

CERDAS MENGUASAI GIT

CERDAS MENGUASAI GIT

Dalam 24 Jam

Rolly M. Awangga
Informatics Research Center



Kreatif Industri Nusantara

Penulis:

Rolly Maulana Awangga

ISBN : 978-602-53897-0-2

Editor:

M. Yusril Helmi Setyawan

Penyunting:

Syafrial Fachrie Pane

Khaera Tunnisia

Diana Asri Wijayanti

Desain sampul dan Tata letak:

Deza Martha Akbar

Penerbit:

Kreatif Industri Nusantara

Redaksi:

Jl. Ligar Nyawang No. 2

Bandung 40191

Tel. 022 2045-8529

Email : awangga@kreatif.co.id

Distributor:

Informatics Research Center

Jl. Sariasisih No. 54

Bandung 40151

Email : irc@poltekpos.ac.id

Cetakan Pertama, 2019

Hak cipta dilindungi undang-undang

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

*'Jika Kamu tidak dapat
menahan lelahnya
belajar, Maka kamu harus
sanggup menahan
perihnya Kebodohan.'*

Imam Syafi'i

CONTRIBUTORS

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

CONTENTS IN BRIEF

1 Chapter 1	1
2 Chapter 2	3
3 Chapter 3	5
4 Chapter 4	7
5 Chapter 5	9
6 Chapter 6	11
7 Chapter 7	13
8 Chapter 8	29
9 Chapter 9	45

DAFTAR ISI

Foreword	xiii
Kata Pengantar	xv
Acknowledgments	xvii
Acronyms	xix
Glossary	xxi
List of Symbols	xxiii
Introduction <i>Rolly Maulana Awangga, S.T., M.T.</i>	xxv
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
7.1	Nurul Izza Hamka - 1174062	13
7.1.1	Teori	13
7.1.2	Praktek Program	17
7.1.3	Bukti Tidak Plagiat	27
8	Chapter 8	29
8.1	Nurul Izza Hamka - 1174062	29
8.1.1	Teori	29
8.1.2	Praktek Program	33
8.1.3	Bukti Tidak Plagiat	43
9	Chapter 9	45
9.1	1174086 - Tia Nur Candida	45
9.1.1	Teori	45
9.1.2	Praktek	49
9.1.3	Bukti Tidak Plagiat	54
9.2	Muhammad Reza Syachrani - 1174084	55
9.2.1	Teori	55
9.2.2	Praktek	60
9.2.3	Bukti Tidak Plagiat	65
9.2.4	Link Video Youtube	65
9.3	1174073 - Ainul Filiani	65
9.3.1	Teori	65
9.3.2	Praktek	69
9.3.3	Penanganan Error	74
9.3.4	Bukti Tidak Plagiat	74
9.4	1174083 - Bakti Qilan Mufid	74
9.4.1	Teori	74
9.4.2	Praktek	81
9.4.3	Penanganan Error	95
9.4.4	Bukti Tidak Plagiat	97
9.4.5	Link Youtube	98
9.5	1174066 - D.Irga B. Naufal Fakhri	98

9.5.1	Teori	98
9.5.2	Praktek	102
9.5.3	Penanganan Error	112
9.5.4	Bukti Tidak Plagiat	112
9.5.5	Link Youtube	112
9.6	1174087 - Ilham Muhammad Ariq	113
9.6.1	Teori	113
9.6.2	Praktek Program	116
9.6.3	Penanganan Error	126
9.6.4	Bukti Tidak Plagiat	126
9.7	1174067 - Kaka Kamaludin	127
9.7.1	Teori	127
9.7.2	Praktek	131
9.7.3	Link Youtube	136
9.8	1174069 - Fanny Shafira Damayanti	136
9.8.1	Teori	136
9.8.2	Praktek	139
9.8.3	Penanganan Error	144
9.8.4	Bukti Tidak Plagiat	144
9.9	1174070 - Arrizal Furqona Gifary	144
9.9.1	Teori	144
9.9.2	Praktek	148
9.9.3	Penanganan Error	153
9.9.4	Bukti Tidak Plagiat	153
9.10	1174054 - Aulyardha Anindita	153
9.10.1	Teori	153
9.10.2	Praktek	157
9.10.3	Penanganan Error	165
9.10.4	Bukti Tidak Plagiat	165
9.10.5	Link Youtube	165
9.11	Difa Al Fansha	165
9.11.1	Teori	165
9.11.2	Praktek	170
9.11.3	Penanganan Error	175
9.11.4	Bukti Tidak Plagiat	175
9.11.5	Link Youtube:	176
9.12	1174077 - Alvan Alvanzah	176
9.12.1	Teori	176

9.12.2	Praktek	180
9.12.3	Penanganan Error	185
9.12.4	Bukti Tidak Plagiat	185
9.13	1174071 - Muhammad Abdul Gani Wijaya	185
9.13.1	Teori	185
9.13.2	Praktek	189
9.13.3	Link Youtube	194
9.14	Nurul Izza Hamka - 1174062	194
9.14.1	Teori	194
9.14.2	Praktek Program	197
9.14.3	Bukti Tidak Plagiat	202

FOREWORD

Sepatah kata dari Kaprodi, Kabag Kemahasiswaan dan Mahasiswa

KATA PENGANTAR

Buku ini diciptakan bagi yang awam dengan git sekalipun.

R. M. AWANGGA

Bandung, Jawa Barat

Februari, 2019

ACKNOWLEDGMENTS

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

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

R. M. A.

ACRONYMS

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

GLOSSARY

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

SYMBOLS

A Amplitude

$\&$ Propositional logic symbol

a Filter Coefficient

B Number of Beats

INTRODUCTION

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

Informatics Research Center
Bandung, Jawa Barat, Indonesia

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

Rumusnya cuman sebagai contoh aja biar keren[?].

$$ABC\mathcal{DEF}\alpha\beta\Gamma\Delta \sum_{def}^{abc} \quad (I.1)$$

BAB 1

CHAPTER 1

BAB 2

CHAPTER 2

BAB 3

CHAPTER 3

BAB 4

CHAPTER 4

BAB 5

CHAPTER 5

BAB 6

CHAPTER 6

BAB 7

CHAPTER 7

7.1 Nurul Izza Hamka - 1174062

7.1.1 Teori

1. Jelaskan kenapa file teks harus dilakukan tokenizer, dilengkapi dengan ilustrasi atau gambar

File harus di lakukan konversi terlebih dahulu menjadi angka sebelum dilakukan inputan menjadi neural network. Keras menyediakan sebuah kelas tokonizer, yang mana tokonizer ini berfungsi untuk melakukan konversi teks menjadi urutan integer indeks atau word count (tf-idf) dan vector binary.

Contoh:

Hasilnya akan tampak seperti ini:

2. Jelaskan konsep dasar K Fold Cross Validation pada dataset komentar Youtube pada kode listing 7.1 dilengkapi dengan ilustrasi atau gambar.

StratifiedKFold berisi presentasi sampel pada setiap kelas.

```

8 from keras.preprocessing.text import Tokenizer
9 tokenizer = Tokenizer()
10 texts = ["Budi makan nasi", "Rudi makan nasi, nasi goreng."]
11 tokenizer.fit_on_texts(texts)
12
13 seq = tokenizer.texts_to_sequences(texts)
14
15 seq1 = tokenizer.texts_to_sequences(["nasi panas sekali"])
16 print("Index: "+str((tokenizer.word_index)))
17 print("Seq. corpus: "+str(seq))
18 print("Seq. untuk 'nasi panas sekali': "+str(seq1))

```

Gambar 7.1

```

In [3]: Index: {`budi`: 4, `nasi`: 3, `makna`: 1, `goreng`: 5}
...: Seq. corpus: [[4, 3, 1], [4, 3, 1, 5]]
...: Seq. untuk "nasi panas sekali": [[1]]

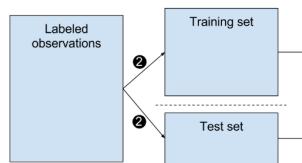
```

Gambar 7.2 Hasil

```

1 kfolds = StratifiedKFold(n_splits=5)
2 splits = kfolds.split(d, d['CLASS'])
3
4 Listing 7.1: K Fold Cross Validation

```

Gambar 7.3 K Fold Cross Validation**Gambar 7.4** Ilustrasi Gambar

3. Jelaskan apa maksudnya kode program for train, test in splits.dilengkapi dengan ilustrasi atau gambar.

Train / Test adalah metode untuk mengukur akurasi model Anda.

Ini disebut Train / Test karena Anda membagi set data menjadi dua set: satu set pelatihan dan satu set pengujian.

Split susunan atau matriks menjadi rangkaian acak train dan test.

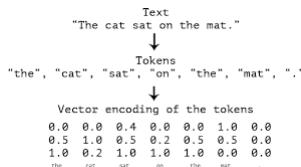
4. Jelaskan apa maksudnya kode program train content = d['CONTENT'].iloc[train_idx] dan test content = d['CONTENT'].iloc[test_idx]. dilengkapi dengan ilustrasi atau gambar.

Maksud dari kode program d['CONTENT'].iloc[train_idx] dan test content = d['CONTENT'].iloc[test_idx] adalah kita mengambil data dari kolom CONTENT yang mana merupakan bagian dari train_idx dan test_idx.

Ilustrasinya adalah Ketika data telah diubah menjadi train dan test maka nantinya kita bebas menampilkannya di kolom mana saja yang kita inginkan.

5. Jelaskan apa maksud dari fungsi tokonizer = Tokonizer (num words = 2000) dan tokonizer.fit on texts (train content), dilengkapi dengan ilustrasi atau gambar.

num words=2000 artinya jumlah kata maksimum yang harus disimpan, berdasarkan frekuensi kata, jadi jumlah kata maksimum adalah 2000.

**Gambar 7.5** Ilustrasi Gambar Nomor 6

```

>>> x = np.array([-1.2, 1.2])
>>> np.absolute(x)
array([1.2, 1.2])

```

Gambar 7.6 Ilustrasi Gambar Nomor 7

```

1 model = Sequential()
2 model.add(Dense(512, input_shape=(2000,)))
3 model.add(Activation('relu'))
4 model.add(Dropout(0.5))
5 model.add(Dense(12))
6 model.add(Activation('softmax'))

```

Listing 7.2. Membuat model Neural Network

Gambar 7.7 Listin 7.2

tokonizer.fit on text (train content) artinya kita akan melakukan fit tokonizer yang hanya untuk data train saja pada kolom CONTENT.

6. Jelaskan apa maksud dari fungsi `d_train.inputs = tokonizer.texts_to_matrix()` dan `d_test.inputs = tokonizer.to_matrix(test content, mode='tfidf')`, dilengkapi dengan ilustrasi kode dan atau gambar.

Maksud dari fungsi yang ada disoal adalah variable `d_train` akan melakukan tokonizer dari teks menjadi bentuk matrix dari data `train_content` yang mode matriknya ada tfidf.

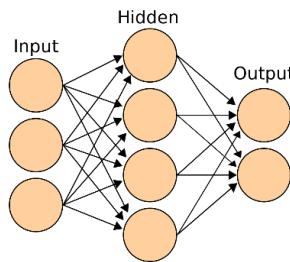
7. Jelaskan apa maksud dari fungsi `d_train.inputs = d_train.inputs / np.amax(np.absolute(d_train))` dan `d_test.inputs = d_test.inputs / np.amax(np.absolute(d_test))`, dilengkapi dengan ilustrasi atau gambar.

Fungsi pada soal diatas adalah membagi matrix tfidf tadi dengan amax, yaitu dengan mengembalikan nilai maks array. Yang hasilnya akan dimasukkan kedalam `d_trains_input` dan `d_test_inputs` dengan nilai absolute.

8. Jelaskan apa maksud fungsi dari `d_train_outputs=np_utils.to_categorical(d['CLASS'].iloc[:])` dan `d_test_outputs=np_utils.to_categorical(d['CLASS'].iloc[test_idx])` dalam kode program, dilengkapi dengan ilustrasi atau gambar.

Pada variable `d_train_output` dan `d_test_output` akan dilakukan one hot encoding dimana `np_utils` akan mengubah bentuk integer ke matrix kelas binear unuk kolom CLAS yang nantinya hanya ada 1 dan 0.

9. Jelaskan apa maksud dari fungsi di listing 7.2. Gambarkan ilustrasi Neural Network nya dari model kode tersebut.



Gambar 7.8 Ilustrasi Gambar Nomor 8

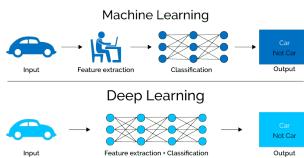
```

1 model.compile(loss='categorical_crossentropy', optimizer='adamax'
2 metrics=['accuracy'])
Listing 7.3: Compile model
  
```

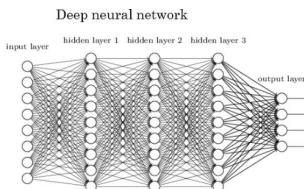
Gambar 7.9 Listing 7.3

- `model = Sequential ()` = pemodelan sequential
- `model.add (Dense (512, input shape = (2000,)))` = pada layer pertama dense dari 512 neouron untuk inputan sebelumnya yang sudah dijadikan matrix dengan jumlah 2000
- `model.add (Activation ('relu'))` = aktivasinya menggunakan function relu, artinya jika ada inputan dengan nilai maks maka inputan tersebut yang dipilih.
- `model.add (Dropout (0.5))` = Dropout akan melakukan pembobotan, dimana pemobobotanya 50% saja.
- `model.add (Dense (2))` = dense 2 mengkategorikan sebanyak 2 neuron.
- `model. Add (Activation ('softmax'))` = aktivasinya menggunakan function softmax.

10. Jelaskan apa maksud dari fungsi di listing 7.3 dengan parameter tersebut.
Melakukan compile dari model sequential dengan loss yang merupakan fungsi optimasi skor dengan categorical_crossentropy, dan menggunakan algoritma adama sebagai optimizer.
11. Jelaskan apa itu Deep Learning
Deep learning adalah metode implemtasi dari machine learning dengan tujuan meniru cara kerja pada otak manusia menggunakan neural network. Singkatnya deep learning ini model yang mempelajari metode komputasi dengan sendirinya.
12. Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning
Deep Neural Network adalah salah satu algoritma yang dapat digunakan dalam pengambilan keputusan, yang mana ini berbasis jaringan saraf. Sedangkan deep learning ini menggunakan deep neural network dalam menyelesaikan masalah pada machine learning.



Gambar 7.10 Ilustrasi Gambar Nomor 11



Gambar 7.11 Ilustrasi Gambar Nomor 12

```
In [1]: import csv
...::: from PIL import Image as pil_image
...::: import keras.preprocessing.image
Using TensorFlow backend.
```

Gambar 7.12 Hasil Nomor 1

13. Jelaskan dengan ilustrasi gambar buatan sendiri(langkah per langkah) bagaimana perhitungan algoritma konvolusi dengan ukuran stride (NPM mod3+1) x (NPM mod3+1) yang terdapat max pooling.(nilai 30)

7.1.2 Praktek Program

- Jelaskan kode program pada blok # In[1]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[1]: import lib
2 import csv
3 from PIL import Image as pil_image
4 import keras.preprocessing.image
```

Kode diatas adalah library untuk melakukan import file CSV. Kemudian untuk module pil image adalah untuk library keras.

Haslinya:

```
In [3]: import random
...: random.shuffle(imgs)
...: split_idx = int(0.8*len(imgs))
...: train = imgs[:split_idx]
...: test = imgs[split_idx:]
```

Gambar 7.13 Hasil Nomor 3

2. Jelaskan kode program pada blok # In[2]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In [2]: load all images (as numpy arrays) and save their classes
2 imgs = []
3 classes = []
4 with open('HASYv2/hasy-data-labels.csv') as csvfile:
5     csvreader = csv.reader(csvfile)
6     i = 0
7     for row in csvreader:
8         if i > 0:
9             img = keras.preprocessing.image.img_to_array(
10                pil_image.open("HASYv2/" + row[0]))
11                img /= 255.0
12                imgs.append((row[0], row[2], img))
13                classes.append(row[2])
14            i += 1
```

Kode diatas untuk menampilkan hasil load dataset dari HASYv2 sebagai file CSV, dan Menginisiasi variabel imgs dan classes dengan variabel array kosong, dan append untuk melampirkan atau menggabungkan data file.

3. Jelaskan kode program pada blok # In[3]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In [3]: shuffle the data, split into 80% train , 20% test
2 import random
3 random.shuffle(imgs)
4 split_idx = int(0.8*len(imgs))
5 train = imgs[:split_idx]
6 test = imgs[split_idx:]
```

import random untuk melakukan random pada vungsi imgs
random.shuffle(imgs) untuk Menginisiasi variabel split.idx dengan nilai integer 80 persen dikali dari pengembalian jumlah dari variabel imgs

Haslinya:

```
In [4]: import numpy as np
...: train_output = np.asarray(list(map(lambda row: row[2], train)))
...: train_input = np.asarray(list(map(lambda row: row[1], train)))
...: test_output = np.asarray(list(map(lambda row: row[1], test)))
...: test_input = np.asarray(list(map(lambda row: row[1], test)))
```

Gambar 7.14 Hasil Nomor 4

```
In [5]: from sklearn.preprocessing import LabelEncoder
...: from sklearn.preprocessing import OneHotEncoder
```

Gambar 7.15 Hasil Nomor 5

4. . Jelaskan kode program pada blok # In[4]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In [4]:
2 import numpy as np
3 train_input = np.asarray(list(map(lambda row: row[2], train)))
4 test_input = np.asarray(list(map(lambda row: row[2], test)))
5 train_output = np.asarray(list(map(lambda row: row[1], train)))
6 test_output = np.asarray(list(map(lambda row: row[1], test)))
```

Yang pertama adalah Melakukan import library numpy dengan inisial np, selanjutnya Menginisiasi variabel train input dengan np method asarray yang mana membuat array dengan isi row 2 dari data train

Haslinya:

5. Jelaskan kode program pada blok # In[5]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri

```
1 # In [5]: import encoder and one hot
2 from sklearn.preprocessing import LabelEncoder
3 from sklearn.preprocessing import OneHotEncoder
```

Pertama Melakukan import library LabelEncode dari sklearn dan kemudian Melakukan import library OneHotEncoder dari sklearn

Haslinya:

6. Jelaskan kode program pada blok # In[6]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[6]: convert class names into one-hot encoding
2 label_encoder = LabelEncoder()
3 integer_encoded = label_encoder.fit_transform(classes)

```

Kode diatas Menginisiasi variabel `label_encoder` dengan *isLabelEncoder* dan *Menginisiasi*

- Jelaskan kode program pada blok # In[7]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[7]: then convert integers into one-hot encoding
2 onehot_encoder = OneHotEncoder(sparse=False)
3 integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
4 onehot_encoder.fit(integer_encoded)

```

Pada kode baris pertama diatas adalah Menginisiasi variabel `onehot_encoder` dengan *isOneHotEncoder*

- Jelaskan kode program pada blok # In[8]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[8]: convert train and test output to one-hot
2 train_output_int = label_encoder.transform(train_output)
3 train_output = onehot_encoder.transform(train_output_int.reshape(
    len(train_output_int), 1))
4 test_output_int = label_encoder.transform(test_output)
5 test_output = onehot_encoder.transform(test_output_int.reshape(
    len(test_output_int), 1))
6 num_classes = len(label_encoder.classes_)
7 print("Number of classes: %d" % num_classes)

```

Pada kode diatas adalah pertama Menconvert data train output menggunakan variabel `label_encoder` kedalam variabel `train_output_int`

Kemudian Menconvert variabel `train_output_int` kedalam fungsi `onehot_encoder` dan selanjutnya pada baris ke3 Menconvert data `test_output` menggunakan variabel `label_encoder`

- Jelaskan kode program pada blok # In[9]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[9]: import sequential
2 from keras.models import Sequential
3 from keras.layers import Dense, Dropout, Flatten
4 from keras.layers import Conv2D, MaxPooling2D

```

Kode diatas adalah:

Baris 1 Melakukan import library Sequential dari Keras

```
In [9]: from keras.models import Sequential
...: from keras.layers import Dense, Dropout, Flatten
...: from keras.layers import Conv2D, MaxPooling2D
```

Gambar 7.16 Hasil Nomor 9

Baris 2 Melakukan import library Dense, Dropout, Flatten dari Keras
 Baris 3 Melakukan import library Conv2D, MaxPooling2D dari Keras.

Haslinya:

10. Jelaskan kode program pada blok # In[10]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[10]: desain jaringan
2 model = Sequential()
3 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
4                  input_shape=np.shape(train_input[0])))
5 model.add(MaxPooling2D(pool_size=(2, 2)))
6 model.add(Conv2D(32, (3, 3), activation='relu'))
7 model.add(MaxPooling2D(pool_size=(2, 2)))
8 model.add(Flatten())
9 model.add(Dense(1024, activation='tanh'))
10 model.add(Dropout(0.5))
11 model.add(Dense(num_classes, activation='softmax'))
12 model.compile(loss='categorical_crossentropy', optimizer='adam',
13                 metrics=['accuracy'])
14 print(model.summary())
```

Baris 1 Menginisiasi variabel model dengan isian library Sequential

Baris 2 variabel model di tambahkan library Conv2D tigapuluhan dua bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan relu dang menggunakan data train;np

Baris 3 variabel model ditambahkan dengan lib MaxPooling2D dengan ketentuan ukuran

Baris 4 variabel model ditambahkan dengan library Conv2D 32bit dengan kernel 3x3

Baris 5 variabel model ditambahkan dengan lib MaxPooling2D dengan ketentuan ukuran

Baris 6 variabel model ditambahkan library Flatten

Baris 7 variabel model ditambahkan library Dense dengan fungsi tanh

Baris 8 variabel model ditambahkan library dropout untuk memangkas data tree sebesar 50

Baris 9 variabel model ditambahkan library Dense dengan data darin num_classes dan fungsi

Baris 10 mengkompile data model untuk mendapatkan loss akurasid dan optimasi

Baris 11 mencetak variabel model kemudian memunculkan kesimpulan berupa data total p

Haslinya:

11. Jelaskan kode program pada blok # In[11]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
In [10]: model = Sequential()
...: model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
...: input_shape=(28, 28, 1)))
...: model.add(MaxPooling2D(pool_size=(2, 2)))
...: model.add(Conv2D(32, (3, 3), activation='relu'))
...: model.add(Flatten())
...: model.add(Dense(32, activation='tanh'))
...: model.add(Dropout(0.5))
...: model.add(Dense(num_classes, activation='softmax'))
...: model.compile(loss='categorical_crossentropy', optimizer='adam',
...: metrics=['accuracy'])
...: print(model.summary())
```

Gambar 7.17 Hasil Nomor 10

```
In [11]: import keras.callbacks
...: # Menginisiasi variabel tensorboard dengan isi lib keras
...: tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-
style')
```

Gambar 7.18 Hasil Nomor 11

```
1 # In[11]: import sequential
2 import keras.callbacks
3 tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-
style')
```

Haslinya:

12. Jelaskan kode program pada blok # In[12]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[12]: 5menit kali 10 epoch = 50 menit
2 # fungsi model titambahkan metod fit untuk mengetahui perhitungan
# dari train_input train_output
3 model.fit(train_input, train_output,
4 # dengan batch size 32 bit
5 #         batch_size=32,
6 #         epochs=10,
7 #         verbose=2,
8 #         validation_split=0.2,
9 #         callbacks=[tensorboard])
10
11 score = model.evaluate(test_input, test_output, verbose=2)
12 print('Test loss:', score[0])
13 print('Test accuracy:', score[1])
```

13. Jelaskan kode program pada blok # In[13]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[13]: try various model configurations and parameters to find
# the best
2 # Melakukan import library time
3 import time
```

```

4 #Menginisiasi variabel result dengan array kosong
5 results = []
6 # melakukan looping dengan ketentuan konvolusi 2 dimensi 1 2
7 for conv2d_count in [1, 2]:
8     # menentukan ukuran besaran fixcel dari data atau konvert 1
9     # fixcel mnjadi data yang berada pada codigan dibawah.
10    for dense_size in [128, 256, 512, 1024, 2048]:
11        # membuat looping untuk memangkas masing-masing data
12        dengan ketentuan 0 persen 25 persen 50 persen dan 75 persen .
13        for dropout in [0.0, 0.25, 0.50, 0.75]:
14            # Menginisiasi variabel model Sequential()
15            model = Sequential()
16            #membuat looping untuk variabel i dengan jarak dari
17            hasil konvolusi.
18            for i in range(conv2d_count):
19                # syarat jika i samadengan bobotnya 0
20                if i == 0:
21                    # Penambahan method add pada variabel model
22                    dengan konvolusi 2 dimensi 32 bit didalamnya dan membuat
23                    kernel dengan ukuran 3 x 3 dan rumus aktifasi relu dan data
24                    shape yang di hitung dari data train .
25                    model.add(Conv2D(32, kernel_size=(3, 3),
26 activation='relu', input_shape=np.shape(train_input[0])))
27                    # jika tidak
28                    else:
29                        # Penambahan method add pada variabel model
30                        dengan konvolusi 2 dimensi 32 bit dengan ukuran kernel 3 x3
31                        dan fungsi aktivasi relu
32                        model.add(Conv2D(32, kernel_size=(3, 3),
33 activation='relu'))
34                        # Penambahan method add pada variabel model
35                        dengan isian method Max pooling berdimensi 2 dengan ukuran
36                        fixcel 2 x 2.
37                        model.add(MaxPooling2D(pool_size=(2, 2)))
38                        # merubah feature gambar menjadi 1 dimensi vektor
39                        model.add(Flatten())
40                        # Penambahan method dense untuk pematatan data dengan
41                        ukuran dense di tentukan dengan rumus fungsi tanh .
42                        model.add(Dense(dense_size , activation='tanh'))
43                        # membuat ketentuan jika pemangkasan lebih besar dari
44                        0 persen
45                        if dropout > 0.0:
46                            # Penambahan method dropout pada model dengan
47                            nilai dari dropout
48                            model.add(Dropout(dropout))
49                            # Penambahan method dense dengan fungsi num
50                            classs dan rumus softmax
51                            model.add(Dense(num_classes , activation='softmax'))
52                            # mongkompile variabel model dengan hasi loss
53                            optimasi dan akurasi matrix
54                            model.compile(loss='categorical_crossentropy',
55 optimizer='adam' , metrics=['accuracy'])
56                            # melakukan log pada dir
57                            log_dir = './logs/conv2d_%d-dense_%d-dropout_%.2f' %
58 (conv2d_count, dense_size , dropout)

```

```

40         # Menginisiasi variabel tensorboard dengan isian dari
41         library keras dan nilai dari lig dir
42         tensorboard = keras.callbacks.TensorBoard(log_dir=
43             log_dir)
44         # Menginisiasi variabel start dengan isian dari
45         library time menggunakan method time
46
47             start = time.time()
48             # Penambahan method fit pada model dengan data dari
49             train input train output nilai batch nilai epoch verbose
50             nilai 20 persen validation split dan callback dengan nilai
51             tnsorboard .
52             model.fit(train_input , train_output , batch_size=32,
53             epochs=10,
54             verbose=0, validation_split=0.2, callbacks
55             =[tensorboard])
56             # Menginisiasi variabel score dengan nilai evaluasi
57             dari model menggunakan data tes input dan tes output
58             score = model.evaluate(test_input , test_output ,
59             verbose=2)
60             # Menginisiasi variabel end
61             end = time.time()
62             # Menginisiasi variabel elapsed
63             elapsed = end - start
64             # mencetak hasil perhitungan
65             print("Conv2D count: %d, Dense size: %d, Dropout: %.2
66             f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count
67             , dense_size , dropout , score[0] , score[1] , elapsed))
68             results.append((conv2d_count , dense_size , dropout ,
69             score[0] , score[1] , elapsed))

```

14. Jelaskan kode program pada blok # In[14]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[14]: rebuild/retrain a model with the best parameters (from
2     the search) and use all data
3 # Menginisiasi variabel model dengan isian library Sequential
4 model = Sequential()
5 # variabel model di tambahkan library Conv2D tigapuluhan dua bit
6     dengan ukuran kernel 3 x 3 dan fungsi penghitungan relu dang
7     menggunakan data train_input
8 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
9     input_shape=np.shape(train_input[0])))
10 # variabel model di tambahkan dengan lib MaxPooling2D dengan
11     ketentuan ukuran 2 x 2 pixcel
12 model.add(MaxPooling2D(pool_size=(2, 2)))
13 # variabel model di tambahkan dengan library Conv2D 32 bit dengan
14     kernel 3 x 3
15 model.add(Conv2D(32, (3, 3), activation='relu'))
16 # variabel model di tambahkan dengan lib MaxPooling2D dengan
17     ketentuan ukuran 2 x 2 pixcel
18 model.add(MaxPooling2D(pool_size=(2, 2)))
19 # variabel model di tambahkan library Flatten

```

```
In [15]: model = Sequential()
.....
fungsi join pada library numpy.concatenate((train_input, test_input)), dimana dimana kita menggabungkan data train dengan data test dengan ukuran kernel 3 x 3 dan
.....
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
    input_shape=(train_input.shape[1], train_input.shape[2], 3)))
.....
model.add(MaxPooling2D(pool_size=(2, 2)))
.....
model.add(Flatten())
.....
model.add(Dense(128, activation='relu'))
.....
model.add(Dropout(0.5))
.....
model.add(Dense(num_classes, activation='softmax'))
.....
# mengkompile data model untuk mendapatkan data loss akurasi dan optimasi
.....
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
.....
# mencetak variabel model kemudian memunculkan kesimpulan berupa
# data total parameter, trainable parameter dan bukan trainable
# parameter
.....
print(model.summary())
```

Gambar 7.19 Hasil Nomor 14

```
13 model.add(Flatten())
14 # variabel model di tambahkan library Dense dengan fungsi tanh
15 model.add(Dense(128, activation='tanh'))
16 # variabel model di tambahkan library dropout untuk memangkas
# data tree sebesar 50 persen
17 model.add(Dropout(0.5))
18 # variabel model di tambahkan library Dense dengan data dari
# num_classes dan fungsi softmax
19 model.add(Dense(num_classes, activation='softmax'))
20 # mengkompile data model untuk mendapatkan data loss akurasi dan
# optimasi
21 model.compile(loss='categorical_crossentropy', optimizer='adam',
# metrics=['accuracy'])
22 # mencetak variabel model kemudian memunculkan kesimpulan berupa
# data total parameter, trainable parameter dan bukan trainable
# parameter
23 print(model.summary())
```

Haslinya:

15. Jelaskan kode program pada blok # In[15]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[15]:join train and test data so we train the network on all
# data we have available to us
2 model.fit(np.concatenate((train_input, test_input)),
3           np.concatenate((train_output, test_output)),
4           batch_size=32, epochs=10, verbose=2)
```

Kode diatas pada baris satu melakukan join numpy menggunakan data *train_input*, *test_input*, *da*

16. Jelaskan kode program pada blok # In[16]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[16]: save the trained model
2 model.save("mathsymbols.model")
```

Kode diatas untuk menyimpan model atau mengekspor model yang telah di jalankan tadi

17. Jelaskan kode program pada blok # In[17]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[17]: save label encoder (to reverse one-hot encoding)
2 np.save('classes.npy', label_encoder.classes_)
```

Kode diatas untuk menyimpan label encoder dengan nama classes.npy

18. Jelaskan kode program pada blok # In[18]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[18]: load the pre-trained model and predict the math symbol
          for an arbitrary image;
2 import keras.models
3 model2 = keras.models.load_model("mathsymbols.model")
4 print(model2.summary())
```

Kode diatas pertama mengimpport library keras model, kemudian Menginisiasi variabel model2 untuk meload model yang telah di simpan tadi dan mencetak hasil model2

19. Jelaskan kode program pada blok # In[19]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[19]: restore the class name to integer encoder
2 label_encoder2 = LabelEncoder()
3 label_encoder2.classes_ = np.load('classes.npy')
4 def predict(img_path):
5     newimg = keras.preprocessing.image.img_to_array(pil_image.
6         open(img_path))
6     newimg /= 255.0
7     # do the prediction
8     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
9
10    # figure out which output neuron had the highest score, and
11    # reverse the one-hot encoding
12    inverted = label_encoder2.inverse_transform([np.argmax(
13        prediction)])
14    print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
15        max(prediction)))
16
17 # In[20]: grab an image (we'll just use a random training image
18 # for demonstration purposes)
18 predict("HASYv2/hasy-data/v2-00010.png")
17 predict("HASYv2/hasy-data/v2-00500.png")
18 predict("HASYv2/hasy-data/v2-00700.png")
```

Kode diatas pertama Menginisiasi variabel label encoder ke 2 dengan isian fungsi label encoder.Kemudian Penambahan method classes dengan data classes yang di eksport tadi, selanjutnya membuat fumgsi predict dengan path img. Dan membagi data yang terdapat pada variabel newimg sebanyak 255 pada kode newimg /= 255.0

20. Jelaskan kode program pada blok # In[20]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

Kode diatas pada baris pertama adalah mencari prediksi menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2/hasy-data/v2-00010.png

Baris kedua mencari prediksi menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2/hasy-data/v2-00500.png

Baris ke3 mencari prediksi menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2/hasy-data/v2-00700.png

7.1.3 Bukti Tidak Plagiat

7.1.4 Link Youtube

<https://www.youtube.com/watch?v=-Oj5X4YD3ek>

BAB 8

CHAPTER 8

8.1 Nurul Izza Hamka - 1174062

8.1.1 Teori

1. Jelaskan kenapa file teks harus dilakukan tokenizer, dilengkapi dengan ilustrasi atau gambar

File harus di lakukan konversi terlebih dahulu menjadi angka sebelum dilakukan inputan menjadi neural network. Keras menyediakan sebuah kelas tokonizer, yang mana tokonizer ini berfungsi untuk melakukan konversi teks menjadi urutan integer indeks atau word count (tf-idf) dan vector binary.

Contoh:

Hasilnya akan tampak seperti ini:

2. Jelaskan konsep dasar K Fold Cross Validation pada dataset komentar Youtube pada kode listing 7.1 dilengkapi dengan ilustrasi atau gambar.

StratifiedKFold berisi presentasi sampel pada setiap kelas.

```

8 from keras.preprocessing.text import Tokenizer
9 tokenizer = Tokenizer()
10 texts = ["Budi makan nasi","Rudi makan nasi, nasi goreng."]
11 tokenizer.fit_on_texts(texts)
12
13 seq = tokenizer.texts_to_sequences(texts)
14
15 seq1 = tokenizer.texts_to_sequences(["nasi panas sekali"])
16 print("Index : "+str((tokenizer.word_index))
17 print("Seq. corpus: "+str(seq))
18 print("Seq. untuk 'nasi panas sekali': "+str(seq1))

```

Gambar 8.1

```

In [3]: Index: [4, 'budi': 3, 'nasi': 1, 'makan': 2, 'goreng': 5]
...: Seq. corpus:[[[3, 1, 1], [4, 2, 1, 1, 5]]]
...: Seq. untuk "nasi panas sekali": [[1]]

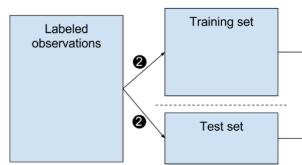
```

Gambar 8.2 Hasil

```

: kfolds = StratifiedKFold(n_splits=5)
: splits = kfolds.split(d, d['CLASS'])
Listing 7.1: K Fold Cross Validation

```

Gambar 8.3 K Fold Cross Validation**Gambar 8.4** Ilustrasi Gambar

3. Jelaskan apa maksudnya kode program for train, test in splits.dilengkapi dengan ilustrasi atau gambar.

Train / Test adalah metode untuk mengukur akurasi model Anda.

Ini disebut Train / Test karena Anda membagi set data menjadi dua set: satu set pelatihan dan satu set pengujian.

Split susunan atau matriks menjadi rangkaian acak train dan test.

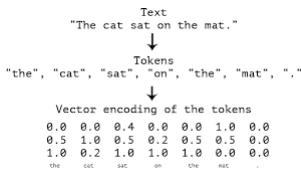
4. Jelaskan apa maksudnya kode program train content = d['CONTENT'].iloc[train_idx] dan test content = d['CONTENT'].iloc[test_idx]. dilengkapi dengan ilustrasi atau gambar.

Maksud dari kode program d['CONTENT'].iloc[train_idx] dan test content = d['CONTENT'].iloc[test_idx] adalah kita mengambil data dari kolom CONTENT yang mana merupakan bagian dari train_idx dan test_idx.

Ilustrasinya adalah Ketika data telah diubah menjadi train dan test maka nantinya kita bebas menampilkannya di kolom mana saja yang kita inginkan.

5. Jelaskan apa maksud dari fungsi tokonizer = Tokonizer (num words = 2000) dan tokonizer.fit on texts (train content), dilengkapi dengan ilustrasi atau gambar.

num words=2000 artinya jumlah kata maksimum yang harus disimpan, berdasarkan frekuensi kata, jadi jumlah kata maksimum adalah 2000.

**Gambar 8.5** Ilustrasi Gambar Nomor 6

```

>>> x = np.array([-1.2, 1.2])
>>> np.absolute(x)
array([1.2, 1.2])

```

Gambar 8.6 Ilustrasi Gambar Nomor 7

```

1 model = Sequential()
2 model.add(Dense(512, input_shape=(2000,)))
3 model.add(Activation('relu'))
4 model.add(Dropout(0.5))
5 model.add(Dense(12))
6 model.add(Activation('softmax'))

```

Listing 7.2. Membuat model Neural Network

Gambar 8.7 Listin 7.2

tokonizer.fit on text (train content) artinya kita akan melakukan fit tokonizer yang hanya untuk data train saja pada kolom CONTENT.

6. Jelaskan apa maksud dari fungsi `d_train.inputs = tokonizer.texts_to_matrix()` dan `d_test.inputs = tokonizer.to_matrix(test content, mode='tfidf')`, dilengkapi dengan ilustrasi kode dan atau gambar.

Maksud dari fungsi yang ada disoal adalah variable `d_train` akan melakukan tokonizer dari teks menjadi bentuk matrix dari data `train_content` yang mode matriknya ada tfidf.

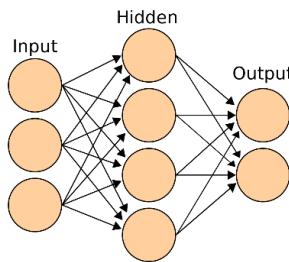
7. Jelaskan apa maksud dari fungsi `d_train.inputs = d_train.inputs / np.amax(np.absolute(d_train))` dan `d_test.inputs = d_test.inputs / np.amax(np.absolute(d_test))`, dilengkapi dengan ilustrasi atau gambar.

Fungsi pada soal diatas adalah membagi matrix tfidf tadi dengan amax, yaitu dengan mengembalikan nilai maks array. Yang hasilnya akan dimasukkan kedalam `d_trains_input` dan `d_test_inputs` dengan nilai absolute.

8. Jelaskan apa maksud fungsi dari `d_train_outputs=np_utils.to_categorical(d['CLASS'].iloc[:])` dan `d_test_outputs=np_utils.to_categorical(d['CLASS'].iloc[test_idx])` dalam kode program, dilengkapi dengan ilustrasi atau gambar.

Pada variable `d_train_output` dan `d_test_output` akan dilakukan one hot encoding dimana `np_utils` akan mengubah bentuk integer ke matrix kelas binear unuk kolom CLAS yang nantinya hanya ada 1 dan 0.

9. Jelaskan apa maksud dari fungsi di listing 7.2. Gambarkan ilustrasi Neural Network nya dari model kode tersebut.



Gambar 8.8 Ilustrasi Gambar Nomor 8

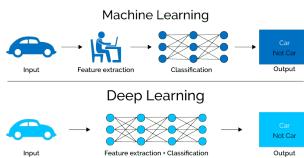
```

1 model.compile(loss='categorical_crossentropy', optimizer='adamax'
2 metrics=['accuracy'])
Listing 7.3: Compile model
  
```

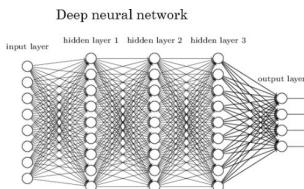
Gambar 8.9 Listing 7.3

- `model = Sequential ()` = pemodelan sequential
- `model.add (Dense (512, input shape = (2000,)))` = pada layer pertama dense dari 512 neouron untuk inputan sebelumnya yang sudah dijadikan matrix dengan jumlah 2000
- `model.add (Activation ('relu'))` = aktivasinya menggunakan function relu, artinya jika ada inputan dengan nilai maks maka inputan tersebut yang dipilih.
- `model.add (Dropout (0.5))` = Dropout akan melakukan pembobotan, dimana pemobobotanya 50% saja.
- `model.add (Dense (2))` = dense 2 mengkategorikan sebanyak 2 neuron.
- `model. Add (Activation ('softmax'))` = aktivasinya menggunakan function softmax.

10. Jelaskan apa maksud dari fungsi di listing 7.3 dengan parameter tersebut.
Melakukan compile dari model sequential dengan loss yang merupakan fungsi optimasi skor dengan categorical_crossentropy, dan menggunakan algoritma adama sebagai optimizer.
11. Jelaskan apa itu Deep Learning
Deep learning adalah metode implemtasi dari machine learning dengan tujuan meniru cara kerja pada otak manusia menggunakan neural network. Singkatnya deep learning ini model yang mempelajari metode komputasi dengan sendirinya.
12. Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning
Deep Neural Network adalah salah satu algoritma yang dapat digunakan dalam pengambilan keputusan, yang mana ini berbasis jaringan saraf. Sedangkan deep learning ini menggunakan deep neural network dalam menyelesaikan masalah pada machine learning.



Gambar 8.10 Ilustrasi Gambar Nomor 11



Gambar 8.11 Ilustrasi Gambar Nomor 12

```
In [1]: import csv
...::: from PIL import Image as pil_image
...::: import keras.preprocessing.image
Using TensorFlow backend.
```

Gambar 8.12 Hasil Nomor 1

13. Jelaskan dengan ilustrasi gambar buatan sendiri(langkah per langkah) bagaimana perhitungan algoritma konvolusi dengan ukuran stride (NPM mod3+1) x (NPM mod3+1) yang terdapat max pooling.(nilai 30)

8.1.2 Praktek Program

- Jelaskan kode program pada blok # In[1]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[1]: import lib
2 import csv
3 from PIL import Image as pil_image
4 import keras.preprocessing.image
```

Kode diatas adalah library untuk melakukan import file CSV. Kemudian untuk module pil image adalah untuk library keras.

Haslinya:

```
In [3]: import random
...: random.shuffle(imgs)
...: split_idx = int(0.8*len(imgs))
...: train = imgs[:split_idx]
...: test = imgs[split_idx:]
```

Gambar 8.13 Hasil Nomor 3

2. Jelaskan kode program pada blok # In[2]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In [2]: load all images (as numpy arrays) and save their classes
2 imgs = []
3 classes = []
4 with open('HASYv2/hasy-data-labels.csv') as csvfile:
5     csvreader = csv.reader(csvfile)
6     i = 0
7     for row in csvreader:
8         if i > 0:
9             img = keras.preprocessing.image.img_to_array(
10                pil_image.open("HASYv2/" + row[0]))
11                img /= 255.0
12                imgs.append((row[0], row[2], img))
13                classes.append(row[2])
14            i += 1
```

Kode diatas untuk menampilkan hasil load dataset dari HASYv2 sebagai file CSV, dan Menginisiasi variabel imgs dan classes dengan variabel array kosong, dan append untuk melampirkan atau menggabungkan data file.

3. Jelaskan kode program pada blok # In[3]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In [3]: shuffle the data, split into 80% train , 20% test
2 import random
3 random.shuffle(imgs)
4 split_idx = int(0.8*len(imgs))
5 train = imgs[:split_idx]
6 test = imgs[split_idx:]
```

import random untuk melakukan random pada vungsi imgs
random.shuffle(imgs) untuk Menginisiasi variabel split.idx dengan nilai integer 80 persen dikali dari pengembalian jumlah dari variabel imgs

Haslinya:

```
In [4]: import numpy as np
...: train_output = np.asarray(list(map(lambda row: row[2], train)))
...: train_input = np.asarray(list(map(lambda row: row[1], train)))
...: test_output = np.asarray(list(map(lambda row: row[1], test)))
...: test_input = np.asarray(list(map(lambda row: row[1], test)))
```

Gambar 8.14 Hasil Nomor 4

```
In [5]: from sklearn.preprocessing import LabelEncoder
...: from sklearn.preprocessing import OneHotEncoder
```

Gambar 8.15 Hasil Nomor 5

4. . Jelaskan kode program pada blok # In[4]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In [4]:
2 import numpy as np
3 train_input = np.asarray(list(map(lambda row: row[2], train)))
4 test_input = np.asarray(list(map(lambda row: row[2], test)))
5 train_output = np.asarray(list(map(lambda row: row[1], train)))
6 test_output = np.asarray(list(map(lambda row: row[1], test)))
```

Yang pertama adalah Melakukan import library numpy dengan inisial np, selanjutnya Menginisiasi variabel train input dengan np method asarray yang mana membuat array dengan isi row 2 dari data train

Haslinya:

5. Jelaskan kode program pada blok # In[5]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri

```
1 # In [5]: import encoder and one hot
2 from sklearn.preprocessing import LabelEncoder
3 from sklearn.preprocessing import OneHotEncoder
```

Pertama Melakukan import library LabelEncode dari sklearn dan kemudian Melakukan import library OneHotEncoder dari sklearn

Haslinya:

6. Jelaskan kode program pada blok # In[6]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[6]: convert class names into one-hot encoding
2 label_encoder = LabelEncoder()
3 integer_encoded = label_encoder.fit_transform(classes)

```

Kode diatas Menginisiasi variabel `label_encoder` dengan *isLabelEncoder* dan *Menginisiasi*

7. Jelaskan kode program pada blok # In[7]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[7]: then convert integers into one-hot encoding
2 onehot_encoder = OneHotEncoder(sparse=False)
3 integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
4 onehot_encoder.fit(integer_encoded)

```

Pada kode baris pertama diatas adalah Menginisiasi variabel `onehot_encoder` dengan *isOneHotEncoder*

8. . Jelaskan kode program pada blok # In[8]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[8]: convert train and test output to one-hot
2 train_output_int = label_encoder.transform(train_output)
3 train_output = onehot_encoder.transform(train_output_int.reshape(
    len(train_output_int), 1))
4 test_output_int = label_encoder.transform(test_output)
5 test_output = onehot_encoder.transform(test_output_int.reshape(
    len(test_output_int), 1))
6 num_classes = len(label_encoder.classes_)
7 print("Number of classes: %d" % num_classes)

```

Pada kode diatas adalah pertama Menconvert data train output menggunakan variabel `label_encoder` kedalam variabel `train_output_int`

Kemudian Menconvert variabel `train_output_int` kedalam fungsi `onehot_encoder` dan selanjutnya pada baris ke3 Menconvert data `test_output` menggunakan variabel `label_encoder`

9. Jelaskan kode program pada blok # In[9]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[9]: import sequential
2 from keras.models import Sequential
3 from keras.layers import Dense, Dropout, Flatten
4 from keras.layers import Conv2D, MaxPooling2D

```

Kode diatas adalah:

Baris 1 Melakukan import library Sequential dari Keras

```
In [9]: from keras.models import Sequential
...: from keras.layers import Dense, Dropout, Flatten
...: from keras.layers import Conv2D, MaxPooling2D
```

Gambar 8.16 Hasil Nomor 9

Baris 2 Melakukan import library Dense, Dropout, Flatten dari Keras
 Baris 3 Melakukan import library Conv2D, MaxPooling2D dari Keras.

Haslinya:

10. Jelaskan kode program pada blok # In[10]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[10]: desain jaringan
2 model = Sequential()
3 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
4                  input_shape=np.shape(train_input[0])))
5 model.add(MaxPooling2D(pool_size=(2, 2)))
6 model.add(Conv2D(32, (3, 3), activation='relu'))
7 model.add(MaxPooling2D(pool_size=(2, 2)))
8 model.add(Flatten())
9 model.add(Dense(1024, activation='tanh'))
10 model.add(Dropout(0.5))
11 model.add(Dense(num_classes, activation='softmax'))
12 model.compile(loss='categorical_crossentropy', optimizer='adam',
13                 metrics=['accuracy'])
14 print(model.summary())
```

Baris 1 Menginisiasi variabel model dengan isian library Sequential

Baris 2 variabel model di tambahkan library Conv2D tigapuluhan dua bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan relu dang menggunakan data train;np

Baris 3 variabel model ditambahkan dengan lib MaxPooling2D dengan ketentuan ukuran

Baris 4 variabel model ditambahkan dengan library Conv2D 32bit dengan kernel 3x3

Baris 5 variabel model ditambahkan dengan lib MaxPooling2D dengan ketentuan ukuran

Baris 6 variabel model ditambahkan library Flatten

Baris 7 variabel model ditambahkan library Dense dengan fungsi tanh

Baris 8 variabel model ditambahkan library dropout untuk memangkas data tree sebesar 50

Baris 9 variabel model ditambahkan library Dense dengan data darinum_classes dan fungsi

Baris 10 mengkompile data model untuk mendapatkan loss akurasid dan optimasi

Baris 11 mencetak variabel model kemudian memunculkan kesimpulan berupa data total p

Haslinya:

11. Jelaskan kode program pada blok # In[11]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
In [10]: model = Sequential()
...: model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
...: input_shape=(28, 28, 1)))
...: model.add(MaxPooling2D(pool_size=(2, 2)))
...: model.add(Conv2D(32, (3, 3), activation='relu'))
...: model.add(Flatten())
...: model.add(Dense(32, activation='tanh'))
...: model.add(Dropout(0.5))
...: model.add(Dense(num_classes, activation='softmax'))
...: model.compile(loss='categorical_crossentropy', optimizer='adam',
...: metrics=['accuracy'])
...: print(model.summary())
```

Gambar 8.17 Hasil Nomor 10

```
In [11]: import keras.callbacks
...: # Menginisiasi variabel tensorboard dengan isi lib keras
...: tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-
style')
```

Gambar 8.18 Hasil Nomor 11

```
1 # In[11]: import sequential
2 import keras.callbacks
3 tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-
style')
```

Haslinya:

12. Jelaskan kode program pada blok # In[12]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[12]: 5menit kali 10 epoch = 50 menit
2 # fungsi model titambahkan metod fit untuk mengetahui perhitungan
# dari train_input train_output
3 model.fit(train_input, train_output,
4 # dengan batch size 32 bit
5 #         batch_size=32,
6 #         epochs=10,
7 #         verbose=2,
8 #         validation_split=0.2,
9 #         callbacks=[tensorboard])
10
11 score = model.evaluate(test_input, test_output, verbose=2)
12 print('Test loss:', score[0])
13 print('Test accuracy:', score[1])
```

13. Jelaskan kode program pada blok # In[13]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[13]: try various model configurations and parameters to find
# the best
2 # Melakukan import library time
3 import time
```

```

4 #Menginisiasi variabel result dengan array kosong
5 results = []
6 # melakukan looping dengan ketentuan konvolusi 2 dimensi 1 2
7 for conv2d_count in [1, 2]:
8     # menentukan ukuran besaran fixcel dari data atau konvert 1
9     # fixcel mnjadi data yang berada pada codigan dibawah.
10    for dense_size in [128, 256, 512, 1024, 2048]:
11        # membuat looping untuk memangkas masing-masing data
12        dengan ketentuan 0 persen 25 persen 50 persen dan 75 persen .
13        for dropout in [0.0, 0.25, 0.50, 0.75]:
14            # Menginisiasi variabel model Sequential()
15            model = Sequential()
16            #membuat looping untuk variabel i dengan jarak dari
17            hasil konvolusi.
18            for i in range(conv2d_count):
19                # syarat jika i samadengan bobotnya 0
20                if i == 0:
21                    # Penambahan method add pada variabel model
22                    dengan konvolusi 2 dimensi 32 bit didalamnya dan membuat
23                    kernel dengan ukuran 3 x 3 dan rumus aktifasi relu dan data
24                    shape yang di hitung dari data train .
25                    model.add(Conv2D(32, kernel_size=(3, 3),
26 activation='relu', input_shape=np.shape(train_input[0])))
27                    # jika tidak
28                    else:
29                        # Penambahan method add pada variabel model
30                        dengan konvolusi 2 dimensi 32 bit dengan ukuran kernel 3 x3
31                        dan fungsi aktivasi relu
32                        model.add(Conv2D(32, kernel_size=(3, 3),
33 activation='relu'))
34                        # Penambahan method add pada variabel model
35                        dengan isian method Max pooling berdimensi 2 dengan ukuran
36                        fixcel 2 x 2.
37                        model.add(MaxPooling2D(pool_size=(2, 2)))
38                        # merubah feature gambar menjadi 1 dimensi vektor
39                        model.add(Flatten())
40                        # Penambahan method dense untuk pematatan data dengan
41                        ukuran dense di tentukan dengan rumus fungsi tanh .
42                        model.add(Dense(dense_size , activation='tanh'))
43                        # membuat ketentuan jika pemangkasan lebih besar dari
44                        0 persen
45                        if dropout > 0.0:
46                            # Penambahan method dropout pada model dengan
47                            nilai dari dropout
48                            model.add(Dropout(dropout))
49                            # Penambahan method dense dengan fungsi num
50                            classs dan rumus softmax
51                            model.add(Dense(num_classes , activation='softmax'))
52                            # mongkompile variabel model dengan hasi loss
53                            optimasi dan akurasi matrix
54                            model.compile(loss='categorical_crossentropy',
55 optimizer='adam' , metrics=['accuracy'])
56                            # melakukan log pada dir
57                            log_dir = './logs/conv2d_%d-dense_%d-dropout_%.2f' %
58 (conv2d_count, dense_size , dropout)

```

```

40         # Menginisiasi variabel tensorboard dengan isian dari
41         library keras dan nilai dari lig dir
42             tensorflow = keras.callbacks.TensorBoard(log_dir=
43                 log_dir)
44             # Menginisiasi variabel start dengan isian dari
45             library time menggunakan method time
46
47             start = time.time()
48             # Penambahan method fit pada model dengan data dari
49             train input train output nilai batch nilai epoch verbose
50             nilai 20 persen validation split dan callback dengan nilai
51             tensorflow.
52             model.fit(train_input, train_output, batch_size=32,
53             epochs=10,
54             verbose=0, validation_split=0.2, callbacks
55             =[tensorflow])
56             # Menginisiasi variabel score dengan nilai evaluasi
57             dari model menggunakan data tes input dan tes output
58             score = model.evaluate(test_input, test_output,
59             verbose=2)
60             # Menginisiasi variabel end
61             end = time.time()
62             # Menginisiasi variabel elapsed
63             elapsed = end - start
64             # mencetak hasil perhitungan
65             print("Conv2D count: %d, Dense size: %d, Dropout: %.2
66             f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count
67             , dense_size, dropout, score[0], score[1], elapsed))
68             results.append((conv2d_count, dense_size, dropout,
69             score[0], score[1], elapsed))

```

14. Jelaskan kode program pada blok # In[14]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[14]: rebuild/retrain a model with the best parameters (from
2     the search) and use all data
3 # Menginisiasi variabel model dengan isian library Sequential
4 model = Sequential()
5 # variabel model di tambahkan library Conv2D tigapuluhan dua bit
6     dengan ukuran kernel 3 x 3 dan fungsi penghitungan relu dang
7     menggunakan data train_input
8 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
9     input_shape=np.shape(train_input[0])))
10 # variabel model di tambahkan dengan lib MaxPooling2D dengan
11     ketentuan ukuran 2 x 2 pixel
12 model.add(MaxPooling2D(pool_size=(2, 2)))
13 # variabel model di tambahkan dengan library Conv2D 32 bit dengan
14     kernel 3 x 3
15 model.add(Conv2D(32, (3, 3), activation='relu'))
16 # variabel model di tambahkan dengan lib MaxPooling2D dengan
17     ketentuan ukuran 2 x 2 pixel
18 model.add(MaxPooling2D(pool_size=(2, 2)))
19 # variabel model di tambahkan library Flatten

```

```
In [15]: model = Sequential()
.....
for i in range(1, 6):
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
                    input_shape=(train_input[0].shape[1], train_input[0].shape[2], 3)))
    ....
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    ....
    model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())

```

Gambar 8.19 Hasil Nomor 14

```
13 model.add(Flatten())
14 # variabel model di tambahkan library Dense dengan fungsi tanh
15 model.add(Dense(128, activation='tanh'))
16 # variabel model di tambahkan library dropout untuk memangkas
# data tree sebesar 50 persen
17 model.add(Dropout(0.5))
18 # variabel model di tambahkan library Dense dengan data dari
# num_classes dan fungsi softmax
19 model.add(Dense(num_classes, activation='softmax'))
20 # mengkompile data model untuk mendapatkan data loss akurasi dan
# optimasi
21 model.compile(loss='categorical_crossentropy', optimizer='adam',
# metrics=['accuracy'])
22 # mencetak variabel model kemudian memunculkan kesimpulan berupa
# data total parameter, trainable parameter dan bukan trainable
# parameter
23 print(model.summary())
```

Haslinya:

15. Jelaskan kode program pada blok # In[15]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[15]:join train and test data so we train the network on all
# data we have available to us
2 model.fit(np.concatenate((train_input, test_input)),
3           np.concatenate((train_output, test_output)),
4           batch_size=32, epochs=10, verbose=2)
```

Kode diatas pada baris satu melakukan join numpy menggunakan data train_input , test_input , train_output dan test_output .

16. Jelaskan kode program pada blok # In[16]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # In[16]: save the trained model
2 model.save("mathsymbols.model")
```

Kode diatas untuk menyimpan model atau mengekspor model yang telah di jalankan tadi

17. Jelaskan kode program pada blok # In[17]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[17]: save label encoder (to reverse one-hot encoding)
2 np.save('classes.npy', label_encoder.classes_)

```

Kode diatas untuk menyimpan label encoder dengan nama classes.npy

18. Jelaskan kode program pada blok # In[18]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[18]: load the pre-trained model and predict the math symbol
          for an arbitrary image;
2 import keras.models
3 model2 = keras.models.load_model("mathsymbols.model")
4 print(model2.summary())

```

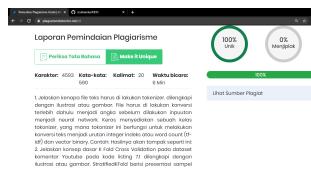
Kode diatas pertama mengimport library keras model, kemudian Menginisiasi variabel model2 untuk meload model yang telah di simpan tadi dan mencetak hasil model2

19. Jelaskan kode program pada blok # In[19]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In[19]: restore the class name to integer encoder
2 label_encoder2 = LabelEncoder()
3 label_encoder2.classes_ = np.load('classes.npy')
4 def predict(img_path):
5     newimg = keras.preprocessing.image.img_to_array(pil_image.
6         open(img_path))
6     newimg /= 255.0
7     # do the prediction
8     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
9
10    # figure out which output neuron had the highest score, and
11    # reverse the one-hot encoding
12    inverted = label_encoder2.inverse_transform([np.argmax(
13        prediction)])
14    print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
15        max(prediction)))
16
17 # In[20]: grab an image (we'll just use a random training image
18 # for demonstration purposes)
18 predict("HASYv2/hasy-data/v2-00010.png")
17 predict("HASYv2/hasy-data/v2-00500.png")
18 predict("HASYv2/hasy-data/v2-00700.png")

```



Gambar 8.20 Bukti Tidak Plagiat

Kode diatas pertama Menginisiasi variabel label encoder ke 2 dengan isian fungsi label encoder.Kemudian Penambahan method classess dengan data classess yang di eksport tadi, selanjutnya membuat fumgsi predict dengan path img. Dan membagi data yang terdapat pada variabel newimg sebanyak 255 pada kode newimg /= 255.0

20. Jelaskan kode program pada blok # In[20]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

Kode diatas pada baris pertama adalah mencari prediksi menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2/hasy-data/v2-00010.png

Baris kedua mencari prediksi menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2/hasy-data/v2-00500.png

Baris ke3 mencari prediksi menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2/hasy-data/v2-00700.png

8.1.3 Bukti Tidak Plagiat

8.1.4 Link youtube

<https://www.youtube.com/watch?v=zm7yEt5M-c>

BAB 9

CHAPTER 9

9.1 1174086 - Tia Nur Candida

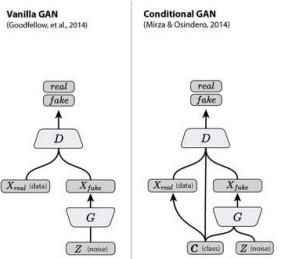
9.1.1 Teori

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

Vanilla GANs biasanya tidak memiliki convolutional Neural Jaringan (CNNs) di jaringan mereka. Conditional GANs (cGANs) adalah perpanjangan dari model GAN. Mereka memungkinkan untuk generasi gambar yang memiliki kondisi tertentu atau atribut dan telah terbukti menjadi lebih baik dari Vanilla GANs sebagai hasilnya.

cGANs adalah jenis GAN yang dikondisikan pada beberapa informasi tambahan. informasi tambahan y ke Generator sebagai lapisan input tambahan. Dalam Vanilla GANs, tidak ada kontrol atas Kategori gambar yang dihasilkan. Ketika kita menambahkan kondisi y ke Generator, kita dapat menghasilkan gambar dari kategori tertentu, menggunakan y, yang mungkin jenis data, seperti label kelas atau data integer. Vanilla GANs bisa belajar hanya satu kategori dan sangat sulit untuk arsitek GANs untuk beberapa kategori. Sebuah cGAN,

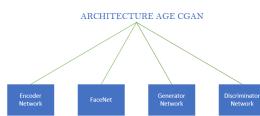
bagaimanapun, dapat digunakan untuk menghasilkan model multi-modal dengan kondisi yang berbeda untuk kategori yang berbeda.



Gambar 9.1 Illustrasi Vanilla GAN dan cGAN

2. Jelaskan dengan ilustrasi gambar sendiri arsitektur dari AgecGAN.

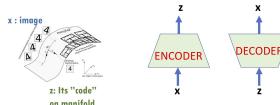
Arsitektur cGAN untuk penuaan wajah sedikit lebih rumit. AgecGan terdiri dari empat jaringan: Encoder, FaceNet, Jaringan Generator, dan jaringan diskriminasi. Dengan Encoder, kita belajar pemetaan invers gambar wajah masukan dan kondisi usia dengan vektor laten. FaceNet adalah jaringan pengenalan wajah yang mempelajari perbedaan antara gambar input x dan gambar yang direkonstruksi. Kami memiliki jaringan Generator, yang mengambil representasi tersembunyi yang terdiri dari gambar wajah dan vektor kondisi dan menghasilkan gambar. Jaringan diskriminasi adalah untuk mendiskriminasikan antara gambar nyata dan gambar palsu. Masalah dengan cGANs adalah bahwa mereka tidak dapat mempelajari tugas pemetaan terbalik masukan gambar x dengan atribut y ke vektor laten z . Solusi untuk masalah ini adalah dengan menggunakan jaringan Encoder. Kita dapat melatih jaringan encoder untuk memperkirakan pemetaan terbalik dari input Images x .



Gambar 9.2 Illustrasi Arsitektur cGAN

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

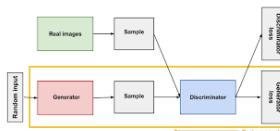
64, 3) dan mengubahnya menjadi vektor 100-dimensi. Jaringan Encoder adalah jaringan syaraf convolutional yang dalam. Jaringan berisi empat convolutional blok dan dua lapisan padat. Setiap blok convolutional berisi lapisan convolutional, lapisan normalisasi batch, dan fungsi aktivasi. Di setiap blok convolutional, setiap lapisan convolutional diikuti oleh lapisan normalisasi batch, kecuali lapisan convolutional pertama.



Gambar 9.3 Illustrasi Network Encoder

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

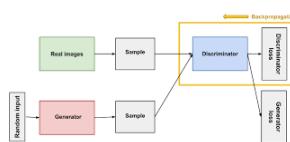
Tujuan utama dari generator adalah untuk menghasilkan gambar dari dimensi (64, 64, 3). Dibutuhkan vektor laten 100 dimensi dan beberapa informasi tambahan, y, dan mencoba untuk menghasilkan gambar yang realistik. Jaringan Generator adalah jaringan neural yang mendalam convolutional juga. Hal ini terdiri dari lapisan padat, upsampling, dan convolutional. Dibutuhkan dua nilai input: vektor kebisingan dan nilai pengkondisian. Nilai pengkondisian adalah informasi tambahan yang diberikan ke jaringan. Untuk Age-cGAN, ini akan menjadi usia.



Gambar 9.4 Illustrasi Network Generator

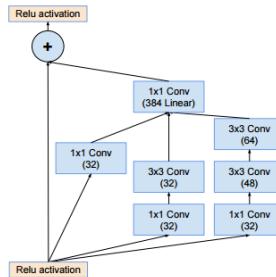
- Jelaskan dengan ilustrasi gambar sendiri arsitektur discriminator network dari Age-cGAN.

Tujuan utama dari jaringan diskriminator adalah untuk mengidentifikasi apakah gambar yang disediakan adalah palsu atau nyata. Hal ini dilakukan dengan melewati gambar melalui serangkaian lapisan sampling bawah dan beberapa lapisan klasifikasi. Dengan kata lain, ini memprediksi Apakah gambar itu nyata atau palsu. Seperti jaringan lain, Jaringan diskriminator lain dalam jaringan convolutional. Ini berisi beberapa blok convolutional. Setiap blok convolutional berisi lapisan convolutional, lapisan normalisasi batch, dan fungsi aktivasi, selain blok convolutional pertama, yang tidak memiliki lapisan normalisasi batch.



Gambar 9.5 Illustrasi Discriminator Network

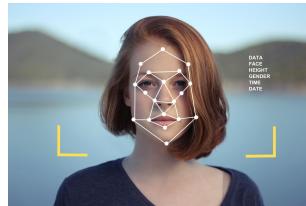
5. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model. pre-trained Inception-ResNet-2 network, sekali disediakan dengan gambar, mengembalikan yang sesuai embedding. Tertanam yang diekstrak untuk gambar asli dan gambar direkonstruksi dapat dihitung dengan menghitung jarak Euclidean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



Gambar 9.6 Illustrasi Inception-ResNet-2 Model.

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

Tujuan utama dari jaringan pengenalan wajah adalah untuk mengenali identitas seseorang dalam gambar yang diberikan.



Gambar 9.7 Illustrasi Face recognition network Age-cGAN.

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

Age-cGAN memiliki beberapa tahapan pelatihan. Seperti disebutkan di bagian

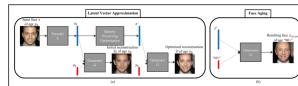
sebelumnya, Age-cGAN memiliki empat jaringan, yang dilatih dalam tiga tahap. Pelatihan AgecGAN terdiri dari tiga tahap:

- pelatihan GAN bersyarat: pada tahap ini, kita melatih jaringan Generator dan jaringan diskriminator.
- awal pendekatan vektor laten: pada tahap ini, kami melatih jaringan Encoder.
- optimasi vektor laten: pada tahap ini, kami mengoptimalkan kedua encoder dan jaringan generator.

8. Berikan contoh perhitungan fungsi training objektif

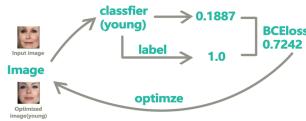
9. Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation

Perkiraan vektor laten awal adalah metode untuk memperkirakan vektor laten untuk mengoptimalkan rekonstruksi gambar wajah. Untuk memperkirakan vektor laten, kami memiliki jaringan Encoder. Kami melatih jaringan Encoder pada gambar yang dihasilkan dan gambar nyata. Setelah dilatih, Jaringan Encoder akan mulai menghasilkan vektor laten dari Distribusi. Tujuan pelatihan fungsi untuk pelatihan jaringan Encoder adalah kehilangan jarak Euclidean.



Gambar 9.8 Ilustrasi Initial latent vector approximation

10. Berikan contoh perhitungan latent vector optimization



Gambar 9.9 Contoh Perhitungan Latent vector optimization

9.1.2 Praktek

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

```
1 # In[1]: Ekstrak File]:
2 import tarfile
```

```

3 tf = tarfile.open("/content/drive/My Drive/Colab Notebooks/
      wiki_crop.tar")
4 tf.extractall(path="/content/drive/My Drive/Colab Notebooks")

```

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

```

1 # In[2. Load Data]:
2 def load_data(wiki_dir, dataset='wiki'):
3     # Load the wiki.mat file
4     meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset
5         )))
6
7     # Load the list of all files
8     full_path = meta[dataset][0, 0]["full_path"][0]
9
10    # List of Matlab serial date numbers
11    dob = meta[dataset][0, 0]["dob"][0]
12
13    # List of years when photo was taken
14    photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
15
16    # Calculate age for all dobs
17    age = [calculate_age(photo_taken[i], dob[i]) for i in range(
18        len(dob))]
19
20    # Create a list of tuples containing a pair of an image path
21    # and age
22    images = []
23    age_list = []
24    for index, image_path in enumerate(full_path):
25        images.append(image_path[0])
26        age_list.append(age[index])
27
28    # Return a list of all images and respective age
29    return images, age_list

```

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

```

1 # In[3. Encoder Bekerja]:
2 def build_encoder():
3     """
4         Encoder Network
5         """
6     input_layer = Input(shape=(64, 64, 3))
7
8     # 1st Convolutional Block
9     enc = Conv2D(filters=32, kernel_size=5, strides=2, padding='same')(input_layer)

```

```

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

```

4. Jelaskan bagaimana kode program The Generator Network bekerja dijelaskan dengan bahawa awam dengan ilustrasi sederhana. Proses Generator agar bekerja dengan baik dibutuhkan representasi dari gambar wajah dan vector kondisi sebagai inputan yang menghasilkan sebuah gambar.

```

1 # In[4. Generator Network Bekerja ]:
2 def build_generator():
3     """
4         Create a Generator Model with hyperparameters values defined
5             as follows
6         """
7
8     latent_dims = 100
9     num_classes = 6
10
11    input_z_noise = Input(shape=(latent_dims,))
12    input_label = Input(shape=(num_classes,))
13
14    x = concatenate([input_z_noise, input_label])
15
16    x = Dense(2048, input_dim=latent_dims + num_classes)(x)

```

```

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)

```

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

```

1 # In[5. Discriminator Network Bekerja]:
2 def build_discriminator():
3     """
4         Create a Discriminator Model with hyperparameters values
5             defined as follows
6         """
7
8     input_shape = (64, 64, 3)
9     label_shape = (6,)
10    image_input = Input(shape=input_shape)
11    label_input = Input(shape=label_shape)
12
13    x = Conv2D(64, kernel_size=3, strides=2, padding='same')(image_input)
14    x = LeakyReLU(alpha=0.2)(x)
15
16    label_input1 = Lambda(expand_label_input)(label_input)
17    x = concatenate([x, label_input1], axis=3)
18
19    x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
20    x = BatchNormalization()(x)
21    x = LeakyReLU(alpha=0.2)(x)
22
23    x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
24    x = BatchNormalization()(x)
25    x = LeakyReLU(alpha=0.2)(x)
26
27    x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
28    x = BatchNormalization()(x)
29    x = LeakyReLU(alpha=0.2)(x)
30
31    x = Flatten()(x)
32    x = Dense(1, activation='sigmoid')(x)
33
34    model = Model(inputs=[image_input, label_input], outputs=[x])
35    return model

```

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

```

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

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

```

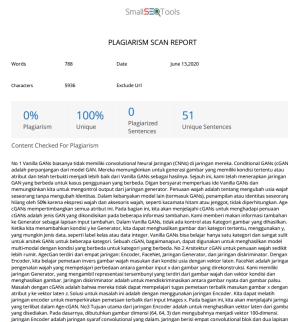
1 # In [7. Laten Vector]:
2     """
3         Train encoder
4     """
5
6     if TRAIN_ENCODER:
7         # Build and compile encoder
8         encoder = build_encoder()
9         encoder.compile(loss=euclidean_distance_loss, optimizer='adam')
10
11        # Load the generator network's weights
12        try:
13            generator.load_weights("generator.h5")
14        except Exception as e:
15            print("Error:", e)
16
17        z_i = np.random.normal(0, 1, size=(5000, z_shape))
18
19        y = np.random.randint(low=0, high=6, size=(5000,), dtype=
20                             np.int64)
21        num_classes = len(set(y))
22        y = np.reshape(np.array(y), [len(y), 1])
23        y = to_categorical(y, num_classes=num_classes)
```

```

23
24     for epoch in range(epochs):
25         print("Epoch:", epoch)
26
27         encoder_losses = []
28
29         number_of_batches = int(z_i.shape[0] / batch_size)
30         print("Number of batches:", number_of_batches)
31         for index in range(number_of_batches):
32             print("Batch:", index + 1)
33
34             z_batch = z_i[index * batch_size:(index + 1) *
35             batch_size]
35             y_batch = y[index * batch_size:(index + 1) *
36             batch_size]
37
38             generated_images = generator.predict_on_batch([
39             z_batch, y_batch])
40
41             # Train the encoder model
42             encoder_loss = encoder.train_on_batch(
43             generated_images, z_batch)
44             print("Encoder loss:", encoder_loss)
45
46             encoder_losses.append(encoder_loss)
47
48             # Write the encoder loss to Tensorboard
49             write_log(tensorboard, "encoder_loss", np.mean(
50             encoder_losses), epoch)
51
52             # Save the encoder model
53             encoder.save_weights("encoder.h5")

```

9.1.3 Bukti Tidak Plagiat



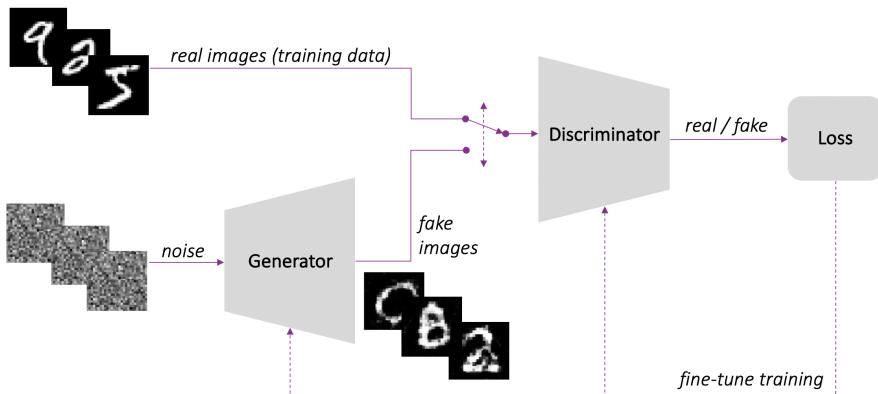
Gambar 9.10 Tidak Melakukan Plagiat Pada Ch 9

9.2 Muhammad Reza Syachrani - 1174084

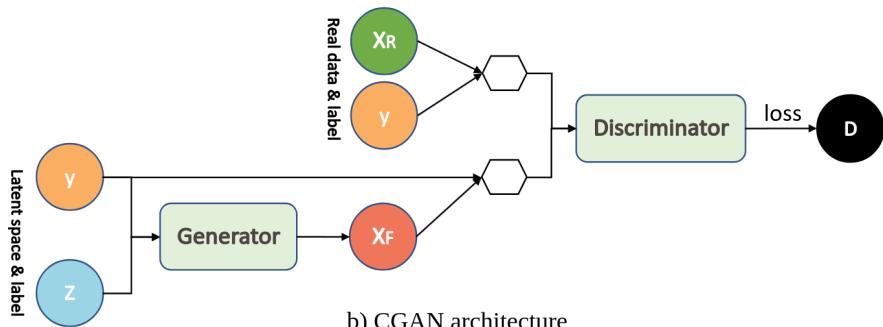
9.2.1 Teori

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

Vanilla GAN adalah tipe GAN paling sederhana. Di sini, Generator dan Diskriminatator adalah perceptron multi-layer sederhana. sedangkan pada CGAN (Conditional GAN) parameter ditambahkan 'y' ke Generator untuk menghasilkan data yang sesuai. Label juga dimasukkan ke dalam input ke Diskriminatator agar Diskriminatator membantu membedakan data nyata dari data yang dihasilkan palsu. Untuk ilustrasi, lihat gambar berikut:



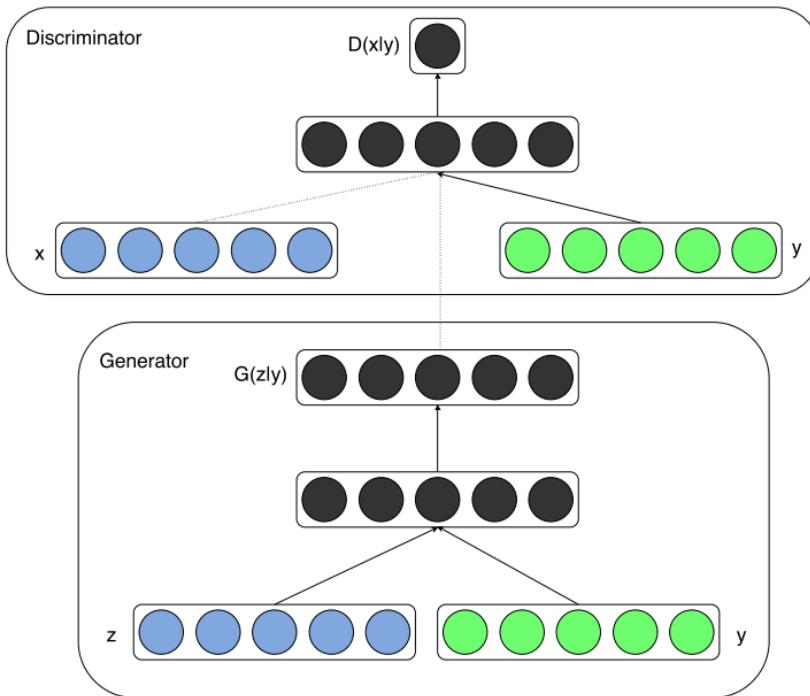
Gambar 9.11 Teori 1



b) CGAN architecture

Gambar 9.12 Teori 1

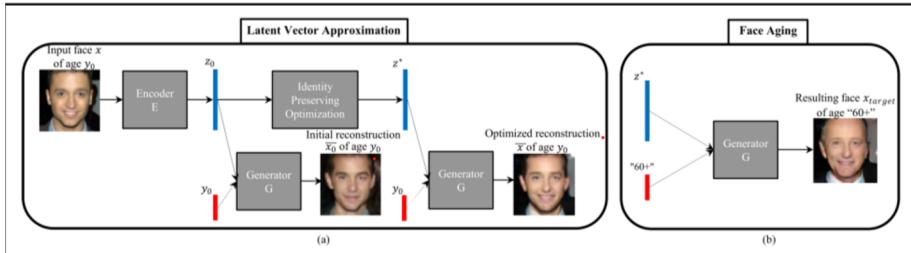
2. Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN.
 Age-cGan terdiri dari empat jaringan: encoder, FaceNet, jaringan generator, dan jaringan diskriminator Untuk ilustrasi, lihat gambar berikut:



Gambar 9.13 Teori 2

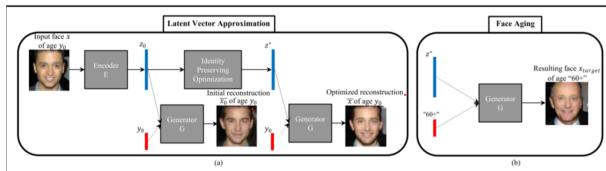
3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari Agec-GAN
 jaringan encoder adalah untuk menghasilkan vektor laten dari gambar yang disediakan. Pada dasarnya, ini mengambil gambar dari dimensi (64, 64, 3) dan mengubahnya menjadi vektor 100 dimensi. Jaringan encoder adalah jaringan saraf convolutional yang mendalam.

Untuk ilustrasi, lihat gambar berikut:

**Gambar 9.14** Teori 3

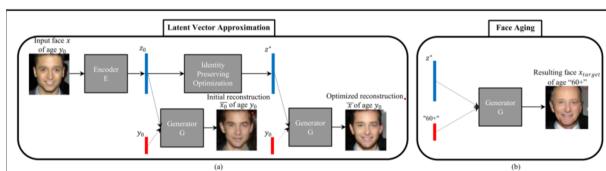
4. Jelaskan dengan ilustrasi gambar sendiri arsitektur generator network dari Agec-GAN.

jaringan generator adalah untuk menghasilkan gambar dengan dimensi (64, 64, 3). Dibutuhkan vektor laten 100 dimensi dan beberapa informasi tambahan, y , dan mencoba menghasilkan gambar yang realistik.

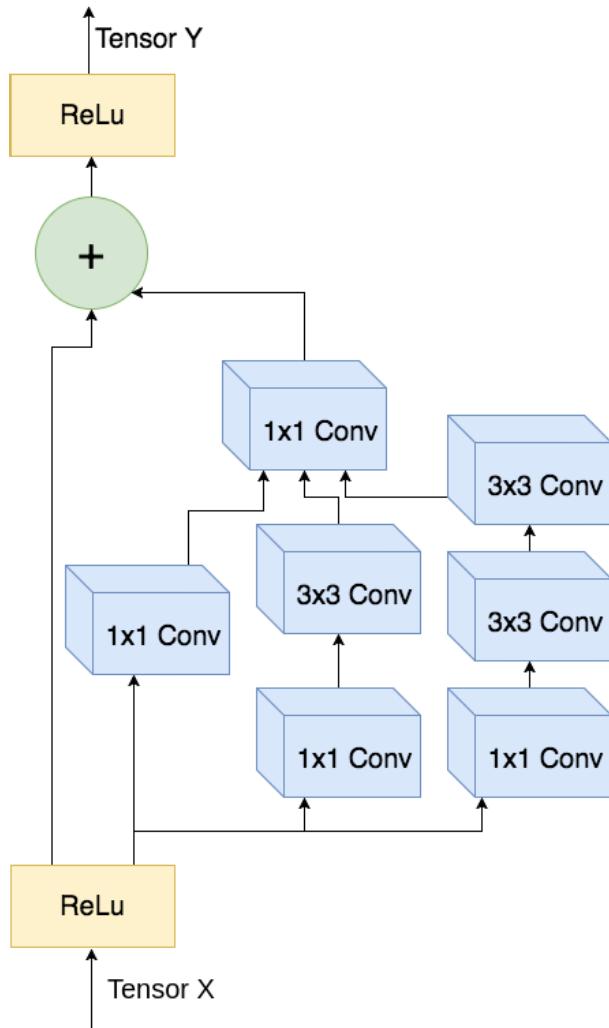
**Gambar 9.15** Teori 4

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

jaringan diskriminator adalah untuk mengidentifikasi apakah gambar yang disediakan adalah palsu atau nyata. Ini dilakukan dengan melewatkannya melalui serangkaian lapisan downsampling dan beberapa lapisan klasifikasi. ilustrasi pada gambar berikut :

**Gambar 9.16** Teori 5

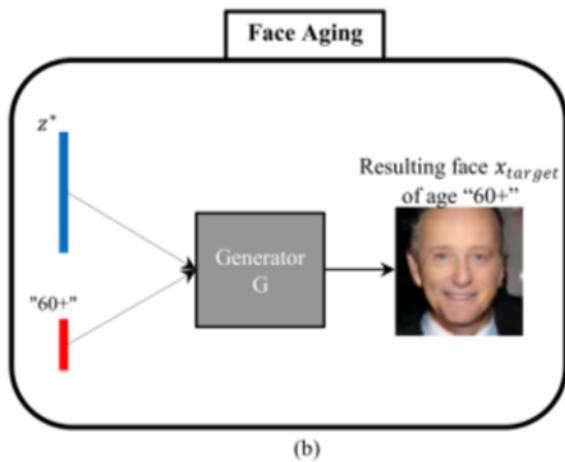
6. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model Pre-Trained Network atau Transfer Learning merupakan suatu metode penyelesaian yang memanfaatkan model yang sudah dilatih terhadap suatu dataset untuk menyelesaikan masalah dengan cara menggunakan sebagai starting point, memodifikasi dan mengupdate parameternya, sehingga sesuai dengan dataset yang baru.



Gambar 9.17 Teori 6

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

Face recognition network Age-cGAN adalah untuk mengenali identitas seseorang dalam gambar yang diberikan. dengan mempelajari perbedaan antara gambar input x dan gambar yang direkonstruksi



Gambar 9.18 Teori 7

8. Sebutkan dan jelaskan serta diertai contoh-contoh tahapan dari Age-cGAN
Pelatihan Age-cGAN terdiri dari tiga tahap:
 - (a) Conditional GAN training: Pada tahap ini, kami melatih jaringan generator dan jaringan diskriminator.
 - (b) Initial latent vector approximation : Pada tahap ini, kami melatih jaringan pembuat enkode.
 - (c) Latent vector optimization: Pada tahap ini, kami mengoptimalkan encoder dan jaringan generator.
9. Berikan contoh perhitungan fungsi training objektif
Objektif Training ialah untuk meminimalkan loss function sebagai log likelihood function yang diberikan pada persamaan dimana D melambangkan training data.

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z}|\mathbf{y})))].$$

Gambar 9.19 Teori 9

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

Latent vector approximation kemampuan untuk membuat gambar yang realistik dan tajam serta menghasilkan gambar wajah pada usia target.

11. Berikan contoh perhitungan latent vector optimization

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

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

Gambar 9.20 Teori 11

9.2.2 Praktek

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

- Login ke google colab menggunakan akun google
- Mount google drive
- Lakukan proses unzip melalui notebook python di google colab, unzip pakai codingan
- Selesai

```
1 import tarfile
2 tf = tarfile.open("/content/drive/My Drive/Colab Notebooks/CH9/
    wiki_crop.tar")
3 tf.extractall(path="/content/drive/My Drive/Colab Notebooks/CH9/
    wiki_crop")
```

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

Dibawah ini merupakan code untuk melakukan fungsi perhitungan usia.

```
1 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(
16      len(dob))]
17
18    # Create a list of tuples containing a pair of an image path
19    # and age
20    images = []
21    age_list = []
22    for index, image_path in enumerate(full_path):
23        images.append(image_path[0])
24        age_list.append(age[index])
25
26    # Return a list of all images and respective age
27    return images, age_list

```

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

```

1 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='same')(input_layer)
9     # enc = BatchNormalization()(enc)
10    enc = LeakyReLU(alpha=0.2)(enc)
11
12    # 2nd Convolutional Block
13    enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='same')(enc)
14    enc = BatchNormalization()(enc)
15    enc = LeakyReLU(alpha=0.2)(enc)
16
17    # 3rd Convolutional Block
18    enc = Conv2D(filters=128, kernel_size=5, strides=2, 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

```

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

Proses Generator agar bekerja dengan baik dibutuhkan representasi dari gambar wajah dan vector kondisi sebagai inputan yang menghasilkan sebuah gambar.

```

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

```

```

37     model = Model(inputs=[input_z_noise, input_label], outputs=[x])
38     return model
39

```

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

Proses Discriminator untuk membedakan antara gambar asli dan gambar palsu.

```

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

```

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

Proses Training cGAN ini dengan load file .mat pada dataset lalu epoch sebanyak 500 kali.

```

1 if __name__ == '__main__':
2     # Define hyperparameters
3     data_dir = "wiki_crop"
4     wiki_dir = os.path.join(data_dir, "wiki_crop")
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)

```

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 predksi epoch yang telah di buat sebanyak 500 kali, dan hasilnya disimpan pada folder result.

```

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) *
30                           batch_size]
31             y_batch = y[index * batch_size:(index + 1) *
32                           batch_size]

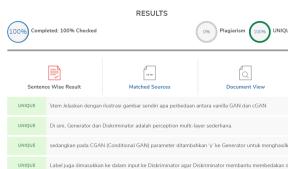
```

```

32         generated_images = generator.predict_on_batch([
33             z_batch, y_batch])
34
35             # Train the encoder model
36             encoder_loss = encoder.train_on_batch(
37                 generated_images, z_batch)
38                 print("Encoder loss:", encoder_loss)
39
40             encoder_losses.append(encoder_loss)
41
42             # Write the encoder loss to Tensorboard
43             write_log(tensorboard, "encoder_loss", np.mean(
44                 encoder_losses), epoch)
45
46             # Save the encoder model
47             encoder.save_weights("encoder.h5")

```

9.2.3 Bukti Tidak Plagiat



Gambar 9.21 plagiarism

9.2.4 Link Video Youtube

<https://youtu.be/fnc2PcVDkOU>

9.3 1174073 - Ainul Filiani

9.3.1 Teori

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

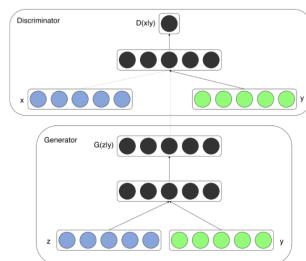
Vanilla GAN Vanilla GAN adalah tipe GAN paling sederhana. Di sini, Generator dan Diskriminator adalah perceptron multi-layer sederhana. Dalam vanilla GAN, algoritma ini sangat sederhana, ia mencoba untuk mengoptimalkan persamaan matematika menggunakan keturunan gradien stokastik. CGAN (Conditional GAN),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



Gambar 9.22 Valina GAN-cGAN

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

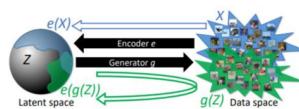
Age cGAN ialah dengan mengkondisikan model pada informasi tambahan dimungkinkan untuk mengarahkan proses pembuatan data. Pengkondisian semacam itu dapat didasarkan pada label kelas.



Gambar 9.23 Age-cGAN

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

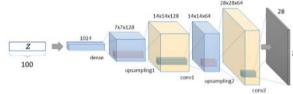
Arsitektur encoder biasanya digunakan untuk memodelkan struktur manifold dan membalikkan encoder untuk memproses data.



Gambar 9.24 Encoder Age cGANr

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

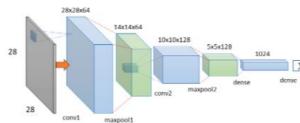
Arsitektur generator adalah sebuah array yang digunakan secara random, yang disebut seed. dari data seed tersebut, generator akan merubahnya menjadi sebuah gambar yang ukuran 28 x 28 dengan menggunakan Convolutional Neural Network.



Gambar 9.25 Network Age cGAN

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

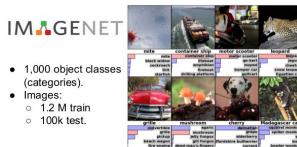
Arsitektur diskriminator adalah CNN yang dapat menerima input gambar yang berukuran 28,28 serta menghasilkan angka biner yang menyatakan apakah data yang diinputkan merupakan dataset asli atau gambar dataset palsu.



Gambar 9.26 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 9.27 Pretrained Inception ResNet

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

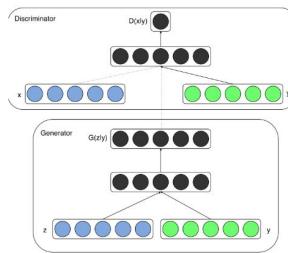
Face Recognition merupakan salah satu sistem yang mengimplementasi Deep Learning yang dapat mengenali wajah secara fisik dari gambar digital atau video frame.



Gambar 9.28 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 diskriminato. dimana untuk tahap generator sendiri membutuhkan vektor laten 100 serta menghasilkan gambar yang realistik dari dimensinya. sedangkan tahap diskriminator itu tahapan dimana memprediksi gambar yang diberikan nyata atau palsu.



Gambar 9.29 Tahap Age cGAN

9. Berikan contoh perhitungan fungsi training objektif.

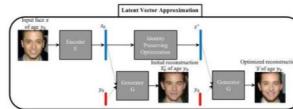
Objektif Training ialah untuk meminimalkan loss function sebagai log likelihood function yang diberikan pada persamaan dimana D melambangkan training data.

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

Gambar 9.30 Training Objektif

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

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



Gambar 9.31 Initial Latent Vector Approximation

- Berikan contoh perhitungan latent vector optimization.

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



Gambar 9.32 Latent Vector Optimization

9.3.2 Praktek

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

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

- Jelaskan bagaimana kode program bekerja untuk melakukan load terhadap dataset yang sudah di ekstrak, termasuk bagaimana penjelasan kode program perhitungan usia. Dibawah ini merupakan code untuk melakukan fungsi perhitungan usia.

```
1 # In[2. Load Data]:
2 def load_data(wiki_dir, dataset='wiki'):
3     # Load the wiki.mat file
4     meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset
5         )))
6
7     # Load the list of all files
8     full_path = meta[dataset][0, 0]["full_path"][0]
9
10    # List of Matlab serial date numbers
11    dob = meta[dataset][0, 0]["dob"][0]
12
13    # List of years when photo was taken
```

```

13 photo_taken = meta[dataset][0, 0][“photo_taken”][0] # year
14
15 # Calculate age for all dobs
16 age = [calculate_age(photo_taken[i], dob[i]) for i in range(
17 len(dob))]
18
19 # Create a list of tuples containing a pair of an image path
20 and age
21 images = []
22 age_list = []
23 for index, image_path in enumerate(full_path):
24     images.append(image_path[0])
25     age_list.append(age[index])
26
27 # Return a list of all images and respective age
28 return images, age_list

```

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

```

1 # In [3]: Encoder Bekerja]:
2 def build_encoder():
3     """
4         Encoder Network
5     """
6     input_layer = Input(shape=(64, 64, 3))
7
8     # 1st Convolutional Block
9     enc = Conv2D(filters=32, kernel_size=5, strides=2, padding='same')(input_layer)
10    # enc = BatchNormalization()(enc)
11    enc = LeakyReLU(alpha=0.2)(enc)
12
13    # 2nd Convolutional Block
14    enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='same')(enc)
15    enc = BatchNormalization()(enc)
16    enc = LeakyReLU(alpha=0.2)(enc)
17
18    # 3rd Convolutional Block
19    enc = Conv2D(filters=128, kernel_size=5, strides=2, padding='same')(enc)
20    enc = BatchNormalization()(enc)
21    enc = LeakyReLU(alpha=0.2)(enc)
22
23    # 4th Convolutional Block
24    enc = Conv2D(filters=256, kernel_size=5, strides=2, padding='same')(enc)
25    enc = BatchNormalization()(enc)
26    enc = LeakyReLU(alpha=0.2)(enc)
27
28    # Flatten layer
29    enc = Flatten()(enc)
30

```

```

31 # 1st Fully Connected Layer
32 enc = Dense(4096)(enc)
33 enc = BatchNormalization()(enc)
34 enc = LeakyReLU(alpha=0.2)(enc)
35
36 # Second Fully Connected Layer
37 enc = Dense(100)(enc)
38
39 # Create a model
40 model = Model(inputs=[input_layer], outputs=[enc])
41 return model

```

4. Jelaskan bagaimana kode program The Generator Network bekerja dengan ilustrasi sederhana. Proses Generator agar bekerja dengan baik dibutuhkan representasi dari gambar wajah dan vector kondisi sebagai inputan yang menghasilkan sebuah gambar.

```

1 # In [4. Generator Network Bekerja]:
2 def build_generator():
3     """
4         Create a Generator Model with hyperparameters values defined
5             as follows
6         """
7     latent_dims = 100
8     num_classes = 6
9
10    input_z_noise = Input(shape=(latent_dims,))
11    input_label = Input(shape=(num_classes,))
12
13    x = concatenate([input_z_noise, input_label])
14
15    x = Dense(2048, input_dim=latent_dims + num_classes)(x)
16    x = LeakyReLU(alpha=0.2)(x)
17    x = Dropout(0.2)(x)
18
19    x = Dense(256 * 8 * 8)(x)
20    x = BatchNormalization()(x)
21    x = LeakyReLU(alpha=0.2)(x)
22    x = Dropout(0.2)(x)
23
24    x = Reshape((8, 8, 256))(x)
25
26    x = UpSampling2D(size=(2, 2))(x)
27    x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
28    x = BatchNormalization(momentum=0.8)(x)
29    x = LeakyReLU(alpha=0.2)(x)
30
31    x = UpSampling2D(size=(2, 2))(x)

```

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

```

1 # In [5. Discriminator Network Bekerja]:
2 def build_discriminator():

```

```

3 """
4 Create a Discriminator Model with hyperparameters values
5 defined as follows
6 """
7
8 input_shape = (64, 64, 3)
9 label_shape = (6,)
10 image_input = Input(shape=input_shape)
11 label_input = Input(shape=label_shape)

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

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

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

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

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

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

28 model = Model(inputs=[image_input, label_input], outputs=[x])
29
30 return model
31
32
33

```

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

```

1 # In[6. Training cGAN]:
2 if __name__ == '__main__':
3     # Define hyperparameters
4     data_dir = "data"
5     wiki_dir = os.path.join(data_dir, "wiki_crop1")
6     epochs = 500
7     batch_size = 2
8     image_shape = (64, 64, 3)
9     z_shape = 100
10    TRAIN_GAN = True
11    TRAIN_ENCODER = False
12    TRAIN_GAN_WITH_FR = False
13    fr_image_shape = (192, 192, 3)

14    # 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)
        adversarial_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2
                               =0.999, epsilon=10e-8)

```

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

```

1 # In [7. Laten Vector]:
2 """
3     Train encoder
4 """
5
6 if TRAIN_ENCODER:
7     # Build and compile encoder
8     encoder = build_encoder()
9     encoder.compile(loss=euclidean_distance_loss, optimizer='adam')
10
11    # Load the generator network's weights
12    try:
13        generator.load_weights("generator.h5")
14    except Exception as e:
15        print("Error:", e)
16
17    z_i = np.random.normal(0, 1, size=(5000, z_shape))
18
19    y = np.random.randint(low=0, high=6, size=(5000,), dtype=np.int64)
20    num_classes = len(set(y))
21    y = np.reshape(np.array(y), [len(y), 1])
22    y = to_categorical(y, num_classes=num_classes)
23
24    for epoch in range(epochs):
25        print("Epoch:", epoch)
26
27    encoder_losses = []
28
29    number_of_batches = int(z_i.shape[0] / batch_size)
30    print("Number of batches:", number_of_batches)
31    for index in range(number_of_batches):
32        print("Batch:", index + 1)
33
34        z_batch = z_i[index * batch_size:(index + 1) *
35                      batch_size]
35        y_batch = y[index * batch_size:(index + 1) *
36                      batch_size]
37
38        generated_images = generator.predict_on_batch([
39            z_batch, y_batch])
40
41        # Train the encoder model
42        encoder_loss = encoder.train_on_batch(
43            generated_images, z_batch)

```

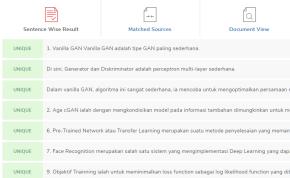
```

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

```

9.3.3 Penanganan Error

9.3.4 Bukti Tidak Plagiat



Gambar 9.33 Bukti Tidak Melakukan Plagiat Chapter 9

9.4 1174083 - Bakti Qilan Mufid

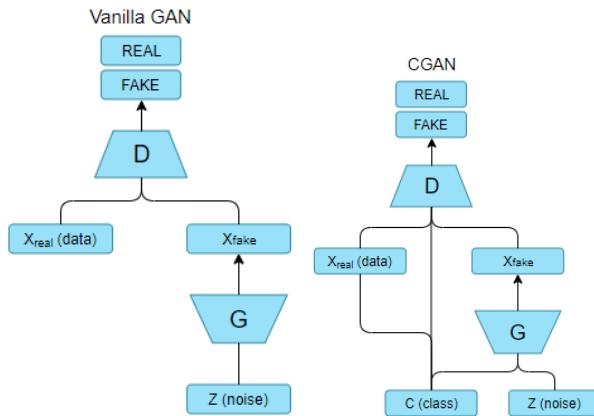
Chapter 9 - Conditional Generative Adversarial Network

9.4.1 Teori

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

1. Vanilla GAN merupakan tipe paling sederhana dari tipe-tipe yang ada pada GAN. Generator dan diskriminator adalah perceptron multi-layer sederhana. Perceptron adalah salah satu metode Jaringan Syaraf Tiruan (JST) sederhana yang menggunakan algoritma training untuk melakukan klasifikasi secara linier. Perceptron digunakan untuk melakukan klasifikasi sederhana dan membagi data untuk menentukan data mana yang masuk dalam klasifikasi dan data mana yang missklasifikasi (diluar klasifikasi). Vanilla GAN mengoptimalkan persamaan matematika menggunakan keturunan gradien stokastik(mempunyai unsur peluang).
2. cGAN(conditional GAN) merupakan tipe GAN yang dikondisikan pada beberapa tambahan informasi. Dalam GAN-projects informasi tambahan itu adalah y yang dimasukan ke generator sebagai lapisan input tambahan.

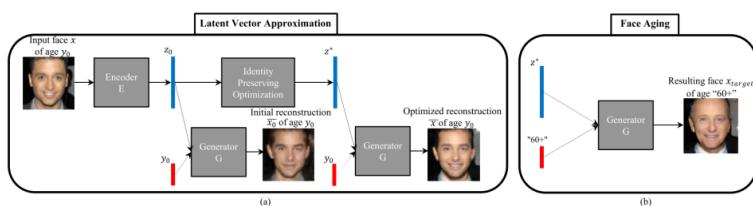
berikut gambaran antara Vanilla GAN dan cGAN



Gambar 9.34 gambaran penjelasan no. 1

9.4.1.2 Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN.
Age-cGAN terdiri dari empat jaringan, yaitu:

- Encoder, pemetaan terbalik dari gambar wajah input dan usia kondisi dengan vektor laten.
- FaceNet(face recognition network), mempelajari perbedaan antara gambar input x dan sebuah gambar yang direkonstruksi.
- Generator, membutuhkan sebuah representasi tersembunyi yang terdiri dari gambar wajah dan kondisi vektor dan akan menghasilkan gambar.
- Discriminator, melakukan diskriminasi antara gambar asli dan gambar palsu.

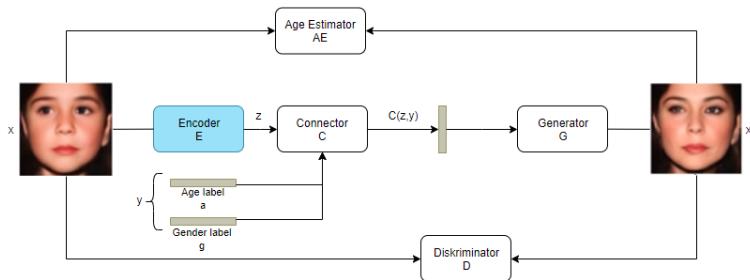


Gambar 9.35 gambaran penjelasan no. 2

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

Tujuan utama dari encoder network adalah menghasilkan latent vector dari gambar

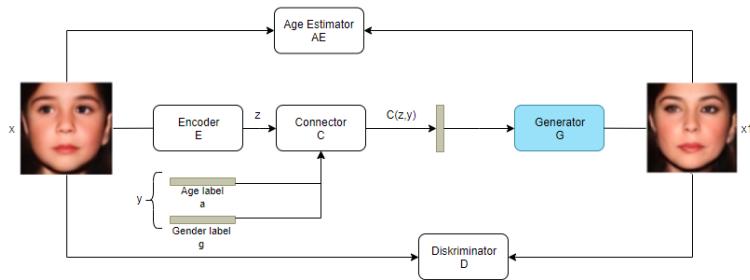
yang sudah disediakan. Pada dasarnya, dibutuhkan gambar dengan dimensi (64, 64, 3) lalu akan diubah menjadi vektor 100-dimensi. encoder network juga merupakan deep CNN. encoder network berisi empat blok konvolusional dan dua dense layer. pada setiap blok konvolusional mengandung sebuah layer konvolusional, batch normalization, dan fungsi aktivasi. Di setiap blok konvolusional, setiap konvolusional lapisan diikuti oleh lapisan normalisasi batch, kecuali yang pertama.



Gambar 9.36 gambaran penjelasan no. 3

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

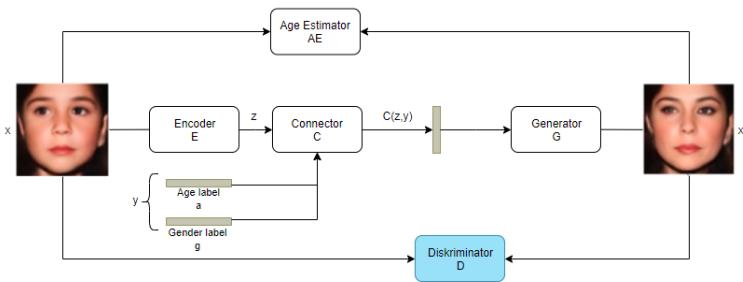
Tujuan utama dari generator network adalah menghasilkan gambar dengan dimensi (64,64,3). dibutuh kan vektor 100-dimensi dan beberapa tambahan informasi, y, dan mencoba menghasilkan gambar yang realistik. generator juga merupakan deep CNN, yang terbuat dari layer dense, upsampling, dan konvolusional layer. Dibutuhkan dua nilai input: sebuah noise vektor dan conditional value. Conditional value adalah informasi tambahan yang diberikan generator network. Untuk Age-cGAN, informasi tersebut merupakan usia.



Gambar 9.37 gambaran penjelasan no. 4

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

Tujuan utama dari diskriminasi network adalah untuk mengidentifikasi apakah gambar itu asli atau palsu, yang diketahui dengan cara mengolah gambar dalam down-sampling dan beberapa lapisan classification. atau bisa disebut memprediksi gambar tersebut palsu atau asli. Diskriminasi network juga merupakan deep CNN. Diskriminasi network juga terdiri dari beberapa konvolusional blok. setiap konvolusional blok memiliki konvolusional layer, kumpulan normalisasi layer, dan fungsi aktivasi. kecuali konvolusional blok pertama yang tidak memiliki kumpulan normalisasi layer.



Gambar 9.38 gambaran penjelasan no. 5

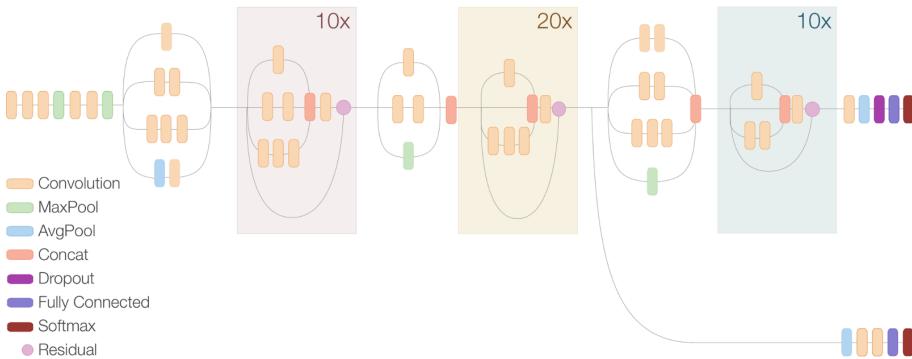
9.4.1.6 Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model.

Inception-ResNet-2 adalah jaringan saraf konvolusional(CNN) yang dilatih oleh lebih dari satu juta gambar dari database ImageNet(<http://www.image-net.org>). Jaringannya memiliki 164 lapisan dan dapat mengklasifikasikan gambar kedalam 1000 kategori objek, seperti keyboard, mouse, pensil, dan maca-macam binatang. hasilnya, jaringan tersebut sudah mempelajari representasi fitur yang kaya untuk berbagai gambar. jaringan ini memiliki ukuran input gambar 299-by-299.

Inception Resnet V2 Network



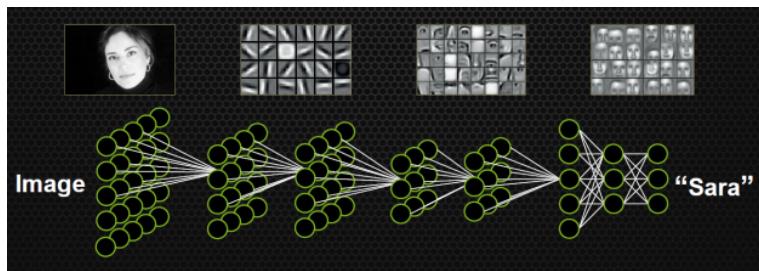
Compressed View



Gambar 9.39 gambaran penjelasan no. 6

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

Tujuan utama dari Face recognition adalah untuk mengenali identitas seseorang dalam gambar yang diberikan.



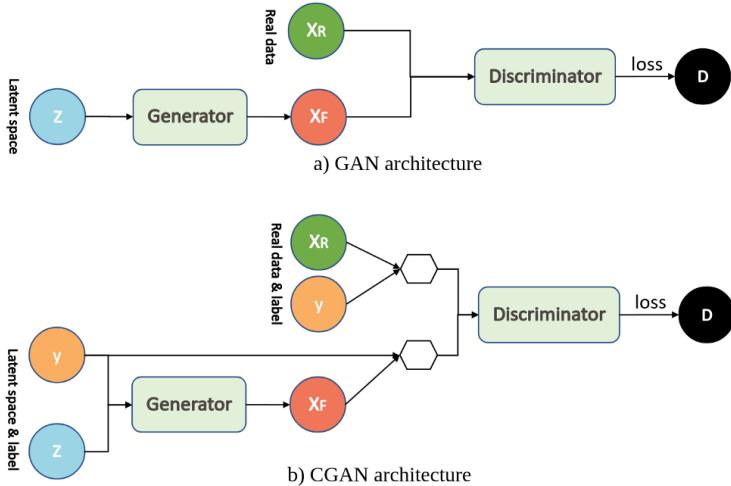
Gambar 9.40 gambaran penjelasan no. 7

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

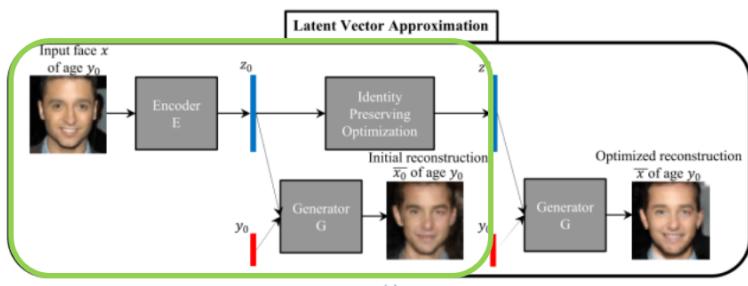
tahapan dari Age-cGAN ada tiga tahap, diantaranya:

1. Conditional GAN training, pada tahap ini Age-cGAN melatih generator dan diskriminator network.
2. Initial latent vector approximation, pada tahap ini Age-cGAN melatih encoder network.

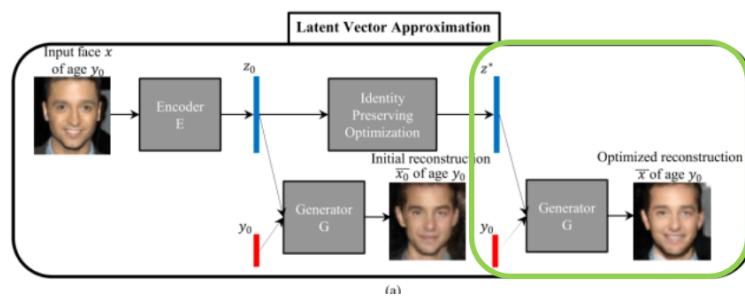
3. Latent vector optimization, pada tahap ini Age-cGAN mengoptimalkan encoder dan generator network.



Gambar 9.41 gambaran penjelasan no. 8(Conditional GAN training)



Gambar 9.42 gambaran penjelasan no. 8(Initial latent vector approximation)



Gambar 9.43 gambaran penjelasan no. 8(Latent vector optimization)

9.4.1.9 Berikan contoh perhitungan fungsi training objektif

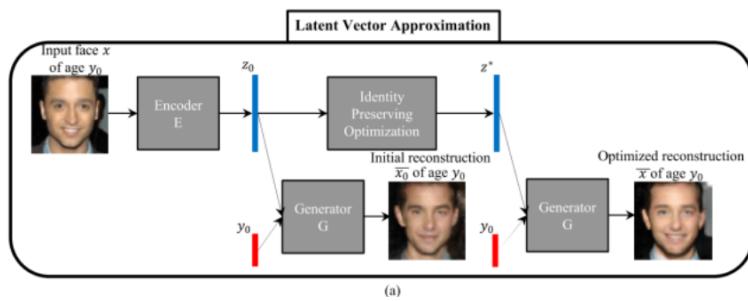
melatih jaringan cGAN melibatkan optimalisasi fungsi $v(\Theta_G, \Theta_D)$. melatih cGAN bisa dianggap game minimax(Algoritma minimax akan melakukan pengecekan pada seluruh kemungkinan yang ada, sehingga akan menghasilkan pohon permainan yang berisi semua kemungkinan permainan tersebut), dimana generator dan diskriminatore keduanya dilatih secara bersamaan. Dalam persamaan sebelumnya P_x merupakan parameter jaringan generator, Θ_D merupakan parameter G dan D, $\log D(x, y)$ merupakan loss dari model diskriminatore, $\log(1 - D(G(z, \tilde{y}), \tilde{y}))$ merupakan loss dari model generator, dan P_{data} merupakan distribusi dari kemungkinan semua gambar.

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

Gambar 9.44 gambaran penjelasan no. 9

9.4.1.10 Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation

Perkiraan latent vektor awal adalah metode yang memperkirakan latent vektor untuk mengoptimalkan rekonstruksi gambar wajah. Untuk itu maka diperlukan network encoder. Network encoder dilatih oleh gambar yang degenerate dan gambar yang nyata. Setelah dilatih, network encoder akan mulai menghasilkan vektor laten dari distribusi yang dipelajari. fungsinya untuk melatih network encoder sebagai Euclidean distance loss.



Gambar 9.45 gambaran penjelasan no. 10

9.4.1.11 Berikan contoh perhitungan latent vector optimization

ketika latent vector optimization berlangsung, Age-cGAN mengoptimalkan network encoder dan network generator secara bersamaan. persamaan yang akan digunakan adalah seperti berikut:

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

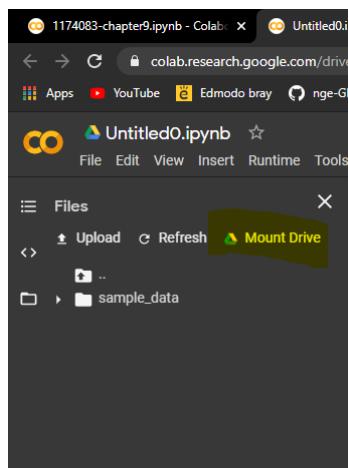
FR adalah jaringan pengenalan wajah. persamaan ini menunjukkan bahwa jarak Euclidean antara gambar asli dan gambar yang direkonstruksi harus minimal. pada tahap ini, Age-cGAN mencoba meminimalkan jarak dan memaksimalkan pelestarian identitas.

9.4.2 Praktek

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

Caranya cukup mudah, kita tinggal ikuti langkah-langkah berikut:

- Login terlebih dahulu ke google colab menggunakan akun google
- lalu buka link datasetnya (<https://drive.google.com/open?id=1NoV357ZvemE5dLCGySN>) lalu copy dataset ke akun google drive.
- Setelah di copy, buka google colab, buat notebook baru dan klik Mount Drive
- Lalu unzip file wiki_crop.tar. maka dataset siap digunakan.



Gambar 9.46 gambar penjelasan praktek soal no.1

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

file dataset wiki_crop.tar berisi 62.328 gambar dan sebuah file bernama wiki.mat yang menampung labelnya. Library spicy.io mempunyai method yang bernama loadmat, dimana method tersebut sangat memudahkan ketika kita ingin me-load file .mat. lalu untuk me-load datanya, kita bisa menggunakan kode 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(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(
16        dob))]
17
18    # Create a list of tuples containing a pair of an image path and
19    # age
20    images = []
21    age_list = []
22    for index, image_path in enumerate(full_path):
23        images.append(image_path[0])
24        age_list.append(age[index])

```

```

24     # Return a list of all images and respective age
25     return images, age_list

```

variabel photo_taken adalah daftar tahun dan dob adalah tanggal serial Matlab nomor untuk gambar yang sesuai dalam daftar. kita bisa mengukur umur seseorang dari seri tanggal dan tahun gambar tersebut diambil, dengan menggunakan kode 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

```

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

Berikut merupakan langkah-langkah encoder network bekerja:

1. Mulai dengan membuat layer input

```

1     input_layer = Input(shape=(64, 64, 3))

```

2. Lalu tambahkan blok konvolusi pertama, yang berisi lapisan konvolusi 2D dan fungsi aktivasi dengan konfigurasi sebagai berikut:

- Filters: 32
- Kernel size: 5
- Strides: 2
- Padding: same
- Activation: LeakyReLU with alpha equal to 0.2:

```

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

```

3. Selanjutnya, tambahkan tiga blok konvolusi lagi, yang masing-masing berisi lapisan konvolusi 2D dan diikuti oleh lapisan normalisasi batch serta fungsi aktivasi, seperti pada konfigurasi berikut:

- Filters: 64, 128, 256
- Kernel size: 5, 5, 5
- Strides: 2, 2, 2
- Padding: same, same, same

- Batch normalization: Each convolutional layer is followed by a batch normalization layer
- Activations: LealyReLU, LeakyReLU, LeakyReLU with alpha equal to 0.2:

```

1      # 2nd Convolutional Block
2      enc = Conv2D(filters=64, kernel_size=5, strides=2,
3      padding='same')(enc)
4      enc = BatchNormalization()(enc)
5      enc = LeakyReLU(alpha=0.2)(enc)

6      # 3rd Convolutional Block
7      enc = Conv2D(filters=128, kernel_size=5, strides=2,
8      padding='same')(enc)
9      enc = BatchNormalization()(enc)
10     enc = LeakyReLU(alpha=0.2)(enc)

11     # 4th Convolutional Block
12     enc = Conv2D(filters=256, kernel_size=5, strides=2,
13     padding='same')(enc)
14     enc = BatchNormalization()(enc)
15     enc = LeakyReLU(alpha=0.2)(enc)

```

4. Selanjutnya meratakan(flatten) output dari blok konvolusi yang terakhir, sebagai berikut:

```

1      # Flatten layer
2      enc = Flatten()(enc)

```

5. Selanjutnya tambahkan dense layer(yang terhubung secara penuh) dan dikikuti dengan batch normaisasi layer serta fungsi aktivasi, dengan konfigurasi berikut:

- Units (nodes): 2,096
- Batch normalization: Yes
- Activation: LeakyReLU with alpha equal to 0.2:

```

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

```

6. Selanjutnya, tambahkan dense layer kedua(yang terhubung secara penuh), dengan konfigurasi berikut:

- Units (nodes): 100
- Activation: None:

```

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

```

7. Terakhir, buat model Keras dan tentukan input serta output untuk network encoder.

```

1      model = Model(inputs=[input_layer], outputs=[enc])

```

9.4.2.4 Jelaskan bagaimana kode program The Generator Network bekerja di-jelaskan dengan bahasa awam dengan ilustrasi sederhana

Berikut merupakan langkah-langkah generator network bekerja:

1. Mulai dengan membuat dua lapisan input ke generator network.

```

1 latent_dims = 100
2 num_classes = 6
3
4 input_z_noise = Input(shape=(latent_dims ,)) #Input layer for
vector z
5 input_label = Input(shape=(num_classes ,)) #Input layer for
conditioning variable

```

2. Selanjutnya, gabungkan input sepanjang dimensi channel, seperti berikut:

```
1 x = concatenate([input_z_noise , input_label])
```

3. Lalu tambahkan blok dense (terhubung secara penuh) dengan konfigurasi berikut:

- Units (nodes): 2,048
- Input dimension: 106
- Activation: LeakyReLU with alpha equal to 0.2
- Dropout: 0.2:

```

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

```

4. Selanjutnya tambahkan blok dense kedua, dengan konfigurasi berikut:

- Units (nodes): 16,384
- Batch normalization: Yes
- Activation: LeakyReLU with alpha equal to 0.2
- Dropout: 0.2:

```

1 x = Dense(256 * 8 * 8)(x)
2 x = BatchNormalization()(x)
3 x = LeakyReLU(alpha=0.2)(x)
4 x = Dropout(0.2)(x)

```

5. Selanjutnya bentuk kembali output dari lapisan dense terakhir ke 3D tensor dengan dimensi(8,8,256):

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

6. Selanjutnya tambahkan blok upsampling yang berisi lapisan upsampling diikuti oleh lapisan konvolusi 2D serta batch normalisasi dengan konfigurasi berikut:

- Upsampling size: (2, 2)
- Filters: 128
- Kernel size: 5
- Padding: same
- Batch normalization: Yes, with momentum equal to 0.8
- Activation: LeakyReLU with alpha equal to 0.2:

```

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

```

7. Selanjutnya, tambahkan blok upsampling lain(mirip dengan lapisan sebelumnya), seperti pada kode berikut, dan konfigurasinya mirip seperti blok sebelumnya, kecuali jumlah filter yang digunakan dalam lapisan konvolusi yang asalnya 128 menjadi 64:

```

1   x = Reshape((8, 8, 256))(x)
2
3   x = UpSampling2D(size=(2, 2))(x)
4   x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
5   x = BatchNormalization(momentum=0.8)(x)
6   x = LeakyReLU(alpha=0.2)(x)
7
8   x = UpSampling2D(size=(2, 2))(x)
9   x = Conv2D(filters=64, kernel_size=5, padding='same')(x)
10  x = BatchNormalization(momentum=0.8)(x)
11  x = LeakyReLU(alpha=0.2)(x)

```

8. Selanjutnya tambahkan blok upsampling terakhir. konfigurasinya mirip dengan lapisan sebelumnya, kecuali tiga lapisan filter akan digunakan dan batch normalisasi tidak digunakan:

```

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

```

9. Terakhir, buat model Keras dan tentukan input serta output untuk generator network:

```

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

```

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

Berikut merupakan langkah-langkah diskriminator network bekerja:

1. Mulailah dengan membuat dua lapisan input, karena pada project ini diskriminator akan memproses dua input:

```

1  input_shape = (64, 64, 3) #Input image shape
2  label_shape = (6,) #Input conditioning variable shape
3  image_input = Input(shape=input_shape) #Two input layers
4  label_input = Input(shape=label_shape) #Two input layers

```

2. Selanjutnya, tambahkan blok konvolusi 2D (fungsi Conv2D + aktivasi) dengan konfigurasi berikut:

- Filters = 64
- Kernel size: 3
- Strides: 2
- Padding: same
- Activation: LeakyReLU with alpha equal to 0.2:

```

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

```

3. Selanjutnya perluas label_input sehingga memiliki bentuk (32,32,6):

```
1  label_input1 = Lambda(expand_label_input)(label_input)
```

fungsi expand_label_input sebagai berikut:

```

1 def expand_label_input(x):
2     x = K.expand_dims(x, axis=1)
3     x = K.expand_dims(x, axis=1)
4     x = K.tile(x, [1, 32, 32, 1])
5     return x

```

4. Selanjutnya gabungkan tensor label yang telah ditransformasi dan output terakhir dari lapisan konvolusi, seperti berikut:

```
1  x = concatenate([x, label_input1], axis=3)
```

5. Tambahkan blok konvolusi (lapisan konvolusi 2D + batch normalisasi + fungsi aktivasi) dengan konfigurasi berikut:

- Filters: 128
- Kernel size: 3
- Strides: 2
- Padding: same
- Batch normalization: Yes
- Activation: LeakyReLU with alpha equal to 0.2:

```

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

```

6. Selanjutnya tambahkan dua blok konvolusi lagi, seperti berikut:

```

1   x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
2   x = BatchNormalization()(x)
3   x = LeakyReLU(alpha=0.2)(x)
4
5   x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
6   x = BatchNormalization()(x)
7   x = LeakyReLU(alpha=0.2)(x)

```

7. Lalu tambahkan lapisan flatten

```
1   x = Flatten()(x)
```

8. lalu tambahkan lapisan dense(lapisan pengklasifikasian) yang menghasilkan probabilitas:

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

9. Terakhir, buat model Keras dan tentukan input serta output untuk diskriminator network.

```
1   model = Model(inputs=[image_input, label_input], outputs=[x])
```

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

Berikut adalah langkah-langkah melatih cGAN:

1. Mulailah dengan menentukan parameter yang diperlukan untuk pelatihan:

```

1   # Define hyperparameters
2   data_dir = "data"
3   wiki_dir = os.path.join(data_dir, "wiki_crop1")
4   epochs = 500
5   batch_size = 2
6   image_shape = (64, 64, 3)
7   z_shape = 100
8   TRAIN_GAN = True
9   TRAIN_ENCODER = False
10  TRAIN_GAN_WITH_FR = False
11  fr_image_shape = (192, 192, 3)

```

2. Selanjutnya, tentukan optimisator untuk pelatihan. Kali ini kita akan menggunakan Adam optimizer yang terdapat di Keras. Untuk inisiasinya seperti berikut

```

1   # Define optimizers
2   # Optimizer for the discriminator network
3   dis_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
4   epsilon=10e-8)

```

```

4 # Optimizer for the generator network
5 gen_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
6 epsilon=10e-8)
# Optimizer for the adversarial network
7 adversarial_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2
=0.999, epsilon=10e-8)
```

gunakan equal rate sama dengan 0,0002, nilai beta_1 sama dengan 0,5, nilai beta_2 sama dengan 0,999, dan nilai epsilon sama dengan 10e-8 semua optimizer.

- Selanjutnya, muat dan kompilasi generator network dan diskriminator network. Dalam Keras kita harus mengkompilasi network terlebih dahulu sebelum melatihnya.

```

1 # Build and compile the discriminator network
2 discriminator = build_discriminator()
3 discriminator.compile(loss=['binary_crossentropy'], optimizer=
dis_optimizer)
4
5 # Build and compile the generator network
6 generator = build_generator()
7 generator.compile(loss=['binary_crossentropy'], optimizer=
gen_optimizer)
```

- Selanjutnya, buat dan kompilasi model adversial, sebagai berikut:

```

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

- Selanjutnya tambahkan TensorBoard untuk menyimpan losses, seperti berikut:

```

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

- Selanjutnya load semua gambar menggunakan fungsi load_data yang didefinisikan pada bagian mempersiapkan data:

```

1 images, age_list = load_data(wiki_dir=wiki_dir, dataset="wiki
")
```

- Selanjutnya konversikan nilai numerik usia ke kategori usia, seperti berikut:

```

1 # Convert age to category
```

Definisi dari fungsi age_to_category seperti kode berikut:

```

1 def age_to_category(age_list):
2     age_list1 = []
3
4     for age in age_list:
5         if 0 < age <= 18:
6             age_category = 0
7         elif 18 < age <= 29:
8             age_category = 1
9         elif 29 < age <= 39:
10            age_category = 2
11        elif 39 < age <= 49:
12            age_category = 3
13        elif 49 < age <= 59:
14            age_category = 4
15        elif age >= 60:
16            age_category = 5
17
18    age_list1.append(age_category)
19
20 return age_list1

```

8. Selanjutnya load semua gambar dan buat ndarray yang berisi semua gambar:

```

1 # Read all images and create an ndarray
2 loaded_images = load_images(wiki_dir, images, (image_shape
[0], image_shape[1]))

```

Definisi dari fungsi load_data seperti kode berikut:

```

1 def load_images(data_dir, image_paths, image_shape):
2     images = None
3
4     for i, image_path in enumerate(image_paths):
5         print()
6         try:
7             # Load image
8             loaded_image = image.load_img(os.path.join(data_dir,
image_path), target_size=image_shape)
9
10            # Convert PIL image to numpy ndarray
11            loaded_image = image.img_to_array(loaded_image)
12
13            # Add another dimension (Add batch dimension)
14            loaded_image = np.expand_dims(loaded_image, axis=0)
15
16            # Concatenate all images into one tensor
17            if images is None:
18                images = loaded_image
19            else:
20                images = np.concatenate([images, loaded_image],
axis=0)
21            except Exception as e:
22                print("Error:", i, e)
23
24 return images

```

9. Selanjutnya buat perulangan(for) berdasarkan jumlah epoch, seperti berikut:

```

1   for epoch in range(epochs):
2       print("Epoch:{} ".format(epoch))
3
4       gen_losses = []
5       dis_losses = []
6
7       number_of_batches = int(len(loader_images) /
batch_size)
8       print("Number of batches:", number_of_batches)

```

10. Selanjutnya buat perulangan yang lain didalam perulangan epoch berdasarkan num_batches, seperti berikut:

```

1   for index in range(number_of_batches):
2       print("Batch:{} ".format(index + 1))

```

11. Selanjutnya buat sample dari batch gambar dari dataset yang asli dan batch dari one-hot encoded age vector:

```

1   images_batch = loader_images[index * batch_size:(index + 1) * batch_size]
2   images_batch = images_batch / 127.5 - 1.0
3   images_batch = images_batch.astype(np.float32)
4
5   y_batch = y[index * batch_size:(index + 1) * batch_size]

```

bentuk dari image_batch (batch_size, 64, 64, 3) dan bentuk dari y_batch (batch_size, 6).

12. Selanjutnya buat sampel batch noise vektor dari distribusi Gaussian, seperti berikut:

```

1   z_noise = np.random.normal(0, 1, size=(batch_size
, z_shape))

```

13. Selanjutnya buat gambar palsu(fake) menggunakan generator network. perlu diingat bahwa kita belum melatih network generator.

```

1   # Generate fake images
2   initial_recon_images = generator.predict_on_batch(
([z_noise, y_batch]))

```

14. Sekarang latih network diskriminator pada gambar yang asli dan juga gambar yang palsu

```

1   d_loss_real = discriminator.train_on_batch([
images_batch, y_batch], real_labels)
2   d_loss_fake = discriminator.train_on_batch([
initial_recon_images, y_batch], fake_labels)

```

15. Lalu latih adversial network dan mempause diskriminator network. atau kita hanya akan melatih generator network.

```

1      # Again sample a batch of noise vectors from a
2      Gaussian(normal) distribution
3      z_noise2 = np.random.normal(0, 1, size=(batch_size, z.shape))
4          # Samples a batch of random age values
5          random_labels = np.random.randint(0, 6,
6          batch_size).reshape(-1, 1)
7          # Convert the random age values to one-hot
8          encoders
9          random_labels = to_categorical(random_labels, 6)
10         # Train the generator network
11         g_loss = adversarial_model.train_on_batch([
12             z_noise2, random_labels], [1] * batch_size)

```

16. Selanjutnya hitung dan cetak lossesnya:

```

1     print("g_loss:{}".format(g_loss))
2     # Add losses to their respective lists
3     gen_losses.append(g_loss)
4     dis_losses.append(d_loss)

```

17. Selanjutnya tulis losses ke TensorBoard untuk divisualisasikan

```

1      # Write losses to Tensorboard
2      write_log(tensorboard, 'g_loss', np.mean(gen_losses),
3      epoch)
4      write_log(tensorboard, 'd_loss', np.mean(dis_losses),
5      epoch)

```

18. Buat sampel dan simpan gambar setiap 10 epoch, seperti berikut:

```

1 if epoch % 10 == 0:
2     images_batch = loaded_images[0:batch_size]
3     images_batch = images_batch / 127.5 - 1.0
4     images_batch = images_batch.astype(np.float32)
5
6     y_batch = y[0:batch_size]
7     z_noise = np.random.normal(0, 1, size=(batch_size,
8     , z.shape))
9
10    gen_images = generator.predict_on_batch([z_noise,
11        y_batch])
12
13    for i, img in enumerate(gen_images[:5]):
14        save_rgb_img(img, path="results/img_{}_{}.png"
15        .format(epoch, i))

```

Letakan blok kode sebelumnya didalam perulangan epoch. setelah setiap 10 epoch, maka akan menghasilkan batch dari gambar palsu dan akan disimpan ke direktori results. untuk fungsi save_rgb_img adalah sebagai fungsi utilitas dan didefinisikan seperti berikut:

```

1 def save_rgb_img(img, path):
2     """
3     Save an rgb image
4     """
5     fig = plt.figure()
6     ax = fig.add_subplot(1, 1, 1)
7     ax.imshow(img)
8     ax.axis("off")
9     ax.set_title("Image")
10
11    plt.savefig(path)
12    plt.close()

```

19. Terakhir simpan kedua model dengan menambahkan baris berikut:

```

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

```

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

Seperti yang sudah diketahui bahwa cGAN tidak belajar untuk memetakan balik dari gambar ke vektor latent. oleh karena itu, encoder yang akan mempelajari pemetaan balik ini dan mampu menghasilkan vektor laten yang dapat digunakan untuk menghasilkan gambar wajah pada usia yang ditargetkan.

Kita telah mendefinisikan hyperparameter yang dibutuhkan dalam pelatihan. Dan untuk langkah-langkahnya seperti berikut:

1. Mulailah dengan membuat network encoder. tambahkan kode berikut untuk membuat dan mengompilasi network encoder.

```

1 # Build and compile encoder
2 encoder = build_encoder()
3 encoder.compile(loss=euclidean_distance_loss, optimizer='adam')

```

Jika belum mendefinisikan euclidean_distance_loss, maka definisikan terlebih dahulu seperti pada kode berikut:

```

1 def euclidean_distance_loss(y_true, y_pred):
2     """
3     Euclidean distance loss
4     """
5     return K.sqrt(K.sum(K.square(y_pred - y_true), axis=-1))

```

2. Selanjutnya load network generator, seperti berikut:

```
1 generator.load_weights("generator.h5")
```

3. Selanjutnya buat sampel batc vektor latent, seperti berikut:

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

4. Selanjutnya buat sampel batch acak angka usia dan konversikan usia angka acak tadi kedalam one-hot encoded vector, seperti berikut:

```

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

```

5. Selanjutnya tambahkan perulangan epoch dan batch dari langkah-langkah. seperti berikut:

```

1     for epoch in range(epochs):
2         print("Epoch:", epoch)
3
4         encoder_losses = []
5
6         number_of_batches = int(z_i.shape[0] / batch_size)
7         print("Number of batches:", number_of_batches)
8         for index in range(number_of_batches):
9             print("Batch:", index + 1)

```

6. Sekarang buat sampel batch vektor latent dan batch one-hot encoded vector dari 1000 sampel, seperti berikut:

```

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

```

7. Selanjutnya hasilkan gambar palsu menggunakan network generator yang sudah dilatih:

```

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

```

8. Lalu latih network encoder pada gambar yang dihasilkan oleh network generator:

```

1     # Train the encoder model
2     encoder_loss = encoder.train_on_batch(
3         generated_images, z_batch)

```

9. Selanjutnya tulis loss encoder ke TensorBoard pada setiap epoch.

```

1     # Write the encoder loss to Tensorboard
2     write_log(tensorboard, "encoder_loss", np.mean(
3         encoder_losses), epoch)

```

10. dan kita perlu menyimpan network encoder yang sudah dilatih. simpan model encoder dengan kode berikut:

```

1     # Save the encoder model
2     encoder.save_weights("encoder.h5")

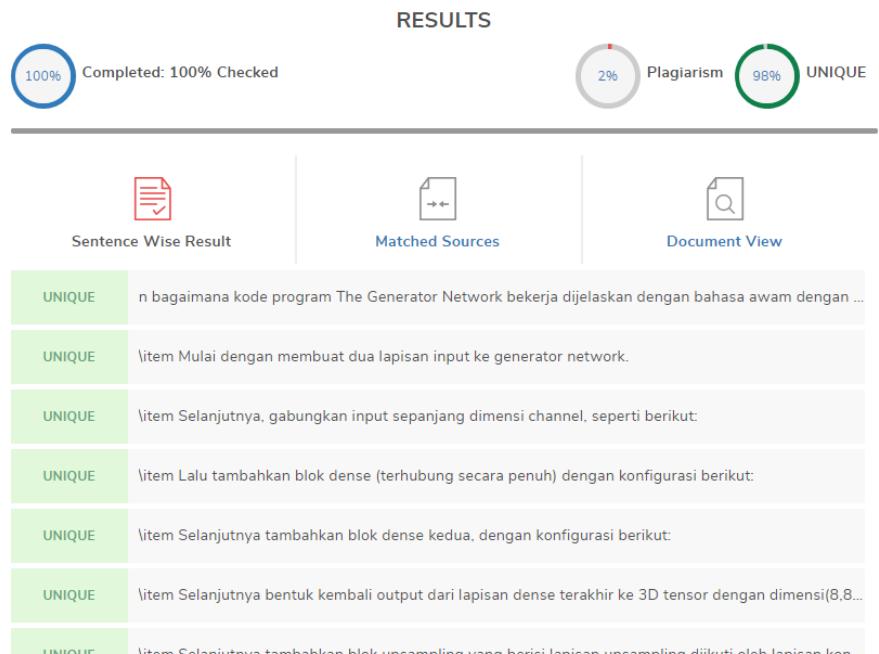
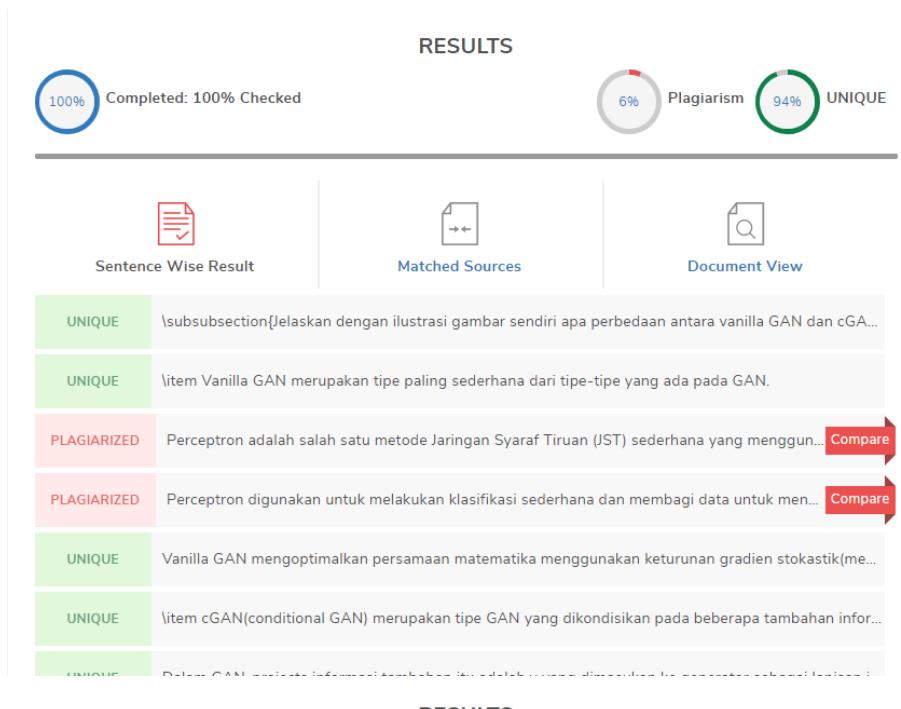
```

9.4.3 Penanganan Error

9.4.3.1 *Terjadi error*

9.4.3.2 *Solusi*

9.4.4 Bukti Tidak Plagiat



Gambar 9.47 Bukti tidak plagiat

9.4.5 Link Youtube

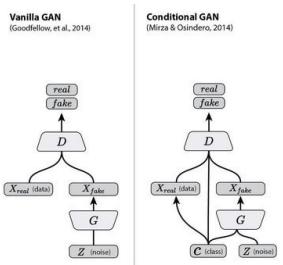
<https://bit.ly/baktiListVideo>

9.5 1174066 - D.Irga B. Naufal Fakhri

9.5.1 Teori

9.5.1.1 Jelaskan dengan ilustrasi gambar sendiri apa perbedaan antara vanilla GAN dan cGAN. Vanilla GANs biasanya tidak memiliki convolutional Neural Jaringan (CNNs) di jaringan mereka. Conditional GANs (cGANs) adalah perpanjangan dari model GAN. Mereka memungkinkan untuk generasi gambar yang memiliki kondisi tertentu atau atribut dan telah terbukti menjadi lebih baik dari Vanilla GANs sebagai hasilnya.

cGANs adalah jenis GAN yang dikondisikan pada beberapa informasi tambahan. informasi tambahan y ke Generator sebagai lapisan input tambahan. Dalam Vanilla GANs, tidak ada kontrol atas Kategori gambar yang dihasilkan. Ketika kita menambahkan kondisi y ke Generator, kita dapat menghasilkan gambar dari kategori tertentu, menggunakan y, yang mungkin jenis data, seperti label kelas atau data integer. Vanilla GANs bisa belajar hanya satu kategori dan sangat sulit untuk arsitek GANs untuk beberapa kategori. Sebuah cGAN, bagaimanapun, dapat digunakan untuk menghasilkan model multi-modal dengan kondisi yang berbeda untuk kategori yang berbeda.

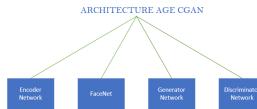


Gambar 9.48 Illustrasi Vanilla GAN dan cGAN

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

Arsitektur cGAN untuk penuaan wajah sedikit lebih rumit. AgecGan terdiri dari empat jaringan: Encoder, FaceNet, Jaringan Generator, dan jaringan diskriminator. Dengan Encoder, kita belajar pemetaan invers gambar wajah masukan dan kondisi usia dengan vektor laten. FaceNet adalah jaringan pengenalan wajah yang mempelajari perbedaan antara gambar input x dan gambar yang direkonstruksi. Kami memiliki jaringan Generator, yang mengambil representasi tersembunyi yang terdiri dari gambar wajah dan vektor kondisi dan menghasilkan gambar. Jaringan diskriminator adalah untuk mendiskriminasikan antara gambar nyata dan gambar palsu. Masalah

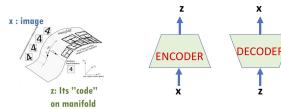
dengan cGANs adalah bahwa mereka tidak dapat mempelajari tugas pemetaan terbalik masukan gambar x dengan atribut y ke vektor laten z . Solusi untuk masalah ini adalah dengan menggunakan jaringan Encoder. Kita dapat melatih jaringan encoder untuk memperkirakan pemetaan terbalik dari input Images x .



Gambar 9.49 Illustrasi Arsitektur cGAN

9.5.1.3 Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari AgecGAN.

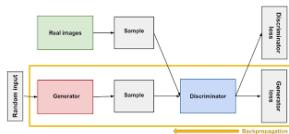
Tujuan utama dari jaringan Encoder adalah untuk menghasilkan vektor laten dari gambar yang disediakan. Pada dasarnya, dibutuhkan gambar dimensi (64, 64, 3) dan mengubahnya menjadi vektor 100-dimensi. Jaringan Encoder adalah jaringan syaraf convolutional yang dalam. Jaringan berisi empat convolutional blok dan dua lapisan padat. Setiap blok convolutional berisi lapisan convolutional, lapisan normalisasi batch, dan fungsi aktivasi. Di setiap blok convolutional, setiap lapisan convolutional diikuti oleh lapisan normalisasi batch, kecuali lapisan convolutional pertama.



Gambar 9.50 Illustrasi Network Encoder

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

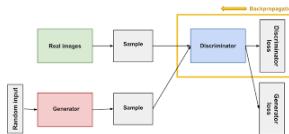
Tujuan utama dari generator adalah untuk menghasilkan gambar dari dimensi (64, 64, 3). Dibutuhkan vektor laten 100 dimensi dan beberapa informasi tambahan, y , dan mencoba untuk menghasilkan gambar yang realistik. Jaringan Generator adalah jaringan neural yang mendalam convolutional juga. Hal ini terdiri dari lapisan padat, upsampling, dan convolutional. Dibutuhkan dua nilai input: vektor kebisikan dan nilai pengkondisian. Nilai pengkondisian adalah informasi tambahan yang diberikan ke jaringan. Untuk Age-cGAN, ini akan menjadi usia.



Gambar 9.51 Illustrasi Network Generator

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

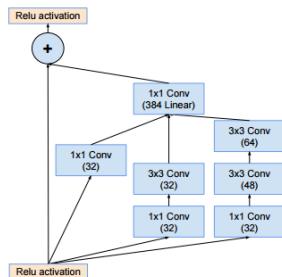
Tujuan utama dari jaringan diskriminasi adalah untuk mengidentifikasi apakah gambar yang disediakan adalah palsu atau nyata. Hal ini dilakukan dengan melalui gambar melalui rangkaian lapisan sampling bawah dan beberapa lapisan klasifikasi. Dengan kata lain, ini memprediksi Apakah gambar itu nyata atau palsu. Seperti jaringan lain, Jaringan diskriminasi lain dalam jaringan convolutional. Ini berisi beberapa blok convolutional. Setiap blok convolutional berisi lapisan convolutional, lapisan normalisasi batch, dan fungsi aktivasi, selain blok convolutional pertama, yang tidak memiliki lapisan normalisasi batch.



Gambar 9.52 Illustrasi Discriminator Network

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

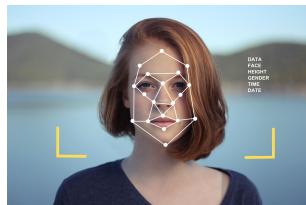
pre-trained Inception-ResNet-2 network, sekali disediakan dengan gambar, mengembalikan yang sesuai embedding. Tertanam yang diekstrak untuk gambar asli dan gambar direkonstruksi dapat dihitung dengan menghitung jarak Euclidean dari yang tertanam.



Gambar 9.53 Illustrasi Inception-ResNet-2 Model.

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

Tujuan utama dari jaringan pengenalan wajah adalah untuk mengenali identitas seseorang dalam gambar yang diberikan.



Gambar 9.54 Illustrasi Face recognition network Age-cGAN.

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

Age-cGAN memiliki beberapa tahapan pelatihan. Seperti disebutkan di bagian sebelumnya, Age-cGAN memiliki empat jaringan, yang dilatih dalam tiga tahap. Pelatihan AgecGAN terdiri dari tiga tahap:

- pelatihan GAN bersyarat: pada tahap ini, kita melatih jaringan Generator dan jaringan diskriminator.
- awal pendekatan vektor laten: pada tahap ini, kami melatih jaringan Encoder.
- optimasi vektor laten: pada tahap ini, kami mengoptimalkan kedua encoder dan jaringan generator.

9.5.1.9 Berikan contoh perhitungan fungsi training objektif

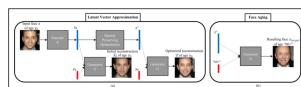
Objektif Trainning ialah untuk meminimalkan loss function sebagai log likelihood function yang diberikan pada persamaan dimana D melambangkan training data.

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

Gambar 9.55 Training Objektif

9.5.1.10 Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation

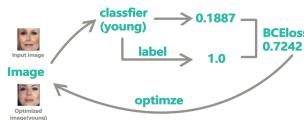
Perkiraan vektor laten awal adalah metode untuk memperkirakan vektor laten untuk mengoptimalkan rekonstruksi gambar wajah. Untuk memperkirakan vektor laten, kami memiliki jaringan Encoder. Kami melatih jaringan Encoder pada gambar yang dihasilkan dan gambar nyata. Setelah dilatih, Jaringan Encoder akan mulai menghasilkan vektor laten dari Distribusi. Tujuan pelatihan fungsi untuk pelatihan jaringan Encoder adalah kehilangan jarak Euclidean.



Gambar 9.56 Illustrasi Initial latent vector approximation

9.5.1.11 Berikan contoh perhitungan latent vector optimization

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



Gambar 9.57 Contoh Perhitungan Latent vector optimization

9.5.2 Praktek

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

```

1 from google.colab import drive
2 drive.mount('/content/drive')
3 # In[1]: Ekstrak File:
4 import tarfile
5 tf = tarfile.open("/content/drive/My Drive/zMachine Learning/
    wiki_crop.tar")
6 tf.extractall(path="/content/drive/My Drive/zMachine Learning/
    wiki_crop/")

```

Kode di atas akan melakukan mount dan extract dataset.

- Masuk ke google colab dengan akun google
- Klik mount google drive
- Lakukan proses untar menggunakan code yang di run google colab
- Selesai

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

```

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

```

```

5         return taken - birth.year
6     else:
7         return taken - birth.year - 1
8
9 #%%
10 def load_data(wiki_dir, dataset='wiki'):
11     # Load the wiki.mat file
12     meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset
13     )))
14
15     # Load the list of all files
16     full_path = meta[dataset][0, 0]["full_path"][0]
17
18     # List of Matlab serial date numbers
19     dob = meta[dataset][0, 0]["dob"][0]
20
21     # List of years when photo was taken
22     photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
23
24     # Calculate age for all dobs
25     age = [calculate_age(photo_taken[i], dob[i]) for i in range(
26         len(dob))]
27
28     # Create a list of tuples containing a pair of an image path
29     # and age
30     images = []
31     age_list = []
32     for index, image_path in enumerate(full_path):
33         images.append(image_path[0])
34         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.

```

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='same')(input_layer)
9     # enc = BatchNormalization()(enc)
10    enc = LeakyReLU(alpha=0.2)(enc)
11
12    # 2nd Convolutional Block
13    enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='same')(enc)
14    enc = BatchNormalization()(enc)
15    enc = LeakyReLU(alpha=0.2)(enc)
16
17    # 3rd Convolutional Block

```

```

18     enc = Conv2D(filters=128, kernel_size=5, strides=2, 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

```

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.

```

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

```

```

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

```

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

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

```

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

```

```

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

```

Diskriminator mencoba untuk membedakan antara gambar asli dan gambar palsu.

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

```

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

```

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

```
81     Train the discriminator network
82     """
83
84     # Generate fake images
85     initial_recon_images = generator.predict_on_batch(
86     ([z_noise, y_batch]))
87
88     d_loss_real = discriminator.train_on_batch([
89     images_batch, y_batch], real_labels)
90     d_loss_fake = discriminator.train_on_batch([
91     initial_recon_images, y_batch], fake_labels)
92
93     d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)
94     print("d_loss:{}".format(d_loss))
95
96     """
97     Train the generator network
98     """
99
100    z_noise2 = np.random.normal(0, 1, size=(batch_size, z_shape))
101    random_labels = np.random.randint(0, 6,
102        batch_size).reshape(-1, 1)
103    random_labels = to_categorical(random_labels, 6)
104
105    g_loss = adversarial_model.train_on_batch([
106        z_noise2, random_labels], [1] * batch_size)
107
108    print("g_loss:{}".format(g_loss))
109
110    gen_losses.append(g_loss)
111    dis_losses.append(d_loss)
112
113    # Write losses to Tensorboard
114    write_log(tensorboard, 'g_loss', np.mean(gen_losses),
115    epoch)
116    write_log(tensorboard, 'd_loss', np.mean(dis_losses),
117    epoch)
118
119    """
120    Generate images after every 10th epoch
121    """
122
123    if epoch % 10 == 0:
124        images_batch = loaded_images[0:batch_size]
125        images_batch = images_batch / 127.5 - 1.0
126        images_batch = images_batch.astype(np.float32)
127
128        y_batch = y[0:batch_size]
129        z_noise = np.random.normal(0, 1, size=(batch_size,
130            z_shape))
131
132        gen_images = generator.predict_on_batch([z_noise,
133            y_batch])
134
135        for i, img in enumerate(gen_images[:5]):
```

```

126         save_rgb_img(img, path="results/img_{:}_{}.png"
127         .format(epoch, i))
128
129     # Save networks
130     try:
131         generator.save_weights("generator.h5")
132         discriminator.save_weights("discriminator.h5")
133     except Exception as e:
134         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.

```

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=
15     np.int64)
16     num_classes = len(set(y))
17     y = np.reshape(np.array(y), [len(y), 1])
18     y = to_categorical(y, num_classes=num_classes)
19
20     for epoch in range(epochs):
21         print("Epoch:", epoch)
22
23         encoder_losses = []
24
25         number_of_batches = int(z_i.shape[0] / batch_size)
26         print("Number of batches:", number_of_batches)
27         for index in range(number_of_batches):
28             print("Batch:", index + 1)
29
30             z_batch = z_i[index * batch_size:(index + 1) *
31             batch_size]
32             y_batch = y[index * batch_size:(index + 1) *
33             batch_size]
34
35             generated_images = generator.predict_on_batch([
36             z_batch, y_batch])
37
38             # Train the encoder model

```

```

35     encoder_loss = encoder.train_on_batch(
36     generated_images, z_batch)
37     print("Encoder loss:", encoder_loss)
38
39     encoder_losses.append(encoder_loss)
40
41     # Write the encoder loss to Tensorboard
42     write_log(tensorboard, "encoder_loss", np.mean(
43     encoder_losses), epoch)
44
45     # Save the encoder model
46     encoder.save_weights("encoder.h5")
47
48 """
49 Optimize the encoder and the generator network
50 """
51 if TRAIN_GAN_WITH_FR:
52
53     # Load the encoder network
54     encoder = build_encoder()
55     encoder.load_weights("encoder.h5")
56
57     # Load the generator network
58     generator.load_weights("generator.h5")
59
60     image_resizer = build_image_resizer()
61     image_resizer.compile(loss=['binary_crossentropy'],
62     optimizer='adam')
63
64     # Face recognition model
65     fr_model = build_fr_model(input_shape=fr_image_shape)
66     fr_model.compile(loss=['binary_crossentropy'],
67     optimizer="adam")
68
69     # Make the face recognition network as non-trainable
70     fr_model.trainable = False
71
72     # Input layers
73     input_image = Input(shape=(64, 64, 3))
74     input_label = Input(shape=(6,))
75
76     # Use the encoder and the generator network
77     latent0 = encoder(input_image)
78     gen_images = generator([latent0, input_label])
79
80     # Resize images to the desired shape
81     resized_images = Lambda(lambda x: K.resize_images(
82     gen_images, height_factor=3, width_factor=3,
83     data_format='channels_last'))(gen_images)
84     embeddings = fr_model(resized_images)
85
86     # Create a Keras model and specify the inputs and outputs
87     # for the network
88     fr_adversarial_model = Model(inputs=[input_image,
89     input_label], outputs=[embeddings])

```

```
83     # Compile the model
84     fr_adversarial_model.compile(loss=euclidean_distance_loss
85     , 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.
104             predict_on_batch(images_batch)
105
106             real_embeddings = fr_model.predict_on_batch(
107             images_batch_resized)
108
109             reconstruction_loss = fr_adversarial_model.
110             train_on_batch([images_batch, y_batch], real_embeddings)
111
112             print("Reconstruction loss:", reconstruction_loss)
113
114             reconstruction_losses.append(reconstruction_loss)
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
126                 , z_shape))
127
128                 gen_images = generator.predict_on_batch([z_noise,
129                 y_batch])
```

```

128
129         for i, img in enumerate(gen_images[:5]):
130             save_rgb_img(img, path="results/img_opt_{}-{}.
131             {}.png".format(epoch, i))
132
133             # Save improved weights for both of the networks
134             generator.save_weights("generator_optimized.h5")
135             encoder.save_weights("encoder_optimized.h5")

```

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

9.5.3 Penanganan Error

9.5.3.1 Error

- OSError



Gambar 9.58 OSError

9.5.3.2 Solusi Error

- OSError

Pastikan folder telah ada

9.5.4 Bukti Tidak Plagiat



Gambar 9.59 Bukti tidak plagiat

9.5.5 Link Youtube

<https://bit.ly/KecerdasanBuatanDirga>

9.6 1174087 - Ilham Muhammad Ariq

9.6.1 Teori

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

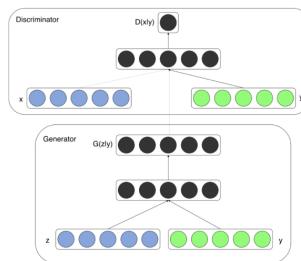
Vanilla GAN adalah tipe GAN paling sederhana. Di sini, Generator dan Diskriminatator adalah perceptron multi-layer sederhana. Dalam vanilla GAN, algoritma ini sangat sederhana, ia mencoba untuk mengoptimalkan persamaan matematika menggunakan keturunan gradien stokastik. CGAN (Conditional GAN), label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



Gambar 9.60 Valina GAN-cGAN

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

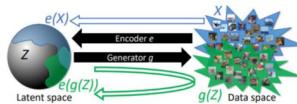
Age cGAN ialah dengan mengkondisikan model pada informasi tambahan dimungkinkan untuk mengarahkan proses pembuatan data. Pengkondisian semacam itu dapat didasarkan pada label kelas.



Gambar 9.61 Age-cGAN

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

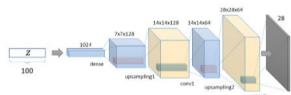
Arsitektur encoder biasanya digunakan untuk memodelkan struktur manifold dan membalikkan encoder untuk memproses data.



Gambar 9.62 Encoder Age cGANr

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

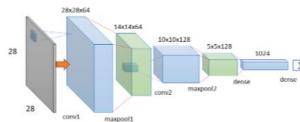
Arsitektur generator adalah sebuah array yang digunakan secara random, yang disebut seed. dari data seed tersebut, generator akan merubahnya menjadi sebuah gambar yang ukuran 28×28 dengan menggunakan Convolutional Neural Network.



Gambar 9.63 Network Age cGAN

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

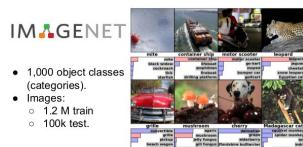
Arsitektur diskriminator adalah CNN yang dapat menerima input gambar yang berukuran $28,28$ serta menghasilkan angka biner yang menyatakan apakah data yang diinputkan merupakan dataset asli atau gambar dataset palsu.



Gambar 9.64 Discriminator Age cGAN

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

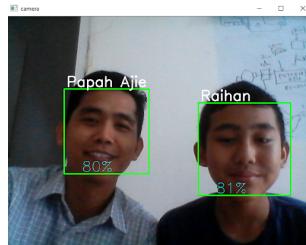
Pre-Trained Network atau Transfer Learning merupakan suatu metode penyelesaian yang memanfaatkan model yang sudah dilatih terhadap suatu dataset untuk menyelesaikan masalah dengan cara menggunakan sebagai starting point, memodifikasi dan mengupdate parameternya, sehingga sesuai dengan dataset yang baru.



Gambar 9.65 Pretrained Inception ResNet

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

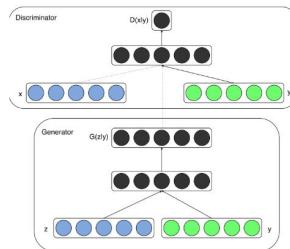
Face Recognition merupakan salah satu sistem yang mengimplementasi Deep Learning yang dapat mengenali wajah secara fisik dari gambar digital atau video frame.



Gambar 9.66 Face recognition network Age-cGAN

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

Pada dari Age-cGan ni terdapat 2 tahapan dengan generator dan diskriminator. dimana untuk tahap generator sendiri membutuhkan vektor laten 100 serta menghasilkan gambar yang realistik dari dimensinya. sedangkan tahap diskriminator itu tahapan dimana memprediksi gambar yang diberikan nyata atau palsu.



Gambar 9.67 Tahap Age cGAN

9. Berikan contoh perhitungan fungsi training objektif.

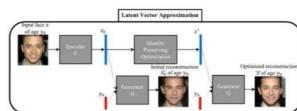
Objektif Trainning ialah untuk meminimalkan loss function sebagai log likelihood function yang diberikan pada persamaan dimana D melambangkan training data.

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

Gambar 9.68 Training Objektif

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

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



Gambar 9.69 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 9.70 Latent Vector Optimization

9.6.2 Praktek Program

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

```

1 from google.colab import drive
2 drive.mount('/content/drive')
3
4 import tarfile

```

```

5 tf = tarfile.open("/content/drive/My Drive/Colab Notebooks/
6 wiki_crop.tar")
tf.extractall(path="/content/drive/My Drive/Colab Notebooks/
Chapter9/data")

```

Kode di atas akan melakukan mount dan extract dataset.

- Login ke google colab menggunakan akun google
- Mount google drive
- Lakukan proses unzip melalui notebook python di google colab, unzip pakai codingan
- Selesai

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

```

1 def 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
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
10    dob = meta[dataset][0, 0]["dob"][0]
11
12    # List of years when photo was taken
13    photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
14
15    # Calculate age for all dobs
16    age = [calculate_age(photo_taken[i], dob[i]) for i in range(
17        len(dob))]
18
19    # Create a list of tuples containing a pair of an image path
20    # and age
21    images = []
22    age_list = []
23    for index, image_path in enumerate(full_path):
24        images.append(image_path[0])
25        age_list.append(age[index])

```

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.

```

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='same')(input_layer)
9     # enc = BatchNormalization()(enc)
10    enc = LeakyReLU(alpha=0.2)(enc)
11
12    # 2nd Convolutional Block
13    enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='same')(enc)
14    enc = BatchNormalization()(enc)
15    enc = LeakyReLU(alpha=0.2)(enc)
16
17    # 3rd Convolutional Block
18    enc = Conv2D(filters=128, kernel_size=5, strides=2, 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

```

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.

```

1 def build_generator():
2     """
3         Create a Generator Model with hyperparameters values defined
4             as follows
5     """
6     latent_dims = 100

```

```

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

```

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

```

```

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

```

Diskriminator mencoba untuk membedakan antara gambar asli dan gambar palsu.

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

```

1 if __name__ == '__main__':
2     # Define hyperparameters
3     data_dir = "data"
4     wiki_dir = os.path.join(data_dir, "wiki_crop1")
5     epochs = 500
6     batch_size = 2
7     image_shape = (64, 64, 3)
8     z_shape = 100
9     TRAIN_GAN = True
10    TRAIN_ENCODER = False
11    TRAIN_GAN_WITH_FR = False
12    fr_image_shape = (192, 192, 3)
13
14    # Define optimizers
15    dis_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
16                        epsilon=10e-8)
17    gen_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
18                        epsilon=10e-8)
19    adversarial_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=
20                                0.999, epsilon=10e-8)
21
22    """
23        Build and compile networks
24    """
25    # Build and compile the discriminator network
26    discriminator = build_discriminator()

```

```
24 discriminator.compile(loss=['binary_crossentropy'], optimizer=dis_optimizer)
25
26 # Build and compile the generator network
27 generator = build_generator()
28 generator.compile(loss=['binary_crossentropy'], optimizer=gen_optimizer)
29
30 # Build and compile the adversarial model
31 discriminator.trainable = False
32 input_z_noise = Input(shape=(100,))
33 input_label = Input(shape=(6,))
34 recons_images = generator([input_z_noise, input_label])
35 valid = discriminator([recons_images, input_label])
36 adversarial_model = Model(inputs=[input_z_noise, input_label],
37                             outputs=[valid])
38 adversarial_model.compile(loss=['binary_crossentropy'],
39                            optimizer=gen_optimizer)
40
41 tensorboard = TensorBoard(log_dir="logs/{}".format(time.time()))
42 tensorboard.set_model(generator)
43 tensorboard.set_model(discriminator)
44 """
45 Load the dataset
46 """
47 images, age_list = load_data(wiki_dir=wiki_dir, dataset="wiki")
48 age_cat = age_to_category(age_list)
49 final_age_cat = np.reshape(np.array(age_cat), [len(age_cat), 1])
50 classes = len(set(age_cat))
51 y = to_categorical(final_age_cat, num_classes=len(set(age_cat)))
52
53 loaded_images = load_images(wiki_dir, images, (image_shape[0], image_shape[1]))
54
55 # Implement label smoothing
56 real_labels = np.ones((batch_size, 1), dtype=np.float32) * 0.9
57 fake_labels = np.zeros((batch_size, 1), dtype=np.float32) * 0.1
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 = []
```

```
68     number_of_batches = int(len(loader_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 = loader_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)  
112  
113        print("g_loss:{}".format(g_loss))  
114  
115        gen_losses.append(g_loss)  
116        dis_losses.append(d_loss)  
117  
118        # Write losses to Tensorboard  
119        write_log(tensorboard, 'g_loss', np.mean(gen_losses),  
120            epoch)  
121        write_log(tensorboard, 'd_loss', np.mean(dis_losses),  
122            epoch)
```

```

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,
122         z_shape))
123
124     gen_images = generator.predict_on_batch([z_noise,
125         y_batch])
126
127     for i, img in enumerate(gen_images[:5]):
128         save_rgb_img(img, path="results/img-{}-{}.png"
129             .format(epoch, i))
130
131     # Save networks
132     try:
133         generator.save_weights("generator.h5")
134         discriminator.save_weights("discriminator.h5")
135     except Exception as e:
136         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.

```

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) *
30                     batch_size]
30         y_batch = y[index * batch_size:(index + 1) *
31                     batch_size]
32
33         generated_images = generator.predict_on_batch([
34             z_batch, y_batch])
35
36         # Train the encoder model
37         encoder_loss = encoder.train_on_batch(
38             generated_images, z_batch)
39         print("Encoder loss:", encoder_loss)
40
41         encoder_losses.append(encoder_loss)
42
43         # Write the encoder loss to Tensorboard
44         write_log(tensorboard, "encoder_loss", np.mean(
45             encoder_losses), epoch)
46
47         # Save the encoder model
48         encoder.save_weights("encoder.h5")
49
50 """
51
52     Optimize the encoder and the generator network
53 """
54
55 if TRAIN_GAN_WITH_FR:
56
57     # Load the encoder network
58     encoder = build_encoder()
59     encoder.load_weights("encoder.h5")
60
61     # Load the generator network
62     generator.load_weights("generator.h5")
63
64     image_resizer = build_image_resizer()
65     image_resizer.compile(loss=['binary_crossentropy'],
66                           optimizer='adam')
67
68     # Face recognition model
69     fr_model = build_fr_model(input_shape=fr_image_shape)
70     fr_model.compile(loss=['binary_crossentropy'], optimizer=
71                       "adam")
72
73     # Make the face recognition network as non-trainable
74     fr_model.trainable = False
75
76     # Input layers
77     input_image = Input(shape=(64, 64, 3))
78     input_label = Input(shape=(6,))
```

```
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(
78     gen_images, height_factor=3, width_factor=3,
79     data_format='channels_last'))(gen_images)
80 embeddings = fr_model(resized_images)
81
82 # Create a Keras model and specify the inputs and outputs
83 # for the network
84 fr_adversarial_model = Model(inputs=[input_image,
85     input_label], outputs=[embeddings])
86
87 # Compile the model
88 fr_adversarial_model.compile(loss=euclidean_distance_loss,
89     optimizer=adversarial_optimizer)
90
91 for epoch in range(epochs):
92     print("Epoch:", epoch)
93
94     reconstruction_losses = []
95
96     number_of_batches = int(len.loaded_images) / batch_size
97     print("Number of batches:", number_of_batches)
98     for index in range(number_of_batches):
99         print("Batch:", index + 1)
100
101         images_batch = loaded_images[index * batch_size:(index + 1) * batch_size]
102         images_batch = images_batch / 127.5 - 1.0
103         images_batch = images_batch.astype(np.float32)
104
105         y_batch = y[index * batch_size:(index + 1) * batch_size]
106
107         images_batch_resized = image_resizer.
108         predict_on_batch(images_batch)
109
110         real_embeddings = fr_model.predict_on_batch(
111             images_batch_resized)
112
113         reconstruction_loss = fr_adversarial_model.
114         train_on_batch([images_batch, y_batch], real_embeddings)
115
116         print("Reconstruction loss:", reconstruction_loss)
117
118         reconstruction_losses.append(reconstruction_loss)
119
120 # Write the reconstruction loss to Tensorboard
121 write_log(tensorboard, "reconstruction_loss", np.mean(
122     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,
126                                      z_shape))
127
128          gen_images = generator.predict_on_batch([z_noise,
129                                                    y_batch])
130
131          for i, img in enumerate(gen_images[:5]):
132              save_rgb_img(img, path="results/img_opt_{}-{}.png".format(epoch, i))
133
134      # Save improved weights for both of the networks
135      generator.save_weights("generator_optimized.h5")
136      encoder.save_weights("encoder_optimized.h5")

```

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

9.6.3 Penanganan Error

1. FileNotFoundError

FileNotFoundError: [Errno 2] No such file or directory: '/content/drive/My Drive/GAN/Modelos/Modelo1/Modelo1.h5'

Gambar 9.71 FileNotFoundError

2. Cara Penanganan Error

- FileNotFoundError

Error tersebut karena salah penempatan file.

9.6.4 Bukti Tidak Plagiat



Gambar 9.72 Bukti Tidak Melakukan Plagiat Chapter 9

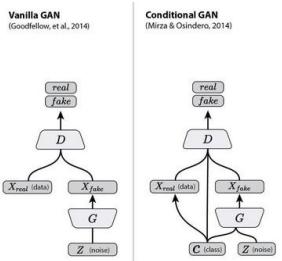
9.7 1174067 - Kaka Kamaludin

9.7.1 Teori

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

Vanilla GANs biasanya tidak memiliki convolutional Neural Jaringan (CNNs) di jaringan mereka. Conditional GANs (cGANs) adalah perpanjangan dari model GAN. Mereka memungkinkan untuk generasi gambar yang memiliki kondisi tertentu atau atribut dan telah terbukti menjadi lebih baik dari Vanilla GANs sebagai hasilnya.

cGANs adalah jenis GAN yang dikondisikan pada beberapa informasi tambahan. informasi tambahan y ke Generator sebagai lapisan input tambahan. Dalam Vanilla GANs, tidak ada kontrol atas Kategori gambar yang dihasilkan. Ketika kita menambahkan kondisi y ke Generator, kita dapat menghasilkan gambar dari kategori tertentu, menggunakan y, yang mungkin jenis data, seperti label kelas atau data integer. Vanilla GANs bisa belajar hanya satu kategori dan sangat sulit untuk arsitek GANs untuk beberapa kategori. Sebuah cGAN, bagaimanapun, dapat digunakan untuk menghasilkan model multi-modal dengan kondisi yang berbeda untuk kategori yang berbeda.

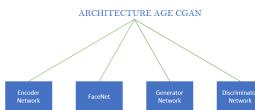


Gambar 9.73 Illustrasi Vanilla GAN dan cGAN

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

Arsitektur cGAN untuk penuaan wajah sedikit lebih rumit. AgecGan terdiri dari empat jaringan: Encoder, FaceNet, Jaringan Generator, dan jaringan diskriminat. Dengan Encoder, kita belajar pemetaan invers gambar wajah masukan dan kondisi usia dengan vektor laten. FaceNet adalah jaringan pengenalan wajah yang mempelajari perbedaan antara gambar input x dan gambar yang direkonstruksi. Kami memiliki jaringan Generator, yang mengambil representasi tersembunyi yang terdiri dari gambar wajah dan vektor kondisi dan menghasilkan gambar. Jaringan diskriminat adalah untuk mendiskriminasikan an-

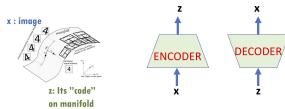
tara gambar nyata dan gambar palsu. Masalah dengan cGANs adalah bahwa mereka tidak dapat mempelajari tugas pemetaan terbalik masukan gambar x dengan atribut y ke vektor laten z . Solusi untuk masalah ini adalah dengan menggunakan jaringan Encoder. Kita dapat melatih jaringan encoder untuk memperkirakan pemetaan terbalik dari input Images x .



Gambar 9.74 Illustrasi Arsitektur cGAN

3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari AgecGAN.

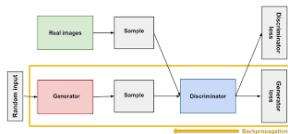
Tujuan utama dari jaringan Encoder adalah untuk menghasilkan vektor laten dari gambar yang disediakan. Pada dasarnya, dibutuhkan gambar dimensi (64, 64, 3) dan mengubahnya menjadi vektor 100-dimensi. Jaringan Encoder adalah jaringan syaraf convolutional yang dalam. Jaringan berisi empat convolutional blok dan dua lapisan padat. Setiap blok convolutional berisi lapisan convolutional, lapisan normalisasi batch, dan fungsi aktivasi. Di setiap blok convolutional, setiap lapisan convolutional diikuti oleh lapisan normalisasi batch, kecuali lapisan convolutional pertama.



Gambar 9.75 Illustrasi Network Encoder

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

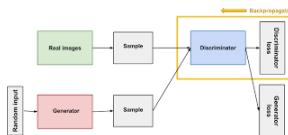
Tujuan utama dari generator adalah untuk menghasilkan gambar dari dimensi (64, 64, 3). Dibutuhkan vektor laten 100 dimensi dan beberapa informasi tambahan, y , dan mencoba untuk menghasilkan gambar yang realistik. Jaringan Generator adalah jaringan neural yang mendalam convolutional juga. Hal ini terdiri dari lapisan padat, upsampling, dan convolutional. Dibutuhkan dua nilai input: vektor kebisingan dan nilai pengkondision. Nilai pengkondision adalah informasi tambahan yang diberikan ke jaringan. Untuk Age-cGAN, ini akan menjadi usia.



Gambar 9.76 Illustrasi Network Generator

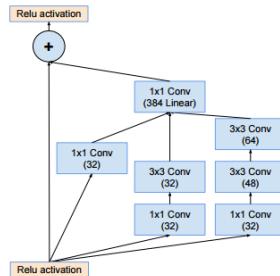
Jelaskan dengan ilustrasi gambar sendiri arsitektur discriminator network dari Age-cGAN.

Tujuan utama dari jaringan diskriminator adalah untuk mengidentifikasi apakah gambar yang disediakan adalah palsu atau nyata. Hal ini dilakukan dengan melewati gambar melalui serangkaian lapisan sampling bawah dan beberapa lapisan klasifikasi. Dengan kata lain, ini memprediksi Apakah gambar itu nyata atau palsu. Seperti jaringan lain, Jaringan diskriminatir lain dalam jaringan convolutional. Ini berisi beberapa blok convolutional. Setiap blok convolutional berisi lapisan convolutional, lapisan normalisasi batch, dan fungsi aktivasi, selain blok convolutional pertama, yang tidak memiliki lapisan normalisasi batch.



Gambar 9.77 Illustrasi Discriminator Network

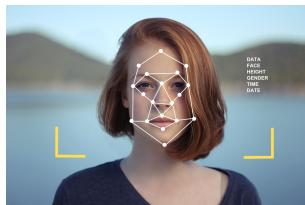
5. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model. pre-trained Inception-ResNet-2 network, sekali disediakan dengan gambar, mengembalikan yang sesuai embedding. Tertanam yang diekstrak untuk gambar asli dan gambar direkonstruksi dapat dihitung dengan menghitung jarak Euclidean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



Gambar 9.78 Illustrasi Inception-ResNet-2 Model.

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

Tujuan utama dari jaringan pengenalan wajah adalah untuk mengenali identitas seseorang dalam gambar yang diberikan.



Gambar 9.79 Illustrasi Face recognition network Age-cGAN.

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

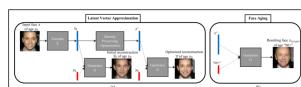
Age-cGAN memiliki beberapa tahapan pelatihan. Seperti disebutkan di bagian sebelumnya, Age-cGAN memiliki empat jaringan, yang dilatih dalam tiga tahap. Pelatihan AgecGAN terdiri dari tiga tahap:

- pelatihan GAN bersyarat: pada tahap ini, kita melatih jaringan Generator dan jaringan diskriminator.
- awal pendekatan vektor laten: pada tahap ini, kami melatih jaringan Encoder.
- optimasi vektor laten: pada tahap ini, kami mengoptimalkan kedua encoder dan jaringan generator.

8. Berikan contoh perhitungan fungsi training objektif

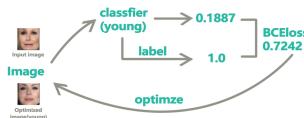
9. Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation

Perkiraan vektor laten awal adalah metode untuk memperkirakan vektor laten untuk mengoptimalkan rekonstruksi gambar wajah. Untuk memperkirakan vektor laten, kami memiliki jaringan Encoder. Kami melatih jaringan Encoder pada gambar yang dihasilkan dan gambar nyata. Setelah dilatih, Jaringan Encoder akan mulai menghasilkan vektor laten dari Distribusi. Tujuan pelatihan fungsi untuk pelatihan jaringan Encoder adalah kehilangan jarak Euclidean.



Gambar 9.80 Illustrasi Initial latent vector approximation

10. Berikan contoh perhitungan latent vector optimization



Gambar 9.81 Contoh Perhitungan Latent vector optimization

9.7.2 Praktek

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

```

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

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

```

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

```
23     age_list.append(age[index])
24
25 # Return a list of all images and respective age
26     return images, age_list
```

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

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

4. Jelaskan bagaimana kode program The Generator Network bekerja dijelaskan dengan bahawa awam dengan ilustrasi sederhana. Proses Generator agar bekerja dengan baik dibutuhkan representasi dari gambar wajah dan vector kondisi sebagai inputan yang menghasilkan sebuah gambar.

```

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

```

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

```

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

```

```

13     label_input1 = Lambda(expand_label_input)(label_input)
14     x = concatenate([x, label_input1], axis=3)
15
16
17     x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
18     x = BatchNormalization()(x)
19     x = LeakyReLU(alpha=0.2)(x)
20
21     x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
22     x = BatchNormalization()(x)
23     x = LeakyReLU(alpha=0.2)(x)
24
25     x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
26     x = BatchNormalization()(x)
27     x = LeakyReLU(alpha=0.2)(x)
28
29     x = Flatten()(x)
30     x = Dense(1, activation='sigmoid')(x)
31
32 model = Model(inputs=[image_input, label_input], outputs=[x])
33

```

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

```

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

```

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

```

1 # In [7. Laten Vector]:
2     .....

```

```
3     Train encoder
4
5
6 if TRAIN_ENCODER:
7     # Build and compile encoder
8     encoder = build_encoder()
9     encoder.compile(loss=euclidean_distance_loss, optimizer='adam')
10
11    # Load the generator network's weights
12    try:
13        generator.load_weights("generator.h5")
14    except Exception as e:
15        print("Error:", e)
16
17    z_i = np.random.normal(0, 1, size=(5000, z_shape))
18
19    y = np.random.randint(low=0, high=6, size=(5000,), dtype=
np.int64)
20    num_classes = len(set(y))
21    y = np.reshape(np.array(y), [len(y), 1])
22    y = to_categorical(y, num_classes=num_classes)
23
24    for epoch in range(epochs):
25        print("Epoch:", epoch)
26
27        encoder_losses = []
28
29        number_of_batches = int(z_i.shape[0] / batch_size)
30        print("Number of batches:", number_of_batches)
31        for index in range(number_of_batches):
32            print("Batch:", index + 1)
33
34            z_batch = z_i[index * batch_size:(index + 1) *
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(
41                generated_images, z_batch)
42            print("Encoder loss:", encoder_loss)
43
44            encoder_losses.append(encoder_loss)
45
46            # Write the encoder loss to Tensorboard
47            write_log(tensorboard, "encoder_loss", np.mean(
48                encoder_losses), epoch)
49
50            # Save the encoder model
51            encoder.save_weights("encoder.h5")
```

9.7.3 Link Youtube

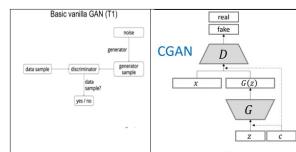
<https://www.youtube.com/playlist?list=PL4dhp4u89PHbhX9jrGyM3N12gmwhY3uIe>

9.8 1174069 - Fanny Shafira Damayanti

9.8.1 Teori

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

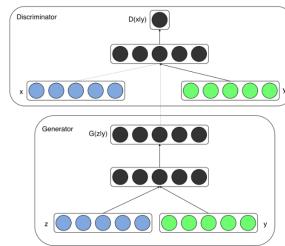
Vanilla GAN adalah tipe GAN paling sederhana. Di sini, Generator dan Diskriminatator adalah perceptron multi-layer sederhana. Dalam vanilla GAN, algoritma ini sangat sederhana, ia mencoba untuk mengoptimalkan persamaan matematika menggunakan keturunan gradien stokastik. CGAN (Conditional GAN), label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



Gambar 9.82 Valina GAN-cGAN

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

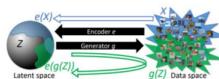
Age cGAN ialah dengan mengkondisikan model pada informasi tambahan dimungkinkan untuk mengarahkan proses pembuatan data. Pengkondisian semacam itu dapat didasarkan pada label kelas.



Gambar 9.83 Age-cGAN

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

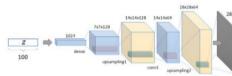
Arsitektur encoder biasanya digunakan untuk memodelkan struktur manifold dan membalikkan encoder untuk memproses data.



Gambar 9.84 Encoder Age cGANr

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

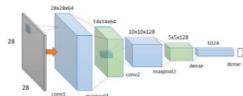
Arsitektur generator adalah sebuah array yang digunakan secara random, yang disebut seed. dari data seed tersebut, generator akan merubahnya menjadi sebuah gambar yang ukuran 28 x 28 dengan menggunakan Convolutional Neural Network.



Gambar 9.85 Network Age cGAN

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

Arsitektur diskriminatator 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 9.86 Discriminator Age cGAN

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

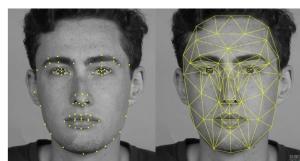
Pre-Trained Network atau Transfer Learning merupakan suatu metode penyelesaian yang memanfaatkan model yang sudah dilatih terhadap suatu dataset untuk menyelesaikan masalah dengan cara menggunakan sebagai starting point, memodifikasi dan mengupdate parameternya, sehingga sesuai dengan dataset yang baru.



Gambar 9.87 Pretrained Inception ResNet

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

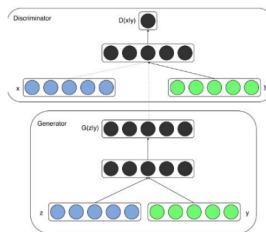
Face Recognition merupakan salah satu sistem yang mengimplementasi Deep Learning yang dapat mengenali wajah secara fisik dari gambar digital atau video frame.



Gambar 9.88 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 realistik dari dimensinya. sedangkan tahap diskriminator itu tahapan dimana memprediksi gambar yang diberikan nyata atau palsu.



Gambar 9.89 Tahap Age cGAN

9. Berikan contoh perhitungan fungsi training objektif.

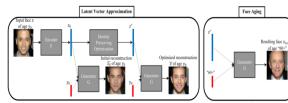
Objektif Training ialah untuk meminimalkan loss function sebagai log likelihood function yang diberikan pada persamaan dimana D melambangkan training data.

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

Gambar 9.90 Training Objektif

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

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

**Gambar 9.91** 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 z_i dalam ruang laten z .

**Gambar 9.92** Latent Vector Optimization

9.8.2 Praktek

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

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

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

```

1 # In[2]. Load Data]:
2 def load_data(wiki_dir, dataset='wiki'):
3     # Load the wiki.mat file
4     meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset
5         )))
6
7     # Load the list of all files
8     full_path = meta[dataset][0, 0]["full_path"][0]
9
10    # List of Matlab serial date numbers
11    dob = meta[dataset][0, 0]["dob"][0]
12
13    # List of years when photo was taken
14    photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
15
16    # Calculate age for all dobs
17    age = [calculate_age(photo_taken[i], dob[i]) for i in range(
18        len(dob))]
19
20    # Create a list of tuples containing a pair of an image path
21    # and age
22    images = []
23    age_list = []
24    for index, image_path in enumerate(full_path):
25        images.append(image_path[0])
26        age_list.append(age[index])
27
28    # Return a list of all images and respective age
29    return images, age_list
30

```

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

```

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

```

```

19 enc = Conv2D(filters=128, kernel_size=5, strides=2, padding='same')(enc)
20 enc = BatchNormalization()(enc)
21 enc = LeakyReLU(alpha=0.2)(enc)
22
23 # 4th Convolutional Block
24 enc = Conv2D(filters=256, kernel_size=5, strides=2, padding='same')(enc)
25 enc = BatchNormalization()(enc)
26 enc = LeakyReLU(alpha=0.2)(enc)
27
28 # Flatten layer
29 enc = Flatten()(enc)
30
31 # 1st Fully Connected Layer
32 enc = Dense(4096)(enc)
33 enc = BatchNormalization()(enc)
34 enc = LeakyReLU(alpha=0.2)(enc)
35
36 # Second Fully Connected Layer
37 enc = Dense(100)(enc)
38
39 # Create a model
40 model = Model(inputs=[input_layer], outputs=[enc])
41 return model

```

4. Jelaskan bagaimana kode program The Generator Network bekerja dengan ilustrasi sederhana. Proses Generator agar bekerja dengan baik dibutuhkan representasi dari gambar wajah dan vector kondisi sebagai inputan yang menghasilkan sebuah gambar.

```

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

```

```

25     x = UpSampling2D(size=(2, 2))(x)
26     x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
27     x = BatchNormalization(momentum=0.8)(x)
28     x = LeakyReLU(alpha=0.2)(x)
29
30     x = UpSampling2D(size=(2, 2))(x)

```

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

```

1 # In [5. Discriminator Network Bekerja ]:
2 def build_discriminator():
3     """
4         Create a Discriminator Model with hyperparameters values
5             defined as follows
6         """
7
8     input_shape = (64, 64, 3)
9     label_shape = (6,)
10    image_input = Input(shape=input_shape)
11    label_input = Input(shape=label_shape)
12
13    x = Conv2D(64, kernel_size=3, strides=2, padding='same')(image_input)
14    x = LeakyReLU(alpha=0.2)(x)
15
16    label_input1 = Lambda(expand_label_input)(label_input)
17    x = concatenate([x, label_input1], axis=3)
18
19    x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
20    x = BatchNormalization()(x)
21    x = LeakyReLU(alpha=0.2)(x)
22
23    x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
24    x = BatchNormalization()(x)
25    x = LeakyReLU(alpha=0.2)(x)
26
27    x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
28    x = BatchNormalization()(x)
29    x = LeakyReLU(alpha=0.2)(x)
30
31    x = Flatten()(x)
32    x = Dense(1, activation='sigmoid')(x)
33
34    model = Model(inputs=[image_input, label_input], outputs=[x])
35    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.

```

1 # In [6. Training cGAN]:
2     if __name__ == '__main__':
3         # Define hyperparameters
4         data_dir = "data"

```

```

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

```

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

```

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

```

```

32             print("Batch:", index + 1)
33
34             z_batch = z_i[index * batch_size:(index + 1) *
batch_size]
35             y_batch = y[index * batch_size:(index + 1) *
batch_size]
36
37             generated_images = generator.predict_on_batch([
z_batch, y_batch])
38
39             # Train the encoder model
40             encoder_loss = encoder.train_on_batch(
generated_images, z_batch)
41             print("Encoder loss:", encoder_loss)
42
43             encoder_losses.append(encoder_loss)
44
45             # Write the encoder loss to Tensorboard
46             write_log(tensorboard, "encoder_loss", np.mean(
encoder_losses), epoch)
47
48             # Save the encoder model
49             encoder.save_weights("encoder.h5")

```

9.8.3 Penanganan Error

9.8.4 Bukti Tidak Plagiat



Gambar 9.93 Bukti Tidak Melakukan Plagiat Chapter 9

9.9 1174070 - Arrizal Furqona Gifary

9.9.1 Teori

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

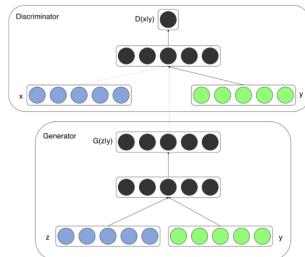
Vanilla GAN Vanilla GAN adalah tipe GAN paling sederhana. Di sini, Generator dan Diskriminator adalah perceptron multi-layer sederhana. Dalam vanilla GAN, algoritma ini sangat sederhana, ia mencoba untuk mengoptimalkan persamaan matematika menggunakan keturunan gradien stokastik. CGAN (Conditional GAN),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



Gambar 9.94 Valina GAN-cGAN

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

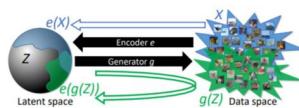
Age cGAN ialah dengan mengkondisikan model pada informasi tambahan dimungkinkan untuk mengarahkan proses pembuatan data. Pengkondisian semacam itu dapat didasarkan pada label kelas.



Gambar 9.95 Age-cGAN

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

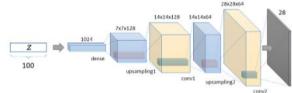
Arsitektur encoder biasanya digunakan untuk memodelkan struktur manifold dan membalikkan encoder untuk memproses data.



Gambar 9.96 Encoder Age cGANr

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

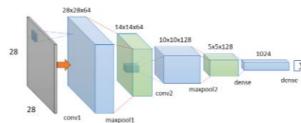
Arsitektur generator adalah sebuah array yang digunakan secara random, yang disebut seed. dari data seed tersebut, generator akan merubahnya menjadi sebuah gambar yang ukuran 28 x 28 dengan menggunakan Convolutional Neural Network.



Gambar 9.97 Network Age cGAN

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

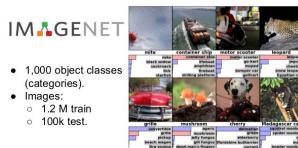
Arsitektur diskriminator adalah CNN yang dapat menerima input gambar yang berukuran 28,28 serta menghasilkan angka biner yang menyatakan apakah data yang diinputkan merupakan dataset asli atau gambar dataset palsu.



Gambar 9.98 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 9.99 Pretrained Inception ResNet

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

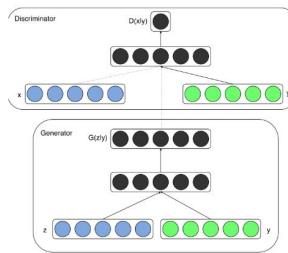
Face Recognition merupakan salah satu sistem yang mengimplementasi Deep Learning yang dapat mengenali wajah secara fisik dari gambar digital atau video frame.



Gambar 9.100 Face recognition network Age-cGAN

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

Pada dari Age-cGan ni terdapat 2 tahapan dengan generator dan diskriminatator. dimana untuk tahap generator sendiri membutuhkan vektor laten 100 serta menghasilkan gambar yang realistik dari dimensinya. sedangkan tahap diskriminatator itu tahapan dimana memprediksi gambar yang diberikan nyata atau palsu.



Gambar 9.101 Tahap Age cGAN

9. Berikan contoh perhitungan fungsi training objektif.

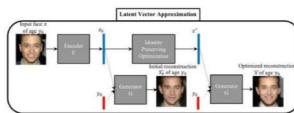
Objektif Training ialah untuk meminimalkan loss function sebagai log likelihood function yang diberikan pada persamaan dimana D melambangkan training data.

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

Gambar 9.102 Training Objektif

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

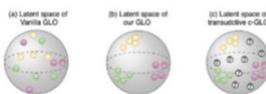
Latent vector appromidation kemampuan untuk membuat gamar yang realistik dan tajam serta menghasilkan gambar wajah pada usia target.



Gambar 9.103 Initial Latent Vector Approximation

- Berikan contoh perhitungan latent vector optimization.

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



Gambar 9.104 Latent Vector Optimization

9.9.2 Praktek

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

```

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

```

- Jelaskan bagaimana kode program bekerja untuk melakukan load terhadap dataset yang sudah di ekstrak, termasuk bagaimana penjelasan kode program perhitungan usia. Dibawah ini merupakan code untuk melakukan fungsi perhitungan usia.

```

1 # In[2. Load Data]:
2 def load_data(wiki_dir, dataset='wiki'):
3     # Load the wiki.mat file
4     meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset
5         )))
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
10    dob = meta[dataset][0, 0]["dob"][0]
11
12    # List of years when photo was taken

```

```

13 photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
14
15 # Calculate age for all dobs
16 age = [calculate_age(photo_taken[i], dob[i]) for i in range(
17 len(dob))]
18
19 # Create a list of tuples containing a pair of an image path
20 and age
21 images = []
22 age_list = []
23 for index, image_path in enumerate(full_path):
24     images.append(image_path[0])
25     age_list.append(age[index])
26
27 # Return a list of all images and respective age
28 return images, age_list

```

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

```

1 # In [3]: Encoder Bekerja]:
2 def build_encoder():
3     """
4         Encoder Network
5     """
6     input_layer = Input(shape=(64, 64, 3))
7
8     # 1st Convolutional Block
9     enc = Conv2D(filters=32, kernel_size=5, strides=2, padding='same')(input_layer)
10    # enc = BatchNormalization()(enc)
11    enc = LeakyReLU(alpha=0.2)(enc)
12
13    # 2nd Convolutional Block
14    enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='same')(enc)
15    enc = BatchNormalization()(enc)
16    enc = LeakyReLU(alpha=0.2)(enc)
17
18    # 3rd Convolutional Block
19    enc = Conv2D(filters=128, kernel_size=5, strides=2, padding='same')(enc)
20    enc = BatchNormalization()(enc)
21    enc = LeakyReLU(alpha=0.2)(enc)
22
23    # 4th Convolutional Block
24    enc = Conv2D(filters=256, kernel_size=5, strides=2, padding='same')(enc)
25    enc = BatchNormalization()(enc)
26    enc = LeakyReLU(alpha=0.2)(enc)
27
28    # Flatten layer
29    enc = Flatten()(enc)
30

```

```

31 # 1st Fully Connected Layer
32 enc = Dense(4096)(enc)
33 enc = BatchNormalization()(enc)
34 enc = LeakyReLU(alpha=0.2)(enc)
35
36 # Second Fully Connected Layer
37 enc = Dense(100)(enc)
38
39 # Create a model
40 model = Model(inputs=[input_layer], outputs=[enc])
41 return model

```

4. Jelaskan bagaimana kode program The Generator Network bekerja dengan ilustrasi sederhana. Proses Generator agar bekerja dengan baik dibutuhkan representasi dari gambar wajah dan vector kondisi sebagai inputan yang menghasilkan sebuah gambar.

```

1 # In [4. Generator Network Bekerja]:
2 def build_generator():
3     """
4         Create a Generator Model with hyperparameters values defined
5             as follows
6         """
7     latent_dims = 100
8     num_classes = 6
9
10    input_z_noise = Input(shape=(latent_dims,))
11    input_label = Input(shape=(num_classes,))
12
13    x = concatenate([input_z_noise, input_label])
14
15    x = Dense(2048, input_dim=latent_dims + num_classes)(x)
16    x = LeakyReLU(alpha=0.2)(x)
17    x = Dropout(0.2)(x)
18
19    x = Dense(256 * 8 * 8)(x)
20    x = BatchNormalization()(x)
21    x = LeakyReLU(alpha=0.2)(x)
22    x = Dropout(0.2)(x)
23
24    x = Reshape((8, 8, 256))(x)
25
26    x = UpSampling2D(size=(2, 2))(x)
27    x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
28    x = BatchNormalization(momentum=0.8)(x)
29    x = LeakyReLU(alpha=0.2)(x)
30
31    x = UpSampling2D(size=(2, 2))(x)

```

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

```

1 # In [5. Discriminator Network Bekerja]:
2 def build_discriminator():

```

```

3      """
4      Create a Discriminator Model with hyperparameters values
5      defined as follows
6      """
7      input_shape = (64, 64, 3)
8      label_shape = (6,)
9      image_input = Input(shape=input_shape)
10     label_input = Input(shape=label_shape)

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

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

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

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

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

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

27     model = Model(inputs=[image_input, label_input], outputs=[x])
28
29     return model
30
31
32
33

```

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

```

1 # In[6. Training cGAN]:
2     if __name__ == '__main__':
3         # Define hyperparameters
4         data_dir = "data"
5         wiki_dir = os.path.join(data_dir, "wiki_crop1")
6         epochs = 500
7         batch_size = 2
8         image_shape = (64, 64, 3)
9         z_shape = 100
10        TRAIN_GAN = True
11        TRAIN_ENCODER = False
12        TRAIN_GAN_WITH_FR = False
13        fr_image_shape = (192, 192, 3)

14
15        # Define optimizers
16        dis_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
17                           epsilon=10e-8)

```

```

17     gen_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
18                           epsilon=10e-8)
        adversarial_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2
                           =0.999, epsilon=10e-8)

```

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

```

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

```

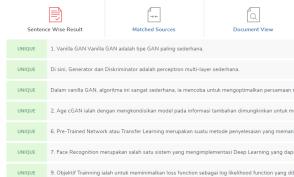
```

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

```

9.9.3 Penanganan Error

9.9.4 Bukti Tidak Plagiat



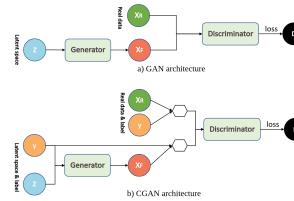
Gambar 9.105 Bukti Tidak Melakukan Plagiat Chapter 9

9.10 1174054 - Aulyardha Anindita

9.10.1 Teori

- 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.

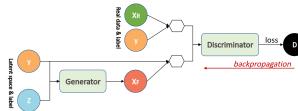


Gambar 9.106 Vanilla GAN dan cGAN

- Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN

Pada Arsitektur Age-CGAN terdapat 4 bagian, yaitu : encoder, faceNet, gener-

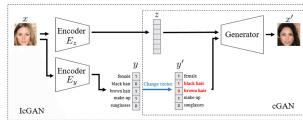
ator dan discriminator. untuk lebih jelasnya bisa dilihat pada ilustrasi gambar dibawah ini.



Gambar 9.107 Arsitektur Age-cGAN

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

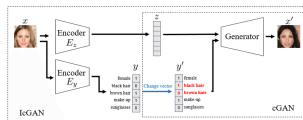
Encoder mempelajari pemetaan terbalik dari gambar wajah input dan kondisi usia dengan vektor laten Z. jaringan encoder menghasilkan vektor laten dari gambar input.jaringan encoder adalah CNN yang mengambil gambar dari dimensi (64,64,3) dan mengubahnya menjadi vektor 100 dimensi. ada empat blok konvolusional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



Gambar 9.108 Arsitektur Encoder Network dari Age-cGAN

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

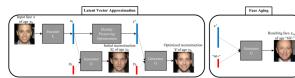
Pada generator dibutuhkan representasi tersembunyi dari gambar wajah dan vektor kondisi sebagai input dan menghasilkan gambar. generator adalah CNN dan dibutuhkan vektor laten 100 dimensi dan vektor kondisi y, dan mencoba menghasilkan gambar realistik dari dimensi (64,64,3). generator memiliki lapisan padat, membingungka dan konvolutif. 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 9.109 Arsitektur Generator Network dari Age-cGAN

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

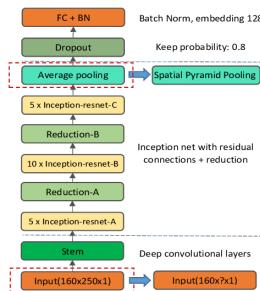
Diskriminator disini berfungsi untuk membedakan antara gambar asli dan gambar palsu. Diskriminatator adalah CNN dan memprediksi gambar yang diberikan adalah nyata atau palsu. Disini terdapat blok konvolusional. Setiap blok konvolusional berisi lapisan konvolusional yang diikuti oleh lapisan normalisasi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



Gambar 9.110 Arsitektur Discriminator Network dari Age-cGAN

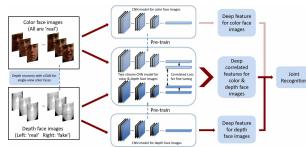
6. Jelaskan dengan ilustrasi gambar sendiri apa itu apa itu pretrained Inception-ResNet-2 Model

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



Gambar 9.111 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 9.112 Arsitektur Face Recognition Network

8. Sebutkan dan jelaskan serta diertai 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 diskriminators dilatih secara bersamaan. Dalam persamaan dibawah ini, θ_G merupakan parameter dari jaringan generator, dan θ_D mewakili parameter G dan D, $\log D(r)$ adalah kehilangan dalam model generator dan p_{data} adalah distribusi dari semua gambar yang mungkin.

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

Gambar 9.113 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 laten, 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 :

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

Gambar 9.114 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.

9.10.2 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(
16         len(dob))]
17
18     # Create a list of tuples containing a pair of an image path
19     # and age
20     images = []
21     age_list = []
22     for index, image_path in enumerate(full_path):
23         images.append(image_path[0])
24         age_list.append(age[index])
25
26     # Return a list of all images and respective age
27     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='same')(input_layer)
9     # enc = BatchNormalization()(enc)
10    enc = LeakyReLU(alpha=0.2)(enc)
11
12    # 2nd Convolutional Block
13    enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='same')(enc)
14    enc = BatchNormalization()(enc)
15    enc = LeakyReLU(alpha=0.2)(enc)
16
17    # 3rd Convolutional Block
18    enc = Conv2D(filters=128, kernel_size=5, strides=2, 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

```

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
7     latent_dims = 100
8     num_classes = 6
9
10    input_z_noise = Input(shape=(latent_dims,))
11    input_label = Input(shape=(num_classes,))
12
13    x = concatenate([input_z_noise, input_label])
14
15    x = Dense(2048, input_dim=latent_dims + num_classes)(x)
16    x = LeakyReLU(alpha=0.2)(x)
17    x = Dropout(0.2)(x)
18
19    x = Dense(256 * 8 * 8)(x)
20    x = BatchNormalization()(x)
21    x = LeakyReLU(alpha=0.2)(x)
22    x = Dropout(0.2)(x)
23
24    x = Reshape((8, 8, 256))(x)
25
26    x = UpSampling2D(size=(2, 2))(x)
27    x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
28    x = BatchNormalization(momentum=0.8)(x)
29    x = LeakyReLU(alpha=0.2)(x)
30
31    x = UpSampling2D(size=(2, 2))(x)
32    x = Conv2D(filters=64, kernel_size=5, padding='same')(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')(x)
38    x = Activation('tanh')(x)

```

```

38     model = Model(inputs=[input_z_noise, input_label], outputs=[x
39         ])
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
4             defined as follows
5         """
6
7     input_shape = (64, 64, 3)
8     label_shape = (6,)
9     image_input = Input(shape=input_shape)
10    label_input = Input(shape=label_shape)
11
12    x = Conv2D(64, kernel_size=3, strides=2, padding='same')(image_input)
13    x = LeakyReLU(alpha=0.2)(x)
14
15    label_input1 = Lambda(expand_label_input)(label_input)
16    x = concatenate([x, label_input1], axis=3)
17
18    x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
19    x = BatchNormalization()(x)
20    x = LeakyReLU(alpha=0.2)(x)
21
22    x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
23    x = BatchNormalization()(x)
24    x = LeakyReLU(alpha=0.2)(x)
25
26    x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
27    x = BatchNormalization()(x)
28    x = LeakyReLU(alpha=0.2)(x)
29
30    x = Flatten()(x)
31    x = Dense(1, activation='sigmoid')(x)
32
33    model = Model(inputs=[image_input, label_input], outputs=[x])
34    return model

```

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_crop1")
5     epochs = 500
6     batch_size = 2
7     image_shape = (64, 64, 3)
8     z_shape = 100
9     TRAIN_GAN = True
10    TRAIN_ENCODER = False
11    TRAIN_GAN_WITH_FR = False
12    fr_image_shape = (192, 192, 3)
13
14    # Define optimizers
15    dis_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
16                          epsilon=10e-8)
17    gen_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
18                          epsilon=10e-8)
19    adversarial_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2
20                                 =0.999, epsilon=10e-8)
21
22    """
23        Build and compile networks
24    """
25
26    # Build and compile the discriminator network
27    discriminator = build_discriminator()
28    discriminator.compile(loss=['binary_crossentropy'], optimizer=
29                           dis_optimizer)
30
31    # Build and compile the generator network
32    generator = build_generator()
33    generator.compile(loss=['binary_crossentropy'], optimizer=
34                       gen_optimizer)
35
36    # Build and compile the adversarial model
37    discriminator.trainable = False
38    input_z_noise = Input(shape=(100,))
39    input_label = Input(shape=(6,))
40    recons_images = generator([input_z_noise, input_label])
41    valid = discriminator([recons_images, input_label])
42    adversarial_model = Model(inputs=[input_z_noise, input_label
43                                     ], outputs=[valid])
44    adversarial_model.compile(loss=['binary_crossentropy'],
45                               optimizer=gen_optimizer)
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)
```

```
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) *
56 0.9
56 fake_labels = np.zeros((batch_size, 1), dtype=np.float32) *
57 0.1
58
59 """
60 Train the generator and the discriminator network
61 """
62 if TRAIN_GAN:
63     for epoch in range(epochs):
64         print("Epoch:{} ".format(epoch))
65
66         gen_losses = []
67         dis_losses = []
68
69         number_of_batches = int(len(loaded_images) / 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
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(
86     ([z_noise, y_batch]))
87
87 d_loss_real = discriminator.train_on_batch([
88     images_batch, y_batch], real_labels)
89     d_loss_fake = discriminator.train_on_batch([
90     initial_recon_images, y_batch], fake_labels)
91
91 d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)
92 print("d_loss:{} ".format(d_loss))
```

```

92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133

```

```

        """
        Train the generator network
        """

        z_noise2 = np.random.normal(0, 1, size=(batch_size, z_shape))
        random_labels = np.random.randint(0, 6, batch_size).reshape(-1, 1)
        random_labels = to_categorical(random_labels, 6)

        g_loss = adversarial_model.train_on_batch([z_noise2, random_labels], [1] * batch_size)

        print("g_loss:{}".format(g_loss))

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

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

    """
    Generate images after every 10th epoch
    """
    if epoch % 10 == 0:
        images_batch = loaded_images[0:batch_size]
        images_batch = images_batch / 127.5 - 1.0
        images_batch = images_batch.astype(np.float32)

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

        gen_images = generator.predict_on_batch([z_noise, y_batch])

        for i, img in enumerate(gen_images[:5]):
            save_rgb_img(img, path="results/img_{}_{}.png"
".format(epoch, i))

    # Save networks
    try:
        generator.save_weights("generator.h5")
        discriminator.save_weights("discriminator.h5")
    except Exception as e:
        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(
36                generated_images, z_batch)
37            print("Encoder loss:", encoder_loss)
38
39            encoder_losses.append(encoder_loss)
40
41            # Write the encoder loss to Tensorboard
42            write_log(tensorboard, "encoder_loss", np.mean(
43                encoder_losses), epoch)
44
45            # Save the encoder model
46            encoder.save_weights("encoder.h5")

```

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

9.10.3 Penanganan Error

1. File Not Found Error

```
#1111111111111111 [Error] 2] No such file or directory: 'C:/Users/DESK/Downloads/n200-apnetcode/200apnets/volumetric_data/soft/3d/test/chair_00000000_1.h5t'
```

Gambar 9.115 File Not FOund Error

2. Cara Penanganan Error

- File Not Found Error

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

9.10.4 Bukti Tidak Plagiat



Gambar 9.116 Bukti Plagiarisme

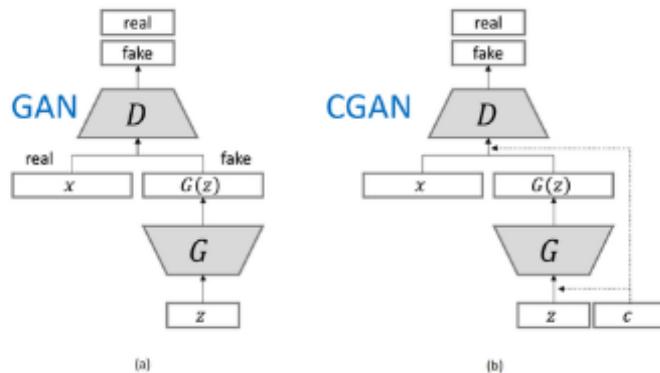
9.10.5 Link Youtube

9.11 Difa Al Fansha

9.11.1 Teori

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

Vanilla GAN memakai data noise yang di proses menjadi data fake sedangkan cGAN memakai latent space atau label untuk generator. Generator dan diskriminator adalah perceptron multi-layer sederhana. perceptron merupakan metode JST sederhana menggunakan algoritma training untuk melakukan klasifikasi secara linear. Perceptron digunakan untuk melakukan klasifikasi sederhana dan membagi data untuk menentukan data ana yang masuk dalam klasifikasi dan data mana yang diluar klasifikasi. Vanilla GAN mengoptimalkan persamaan mtk menggunakan keturunan gradien stokastik.



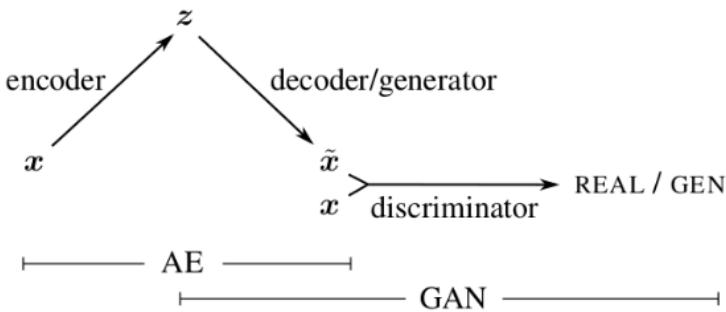
Gambar 9.117

9.11.1.2 Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN

AgecGan terdiri dari empat jaringan: Encoder, FaceNet, Jaringan Generator, dan jaringan diskriminasi. Encoder, kita belajar pemetaan invers gambar wajah masukan dan kondisi usia dengan vektor laten. FaceNet adalah jaringan pengenalan wajah yang mempelajari perbedaan antara gambar input x dan gambar yang direkonstruksi. Kami memiliki jaringan Generator, yang mengambil representasi tersembunyi yang terdiri dari gambar wajah dan vektor kondisi dan menghasilkan gambar. Jaringan diskriminasi adalah untuk mendiskriminasikan antara gambar nyata dan gambar palsu. Masalah dengan cGANs adalah bahwa mereka tidak dapat mempelajari tugas pemetaan terbalik masukan gambar x dengan atribut y ke vektor laten z . Solusi untuk masalah ini adalah dengan menggunakan jaringan Encoder. Kita dapat melatih jaringan encoder untuk memperkirakan pemetaan terbalik dari input Images x

9.11.1.3 Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari AgecGAN

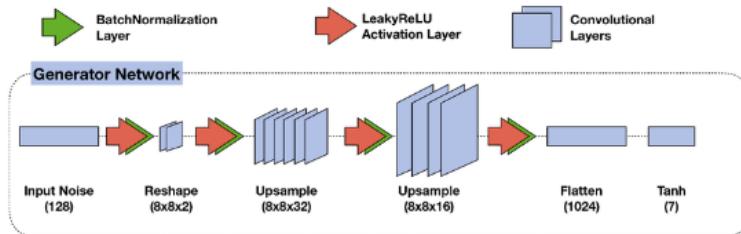
Arsitektur encoder biasanya digunakan untuk memodelkan struktur manifold dan membalikkan encoder untuk memproses data.



Gambar 9.118

9.11.1.4 Jelaskan dengan ilustrasi gambar sendiri arsitektur generator network dari AgecGAN

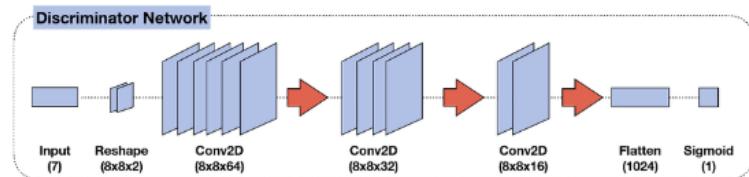
Encoder mempelajari pemetaan terbalik dari gambar wajah input dan kondisi usia dengan vektor laten Z. Jaringan encoder menghasilkan vektor laten dari gambar input. Jaringan Encoder adalah CNN yang mengambil gambar dari dimensi (64, 64, 3) dan mengubahnya menjadi vektor 100 dimensi. Ada empat blok konvolusional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



Gambar 9.119

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

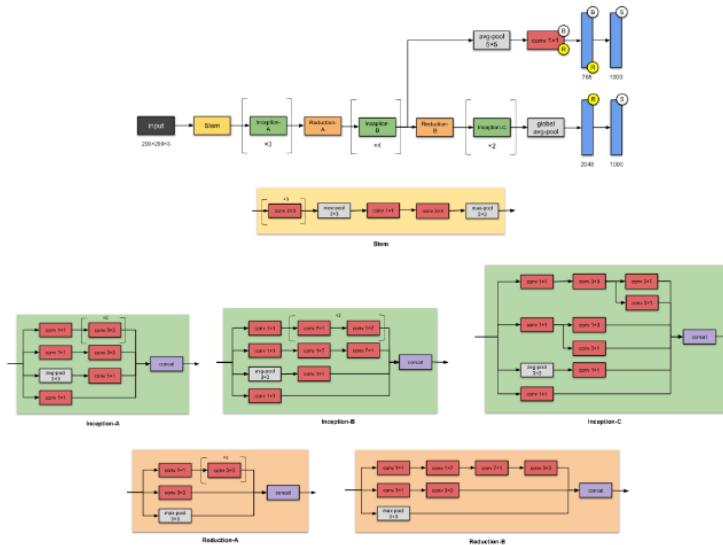
Arsitektur diskriminator adalah CNN yang dapat menerima input gambar yang berukuran 28,28 serta menghasilkan angka biner yang menyatakan apakah data yang diinputkan merupakan dataset asli atau gambar dataset palsu.



Gambar 9.120

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

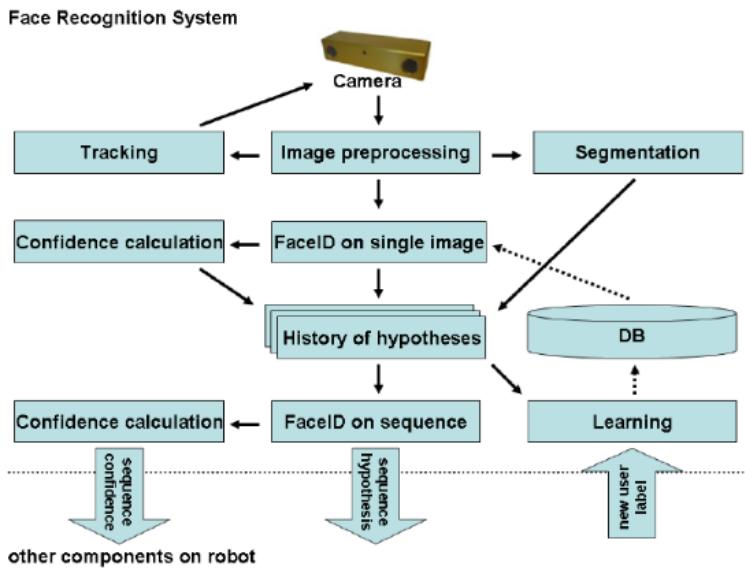
Pre-Trained Network atau Transfer Learning merupakan suatu metode penyelesaian yang memanfaatkan model yang sudah dilatih terhadap suatu dataset untuk menyelesaikan masalah dengan cara menggunakan sebagai starting point, memodifikasi dan mengupdate parameternya, sehingga sesuai dengan dataset yang baru. Inception resNet 2 adalah jaringan saran konvolusional yang dilatih lebih dari satu juta gambar



Gambar 9.121

9.11.1.7 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 9.122

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

Terdapat 3 tahapan :

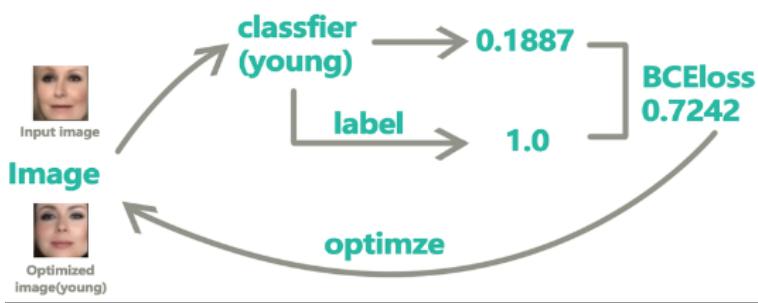
- Conditional GAN, Melatih generator dan discriminator.
- Initial Lantent Vector Approxmation, Melatih encoder.
- Latent vector optimization, Mengoptimalkan encode dan generator

9.11.1.9 Berikan contoh perhitungan fungsi training objektif

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

9.11.1.10 Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation

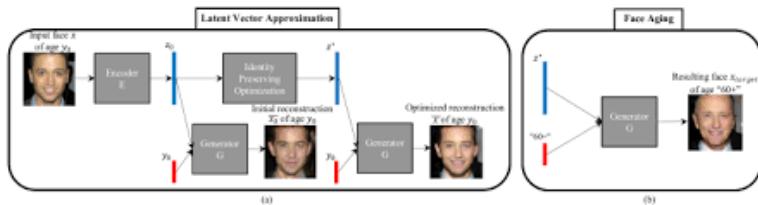
Latent vector approximation kemampuan untuk membuat gamar yang realistik dan tajam serta menghasilkan gambar wajah pada usia target.



Gambar 9.123

9.11.1.11 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 z_i dalam ruang laten z .



Gambar 9.124

9.11.2 Praktek

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

```
1 # In[1. Ekstrak File]:
2 import tarfile
3 tf = tarfile.open("/content/drive/My Drive/Kecerdasan Buatan/
    wiki_crop.tar")
4 tf.extractall(path="/content/drive/My Drive/Kecerdasan Buatan")
```

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

```
1 # In[2. Load Data]:
```

```

2 def load_data(wiki_dir, dataset='wiki'):
3     # Load the wiki.mat file
4     meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset)))
5
6     # Load the list of all files
7     full_path = meta[dataset][0, 0]["full_path"][0]
8
9     # List of Matlab serial date numbers
10    dob = meta[dataset][0, 0]["dob"][0]
11
12    # List of years when photo was taken
13    photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
14
15    # Calculate age for all dobs
16    age = [calculate_age(photo_taken[i], dob[i]) for i in range(len(dob))]
17
18    # Create a list of tuples containing a pair of an image path and
19    # age
20    images = []
21    age_list = []
22    for index, image_path in enumerate(full_path):
23        images.append(image_path[0])
24        age_list.append(age[index])
25
26    # Return a list of all images and respective age
27    return images, age_list

```

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

Proses Encoder berfungsi untuk mempelajari pemetaan terbalik dari gambar wajah dan kondisi usia dengan vector latent Z.

```

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

```

```

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

```

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

Proses Generator agar bekerja dengan baik dibutuhkan representasi dari gambar wajah dan vector kondisi sebagai inputan yang menghasilkan sebuah gambar.

```

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

```

```

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)

```

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

Proses Discriminator untuk membedakan antara gambar asli dan gambar palsu.

```

1 # In [5. Discriminator Network Bekerja]:
2 def build_discriminator():
3     """
4         Create a Discriminator Model with hyperparameters values defined
5         as follows
6         """
7
8     input_shape = (64, 64, 3)
9     label_shape = (6,)
10    image_input = Input(shape=input_shape)
11    label_input = Input(shape=label_shape)
12
13    x = Conv2D(64, kernel_size=3, strides=2, padding='same')(image_input)
14    x = LeakyReLU(alpha=0.2)(x)
15
16    label_input1 = Lambda(expand_label_input)(label_input)
17    x = concatenate([x, label_input1], axis=3)
18
19    x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
20    x = BatchNormalization()(x)
21    x = LeakyReLU(alpha=0.2)(x)
22
23    x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
24    x = BatchNormalization()(x)
25    x = LeakyReLU(alpha=0.2)(x)
26
27    x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
28    x = BatchNormalization()(x)
29    x = LeakyReLU(alpha=0.2)(x)
30
31    x = Flatten()(x)
32    x = Dense(1, activation='sigmoid')(x)
33
34    model = Model(inputs=[image_input, label_input], outputs=[x])
35    return model

```

9.11.2.6 Jelaskan bagaimana kode program Training cGAN bekerja dijelaskan dengan bahawa awam dengan ilustrasi sederhana

Proses Training cGAN ini dengan load file .mat pada dataset lalu epoch sebanyak 500 kali.

```

1 # In [6. Training cGAN]:
2     if __name__ == '__main__':
3         # Define hyperparameters

```

```

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

```

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

```

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

Initial dan Latent Vector Approximation bekerja melakukan predicsi epoch yang telah di buat sebanyak 500 kali, dan nanti hasilnya ada di folder result.

9.11.3 Penangan Error

9.11.3.1 Screenshoots Error

9.11.3.2 Tuliskan kode eror dan jenis errornya

9.11.3.3 Solusi pemecahan masalah error tersebut

9.11.4 Bukti Tidak Plagiat



Gambar 9.125 Bukti Tidak Plagiat

9.11.5 Link Youtube:

<https://www.youtube.com/watch?v=RYcrSdyEaQc>

9.12 1174077 - Alvan Alvanzah

9.12.1 Teori

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

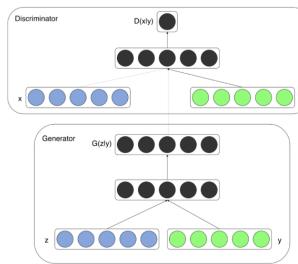
Vanilla GAN Vanilla GAN adalah tipe GAN paling sederhana. Di sini, Generator dan Diskriminator adalah perceptron multi-layer sederhana. Dalam vanilla GAN, algoritma ini sangat sederhana, ia mencoba untuk mengoptimalkan persamaan matematika menggunakan keturunan gradien stokastik. CGAN (Conditional GAN),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



Gambar 9.126 Valina GAN-cGAN

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

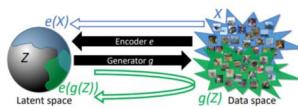
Age cGAN ialah dengan mengkondisikan model pada informasi tambahan dimungkinkan untuk mengarahkan proses pembuatan data. Pengkondisian semacam itu dapat didasarkan pada label kelas.



Gambar 9.127 Age-cGAN

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

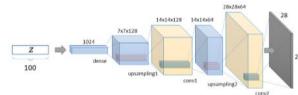
Arsitektur encoder biasanya digunakan untuk memodelkan struktur manifold dan membalikkan encoder untuk memproses data.



Gambar 9.128 Encoder Age cGANr

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

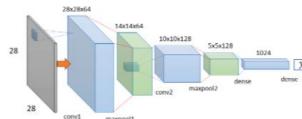
Arsitektur generator adalah sebuah array yang digunakan secara random, yang disebut seed. dari data seed tersebut, generator akan merubahnya menjadi sebuah gambar yang ukuran 28×28 dengan menggunakan Convolutional Neural Network.



Gambar 9.129 Network Age cGAN

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

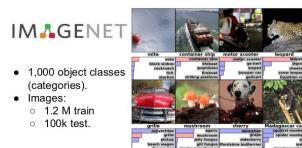
Arsitektur diskriminatator adalah CNN yang dapat menerima input gambar yang berukuran $28,28$ serta menghasilkan angka biner yang menyatakan apakah data yang diinputkan merupakan dataset asli atau gambar dataset palsu.



Gambar 9.130 Discriminator Age cGAN

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

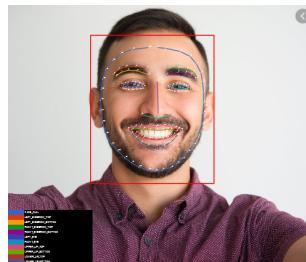
Pre-Trained Network atau Transfer Learning merupakan suatu metode penyelesaian yang memanfaatkan model yang sudah dilatih terhadap suatu dataset untuk menyelesaikan masalah dengan cara menggunakan sebagai starting point, memodifikasi dan mengupdate parameternya, sehingga sesuai dengan dataset yang baru.



Gambar 9.131 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.

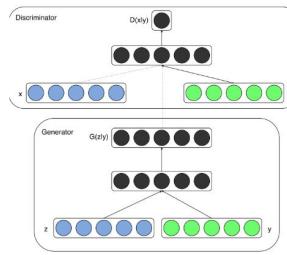


Gambar 9.132 Face recognition network Age-cGAN

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

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

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



Gambar 9.133 Tahap Age cGAN

9. Berikan contoh perhitungan fungsi training objektif.

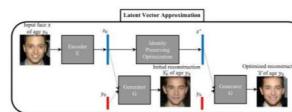
Objektif Training ialah untuk meminimalkan loss function sebagai log likelihood function yang diberikan pada persamaan dimana D melambangkan training data.

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

Gambar 9.134 Training Objektif

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

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



Gambar 9.135 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 z_i dalam ruang laten z .



Gambar 9.136 Latent Vector Optimization

9.12.2 Praktek

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

```

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

Kode di atas akan melakukan mount dan extract data.

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

```

1 # In[2. Load Data]:
2 def load_data(wiki_dir, dataset='wiki'):
3     # Load the wiki.mat file
4     meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset
5         )))
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
10    dob = meta[dataset][0, 0]["dob"][0]
11
12    # List of years when photo was taken
13    photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
14
15    # Calculate age for all dobs
16    age = [calculate_age(photo_taken[i], dob[i]) for i in range(
17        len(dob))]
18
18    # Create a list of tuples containing a pair of an image path
19    # and age
20    images = []
21    age_list = []
22    for index, image_path in enumerate(full_path):
23        images.append(image_path[0])
24        age_list.append(age[index])
25
25    # Return a list of all images and respective age
```

```
26     return images, age_list
```

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.

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

Proses Encoder berfungsi untuk mempelajari pemetaan terbalik dari gambar wajah dan kondisi usia dengan vector latent Z.

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

```

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

```

Proses Generator agar bekerja dengan baik dibutuhkan representasi dari gambar wajah dan vector kondisi sebagai inputan yang menghasilkan sebuah gambar.

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

```

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

```

```

14     label_input1 = Lambda(expand_label_input)(label_input)
15     x = concatenate([x, label_input1], axis=3)
16
17     x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
18     x = BatchNormalization()(x)
19     x = LeakyReLU(alpha=0.2)(x)
20
21     x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
22     x = BatchNormalization()(x)
23     x = LeakyReLU(alpha=0.2)(x)
24
25     x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
26     x = BatchNormalization()(x)
27     x = LeakyReLU(alpha=0.2)(x)
28
29     x = Flatten()(x)
30     x = Dense(1, activation='sigmoid')(x)
31
32 model = Model(inputs=[image_input, label_input], outputs=[x])
33 return model

```

Proses Discriminator untuk membedakan antara gambar asli dan gambar palsu.

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

```

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

```

Proses Training cGAN ini 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.

```

1 # In [7. Latent Vector]:
2     ....

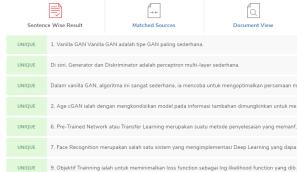
```

```
3     Train encoder
4
5
6 if TRAIN_ENCODER:
7     # Build and compile encoder
8     encoder = build_encoder()
9     encoder.compile(loss=euclidean_distance_loss, optimizer='adam')
10
11    # Load the generator network's weights
12    try:
13        generator.load_weights("generator.h5")
14    except Exception as e:
15        print("Error:", e)
16
17    z_i = np.random.normal(0, 1, size=(5000, z_shape))
18
19    y = np.random.randint(low=0, high=6, size=(5000,), dtype=
np.int64)
20    num_classes = len(set(y))
21    y = np.reshape(np.array(y), [len(y), 1])
22    y = to_categorical(y, num_classes=num_classes)
23
24    for epoch in range(epochs):
25        print("Epoch:", epoch)
26
27        encoder_losses = []
28
29        number_of_batches = int(z_i.shape[0] / batch_size)
30        print("Number of batches:", number_of_batches)
31        for index in range(number_of_batches):
32            print("Batch:", index + 1)
33
34            z_batch = z_i[index * batch_size:(index + 1) *
batch_size]
35            y_batch = y[index * batch_size:(index + 1) *
batch_size]
36
37            generated_images = generator.predict_on_batch([
z_batch, y_batch])
38
39            # Train the encoder model
40            encoder_loss = encoder.train_on_batch(
41                generated_images, z_batch)
42            print("Encoder loss:", encoder_loss)
43
44            encoder_losses.append(encoder_loss)
45
46            # Write the encoder loss to Tensorboard
47            write_log(tensorboard, "encoder_loss", np.mean(
48                encoder_losses), epoch)
49
50            # Save the encoder model
51            encoder.save_weights("encoder.h5")
```

Initial dan Latent Vector Approximation bekerja melakukan predicsi epoch yang telah di buat sebanyak 500 kali, dan nanti hasilnya ada di folder result.

9.12.3 Penanganan Error

9.12.4 Bukti Tidak Plagiat



Gambar 9.137 Bukti Tidak Melakukan Plagiat Chapter 9

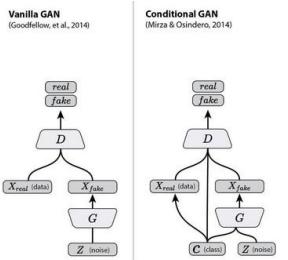
9.13 1174071 - Muhammad Abdul Gani Wijaya

9.13.1 Teori

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

Vanilla GANs biasanya tidak memiliki convolutional Neural Jaringan (CNNs) di jaringan mereka. Conditional GANs (cGANs) adalah perpanjangan dari model GAN. Mereka memungkinkan untuk generasi gambar yang memiliki kondisi tertentu atau atribut dan telah terbukti menjadi lebih baik dari Vanilla GANs sebagai hasilnya.

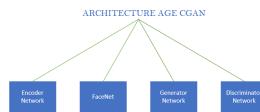
cGANs adalah jenis GAN yang dikondisikan pada beberapa informasi tambahan. informasi tambahan y ke Generator sebagai lapisan input tambahan. Dalam Vanilla GANs, tidak ada kontrol atas Kategori gambar yang dihasilkan. Ketika kita menambahkan kondisi y ke Generator, kita dapat menghasilkan gambar dari kategori tertentu, menggunakan y, yang mungkin jenis data, seperti label kelas atau data integer. Vanilla GANs bisa belajar hanya satu kategori dan sangat sulit untuk arsitek GANs untuk beberapa kategori. Sebuah cGAN, bagaimanapun, dapat digunakan untuk menghasilkan model multi-modal dengan kondisi yang berbeda untuk kategori yang berbeda.



Gambar 9.138 Illustrasi Vanilla GAN dan cGAN

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

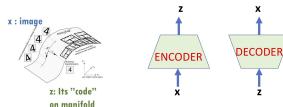
Arsitektur cGAN untuk penuaan wajah sedikit lebih rumit. AgecGan terdiri dari empat jaringan: Encoder, FaceNet, Jaringan Generator, dan jaringan diskriminat. Dengan Encoder, kita belajar pemetaan invers gambar wajah masukan dan kondisi usia dengan vektor laten. FaceNet adalah jaringan pengenalan wajah yang mempelajari perbedaan antara gambar input x dan gambar yang direkonstruksi. Kami memiliki jaringan Generator, yang mengambil representasi tersembunyi yang terdiri dari gambar wajah dan vektor kondisi dan menghasilkan gambar. Jaringan diskriminat adalah untuk mendiskriminasikan antara gambar nyata dan gambar palsu. Masalah dengan cGANs adalah bahwa mereka tidak dapat mempelajari tugas pemetaan terbalik masukan gambar x dengan atribut y ke vektor laten z . Solusi untuk masalah ini adalah dengan menggunakan jaringan Encoder. Kita dapat melatih jaringan encoder untuk memperkirakan pemetaan terbalik dari input Images x .



Gambar 9.139 Illustrasi Arsitektur cGAN

3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari AgecGAN.

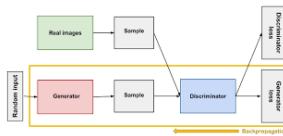
Tujuan utama dari jaringan Encoder adalah untuk menghasilkan vektor laten dari gambar yang disediakan. Pada dasarnya, dibutuhkan gambar dimensi (64, 64, 3) dan mengubahnya menjadi vektor 100-dimensi. Jaringan Encoder adalah jaringan syaraf convolutional yang dalam. Jaringan berisi empat convolutional blok dan dua lapisan padat. Setiap blok convolutional berisi lapisan convolutional, lapisan normalisasi batch, dan fungsi aktivasi. Di setiap blok convolutional, setiap lapisan convolutional diikuti oleh lapisan normalisasi batch, kecuali lapisan convolutional pertama.



Gambar 9.140 Illustrasi Network Encoder

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

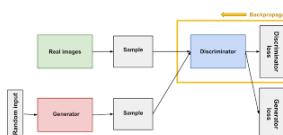
Tujuan utama dari generator adalah untuk menghasilkan gambar dari dimensi (64, 64, 3). Dibutuhkan vektor laten 100 dimensi dan beberapa informasi tambahan, y, dan mencoba untuk menghasilkan gambar yang realistik. Jaringan Generator adalah jaringan neural yang mendalam convolutional juga. Hal ini terdiri dari lapisan padat, upsampling, dan convolutional. Dibutuhkan dua nilai input: vektor kebisingan dan nilai pengkondisian. Nilai pengkondisian adalah informasi tambahan yang diberikan ke jaringan. Untuk Age-cGAN, ini akan menjadi usia.



Gambar 9.141 Illustrasi Network Generator

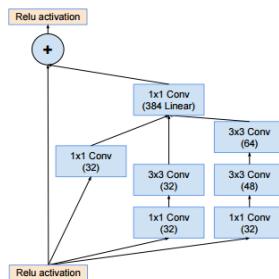
- Jelaskan dengan ilustrasi gambar sendiri arsitektur discriminator network dari Age-cGAN.

Tujuan utama dari jaringan diskriminator adalah untuk mengidentifikasi apakah gambar yang disediakan adalah palsu atau nyata. Hal ini dilakukan dengan melewati gambar melalui serangkaian lapisan sampling bawah dan beberapa lapisan klasifikasi. Dengan kata lain, ini memprediksi Apakah gambar itu nyata atau palsu. Seperti jaringan lain, Jaringan diskriminator lain dalam jaringan convolutional. Ini berisi beberapa blok convolutional. Setiap blok convolutional berisi lapisan convolutional, lapisan normalisasi batch, dan fungsi aktivasi, selain blok convolutional pertama, yang tidak memiliki lapisan normalisasi batch.



Gambar 9.142 Illustrasi Discriminator Network

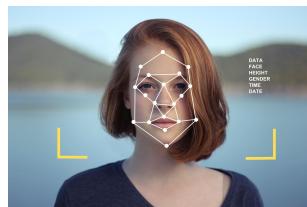
5. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model. pre-trained Inception-ResNet-2 network, sekali disediakan dengan gambar, mengembalikan yang sesuai embedding. Tertanam yang diekstrak untuk gambar asli dan gambar direkonstruksi dapat dihitung dengan menghitung jarak Euclidean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



Gambar 9.143 Illustrasi Inception-ResNet-2 Model.

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

Tujuan utama dari jaringan pengenalan wajah adalah untuk mengenali identitas seseorang dalam gambar yang diberikan.



Gambar 9.144 Illustrasi Face recognition network Age-cGAN.

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

Age-cGAN memiliki beberapa tahapan pelatihan. Seperti disebutkan di bagian sebelumnya, Age-cGAN memiliki empat jaringan, yang dilatih dalam tiga tahap. Pelatihan AgecGAN terdiri dari tiga tahap:

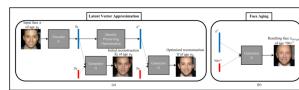
- pelatihan GAN bersyarat: pada tahap ini, kita melatih jaringan Generator dan jaringan diskriminator.
- awal pendekatan vektor laten: pada tahap ini, kami melatih jaringan Encoder.

- optimasi vektor laten: pada tahap ini, kami mengoptimalkan kedua encoder dan jaringan generator.

8. Berikan contoh perhitungan fungsi training objektif

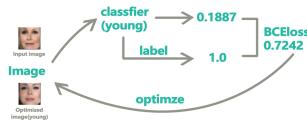
9. Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation

Perkiraan vektor laten awal adalah metode untuk memperkirakan vektor laten untuk mengoptimalkan rekonstruksi gambar wajah. Untuk memperkirakan vektor laten, kami memiliki jaringan Encoder. Kami melatih jaringan Encoder pada gambar yang dihasilkan dan gambar nyata. Setelah dilatih, Jaringan Encoder akan mulai menghasilkan vektor laten dari Distribusi. Tujuan pelatihan fungsi untuk pelatihan jaringan Encoder adalah kehilangan jarak Euclidean.



Gambar 9.145 Illustrasi Initial latent vector approximation

10. Berikan contoh perhitungan latent vector optimization



Gambar 9.146 Contoh Perhitungan Latent vector optimization

9.13.2 Praktek

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

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

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

```

1 # In[2. Load Data]:
2 def load_data(wiki_dir, dataset='wiki'):
3     # Load the wiki.mat file
4     meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset
5         )))
6
7     # Load the list of all files
8     full_path = meta[dataset][0, 0]["full_path"][0]
9
10    # List of Matlab serial date numbers
11    dob = meta[dataset][0, 0]["dob"][0]
12
13    # List of years when photo was taken
14    photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
15
16    # Calculate age for all dobs
17    age = [calculate_age(photo_taken[i], dob[i]) for i in range(
18        len(dob))]
19
20    # Create a list of tuples containing a pair of an image path
21    # and age
22    images = []
23    age_list = []
24    for index, image_path in enumerate(full_path):
25        images.append(image_path[0])
26        age_list.append(age[index])
27
28    # Return a list of all images and respective age
29    return images, age_list
30

```

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

```

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

```

```

19 enc = Conv2D(filters=128, kernel_size=5, strides=2, padding='same')(enc)
20 enc = BatchNormalization()(enc)
21 enc = LeakyReLU(alpha=0.2)(enc)
22
23 # 4th Convolutional Block
24 enc = Conv2D(filters=256, kernel_size=5, strides=2, padding='same')(enc)
25 enc = BatchNormalization()(enc)
26 enc = LeakyReLU(alpha=0.2)(enc)
27
28 # Flatten layer
29 enc = Flatten()(enc)
30
31 # 1st Fully Connected Layer
32 enc = Dense(4096)(enc)
33 enc = BatchNormalization()(enc)
34 enc = LeakyReLU(alpha=0.2)(enc)
35
36 # Second Fully Connected Layer
37 enc = Dense(100)(enc)
38
39 # Create a model
40 model = Model(inputs=[input_layer], outputs=[enc])
41 return model

```

4. Jelaskan bagaimana kode program The Generator Network bekerja dijelaskan dengan bahawa awam dengan ilustrasi sederhana. Proses Generator agar bekerja dengan baik dibutuhkan representasi dari gambar wajah dan vector kondisi sebagai inputan yang menghasilkan sebuah gambar.

```

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

```

```

25     x = UpSampling2D(size=(2, 2))(x)
26     x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
27     x = BatchNormalization(momentum=0.8)(x)
28     x = LeakyReLU(alpha=0.2)(x)
29
30     x = UpSampling2D(size=(2, 2))(x)

```

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

```

1 # In [5. Discriminator Network Bekerja]:
2 def build_discriminator():
3     """
4         Create a Discriminator Model with hyperparameters values
5             defined as follows
6         """
7
8     input_shape = (64, 64, 3)
9     label_shape = (6,)
10    image_input = Input(shape=input_shape)
11    label_input = Input(shape=label_shape)
12
13    x = Conv2D(64, kernel_size=3, strides=2, padding='same')(image_input)
14    x = LeakyReLU(alpha=0.2)(x)
15
16    label_input1 = Lambda(expand_label_input)(label_input)
17    x = concatenate([x, label_input1], axis=3)
18
19    x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
20    x = BatchNormalization()(x)
21    x = LeakyReLU(alpha=0.2)(x)
22
23    x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
24    x = BatchNormalization()(x)
25    x = LeakyReLU(alpha=0.2)(x)
26
27    x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
28    x = BatchNormalization()(x)
29    x = LeakyReLU(alpha=0.2)(x)
30
31    x = Flatten()(x)
32    x = Dense(1, activation='sigmoid')(x)
33
34    model = Model(inputs=[image_input, label_input], outputs=[x])
35    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.

```

1 # In [6. Training cGAN]:
2     if __name__ == '__main__':
3         # Define hyperparameters
4         data_dir = "data"

```

```

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

```

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

```

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

```

```

32             print("Batch:", index + 1)
33
34             z_batch = z_i[index * batch_size:(index + 1) *
batch_size]
35             y_batch = y[index * batch_size:(index + 1) *
batch_size]
36
37             generated_images = generator.predict_on_batch([
z_batch, y_batch])
38
39             # Train the encoder model
40             encoder_loss = encoder.train_on_batch(
generated_images, z_batch)
41             print("Encoder loss:", encoder_loss)
42
43             encoder_losses.append(encoder_loss)
44
45             # Write the encoder loss to Tensorboard
46             write_log(tensorboard, "encoder_loss", np.mean(
encoder_losses), epoch)
47
48             # Save the encoder model
49             encoder.save_weights("encoder.h5")

```

9.13.3 Link Youtube

<https://www.youtube.com/watch?v=pG7H70LNONU>

9.14 Nurul Izza Hamka - 1174062

9.14.1 Teori

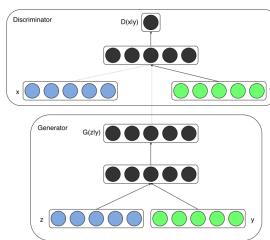
1. Jelaskan dengan ilustrasi gambar sendiri apa perbedaan antara vanilla GAN dan cGAN.

Vanilla GAN biasanya tidak memiliki jaringan saraf convolutional (CNN) di jaringan mereka. Ini diusulkan untuk pertama kalinya dengan diperkenalkannya DCGAN. Dalam vanilla GANs, tidak ada kontrol atas kategori gambar yang dihasilkan. Vanilla GAN hanya dapat mempelajari satu kategori dan sangat sulit untuk merancang GAN untuk beberapa kategori.

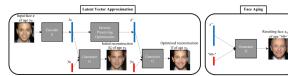
Sedangkan Conditional GANs (cGANs) adalah perpanjangan dari model GAN. Mereka memungkinkan untuk generasi gambar yang memiliki kondisi atau atribut tertentu dan telah terbukti lebih baik daripada vanilla GAN sebagai hasilnya. cGANs memperluas gagasan vanilla GAN dan memungkinkan kami untuk mengontrol output dari jaringan generator.

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

Age-cGAN terdiri dari empat jaringan: encoder, FaceNet, jaringan generator, dan jaringan diskriminator. Dengan encoder, kita belajar pemetaan terbalik



Gambar 9.147 Ilustrasi Nomor 1



Gambar 9.148 Ilustrasi Nomor 2

dari gambar wajah input dan kondisi usia dengan vektor laten. FaceNet adalah jaringan pengenalan wajah yang mempelajari perbedaan antara gambar input x dan gambar yang direkonstruksi.

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

Jaringan encoder adalah jaringan saraf convolutional yang mendalam. Tujuan utama dari jaringan encoder adalah untuk mengasilkan sebuah vector laten dari gambar yang telah digunakan. Pada dasarnya, ini mengambil gambar dari dimensi (64, 64, 3) dan mengubahnya menjadi vektor 100 dimensi.

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

A Generator Network: Dibutuhkan representasi tersembunyi dari gambar wajah dan vektor kondisi sebagai input dan menghasilkan gambar.

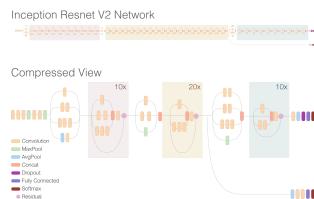
Jaringan Generator adalah CNN dan dibutuhkan vektor laten 100 dimensi dan vektor kondisi y , dan mencoba menghasilkan gambar realistik dari dimensi (64, 64, 3)

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

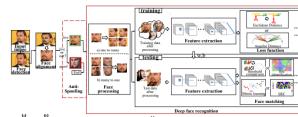
Jaringan Diskriminator, mencoba membedakan antara gambar asli dan gambar palsu.

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

Inception-ResNet-v2 adalah jaringan saraf convolutional yang dilatih pada lebih dari satu juta gambar dari database ImageNet. Jaringannya memiliki 164 lapisan



Gambar 9.149 Ilustrasi Nomor 6



Gambar 9.150 Ilustrasi Nomor 7

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

Gambar 9.151 Perhitungan Fungsi Training Objektif

dan dapat mengklasifikasikan gambar ke dalam 1000 kategori objek, seperti keyboard, mouse, pensil, dan banyak binatang.

7. Jelaskan dengan ilustrasi gambar sendiri arsitektur Face reconition network Age-cGAN.
- Tujuan utama dari Face reconition adalah untuk mengenali identitas seseorang dalam gambar yang diberikan.
- Sebutkan dan jelaskan disertai contoh-contoh tahapan dari Age-cGAN.
 - a. Conditional GAN training, Pada tahap ini adalah melatih generator jaringan dan jaringan diskriminator. Setelah dilatih, jaringan generator dapat menghasilkan gambar wajah yang buram.
 - b. Initial latent vector approximation, Pada tahap ini adalah melatih jaringan encoder. Setelah dilatih, jaringan encoder akan mulai menghasilkan vektor laten dari distribusi yang dipelajari. Fungsi tujuan pelatihan untuk melatih jaringan encoder adalah kehilangan jarak Euclidean.
 - c. Latent vector optimization, Pada tahap ini adalah mengoptimalkan encoder dan jaringan generator.
9. Berikan contoh perhitungan fungsi training objektif.

Dimana:

- $\log D(x, y)$ adalah kerugian untuk model Diskriminator.
- $\log (1 - D(x, y'), y')$ adalah kerugian untuk model Generator.
- $P(\text{data})$ adalah distribusi dari semua gambar yang mungkin.

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

Gambar 9.152 Peritungang Latent Vector Optimazation

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

Dimana:

FR adalah jaringan pengenalan wajah. Persamaan ini menunjukkan bahwa jarak Euclidean antara gambar asli dan gambar yang direkonstruksi harus minimal. Pada tahap ini, kami mencoba meminimalkan jarak untuk memaksimalkan pelitearian identitas.

9.14.2 Praktek Program

1. Jelaskan bagaimana cara extract file dari Age-cGAN menggunakan google colab.

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

Kode diatas adalah untuk melakukan ekstrak file menggunakan google colab.

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

```
1 # In[2. Load Data]:
2 def load_data(wiki_dir, dataset='wiki'):
3
```

```

4 meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset
5  )))
6 full_path = meta[dataset][0, 0]["full_path"][0]
7 dob = meta[dataset][0, 0]["dob"][0]
8 photo_taken = meta[dataset][0, 0]["photo_taken"][0] # year
9
10 age = [calculate_age(photo_taken[i], dob[i]) for i in range(
11   len(dob))]
12
13 images = []
14 age_list = []
15 for index, image_path in enumerate(full_path):
16   images.append(image_path[0])
17   age_list.append(age[index])
18
19 return images, age_list
20
21 # In[3. The Encoder Network bekerja ]
22 def build_encoder():
23   """
24     Encoder Network
25   """
26

```

Pertama Load file wiki.mat

kedua load semua file

ketiga Daftar nomor tanggal seri Matlab

keempat Daftar tahun ketika foto diambil

kelima Hitung usia untuk semua dobs

selanjutnya Buat daftar tupel yang berisi sepasang jalur dan usia gambar Terakhir Kembalikan daftar semua gambar dan usia masing-masing

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

```

1 # In[3. The Encoder Network bekerja ]
2 def build_encoder():
3   """
4     Encoder Network
5   """
6   input_layer = Input(shape=(64, 64, 3))
7   # 1st Convolutional Block
8   enc = Conv2D(filters=32, kernel_size=5, strides=2, padding='same')(input_layer)
9   enc = LeakyReLU(alpha=0.2)(enc)
10
11   # 2nd Convolutional Block
12   enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='same')(enc)
13   enc = BatchNormalization()(enc)

```

```

14 enc = LeakyReLU(alpha=0.2)(enc)
15
16 # 3rd Convolutional Block
17 enc = Conv2D(filters=128, kernel_size=5, strides=2, padding='same')(enc)
18 enc = BatchNormalization()(enc)
19 enc = LeakyReLU(alpha=0.2)(enc)
20
21 # 4th Convolutional Block
22 enc = Conv2D(filters=256, kernel_size=5, strides=2, padding='same')(enc)
23 enc = BatchNormalization()(enc)
24 enc = LeakyReLU(alpha=0.2)(enc)
25
26 # Flatten layer
27 enc = Flatten()(enc)
28
29 # 1st Fully Connected Layer
30 enc = Dense(4096)(enc)
31 enc = BatchNormalization()(enc)
32 enc = LeakyReLU(alpha=0.2)(enc)
33
34 # Second Fully Connected Layer
35 enc = Dense(100)(enc)
36
37 # Create a model
38 model = Model(inputs=[input_layer], outputs=[enc])
39 return model

```

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

```

1 # In[4]: The Generator Network Bekerja ]:
2 def build_generator():
3     """
4         Create a Generator Model with hyperparameters values defined
5             as follows
6     """
7     latent_dims = 100
8     num_classes = 6
9
10    input_z_noise = Input(shape=(latent_dims,))
11    input_label = Input(shape=(num_classes,))
12
13    x = concatenate([input_z_noise, input_label])
14
15    x = Dense(2048, input_dim=latent_dims + num_classes)(x)
16    x = LeakyReLU(alpha=0.2)(x)
17    x = Dropout(0.2)(x)
18
19    x = Dense(256 * 8 * 8)(x)
20    x = BatchNormalization()(x)
21    x = LeakyReLU(alpha=0.2)(x)

```

```

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
41 return model

```

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

```

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

```

```

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

```

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

```

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

```

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

```

1 # In[7. Initial latent vector approximation]:
2     """
3         Train encoder
4     """
5
6     if TRAIN_ENCODER:
7         # Build and compile encoder
8         encoder = build_encoder()
9         encoder.compile(loss=euclidean_distance_loss, optimizer='adam')
10
11        # Load the generator network's weights
12        try:
13            generator.load_weights("generator.h5")

```

```

14         except Exception as e:
15             print("Error:", e)
16
17     z_i = np.random.normal(0, 1, size=(5000, z_shape))
18
19     y = np.random.randint(low=0, high=6, size=(5000,), dtype=
20     np.int64)
21     num_classes = len(set(y))
22     y = to_categorical(y, num_classes=num_classes)
23
24     for epoch in range(epochs):
25         print("Epoch:", epoch)
26
27         encoder_losses = []
28
29         number_of_batches = int(z_i.shape[0] / batch_size)
30         print("Number of batches:", number_of_batches)
31         for index in range(number_of_batches):
32             print("Batch:", index + 1)
33
34             z_batch = z_i[index * batch_size:(index + 1) *
35             batch_size]
35             y_batch = y[index * batch_size:(index + 1) *
36             batch_size]
37
38             generated_images = generator.predict_on_batch([
39                 z_batch, y_batch])
40
41             # Train the encoder model
42             encoder_loss = encoder.train_on_batch(
43                 generated_images, z_batch)
44             print("Encoder loss:", encoder_loss)
45
46             encoder_losses.append(encoder_loss)
47
48             # Write the encoder loss to Tensorboard
49             write_log(tensorboard, "encoder_loss", np.mean(
50                 encoder_losses), epoch)
51
52             # Save the encoder model
53             encoder.save_weights("encoder.h5")

```

9.14.3 Bukti Tidak Plagiat

9.14.4 Link Youtube

<https://www.youtube.com/watch?v=GV18ejp6DE>



Gambar 9.153 Hasil Tidak Plagiat

