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[1].

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

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

BAB 9

CHAPTER 9

9.1 1174008 - Arjun Yuda Firwanda

9.1.1 Teori

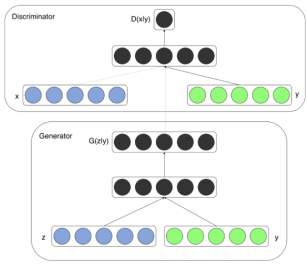
1. Jelaskan dengan ilustrasi gambar sendiri apa perbedaan antara vanilla GAN dan cGAN.

   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.

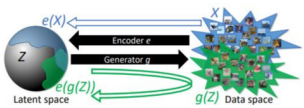
   Age cGAN ialah dengan mengkondisikan model pada informasi tambahan dimungkinkan untuk mengarahkan proses pembuatan data. Pengkondisian semacam itu dapat didasarkan pada label kelas.



Gambar 9.2 Age-cGAN

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

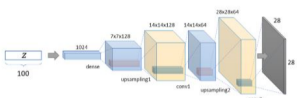
   Arsitektur encoder biasanya digunakan untuk memodelkan struktur manifold dan membalikkan encoder untuk memproses data.



Gambar 9.3 Encoder Age cGANr

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

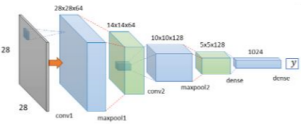
   Arsitektur generator adalah sebuah array yang digunakan secara random, yang disebut seed. dari data seed tersebut, generator akan merubahnya menjadi sebuah gambar yang ukuran 28 x 28 dengan menggunakan Convolutional Neural Network.



Gambar 9.4 Network Age cGAN

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

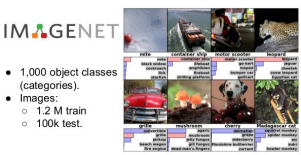
Arsitektur diskriminator adalah CNN yang dapat menerima input gambar yang berukuran 28,28 serta menghasilkan angka biner yang menyatakan apakah data yang diinputkan merupakan dataset asli atau gambar dataset palsu.



Gambar 9.5 Discriminator Age cGAN

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

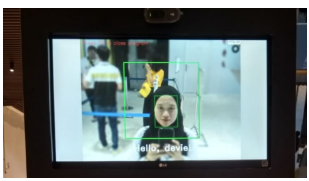
   Pre-Trained Network atau Transfer Learning merupakan suatu metode penyelesaian yang memanfaatkan model yang sudah dilatih terhadap suatu dataset untuk menyelesaikan masalah dengan cara menggunakan sebagai starting point, memodifikasi dan mengupdate parameternya, sehingga sesuai dengan dataset yang baru.



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.

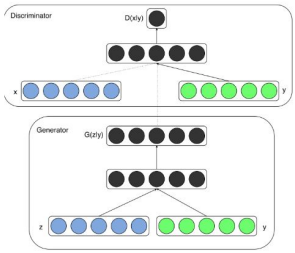


Gambar 9.7 Face recognition network Age-cGAN

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

   Pada dari Age-cGan ni terdapat 2 tahapan dengan generator dan diskriminator. 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.

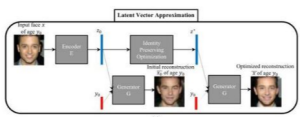
   Objektif Trainning ialah untuk meminimalkan loss function sebagai log likelihood function yang diberikan pada persamaan dimana D melambangkan trainning data.

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

Gambar 9.9 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 lantent optimization menggunakan metode yang relatif sederhana, tergantung pada jumlah kecil parameter yang diperlukan, sehingga pada latent optimization dapat memetakan setiap gambar x dari dataset ke vektor acak dimensi rendah zi dalam ruang laten z.

Gambar 9.11 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)
```

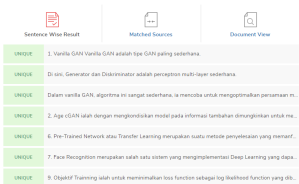
7. Jelaskan bagaimana kode program Initial dan latent vector approximation bekerja dijelaskan dengan bahawa awam dengan ilustrasi sederhana. Initial dan Latent Vector Approximation bekerja melakukan predicsi epoch yang telah di buat sebanyak 500 kali, dan nanti hasilnya ada di folder result.

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

    if TRAIN_ENCODER:
        # Build and compile encoder
        encoder = build_encoder()
```

```
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

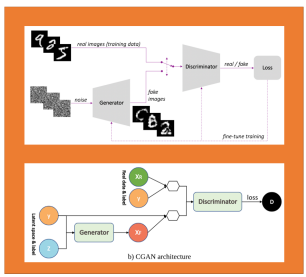


Gambar 9.12 Bukti Tidak Melakukan Plagiat Chapter 9

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:



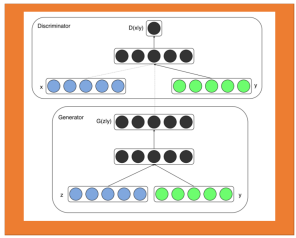
Gambar 9.13 Teori 1

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

   Untuk arsitektur dari Age cGAN mempunyai 4 yaitu :

   - Encoder
   - FaceNet
   - Generator
   - Discriminator

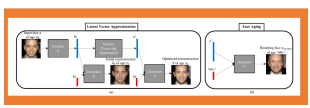
Untuk ilustrasi, lihat gambar berikut:



Gambar 9.14 Teori 2

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

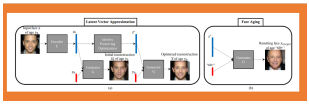
   Encoder mempelajari pemetaan terbalik dari gambar wajah input dan kondisi usia dengan vektor laten Z. Jaringan encoder menghasilkan vektor laten dari gambar input. Jaringan Encoder adalah CNN yang mengambil gambar dari dimensi (64, 64, 3) dan mengubahnya menjadi vektor 100 dimensi. Ada empat blok konvolusional 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

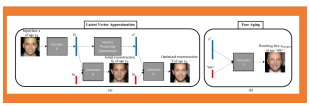
4. Jelaskan dengan ilustrasi gambar sendiri arsitektur generator network dari Agec-GAN.

   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.



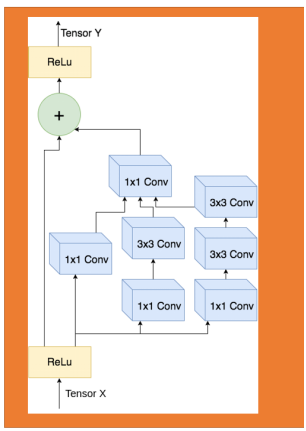
Gambar 9.16 Teori 4

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

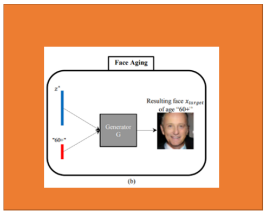
6. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model.
   Model Inception-ResNet v2 adalah model yang ciptakan untuk keperluan klasifikasi image dengan bobot di ImageNet.



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.

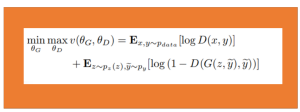


Gambar 9.19 Teori 7

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 (data) adalah distribusi dari semua gambar yang mungkin.



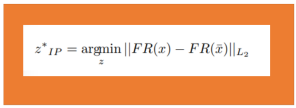
Gambar 9.20 Teori 9

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

    Initial latent vector approximation: Encoder network training adakah seubah 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



Gambar 9.21 Teori 11

9.2.2 Praktek Program

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

```
from google.colab import drive
drive.mount('/content/drive')

import tarfile
tf = tarfile.open("/content/drive/My Drive/Colab Notebooks/GIT CH
    9/data/wiki_crop.tar")
tf.extractall(path="/content/drive/My Drive/Colab Notebooks/GIT
   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.

```
def calculate_age(taken, dob):
    birth = datetime.fromordinal(max(int(dob) - 366, 1))

    if birth.month < 7:
        return taken - birth.year
    else:
        return taken - birth.year - 1
```

```
#%%
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])
```

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

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

```
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
```

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

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

```
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)
    x = Conv2D(filters=64, kernel_size=5, padding='same')(x)
    x = BatchNormalization(momentum=0.8)(x)
    x = LeakyReLU(alpha=0.2)(x)

    x = UpSampling2D(size=(2, 2))(x)
    x = Conv2D(filters=3, kernel_size=5, padding='same')(x)
    x = Activation('tanh')(x)

    model = Model(inputs=[input_z_noise, input_label], outputs=[x
    ])
    return model
```

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

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

```
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
```

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.

```
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)

    """
    Build and compile networks
    """
    # Build and compile the discriminator network
    discriminator = build_discriminator()
    discriminator.compile(loss=['binary_crossentropy'], optimizer
    =dis_optimizer)

    # Build and compile the generator network
    generator = build_generator()
    generator.compile(loss=['binary_crossentropy'], optimizer=
    gen_optimizer)

    # Build and compile the adversarial model
    discriminator.trainable = False
    input_z_noise = Input(shape=(100,))
    input_label = Input(shape=(6,))
    recons_images = generator([input_z_noise, input_label])
    valid = discriminator([recons_images, input_label])
    adversarial_model = Model(inputs=[input_z_noise, input_label
    ], outputs=[valid])
    adversarial_model.compile(loss=['binary_crossentropy'],
    optimizer=gen_optimizer)

```

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

    """
    Load the dataset
    """
    images, age_list = load_data(wiki_dir=wiki_dir, dataset="wiki
    ")
    age_cat = age_to_category(age_list)
    final_age_cat = np.reshape(np.array(age_cat), [len(age_cat),
    1])
    classes = len(set(age_cat))
    y = to_categorical(final_age_cat, num_classes=len(set(age_cat
    )))

    loaded_images = load_images(wiki_dir, images, (image_shape
    [0], image_shape[1]))

    # Implement label smoothing
    real_labels = np.ones((batch_size, 1), dtype=np.float32) *
    0.9
    fake_labels = np.zeros((batch_size, 1), dtype=np.float32) *
    0.1

    """
    Train the generator and the discriminator network
    """
    if TRAIN_GAN:
        for epoch in range(epochs):
            print("Epoch:{}".format(epoch))

            gen_losses = []
            dis_losses = []

            number_of_batches = int(len(loaded_images) /
    batch_size)
            print("Number of batches:", number_of_batches)
            for index in range(number_of_batches):
                print("Batch:{}".format(index + 1))

                images_batch = loaded_images[index * batch_size:(
    index + 1) * batch_size]
                images_batch = images_batch / 127.5 - 1.0
                images_batch = images_batch.astype(np.float32)

                y_batch = y[index * batch_size:(index + 1) *
    batch_size]
                z_noise = np.random.normal(0, 1, size=(batch_size
    , z_shape))

                """
                Train the discriminator network
                """

```

```
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
```

```
        try:
            generator.save_weights("generator.h5")
            discriminator.save_weights("discriminator.h5")
        except Exception as e:
            print("Error:", e)
```

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

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

```
    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 = []

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

                z_batch = z_i[index * batch_size:(index + 1) * batch_size]
                y_batch = y[index * batch_size:(index + 1) * batch_size]

                generated_images = generator.predict_on_batch([z_batch, y_batch])

                # Train the encoder model
                encoder_loss = encoder.train_on_batch(generated_images, z_batch)
                print("Encoder loss:", encoder_loss)

                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")
45
46      """
47      Optimize the encoder and the generator network
48      """
49      if TRAIN_GAN_WITH_FR:
50
51          # Load the encoder network
52          encoder = build_encoder()
53          encoder.load_weights("encoder.h5")
54
55          # Load the generator network
56          generator.load_weights("generator.h5")
57
58          image_resizer = build_image_resizer()
59          image_resizer.compile(loss=['binary_crossentropy'],
    optimizer='adam')
60
61          # Face recognition model
62          fr_model = build_fr_model(input_shape=fr_image_shape)
63          fr_model.compile(loss=['binary_crossentropy'], optimizer=
    "adam")
64
65          # Make the face recognition network as non-trainable
66          fr_model.trainable = False
67
68          # Input layers
69          input_image = Input(shape=(64, 64, 3))
70          input_label = Input(shape=(6,))
71
72          # Use the encoder and the generator network
73          latent0 = encoder(input_image)
74          gen_images = generator([latent0, input_label])
75
76          # Resize images to the desired shape
77          resized_images = Lambda(lambda x: K.resize_images(
    gen_images, height_factor=3, width_factor=3,
78
    data_format='channels_last'))(gen_images)
79          embeddings = fr_model(resized_images)
80
81          # Create a Keras model and specify the inputs and outputs
     for the network
82          fr_adversarial_model = Model(inputs=[input_image,
    input_label], outputs=[embeddings])
83
84          # Compile the model
85          fr_adversarial_model.compile(loss=euclidean_distance_loss
    , optimizer=adversarial_optimizer)
86
```

```
87          for epoch in range(epochs):
88              print("Epoch:", epoch)
89
90              reconstruction_losses = []
91
92              number_of_batches = int(len(loaded_images) /
    batch_size)
93              print("Number of batches:", number_of_batches)
94              for index in range(number_of_batches):
95                  print("Batch:", index + 1)
96
97                  images_batch = loaded_images[index * batch_size:(
    index + 1) * batch_size]
98                  images_batch = images_batch / 127.5 - 1.0
99                  images_batch = images_batch.astype(np.float32)
100
101                 y_batch = y[index * batch_size:(index + 1) *
    batch_size]
102
103                 images_batch_resized = image_resizer.
    predict_on_batch(images_batch)
104
105                 real_embeddings = fr_model.predict_on_batch(
    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 = loaded_images[0:batch_size]
121                 images_batch = images_batch / 127.5 - 1.0
122                 images_batch = images_batch.astype(np.float32)
123
124                 y_batch = y[0:batch_size]
125                 z_noise = np.random.normal(0, 1, size=(batch_size
    , 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
```

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

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

9.2.3 Penanganan Error

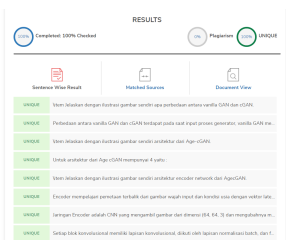
1. ValueError

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



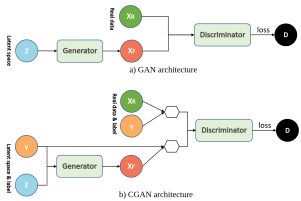
Gambar 9.23 Bukti Tidak Melakukan Plagiat Chapter 9, CIE COPAS

9.3 1174017 - Muh. Rifky Prananda

9.3.1 Teori

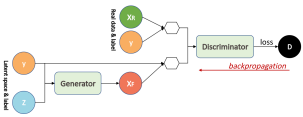
1. Jelaskan dengan ilustrasi gambar sendiri apa perbedaan antara vanilla GAN dan cGAN
   Perbedaan antara vanilla GAN dan CGAN terletak pada input proses suatu generator, pada vanilla GAN kita menggunakan data noise yang kemudian diproses

menjadi suatu data fake atau palsu sedangkan pada cGAN kita menggunakan latent space atau label pada suatu generator.



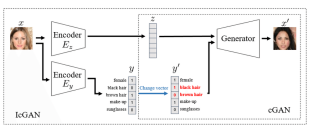
Gambar 9.24 Vanilla GAN dan cGAN

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 9.25 Arsitektur Age-cGAN

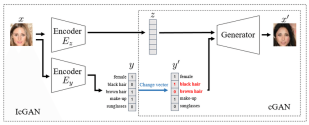
3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari Age-cGAN
   Encoder mempelajari pemetaan terbalik dari gambar wajah input dan kondisi usia dengan vektor laten Z. jaringan encoder menghasilkan vektor laten dari gambar input.jaringan encoder adalah CNN yang mengambil gambar 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 9.26 Arsitektur Encoder Network dari Age-cGAN

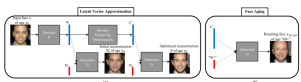
4. Jelaskan dengan ilustrasi gambar sendiri arsitektur generator network dari Age-cGAN
   Pada generator dibutuhkan representasi tersembunyi dari gambar wajah dan vektor kondisi sebagai input dan menghasilkan gambar. generator adalah CNN

dan dibutuhkan vektor laten 100 dimensi dan vektor kondisi y, dan mencoba menghasilkan gambar realistis dari dimensi (64,64,3). generator memiliki lapisan padat, membingungka dan konvlolusional. 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.



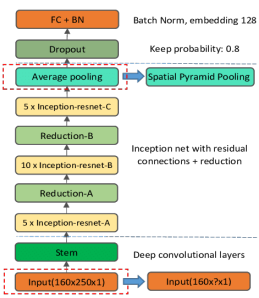
Gambar 9.27 Arsitektur Generator Network dari Age-cGAN

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.



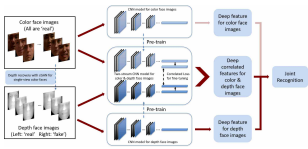
Gambar 9.28 Arsitektur Discriminator Network dari Age-cGAN

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 9.29 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 spenuhnya dapat digunakan. Embedding yang diekstraksi untuk gambar asli dan gambar yang direkonstruksi dapat dihitung dengan menghitung jarak Euclidean dari embeddings.



Gambar 9.30 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, a merupakan parameter dari jaringan generator, dan n mewakili parameter G dan D, logD(r) adalah kehilangan dalam model generator dan Pdata adalah distribusi dari semua gambar yang mungkin.

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

Gambar 9.31 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 menghaslkan 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 :

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

Gambar 9.32 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.

9.3.2 Praktek

1. Nomor 1

```
from google.colab import drive
drive.mount('/content/drive')

import tarfile
tf = tarfile.open("/content/drive/My Drive/Chapter 9 AI/wiki_crop.tar")
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 mengestract bisa menggunakan codingan seperti pada kode diatas

2. Nomor 2

```
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
```

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

3. Nomor 3

```
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
```

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

```
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)
```

```
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
```

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

```
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)

    """
    Build and compile networks
    """
    # Build and compile the discriminator network
    discriminator = build_discriminator()
    discriminator.compile(loss=['binary_crossentropy'], optimizer
    =dis_optimizer)

    # Build and compile the generator network
    generator = build_generator()
    generator.compile(loss=['binary_crossentropy'], optimizer=
    gen_optimizer)

    # Build and compile the adversarial model
    discriminator.trainable = False
    input_z_noise = Input(shape=(100,))
    input_label = Input(shape=(6,))
    recons_images = generator([input_z_noise, input_label])
    valid = discriminator([recons_images, input_label])
    adversarial_model = Model(inputs=[input_z_noise, input_label
    ], outputs=[valid])
    adversarial_model.compile(loss=['binary_crossentropy'],
    optimizer=gen_optimizer)

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

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

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

7. Nomor 7

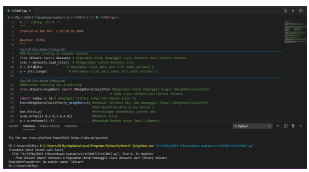
```
if TRAIN_ENCODER:
        # Build and compile encoder
        encoder = build_encoder()
        encoder.compile(loss=euclidean_distance_loss, optimizer='
    adam')

        # 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 = []

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

                z_batch = z_i[index * batch_size:(index + 1) *
    batch_size]
                y_batch = y[index * batch_size:(index + 1) *
    batch_size]

                generated_images = generator.predict_on_batch([
    z_batch, y_batch])

                # Train the encoder model
                encoder_loss = encoder.train_on_batch(
    generated_images, z_batch)
                print("Encoder loss:", encoder_loss)

                encoder_losses.append(encoder_loss)

            # 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.

9.3.3 Penanganan Error

1. File Not Found Error

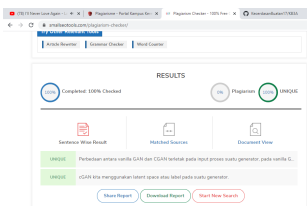


Gambar 9.33 File Not FOund Error

2. Cara Penanganan Error

   - File Not Found Error
     Error tersebut karena disebabkan gagal load dataset karena salah penamaan direktori.

9.3.4 Bukti Tidak Plagiat



Gambar 9.34 Bukti Plagiarisme

9.3.5 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