

CERDAS MENGUASAI GIT

CERDAS MENGUASAI GIT

Dalam 24 Jam

Rolly M. Awangga
Informatics Research Center



Kreatif Industri Nusantara

Penulis:

Rolly Maulana Awangga

ISBN : 978-602-53897-0-2

Editor:

M. Yusril Helmi Setyawan

Penyunting:

Syafrial Fachrie Pane

Khaera Tunnisa

Diana Asri Wijayanti

Desain sampul dan Tata letak:

Deza Martha Akbar

Penerbit:

Kreatif Industri Nusantara

Redaksi:

Jl. Ligar Nyawang No. 2

Bandung 40191

Tel. 022 2045-8529

Email : awangga@kreatif.co.id

Distributor:

Informatics Research Center

Jl. Sariasih No. 54

Bandung 40151

Email : irc@poltekpos.ac.id

Cetakan Pertama, 2019

Hak cipta dilindungi undang-undang

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

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

CONTRIBUTORS

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

CONTENTS IN BRIEF

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[?].

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

BAB 1

CHAPTER 1

BAB 2

CHAPTER 2

BAB 3

CHAPTER 3

BAB 4

CHAPTER 4

BAB 5

CHAPTER 5

BAB 6

CHAPTER 6

BAB 7

CHAPTER 7

BAB 8

CHAPTER 8

BAB 9

CHAPTER 9

9.1 1174008 - Arjun Yuda Firwanda

9.1.1 Teori

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

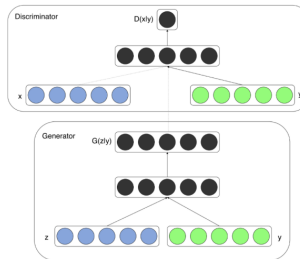
Vanilla GAN Vanilla GAN adalah tipe GAN paling sederhana. Di sini, Generator dan Diskriminator adalah perceptron multi-layer sederhana. Dalam vanilla GAN, algoritma ini sangat sederhana, ia mencoba untuk mengoptimalkan persamaan matematika menggunakan keturunan gradien stokastik. CGAN (Conditional GAN), label bertindak sebagai ekstensi ke ruang laten z untuk menghasilkan dan membedakan gambar dengan lebih baik.



Gambar 9.1 Valina GAN-cGAN

2. Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN.

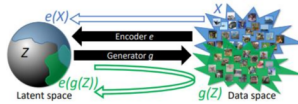
Age cGAN ialah dengan mengkondisikan model pada informasi tambahan dimungkinkan untuk mengarahkan proses pembuatan data. Pengkondisian semacam itu dapat didasarkan pada label kelas.



Gambar 9.2 Age-cGAN

3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari Agec-GAN.

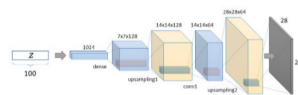
Arsitektur encoder biasanya digunakan untuk memodelkan struktur manifold dan membalikkan encoder untuk memproses data.



Gambar 9.3 Encoder Age cGANr

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

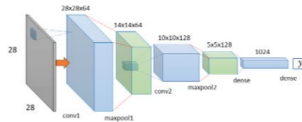
Arsitektur generator adalah sebuah array yang digunakan secara random, yang disebut seed. dari data seed tersebut, generator akan merubahnya menjadi sebuah gambar yang ukuran 28 x 28 dengan menggunakan Convolutional Neural Network.



Gambar 9.4 Network Age cGAN

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

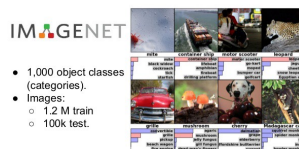
Arsitektur diskriminator adalah CNN yang dapat menerima input gambar yang berukuran 28,28 serta menghasilkan angka biner yang menyatakan apakah data yang diinputkan merupakan dataset asli atau gambar dataset palsu.



Gambar 9.5 Discriminator Age cGAN

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

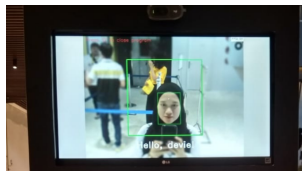
Pre-Trained Network atau Transfer Learning merupakan suatu metode penyelesaian yang memanfaatkan model yang sudah dilatih terhadap suatu dataset untuk menyelesaikan masalah dengan cara menggunakan sebagai starting point, memodifikasi dan mengupdate parameternya, sehingga sesuai dengan dataset yang baru.



Gambar 9.6 Pretrained Inception ResNet

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

Face Recognition merupakan salah satu sistem yang mengimplementasi Deep Learning yang dapat mengenali wajah secara fisik dari gambar digital atau video frame.

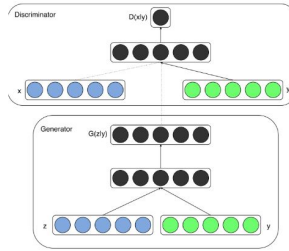


Gambar 9.7 Face recognition network Age-cGAN

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

Pada dari Age-cGan ni terdapat 2 tahapan dengan generator dan diskrimina-
tor. dimana untuk tahap generator sendiri membutuhkan vektor laten 100 serta

menghasilkan gambar yang realistis dari dimensinya. sedangkan tahap diskriminator itu tahapan dimana memprediksi gambar yang diberikan nyata atau palsu.



Gambar 9.8 Tahap Age cGAN

9. Berikan contoh perhitungan fungsi training objektif.

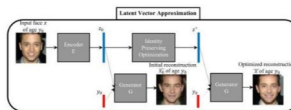
Objektif Training ialah untuk meminimalkan loss function sebagai log likelihood function yang diberikan pada persamaan dimana D melambangkan training data.

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

Gambar 9.9 Training Objektif

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

Latent vector approdimation kemampuan untuk membuat gamar yang realistis dan tajam serta menghasilkan gambar wajah pada usia target.



Gambar 9.10 Initial Latent Vector Approximation

11. Berikan contoh perhitungan latent vector optimization.

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



Gambar 9.11 Latent Vector Optimization

9.1.2 Praktek

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

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

2. Jelaskan bagaimana kode program bekerja untuk melakukan load terhadap dataset yang sudah di ekstrak, termasuk bagaimana penjelasan kode program perhitungan usia. 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 bahasa awam dengan ilustrasi sederhana. Proses Encoder berfungsi untuk mempelajari pemetaan terbalik dari gambar wajah dan kondisi usia dengan vector latent Z .

```

1 # In[3. Encoder Bekerja]:
2 def build_encoder():
3     """
4     Encoder Network
5     """
6     input_layer = Input(shape=(64, 64, 3))
7
8     # 1st Convolutional Block
9     enc = Conv2D(filters=32, kernel_size=5, strides=2, padding='
10     same')(input_layer)
11     # enc = BatchNormalization()(enc)
12     enc = LeakyReLU(alpha=0.2)(enc)
13
14     # 2nd Convolutional Block
15     enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='
16     same')(enc)
17     enc = BatchNormalization()(enc)
18     enc = LeakyReLU(alpha=0.2)(enc)
19
20     # 3rd Convolutional Block
21     enc = Conv2D(filters=128, kernel_size=5, strides=2, padding='
22     same')(enc)
23     enc = BatchNormalization()(enc)
24     enc = LeakyReLU(alpha=0.2)(enc)
25
26     # 4th Convolutional Block
27     enc = Conv2D(filters=256, kernel_size=5, strides=2, padding='
28     same')(enc)
29     enc = BatchNormalization()(enc)
30     enc = LeakyReLU(alpha=0.2)(enc)
31
32     # Flatten layer
33     enc = Flatten()(enc)
34
35     # 1st Fully Connected Layer
36     enc = Dense(4096)(enc)
37     enc = BatchNormalization()(enc)
38     enc = LeakyReLU(alpha=0.2)(enc)
39
40     # Second Fully Connected Layer
41     enc = Dense(100)(enc)
42
43     # Create a model
44     model = Model(inputs=[input_layer], outputs=[enc])
45     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     latent_dims = 100
8     num_classes = 6
9
10    input_z_noise = Input(shape=(latent_dims,))
11    input_label = Input(shape=(num_classes,))
12
13    x = concatenate([input_z_noise, input_label])
14
15    x = Dense(2048, input_dim=latent_dims + num_classes)(x)
16    x = LeakyReLU(alpha=0.2)(x)
17    x = Dropout(0.2)(x)
18
19    x = Dense(256 * 8 * 8)(x)
20    x = BatchNormalization()(x)
21    x = LeakyReLU(alpha=0.2)(x)
22    x = Dropout(0.2)(x)
23
24    x = Reshape((8, 8, 256))(x)
25
26    x = UpSampling2D(size=(2, 2))(x)
27    x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
28    x = BatchNormalization(momentum=0.8)(x)
29    x = LeakyReLU(alpha=0.2)(x)
30
31    x = UpSampling2D(size=(2, 2))(x)

```

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

```

1 # In[5. Discriminator Network Bekerja]:
2 def build_discriminator():
3     """
4     Create a Discriminator Model with hyperparameters values
5     defined as follows
6     """
7     input_shape = (64, 64, 3)
8     label_shape = (6,)
9     image_input = Input(shape=input_shape)
10    label_input = Input(shape=label_shape)
11
12    x = Conv2D(64, kernel_size=3, strides=2, padding='same')(
13        image_input)
14    x = LeakyReLU(alpha=0.2)(x)
15
16    label_input1 = Lambda(expand_label_input)(label_input)
17    x = concatenate([x, label_input1], axis=3)
18
19    x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
20    x = BatchNormalization()(x)

```



```

19     x = LeakyReLU(alpha=0.2)(x)
20
21     x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
22     x = BatchNormalization()(x)
23     x = LeakyReLU(alpha=0.2)(x)
24
25     x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
26     x = BatchNormalization()(x)
27     x = LeakyReLU(alpha=0.2)(x)
28
29     x = Flatten()(x)
30     x = Dense(1, activation='sigmoid')(x)
31
32     model = Model(inputs=[image_input, label_input], outputs=[x])
33     return model

```

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

```

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

```

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

```

1 # In[7. Laten Vector]:
2     """
3     Train encoder
4     """
5
6     if TRAIN_ENCODER:
7         # Build and compile encoder
8         encoder = build_encoder()

```

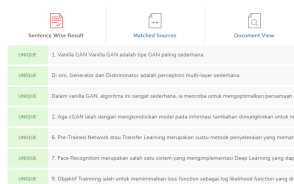
```

9         encoder.compile(loss=euclidean_distance_loss, optimizer='
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(
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



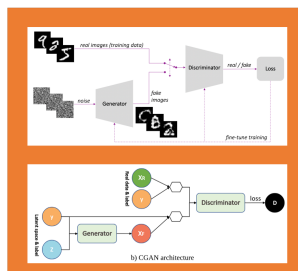
Gambar 9.12 Bukti Tidak Melakukan Plagiat Chapter 9

9.2 1174021 - Muhammad Fahmi

9.2.1 Soal Teori

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

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



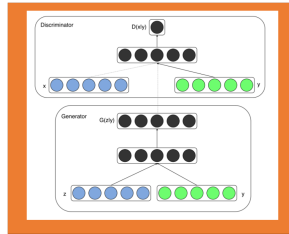
Gambar 9.13 Teori 1

2. Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN.

Untuk arsitektur dari Age cGAN mempunyai 4 yaitu :

- Encoder
- FaceNet
- Generator
- Discriminator

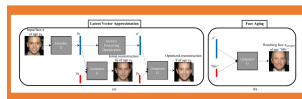
Untuk ilustrasi, lihat gambar berikut:



Gambar 9.14 Teori 2

3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari Agec-GAN.

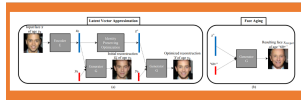
Encoder mempelajari pemetaan terbalik dari gambar wajah input dan kondisi usia dengan vektor laten Z . Jaringan encoder menghasilkan vektor laten dari gambar input. Jaringan Encoder adalah CNN yang mengambil gambar dari dimensi $(64, 64, 3)$ dan mengubahnya menjadi vektor 100 dimensi. Ada empat blok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



Gambar 9.15 Teori 3

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

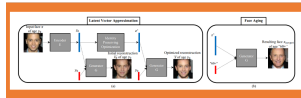
Generator dibutuhkan representasi tersembunyi dari gambar wajah dan vektor kondisi sebagai input dan menghasilkan gambar. Generator adalah CNN dan dibutuhkan vektor laten 100 dimensi dan vektor kondisi y , dan mencoba menghasilkan gambar realistis dari dimensi $(64, 64, 3)$. Generator memiliki lapisan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



Gambar 9.16 Teori 4

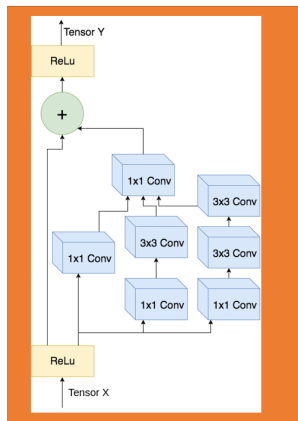
5. Jelaskan dengan ilustrasi gambar sendiri arsitektur discriminator network dari Age-cGAN.

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



Gambar 9.17 Teori 5

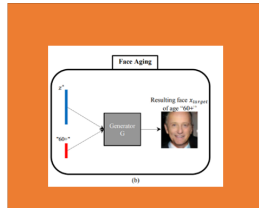
6. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model. Model Inception-ResNet v2 adalah model yang ciptakan untuk keperluan klasifikasi image dengan bobot di ImageNet.



Gambar 9.18 Teori 6

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

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



Gambar 9.19 Teori 7

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

- Input
- Training
- Testing

9. Berikan contoh perhitungan fungsi training objektif

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

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

Gambar 9.20 Teori 9

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

Initial latent vector approximation: Encoder network training adakah sebuah metode perkiraan awal vektor laten digunakan untuk memperkirakan vektor laten untuk mengoptimalkan rekonstruksi gambar wajah. Encoder adalah jaringan

saraf yang mendekati vektor laten. Kami melatih jaringan encoder pada gambar yang dihasilkan dan gambar nyata. Setelah dilatih, jaringan encoder akan mulai menghasilkan vektor laten dari distribusi yang dipelajari. Fungsi tujuan pelatihan untuk melatih jaringan encoder adalah kehilangan jarak Euclidean.

11. Berikan contoh perhitungan latent vector optimization

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

Gambar 9.21 Teori 11

9.2.2 Praktek Program

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

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

Kode di atas akan melakukan mount dan extract dataset.

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

```
1 def calculate_age(taken, dob):
2     birth = datetime.fromordinal(max(int(dob) - 366, 1))
3
4     if birth.month < 7:
5         return taken - birth.year
6     else:
7         return taken - birth.year - 1
```

```

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

```

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

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

```

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

```



```

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

```

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

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

```

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

```

```

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

```

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

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

```

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

```

```

31     model = Model(inputs=[image_input, label_input], outputs=[x])
32     return model

```

Diskriminator mencoba untuk membedakan antara gambar asli dan gambar palsu.

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

```

1  if __name__ == '__main__':
2      # Define hyperparameters
3      data_dir = "data"
4      wiki_dir = os.path.join(data_dir, "wiki_crop1")
5      epochs = 500
6      batch_size = 2
7      image_shape = (64, 64, 3)
8      z_shape = 100
9      TRAIN_GAN = True
10     TRAIN_ENCODER = False
11     TRAIN_GAN_WITH_FR = False
12     fr_image_shape = (192, 192, 3)
13
14     # Define optimizers
15     dis_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
16                          epsilon=10e-8)
17     gen_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
18                          epsilon=10e-8)
19     adversarial_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=
20     =0.999, epsilon=10e-8)
21
22     """
23     Build and compile networks
24     """
25
26     # Build and compile the discriminator network
27     discriminator = build_discriminator()
28     discriminator.compile(loss=['binary_crossentropy'], optimizer
29     =dis_optimizer)
30
31     # Build and compile the generator network
32     generator = build_generator()
33     generator.compile(loss=['binary_crossentropy'], optimizer=
34     gen_optimizer)
35
36     # Build and compile the adversarial model
37     discriminator.trainable = False
38     input_z_noise = Input(shape=(100,))
39     input_label = Input(shape=(6,))
40     recons_images = generator([input_z_noise, input_label])
41     valid = discriminator([recons_images, input_label])
42     adversarial_model = Model(inputs=[input_z_noise, input_label
43     ], outputs=[valid])
44     adversarial_model.compile(loss=['binary_crossentropy'],
45     optimizer=gen_optimizer)

```

```

39     tensorboard = TensorBoard(log_dir="logs/{}".format(time.time
40     ()))
41     tensorboard.set_model(generator)
42     tensorboard.set_model(discriminator)
43
44     """
45     Load the dataset
46     """
47     images, age_list = load_data(wiki_dir=wiki_dir, dataset="wiki
48     ")
49     age_cat = age_to_category(age_list)
50     final_age_cat = np.reshape(np.array(age_cat), [len(age_cat),
51     1])
52     classes = len(set(age_cat))
53     y = to_categorical(final_age_cat, num_classes=len(set(age_cat)
54     ))
55
56     loaded_images = load_images(wiki_dir, images, (image_shape
57     [0], image_shape[1]))
58
59     # Implement label smoothing
60     real_labels = np.ones((batch_size, 1), dtype=np.float32) *
61     0.9
62     fake_labels = np.zeros((batch_size, 1), dtype=np.float32) *
63     0.1
64
65     """
66     Train the generator and the discriminator network
67     """
68     if TRAIN_GAN:
69         for epoch in range(epochs):
70             print("Epoch:{}".format(epoch))
71
72             gen_losses = []
73             dis_losses = []
74
75             number_of_batches = int(len(loaded_images) /
76             batch_size)
77             print("Number of batches:", number_of_batches)
78             for index in range(number_of_batches):
79                 print("Batch:{}".format(index + 1))
80
81                 images_batch = loaded_images[index * batch_size:(
82                 index + 1) * batch_size]
83                 images_batch = images_batch / 127.5 - 1.0
84                 images_batch = images_batch.astype(np.float32)
85
86                 y_batch = y[index * batch_size:(index + 1) *
87                 batch_size]
88                 z_noise = np.random.normal(0, 1, size=(batch_size
89                 , z_shape))
90
91             """
92             Train the discriminator network
93             """

```

```

84         # Generate fake images
85         initial_recon_images = generator.predict_on_batch
([z_noise, y_batch])
86
87         d_loss_real = discriminator.train_on_batch([
images_batch, y_batch], real_labels)
88         d_loss_fake = discriminator.train_on_batch([
initial_recon_images, y_batch], fake_labels)
89
90         d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)
91         print("d_loss:{}".format(d_loss))
92
93         """
94         Train the generator network
95         """
96
97         z_noise2 = np.random.normal(0, 1, size=(
batch_size, z_shape))
98         random_labels = np.random.randint(0, 6,
batch_size).reshape(-1, 1)
99         random_labels = to_categorical(random_labels, 6)
100
101         g_loss = adversarial_model.train_on_batch([
z_noise2, random_labels], [1] * batch_size)
102
103         print("g_loss:{}".format(g_loss))
104
105         gen_losses.append(g_loss)
106         dis_losses.append(d_loss)
107
108         # Write losses to Tensorboard
109         write_log(tensorboard, 'g_loss', np.mean(gen_losses),
epoch)
110         write_log(tensorboard, 'd_loss', np.mean(dis_losses),
epoch)
111
112         """
113         Generate images after every 10th epoch
114         """
115         if epoch % 10 == 0:
116             images_batch = loaded_images[0:batch_size]
117             images_batch = images_batch / 127.5 - 1.0
118             images_batch = images_batch.astype(np.float32)
119
120             y_batch = y[0:batch_size]
121             z_noise = np.random.normal(0, 1, size=(batch_size
, z_shape))
122
123             gen_images = generator.predict_on_batch([z_noise,
y_batch])
124
125             for i, img in enumerate(gen_images[:5]):
126                 save_rgb_img(img, path="results/img-{}-{}.png
".format(epoch, i))
127
128         # Save networks

```

```

129         try :
130             generator.save_weights("generator.h5")
131             discriminator.save_weights("discriminator.h5")
132         except Exception as e:
133             print("Error:", e)

```

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

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

```

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

```



```

87         for epoch in range(epochs):
88             print("Epoch:", epoch)
89
90             reconstruction_losses = []
91
92             number_of_batches = int(len(loader.images) /
batch_size)
93             print("Number of batches:", number_of_batches)
94             for index in range(number_of_batches):
95                 print("Batch:", index + 1)
96
97                 images_batch = loader.images[index * batch_size:(
index + 1) * batch_size]
98                 images_batch = images_batch / 127.5 - 1.0
99                 images_batch = images_batch.astype(np.float32)
100
101                 y_batch = y[index * batch_size:(index + 1) *
batch_size]
102
103                 images_batch_resized = image_resizer.
predict_on_batch(images_batch)
104
105                 real_embeddings = fr_model.predict_on_batch(
images_batch_resized)
106
107                 reconstruction_loss = fr_adversarial_model.
train_on_batch([images_batch, y_batch], real_embeddings)
108
109                 print("Reconstruction loss:", reconstruction_loss
)
110
111                 reconstruction_losses.append(reconstruction_loss)
112
113             # Write the reconstruction loss to Tensorboard
114             write_log(tensorboard, "reconstruction_loss", np.mean
(reconstruction_losses), epoch)
115
116             """
117             Generate images
118             """
119             if epoch % 10 == 0:
120                 images_batch = loader.images[0:batch_size]
121                 images_batch = images_batch / 127.5 - 1.0
122                 images_batch = images_batch.astype(np.float32)
123
124                 y_batch = y[0:batch_size]
125                 z_noise = np.random.normal(0, 1, size=(batch_size
, z_shape))
126
127                 gen_images = generator.predict_on_batch([z_noise,
y_batch])
128
129                 for i, img in enumerate(gen_images[:5]):
130                     save_rgb_img(img, path="results/img_opt_{_}
_{_}.png".format(epoch, i))
131

```



```

132 # Save improved weights for both of the networks
133 generator.save_weights("generator_optimized.h5")
134 encoder.save_weights("encoder_optimized.h5")

```

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

9.2.3 Penanganan Error

1. ValueError



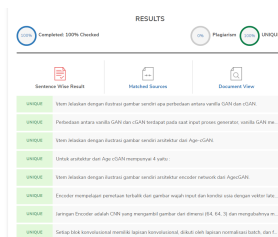
Gambar 9.22 FileNotFoundError

2. Cara Penanganan Error

- FileNotFoundError

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

9.2.4 Bukti Tidak Plagiat



Gambar 9.23 Bukti Tidak Melakukan Plagiat Chapter 9, CIE COPAS

BAB 10

CHAPTER 10

BAB 11

CHAPTER 11

BAB 12

CHAPTER 12

BAB 13

CHAPTER 13

BAB 14

CHAPTER 14
