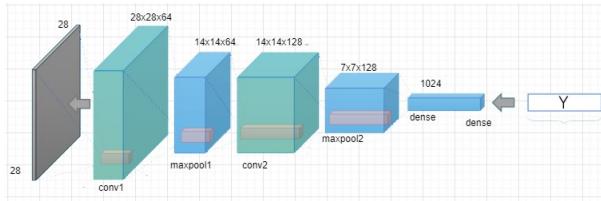
### 8.2.1.4 Bagaimana arsitektur diskriminator dibuat

aristekturn diskriminator didalam GAN terdiri dari lima lapisan, yaitu: satu input layer, tiga dense layer dan satu output layer. Diskriminator pada GAN-Project:



**Gambar 8.25** Arsitektur Diskriminatator pada GAN-Project

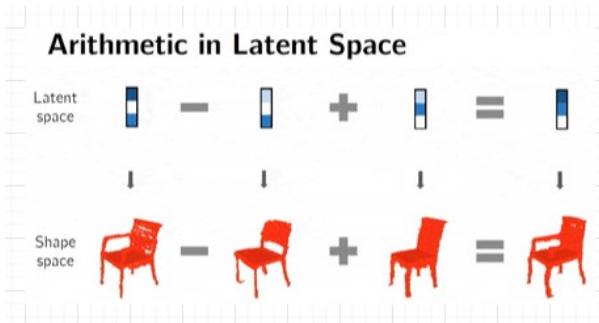
Dengan ilustrasi sebagai berikut:



**Gambar 8.26** Penjelasan no.4

#### 8.2.1.5 Apa itu latent space.

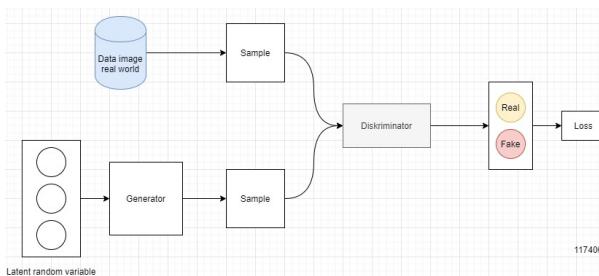
Latent space adalah space atau spot yang tersembunyi, sehingga akan mengambil titik data yang dekat dan yang mirip. Dengan ilustrasi sebagai berikut:



**Gambar 8.27** Penjelasan no.5

#### 8.2.1.6 Apa itu adversarial play

Dalam GAN adversarial play ialah persaingan antara generator dan diskriminator.



**Gambar 8.28** Penjelasan no.6

#### 8.2.1.7 Jelaskan dengan ilustrasi gambar apa itu Nash equilibrium

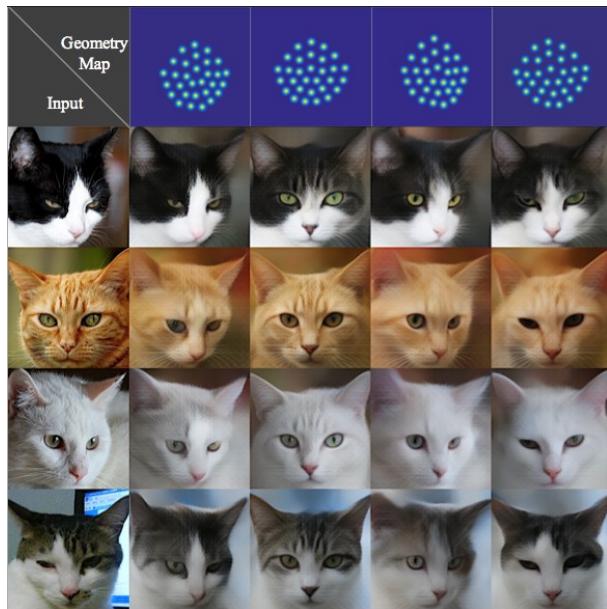
Nash equilibrium adalah strategi yang dipilih oleh masing-masing pemain, dengan strategi pemain lain tertentu. Strategi ini merupakan strategi terbaik dari masing-masing pemain dengan mempertimbangkan strategi tertentu yang diambil oleh pemain lainnya, atau yang disebut dengan Best Respons.

		TOM	
		A	B
SAM	A	1,1	1,-1
	B	-1,1	0,0

Gambar 8.29 Penjelasan no.7

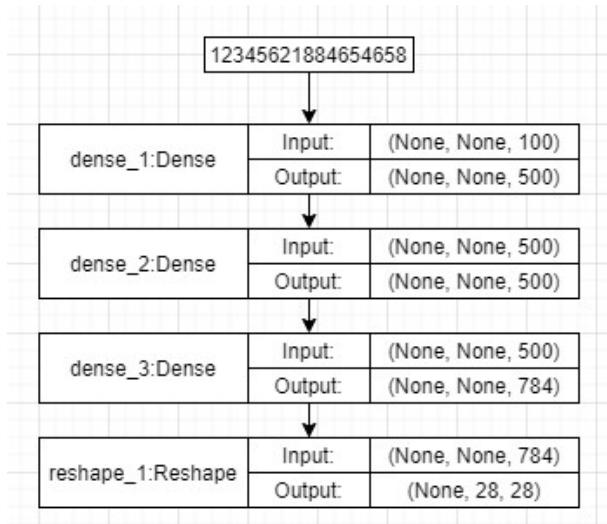
#### 8.2.1.8 Sebutkan dan jelaskan contoh-contoh implementasi dari GAN

Contoh implementasi dari GAN salah satunya adalah men-generate wajah kucing



**Gambar 8.30** Penjelasan no.8

**8.2.1.9 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)**



**Gambar 8.31** Arsitektur generator pada GAN-Project

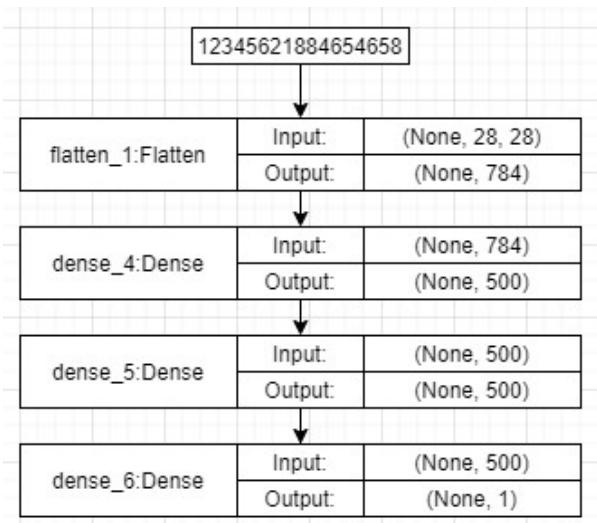
```

1 def generator_model():
2     model = Sequential([
3         Dense(1024, input_dim=100, activation='tanh'),
4         Dense(128*7*7),
5         Reshape((7, 7, 128)),
6         UpSampling2D(size=(2, 2)),
7         Conv2D(64, (5, 5), padding='same', activation='tanh'),
8         UpSampling2D(size=(2, 2)),
9         Conv2D(1, (5, 5), padding='same', activation='tanh')
10    ])
11    return model
12
13 generator_model().summary()

```

**Listing 8.20** Kode program arsitektur generator

**8.2.1.10 Berikan contoh dengan ilustrasi dari arsitektur diskriminatore dengan sebuah input layer, 3 buah hidden layer, dan satu output layer.**



**Gambar 8.32** Arsitektur Diskriminatore pada GAN-Project

```

1 def discriminator_model():
2     model = Sequential([
3         Conv2D(64, (5, 5), padding='same', input_shape=(28, 28, 1),
4             activation='tanh'),
5         MaxPooling2D(pool_size=(2, 2)),
6         Conv2D(128, (5, 5), activation='tanh'),
7         MaxPooling2D(pool_size=(2, 2)),
8         Flatten(),
9         Dense(1024, activation='tanh'),
10        Dense(1, activation='sigmoid')
11    ])
12
13 discriminator_model().summary()

```

```

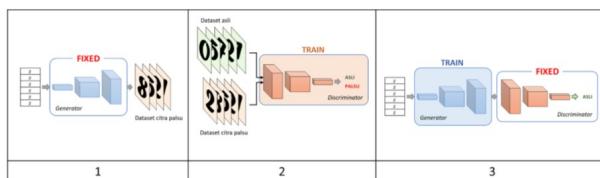
10     ])
11     return model
12
13 discriminator_model().summary()

```

**Listing 8.21** Kode program arsitektur diskriminat

#### **8.2.1.11 Kaitan output dan input antara generator dan diskriminat tersebut. Jelaskan kenapa inputan dan outputan seperti itu.**

Kaitan antara output pada generator dan inputan pada diskriminat adalah suatu keterhubungan yang akan menghasilkan data yang benar, karena output dari generator merupakan inoutan pada diskriminat



**Gambar 8.33** Penjelasan no.9

#### **8.2.1.12 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.**

diantara keduanya yang paling terkenal adalah KL divergence, karena KL divergence memiliki beberapa sifat/rumus yang bagus .salah satunya adalah  $KL[q; p]$  jenis daerah yang di mana  $q(x)$  memiliki massa non-null dan  $p(x)$  memiliki massa null. Ini mungkin terlihat seperti bug, tetapi sebenarnya ini merupakan fitur dalam situasi tertentu. sedangkan JS tidak memilikinya,

#### **8.2.1.13 Fungsi objektif yang berfungsi untuk mengukur kesamaan antara gambar yang dibuat dengan yang asli.**

fungsi objektif merupakan fungsi yang akan mengambil parameter data dan model sebagai argumen, dan dapat dievaluasi untuk mengembalikan angka sehingga mampu membedakan gambar yang asli dan yang di buat/fake.

#### **8.2.1.14 Scoring algoritma selain mean square error atau cross entropy seperti The Inception Score dan The Frechet Inception distance.**

Inception Score digunakan untuk mengukur seberapa realistik output dari GAN, dimana ada dua parameter, yaitu: gambarnya punya variasi dan setiap gambar jelas terlihat seperti sesuatu. Frechet Inception Distance adalah ukuran kesamaan antara dua dataset gambar. Itu terbukti berkorelasi baik dengan penilaian manusia terhadap kualitas visual dan paling sering digunakan untuk mengevaluasi kualitas sampel Generative Adversarial Networks. FID dihitung dengan menghitung jarak Fréchet antara dua Gaussians dipasang ke representasi fitur dari jaringan Inception.

### 8.2.1.15 Kelebihan dan kekurangan GAN

#### ▪ Kelebihan

1. GAN Menghasilkan data baru yang bisa hampir mirip dengan data asli. Karena hasil pelatihannya, GAN dapat menghasilkan data gambar, teks, audio, dan video yang dapat dibilang hampir mirip dengan yang aslinya. Berkat hal tersebut, GAN dapat digunakan dalam sistem marketing, e-commerce, games, iklan, dan industri lainnya
2. GAN mempelajari representasi data secara internal sehingga beberapa masalah pada machine learning dapat diatasi dengan mudah
3. Discriminator yang sudah dilatih dapat menjadi sebuah classifier atau pendekripsi jika data sudah sesuai. Karena Discriminator yang akan menjadi tidak efisien berkat seringnya dilatih
4. GAN dapat dilatih menggunakan data yang belum dilabeled

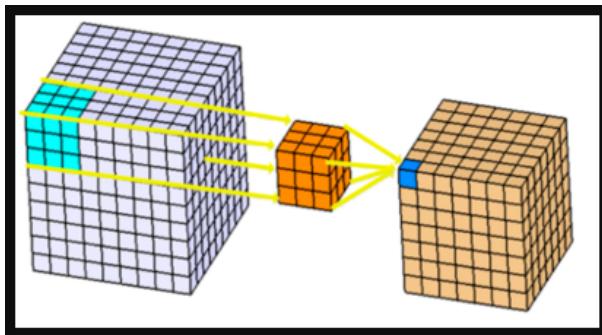
#### ▪ Kekurangan

1. Data saat diproses oleh metode gan tidak konvergensi
2. Jenis sampel yang dihasilkan oleh generator terbatas karena modenya terbatas
3. Ketidak seimbangnya antara generator dan discriminator dapat menyebabkan overfitting atau terlalu dekat dengan hasil sampel
4. Sangat sensitif dengan data yang sudah diinisiasi sebelumnya

### 8.2.2 Praktek

#### 8.2.2.1 Jelaskan apa itu 3D convolutions

3D convolutions adalah operasi konvolusi 3D yang menerapkan filter 3D ke data input dengan tiga arah, yaitu x,y, dan z. fitur ini menciptakan sebuah daftar peta yang ditumpuk.



Gambar 8.34 Penjelasan praktek no.1

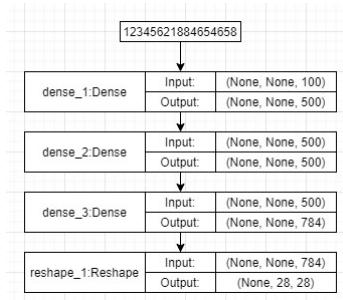
**8.2.2.2 Jelaskan dengan kode program arsitektur dari generator networknya, beserta penjelasan input dan output dari generator network.**

```

1 def build_generator():
2     """
3         Create a Generator Model with hyperparameters values defined as
4         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')(a)
28        a = BatchNormalization()(a, training=True)
29        a = Activation(activation=gen_activations[i + 1])(a)
30
31    gen_model = Model(inputs=[input_layer], outputs=[a])
32    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.



**Gambar 8.35** gambaran proses generator network

### 8.2.2.3 Jelaskan dengan kode program arsitektur dari diskriminatator network, beserta penjelasan input dan outputnya.

```

1 def build_discriminator():
2     """
3         Create a Discriminator Model using hyperparameters values defined
4             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    dis_convolutional_blocks = 5
16
17    dis_input_layer = Input(shape=dis_input_shape)
18
19    # The first 3D Convolutional block
20    a = Conv3D(filters=dis_filters[0],
21                kernel_size=dis_kernel_sizes[0],
22                strides=dis_strides[0],
23                padding=dis_paddings[0])(dis_input_layer)
24    # a = BatchNormalization()(a, training=True)
25    a = LeakyReLU(dis_alphas[0])(a)
26
27    # Next 4 3D Convolutional Blocks
28    for i in range(dis_convolutional_blocks - 1):
29        a = Conv3D(filters=dis_filters[i + 1],
30                    kernel_size=dis_kernel_sizes[i + 1],
31                    strides=dis_strides[i + 1],
32                    padding=dis_paddings[i + 1])(a)
33        a = BatchNormalization()(a, training=True)
34        if dis_activations[i + 1] == 'leaky_relu':
35            a = LeakyReLU(dis_alphas[i + 1])(a)
36        elif dis_activations[i + 1] == 'sigmoid':
37            a = Activation(activation='sigmoid')(a)

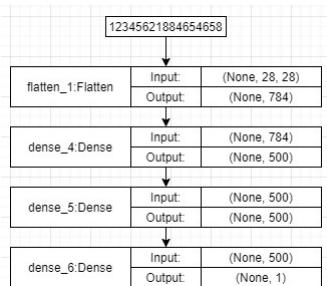
```

```

37     dis_model = Model(inputs=[dis_input_layer], outputs=[a])
38     return dis_model
39

```

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.



**Gambar 8.36** gambaran proses diskriminator network

#### 8.2.2.4 Jelaskan proses training 3D-GANs

Proses training 3D-GANs dilakukan seperti langkah-langkah berikut:

- Terdapat sebuah vektor noise dengan dimensi 200 dari distribusi Gaussian(normal).
- Meng-generate gambar palsu menggunakan model generator.
- Melatih jaringan generator dengan gambar yang asli(sampel dari data yang real) dan dengan gambar palsu yang dihasilkan oleh generator.
- Gunakan adversarial model untuk melatih generator model, jangan melatih diskriminator model.
- Ulangi langkah ini dengan jumlah epoch tertentu.

#### 8.2.2.5 Jelaskan bagaimana melakukan settingan awal chapter 02 untuk memenuhi semua kebutuhan sebelum melanjutkan ke tahapan persiapan data.

Persiapkan data 3DShapeNets, setelah itu lakukan seperti dibawah ini:

- membaca buku panduan dan keterangan dari Generative-Adversarial-Network project ini. lalu clone github <https://github.com/PacktPublishing/Generative-Adversarial-Networks-Projects>

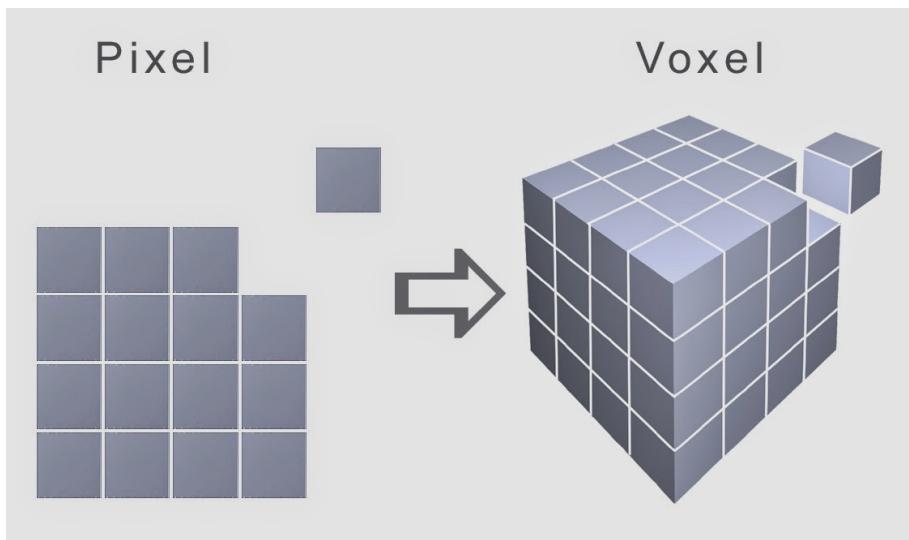
- siapkan komputer atau bisa menggunakan google colab
- install library sesuai requirements.txt atau membuat virtual env baru dan install librarynya.

**8.2.2.6 Jelaskan tentang dataset yang digunakan, dari mulai tempat unduh, cara membuka dan melihat data. sampai deskripsi dari isi dataset dengan detail penjelasan setiap folder/file yang membuat orang awam paham.**

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

**8.2.2.7 Jelaskan apa itu voxel dengan ilustrasi dan bahasa paling awam**

Secara singkat, voxel dapat diartikan sebagai 3D pixel atau pixel dalam bentuk 3D



**Gambar 8.37** Gambar Voxel dan Pixel

```

1 #In []: Testing Display
2 import scipy.io as io
3 voxels = io.loadmat("/content/3DShapeNets/volumetric_data/door/30/
4 test/door_000000011_1.mat")['instance']
5 import numpy as np
6 voxels = np.pad(voxels, (1, 1), 'constant', constant_values=(0, 0))
7
8 import scipy.ndimage as nd
9 voxels = nd.zoom(voxels, (2, 2, 2), mode='constant', order=0)
10
11 import matplotlib.pyplot as plt
12 from mpl_toolkits.mplot3d import Axes3D

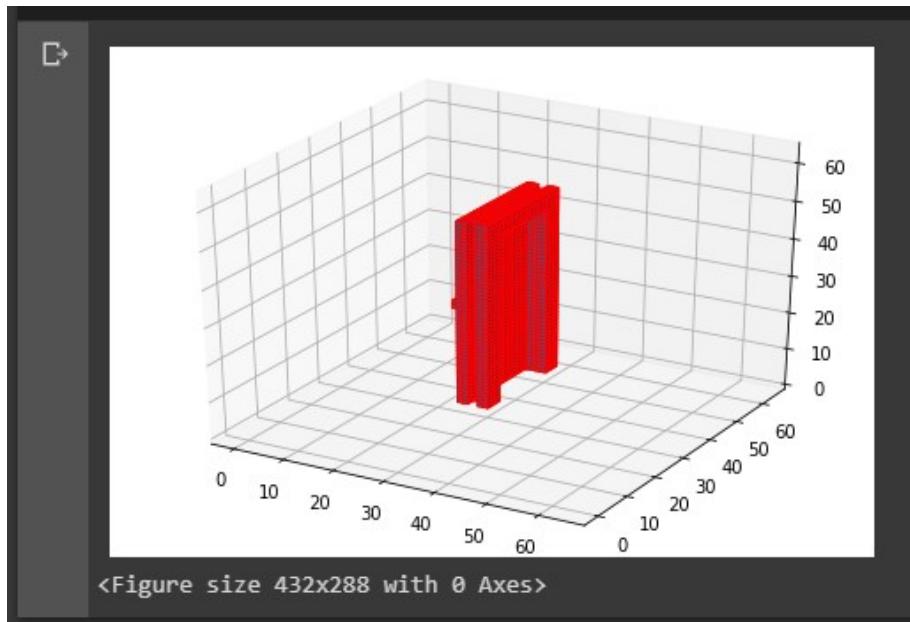
```

```

13 fig = plt.figure()
14 ax = Axes3D(fig)
15 ax.voxels(voxels, edgecolor="red")
16
17 plt.show()
18 plt.savefig('/content/drive/My Drive/zMachine Learning/3DShape/data')

```

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



**Gambar 8.38** Gambar Voxel dan Pixel

#### 8.2.2.9 Buka file run.py jelaskan perbaris kode pada fungsi untuk membuat generator yaitu build generator

```

1 def build_generator():
2     """
3         Create a Generator Model with hyperparameters values defined as
4         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)

```

```

10 gen_activations = ['relu', 'relu', 'relu', 'relu', 'sigmoid']
11 gen_convolutional_blocks = 5
12
13 input_layer = Input(shape=gen_input_shape)
14
15 # First 3D transpose convolution(or 3D deconvolution) block
16 a = Deconv3D(filters=gen_filters[0],
17               kernel_size=gen_kernel_sizes[0],
18               strides=gen_strides[0])(input_layer)
19 a = BatchNormalization()(a, training=True)
20 a = Activation(activation='relu')(a)
21
22 # Next 4 3D transpose convolution(or 3D deconvolution) blocks
23 for i in range(gen_convolutional_blocks - 1):
24     a = Deconv3D(filters=gen_filters[i + 1],
25                   kernel_size=gen_kernel_sizes[i + 1],
26                   strides=gen_strides[i + 1], padding='same')(a)
27     a = BatchNormalization()(a, training=True)
28     a = Activation(activation=gen_activations[i + 1])(a)
29
30 gen_model = Model(inputs=[input_layer], outputs=[a])
31 return gen_model

```

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

#### **8.2.2.10 jelaskan juga fungsi untuk membangun diskriminator pada fungsi build discriminator.**

```

1 def build_discriminator():
2     """
3         Create a Discriminator Model using hyperparameters values defined
4         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    dis_convolutional_blocks = 5
16
17    dis_input_layer = Input(shape=dis_input_shape)
18
19    # The first 3D Convolutional block
20    a = Conv3D(filters=dis_filters[0],
21               kernel_size=dis_kernel_sizes[0],
22               strides=dis_strides[0],
23               padding=dis_paddings[0])(dis_input_layer)
24    # a = BatchNormalization()(a, training=True)
25    a = LeakyReLU(dis_alphas[0])(a)

```

```

1 # Next 4 3D Convolutional Blocks
2 for i in range(dis_convolutional_blocks - 1):
3     a = Conv3D(filters=dis_filters[i + 1],
4                 kernel_size=dis_kernel_sizes[i + 1],
5                 strides=dis_strides[i + 1],
6                 padding=dis_paddings[i + 1])(a)
7     a = BatchNormalization()(a, training=True)
8     if dis_activations[i + 1] == 'leaky_relu':
9         a = LeakyReLU(dis_alphas[i + 1])(a)
10    elif dis_activations[i + 1] == 'sigmoid':
11        a = Activation(activation='sigmoid')(a)
12
13    dis_model = Model(inputs=[dis_input_layer], outputs=[a])
14
15 return dis_model

```

Kode di atas befungsi untuk membangun diskriminatator berfungsi untuk mendefenisikan seluruh gambar yang sudah di load generator sebagai gambar fake dan real.

#### **8.2.2.11 jelaskan apa maksud dari kode program name == 'main'**

```
1 if __name__ == '__main__':
```

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.

#### **8.2.2.12 jelaskan secara detil perbaris dan per parameter apa arti dari kode program :**

```

1 #In []: Soal 12
2 """
3     Specify Hyperparameters
4 """
5 object_name = "chair" #Membuat variable
6 data_dir = "/content/3DShapeNets/volumetric_data/" \
7             "{}/30/train/*.mat".format(object_name) #Memasukkan
8 direktori dari data training
9 gen_learning_rate = 0.0025 #Membuat rate pada generator
10 dis_learning_rate = 1e-5 #Membuat rate pada diskriminatator
11 beta = 0.5 #Menentukan beta
12 batch_size = 1 #Menentukan ukuran batch
13 z_size = 200 #Menentukan ukuran data training
14 epochs = 10 #Menentukan masa periode. setiap epoch memiliki 1
15 batch
16 MODE = "train" #Menentukan variable isinya train

```

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

#### **8.2.2.13 Jelaskan secara detil dari kode program pembuatan dan kompilasi arsitektur berikut :**

```

1 #In []: 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.

#### **8.2.2.14 Jelaskan secara detil kode program untuk membuat dan melakukan kompilasi model adversarial berikut:**

```

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

#### **8.2.2.15 Jelaskan Ekstrak dan load data kursi dengan menggunakan fungsi getVoxelsFormat dan get3DImages yang digunakan pada kode program berikut :**

```

1 #In []: Soal_15
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 befungsi untuk melakukan load data pada dataset.

#### **8.2.2.16 Jelaskan maksud dari kode program instansiasi TensorBoard yang menambahkan generator dan diskriminator pada program berikut:**

```

1 #In []: Soal_16

```

```

1 tensorboard = TensorBoard(log_dir="/content/drive/My Drive/
2 zMachine Learning/3DShape/logs/{}".format(time.time()))
3 tensorboard.set_model(generator)
4 tensorboard.set_model(discriminator)

```

Kode di atas berfungsi untuk membuat tensorboard untuk mencatat log yang akan dihasilkan dari training data kita.

#### **8.2.2.17 Jelaskan apa fungsi dari np.reshape ones zeros pada kode program berikut dengan parameternya:**

```

1 #In []: Soal 17
2 labels_real = np.reshape(np.ones((batch_size,)), (-1, 1, 1, 1, 1))
3 labels_fake = np.reshape(np.zeros((batch_size,)), (-1, 1, 1, 1, 1))

```

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

#### **8.2.2.18 Jelaskan kenapa harus ada perulangan dalam meraih epoch. Dan jelaskan apa itu epoch terkait kode program berikut:**

```

1 #In []: Soal 18
2     if MODE == 'train':
3         for epoch in range(epochs):
4             print("Epoch:", epoch)
5
6             gen_losses = []
7             dis_losses = []

```

Kode di atas befungsi untuk melakukan training epoch, karena jika epoch semakin banyak maka kualitas training yang dihasilkan akan semakin baik dan training epoch akan dilakukan jika variable mode berisikan train.

#### **8.2.2.19 Jelaskan apa itu batches dan kaitannya dengan kode program berikut, dan kenapa berada di dalam epoch:**

```

1 #In []: Soal 19
2     number_of_batches = int(volumes.shape[0] / batch_size)
3     print("Number of batches:", number_of_batches)
4     for index in range(number_of_batches):
5         print("Batch:", index + 1)

```

Batch adalah jumlah file yang akan di training.

#### **8.2.2.20 Berikut adalah kode program pengambilan gambar dan noise. Jelaskan apa fungsi np.random.normal serta astype, serta jelaskan apa arti parameter titik dua dan jelaskan isi dari z sample dan volumes batch:**

```

1 #In []: Soal 20
2             z_sample = np.random.normal(0, 0.33, size=[batch_size
3                 , 1, 1, 1, z_size]).astype(np.float32)
4                     volumes_batch = volumes[index * batch_size:(index +
5                         ) * batch_size, :, :, :]

```

Kode di atas befungsi untuk gambar menjadi bersih dari noise dan juga menye-suaikan shapenya.

**8.2.2.21 Berikut adalah kode program generator gambar palsu. Jelaskan apa fungsi generator.predict on batch, serta jelaskan apa arti parameter z sample:**

```

1 #In []: Soal 21
2             # Next, generate volumes using the generate network
3             gen_volumes = generator.predict_on_batch(z_sample)

```

Kode di atas befungsi untuk membuat sample gambar atau mempredict gambar yang nantinya akan diteruskan ke diskriminator.

**8.2.2.22 Berikut adalah kode program training diskriminator dengan gambar palsu dari generator dan gambar asil. Jelaskan apa maksudnya harus dilakukan training diskriminator secara demikian dan jelaskan apa isi loss fake dan loss real serta d loss dan fungsi train on batch.**

```

1 #In []: Soal 22
2             """
3             Train the discriminator network
4             """
5             discriminator.trainable = True
6             if index % 2 == 0:
7                 loss_real = discriminator.train_on_batch(
8                     volumes_batch, labels_real)
9                 loss_fake = discriminator.train_on_batch(
10                    gen_volumes, labels_fake)
11
12                 d_loss = 0.5 * np.add(loss_real, loss_fake)
13                 print("d_loss:{})".format(d_loss))
14
15             else:
16                 d_loss = 0.0
17
18             discriminator.trainable = False

```

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

**8.2.2.23 Berikut adalah kode program training model adversarial yang terda-pat generator dan diskriminator. Jelaskan apa bagaimana proses terbentuknya parameter z dan g loss:**

```

1 #In []: Soal 23
2         """
3             Train the generator network
4             """
5             z = np.random.normal(0, 0.33, size=[batch_size , 1, 1,
6             1, z_size]).astype(np.float32)
7             g_loss = adversarial_model.train_on_batch(z,
8             labels_real)
9             print("g_loss:{}".format(g_loss))
10            gen_losses.append(g_loss)
11            dis_losses.append(d_loss)

```

Kode di atas befungsi untuk melakukan print gloss untuk generator dan juga dloss untuk diskriminatator.

**8.2.2.24 Berikut adalah kode program generate dan menyimpan gambar 3D setelah beberapa saat setiap epoch. Jelaskan mengapa ada perulangan dengan parameter tersebut, serta jelaskan arti setiap variabel beserta perlihatkan isinya dan artikan isinya :**

```

1 #In []: Soal 24
2         # Every 10th mini-batch , generate volumes and save
3         them
4         if index % 10 == 0:
5             z_sample2 = np.random.normal(0, 0.33, size=[batch_size , 1, 1, 1, z_size]).astype(np.float32)
6             generated_volumes = generator.predict(z_sample2 ,
7             verbose=3)
8             for i, generated_volume in enumerate(
9             generated_volumes[:5]):
10                voxels = np.squeeze(generated_volume)
11                voxels[voxels < 0.5] = 0.
12                voxels[voxels >= 0.5] = 1.
13                saveFromVoxels(voxels , "/content/drive/My
Drive/zMachine_Learning/3DShape/results/img_{}-{}-{}".format(
epoch , index , i))

```

melakukan perulangan dimana setiap terdapat 10 mini-batch akan meng-generate volumenya dan akan menyimpannya lalu membandingkannya.

**8.2.2.25 Berikut adalah kode program menyimpan average losses setiap epoch. Jelaskan apa itu tensorboard dan setiap parameter yang digunakan pada kode program ini :**

```

1 #In []: Soal 25
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)

```

menuliskan log losses ke Tensorboard agar disimpan

**8.2.2.26 Berikut adalah kode program menyimpan model. Jelaskan apa itu format h5 dan penjelasan dari kode program berikut :**

```

1 #In []: Soal 26
2     """
3         Save models
4     """
5         generator.save_weights(os.path.join("models", "generator_weights.h5"))
6         discriminator.save_weights(os.path.join("models", "discriminator_weights.h5"))

```

untuk menyimpan berat model generator dan model diskriminator kedalam file h5.

H5 adalah Hierarchical Data Format 5 File. File H5 adalah file data yang disimpan dalam Format Data Hirarki (HDF). Ini berisi array multidimensi data ilmiah. File H5 biasanya digunakan di luar angkasa, fisika, teknik, keuangan, penelitian akademis, genomik, astronomi, instrumen elektronik, dan bidang medis.

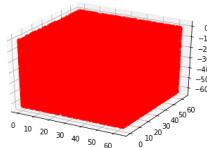
**8.2.2.27 Berikut adalah kode program testing model. Jelaskan dengan ilustrasi gambar dari mulai meload hingga membuat gambar 3D dengan menggunakan z sample, bisakah parameter z sample tersebut diubah? :**

```

1 #In []: Soal 27
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", "generator_weights.h5"), True)
9         discriminator.load_weights(os.path.join("models", "discriminator_weights.h5"), True)
10
11         # Generate 3D models
12         z_sample = np.random.normal(0, 1, size=[batch_size, 1, 1, 1,
13             z_size]).astype(np.float32)
14         generated_volumes = generator.predict(z_sample, verbose=3)
15
16         for i, generated_volume in enumerate(generated_volumes[:2]):
17             voxels = np.squeeze(generated_volume)
18             voxels[voxels < 0.5] = 0.
19             voxels[voxels >= 0.5] = 1.
20             saveFromVoxels(voxels, "/content/drive/My Drive/zMachine
Learning/3DShape/results/gen_{}".format(i))

```

untuk membuat mode predict/prediksi. dimana dia melakukan pembuatan model generator, model diskriminator dan meload data h5 yang sudah dibuat tadi dan menggenerate nya kedalam model 3D, lalu menyimpan model voxel nya ke folder results.



**Gambar 8.39** Salah satu hasil

### 8.2.3 Penanganan Error

#### 8.2.3.1 Error

- OSError



**Gambar 8.40** OSError

#### 8.2.3.2 Solusi Error

- OSError

Pastikan folder telah ada

### 8.2.4 Bukti Tidak Plagiat



**Gambar 8.41** Bukti tidak plagiat

### 8.2.5 Link Youtube

<https://bit.ly/KecerdasanBuatanDirga>

## 8.3 1174087 - Ilham Muhammad Ariq

### 8.3.1 Teori

#### 8.3.1.1 Apa itu generator dengan perumpamaan anda sebagai mahasiswa sebagai generatornya.

Generator adalah pembuat sesuatu atau dapat menghasilkan sesuatu. Jika perumpamaannya mahasiswa maka seperti ini :

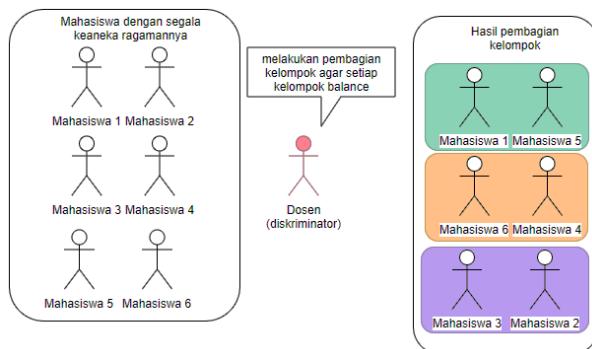


Gambar 8.42 No 1

#### 8.3.1.2 Jelaskan dengan ilustrasi gambar sendiri apa itu diskriminator dengan perumpamaan dosen anda sebagai diskriminatorya.

Arti kata diskriminator adalah rangkaian dalam berbagai alat. Berasal dari kata diskriminasi yang merujuk pada pelayanan yang tidak adil terhadap individu tertentu, di mana layanan ini dibuat berdasarkan karakteristik yang diwakili oleh individu tersebut. dalam machine learning, Discrimination itu bisa ada dua, yaitu:

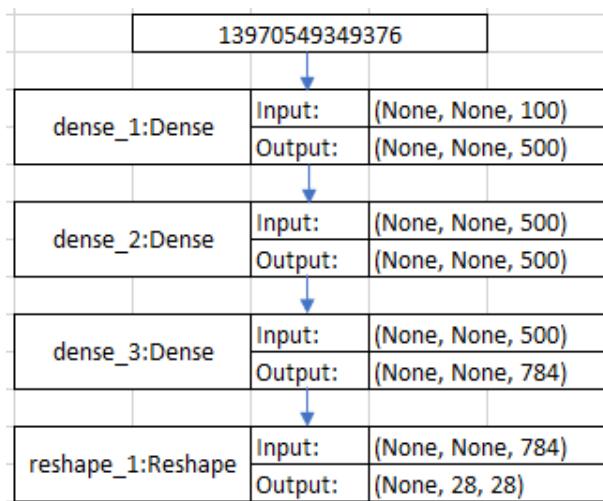
1. Disparate Treatment (perawatan berbeda) atau membedakan/ mengklasifikasikan sesuatu atau seseorang
2. Disparate Impact (dampak berbeda) ialah melihat konsekuensi klasifikasi/ pengambilan keputusan pada kelompok tertentu.



Gambar 8.43 No 2

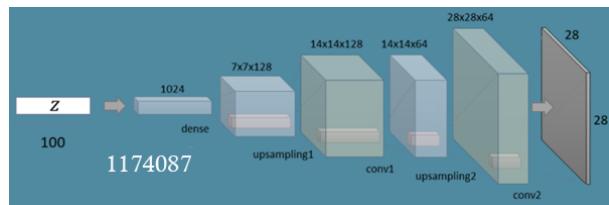
### 8.3.1.3 Bagaimana arsitektur generator dibuat

Aristekturn generator didalam GAN terdiri dari lima lapisan, yaitu: satu input layer, tiga dense layer dan satu output layer.



**Gambar 8.44** Arsitektur generator pada GAN-Project

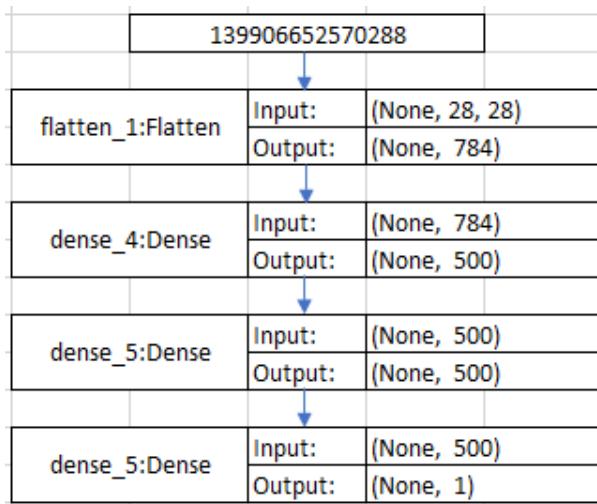
Dengan ilustrasi sebagai berikut:



**Gambar 8.45** No 3

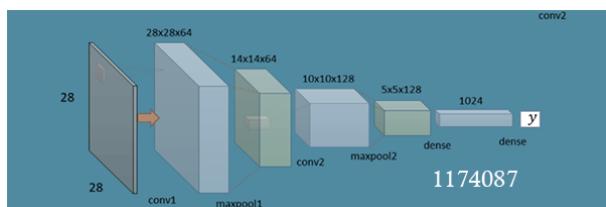
### 8.3.1.4 Bagaimana arsitektur diskriminator dibuat

aristekturn diskriminator didalam GAN terdiri dari lima lapisan, yaitu: satu input layer, tiga dense layer dan satu output layer. Diskriminator pada GAN-Project:



**Gambar 8.46** Arsitektur Diskriminatator pada GAN-Project

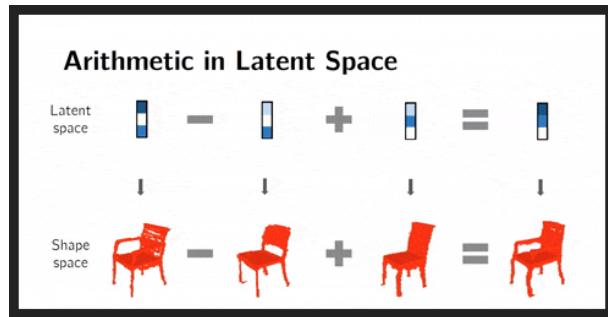
Dengan ilustrasi sebagai berikut:



**Gambar 8.47** No 4

#### 8.3.1.5 Apa itu latent space.

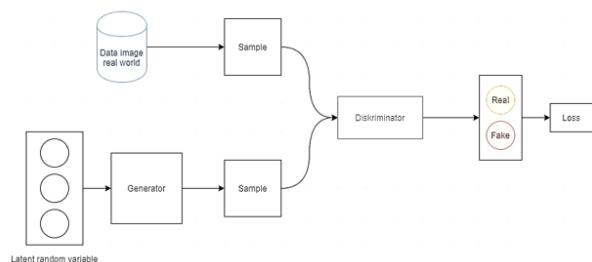
Latent space adalah space atau spot yang tersembunyi, sehingga akan mengambil titik data yang dekat dan yang mirip. Dengan ilustrasi sebagai berikut:



Gambar 8.48 No 5

### 8.3.1.6 Apa itu adversarial play

Dalam GAN adversarial play ialah persaingan antara generator dan diskriminator.



Gambar 8.49 No 6

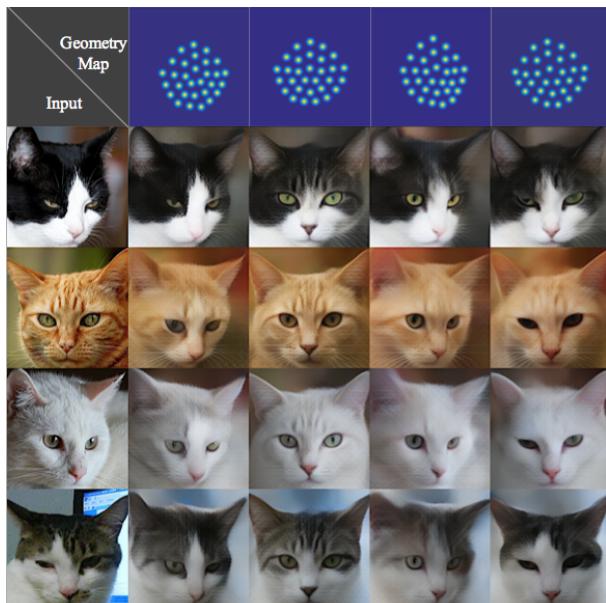
### 8.3.1.7 Jelaskan dengan ilustrasi gambar apa itu Nash equilibrium

Nash equilibrium adalah strategi yang dipilih oleh masing-masing pemain, dengan strategi pemain lain tertentu. Strategi ini merupakan strategi terbaik dari masing-masing pemain dengan mempertimbangkan strategi tertentu yang diambil oleh pemain lainnya, atau yang disebut dengan Best Respons.

		TOM	
		A	B
SAM	A	1,1	1,-1
	B	-1,1	0,0

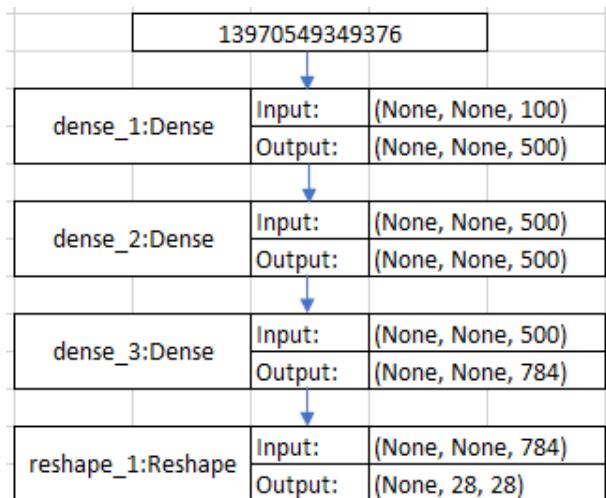
Gambar 8.50 No 7

**8.3.1.8 Sebutkan dan jelaskan contoh-contoh implementasi dari GAN**  
Contoh implementasi dari GAN salah satunya adalah men-generate wajah kucing



Gambar 8.51 No 8

**8.3.1.9 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)**



**Gambar 8.52 Arsitektur generator pada GAN-Project**

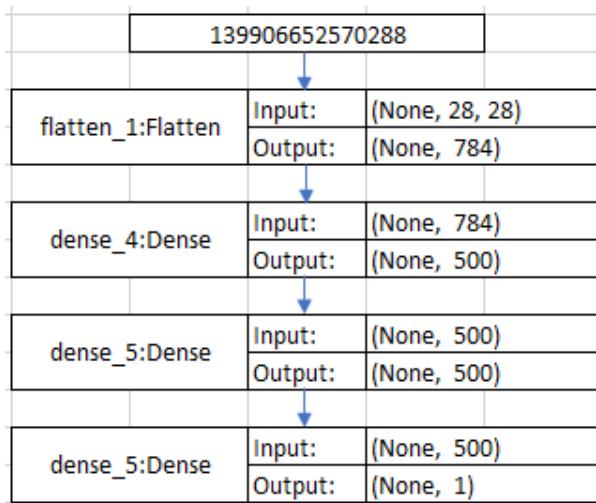
```

1 def generator_model():
2     model = Sequential([
3         Dense(1024, input_dim=100, activation='tanh'),
4         Dense(128*7*7),
5         Reshape((7, 7, 128)),
6         UpSampling2D(size=(2, 2)),
7         Conv2D(64, (5, 5), padding='same', activation='tanh'),
8         UpSampling2D(size=(2, 2)),
9         Conv2D(1, (5, 5), padding='same', activation='tanh')
10    ])
11    return model
12
13 generator_model().summary()

```

**Listing 8.22 Kode program arsitektur generator**

**8.3.1.10 Berikan contoh dengan ilustrasi dari arsitektur dikriminator dengan sebuah input layer, 3 buah hidden layer, dan satu output layer.**



**Gambar 8.53** Arsitektur Diskriminator pada GAN-Project

```

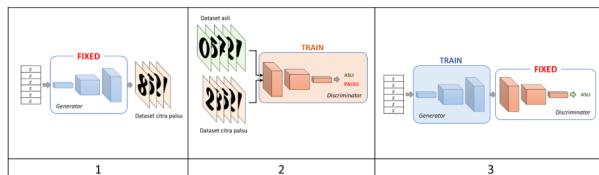
1 def discriminator_model():
2     model = Sequential([
3         Conv2D(64, (5, 5), padding='same', input_shape=(28, 28, 1),
4             activation='tanh'),
5         MaxPooling2D(pool_size=(2, 2)),
6         Conv2D(128, (5, 5), activation='tanh'),
7         MaxPooling2D(pool_size=(2, 2)),
8         Flatten(),
9         Dense(1024, activation='tanh'),
10        Dense(1, activation='sigmoid')
11    ])
12    return model
13
14 discriminator_model().summary()

```

**Listing 8.23** Kode program arsitektur diskriminator

### 8.3.1.11 Kaitan output dan input antara generator dan diskriminator tersebut. Jelaskan kenapa inputan dan outputan seperti itu.

Kaitan antara output pada generator dan inputan pada diskriminator adalah suatu keterhubungan yang akan menghasilkan data yang benar, karena output dari generator merupakan inoutan pada diskriminator



Gambar 8.54 No 9

#### **8.3.1.12 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.**

diantara keduanya yang paling terkenal adalah KL divergence, karena KL divergence memiliki beberapa sifat/rumus yang bagus .salah satunya adalah  $KL[q; p]$  jenis daerah yang di mana  $q(x)$  memiliki massa non-null dan  $p(x)$  memiliki massa null. Ini mungkin terlihat seperti bug, tetapi sebenarnya ini merupakan fitur dalam situasi tertentu. sedangkan JS tidak memiliki

#### **8.3.1.13 Fungsi objektif yang berfungsi untuk mengukur kesamaan antara gambar yang dibuat dengan yang asli.**

fungsi objektif merupakan fungsi yang akan mengambil parameter data dan model sebagai argumen, dan dapat dievaluasi untuk mengembalikan angka sehingga mampu membedakan gambar yang asli dan yang di buat/fake.

#### **8.3.1.14 Scoring algoritma selain mean square error atau cross entropy seperti The Inception Score dan The Frechet Inception distance.**

Inception Score digunakan untuk mengukur seberapa realistik output dari GAN, di mana ada dua parameter, yaitu: gambarnya punya variasi dan setiap gambar jelas terlihat seperti sesuatu. Frechet Inception Distance adalah ukuran kesamaan antara dua dataset gambar. Itu terbukti berkorelasi baik dengan penilaian manusia terhadap kualitas visual dan paling sering digunakan untuk mengevaluasi kualitas sampel Generative Adversarial Networks. FID dihitung dengan menghitung jarak Fréchet antara dua Gaussians dipasang ke representasi fitur dari jaringan Inception.

#### **8.3.1.15 Kelebihan dan kekurangan GAN**

- Kelebihan

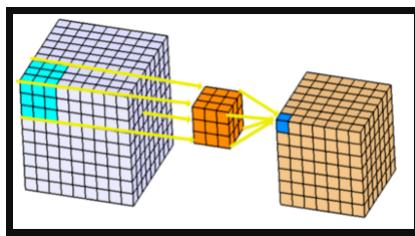
1. GAN Menghasilkan data baru yang bisa hampir mirip dengan data asli. Karena hasil pelatihannya, GAN dapat menghasilkan data gambar, teks, audio, dan video yang dapat dibilang hampir mirip dengan yang aslinya. Berkat hal tersebut, GAN dapat digunakan dalam sistem marketing, e-commerce, games, iklan, dan industri lainnya
2. GAN mempelajari representasi data secara internal sehingga beberapa masalah pada machine learning dapat diatasi dengan mudah

3. Discriminator yang sudah dilatih dapat menjadi sebuah classifier atau pen-deteksi jika data sudah sesuai. Karena Discriminator yang akan menjadi tidak efisien berkat seringnya dilatih
  4. GAN dapat dilatih menggunakan data yang belum dilabeled
- Kekurangan
    1. Data saat diproses oleh metode gan tidak konvergensi
    2. Jenis sampel yang dihasilkan oleh generator terbatas karena modenya ter-batas
    3. Ketidak seimbangnya antara generator dan discriminator dapat menyebabkan overfitting atau terlalu dekat dengan hasil sampel
    4. Sangat sensitif dengan data yang sudah diinisiasi sebelumnya

### 8.3.2 Praktek

#### 8.3.2.1 Jelaskan apa itu 3D convolutions

3D convolutions merupakan operasi konvolusi 3D yang menerapkan filter 3D ke data input dengan tiga arah, yaitu x,y, dan z. fitur ini menciptakan sebuah daftar peta yang ditumpuk. agar lebih mudah saya menggunakan sebuah ilustrasi seperti berikut:



Gambar 8.55 gambaran 3D Convolutions

#### 8.3.2.2 Jelaskan dengan kode program arsitektur dari generator networknya, beserta penjelasan input dan output dari generator network.

```

1 def build_generator():#nama kelasnya
2     """
3         Create a Generator Model with hyperparameters values defined as
4         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

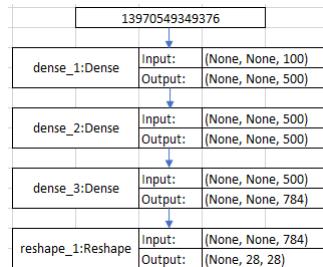
```

```

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

```

kode diatas merupakan generator network yang terdiri dari lima layer, satu input layer, tiga hidden layer, dan satu output layer.Dan prosesnya dapat digambarkan seperti berikut:



**Gambar 8.56** gambaran proses generator network

penjelasan

- Input layer mengambil 100-dimensi sampel dari distribusi Gaussian(normal) dan meneruskan tensor ke. hidden layer pertama tanpa ada modifikasi
- ketiga hidden layer adalah dense layer dengan unit masing-masing 500, 500, dan 784. dimana dense layer pertama mengkonversi bentuk tensor (batch\_size, 100) ke bentuk tensor(batch\_size, 500). sedangkan pada layer dense kedua menghasilkan bentuk tensor (batch\_size, 500), dan layer dense ketiga menghasilkan (batch\_size, 784)
- Output layer, tensor akan dibentuk kembali dari bentuk tensor (batch\_size, 784) menjadi (batch\_size, 28, 28) artinya hasil dari generator ini akan menghasilkan banyak gambar, dimana setiap gambarnya memiliki ukuran (28, 28).

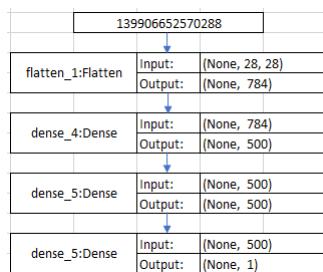
**8.3.2.3 Jelaskan dengan kode program arsitektur dari diskriminator network, beserta penjelasan input dan outputnya.**

```

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

```

kode diatas merupakan diskriminator network, dimana prosesnya dapat digambarkan seperti berikut:



**Gambar 8.57** gambaran proses diskriminator network

penjelasan

- awalnya diskriminator menerima input dengan bentuk 28x28.
- Input layer mengambil tensor input dan meneruskannya ke hidden layer pertama tanpa modifikasi apapun.
- lalu flattens layer akan meratakan tensor menjadi 784-dimensi vektor, yang akan diteruskan ke hidden layer (dense layer) pertama. hidden layer pertama dan kedua aka memodifikasinya menjadi vektor 500-dimensi.
- Output layer mesuk kedalam dense layer, dengan satu unit neuron dan sigmoid sebagai fungsi aktivasi. sigmoid akan menghasilkan nilai tunggal, 0 atau 1. Nilai 0 akan menunjukan bahwa gambar yang diberikan palsu. sebaliknya jika nilai 1 maka gambar asli.

#### 8.3.2.4 Jelaskan proses training 3D-GANs

Proses training 3D-GANs dilakukan seperti langkah-langkah berikut:

- Terdapat sebuah vektor noise dengan dimensi 200 dari distribusi Gaussian(normal).
- Meng-generate gambar palsu menggunakan model generator.
- Melatih jaringan generator dengan gambar yang asli(sampel dari data yang real) dan dengan gambar palsu yang dihasilkan oleh generator.
- Gunakan adversarial model untuk melatih generator model, jangan melatih diskriminator model.
- Ulangi langkah ini dengan jumlah epoch tertentu.

#### 8.3.2.5 Jelaskan bagaimana melakukan settingan awal chapter 02 untuk memenuhi semua kebutuhan sebelum melanjutkan ke tahapan persiapan data.

Sebelum melakukan tahapan persiapan data, kita harus melakukan beberapa hal seperti berikut:

- membaca buku panduan atau keterangan dari Generative-Adversarial-Network project ini. bisa didownload di portal kampus keren atau di github <https://github.com/PacktPublishing/Generative-Adversarial-Networks-Projects>
- persiapkan laptop/pc dengan spek yang cukup bagus(tidak kentang). jika tidak ada, bisa menggunakan google colab(cara penggunaanya akan dijelaskan di video).
- usahakan versi yang terinstall sama dengan versi yang ada pada requirement.txt agar terhindar dari error.

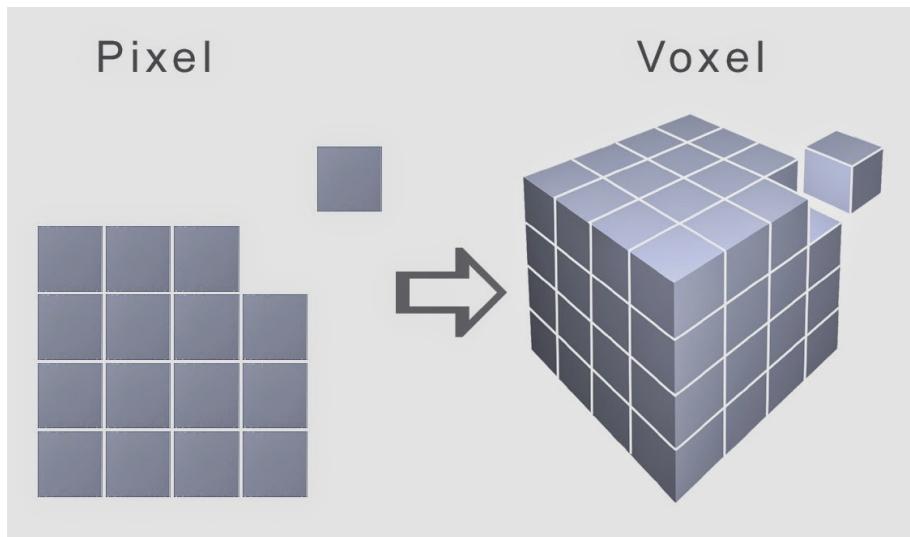
#### ***8.3.2.6 Jelaskan tentang dataset yang digunakan, dari mulai tempat unduh, cara membuka dan melihat data. sampai deskripsi dari isi dataset dengan detail penjelasan setiap folder/file yang membuat orang awam paham.***

Penjelasan dataset

- dataset bisa didownload pada halaman: <http://3dshapenets.cs.princeton.edu/3DShapeNets>
- Untuk membuka data, setelah didownload cukup di ekstrak saja menggunakan winrar/ 7zip
- Untuk mengetahui lebih jelasnya tentang dataset ini bisa dibuka saja file README.md, disana akan dijelaskan semuanya.

#### ***8.3.2.7 Jelaskan apa itu voxel dengan ilustrasi dan bahasa paling awam***

Secara singkat, voxel dapat diartikan sebagai 3D pixel atau pixel dalam bentuk 3D. seperti pada ilustrasi berikut:



**Gambar 8.58 Pixel VS Voxel**

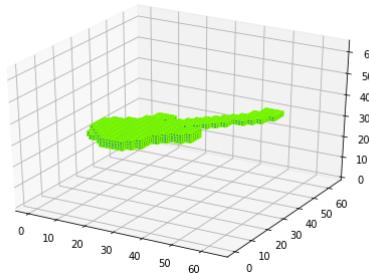
### 8.3.2.8 Visualisasikan dataset tersebut dalam tampilan visual plot, jelaskan cara melakukan visualisasinya

```

1 import scipy.io as io
2 import matplotlib.pyplot as plt
3 import scipy.ndimage as nd
4 import numpy as np
5 from mpl_toolkits.mplot3d import Axes3D
6
7 voxels = io.loadmat("3DShapeNets/volumetric_data/guitar/30/test/
8     guitar_000000003_1.mat")['instance']
9 voxels = np.pad(voxels, (1, 1), 'constant', constant_values=(0, 0))
10 voxels = nd.zoom(voxels, (2, 2, 2), mode='constant', order=0)
11
12 fig = plt.figure()
13 ax = Axes3D(fig)
14 ax.voxels(voxels, edgecolor="chartreuse")
15 plt.show()
16 plt.savefig('guitar')

```

Kode diatas berfungsi untuk melakukan visualisasi dataset dalam tampilan plot. dengan tahapan import library, load data file .mat, dan lakukan read memakai matplotlib. dan hasilnya seperti berikut



**Gambar 8.59** Visualisasi dataset(contoh)

### 8.3.2.9 Buka file run.py jelaskan perbaris kode pada fungsi untuk membuat generator yaitu build\_generator

```

1 z_size = 200
2 gen_filters = [512, 256, 128, 64, 1]
3 gen_kernel_sizes = [4, 4, 4, 4, 4]
4 gen_strides = [1, 2, 2, 2, 2]
5 gen_input_shape = (1, 1, 1, z_size)
6 gen_activations = ['relu', 'relu', 'relu', 'relu', 'sigmoid']
7 gen_convolutional_blocks = 5

```

potongan kode diatas berfungsi sebagai penentuan nilai untuk hyperparameters yang berbeda.

```
1 input_layer = Input(shape=gen_input_shape)
```

potongan kode diatas berfungsi untuk membuat input layer

```
1 # First 3D transpose convolution(or 3D deconvolution) block
2 a = Deconv3D(filters=gen_filters[0],
3               kernel_size=gen_kernel_sizes[0],
4               strides=gen_strides[0])(input_layer)
5 a = BatchNormalization()(a, training=True)
6 a = Activation(activation='relu')(a)
```

potongan kode diatas berfungsi untuk menambahkan 3D transpose pertama.

```
1 # Next 4 3D transpose convolution(or 3D deconvolution) blocks
2 for i in range(gen_convolutional_blocks - 1):
3     a = Deconv3D(filters=gen_filters[i + 1],
4                   kernel_size=gen_kernel_sizes[i + 1],
5                   strides=gen_strides[i + 1], padding='same')(a)
6     a = BatchNormalization()(a, training=True)
7     a = Activation(activation=gen_activations[i + 1])(a)
```

potongan kode diatas berfungsi untuk menambahkan empat 3D transpose lainnya.

```
1 gen_model = Model(inputs=[input_layer], outputs=[a])
```

potongan kode diatas berfungsi untuk membuat model keras dan menentukan input dan output untuk network generator.

### **8.3.2.10 jelaskan juga fungsi untuk membangun diskriminator pada fungsi build discriminator.**

```
1 dis_input_shape = (64, 64, 64, 1)
2 dis_filters = [64, 128, 256, 512, 1]
3 dis_kernel_sizes = [4, 4, 4, 4, 4]
4 dis_strides = [2, 2, 2, 2, 1]
5 dis_paddings = ['same', 'same', 'same', 'same', 'valid']
6 dis_alphas = [0.2, 0.2, 0.2, 0.2, 0.2]
7 dis_activations = ['leaky_relu', 'leaky_relu', 'leaky_relu',
```

potongan kode diatas berfungsi sebagai penentuan nilai untuk hyperparameters yang berbeda.

```
1 dis_input_layer = Input(shape=dis_input_shape)
```

potongan kode diatas berfungsi untuk membuat input layer, berupa gambar 3D dengan dimensi 64x64x64x1

```
1 # The first 3D Convolutional block
2 a = Conv3D(filters=dis_filters[0],
3            kernel_size=dis_kernel_sizes[0],
4            strides=dis_strides[0],
5            padding=dis_paddings[0])(dis_input_layer)
6 # a = BatchNormalization()(a, training=True)
7 a = LeakyReLU(dis_alphas[0])(a)
```

potongan kode diatas berfungsi untuk menambahkan 3D transpose pertama.

```

1 # Next 4 3D Convolutional Blocks
2 for i in range(dis_convolutional_blocks - 1):
3     a = Conv3D(filters=dis_filters[i + 1],
4                 kernel_size=dis_kernel_sizes[i + 1],
5                 strides=dis_strides[i + 1],
6                 padding=dis_paddings[i + 1])(a)
7     a = BatchNormalization()(a, training=True)
8     if dis_activations[i + 1] == 'leaky_relu':
9         a = LeakyReLU(dis_alphas[i + 1])(a)
10    elif dis_activations[i + 1] == 'sigmoid':
11        a = Activation(activation='sigmoid')(a)

```

potongan kode diatas berfungsi untuk menambahkan empat 3D transpose lainnya.

```
1 dis_model = Model(inputs=[dis_input_layer], outputs=[a])
```

potongan kode diatas berfungsi untuk membuat model keras dan menentukan input dan output untuk network generator.

### **8.3.2.11 jelaskan apa maksud dari kode program name == 'main'**

```
1 if __name__ == '__main__':
```

**Listing 8.24** Kode program run.py

maksudnya kode tersebut memiliki fungsi jika interpreter python menjalankan kode tersebut, maka akan menetapkan variable name untuk memiliki nilai main. jika file ini di import dari modul lain, maka name akan ditetapkan ke nama modul tersebut.

### **8.3.2.12 jelaskan secara detil perbaris dan per parameter apa arti dari kode program :**

Penjelasan kode:

```

1 # In[ soal_12]
2     object_name = "airplane" #membuat variabel
3     data_dir = "3DShapeNets/volumetric_data/" \
4                 "{}/30/train/*.mat".format(object_name) #menunjukkan
5     direktori dari variable sebelumnya
6     gen_learning_rate = 0.0025 #membuat rate pada generator
7     dis_learning_rate = 1e-5 #membuat rate pada diskriminatot
8     beta = 0.5 #
9     batch_size = 1 #menentukan ukuran kumpulan gambar
10    z_size = 200 #
11    epochs = 1 #menetukkan masa periode. setiap epoch memiliki semua
12    batch
13    MODE = "train" #membuat traon mode

```

**Listing 8.25** Kode program run.py

### **8.3.2.13 Jelaskan secara detil dari kode program pembuatan dan kompilasi arsitektur berikut :**

```

1     gen_optimizer = Adam(lr=gen_learning_rate, beta_1=beta)
2     dis_optimizer = Adam(lr=dis_learning_rate, beta_1=beta)
3
4     discriminator = build_discriminator()
5     discriminator.compile(loss='binary_crossentropy', optimizer=
6         dis_optimizer)
7
8     generator = build_generator()
9     generator.compile(loss='binary_crossentropy', optimizer=
10        gen_optimizer)

```

**Listing 8.26** Kode program run.py

penjelasanya ialah membuat model dari generator model dan diskriminatormodel,

#### **8.3.2.14 Jelaskan secara detil kode program untuk membuat dan melakukan kompilasi model adversarial berikut:**

```

1 # In[ soal 14]
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=[validity])
8     adversarial_model.compile(loss='binary_crossentropy', optimizer=
9         gen_optimizer)

```

**Listing 8.27** Kode program run.py

penjelasanya ialah membuat model diskriminatormodel tidak di training dan membuat model adversarial.

#### **8.3.2.15 Jelaskan Ekstrak dan load data kursi dengan menggunakan fungsi getVoxelsFormat dan get3DImages yang digunakan pada kode program berikut :**

```

1 # In[ soal 15]
2     print("Loading data...")
3     volumes = get3DImages(data_dir=data_dir)
4     volumes = volumes [..., np.newaxis].astype(np.float)
5     print("Data loaded...")

```

**Listing 8.28** Kode program run.py

penjelasanya ialah melakukan print loading data, membuat variabel volumes untuk mendapatkan data gamabr 3D.

#### **8.3.2.16 Jelaskan maksud dari kode program instansiasi TensorBoard yang menambahkan generator dan diskriminatormodel pada program berikut:**

```

1 # In[ soal_16]
2     tensorboard = TensorBoard(log_dir="logs/{}".format(time.time()))
3     tensorboard.set_model(generator)
4     tensorboard.set_model(discriminator)

```

**Listing 8.29** Kode program run.py

penjelasanya ialah membuat variabel tensorboard untuk melakukan pencatatan log, dan men-set model generator dan model diskriminatot

### **8.3.2.17 Jelaskan apa fungsi dari np reshape ones zeros pada kode program berikut dengan parameternya:**

Penjelasan kode:

```

1 # In[ soal_17]
2     labels_real = np.reshape(np.ones((batch_size,)), (-1, 1, 1, 1, 1))
3     labels_fake = np.reshape(np.zeros((batch_size,)), (-1, 1, 1, 1, 1))

```

**Listing 8.30** Kode program run.py

membuat label\_real untuk menetapkan bahwa jika output 1 itu menunjukan gambar real, dan label\_fake untuk menunjukan bahwa output 0 untuk data fake/palsu.

### **8.3.2.18 Jelaskan kenapa harus ada perulangan dalam meraih epoch. Dan jelaskan apa itu epoch terkait kode program berikut:**

```

1 # In[ soal_18]
2     if MODE == 'train':
3         for epoch in range(epochs):
4             print("Epoch:", epoch)
5
6             gen_losses = []
7             dis_losses = []

```

**Listing 8.31** Kode program run.py

melakukan perulangan jika MODE = train maka akan diulangi terus menerus sampai epoch(periode) terpenuhi dengan disertai keterangan generator dan diskriminatot losses.

### **8.3.2.19 Jelaskan apa itu batches dan kaitannya dengan kode program berikut, dan kenapa berada di dalam epoch:**

```

1 # In[ soal_19]
2     number_of_batches = int(volumes.shape[0] / batch_size)
3     print("Number of batches:", number_of_batches)
4     for index in range(number_of_batches):
5         print("Batch:", index + 1)

```

**Listing 8.32** Kode program run.py

menghitung jumlah batch yang telah ditrain

**8.3.2.20 Berikut adalah kode program pengambilan gambar dan noise. Jelaskan apa fungsi `np.random.normal` serta `astype`, serta jelaskan apa arti parameter titik dua dan jelaskan isi dari `z sample` dan `volumes batch`:**

```

1 # In[ soal 20]
2             z_sample = np.random.normal(0, 0.33, size=[batch_size
3 , 1, 1, 1, z_size]).astype(np.float32)
4             volumes_batch = volumes[index * batch_size:(index +
5 1) * batch_size, :, :, :]

```

**Listing 8.33** Kode program run.py

untuk membersihkan gambar dari noise dan menyesuaikan shape.

**8.3.2.21 Berikut adalah kode program generator gambar palsu. Jelaskan apa fungsi `generator.predict_on_batch`, serta jelaskan apa arti parameter `z sample`:**

```

1 # In[ soal 21]
2             # Next, generate volumes using the generate network
3             gen_volumes = generator.predict_on_batch(z_sample)

```

**Listing 8.34** Kode program run.py

untuk meng-generate volume menggunakan model jaringan generator.

**8.3.2.22 Berikut adalah kode program training diskriminator dengan gambar palsu dari generator dan gambar asli. Jelaskan apa maksudnya harus dilakukan training diskriminator secara demikian dan jelaskan apa isi `loss fake` dan `loss real` serta `d loss` dan fungsi `train on batch`.**

```

1 # In[ soal 22]
2             discriminator.trainable = True
3             if index % 2 == 0:
4                 loss_real = discriminator.train_on_batch(
5                     volumes_batch, labels_real)
6                 loss_fake = discriminator.train_on_batch(
7                     gen_volumes, labels_fake)
8
9                 d_loss = 0.5 * np.add(loss_real, loss_fake)
10                print("d_loss:{:.4f}".format(d_loss))
11
12            else:
13                d_loss = 0.0

```

**Listing 8.35** Kode program run.py

untuk membuat diskriminator bisa meload gambar fake dan real dari model generator. dengan disertai generatot dan diskriminatot loss, agar terlihat seberapa baik kualitas yang dihasilkan.

**8.3.2.23 Berikut adalah kode program training model adversarial yang terdapat generator dan diskriminatot. Jelaskan apa bagaimana proses terbentuknya**

**parameter z dan g loss:**

```

1 # In[ soal_23]
2         discriminator.trainable = False
3         """
4             Train the generator network
5             """
6         z = np.random.normal(0, 0.33, size=[batch_size, 1, 1,
7             1, z_size]).astype(np.float32)
8         g_loss = adversarial_model.train_on_batch(z,
9             labels_real)
10        print("g_loss:{:.4f}".format(g_loss))
11        gen_losses.append(g_loss)
12        dis_losses.append(d_loss)

```

**Listing 8.36** Kode program run.py

untuk mentrain model generator dan variabel g\_loss untuk melakukan perbandingan antara label gambar yang asli.

**8.3.2.24 Berikut adalah kode program generate dan menyimpan gambar 3D setelah beberapa saat setiap epoch. Jelaskan mengapa ada perulangan dengan parameter tersebut, serta jelaskan arti setiap variabel beserta perlihatkan isinya dan artikan isinya :**

```

1 # In[ soal_24]
2         # Every 10th mini-batch, generate volumes and save
3         them
4         if index % 10 == 0:
5             z_sample2 = np.random.normal(0, 0.33, size=[batch_size, 1, 1, 1, z_size]).astype(np.float32)
6             generated_volumes = generator.predict(z_sample2,
7                 verbose=3)
8             for i, generated_volume in enumerate(
9                 generated_volumes[:5]):
10                voxels = np.squeeze(generated_volume)
11                voxels[voxels < 0.5] = 0.
12                voxels[voxels >= 0.5] = 1.
13                saveFromVoxels(voxels, "results/img_{:}-{:}-{:}".
14 .format(epoch, index, i))

```

**Listing 8.37** Kode program run.py

untuk melakukan perulangan dimana setiap terdapat 10 mini-batch akan meng-generate volumenya dan akan menyimpannya.

**8.3.2.25 Berikut adalah kode program menyimpan average losses setiap epoch. Jelaskan apa itu tensorboard dan setiap parameter yang digunakan pada kode program ini :**

```

1 # In[ soal_25]

```

```

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)

```

**Listing 8.38** Kode program run.py

untuk menuliskan log losses ke Tensorboard

**8.3.2.26 Berikut adalah kode program menyimpan model. Jelaskan apa itu format h5 dan penjelasan dari kode program berikut :**

```

1 # In[ soal_26]
2     generator.save_weights(os.path.join("models", "generator_weights.h5"))
3     discriminator.save_weights(os.path.join("models", "discriminator_weights.h5"))

```

**Listing 8.39** Kode program run.py

untuk menyimpan berat model generator dan model diskriminasi kedalam file h5.

H5 adalah Hierarchical Data Format 5 File. File H5 adalah file data yang disimpan dalam Format Data Hirarki (HDF). Ini berisi array multidimensi data ilmiah. File H5 biasanya digunakan di luar angkasa, fisika, teknik, keuangan, penelitian akademis, genomik, astronomi, instrumen elektronik, dan bidang medis.

**8.3.2.27 Berikut adalah kode program testing model. Jelaskan dengan ilustrasi gambar dari mulai meload hingga membuat gambar 3D dengan menggunakan z sample, bisakah parameter z sample tersebut diubah? :**

```

1     generator.load_weights(os.path.join("models", "generator_weights.h5"), True)
2     discriminator.load_weights(os.path.join("models", "discriminator_weights.h5"), True)
3
4     # Generate 3D models
5     z_sample = np.random.normal(0, 1, size=[batch_size, 1, 1, 1,
6     z_size]).astype(np.float32)
7     generated_volumes = generator.predict(z_sample, verbose=3)
8
9     for i, generated_volume in enumerate(generated_volumes[:2]):
10        voxels = np.squeeze(generated_volume)
11        voxels[voxels < 0.5] = 0.
12        voxels[voxels >= 0.5] = 1.
13        saveFromVoxels(voxels, "results/gen_{}".format(i))

```

**Listing 8.40** Kode program run.py

untuk membuat mode predict/prediksi. dimana dia melakukan pembuatan model generator, model diskriminasi dan meload data h5 yang sudah dibuat tadi dan menggenerate nya kedalam model 3D, lalu menyimpan model voxel nya ke folde results.

### 8.3.3 Penanganan Error

#### 1. SS Error

```
FileNotFoundError: [Errno 2] No such file or directory: '200aplets/volumetric_data/guitar/20/test/
guitar_000000001_1.mot'
```

**Gambar 8.60** File Not Found error

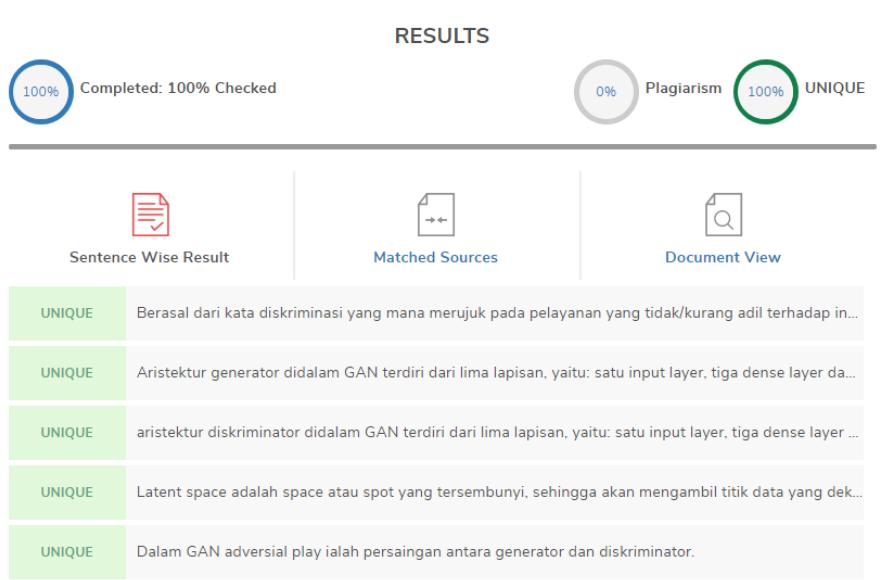
#### 2. Jenis Error

- File Not Found Error

#### 3. Cara Penanganan

Dengan cara menyesuaikan letak file yang ingin dibaca / digunakan

### 8.3.4 Bukti Tidak Plagiat



**Gambar 8.61** Bukti tidak plagiat

### 8.3.5 Link Youtube

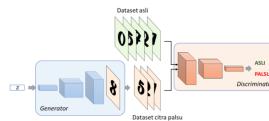
<https://youtu.be/hW96rJM9VD8>

## 8.4 1174069 - Fanny Shafira Damayanti

### 8.4.1 Soal Teori

1. Jelaskan dengan ilustrasi gambar sendiri apa itu generator dengan perumpamaan anda sebagai mahasiswa sebagai generatornya.

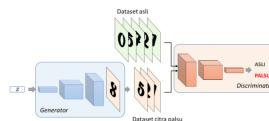
Tugas Generator sekarang sedang dibuat untuk membuat koleksi gambar palsu, yang saat ini dilihat oleh Diskriminator. Diskriminator tidak dapat membedakan antara yang asli dan yang palsu. Untuk ilustrasi, lihat gambar berikut:



**Gambar 8.62** Teori 1

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

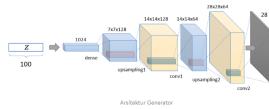
Diskriminator adalah CNN yang menerima input gambar yang dimiliki dan menghasilkan angka biner yang meminta input gambar, lalu menghasilkan gambar dari dataset asli atau menghasilkan gambar palsu. Untuk ilustrasi, lihat gambar berikut:



**Gambar 8.63** Teori 2

3. Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur generator dibuat.

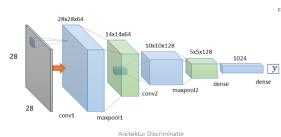
Aksitektur generator dibuat bisa dijelaskan pada gambar berikut :



**Gambar 8.64** Teori 3

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

Aksitektur diskriminatator dibuat bisa dijelaskan pada gambar berikut :



**Gambar 8.65** Teori 4

5. Jelaskan dengan ilustrasi gambar apa itu latent space.

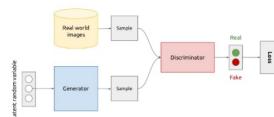
Latent space dijelaskan pada gambar berikut :



**Gambar 8.66** Teori 5

6. Jelaskan dengan ilustrasi gambar apa itu adversarial play.

Adversarial play dijelaskan pada gambar berikut :



**Gambar 8.67** 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 :

		PRISONER 2	
		Confess	Lie
PRISONER 1	Confess	-8, -8	0, -10
	Lie	-10, 0	-1, -1

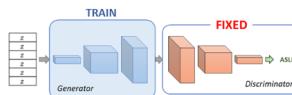
**Gambar 8.68** Teori 7

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 perhatikan 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).

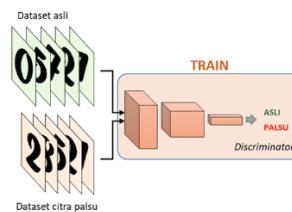
Untuk penjelasan tersebut dijelaskan pada gambar dibawah ini :



**Gambar 8.69** Teori 9

10. Berikan contoh dengan ilustrasi dari arsitektur dikriminatator dengan sebuah input layer, 3 buah hidden layer, dan satu output layer.

Untuk penjelasan tersebut dijelaskan pada gambar dibawah ini :



**Gambar 8.70** Teori 10

11. Jelaskan bagaimana kaitan output dan input antara generator dan diskriminatator tersebut. Jelaskan kenapa inputan dan outputan seperti itu.

Gambar dari Generator yang berhasil di deteksi oleh Diskriminatator sebagai gambar fake, akan dikembalikan dengan feedback pke generator. Kini Generator bertugas untuk bisa membuat sekumpulan gambar palsu, yang nantinya dapat dilihat oleh Diskriminatator, lalu, Diskriminatator 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.

#### 8.4.2 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 pendekripsi 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 Create a Generator Model with hyperparameters values defined  
2 as follows  
3     """  
4     z_size = 200  
5     gen_filters = [512, 256, 128, 64, 1]
```

```

5   gen_kernel_sizes = [4, 4, 4, 4, 4]
6   gen_strides = [1, 2, 2, 2, 2]
7   gen_input_shape = (1, 1, 1, z_size)
8   gen_activations = ['relu', 'relu', 'relu', 'relu', 'sigmoid']
9   gen_convolutional_blocks = 5
10
11  input_layer = Input(shape=gen_input_shape)
12
13  # First 3D transpose convolution(or 3D deconvolution) block
14  a = Deconv3D(filters=gen_filters[0],
15                kernel_size=gen_kernel_sizes[0],
16                strides=gen_strides[0])(input_layer)
17  a = BatchNormalization()(a, training=True)
18  a = Activation(activation='relu')(a)
19
20  # Next 4 3D transpose convolution(or 3D deconvolution) blocks
21  for i in range(gen_convolutional_blocks - 1):
22      a = Deconv3D(filters=gen_filters[i + 1],
23                    kernel_size=gen_kernel_sizes[i + 1],
24                    strides=gen_strides[i + 1], padding='same')(a)
25      a = BatchNormalization()(a, training=True)
26      a = Activation(activation=gen_activations[i + 1])(a)
27
28  gen_model = Model(inputs=[input_layer], outputs=[a])
29  return gen_model
30
31 #%% soal10

```

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

```

```

19     strides=dis_strides[0],
20     padding=dis_paddings[0])(dis_input_layer)
21 # a = BatchNormalization()(a, training=True)
22 a = LeakyReLU(dis_alphas[0])(a)
23
24 # Next 4 3D Convolutional Blocks
25 for i in range(dis_convolutional_blocks - 1):
26     a = Conv3D(filters=dis_filters[i + 1],
27                 kernel_size=dis_kernel_sizes[i + 1],
28                 strides=dis_strides[i + 1],
29                 padding=dis_paddings[i + 1])(a)
30     a = BatchNormalization()(a, training=True)
31     if dis_activations[i + 1] == 'leaky_relu':
32         a = LeakyReLU(dis_alphas[i + 1])(a)
33     elif dis_activations[i + 1] == 'sigmoid':
34         a = Activation(activation='sigmoid')(a)
35
36 dis_model = Model(inputs=[dis_input_layer], outputs=[a])
37 return dis_model
38
39 #%%%

```

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

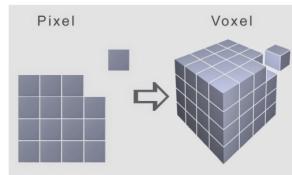
#### 6. Soal 6

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 8.71** Praktek Soal 7

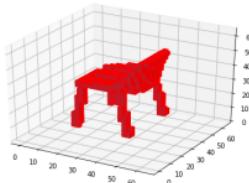
## 8. Soal 8

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun May 17 14:56:53 2020
4
5 @author: FannyShafira
6 """
7
8 # In []
9
10 import scipy.io as io
11 voxels = io.loadmat("data/3DShapeNets/volumetric_data/chair/30/
12     test/chair_000000000_1.mat")['instance']
13 #%%
14 import numpy as np
15 voxels = np.pad(voxels, (1, 1), 'constant', constant_values=(0,
16     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 plt.show()
27 plt.savefig('data')

```

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



Gambar 8.72 Hasil Soal 8

## 9. Soal 9

```

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')(a)
28        a = BatchNormalization()(a, training=True)
29        a = Activation(activation=gen_activations[i + 1])(a)
30
31    gen_model = Model(inputs=[input_layer], outputs=[a])
32
33    #%% soal10

```

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

## 10. Soal 10

```

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    dis_convolutional_blocks = 5
16
17    dis_input_layer = Input(shape=dis_input_shape)
18
19    # The first 3D Convolutional block
20    a = Conv3D(filters=dis_filters[0],
21                kernel_size=dis_kernel_sizes[0],
22                strides=dis_strides[0],
23                padding=dis_paddings[0])(dis_input_layer)
24    # a = BatchNormalization()(a, training=True)
25    a = LeakyReLU(dis_alphas[0])(a)
26
27    # Next 4 3D Convolutional Blocks
28    for i in range(dis_convolutional_blocks - 1):
29        a = Conv3D(filters=dis_filters[i + 1],
30                    kernel_size=dis_kernel_sizes[i + 1],
31                    strides=dis_strides[i + 1],
32                    padding=dis_paddings[i + 1])(a)
33        a = BatchNormalization()(a, training=True)
34        if dis_activations[i + 1] == 'leaky_relu':
35            a = LeakyReLU(dis_alphas[i + 1])(a)
36        elif dis_activations[i + 1] == 'sigmoid':
37            a = Activation(activation='sigmoid')(a)
38
39    dis_model = Model(inputs=[dis_input_layer], outputs=[a])
40    return dis_model
41
42 #%%

```

Kode di atas befungsi untuk membangun diskriminator berfungsi untuk mendefenisikan 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   data_dir = "data/3DShapeNets/volumetric_data/" \
2       "*/30/train/*.mat".format(object_name)
3   gen_learning_rate = 0.0025
4   dis_learning_rate = 1e-5
5   beta = 0.5
6   batch_size = 1
7   z_size = 200
8   epochs = 1
9   MODE = "train"
10
11 #%% soal 13

```

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

#### 13. Soal 13

```

1 Create models
2 """
3     gen_optimizer = Adam(lr=gen_learning_rate, beta_1=beta)
4     dis_optimizer = Adam(lr=dis_learning_rate, beta_1=beta)
5
6     discriminator = build_discriminator()
7     discriminator.compile(loss='binary_crossentropy', optimizer=
8         dis_optimizer)
9
10    generator = build_generator()
11    generator.compile(loss='binary_crossentropy', optimizer=
12        gen_optimizer)
13
14 #%% soal14

```

Kode di atas menggunakan Adam sebagai algoritma pengoptimalan dan binary\_crossentropy sebagai kerugian loss.

#### 14. Soal 14

```

1
2     input_layer = Input(shape=(1, 1, 1, z_size))
3     generated_volumes = generator(input_layer)
4     validity = discriminator(generated_volumes)
5     adversarial_model = Model(inputs=[input_layer], outputs=[validity])
6     adversarial_model.compile(loss='binary_crossentropy',
7         optimizer=gen_optimizer)
8
9 #%% soal15

```

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

#### 15. Soal 15

```

1     volumes = get3DImages( data_dir=data_dir )
2     volumes = volumes[... , np.newaxis].astype(np.float)
3     print("Data loaded...")
4
5 %% soal16

```

Kode di atas befungsi untuk melakukan load data pada dataset.

#### 16. Soal 16

```

1     tensorboard.set_model(generator)
2     tensorboard.set_model(discriminator)
3
4 %% soal17

```

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

#### 17. Soal 17

```

1     labels_fake = np.reshape(np.zeros((batch_size,)), (-1, 1, 1,
2     1))
3
4 %% soal18

```

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

#### 18. Soal 18

```

1         for epoch in range(epochs):
2             print("Epoch:", epoch)
3
4             gen_losses = []
5             dis_losses = []
6
7 %% soal19

```

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

#### 19. Soal 19

```

1     print("Number of batches:", number_of_batches)
2         for index in range(number_of_batches):
3             print("Batch:", index + 1)
4
5 #%% soal20

```

Batch adalah jumlah file yang akan di training.

## 20. Soal 20

```

1     volumes_batch = volumes[index * batch_size:(index
2         + 1) * batch_size, :, :, :]
3
3 #%% soal21

```

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

## 21. Soal 21

```

1
2
3
1     gen_volumes = generator.predict_on_batch(z_sample
2         )
3
3     .....

```

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

## 22. Soal 22

```

1
2
3
4
5
6
7
8
9
10
11
1     if index % 2 == 0:
2         loss_real = discriminator.train_on_batch(
3             volumes_batch, labels_real)
4             loss_fake = discriminator.train_on_batch(
5                 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
11 else:
12     d_loss = 0.0
11 #%% soal23

```

Kode di atas befungsi 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      """
2          Train the generator network
3      """
4          z = np.random.normal(0, 0.33, size=[batch_size,
5              1, 1, 1, z_size]).astype(np.float32)
6          g_loss = adversarial_model.train_on_batch(z,
7              labels_real)
8          print("g_loss:{}".format(g_loss))
9
10         gen_losses.append(g_loss)
11         dis_losses.append(d_loss)
12
13 #%% soal24

```

Kode di atas befungsi untuk melakukan print gloss untuk generator dan juga dloss untuk diskriminatator.

#### 24. Soal 24

```

1      if index % 10 == 0:
2          z_sample2 = np.random.normal(0, 0.33, size=[batch_size,
3              1, 1, 1, z_size]).astype(np.float32)
4          generated_volumes = generator.predict(
5              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.
11             saveFromVoxels(voxels, "results/img_{}-{}"
12             .format(epoch, index, i))
13
14 #%% soal25

```

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

#### 25. Soal 25

```

1      write_log(tensorboard, 'g_loss', np.mean(gen_losses),
2      epoch)
3      write_log(tensorboard, 'd_loss', np.mean(dis_losses),
4      epoch)
5      """

```

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

#### 26. Soal 26

```

1     discriminator.save_weights(os.path.join("models", " "
2         "discriminator_weights.h5"))
3 #%% soal27

```

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

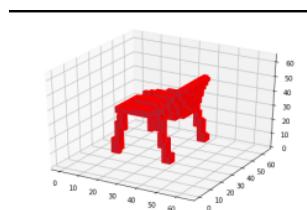
## 27. Soal 27

```

1 # Create models
2 generator = build_generator()
3 discriminator = build_discriminator()
4
5 # Load model weights
6 generator.load_weights(os.path.join("models", " "
7     "generator_weights.h5"), True)
8 discriminator.load_weights(os.path.join("models", " "
9     "discriminator_weights.h5"), True)
10
11 # Generate 3D models
12 z_sample = np.random.normal(0, 1, size=[batch_size, 1, 1,
13     1, z_size]).astype(np.float32)
14 generated_volumes = generator.predict(z_sample, verbose
15     =3)
16
17 for i, generated_volume in enumerate(generated_volumes
18     [:2]):
19     voxels = np.squeeze(generated_volume)
20     voxels[voxels < 0.5] = 0.
21     voxels[voxels >= 0.5] = 1.
22     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 diskriminat. Untuk ilustrasi gambar sebagai berikut :



**Gambar 8.73** Praktek Soal 27

### 8.4.3 Penanganan Error

#### 1. ValueError

```

main(filename, samplesize):
    file = open(filename, "r")
    lines = file.readlines()
    file.close()
    samplesize = int(samplesize)
    samples = []
    for i in range(samplesize):
        sample = random.choice(lines)
        samples.append(sample)
    samples = np.random.choice(samples, size=samples)
    file = open("samples.txt", "w")
    file.write(samples)
    file.close()
    print("Samples have been saved to samples.txt")
    print("File has been created at", os.getcwd())
    os.remove(filename)

```

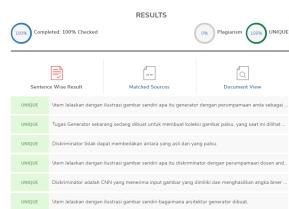
**Gambar 8.74** ValueError

## 2. Cara Penanganan Error

- **ValueError**

Error tersebut karena disebabkan gagal load dataset karena salah penamaan direktori.

### 8.4.4 Bukti Tidak Plagiat



**Gambar 8.75** Bukti Tidak Melakukan Plagiat Chapter 8

## 8.5 1174086 { Tia Nur Candida

### 8.5.1 Teori

1. Jelaskan dengan ilustrasi gambar sendiri apa itu generator dengan perumpamaan anda sebagai mahasiswa sebagai generatornya.

Generator adalah sebuah jaringan yang merubah inputan vektor menjadi gambar, seperti ada inputan vektor secara acak yaitu angka sembarang, maka angka tersebut akan diubah menjadi gambar yang sembarang pula. Sebagai ilustrasi apabila mahasiswa sebagai generator, misalkan inputan vektor tersebut adalah bahan-bahan membuat kue, maka hasil dari generator tersebut juga akan acak, bisa kue bolu, cupcake, ataupun kue lainnya.



**Gambar 8.76** Teori No 1

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

Diskriminator adalah sebuah jaringan yang mengeluarkan klasifikasi untuk menyatakan input gambar adalah asli dari dataset atau buatan dari generator. Lebih mudahnya diskriminator digunakan agar hasil gambar dari generator sesuai dengan yang diinginkan. sebagai ilustrasi nya dosen menyuruh mahasiswanya membuat kue bolu, maka hasil yang akan dibuat adalah kue bolu.



**Gambar 8.77** Teori No 2

3. Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur generator dibuat  
Generator menggunakan input array random yang bernama seed, dimana akan diubah menjadi sebuah gambar dengan menggunakan Convolutional Neural Network yang dapat dilihat pada gambar.



**Gambar 8.78** Teori No 3

4. Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur diskriminatator dibuat

Diskriminator adalah CNN yang menerima inputan image lalu menghasilkan angka biner yang dapat menyatakan apakah gambar tersebut asli atau sesuai dengan dataset asli



**Gambar 8.79** Teori No 4

5. Jelaskan dengan ilustrasi gambar apa itu latent space

latent space adalah representasi dari data yang di kompress, untuk contohnya misalnya ada 3 gambar yaitu 2 kursi yang berbeda dan 1 meja, maka hal hal yang mirip dai kursi tersebut(ciri-ciri utamanya), seeperti itulah latent space



**Gambar 8.80** Teori No 5

6. apa itu adversarial play

adversarial play adalah dimana para jaringan di latih, dimana jaringan satu dan lainnya saling berkompetisi. dapat disimpulkan dimana jaringan generator dan jaringan discriminator saling bertemu berulang ulang kali

7. Jelaskan dengan ilustrasi gambar apa itu Nash equilibrium

Nash equilibrium adalah konsep dalam teori permainan di mana hasil optimal dari permainan adalah di mana tidak ada insentif untuk menyimpang dari strategi awal mereka. Lebih khusus lagi, keseimbangan Nash adalah konsep teori permainan di mana hasil optimal dari permainan adalah di mana tidak ada pemain yang memiliki insentif untuk menyimpang dari strategi yang dipilihnya setelah mempertimbangkan pilihan lawan.

	A	B
A	1,1	1,-1
B	-1,1	0,0

**Gambar 8.81** Teori No 7

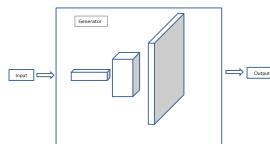
8. Sebutkan dan jelaskan contoh-contoh implementasi dari GAN

Pada bidang mode, seni, dan iklan, GAN dapat digunakan untuk membuat foto-foto model fashion imajiner tanpa perlu menyewa model. Pada bidang sains GAN dapat meningkatkan citra astronomi dan mensimulasikan pelensa gravitasi untuk penelitian materi gelap.

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)

```
gen=Sequential() #Inisiasi dari sequensial
gen.add(Dense(units=200, input_dim=np.shape(train_input)[1])
gen.add(Dense(units=400))#Menambah dense layer dengan batch
gen.add(Dense(units=784, activation='tanh')) #Menambah dense
gen.compile(loss='binary_crossentropy', optimizer=adam\opt
gen.summary() #Memproses data yang sudah disetting dan mena
```

Pada contoh tersebut, data akan diambil dari hasil proses sebelumnya yaitu proses ekstrasi data gambar. Dari contoh ini, terdapat 3 layer dense, 1 input layer dan 1 output layer. Dari input layer nanti akan dimasukkan terlebih dahulu ke dense layer pertama lalu diproses oleh 2 layer selanjutnya dan terakhir akan ditampilkan oleh layer output.

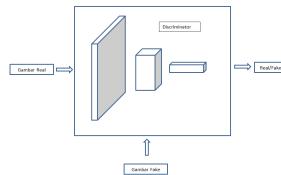
**Gambar 8.82** Teori No 9

10. Berikan contoh dengan ilustrasi dari arsitektur dikriminatator dengan sebuah input layer, 3 buah hidden layer, dan satu output layer.

```

diskrim=Sequential() #Inisiasi dari sequensial
diskrim.add(Dense(units=784, input_dim=np.shape(train_input)
diskrim.add(Dense(units=400)) #Mensetting Dense
diskrim.add(Dense(units=200, activation='sigmoid')) #Mensett
diskrim.compile(loss='binary_crossentropy', optimizer=adam)
diskrim.summary()#Memproses data yang sudah disetting dan me
  
```

Pada contoh tersebut, data akan diambil dari hasil proses sebelumnya yaitu proses ekstrasi data gambar. Dari contoh ini, terdapat 3 layer dense, 1 input layer dan 1 output layer. Pada proses ini, seluruh data akan dibandingkan dengan data sebelumnya yaitu dari generator dan dari data aslinya yang sudah dijadikan data vector.

**Gambar 8.83** Teori No 10

11. Jelaskan bagaimana kaitan output dan input antara generator dan diskriminatator tersebut. Jelaskan kenapa inputan dan outputan seperti itu

Pada kedua metode tersebut, akan disebutkan berapa akurasi dari setiap metode. Pada setiap metode tersebut (Discriminator dan generator) akan dilakukan pelatihan dan akan dibandingkan hasilnya. Generator akan menghasilkan data baru sesuai dengan hasil latihan dan dari data tersebut, discriminator akan membandingkan dengan data set apakah data tersebut "asli" atau tidak.

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

relative entropy adalah ukuran dari bagaimana satu distribusi probabilitas berbeda dari yang kedua, distribusi probabilitas referensi, Divergensi Jensen-Shannon adalah ukuran divergensi berprinsip yang selalu terbatas untuk variabel acak terbatas.

13. Jelaskan apa itu fungsi objektif yang berfungsi untuk mengukur kesamaan antara gambar yang dibuat dengan yang asli.

Fungsi objektif adalah fungsi yang digunakan sebagai penunjuk berapa nilai kesamaan antara gambar yang dibuat dengan yang asli

14. Jelaskan apa itu scoring algoritma selain mean square error atau cross entropy seperti The Inception Score dan The Frechet Inception distance.

Inception Score digunakan untuk mengukur seberapa realistik output dari GAN, dimana ada dua parameter, yaitu : gambarnya punya variasi dan setiap gambar jelas terlihat seperti sesuatu. Frechet Inception Distance adalah ukuran kesamaan antara dua dataset gambar. Itu terbukti berkorelasi baik dengan penilaian manusia terhadap kualitas visual dan paling sering digunakan untuk mengevaluasi kualitas sampel Generative Adversarial Networks. FID dihitung dengan menghitung jarak Fréchet antara dua Gaussians dipasang ke representasi fitur dari jaringan Inception.

15. Jelaskan kelebihan dan kekurangan GAN

- Kelebihan

- (a) GAN Menghasilkan data baru yang bisa hampir mirip dengan data asli. Karena hasil pelatihannya, GAN dapat menghasilkan data gambar, teks, audio, dan video yang dapat dibilang hampir mirip dengan yang aslinya. Berkat hal tersebut, GAN dapat digunakan dalam sistem marketing, e-commerce, games, iklan, dan industri lainnya
- (b) GAN mempelajari representasi data secara internal sehingga beberapa masalah pada machine learning dapat diatasi dengan mudah
- (c) Discriminator yang sudah dilatih dapat menjadi sebuah classifier atau pendekripsi jika data sudah sesuai. Karena Discriminator yang akan menjadi tidak efisien berkat seringnya dilatih
- (d) GAN dapat dilatih menggunakan data yang belum dilabeled

- Kekurangan

- (a) Data saat diproses oleh metode gan tidak konvergensi
- (b) Jenis sampel yang dihasilkan oleh generator terbatas karena modanya terbatas
- (c) Ketidak seimbangnya antara generator dan discriminator dapat menyebabkan overfitting atau terlalu dekat dengan hasil sampel
- (d) Sangat sensitif dengan data yang sudah diinisiasi sebelumnya

### 8.5.2 Praktek

1. Jelaskan apa itu 3D convolutions Konvolusi 3D. Konvolusi 3D menerapkan filter 3 dimensi ke dataset dan filter bergerak 3 arah (x, y, z) untuk menghitung representasi fitur level rendah. Bentuk output mereka adalah ruang volume 3 dimensi seperti kubus atau kuboid.
2. Jelaskan dengan kode program arsitektur dari generator networknya, beserta penjelasan input dan output dari generator network.

```

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    dis_convolutional_blocks = 5
16
17    dis_input_layer = Input(shape=dis_input_shape)
18
19    # The first 3D Convolutional block
20    a = Conv3D(filters=dis_filters[0],
21                kernel_size=dis_kernel_sizes[0],
22                strides=dis_strides[0],
23                padding=dis_paddings[0])(dis_input_layer)
24    # a = BatchNormalization()(a, training=True)
25    a = LeakyReLU(dis_alphas[0])(a)
26
27    # Next 4 3D Convolutional Blocks
28    for i in range(dis_convolutional_blocks - 1):
29        a = Conv3D(filters=dis_filters[i + 1],
30                    kernel_size=dis_kernel_sizes[i + 1],
31                    strides=dis_strides[i + 1],
32                    padding=dis_paddings[i + 1])(a)
33        a = BatchNormalization()(a, training=True)
34        if dis_activations[i + 1] == 'leaky_relu':
35            a = LeakyReLU(dis_alphas[i + 1])(a)
36        elif dis_activations[i + 1] == 'sigmoid':
37            a = Activation(activation='sigmoid')(a)
38
39    dis_model = Model(inputs=[dis_input_layer], outputs=[a])
40    return dis_model

```

generator ialah `g_loss`, Bentuk jaringan Generator dapat dilihat berkebalikan dengan struktur jaringan saraf pada umumnya. Jaringan Generator menerima input sebuah vektor angka `z`, kemudian mengubahnya menjadi output gambar tiga dimensi.

3. Jelaskan dengan kode program arsitektur dari diskriminator network, beserta penjelasan input dan outputnya.

```

1 def build_generator():
2     """
3         Create a Generator Model with hyperparameters values defined
4             as follows
5         """
6
7     z_size = 200
8     gen_filters = [512, 256, 128, 64, 1]
9     gen_kernel_sizes = [4, 4, 4, 4, 4]
10    gen_strides = [1, 2, 2, 2, 2]
11    gen_input_shape = (1, 1, 1, z_size)
12    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')(a)
29
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

```

diskriminato 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.

4. Jelaskan proses training 3D-GANs proses training 3D gan yaitu dengan melakukan epoch sebanyak yang ditentukan.
5. Jelaskan bagaimana melakukan settingan awal chapter 02 untuk memenuhi semua kebutuhan sebelum melanjutkan ke tahapan persiapan data. 1. clone github  
2. download dataset 3. buat folder baru logs dan results
6. Jelaskan tentang dataset yang digunakan, dari mulai tempat unduh, cara membuka dan melihat data. Sampai deskripsi dari isi dataset dengan detail penjelasan setiap folder/file yang membuat orang awam paham. dataset 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. Jelaskan apa itu voxel dengan ilustrasi dan bahasa paling awam 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
8. Visualisasikan dataset tersebut dalam tampilan visual plot, jelaskan cara melakukan visualisasinya 1. import library 2. load data file .mat 3. lalu read memakai matplotlib
9. buka file run.py jelaskan perbaris kode pada fungsi untuk membuat generator yaitu build generator. membuat generator yaitu dengan ketentuan gen sebagai variabel dan membuat fungsi atau variabel gen\_model lalu dilakukan return
10. jelaskan juga fungsi untuk membangun diskriminator pada fungsi build discriminator. membangun diskriminator berfungsi untuk mendefenisikan seluruh gambar yang sudah di load generator sebagai gambar fake dan real
11. Jelaskan kode program

```
1 if __name__ == '__main__':
```

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

12. Jelaskan kode program

```
1 """
2     Specify Hyperparameters
3 """
4
5     object_name = "chair"
6     data_dir = "3DShapeNets/volumetric_data/" \
7                 "{}/train/*.mat".format(object_name)
8     gen_learning_rate = 0.0025
9     dis_learning_rate = 10e-5
10    beta = 0.5
11    batch_size = 1
12    z_size = 200
13    epochs = 10
14    MODE = "train"
```

artinya adalah load dataset yang hanya dalam folder chair data train

13. Jelaskan kode program

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

```

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

```

disini menggunakan Adam sebagai algoritma pengoptimalan dan binary\_crossentropy sebagai kerugian loss.

#### 14. Jelaskan kode program

```

1     input_layer = Input(shape=(1, 1, 1, z_size))
2     generated_volumes = generator(input_layer)
3     validity = discriminator(generated_volumes)
4     adversarial_model = Model(inputs=[input_layer], outputs=[validity])
5     adversarial_model.compile(loss='binary_crossentropy',
       optimizer=gen_optimizer)

```

ini 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. Jelaskan kode program

```

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

```

ini melakukan load data pada dataset

#### 16. Jelaskan kode program

```

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

```

ini berfungsi untuk membuat tensorboard yang nantinya bisa diakses melalui localhost

#### 17. Jelaskan kode program

```

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

```

fungsi ini ialah untuk melakukan reshape agar shape yang dihasilkan tidak terlalu besar.

#### 18. Jelaskan kode program

```

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

```

karena jika epoch semakin banyak maka kualitas training yang dihasilkan akan semakin baik

#### 19. Jelaskan kode program

```

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

```

batch adalah jumlah file yang akan di training

#### 20. Jelaskan kode program

```

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

```

ini adalah untuk membuat gambar bersih dari noise dan juga menyesuaikan shape

#### 21. Jelaskan kode program

```

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

```

ialah membuat sample gambar palsu yang akan diteruskan ke diskriminatore

#### 22. Jelaskan kode program

```

1     """
2     Train the discriminator network
3     """
4     discriminator.trainable = True
5     if index % 2 == 0:
6         loss_real = discriminator.train_on_batch(
7             volumes_batch, labels_real)
8         loss_fake = discriminator.train_on_batch(
9             gen_volumes, labels_fake)
10
11         d_loss = 0.5 * np.add(loss_real, loss_fake)
12         print("d_loss:{}".format(d_loss))
13

```

diskriminat bisa load gambar fake dan real dari generator, oleh karena itu ada generator loss dan diskriminat loss untuk melihat seberapa baik kualitas yang dihasilkan

#### 23. Jelaskan kode program

```

1      z = np.random.normal(0, 0.33, size=[batch_size,
2          1, 1, 1, z_size]).astype(np.float32)
3          g_loss = adversarial_model.train_on_batch(z,
4          labels_real)
5          print("g_loss:{}".format(g_loss))
6
7          gen_losses.append(g_loss)
8          dis_losses.append(d_loss)

```

dengan melakukan print g\_loss untuk generator dan juga d\_loss untuk diskriminat

#### 24. Jelaskan kode program

```

1      # Every 10th mini-batch, generate volumes and
2      save them
3      if index % 10 == 0:
4          z_sample2 = np.random.normal(0, 0.33, size=[batch_size,
5              1, 1, 1, z_size]).astype(np.float32)
6          generated_volumes = generator.predict(
7              z_sample2, verbose=3)
8          for i, generated_volume in enumerate(
9              generated_volumes[:5]):
10             voxels = np.squeeze(generated_volume)
11             voxels[voxels < 0.5] = 0.
12             voxels[voxels >= 0.5] = 1.
13             saveFromVoxels(voxels, "results/img_{}_{}".format(epoch, index, i))

```

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

#### 25. Jelaskan kode program

```

1      # Write losses to Tensorboard
2      write_log(tensorboard, 'g_loss', np.mean(gen_losses),
3      epoch)
4      write_log(tensorboard, 'd_loss', np.mean(dis_losses),
5      epoch)

```

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

#### 26. Jelaskan kode program

```

1      """
2      Save models
3      """
4      generator.save_weights(os.path.join("models",
5          "generator_weights.h5"))

```

```

5     discriminator.save_weights(os.path.join("models", "
6         discriminator_weights.h5"))

```

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

## 27. Jelaskan kode program

```

1 if MODE == 'predict':
2     # Create models
3     generator = build_generator()
4     discriminator = build_discriminator()
5
6     # Load model weights
7     generator.load_weights(os.path.join("models", "
8         generator_weights.h5"), True)
9     discriminator.load_weights(os.path.join("models", "
10        discriminator_weights.h5"), True)
11
12     # Generate 3D models
13     z_sample = np.random.normal(0, 1, size=[batch_size, 1, 1,
14     1, z_size]).astype(np.float32)
15     generated_volumes = generator.predict(z_sample, verbose
16     =3)
17
18     for i, generated_volume in enumerate(generated_volumes
19     [:2]):
20         voxels = np.squeeze(generated_volume)
21         voxels[voxels < 0.5] = 0.
22         voxels[voxels >= 0.5] = 1.
23         saveFromVoxels(voxels, "results/gen_{}".format(i))

```

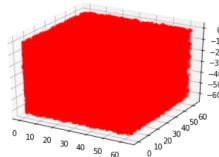
ini adalah tahap akhir untuk melakukan testing dari model yang telah dibuat dan buat model dari yang sudah di create sebelumnya yaitu generator dan diskriminato.

```

...           saveFromVoxels(voxels,
Loading data...
Data loaded...
Epoch: 0
Number of batches: 10
Batch: 1
d_loss:0.6931722164154053
g_loss:0.6931138038635254
Batch: 2
g_loss:0.6931138038635254
Batch: 3
d_loss:0.6931579113006592
g_loss:0.6931090354919434
Batch: 4
g_loss:0.6931090354919434
Batch: 5
d_loss:0.6931560039520264
g loss:0.693108081817627

```

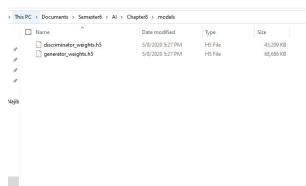
**Gambar 8.84** Result



**Gambar 8.85** Result



**Gambar 8.86** Result



**Gambar 8.87** Result

## 8.6 1174079 - Chandra Kirana Poetra

Chapter 8 - Perkenalan Generative Adversarial Network

### 8.6.1 Teori

**8.6.1.1 Jelaskan dengan ilustrasi gambar sendiri apa itu generator dengan perumpamaan anda sebagai mahasiswa sebagai generatornya.**

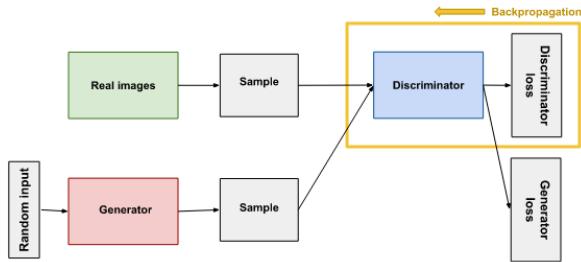


**Gambar 8.88** gambaran penjelasan no. 1



**Gambar 8.89** gambaran penjelasan no. 1

Pertama, mahasiswa akan mencoba untuk meniru data asli seperti gambar diatas yaitu uang dengan cara menggambarnya, sekilas memang terlihat sangat jelas perbedaan antara keduanya dan disini pihak discriminator yaitu dosen akan mampu untuk membedakan gambar yang digambar oleh mahasiswa dengan data asli, setelah itu mahasiswa akan mencoba berulang kali hingga data yang ditirunya mirip seperti pada gambar kedua, bila dilihat pada gambar kedua sangat mirip sekali antara satu sama lain sehingga pihak discriminator kesulitan membedakannya dan melabelinya sebagai data real.



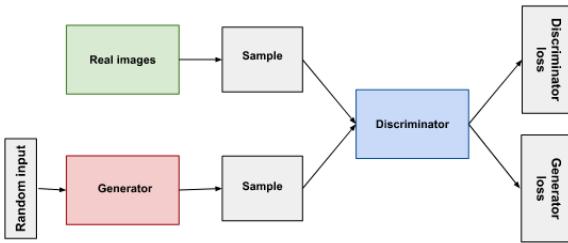
**Gambar 8.90** gambaran penjelasan no. 2

#### **8.6.1.2 Jelaskan dengan ilustrasi gambar sendiri apa itu diskriminator dengan perumpamaan dosen anda sebagai diskriminornya.**

Diskriminator sederhananya adalah suatu hal yang digunakan untuk melakukan klasifikasi yang membedakan data asli dengan data yang dihasilkan oleh generator, disini dosen bisa digambarkan sebagai seorang diskriminotor yang mengevaluasi tiap mahasiswanya, misalkan mahasiswa menggambar suatu tiruan dari data A, nah dosen akan mampu membedakan gambar yang asli dengan gambar yang dibuat oleh mahasiswa karena dosen sudah memiliki klasifikasinya tersendiri

#### **8.6.1.3 Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur generator dibuat**

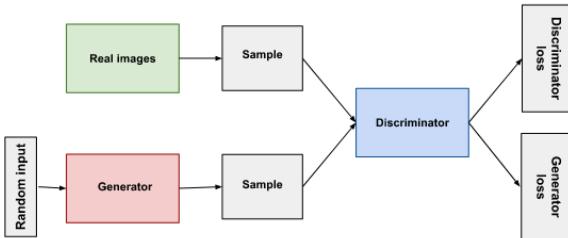
Arsitektur GAN, terdiri dari generator yang menghasilkan data sample dan juga ada data asli yang dijadikan sampel kemudian diinputkan ke discriminator yang kemudian akan melabeli data mana yang real dan tiruan



**Gambar 8.91** gambar arsitektur generator pada GAN-Project

#### 8.6.1.4 Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur generator dibuat

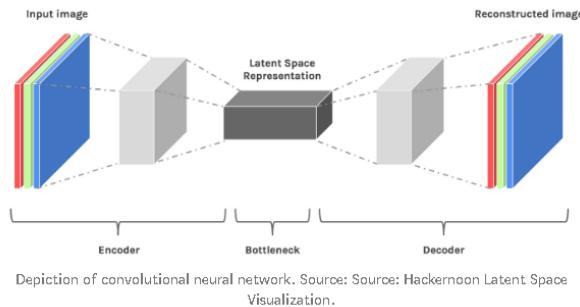
Arsitektur GAN, terdiri dari generator yang menghasilkan data sample dan juga ada data asli yang dijadikan sampel kemudian diinputkan ke discriminator yang kemudian akan melabeli data mana yang real dan tiruan



**Gambar 8.92** gambar arsitektur generator pada GAN-Project

#### 8.6.1.5 Jelaskan dengan ilustrasi gambar apa itu latent space.

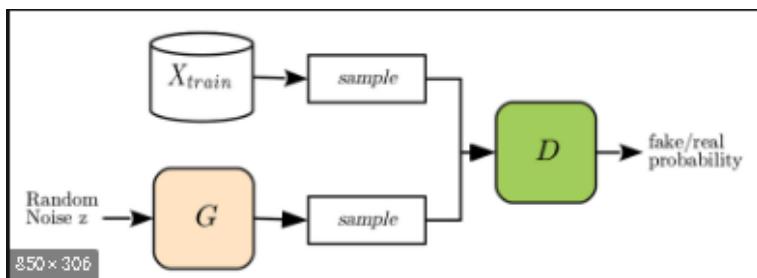
Latent space sederhananya adalah bentuk representasi dari data yang di compress, bisa dibilang di machine learning pun kita perlu melakukan optimisasi data dengan cara melakukan compress pada data yang akan kita gunakan agar sizenya tidak terlalu besar



**Gambar 8.93** gambaran penjelasan no. 5

#### 8.6.1.6 *Jelaskan dengan ilustrasi gambar apa itu adversarial play*

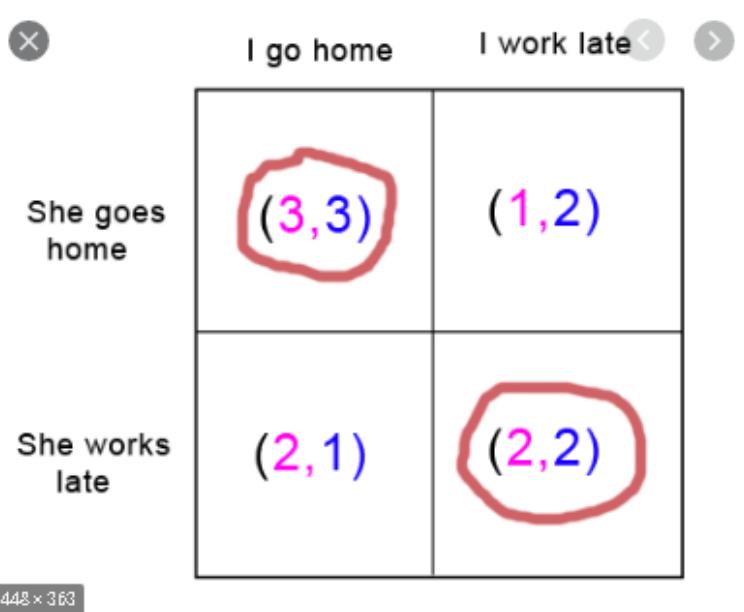
Merupakan proses yang terjadi antara generator dengan diskriminator dalam gan untuk membuat diskriminator salah dalam melabeli suatu data.



**Gambar 8.94** gambaran penjelasan no. 6

#### 8.6.1.7 *Jelaskan dengan ilustrasi gambar apa itu Nash equilibrium*

Suatu situasi stabil dari suatu sistem yang berkaitan dengan interaksi dari banyak partisipan, dimana tidak ada satupun orang yang bisa menggantikan strateginya jika yang lain tidak mengantikannya juga



**Gambar 8.95** gambaran penjelasan no. 7

#### 8.6.1.8 Sebutkan dan jelaskan contoh-contoh implementasi dari GAN

Contohnya adalah sebagai berikut, mereka adalah orang-orang yang dihasilkan oleh algoritma ini, alias tidak ada di dunia ini.



**Gambar 8.96** gambaran penjelasan no. 8

#### 8.6.1.9 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)

Berikut adalah kode programnya, untuk gambar sudah di sertakan di nomor sebelumnya

```

1 def generator_model():
2     model = Sequential([
3         Dense(1024, input_dim=100, activation='tanh'),
4         Dense(128*7*7),
5         Reshape((7, 7, 128)),
6         UpSampling2D(size=(2, 2)),
7         Conv2D(64, (5, 5), padding='same', activation='tanh'),
8         UpSampling2D(size=(2, 2)),
9         Conv2D(1, (5, 5), padding='same', activation='tanh')
10    ])
11    return model
12
13 generator_model().summary()

```

**Listing 8.41** Kode program arsitektur generator

#### **8.6.1.10 Berikan contoh dengan ilustrasi dari arsitektur dikriminator dengan sebuah input layer, 3 buah hidden layer, dan satu output layer.**

untuk gambar arsitektur nya ada pada gambar no. 4 sedangkan untuk kode programnya seperti berikut:

```

1 def discriminator_model():
2     model = Sequential([
3         Conv2D(64, (5, 5), padding='same', input_shape=(28, 28, 1),
4         activation='tanh'),
5         MaxPooling2D(pool_size=(2, 2)),
6         Conv2D(128, (5, 5), activation='tanh'),
7         MaxPooling2D(pool_size=(2, 2)),
8         Flatten(),
9         Dense(1024, activation='tanh'),
10        Dense(1, activation='sigmoid')
11    ])
12    return model
13 discriminator_model().summary()

```

**Listing 8.42** Kode program arsitektur diskriminator

#### **8.6.1.11 Jelaskan bagaimana kaitan output dan input antara generator dan diskriminator tersebut. Jelaskan kenapa inputan dan outputan seperti itu.**

Kaitannya antara output dengan input adalah pelabelan data, generator akan terus berusaha menghasilkan data fake yang pada akhirnya karena sangat mirip maka diskriminator akan melabelinya asli.

#### **8.6.1.12 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.**

Kullback–Leibler divergence mengukur bagaimana suatu distribusi probabilitas berbeda

dari yang kedua, dari kedua algoritma diatas, yang paling umum diketahui oleh banyak orang adalah KL divergence, kl divergence memiliki beberapa sisi bagus contohnya adalah  $KL[q; p]$  jenis daerah yang di mana  $q(x)$  memiliki massa non-null dan  $p(x)$  memiliki massa null. memang terlihat seperti bug ,tetapi dalam situasi tertentu ini adalah fitur. dan JS tidak memilikinya,

**8.6.1.13 Jelaskan apa itu fungsi objektif yang berfungsi untuk mengukur kesamaan antara gambar yang dibuat dengan yang asli.**

fungsi objektif adalah suatu fungsi yang akan mengambil beberapa parameter di suatu data dan juga model yang kemudian dijadikan argumen untuk dievaluasi, sehingga nantinya akan mengembalikan angka yang mampu membedakan antara gambar asli dan palsu.

**8.6.1.14 Jelaskan apa itu scoring algoritma selain mean square error atau cross entropy seperti The Inception Score dan The Frechet Inception distance.**

Inception score digunakan untuk mengukur kualitas dari gambar yang digenerated, khususnya yang tiruan,sintetik seperti yang dihasilkan algoritma GAN, sedangkan Frechet Inception distance digunakan untuk mengukur dua data set gambar, berkaitan sekali dengan kualitas visual manusia dalam menentukan gambar, dan algoritma ini sering digunakan di GAN

**8.6.1.15 Jelaskan kelebihan dan kekurangan GAN**

▪ Kelebihan

1. GAN merupakan metode bagus untuk melakukan training classifier dalam metode semi supervised
2. GAN melakukan proses generate yang cepat dibandingkan dengan metode lainnya.
3. GAN tidak butuh pengukuran monte carlo
4. GAN lebih mudah digunakan dibandingkan dengan VAE

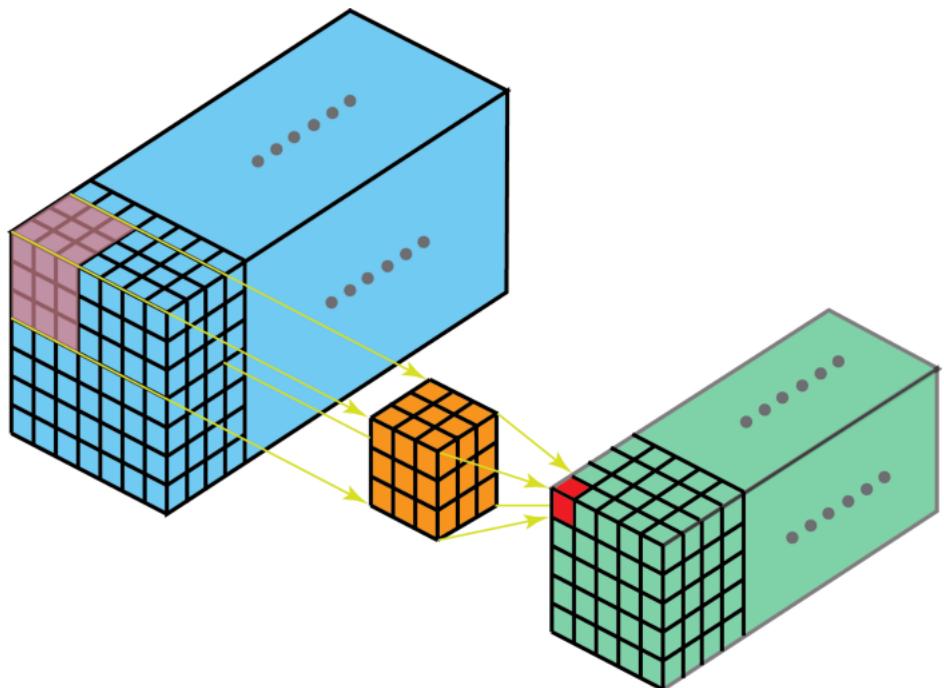
▪ Kekurangan

1. Melakukan training gan membutuhkan kita untuk mencari nash quilibrium
2. Agak sulit untuk generate data text
3. Dibandingkan dengan boltzmann machine, gan sulit untuk menebak value dari suatu pixel
4. Sangat sensitif dengan data yang sudah diinisiasi sebelumnya

## 8.6.2 Praktek

### 8.6.2.1 Jelaskan apa itu 3D convolutions

3D convolutions merupakan operasi konvolusi 3D yang menerapkan filter 3D ke data input dengan tiga arah, yaitu x,y, dan z. fitur ini menciptakan sebuah daftar peta yang ditumpuk. agar lebih mudah saya menggunakan sebuah ilustrasi seperti berikut:



**Gambar 8.97** gambaran 3D Convolutions

### 8.6.2.2 Jelaskan dengan kode program arsitektur dari generator networknya, beserta penjelasan input dan output dari generator network.

```

1 def build_generator():#nama kelasnya
2     """
3         Create a Generator Model with hyperparameters values defined as
4         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 input_layer = Input(shape=gen_input_shape)
14
15 # First 3D transpose convolution(or 3D deconvolution) block
16 a = Deconv3D(filters=gen_filters[0],
17               kernel_size=gen_kernel_sizes[0],
18               strides=gen_strides[0])(input_layer)
19 a = BatchNormalization()(a, training=True)
20 a = Activation(activation='relu')(a)
21
22 # Next 4 3D transpose convolution(or 3D deconvolution) blocks
23 for i in range(gen_convolutional_blocks - 1):
24     a = Deconv3D(filters=gen_filters[i + 1],
25                   kernel_size=gen_kernel_sizes[i + 1],
26                   strides=gen_strides[i + 1], padding='same')(a)
27     a = BatchNormalization()(a, training=True)
28     a = Activation(activation=gen_activations[i + 1])(a)
29
30 gen_model = Model(inputs=[input_layer], outputs=[a])
31

```

penjelasan

- Input layer mengambil 100-dimensi sampel dari distribusi Gaussian(normal) dan meneruskan tensor ke. hidden layer pertama tanpa ada modifikasi
- ketiga hidden layer adalah dense layer dengan unit masing-masing 500, 500, dan 784. dimana dense layer pertama mengkonversi bentuk tensor (batch\_size, 100) ke bentuk tensor(batch\_size, 500). sedangkan pada layer dense kedua menghasilkan bentuk tensor (batch\_size, 500), dan layer dense ketiga menghasilkan (batch\_size, 784)
- Output layer, tensor akan dibentuk kembali dari bentuk tensor (batch\_size, 784) menjadi (batch\_size, 28, 28) artinya hasil dari generator ini akan menghasilkan banyak gambar, dimana setiap gambarnya memiliki ukuran (28, 28).

#### **8.6.2.3 Jelaskan dengan kode program arsitektur dari diskriminatator network, beserta penjelasan input dan outputnya.**

```

1 def build_discriminator():
2     """
3         Create a Discriminator Model using hyperparameters values defined
4         as follows
5     """
6     dis_input_shape = (64, 64, 64, 1)
7     dis_filters = [64, 128, 256, 512, 1]
8     dis_kernel_sizes = [4, 4, 4, 4, 4]
9     dis_strides = [2, 2, 2, 2, 1]
10    dis_paddings = ['same', 'same', 'same', 'same', 'valid']
11    dis_alphas = [0.2, 0.2, 0.2, 0.2, 0.2]
12    dis_activations = ['leaky_relu', 'leaky_relu', 'leaky_relu',
13                      'leaky_relu', 'sigmoid']
14    dis_convolutional_blocks = 5

```

```

15 dis_input_layer = Input(shape=dis_input_shape)
16
17 # The first 3D Convolutional block
18 a = Conv3D(filters=dis_filters[0],
19             kernel_size=dis_kernel_sizes[0],
20             strides=dis_strides[0],
21             padding=dis_paddings[0])(dis_input_layer)
22 # a = BatchNormalization()(a, training=True)
23 a = LeakyReLU(dis_alphas[0])(a)
24
25 # Next 4 3D Convolutional Blocks
26 for i in range(dis_convolutional_blocks - 1):
27     a = Conv3D(filters=dis_filters[i + 1],
28                 kernel_size=dis_kernel_sizes[i + 1],
29                 strides=dis_strides[i + 1],
30                 padding=dis_paddings[i + 1])(a)
31     a = BatchNormalization()(a, training=True)
32     if dis_activations[i + 1] == 'leaky_relu':
33         a = LeakyReLU(dis_alphas[i + 1])(a)
34     elif dis_activations[i + 1] == 'sigmoid':
35         a = Activation(activation='sigmoid')(a)
36
37 dis_model = Model(inputs=[dis_input_layer], outputs=[a])
38 return dis_model

```

penjelasan

- awalnya diskriminator menerima input dengan bentuk 28x28.
- Input layer mengambil tensor input dan meneruskannya ke hidden layer pertama tanpa modifikasi apapun.
- lalu flattens layer akan meratakan tensor menjadi 784-dimensi vektor, yang akan diteruskan ke hidden layer (dense layer) pertama. hidden layer pertama dan kedua aka memodifikasinya menjadi vektor 500-dimensi.
- Output layer mesuk kedalam dense layer, dengan satu unit neuron dan sigmoid sebagai fungsi aktivasi. sigmoid akan menghasilkan nilai tunggal, 0 atau 1. Nilai 0 akan menunjukan bahwa gambar yang diberikan palsu. sebaliknya jika nilai 1 maka gambar asli.

#### **8.6.2.4 Jelaskan proses training 3D-GANs**

Proses training 3D-GANs dilakukan seperti langkah-langkah berikut:

- Terdapat sebuah vektor noise dengan dimensi 200 dari distribusi Gaussian(normal).
- Meng-generate gambar palsu menggunakan model generator.
- Melatih jaringan generator dengan gambar yang asli(sampel dari data yang real) dan dengan gambar palsu yang dihasilkan oleh generator.
- Gunakan adversarial model untuk melatih generator model, jangan melatih diskriminator model.
- Ulangi langkah ini dengan jumlah epoch tertentu.

### **8.6.2.5 Jelaskan bagaimana melakukan settingan awal chapter 02 untuk memenuhi semua kebutuhan sebelum melanjutkan ke tahapan persiapan data.**

Sebelum melakukan tahapan persiapan data, kita harus melakukan beberapa hal seperti berikut:

- membaca buku panduan atau keterangan dari Generative-Adversial-Network project ini. bisa didownload di portal kampus keren atau di github <https://github.com/PacktPublishing/Generative-Adversarial-Networks-Projects>
- persiapkan laptop/pc dengan spek yang cukup bagus(tidak kentang). jika tidak ada, bisa menggunakan google colab(cara penggunaanya akan dijelaskan di video).
- usahakan versi yang terinstall sama dengan versi yang ada pada requirement.txt agar terhindar dari error.

### **8.6.2.6 Jelaskan tentang dataset yang digunakan, dari mulai tempat unduh, cara membuka dan melihat data. sampai deskripsi dari isi dataset dengan detail penjelasan setiap folder/file yang membuat orang awam paham.**

Penjelasan dataset

- dataset bisa didownload pada halaman: <http://3dshapenets.cs.princeton.edu/3DShapeNets/>
- Untuk membuka data, setelah didownload cukup di eksrak saja menggunakan winrar/ 7zip
- Untuk mengetahui lebih jelasnya tentang dataset ini bisa dibuka saja file README.md, disana akan dijelaskan semuanya.

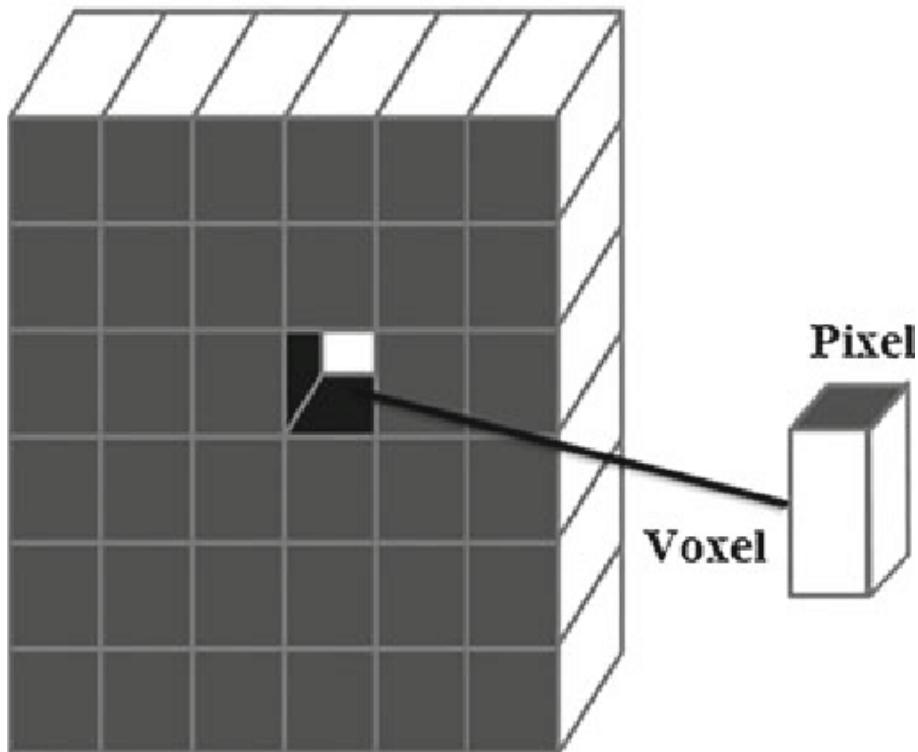
3D	5/27/2015 2:25 AM	File folder	
bp	5/27/2015 2:58 AM	File folder	
generative	5/27/2015 3:07 AM	File folder	
util	5/27/2015 6:14 AM	File folder	
volumetric_data	4/16/2015 4:14 AM	File folder	
voxelization	1/12/2015 11:36 PM	File folder	
.gitignore	1/12/2015 11:36 PM	Text Document	1 KB
discriminative_10_class.mat	5/27/2015 3:19 AM	Microsoft Access ...	147,828 KB
discriminative_40_class.mat	5/27/2015 3:21 AM	Microsoft Access ...	148,607 KB
generate_completions.m	5/25/2015 8:55 AM	Objective C Sourc...	3 KB
generative_10_class.mat	5/27/2015 7:03 AM	Microsoft Access ...	46,284 KB
generative_40_class.mat	5/27/2015 7:08 AM	Microsoft Access ...	59,184 KB
kernels.m	4/15/2015 5:42 AM	Objective C Sourc...	1 KB
kFunctions.cu	1/12/2015 11:36 PM	CU File	39 KB
kFunctions.ptx	1/29/2015 8:01 AM	PTX File	127 KB
kFunctions2.cu	1/29/2015 7:53 AM	CU File	39 KB
kFunctions2.ptx	1/29/2015 7:53 AM	PTX File	127 KB
NBV_onestep_test.m	5/27/2015 7:15 AM	Objective C Sourc...	3 KB
README.md	5/27/2015 7:00 AM	Markdown Source...	3 KB
rec_completion_test.m	5/25/2015 8:56 AM	Objective C Sourc...	10 KB
rec_test.m	5/25/2015 9:22 AM	Objective C Sourc...	3 KB
run_finetuning.m	4/16/2015 6:10 AM	Objective C Sourc...	1 KB
run_pretrain.m	4/16/2015 6:11 AM	Objective C Sourc...	4 KB
sample_test_classification.m	5/25/2015 9:15 AM	Objective C Sourc...	3 KB
sample_test_extreme.m	5/25/2015 9:15 AM	Objective C Sourc...	5 KB

**Gambar 8.98** Isi dari dataset.zip

### 8.6.2.7 Jelaskan apa itu voxel dengan ilustrasi dan bahasa paling awam

Secara singkat, voxel dapat diartikan sebagai 3D pixel atau pixel dalam bentuk 3D, seperti pada ilustrasi berikut:

## Matrix of image



Gambar 8.99 Pixel VS Voxel

### 8.6.2.8 Visualisasikan dataset tersebut dalam tampilan visual plot, jelaskan cara melakukan visualisasinya

```

1 import scipy.io as io
2 import matplotlib.pyplot as plt
3 import scipy.ndimage as nd
4 import numpy as np
5 from mpl_toolkits.mplot3d import Axes3D
6
7 voxels = io.loadmat("/content/3DShapeNets/volumetric_data/flower_pot
8 /30/test/flower_pot_000000010_1.mat")['instance']
9 voxels = np.pad(voxels, (1, 1), 'constant', constant_values=(0, 0))
10 voxels = nd.zoom(voxels, (2, 2, 2), mode='constant', order=0)

```

```

11 fig = plt.figure()
12 ax = Axes3D(fig)
13 ax.voxels(voxels, edgecolor="chartreuse")
14
15 plt.show()
16 plt.savefig('flower_pot')

```

Kode diatas berfungsi untuk melakukan visualisasi dataset dalam tampilan plot. dengan tahapan import library, load data file .mat, dan lakukan read memakai matplotlib. dan hasilnya seperti berikut

#### **8.6.2.9 Buka file run.py jelaskan perbaris kode pada fungsi untuk membuat generator yaitu build generator**

```

1 z_size = 200
2 gen_filters = [512, 256, 128, 64, 1]
3 gen_kernel_sizes = [4, 4, 4, 4, 4]
4 gen_strides = [1, 2, 2, 2, 2]
5 gen_input_shape = (1, 1, 1, z_size)
6 gen_activations = ['relu', 'relu', 'relu', 'relu', 'sigmoid']
7 gen_convolutional_blocks = 5

```

potongan kode diatas berfungsi sebagai penentuan nilai untuk hyperparameters yang berbeda.

```
1 input_layer = Input(shape=gen_input_shape)
```

potongan kode diatas berfungsi untuk membuat input layer

```

1 # First 3D transpose convolution(or 3D deconvolution) block
2 a = Deconv3D(filters=gen_filters[0],
3               kernel_size=gen_kernel_sizes[0],
4               strides=gen_strides[0])(input_layer)
5 a = BatchNormalization()(a, training=True)
6 a = Activation(activation='relu')(a)

```

potongan kode diatas berfungsi untuk menambahkan 3D transpose pertama.

```

1 # Next 4 3D transpose convolution(or 3D deconvolution) blocks
2 for i in range(gen_convolutional_blocks - 1):
3     a = Deconv3D(filters=gen_filters[i + 1],
4                   kernel_size=gen_kernel_sizes[i + 1],
5                   strides=gen_strides[i + 1], padding='same')(a)
6     a = BatchNormalization()(a, training=True)
7     a = Activation(activation=gen_activations[i + 1])(a)

```

potongan kode diatas berfungsi untuk menambahkan empat 3D transpose lainya.

```
1 gen_model = Model(inputs=[input_layer], outputs=[a])
```

potongan kode diatas berfungsi untuk membuat model keras dan menentukan input dan output untuk network generator.

#### **8.6.2.10 jelaskan juga fungsi untuk membangun diskriminatator pada fungsi build discriminator.**

```

1 dis_input_shape = (64, 64, 64, 1)
2 dis_filters = [64, 128, 256, 512, 1]
3 dis_kernel_sizes = [4, 4, 4, 4, 4]
4 dis_strides = [2, 2, 2, 2, 1]
5 dis_paddings = ['same', 'same', 'same', 'same', 'valid']
6 dis_alphas = [0.2, 0.2, 0.2, 0.2, 0.2]
7 dis_activations = ['leaky_relu', 'leaky_relu', 'leaky_relu',

```

potongan kode diatas berfungsi sebagai penentuan nilai untuk hyperparameters yang berbeda.

```

1 dis_input_layer = Input(shape=dis_input_shape)

```

potongan kode diatas berfungsi untuk membuat input layer, berupa gambar 3D dengan dimensi 64x64x64x1

```

1 # The first 3D Convolutional block
2 a = Conv3D(filters=dis_filters[0],
3             kernel_size=dis_kernel_sizes[0],
4             strides=dis_strides[0],
5             padding=dis_paddings[0])(dis_input_layer)
6 # a = BatchNormalization()(a, training=True)
7 a = LeakyReLU(dis_alphas[0])(a)

```

potongan kode diatas berfungsi untuk menambahkan 3D transpose pertama.

```

1 # Next 4 3D Convolutional Blocks
2 for i in range(dis_convolutional_blocks - 1):
3     a = Conv3D(filters=dis_filters[i + 1],
4                 kernel_size=dis_kernel_sizes[i + 1],
5                 strides=dis_strides[i + 1],
6                 padding=dis_paddings[i + 1])(a)
7     a = BatchNormalization()(a, training=True)
8     if dis_activations[i + 1] == 'leaky_relu':
9         a = LeakyReLU(dis_alphas[i + 1])(a)
10    elif dis_activations[i + 1] == 'sigmoid':
11        a = Activation(activation='sigmoid')(a)

```

potongan kode diatas berfungsi untuk menambahkan empat 3D transpose lainnya.

```

1 dis_model = Model(inputs=[dis_input_layer], outputs=[a])

```

potongan kode diatas berfungsi untuk membuat model keras dan menentukan input dan output untuk network generator.

#### **8.6.2.11 jelaskan apa maksud dari kode `program name == 'main'`**

```

1 if __name__ == '__main__':

```

**Listing 8.43 Kode program run.py**

maksudnya kode tersebut memiliki fungsi jika interpreter python menjalankan kode tersebut, maka akan menetapkan variable name untuk memiliki nilai main. jika file ini di import dari modul lain, maka name akan ditetapkan ke nama modul tersebut.

**8.6.2.12 Jelaskan secara detil perbaris dan per parameter apa arti dari kode program :**

Penjelasan kode:

```

1 # In[ soal_12]
2     object_name = "airplane" #membuat variabel
3     data_dir = "data/3DShapeNets/volumetric_data/" \
4             "{}/30/train/*.mat".format(object_name) #menunjukan
5             direktori dari variable sebelumnya
6     gen_learning_rate = 0.0025 #membuat rate pada generator
7     dis_learning_rate = 10e-5 #membuat rate pada diskriminat
8     beta = 0.5 #
9     batch_size = 1 #menentukan ukuran kumpulan gambar
10    z_size = 200 #
11    epochs = 1 #menetukan masa periode. setiap epoch memiliki semua
12        batch
13    MODE = "train" #membuat traon mode

```

**Listing 8.44** Kode program run.py

**8.6.2.13 Jelaskan secara detil dari kode program pembuatan dan kompilasi arsitektur berikut :**

```

1     gen_optimizer = Adam(lr=gen_learning_rate , beta_1=beta)
2     dis_optimizer = Adam(lr=dis_learning_rate , beta_1=beta)
3
4     discriminator = build_discriminator()
5     discriminator.compile(loss='binary_crossentropy' , optimizer=
6         dis_optimizer)
7
8     generator = build_generator()
9     generator.compile(loss='binary_crossentropy' , optimizer=
10        gen_optimizer)

```

**Listing 8.45** Kode program run.py

penjelasannya ialah membuat model dari generator model dan diskriminat model,

**8.6.2.14 Jelaskan secara detil kode program untuk membuat dan melakukan kompilasi model adversarial berikut:**

```

1 # In[ soal_14]
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=[validity
8         ])
9     adversarial_model.compile(loss='binary_crossentropy' , optimizer=
10        gen_optimizer)

```

**Listing 8.46** Kode program run.py

penjelasannya ialah membuat model diskriminat tidak di training dan membuat model adversarial.

**8.6.2.15 Jelaskan Ekstrak dan load data kursi dengan menggunakan fungsi `getVoxelsFormat` dan `get3DImages` yang digunakan pada kode program berikut :**

```
1 # In[ soal_15]
2     print("Loading data ...")
3     volumes = get3DImages(data_dir=data_dir)
4     volumes = volumes [..., np.newaxis].astype(np.float)
5     print("Data loaded ...")
```

**Listing 8.47** Kode program run.py

penjelasanya ialah melakukan print loading data, membuat variabel volumes untuk mendapatkan data gamabr 3D.

**8.6.2.16 Jelaskan maksud dari kode program instansiasi TensorBoard yang menambahkan generator dan diskriminator pada program berikut:**

```
1 # In[ soal_16]
2     tensorboard = TensorBoard(log_dir="logs/{}".format(time.time()))
3     tensorboard.set_model(generator)
4     tensorboard.set_model(discriminator)
```

**Listing 8.48** Kode program run.py

penjelasanya ialah membuat variabel tensorboard untuk melakukan pencatatan log, dan men-set model generator dan model diskriminator

**8.6.2.17 Jelaskan apa fungsi dari `np.reshape ones zeros` pada kode program berikut dengan parameternya:**

Penjelasan kode:

```
1 # In[ soal_17]
2     labels_real = np.reshape(np.ones((batch_size,)), (-1, 1, 1, 1, 1))
3     labels_fake = np.reshape(np.zeros((batch_size,)), (-1, 1, 1, 1, 1))
```

**Listing 8.49** Kode program run.py

membuat label\_real untuk menetapkan bahwa jika output 1 itu menunjukan gambar real, dan label\_fake untuk menunjukan bahwa output 0 untuk data fake/palsu.

**8.6.2.18 Jelaskan kenapa harus ada perulangan dalam meraih epoch. Dan jelaskan apa itu epoch terkait kode program berikut:**

```
1 # In[ soal_18]
2     if MODE == 'train':
3         for epoch in range(epochs):
4             print("Epoch:", epoch)
5
6             gen_losses = []
```

```
7 dis_losses = []
```

**Listing 8.50** Kode program run.py

melakukan perulangan jika MODE = train maka akan diulangi terus menerus sampai epoch(periode) terpenuhi dengan disertai keterangan generator dan diskriminatator losses.

**8.6.2.19 Jelaskan apa itu batches dan kaitannya dengan kode program berikut, dan kenapa berada di dalam epoch:**

```
1 # In[ soal 19]
2     number_of_batches = int(volumes.shape[0] / batch_size)
3     print("Number of batches:", number_of_batches)
4     for index in range(number_of_batches):
5         print("Batch:", index + 1)
```

**Listing 8.51** Kode program run.py

menghitung jumlah batch yang telah ditrain

**8.6.2.20 Berikut adalah kode program pengambilan gambar dan noise. Jelaskan apa fungsi np.random.normal serta astype, serta jelaskan apa arti parameter titik dua dan jelaskan isi dari z sample dan volumes batch:**

```
1 # In[ soal 20]
2     z_sample = np.random.normal(0, 0.33, size=[batch_size
3     , 1, 1, 1, z_size]).astype(np.float32)
4     volumes_batch = volumes[index * batch_size:(index +
5     1) * batch_size, :, :, :]
```

**Listing 8.52** Kode program run.py

untuk membersihkan gambar dari noise dan menyesuaikan shape.

**8.6.2.21 Berikut adalah kode program generator gambar palsu. Jelaskan apa fungsi generator.predict on batch, serta jelaskan apa arti parameter z sample:**

```
1 # In[ soal 21]
2     # Next, generate volumes using the generate network
3     gen_volumes = generator.predict_on_batch(z_sample)
```

**Listing 8.53** Kode program run.py

untuk meng-generate volume menggunakan model jaringan generator.

**8.6.2.22 Berikut adalah kode program training diskriminatator dengan gambar palsu dari generator dan gambar asli. Jelaskan apa maksudnya harus dilakukan training diskriminatator secara demikian dan jelaskan apa isi loss fake dan loss real serta d loss dan fungsi train on batch.**

```

1 # In[ soal_22]
2     discriminator.trainable = True
3     if index % 2 == 0:
4         loss_real = discriminator.train_on_batch(
5             volumes_batch, labels_real)
6         loss_fake = discriminator.train_on_batch(
7             gen_volumes, labels_fake)
8
9         d_loss = 0.5 * np.add(loss_real, loss_fake)
10        print("d_loss:{:.2f}".format(d_loss))
11
12    else:
13        d_loss = 0.0

```

**Listing 8.54** Kode program run.py

untuk membuat diskriminasi bisa meload gambar fake dan real dari model generator. dengan disertai generasi dan diskriminasi loss, agar terlihat seberapa baik kualitas yang dihasilkan.

**8.6.2.23 Berikut adalah kode program training model adversarial yang terdapat generator dan diskriminasi. Jelaskan apa bagaimana proses terbentuknya parameter z dan g loss:**

```

1 # In[ soal_23]
2     discriminator.trainable = False
3     """
4         Train the generator network
5     """
6     z = np.random.normal(0, 0.33, size=[batch_size, 1, 1,
7         1, z_size]).astype(np.float32)
8     g_loss = adversarial_model.train_on_batch(z,
9         labels_real)
10    print("g_loss:{:.2f}".format(g_loss))
11
12    gen_losses.append(g_loss)
13    dis_losses.append(d_loss)

```

**Listing 8.55** Kode program run.py

untuk mentrain model generator dan variabel g\_loss untuk melakukan perbandingan antara label gambar yang asli.

**8.6.2.24 Berikut adalah kode program generate dan menyimpan gambar 3D setelah beberapa saat setiap epoch. Jelaskan mengapa ada perulangan dengan parameter tersebut, serta jelaskan arti setiap variabel beserta perlihatkan isinya dan artikan isinya :**

```

1 # In[ soal_24]
2         # Every 10th mini-batch, generate volumes and save
3         them
4         if index % 10 == 0:

```

```

4         z_sample2 = np.random.normal(0, 0.33, size=[
5             batch_size, 1, 1, 1, z_size]).astype(np.float32)
6             generated_volumes = generator.predict(z_sample2,
7                 verbose=3)
8             for i, generated_volume in enumerate(
9                 generated_volumes[:5]):
10                 voxels = np.squeeze(generated_volume)
11                 voxels[voxels < 0.5] = 0.
12                 voxels[voxels >= 0.5] = 1.
13                 saveFromVoxels(voxels, "results/img_{:}-{:}-{:}""
14 .format(epoch, index, i))

```

**Listing 8.56** Kode program run.py

untuk melakukan perulangan dimana setiap terdapat 10 mini-batch akan meng-generate volumenya dan akan menyimpannya.

**8.6.2.25 Berikut adalah kode program menyimpan average losses setiap epoch. Jelaskan apa itu tensorboard dan setiap parameter yang digunakan pada kode program ini :**

```

1 # In[ soal 25]
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)

```

**Listing 8.57** Kode program run.py

untuk menuliskan log losses ke Tensorboard

**8.6.2.26 Berikut adalah kode program menyimpan model. Jelaskan apa itu format h5 dan penjelasan dari kode program berikut :**

```

1 # In[ soal 26]
2     generator.save_weights(os.path.join("models", "
3     generator_weights.h5"))
4     discriminator.save_weights(os.path.join("models", "
5     discriminator_weights.h5"))

```

**Listing 8.58** Kode program run.py

untuk menyimpan berat model generator dan model diskriminator kedalam file h5.

H5 adalah Hierarchical Data Format 5 File. File H5 adalah file data yang disimpan dalam Format Data Hirarki (HDF). Ini berisi array multidimensi data ilmiah. File H5 biasanya digunakan di luar angkasa, fisika, teknik, keuangan, penelitian akademis, genomik, astronomi, instrumen elektronik, dan bidang medis.

**8.6.2.27 Berikut adalah kode program testing model. Jelaskan dengan ilustrasi gambar dari mulai meload hingga membuat gambar 3D dengan menggunakan z sample, bisakah parameter z sample tersebut diubah? :**

```

1     generator.load_weights(os.path.join("models", "generator_weights.h5"), True)
2     discriminator.load_weights(os.path.join("models", "discriminator_weights.h5"), True)
3
4     # Generate 3D models
5     z_sample = np.random.normal(0, 1, size=[batch_size, 1, 1, 1,
6     z_size]).astype(np.float32)
7     generated_volumes = generator.predict(z_sample, verbose=3)
8
9     for i, generated_volume in enumerate(generated_volumes[:2]):
10        voxels = np.squeeze(generated_volume)
11        voxels[voxels < 0.5] = 0.
12        voxels[voxels >= 0.5] = 1.
13        saveFromVoxels(voxels, "results/gen_{}".format(i))

```

**Listing 8.59** Kode program run.py

untuk membuat mode predict/prediksi. dimana dia melakukan pembuatan model generator, model diskriminatator dan meload data h5 yang sudah dibuat tadi dan menggenerate nya kedalam model 3D, lalu menyimpan model voxel nya ke folde results.

### 8.6.3 Penanganan Error

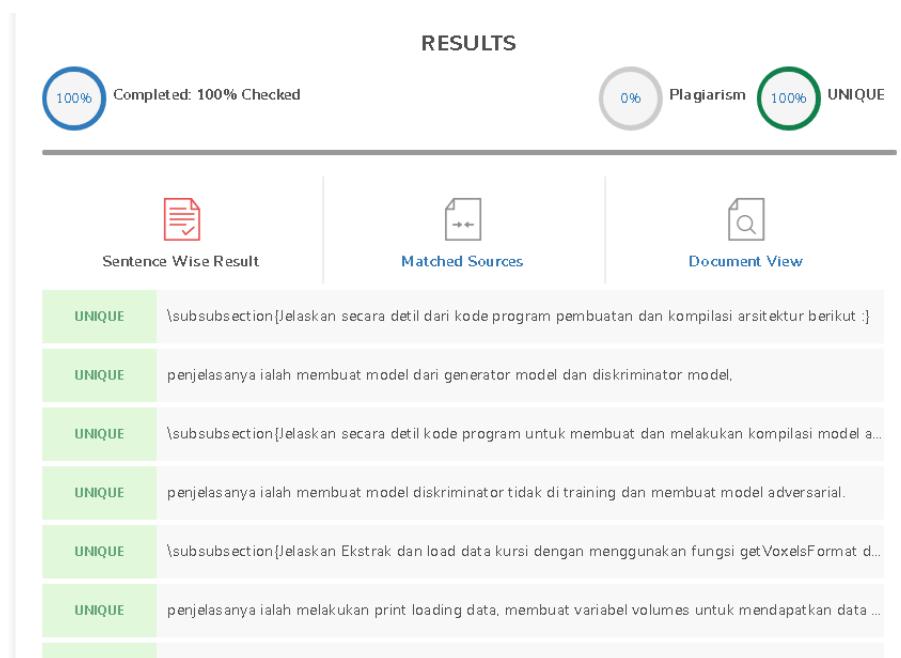
#### 8.6.3.1 Terjadi error

1. terjadi error Summary has no attribute, seperti pada gambar berikut:

#### 8.6.3.2 Solusi

1. solusi dari error 1 ialah: dengan menginstall versi tensorflow 1.xx

## 8.6.4 Bukti Tidak Plagiat



**Gambar 8.100** Bukti tidak plagiat

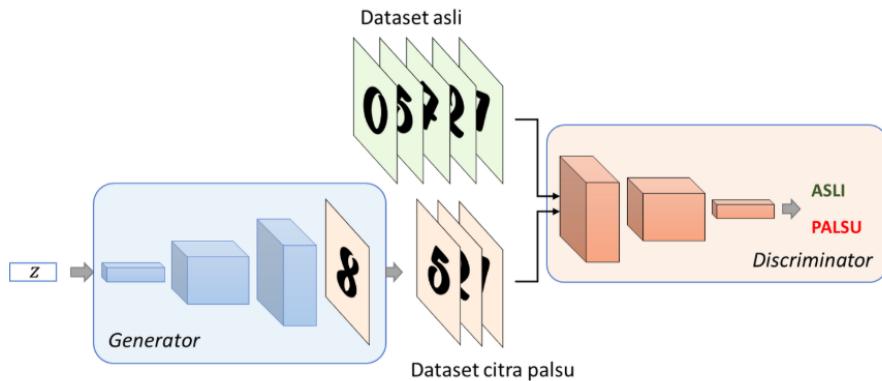
## 8.6.5 Link Youtube

[shorturl.at/cGIO2](http://shorturl.at/cGIO2)

## 8.7 Muhammad Reza Syachrani - 1174084

### 8.7.1 Teori

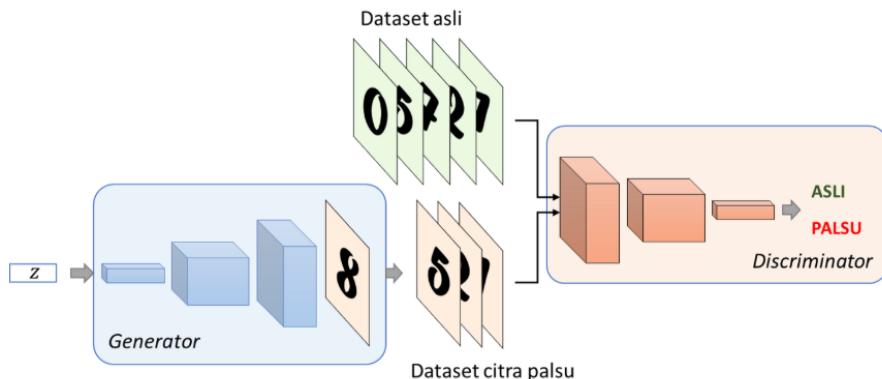
1. Jelaskan dengan ilustrasi gambar sendiri apa itu generator dengan perumpamaan anda sebagai mahasiswa sebagai generatornya.  
Tugas Generator sekarang sedang dibuat untuk membuat koleksi gambar palsu, yang saat ini dilihat oleh Diskriminator. Diskriminator tidak dapat membedakan antara yang asli dan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



**Gambar 8.101** Teori 1

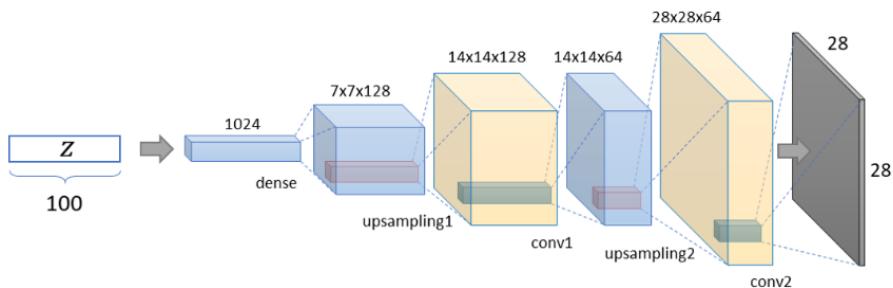
2. Jelaskan dengan ilustrasi gambar sendiri apa itu diskriminator dengan perumpamaan dosen anda sebagai diskriminornya

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:



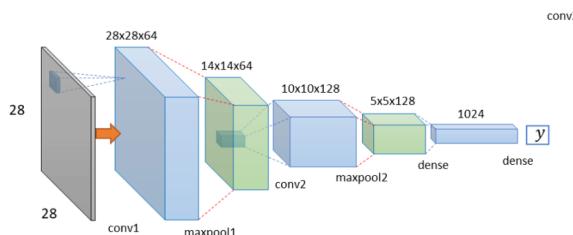
**Gambar 8.102** Teori 2

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

**Gambar 8.103** Teori 3

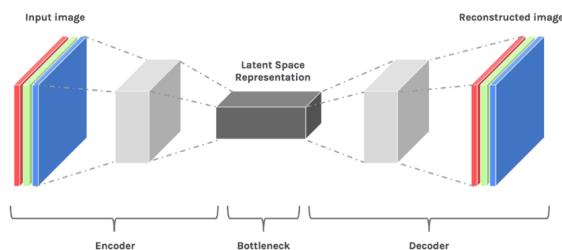
4. Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur diskriminator dibuat.

Arsitektur diskriminator dibuat bisa dijelaskan pada gambar berikut :

**Gambar 8.104** Teori 4

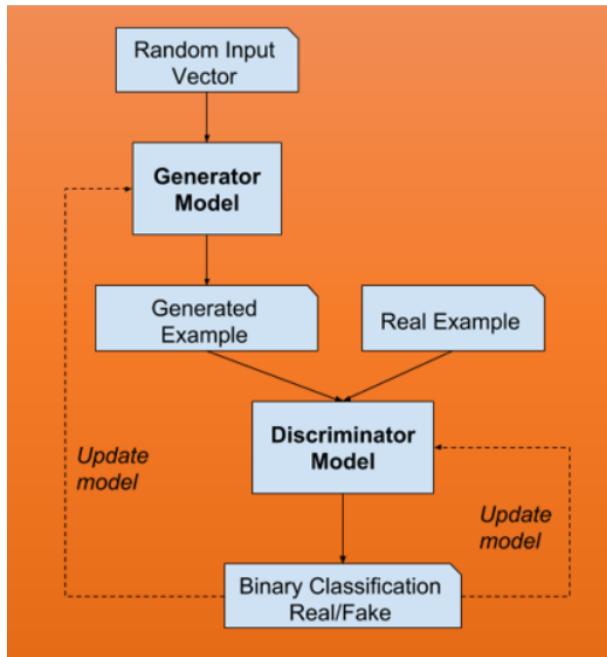
5. Jelaskan dengan ilustrasi gambar apa itu latent space.

Latent space dijelaskan pada gambar berikut :

**Gambar 8.105** Teori 5

6. Jelaskan dengan ilustrasi gambar apa itu adversarial play  
adversarial play ialah persaingan antara generator dan diskriminat

or. bisa dilihat pada ilustrasi gambar berikut:



Gambar 8.106 Teori 6

7. Jelaskan dengan ilustrasi gambar apa itu Nash equilibrium.

Nash equilibrium konsep teori permainan di mana hasil optimal dari permainan adalah di mana tidak ada pemain yang memiliki insentif untuk menyimpang dari strategi yang dipilih setelah mempertimbangkan pilihan lawan Nash equilibrium pada gambar berikut :

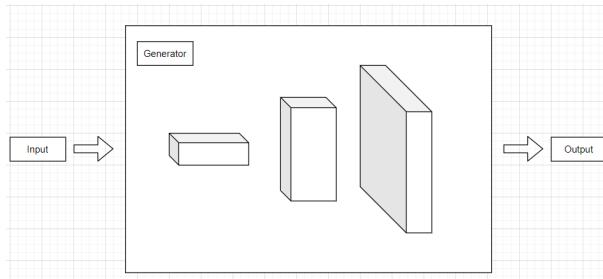
		Player 2	Option A	Option B	Option C
		Player 1	Option A	Option B	Option C
Player 1	Option A	0, 0	25, 40	5, 10	
	Option B	40, 25	0, 0	5, 15	
	Option C	10, 5	15, 5	10, 10	

Gambar 8.107 Teori 7

8. Sebutkan dan jelaskan contoh-contoh implementasi dari GAN

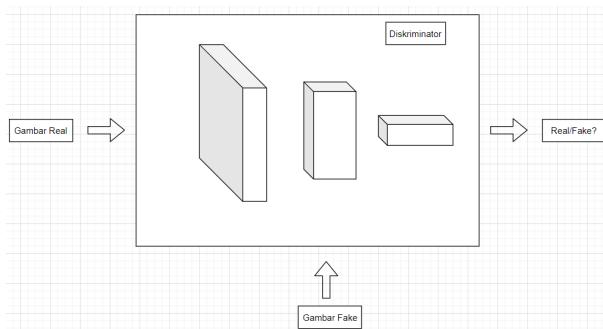
Implementasi 3DGAN yaitu pada MAPS dan juga IKEA. Karena pada maps dan juga ikea sudah menerapkan bentuk 3 dimensi yang bisa lebih menarik perhatikan pengguna.

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).



**Gambar 8.108** Teori 9

10. Berikan contoh dengan ilustrasi dari arsitektur dikriminator dengan sebuah input layer, 3 buah hidden layer, dan satu output layer.



**Gambar 8.109** Teori 10

11. Jelaskan bagaimana kaitan output dan input antara generator dan diskriminat tersebut. Jelaskan kenapa inputan dan outputan seperti itu  
Pada setiap metode tersebut (Discriminator generator) akan dilakukan pelatihan dan akan dibandingkan hasilnya. Generator akan menghasilkan data baru sesuai dengan hasil latihan dan dari data tersebut, discriminator akan membandingkan dengan data set apakah data tersebut asli atau tidak.
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.  
relative entropy adalah ukuran dari bagaimana satu distribusi probabilitas berbeda

dari yang kedua, distribusi probabilitas referensi, Divergensi Jensen-Shannon adalah ukuran divergensi berprinsip yang selalu terbatas untuk variabel acak terbatas.

13. Jelaskan apa itu fungsi objektif yang berfungsi untuk mengukur kesamaan antara gambar yang dibuat dengan yang asli.

Fungsi objektif adalah fungsi yang digunakan sebagai penujuk berapa nilai kesamaan antara gambar yang dibuat dengan yang asli

14. Jelaskan apa itu scoring algoritma selain mean square error atau cross entropy seperti The Inception Score dan The Frechet Inception distance.

Inception Score digunakan untuk mengukur seberapa realistik output dari GAN, dimana ada dua parameter, yaitu: gambarnya punya variasi dan setiap gambar jelas terlihat seperti sesuatu. Frechet Inception Distance adalah ukuran kesamaan antara dua dataset gambar. Itu terbukti berkorelasi baik dengan penilaian manusia terhadap kualitas visual dan paling sering digunakan untuk mengevaluasi kualitas sampel Generative Adversarial Networks. FID dihitung dengan menghitung jarak Frechet antara dua Gaussians dipasang ke representasi fitur dari jaringan Inception.

15. Jelaskan kelebihan dan kekurangan GAN

Keunggulan GAN Menghasilkan data baru yang bisa hampir mirip dengan data asli. Karena hasil pelatihannya, GAN dapat menghasilkan data gambar, teks, audio, dan video yang dapat dibilang hampir mirip dengan yang aslinya. Berkat hal tersebut, GAN dapat digunakan dalam sistem marketing, e-commerce, games, iklan, dan industri lainnya GAN mempelajari representasi data secara internal sehingga beberapa masalah pada machine learning dapat diatasi dengan mudah. Discriminator yang sudah dilatih dapat menjadi sebuah classifier atau pendeteksi jika data sudah sesuai. Karena Discriminator yang akan menjadi tidak efisien berkat seringnya dilatih GAN dapat dilatih menggunakan data yang belum dilabeled

Kerugian Data saat diproses oleh metode gan tidak konvergensi Jenis sampel yang dihasilkan oleh generator terbatas karena modenya terbatas Ketidak seimbangnya antara generator dan discriminator dapat menyebabkan overfitting atau terlalu dekat dengan hasil sampel Sangat sensitif dengan data yang sudah diinisiasi sebelumnya.

### 8.7.2 Praktek

1. Jelaskan apa itu 3D convolutions

Konvolusi 3D menerapkan filter 3 dimensi ke dataset dan filter bergerak 3 arah ( $x, y, z$ ) untuk menghitung representasi fitur level rendah. Bentuk output mereka adalah ruang volume 3 dimensi seperti kubus atau kuboid.

2. Jelaskan dengan kode program arsitektur dari generator networknya, beserta penjelasan input dan output dari generator network.

```

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')(a)
28        a = BatchNormalization()(a, training=True)
29        a = Activation(activation=gen_activations[i + 1])(a)
30
31    gen_model = Model(inputs=[input_layer], outputs=[a])
32    return gen_model

```

generator ialah g\_loss, Bentuk jaringan Generator dapat dilihat berkebalikan dengan struktur jaringan saraf pada umumnya. Jaringan Generator menerima input sebuah vektor angka z, kemudian mengubahnya menjadi output gambar tiga dimensi.

3. Jelaskan dengan kode program arsitektur dari diskriminator network, beserta penjelasan input dan outputnya.

```

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     dis_convolutional_blocks = 5
16
17     dis_input_layer = Input(shape=dis_input_shape)
18
19     # The first 3D Convolutional block
20     a = Conv3D(filters=dis_filters[0],
21                 kernel_size=dis_kernel_sizes[0],
22                 strides=dis_strides[0],
23                 padding=dis_paddings[0])(dis_input_layer)
24     # a = BatchNormalization()(a, training=True)
25     a = LeakyReLU(dis_alphas[0])(a)
26
27     # Next 4 3D Convolutional Blocks
28     for i in range(dis_convolutional_blocks - 1):
29         a = Conv3D(filters=dis_filters[i + 1],
30                     kernel_size=dis_kernel_sizes[i + 1],
31                     strides=dis_strides[i + 1],
32                     padding=dis_paddings[i + 1])(a)
33         a = BatchNormalization()(a, training=True)
34         if dis_activations[i + 1] == 'leaky_relu':
35             a = LeakyReLU(dis_alphas[i + 1])(a)
36         elif dis_activations[i + 1] == 'sigmoid':
37             a = Activation(activation='sigmoid')(a)
38
39     dis_model = Model(inputs=[dis_input_layer], outputs=[a])
40     return dis_model

```

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

4. Jelaskan proses training 3D-GANs  
proses training 3D gan yaitu dengan melakukan epoch sebanyak yang ditentukan.
5. Jelaskan bagaimana melakukan settingan awal chapter 02 untuk memenuhi semua kebutuhan sebelum melanjutkan ke tahapan persiapan data.

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

6. Jelaskan tentang dataset yang digunakan, dari mulai tempat unduh, cara membuka dan melihat data. Sampai deskripsi dari isi dataset dengan detail penjelasan setiap folder/file yang membuat orang awam paham.

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. Jelaskan apa itu voxel dengan ilustrasi dan bahasa paling awam.

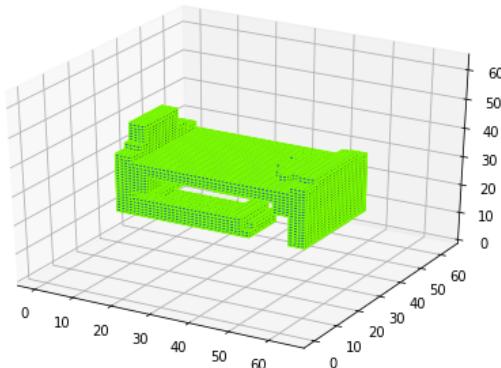
Volume pixel atau voxel adalah titik dalam ruang tiga dimensi. Sebuah voxel mendefinisikan posisi dengan tiga koordinat dalam arah x, y, dan z. Sebuah voxel adalah unit dasar untuk mewakili gambar 3D.

8. Visualisasikan dataset tersebut dalam tampilan visual plot, jelaskan cara melakukan visualisasinya.

```

1 import scipy.io as io
2 import matplotlib.pyplot as plt
3 import scipy.ndimage as nd
4 import numpy as np
5 from mpl_toolkits.mplot3d import Axes3D
6
7 voxels = io.loadmat("D:/New folder/KB3C/src/1174084/8/3 DShapeNets
8     /volumetric_data/bed/30/test/bed_000000000_1.mat")['instance'
9     ]
10 voxels = np.pad(voxels, (1, 1), 'constant', constant_values=(0,
11     0))
12 voxels = nd.zoom(voxels, (2, 2, 2), mode='constant', order=0)
13
14 fig = plt.figure()
15 ax = Axes3D(fig)
16 ax.voxels(voxels, edgecolor="chartreuse")
17 plt.show()
18 plt.savefig('bed')
```

langkah-langkah seperti ini : import library, load data file .mat dan lakukan read memakai matplotlib



**Gambar 8.110** visualisasi

9. buka file run.py jelaskan perbaris kode pada fungsi untuk membuat generator yaitu build generator.

```

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')(a)
28        a = BatchNormalization()(a, training=True)
29        a = Activation(activation=gen_activations[i + 1])(a)
30

```

```

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

```

Berfungsi untuk membuat generator yaitu dengan ketentuan gen sebagai variabel dan membuat fungsi atau variabel genmodel lalu dilakukan return.

10. jelaskan juga fungsi untuk membangun diskriminator pada fungsi build discriminator.

```

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    dis_convolutional_blocks = 5
16
17    dis_input_layer = Input(shape=dis_input_shape)
18
19    # The first 3D Convolutional block
20    a = Conv3D(filters=dis_filters[0],
21                kernel_size=dis_kernel_sizes[0],
22                strides=dis_strides[0],
23                padding=dis_paddings[0])(dis_input_layer)
24    # a = BatchNormalization()(a, training=True)
25    a = LeakyReLU(dis_alphas[0])(a)
26
27    # Next 4 3D Convolutional Blocks
28    for i in range(dis_convolutional_blocks - 1):
29        a = Conv3D(filters=dis_filters[i + 1],
30                    kernel_size=dis_kernel_sizes[i + 1],
31                    strides=dis_strides[i + 1],
32                    padding=dis_paddings[i + 1])(a)
33        a = BatchNormalization()(a, training=True)
34        if dis_activations[i + 1] == 'leaky_relu':
35            a = LeakyReLU(dis_alphas[i + 1])(a)
36        elif dis_activations[i + 1] == 'sigmoid':
37            a = Activation(activation='sigmoid')(a)
38
39    dis_model = Model(inputs=[dis_input_layer], outputs=[a])
40    return dis_model

```

Berfungsi untuk membangun diskriminator berfungsi untuk mendefenisikan seluruh gambar yang sudah di load generator sebagai gambar fake dan real.

11. Jelaskan kode program name = main

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

```

1
2 if __name__ == '__main__':
3     """
4     Specify Hyperparameters
5     """

```

12. Jelaskan kode program

```

1
2 object_name = "airplane"
3 data_dir = "D:/New folder/KB3C/src/1174084/8/3 DShapeNets/
4     volumetric_data/" \
5         "{}/30/train/*.mat".format(object_name)
6 gen_learning_rate = 0.0025
7 dis_learning_rate = 10e-5
8 beta = 0.5
9 batch_size = 1
10 z_size = 200
11 epochs = 1
12 MODE = "train"

```

Berfungsi untuk melakukan load dataset dengan ketentuan data yang hanya dalam folder chair pada data train.

13. Jelaskan kode program pembuatan dan kompilasi arsitektur

```

1 """
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)

```

Menggunakan Adam sebagai algoritma pengoptimalan dan binary\_crossentropy sebagai kerugian loss.

14. Jelaskan kode program membuat dan melakukan kompilasi model adversarial

```

1
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=[validity])
8     adversarial_model.compile(loss='binary_crossentropy',
optimizer=gen_optimizer)

```

Ini 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. Jelaskan kode program Ekstrak dan load data kursi dengan menggunakan fungsi getVoxelsFormat dan get3DImages

```

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

```

Berfungsi untuk melakukan load data pada dataset.

16. Jelaskan kode program instansiasi TensorBoard yang menambahkan generator dan diskriminator

```

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

```

Berfungsi untuk membuat tensorboard yang nantinya bisa diakses melalui localhost.

17. Jelaskan kode program, fungsi dari np reshape ones zeros

```

1     labels_real = np.reshape(np.ones((batch_size,)), (-1, 1, 1,
2     1, 1))
3     labels_fake = np.reshape(np.zeros((batch_size,)), (-1, 1, 1,
4     1, 1))

```

Berfungsi untuk melakukan reshape agar shape yang dihasilkan tidak terlalu besar. Dengan membuat variabel real dan fake.

18. Jelaskan kode, kenapa harus ada perulangan dalam meraih epoch

```

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

```

Berfungsi untuk melakukan training epoch, karena jika epoch semakin banyak maka kualitas training yang dihasilkan akan semakin baik.

19. Jelaskan kode program batches

```

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

```

Batch adalah jumlah file yang akan di training.

20. Jelaskan kode program pengambilan gambar dan noise

```

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

```

Berfungsi untuk membuat gambar bersih dari noise dan juga menyesuaikan shape.

21. Jelaskan kode program generator gambar palsu

```

1             # Next, generate volumes using the generate
2             network
3             gen_volumes = generator.predict_on_batch(z_sample)
4             """
5             Train the discriminator network
6             """

```

Berfungsi untuk membuat sample gambar palsu yang akan diteruskan ke diskriminator.

22. Jelaskan kode program training diskriminator dengan gambar palsu dari generator dan gambar asli

```

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

```

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. Jelaskan kode program training model adversarial yang terdapat generator dan diskriminato

```

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

```

Befungsi untuk melakukan print gloss untuk generator dan juga dloss untuk diskriminato.

24. Jelaskan kode program generate dan menyimpan gambar 3D setelah beberapa saat setiap epoch

```

1
2             # Every 10th mini-batch, generate volumes and
3             save them
4             if index % 10 == 0:
5                 z_sample2 = np.random.normal(0, 0.33, size=[batch_size,
6                 1, 1, 1, z_size]).astype(np.float32)
7                 generated_volumes = generator.predict(
8                 z_sample2, verbose=3)
9                 for i, generated_volume in enumerate(
10                generated_volumes[:5]):
11                     voxels = np.squeeze(generated_volume)
12                     voxels[voxels < 0.5] = 0.
13                     voxels[voxels >= 0.5] = 1.
14                     saveFromVoxels(voxels, "D:/New folder/
15 KB3C/src/1174084/8/3DShapeNets/voxelization/results/img_{}-{}-{}".
16 format(epoch, index, i))

```

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

25. Jelaskan kode program menyimpan average losses setiap epoch

```

1
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)

```

```

5      """
6      Save models
7      """
8

```

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

## 26. Jelaskan kode program menyimpan model

```

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

```

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

## 27. Jelaskan kode program testing model

```

1  if MODE == 'predict':
2      # Create models
3      generator = build_generator()
4      discriminator = build_discriminator()
5
6      # Load model weights
7      generator.load_weights(os.path.join("models", "generator_weights.h5"), True)
8      discriminator.load_weights(os.path.join("models", "discriminator_weights.h5"), True)
9
10     # Generate 3D models
11     z_sample = np.random.normal(0, 1, size=[batch_size, 1, 1,
12         1, z_size]).astype(np.float32)
13     generated_volumes = generator.predict(z_sample, verbose
14     =3)
15
16     for i, generated_volume in enumerate(generated_volumes
17         [:2]):
18         voxels = np.squeeze(generated_volume)
19         voxels[voxels < 0.5] = 0.
20         voxels[voxels >= 0.5] = 1.
21         saveFromVoxels(voxels, "D:/New folder/KB3C/src
22         /1174084/8/3DShapeNets/voxelization/results/gen_{}`".format(i)
23     )

```

Ini adalah tahap akhir untuk melakukan testing dari model yang telah dibuat dan buat model dari yang sudah di create sebelumnya yaitu generator dan diskriminato.

### 8.7.3 Bukti Tidak Plagiat



**Gambar 8.111** plagiarism

### 8.7.4 Link Video Youtube

<https://youtu.be/qETedUr2Le4>



# BAB 9

---

# CHAPTER 9

---

## 9.1 1174086 - Tia Nur Candida

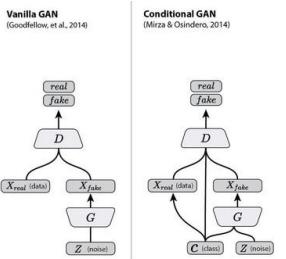
### 9.1.1 Teori

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

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 y 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,

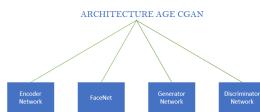
bagaimanapun, dapat digunakan untuk menghasilkan model multi-modal dengan kondisi yang berbeda untuk kategori yang berbeda.



**Gambar 9.1** Illustrasi Vanilla GAN dan cGAN

2. Jelaskan dengan ilustrasi gambar sendiri arsitektur dari AgecGAN.

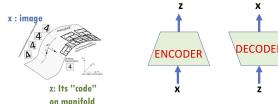
Arsitektur cGAN untuk penuaan wajah sedikit lebih rumit. AgecGan terdiri dari empat jaringan: Encoder, FaceNet, Jaringan Generator, dan jaringan diskriminasi. Dengan Encoder, kita belajar pemetaan invers gambar wajah masukan dan kondisi usia dengan vektor laten. FaceNet adalah jaringan pengenalan wajah yang mempelajari perbedaan antara gambar input  $x$  dan gambar yang direkonstruksi. Kami memiliki jaringan Generator, yang mengambil representasi tersembunyi yang terdiri dari gambar wajah dan vektor kondisi dan menghasilkan gambar. Jaringan diskriminasi adalah untuk mendiskriminasikan antara gambar nyata dan gambar palsu. Masalah dengan cGANs adalah bahwa mereka tidak dapat mempelajari tugas pemetaan terbalik masukan gambar  $x$  dengan atribut  $y$  ke vektor laten  $z$ . Solusi untuk masalah ini adalah dengan menggunakan jaringan Encoder. Kita dapat melatih jaringan encoder untuk memperkirakan pemetaan terbalik dari input Images  $x$ .



**Gambar 9.2** Illustrasi Arsitektur cGAN

3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari AgecGAN.
- Tujuan utama dari jaringan Encoder adalah untuk menghasilkan vektor laten dari gambar yang disediakan. Pada dasarnya, dibutuhkan gambar dimensi (64,

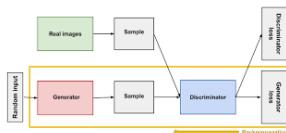
64, 3) dan mengubahnya menjadi vektor 100-dimensi. Jaringan Encoder adalah jaringan syaraf convolutional yang dalam. Jaringan berisi empat convolutional blok dan dua lapisan padat. Setiap blok convolutional berisi lapisan convolutional, lapisan normalisasi batch, dan fungsi aktivasi. Di setiap blok convolutional, setiap lapisan convolutional diikuti oleh lapisan normalisasi batch, kecuali lapisan convolutional pertama.



**Gambar 9.3** Illustrasi Network Encoder

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

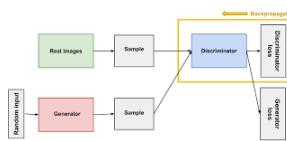
Tujuan utama dari generator adalah untuk menghasilkan gambar dari dimensi (64, 64, 3). Dibutuhkan vektor laten 100 dimensi dan beberapa informasi tambahan, y, dan mencoba untuk menghasilkan gambar yang realistik. Jaringan Generator adalah jaringan neural yang mendalam convolutional juga. Hal ini terdiri dari lapisan padat, upsampling, dan convolutional. Dibutuhkan dua nilai input: vektor kebisingan dan nilai pengkondisian. Nilai pengkondisian adalah informasi tambahan yang diberikan ke jaringan. Untuk Age-cGAN, ini akan menjadi usia.



**Gambar 9.4** Illustrasi Network Generator

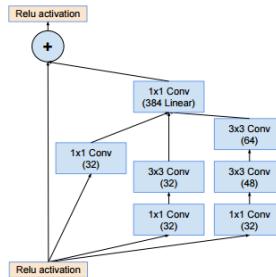
- Jelaskan dengan ilustrasi gambar sendiri arsitektur discriminator network dari Age-cGAN.

Tujuan utama dari jaringan diskriminator adalah 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** Illustrasi Discriminator Network

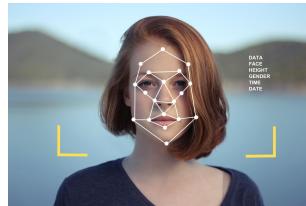
5. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model. pre-trained Inception-ResNet-2 network, sekali disediakan dengan gambar, mengembalikan yang sesuai embedding. Tertanam yang diekstrak untuk gambar asli dan gambar direkonstruksi dapat dihitung dengan menghitung jarak Euclidean dari yang tertanam.



**Gambar 9.6** Illustrasi Inception-ResNet-2 Model.

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

Tujuan utama dari jaringan pengenalan wajah adalah untuk mengenali identitas seseorang dalam gambar yang diberikan.



**Gambar 9.7** Illustrasi Face recognition network Age-cGAN.

7. . Sebutkan dan jelaskan serta di sertai contoh-contoh tahapan dari Age-cGAN

Age-cGAN memiliki beberapa tahapan pelatihan. Seperti disebutkan di bagian

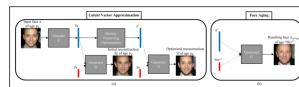
sebelumnya, Age-cGAN memiliki empat jaringan, yang dilatih dalam tiga tahap. Pelatihan AgecGAN terdiri dari tiga tahap:

- 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.

8. Berikan contoh perhitungan fungsi training objektif

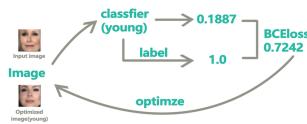
9. Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation

Perkiraan vektor laten awal 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 9.8** Illustrasi Initial latent vector approximation

10. Berikan contoh perhitungan latent vector optimization



**Gambar 9.9** Contoh Perhitungan Latent vector optimization

### 9.1.2 Praktek

1. Jelaskan bagaimana cara ekstrak file dataset Age-cGAN menggunakan google colab. Menggunakan Google Colab, dimana membuat notebooks baru, kemudian membuat ekstraksi file dari link dataset.

```
1 # In[1]: Ekstrak File]:
2 import tarfile
```

```

3 tf = tarfile.open("/content/drive/My Drive/Colab Notebooks/
      wiki_crop.tar")
4 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. Dibawah ini merupakan code untuk melakukan fungsi perhitungan usia.

```

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
7     # Load the list of all files
8     full_path = meta[dataset][0, 0]["full_path"][0]
9
10    # List of Matlab serial date numbers
11    dob = meta[dataset][0, 0]["dob"][0]
12
13    # List of years when photo was taken
14    photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
15
16    # Calculate age for all dobs
17    age = [calculate_age(photo_taken[i], dob[i]) for i in range(
18        len(dob))]
19
20    # Create a list of tuples containing a pair of an image path
21    # and age
22    images = []
23    age_list = []
24    for index, image_path in enumerate(full_path):
25        images.append(image_path[0])
26        age_list.append(age[index])
27
28    # Return a list of all images and respective age
29    return images, age_list

```

3. Jelaskan bagaimana kode program The Encoder Network bekerja dijelaskan dengan bahawa 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='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='same')(enc)
15 enc = BatchNormalization()(enc)
16 enc = LeakyReLU(alpha=0.2)(enc)
17
18 # 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 bahawa 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
8     latent_dims = 100
9     num_classes = 6
10
11    input_z_noise = Input(shape=(latent_dims,))
12    input_label = Input(shape=(num_classes,))
13
14    x = concatenate([input_z_noise, input_label])
15
16    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
30     x = UpSampling2D(size=(2, 2))(x)

```

5. Jelaskan bagaimana kode program The Discriminator Network bekerja dijelaskan dengan bahawa 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
8     input_shape = (64, 64, 3)
9     label_shape = (6,)
10    image_input = Input(shape=input_shape)
11    label_input = Input(shape=label_shape)
12
13    x = Conv2D(64, kernel_size=3, strides=2, padding='same')(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])
35    return model

```

6. Jelaskan bagaimana kode program Training cGAN bekerja dijelaskan dengan bahawa 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 bahawa awam dengan ilustrasi sederhana. Initial dan Latent Vector Approximation bekerja melakukan predicsi epoch yang telah di buat sebanyak 500 kali, dan nanti hasilnya ada di folder result.

```

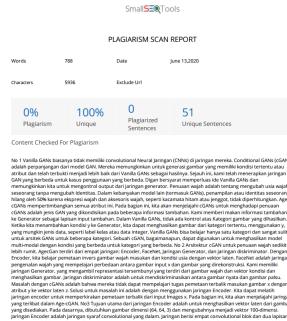
1 # In [7. Laten Vector]:
2     """
3         Train encoder
4     """
5
6     if TRAIN_ENCODER:
7         # Build and compile encoder
8         encoder = build_encoder()
9         encoder.compile(loss=euclidean_distance_loss, optimizer='adam')
10
11        # Load the generator network's weights
12        try:
13            generator.load_weights("generator.h5")
14        except Exception as e:
15            print("Error:", e)
16
17        z_i = np.random.normal(0, 1, size=(5000, z_shape))
18
19        y = np.random.randint(low=0, high=6, size=(5000,), dtype=
20                             np.int64)
21        num_classes = len(set(y))
22        y = np.reshape(np.array(y), [len(y), 1])
23        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]
35             y_batch = y[index * batch_size:(index + 1) *
36             batch_size]
37
38             generated_images = generator.predict_on_batch([
39             z_batch, y_batch])
40
41             # Train the encoder model
42             encoder_loss = encoder.train_on_batch(
43             generated_images, z_batch)
44             print("Encoder loss:", encoder_loss)
45
46             encoder_losses.append(encoder_loss)
47
48             # Write the encoder loss to Tensorboard
49             write_log(tensorboard, "encoder_loss", np.mean(
50             encoder_losses), epoch)
51
52             # Save the encoder model
53             encoder.save_weights("encoder.h5")

```

### 9.1.3 Bukti Tidak Plagiat



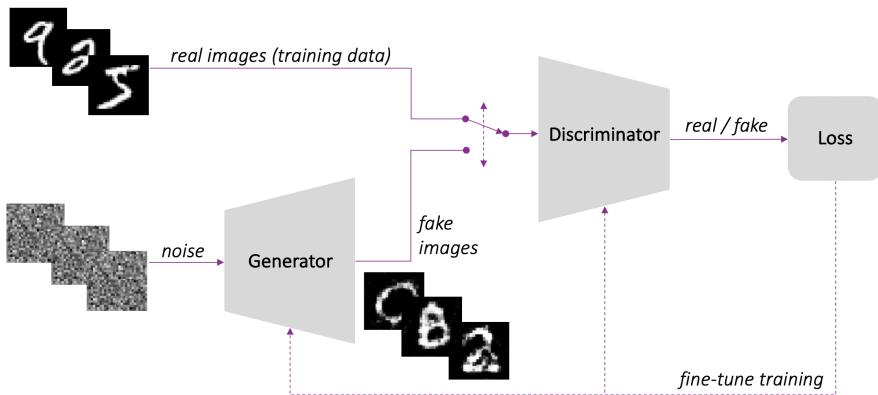
Gambar 9.10 Tidak Melakukan Plagiat Pada Ch 9

## 9.2 Muhammad Reza Syachrani - 1174084

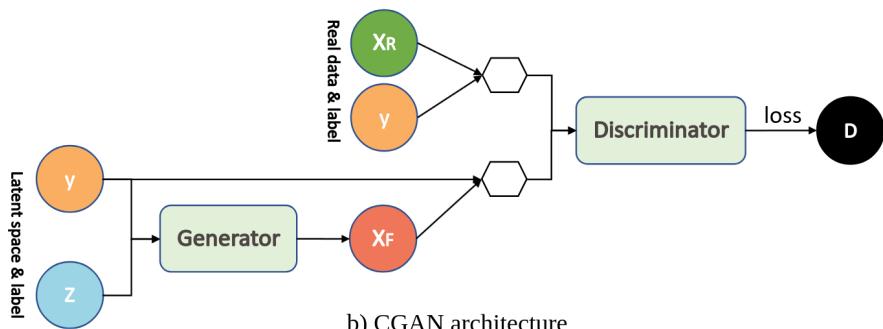
### 9.2.1 Teori

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

Vanilla GAN adalah tipe GAN paling sederhana. Di sini, Generator dan Diskriminatator adalah perceptron multi-layer sederhana. sedangkan pada CGAN (Conditional GAN) parameter ditambahkan 'y' ke Generator untuk menghasilkan data yang sesuai. Label juga dimasukkan ke dalam input ke Diskriminatator agar Diskriminatator membantu membedakan data nyata dari data yang dihasilkan palsu. Untuk ilustrasi, lihat gambar berikut:



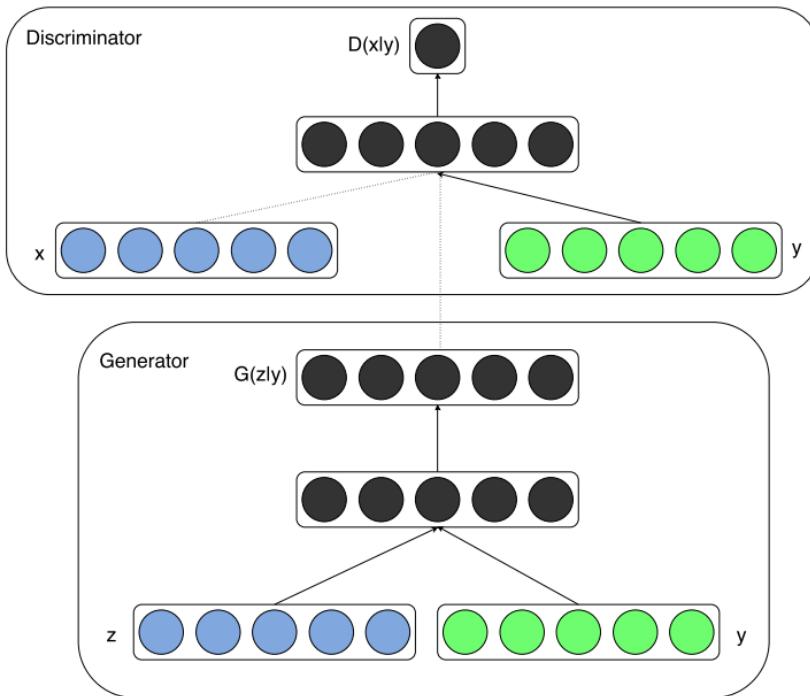
Gambar 9.11 Teori 1



b) CGAN architecture

Gambar 9.12 Teori 1

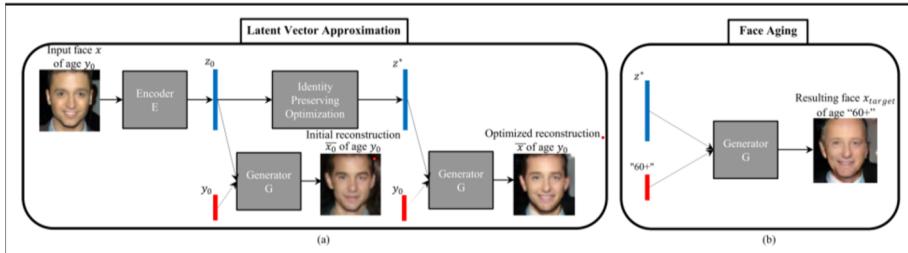
2. Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN.  
 Age-cGan terdiri dari empat jaringan: encoder, FaceNet, jaringan generator, dan jaringan diskriminator Untuk ilustrasi, lihat gambar berikut:



**Gambar 9.13** Teori 2

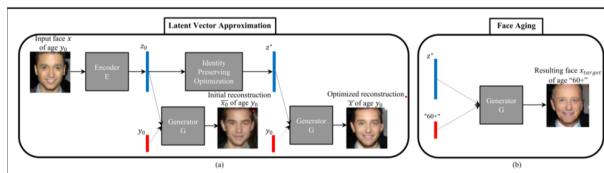
3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari Agec-GAN  
 jaringan encoder adalah untuk menghasilkan vektor laten dari gambar yang disediakan. Pada dasarnya, ini mengambil gambar dari dimensi (64, 64, 3) dan mengubahnya menjadi vektor 100 dimensi. Jaringan encoder adalah jaringan saraf convolutional yang mendalam.

Untuk ilustrasi, lihat gambar berikut:

**Gambar 9.14** Teori 3

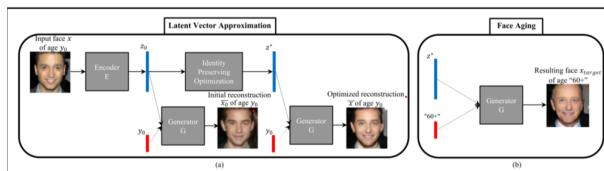
4. Jelaskan dengan ilustrasi gambar sendiri arsitektur generator network dari Agec-GAN.

jaringan generator adalah untuk menghasilkan gambar dengan dimensi (64, 64, 3). Dibutuhkan vektor laten 100 dimensi dan beberapa informasi tambahan,  $y$ , dan mencoba menghasilkan gambar yang realistik.

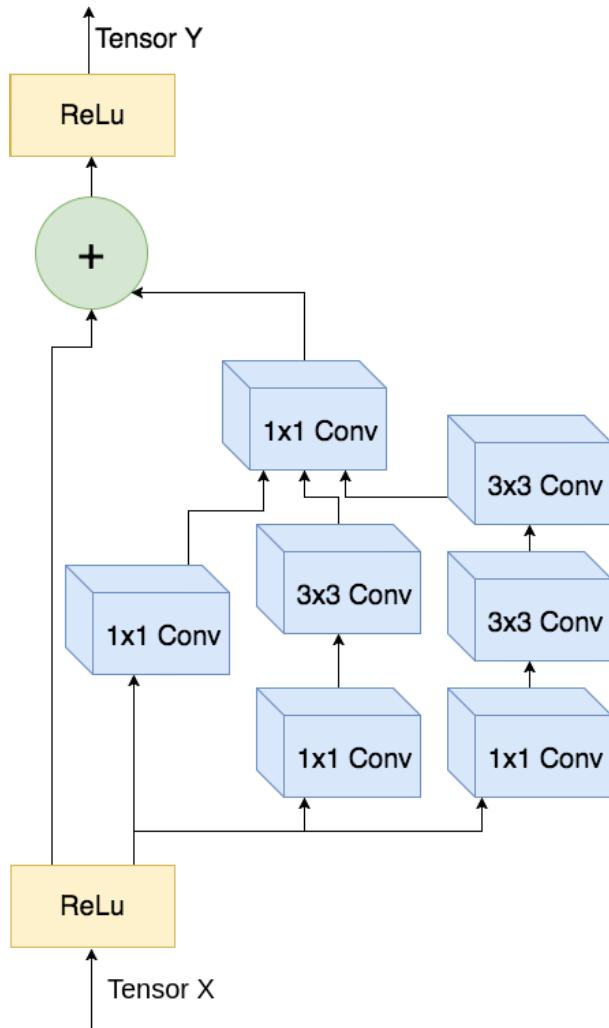
**Gambar 9.15** Teori 4

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

jaringan diskriminasi adalah untuk mengidentifikasi apakah gambar yang disediakan adalah palsu atau nyata. Ini dilakukan dengan melewatkannya melalui serangkaian lapisan downsampling dan beberapa lapisan klasifikasi. ilustrasi pada gambar berikut :

**Gambar 9.16** Teori 5

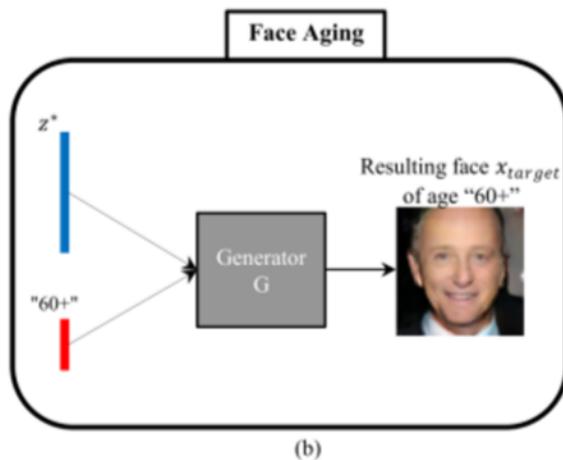
6. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model Pre-Trained Network atau Transfer Learning merupakan suatu metode penyelesaian yang memanfaatkan model yang sudah dilatih terhadap suatu dataset untuk menyelesaikan masalah dengan cara menggunakan sebagai starting point, memodifikasi dan mengupdate parameternya, sehingga sesuai dengan dataset yang baru.



**Gambar 9.17** Teori 6

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

Face recognition network Age-cGAN adalah untuk mengenali identitas seseorang dalam gambar yang diberikan. dengan mempelajari perbedaan antara gambar input  $x$  dan gambar yang direkonstruksi



**Gambar 9.18** Teori 7

8. Sebutkan dan jelaskan serta diertai contoh-contoh tahapan dari Age-cGAN  
Pelatihan Age-cGAN terdiri dari tiga tahap:
  - (a) Conditional GAN training: Pada tahap ini, kami melatih jaringan generator dan jaringan diskriminator.
  - (b) Initial latent vector approximation : Pada tahap ini, kami melatih jaringan pembuat enkode.
  - (c) Latent vector optimization: Pada tahap ini, kami mengoptimalkan 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.

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z}|\mathbf{y})))].$$

**Gambar 9.19** Teori 9

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

Latent vector approximation kemampuan untuk membuat gambar yang realistik dan tajam serta menghasilkan gambar wajah pada usia target.

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  $z$ .

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

**Gambar 9.20** Teori 11

### 9.2.2 Praktek

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

- Login ke google colab menggunakan akun google
- Mount google drive
- Lakukan proses unzip melalui notebook python di google colab, unzip pakai codingan
- Selesai

```
1 import tarfile
2 tf = tarfile.open("/content/drive/My Drive/Colab Notebooks/CH9/
    wiki_crop.tar")
3 tf.extractall(path="/content/drive/My Drive/Colab Notebooks/CH9/
    wiki_crop")
```

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

Dibawah ini merupakan code untuk melakukan fungsi perhitungan usia.

```
1 def load_data(wiki_dir, dataset='wiki'):
2     # Load the wiki.mat file
3     meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset
        )))
4
5     # Load the list of all files
6     full_path = meta[dataset][0, 0]["full_path"][0]
7
8     # List of Matlab serial date numbers
```

```

9     dob = meta[dataset][0, 0]["dob"][0]
10
11    # List of years when photo was taken
12    photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
13
14    # Calculate age for all dobs
15    age = [calculate_age(photo_taken[i], dob[i]) for i in range(
16        len(dob))]
17
18    # Create a list of tuples containing a pair of an image path
19    # and age
20    images = []
21    age_list = []
22    for index, image_path in enumerate(full_path):
23        images.append(image_path[0])
24        age_list.append(age[index])
25
26    # Return a list of all images and respective age
27    return images, age_list

```

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

```

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='same')(input_layer)
9     # enc = BatchNormalization()(enc)
10    enc = LeakyReLU(alpha=0.2)(enc)
11
12    # 2nd Convolutional Block
13    enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='same')(enc)
14    enc = BatchNormalization()(enc)
15    enc = LeakyReLU(alpha=0.2)(enc)
16
17    # 3rd Convolutional Block
18    enc = Conv2D(filters=128, kernel_size=5, strides=2, padding='same')(enc)
19    enc = BatchNormalization()(enc)
20    enc = LeakyReLU(alpha=0.2)(enc)
21
22    # 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

```

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

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

```

1 def build_generator():
2     """
3         Create a Generator Model with hyperparameters values defined
4         as follows
5     """
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
26    x = UpSampling2D(size=(2, 2))(x)
27    x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
28    x = BatchNormalization(momentum=0.8)(x)
29    x = LeakyReLU(alpha=0.2)(x)
30
31    x = UpSampling2D(size=(2, 2))(x)
32    x = Conv2D(filters=64, kernel_size=5, padding='same')(x)
33    x = BatchNormalization(momentum=0.8)(x)
34    x = LeakyReLU(alpha=0.2)(x)
35
36    x = UpSampling2D(size=(2, 2))(x)
37    x = Conv2D(filters=3, kernel_size=5, padding='same')(x)
38    x = Activation('tanh')(x)

```

```

37     model = Model(inputs=[input_z_noise, input_label], outputs=[x])
38     return model
39

```

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

Proses Discriminator untuk membedakan antara gambar asli dan gambar palsu.

```

1 def build_discriminator():
2     """
3         Create a Discriminator Model with hyperparameters values
4         defined as follows
5     """
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')(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

```

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

Proses Training cGAN ini dengan load file .mat pada dataset lalu epoch sebanyak 500 kali.

```

1 if __name__ == '__main__':
2     # Define hyperparameters
3     data_dir = "wiki_crop"
4     wiki_dir = os.path.join(data_dir, "wiki_crop")
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,
epsilon=10e-8)
adversarial_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2
=0.999, epsilon=10e-8)

```

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

Initial dan Latent Vector Approximation bekerja melakukan predksi epoch yang telah di buat sebanyak 500 kali, dan hasilnya disimpan pada folder result.

```

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([
33             z_batch, y_batch])
34
35             # Train the encoder model
36             encoder_loss = encoder.train_on_batch(
37                 generated_images, z_batch)
38                 print("Encoder loss:", encoder_loss)
39
40             encoder_losses.append(encoder_loss)
41
42             # Write the encoder loss to Tensorboard
43             write_log(tensorboard, "encoder_loss", np.mean(
44                 encoder_losses), epoch)
45
46             # Save the encoder model
47             encoder.save_weights("encoder.h5")

```

### 9.2.3 Bukti Tidak Plagiat



Gambar 9.21 plagiarism

### 9.2.4 Link Video Youtube

<https://youtu.be/fnc2PcVDkOU>

## 9.3 1174083 - Bakti Qilan Mufid

Chapter 9 - Conditional Generative Adversarial Network

### 9.3.1 Teori

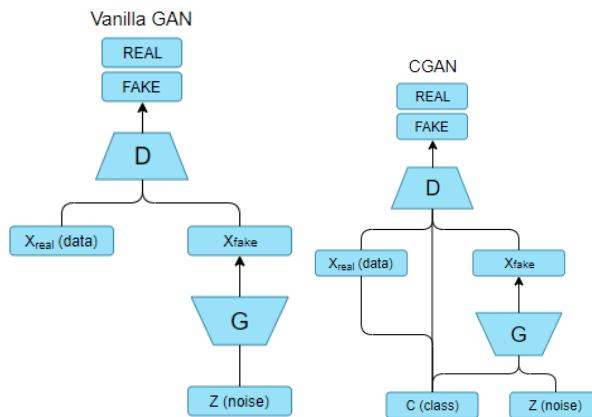
#### 9.3.1.1 Jelaskan dengan ilustrasi gambar sendiri apa perbedaan antara vanilla GAN dan cGAN.

1. Vanilla GAN merupakan tipe paling sederhana dari tipe-tipe yang ada pada GAN. Generator dan diskriminatator adalah perceptron multi-layer sederhana. Perceptron adalah salah satu metode Jaringan Syaraf Tiruan (JST) sederhana yang menggunakan algoritma training untuk melakukan klasifikasi secara linier. Perceptron digunakan untuk melakukan klasifikasi sederhana dan membagi data untuk menentukan data mana yang masuk dalam klasifikasi dan data

mana yang missclasifikasi (diluar klasifikasi). Vanilla GAN mengoptimalkan persamaan matematika menggunakan keturunan gradien stokastik(mempunyai unsur peluang).

2. cGAN(conditional GAN) merupakan tipe GAN yang dikondisikan pada beberapa tambahan informasi. Dalam GAN-projects informasi tambahan itu adalah  $y$  yang dimasukan ke generator sebagai lapisan input tambahan.

berikut gambaran antara Vanilla GAN dan cGAN

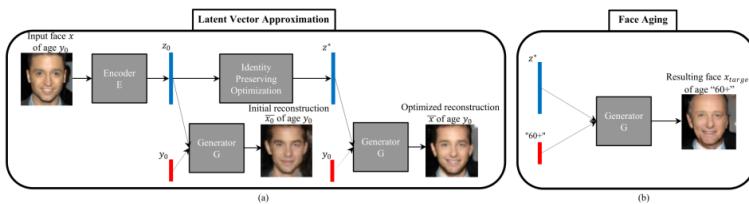


**Gambar 9.22** gambaran penjelasan no. 1

### 9.3.1.2 Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN.

Age-cGAN terdiri dari empat jaringan, yaitu:

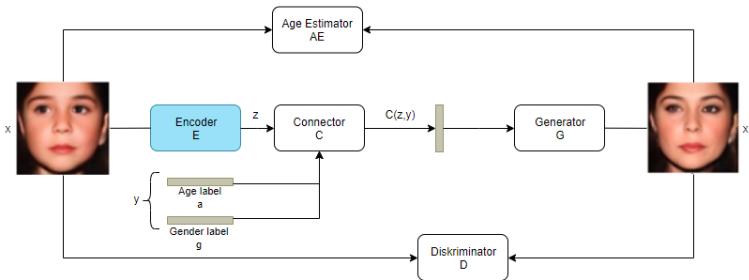
- Encoder, pemetaan terbalik dari gambar wajah input dan usia kondisi dengan vektor laten.
- FaceNet(face recognition network), mempelajari perbedaan antara gambar input  $x$  dan sebuah gambar yang direkonstruksi.
- Generator, membutuhkan sebuah representasi tersembunyi yang terdiri dari gambar wajah dan kondisi vektor dan akan menghasilkan gambar.
- Discriminator, melakukan diskriminasi antara gambar asli dan gambar palsu.



**Gambar 9.23** gambaran penjelasan no. 2

#### 9.3.1.3 Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari Age-cGAN.

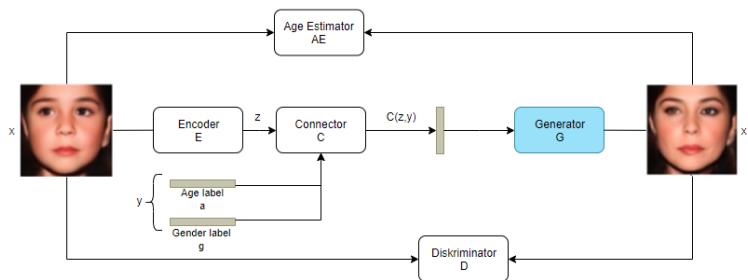
Tujuan utama dari encoder network adalah menghasilkan latent vector dari gambar yang sudah disediakan. Pada dasarnya, dibutuhkan gambar dengan dimensi (64, 64, 3) lalu akan diubah menjadi vektor 100-dimensi. encoder network juga merupakan deep CNN. encoder network berisi empat blok konvolusional dan dua dense layer. pada setiap blok konvolusional mengandung sebuah layer konvolusional, batch normalization, dan fungsi aktivasi. Di setiap blok konvolusional, setiap konvolusional lapisan diikuti oleh lapisan normalisasi batch, kecuali yang pertama.



**Gambar 9.24** gambaran penjelasan no. 3

#### 9.3.1.4 Jelaskan dengan ilustrasi gambar sendiri arsitektur generator network dari Age-cGAN.

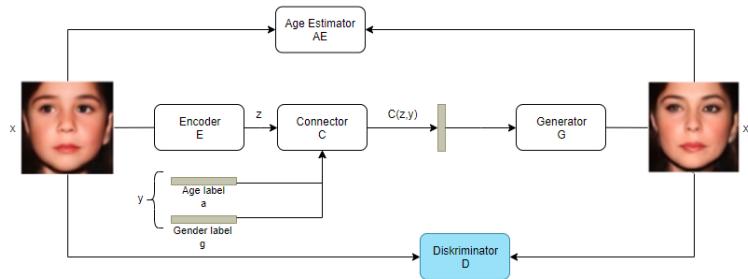
Tujuan utama dari generator network adalah menghasilkan gambar dengan dimensi (64,64,3). dibutuhkan vektor 100-dimensi dan beberapa tambahan informasi,  $y$ , dan mencoba menghasilkan gambar yang realistik. generator juga merupakan deep CNN, yang terbuat dari layer dense, upsampling, dan konvolusional layer. Dibutuhkan dua nilai input: sebuah noise vektor dan conditional value. Conditional value adalah informasi tambahan yang diberikan generator network. Untuk Age-cGAN, informasi tersebut merupakan usia.



**Gambar 9.25** gambaran penjelasan no. 4

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

Tujuan utama dari diskriminator network adalah untuk mengidentifikasi apakah gambar itu asli atau palsu, yang diketahui dengan cara mengolah gambar dalam down-sampling dan beberapa lapisan classification. atau bisa disebut memprediksi gambar tersebut palsu atau asli. Diskriminator network juga merupakan deep CNN. Diskriminator network juga terdiri dari beberapa konvolusional blok. setiap konvolusional blok memiliki konvolusional layer, kumpulan normalisasi layer, dan fungsi aktivasi. kecuali konvolusional blok pertama yang tidak memiliki kumpulan normalisasi layer.



**Gambar 9.26** gambaran penjelasan no. 5

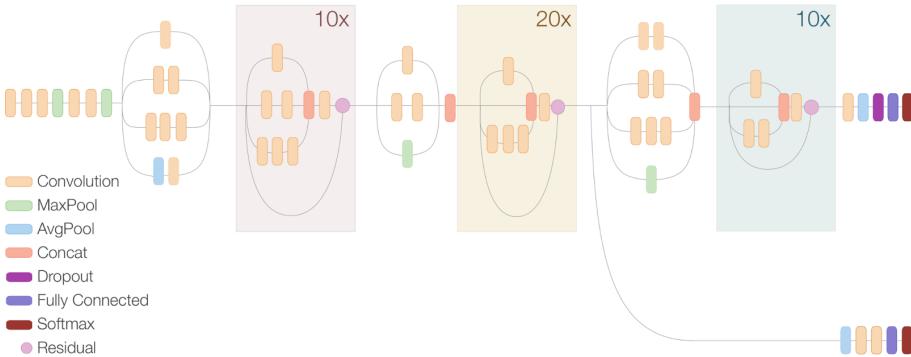
### 9.3.1.6 Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model.

Inception-ResNet-2 adalah jaringan saraf konvolusional(CNN) yang dilatih oleh lebih dari satu juta gambar dari database ImageNet(<http://www.image-net.org>). Jaringannya memiliki 164 lapisan dan dapat mengklasifikasikan gambar kedalam 1000 kategori objek, seperti keyboard, mouse, pensil, dan maca-macam binatang. hasilnya, jaringan tersebut sudah mempelajari representasi fitur yang kaya untuk berbagai gambar. Jaringan ini memiliki ukuran input gambar 299-by-299.

## Inception Resnet V2 Network



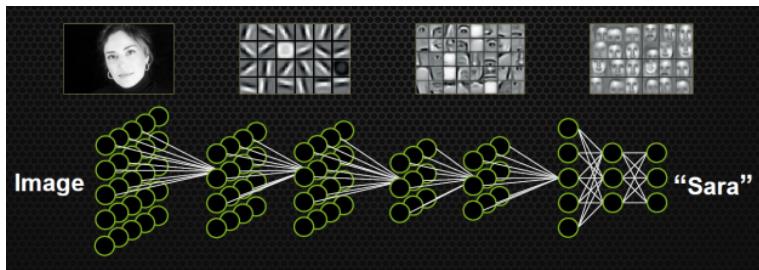
## Compressed View



**Gambar 9.27** gambaran penjelasan no. 6

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

Tujuan utama dari Face recognition adalah untuk mengenali identitas seseorang dalam gambar yang diberikan.



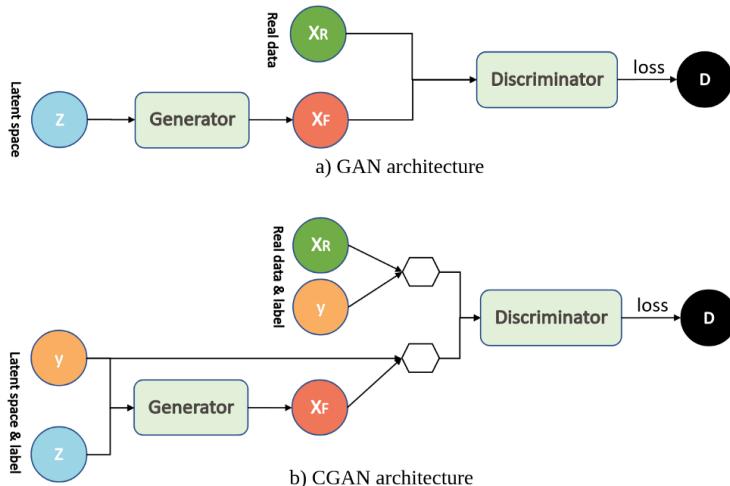
**Gambar 9.28** gambaran penjelasan no. 7

### 9.3.1.8 Sebutkan dan jelaskan serta di sertai contoh-contoh tahapan dari Age-cGAN

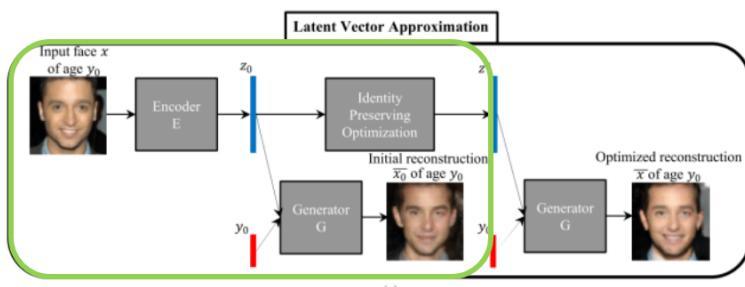
tahapan dari Age-cGAN ada tiga tahap, diantaranya:

1. Conditional GAN training, pada tahap ini Age-cGAN melatih generator dan diskriminator network.
2. Initial latent vector approximation, pada tahap ini Age-cGAN melatih encoder network.

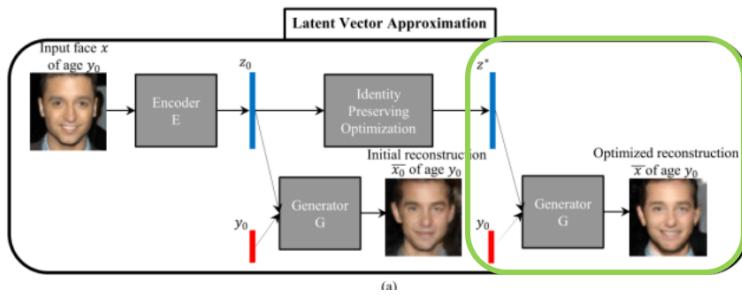
3. Latent vector optimization, pada tahap ini Age-cGAN mengoptimalkan encoder dan generator network.



**Gambar 9.29** gambaran penjelasan no. 8(Conditional GAN training)



**Gambar 9.30** gambaran penjelasan no. 8(Initial latent vector approximation)



**Gambar 9.31** gambaran penjelasan no. 8(Latent vector optimization)

#### 9.3.1.9 Berikan contoh perhitungan fungsi training objektif

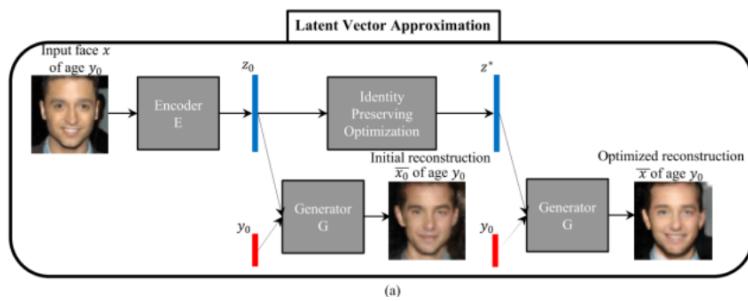
melatih jaringan cGAN melibatkan optimalisasi fungsi  $\nu(\Theta_G, \Theta_D)$ . melatih cGAN bisa dianggap game minimax(Algoritma minimax akan melakukan pengecekan pada seluruh kemungkinan yang ada, sehingga akan menghasilkan pohon permainan yang berisi semua kemungkinan permainan tersebut), dimana generator dan diskriminatore keduanya dilatih secara bersamaan. Dalam persamaan sebelumnya  $P_x$  merupakan parameter jaringan generator,  $\Theta_D$  merupakan parameter G dan D,  $\log D(x, y)$  merupakan loss dari model diskriminatore,  $\log(1 - D(G(z, \tilde{y}), \tilde{y}))$  merupakan loss dari model generator, dan  $P_{data}$  merupakan distribusi dari kemungkinan semua gambar.

$$\min_{\theta_G} \max_{\theta_D} \nu(\theta_G, \theta_D) = E_{x,y \sim p_{data}} [\log D(x, y)] + E_{z \sim p_z(z), \tilde{y} \sim p_{\tilde{y}}} [\log (1 - D(G(z, \tilde{y}), \tilde{y}))]$$

**Gambar 9.32** gambaran penjelasan no. 9

#### 9.3.1.10 Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation

Perkiraan latent vektor awal adalah metode yang memperkirakan latent vektor untuk mengoptimalkan rekonstruksi gambar wajah. Untuk itu maka diperlukan network encoder. Network encoder dilatih oleh gambar yang degenerate dan gambar yang nyata. Setelah dilatih, network encoder akan mulai menghasilkan vektor laten dari distribusi yang dipelajari. fungsinya untuk melatih network encoder sebagai Euclidean distance loss.



**Gambar 9.33** gambaran penjelasan no. 10

### 9.3.1.11 Berikan contoh perhitungan latent vector optimization

ketika latent vector optimization berlangsung, Age-cGAN mengoptimalkan network encoder dan network generator secara bersamaan. persamaan yang akan digunakan adalah seperti berikut:

$$z * IP = \operatorname{argmin}_z \|FR(x) - FR(\bar{x})\| L_2$$

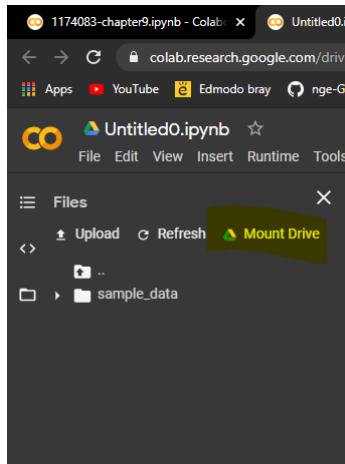
*FR* adalah jaringan pengenalan wajah. persamaan ini menunjukkan bahwa jarak Euclidean antara gambar asli dan gambar yang direkonstruksi harus minimal. pada tahap ini, Age-cGAN mencoba meminimalkan jarak dan memaksimalkan pelestarian identitas.

## 9.3.2 Praktek

### 9.3.2.1 Jelaskan bagaimana cara ekstrak file dataset Age-cGAN menggunakan google colab

Caranya cukup mudah, kita tinggal ikuti langkah-langkah berikut:

- Login terlebih dahulu ke google colab menggunakan akun google
- lalu buka link datasetnya (<https://drive.google.com/open?id=1NoV357ZvemE5dLCGySN>) lalu copy dataset ke akun google drive.
- Setelah di copy, buka google colab, buat notebook baru dan klik Mount Drive
- Lalu unzip file wiki\_crop.tar. maka dataset siap digunakan.



Gambar 9.34 gambar penjelasan praktek soal no.1

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

file dataset wiki\_crop.tar berisi 62.328 gambar dan sebuah file bernama wiki.mat yang menampung labelnya. Library spicy.io mempunyai method yang bernama loadmat, dimana method tersebut sangat memudahkan ketika kita ingin me-load file .mat. lalu untuk me-load datanya, kita bisa menggunakan kode berikut:

```

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

```

```

24 # Return a list of all images and respective age
25     return images, age_list

```

variabel photo\_taken adalah daftar tahun dan dob adalah tanggal serial Matlab nomor untuk gambar yang sesuai dalam daftar. kita bisa mengukur umur seseorang dari seri tanggal dan tahun gambar tersebut diambil, dengan menggunakan kode berikut:

```

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

```

### 9.3.2.3 Jelaskan bagaimana kode program The Encoder Network bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana

Berikut merupakan langkah-langkah encoder network bekerja:

1. Mulai dengan membuat layer input

```

1 input_layer = Input(shape=(64, 64, 3))

```

2. Lalu tambahkan blok konvolusi pertama, yang berisi lapisan konvolusi 2D dan fungsi aktivasi dengan konfigurasi sebagai berikut:

- Filters: 32
- Kernel size: 5
- Strides: 2
- Padding: same
- Activation: LeakyReLU with alpha equal to 0.2:

```

1 # 1st Convolutional Block
2 enc = Conv2D(filters=32, kernel_size=5, strides=2, padding='same')(input_layer)
3 # enc = BatchNormalization()(enc)
4 enc = LeakyReLU(alpha=0.2)(enc)

```

3. Selanjutnya, tambahkan tiga blok konvolusi lagi, yang masing-masing berisi lapisan konvolusi 2D dan diikuti oleh lapisan normalisasi batch serta fungsi aktivasi, seperti pada konfigurasi berikut:

- Filters: 64, 128, 256
- Kernel size: 5, 5, 5
- Strides: 2, 2, 2
- Padding: same, same, same

- Batch normalization: Each convolutional layer is followed by a batch normalization layer
- Activations: LealyReLU, LeakyReLU, LeakyReLU with alpha equal to 0.2:

```

1      # 2nd Convolutional Block
2      enc = Conv2D(filters=64, kernel_size=5, strides=2,
3          padding='same')(enc)
4      enc = BatchNormalization()(enc)
5      enc = LeakyReLU(alpha=0.2)(enc)

6      # 3rd Convolutional Block
7      enc = Conv2D(filters=128, kernel_size=5, strides=2,
8          padding='same')(enc)
9      enc = BatchNormalization()(enc)
10     enc = LeakyReLU(alpha=0.2)(enc)

11     # 4th Convolutional Block
12     enc = Conv2D(filters=256, kernel_size=5, strides=2,
13         padding='same')(enc)
14     enc = BatchNormalization()(enc)
15     enc = LeakyReLU(alpha=0.2)(enc)

```

4. Selanjutnya meratakan(flatten) output dari blok konvolusi yang terakhir, sebagai berikut:

```

1      # Flatten layer
2      enc = Flatten()(enc)

```

5. Selanjutnya tambahkan dense layer(yang terhubung secara penuh) dan dikikuti dengan batch normaisasi layer serta fungsi aktivasi, dengan konfigurasi berikut:

- Units (nodes): 2,096
- Batch normalization: Yes
- Activation: LeakyReLU with alpha equal to 0.2:

```

1      # 1st Fully Connected Layer
2      enc = Dense(4096)(enc)
3      enc = BatchNormalization()(enc)
4      enc = LeakyReLU(alpha=0.2)(enc)

```

6. Selanjutnya, tambahkan dense layer kedua(yang terhubung secara penuh), dengan konfigurasi berikut:

- Units (nodes): 100
- Activation: None:

```

1      # Second Fully Connected Layer
2      enc = Dense(100)(enc)

```

7. Terakhir, buat model Keras dan tentukan input serta output untuk network encoder.

```

1      model = Model(inputs=[input_layer], outputs=[enc])

```

### 9.3.2.4 Jelaskan bagaimana kode program The Generator Network bekerja di-jelaskan dengan bahasa awam dengan ilustrasi sederhana

Berikut merupakan langkah-langkah generator network bekerja:

1. Mulai dengan membuat dua lapisan input ke generator network.

```

1 latent_dims = 100
2 num_classes = 6
3
4 input_z_noise = Input(shape=(latent_dims ,)) #Input layer for
vector z
5 input_label = Input(shape=(num_classes ,)) #Input layer for
conditioning variable

```

2. Selanjutnya, gabungkan input sepanjang dimensi channel, seperti berikut:

```
1 x = concatenate([input_z_noise , input_label])
```

3. Lalu tambahkan blok dense (terhubung secara penuh) dengan konfigurasi berikut:

- Units (nodes): 2,048
- Input dimension: 106
- Activation: LeakyReLU with alpha equal to 0.2
- Dropout: 0.2:

```

1 x = Dense(2048 , input_dim=latent_dims + num_classes )(x)
2 x = LeakyReLU(alpha=0.2)(x)
3 x = Dropout(0.2)(x)

```

4. Selanjutnya tambahkan blok dense kedua, dengan konfigurasi berikut:

- Units (nodes): 16,384
- Batch normalization: Yes
- Activation: LeakyReLU with alpha equal to 0.2
- Dropout: 0.2:

```

1 x = Dense(256 * 8 * 8)(x)
2 x = BatchNormalization()(x)
3 x = LeakyReLU(alpha=0.2)(x)
4 x = Dropout(0.2)(x)

```

5. Selanjutnya bentuk kembali output dari lapisan dense terakhir ke 3D tensor dengan dimensi(8,8,256):

```
1 x = Reshape((8 , 8 , 256))(x)
```

6. Selanjutnya tambahkan blok upsampling yang berisi lapisan upsampling diikuti oleh lapisan konvolusi 2D serta batch normalisasi dengan konfigurasi berikut:

- Upsampling size: (2, 2)
- Filters: 128
- Kernel size: 5
- Padding: same
- Batch normalization: Yes, with momentum equal to 0.8
- Activation: LeakyReLU with alpha equal to 0.2:

```

1   x = UpSampling2D(size=(2, 2))(x)
2   x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
3   x = BatchNormalization(momentum=0.8)(x)
4   x = LeakyReLU(alpha=0.2)(x)

```

7. Selanjutnya, tambahkan blok upsampling lain(mirip dengan lapisan sebelumnya), seperti pada kode berikut, dan konfigurasinya mirip seperti blok sebelumnya, kecuali jumlah filter yang digunakan dalam lapisan konvolusi yang asalnya 128 menjadi 64:

```

1   x = Reshape((8, 8, 256))(x)
2
3   x = UpSampling2D(size=(2, 2))(x)
4   x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
5   x = BatchNormalization(momentum=0.8)(x)
6   x = LeakyReLU(alpha=0.2)(x)
7
8   x = UpSampling2D(size=(2, 2))(x)
9   x = Conv2D(filters=64, kernel_size=5, padding='same')(x)
10  x = BatchNormalization(momentum=0.8)(x)
11  x = LeakyReLU(alpha=0.2)(x)

```

8. Selanjutnya tambahkan blok upsampling terakhir. konfigurasinya mirip dengan lapisan sebelumnya, kecuali tiga lapisan filter akan digunakan dan batch normalisasi tidak digunakan:

```

1   x = UpSampling2D(size=(2, 2))(x)
2   x = Conv2D(filters=3, kernel_size=5, padding='same')(x)
3   x = Activation('tanh')(x)

```

9. Terakhir, buat model Keras dan tentukan input serta output untuk generator network:

```

1   model = Model(inputs=[input_z_noise, input_label], outputs=[x])

```

### 9.3.2.5 Jelaskan bagaimana kode program The Discriminator Network bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana

Berikut merupakan langkah-langkah diskriminator network bekerja:

1. Mulailah dengan membuat dua lapisan input, karena pada project ini diskriminator akan memproses dua input:

```

1  input_shape = (64, 64, 3) #Input image shape
2  label_shape = (6,) #Input conditioning variable shape
3  image_input = Input(shape=input_shape) #Two input layers
4  label_input = Input(shape=label_shape) #Two input layers

```

2. Selanjutnya, tambahkan blok konvolusi 2D (fungsi Conv2D + aktivasi) dengan konfigurasi berikut:

- Filters = 64
- Kernel size: 3
- Strides: 2
- Padding: same
- Activation: LeakyReLU with alpha equal to 0.2:

```

1  x = Conv2D(64, kernel_size=3, strides=2, padding='same')(image_input)
2  x = LeakyReLU(alpha=0.2)(x)

```

3. Selanjutnya perluas label\_input sehingga memiliki bentuk (32,32,6):

```
1  label_input1 = Lambda(expand_label_input)(label_input)
```

fungsi expand\_label\_input sebagai berikut:

```

1 def expand_label_input(x):
2     x = K.expand_dims(x, axis=1)
3     x = K.expand_dims(x, axis=1)
4     x = K.tile(x, [1, 32, 32, 1])
5     return x

```

4. Selanjutnya gabungkan tensor label yang telah ditransformasi dan output terakhir dari lapisan konvolusi, seperti berikut:

```
1  x = concatenate([x, label_input1], axis=3)
```

5. Tambahkan blok konvolusi (lapisan konvolusi 2D + batch normalisasi + fungsi aktivasi) dengan konfigurasi berikut:

- Filters: 128
- Kernel size: 3
- Strides: 2
- Padding: same
- Batch normalization: Yes
- Activation: LeakyReLU with alpha equal to 0.2:

```

1 x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
2 x = BatchNormalization()(x)
3 x = LeakyReLU(alpha=0.2)(x)

```

6. Selanjutnya tambahkan dua blok konvolusi lagi, seperti berikut:

```

1 x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
2 x = BatchNormalization()(x)
3 x = LeakyReLU(alpha=0.2)(x)
4
5 x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
6 x = BatchNormalization()(x)
7 x = LeakyReLU(alpha=0.2)(x)

```

7. Lalu tambahkan lapisan flatten

```
1 x = Flatten()(x)
```

8. lalu tambahkan lapisan dense(lapisan pengklasifikasian) yang menghasilkan probabilitas:

```
1 x = Dense(1, activation='sigmoid')(x)
```

9. Terakhir, buat model Keras dan tentukan input serta output untuk diskriminator network.

```
1 model = Model(inputs=[image_input, label_input], outputs=[x])
```

### **9.3.2.6 Jelaskan bagaimana kode program Training cGAN bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana**

Berikut adalah langkah-langkah melatih cGAN:

1. Mulailah dengan menentukan parameter yang diperlukan untuk pelatihan:

```

1 # Define hyperparameters
2 data_dir = "data"
3 wiki_dir = os.path.join(data_dir, "wiki_crop1")
4 epochs = 500
5 batch_size = 2
6 image_shape = (64, 64, 3)
7 z_shape = 100
8 TRAIN_GAN = True
9 TRAIN_ENCODER = False
10 TRAIN_GAN_WITH_FR = False
11 fr_image_shape = (192, 192, 3)

```

2. Selanjutnya, tentukan optimisator untuk pelatihan. Kali ini kita akan menggunakan Adam optimizer yang terdapat di Keras. Untuk inisiasinya seperti berikut

```

1 # Define optimizers
2 # Optimizer for the discriminator network
3 dis_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
4 epsilon=10e-8)

```

```

4 # Optimizer for the generator network
5     gen_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
6                           epsilon=10e-8)
7 # Optimizer for the adversarial network
8     adversarial_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2
9                           =0.999, epsilon=10e-8)
```

gunakan equal rate sama dengan 0,0002, nilai beta\_1 sama dengan 0,5, nilai beta\_2 sama dengan 0,999, dan nilai epsilon sama dengan 10e-8 semua optimizer.

- Selanjutnya, muat dan kompilasi generator network dan diskriminator network. Dalam Keras kita harus mengkompilasi network terlebih dahulu sebelum melatihnya.

```

1 # Build and compile the discriminator network
2 discriminator = build_discriminator()
3 discriminator.compile(loss=['binary_crossentropy'], optimizer=
4                         =dis_optimizer)
5
6 # Build and compile the generator network
7 generator = build_generator()
8 generator.compile(loss=['binary_crossentropy'], optimizer=
9                         =gen_optimizer)
```

- Selanjutnya, buat dan kompilasi model adversial, sebagai berikut:

```

1 # Build and compile the adversarial model
2 discriminator.trainable = False
3 input_z_noise = Input(shape=(100,))
4 input_label = Input(shape=(6,))
5 recons_images = generator([input_z_noise, input_label])
6 valid = discriminator([recons_images, input_label])
7 adversarial_model = Model(inputs=[input_z_noise, input_label],
8                             outputs=[valid])
9 adversarial_model.compile(loss=['binary_crossentropy'],
10                           optimizer=gen_optimizer)
```

- Selanjutnya tambahkan TensorBoard untuk menyimpan losses, seperti berikut:

```

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

- Selanjutnya load semua gambar menggunakan fungsi load\_data yang didefinisikan pada bagian mempersiapkan data:

```

1 images, age_list = load_data(wiki_dir=wiki_dir, dataset="wiki")
```

- Selanjutnya konversikan nilai numerik usia ke kategori usia, seperti berikut:

```

1 # Convert age to category
```

Definisi dari fungsi age\_to\_category seperti kode berikut:

```

1 def age_to_category(age_list):
2     age_list1 = []
3
4     for age in age_list:
5         if 0 < age <= 18:
6             age_category = 0
7         elif 18 < age <= 29:
8             age_category = 1
9         elif 29 < age <= 39:
10            age_category = 2
11        elif 39 < age <= 49:
12            age_category = 3
13        elif 49 < age <= 59:
14            age_category = 4
15        elif age >= 60:
16            age_category = 5
17
18        age_list1.append(age_category)
19
20    return age_list1

```

8. Selanjutnya load semua gambar dan buat ndarray yang berisi semua gambar:

```

1 # Read all images and create an ndarray
2 loaded_images = load_images(wiki_dir, images, (image_shape
[0], image_shape[1]))

```

Definisi dari fungsi load\_data seperti kode berikut:

```

1 def load_images(data_dir, image_paths, image_shape):
2     images = None
3
4     for i, image_path in enumerate(image_paths):
5         print()
6         try:
7             # Load image
8             loaded_image = image.load_img(os.path.join(data_dir,
image_path), target_size=image_shape)
9
10            # Convert PIL image to numpy ndarray
11            loaded_image = image.img_to_array(loaded_image)
12
13            # Add another dimension (Add batch dimension)
14            loaded_image = np.expand_dims(loaded_image, axis=0)
15
16            # Concatenate all images into one tensor
17            if images is None:
18                images = loaded_image
19            else:
20                images = np.concatenate([images, loaded_image],
axis=0)
21            except Exception as e:
22                print("Error:", i, e)
23
24    return images

```

9. Selanjutnya buat perulangan(for) berdasarkan jumlah epoch, seperti berikut:

```

1   for epoch in range(epochs):
2       print("Epoch:{}".format(epoch))
3
4       gen_losses = []
5       dis_losses = []
6
7       number_of_batches = int(len.loaded_images) /
batch_size)
8       print("Number of batches:", number_of_batches)

```

10. Selanjutnya buat perulangan yang lain didalam perulangan epoch berdasarkan num\_batches, seperti berikut:

```

1   for index in range(number_of_batches):
2       print("Batch:{}".format(index + 1))

```

11. Selanjutnya buat sample dari batch gambar dari dataset yang asli dan batch dari one-hot encoded age vector:

```

1   images_batch = loaded_images[index * batch_size:(
index + 1) * batch_size]
2   images_batch = images_batch / 127.5 - 1.0
3   images_batch = images_batch.astype(np.float32)
4
5   y_batch = y[index * batch_size:(index + 1) *
batch_size]

```

bentuk dari image\_batch (batch\_size, 64, 64, 3) dan bentuk dari y\_batch (batch\_size, 6).

12. Selanjutnya buat sampel batch noise vektor dari distribusi Gaussian, seperti berikut:

```

1   z_noise = np.random.normal(0, 1, size=(batch_size
, z_shape))

```

13. Selanjutnya buat gambar palsu(fake) menggunakan generator network. perlu diingat bahwa kita belum melatih network generator.

```

1   # Generate fake images
2   initial_recon_images = generator.predict_on_batch(
[ z_noise , y_batch ])

```

14. Sekarang latih network diskriminator pada gambar yang asli dan juga gambar yang palsu

```

1   d_loss_real = discriminator.train_on_batch([
images_batch , y_batch ], real_labels)
2   d_loss_fake = discriminator.train_on_batch([
initial_recon_images , y_batch ], fake_labels)

```

15. Lalu latih adversial network dan mempause diskriminator network. atau kita hanya akan melatih generator network.

```

1      # Again sample a batch of noise vectors from a
2      Gaussian(normal) distribution
3      z_noise2 = np.random.normal(0, 1, size=(batch_size, z.shape))
4          # Samples a batch of random age values
5          random_labels = np.random.randint(0, 6,
6          batch_size).reshape(-1, 1)
7          # Convert the random age values to one-hot
8          encoders
9          random_labels = to_categorical(random_labels, 6)
10         # Train the generator network
11         g_loss = adversarial_model.train_on_batch([
12             z_noise2, random_labels], [1] * batch_size)

```

16. Selanjutnya hitung dan cetak lossesnya:

```

1     print("g_loss:{}".format(g_loss))
2     # Add losses to their respective lists
3     gen_losses.append(g_loss)
4     dis_losses.append(d_loss)

```

17. Selanjutnya tulis losses ke TensorBoard untuk divisualisasikan

```

1      # Write losses to Tensorboard
2      write_log(tensorboard, 'g_loss', np.mean(gen_losses),
3      epoch)
4      write_log(tensorboard, 'd_loss', np.mean(dis_losses),
5      epoch)

```

18. Buat sampel dan simpan gambar setiap 10 epoch, seperti berikut:

```

1 if epoch % 10 == 0:
2     images_batch = loaded_images[0:batch_size]
3     images_batch = images_batch / 127.5 - 1.0
4     images_batch = images_batch.astype(np.float32)
5
6     y_batch = y[0:batch_size]
7     z_noise = np.random.normal(0, 1, size=(batch_size,
8     , z.shape))
9
10    gen_images = generator.predict_on_batch([z_noise,
11        y_batch])
12
13    for i, img in enumerate(gen_images[:5]):
14        save_rgb_img(img, path="results/img_{}_{}.png"
15        .format(epoch, i))

```

Letakan blok kode sebelumnya didalam perulangan epoch. setelah setiap 10 epoch, maka akan menghasilkan batch dari gambar palsu dan akan disimpan ke direktori results. untuk fungsi save\_rgb\_img adalah sebagai fungsi utilitas dan didefinisikan seperti berikut:

```

1 def save_rgb_img(img, path):
2     """
3     Save an rgb image
4     """
5     fig = plt.figure()
6     ax = fig.add_subplot(1, 1, 1)
7     ax.imshow(img)
8     ax.axis("off")
9     ax.set_title("Image")
10
11 plt.savefig(path)
12 plt.close()

```

19. Terakhir simpan kedua model dengan menambahkan baris berikut:

```

1 # Save improved weights for both of the networks
2 generator.save_weights("generator_optimized.h5")
3 encoder.save_weights("encoder_optimized.h5")

```

### **9.3.2.7 Jelaskan bagaimana kode program Initial dan latent vector approximation bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana**

Seperti yang sudah diketahui bahwa cGAN tidak belajar untuk memetakan balik dari gambar ke vektor latent. oleh karena itu, encoder yang akan mempelajari pemetaan balik ini dan mampu menghasilkan vektor laten yang dapat digunakan untuk menghasilkan gambar wajah pada usia yang ditargetkan.

Kita telah mendefinisikan hyperparameter yang dibutuhkan dalam pelatihan. Dan untuk langkah-langkahnya seperti berikut:

1. Mulailah dengan membuat network encoder. tambahkan kode berikut untuk membuat dan mengompilasi network encoder.

```

1 # Build and compile encoder
2 encoder = build_encoder()
3 encoder.compile(loss=euclidean_distance_loss, optimizer='adam')

```

Jika belum mendefinisikan euclidean\_distance\_loss, maka definisikan terlebih dahulu seperti pada kode berikut:

```

1 def euclidean_distance_loss(y_true, y_pred):
2     """
3     Euclidean distance loss
4     """
5     return K.sqrt(K.sum(K.square(y_pred - y_true), axis=-1))

```

2. Selanjutnya load network generator, seperti berikut:

```
1 generator.load_weights("generator.h5")
```

3. Selanjutnya buat sampel batc vektor latent, seperti berikut:

```
1 z_i = np.random.normal(0, 1, size=(5000, z_shape))
```

4. Selanjutnya buat sampel batch acak angka usia dan konversikan usia angka acak tadi kedalam one-hot encoded vector, seperti berikut:

```

1     y = np.random.randint(low=0, high=6, size=(5000,), dtype=
2         np.int64)
3     num_classes = len(set(y))
4     y = np.reshape(np.array(y), [len(y), 1])
5     y = to_categorical(y, num_classes=num_classes)

```

5. Selanjutnya tambahkan perulangan epoch dan batch dari langkah-langkah. seperti berikut:

```

1     for epoch in range(epochs):
2         print("Epoch:", epoch)
3
4         encoder_losses = []
5
6         number_of_batches = int(z_i.shape[0] / batch_size)
7         print("Number of batches:", number_of_batches)
8         for index in range(number_of_batches):
9             print("Batch:", index + 1)

```

6. Sekarang buat sampel batch vektor latent dan batch one-hot encoded vector dari 1000 sampel, seperti berikut:

```

1     z_batch = z_i[index * batch_size:(index + 1) *
2                     batch_size]
3     y_batch = y[index * batch_size:(index + 1) *
4                     batch_size]

```

7. Selanjutnya hasilkan gambar palsu menggunakan network generator yang sudah dilatih:

```

1     generated_images = generator.predict_on_batch([
2         z_batch, y_batch])

```

8. Lalu latih network encoder pada gambar yang dihasilkan oleh network generator:

```

1     # Train the encoder model
2     encoder_loss = encoder.train_on_batch(
3         generated_images, z_batch)

```

9. Selanjutnya tulis loss encoder ke TensorBoard pada setiap epoch.

```

1     # Write the encoder loss to Tensorboard
2     write_log(tensorboard, "encoder_loss", np.mean(
3         encoder_losses), epoch)

```

10. dan kita perlu menyimpan network encoder yang sudah dilatih. simpan model encoder dengan kode berikut:

```

1     # Save the encoder model
2     encoder.save_weights("encoder.h5")

```

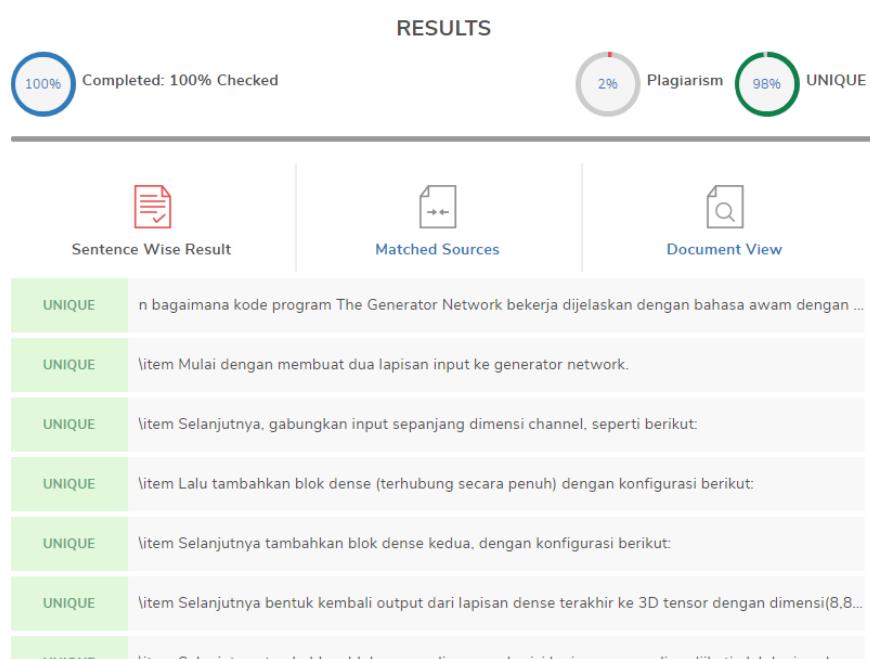
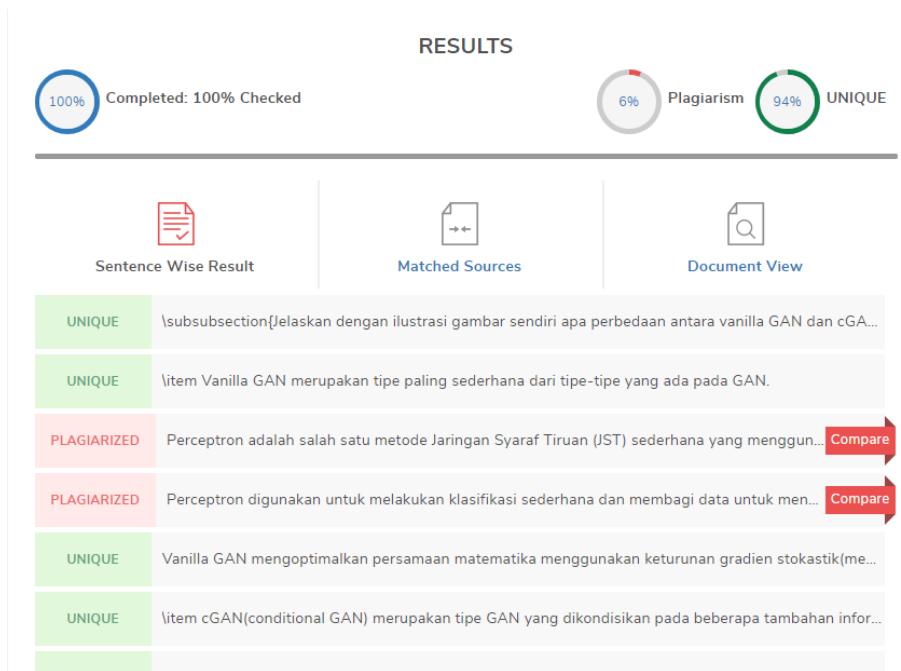
### 9.3.3 Penanganan Error

#### 9.3.3.1 *Terjadi error*

#### 9.3.3.2 *Solusi*



### 9.3.4 Bukti Tidak Plagiat



Gambar 9.35 Bukti tidak plagiat

### 9.3.5 Link Youtube

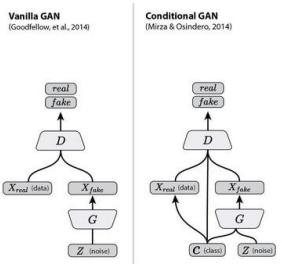
<https://bit.ly/baktiListVideo>

## 9.4 1174066 - D.Irga B. Naufal Fakhri

### 9.4.1 Teori

**9.4.1.1 Jelaskan dengan ilustrasi gambar sendiri apa perbedaan antara vanilla GAN dan cGAN.** 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 y 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.

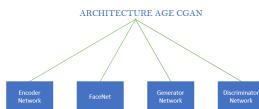


**Gambar 9.36** Illustrasi Vanilla GAN dan cGAN

**9.4.1.2 Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN.**

Arsitektur cGAN untuk penuaan wajah sedikit lebih rumit. AgecGan terdiri dari empat jaringan: Encoder, FaceNet, Jaringan Generator, dan jaringan diskriminator. Dengan Encoder, kita belajar pemetaan invers gambar wajah masukan dan kondisi usia dengan vektor latent. FaceNet adalah jaringan pengenalan wajah yang mempelajari perbedaan antara gambar input x dan gambar yang direkonstruksi. Kami memiliki jaringan Generator, yang mengambil representasi tersembunyi yang terdiri dari gambar wajah dan vektor kondisi dan menghasilkan gambar. Jaringan diskriminator adalah untuk mendiskriminasikan antara gambar nyata dan gambar palsu. Masalah

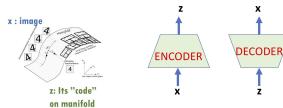
dengan cGANs adalah bahwa mereka tidak dapat mempelajari tugas pemetaan terbalik masukan gambar  $x$  dengan atribut  $y$  ke vektor laten  $z$ . Solusi untuk masalah ini adalah dengan menggunakan jaringan Encoder. Kita dapat melatih jaringan encoder untuk memperkirakan pemetaan terbalik dari input Images  $x$ .



**Gambar 9.37** Illustrasi Arsitektur cGAN

#### 9.4.1.3 Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari AgecGAN.

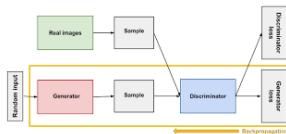
Tujuan utama dari jaringan Encoder adalah untuk menghasilkan vektor laten dari gambar yang disediakan. Pada dasarnya, dibutuhkan gambar dimensi (64, 64, 3) dan mengubahnya menjadi vektor 100-dimensi. Jaringan Encoder adalah jaringan syaraf convolutional yang dalam. Jaringan berisi empat convolutional blok dan dua lapisan padat. Setiap blok convolutional berisi lapisan convolutional, lapisan normalisasi batch, dan fungsi aktivasi. Di setiap blok convolutional, setiap lapisan convolutional diikuti oleh lapisan normalisasi batch, kecuali lapisan convolutional pertama.



**Gambar 9.38** Illustrasi Network Encoder

#### 9.4.1.4 Jelaskan dengan ilustrasi gambar sendiri arsitektur generator network dari AgecGAN.

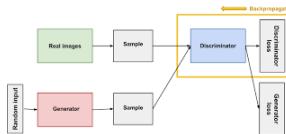
Tujuan utama dari generator adalah untuk menghasilkan gambar dari dimensi (64, 64, 3). Dibutuhkan vektor laten 100 dimensi dan beberapa informasi tambahan,  $y$ , dan mencoba untuk menghasilkan gambar yang realistik. Jaringan Generator adalah jaringan neural yang mendalam convolutional juga. Hal ini terdiri dari lapisan padat, upsampling, dan convolutional. Dibutuhkan dua nilai input: vektor kebisikan dan nilai pengkondisian. Nilai pengkondisian adalah informasi tambahan yang diberikan ke jaringan. Untuk Age-cGAN, ini akan menjadi usia.



**Gambar 9.39** Illustrasi Network Generator

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

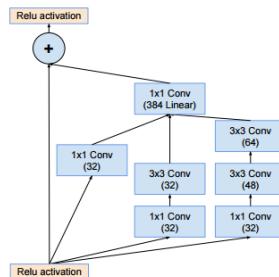
Tujuan utama dari jaringan diskriminasi adalah untuk mengidentifikasi apakah gambar yang disediakan adalah palsu atau nyata. Hal ini dilakukan dengan melalui gambar melalui rangkaian lapisan sampling bawah dan beberapa lapisan klasifikasi. Dengan kata lain, ini memprediksi Apakah gambar itu nyata atau palsu. Seperti jaringan lain, Jaringan diskriminasi 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 9.40** Illustrasi Discriminator Network

#### 9.4.1.6 Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model.

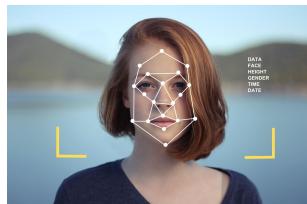
pre-trained Inception-ResNet-2 network, sekali disediakan dengan gambar, mengembalikan yang sesuai embedding. Tertanam yang diekstrak untuk gambar asli dan gambar direkonstruksi dapat dihitung dengan menghitung jarak Euclidean dari yang tertanam.



**Gambar 9.41** Illustrasi Inception-ResNet-2 Model.

#### **9.4.1.7 Jelaskan dengan ilustrasi gambar sendiri arsitektur Face recognition network Age-cGAN.**

Tujuan utama dari jaringan pengenalan wajah adalah untuk mengenali identitas seseorang dalam gambar yang diberikan.



**Gambar 9.42** Illustrasi Face recognition network Age-cGAN.

#### **9.4.1.8 Sebutkan dan jelaskan serta di sertai contoh-contoh tahapan dari Age-cGAN**

Age-cGAN memiliki beberapa tahapan pelatihan. Seperti disebutkan di bagian sebelumnya, 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.

#### **9.4.1.9 Berikan contoh perhitungan fungsi training objektif**

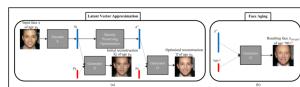
Objektif Trainning ialah untuk meminimalkan loss function sebagai log likelihood function yang diberikan pada persamaan dimana D melambangkan training data.

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

**Gambar 9.43** Training Objektif

#### **9.4.1.10 Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation**

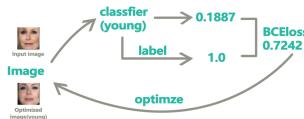
Perkiraan vektor laten awal adalah metode untuk memperkirakan vektor laten untuk mengoptimalkan rekonstruksi gambar wajah. Untuk memperkirakan vektor laten, kami memiliki jaringan Encoder. Kami melatih jaringan Encoder pada gambar yang dihasilkan dan gambar nyata. Setelah dilatih, Jaringan Encoder akan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



**Gambar 9.44** Illustrasi Initial latent vector approximation

#### 9.4.1.11 Berikan contoh perhitungan latent vector optimization

Latent vector approximation kemampuan untuk membuat gambar yang realistik dan tajam serta menghasilkan gambar wajah pada usia target.



**Gambar 9.45** Contoh Perhitungan Latent vector optimization

#### 9.4.2 Praktek

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

```
1 from google.colab import drive
2 drive.mount('/content/drive')
3 # In[1]: Ekstrak File:
4 import tarfile
5 tf = tarfile.open("/content/drive/My Drive/zMachine Learning/
    wiki_crop.tar")
6 tf.extractall(path="/content/drive/My Drive/zMachine Learning/
    wiki_crop/")
```

Kode di atas akan melakukan mount dan extract dataset.

- Masuk ke google colab dengan akun google
- Klik mount google drive
- Lakukan proses untar menggunakan code yang di run google colab
- 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
15     # Load the list of all files
16     full_path = meta[dataset][0, 0]["full_path"][0]
17
18     # List of Matlab serial date numbers
19     dob = meta[dataset][0, 0]["dob"][0]
20
21     # List of years when photo was taken
22     photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
23
24     # Calculate age for all dobs
25     age = [calculate_age(photo_taken[i], dob[i]) for i in range(
26         len(dob))]
27
28     # Create a list of tuples containing a pair of an image path
29     # and age
30     images = []
31     age_list = []
32     for index, image_path in enumerate(full_path):
33         images.append(image_path[0])
34         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 bahawa 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='same')(input_layer)
9     # enc = BatchNormalization()(enc)
10    enc = LeakyReLU(alpha=0.2)(enc)
11
12    # 2nd Convolutional Block
13    enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='same')(enc)
14    enc = BatchNormalization()(enc)
15    enc = LeakyReLU(alpha=0.2)(enc)
16
17    # 3rd Convolutional Block

```

```

18 enc = Conv2D(filters=128, kernel_size=5, strides=2, padding='same')(enc)
19 enc = BatchNormalization()(enc)
20 enc = LeakyReLU(alpha=0.2)(enc)
21
22 # 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 bahawa awam dengan ilustrasi sederhana.

```

1 def build_generator():
2     """
3         Create a Generator Model with hyperparameters values defined
4         as follows
5         """
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)

```

```

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

```

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

- Jelaskan bagaimana kode program The Discriminator Network bekerja dijelaskan dengan bahawa awam dengan ilustrasi sederhana.

```

1 def build_discriminator():
2     """
3         Create a Discriminator Model with hyperparameters values
4         defined as follows
5     """
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')(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)

```

```

28     x = Flatten()(x)
29     x = Dense(1, activation='sigmoid')(x)
30
31 model = Model(inputs=[image_input, label_input], outputs=[x])
32     return model

```

Diskriminatator 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_crop1")
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])

```

```
37     adversarial_model.compile(loss=['binary_crossentropy'],
38                                 optimizer=gen_optimizer)
39
40     tensorboard = TensorBoard(log_dir="logs/{}".format(time.time()))
41     tensorboard.set_model(generator)
42     tensorboard.set_model(discriminator)
43
44     """
45     Load the dataset
46     """
47     images, age_list = load_data/wiki_dir=wiki_dir, dataset="wiki")
48     age_cat = age_to_category(age_list)
49     final_age_cat = np.reshape(np.array(age_cat), [len(age_cat),
50                                         1])
51     classes = len(set(age_cat))
52     y = to_categorical(final_age_cat, num_classes=len(set(age_cat)))
53
54     loaded_images = load_images/wiki_dir, images, (image_shape[0],
55                                                 image_shape[1]))
56
57     # Implement label smoothing
58     real_labels = np.ones((batch_size, 1), dtype=np.float32) *
59     0.9
60     fake_labels = np.zeros((batch_size, 1), dtype=np.float32) *
61     0.1
62
63     """
64     Train the generator and the discriminator network
65     """
66
67     if TRAIN_GAN:
68         for epoch in range(epochs):
69             print("Epoch:{}".format(epoch))
70
71             gen_losses = []
72             dis_losses = []
73
74             number_of_batches = int(len(loaded_images) /
75             batch_size)
76             print("Number of batches:", number_of_batches)
77             for index in range(number_of_batches):
78                 print("Batch:{}".format(index + 1))
79
80                 images_batch = loaded_images[index * batch_size:(index +
81                                         1) * batch_size]
82                 images_batch = images_batch / 127.5 - 1.0
83                 images_batch = images_batch.astype(np.float32)
84
85                 y_batch = y[index * batch_size:(index + 1) *
86                             batch_size]
87                 z_noise = np.random.normal(0, 1, size=(batch_size,
88                                              z_shape))
89
90                 """
```

```

81     Train the discriminator network
82     """
83
84     # Generate fake images
85     initial_recon_images = generator.predict_on_batch(
86     ([z_noise, y_batch]))
87
88     d_loss_real = discriminator.train_on_batch([
89     images_batch, y_batch], real_labels)
90     d_loss_fake = discriminator.train_on_batch([
91     initial_recon_images, y_batch], fake_labels)
92
93     d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)
94     print("d_loss:{}".format(d_loss))
95
96     """
97     Train the generator network
98     """
99
100    z_noise2 = np.random.normal(0, 1, size=(batch_size, z_shape))
101    random_labels = np.random.randint(0, 6,
102        batch_size).reshape(-1, 1)
103    random_labels = to_categorical(random_labels, 6)
104
105    g_loss = adversarial_model.train_on_batch([
106        z_noise2, random_labels], [1] * batch_size)
107
108    print("g_loss:{}".format(g_loss))
109
110    gen_losses.append(g_loss)
111    dis_losses.append(d_loss)
112
113    # Write losses to Tensorboard
114    write_log(tensorboard, 'g_loss', np.mean(gen_losses),
115    epoch)
116    write_log(tensorboard, 'd_loss', np.mean(dis_losses),
117    epoch)
118
119    """
120    Generate images after every 10th epoch
121    """
122
123    if epoch % 10 == 0:
124        images_batch = loaded_images[0:batch_size]
125        images_batch = images_batch / 127.5 - 1.0
126        images_batch = images_batch.astype(np.float32)
127
128        y_batch = y[0:batch_size]
129        z_noise = np.random.normal(0, 1, size=(batch_size,
130            z_shape))
131
132        gen_images = generator.predict_on_batch([z_noise,
133            y_batch])
134
135        for i, img in enumerate(gen_images[:5]):
```

```

126         save_rgb_img(img, path="results/img_{i}.png")
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 bahawa 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) *
30                         batch_size]
31             y_batch = y[index * batch_size:(index + 1) *
32                         batch_size]
33
34             generated_images = generator.predict_on_batch([
z_batch, y_batch])
35
36             # Train the encoder model

```

```
35     encoder_loss = encoder.train_on_batch(
36         generated_images, z_batch)
37         print("Encoder loss:", encoder_loss)
38
39         encoder_losses.append(encoder_loss)
40
41         # Write the encoder loss to Tensorboard
42         write_log(tensorboard, "encoder_loss", np.mean(
43             encoder_losses), epoch)
44
45         # Save the encoder model
46         encoder.save_weights("encoder.h5")
47
48         """
49         Optimize the encoder and the generator network
50         """
51
52         if TRAIN_GAN_WITH_FR:
53
54             # Load the encoder network
55             encoder = build_encoder()
56             encoder.load_weights("encoder.h5")
57
58             # Load the generator network
59             generator.load_weights("generator.h5")
60
61             image_resizer = build_image_resizer()
62             image_resizer.compile(loss=['binary_crossentropy'],
63                                   optimizer='adam')
64
65             # Face recognition model
66             fr_model = build_fr_model(input_shape=fr_image_shape)
67             fr_model.compile(loss=['binary_crossentropy'],
68                               optimizer="adam")
69
70             # Make the face recognition network as non-trainable
71             fr_model.trainable = False
72
73             # Input layers
74             input_image = Input(shape=(64, 64, 3))
75             input_label = Input(shape=(6,))
76
77             # Use the encoder and the generator network
78             latent0 = encoder(input_image)
79             gen_images = generator([latent0, input_label])
80
81             # Resize images to the desired shape
82             resized_images = Lambda(lambda x: K.resize_images(
83                 gen_images, height_factor=3, width_factor=3,
84                 data_format='channels_last'))(gen_images)
85             embeddings = fr_model(resized_images)
86
87             # Create a Keras model and specify the inputs and outputs
88             # for the network
89             fr_adversarial_model = Model(inputs=[input_image,
90                 input_label], outputs=[embeddings])
```

```
83      # Compile the model
84      fr_adversarial_model.compile(loss=euclidean_distance_loss
85      , optimizer=adversarial_optimizer)
86
87      for epoch in range(epochs):
88          print("Epoch:", epoch)
89
90          reconstruction_losses = []
91
92          number_of_batches = int(len.loaded_images) / 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:(index + 1) * batch_size]
98              images_batch = images_batch / 127.5 - 1.0
99              images_batch = images_batch.astype(np.float32)
100
101              y_batch = y[index * batch_size:(index + 1) * batch_size]
102
103              images_batch_resized = image_resizer.predict_on_batch(images_batch)
104
105              real_embeddings = fr_model.predict_on_batch(
106                  images_batch_resized)
107
108              reconstruction_loss = fr_adversarial_model.
109              train_on_batch([images_batch, y_batch], real_embeddings)
110
111              print("Reconstruction loss:", reconstruction_loss)
112          )
113
114          reconstruction_losses.append(reconstruction_loss)
115
116          """
117          Generate images
118          """
119          if epoch % 10 == 0:
120              images_batch = loaded_images[0:batch_size]
121              images_batch = images_batch / 127.5 - 1.0
122              images_batch = images_batch.astype(np.float32)
123
124              y_batch = y[0:batch_size]
125              z_noise = np.random.normal(0, 1, size=(batch_size
126              , z_shape))
127
128              gen_images = generator.predict_on_batch([z_noise,
129                  y_batch])
```

```

128
129         for i, img in enumerate(gen_images[:5]):
130             save_rgb_img(img, path="results/img_opt_{}-{}.
131             {}.png".format(epoch, i))
132
133             # Save improved weights for both of the networks
134             generator.save_weights("generator_optimized.h5")
135             encoder.save_weights("encoder_optimized.h5")

```

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

### 9.4.3 Penanganan Error

#### 9.4.3.1 Error

- OSError



**Gambar 9.46** OSError

#### 9.4.3.2 Solusi Error

- OSError

Pastikan folder telah ada

### 9.4.4 Bukti Tidak Plagiat



**Gambar 9.47** Bukti tidak plagiat

### 9.4.5 Link Youtube

<https://bit.ly/KecerdasanBuatanDirga>

## 9.5 1174087 - Ilham Muhammad Ariq

### 9.5.1 Teori

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

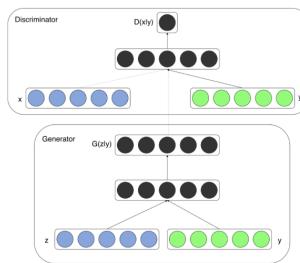
Vanilla GAN adalah tipe GAN paling sederhana. Di sini, Generator dan Diskriminasi adalah perceptron multi-layer sederhana. Dalam vanilla GAN, algoritma ini sangat sederhana, ia mencoba untuk mengoptimalkan persamaan matematika menggunakan keturunan gradien stokastik. CGAN (Conditional GAN), label bertindak sebagai ekstensi ke ruang laten  $z$  untuk menghasilkan dan membedakan gambar dengan lebih baik.



**Gambar 9.48** Valina GAN-cGAN

2. Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN.

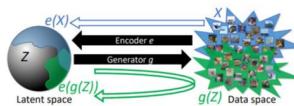
Age cGAN ialah dengan mengkondisikan model pada informasi tambahan dimungkinkan untuk mengarahkan proses pembuatan data. Pengkondisian semacam itu dapat didasarkan pada label kelas.



**Gambar 9.49** Age-cGAN

3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari Agec-GAN.

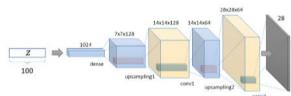
Arsitektur encoder biasanya digunakan untuk memodelkan struktur manifold dan membalikkan encoder untuk memproses data.



**Gambar 9.50** Encoder Age cGANr

4. Jelaskan dengan ilustrasi gambar sendiri arsitektur generator network dari Agec-GAN.

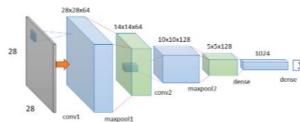
Arsitektur generator adalah sebuah array yang digunakan secara random, yang disebut seed. dari data seed tersebut, generator akan merubahnya menjadi sebuah gambar yang ukuran  $28 \times 28$  dengan menggunakan Convolutional Neural Network.



**Gambar 9.51** Network Age cGAN

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

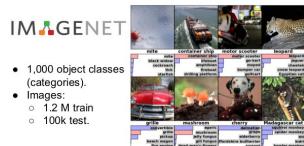
Arsitektur diskriminator adalah CNN yang dapat menerima input gambar yang berukuran  $28,28$  serta menghasilkan angka biner yang menyatakan apakah data yang diinputkan merupakan dataset asli atau gambar dataset palsu.



**Gambar 9.52** Discriminator Age cGAN

6. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model.

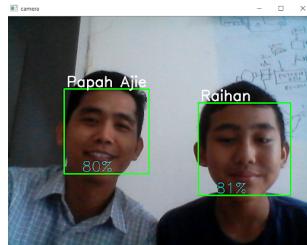
Pre-Trained Network atau Transfer Learning merupakan suatu metode penyelesaian yang memanfaatkan model yang sudah dilatih terhadap suatu dataset untuk menyelesaikan masalah dengan cara menggunakan sebagai starting point, memodifikasi dan mengupdate parameternya, sehingga sesuai dengan dataset yang baru.



**Gambar 9.53** Pretrained Inception ResNet

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

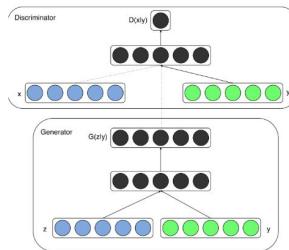
Face Recognition merupakan salah satu sistem yang mengimplementasi Deep Learning yang dapat mengenali wajah secara fisik dari gambar digital atau video frame.



**Gambar 9.54** Face recognition network Age-cGAN

8. Sebutkan dan jelaskan serta diertai contoh-contoh tahapan dari Age-cGAN.

Pada dari Age-cGan ni terdapat 2 tahapan dengan generator dan diskriminator. dimana untuk tahap generator sendiri membutuhkan vektor laten 100 serta menghasilkan gambar yang realistik dari dimensinya. sedangkan tahap diskriminator itu tahapan dimana memprediksi gambar yang diberikan nyata atau palsu.



**Gambar 9.55** Tahap Age cGAN

9. Berikan contoh perhitungan fungsi training objektif.

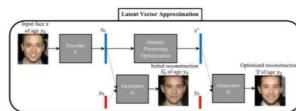
Objektif Trainning ialah untuk meminimalkan loss function sebagai log likelihood function yang diberikan pada persamaan dimana D melambangkan training data.

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

**Gambar 9.56** Training Objektif

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

Latent vector approdimation kemampuan untuk membuat gamar yang realistik dan tajam serta menghasilkan gambar wajah pada usia target.



**Gambar 9.57** Initial Latent Vector Approximation

11. Berikan contoh perhitungan latent vector optimization.

Perhitungan lantent 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 zi dalam ruang laten z.



**Gambar 9.58** Latent Vector Optimization

### 9.5.2 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/
6 wiki_crop.tar")
tf.extractall(path="/content/drive/My Drive/Colab Notebooks/
Chapter9/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 load_data(wiki_dir, dataset='wiki'):
2     # Load the wiki.mat file
3     meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset
4         )))
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])

```

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

3. Jelaskan bagaimana kode program The Encoder Network bekerja dijelaskan dengan bahawa 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='same')(input_layer)
9     # enc = BatchNormalization()(enc)
10    enc = LeakyReLU(alpha=0.2)(enc)
11
12    # 2nd Convolutional Block
13    enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='same')(enc)
14    enc = BatchNormalization()(enc)
15    enc = LeakyReLU(alpha=0.2)(enc)
16
17    # 3rd Convolutional Block
18    enc = Conv2D(filters=128, kernel_size=5, strides=2, padding='same')(enc)
19    enc = BatchNormalization()(enc)
20    enc = LeakyReLU(alpha=0.2)(enc)
21
22    # 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 bahawa 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

```

```

6   num_classes = 6
7
8   input_z_noise = Input(shape=(latent_dims,))
9   input_label = Input(shape=(num_classes,))
10
11  x = concatenate([input_z_noise, input_label])
12
13  x = Dense(2048, input_dim=latent_dims + num_classes)(x)
14  x = LeakyReLU(alpha=0.2)(x)
15  x = Dropout(0.2)(x)
16
17  x = Dense(256 * 8 * 8)(x)
18  x = BatchNormalization()(x)
19  x = LeakyReLU(alpha=0.2)(x)
20  x = Dropout(0.2)(x)
21
22  x = Reshape((8, 8, 256))(x)
23
24  x = UpSampling2D(size=(2, 2))(x)
25  x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
26  x = BatchNormalization(momentum=0.8)(x)
27  x = LeakyReLU(alpha=0.2)(x)
28
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 bahawa awam dengan ilustrasi sederhana.

```

1 def build_discriminator():
2     """
3         Create a Discriminator Model with hyperparameters values
4         defined as follows
5     """
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')(image_input)

```

```
11 x = LeakyReLU(alpha=0.2)(x)
12
13 label_input1 = Lambda(expand_label_input)(label_input)
14 x = concatenate([x, label_input1], axis=3)
15
16 x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
17 x = BatchNormalization()(x)
18 x = LeakyReLU(alpha=0.2)(x)
19
20 x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
21 x = BatchNormalization()(x)
22 x = LeakyReLU(alpha=0.2)(x)
23
24 x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
25 x = BatchNormalization()(x)
26 x = LeakyReLU(alpha=0.2)(x)
27
28 x = Flatten()(x)
29 x = Dense(1, activation='sigmoid')(x)
30
31 model = Model(inputs=[image_input, label_input], outputs=[x])
32 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_crop1")
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=0.999,
20                                 epsilon=10e-8)
21
22    """
23        Build and compile networks
24    """
25    # Build and compile the discriminator network
26    discriminator = build_discriminator()
```

```
24 discriminator.compile(loss=['binary_crossentropy'], optimizer=dis_optimizer)
25
26 # Build and compile the generator network
27 generator = build_generator()
28 generator.compile(loss=['binary_crossentropy'], optimizer=gen_optimizer)
29
30 # Build and compile the adversarial model
31 discriminator.trainable = False
32 input_z_noise = Input(shape=(100,))
33 input_label = Input(shape=(6,))
34 recons_images = generator([input_z_noise, input_label])
35 valid = discriminator([recons_images, input_label])
36 adversarial_model = Model(inputs=[input_z_noise, input_label],
37                             outputs=[valid])
38 adversarial_model.compile(loss=['binary_crossentropy'],
39                            optimizer=gen_optimizer)
40
41 tensorboard = TensorBoard(log_dir="logs/{}".format(time.time()))
42 tensorboard.set_model(generator)
43 tensorboard.set_model(discriminator)
44 """
45 Load the dataset
46 """
47 images, age_list = load_data(wiki_dir=wiki_dir, dataset="wiki")
48 age_cat = age_to_category(age_list)
49 final_age_cat = np.reshape(np.array(age_cat), [len(age_cat), 1])
50 classes = len(set(age_cat))
51 y = to_categorical(final_age_cat, num_classes=len(set(age_cat)))
52
53 loaded_images = load_images(wiki_dir, images, (image_shape[0], image_shape[1]))
54
55 # Implement label smoothing
56 real_labels = np.ones((batch_size, 1), dtype=np.float32) * 0.9
57 fake_labels = np.zeros((batch_size, 1), dtype=np.float32) * 0.1
58 """
59 Train the generator and the discriminator network
60 """
61 if TRAIN_GAN:
62     for epoch in range(epochs):
63         print("Epoch:{}".format(epoch))
64
65         gen_losses = []
66         dis_losses = []
```

```
68         number_of_batches = int(len.loaded_images) /  
69         batch_size)  
70         print("Number of batches:", number_of_batches)  
71         for index in range(number_of_batches):  
72             print("Batch:{}".format(index + 1))  
73  
74             images_batch = loaded_images[index * batch_size:(  
75                 index + 1) * batch_size]  
76             images_batch = images_batch / 127.5 - 1.0  
77             images_batch = images_batch.astype(np.float32)  
78  
79             y_batch = y[index * batch_size:(index + 1) *  
80                 batch_size]  
81             z_noise = np.random.normal(0, 1, size=(batch_size  
82                 , z_shape))  
83  
84             """  
85             Train the discriminator network  
86             """  
87  
88             # Generate fake images  
89             initial_recon_images = generator.predict_on_batch  
90             ([z_noise, y_batch])  
91  
92             d_loss_real = discriminator.train_on_batch([  
93                 images_batch, y_batch], real_labels)  
94             d_loss_fake = discriminator.train_on_batch([  
95                 initial_recon_images, y_batch], fake_labels)  
96  
97             d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)  
98             print("d_loss:{}".format(d_loss))  
99  
100            """  
101            Train the generator network  
102            """  
103  
104            z_noise2 = np.random.normal(0, 1, size=(  
105                batch_size, z_shape))  
106            random_labels = np.random.randint(0, 6,  
107                batch_size).reshape(-1, 1)  
108            random_labels = to_categorical(random_labels, 6)  
109  
110            g_loss = adversarial_model.train_on_batch([  
111                z_noise2, random_labels], [1] * batch_size)  
112  
113            print("g_loss:{}".format(g_loss))  
114  
115            gen_losses.append(g_loss)  
116            dis_losses.append(d_loss)  
117  
118            # Write losses to Tensorboard  
119            write_log(tensorboard, 'g_loss', np.mean(gen_losses),  
120 epoch)  
121            write_log(tensorboard, 'd_loss', np.mean(dis_losses),  
122 epoch)
```

```

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,
122         z_shape))
123
124     gen_images = generator.predict_on_batch([z_noise,
125         y_batch])
126
127     for i, img in enumerate(gen_images[:5]):
128         save_rgb_img(img, path="results/img-{}-{}.png"
129             .format(epoch, i))
130
131     # Save networks
132     try:
133         generator.save_weights("generator.h5")
134         discriminator.save_weights("discriminator.h5")
135     except Exception as e:
136         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 bahawa 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) *
30                         batch_size]
30         y_batch = y[index * batch_size:(index + 1) *
31                         batch_size]
32
33         generated_images = generator.predict_on_batch([
34             z_batch, y_batch])
35
36         # Train the encoder model
37         encoder_loss = encoder.train_on_batch(
38             generated_images, z_batch)
39         print("Encoder loss:", encoder_loss)
40
41         encoder_losses.append(encoder_loss)
42
43         # Write the encoder loss to Tensorboard
44         write_log(tensorboard, "encoder_loss", np.mean(
45             encoder_losses), epoch)
46
47         # Save the encoder model
48         encoder.save_weights("encoder.h5")
49
50     """
51     Optimize the encoder and the generator network
52     """
53
54     if TRAIN_GAN_WITH_FR:
55
56         # Load the encoder network
57         encoder = build_encoder()
58         encoder.load_weights("encoder.h5")
59
60         # Load the generator network
61         generator.load_weights("generator.h5")
62
63         image_resizer = build_image_resizer()
64         image_resizer.compile(loss=['binary_crossentropy'],
65                               optimizer='adam')
66
67         # Face recognition model
68         fr_model = build_fr_model(input_shape=fr_image_shape)
69         fr_model.compile(loss=['binary_crossentropy'], optimizer=
70                           "adam")
71
72         # Make the face recognition network as non-trainable
73         fr_model.trainable = False
74
75         # Input layers
76         input_image = Input(shape=(64, 64, 3))
77         input_label = Input(shape=(6,))
```

```
72     # Use the encoder and the generator network
73     latent0 = encoder(input_image)
74     gen_images = generator([latent0, input_label])
75
76     # Resize images to the desired shape
77     resized_images = Lambda(lambda x: K.resize_images(
78         gen_images, height_factor=3, width_factor=3,
79         data_format='channels_last'))(gen_images)
80     embeddings = fr_model(resized_images)
81
82     # Create a Keras model and specify the inputs and outputs
83     # for the network
84     fr_adversarial_model = Model(inputs=[input_image,
85         input_label], outputs=[embeddings])
86
87     # Compile the model
88     fr_adversarial_model.compile(loss=euclidean_distance_loss,
89         optimizer=adversarial_optimizer)
90
91     for epoch in range(epochs):
92         print("Epoch:", epoch)
93
94         reconstruction_losses = []
95
96         number_of_batches = int(len.loaded_images) / batch_size
97         print("Number of batches:", number_of_batches)
98         for index in range(number_of_batches):
99             print("Batch:", index + 1)
100
101             images_batch = loaded_images[index * batch_size:(index + 1) * batch_size]
102             images_batch = images_batch / 127.5 - 1.0
103             images_batch = images_batch.astype(np.float32)
104
105             y_batch = y[index * batch_size:(index + 1) * batch_size]
106
107             images_batch_resized = image_resizer.
108             predict_on_batch(images_batch)
109
110             real_embeddings = fr_model.predict_on_batch(
111                 images_batch_resized)
112
113             reconstruction_loss = fr_adversarial_model.
114             train_on_batch([images_batch, y_batch], real_embeddings)
115
116             print("Reconstruction loss:", reconstruction_loss)
117
118             reconstruction_losses.append(reconstruction_loss)
119
120             # Write the reconstruction loss to Tensorboard
121             write_log(tensorboard, "reconstruction_loss", np.mean(
122                 reconstruction_losses), epoch)
```

```

115      """
116      """
117      Generate images
118      """
119      if epoch % 10 == 0:
120          images_batch = loaded_images[0:batch_size]
121          images_batch = images_batch / 127.5 - 1.0
122          images_batch = images_batch.astype(np.float32)
123
124          y_batch = y[0:batch_size]
125          z_noise = np.random.normal(0, 1, size=(batch_size
126                                      , z_shape))
127
128          gen_images = generator.predict_on_batch([z_noise,
129                                                    y_batch])
130
131          for i, img in enumerate(gen_images[:5]):
132              save_rgb_img(img, path="results/img_opt_{}-{}".
133                           format(epoch, i))
134
135      # Save improved weights for both of the networks
136      generator.save_weights("generator_optimized.h5")
137      encoder.save_weights("encoder_optimized.h5")

```

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

### 9.5.3 Penanganan Error

#### 1. FileNotFoundError

FileNotFoundError: [Errno 2] No such file or directory: '/content/drive/MyDrive/Colab Notebooks/Chapter/latent2d/2023-07-10T14-15-10Z/latent2d\_2023-07-10T14-15-10Z.h5'

**Gambar 9.59** FileNotFoundError

#### 2. Cara Penanganan Error

- FileNotFoundError

Error tersebut karena salah penempatan file.

### 9.5.4 Bukti Tidak Plagiat



**Gambar 9.60** Bukti Tidak Melakukan Plagiat Chapter 9

