

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

1 Chapter 1	1
2 Chapter 2	3
3 Chapter 3	5
4 Chapter 4	7
5 Chapter 5	9
6 Chapter 6	11
7 Chapter 7	13
8 Chapter 8	15
9 Chapter 9	29
10 Chapter 10	39
11 Chapter 11	41
12 Chapter 12	43
13 Chapter 13	45
14 Chapter 14	47

DAFTAR ISI

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

5	Chapter 5	9
6	Chapter 6	11
7	Chapter 7	13
8	Chapter 8	15
8.1	Dwi Septiani Tsaniyah - 1174003	15
8.1.1	Teori	15
8.1.2	Pemrograman	20
9	Chapter 9	29
9.1	Dwi Septiani Tsaniyah - 1174003	29
9.1.1	Teori	29
9.1.2	Praktek	33
9.1.3	Penanganan Error	38
9.1.4	Bukti Tidak Plagiat	38
10	Chapter 10	39
11	Chapter 11	41
12	Chapter 12	43
13	Chapter 13	45
14	Chapter 14	47

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 Dwi Septiani Tsaniyah - 1174003

8.1.1 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 8.1 Ilustrasi

2. Jelaskan dengan ilustrasi gambar sendiri apa itu diskriminator dengan perumpamaan dosen anda sebagai diskriminornya. 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 ilustrasinya dosen menyuruh mahasiswanya membuat kue bolu, maka hasil yang akan dibuat adalah kue bolu.



Gambar 8.2 Ilustrasi

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 8.3 Ilustrasi

4. Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur diskriminator 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 8.4 Ilustrasi

5. Jelaskan dengan ilustrasi gambar apa itu latent space. Latent space adalah representasi dari data yang dikompresi, untuk contohnya misalkan 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 8.5 Ilustrasi

6. Jelaskan dengan ilustrasi gambar 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

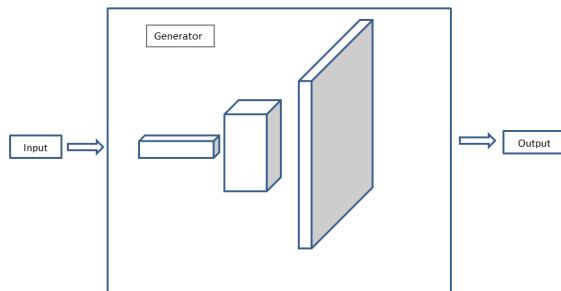
Gambar 8.6 Ilustrasi

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 pelensaan 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(t)
gen.add(Dense(units=400)) #Menambah dense lay
gen.add(Dense(units=784, activation='tanh'))
```

```
gen.compile(loss='binary_crossentropy', opti
gen.summary() #Memproses data yang sudah di
```

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.

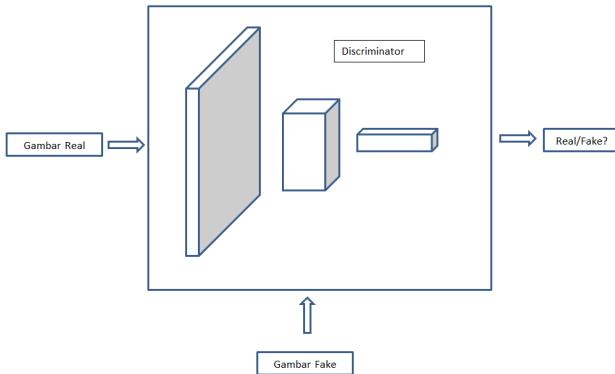


Gambar 8.7 Ilustrasi

10. Berikan contoh dengan ilustrasi dari arsitektur dikriminator 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(X)[1]))
diskrim.add(Dense(units=400)) #Mensetting Dense
diskrim.add(Dense(units=200, activation='sigmoid'))
diskrim.compile(loss='binary_crossentropy',
diskrim.summary() #Memproses data yang sudah di
```

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 8.8 Ilustrasi

11. Jelaskan bagaimana kaitan output dan input antara generator dan diskriminator tersebut. Jelaskan kenapa inputan dan outputan seperti itu. Pada kedua metode tersebut, akan disebutkan berapa akurasi dari setiap metode. 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 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 Frchet antara dua Gaussians dipasang ke representasi fitur dari jaringan Inception.

15. Jelaskan kelebihan dan kekurangan GAN Keungungan 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 pendekripsi 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

8.1.2 Pemrograman

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)
```

```

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
39     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.

- Jelaskan dengan kode program arsitektur dari diskriminatator 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     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])

```

31 `return gen_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.

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 diskriminatator pada fungsi build discriminator. membangun diskriminatator berfungsi untuk mendefenisikan seluruh gambar yang sudah di load generator sebagai gambar fake dan real
11. Jelaskan kode program

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

```

artinya adalah load dataset yang hanya dalam folder chair data train

13. Jelaskan kode program

```

1   """
2     Create models
3   """
4   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)
3         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,
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=[batch_size, 1, 1, 1, z_size]).astype(np.float32)
2                     volumes_batch = volumes[index * batch_size:(index
+ 1) * batch_size, :, :, :]
```

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

21. Jelaskan kode program

```

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

```

ialah membuat sample gambar palsu yang akan diteruskan ke diskriminat

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
12           else:
13               d_loss = 0.0

```

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
6           gen_losses.append(g_loss)
7           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
11                   voxels = np.squeeze(generated_volume)
12                   voxels[voxels < 0.5] = 0.

```

```

8                     voxels[ voxels >= 0.5] = 1.
9             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) ,
epoch)
3      write_log(tensorboard , 'd_loss' , np.mean(dis_losses) ,
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" , "
generator_weights.h5"))
5      discriminator.save_weights(os.path.join("models" , "
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

```

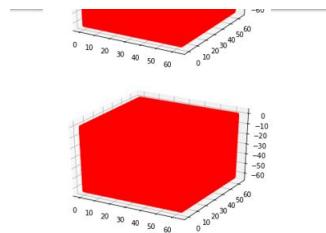
1  if MODE == 'predict':
2      # Create models
3      generator = build_generator()
4      discriminator = build_discriminator()
5
6      # Load model weights
7      generator.load_weights(os.path.join("models" , "
generator_weights.h5") , True)
8      discriminator.load_weights(os.path.join("models"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

```

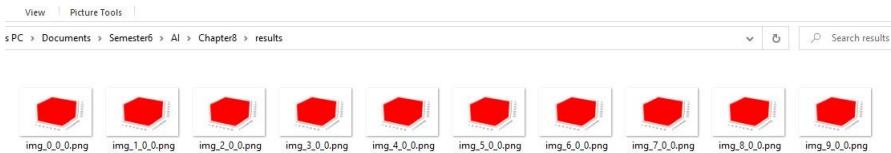
ini adalah tahap akhir untuk melakukan testing dari model yang telah dibuat dan buat model dari yang sudah di create sebelumnya yaitu generator dan diskriminator.

```
-----  
Loading data...  
Data loaded...  
Epoch: 0  
Number of batches: 10  
Batch: 1  
d_loss:0.6931722164154053  
g_loss:0.6931138038635254  
Batch: 2  
d_loss:0.6931138038635254  
g_loss:0.6931576728820801  
Batch: 3  
d_loss:0.6931576728820801  
g_loss:0.69310909354919434  
Batch: 4  
d_loss:0.69310909354919434  
g_loss:0.6931560039520264  
Batch: 5  
d_loss:0.693108081817627  
Batch: 6  
g_loss:0.693108081817627  
-----
```

Gambar 8.9 hasil



Gambar 8.10 hasil



Gambar 8.11 hasil

8.2 1174012 - Damara Benedikta

8.2.1 Soal Teori

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

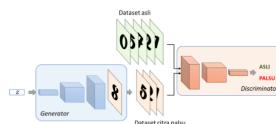
Generator merupakan sebuah arsitektur jaringan saraf tiruan yang bertujuan untuk membentuk atau membangkitkan suatu data yang benar-benar baru, dari tidak ada menjadi ada.



Gambar 8.12 Teori 1

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

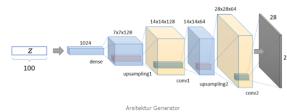
Discriminator merupakan jaringan klasifikasi biner yang menerima input gambar tiga dimensi dan mengeluarkan klasifikasi menyatakan input gambar adalah gambar asli dari dataset atau merupakan gambar buatan Generator. Discriminator dilatih dengan sekumpulan data yang dibangkitkan oleh Generator, dan sekumpulan data dari dataset, dan dilatih untuk bisa membedakan keduanya.



Gambar 8.13 Teori 2

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

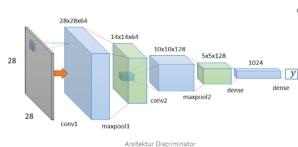
Arsitektur Generator Input Generator adalah sebuah array yang dibangkitkan secara random, disebut seed. Di sini kami tetapkan ukuran input seed adalah [1x100]. Dari masukan seed tersebut, Generator akan mengubahnya menjadi sebuah gambar berukuran [28x28] menggunakan Convolutional Neural Network.



Gambar 8.14 Teori 3

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

Arsitektur administrator Diskriminator merupakan CNN yang menerima input image berukuran [28,28] dan menghasilkan angka biner yang menyatakan apakah input gambar merupakan gambar dari dataset asli (kelas 1) atau merupakan gambar baru/gambar palsu (kelas 0)

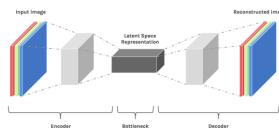


Gambar 8.15 Teori 4

5. Jelaskan dengan ilustrasi gambar apa itu latent space.

Latent space Biasanya itu adalah hipersphere 100 dimensi dengan setiap variabel diambil dari distribusi Gaussian dengan rata-rata nol dan standar deviasi satu. Melalui pelatihan, generator belajar memetakan titik ke ruang laten dengan gambar output spesifik dan pemetaan ini akan berbeda setiap kali model dilatih.

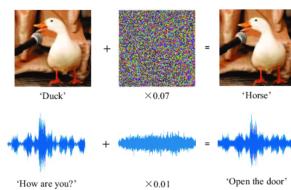
Ruang laten memiliki struktur ketika ditafsirkan oleh model generator, dan struktur ini dapat ditanyakan dan dinavigasi untuk model yang diberikan.



Gambar 8.16 Teori 5

6. Jelaskan dengan ilustrasi gambar apa itu adversarial play.

Adversarial play melibatkan dua orang atau dua pihak yang saling menentang: dari, berkaitan dengan, atau karakteristik prosedur permusuhan atau permusuhan



Gambar 8.17 Teori 6

7. Jelaskan dengan ilustrasi gambar apa itu Nash equilibrium.

Nash equilibrium adalah konsep solusi menggunakan serangkaian strategi dalam negosiasi yang melibatkan 2 pihak atau lebih dimana masing-masing pihak di-asumsikan telah mengetahui strategi lawan, dan masing-masing pihak hanya dapat memperoleh keuntungan dengan mengubah strateginya sendiri.

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

Gambar 8.18 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).

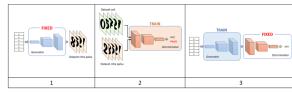
Untuk penjelasan tersebut dijelaskan pada gambar dibawah ini :



Gambar 8.19 Teori 9

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

Secara berurutan, untuk setiap epoch pelatihan akan melakukan tiga tahapan yaitu: Bangkitkan data citra palsu menggunakan Generator sejumlah dataset citra asli Latih Discriminator untuk bisa membedakan dataset citra asli dari dataset citra palsu Gunakan Discriminator yang sudah dilatih untuk melatih Generator agar bisa membangkitkan dataset citra palsu yang dinilai asli oleh Discriminator



Gambar 8.20 Teori 10

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

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

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

```

31

.....

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

```

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 Diskriminator sebagai gambar fake, akan dikembalikan dengan feedback pke generator. Kini Generator bertugas untuk bisa membuat sekumpulan gambar palsu, yang nantinya dapat dilihat oleh Diskriminator, lalu, Diskriminator tidak bisa membedakan fake dan real.

4. Soal 4

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

5. Soal 5

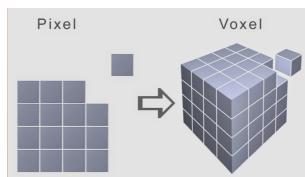
- Clone github
- Download dataset
- Buat folder baru logs dan results

6. Soal 6

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

7. Soal 7

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



Gambar 8.21 Praktek Soal 7

8. Soal 8

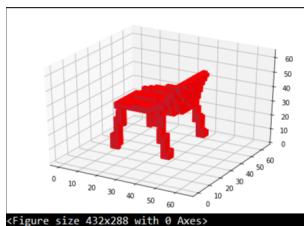
```

1
2
3 # In []
4
5 import scipy.io as io
6 voxels = io.loadmat("data/3DShapeNets/volumetric_data/chair/30/
7 test/chair_000000000_1.mat")['instance']
8 import numpy as np

```

```
9 voxels = np.pad(voxels, (1, 1), 'constant', constant_values=(0, 0))
10 #%%%
11 import scipy.ndimage as nd
12 voxels = nd.zoom(voxels, (2, 2, 2), mode='constant', order=0)
13 #%%%
14 import matplotlib.pyplot as plt
15 from mpl_toolkits.mplot3d import Axes3D
16 fig = plt.figure()
17 ax = Axes3D(fig)
18 ax.voxels(voxels, edgecolor="red")
19
20 plt.show()
21 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 8.22 Hasil Soal 8

9. Soal 9

```

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
8
9 input_layer = Input(shape=gen_input_shape)
10
11 # First 3D transpose convolution(or 3D deconvolution) block
12 a = Deconv3D(filters=gen_filters[0],
13               kernel_size=gen_kernel_sizes[0],
14               strides=gen_strides[0])(input_layer)
15 a = BatchNormalization()(a, training=True)
16 a = Activation(activation='relu')(a)
17
18 # Next 4 3D transpose convolution(or 3D deconvolution) blocks
19 for i in range(gen_convolutional_blocks - 1):
20     a = Deconv3D(filters=gen_filters[i + 1],
21                   kernel_size=gen_kernel_sizes[i + 1],
22                   strides=gen_strides[i + 1])(a)
23     a = BatchNormalization()(a, training=True)
24     a = Activation(activation='relu')(a)
25
26 model = Model(inputs=[input_layer], outputs=a)
27
28 # Print the summary of the model
29 print(model.summary())

```

```

22             strides=gen_strides[i + 1], padding='same'))(
23     a)
24     a = BatchNormalization()(a, training=True)
25     a = Activation(activation=gen_activations[i + 1])(a)
26
27     gen_model = Model(inputs=[input_layer], outputs=[a])
28
29 #%% soal10
30
31
32 def build_discriminator():
33     """

```

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

```

```

36     return dis_model
37
38 #%%%
39
40 def write_log(callback, name, value, batch_no):
41     summary = tf.Summary()
42     summary.value = summary.value.add()

```

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

11. Soal 11

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

12. Soal 12

```

1     beta = 0.5
2     batch_size = 1
3     z_size = 200
4     epochs = 1
5     MODE = "train"
6
7 #%% soal 13
8 """
9     Create models
10 """
11 gen_optimizer = Adam(lr=gen_learning_rate, beta_1=beta)

```

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

13. Soal 13

```

1     discriminator = build_discriminator()
2     discriminator.compile(loss='binary_crossentropy', optimizer=
3     dis_optimizer)
4
5     generator = build_generator()
6     generator.compile(loss='binary_crossentropy', optimizer=
7     gen_optimizer)
8
9 #%% soal14
10    discriminator.trainable = False
11
12    input_layer = Input(shape=(1, 1, 1, z_size))
13    generated_volumes = generator(input_layer)

```

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

14. Soal 14

```

1 adversarial_model = Model(inputs=[input_layer], outputs=[validity])
2 adversarial_model.compile(loss='binary_crossentropy',
3 optimizer=gen_optimizer)
4 #%% soal15
5 print("Loading data...")
6 volumes = get3DImages(data_dir=data_dir)
7 volumes = volumes [..., np.newaxis].astype(np.float)
8 print("Data loaded...")

```

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

15. Soal 15

```

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

```

Kode di atas befungsi untuk melakukan load data pada dataset.

16. Soal 16

```

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))
#%% soal18

```

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

17. Soal 17

```

1 for epoch in range(epochs):
2     print("Epoch:", epoch)

```

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

18. Soal 18

```

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

```

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 #%% soal20
2             z_sample = np.random.normal(0, 0.33, size=[batch_size, 1, 1, 1, z_size]).astype(np.float32)
3             volumes_batch = volumes[index * batch_size:(index + 1) * batch_size, :, :, :]
4
5 #%% soal21

```

Batch adalah jumlah file yang akan di training.

20. Soal 20

```

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

```

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

21. Soal 21

```

1             .....
2
3 #%% soal22

```

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

22. Soal 22

```

1             d_loss = 0.5 * np.add(loss_real, loss_fake)
2             print("d_loss:{}".format(d_loss))
3
4         else:
5             d_loss = 0.0
6

```

```

7 #%% soal23
8
9
10 discriminator.trainable = False
11 """
12     Train the generator network
13 """

```

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

23. Soal 23

```

1 g_loss = adversarial_model.train_on_batch(z,
2 labels_real)
3 print("g_loss:{}".format(g_loss))
4
5 gen_losses.append(g_loss)
6 dis_losses.append(d_loss)
7
8 #%% soal24
9         # Every 10th mini-batch, generate volumes and
10        save them
11        if index % 10 == 0:
12            z_sample2 = np.random.normal(0, 0.33, size=[batch_size, 1, 1, 1, z_size]).astype(np.float32)
13            generated_volumes = generator.predict(
14                z_sample2, verbose=3)

```

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

24. Soal 24

```

1 voxels = np.squeeze(generated_volume)
2 voxels[voxels < 0.5] = 0.
3 voxels[voxels >= 0.5] = 1.
4 saveFromVoxels(voxels, "results/img_{}-{}".format(epoch, index, i))
5
6 #%% soal25
7         # Write losses to Tensorboard
8         write_log(tensorboard, 'g_loss', np.mean(gen_losses),
9 epoch)
10        write_log(tensorboard, 'd_loss', np.mean(dis_losses),
11 epoch)

```

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

25. Soal 25

```

1 Save models
2 """
3
4 #%% soal26

```

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

26. Soal 26

```

1 # Create models
2 generator = build_generator()
3 discriminator = build_discriminator()

```

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

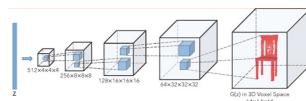
27. Soal 27

```

1 # Load model weights
2 generator.load_weights(os.path.join("models", "generator_weights.h5"), True)
3 discriminator.load_weights(os.path.join("models", "discriminator_weights.h5"), True)
4
5 # Generate 3D models
6 z_sample = np.random.normal(0, 1, size=[batch_size, 1, 1,
7 1, z_size]).astype(np.float32)
8 generated_volumes = generator.predict(z_sample, verbose
=3)
9
10 for i, generated_volume in enumerate(generated_volumes
[1:2]):
11     voxels = np.squeeze(generated_volume)
12     voxels[voxels < 0.5] = 0.
13     voxels[voxels >= 0.5] = 1.
14     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 8.23 Praktek Soal 27

BAB 9

CHAPTER 9

9.1 Dwi Septiani Tsaniyah - 1174003

9.1.1 Teori

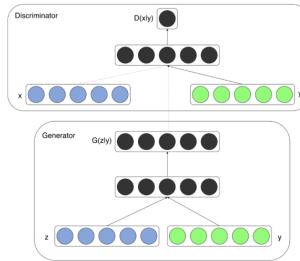
1. Jelaskan dengan ilustrasi gambar sendiri apa perbedaan antara vanilla GAN dan cGAN. Vanilla GANs biasanya tidak memiliki convolutional Neural Jaringan (CNNs) di jaringan mereka. Conditional GANs (cGANs) adalah perpanjangan dari model GAN. Mereka memungkinkan untuk generasi gambar yang memiliki kondisi 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 Age-cGAN.

Arsitektur cGAN untuk penuaan wajah sedikit lebih rumit. AgecGan terdiri dari empat jaringan: Encoder, FaceNet, Jaringan Generator, dan jaringan diskriminat. 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 diskriminat 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 .

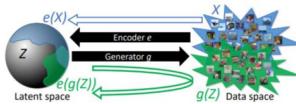


Gambar 9.2 Illustrasi Arsitektur cGAN

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 con-

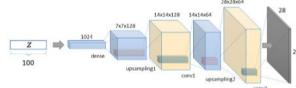
volutional, setiap lapisan convolutional diikuti oleh lapisan normalisasi batch, kecuali lapisan convolutional pertama.



Gambar 9.3 Illustrasi Network Encoder

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

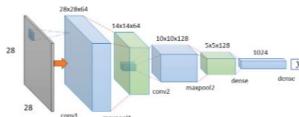
Tujuan utama dari generator adalah untuk menghasilkan gambar dari dimensi (64, 64, 3). Dibutuhkan vektor laten 100 dimensi dan beberapa informasi tambahan, y , dan mencoba untuk menghasilkan gambar yang realistik. Jaringan Generator adalah jaringan neural yang mendalam convolutional juga. Hal ini terdiri dari lapisan padat, upsampling, dan convolutional. Dibutuhkan dua nilai input: vektor 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

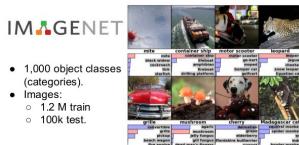
Jelaskan dengan ilustrasi gambar sendiri arsitektur discriminator network dari Age-cGAN.

Tujuan utama dari jaringan diskriminasi 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 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 9.5 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 diertai 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

Objektif Training ialah untuk meminimalkan loss function sebagai log likeli-

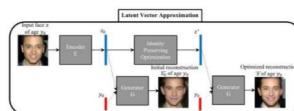
hood function yang diberikan pada persamaan dimana D melambangkan training data.

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

Gambar 9.8 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 9.9 Illustrasi Initial latent vector approximation

10. Berikan contoh perhitungan latent vector optimization



Gambar 9.10 Contoh Perhitungan Latent vector optimization

9.1.2 Praktek

1. Jelaskan bagaimana cara ekstrak le 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/
4     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 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)
17    x = LeakyReLU(alpha=0.2)(x)
18    x = Dropout(0.2)(x)
19
20    x = Dense(256 * 8 * 8)(x)
21    x = BatchNormalization()(x)
22    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')(
14        image_input)
15    x = LeakyReLU(alpha=0.2)(x)
16
17    label_input1 = Lambda(expand_label_input)(label_input)
18    x = concatenate([x, label_input1], axis=3)
19
20    x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
21    x = BatchNormalization()(x)
22    x = LeakyReLU(alpha=0.2)(x)
23
24    x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
25    x = BatchNormalization()(x)
26    x = LeakyReLU(alpha=0.2)(x)
27
28    x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
29    x = BatchNormalization()(x)
30    x = LeakyReLU(alpha=0.2)(x)
31
32    x = Flatten()(x)
33    x = Dense(1, activation='sigmoid')(x)
34
35    model = Model(inputs=[image_input, label_input], outputs=[x])
36    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=np.int64)
20    num_classes = len(set(y))
21    y = np.reshape(np.array(y), [len(y), 1])
22    y = to_categorical(y, num_classes=num_classes)
23
24    for epoch in range(epochs):
25        print("Epoch:", epoch)
26
27        encoder_losses = []

```

```

28
29         number_of_batches = int(z_i.shape[0] / batch_size)
30         print("Number of batches:", number_of_batches)
31         for index in range(number_of_batches):
32             print("Batch:", index + 1)
33
34             z_batch = z_i[index * batch_size:(index + 1) *
35             batch_size]
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 Penanganan Error

9.1.4 Bukti Tidak Plagiat



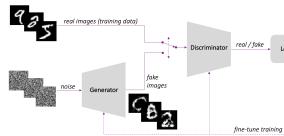
Gambar 9.11 Bukti Tidak Melakukan Plagiat Chapter 9

9.2 1174012 - Damara Benedikta

9.2.1 Teori

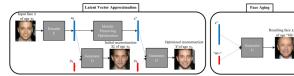
1. Jelaskan dengan ilustrasi gambar sendiri apa perbedaan antara vanilla GAN dan cGAN. Generator dan Disciminator pada Vanilla GAN adalah pembelajaran multi-layer yang sederhana, biasanya tidak memiliki convolutional Neural Jaringan (CNNs) di jaringannya dimana algoritmanya sederhana yang mencoba untuk mengoptimasi persamman matematika menggunakan penurunan gradien stokastik. Sedangkan cGAN dapat dideskripsikan sebagai deep learing pada parameter kondisional diberlakukan.Jenis GAN yang dikondisikan pada beberapa

informasi tambahan. informasi tambahan y ke Generator sebagai lapisan input tambahan.



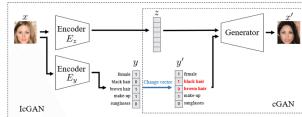
Gambar 9.12 Vanilla GAN-cGAN

2. Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN. Age cGAN mempunyai 4 jaringan Encoder, FaceNet, Jaringan Generator, dan jaringan diskriminator.yang di latih dalam 3 tahapan yaitu: Conditional GAN Training,Initial latent vector approximation,dan Latent vector optimization.



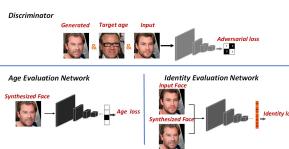
Gambar 9.13 Age-cGAN

3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari AgecGAN. Arsitektur encoder biasanya digunakan untuk mengenerate latent vector dari gambar yang akan diinputkan yang nantinya akan diteruskan ke generator. dimana 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. etiap blok convolutional berisi lapisan convolutional, lapisan normalisasi batch, dan fungsi aktivasi.



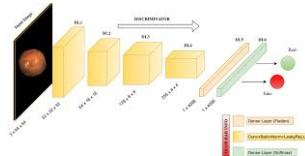
Gambar 9.14 Encoder Age cGANr

4. Jelaskan dengan ilustrasi gambar sendiri arsitektur generator network dari AgecGAN. Arsitektur generator adalah sebuah CNN dan mengambil 100-dimensional latent vector sebagai inputannya yang akan menghasilkan gambar realistik dalam dimensi (64,64,3). 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.



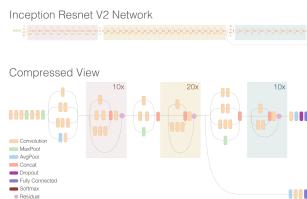
Gambar 9.15 Network Age cGAN

5. Jelaskan dengan ilustrasi gambar sendiri arsitektur discriminator network dari Age-cGAN. Arsitektur diskriminator adalah CNN yang fungsinya untuk memprediksi apakah gambar yang diberikan palsu atau asli. 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 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.



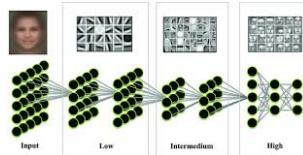
Gambar 9.16 Discriminator Age cGAN

6. Jelaskan dengan ilustrasi gambar apa itu pre-trained 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.



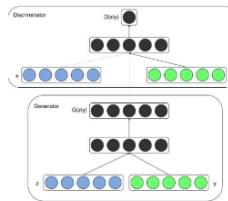
Gambar 9.17 Pretrained Inception ResNet

7. Jelaskan dengan ilustrasi gambar sendiri arsitektur Face recognition network Age-cGAN. Face Recognition digunakan untuk mengenali identitas seseorang dari gambar yang diberikan.



Gambar 9.18 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 diskriminato. 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.



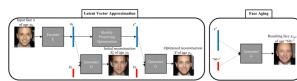
Gambar 9.19 Tahap Age cGAN

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 trainning data.

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

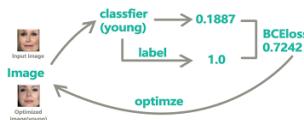
Gambar 9.20 Training Objektif

10. Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation. Latent vector approdimation kemampuan untuk membuat gambar yang realistik dan tajam serta menghasilkan gambar wajah pada usia target.



Gambar 9.21 Initial Latent Vector Approximation

- 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 z dalam ruang laten Z .



Gambar 9.22 Latent Vector Optimization

9.2.2 Praktek

- Jelaskan bagaimana cara ekstrak le dataset Age-cGAN menggunakan google colab. Kode dibawah ini digunakan untuk 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/
4 wiki_crop.tar")
4 tf.extractall(path="/content/drive/My Drive/Colab Notebooks")
```

- Jelaskan bagaimana kode program bekerja untuk melakukan load terhadap dataset yang sudah di ekstrak, termasuk bagaimana penjelasan kode program perhitungan usia. Kode dibawah berfungsi untuk melakukan fungsi perhitungan usia.

```
1 # In[2. Load Data]:
2 def load_data(wiki_dir, dataset='wiki'):
3     # MenLoad file wiki.mat
4     meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset)))
5
6     # MenLoad list dari semua file
7     full_path = meta[dataset][0, 0]["full_path"][0]
8
9     # List dari serial angka tanggal Matlab
10    dob = meta[dataset][0, 0]["dob"][0]
11
12    # List tahun kapan foto diambil
13    photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
14
```

```

15 # Menghitung umur dari semua dob
16 age = [calculate_age(photo_taken[i], dob[i]) for i in range(
17     len(dob))]
18
19 # Membuat list tupel yang berisikan pasangan antara lokasi
20 # gambar dan umur
21 images = []
22 age_list = []
23 for index, image_path in enumerate(full_path):
24     images.append(image_path[0])
25     age_list.append(age[index])
26
27 # Mengembalikan nilai list dari semua gambar dan umurnya
28 # masing-masing
29 return images, age_list

```

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

```

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)
17    x = LeakyReLU(alpha=0.2)(x)
18    x = Dropout(0.2)(x)
19
20    x = Dense(256 * 8 * 8)(x)
21    x = BatchNormalization()(x)
22    x = LeakyReLU(alpha=0.2)(x)
23    x = Dropout(0.2)(x)
24
25    x = Reshape((8, 8, 256))(x)
26
27    x = UpSampling2D(size=(2, 2))(x)
28    x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
29    x = BatchNormalization(momentum=0.8)(x)
30    x = LeakyReLU(alpha=0.2)(x)
31
32    x = UpSampling2D(size=(2, 2))(x)

```

5. Jelaskan bagaimana kode program The Discriminator Network bekerja dijelaskan dengan bahawa awam dengan ilustrasi sederhana. Berikut adalah proses Discriminator yang digunakan untuk membedakan antara gambar asli ataupun gambar palsu yang dihasilkan oleh generator.

```
1 # In [5. Discriminator Network Bekerja ]:
```

```

2 def build_discriminator():
3     """
4         Create a Discriminator Model with hyperparameters values
5             defined as follows
6         """
7     input_shape = (64, 64, 3)
8     label_shape = (6,)
9     image_input = Input(shape=input_shape)
10    label_input = Input(shape=label_shape)
11
12    x = Conv2D(64, kernel_size=3, strides=2, padding='same')(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 # 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)

```

```

17     gen_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
18                           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 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=
np.int64)
20    num_classes = len(set(y))
21    y = np.reshape(np.array(y), [len(y), 1])
22    y = to_categorical(y, num_classes=num_classes)
23
24    for epoch in range(epochs):
25        print("Epoch:", epoch)
26
27    encoder_losses = []
28
29    number_of_batches = int(z_i.shape[0] / batch_size)
30    print("Number of batches:", number_of_batches)
31    for index in range(number_of_batches):
32        print("Batch:", index + 1)
33
34        z_batch = z_i[index * batch_size:(index + 1) *
batch_size]
35        y_batch = y[index * batch_size:(index + 1) *
batch_size]
36
37        generated_images = generator.predict_on_batch([
z_batch, y_batch])
38
39        # Train the encoder model
40        encoder_loss = encoder.train_on_batch(
generated_images, z_batch)

```

```
41         print("Encoder loss:", encoder_loss)
42
43         encoder_losses.append(encoder_loss)
44
45         # Write the encoder loss to Tensorboard
46         write_log(tensorboard, "encoder_loss", np.mean(
47             encoder_losses), epoch)
48
49         # Save the encoder model
50         encoder.save_weights("encoder.h5")
```


BAB 10

CHAPTER 10

BAB 11

CHAPTER 11

BAB 12

CHAPTER 12

BAB 13

CHAPTER 13

BAB 14

CHAPTER 14
