

CERDAS MENGUASAI GIT

CERDAS MENGUASAI GIT

Dalam 24 Jam

Rolly M. Awangga
Informatics Research Center



Kreatif Industri Nusantara

Penulis:

Rolly Maulana Awangga

ISBN : 978-602-53897-0-2

Editor:

M. Yusril Helmi Setyawan

Penyunting:

Syafrial Fachrie Pane

Khaera Tunnisa

Diana Asri Wijayanti

Desain sampul dan Tata letak:

Deza Martha Akbar

Penerbit:

Kreatif Industri Nusantara

Redaksi:

Jl. Ligar Nyawang No. 2

Bandung 40191

Tel. 022 2045-8529

Email : awangga@kreatif.co.id

Distributor:

Informatics Research Center

Jl. Sariasih No. 54

Bandung 40151

Email : irc@poltekpos.ac.id

Cetakan Pertama, 2019

Hak cipta dilindungi undang-undang

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

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

CONTRIBUTORS

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

CONTENTS IN BRIEF

1 Chapter 1	1
2 Chapter 2	3
3 Chapter 3	5
4 Chapter 4	7
5 Chapter 5	9
6 Chapter 6	11
7 Chapter 7	13
8 Chapter 8	15
9 Chapter 9	17
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	xxiii
<i>Rolly Maulana Awangga, S.T., M.T.</i>	
1 Chapter 1	1
2 Chapter 2	3
3 Chapter 3	5
4 Chapter 4	7
	ix

5	Chapter 5	9
6	Chapter 6	11
7	Chapter 7	13
8	Chapter 8	15
9	Chapter 9	17
9.1	1174006 - Kadek Diva Krishna Murti	17
9.1.1	Teori	17
9.1.2	Praktek	21
9.1.3	Penanganan Error	26
9.2	1174017 - Muh. Rifky Prananda	26
9.2.1	Teori	26
9.2.2	Praktek	30
9.2.3	Penanganan Error	37
9.2.4	Bukti Tidak Plagiat	38
9.2.5	Link Youtube	38
10	Chapter 10	39
11	Chapter 11	41
12	Chapter 12	43
13	Chapter 13	45
14	Chapter 14	47
	Daftar Pustaka	49
	Index	51

FOREWORD

Sepatah kata dari Kaprodi, Kabag Kemahasiswaan dan Mahasiswa

KATA PENGANTAR

Buku ini diciptakan bagi yang awam dengan git sekalipun.

R. M. AWANGGA

*Bandung, Jawa Barat
Februari, 2019*

ACKNOWLEDGMENTS

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

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

R. M. A.

ACRONYMS

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

GLOSSARY

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

SYMBOLS

- A Amplitude
- $\&$ Propositional logic symbol
- a Filter Coefficient

- \mathcal{B} Number of Beats

INTRODUCTION

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

Informatics Research Center
Bandung, Jawa Barat, Indonesia

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

Rumusnya cuman sebagai contoh aja biar keren[1].

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

BAB 1

CHAPTER 1

BAB 2

CHAPTER 2

BAB 3

CHAPTER 3

BAB 4

CHAPTER 4

BAB 5

CHAPTER 5

BAB 6

CHAPTER 6

BAB 7

CHAPTER 7

BAB 8

CHAPTER 8

BAB 9

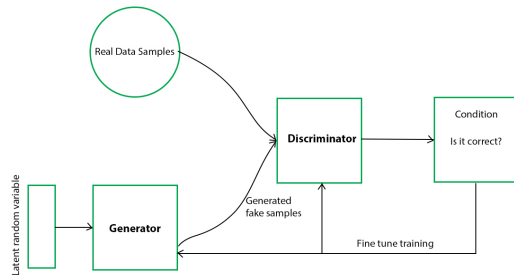
CHAPTER 9

9.1 1174006 - Kadek Diva Krishna Murti

9.1.1 Teori

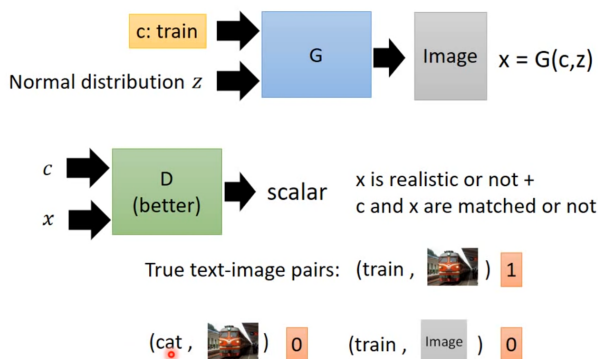
1. Jelaskan dengan ilustrasi gambar sendiri apa perbedaan antara vanilla GAN dan cGAN.

Vanilla GAN merupakan tipe GAN yang paling sederhana. Ia memiliki dua network diantaranya generator network dan discriminator network. Kedua network tersebut akan dilatih dan diadu satu sama lain. Generator network dilatih untuk membodohi discriminator network dengan membuat data baru dari data asli. Discriminator dilatih untuk tidak dibodohi oleh generator network



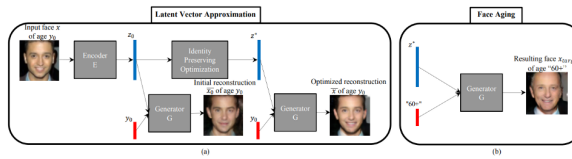
CGAN merupakan tipe GAN yang menerapkan metode deep learning dimana memiliki beberapa parameter kondisional pada generator dan discriminator. Parameter tersebut bisa berupa informasi label untuk membantu membedakan data asli dan data palsu.

Conditional GAN



2. Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN. Age-cGAN terdiri dari empat network, diantaranya:

- Encoder, network ini memetakan data input ke dalam bentuk latent vector.
- FaceNet, network ini belajar mengenali perbedaan antara data input dengan data hasil generate
- Generator Network, network ini mengenerate data baru dari inputan yang ada.
- Discriminator Network, network ini mencoba membedakan antara data asli dan data palsu.



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

Encoder network memetakan data input ke dalam bentuk latent vector.

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

Generator network mengenerate data baru dari inputan yang ada.

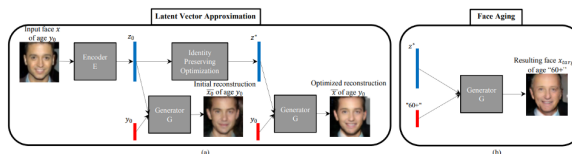
5. Jelaskan dengan ilustrasi gambar sendiri arsitektur discriminator network dari Agec-GAN.

Discriminator network mencoba membedakan antara data asli dan data palsu.

6. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model. Pretrained Inception-ResNet-2 Model adalah sebuah model terlatih berdasar jutaan gambar dari ImageNet

7. Jelaskan dengan ilustrasi gambar sendiri arsitektur Face recognition network Agec-GAN.

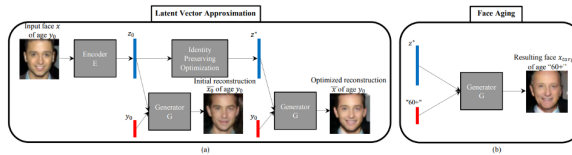
- Encoder, network ini memetakan data input ke dalam bentuk latent vector.
- FaceNet, network ini belajar mengenali perbedaan antara data input dengan data hasil generate
- Generator Network, network ini mengenerate data baru dari inputan yang ada.
- Discriminator Network, network ini mencoba membedakan antara data asli dan data palsu.



8. Sebutkan dan jelaskan serta di sertai contoh-contoh tahapan dari Agec-GAN

- Encoder, network ini memetakan data input ke dalam bentuk latent vector.
- FaceNet, network ini belajar mengenali perbedaan antara data input dengan data hasil generate

- Generator Network, network ini mengenerate data baru dari inputan yang ada.
- Discriminator Network, network ini mencoba membedakan antara data asli dan data palsu.



9. Berikan contoh perhitungan fungsi training objektif

Rumus yang dipakai untuk menerapkan fungsi training objektif adalah sebagai berikut:

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

Penjelasan:

- $\log D(x, y)$ adalah kerugian pada model Diskriminator.
 - $\log (1 - D(G(x, y), y))$ adalah kerugian pada model Generator.
 - $P(data)$ adalah distribusi dari semua gambar yang memungkinkan.
10. Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation
- Initial latent vector approximation merupakan metode yang digunakan untuk memperkirakan vektor latent untuk mengoptimalkan rekonstruksi gambar wajah.
 - Encoder adalah jaringan saraf yang mendekati vektor latent.
 - Encoder network dilatih pada gambar yang dihasilkan dan pada gambar asli.
 - Setelah dilatih, jaringan encoder akan mulai menghasilkan vektor laten dari distribusi yang dipelajari.
 - Fungsi objektif pelatihan dari pelatihan jaringan network adalah Euclidean distance loss.
11. Berikan contoh perhitungan latent vector optimization
- Latent vector optimization merupakan proses untuk meningkatkan kinerja dari encoder dan generator network secara sekaligus. Rumus yang dipakai untuk

menerapkan latent vector optimization adalah sebagai berikut:

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

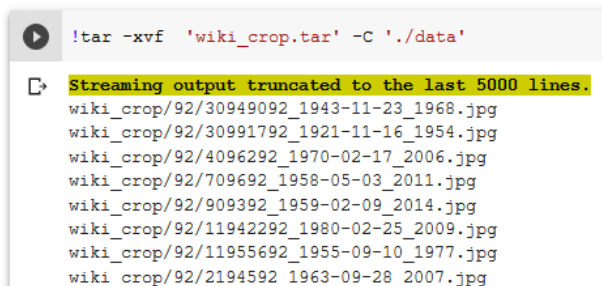
Penjelasan:

- FR adalah face recognition network untuk mengenali identitas orang berdasar inputan gambar wajah
- Persamaan jarak Euclidean antara gambar asli dan gambar rekonstruksi
- Meminimalkan jarak Euclidean ini harus meningkatkan pemeliharaan identitas pada rekonstruksi gambar

9.1.2 Praktek

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

```
1 !tar -xvf 'wiki_crop.tar' -C './data'
```



```
!tar -xvf 'wiki_crop.tar' -C './data'
```

Streaming output truncated to the last 5000 lines.

```
wiki_crop/92/30949092_1943-11-23_1968.jpg
wiki_crop/92/30991792_1921-11-16_1954.jpg
wiki_crop/92/4096292_1970-02-17_2006.jpg
wiki_crop/92/709692_1958-05-03_2011.jpg
wiki_crop/92/909392_1959-02-09_2014.jpg
wiki_crop/92/11942292_1980-02-25_2009.jpg
wiki_crop/92/11955692_1955-09-10_1977.jpg
wiki_crop/92/2194592_1963-09-28_2007.jpg
```

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

Cara load dataset, adalah sebagai berikut:

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

```

9         dob = meta[dataset][0, 0]["dob"][0]
10
11         # List of years when photo was taken
12         photo_taken = meta[dataset][0, 0]["photo_taken"][0]
13
14         # year
15
16         # Calculate age for all dobs
17         age = [calculate_age(photo_taken[i], dob[i]) for i in
18                 range(len(dob))]
19
20         # Create a list of tuples containing a pair of an
21         image path 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

```

Penjelasan kode program perhitungan usia, adalah sebagai berikut:

```

1     def calculate_age(taken, dob):
2         birth = datetime.fromordinal(max(int(dob) - 366, 1))
3
4         if birth.month < 7:
5             return taken - birth.year
6         else:
7             return taken - birth.year - 1
8

```

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,
9           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,
15           padding='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,
21           padding='same')(enc)

```

```

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

```

Encoder, network ini memetakan data input ke dalam bentuk latent vector.

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
4         defined 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)

```



```

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

```

Generator Network, network ini mengenerate data baru dari inputan yang ada.

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
4         values 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='
12         same')(image_input)
13         x = LeakyReLU(alpha=0.2)(x)
14
15         label_input1 = Lambda(expand_label_input)(label_input
16         )
17         x = concatenate([x, label_input1], axis=3)
18
19         x = Conv2D(128, kernel_size=3, strides=2, padding='
20         same')(x)
21         x = BatchNormalization()(x)
22         x = LeakyReLU(alpha=0.2)(x)
23
24         x = Conv2D(256, kernel_size=3, strides=2, padding='
25         same')(x)
26         x = BatchNormalization()(x)
27         x = LeakyReLU(alpha=0.2)(x)
28
29         x = Conv2D(512, kernel_size=3, strides=2, padding='
30         same')(x)

```

```

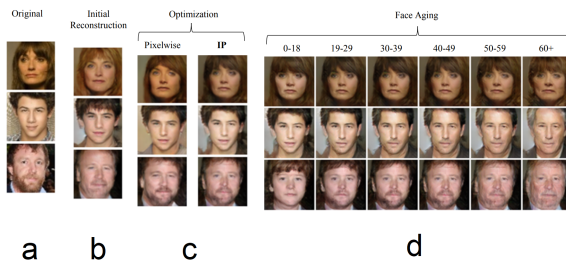
25     x = BatchNormalization()(x)
26     x = LeakyReLU(alpha=0.2)(x)
27
28     x = Flatten()(x)
29     x = Dense(1, activation='sigmoid')(x)
30
31     model = Model(inputs=[image_input, label_input],
32                   outputs=[x])
33     return model

```

Discriminator Network, network ini mencoba membedakan antara data asli dan data palsu.

6. Jelaskan bagaimana kode program Training cGAN bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana Cara kerja program Training cGAN, adalah sebagai berikut:

- Gambar asli yang akan diproses.
- Rekonstruksi gambar yang telah digenerate menggunakan initial latent approximations z_0 .
- Rekonstruksi gambar yang telah digenerate menggunakan "Pixelwise" and "Identity-Preserving" lalu dilakukan optimasi menggunakan latent approximations: $z * \text{pixel}$ and $z * \text{IP}$.
- penentuan umur berdasar gambar generate yang telah direkonstruksi menggunakan identity-preserving $z * \text{IP}$ latent approximations and kondisi dari berbagai label umur y (one per column).



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

- Initial latent vector approximation merupakan metode yang digunakan untuk memperkirakan vektor latent untuk mengoptimalkan rekonstruksi gambar wajah.
- Encoder adalah jaringan saraf yang mendekati vektor latent.
- Encoder network dilatih pada gambar yang dihasilkan dan pada gambar asli.

- Setelah dilatih, jaringan encoder akan mulai menghasilkan vektor laten dari distribusi yang dipelajari.
- Fungsi objektif pelatihan dari pelatihan jaringan network adalah Euclidean distance loss.

9.1.3 Penanganan Error

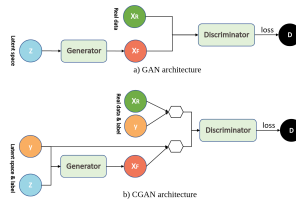
1. Skinsut error
2. Tuliskan kode eror dan jenis errornya
3. Solusi pemecahan masalah error tersebut

9.2 1174017 - Muh. Rifky Prananda

9.2.1 Teori

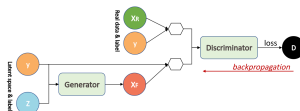
1. Jelaskan dengan ilustrasi gambar sendiri apa perbedaan antara vanilla GAN dan cGAN

Perbedaan antara vanilla GAN dan CGAN terletak pada input proses suatu generator, pada vanilla GAN kita menggunakan data noise yang kemudian diproses menjadi suatu data fake atau palsu sedangkan pada cGAN kita menggunakan latent space atau label pada suatu generator.



Gambar 9.1 Vanilla GAN dan cGAN

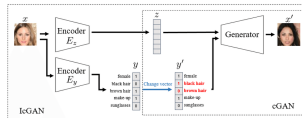
2. Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN
 Pada Arsitektur Age-CGAN terdapat 4 bagian, yaitu : encoder, faceNet, generator dan discriminator. untuk lebih jelasnya bisa dilihat pada ilustrasi gambar dibawah ini.



Gambar 9.2 Arsitektur Age-cGAN

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

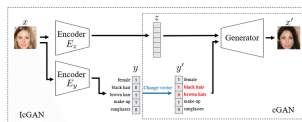
Encoder mempelajari pemetaan terbalik dari gambar wajah input dan kondisi usia dengan vektor laten Z . jaringan encoder menghasilkan vektor laten dari gambar input. jaringan encoder adalah CNN yang mengambil gambar dari dimensi (64,64,3) dan mengubahnya menjadi vektor 100 dimensi. ada empat blok konvolusional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



Gambar 9.3 Arsitektur Encoder Network dari Age-cGAN

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

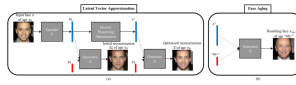
Pada generator dibutuhkan representasi tersembunyi dari gambar wajah dan vektor kondisi sebagai input dan menghasilkan gambar. generator adalah CNN dan dibutuhkan vektor laten 100 dimensi dan vektor kondisi y , dan mencoba menghasilkan gambar realistis dari dimensi (64,64,3). generator memiliki lapisan padat, membingungkan dan konvolusional. lalu 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 9.4 Arsitektur Generator Network dari Age-cGAN

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

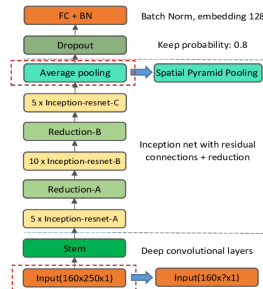
Diskriminator disini berfungsi untuk membedakan antara gambar asli dan gambar palsu. Diskriminator adalah CNN dan memprediksi gambar yang diberikan adalah nyata atau palsu. Disini terdapat blok konvolusional. Setiap blok konvolusional berisi lapisan konvolusional yang diikuti oleh lapisan normalisasi batch, dan fungsi aktifasi, kecuali blok konvolusional pertama, yang tidak memiliki normalisasi batch.



Gambar 9.5 Arsitektur Discriminator Network dari Age-cGAN

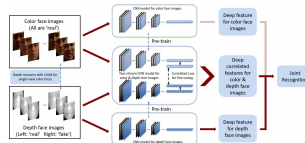
6. Jelaskan dengan ilustrasi gambar sendiri apa itu apa itu pretrained Inception-ResNet-2 Model

Pretrained Inception-ResNet-2 Model adalah suatu model yang diciptakan untuk keperluan klasifikasi image dengan bobot di ImageNet.



Gambar 9.6 Pretrained Inception-ResNet-2 Model

7. Jelaskan dengan ilustrasi gambar arsitektur Face recognition network Age-cGAN
FaceNet merupakan suatu jaringan pengenalan wajah yang mempelajari perbedaan antara gambar input x dan gambar yang direkonstruksi x . FaceNet ini dapat 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 9.7 Arsitektur Face Recognition Network

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

- Input adalah semua data dan perintah yang dimasukkan yang kemudian nantinya akan diproses

- Training, adalah suatu proses yang dimana data-data akan digunakan dalam proses training atau learning
- Testing, adalah suatu proses yang melakukan evaluasi terhadap performa algoritma tersebut.

9. Berikan contoh perhitungan fungsi training objektif

Pada training network cGAN melibatkan fungsi optimalisasi. Melatih cGAN dapat dianggap sebagai permainan minimax, dimana generator dan diskriminator dilatih secara bersamaan. Dalam persamaan dibawah ini, θ merupakan parameter dari jaringan generator, dan ϕ mewakili parameter G dan D, $\log D(x)$ adalah kehilangan dalam model generator dan P_{data} adalah distribusi dari semua gambar yang mungkin.

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

Gambar 9.8 Perhitungan Fungsi Training Objektif

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

Initial latent vector approximation adalah suatu metode untuk memperkirakan vektor laten untuk mengoptimalkan rekonstruksi gambar wajah. untuk memperkirakan vektor latent, kami memiliki jaringan pembuat encode. yaitu dengan melatih jaringan encoder pada gambar yang dihasilkan dan gambar nyata. setelah dilatih, jaringan encoder akan menghasilkan vektor laten dari distribusi bersandar. fungsi tujuan training untuk melatih jaringan encoder yaitu kehilangan jarak euclidean.

11. Berikan contoh perhitungan latent vector optimization

Selama optimasi vektor laten, dengan mengoptimalkan jaringan encoder dan jaringan generator secara bersamaan. persamaan yang kami gunakan untuk optimasi vektor laten adalah sebagai berikut :

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

Gambar 9.9 Perhitungan latent vector optimization

Pada persamaan diatas menunjukkan bahwa jarak euclidean antara gambar asli dan gambar yang direkonstruksi harus minimal. pada tahap ini, kita bisa mencoba meminimalkan jarak untuk memaksimalkan pelestarian identitas.

9.2.2 Praktek

1. Nomor 1

```

1 from google.colab import drive
2 drive.mount('/content/drive')
3
4 import tarfile
5 tf = tarfile.open("/content/drive/My Drive/Chapter 9 AI/wiki_crop
   .tar")
6 tf.extractall(path="/content/drive/My Drive/Chapter 9 AI")

```

Pada kode diatas yaitu menghubungkan google drive dan mengextract dataset. adapun langkah-langkahnya bisa dilihat pada gambar berikut :

- Pertama, login terlebih dahulu ke akun google masing-masing dan masuk ke google colab
- sambungkan google drive dengan google colab
- Melakukan proses extract melalui notebook python di google colab. untuk mengextract bisa menggunakan codingan seperti pada kode diatas

2. Nomor 2

```

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

```

```
25     return images, age_list
```

Maksud dari kode diatas yaitu untuk melakukan load data dan melakukan fungsi perhitungan usia

3. Nomor 3

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

Maksud encoder dalam kode diatas yaitu untuk mempelajari pemetaan terbalik dari gambar wajah yang diinput dan kondisi usia dengan vektor laten Z

4. Nomor 4

```

1 def build_generator():
2     """
3     Create a Generator Model with hyperparameters values defined
4     as follows
5     """
6     latent_dims = 100
7     num_classes = 6
8
9     input_z_noise = Input(shape=(latent_dims,))
10    input_label = Input(shape=(num_classes,))
11
12    x = concatenate([input_z_noise, input_label])
13
14    x = Dense(2048, input_dim=latent_dims + num_classes)(x)
15    x = LeakyReLU(alpha=0.2)(x)
16    x = Dropout(0.2)(x)
17
18    x = Dense(256 * 8 * 8)(x)
19    x = BatchNormalization()(x)
20    x = LeakyReLU(alpha=0.2)(x)
21    x = Dropout(0.2)(x)
22
23    x = Reshape((8, 8, 256))(x)
24
25    x = UpSampling2D(size=(2, 2))(x)
26    x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
27    x = BatchNormalization(momentum=0.8)(x)
28    x = LeakyReLU(alpha=0.2)(x)
29
30    x = UpSampling2D(size=(2, 2))(x)
31    x = Conv2D(filters=64, kernel_size=5, padding='same')(x)
32    x = BatchNormalization(momentum=0.8)(x)
33    x = LeakyReLU(alpha=0.2)(x)
34
35    x = UpSampling2D(size=(2, 2))(x)
36    x = Conv2D(filters=3, kernel_size=5, padding='same')(x)
37    x = Activation('tanh')(x)
38
39    model = Model(inputs=[input_z_noise, input_label], outputs=[x])
40    return model

```

Maksud generator dalam kode diatas yaitu generator network mampu bekerja dengan baik dengan membutuhkan representasi tersembunyi dari gambar wajah dan vektor kondisi sebagai input dan menghasilkan gambar

5. Nomor 5

```

1 def build_discriminator():

```

```

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

```

Maksud diskriminator pada kode diatas yaitu untuk membedakan antara gambar yang asli dan gambar yang palsu

6. Nomor 6

```

1 if __name__ == '__main__':
2     # Define hyperparameters
3     data_dir = "data"
4     wiki_dir = os.path.join(data_dir, "wiki_cropl")
5     epochs = 500
6     batch_size = 2
7     image_shape = (64, 64, 3)
8     z_shape = 100
9     TRAIN_GAN = True
10    TRAIN_ENCODER = False
11    TRAIN_GAN_WITH_FR = False
12    fr_image_shape = (192, 192, 3)
13
14    # Define optimizers

```

```

15     dis_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
16         epsilon=10e-8)
17     gen_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
18         epsilon=10e-8)
19     adversarial_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2
20         =0.999, epsilon=10e-8)
21
22     """
23     Build and compile networks
24     """
25
26     # Build and compile the discriminator network
27     discriminator = build_discriminator()
28     discriminator.compile(loss=['binary_crossentropy'], optimizer
29         =dis_optimizer)
30
31     # Build and compile the generator network
32     generator = build_generator()
33     generator.compile(loss=['binary_crossentropy'], optimizer=
34         gen_optimizer)
35
36     # Build and compile the adversarial model
37     discriminator.trainable = False
38     input_z_noise = Input(shape=(100,))
39     input_label = Input(shape=(6,))
40     recons_images = generator([input_z_noise, input_label])
41     valid = discriminator([recons_images, input_label])
42     adversarial_model = Model(inputs=[input_z_noise, input_label
43         ], outputs=[valid])
44     adversarial_model.compile(loss=['binary_crossentropy'],
45         optimizer=gen_optimizer)
46
47     tensorboard = TensorBoard(log_dir="logs/{}".format(time.time
48         ()))
49     tensorboard.set_model(generator)
50     tensorboard.set_model(discriminator)
51
52     """
53     Load the dataset
54     """
55
56     images, age_list = load_data(wiki_dir=wiki_dir, dataset="wiki
57         ")
58     age_cat = age_to_category(age_list)
59     final_age_cat = np.reshape(np.array(age_cat), [len(age_cat),
60         1])
61     classes = len(set(age_cat))
62     y = to_categorical(final_age_cat, num_classes=len(set(age_cat
63         )))
64
65     loaded_images = load_images(wiki_dir, images, (image_shape
66         [0], image_shape[1]))
67
68     # Implement label smoothing
69     real_labels = np.ones((batch_size, 1), dtype=np.float32) *
70         0.9
71     fake_labels = np.zeros((batch_size, 1), dtype=np.float32) *
72         0.1

```

```

57 """
58 Train the generator and the discriminator network
59 """
60
61 if TRAIN_GAN:
62     for epoch in range(epochs):
63         print("Epoch:{}".format(epoch))
64
65         gen_losses = []
66         dis_losses = []
67
68         number_of_batches = int(len(loaded_images) /
69 batch_size)
70         print("Number of batches:", number_of_batches)
71         for index in range(number_of_batches):
72             print("Batch:{}".format(index + 1))
73
74             images_batch = loaded_images[index * batch_size:(
75 index + 1) * batch_size]
76             images_batch = images_batch / 127.5 - 1.0
77             images_batch = images_batch.astype(np.float32)
78
79             y_batch = y[index * batch_size:(index + 1) *
80 batch_size]
81             z_noise = np.random.normal(0, 1, size=(batch_size
82 , z_shape))
83
84             """
85             Train the discriminator network
86             """
87
88             # Generate fake images
89             initial_recon_images = generator.predict_on_batch
90 ([z_noise, y_batch])
91
92             d_loss_real = discriminator.train_on_batch([
93 images_batch, y_batch], real_labels)
94             d_loss_fake = discriminator.train_on_batch([
95 initial_recon_images, y_batch], fake_labels)
96
97             d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)
98             print("d_loss:{}".format(d_loss))
99
100             """
101             Train the generator network
102             """
103
104             z_noise2 = np.random.normal(0, 1, size=(
105 batch_size, z_shape))
106             random_labels = np.random.randint(0, 6,
107 batch_size).reshape(-1, 1)
108             random_labels = to_categorical(random_labels, 6)
109
110             g_loss = adversarial_model.train_on_batch([
111 z_noise2, random_labels], [1] * batch_size)

```

```

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)

```

Maksud dari kode diatas yaitu sebagai proses training dengan meload file.mat pada dataset, lalu kita melakukan epoch sebanyak 500 kali.

7. Nomor 7

```

1  if TRAIN_ENCODER:
2      # Build and compile encoder
3      encoder = build_encoder()
4      encoder.compile(loss=euclidean_distance_loss, optimizer='
adam')
5
6      # Load the generator network's weights
7      try:
8          generator.load_weights("generator.h5")
9      except Exception as e:
10         print("Error:", e)
11

```

```

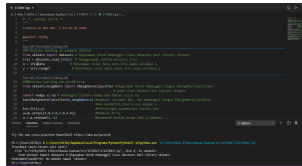
12     z_i = np.random.normal(0, 1, size=(5000, z_shape))
13
14     y = np.random.randint(low=0, high=6, size=(5000,), dtype=
np.int64)
15     num_classes = len(set(y))
16     y = np.reshape(np.array(y), [len(y), 1])
17     y = to_categorical(y, num_classes=num_classes)
18
19     for epoch in range(epochs):
20         print("Epoch:", epoch)
21
22         encoder_losses = []
23
24         number_of_batches = int(z_i.shape[0] / batch_size)
25         print("Number of batches:", number_of_batches)
26         for index in range(number_of_batches):
27             print("Batch:", index + 1)
28
29             z_batch = z_i[index * batch_size:(index + 1) *
batch_size]
30             y_batch = y[index * batch_size:(index + 1) *
batch_size]
31
32             generated_images = generator.predict_on_batch([
z_batch, y_batch])
33
34             # Train the encoder model
35             encoder_loss = encoder.train_on_batch(
generated_images, z_batch)
36             print("Encoder loss:", encoder_loss)
37
38             encoder_losses.append(encoder_loss)
39
40             # Write the encoder loss to Tensorboard
41             write_log(tensorboard, "encoder_loss", np.mean(
encoder_losses), epoch)
42
43             # Save the encoder model
44             encoder.save_weights("encoder.h5")

```

Maksud dari kode diatas yaitu dengan membuat model .h5 lalu meload data dengan menghasilkan result.

9.2.3 Penanganan Error

1. File Not Found Error



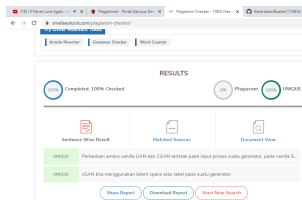
Gambar 9.10 File Not FOund Error

2. Cara Penanganan Error

▪ File Not Found Error

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

9.2.4 Bukti Tidak Plagiat



Gambar 9.11 Bukti Plagiarisme

9.2.5 Link Youtube

BAB 10

CHAPTER 10

BAB 11

CHAPTER 11

BAB 12

CHAPTER 12

BAB 13

CHAPTER 13

BAB 14

CHAPTER 14

DAFTAR PUSTAKA

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

Index

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