CERDAS MENGUASAI GIT

CERDAS MENGUASAI GIT
Dalam 24 Jam

Rolly M. Awangga
Informatics Research Center

Kreatif Industri Nusantara

Penulis:
Rolly Maulana Awangga

ISBN : 978-602-53897-0-2

Editor:
M. Yusril Helmi Setyawan

Penyunting:
Syafrial Fachrie Pane
Khaera Tunnisa
Diana Asri Wijayanti

Desain sampul dan Tata letak:
Deza Martha Akbar

Penerbit:
Kreatif Industri Nusantara

Redaksi:
Jl. Ligar Nyawang No. 2
Bandung 40191
Tel. 022 2045-8529
Email : awangga@kreatif.co.id

Distributor:
Informatics Research Center
Jl. Sariasih No. 54
Bandung 40151
Email : irc@poltekpos.ac.id

Cetakan Pertama, 2019

'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

$\mathcal{B}$ Number of Beats

INTRODUCTION

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

Informatics Research Center
Bandung, Jawa Barat, Indonesia

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

Rumusnya cuman sebagai contoh aja biar keren[**?**].

$$ABC\mathcal{D}\mathcal{E}\mathcal{F}\alpha\beta\Gamma\Delta\sum_{def}^{abc} \tag{I.1}$$

BAB 1

CHAPTER 1

BAB 2

CHAPTER 2

BAB 3

CHAPTER 3

BAB 4

CHAPTER 4

BAB 5

CHAPTER 5

BAB 6

CHAPTER 6

BAB 7

CHAPTER 7

BAB 8

CHAPTER 8

8.1 Dwi Septiani Tsaniyah - 1174003

8.1.1 Teori

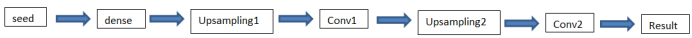
1. Jelaskan dengan ilustrasi gambar sendiri apa itu generator dengan perumpamaan anda sebagai mahasiswa sebagai generatornya. Generator adalah sebuah jaringan yang merubah inputan vektor menjadi gambar, seperti ada inputan vektor secara acak yaitu angka sembarang, maka angka tersebut akan diubah menjadi gambar yang sembarang pula. Sebagai ilustrasi apabila mahasiswa sebagai generator, mmisalkan 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 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.

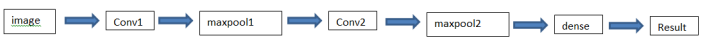
Gambar 8.2 Ilustrasi

3. Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur generator dibuat Generator menggunkan 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 di kompress, untuk contohnya misalnya ada 3 gambar yaitu 2 kursi yang berebeda dan 1 meja, maka hal hal yang mirip dai kursi tersebut(ciri-ciri utamanya), seeprti 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

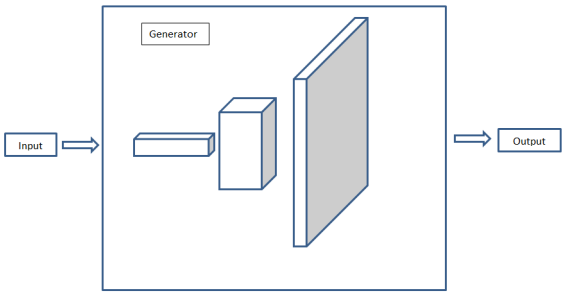
8. Sebutkan dan jelaskan contoh-contoh implementasi dari GAN Pada bidang mode, seni, dan iklan, GAN dapar digunakan untuk membuat foto-foto model fashion imajinier tanpa perlu menyewa model. Pada bidang sains GAN dapat meningkatkan citra astronomi dan mensimulasikan pelensaan gravitasi untuk penelitian 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 dis
```

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.

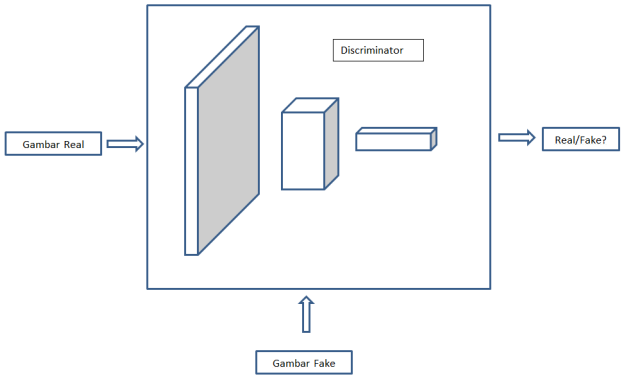


Gambar 8.7 Ilustrasi

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

```
diskrim=Sequential()#Inisiasi dari sequensia
diskrim.add(Dense(units=784,input_dim=np.sha
diskrim.add(Dense(units=400)) #Mensetting De
diskrim.add(Dense(units=200, activation='sig
diskrim.compile(loss='binary_crossentropy',
diskrim.summary()#Memproses data yang sudah
```

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 penujuk berapa nilai kesamaan anatara 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 realistis 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 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

8.1.2 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.

```
def build_discriminator():
    """
    Create a Discriminator Model using hyperparameters values
    defined as follows
    """

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

    dis_input_layer = Input(shape=dis_input_shape)

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

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

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
       as follows
4      """
5      z_size = 200
6      gen_filters = [512, 256, 128, 64, 1]
7      gen_kernel_sizes = [4, 4, 4, 4, 4]
8      gen_strides = [1, 2, 2, 2, 2]
9      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
```

diskrimanator 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 ketentukan 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

```
    """
    Specify Hyperparameters
    """
    object_name = "chair"
    data_dir = "3DShapeNets/volumetric_data/" \
               "{}/30/train/*.mat".format(object_name)
    gen_learning_rate = 0.0025
    dis_learning_rate = 10e-5
    beta = 0.5
    batch_size = 1
    z_size = 200
    epochs = 10
    MODE = "train"
```

artinya adalah load dataset yang hanya dalam folder chair data train

13. Jelaskan kode program

```
    """
    Create models
    """
    gen_optimizer = Adam(lr=gen_learning_rate, beta_1=beta)
    dis_optimizer = Adam(lr=dis_learning_rate, beta_1=beta)

    discriminator = build_discriminator()
    discriminator.compile(loss='binary_crossentropy', optimizer=
    dis_optimizer)

    generator = build_generator()
    generator.compile(loss='binary_crossentropy', optimizer=
    gen_optimizer)
```

disini menggunakan Adam sebagai algoritma pengoptimalan dan binary crossentropy sebagai kerugian loss.

14. Jelaskan kode program

```
    input_layer = Input(shape=(1, 1, 1, z_size))
    generated_volumes = generator(input_layer)
    validity = discriminator(generated_volumes)
    adversarial_model = Model(inputs=[input_layer], outputs=[
    validity])
    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

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

ini melakukan load data pada dataset

16. Jelaskan kode program

```
tensorboard = TensorBoard(log_dir="logs/{}".format(time.time())
    )
tensorboard.set_model(generator)
tensorboard.set_model(discriminator)
```

ini berfungsi untuk membuat tensorboard yang nantinya bisa diakses melalui localhost

17. Jelaskan kode program

```
labels_real = np.reshape(np.ones((batch_size,)), (-1, 1, 1,
1, 1))
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

```
if MODE == 'train':
    for epoch in range(epochs):
        print("Epoch:", epoch)

        gen_losses = []
        dis_losses = []
```

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

19. Jelaskan kode program

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

batch adalah jumlah file yang akan di training

20. Jelaskan kode program

```
            z_sample = np.random.normal(0, 0.33, size=[
batch_size, 1, 1, 1, z_size]).astype(np.float32)
            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

```
                # Next, generate volumes using the generate
    network
                gen_volumes = generator.predict_on_batch(z_sample
    )
```

ialah membuat sample gambar palsu yang akan diteruskan ke diskriminator

22. Jelaskan kode program

```
                """
                Train the discriminator network
                """
                discriminator.trainable = True
                if index % 2 == 0:
                    loss_real = discriminator.train_on_batch(
    volumes_batch, labels_real)
                    loss_fake = discriminator.train_on_batch(
    gen_volumes, labels_fake)

                    d_loss = 0.5 * np.add(loss_real, loss_fake)
                    print("d_loss:{}".format(d_loss))

                else:
                    d_loss = 0.0
```

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

```
                z = np.random.normal(0, 0.33, size=[batch_size,
    1, 1, 1, z_size]).astype(np.float32)
                g_loss = adversarial_model.train_on_batch(z,
    labels_real)
                print("g_loss:{}".format(g_loss))

                gen_losses.append(g_loss)
                dis_losses.append(d_loss)
```

dengan melakukan print g_loss untuk generator dan juga d loss untuk diskriminator

24. Jelaskan kode program

```
                # Every 10th mini-batch, generate volumes and
    save them
                if index % 10 == 0:
                    z_sample2 = np.random.normal(0, 0.33, size=[
    batch_size, 1, 1, 1, z_size]).astype(np.float32)
                    generated_volumes = generator.predict(
    z_sample2, verbose=3)
                    for i, generated_volume in enumerate(
    generated_volumes[:5]):
                        voxels = np.squeeze(generated_volume)
                        voxels[voxels < 0.5] = 0.
```

```
                        voxels[voxels >= 0.5] = 1.
                        saveFromVoxels(voxels, "results/img_{}_{}
_{}".format(epoch, index, i))
```

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

25. Jelaskan kode program

```
            # Write losses to Tensorboard
            write_log(tensorboard, 'g_loss', np.mean(gen_losses),
 epoch)
            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

```
        """
        Save models
        """
        generator.save_weights(os.path.join("models", "
generator_weights.h5"))
        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

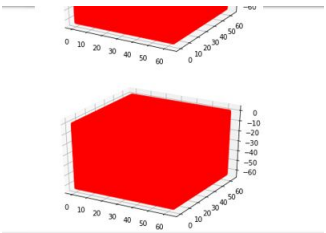
27. jelaskan kode program

```
    if MODE == 'predict':
        # Create models
        generator = build_generator()
        discriminator = build_discriminator()

        # Load model weights
        generator.load_weights(os.path.join("models", "
generator_weights.h5"), True)
        discriminator.load_weights(os.path.join("models", "
discriminator_weights.h5"), True)

        # Generate 3D models
        z_sample = np.random.normal(0, 1, size=[batch_size, 1, 1,
 1, z_size]).astype(np.float32)
        generated_volumes = generator.predict(z_sample, verbose
=3)

        for i, generated_volume in enumerate(generated_volumes
[:2]):
            voxels = np.squeeze(generated_volume)
            voxels[voxels < 0.5] = 0.
            voxels[voxels >= 0.5] = 1.
            saveFromVoxels(voxels, "results/gen_{}".format(i))
```

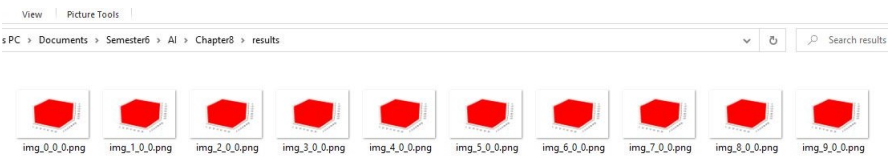
ini adalah tahap akhir untuk melakukan testing dari model yang telah dibuat dan buat model dari yang sudah di create sebelumnya yaitu generator dan diskriminator.

```
Loading data...
Data loaded...
Epoch: 0
Number of batches: 10
Batch: 1
d_loss:0.6931722164154053
g_loss:0.6931138038635254
Batch: 2
g_loss:0.6931138038635254
Batch: 3
d_loss:0.6931576728820801
g_loss:0.6931090354919434
Batch: 4
g_loss:0.6931090354919434
Batch: 5
d_loss:0.6931560039520264
g_loss:0.693108081817627
Batch: 6
g_loss:0.693108081817627
```

Gambar 8.9 hasil



Gambar 8.10 hasil



Gambar 8.11 hasil

BAB 9

CHAPTER 9

9.1 Dwi Septiani Tsaniyah - 1174003

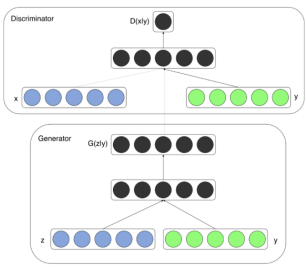
9.1.1 Teori

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.

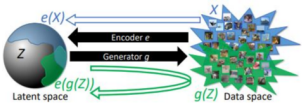
   Arsitektur cGAN untuk penuaan wajah sedikit lebih rumit. AgecGan terdiri dari empat jaringan: Encoder, FaceNet, Jaringan Generator, dan jaringan diskriminator. Dengan Encoder, kita belajar pemetaan invers gambar wajah masukan dan kondisi usia dengan vektor laten. FaceNet adalah jaringan pengenalan wajah 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 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 9.2 Illustrasi Arsitektur cGAN

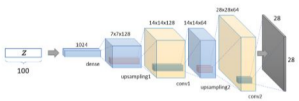
3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari AgecGAN.
   Tujuan utama dari jaringan Encoder adalah untuk menghasilkan vektor laten dari gambar yang disediakan. Pada dasarnya, dibutuhkan gambar dimensi (64, 64, 3) dan mengubahnya menjadi vektor 100-dimensi. Jaringan Encoder 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

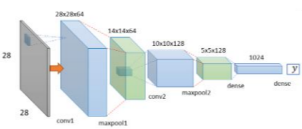
4. Jelaskan dengan ilustrasi gambar sendiri arsitektur generator network dari Agec-GAN.
   Tujuan utama dari generator adalah untuk menghasilkan gambar dari dimensi (64, 64, 3). Dibutuhkan vektor laten 100 dimensi dan beberapa informasi tambahan, y, dan mencoba untuk menghasilkan gambar yang realistis. 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

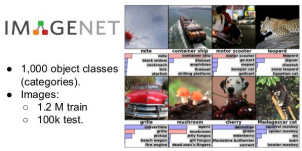
Jelaskan dengan ilustrasi gambar sendiri arsitektur discriminator network dari Age-cGAN.

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

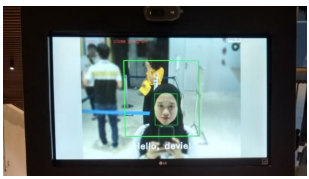
5. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model.
pre-trained Inception-ResNet-2 network, sekali disediakan dengan gambar, mengembalikan yang sesuai embedding. Tertanam yang diekstrak untuk gambar asli dan gambar direkonstruksi dapat dihitung dengan menghitung jarak Euclidean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

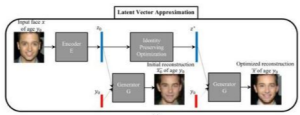
8. Berikan contoh perhitungan fungsi training objektif
Objektif Trainning ialah untuk meminimalkan loss function sebagai log likeli-

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

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

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.

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

```
# In[2. Load Data]:
def load_data(wiki_dir, dataset='wiki'):
    # Load the wiki.mat file
    meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset
    )))

    # Load the list of all files
    full_path = meta[dataset][0, 0]["full_path"][0]

    # List of Matlab serial date numbers
    dob = meta[dataset][0, 0]["dob"][0]

    # List of years when photo was taken
    photo_taken = meta[dataset][0, 0]["photo_taken"][0]  # year

    # Calculate age for all dobs
    age = [calculate_age(photo_taken[i], dob[i]) for i in range(
    len(dob))]

    # Create a list of tuples containing a pair of an image path
    and age
    images = []
    age_list = []
    for index, image_path in enumerate(full_path):
        images.append(image_path[0])
        age_list.append(age[index])

    # Return a list of all images and respective age
    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.

```
# In[3. Encoder Bekerja]:
def build_encoder():
    """
    Encoder Network
    """
    input_layer = Input(shape=(64, 64, 3))

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

    # 2nd Convolutional Block
    enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='
    same')(enc)
```

```
    enc = BatchNormalization()(enc)
    enc = LeakyReLU(alpha=0.2)(enc)

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

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

    # Flatten layer
    enc = Flatten()(enc)

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

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

    # Create a model
    model = Model(inputs=[input_layer], outputs=[enc])
    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.

```
# In[4. Generator Network Bekerja]:
def build_generator():
    """
    Create a Generator Model with hyperparameters values defined
    as follows
    """
    latent_dims = 100
    num_classes = 6

    input_z_noise = Input(shape=(latent_dims,))
    input_label = Input(shape=(num_classes,))

    x = concatenate([input_z_noise, input_label])

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

    x = Dense(256 * 8 * 8)(x)
    x = BatchNormalization()(x)
    x = LeakyReLU(alpha=0.2)(x)
```

```
    x = Dropout(0.2)(x)

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

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

    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.

```
# In[5.Discriminator Network Bekerja]:
def build_discriminator():
    """
    Create a Discriminator Model with hyperparameters values
    defined as follows
    """
    input_shape = (64, 64, 3)
    label_shape = (6,)
    image_input = Input(shape=input_shape)
    label_input = Input(shape=label_shape)

    x = Conv2D(64, kernel_size=3, strides=2, padding='same')(
    image_input)
    x = LeakyReLU(alpha=0.2)(x)

    label_input1 = Lambda(expand_label_input)(label_input)
    x = concatenate([x, label_input1], axis=3)

    x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
    x = BatchNormalization()(x)
    x = LeakyReLU(alpha=0.2)(x)

    x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
    x = BatchNormalization()(x)
    x = LeakyReLU(alpha=0.2)(x)

    x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
    x = BatchNormalization()(x)
    x = LeakyReLU(alpha=0.2)(x)

    x = Flatten()(x)
    x = Dense(1, activation='sigmoid')(x)

    model = Model(inputs=[image_input, label_input], outputs=[x])
    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 sebanuak 500 kali.

```
# In[6. Training cGAN]:
    if __name__ == '__main__':
    # Define hyperparameters
    data_dir = "data"
    wiki_dir = os.path.join(data_dir, "wiki_crop1")
    epochs = 500
    batch_size = 2
    image_shape = (64, 64, 3)
    z_shape = 100
    TRAIN_GAN = True
    TRAIN_ENCODER = False
    TRAIN_GAN_WITH_FR = False
    fr_image_shape = (192, 192, 3)

    # Define optimizers
    dis_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
    epsilon=10e-8)
    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 predicsi epoch yang telah di buat sebanyak 500 kali, dan nanti hasilnya ada di folder result.

```
# In[7. Laten Vector]:
    """
    Train encoder
    """

    if TRAIN_ENCODER:
        # Build and compile encoder
        encoder = build_encoder()
        encoder.compile(loss=euclidean_distance_loss, optimizer='
    adam')

        # Load the generator network's weights
        try:
            generator.load_weights("generator.h5")
        except Exception as e:
            print("Error:", e)

        z_i = np.random.normal(0, 1, size=(5000, z_shape))

        y = np.random.randint(low=0, high=6, size=(5000,), dtype=
    np.int64)
        num_classes = len(set(y))
        y = np.reshape(np.array(y), [len(y), 1])
        y = to_categorical(y, num_classes=num_classes)

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

            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(
    encoder_losses), epoch)
47
48      # Save the encoder model
49      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

BAB 10

CHAPTER 10

BAB 11

CHAPTER 11

BAB 12

CHAPTER 12

BAB 13

CHAPTER 13

BAB 14

CHAPTER 14