

**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

---

|                    |           |
|--------------------|-----------|
| <b>1 Chapter 1</b> | <b>1</b>  |
| <b>2 Chapter 2</b> | <b>3</b>  |
| <b>3 Chapter 3</b> | <b>5</b>  |
| <b>4 Chapter 4</b> | <b>7</b>  |
| <b>5 Chapter 5</b> | <b>9</b>  |
| <b>6 Chapter 6</b> | <b>11</b> |
| <b>7 Chapter 7</b> | <b>13</b> |



# DAFTAR ISI

---

|                                                          |          |
|----------------------------------------------------------|----------|
| Foreword                                                 | xi       |
| Kata Pengantar                                           | xiii     |
| Acknowledgments                                          | xv       |
| Acronyms                                                 | xvii     |
| Glossary                                                 | xix      |
| List of Symbols                                          | xxi      |
| Introduction<br><i>Rolly Maulana Awangga, S.T., M.T.</i> | xxiii    |
| <b>1 Chapter 1</b>                                       | <b>1</b> |
| <b>2 Chapter 2</b>                                       | <b>3</b> |
| <b>3 Chapter 3</b>                                       | <b>5</b> |
| <b>4 Chapter 4</b>                                       | <b>7</b> |
|                                                          | ix       |

|          |                                   |           |
|----------|-----------------------------------|-----------|
| <b>5</b> | <b>Chapter 5</b>                  | <b>9</b>  |
| <b>6</b> | <b>Chapter 6</b>                  | <b>11</b> |
| <b>7</b> | <b>Chapter 7</b>                  | <b>13</b> |
| 7.1      | 1174027 - Harun Ar - Rasyid       | 13        |
| 7.1.1    | Teori                             | 13        |
| 7.1.2    | Praktek                           | 18        |
| 7.1.3    | Penanganan Error                  | 26        |
| 7.1.4    | Bukti Tidak Plagiat               | 27        |
| 7.2      | 1174051 Evietania Charis Sujadi   | 27        |
| 7.2.1    | Teori                             | 27        |
| 7.2.2    | Praktek                           | 33        |
| 7.2.3    | Penanganan Error                  | 55        |
| 7.3      | 1174021 - Muhammad Fahmi          | 55        |
| 7.3.1    | Soal Teori                        | 55        |
| 7.3.2    | Praktek Program                   | 61        |
| 7.3.3    | Penanganan Error                  | 73        |
| 7.3.4    | Bukti Tidak Plagiat               | 74        |
| 7.4      | 1174026 Felix Lase                | 74        |
| 7.4.1    | Teori                             | 74        |
| 7.4.2    | Praktek                           | 80        |
| 7.4.3    | Penanganan Error                  | 102       |
| 7.5      | 1174017 Muh. Rifky Prananda       | 103       |
| 7.5.1    | Teori                             | 103       |
| 7.5.2    | Praktek                           | 109       |
| 7.5.3    | Penanganan Error                  | 130       |
| 7.6      | 1174009 - Dwi Yulianingsih        | 131       |
| 7.6.1    | Teori                             | 131       |
| 7.6.2    | Pemrograman                       | 136       |
| 7.7      | 1174096 - Nico Ekklesia Sembiring | 150       |
| 7.7.1    | Soal Teori                        | 150       |
| 7.7.2    | Praktek Program                   | 154       |
| 7.7.3    | Penanganan Error                  | 167       |
| 7.7.4    | Bukti Tidak Plagiat               | 168       |
|          | Daftar Pustaka                    | 169       |
|          | Index                             | 171       |

# **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[1].

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



# **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

---

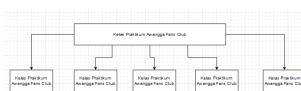
# CHAPTER 7

---

### 7.1 1174027 - Harun Ar - Rasyid

#### 7.1.1 Teori

1. Jelaskan kenapa file teks harus di lakukan tokenizer. dilengkapi dengan ilustrasi atau gambar.



**Gambar 7.1** Illustrasi Tokenizer

2. Jelaskan konsep dasar K Fold Cross Validation pada dataset komentar Youtube pada kode listing 7.7.dilengkapi dengan ilustrasi atau gambar.

```
1 kfolds = StratifiedKFold(n_splits=5)
```

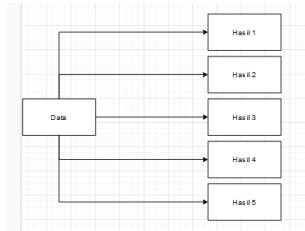
```

2     splits = kfold.split(d, d['CLASS'])
3

```

### **Listing 7.1 K Fold Cross Validation**

Pada koding diatas terdapat variabel kfold yang didalamnya berisi parameter split yang diisikan nilai 5. hal tersebut dimaksudkan untuk membuat pengolahan data akan diulang setiap datanya sebanyak lima kali dengan atribut class sebagai acuan pengolahan datanya. Lalu kemudian akan dihasilkan akurasi dari pengulangan data tersebut sebesar sekian persen tergantung datanya



**Gambar 7.2** Illustrasi K Fold Cross Validation

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

For train digunakan untuk melakukan training atau pelatihan pada data yang sudah dideklarasikan sebelumnya. Sedangkan test in split digunakan untuk membatasi jumlah data yang akan diinputkan atau data yang akan digunakan.

```

In [5]: import numpy as np
.....
from sklearn.model_selection import train_test_split
.....
print(x[1])
[[2 3]
 [2 3]
 [2 3]
 [2 3]
 [2 3]
In [6]: [[x[1]]]
.....
X_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3, random_state=4)
.....
print(x[1])
[[2 3]
 [2 3]
 [2 3]
 [2 3]
 [2 3]

```

**Gambar 7.3** Illustrasi For train dan test in split

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 tersebut adalah membaca isian kolom pada field yang bernama CONTENT sebagai data training dan data testing untuk program

| No | Nama       | Content                                                   |
|----|------------|-----------------------------------------------------------|
| 1  | Mobil      | Kendaraan darat yang biasanya memiliki roda 4             |
| 2  | Motor      | Kendaraan darat yang biasanya memiliki roda 2             |
| 3  | Traktor    | Kendaraan darat yang dipakai untuk memperbaik sawah       |
| 4  | Helikopter | Kendaraan udara yang memiliki baing - baing untuk terbang |

**Gambar 7.4** Illustrasi penggunaan kolom Content

Jelaskan apa maksud dari fungsi `tokenizer = Tokenizer(num_words=2000)` dan `tokenizer.fit_on_texts(train_content)`, dilengkapi dengan ilustrasi atau gambar.

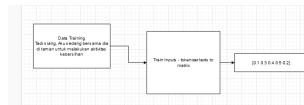
- `tokenizer = Tokenizer(num_words=2000)` digunakan untuk membaca kalimat yang telah dibuat menjadi token sebanyak 2000 kata
- `fit_on_texts` digunakan untuk membuat membaca data token teks yang telah dimasukan kedalam fungsi yaitu fungsi `train_konten`



**Gambar 7.5** Illustrasi fit tokenizer dan num\_word=2000

5. Jelaskan apa maksud dari fungsi `d_train_inputs = tokenizer.texts_to_matrix(train_content, mode='tfidf')` dan `d_test_inputs = tokenizer.texts_to_matrix(test_content, mode='tfidf')`, dilengkapi dengan ilustrasi kode dan atau gambar.

Untuk digunakan sebagai pengubah urutan teks yang tadi telah dilakukan tokenizer menjadi matriks yang berurutan seperti tf idf



**Gambar 7.6** Illustrasi d train inputs = tokenizer.texts to matrix

6. Jelaskan apa maksud dari fungsi `d_train_inputs = d_train_inputs/np.absolute(d_train_inputs).max()` dan `d_test_inputs = d_test_inputs/np.absolute(d_test_inputs).max()`, dilengkapi dengan ilustrasi atau gambar.

Fungsi tersebut digunakan untuk membagi matriks tfidf dengan penentuan maksimum array sepanjang sumbu sehingga akan menimbulkan garis ke bawah dan ke atas yang membentuk gambar v. Lalu hasil tersebut akan dimasukkan ke variabel d train input dan d test input dengan methode absolute. Yang berarti tanpa bilangan negatif.

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

Maksud dari fungsi tersebut yaitu untuk merubah nilai vektor yang ada pada atribut class menjadi bentuk matrix dengan pengurutan berdasarkan data index training dan testing.

8. Jelaskan apa maksud dari fungsi di listing 7.8. Gambarkan ilustrasi Neural Network nya dari model kode tersebut.

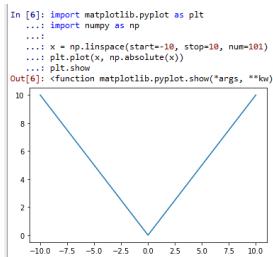
```

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(2))
6 model.add(Activation('softmax'))
7

```

**Listing 7.2** Membuat model Neural Network

model = sequential berarti variabel model berisi method sequential yang berguna untuk searching data dengan menerima parameter atau argumen kunci dengan langkah tertentu untuk mencari data yang telah diolah. Kemudian model akan ditambahkan method add dengan dense yang berarti data - data yang diinputkan akan terhubung, dengan data 612 dan 2000 data kata atau word kemudian model tersebut di masukan fungsi activation dengan rumus atau metode relu. setelah itu data akan di dropout 0.5atau dipangkas sebanyak 50 persen dikarenakan pada pohon bobot terlalu akurat terhadap data.

**Gambar 7.7** Illustrasi d train inputs

```

In [10]: labels = [0,2,1,2,0,1]
...: keras.utils.to_categorical(labels)Out[11]:
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.]], dtype=float32)

```

**Gambar 7.8** Illustrasi train outputs = np utils.to categorical

9. Jelaskan apa maksud dari fungsi di listing 7.9 dengan parameter tersebut.

```

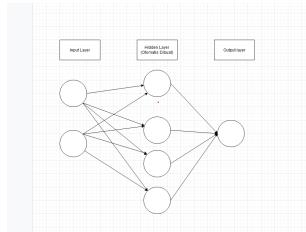
1 model.compile(loss='categorical_crossentropy', optimizer='adamax',
2                 metrics=['accuracy'])
3

```

**Listing 7.3** Compile model

model tersebut kemudian di compile atau di kembalikan kembali fungsi nilainya yang mana akan mengembalikan fungsi nilai loss nya berapa yang diambil dari

fungsi adamax yang berberguna untuk mengetahui nilai lossnya kemudian metrics = accuracy merupakan akurasi dari nilai matrixnya. kemudian terdiri atas beberapa layer atau hidden layer. perbedaan antara deep learning dan DNN atau Deep Neural Network yaitu deep learning merupakan pemakai algoritma dari DNN dan DNN merupakan algoritma yang ada pada deep learning.



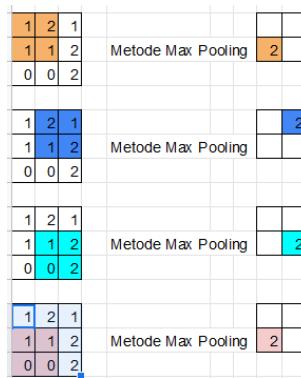
**Gambar 7.9** Illustrasi Neural Network

10. Jelaskan apa itu Deep Learning

Deep learning merupakan salah satu algoritma yang seperti Neural Network yang menggunakan meta data sebagai inputan dan mengolahnya menggunakan layer layer yang tersembunyi.

11. Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning  
Deep Neural Network merupakan algoritma jaringan syaraf yang melakukan pembobotan terhadap data yang sudah ada sebagai acuan untuk data inputan selanjutnya.

12. Jelaskan dengan ilustrasi gambar buatan sendiri(langkah per langkah) bagaimana perhitungan algoritma konvolusi dengan ukuran stride  $(NPM \bmod 3 + 1) \times (NPM \bmod 3 + 1)$  yang terdapat max pooling.(nilai 30)  
sebelum membuat ilustrasi perlu di ketahui apa itu stride, stride adalah acuan atau parameter yang menentukan pergeseran pada filter fixcel. sebagai contoh nilai stride 1 yang berarti filter akan bergeser sebanyak satu fixcel secara vertical dan horizontal. selanjutnya apa itu max pooling contoh pada suatu gambar di tentukan Max Pooling dari  $3 \times 3$  dengan stride 1 yang berarti setiap pergeseran 1 pixcel akan diambil nilai terbesar dari pixcel  $3 \times 3$  tersebut.



**Gambar 7.10** Illustrasi perhitungan stride 1 max pooling

### 7.1.2 Praktek

1. 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 # menimpor libtari CSV untuk mengolah data ber ekstensi csv
3 import csv
4 #kemudian Melakukan import library Image yang berguna untuk dari
    PIL atau Python Imaging Library yang berguna untuk mengolah
    data berupa gambar
5 from PIL import Image as pil_image
6 # kemudian Melakukan import library keras yang menggunakan method
    preprocessing yang digunakan untuk membuat neural network
7 import keras.preprocessing.image

```

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 #Menginisiasi variabel imgs dan classes dengan variabel array
    kosong
3 imgs = []
4 classes = []
5 #membuka file hasy-data-labels.csv yang berada di foleder HASYv2
    yang di inisialisasi menjadi csvfile
6 with open('HASYv2/hasy-data-labels.csv') as csvfile:
7     #Menginisiasi variabel csvreader yang berisi method csv.
        reader yang membaca variabel csvfile
8     csvreader = csv.reader(csvfile)
9     # Menginisiasi variabel i dengan isi 0
10    i = 0
11    # membuat looping pada variabel csvreader
12    for row in csvreader:

```

```

13     # dengan ketentuan jika i lebih kecil daripada o
14     if i > 0:
15         # dibuat variabel img dengan isi keras untuk aktivasi
16         # neural network fungsi yang membaca data yang berada dalam
17         # folder HASYv2 dengan input nilai -1.0 dan 1.0
18         img = keras.preprocessing.image.img_to_array(
19             pil_image.open("HASYv2/" + row[0]))
20         #Pembagian data yang ada pada fungsi img sebanyak
21         # 255.0
22         img /= 255.0
23         # Penambahan nilai baru pada imgs pada row ke 1 2 dan
24         # dilanjutkan dengan variabel img
25         imgs.append((row[0], row[2], img))
26         # Penambahan nilai pada row ke 2 pada variabel
27         classes
28             classes.append(row[2])
29             # penambahan nilai satu pada variabel i
30
31     i += 1

```

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 # Melakukan import library random
3 import random
4 # melakukan random pada vungsi imgs
5 random.shuffle(imgs)
6 # Menginisiasi variabel split_idx dengan nilai integer 80 persen
# dikali dari pengembalian jumlah dari variabel imgs
7 split_idx = int(0.8*len(imgs))
8 # Menginisiasi variabel train dengan isi lebih besar split idx
9 train = imgs[:split_idx]
10 # Menginisiasi variabel test dengan isi lebih kecil split idx
11 test = imgs[split_idx:]

```

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 # Melakukan import library numpy dengan inisial np
3 import numpy as np
4 # Menginisiasi variabel train_input dengan np method asarray yang
# mana membuat array dengan isi row 2 dari data train
5 train_input = np.asarray(list(map(lambda row: row[2], train)))
6 # membuat test_input dengan np method asarray yang mana
# membuat array dengan isi row 2 dari data test
7 test_input = np.asarray(list(map(lambda row: row[2], test)))
8 # Menginisiasi variabel train_output dengan np method asarray
# yang mana membuat array dengan isi row 1 dari data train
9 train_output = np.asarray(list(map(lambda row: row[1], train)))
10 # Menginisiasi variabel test_output dengan np method asarray yang
# mana membuat array dengan isi row 1 dari data test
11 test_output = np.asarray(list(map(lambda row: row[1], test)))

```

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 # Melakukan import library LabelEncode dari sklearn
3 from sklearn.preprocessing import LabelEncoder
4 # Melakukan import library OneHotEncoder dari sklearn
5 from sklearn.preprocessing import OneHotEncoder

```

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
3 # Menginisiasi variabel label_encoder dengan isi LabelEncoder
4 label_encoder = LabelEncoder()
5 # Menginisiasi variabel integer_encoded yang berfungsi untuk
    # Menconvert variabel classes kedalam bentuk integer
6 integer_encoded = label_encoder.fit_transform(classes)

```

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 # Menginisiasi variabel onehot_encoder dengan isi OneHotEncoder
3 onehot_encoder = OneHotEncoder(sparse=False)
4 # mengisi variabel integer_encoded dengan isi integer_encoded
    # yang telah di convert pada fungsi sebelumnya
5 integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
6 # Menconvert variabel integer_encoded kedalam onehot_encoder
7 onehot_encoder.fit(integer_encoded)

```

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 # Menconvert data train output menggunakan variabel
    # label_encoder kedalam variabel train_output_int
3 train_output_int = label_encoder.transform(train_output)
4 # Menconvert variabel train_output_int kedalam fungsi
    # onehot_encoder
5 train_output = onehot_encoder.transform(train_output_int.reshape(
    len(train_output_int), 1))
6 # Menconvert data test_output menggunakan variabel label_encoder
    # kedalam variabel test_output_int
7 test_output_int = label_encoder.transform(test_output)
8 # Menconvert variabel test_output_int kedalam fungsi
    # onehot_encoder

```

```

9 test_output = onehot_encoder.transform(test_output_int.reshape(
10    len(test_output_int), 1))
11 # Menginisiasi variabel num_classes dengan isi variabel
12   label_encoder.classes_
13 num_classes = len(label_encoder.classes_)
14 # mencetak hasil dari nomer Class berupa persen
15 print("Number of classes: %d" % num_classes)

```

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 # Melakukan import library Sequential dari Keras
3 from keras.models import Sequential
4 # Melakukan import library Dense, Dropout, Flatten dari Keras
5 from keras.layers import Dense, Dropout, Flatten
6 # Melakukan import library Conv2D, MaxPooling2D dari Keras
7 from keras.layers import Conv2D, MaxPooling2D

```

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 # Menginisiasi variabel model dengan isian library Sequential
3 model = Sequential()
4 # variabel model di tambahkan library Conv2D tigapuluhan dua bit
      dengan ukuran kernel 3 x 3 dan fungsi penghitungan relu dang
      menggunakan data train_input
5 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
6                  input_shape=np.shape(train_input[0])))
7 # variabel model di tambahkan dengan lib MaxPooling2D dengan
      ketentuan ukuran 2 x 2 pixcel
8 model.add(MaxPooling2D(pool_size=(2, 2)))
9 # variabel model di tambahkan dengan library Conv2D 32bit dengan
      kernel 3 x 3
10 model.add(Conv2D(32, (3, 3), activation='relu'))
11 # variabel model di tambahkan dengan lib MaxPooling2D dengan
      ketentuan ukuran 2 x 2 pixcel
12 model.add(MaxPooling2D(pool_size=(2, 2)))
13 # variabel model di tambahkan library Flatten
14 model.add(Flatten())
15 # variabel model di tambahkan library Dense dengan fungsi tanh
16 model.add(Dense(1024, activation='tanh'))
17 # variabel model di tambahkan library dropout untuk memangkas
      data tree sebesar 50 persen
18 model.add(Dropout(0.5))
19 # variabel model di tambahkan library Dense dengan data dari
      num_classes dan fungsi softmax
20 model.add(Dense(num_classes, activation='softmax'))
21 # mengkompile data model untuk mendapatkan data loss akurasi dan
      optimasi
22 model.compile(loss='categorical_crossentropy', optimizer='adam',
23 metrics=['accuracy'])

```

```

24 # mencetak variabel model kemudian memunculkan kesimpulan berupa
    data total parameter, trainable paremeter dan bukan trainable
    parameter
25 print(model.summary())

```

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.

```

1 # In[11]: import sequential
2 # Melakukan import library keras callbacks
3 import keras.callbacks
4 # Menginisiasi variabel tensorboard dengan isi lib keras
5 tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-
    style')

```

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
        fixcel mnjadi data yang berada pada codigan dibawah.
9     for dense_size in [128, 256, 512, 1024, 2048]:
10         # membuat looping untuk memangkas masing-masing data
            dengan ketentuan 0 persen 25 persen 50 persen dan 75 persen.
11         for dropout in [0.0, 0.25, 0.50, 0.75]:
12             # Menginisiasi variabel model Sequential

```

```

13         model = Sequential()
14         #membuat looping untuk variabel i dengan jarak dari
15         hasil konvolusi.
16         for i in range(conv2d_count):
17             # syarat jika i samadengan bobotnya 0
18             if i == 0:
19                 # Penambahan method add pada variabel model
20                 dengan konvolusi 2 dimensi 32 bit didalamnya dan membuat
21                 kernel dengan ukuran 3 x 3 dan rumus aktifasi relu dan data
22                 shape yang di hitung dari data train.
23                 model.add(Conv2D(32, kernel_size=(3, 3),
24                 activation='relu', input_shape=np.shape(train_input[0])))
25                     # jika tidak
26                     else:
27                         # Penambahan method add pada variabel model
28                         dengan konvolusi 2 dimensi 32 bit dengan ukuran kernel 3 x3
29                         dan fungsi aktivasi relu
30                         model.add(Conv2D(32, kernel_size=(3, 3),
31                         activation='relu'))
32                         # Penambahan method add pada variabel model
33                         dengan isian method Max pooling berdimensi 2 dengan ukuran
34                         fixcel 2 x 2.
35                         model.add(MaxPooling2D(pool_size=(2, 2)))
36                         # merubah feature gambar menjadi 1 dimensi vektor
37                         model.add(Flatten())
38                         # Penambahan method dense untuk pematatan data dengan
39                         ukuran dense di tentukan dengan rumus fungsi tanh.
40                         model.add(Dense(dense_size, activation='tanh'))
41                         # membuat ketentuan jika pemangkasan lebih besar dari
42                         0 persen
43                         if dropout > 0.0:
44                             # Penambahan method dropout pada model dengan
45                             nilai dari dropout
46                             model.add(Dropout(dropout))
47                             # Penambahan method dense dengan fungsi num
48                             classs dan rumus softmax
49                             model.add(Dense(num_classes, activation='softmax'))
50                             # mongkompile variabel model dengan hasi loss
51                             optimasi dan akurasi matrix
52                             model.compile(loss='categorical_crossentropy',
53                             optimizer='adam', metrics=['accuracy'])
54                             # melakukan log pada dir
55                             log_dir = './logs/conv2d_%d-dense_%d-dropout_%.2f' %
56 (conv2d_count, dense_size, dropout)
57                             # Menginisiasi variabel tensorboard dengan isian dari
58                             library keras dan nilai dari lig dir
59                             tensorflow = keras.callbacks.TensorBoard(log_dir=
60 log_dir)
61                             # Menginisiasi variabel start dengan isian dari
62                             library time menggunakan method time
63
64                             start = time.time()
65                             # Penambahan method fit pada model dengan data dari
66                             train input train output nilai batch nilai epoch verbose
67                             nilai 20 persen validation split dan callback dengan nilai
68                             tnsorboard .

```

```

46         model.fit(train_input, train_output, batch_size=32,
47                      epochs=10,
48                               verbose=0, validation_split=0.2, callbacks
49                               =[tensorboard])
50                               # Menginisiasi variabel score dengan nilai evaluasi
51                               dari model menggunakan data tes input dan tes output
52                               score = model.evaluate(test_input, test_output,
53                               verbose=2)
54                               # Menginisiasi variabel end
55                               end = time.time()
56                               # Menginisiasi variabel elapsed
57                               elapsed = end - start
58                               # mencetak hasil perhitungan
59                               print("Conv2D count: %d, Dense size: %d, Dropout: %.2f
60 f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count
61 , dense_size, dropout, score[0], score[1], elapsed))
62                               results.append((conv2d_count, dense_size, dropout,
63 , 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 32bit 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
20 model.add(Flatten())
21 # variabel model di tambahkan library Dense dengan fungsi tanh
22 model.add(Dense(128, activation='tanh'))
23 # variabel model di tambahkan library dropout untuk memangkas
24     data tree sebesar 50 persen
25 model.add(Dropout(0.5))
26 # variabel model di tambahkan library Dense dengan data dari
27     num_classes dan fungsi softmax
28 model.add(Dense(num_classes, activation='softmax'))
29 # mengkompile data model untuk mendapatkan data loss akurasi dan
30     optimasi
31 model.compile(loss='categorical_crossentropy', optimizer='adam',
32               metrics=['accuracy'])

```

```

22 # mencetak variabel model kemudian memunculkan kesimpulan berupa
    data total parameter, trainable paremeter dan bukan trainable
    parameter
23 print(model.summary())

```

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 # melakukan join numpy menggunakan data train_input test_input
3 model.fit(np.concatenate((train_input, test_input)),
4         # kelanjutan data yang di gunakan pada join
        train_output test_output
5         np.concatenate((train_output, test_output)),
6         #menggunakan ukuran 32 bit dan epoch 10
7         batch_size=32, epochs=10, verbose=2)

```

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 #menyimpan model atau mengeksport model yang telah di jalantadi
3 model.save("mathsymbols.model")

```

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 # menyimpan label encoder dengan nama classes.npy
3 np.save('classes.npy', label_encoder.classes_)

```

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 # the code below could be placed in a separate file
3 # mengimpport library keras model
4 import keras.models
5 # Menginisiasi variabel model2 untuk meload model yang telah di
    simpan tadi
6 model2 = keras.models.load_model("mathsymbols.model")
7 # mencetak hasil model2
8 print(model2.summary())

```

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 # Menginisiasi variabel label encoder ke 2 dengan isian fungsi
   label encoder.
3 label_encoder2 = LabelEncoder()
4 # Penambahan method classes dengan data classes yang di eksport
   tadi
5 label_encoder2.classes_ = np.load('classes.npy')
6 # membuat fungsi predict dengan path img
7 def predict(img_path):
8     # Menginisiasi variabel newimg dengan membuat immage menjadi
       array dan membuka data berdasarkan img path
9     newimg = keras.preprocessing.image.img_to_array(pil_image.
       open(img_path))
10    # membagi data yang terdapat pada variabel newimg sebanyak
11      255
12    newimg /= 255.0
13
14    # do the prediction
15    # Menginisiasi variabel predivtion dengan isian variabel
       model2 menggunakan fungsi predic dengan syarat variabel
       newimg dengan data reshape
16    prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
17
18    # figure out which output neuron had the highest score, and
       reverse the one-hot encoding
19    # Menginisiasi variabel inverted dengan label encoder2 dan
       menggunakan argmax untuk mencari skor luaran tertinggi
20    inverted = label_encoder2.inverse_transform([np.argmax(
       prediction)])
21    # mencetak prediksi gambar dan confidence dari gambar.
22    print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
       max(prediction)))

```

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.

```

1 # In[20]: grab an image (we'll just use a random training image
   for demonstration purposes)
2 # mencari prediksi menggunakan fungsi prediksi yang di buat tadi
   dari data di HASYv2/hasy-data/v2-00010.png
3 predict("HASYv2/hasy-data/v2-00010.png")
4 # mencari prediksi menggunakan fungsi prediksi yang di buat tadi
   dari data di HASYv2/hasy-data/v2-00500.png
5 predict("HASYv2/hasy-data/v2-00500.png")
6 # mencari prediksi menggunakan fungsi prediksi yang di buat tadi
   dari data di HASYv2/hasy-data/v2-00700.png
7 predict("HASYv2/hasy-data/v2-00700.png")

```

### 7.1.3 Penanganan Error

1. SS Error

```
File "D:/Kuliah/Semester 6/Kecerdasan Buatan/Bahan/src/1174051/6/1174051.py", line 9,
  import librosa
ModuleNotFoundError: No module named 'librosa'
```

**Gambar 7.11** No Module Name error

## 2. Jenis Error

- No Module

## 3. Cara Penanganan

Dengan cara melakukan instalasi module yang bersangkutan / menginstal library yang digunakan

### 7.1.4 Bukti Tidak Plagiat

**Gambar 7.12** Tidak Melakukan Plagiat Pada Ch 7

## 7.2 1174051 Evietania Charis Sujadi

### 7.2.1 Teori

**7.2.1. Soal No. 1** Kenapa file teks harus dilakukan tokenizer, dilengkapi dengan ilustrasi atau gambar.

Tokenizer adalah proses untuk membagi kalimat menjadi beberapa teks, hal ini sangat di perlukan dalam AI karena nanti setiap teks akan di hitung bobotnya dan akan memunculkan nilai vektor sehingga teks tersebut bisa di gunakan sebagai data untuk memprediksi teks yang muncul dalam satu kalimat sedangkan proses tkenizer merupakan caramembagi bagi teks dari suatu kalimat biasanya pembagi kalimat tersebut merupakan spasi dalam suatu kalimat.

- Ilustrasi Gambar:

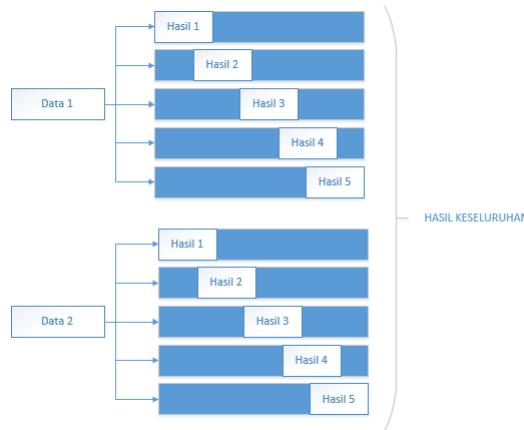
**Gambar 7.13** Tokenizer

**7.2.1.2 Soal No. 2** Konsep dasar K Fold Cross Validation pada dataset komentar Youtube, dilengkapi dengan ilustrasi atau gambar.

```
kfold = StratifiedKFold(n_splits=5)
splits = kfold.split(d, d['CLASS'])
```

pada codingan tersebut terdapat kfold sebagai variabel yang didalamnya terdiri dari split 5 yang berarti pengulangan terhadap pengolahan masing lima kali pada kasus ini terdapat data sebanyak 5 berarti ke lima data tersebut di ulang sebanyak lima kali dengan atribut class sebagai acuan pengolahan datanya kemudian akan dihasilkan akurasi dari pengulangan data tersebut sebesar sekian persen tergantung datanya.

- Ilustrasi Gambar:

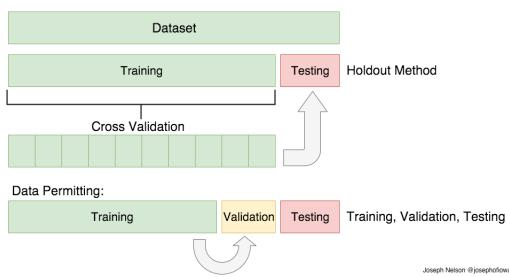


**Gambar 7.14** Konsep dasar K Fold Cross Validation

**7.2.1.3 Soal No. 3** Maksud kode program for train dan test in splits, dilengkapi dengan ilustrasi atau gambar.

Untuk melakukan pengujian atas data pada dataset sudah di tidak terjadi penumpukan dan split. Dimana di setiap class tidak akan muncul id yang sama.

- Ilustrasi Gambar :



**Gambar 7.15** Train dan Test in Split

**7.2.1.4 Soal No. 4** Maksud kode program `train content = d['CONTENT'].iloc[train idx]` dan `test content = d['CONTENT'].iloc[test idx]`, dilengkapi dengan ilustrasi atau gambar.

Untuk mengambil data pada kolom CONTENT yang merupakan bagian dari train idx dan test idx.

- Ilustrasi Gambar:



**Gambar 7.16** Train content

**7.2.1.5 Soal No. 5** Maksud dari fungsi `tokenizer = Tokenizer(num words=2000)` dan `tokenizer.fit on texts(train content)`, dilengkapi dengan ilustrasi atau gambar.

Yang pertama, yaitu fungsi tokenizer ialah mem-vektorisasi jumlah kata yang ingin diubah kedalam bentuk 2000 kata.

Yang kedua, untuk melakukan fit tokenizer untuk dat其实nya dengan data test nya untuk kolom CONTENT saja.

- Ilustrasi Gambar :

## ✓ Tokenization

is a process of breaking up a piece of **text** into many pieces, such as sentences and words. It works by separating **words** using spaces and punctuation.

```
[1]: from nltk.tokenize import sent_tokenize
      sentence = "I love ice cream. I also like steak."
      sent_tokenize(sentence)

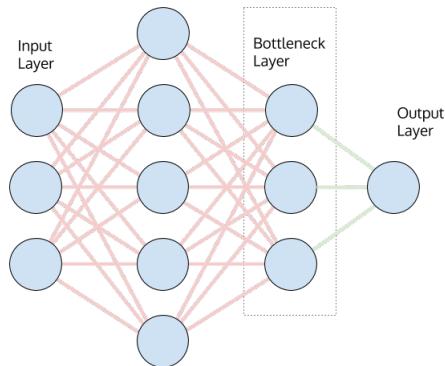
[1]: ['I love ice cream.', 'I also like steak.']}
```

**Gambar 7.17** Tokenizer

**7.2.1.6 Soal No. 6** Maksud dari fungsi d train inputs = tokenizer.texts to matrix(train content, mode='tfidf') dan d test inputs = tokenizer.texts to matrix(test content, mode='tfidf'), dilengkapi dengan ilustrasi atau gambar.

Variabel d train input untuk melakukan tokenizer dari bentuk teks ke matrix dari data train content dengan mode tfidf dan variabel d test inputs sama saja untuk data test.

- Ilustrasi Gambar:

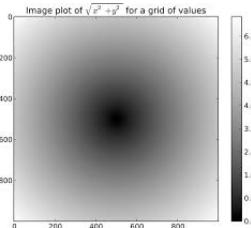


**Gambar 7.18** Train Inputs 1

**7.2.1.7 Soal No. 7** Maksud dari fungsi d train inputs = d train inputs/np.amax(np.absolute(d train inputs)) dan d test inputs = d test inputs/np.amax(np.absolute(d test inputs)) , dilengkapi dengan ilustrasi atau gambar.

Akan membagi matrix tfidf dengan amax untuk mengembalikan array atau maksimum array. Kemudian hasilnya dimasukan dalam variabel d train inputs untuk data train dan d test inputs untuk data test dengan nominal bilangan tanpa ada bilangan negatif dan koma.

- Ilustrasi Gambar :

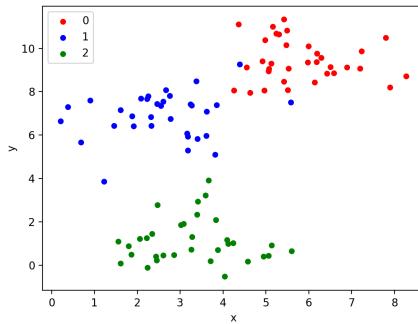


**Gambar 7.19** Train Inputs 2

**7.2.1.8 Soal No. 8** Maksud dari d train outputs = np utils.to categorical(d['CLASS'].iloc[train\_idx]) dan d test outputs = np utils.to categorical(d['CLASS'].iloc[test\_idx])

Fungsi pada kode program tersebut ditujukan untuk melakukan one-hot encoding supaya bisa masuk dan digunakan pada neural network. One-hot encoding diambil dari class yang berarti hanya terdapat 2 neuron, yaitu satu nol(1,0) atau nol satu(0,1) karena pilihannya hanya ada dua.

- Ilustrasi Gambar:



**Gambar 7.20** Compile model

**7.2.1.9 Soal No. 9** Maksud dari listing 7.2.

Fungsi kode program tersebut untuk melakukan pemodelan dengan sequential, membandingkan setiap satu larik elemen dengan cara satu persatu secara beruntun. Terdapat 512 neuron inputan dengan input shape 2000 vektor yang sudah di-normalisasi. Lalu model dilakukan aktivasi dengan fungsi 'relu'. Kemudian pemotongan bobot supaya tidak overfitting sebesar 50 persen dari neuron inputan 512. Lalu pada layer output terdapat 2 neuron outputan yaitu nol (1,0) atau nol satu (0,1). Kemudian outputan tersebut diaktivasi menggunakan fungsi softmax.

#### 7.2.1.10 **Soal No. 10** Maksud dari listing 7.3.

Fungsi kode program tersebut untuk model yang telah dibuat selanjutnya dicompile dengan menggunakan algoritma optimisasi, fungsi loss, dan fungsi metrik.

#### 7.2.1.11 **Soal No. 11** Deep Learning

Deep learning, yang bisa diartikan sebagai rangkaian metode untuk melatih jaringan saraf buatan multi-lapisan. Ternyata, metode ini efektif dalam mengidentifikasi pola dari data. Manakala media membicarakan jaringan saraf, kemungkinan yang dimaksud adalah deep learning.

#### 7.2.1.12 **Soal No. 12** Deep Neural Network, dan apa bedanya dengan Deep Learning

Algoritma DNN (Deep Neural Networks) adalah salah satu algoritma berbasis jaringan saraf yang dapat digunakan untuk pengambilan keputusan. Contoh yang dibahas kali ini adalah mengenai penentuan penerimaan pengajuan kredit sepeda motor baru berdasarkan kelompok data yang sudah ada.

Pembedannya dengan Deep Learning adalah terletak pada kedalaman model, deep learning adalah frasa yang digunakan untuk jaringan saraf yang kompleks.

#### 7.2.1.13 **Soal No. 13** Jelaskan dengan ilustrasi gambar buatan sendiri, bagaimana perhitungan algoritma konvolusi dengan ukuran stride $(NPM \bmod 3 + 1) \times (NPM \bmod 3 + 1)$ yang terdapat max pooling.(nilai 30)

Karena NPM saya 1174051 dan hasil dari  $(NPM \bmod 3) + 1 = 2$ , maka saya menggunakan matrik kernel berukuran  $2 \times 2$ . Misalkan  $f(x,y)$  yang digunakan berukuran  $3 \times 3$  dan kernel atau mask berukuran  $2 \times 2$  adalah sebagai berikut:

Gambar Matriks:

$$F(x,y) = \begin{matrix} 2 & 3 & 5 \\ 1 & 0 & 8 \\ 7 & 2 & 0 \end{matrix} \quad \begin{matrix} 1 & 0 \\ 2 & 4 \end{matrix}$$

**Gambar 7.21** Perhitungan algoritma konvolusi

Penyelesaian dari operasi konvolusi antara  $f(x,y)$  dengan kernel  $g(x,y)$  adalah  $f(x,y) * g(x,y)$

- Tempatkan matrik kernel di sebelah kiri atas, lalu hitung nilai piksel pada posisi (0,0) dari kernel tersebut. Konvolusi dihitung dengan cara berikut :

$$(2 \times 1) + (3 \times 0) + (1 \times 2) + (0 \times 4)$$

Sehingga didapat hasil konvolusi = 4

- Tempatkan matrik kernel di sebelah kanan atas, lalu hitung nilai piksel pada posisi (0,0) dari kernel tersebut. Konvolusi dihitung dengan cara berikut :

$$(3 \times 1) + (5 \times 0) + (0 \times 2) + (8 \times 4)$$

Sehingga didapat hasil konvolusi = 35

- Tempatkan matrik kernel di sebelah kiri bawah, lalu hitung nilai piksel pada posisi (0,0) dari kernel tersebut. Konvolusi dihitung dengan cara berikut :

$$(1 \times 1) + (0 \times 0) + (7 \times 2) + (2 \times 4)$$

Sehingga didapat hasil konvolusi = 23

- Tempatkan matrik kernel di sebelah kanan bawah, lalu hitung nilai piksel pada posisi (0,0) dari kernel tersebut. Konvolusi dihitung dengan cara berikut :

$$(0 \times 1) + (8 \times 0) + (2 \times 2) + (0 \times 4)$$

Sehingga didapat hasil konvolusi = 4

Hasil:

$$\begin{matrix} 4 & 35 \\ 23 & 4 \end{matrix}$$

**Gambar 7.22** Hasil

## 7.2.2 Praktek

**7.2.2.1 Soal No. 1** Jelaskan arti dari setiap baris kode program pada blok # In[1] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:46:02 2020
4
5 @author: User
6 """
7
8 import csv
9 from PIL import Image as pil_image
10 import keras.preprocessing.image

```

▪ Penjelasan:

Baris Code 1: Memasukkan atau mengimport file csv

Baris Code 2: Memasukkan module image sebagai pil\_image dari library PIL

Baris Code 3: Memasukkan atau mengimport fungsi keras.preprocessing.image

▪ Hasil output:

```

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

```

**Gambar 7.23** 1

**7.2.2.2 Soal No. 2** Jelaskan arti dari setiap baris kode program pada blok # In[2] dan hasil luarannya.

▪ Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:25 2020
4
5 @author: User
6 """
7
8 imgs = []
9 classes = []
10 with open('HASYv2/hasy-data-labels.csv') as csvfile:
11     csvreader = csv.reader(csvfile)
12     i = 0
13     for row in csvreader:
14         if i > 0:
15             img = keras.preprocessing.image.img_to_array(
16                 pil_image.open("HASYv2/" + row[0]))
17             # neuron activation functions behave best when input
18             # values are between 0.0 and 1.0 (or -1.0 and 1.0),
19             # so we rescale each pixel value to be in the range
20             # 0.0 to 1.0 instead of 0–255
21             img /= 255.0
22             imgs.append((row[0], row[2], img))
23             classes.append(row[2])
24             i += 1

```

- Penjelasan:

Baris Code 1: Membuat variabel imgs tanpa parameter

Baris Code 2: Membuat variabel classes tanpa parameter

Baris Code 3: Membuka file HASYv2/hasy-data-labels.csv

Baris Code 4: Membuat variabel csvreader untuk pembacaan dari file csv yang dimasukkan

Baris Code 5: Membuat variabel i dengan parameter 0

Baris Code 6: Mengeksekusi baris dari pembacaan csv

Baris Code 7: Mengaplikasikan perintah "if" dengan ketentuan variabel i lebih besar dari angka 0, maka akan dilanjutkan ke perintah berikutnya

Baris Code 8: Membuat variabel img yang mengubah image menjadi bentuk array dari file HASYv2 yang dibuka dengan row berparameter 0.

Baris Code 9: Membuat variabel img atau dengan nilai 255.0

Baris Code 10: Mendefinisikan fungsi imgs.append dimana merupakan proses melampirkan atau menggabungkan data dengan file lain atau set data yang ditentukan dengan 3 parameter yaitu row[0], row[2] dan variabel img.

Baris Code 11: Mendefinisikan fungsi append kembali dari variabel classes dengan parameternya row[2].

Baris Code 12: Mendefinisikan fungsi dimana i variabel i akan ditambah nilainya sehingga akan bernilai 1.

- Hasil output:

```
In [2]: imgs = []
...: classes = []
...: with open('HASYv2/hasy-data-labels.csv') as csvfile:
...:     csvreader = csv.reader(csvfile)
...:     i = 0
...:     for row in csvreader:
...:         if i > 0:
...:             img =
...:             keras.preprocessing.image.img_to_array(pil_image.open("HASYv2/" + row[0]))
...:             # neuron activation functions behave best when input values are
...:             # between 0.0 and 1.0 (or -1.0 and 1.0),
...:             # so we rescale each pixel value to be in the range 0.0 to 1.0
...:             instead of 0-255
...:             img /= 255.0
...:             imgs.append((row[0], row[2], img))
...:             classes.append(row[2])
...:             i += 1
```

**Gambar 7.24** 2

**7.2.2.3 Soal No. 3** Jelaskan arti dari setiap baris kode program pada blok # In[3] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020
4
5 @author: User
6 """
7
8 import random
9 random.shuffle(imgs)
10 split_idx = int(0.8*len(imgs))
11 train = imgs[:split_idx]
12 test = imgs[split_idx:]

```

▪ Penjelasan:

Baris Code 1: Memasukkan module random

Baris Code 2: Melakukan pengocokan atau pengacakan pada module random dengan parameter variabelnya imgs

Baris Code 3: Membagi dan memecah index dalam bentuk integer dengan mengkalikan nilai 0,8 dan fungsi len yang akan mengembalikan jumlah item dalam datanya dari variabel imgs

Baris Code 4: Membuat variabel train yang mengeksekusi imgs dengan pemecahan index awal pada data

Baris Code 5: Membuat variabel test yang mengeksekusi imgs dengan pemecahan index akhir pada data

▪ Hasil output:

```

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

```

**Gambar 7.25** 3

**7.2.2.4 Soal No. 4** Jelaskan arti dari setiap baris kode program pada blok # In[4] dan hasil luarannya.

▪ Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020
4
5 @author: User
6 """
7
8 import numpy as np
9

```

```

10 train_input = np.asarray(list(map(lambda row: row[2], train)))
11 test_input = np.asarray(list(map(lambda row: row[2], test)))
12
13 train_output = np.asarray(list(map(lambda row: row[1], train)))
14 test_output = np.asarray(list(map(lambda row: row[1], test)))

```

▪ Penjelasan:

Baris Code 1: Mengimport library numpy sebagai np

Baris Code 2: Membuat variabel train\_input untuk input menjadi sebuah array dari np menggunakan fungsi list untuk mengkoleksikan data yang dipilih dan diubah. Didalamnya diterapkan fungsi map untuk mengembalikan iterator dari datanya dengan memfungskan lamda pada row dengan parameter [2] untuk membuat objek fungsi menjadi lebih kecil dan mudah dieksekusi dari variabel train.

Baris Code 3: Membuat variabel test\_input dengan fungsi seperti train\_input yang membedakan hanya datanya atau inputan yang diproses berasal dari variabel test

Baris Code 4: Membuat variabel train\_output untuk mengubah keluaran menjadi sebuah array dari np dengan menggunakan fungsi list untuk mengkoleksi data yang dipilih dan diubah. Didalamnya diterapkan fungsi map untuk mengembalikan iterator dari datanya dengan memfungskan lamda pada row dengan parameter[1] untuk membuat objek fungsi menjadi lebih kecil dan mudah dieksekusi dari variabel train.

Baris Code 5: Membuat variabel test\_output dengan fungsi yang sama seperti train\_output yang membedakan hanya datanya atau inputan yang diproses berasal dari variabel test

▪ Hasil output:

```

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

```

Gambar 7.26 4

**7.2.2.5 Soal No. 5** Jelaskan arti dari setiap baris kode program pada blok # In[5] dan hasil luarannya.

▪ Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020

```

```

4
5 @author: User
6 """
7
8 from sklearn.preprocessing import LabelEncoder
9 from sklearn.preprocessing import OneHotEncoder

```

▪ Penjelasan:

Baris Code 1: Memasukkan modul atau fungsi LabelEncoder dari sklearn.preprocessing untuk dapat digunakan menormalkan label dimana label encoder didefinisikan dengan nilai antara 0 dan n\_classes-1.

Baris Code 2: Memasukkan modul atau fungsi OneHotEncoder dari sklearn.preprocessing untuk mendefinisikan fitur input dimana mengambil nilai dalam kisaran [0, maks (nilai)).

▪ Hasil output:

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

Gambar 7.27 5

**7.2.2.6 Soal No. 6** Jelaskan arti dari setiap baris kode program pada blok # In[6] dan hasil luarannya.

▪ Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020
4
5 @author: User
6 """
7
8 label_encoder = LabelEncoder()
9 integer_encoded = label_encoder.fit_transform(classes)

```

▪ Penjelasan:

Baris Code 1: Membuat variabel label\_encoder dengan modul atau fungsi dari LabelEncoder tanpa parameter

Baris Code 2: Membuat variabel integer\_encoded dengan fungsi label\_encoder.fit\_transform() dari variabel classes yang akan mengembalikan beberapa data yang diubah kembali dari variabel label\_encoder.

▪ Hasil output:

```
In [6]: label_encoder = LabelEncoder()
...: integer_encoded = label_encoder.fit_transform(classes)
```

Gambar 7.28 6

**7.2.2.7 Soal No. 7** Jelaskan arti dari setiap baris kode program pada blok # In[7] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020
4
5 @author: User
6 """
7
8 onehot_encoder = OneHotEncoder(sparse=False)
9 integer_encoded = integer_encoded.reshape(len(integer_encoded),
10                                         1)
10 onehot_encoder.fit(integer_encoded)
```

- Penjelasan:

Baris 1: Membuat variabel onehot\_encoder yang memanggil fungsi OneHotEncoder tanpa mengembalikan matriks karena sparse=false.

Baris 2: Membuat variabel integer\_encoded memanggil variabel integer\_encoded pada kode program 6 untuk dieksekusi memberikan bentuk baru ke array tanpa mengubah datanya dari mengembalikan panjang nilai dari integer\_encoded.

Baris 3: Onehotencoding melakukan fitting pada integer\_encoded.

- Hasil output:

```

In [7]: onehot_encoder = OneHotEncoder(sparse=False)
...: integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
...: onehot_encoder.fit(integer_encoded)
D:\Anaconda\lib\site-packages\sklearn\preprocessing\_encoders.py:371: FutureWarning:
The handling of integer data will change in version 0.22. Currently, the categories
are determined based on the range [0, max(values)], while in the future they will be
determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify
"categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories
to integers, then you can now use the OneHotEncoder directly.
warnings.warn(msg, FutureWarning)
Out[7]:
OneHotEncoder(categorical_features=None, categories=None,
               dtype=<class 'numpy.float64'>, handle_unknown='error',
               n_values=None, sparse=False)
```

Gambar 7.29 7

**7.2.2.8 Soal No. 8** Jelaskan arti dari setiap baris kode program pada blok # In[8] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
```

```

3 Created on Mon Apr 13 23:47:28 2020
4
5 @author: User
6 """
7
8 train_output_int = label_encoder.transform(train_output)
9 train_output = onehot_encoder.transform(train_output_int.reshape(
10     len(train_output_int), 1))
11 test_output_int = label_encoder.transform(test_output)
12 test_output = onehot_encoder.transform(test_output_int.reshape(
13     len(test_output_int), 1))
14
15 num_classes = len(label_encoder.classes_)
16 print("Number of classes: %d" % num_classes)

```

▪ Penjelasan:

Baris 1: Membuat variabel train\_output\_int yang mengeksekusi label\_encoder dengan mengubah nilai dari parameter variabel train\_output.

Baris 2: Membuat variabel train\_output yang mengeksekusi variabel onehot\_encoder dari kode program 7 dengan mengubah nilai dari variabel parameter train\_output\_int yang datanya sudah diubah kedalam bentuk array dan panjang nilai dari train\_output\_int telah dikembalikan.

Baris 3: Membuat variabel test\_output\_int yang mengeksekusi label\_encoder dengan mengubah nilai dari parameter variabel test\_output.

Baris 4: Membuat variabel test\_output yang mengeksekusi variabel onehot\_encoder dari kode program 7 dengan mengubah nilai dari variabel parameter test\_output\_int yang datanya sudah diubah kedalam bentuk array dan panjang nilai dari test\_output\_int telah dikembalikan.

Baris 5: Membuat variabel num\_classes untuk mengetahui jumlah class dari label\_encoder

Baris 6: Perintah print digunakan untuk memunculkan hasil dari variabel num\_classes

```

In [8]: train_output_int = label_encoder.transform(train_output)
...: train_output =
...: onehot_encoder.transform(train_output_int.reshape(len(train_output_int), 1))
...: test_output_int = label_encoder.transform(test_output)
...: test_output =
...: onehot_encoder.transform(test_output_int.reshape(len(test_output_int), 1))
...:
...: num_classes = len(label_encoder.classes_)
...: print("Number of classes: %d" % num_classes)
Number of classes: 369

```

Gambar 7.30 8

▪ Hasil output:

**7.2.2.9 Soal No. 9** Jelaskan arti dari setiap baris kode program pada blok # In[9] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:28 2020
4
5 @author: User
6 """
7
8 from keras.models import Sequential
9 from keras.layers import Dense, Dropout, Flatten
10 from keras.layers import Conv2D, MaxPooling2D

```

- Penjelasan:

Baris 1: Melakukan importing fungsi model sequential dari library keras.

Baris 2: Melakukan importing fungsi layer dense, dropout, dan flatten dari library keras.

Baris 3: Melakukan importing fungsi layer Conv2D dan MaxPooling2D dari library keras.

- Hasil output:

```

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

```

**Gambar 7.31 9**

**7.2.2.10 Soal No. 10** Jelaskan arti dari setiap baris kode program pada blok # In[10] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 model = Sequential()
9 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
10                  input_shape=np.shape(train_input[0])))
11 model.add(MaxPooling2D(pool_size=(2, 2)))
12 model.add(Conv2D(32, (3, 3), activation='relu'))
13 model.add(MaxPooling2D(pool_size=(2, 2)))
14 model.add(Flatten())
15 model.add(Dense(1024, activation='tanh'))

```

```
16 model.add(Dropout(0.5))
17 model.add(Dense(num_classes, activation='softmax'))
18
19 model.compile(loss='categorical_crossentropy', optimizer='adam',
20                 metrics=['accuracy'])
21
22 print(model.summary())
```

- Penjelasan:

Baris 1: Melakukan pemodelan Sequential.

Baris 2: Menambahkan Konvolusi 2D dengan 32 filter konvolusi masing-masing berukuran 3x3 dengan algoritam activation relu dengan data dari train input mulai dari baris nol.

Baris 3: Menambahkan Max Pooling dengan matriks 2x2.

Baris 4: Penambahan Konvolusi 2D dengan 32 filter, konvolusi masing-masing berukuran 3x3 dengan algoritam activation relu.

Baris 5 Menambahkan Max Pooling dengan matriks 2x2.

Baris 6: Mendefinisikan inputan dengan 1024 neuron dan menggunakan algoritma tanh untuk activationnya.

Baris 7: Dropout terdiri dari pengaturan secara acak tingkat pecahan unit input ke 0 pada setiap pembaruan selama waktu pelatihan, yang membantu mencegah overfitting sebesar 50% .

Baris 8: Untuk output layer menggunakan data dari variabel num classes dengan fungsi activationnya softmax.

Baris 9: Konfigurasi proses pembelajaran, melalui metode compile, sebelum melatih suatu model.

Baris 10: Menampilkan model yang telah dibuat.

- Hasil output:

```

...: model.add(Flatten())
...: model.add(Dense(1024, 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())
WARNING:tensorflow:From D:\Anaconda\lib\site-packages\tensorflow\python\framework
\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From D:\Anaconda\lib\site-packages\keras\backend
\tensorflow_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with
keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.

Layer (type)          Output Shape         Param #
=====
conv2d_1 (Conv2D)     (None, 30, 30, 32)      896
max_pooling2d_1 (MaxPooling2D) (None, 15, 15, 32)    0
conv2d_2 (Conv2D)     (None, 13, 13, 32)      9248
max_pooling2d_2 (MaxPooling2D) (None, 6, 6, 32)      0
flatten_1 (Flatten)   (None, 1152)           0
dense_1 (Dense)       (None, 1024)          1180672
dropout_1 (Dropout)   (None, 1024)           0
dense_2 (Dense)       (None, 369)            378225
=====
Total params: 1,569,041
Trainable params: 1,569,041
Non-trainable params: 0
=====
```

None

Gambar 7.32 10

**7.2.2.11 Soal No. 11** Jelaskan arti dari setiap baris kode program pada blok # In[11] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:11 2020
4
5 @author: User
6 """
7
8
9 import keras.callbacks
10 tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-
    style')
```

- Penjelasan:

Baris Code 1: Melakukan import library keras.callbacks yang digunakan pada penulisan log untuk TensorBoard, untuk memvisualisasikan grafik dinamis dari pelatihan dan metrik pengujian.

Baris Code 2: Membuat variabel tensorboard untuk mendefinisikan fungsi TensorBoard pada keras.callbacks yang digunakan sebagai alat visualisasi yang disediakan dengan TensorFlow. Dan untuk fungsi log\_dir memanggil data yaitu './logs/mnist-style'

- Hasil output:

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

**Gambar 7.33** 11

**7.2.2.12 Soal No. 12** Jelaskan arti dari setiap baris kode program pada blok # In[12] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:11 2020
4
5 @author: User
6 """
7
8 model.fit(train_input, train_output,
9           batch_size=32,
10          epochs=10,
11          verbose=2,
12          validation_split=0.2,
13          callbacks=[tensorboard])
14
15 score = model.evaluate(test_input, test_output, verbose=2)
16 print('Test loss:', score[0])
17 print('Test accuracy:', score[1])
```

- Penjelasan:

Baris Code 1: Menerapkan fungsi model.fit yang didalamnya memproses train\_input, train\_output

Baris Code 2: Penerapan fungsi yang sama difungsikan batch\_size apabila batch\_size nya tidak ditemukan maka otomatis akan dijadikan nilai 32

Baris Code 3: Penerapan fungsi yang sama, difungsikan epochs dimana perulangan dari berapa kali nilai yang digunakan untuk data, dan jumlahnya ialah 10

Baris Code 4: Mendefinisikan fungsi verbose untuk digunakan sebagai opsi menghasilkan informasi logging dari data yang ditentukan dengan nilai 2

Baris Code 5: Mendefinisikan fungsi validation\_split untuk memecah nilai dari perhitungan validasinya sebesar 0,2. (Fraksi data pelatihan untuk digunakan sebagai data validasi)

Baris Code 6: Mendefinisikan fungsi callbacks dengan parameteranya yang mengeksekusi tensorboard dimana digunakan untuk visualisasikan parameter training, metrik, hiperparameter pada nilai/data yang diproses

Baris Code 7: Mendefinisikan variabel score dengan fungsi evaluate dari model dengan parameter test\_input, tst\_output dan verbose=2 untuk memprediksi output dan input yang diberikan dan kemudian menghitung fungsi metrik yang ditentukan dalam modelnya

Baris Code 8: Mencetak score optimasi dari test dengan ketentuan nilai parameter 0

Baris Code 9: Mencetak score akurasi dari test dengan ketentuan nilai parameter 1

- Hasil output:

```
In [12]: model.fit(train_input, train_output,
...:         batch_size=32,
...:         epochs=10,
...:         verbose=2,
...:         validation_split=0.2,
...:         callbacks=[tensorboard])
...: score = model.evaluate(test_input, test_output, verbose=2)
...: print('Test loss:', score[0])
...: print('Test accuracy:', score[1])
WARNING:tensorflow:From D:\Anaconda\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 107668 samples, validate on 26918 samples
Epoch 1/10
- 2007s - loss: 1.5334 - acc: 0.6294 - val_loss: 0.9824 - val_acc: 0.7285
Epoch 2/10
- 918s - loss: 0.9694 - acc: 0.7321 - val_loss: 0.9162 - val_acc: 0.7494
Epoch 3/10
- 562s - loss: 0.8543 - acc: 0.7556 - val_loss: 0.8883 - val_acc: 0.7530
Epoch 4/10
- 503s - loss: 0.7854 - acc: 0.7699 - val_loss: 0.8441 - val_acc: 0.7666
Epoch 5/10
- 361s - loss: 0.7364 - acc: 0.7796 - val_loss: 0.8491 - val_acc: 0.7663
Epoch 6/10
- 366s - loss: 0.6936 - acc: 0.7899 - val_loss: 0.8522 - val_acc: 0.7692
Epoch 7/10
- 360s - loss: 0.6591 - acc: 0.7962 - val_loss: 0.8580 - val_acc: 0.7668
Epoch 8/10
- 389s - loss: 0.6344 - acc: 0.8017 - val_loss: 0.8496 - val_acc: 0.7654
Epoch 9/10
- 357s - loss: 0.6076 - acc: 0.8072 - val_loss: 0.8597 - val_acc: 0.7638
Epoch 10/10
- 359s - loss: 0.5905 - acc: 0.8116 - val_loss: 0.8889 - val_acc: 0.7637
Test loss: 0.8794204677150739
Test accuracy: 0.7632181175230488
```

Gambar 7.34 12

**7.2.2.13 Soal No. 13** Jelaskan arti dari setiap baris kode program pada blok # In[13] dan hasil luarnya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 import time
9
10 results = []
11 for conv2d_count in [1, 2]:
12     for dense_size in [128, 256, 512, 1024, 2048]:
13         for dropout in [0.0, 0.25, 0.50, 0.75]:
14             model = Sequential()
15             for i in range(conv2d_count):
16                 if i == 0:
```

```

17         model.add(Conv2D(32, kernel_size=(3, 3),
18                         activation='relu', input_shape=np.shape(train_input[0])))
19                         else:
20                             model.add(Conv2D(32, kernel_size=(3, 3),
21                                         activation='relu'))
22                             model.add(MaxPooling2D(pool_size=(2, 2)))
23                             model.add(Flatten())
24                             model.add(Dense(dense_size, activation='tanh'))
25                             if dropout > 0.0:
26                                 model.add(Dropout(dropout))
27                             model.add(Dense(num_classes, activation='softmax'))
28
29             model.compile(loss='categorical_crossentropy',
30                           optimizer='adam', metrics=['accuracy'])
31
32             log_dir = './logs/conv2d_%d-dense_%d-dropout_%.2f' %
33             (conv2d_count, dense_size, dropout)
34             tensorboard = keras.callbacks.TensorBoard(log_dir=
35             log_dir)
36
37             start = time.time()
38             model.fit(train_input, train_output, batch_size=32,
39             epochs=10,
40                     verbose=0, validation_split=0.2, callbacks
41             =[tensorboard])
42             score = model.evaluate(test_input, test_output,
43             verbose=2)
44             end = time.time()
45             elapsed = end - start
46             print("Conv2D count: %d, Dense size: %d, Dropout: %.2
47 f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count
48             , dense_size, dropout, score[0], score[1], elapsed))
49             results.append((conv2d_count, dense_size, dropout,
50             score[0], score[1], elapsed))

```

▪ Penjelasan:

Baris 1: Import atau memasukkan modul time

Baris 2: Variabel result berisikan array kosong

Baris 3: Menggunakan convolution 2D yang berarti memiliki 1 atau 2 layer

Baris 4: Mendefinisikan dense size dengan ukuran 128, 256, 512, 1024, 2048

Baris 5: Mendefinsikan drop out dengan 0, 25%, 50%, dan 75%

Baris 6: Melakukan pemodelan Sequential

Baris 7: Jika ini adalah layer pertama, akan memasukkan bentuk input.

Baris 8: Kalau tidak kita hanya akan menambahkan layer.

Baris 9: Kemudian, setelah menambahkan layer konvolusi, lakukan hal yang sama dengan max pooling.

Baris 10: Lalu, ratakan atau flatten dan menambahkan dense size ukuran apa pun yang berasal dari dense size. Dimana akan selalu menggunakan algoritma tanh

Baris 11: Jika dropout digunakan, akan menambahkan layer dropout. Dropout ini berarti misalnya 50%, bahwa setiap kali memperbarui bobot setelah setiap batch, ada peluang 50% untuk setiap bobot yang tidak akan diperbarui

Baris 12: Menempatkan di antara dua lapisan padat untuk dihindarkan dari melindunginya dari overfitting. Lapisan terakhir akan selalu menjadi jumlah kelas karena itu harus, dan menggunakan softmax. Itu dikompilasi dengan cara yang sama.

Baris 13: Atur direktori log yang berbeda untuk TensorBoard sehingga dapat membedakan konfigurasi yang berbeda.

Baris 14: Variabel start akan memanggil modul time

Baris 15: Melakukan fit atau compile

Baris 16: Melakukan scoring dengan evaluate yang akan menampilkan data loss dan accuracy dari model

Baris 17: 'End' merupakan variabel untuk melihat waktu akhir pada saat pemodelan berhasil dilakukan.

Baris 18: Menampilkan hasil dari skrip diatas

- Hasil output:

```
...:
...:         model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
...:
...:         log_dir = './logs/conv2d_%d-dense_%d-dropout_.2f' % (conv2d_count,
dense_size, dropout)
...:         tensorboard = keras.callbacks.TensorBoard(log_dir=log_dir)
...:
...:         start = time.time()
...:         model.fit(train_input, train_output, batch_size=32, epochs=10,
...:                   verbose=0, validation_split=0.2, callbacks=[tensorboard])
...:         score = model.evaluate(test_input, test_output, verbose=2)
...:         end = time.time()
...:         elapsed = end - start
...:         print("Conv2D count: %d, Dense size: %d, Dropout: %.2f - Loss: %.
2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count, dense_size, dropout, score[0],
score[1], elapsed))
...:         results.append((conv2d_count, dense_size, dropout, score[0],
score[1], elapsed))
Conv2D count: 1, Dense size: 128, Dropout: 0.00 - Loss: 1.13, Accuracy: 0.74, Time: 1540
sec
Conv2D count: 1, Dense size: 128, Dropout: 0.25 - Loss: 0.93, Accuracy: 0.76, Time: 1579
sec
Conv2D count: 1, Dense size: 128, Dropout: 0.50 - Loss: 0.81, Accuracy: 0.77, Time: 1599
sec
Conv2D count: 1, Dense size: 128, Dropout: 0.75 - Loss: 0.82, Accuracy: 0.77, Time: 1627
```

**Gambar 7.35** 13

**7.2.2.14 Soal No. 14** Jelaskan arti dari setiap baris kode program pada blok # In[14] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 model = Sequential()
9 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
10                 input_shape=np.shape(train_input[0])))
11 model.add(MaxPooling2D(pool_size=(2, 2)))
12 model.add(Conv2D(32, (3, 3), activation='relu'))
13 model.add(MaxPooling2D(pool_size=(2, 2)))
14 model.add(Flatten())
15 model.add(Dense(128, activation='tanh'))
16 model.add(Dropout(0.5))
17 model.add(Dense(num_classes, activation='softmax'))
18 model.compile(loss='categorical_crossentropy', optimizer='adam',
19                 metrics=['accuracy'])
20 print(model.summary())
```

- Penjelasan:

Baris 1: Melakukan pemodelan Sequential

Baris 2: Menambahkan Convolutio 2D dengan dmensi 32, dan ukuran matriks 3x3 dengan function aktivasi yang digunakan yaitu relu dan menampilkan input shape

Baris 3: Melakukan Max Pooling 2D dengan ukuran matriks 2x2

Baris 4: Melakukan Convolusi lagi dengan kriteria yang sama tanpa menambahkan input, ini dilakukan untuk mendapatkan data yang terbaik

Baris 5: Flatten digunakan untuk meratakan inputan atau masukkan data

Baris 6: Menambahkan dense input sebanyak 128 neuron dengan menggunakan function aktivasi tanh.

Baris 7: Dropout sebanyak 50% untuk menghindari overfitting

Baris 8: Menambahkan dense pada model untuk output dimana layerini akan menjadi jumlah dari class yang ada.

Baris 9: Mengcompile model yang didefinisikan diatas

Baris 10: Menampilkan ringkasan dari pemodelan yang dilakukan

- Hasil output:

```
In [14]: model = Sequential()
...: model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=np.shape(train_input[0])))
...: model.add(MaxPooling2D(pool_size=(2, 2)))
...: model.add(Conv2D(32, (3, 3), activation='relu'))
...: model.add(MaxPooling2D(pool_size=(2, 2)))
...: model.add(Flatten())
...: model.add(Dense(128, 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())

Layer (type)          Output Shape         Param #
=====
```

| Layer (type)                  | Output Shape       | Param # |
|-------------------------------|--------------------|---------|
| conv2d_3 (Conv2D)             | (None, 30, 30, 32) | 896     |
| max_pooling2d_3 (MaxPooling2) | (None, 15, 15, 32) | 0       |
| conv2d_4 (Conv2D)             | (None, 13, 13, 32) | 9248    |
| max_pooling2d_4 (MaxPooling2) | (None, 6, 6, 32)   | 0       |
| flatten_2 (Flatten)           | (None, 1152)       | 0       |
| dense_3 (Dense)               | (None, 128)        | 147584  |
| dropout_2 (Dropout)           | (None, 128)        | 0       |
| dense_4 (Dense)               | (None, 369)        | 47601   |

```
Total params: 205,329
Trainable params: 205,329
Non-trainable params: 0
```

---

None

Gambar 7.36 14

**7.2.2.15 Soal No. 15** Jelaskan arti dari setiap baris kode program pada blok # In[15] dan hasil luarnanya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 model.fit(np.concatenate((train_input, test_input)),
9           np.concatenate((train_output, test_output)),
10          batch_size=32, epochs=10, verbose=2)
```

- Penjelasan:

Baris 1: Melakukan fit dengan join data train dan test agar dapat dilakukan pelatihan untuk jaringan pada semua data yang dimiliki.

- Hasil output:

```
In [26]: model.fit(np.concatenate((train_input, test_input)),
...:                 np.concatenate((train_output, test_output)),
...:                 batch_size=32, epochs=10, verbose=2)
Epoch 1/10
- 247s - loss: 1.7949 - acc: 0.5843
Epoch 2/10
- 236s - loss: 1.0797 - acc: 0.7064
Epoch 3/10
- 235s - loss: 0.9648 - acc: 0.7308
Epoch 4/10
- 237s - loss: 0.9060 - acc: 0.7434
Epoch 5/10
- 237s - loss: 0.8671 - acc: 0.7519
Epoch 6/10
- 236s - loss: 0.8362 - acc: 0.7573
Epoch 7/10
- 238s - loss: 0.8137 - acc: 0.7631
Epoch 8/10
- 239s - loss: 0.7980 - acc: 0.7652
Epoch 9/10
- 239s - loss: 0.7831 - acc: 0.7692
Epoch 10/10
- 239s - loss: 0.7695 - acc: 0.7724
Out[26]: <keras.callbacks.History at 0x14c1941f5f8>
```

Gambar 7.37 15

**7.2.2.16 Soal No. 16** Jelaskan arti dari setiap baris kode program pada blok # In[16] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 model.save("mathsymbols.model")
```

- Penjelasan:

Baris 1: Menyimpan model yang telah di latih dengan nama mathsymbols.model

- Hasil output:

```
In [22]: model.save("mathsymbols.model")
```

Gambar 7.38 16

**7.2.2.17 Soal No. 17** Jelaskan arti dari setiap baris kode program pada blok # In[17] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 np.save('classes.npy', label_encoder.classes_)
```

- Penjelasan:

Baris 1: Menyimpan label enkoder (untuk membalikkan one-hot encoder) dengan nama classes.npy

- Hasil output:

```
In [23]: np.save('classes.npy', label_encoder.classes_)
```

**Gambar 7.39** 17

**7.2.2.18 Soal No. 18** Jelaskan arti dari setiap baris kode program pada blok # In[18] dan hasil luarnya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:53:35 2020
4
5 @author: User
6 """
7
8 import keras.models
9 model2 = keras.models.load_model("mathsymbols.model")
10 print(model2.summary())
```

- Penjelasan:

Baris 1: Memasukkan atau mengimport models dari librari Keras

Baris 2: Variabel model2 akan memanggil model yang telah disave sebelumnya

Baris 3: Menampilkan ringkasan dari hasil pemodelan

- Hasil output:

```
In [24]: import keras.models
...: model2 = keras.models.load_model("mathsymbols.model")
...: print(model2.summary())

```

| Layer (type)                   | Output Shape       | Param # |
|--------------------------------|--------------------|---------|
| conv2d_3 (Conv2D)              | (None, 30, 30, 32) | 896     |
| max_pooling2d_3 (MaxPooling2D) | (None, 15, 15, 32) | 0       |
| conv2d_4 (Conv2D)              | (None, 13, 13, 32) | 9248    |
| max_pooling2d_4 (MaxPooling2D) | (None, 6, 6, 32)   | 0       |
| flatten_2 (Flatten)            | (None, 1152)       | 0       |
| dense_3 (Dense)                | (None, 128)        | 147584  |
| dropout_2 (Dropout)            | (None, 128)        | 0       |
| dense_4 (Dense)                | (None, 369)        | 47601   |

Total params: 205,329  
Trainable params: 205,329  
Non-trainable params: 0

---

None

**Gambar 7.40** 18

**7.2.2.19 Soal No. 19** Jelaskan arti dari setiap baris kode program pada blok # In[19] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:53:36 2020
4
5 @author: User
6 """
7
8 label_encoder2 = LabelEncoder()
9 label_encoder2.classes_ = np.load('classes.npy')
10
11 def predict(img_path):
12     newimg = keras.preprocessing.image.img_to_array(pil_image.
13         open(img_path))
14     newimg /= 255.0
15
16     # do the prediction
17     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
18
19     # figure out which output neuron had the highest score, and
20     # reverse the one-hot encoding
21     inverted = label_encoder2.inverse_transform([np.argmax(
22         prediction)]) # argmax finds highest-scoring output
23     print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
24         max(prediction)))
```

- Penjelasan:

Baris 1: Memanggil fungsi LabelEncoder

Baris 2: Variabel label encoder akan memanggil class yang disimpan sebelumnya.

Baris 3: Function Predict akan mengubah gambar kedalam bentuk array

Baris 4: Variabel prediction akan melakukan prediksi untuk model2 dengan reshape variabel newimg dengan bentukarray 4D.

Baris 5: Variabel inverted akan mencari nilai tertinggi output dari hasil prediksi tadi

- Hasil output:

```
In [25]: label_encoder2 = LabelEncoder()
...: label_encoder2.classes_ = np.load('classes.npy')
...:
...: def predict(img_path):
...:     newimg =
keras.preprocessing.image.img_to_array(pil_image.open(img_path))
...:     newimg /= 255.0
...:
...:     # do the prediction
...:     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
...:
...:     # figure out which output neuron had the highest score, and reverse the
one-hot encoding
...:     inverted = label_encoder2.inverse_transform([np.argmax(prediction)]) # argmax finds highest-scoring output
...:     print("Prediction: %s, confidence: %.2f" % (inverted[0],
np.max(prediction)))
```

**Gambar 7.41** 19

**7.2.2.20 Soal No. 20** Jelaskan arti dari setiap baris kode program pada blok # In[20] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:54:14 2020
4
5 @author: User
6 """
7
8 predict("HASYv2/hasy-data/v2-00010.png")
9
10 predict("HASYv2/hasy-data/v2-00500.png")
11
12 predict("HASYv2/hasy-data/v2-00700.png")
```

- Penjelasan:

- Baris 1: Melakukan prediksi dari pelatihan dari gambar v2-00010.png  
 Baris 2: Melakukan prediksi dari pelatihan dari gambar v2-00500.png  
 Baris 3: Melakukan prediksi dari pelatihan dari gambar v2-00700.png

- Hasil output:

```
In [26]: predict("HASYv2/hasy-data/v2-00010.png")
...:
...: predict("HASYv2/hasy-data/v2-00500.png")
...:
...: predict("HASYv2/hasy-data/v2-00700.png")
Prediction: \pitchfork, confidence: 0.00
Prediction: \copyright, confidence: 0.00
Prediction: J, confidence: 0.00
```

**Gambar 7.42** 20

### 7.2.3 Penanganan Error

#### 7.2.3.1 Penanganan Error

- Screenshoot:

```
Traceback (most recent call last):
File "<ipython-input-3-fd9abfd31556>", line 1, in <module>
    model.fit(np.concatenate((train_input, test_input)),
NameError: name 'model' is not defined
```

**Gambar 7.43** Error

- Code Error:

NameError: name 'model' is not defined

- Penanganan Error:

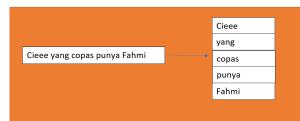
Berdasarkan error maka penyelesaiannya ialah melakukan pendefinisian variabel model sehingga code dapat dijalankan. Lakukan running pada code yang berada diatasnya dimana mendefinisikan variabel model.

## 7.3 1174021 - Muhammad Fahmi

### 7.3.1 Soal Teori

1. Jelaskan kenapa file teks harus di lakukan tokenizer. dilengkapi dengan ilustrasi atau gambar.  
 Karena MTokenizer merupakan proses membagi teks yang dapat berupa kalimat, paragraf atau dokumen menjadi kata-kata atau bagian-bagian tertentu dalam

kalimat tersebut. Sebagai contoh dari kalimat "Cieeee yang copas punya fahmi", kalimat itu menjadi beberapa bagian yaitu "cieeee", "yang", "copas", "punya", "fahmi". Yang menjadi acuan yakni tanda baca dan spasi. Untuk ilustrasi, lihat gambar berikut:

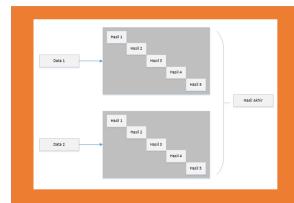


**Gambar 7.44** Teori 1

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

```
1 kfold = StratifiedKFold(n_splits=5)
2
```

Konsep sederhana dari K Fold Cross Validation ialah Pada code tersebut terdapat kfold yang bertujuan untuk melakukan split data menjadi 5 bagian dari dataset komentar Youtube tersebut. Sehingga dari setiap data yang sudah dibagi tersebut akan menghasilkan persentase dari setiap bagiannya, untuk menghasilkan hasil akhir dengan persentase yang cukup baik. Untuk ilustrasi, lihat gambar berikut:



**Gambar 7.45** Teori 2

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

Untuk penjelasannya yaitu For train berfungsi untuk membagi data tersebut menjadi data training. Sedangkan test in splits berfungsi untuk menguji apakah dataset tersebut sudah dibagi menjadi beberapa bagian atau masih menumpuk. Untuk ilustrasi, lihat gambar berikut:

**Gambar 7.46** Teori 3

- 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.

Fungsi dalam kode tersebut berfungsi untuk mengambil data pada kolom atau index CONTENT yang merupakan bagian dari train\_idx dan test\_idx. Contoh sederhananya ketika data telah diubah menjadi data train dan data test maka kita dapat memilihnya untuk ditampilkan pada kolom yang di inginkan. Namun, untuk ilustrasi lihat gambar berikut:

```

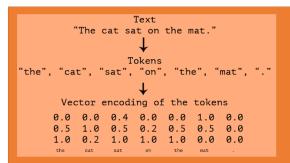
1 import pandas as pd
2
3 mydict = [{ 'a': 1, 'b': 2, 'c': 3, 'd': 4},
4           { 'a': 100, 'b': 200, 'c': 300, 'd': 400},
5           { 'a': 1000, 'b': 2000, 'c': 3000, 'd': 4000 }]
6 df = pd.DataFrame(mydict)
7 df

```

|                       |   |
|-----------------------|---|
| a                     | 1 |
| b                     | 2 |
| c                     | 3 |
| d                     | 4 |
| Name: 0, dtype: int64 |   |

**Gambar 7.47** Teori 4

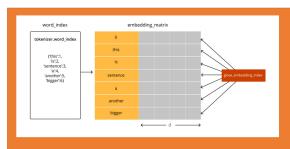
- Jelaskan apa maksud dari fungsi tokenizer = Tokenizer(num words=2000) dan tokenizer.fit on texts(train content), dilengkapi dengan ilustrasi atau gambar. Fungsi tokenizer ini berfungsi untuk melakukan vektorisasi data kedalam bentuk token sebanyak 2000 kata. Dan selanjutnya akan melakukan fit tokenizer hanya untuk data training saja tidak dengan data testingnya. Untuk ilustrasi lihat gambar berikut:



Gambar 7.48 Teori 5

6. Jelaskan apa maksud dari fungsi `d train inputs = tokenizer.texts to matrix(train content, mode='tfidf')` dan `d test inputs = tokenizer.texts to matrix(test content, mode='tfidf')`, dilengkapi dengan ilustrasi kode dan atau gambar.

Maksud dari baris diatas ialah untuk memasukkan text ke sebuah matrix dengan mode tfidf dan menginputkan data testing untuk di terjemahkan ke sebuah matriks. Untuk ilustrasi, lihat gambar berikut:



Gambar 7.49 Teori 6

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

Fungsi `np.amax` adalah nilai Maksimal. Jika sumbu tidak ada, hasilnya adalah nilai skalar. Jika sumbu diberikan, hasilnya adalah array dimensi `a.ndim - 1`. Untuk ilustrasi, lihat gambar berikut:

```

>>> a = np.arange(4).reshape((2,2))
>>> a
array([[0, 1],
       [2, 3]])
>>> np.amax(a)
# Maximum of the flattened array
3
>>> np.amax(a, axis=0) # Maxima along the first axis
array([1, 1])
>>> np.amax(a, axis=1) # Maxima along the second axis
array([1, 3])
>>> a > 1, a > 1, where=[False, True], initial=-1, axis=0)
array([-1, 3])
>>> b = np.arange(5, dtype=float)
>>> b[2] = np.nan
>>> b
array([ 0. ,  1. ,  nan,  3. ,  4. ])
>>> np.sum(b, where=np.isnan(b), initial=-1)
4.0
>>> np.sum(b)
4.0

```

Gambar 7.50 Teori 7

8. Jelaskan apa maksud fungsi dari `d train outputs = np utils.to categorical(d['CLASS'].iloc[idx])` dan `d test outputs = np utils.to categorical(d['CLASS'].iloc[test idx])` dalam kode program, dilengkapi dengan ilustrasi atau gambar.

Fungsi dari baris kode tersebut ialah membuat train outputs dengan kategori dari class lalu dengan ketentuan iloc train idx. Kemudian membuat keluaran sebagai output. Untuk ilustrasi, lihat gambar berikut:

```

>>> Y_train
array([[-1.,  0.,  0.],
       [ 0., -1.,  0.],
       [ 0.,  0., -1.],
       [-1., -1.,  0.],
       [ 0., -1., -1.],
       [ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.],
       [ 1.,  1.,  0.],
       [ 1.,  0., -1.],
       [ 0.,  1., -1.],
       [ 1., -1.,  0.],
       [-1.,  1.,  0.],
       [ 0.,  0., -1.],
       [ 1.,  0.,  1.],
       [ 0.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1., -1.],
       [-1., -1., -1.]])
>>> Y_train[0:10]
array([-1.,  0.,  0.])
array([ 0., -1.,  0.])
array([ 0.,  0., -1.])
array([-1., -1.,  0.])
array([ 0., -1., -1.])
array([-1.,  0.,  0.])
array([ 0.,  1.,  0.])
array([ 0.,  0.,  1.])
array([ 1.,  1.,  0.])
array([ 1.,  0., -1.])
array([ 0.,  1., -1.])
array([ 1., -1.,  0.])
array([-1.,  1.,  0.])
array([ 0.,  0., -1.])
array([ 1.,  0.,  1.])
array([ 0.,  1.,  1.])
array([ 1.,  1.,  1.])
array([ 1.,  1., -1.])
array([-1., -1., -1.])
>>> np_utils.to_categorical(Y_train)
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.],
       [ 1.,  1.,  0.],
       [ 1.,  0., -1.],
       [ 0.,  1., -1.],
       [ 1., -1.,  0.],
       [-1.,  1.,  0.],
       [ 0.,  0., -1.],
       [ 1.,  0.,  1.],
       [ 0.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1., -1.],
       [-1., -1., -1.]])

```

**Gambar 7.51** Teori 8

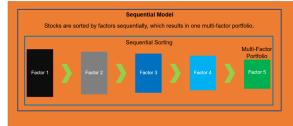
9. Jelaskan apa maksud dari fungsi di listing 7.2. Gambarkan ilustrasi Neural Network nya dari model kode tersebut.

```

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(2))
6 model.add(Activation('softmax'))

```

Fungsi dari baris kode tersebut ialah model perlu mengetahui bentuk input apa yang harus diharapkan. Untuk alasan ini, lapisan pertama dalam model Sequential (dan hanya yang pertama, karena lapisan berikut dapat melakukan inferensi bentuk otomatis) perlu menerima informasi tentang bentuk inputnya.



**Gambar 7.52** Teori 9

10. Jelaskan apa maksud dari fungsi di listing 7.3 dengan parameter tersebut.

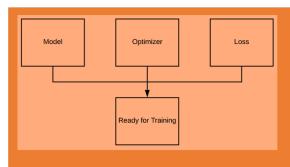
```

1 model.compile(loss='categorical_crossentropy', optimizer='adamax',
2                 metrics=['accuracy'])

```

Fungsi dari baris kode tersebut ialah bisa meneruskan nama fungsi loss yang ada, atau melewati fungsi simbolis TensorFlow yang mengembalikan skalar untuk setiap titik data dan mengambil dua argumen y\_true: True label. dan y\_pred:

Prediksi. Tujuan yang dioptimalkan sebenarnya adalah rata-rata dari array output di semua titik data.



**Gambar 7.53** Teori 10

11. Jelaskan apa itu Deep Learning.

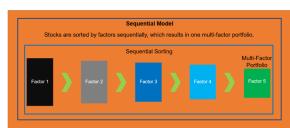
Deep Learning adalah salah satu cabang dari ilmu machine learning yang terdiri dari algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam machine learning dapat digunakan baik untuk supervised learning dan unsupervised learning.

12. Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning.

DNN adalah salah satu algoritma berbasis jaringan saraf tiruan yang memiliki dari 1 lapisan saraf tersembunyi yang dapat digunakan untuk pengambilan keputusan. Perbedaannya dengan deep learning, yakni: DNN dapat menentukan dan mencerna karakteristik tertentu di suatu rangkaian data, kapabilitas lebih kompleks untuk mempelajari, mencerna, dan mengklasifikasikan data, serta dibagi ke dalam berbagai lapisan dengan fungsi yang berbeda-beda.

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.

1174021 mod3+1 x 1174021 mod3+1 = 2 x 2, adapun ilustrasi gambar nya sebagai berikut :



**Gambar 7.54** Teori 9

### 7.3.2 Praktek Program

#### 1. Soal 1

```

1 # In[1]:import lib
2 # mengimport library CSV untuk mengolah data ber ekstensi csv
3 import csv
4 #kemudian mengimport librari Image yang berguna untuk dari PIL
      atau Python Imaging Library yang berguna untuk mengolah data
      berupa gambar
5 from PIL import Image as pil_image
6 # kemudian mengimport librari keras yang menggunakan method
      preprocessing yang digunakan untuk membuat neutal network
7 import keras.preprocessing.image

```

Kode di atas menjelaskan tentang library yang akan di pakai yaitu import file csv, lalu load module pil image dan juga import library keras, hasilnya adalah sebagai berikut:

```

In [1]: import CSV
        #mengimport librari Image yang berguna untuk dari PIL atau Python
        # imaging atau Python Imaging Library yang berguna untuk mengolah data berupa gambar
        from PIL import Image as PIL_image
        # kemudian mengimport librari keras yang menggunakan method preprocessing yang
        # digunakan untuk membuat neutral network
        import keras.preprocessing.image
Using TensorFlow backend.

```

**Gambar 7.55** Hasil Soal 1.

#### 2. Soal 2

```

1 # In[2]:load all images (as numpy arrays) and save their classes
2 #membuat variabel imgs dengan variabel kosong
3 imgs = []
4 #membuat variabel classes dengan variabel kosong
5 classes = []
6 #membuaka file hasy-data-labels.csv yang berada di foleder HASYv2
      yang di inisialisasi menjadi csvfile
7 with open('HASYv2/hasy-data-labels.csv') as csvfile:
8     #membuat variabel csvreader yang berisi method csv.reader
      yang membaca variabel csvfile
9     csvreader = csv.reader(csvfile)
10    # membuat variabel i dengan isi 0
11    i = 0
12    # membuat looping pada variabel csvreader
13    for row in csvreader:
14        # dengan ketentuan jika i lebihkecil daripada o
15        if i > 0:
16            # dibuat variabel img dengan isi keras untuk aktivasi
              neural network fungsi yang membaca data yang berada dalam
              folder HASYv2 dengan input nilai -1.0 dan 1.0
17            img = keras.preprocessing.image.img_to_array(
18                pil_image.open("HASYv2/" + row[0]))
19                # neuron activation functions behave best when input
                  values are between 0.0 and 1.0 (or -1.0 and 1.0),
                  # so we rescale each pixel value to be in the range
                  0.0 to 1.0 instead of 0-255

```

```

20         #membagi data yang ada pada fungsi img sebanyak 255.0
21         img /= 255.0
22         # menambah nilai baru pada imgs pada row ke 1 2 dan
23         dilanjutkan dengan variabel img
24         imgs.append((row[0], row[2], img))
25         # menambahkan nilai pada row ke 2 pada variabel
26         classes
27             classes.append(row[2])
28             # penambahan nilai satu pada variabel i
29             i += 1

```

Kode di atas akan menampilkan hasil dari proses load dataset dari HASYv2 sebagai file csv, membuat variabel i dengan parameter 0, lalu perintah perulangan if dengan  $i \leq 0$ , variabel img dengan nilai 255.0, imgs.append yaitu proses melampirkan atau menggabungkan data dengan file. Berikut adalah hasil setelah saya lakukan running dan pembacaan file audio :



**Gambar 7.56** Hasil Soal 2.

### 3. Soal 3

```

1 # In [3]: shuffle the data, split into 80% train , 20% test
2 # mengimport library random
3 import random
4 # melakukan random pada vungsi imgs
5 random.shuffle(imgs)
6 # membuat variabel split_idx dengan nilai integer 80 persen
# dikali dari pengembalian jumlah dari variabel imgs
7 split_idx = int(0.8*len(imgs))
8 # membuat variabel train dengan isi lebih besar split idx
9 train = imgs[:split_idx]
10 # membuat variabel test dengan isi lebih kecil split idx
11 test = imgs[split_idx:]

```

Perintah import random yaitu sebagai library untuk menghasilkan nilai acak, disini kita membuat data menjadi 2 bagian yaitu 80% sebagai training dan 20% sebagai data testing. Hasilnya adalah sebagai berikut :

```

In [4]: import random
.... random.shuffle(imgs)
.... split_idx = int(0.8*len(imgs))
.... train = imgs[:split_idx]
.... test = imgs[split_idx:]

```

**Gambar 7.57** Hasil Soal 3.

### 4. Soal 4

```

1 # In[4]:
2 # mengimport librari numpy dengan inisial np
3 import numpy as np
4 # membuat variabel train_input dengan np method asarray yang mana
   membuat array dengan isi row 2 dari data train
5 train_input = np.asarray(list(map(lambda row: row[2], train)))
6 # membuat test_input dengan np method asarray yang mana
   membuat array dengan isi row 2 dari data test
7 test_input = np.asarray(list(map(lambda row: row[2], test)))
8 # membuat variabel train_output dengan np method asarray yang mana
   membuat array dengan isi row 1 dari data train
9 train_output = np.asarray(list(map(lambda row: row[1], train)))
10 # membuat variabel test_output dengan np method asarray yang mana
    membuat array dengan isi row 1 dari data test
11 test_output = np.asarray(list(map(lambda row: row[1], test)))

```

Kode di atas dapat digunakan untuk melakukan fungsi yang sebelumnya telah kita lakukan yaitu dengan membuat variabel baru untuk menampung data train input dan test input lalu keluarannya sebagai output. Hasilnya adalah sebagai berikut :

```

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

```

**Gambar 7.58** Hasil Soal 4.

## 5. Soal 5

```

1 # In[5]: import encoder and one hot
2 # mengimport librari LabelEncode dari sklearn
3 from sklearn.preprocessing import LabelEncoder
4 # mengimport librari OneHotEncoder dari sklearn
5 from sklearn.preprocessing import OneHotEncoder

```

Kode diatas untuk melakukan import library yang berfungsi sebagai label encoder dan one hot encoder. Hasilnya adalah sebagai berikut :

```

In [6]: from sklearn.preprocessing import LabelEncoder
... from sklearn.preprocessing import OneHotEncoder

```

**Gambar 7.59** Hasil Soal 5.

## 6. Soal 6

```

1 # In[6]: convert class names into one-hot encoding
2
3 # membuat variabel label_encoder dengan isi LabelEncoder
4 label_encoder = LabelEncoder()
5 # membuat variabel integer_encoded yang berfungsi untuk
   mengkonvert variabel classes kedalam bentuk integer
6 integer_encoded = label_encoder.fit_transform(classes)

```

Kode diatas berfungsi untuk melakukan convert class names ke one-hot encoding. Hasilnya adalah sebagai berikut :

```
In [47]: label_encoder = LabelEncoder()
... # membuat variabel integer_encoded yang
berfungsi untuk mengkonvert variabel classes kedalam
bentuk integer
... integer_encoded =
label_encoder.fit_transform(classes)
```

**Gambar 7.60** Hasil Soal 6.

## 7. Soal 7

```
1 # In[7]: then convert integers into one-hot encoding
2 # membuat variabel onehot_encoder dengan isi OneHotEncoder
3 onehot_encoder = OneHotEncoder(sparse=False)
4 # mengisi variabel integer_encoded dengan isi integer_encoded
# yang telah di convert pada fungsi sebelumnya
5 integer_encoded = integer_encoded.reshape(len(integer_encoded),
1)
6 # mengkonvert variabel integer_encoded kedalam onehot_encoder
7 onehot_encoder.fit(integer_encoded)
```

Kode di atas befungsi untuk melakukan convert integers ke fungsi one hot encoding. Hasilnya adalah sebagai berikut :

```
In [8]: onehot_encoder = OneHotEncoder(sparse=False)
... integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
... onehot_encoder.fit(integer_encoded)
Out[8]:
OneHotEncoder(categories='auto', drop=None, dtype=<class 'numpy.float64'>,
handle_unknown='error', sparse=False)
```

**Gambar 7.61** Hasil Soal 7.

## 8. Soal 8

```
1 # In[8]: convert train and test output to one-hot
2 # mengkonvert data train output menggunakan variabel
# label_encoder kedalam variabel train_output_int
3 train_output_int = label_encoder.transform(train_output)
4 # mengkonvert variabel train_output_int kedalam fungsi
# onehot_encoder
5 train_output = onehot_encoder.transform(train_output_int.reshape(
len(train_output_int), 1))
6 # mengkonvert data test_output menggunakan variabel label_encoder
# kedalam variabel test_output_int
7 test_output_int = label_encoder.transform(test_output)
8 # mengkonvert variabel test_output_int kedalam fungsi
# onehot_encoder
9 test_output = onehot_encoder.transform(test_output_int.reshape(
len(test_output_int), 1))
10 # membuat variabel num_classes dengan isi variabel label_encoder
# dan classes
11 num_classes = len(label_encoder.classes_)
12 # mencetak hasil dari nomer Class berupa persen
13 print("Number of classes: %d" % num_classes)
```

Kode di atas befungsi untuk melakukan convert train dan test output ke fungsi one hot encoding. Hasilnya adalah sebagai berikut :

```
In [9]: train_output_int = label_encoder.transform(train_output)
train_output_int
train_output_int = np.reshape(train_output_int, (train_output_int.shape[0], 3))
test_output_int = label_encoder.transform(test_output)
test_output_int
test_output_int = np.reshape(test_output_int, (test_output_int.shape[0], 3))
num_classes = len(label_encoder.classes_)
num_classes
Number of classes: 369
```

**Gambar 7.62** Hasil Soal 8.

## 9. Soal 9

```
1 # In[9]: import sequential
2 # mengimport librari Sequential dari Keras
3 from keras.models import Sequential
4 # mengimport librari Dense, Dropout, Flatten dari Keras
5 from keras.layers import Dense, Dropout, Flatten
6 # mengimport librari Conv2D, MaxPooling2D dari Keras
7 from keras.layers import Conv2D, MaxPooling2D
```

Kode di atas befungsi untuk melakukan import sequential dari library keras. Hasilnya adalah sebagai berikut :

```
In [10]: from keras.models import Sequential
... from keras.layers import Dense, Dropout, Flatten
... from keras.layers import Conv2D, MaxPooling2D
```

**Gambar 7.63** Hasil Soal 9.

## 10. Soal 10

```
1 # In[10]: desain jaringan
2 # membuat variabel model dengan isian librari Sequential
3 model = Sequential()
4 # variabel model di tambahkan librari Conv2D tigapuluhan dua bit
# dengan ukuran kernel 3 x 3 dan fungsi penghitungan relu dang
# menggunakan data train_input
5 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
6                  input_shape=np.shape(train_input[0])))
7 # variabel model di tambahkan dengan lib MaxPooling2D dengan
# ketentuan ukuran 2 x 2 pixcel
8 model.add(MaxPooling2D(pool_size=(2, 2)))
9 # variabel model di tambahkan dengan librari Conv2D 32bit dengan
# kernel 3 x 3
10 model.add(Conv2D(32, (3, 3), activation='relu'))
11 # variabel model di tambahkan dengan lib MaxPooling2D dengan
# ketentuan ukuran 2 x 2 pixcel
12 model.add(MaxPooling2D(pool_size=(2, 2)))
13 # variabel model di tambahkan librari Flatten
14 model.add(Flatten())
15 # variabel model di tambahkan librari Dense dengan fungsi tanh
16 model.add(Dense(1024, activation='tanh'))
```

```

17 # variabel model di tambahkan librari dropout untuk memangkas
    data tree sebesar 50 persen
18 model.add(Dropout(0.5))
19 # variabel model di tambahkan librari Dense dengan data dari
    num_classes dan fungsi softmax
20 model.add(Dense(num_classes, activation='softmax'))
21 # mengkompile data model untuk mendapatkan data loss akurasi dan
    optimasi
22 model.compile(loss='categorical_crossentropy', optimizer='adam',
    metrics=['accuracy'])
23 # mencetak variabel model kemudian memunculkan kesimpulan berupa
    data total parameter, trainable paremeter dan bukan trainable
    parameter
24 print(model.summary())
25

```

Kode di atas befungsi untuk melihat detail desain pada jaringan model yang telah di definisikan. Hasilnya adalah sebagai berikut :

```

        .....model.add(conv2d_0, kernel_size=(3, 3), activation="relu")
        .....model.add(conv2d_0, kernel_size=(3, 3), activation="relu"))
        .....model.add(conv2d_0, (3, 3), activation="relu"))
        .....model.add(maxpooling2d_0, pool_size=(2, 2))
        .....model.add(dense_0, activation="tanh"))
        .....model.add(dense_0)
        .....model.add(Dense(num_classes, activation="softmax"))
        .....model.compile(loss='categorical_crossentropy', optimizer='adam',
    metrics=['accuracy'])
print(model.summary())
Model: sequential_5
Layer (Type)          Output Shape         Param #
conv2d_0 (Conv2D)      (None, 10, 10, 32)     996
max_pooling2d_0 (MaxPooling2D) (None, 5, 5, 32)     0
conv2d_1 (Conv2D)      (None, 5, 5, 32)     9248
max_pooling2d_1 (MaxPooling2D) (None, 6, 6, 32)     0
flatten_0 (Flatten)   (None, 352)       0
dense_0 (Dense)       (None, 28)        147584
dropout_4 (Dropout)   (None, 28)        0
dense_10 (Dense)     (None, 69)        47601
Total params: 205,329
Trainable params: 205,329
Non-trainable params: 0

```

**Gambar 7.64** Hasil Soal 10.

## 11. Soal 11

```

1 # In[11]: import sequential
2 # mengimport librari keras callbacks
3 import keras.callbacks
4 # membuat variabel tensorboard dengan isi lib keras
5 tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-
    style')

```

Kode di atas befungsi untuk melakukan load callbacks pada library keras. Hasilnya adalah sebagai berikut :

```

In [12]: import keras.callbacks
tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-style')

```

**Gambar 7.65** Hasil Soal 11.

## 12. Soal 12

```

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])

```

Kode di atas befungsi untuk melakukan load epoch yang akan di training dan diberikan hasil selama 10 kali epoch. Hasilnya adalah sebagai berikut :

```

Epoch 0/10
  37/37 [==============================] - 0s - loss: 1.5816 - val_loss: 1.4932 - val_accuracy: 0.7182
WARNING:tensorflow:From C:\Users\Arafat\PycharmCond3\white package\keras\utils\batch_
processing.py:57: The name tf.summary is deprecated. Please use tf.compat.v1.summary instead.
Epoch 1/10
  37/37 [==============================] - 0s - loss: 1.4981 - accuracy: 0.7248 - val_loss: 0.9669 - val_accuracy: 0.7523
Epoch 2/10
  37/37 [==============================] - 0s - loss: 0.8755 - accuracy: 0.7477 - val_loss: 0.8582 - val_accuracy: 0.7962
Epoch 3/10
  37/37 [==============================] - 0s - loss: 0.8228 - accuracy: 0.7632 - val_loss: 0.8136 - val_accuracy: 0.7929
Epoch 4/10
  37/37 [==============================] - 0s - loss: 0.7751 - accuracy: 0.7716 - val_loss: 0.8429 - val_accuracy: 0.7818
Epoch 5/10
  37/37 [==============================] - 0s - loss: 0.7356 - accuracy: 0.7804 - val_loss: 0.8562 - val_accuracy: 0.7842
Epoch 6/10
  37/37 [==============================] - 0s - loss: 0.7264 - accuracy: 0.7819 - val_loss: 0.8566 - val_accuracy: 0.7854
Epoch 7/10
  37/37 [==============================] - 0s - loss: 0.7165 - accuracy: 0.7824 - val_loss: 0.8549 - val_accuracy: 0.7879
Epoch 8/10
  37/37 [==============================] - 0s - loss: 0.6532 - accuracy: 0.7975 - val_loss: 0.8535 - val_accuracy: 0.7981
Epoch 9/10
  37/37 [==============================] - 0s - loss: 0.6113 - accuracy: 0.8021 - val_loss: 0.8792 - val_accuracy: 0.8013
test loss: 0.7094048188010123
test accuracy: 0.7055150000000001

```

Gambar 7.66 Hasil Soal 12.

### 13. Soal 13

```

1 # In[13]: try various model configurations and parameters to find
   the best
2 # mengimport librari time
3 import time
4 #membuat 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
       fixcel mnjadi data yang berada pada codigan dibawah.
9     for dense_size in [128, 256, 512, 1024, 2048]:
10        # membuat looping untuk memangkas masing-masing data
           dengan ketentuan 0 persen 25 persen 50 persen dan 75 persen.
11        for dropout in [0.0, 0.25, 0.50, 0.75]:
12            # membuat variabel model Sequential
13            model = Sequential()
14            #membuat looping untuk variabel i dengan jarak dari
           hasil konvolusi.
15            for i in range(conv2d_count):
16                # syarat jika i samadengan bobotnya 0
17                if i == 0:
18                    # menambahkan method add pada variabel model
           dengan konvolusi 2 dimensi 32 bit didalamnya dan membuat

```

```
kernel dengan ukuran 3 x 3 dan rumus aktifasi relu dan data
shape yang di hitung dari data train.
    model.add(Conv2D(32, kernel_size=(3, 3),
activation='relu', input_shape=np.shape(train_input[0])))
        # jika tidak
    else:
        # menambahkan method add pada variabel model
dengan konvolusi 2 dimensi 32 bit dengan ukuran kernel 3 x3
dan fungsi aktivasi relu
    model.add(Conv2D(32, kernel_size=(3, 3),
activation='relu'))
        # menambahkan method add pada variabel model
dengan isian method Max pooling berdimensi 2 dengan ukuran
fixcel 2 x 2.
    model.add(MaxPooling2D(pool_size=(2, 2)))
        # merubah feature gambar menjadi 1 dimensi vektor
    model.add(Flatten())
        # menambahkan method dense untuk pematatan data
dengan ukuran dense di tentukan dengan rumus fungsi tanh.
    model.add(Dense(dense_size, activation='tanh'))
        # membuat ketentuan jika pemangkasan lebih besar dari
0 persen
    if dropout > 0.0:
        # menambahkan method dropout pada model dengan
nilai dari dropout
        model.add(Dropout(dropout))
        # menambahkan method dense dengan fungsi num
classs dan rumus softmax
        model.add(Dense(num_classes, activation='softmax'))
        # mongkompile variabel model dengan hasi loss
optimasi dan akurasi matrix
        model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
        # melakukan log pada dir
        log_dir = './logs/conv2d_%d-dense_%d-dropout_%.2f' %
(conv2d_count, dense_size, dropout)
        # membuat variabel tensorflow dengan isian dari
librari keras dan nilai dari lig dir
        tensorboard = keras.callbacks.TensorBoard(log_dir=
log_dir)
        # membuat variabel start dengan isian dari librari
time menggunakan method time

        start = time.time()
        # menambahkan method fit pada model dengan data dari
train input train output nilai batch nilai epoch verbose
nilai 20 persen validation split dan callback dengan nilai
tnsorboard.
        model.fit(train_input, train_output, batch_size=32,
epochs=10,
            verbose=0, validation_split=0.2, callbacks
=[tensorboard])
        # membuat variabel score dengan nilai evaluasi dari
model menggunakan data tes input dan tes output
        score = model.evaluate(test_input, test_output,
verbose=2)
```

```
50     # membuat variabel end  
51     end = time.time()  
52     # membuat variabel elapsed  
53     elapsed = end - start  
54     # mencetak hasil perhitungan  
55     print("Conv2D count: %d, Dense size: %d, Dropout: %.2f  
      f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count  
      , dense_size, dropout, score[0], score[1], elapsed))  
56     results.append((conv2d_count, dense_size, dropout,  
      score[0], score[1], elapsed))
```

Kode di atas befungsi untuk melakukan konfigurasi model dengan parameter yang ditentukan dan mencoba mencari data yang terbaik. Hasilnya adalah sebagai berikut :

```
conv2d count: 1, dense size: 128, Dropout: 0.00 Loss: 1.19, Accuracy: 0.74, Time: 413 sec  
conv2d count: 1, dense size: 128, Dropout: 0.25 Loss: 0.85, Accuracy: 0.74, Time: 451 sec  
conv2d count: 1, dense size: 128, Dropout: 0.50 Loss: 0.85, Accuracy: 0.74, Time: 450 sec  
conv2d count: 1, dense size: 128, Dropout: 0.75 Loss: 0.85, Accuracy: 0.74, Time: 450 sec  
conv2d count: 1, dense size: 256, Dropout: 0.00 Loss: 1.36, Accuracy: 0.74, Time: 480 sec  
conv2d count: 1, dense size: 256, Dropout: 0.25 Loss: 0.85, Accuracy: 0.74, Time: 480 sec  
conv2d count: 1, dense size: 256, Dropout: 0.50 Loss: 0.85, Accuracy: 0.74, Time: 480 sec  
conv2d count: 1, dense size: 256, Dropout: 0.75 Loss: 0.85, Accuracy: 0.74, Time: 480 sec  
conv2d count: 1, dense size: 512, Dropout: 0.00 Loss: 1.09, Accuracy: 0.74, Time: 488 sec  
conv2d count: 1, dense size: 512, Dropout: 0.25 Loss: 0.85, Accuracy: 0.74, Time: 488 sec  
conv2d count: 1, dense size: 512, Dropout: 0.50 Loss: 0.85, Accuracy: 0.74, Time: 488 sec  
conv2d count: 1, dense size: 512, Dropout: 0.75 Loss: 0.85, Accuracy: 0.74, Time: 488 sec  
conv2d count: 2, dense size: 128, Dropout: 0.00 Loss: 1.19, Accuracy: 0.74, Time: 512 sec  
conv2d count: 2, dense size: 128, Dropout: 0.25 Loss: 0.85, Accuracy: 0.74, Time: 512 sec  
conv2d count: 2, dense size: 128, Dropout: 0.50 Loss: 0.85, Accuracy: 0.74, Time: 512 sec  
conv2d count: 2, dense size: 128, Dropout: 0.75 Loss: 0.85, Accuracy: 0.74, Time: 512 sec  
conv2d count: 2, dense size: 256, Dropout: 0.00 Loss: 1.39, Accuracy: 0.74, Time: 512 sec  
conv2d count: 2, dense size: 256, Dropout: 0.25 Loss: 0.85, Accuracy: 0.74, Time: 512 sec  
conv2d count: 2, dense size: 256, Dropout: 0.50 Loss: 0.85, Accuracy: 0.74, Time: 512 sec  
conv2d count: 2, dense size: 256, Dropout: 0.75 Loss: 0.85, Accuracy: 0.74, Time: 512 sec  
conv2d count: 2, dense size: 512, Dropout: 0.00 Loss: 1.09, Accuracy: 0.74, Time: 540 sec  
conv2d count: 2, dense size: 512, Dropout: 0.25 Loss: 0.85, Accuracy: 0.74, Time: 540 sec  
conv2d count: 2, dense size: 512, Dropout: 0.50 Loss: 0.85, Accuracy: 0.74, Time: 540 sec  
conv2d count: 2, dense size: 512, Dropout: 0.75 Loss: 0.85, Accuracy: 0.74, Time: 540 sec  
conv2d count: 3, dense size: 128, Dropout: 0.00 Loss: 1.19, Accuracy: 0.74, Time: 572 sec  
conv2d count: 3, dense size: 128, Dropout: 0.25 Loss: 0.85, Accuracy: 0.74, Time: 572 sec  
conv2d count: 3, dense size: 128, Dropout: 0.50 Loss: 0.85, Accuracy: 0.74, Time: 572 sec  
conv2d count: 3, dense size: 128, Dropout: 0.75 Loss: 0.85, Accuracy: 0.74, Time: 572 sec  
conv2d count: 3, dense size: 256, Dropout: 0.00 Loss: 1.39, Accuracy: 0.74, Time: 572 sec  
conv2d count: 3, dense size: 256, Dropout: 0.25 Loss: 0.85, Accuracy: 0.74, Time: 572 sec  
conv2d count: 3, dense size: 256, Dropout: 0.50 Loss: 0.85, Accuracy: 0.74, Time: 572 sec  
conv2d count: 3, dense size: 256, Dropout: 0.75 Loss: 0.85, Accuracy: 0.74, Time: 572 sec  
conv2d count: 3, dense size: 512, Dropout: 0.00 Loss: 1.09, Accuracy: 0.74, Time: 588 sec  
conv2d count: 3, dense size: 512, Dropout: 0.25 Loss: 0.85, Accuracy: 0.74, Time: 588 sec  
conv2d count: 3, dense size: 512, Dropout: 0.50 Loss: 0.85, Accuracy: 0.74, Time: 588 sec  
conv2d count: 3, dense size: 512, Dropout: 0.75 Loss: 0.85, Accuracy: 0.74, Time: 588 sec  
conv2d count: 4, dense size: 128, Dropout: 0.00 Loss: 1.19, Accuracy: 0.74, Time: 620 sec  
conv2d count: 4, dense size: 128, Dropout: 0.25 Loss: 0.85, Accuracy: 0.74, Time: 620 sec  
conv2d count: 4, dense size: 128, Dropout: 0.50 Loss: 0.85, Accuracy: 0.74, Time: 620 sec  
conv2d count: 4, dense size: 128, Dropout: 0.75 Loss: 0.85, Accuracy: 0.74, Time: 620 sec  
conv2d count: 4, dense size: 256, Dropout: 0.00 Loss: 1.39, Accuracy: 0.74, Time: 620 sec  
conv2d count: 4, dense size: 256, Dropout: 0.25 Loss: 0.85, Accuracy: 0.74, Time: 620 sec  
conv2d count: 4, dense size: 256, Dropout: 0.50 Loss: 0.85, Accuracy: 0.74, Time: 620 sec  
conv2d count: 4, dense size: 256, Dropout: 0.75 Loss: 0.85, Accuracy: 0.74, Time: 620 sec  
conv2d count: 4, dense size: 512, Dropout: 0.00 Loss: 1.09, Accuracy: 0.74, Time: 640 sec  
conv2d count: 4, dense size: 512, Dropout: 0.25 Loss: 0.85, Accuracy: 0.74, Time: 640 sec  
conv2d count: 4, dense size: 512, Dropout: 0.50 Loss: 0.85, Accuracy: 0.74, Time: 640 sec  
conv2d count: 4, dense size: 512, Dropout: 0.75 Loss: 0.85, Accuracy: 0.74, Time: 640 sec
```

Gambar 7.67 Hasil Soal 13.

#### 14. Soal 14

```
1 # In[14]: rebuild/retrain a model with the best parameters (from  
  the search) and use all data  
2 # membuat variabel model dengan isian librari Sequential  
3 model = Sequential()  
4 # variabel model di tambahkan librari Conv2D tigapuluh dua bit  
  dengan ukuran kernel 3 x 3 dan fungsi penghitungan relu dang  
  menggunakan data train_input  
5 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',  
  input_shape=np.shape(train_input[0])))  
6 # variabel model di tambahkan dengan lib MaxPooling2D dengan  
  ketentuan ukuran 2 x 2 pixcel  
7 model.add(MaxPooling2D(pool_size=(2, 2)))  
8 # variabel model di tambahkan dengan librari Conv2D 32bit dengan  
  kernel 3 x 3  
9 model.add(Conv2D(32, (3, 3), activation='relu'))  
10 # variabel model di tambahkan dengan lib MaxPooling2D dengan  
  ketentuan ukuran 2 x 2 pixcel  
11 model.add(MaxPooling2D(pool_size=(2, 2)))  
12 # variabel model di tambahkan librari Flatten  
13 model.add(Flatten())  
14 # variabel model di tambahkan librari Dense dengan fungsi tanh  
15 model.add(Dense(128, activation='tanh'))
```

```

16 # variabel model di tambahkan librari dropout untuk memangkas
    data tree sebesar 50 persen
17 model.add(Dropout(0.5))
18 # variabel model di tambahkan librari 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 paremeter dan bukan trainable
    parameter
23 print(model.summary())

```

Kode di atas befungsi untuk membuat data training kembali dengan model yang sudah di tentukan dan mencari yang terbaik dari hasil training tersebut. Hasilnya adalah sebagai berikut :

| Layer (Type)                   | Output Shape       | Param # |
|--------------------------------|--------------------|---------|
| conv2d_0 (Conv2D)              | (None, 36, 36, 32) | 896     |
| max_pooling2d_0 (MaxPooling2D) | (None, 15, 15, 32) | 0       |
| conv2d_1 (Conv2D)              | (None, 11, 11, 32) | 9248    |
| max_pooling2d_1 (MaxPooling2D) | (None, 6, 6, 32)   | 0       |
| Flatten_5 (Flatten)            | (None, 1152)       | 0       |
| dense_9 (Dense)                | (None, 128)        | 147584  |
| dropout_4 (Dropout)            | (None, 128)        | 0       |
| dense_10 (Dense)               | (None, 36)         | 47681   |

Total params: 265,129  
Trainable params: 205,129  
Non-trainable params: 60

Gambar 7.68 Hasil Soal 14.

## 15. Soal 15

```

1 # In[15]:join train and test data so we train the network on all
    data we have available to us
2 # melakukan join numpy menggunakan data train_input test_input
3 model.fit(np.concatenate((train_input, test_input)),
4           # kelanjutan data yang di gunakan pada join
        train_output test_output
5           np.concatenate((train_output, test_output)),
6           #menggunakan ukuran 32 bit dan epoch 10
7           batch_size=32, epochs=10, verbose=2)

```

Kode di atas befungsi untuk melakukan join terhadap data train dan test dengan seluruh konfigurasi yang telah di tentukan sebelumnya. Hasilnya adalah sebagai berikut :

```

1 In [16]: model.fit(ep.concatenate((train_input, test_input)),
...                   np.concatenate((train_output, test_output)),
...                   batch_size=1, epochs=10, verbose=1)
Epoch 1/10
  126s - loss: 1.7795 - accuracy: 0.5871
Epoch 2/10
  126s - loss: 1.6744 - accuracy: 0.7074
Epoch 3/10
  138s - loss: 0.9564 - accuracy: 0.7320
Epoch 4/10
  115s - loss: 0.8977 - accuracy: 0.7458
Epoch 5/10
  118s - loss: 0.8573 - accuracy: 0.7527
Epoch 6/10
  116s - loss: 0.8297 - accuracy: 0.7586
Epoch 7/10
  126s - loss: 0.8075 - accuracy: 0.7632
Epoch 8/10
  126s - loss: 0.7917 - accuracy: 0.7663
Epoch 9/10
  129s - loss: 0.7765 - accuracy: 0.7797
Epoch 10/10
  118s - loss: 0.7637 - accuracy: 0.7718
Out[16]: <keras.callbacks.History at 0x21637ebc3d8>

```

**Gambar 7.69** Hasil Soal 15.

## 16. Soal 16

```

1 # In[16]: save the trained model
2 # menyimpan model atau mengekspor model yang telah di jalantadi
3 model.save("mathsymbols.model")

```

Kode di atas befungsi untuk melakukan save pada model yang akan disimpan sebagai mathsymbols.model. Hasilnya adalah sebagai berikut :

```

# save the trained model
model.save("mathsymbols.model")

```

**Gambar 7.70** Hasil Soal 16.

## 17. Soal 17

```

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

```

Kode di atas befungsi untuk melakukan save pada model yang akan disimpan sebagai classes.npy dengan reverse ke fungsi one hot encoding. Hasilnya adalah sebagai berikut :

```

# save Label encoder (to reverse one-hot encoding)
np.save('classes.npy', label_encoder.classes_)

```

**Gambar 7.71** Hasil Soal 17.

## 18. Soal 18

```

1 # In[18]: load the pre-trained model and predict the math symbol
      for an arbitrary image;
2 # the code below could be placed in a separate file
3 # mengimpport librari keras model
4 import keras.models

```

```

5 # membuat variabel model2 untuk meload model yang telah di simpan
  tadi
6 model2 = keras.models.load_model("mathsymbols.model")
7 # mencetak hasil model2
8 print(model2.summary())

```

Kode di atas befungsi untuk melakukan load data yang sudah di train dan juga memprediksikan hasil. Hasilnya adalah sebagai berikut :

| Layer (Type)                    | Output Shape       | Param # |
|---------------------------------|--------------------|---------|
| conv2d_68 (Conv2D)              | (None, 10, 10, 32) | 896     |
| max_pooling2d_68 (MaxPooling2D) | (None, 5, 5, 32)   | 0       |
| conv2d_69 (Conv2D)              | (None, 13, 13, 32) | 9248    |
| max_pooling2d_69 (MaxPooling2D) | (None, 6, 6, 32)   | 0       |
| Flatten_45 (Flatten)            | (None, 312)        | 0       |
| dense_89 (Dense)                | (None, 128)        | 147584  |
| dropout_34 (Dropout)            | (None, 128)        | 0       |
| dense_90 (Dense)                | (None, 369)        | 47681   |

Total params: 205,329  
Trainable params: 205,329  
Non-trainable params: 0  
None

**Gambar 7.72** Hasil Soal 18.

## 19. Soal 19

```

1 # In[19]: restore the class name to integer encoder
2 # membuat variabel label encoder ke 2 dengan isian fungsi label
  encoder.
3 label_encoder2 = LabelEncoder()
4 # menambahkan method classess dengan data classess yang di
  eksport tadi
5 label_encoder2.classes_ = np.load('classes.npy')
6 # membuat fungsi predict dengan path img
7 def predict(img_path):
8     # membuat variabel newimg dengan membuat immage menjadi array
      dan membuka data berdasarkan img path
9     newimg = keras.preprocessing.image.img_to_array(pil_image.
      open(img_path))
10    # membagi data yang terdapat pada variabel newimg sebanyak
      255
11    newimg /= 255.0
12
13    # do the prediction
14    # membuat variabel predivtion dengan isian variabel model2
      menggunakan fungsi predic dengan syarat variabel newimg
      dengan data reshape
15    prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
16
17    # figure out which output neuron had the highest score, and
      reverse the one-hot encoding
18    # membuat variabel inverted dengan label encoder2 dan
      menggunakan argmax untuk mencari skor luaran tertinggi
19    inverted = label_encoder2.inverse_transform([np.argmax(
      prediction)])
20    # mencetak prediksi gambar dan confidence dari gambar.

```

```
21     print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
max(prediction)))
```

Kode di atas befungsi untuk melakukan restore terhadap nama class pada fungsi integer encoder, dengan membuat fungsi predict. Hasilnya adalah sebagai berikut :

| Layer (Type)                  | Output Shape       | Param # |
|-------------------------------|--------------------|---------|
| conv2d_68 (Conv2D)            | (None, 10, 10, 32) | 896     |
| max_pooling2d_68 (MaxPooling) | (None, 5, 5, 32)   | 0       |
| conv2d_69 (Conv2D)            | (None, 13, 13, 32) | 9248    |
| max_pooling2d_69 (MaxPooling) | (None, 6, 6, 32)   | 0       |
| flatten_45 (Flatten)          | (None, 1152)       | 0       |
| dense_89 (Dense)              | (None, 128)        | 147584  |
| dropout_34 (Dropout)          | (None, 128)        | 0       |
| dense_90 (Dense)              | (None, 369)        | 47601   |
| Total params: 205,229         |                    |         |
| Trainable params: 205,229     |                    |         |
| Non-trainable params: 0       |                    |         |
| None                          |                    |         |

**Gambar 7.73** Hasil Soal 19.

## 20. Soal 20

```
1 # In[20]: grab an image (we'll just use a random training image
   for demonstration purposes)
2 # mencari prediksi menggunakan fungsi prediksi yang di buat tadi
   dari data di HASYv2/hasy-data/v2-00010.png
3 predict("HASYv2/hasy-data/v2-00010.png")
4 # mencari prediksi menggunakan fungsi prediksi yang di buat tadi
   dari data di HASYv2/hasy-data/v2-00500.png
5 predict("HASYv2/hasy-data/v2-00500.png")
6 # mencari prediksi menggunakan fungsi prediksi yang di buat tadi
   dari data di HASYv2/hasy-data/v2-00700.png
7 predict("HASYv2/hasy-data/v2-00700.png")
```

Kode di atas befungsi untuk menunjukkan hasil akhir prediski. Hasilnya adalah sebagai berikut :

```
# grab an image (we'll just use a random training image for demonstration purposes)
predict("HASYv2/hasy-data/v2-00010.png")
prediction: A_4, confidence: 0.47
predict("HASYv2/hasy-data/v2-00500.png")
prediction: V_2, confidence: 0.56
predict("HASYv2/hasy-data/v2-00700.png")
prediction: V_alpha, confidence: 0.88
```

**Gambar 7.74** Hasil Soal 20.

### 7.3.3 Penanganan Error

#### 1. KeyboardInterrupt

```
file = open('input/00-distracteddriving_1.jpg', 'rb')
img = keras.preprocessing.image.load_img(file, target_size=(256, 256))
x = img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
x = np.array(x)
fp = multilabel_confusion_matrix(y_true, y_pred)
KeyboardInterrupt
```

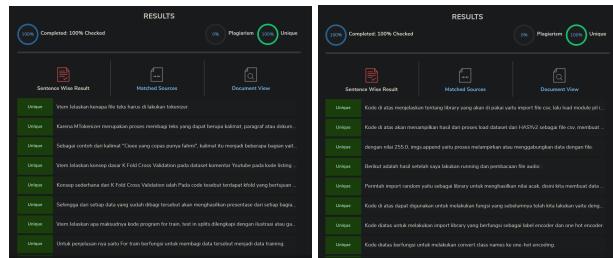
**Gambar 7.75** KeyboardInterrupt

## 2. Cara Penanganan Error

- KeyboardInterrupt

Error tersebut karena disebabkan oleh human typo yang melakukan klik pada keyboard saat running.

### 7.3.4 Bukti Tidak Plagiat



Gambar 7.76   Bukti Tidak Melakukan Plagiat Chapter 7

## 7.4 1174026 Felix Lase

### 7.4.1 Teori

#### 7.4.1.1 Soal No. 1

Kenapa file teks harus dilakukan tokenizer, dilengkapi dengan ilustrasi atau gambar.

Karena tokenizer ini berfungsi untuk mengkonversi teks menjadi urutan integer indeks kata atau vektor binary, word count atau tf-idf. Text harus dilakukan tokenizer agar dapat dirubah menjadi vektor. Dari perubahan ke vektor tersebut maka data/textnya dapat dibaca oleh komputer (terkomputerisasi).

- Ilustrasi Gambar:

#### 7.4.1.2 Soal No. 2

Konsep dasar K Fold Cross Validation pada dataset komentar Youtube, dilengkapi dengan ilustrasi atau gambar.

Pada Starti

edKFold memiliki input untuk setiap class yang terbagi menjadi 5 class pada setiap class-nya. Lalu dimasukkan kedalam class dataset youtube.

- Ilustrasi Gambar:

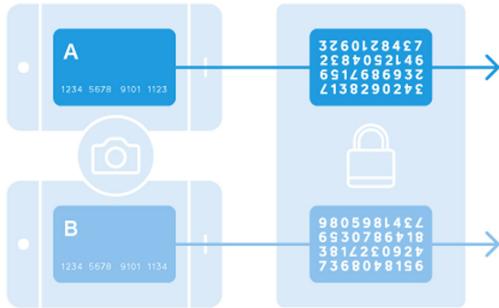
#### 7.4.1.3 Soal No. 3

Maksud kode program for train dan test in splits, dilengkapi dengan ilustrasi atau gambar.

Untuk melakukan pengujian atas data pada dataset sudah di tidak terjadi penumpukan dan split. Dimana di setiap class tidak akan muncul id yang sama.

- Ilustrasi Gambar :

## Tokenization Simplified

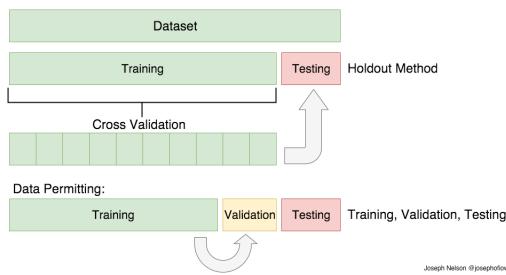


**Gambar 7.77** Tokenizer

|   |   |                |   |   |   |   |   |   |    |
|---|---|----------------|---|---|---|---|---|---|----|
| 1 | 2 | 3              | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3              | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3              | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3              | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3              | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3              | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3              | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3              | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3              | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|   |   |                |   |   |   |   |   |   |    |
|   |   | Data Pengujian |   |   |   |   |   |   |    |
|   |   | Data Pelatihan |   |   |   |   |   |   |    |

Gambar 1 – Skema 10 fold CV

**Gambar 7.78** Konsep dasar K Fold Cross Validation



**Gambar 7.79** Train dan Test in Split

**7.4.1.4 Soal No. 4** Maksud kode program train content = d['CONTENT'].iloc[train idx] dan test content = d['CONTENT'].iloc[test idx], dilengkapi dengan ilustrasi atau gambar.

Untuk mengambil data pada kolom CONTENT yang merupakan bagian dari train idx dan test idx.

- Ilustrasi Gambar:



**Gambar 7.80** Train content

**7.4.1.5 Soal No. 5** Maksud dari fungsi tokenizer = Tokenizer(num words=2000) dan tokenizer.fit on texts(train content), dilengkapi dengan ilustrasi atau gambar.

Yang pertama, yaitu fungsi tokenizer ialah mem-vektorisasi jumlah kata yang ingin diubah kedalam bentuk 2000 kata.

Yang kedua, untuk melakukan fit tokenizer untuk dat trainnya dengan data test nya untuk kolom CONTENT saja.

- Ilustrasi Gambar :

**7.4.1.6 Soal No. 6** Maksud dari fungsi d train inputs = tokenizer.texts to matrix(train content, mode='tfidf') dan d test inputs = tokenizer.texts to matrix(test content, mode='tfidf'), dilengkapi dengan ilustrasi atau gambar.

Variabel d train input untukmelakukan tokenizer dari bentuk teks ke matrix dari data train content dengan mode tfidf dan variabel d test inputs sama saja untuk data test.

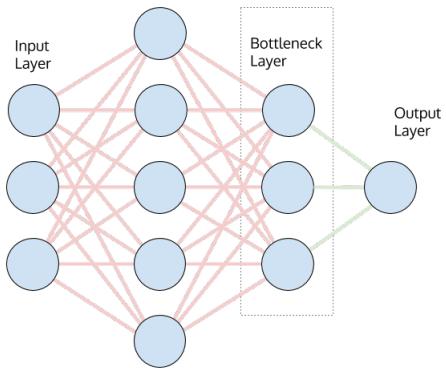
- Ilustrasi Gambar:

## ✓ Tokenization

is a process of breaking up a piece of `text` into many pieces, such as sentences and words. It works by separating `words` using spaces and punctuation.

```
[1]: from nltk.tokenize import sent_tokenize  
      sentence = "I love ice cream. I also like steak."  
      sent_tokenize(sentence)  
[1]: ['I love ice cream.', 'I also like steak.']}
```

**Gambar 7.81** Tokenizer

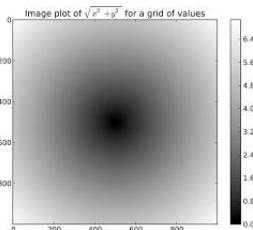


**Gambar 7.82** Train Inputs 1

**7.4.1.7 Soal No. 7** Maksud dari fungsi  $d\text{ train inputs} = d\text{ train inputs}/np.\text{amax}(np.\text{absolute}(d\text{ train inputs}))$  dan  $d\text{ test inputs} = d\text{ test inputs}/np.\text{amax}(np.\text{absolute}(d\text{ test inputs}))$ , dilengkapi dengan ilustrasi atau gambar.

Akan membagi matrix tfidf dengan amax untuk mengembalikan array atau maksimum array. Kemudian hasilnya dimasukan dalam variabel  $d\text{ train inputs}$  untuk data train dan  $d\text{ test inputs}$  untuk data test dengan nominal bilangan tanpa ada bilangan negatif dan koma.

- Ilustrasi Gambar :

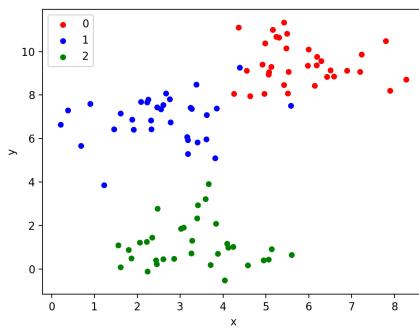


**Gambar 7.83** Train Inputs 2

**7.4.1.8 Soal No. 8** Maksud dari  $d\text{ train outputs} = np.\text{utils.to categorical}(d['CLASS'].iloc[train_idx])$  dan  $d\text{ test outputs} = np.\text{utils.to categorical}(d['CLASS'].iloc[test_idx])$

Fungsi pada kode program tersebut ditujukan untuk melakukan one-hot encoding supaya bisa masuk dan digunakan pada neural network. One-hot encoding diambil dari class yang berarti hanya terdapat 2 neuron, yaitu satu nol(1,0) atau nol satu(0,1) karena pilihannya hanya ada dua.

- Ilustrasi Gambar:



**Gambar 7.84** Compile model

**7.4.1.9 Soal No. 9** Maksud dari listing 7.2.

Fungsi kode program tersebut untuk melakukan pemodelan dengan sequential, membandingkan setiap satu larik elemen dengan cara satu persatu secara beruntun. Terdapat 512 neuron inputan dengan input shape 2000 vektor yang sudah di-normalisasi. Lalu model dilakukan aktivasi dengan fungsi 'relu'. Kemudian pemotongan bobot supaya tidak overfitting sebesar 50 persen dari neuron inputan 512. Lalu pada layer output terdapat 2 neuron outputan yaitu nol (1,0) atau nol satu (0,1). Kemudian outputan tersebut diaktivasi menggunakan fungsi softmax.

#### **7.4.1.10 Soal No. 10** Maksud dari listing 7.3.

Fungsi kode program tersebut untuk model yang telah dibuat selanjutnya dicompile dengan menggunakan algoritma optimisasi, fungsi loss, dan fungsi metrik.

#### **7.4.1.11 Soal No. 11** Deep Learning

Deep learning, yang bisa diartikan sebagai rangkaian metode untuk melatih jaringan saraf buatan multi-lapisan. Ternyata, metode ini efektif dalam mengidentifikasi pola dari data. Manakala media membicarakan jaringan saraf, kemungkinan yang dimaksud adalah deep learning.

#### **7.4.1.12 Soal No. 12** Deep Neural Network, dan apa bedanya dengan Deep Learning

Algoritma DNN (Deep Neural Networks) adalah salah satu algoritma berbasis jaringan saraf yang dapat digunakan untuk pengambilan keputusan. Contoh yang dibahas kali ini adalah mengenai penentuan penerimaan pengajuan kredit sepeda motor baru berdasarkan kelompok data yang sudah ada.

Pebedaannya dengan Deep Learning adalah terletak pada kedalaman model, deep learning adalah frasa yang digunakan untuk jaringan saraf yang kompleks.

#### **7.4.1.13 Soal No. 13** Jelaskan dengan ilustrasi gambar buatan sendiri, bagaimana perhitungan algoritma konvolusi dengan ukuran stride ( $NPM \bmod 3 + 1$ )x( $NPM \bmod 3 + 1$ ) yang terdapat max pooling.(nilai 30)

Karena  $NPM$  saya 1164067 dan hasil dari  $(NPM \bmod 3) + 1 = 2$ , maka saya menggunakan matrik kernel berukuran  $2 \times 2$ . Misalkan  $f(x,y)$  yang digunakan berukuran  $3 \times 3$  dan kernel atau mask berukuran  $2 \times 2$  adalah sebagai berikut:

Gambar Matriks:

Penyelesaian dari operasi konvolusi antara  $f(x,y)$  dengan kernel  $g(x,y)$  adalah  $f(x,y) * g(x,y)$

- Tempatkan matrik kernel di sebelah kiri atas, lalu hitung nilai piksel pada posisi (0,0) dari kernel tersebut. Konvolusi dihitung dengan cara berikut :

$$F(x,y) = \begin{matrix} 2 & 3 & 5 \\ 1 & 0 & 8 \\ 7 & 2 & 0 \end{matrix} \quad \begin{matrix} 1 & 0 \\ 2 & 4 \end{matrix}$$

**Gambar 7.85** Perhitungan algoritma konvolusi

$$(2 \times 1) + (3 \times 0) + (1 \times 2) + (0 \times 4)$$

Sehingga didapat hasil konvolusi = 4

- Tempatkan matrik kernel di sebelah kanan atas, lalu hitung nilai piksel pada posisi (0,0) dari kernel tersebut. Konvolusi dihitung dengan cara berikut :

$$(3 \times 1) + (5 \times 0) + (0 \times 2) + (8 \times 4)$$

Sehingga didapat hasil konvolusi = 35

- Tempatkan matrik kernel di sebelah kiri bawah, lalu hitung nilai piksel pada posisi (0,0) dari kernel tersebut. Konvolusi dihitung dengan cara berikut :

$$(1 \times 1) + (0 \times 0) + (7 \times 2) + (2 \times 4)$$

Sehingga didapat hasil konvolusi = 23

- Tempatkan matrik kernel di sebelah kanan bawah, lalu hitung nilai piksel pada posisi (0,0) dari kernel tersebut. Konvolusi dihitung dengan cara berikut :

$$(0 \times 1) + (8 \times 0) + (2 \times 2) + (0 \times 4)$$

Sehingga didapat hasil konvolusi = 4

Hasil:

$$\begin{matrix} 4 & 35 \\ 23 & 4 \end{matrix}$$

**Gambar 7.86** Hasil

## 7.4.2 Praktek

**7.4.2.1 Soal No. 1** Jelaskan arti dari setiap baris kode program pada blok # In[1] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:46:02 2020
4
5 @author: User

```

```

6 """
7
8 import csv
9 from PIL import Image as pil_image
10 import keras.preprocessing.image

```

- Penjelasan:

Baris Code 1: Memasukkan atau mengimport file csv

Baris Code 2: Memasukkan module image sebagai pil\_image dari library PIL

Baris Code 3: Memasukkan atau mengimport fungsi keras.preprocessing.image

- Hasil output:

```

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

```

**Gambar 7.87** 1

**7.4.2.2 Soal No. 2** Jelaskan arti dari setiap baris kode program pada blok # In[2] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:25 2020
4
5 @author: User
6 """
7
8 imgs = []
9 classes = []
10 with open('HASYv2/hasy-data-labels.csv') as csvfile:
11     csvreader = csv.reader(csvfile)
12     i = 0
13     for row in csvreader:
14         if i > 0:
15             img = keras.preprocessing.image.img_to_array(
16                 pil_image.open("HASYv2/" + row[0]))
17             # neuron activation functions behave best when input
18             # values are between 0.0 and 1.0 (or -1.0 and 1.0),
19             # so we rescale each pixel value to be in the range
20             # 0.0 to 1.0 instead of 0-255
21             img /= 255.0
22             imgs.append((row[0], row[2], img))
23             classes.append(row[2])
24         i += 1

```

- Penjelasan:

Baris Code 1: Membuat variabel imgs tanpa parameter

Baris Code 2: Membuat variabel classes tanpa parameter

Baris Code 3: Membuka file HASYv2/hasy-data-labels.csv

Baris Code 4: Membuat variabel csvreader untuk pembacaan dari file csv yang dimasukkan

Baris Code 5: Membuat variabel i dengan parameter 0

Baris Code 6: Mengeksekusi baris dari pembacaan csv

Baris Code 7: Mengaplikasikan perintah "if" dengan ketentuan variabel i lebih besar dari angka 0, maka akan dilanjutkan ke perintah berikutnya

Baris Code 8: Membuat variabel img yang mengubah image menjadi bentuk array dari file HASYv2 yang dibuka dengan row berparameter 0.

Baris Code 9: Membuat variabel img atau dengan nilai 255.0

Baris Code 10: Mendefinisikan fungsi imgs.append dimana merupakan proses melampirkan atau menggabungkan data dengan file lain atau set data yang ditentukan dengan 3 parameter yaitu row[0], row[2] dan variabel img.

Baris Code 11: Mendefinisikan fungsi append kembali dari variabel classes dengan parameternya row[2].

Baris Code 12: Mendefinisikan fungsi dimana i variabel i akan ditambah nilainya sehingga akan bernilai 1.

- Hasil output:

```
In [2]: imgs = []
...: classes = []
...: with open('HASYv2/hasy-data-labels.csv') as csvfile:
...:     csvreader = csv.reader(csvfile)
...:     i = 0
...:     for row in csvreader:
...:         if i > 0:
...:             img =
keras.preprocessing.image.img_to_array(pil_image.open("HASYv2/" + row[0]))
...:
# neuron activation functions behave best when input values are
between 0.0 and 1.0 (or -1.0 and 1.0),
...:
# so we rescale each pixel value to be in the range 0.0 to 1.0
instead of 0-255
...:             img /= 255.0
...:             imgs.append((row[0], row[2], img))
...:             classes.append(row[2])
...:             i += 1
```

**Gambar 7.88** 2

**7.4.2.3 Soal No. 3** Jelaskan arti dari setiap baris kode program pada blok # In[3] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020
4
5 @author: User
6 """
7
8 import random
9 random.shuffle(imgs)
10 split_idx = int(0.8*len(imgs))
11 train = imgs[:split_idx]
12 test = imgs[split_idx:]

```

▪ Penjelasan:

Baris Code 1: Memasukkan module random

Baris Code 2: Melakukan pengocokan atau pengacakan pada module random dengan parameter variabelnya imgs

Baris Code 3: Membagi dan memecah index dalam bentuk integer dengan mengkalikan nilai 0,8 dan fungsi len yang akan mengembalikan jumlah item dalam datanya dari variabel imgs

Baris Code 4: Membuat variabel train yang mengeksekusi imgs dengan pemecahan index awal pada data

Baris Code 5: Membuat variabel test yang mengeksekusi imgs dengan pemecahan index akhir pada data

▪ Hasil output:

```

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

```

**Gambar 7.89** 3

**7.4.2.4 Soal No. 4** Jelaskan arti dari setiap baris kode program pada blok # In[4] dan hasil luarannya.

▪ Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020
4
5 @author: User
6 """
7
8 import numpy as np
9

```

```

10 train_input = np.asarray(list(map(lambda row: row[2], train)))
11 test_input = np.asarray(list(map(lambda row: row[2], test)))
12
13 train_output = np.asarray(list(map(lambda row: row[1], train)))
14 test_output = np.asarray(list(map(lambda row: row[1], test)))

```

▪ Penjelasan:

Baris Code 1: Mengimport library numpy sebagai np

Baris Code 2: Membuat variabel train\_input untuk input menjadi sebuah array dari np menggunakan fungsi list untuk mengkoleksikan data yang dipilih dan diubah. Didalamnya diterapkan fungsi map untuk mengembalikan iterator dari datanya dengan memfungskikan lamda pada row dengan parameter [2] untuk membuat objek fungsi menjadi lebih kecil dan mudah dieksekusi dari variabel train.

Baris Code 3: Membuat variabel test\_input dengan fungsi seperti train\_input yang membedakan hanya datanya atau inputan yang diproses berasal dari variabel test

Baris Code 4: Membuat variabel train\_output untuk mengubah keluaran menjadi sebuah array dari np dengan menggunakan fungsi list untuk mengkoleksi data yang dipilih dan diubah. Didalamnya diterapkan fungsi map untuk mengembalikan iterator dari datanya dengan memfungskikan lamda pada row dengan parameter[1] untuk membuat objek fungsi menjadi lebih kecil dan mudah dieksekusi dari variabel train.

Baris Code 5: Membuat variabel test\_output dengan fungsi yang sama seperti train\_output yang membedakan hanya datanya atau inputan yang diproses berasal dari variabel test

▪ Hasil output:

```

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

```

Gambar 7.90 4

**7.4.2.5 Soal No. 5** Jelaskan arti dari setiap baris kode program pada blok # In[5] dan hasil luarannya.

▪ Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020

```

```

4
5 @author: User
6 """
7
8 from sklearn.preprocessing import LabelEncoder
9 from sklearn.preprocessing import OneHotEncoder

```

- Penjelasan:

Baris Code 1: Memasukkan modul atau fungsi LabelEncoder dari sklearn.preprocessing untuk dapat digunakan menormalkan label dimana label encoder didefinisikan dengan nilai antara 0 dan n\_classes-1.

Baris Code 2: Memasukkan modul atau fungsi OneHotEncoder dari sklearn.preprocessing untuk mendefinisikan fitur input dimana mengambil nilai dalam kisaran [0, maks (nilai)).

- Hasil output:

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

**Gambar 7.91** 5

**7.4.2.6 Soal No. 6** Jelaskan arti dari setiap baris kode program pada blok # In[6] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020
4
5 @author: User
6 """
7
8 label_encoder = LabelEncoder()
9 integer_encoded = label_encoder.fit_transform(classes)

```

- Penjelasan:

Baris Code 1: Membuat variabel label\_encoder dengan modul atau fungsi dari LabelEncoder tanpa parameter

Baris Code 2: Membuat variabel integer\_encoded dengan fungsi label\_encoder.fit\_transform dari variabel classes yang akan mengembalikan beberapa data yang diubah kembali dari variabel label\_encoder.

- Hasil output:

```
In [6]: label_encoder = LabelEncoder()
...: integer_encoded = label_encoder.fit_transform(classes)
```

Gambar 7.92 6

**7.4.2.7 Soal No. 7** Jelaskan arti dari setiap baris kode program pada blok # In[7] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020
4
5 @author: User
6 """
7
8 onehot_encoder = OneHotEncoder(sparse=False)
9 integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
10 onehot_encoder.fit(integer_encoded)
```

- Penjelasan:

Baris 1: Membuat variabel onehot\_encoder yang memanggil fungsi OneHotEncoder tanpa mengembalikan matriks karena sparse=false.

Baris 2: Membuat variabel integer\_encoded memanggil variabel integer\_encoded pada kode program 6 untuk dieksekusi memberikan bentuk baru ke array tanpa mengubah datanya dari mengembalikan panjang nilai dari integer\_encoded.

Baris 3: Onehotencoding melakukan fitting pada integer\_encoded.

- Hasil output:

```
In [7]: onehot_encoder = OneHotEncoder(sparse=False)
...: integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
...: onehot_encoder.fit(integer_encoded)
D:\Anaconda\lib\site-packages\sklearn\preprocessing\_encoders.py:371: FutureWarning:
The handling of integer data will change in version 0.22. Currently, the categories
are determined based on the range [0, max(values)], while in the future they will be
determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify
"categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories
to integers, then you can now use the OneHotEncoder directly.
    warnings.warn(msg, FutureWarning)
Out[7]:
OneHotEncoder(categorical_features=None, categories=None,
               dtype=<class 'numpy.float64'>, handle_unknown='error',
               n_values=None, sparse=False)
```

Gambar 7.93 7

**7.4.2.8 Soal No. 8** Jelaskan arti dari setiap baris kode program pada blok # In[8] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:28 2020
4
5 @author: User
6 """
7
8 train_output_int = label_encoder.transform(train_output)
9 train_output = onehot_encoder.transform(train_output_int.reshape(
10     len(train_output_int), 1))
11 test_output_int = label_encoder.transform(test_output)
12 test_output = onehot_encoder.transform(test_output_int.reshape(
13     len(test_output_int), 1))
14 num_classes = len(label_encoder.classes_)
15 print("Number of classes: %d" % num_classes)
```

- Penjelasan:

Baris 1: Membuat variabel `train_output_int` yang mengeksekusi `label_encoder` dengan mengubah nilai dari parameter variabel `train_output`.

Baris 2: Membuat variabel `train_output` yang mengeksekusi variabel `onehot_encoder` dari kode program 7 dengan mengubah nilai dari variabel parameter `train_output_int` yang datanya sudah diubah kedalam bentuk array dan panjang nilai dari `train_output_int` telah dikembalikan.

Baris 3: Membuat variabel `test_output_int` yang mengeksekusi `label_encoder` dengan mengubah nilai dari parameter variabel `test_output`.

Baris 4: Membuat variabel `test_output` yang mengeksekusi variabel `onehot_encoder` dari kode program 7 dengan mengubah nilai dari variabel parameter `test_output_int` yang datanya sudah diubah kedalam bentuk array dan panjang nilai dari `test_output_int` telah dikembalikan.

Baris 5: Membuat variabel `num_classes` untuk mengetahui jumlah class dari `label_encoder`

Baris 6: Perintah `print` digunakan untuk memunculkan hasil dari variabel `num_classes`

- Hasil output:

**7.4.2.9 Soal No. 9** Jelaskan arti dari setiap baris kode program pada blok # In[9] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:28 2020
```

```
In [8]: train_output_int = label_encoder.transform(train_output)
      ...: train_output =
onehot_encoder.transform(train_output_int.reshape(len(train_output_int), 1))
      ...: test_output_int = label_encoder.transform(test_output)
      ...: test_output =
onehot_encoder.transform(test_output_int.reshape(len(test_output_int), 1))
      ...
      ...: num_classes = len(label_encoder.classes_)
      ...: print("Number of classes: %d" % num_classes)
Number of classes: 369
```

Gambar 7.94 8

```
4
5 @author: User
6 """
7
8 from keras.models import Sequential
9 from keras.layers import Dense, Dropout, Flatten
10 from keras.layers import Conv2D, MaxPooling2D
```

▪ Penjelasan:

Baris 1: Melakukan importing fungsi model sequential dari library keras.

Baris 2: Melakukan importing fungsi layer dense, dropout, dan flatten dari library keras.

Baris 3: Melakukan importing fungsi layer Conv2D dan MaxPooling2D dari library keras.

▪ Hasil output:

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

Gambar 7.95 9

**7.4.2.10 Soal No. 10** Jelaskan arti dari setiap baris kode program pada blok # In[10] dan hasil luarannya.

▪ Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 model = Sequential()
9 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
10                  input_shape=np.shape(train_input[0])))
```

```

11 model.add(MaxPooling2D(pool_size=(2, 2)))
12 model.add(Conv2D(32, (3, 3), activation='relu'))
13 model.add(MaxPooling2D(pool_size=(2, 2)))
14 model.add(Flatten())
15 model.add(Dense(1024, activation='tanh'))
16 model.add(Dropout(0.5))
17 model.add(Dense(num_classes, activation='softmax'))
18
19 model.compile(loss='categorical_crossentropy', optimizer='adam',
20                 metrics=['accuracy'])
21
22 print(model.summary())

```

▪ Penjelasan:

Baris 1: Melakukan pemodelan Sequential.

Baris 2: Menambahkan Konvolusi 2D dengan 32 filter konvolusi masing-masing berukuran 3x3 dengan algoritam activation relu dengan data dari train input mulai dari baris nol.

Baris 3: Menambahkan Max Pooling dengan matriks 2x2.

Baris 4: Penambahan Konvolusi 2D dengan 32 filter, konvolusi masing-masing berukuran 3x3 dengan algoritam activation relu.

Baris 5 Menambahkan Max Pooling dengan matriks 2x2.

Baris 6: Mendefinisikan inputan dengan 1024 neuron dan menggunakan algoritma tanh untuk activationnya.

Baris 7: Dropout terdiri dari pengaturan secara acak tingkat pecahan unit input ke 0 pada setiap pembaruan selama waktu pelatihan, yang membantu mencegah overfitting sebesar 50% .

Baris 8: Untuk output layer menggunakan data dari variabel num classes dengan fungsi activationnya softmax.

Baris 9: Konfigurasi proses pembelajaran, melalui metode compile, sebelum melatih suatu model.

Baris 10: Menampilkan model yang telah dibuat.

▪ Hasil output:

**7.4.2.11 Soal No. 11** Jelaskan arti dari setiap baris kode program pada blok # In[11] dan hasil luarannya.

▪ Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:11 2020
4
5 @author: User
6 """

```

```

....: model.add(Flatten())
....: model.add(Dense(1024, 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())
WARNING:tensorflow:From D:\Anaconda\lib\site-packages\tensorflow\python\framework\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From D:\Anaconda\lib\site-packages\keras\backend\tensorflow_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.



| Layer (type)                   | Output Shape       | Param # |
|--------------------------------|--------------------|---------|
| conv2d_1 (Conv2D)              | (None, 30, 30, 32) | 896     |
| max_pooling2d_1 (MaxPooling2D) | (None, 15, 15, 32) | 0       |
| conv2d_2 (Conv2D)              | (None, 13, 13, 32) | 9248    |
| max_pooling2d_2 (MaxPooling2D) | (None, 6, 6, 32)   | 0       |
| flatten_1 (Flatten)            | (None, 1152)       | 0       |
| dense_1 (Dense)                | (None, 1024)       | 1180672 |
| dropout_1 (Dropout)            | (None, 1024)       | 0       |
| dense_2 (Dense)                | (None, 369)        | 378225  |


Total params: 1,569,041
Trainable params: 1,569,041
Non-trainable params: 0

```

---

None

```
7  
8  
9 import keras.callbacks  
10 tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-  
    style')
```

- Penjelasan:

Baris Code 1: Melakukan import library keras.callbacks yang digunakan pada penulisan log untuk TensorBoard, untuk memvisualisasikan grafik dinamis dari pelatihan dan metrik pengujian.

Baris Code 2: Membuat variabel tensorboard untuk mendefinisikan fungsi TensorBoard pada keras.callbacks yang digunakan sebagai alat visualisasi yang disediakan dengan TensorFlow. Dan untuk fungsi log\_dir memanggil data yaitu './logs/mnist-style'

- Hasil output:

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

Gambar 7.97 11

**7.4.2.12 Soal No. 12** Jelaskan arti dari setiap baris kode program pada blok # In[12] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-  
2 """  
3 Created on Mon Apr 13 23:50:11 2020  
4  
5 @author: User  
6 """  
7  
8 model.fit(train_input, train_output,  
9             batch_size=32,  
10            epochs=10,  
11            verbose=2,  
12            validation_split=0.2,  
13            callbacks=[tensorboard])  
14  
15 score = model.evaluate(test_input, test_output, verbose=2)  
16 print('Test loss:', score[0])  
17 print('Test accuracy:', score[1])
```

- Penjelasan:

Baris Code 1: Menerapkan fungsi model.fit yang didalamnya memproses train\_input, train\_output

Baris Code 2: Penerapan fungsi yang sama difungsikan batch\_size apabila batch\_sizenya tidak ditemukan maka otomatis akan dijadikan nilai 32

Baris Code 3: Penerapan fungsi yang sama, difungsikan epochs dimana perulangan dari berapa kali nilai yang digunakan untuk data, dan jumlahnya ialah 10

Baris Code 4: Mendefinisikan fungsi verbose untuk digunakan sebagai opsi menghasilkan informasi logging dari data yang ditentukan dengan nilai 2

Baris Code 5: Mendefinisikan fungsi validation\_split untuk memecah nilai dari perhitungan validasinya sebesar 0,2. (Fraksi data pelatihan untuk digunakan sebagai data validasi)

Baris Code 6: Mendefinisikan fungsi callbacks dengan parameteranya yang mengeksekusi tensorboard dimana digunakan untuk visualisasikan parameter training, metrik, hiperparameter pada nilai/data yang diproses

Baris Code 7: Mendefinisikan variabel score dengan fungsi evaluate dari model dengan parameter test\_input, test\_output dan verbose=2 untuk memprediksi output dan input yang diberikan dan kemudian menghitung fungsi metrik yang ditentukan dalam modelnya

Baris Code 8: Mencetak score optimasi dari test dengan ketentuan nilai parameter 0

Baris Code 9: Mencetak score akurasi dari test dengan ketentuan nilai parameter 1

- Hasil output:

**7.4.2.13 Soal No. 13** Jelaskan arti dari setiap baris kode program pada blok # In[13] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 import time
9
10 results = []
11 for conv2d_count in [1, 2]:
12     for dense_size in [128, 256, 512, 1024, 2048]:
13         for dropout in [0.0, 0.25, 0.50, 0.75]:
14             model = Sequential()
15             for i in range(conv2d_count):
16                 if i == 0:
17                     model.add(Conv2D(32, kernel_size=(3, 3),
activation='relu', input_shape=np.shape(train_input[0])))
18                 else:

```

```
In [12]: model.fit(train_input, train_output,
...:     batch_size=32,
...:     epochs=10,
...:     verbose=2,
...:     validation_split=0.2,
...:     callbacks=[tensorboard])
...:
...: score = model.evaluate(test_input, test_output, verbose=2)
...: print('Test loss:', score[0])
...: print('Test accuracy:', score[1])
WARNING:tensorflow:From D:\Anaconda\lib\site-packages\tensorflow\python\ops\numpy_ops.py:3066: to_int32 (from tensorflow.python.ops.numpy_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 107668 samples, validate on 26918 samples
Epoch 1/10
- 2007s - loss: 1.5334 - acc: 0.6294 - val_loss: 0.9824 - val_acc: 0.7285
Epoch 2/10
- 918s - loss: 0.9694 - acc: 0.7321 - val_loss: 0.9162 - val_acc: 0.7494
Epoch 3/10
- 562s - loss: 0.8543 - acc: 0.7556 - val_loss: 0.8883 - val_acc: 0.7530
Epoch 4/10
- 503s - loss: 0.7854 - acc: 0.7699 - val_loss: 0.8441 - val_acc: 0.7666
Epoch 5/10
- 361s - loss: 0.7364 - acc: 0.7796 - val_loss: 0.8491 - val_acc: 0.7663
Epoch 6/10
- 366s - loss: 0.6936 - acc: 0.7899 - val_loss: 0.8522 - val_acc: 0.7692
Epoch 7/10
- 360s - loss: 0.6591 - acc: 0.7962 - val_loss: 0.8580 - val_acc: 0.7668
Epoch 8/10
- 389s - loss: 0.6344 - acc: 0.8017 - val_loss: 0.8496 - val_acc: 0.7654
Epoch 9/10
- 357s - loss: 0.6076 - acc: 0.8072 - val_loss: 0.8597 - val_acc: 0.7638
Epoch 10/10
- 359s - loss: 0.5905 - acc: 0.8116 - val_loss: 0.8889 - val_acc: 0.7637
Test loss: 0.8794204677150739
Test accuracy: 0.7632181175230488
```

Gambar 7.98 12

```

19         model.add(Conv2D(32, kernel_size=(3, 3),
20                           activation='relu'))
21         model.add(MaxPooling2D(pool_size=(2, 2)))
22         model.add(Flatten())
23         model.add(Dense(dense_size, activation='tanh'))
24         if dropout > 0.0:
25             model.add(Dropout(dropout))
26         model.add(Dense(num_classes, activation='softmax'))
27
28         model.compile(loss='categorical_crossentropy',
29                         optimizer='adam', metrics=['accuracy'])
30
31         log_dir = './logs/conv2d_%d-dense_%d-dropout_%.2f' %
32         (conv2d_count, dense_size, dropout)
33         tensorboard = keras.callbacks.TensorBoard(log_dir=
34         log_dir)
35
36         start = time.time()
37         model.fit(train_input, train_output, batch_size=32,
38         epochs=10,
39                     verbose=0, validation_split=0.2, callbacks
=[tensorboard])
40         score = model.evaluate(test_input, test_output,
41         verbose=2)
42         end = time.time()
43         elapsed = end - start
44         print("Conv2D count: %d, Dense size: %d, Dropout: %.2
45 f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count
46 , dense_size, dropout, score[0], score[1], elapsed))
47         results.append((conv2d_count, dense_size, dropout,
48 score[0], score[1], elapsed))

```

▪ Penjelasan:

Baris 1: Import atau memasukkan modul time

Baris 2: Variabel result berisikan array kosong

Baris 3: Menggunakan convolution 2D yang berarti memiliki 1 atau 2 layer

Baris 4: Mendefinisikan dense size dengan ukuran 128, 256, 512, 1024, 2048

Baris 5: Mendefinsikan drop out dengan 0, 25%, 50%, dan 75%

Baris 6: Melakukan pemodelan Sequential

Baris 7: Jika ini adalah layer pertama, akan memasukkan bentuk input.

Baris 8: Kalau tidak kita hanya akan menambahkan layer.

Baris 9: Kemudian, setelah menambahkan layer konvolusi, lakukan hal yang sama dengan max pooling.

Baris 10: Lalu, ratakan atau flatten dan menambahkan dense size ukuran apa pun yang berasal dari dense size. Dimana akan selalu menggunakan algoritma tanh

Baris 11: Jika dropout digunakan, akan menambahkan layer dropout. Dropout ini berarti misalnya 50%, bahwa setiap kali memperbarui bobot setelah setiap batch, ada peluang 50% untuk setiap bobot yang tidak akan diperbarui

Baris 12: Menempatkan di antara dua lapisan padat untuk dihindarkan dari melindunginya dari overfitting. Lapisan terakhir akan selalu menjadi jumlah kelas karena itu harus, dan menggunakan softmax. Itu dikompilasi dengan cara yang sama.

Baris 13: Atur direktori log yang berbeda untuk TensorBoard sehingga dapat membedakan konfigurasi yang berbeda.

Baris 14: Variabel start akan memanggil modul time

Baris 15: Melakukan fit atau compile

Baris 16: Melakukan scoring dengan evaluate yang akan menampilkan data loss dan accuracy dari model

Baris 17: 'End' merupakan variabel untuk melihat waktu akhir pada saat pemodelan berhasil dilakukan.

Baris 18: Menampilkan hasil dari skrip diatas

- Hasil output:

```
...:
...:                         model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
...:
...:                         log_dir = './logs/conv2d_%d-dense_%d-dropout_.2f' % (conv2d_count,
dense_size, dropout)
...:                         tensorboard = keras.callbacks.TensorBoard(log_dir=log_dir)
...:
...:                         start = time.time()
...:                         model.fit(train_input, train_output, batch_size=32, epochs=10,
...:                                   verbose=0, validation_split=0.2, callbacks=[tensorboard])
...:                         score = model.evaluate(test_input, test_output, verbose=2)
...:                         end = time.time()
...:                         elapsed = end - start
...:                         print("Conv2D count: %d, Dense size: %d, Dropout: %.2f - Loss: %.
2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count, dense_size, dropout, score[0],
score[1], elapsed))
...:                         results.append((conv2d_count, dense_size, dropout, score[0],
score[1], elapsed))
Conv2D count: 1, Dense size: 128, Dropout: 0.00 - Loss: 1.13, Accuracy: 0.74, Time: 1540
sec
Conv2D count: 1, Dense size: 128, Dropout: 0.25 - Loss: 0.93, Accuracy: 0.76, Time: 1579
sec
Conv2D count: 1, Dense size: 128, Dropout: 0.50 - Loss: 0.81, Accuracy: 0.77, Time: 1599
sec
Conv2D count: 1, Dense size: 128, Dropout: 0.75 - Loss: 0.82, Accuracy: 0.77, Time: 1627
```

Gambar 7.99 13

**7.4.2.14 Soal No. 14** Jelaskan arti dari setiap baris kode program pada blok # In[14] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 model = Sequential()
9 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
10                 input_shape=np.shape(train_input[0])))
11 model.add(MaxPooling2D(pool_size=(2, 2)))
12 model.add(Conv2D(32, (3, 3), activation='relu'))
13 model.add(MaxPooling2D(pool_size=(2, 2)))
14 model.add(Flatten())
15 model.add(Dense(128, activation='tanh'))
16 model.add(Dropout(0.5))
17 model.compile(loss='categorical_crossentropy', optimizer='adam',
18                 metrics=['accuracy'])
18 print(model.summary())

```

▪ Penjelasan:

Baris 1: Melakukan pemodelan Sequential

Baris 2: Menambahkan Convolutio 2D dengan dmensi 32, dan ukuran matriks 3x3 dengan function aktivasi yang digunakan yaitu relu dan menampilkan input shape

Baris 3: Melakukan Max Pooling 2D dengan ukuran matriks 2x2

Baris 4: Melakukan Convolusi lagi dengan kriteria yang sama tanpa menambahkan input, ini dilakukan untuk mendapatkan data yang terbaik

Baris 5: Flatten digunakan untuk meratakan inputan atau masukkan data

Baris 6: Menambahkan dense input sebanyak 128 neuron dengan menggunakan function aktivasi tanh.

Baris 7: Dropout sebanyak 50% untuk menghindari overfitting

Baris 8: Menambahkan dense pada model untuk output dimana layerini akan menjadi jumlah dari class yang ada.

Baris 9: Mengcompile model yang didefinisikan diatas

Baris 10: Menampilkan ringkasan dari pemodelan yang dilakukan

▪ Hasil output:

**7.4.2.15 Soal No. 15** Jelaskan arti dari setiap baris kode program pada blok # In[15] dan hasil luarannya.

▪ Code:

```
In [14]: model = Sequential()
...: model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=np.shape(train_input[0])))
...: model.add(MaxPooling2D(pool_size=(2, 2)))
...: model.add(Conv2D(32, (3, 3), activation='relu'))
...: model.add(MaxPooling2D(pool_size=(2, 2)))
...: model.add(Flatten())
...: model.add(Dense(128, 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())

Layer (type)                 Output Shape              Param #
=====conv2d_3 (Conv2D)        (None, 30, 30, 32)      896
=====max_pooling2d_3 (MaxPooling2D) (None, 15, 15, 32)   0
=====conv2d_4 (Conv2D)        (None, 13, 13, 32)      9248
=====max_pooling2d_4 (MaxPooling2D) (None, 6, 6, 32)    0
=====flatten_2 (Flatten)     (None, 1152)             0
=====dense_3 (Dense)         (None, 128)              147584
=====dropout_2 (Dropout)     (None, 128)              0
=====dense_4 (Dense)         (None, 369)              47601
=====
Total params: 205,329
Trainable params: 205,329
Non-trainable params: 0
```

---

None

Gambar 7.100 14

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 model.fit(np.concatenate((train_input, test_input)),
9         np.concatenate((train_output, test_output)),
10        batch_size=32, epochs=10, verbose=2)

```

- Penjelasan:

Baris 1: Melakukan fit dengan join data train dan test agar dapat dilakukan pelatihan untuk jaringan pada semua data yang dimiliki.

- Hasil output:

```

In [26]: model.fit(np.concatenate((train_input, test_input)),
...:                 np.concatenate((train_output, test_output)),
...:                 batch_size=32, epochs=10, verbose=2)
Epoch 1/10
- 247s - loss: 1.7949 - acc: 0.5843
Epoch 2/10
- 236s - loss: 1.0797 - acc: 0.7064
Epoch 3/10
- 235s - loss: 0.9648 - acc: 0.7308
Epoch 4/10
- 237s - loss: 0.9060 - acc: 0.7434
Epoch 5/10
- 237s - loss: 0.8671 - acc: 0.7519
Epoch 6/10
- 236s - loss: 0.8362 - acc: 0.7573
Epoch 7/10
- 238s - loss: 0.8137 - acc: 0.7631
Epoch 8/10
- 239s - loss: 0.7980 - acc: 0.7652
Epoch 9/10
- 239s - loss: 0.7831 - acc: 0.7692
Epoch 10/10
- 239s - loss: 0.7695 - acc: 0.7724
Out[26]: <keras.callbacks.History at 0x14c1941f5f8>

```

**Gambar 7.101** 15

**7.4.2.16 Soal No. 16** Jelaskan arti dari setiap baris kode program pada blok # In[16] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 model.save("mathsymbols.model")

```

- Penjelasan:

Baris 1: Menyimpan model yang telah di latih dengan nama mathsymbols.model

- Hasil output:

```
In [22]: model.save("mathsymbols.model")
```

Gambar 7.102 16

**7.4.2.17 Soal No. 17** Jelaskan arti dari setiap baris kode program pada blok # In[17] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 np.save('classes.npy', label_encoder.classes_)
```

- Penjelasan:

Baris 1: Menyimpan label enkoder (untuk membalikkan one-hot encoder) dengan nama classes.npy

- Hasil output:

```
In [23]: np.save('classes.npy', label_encoder.classes_)
```

Gambar 7.103 17

**7.4.2.18 Soal No. 18** Jelaskan arti dari setiap baris kode program pada blok # In[18] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:53:35 2020
4
5 @author: User
6 """
7
8 import keras.models
9 model2 = keras.models.load_model("mathsymbols.model")
10 print(model2.summary())
```

- Penjelasan:

Baris 1: Memasukkan atau mengimport models dari librari Keras

Baris 2: Variabel model2 akan memanggil model yang telah disave sebelumnya

Baris 3: Menampilkan ringkasan dari hasil pemodelan

- Hasil output:

```
In [24]: import keras.models
...: model2 = keras.models.load_model("mathsymbols.model")
...: print(model2.summary())

Layer (type)                 Output Shape              Param #
=====
conv2d_3 (Conv2D)            (None, 30, 30, 32)      896
max_pooling2d_3 (MaxPooling2 (None, 15, 15, 32)      0
conv2d_4 (Conv2D)            (None, 13, 13, 32)      9248
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 32)        0
flatten_2 (Flatten)          (None, 1152)             0
dense_3 (Dense)              (None, 128)              147584
dropout_2 (Dropout)          (None, 128)              0
dense_4 (Dense)              (None, 369)              47601
=====
Total params: 205,329
Trainable params: 205,329
Non-trainable params: 0
=====
None
```

**Gambar 7.104** 18

**7.4.2.19 Soal No. 19** Jelaskan arti dari setiap baris kode program pada blok # In[19] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:53:36 2020
4
5 @author: User
6 """
7
8 label_encoder2 = LabelEncoder()
9 label_encoder2.classes_ = np.load('classes.npy')
10
11 def predict(img_path):
12     newimg = keras.preprocessing.image.img_to_array(pil_image.
13         open(img_path))
14     newimg /= 255.0
```

```

15 # do the prediction
16 prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
17
18 # figure out which output neuron had the highest score, and
19 # reverse the one-hot encoding
20 inverted = label_encoder2.inverse_transform([np.argmax(
    prediction)]) # argmax finds highest-scoring output
    print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
        max(prediction)))

```

▪ Penjelasan:

Baris 1: Memanggil fungsi LabelEncoder

Baris 2: Variabel label encoder akan memanggil class yang disimpan sebelumnya.

Baris 3: Function Predict akan mengubah gambar kedalam bentuk array

Baris 4: Variabel prediction akan melakukan prediksi untuk model2 dengan reshape variabel newimg dengan bentukarray 4D.

Baris 5: Variabel inverted akan mencari nilai tertinggi output dari hasil prediksi tadi

▪ Hasil output:

```

In [25]: label_encoder2 = LabelEncoder()
...: label_encoder2.classes_ = np.load('classes.npy')
...:
...: def predict(img_path):
...:     newimg =
keras.preprocessing.image.img_to_array(pil_image.open(img_path))
...:     newimg /= 255.0
...:
...:     # do the prediction
...:     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
...:
...:     # figure out which output neuron had the highest score, and reverse the
one-hot encoding
...:     inverted = label_encoder2.inverse_transform([np.argmax(prediction)]) # #
argmax finds highest-scoring output
...:     print("Prediction: %s, confidence: %.2f" % (inverted[0],
np.max(prediction)))

```

Gambar 7.105 19

**7.4.2.20 Soal No. 20** Jelaskan arti dari setiap baris kode program pada blok # In[20] dan hasil luarannya.

▪ Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:54:14 2020
4
5 @author: User

```

```

6 """
7
8 predict("HASYv2/hasy-data/v2-00010.png")
9
10 predict("HASYv2/hasy-data/v2-00500.png")
11
12 predict("HASYv2/hasy-data/v2-00700.png")

```

- Penjelasan:

Baris 1: Melakukan prediksi dari pelatihan dari gambar v2-00010.png

Baris 2: Melakukan prediksi dari pelatihan dari gambar v2-00500.png

Baris 3: Melakukan prediksi dari pelatihan dari gambar v2-00700.png

- Hasil output:

```

In [26]: predict("HASYv2/hasy-data/v2-00010.png")
...
...: predict("HASYv2/hasy-data/v2-00500.png")
...
...: predict("HASYv2/hasy-data/v2-00700.png")
Prediction: \pitchfork, confidence: 0.00
Prediction: \copyright, confidence: 0.00
Prediction: J, confidence: 0.00

```

**Gambar 7.106** 20

### 7.4.3 Penanganan Error

#### 7.4.3.1 Penanganan Error

- Screenshoot:

```

Traceback (most recent call last):
File "<ipython-input-3-fd9abfd31556>", line 1, in <module>
    model.fit(np.concatenate((train_input, test_input)),
NameError: name 'model' is not defined

```

**Gambar 7.107** Error

- Code Error:

NameError: name 'model' is not defined

- Penanganan Error:

Berdasarkan error maka penyelesaiannya ialah melakukan pendefinisian variabel model sehingga code dapat dijalankan. Lakukan running pada code yang berada diatasnya dimana mendefinisikan variabel model.

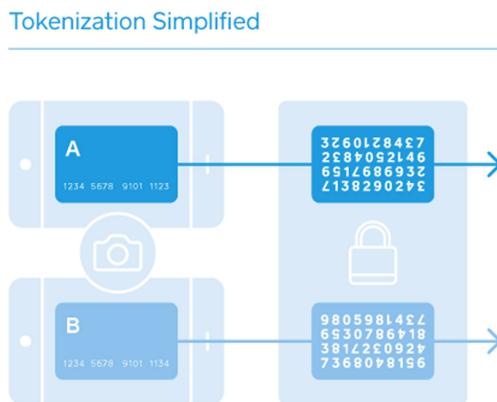
## 7.5 1174017 Muh. Rifky Prananda

### 7.5.1 Teori

**7.5.1.1 Soal No. 1** Kenapa file teks harus dilakukan tokenizer, dilengkapi dengan ilustrasi atau gambar.

Karena tokenizer ini berfungsi untuk mengkonversi teks menjadi urutan integer indeks kata atau vektor binary, word count atau tf-idf. Text harus dilakukan tokenizer agar dapat dirubah menjadi vektor. Dari perubahan ke vektor tersebut maka data/textnya dapat dibaca oleh komputer (terkomputerisasi).

- Ilustrasi Gambar:



**Gambar 7.108** Tokenizer

**7.5.1.2 Soal No. 2** Konsep dasar K Fold Cross Validation pada dataset komentar Youtube, dilengkapi dengan ilustrasi atau gambar.

Pada Starti

edKFold memiliki input untuk setiap class yang terbagi menjadi 5 class pada setiap class-nya. Lalu dimasukkan kedalam class dataset youtube.

- Ilustrasi Gambar:

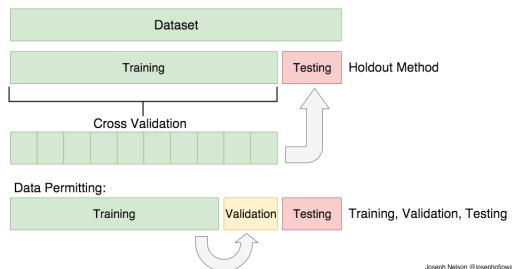
**7.5.1.3 Soal No. 3** Maksud kode program for train dan test in splits, dilengkapi dengan ilustrasi atau gambar.

Untuk melakukan pengujian atas data pada dataset sudah di tidak terjadi penumpukan dan split. Dimana di setiap class tidak akan muncul id yang sama.

- Ilustrasi Gambar :

|   |   |   |   |                |   |   |   |   |    |
|---|---|---|---|----------------|---|---|---|---|----|
| 1 | 2 | 3 | 4 | 5              | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3 | 4 | 5              | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3 | 4 | 5              | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3 | 4 | 5              | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3 | 4 | 5              | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3 | 4 | 5              | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3 | 4 | 5              | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3 | 4 | 5              | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3 | 4 | 5              | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3 | 4 | 5              | 6 | 7 | 8 | 9 | 10 |
|   |   |   |   | Data Pengujian |   |   |   |   |    |
|   |   |   |   | Data Pelatihan |   |   |   |   |    |

Gambar 1 – Skema 10 fold CV

**Gambar 7.109** Konsep dasar K Fold Cross Validation**Gambar 7.110** Train dan Test in Split

**7.5.1.4 Soal No. 4** Maksud kode program train content = d['CONTENT'].iloc[train idx] dan test content = d['CONTENT'].iloc[test idx], dilengkapi dengan ilustrasi atau gambar.

Untuk mengambil data pada kolom CONTENT yang merupakan bagian dari train idx dan test idx.

- Ilustrasi Gambar:



Gambar 7.111 Train content

**7.5.1.5 Soal No. 5** Maksud dari fungsi tokenizer = Tokenizer(num words=2000) dan tokenizer.fit on texts(train content), dilengkapi dengan ilustrasi atau gambar.

Yang pertama, yaitu fungsi tokenizer ialah mem-vektorisasi jumlah kata yang ingin diubah kedalam bentuk token 2000 kata.

Yang kedua, untuk melakukan fit tokenizer untuk dat trainnya dengan data test nya untuk kolom CONTENT saja.

- Ilustrasi Gambar :

✓ Tokenization

is a process of breaking up a piece of **text** into many pieces, such as sentences and words. It works by separating **words** using spaces and punctuation.

```

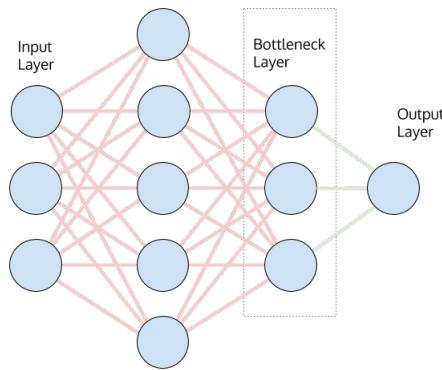
[1]: from nltk.tokenize import sent_tokenize
      sentence = "I love ice cream. I also like steak."
      sent_tokenize(sentence)
[1]: ['I love ice cream.', 'I also like steak.']
  
```

Gambar 7.112 Tokenizer

**7.5.1.6 Soal No. 6** Maksud dari fungsi d train inputs = tokenizer.texts to matrix(train content, mode='tfidf') dan d test inputs = tokenizer.texts to matrix(test content, mode='tfidf'), dilengkapi dengan ilustrasi atau gambar.

Variabel d train input untukmelakukan tokenizer dari bentuk teks ke matrix dari data train content dengan mode tfidf dan variabel d test inputs sama saja untuk data test.

- Ilustrasi Gambar:

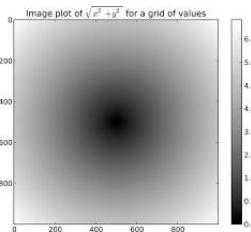


**Gambar 7.113** Train Inputs 1

**7.5.1.7 Soal No. 7** Maksud dari fungsi  $d\text{ train inputs} = d\text{ train inputs}/\text{np.amax}(\text{np.absolute}(d\text{ train inputs}))$  dan  $d\text{ test inputs} = d\text{ test inputs}/\text{np.amax}(\text{np.absolute}(d\text{ test inputs}))$ , dilengkapi dengan ilustrasi atau gambar.

Akan membagi matrix tfidf dengan amax untuk mengembalikan array atau maksimum array. Kemudian hasilnya dimasukan dalam variabel  $d\text{ train inputs}$  untuk data train dan  $d\text{ test inputs}$  untuk data test dengan nominal bilangan tanpa ada bilangan negatif dan koma.

- Ilustrasi Gambar :

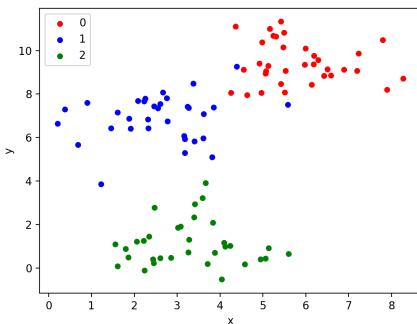


**Gambar 7.114** Train Inputs 2

**7.5.1.8 Soal No. 8** Maksud dari  $d\text{ train outputs} = \text{np utils.to categorical}(d['CLASS'].iloc[train idx])$  dan  $d\text{ test outputs} = \text{np utils.to categorical}(d['CLASS'].iloc[test idx])$

Fungsi pada kode program tersebut ditujukan untuk melakukan one-hot encoding supaya bisa masuk dan digunakan pada neural network. One-hot encoding diambil dari class yang berarti hanya terdapat 2 neuron, yaitu satu nol(1,0) atau nol satu(0,1) karena pilihannya hanya ada dua.

- Ilustrasi Gambar:



**Gambar 7.115** Compile model

#### 7.5.1.9 **Soal No. 9** Maksud dari listing 7.2.

Fungsi kode program tersebut untuk melakukan pemodelan dengan sequential, membandingkan setiap satu larik elemen dengan cara satu persatu secara beruntun. Terdapat 512 neuron inputan dengan input shape 2000 vektor yang sudah di-normalisasi. Lalu model dilakukan aktivasi dengan fungsi 'relu'. Kemudian pemotongan bobot supaya tidak overfitting sebesar 50 persen dari neuron inputan 512. Lalu pada layer output terdapat 2 neuron outputan yaitu nol (1,0) atau nol satu (0,1). Kemudian outputan tersebut diaktifasi menggunakan fungsi softmax.

#### 7.5.1.10 **Soal No. 10** Maksud dari listing 7.3.

Fungsi kode program tersebut untuk model yang telah dibuat selanjutnya dicompile dengan menggunakan algoritma optimisasi, fungsi loss, dan fungsi metrik.

#### 7.5.1.11 **Soal No. 11** Deep Learning

Deep learning, yang bisa diartikan sebagai rangkaian metode untuk melatih jaringan saraf buatan multi-lapisan. Ternyata, metode ini efektif dalam mengidentifikasi pola dari data. Manakala media membicarakan jaringan saraf, kemungkinan yang dimaksud adalah deep learning.

**7.5.1.12 Soal No. 12** Deep Neural Network, dan apa bedanya dengan Deep Learning

Algoritma DNN (Deep Neural Networks) adalah salah satu algoritma berbasis jaringan saraf yang dapat digunakan untuk pengambilan keputusan. Contoh yang dibahas kali ini adalah mengenai penentuan penerimaan pengajuan kredit sepeda motor baru berdasarkan kelompok data yang sudah ada.

Pembedannya dengan Deep Learning adalah terletak pada kedalaman model, deep learning adalah frasa yang digunakan untuk jaringan saraf yang kompleks.

**7.5.1.13 Soal No. 13** Jelaskan dengan ilustrasi gambar buatan sendiri, bagaimana perhitungan algoritma konvolusi dengan ukuran stride (NPM mod3+1)x(NPM mod3+1) yang terdapat max pooling.(nilai 30)

Karena NPM saya 1164067 dan hasil dari  $(NPM \bmod 3) + 1 = 2$ , maka saya menggunakan matrik kernel berukuran  $2 \times 2$ . Misalkan  $f(x,y)$  yang digunakan berukuran  $3 \times 3$  dan kernel atau mask berukuran  $2 \times 2$  adalah sebagai berikut:

Gambar Matriks:

$$F(x,y) = \begin{matrix} 2 & 3 & 5 \\ 1 & 0 & 8 \\ 7 & 2 & 0 \end{matrix} \quad \begin{matrix} 1 & 0 \\ 2 & 4 \end{matrix}$$

**Gambar 7.116** Perhitungan algoritma konvolusi

Penyelesaian dari operasi konvolusi antara  $f(x,y)$  dengan kernel  $g(x,y)$  adalah  $f(x,y) * g(x,y)$

- Tempatkan matrik kernel di sebelah kiri atas, lalu hitung nilai piksel pada posisi (0,0) dari kernel tersebut. Konvolusi dihitung dengan cara berikut :

$$(2 \times 1) + (3 \times 0) + (1 \times 2) + (0 \times 4)$$

Sehingga didapat hasil konvolusi = 4

- Tempatkan matrik kernel di sebelah kanan atas, lalu hitung nilai piksel pada posisi (0,0) dari kernel tersebut. Konvolusi dihitung dengan cara berikut :

$$(3 \times 1) + (5 \times 0) + (0 \times 2) + (8 \times 4)$$

Sehingga didapat hasil konvolusi = 35

- Tempatkan matrik kernel di sebelah kiri bawah, lalu hitung nilai piksel pada posisi (0,0) dari kernel tersebut. Konvolusi dihitung dengan cara berikut :

$$(1 \times 1) + (0 \times 0) + (7 \times 2) + (2 \times 4)$$

Sehingga didapat hasil konvolusi = 23

- Tempatkan matrik kernel di sebelah kanan bawah, lalu hitung nilai piksel pada posisi (0,0) dari kernel tersebut. Konvolusi dihitung dengan cara berikut :  
 $(0 \times 1) + (8 \times 0) + (2 \times 2) + (0 \times 4)$   
Sehingga didapat hasil konvolusi = 4  
Hasil:

$$\begin{matrix} & 4 & 35 \\ 23 & & 4 \end{matrix}$$

**Gambar 7.117** Hasil

## 7.5.2 Praktek

- 7.5.2.1 Soal No. 1** Jelaskan arti dari setiap baris kode program pada blok # In[1] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:46:02 2020
4
5 @author: User
6 """
7
8 import csv
9 from PIL import Image as pil_image
10 import keras.preprocessing.image

```

- Penjelasan:

Baris Code 1: Memasukkan atau mengimport file csv

Baris Code 2: Memasukkan module image sebagai pil\_image dari library PIL

Baris Code 3: Memasukkan atau mengimport fungsi keras.preprocessing.image

- Hasil output:

```

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

```

**Gambar 7.118** 1

**7.5.2.2 Soal No. 2** Jelaskan arti dari setiap baris kode program pada blok # In[2] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:25 2020
4
5 @author: User
6 """
7
8 imgs = []
9 classes = []
10 with open('HASYv2/hasy-data-labels.csv') as csvfile:
11     csvreader = csv.reader(csvfile)
12     i = 0
13     for row in csvreader:
14         if i > 0:
15             img = keras.preprocessing.image.img_to_array(
16                 pil_image.open("HASYv2/" + row[0]))
17                 # neuron activation functions behave best when input
18                 # values are between 0.0 and 1.0 (or -1.0 and 1.0),
19                 # so we rescale each pixel value to be in the range
20                 # 0.0 to 1.0 instead of 0-255
21                 img /= 255.0
22             imgs.append((row[0], row[2], img))
23             classes.append(row[2])
24         i += 1

```

- Penjelasan:

Baris Code 1: Membuat variabel imgs tanpa parameter

Baris Code 2: Membuat variabel classes tanpa parameter

Baris Code 3: Membuka file HASYv2/hasy-data-labels.csv

Baris Code 4: Membuat variabel csvreader untuk pembacaan dari file csv yang dimasukkan

Baris Code 5: Membuat variabel i dengan parameter 0

Baris Code 6: Mengeksekusi baris dari pembacaan csv

Baris Code 7: Mengaplikasikan perintah "if" dengan ketentuan variabel i lebih besar dari angka 0, maka akan dilanjutkan ke perintah berikutnya

Baris Code 8: Membuat variabel img yang mengubah image menjadi bentuk array dari file HASYv2 yang dibuka dengan row berparameter 0.

Baris Code 9: Membuat variabel img atau dengan nilai 255.0

Baris Code 10: Mendefinisikan fungsi imgs.append dimana merupakan proses melampirkan atau menggabungkan data dengan file lain atau set data yang ditentukan dengan 3 parameter yaitu row[0], row[2] dan variabel img.

Baris Code 11: Mendefinisikan fungsi append kembali dari variabel classes dengan parameternya row[2].

Baris Code 12: Mendefinisikan fungsi dimana i variabel i akan ditambah nilainya sehingga akan bernilai 1.

- Hasil output:

```
In [2]: imgs = []
...: classes = []
...: with open('HASYv2/hasy-data-labels.csv') as csvfile:
...:     csvreader = csv.reader(csvfile)
...:     i = 0
...:     for row in csvreader:
...:         if i > 0:
...:             img =
...:             keras.preprocessing.image.img_to_array(pil_image.open("HASYv2/" + row[0]))
...:             # neuron activation functions behave best when input values are
between 0.0 and 1.0 (or -1.0 and 1.0),
...:             # so we rescale each pixel value to be in the range 0.0 to 1.0
instead of 0-255
...:             img /= 255.0
...:             imgs.append((row[0], row[2], img))
...:             classes.append(row[2])
...:             i += 1
```

**Gambar 7.119** 2

**7.5.2.3 Soal No. 3** Jelaskan arti dari setiap baris kode program pada blok # In[3] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020
4
5 @author: User
6 """
7
8 import random
9 random.shuffle(imgs)
10 split_idx = int(0.8*len(imgs))
11 train = imgs[:split_idx]
12 test = imgs[split_idx:]
```

- Penjelasan:

Baris Code 1: Memasukkan module random

Baris Code 2: Melakukan pengocokan atau pengacakkan pada module random dengan parameter variabelnya imgs

Baris Code 3: Membagi dan memecah index dalam bentuk integer dengan mengkalikan nilai 0,8 dan fungsi len yang akan mengembalikan jumlah item dalam datanya dari variabel imgs

Baris Code 4: Membuat variabel train yang mengeksekusi imgs dengan pemecahan index awal pada data

Baris Code 5: Membuat variabel test yang mengeksekusi imgs dengan pemecahan index akhir pada data

- Hasil output:

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

**Gambar 7.120** 3

**7.5.2.4 Soal No. 4** Jelaskan arti dari setiap baris kode program pada blok # In[4] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020
4
5 @author: User
6 """
7
8 import numpy as np
9
10 train_input = np.asarray(list(map(lambda row: row[2], train)))
11 test_input = np.asarray(list(map(lambda row: row[2], test)))
12
13 train_output = np.asarray(list(map(lambda row: row[1], train)))
14 test_output = np.asarray(list(map(lambda row: row[1], test)))
```

- Penjelasan:

Baris Code 1: Mengimport library numpy sebagai np

Baris Code 2: Membuat variabel train\_input untuk input menjadi sebuah array dari np menggunakan fungsi list untuk mengkoleksikan data yang dipilih dan diubah. Didalamnya diterapkan fungsi map untuk mengembalikan iterator dari datanya dengan memfungksikan lamda pada row dengan parameter [2] untuk membuat objek fungsi menjadi lebih kecil dan mudah dieksekusi dari variabel train.

Baris Code 3: Membuat variabel test\_input dengan fungsi seperti train\_input yang membedakan hanya datanya atau inputan yang diproses berasal dari variabel test

Baris Code 4: Membuat variabel train\_output untuk mengubah keluaran menjadi sebuah array dari np dengan menggunakan fungsi list untuk mengkoleksi

data yang dipilih dan diubah. Didalamnya diterapkan fungsi map untuk mengembalikan iterator dari datanya dengan memfungskan lamda pada row dengan parameter[1] untuk membuat objek fungsi menjadi lebih kecil dan mudah dieksekusi dari variabel train.

Baris Code 5: Membuat variabel test\_output dengan fungsi yang sama seperti train\_output yang membedakan hanya datanya atau inputan yang diproses berdasarkan dari variabel test

- Hasil output:

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

**Gambar 7.121** 4

**7.5.2.5 Soal No. 5** Jelaskan arti dari setiap baris kode program pada blok # In[5] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020
4
5 @author: User
6 """
7
8 from sklearn.preprocessing import LabelEncoder
9 from sklearn.preprocessing import OneHotEncoder
```

- Penjelasan:

Baris Code 1: Memasukkan modul atau fungsi LabelEncoder dari sklearn.preprocessing untuk dapat digunakan menormalkan label dimana label encoder didefinisikan dengan nilai antara 0 dan n\_classes-1.

Baris Code 2: Memasukkan modul atau fungsi OneHotEncoder dari sklearn.preprocessing untuk mendefinisikan fitur input dimana mengambil nilai dalam kisaran [0, maks (nilai)).

- Hasil output:

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

**Gambar 7.122** 5

**7.5.2.6 Soal No. 6** Jelaskan arti dari setiap baris kode program pada blok # In[6] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020
4
5 @author: User
6 """
7
8 label_encoder = LabelEncoder()
9 integer_encoded = label_encoder.fit_transform(classes)

```

- Penjelasan:

Baris Code 1: Membuat variabel label\_encoder dengan modul atau fungsi dari LabelEncoder tanpa parameter

Baris Code 2: Membuat variabel integer\_encoded dengan fungsi label\_encoder.fit\_transform dari variabel classes yang akan mengembalikan beberapa data yang diubah kembali dari variabel label\_encoder.

- Hasil output:

```
In [6]: label_encoder = LabelEncoder()
...: integer_encoded = label_encoder.fit_transform(classes)
```

Gambar 7.123 6

**7.5.2.7 Soal No. 7** Jelaskan arti dari setiap baris kode program pada blok # In[7] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:27 2020
4
5 @author: User
6 """
7
8 onehot_encoder = OneHotEncoder(sparse=False)
9 integer_encoded = integer_encoded.reshape(len(integer_encoded),
10 1)
onehot_encoder.fit(integer_encoded)

```

- Penjelasan:

Baris 1: Membuat variabel onehot\_encoder yang memanggil fungsi OneHotEncoder tanpa mengembalikan matriks karena sparse=false.

Baris 2: Membuat variabel integer\_encoded memanggil variabel integer\_encoded pada kode program 6 untuk dieksekusi memberikan bentuk baru ke array tanpa mengubah datanya dari mengembalikan panjang nilai dari integer\_encoded.

Baris 3: Onehotencoding melakukan fitting pada integer\_encoded.

- Hasil output:

```
In [7]: onehot_encoder = OneHotEncoder(sparse=False)
...: integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
...: onehot_encoder.fit(integer_encoded)
D:\Anaconda\lib\site-packages\sklearn\preprocessing\_encoders.py:371: FutureWarning:
The handling of integer data will change in version 0.22. Currently, the categories
are determined based on the range [0, max(values)], while in the future they will be
determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify
"categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories
to integers, then you can now use the OneHotEncoder directly.
    warnings.warn(msg, FutureWarning)
Out[7]:
OneHotEncoder(categorical_features=None, categories=None,
               dtype=<class 'numpy.float64'>, handle_unknown='error',
               n_values=None, sparse=False)
```

Gambar 7.124 7

**7.5.2.8 Soal No. 8** Jelaskan arti dari setiap baris kode program pada blok # In[8] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:28 2020
4
5 @author: User
6 """
7
8 train_output_int = label_encoder.transform(train_output)
9 train_output = onehot_encoder.transform(train_output_int.reshape(
10     len(train_output_int), 1))
11 test_output_int = label_encoder.transform(test_output)
12 test_output = onehot_encoder.transform(test_output_int.reshape(
13     len(test_output_int), 1))
14 num_classes = len(label_encoder.classes_)
15 print("Number of classes: %d" % num_classes)
```

- Penjelasan:

Baris 1: Membuat variabel train\_output\_int yang mengeksekusi label\_encoder dengan mengubah nilai dari parameter variabel train\_output.

Baris 2: Membuat variabel train\_output yang mengeksekusi variabel onehot\_encoder dari kode program 7 dengan mengubah nilai dari variabel parameter train\_output\_int

yang datanya sudah diubah kedalam bentuk array dan panjang nilai dari train\_output\_int telah dikembalikan.

Baris 3: Membuat variabel test\_output\_int yang mengeksekusi label\_encoder dengan mengubah nilai dari parameter variabel test\_output.

Baris 4: Membuat variabel test\_output yang mengeksekusi variabel onehot\_encoder dari kode program 7 dengan mengubah nilai dari variabel parameter test\_output\_int yang datanya sudah diubah kedalam bentuk array dan panjang nilai dari test\_output\_int telah dikembalikan.

Baris 5: Membuat variabel num\_classes untuk mengetahui jumlah class dari label\_encoder

Baris 6: Perintah print digunakan untuk memunculkan hasil dari variabel num\_classes

```
In [8]: train_output_int = label_encoder.transform(train_output)
...: train_output =
onehot_encoder.transform(train_output_int.reshape(len(train_output_int), 1))
...: test_output_int = label_encoder.transform(test_output)
...: test_output =
onehot_encoder.transform(test_output_int.reshape(len(test_output_int), 1))
...:
...: num_classes = len(label_encoder.classes_)
...: print("Number of classes: %d" % num_classes)
Number of classes: 369
```

**Gambar 7.125** 8

- Hasil output:

**7.5.2.9 Soal No. 9** Jelaskan arti dari setiap baris kode program pada blok # In[9] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:47:28 2020
4
5 @author: User
6 """
7
8 from keras.models import Sequential
9 from keras.layers import Dense, Dropout, Flatten
10 from keras.layers import Conv2D, MaxPooling2D
```

- Penjelasan:

Baris 1: Melakukan importing fungsi model sequential dari library keras.

Baris 2: Melakukan importing fungsi layer dense, dropout, dan flatten dari library keras.

Baris 3: Melakukan importing fungsi layer Conv2D dan MaxPooling2D dari library keras.

- Hasil output:

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

**Gambar 7.126** 9

**7.5.2.10 Soal No. 10** Jelaskan arti dari setiap baris kode program pada blok # In[10] dan hasil luarannya.

- Code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 model = Sequential()
9 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
10                  input_shape=np.shape(train_input[0])))
11 model.add(MaxPooling2D(pool_size=(2, 2)))
12 model.add(Conv2D(32, (3, 3), activation='relu'))
13 model.add(MaxPooling2D(pool_size=(2, 2)))
14 model.add(Flatten())
15 model.add(Dense(1024, activation='tanh'))
16 model.add(Dropout(0.5))
17 model.add(Dense(num_classes, activation='softmax'))
18
19 model.compile(loss='categorical_crossentropy', optimizer='adam',
20                 metrics=['accuracy'])
21
22 print(model.summary())
```

- Penjelasan:

Baris 1: Melakukan pemodelan Sequential.

Baris 2: Menambahkan Konvolusi 2D dengan 32 filter konvolusi masing-masing berukuran 3x3 dengan algoritam activation relu dengan data dari train input mulai dari baris nol.

Baris 3: Menambahkan Max Pooling dengan matriks 2x2.

Baris 4: Penambahan Konvolusi 2D dengan 32 filter, konvolusi masing-masing berukuran 3x3 dengan algoritam activation relu.

Baris 5 Menambahkan Max Pooling dengan matriks 2x2.

Baris 6: Mendefinisikan inputan dengan 1024 neuron dan menggunakan algoritma tanh untuk activationnya.

Baris 7: Dropout terdiri dari pengaturan secara acak tingkat pecahan unit input ke 0 pada setiap pembaruan selama waktu pelatihan, yang membantu mencegah overfitting sebesar 50% .

Baris 8: Untuk output layer menggunakan data dari variabel num classes dengan fungsi activationnya softmax.

Baris 9: Konfigurasi proses pembelajaran, melalui metode compile, sebelum melatih suatu model.

Baris 10: Menampilkan model yang telah dibuat.

- Hasil output:

```
...: model.add(Flatten())
...: model.add(Dense(1024, 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())
WARNING:tensorflow:From D:\Anaconda\lib\site-packages\tensorflow\python\framework
\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From D:\Anaconda\lib\site-packages\keras\backend
\backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with
keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.
-----
```

| Layer (type)                   | Output Shape       | Param # |
|--------------------------------|--------------------|---------|
| conv2d_1 (Conv2D)              | (None, 30, 30, 32) | 896     |
| max_pooling2d_1 (MaxPooling2D) | (None, 15, 15, 32) | 0       |
| conv2d_2 (Conv2D)              | (None, 13, 13, 32) | 9248    |
| max_pooling2d_2 (MaxPooling2D) | (None, 6, 6, 32)   | 0       |
| flatten_1 (Flatten)            | (None, 1152)       | 0       |
| dense_1 (Dense)                | (None, 1024)       | 1180672 |
| dropout_1 (Dropout)            | (None, 1024)       | 0       |
| dense_2 (Dense)                | (None, 369)        | 378225  |

```
=====
Total params: 1,569,041
Trainable params: 1,569,041
Non-trainable params: 0
-----
```

---

None

**Gambar 7.127** 10

**7.5.2.11 Soal No. 11** Jelaskan arti dari setiap baris kode pada blok # In[11] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:11 2020
4
5 @author: User
6 """
7
8
9 import keras.callbacks
10 tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-
    style')

```

- Penjelasan:

Baris Code 1: Melakukan import library keras.callbacks yang digunakan pada penulisan log untuk TensorBoard, untuk memvisualisasikan grafik dinamis dari pelatihan dan metrik pengujian.

Baris Code 2: Membuat variabel tensorboard untuk mendefinisikan fungsi TensorBoard pada keras.callbacks yang digunakan sebagai alat visualisasi yang disediakan dengan TensorFlow. Dan untuk fungsi log\_dir memanggil data yaitu './logs/mnist-style'

- Hasil output:

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

**Gambar 7.128** 11

**7.5.2.12 Soal No. 12** Jelaskan arti dari setiap baris kode program pada blok # In[12] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:11 2020
4
5 @author: User
6 """
7
8 model.fit(train_input, train_output,
9            batch_size=32,
10           epochs=10,
11           verbose=2,
12           validation_split=0.2,
13           callbacks=[tensorboard])
14
15 score = model.evaluate(test_input, test_output, verbose=2)
16 print('Test loss:', score[0])
17 print('Test accuracy:', score[1])

```

- Penjelasan:

Baris Code 1: Menerapkan fungsi model.fit yang didalamnya memproses train\_input, train\_output

Baris Code 2: Penerapan fungsi yang sama difungsikan batch\_size apabila batch\_sizenya tidak ditemukan maka otomatis akan dijadikan nilai 32

Baris Code 3: Penerapan fungsi yang sama, difungsikan epochs dimana perulangan dari berapa kali nilai yang digunakan untuk data, dan jumlahnya ialah 10

Baris Code 4: Mendefinisikan fungsi verbose untuk digunakan sebagai opsi menghasilkan informasi logging dari data yang ditentukan dengan nilai 2

Baris Code 5: Mendefinisikan fungsi validation\_split untuk memecah nilai dari perhitungan validasinya sebesar 0,2. (Fraksi data pelatihan untuk digunakan sebagai data validasi)

Baris Code 6: Mendefinisikan fungsi callbacks dengan parameternya yang mengeksekusi tensorboard dimana digunakan untuk visualisasikan parameter training, metrik, hiperparameter pada nilai/data yang diproses

Baris Code 7: Mendefinisikan variabel score dengan fungsi evaluate dari model dengan parameter test\_input, tst\_output dan verbose=2 untuk memprediksi output dan input yang diberikan dan kemudian menghitung fungsi metrik yang ditentukan dalam modelnya

Baris Code 8: Mencetak score optimasi dari test dengan ketentuan nilai parameter 0

Baris Code 9: Mencetak score akurasi dari test dengan ketentuan nilai parameter 1

- Hasil output:

**7.5.2.13 Soal No. 13** Jelaskan arti dari setiap baris kode program pada blok # In[13] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 import time
9
10 results = []
11 for conv2d_count in [1, 2]:
12     for dense_size in [128, 256, 512, 1024, 2048]:
13         for dropout in [0.0, 0.25, 0.50, 0.75]:
14             model = Sequential()

```

```
In [12]: model.fit(train_input, train_output,
...:     batch_size=32,
...:     epochs=10,
...:     verbose=2,
...:     validation_split=0.2,
...:     callbacks=[tensorboard])
...:
...: score = model.evaluate(test_input, test_output, verbose=2)
...: print('Test loss:', score[0])
...: print('Test accuracy:', score[1])
WARNING:tensorflow:From D:\Anaconda\lib\site-packages\tensorflow\python\ops\numpy_ops.py:3066: to_int32 (from tensorflow.python.ops.numpy_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 107668 samples, validate on 26918 samples
Epoch 1/10
- 2007s - loss: 1.5334 - acc: 0.6294 - val_loss: 0.9824 - val_acc: 0.7285
Epoch 2/10
- 918s - loss: 0.9694 - acc: 0.7321 - val_loss: 0.9162 - val_acc: 0.7494
Epoch 3/10
- 562s - loss: 0.8543 - acc: 0.7556 - val_loss: 0.8883 - val_acc: 0.7530
Epoch 4/10
- 503s - loss: 0.7854 - acc: 0.7699 - val_loss: 0.8441 - val_acc: 0.7666
Epoch 5/10
- 361s - loss: 0.7364 - acc: 0.7796 - val_loss: 0.8491 - val_acc: 0.7663
Epoch 6/10
- 366s - loss: 0.6936 - acc: 0.7899 - val_loss: 0.8522 - val_acc: 0.7692
Epoch 7/10
- 360s - loss: 0.6591 - acc: 0.7962 - val_loss: 0.8580 - val_acc: 0.7668
Epoch 8/10
- 389s - loss: 0.6344 - acc: 0.8017 - val_loss: 0.8496 - val_acc: 0.7654
Epoch 9/10
- 357s - loss: 0.6076 - acc: 0.8072 - val_loss: 0.8597 - val_acc: 0.7638
Epoch 10/10
- 359s - loss: 0.5905 - acc: 0.8116 - val_loss: 0.8889 - val_acc: 0.7637
Test loss: 0.8794204677150739
Test accuracy: 0.7632181175230488
```

Gambar 7.129 12

```

15         for i in range(conv2d_count):
16             if i == 0:
17                 model.add(Conv2D(32, kernel_size=(3, 3),
18 activation='relu', input_shape=np.shape(train_input[0])))
19             else:
20                 model.add(Conv2D(32, kernel_size=(3, 3),
21 activation='relu'))
22                 model.add(MaxPooling2D(pool_size=(2, 2)))
23                 model.add(Flatten())
24                 model.add(Dense(dense_size, activation='tanh'))
25                 if dropout > 0.0:
26                     model.add(Dropout(dropout))
27                 model.add(Dense(num_classes, activation='softmax'))
28
29                 model.compile(loss='categorical_crossentropy',
30 optimizer='adam', metrics=['accuracy'])
31
32                 log_dir = './logs/conv2d_%d-dense_%d-dropout_%.2f' %
33 (conv2d_count, dense_size, dropout)
34                 tensorboard = keras.callbacks.TensorBoard(log_dir=
log_dir)
35
36                 start = time.time()
37                 model.fit(train_input, train_output, batch_size=32,
38 epochs=10,
39 verbose=0, validation_split=0.2, callbacks
=[tensorboard])
40                 score = model.evaluate(test_input, test_output,
verbose=2)
41                 end = time.time()
42                 elapsed = end - start
43                 print("Conv2D count: %d, Dense size: %d, Dropout: %.2
f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count
, dense_size, dropout, score[0], score[1], elapsed))
44                 results.append((conv2d_count, dense_size, dropout,
score[0], score[1], elapsed))

```

▪ Penjelasan:

Baris 1: Import atau memasukkan modul time

Baris 2: Variabel result berisikan array kosong

Baris 3: Menggunakan convolution 2D yang berarti memiliki 1 atau 2 layer

Baris 4: Mendefinisikan dense size dengan ukuran 128, 256, 512, 1024, 2048

Baris 5: Mendefinsikan drop out dengan 0, 25%, 50%, dan 75%

Baris 6: Melakukan pemodelan Sequential

Baris 7: Jika ini adalah layer pertama, akan memasukkan bentuk input.

Baris 8: Kalau tidak kita hanya akan menambahkan layer.

Baris 9: Kemudian, setelah menambahkan layer konvolusi, lakukan hal yang sama dengan max pooling.

Baris 10: Lalu, ratakan atau flatten dan menambahkan dense size ukuran apa pun yang berasal dari dense size. Dimana akan selalu menggunakan algoritma tanh

Baris 11: Jika dropout digunakan, akan menambahkan layer dropout. Dropout ini berarti misalnya 50%, bahwa setiap kali memperbarui bobot setelah setiap batch, ada peluang 50% untuk setiap bobot yang tidak akan diperbarui

Baris 12: Menempatkan di antara dua lapisan padat untuk dihindarkan dari melindunginya dari overfitting. Lapisan terakhir akan selalu menjadi jumlah kelas karena itu harus, dan menggunakan softmax. Itu dikompilasi dengan cara yang sama.

Baris 13: Atur direktori log yang berbeda untuk TensorBoard sehingga dapat membedakan konfigurasi yang berbeda.

Baris 14: Variabel start akan memanggil modul time

Baris 15: Melakukan fit atau compile

Baris 16: Melakukan scoring dengan evaluate yang akan menampilkan data loss dan accuracy dari model

Baris 17: 'End' merupakan variabel untuk melihat waktu akhir pada saat pemodelan berhasil dilakukan.

Baris 18: Menampilkan hasil dari skrip diatas

- Hasil output:

```
....  
....      model.compile(loss='categorical_crossentropy', optimizer='adam',  
metrics=['accuracy'])  
....  
....      log_dir = './logs/conv2d_%d-dense_%d-dropout_.2f' % (conv2d_count,  
dense_size, dropout)  
....      tensorboard = keras.callbacks.TensorBoard(log_dir=log_dir)  
....  
....      start = time.time()  
....      model.fit(train_input, train_output, batch_size=32, epochs=10,  
....                  verbose=0, validation_split=0.2, callbacks=[tensorboard])  
....      score = model.evaluate(test_input, test_output, verbose=2)  
....      end = time.time()  
....      elapsed = end - start  
....      print("Conv2D count: %d, Dense size: %d, Dropout: %.2f - Loss: %.  
2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count, dense_size, dropout, score[0],  
score[1], elapsed))  
....      results.append((conv2d_count, dense_size, dropout, score[0],  
score[1], elapsed))  
Conv2D count: 1, Dense size: 128, Dropout: 0.00 - Loss: 1.13, Accuracy: 0.74, Time: 1540  
sec  
Conv2D count: 1, Dense size: 128, Dropout: 0.25 - Loss: 0.93, Accuracy: 0.76, Time: 1579  
sec  
Conv2D count: 1, Dense size: 128, Dropout: 0.50 - Loss: 0.81, Accuracy: 0.77, Time: 1599  
sec  
Conv2D count: 1, Dense size: 128, Dropout: 0.75 - Loss: 0.82, Accuracy: 0.77, Time: 1627
```

**7.5.2.14 Soal No. 14** Jelaskan arti dari setiap baris kode program pada blok # In[14] dan hasil luarnya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 model = Sequential()
9 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
10                 input_shape=np.shape(train_input[0])))
11 model.add(MaxPooling2D(pool_size=(2, 2)))
12 model.add(Conv2D(32, (3, 3), activation='relu'))
13 model.add(MaxPooling2D(pool_size=(2, 2)))
14 model.add(Flatten())
15 model.add(Dense(128, activation='tanh'))
16 model.add(Dropout(0.5))
17 model.compile(loss='categorical_crossentropy', optimizer='adam',
18                 metrics=['accuracy'])
18 print(model.summary())

```

- Penjelasan:

Baris 1: Melakukan pemodelan Sequential

Baris 2: Menambahkan Convolutio 2D dengan dmensi 32, dan ukuran matriks 3x3 dengan function aktivasi yang digunakan yaitu relu dan menampilkan input shape

Baris 3: Melakukan Max Pooling 2D dengan ukuran matriks 2x2

Baris 4: Melakukan Convolusi lagi dengan kriteria yang sama tanpa menambahkan input, ini dilakukan untuk mendapatkan data yang terbaik

Baris 5: Flatten digunakan untuk meratakan inputan atau masukkan data

Baris 6: Menambahkan dense input sebanyak 128 neuron dengan menggunakan function aktivasi tanh.

Baris 7: Dropout sebanyak 50% untuk menghindari overfitting

Baris 8: Menambahkan dense pada model untuk output dimana layerini akan menjadi jumlah dari class yang ada.

Baris 9: Mengcompile model yang didefinisikan diatas

Baris 10: Menampilkan ringkasan dari pemodelan yang dilakukan

- Hasil output:

```
In [14]: model = Sequential()
...: model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=np.shape(train_input[0])))
...: model.add(MaxPooling2D(pool_size=(2, 2)))
...: model.add(Conv2D(32, (3, 3), activation='relu'))
...: model.add(MaxPooling2D(pool_size=(2, 2)))
...: model.add(Flatten())
...: model.add(Dense(128, 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())
```

| Layer (type)                   | Output Shape       | Param # |
|--------------------------------|--------------------|---------|
| conv2d_3 (Conv2D)              | (None, 30, 30, 32) | 896     |
| max_pooling2d_3 (MaxPooling2D) | (None, 15, 15, 32) | 0       |
| conv2d_4 (Conv2D)              | (None, 13, 13, 32) | 9248    |
| max_pooling2d_4 (MaxPooling2D) | (None, 6, 6, 32)   | 0       |
| flatten_2 (Flatten)            | (None, 1152)       | 0       |
| dense_3 (Dense)                | (None, 128)        | 147584  |
| dropout_2 (Dropout)            | (None, 128)        | 0       |
| dense_4 (Dense)                | (None, 369)        | 47601   |
| Total params:                  | 205,329            |         |
| Trainable params:              | 205,329            |         |
| Non-trainable params:          | 0                  |         |

---

```
None
```

**Gambar 7.131** 14

**7.5.2.15 Soal No. 15** Jelaskan arti dari setiap baris kode program pada blok # In[15] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 model.fit(np.concatenate((train_input, test_input)),
9           np.concatenate((train_output, test_output)),
10          batch_size=32, epochs=10, verbose=2)

```

- Penjelasan:

Baris 1: Melakukan fit dengan join data train dan test agar dapat dilakukan pelatihan untuk jaringan pada semua data yang dimiliki.

- Hasil output:

```

In [26]: model.fit(np.concatenate((train_input, test_input)),
...:           np.concatenate((train_output, test_output)),
...:           batch_size=32, epochs=10, verbose=2)
Epoch 1/10
- 247s - loss: 1.7949 - acc: 0.5843
Epoch 2/10
- 236s - loss: 1.0797 - acc: 0.7064
Epoch 3/10
- 235s - loss: 0.9648 - acc: 0.7308
Epoch 4/10
- 237s - loss: 0.9060 - acc: 0.7434
Epoch 5/10
- 237s - loss: 0.8671 - acc: 0.7519
Epoch 6/10
- 236s - loss: 0.8362 - acc: 0.7573
Epoch 7/10
- 238s - loss: 0.8137 - acc: 0.7631
Epoch 8/10
- 239s - loss: 0.7980 - acc: 0.7652
Epoch 9/10
- 239s - loss: 0.7831 - acc: 0.7692
Epoch 10/10
- 239s - loss: 0.7695 - acc: 0.7724
Out[26]: <keras.callbacks.History at 0x14c1941f5f8>

```

Gambar 7.132 15

**7.5.2.16 Soal No. 16** Jelaskan arti dari setiap baris kode program pada blok # In[16] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """

```

```

3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 model.save("mathsymbols.model")

```

- Penjelasan:

Baris 1: Menyimpan model yang telah di latih dengan nama mathsymbols.model

- Hasil output:

```
In [22]: model.save("mathsymbols.model")
```

**Gambar 7.133** 16

**7.5.2.17 Soal No. 17** Jelaskan arti dari setiap baris kode program pada blok # In[17] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:50:12 2020
4
5 @author: User
6 """
7
8 np.save('classes.npy', label_encoder.classes_)

```

- Penjelasan:

Baris 1: Menyimpan label enkoder (untuk membalikkan one-hot encoder) dengan nama classes.npy

- Hasil output:

```
In [23]: np.save('classes.npy', label_encoder.classes_)
```

**Gambar 7.134** 17

**7.5.2.18 Soal No. 18** Jelaskan arti dari setiap baris kode program pada blok # In[18] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:53:35 2020
4

```

```

5 @author: User
6 """
7
8 import keras.models
9 model2 = keras.models.load_model("mathsymbols.model")
10 print(model2.summary())

```

▪ Penjelasan:

Baris 1: Memasukkan atau mengimport models dari librari Keras

Baris 2: Variabel model2 akan memanggil model yang telah disave sebelumnya

Baris 3: Menampilkan ringkasan dari hasil pemodelan

▪ Hasil output:

```

In [24]: import keras.models
      ...: model2 = keras.models.load_model("mathsymbols.model")
      ...: print(model2.summary())

Layer (type)                 Output Shape              Param #
=====
conv2d_3 (Conv2D)            (None, 30, 30, 32)      896
max_pooling2d_3 (MaxPooling2 (None, 15, 15, 32)      0
conv2d_4 (Conv2D)            (None, 13, 13, 32)      9248
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 32)        0
flatten_2 (Flatten)          (None, 1152)             0
dense_3 (Dense)              (None, 128)              147584
dropout_2 (Dropout)          (None, 128)              0
dense_4 (Dense)              (None, 369)              47601
=====
Total params: 205,329
Trainable params: 205,329
Non-trainable params: 0
=====
None

```

Gambar 7.135 18

**7.5.2.19 Soal No. 19** Jelaskan arti dari setiap baris kode program pada blok # In[19] dan hasil luarnya.

▪ Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:53:36 2020
4
5 @author: User
6 """
7

```

```

8 label_encoder2 = LabelEncoder()
9 label_encoder2.classes_ = np.load('classes.npy')
10
11 def predict(img_path):
12     newimg = keras.preprocessing.image.img_to_array(pil_image.
13         open(img_path))
14     newimg /= 255.0
15
16     # do the prediction
17     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
18
19     # figure out which output neuron had the highest score, and
20     # reverse the one-hot encoding
21     inverted = label_encoder2.inverse_transform([np.argmax(
22         prediction)]) # argmax finds highest-scoring output
23     print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
24         max(prediction)))

```

▪ Penjelasan:

Baris 1: Memanggil fungsi LabelEncoder

Baris 2: Variabel label encoder akan memanggil class yang disimpan sebelumnya.

Baris 3: Function Predict akan mengubah gambar kedalam bentuk array

Baris 4: Variabel prediction akan melakukan prediksi untuk model2 dengan reshape variabel newimg dengan bentukarray 4D.

Baris 5: Variabel inverted akan mencari nilai tertinggi output dari hasil prediksi tadi

▪ Hasil output:

```

In [25]: label_encoder2 = LabelEncoder()
...: label_encoder2.classes_ = np.load('classes.npy')
...:
...: def predict(img_path):
...:     newimg =
keras.preprocessing.image.img_to_array(pil_image.open(img_path))
...:     newimg /= 255.0
...:
...:     # do the prediction
...:     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
...:
...:     # figure out which output neuron had the highest score, and reverse the
one-hot encoding
...:     inverted = label_encoder2.inverse_transform([np.argmax(prediction)]) # argmax finds highest-scoring output
...:     print("Prediction: %s, confidence: %.2f" % (inverted[0],
np.max(prediction)))

```

Gambar 7.136 19

**7.5.2.20 Soal No. 20** Jelaskan arti dari setiap baris kode program pada blok # In[20] dan hasil luarannya.

- Code:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 13 23:54:14 2020
4
5 @author: User
6 """
7
8 predict("HASYv2/hasy-data/v2-00010.png")
9
10 predict("HASYv2/hasy-data/v2-00500.png")
11
12 predict("HASYv2/hasy-data/v2-00700.png")

```

- Penjelasan:

Baris 1: Melakukan prediksi dari pelatihan dari gambar v2-00010.png

Baris 2: Melakukan prediksi dari pelatihan dari gambar v2-00500.png

Baris 3: Melakukan prediksi dari pelatihan dari gambar v2-00700.png

- Hasil output:

```

In [26]: predict("HASYv2/hasy-data/v2-00010.png")
...:
...: predict("HASYv2/hasy-data/v2-00500.png")
...:
...: predict("HASYv2/hasy-data/v2-00700.png")
Prediction: \pitchfork, confidence: 0.00
Prediction: \copyright, confidence: 0.00
Prediction: J, confidence: 0.00

```

**Gambar 7.137** 20

### 7.5.3 Penanganan Error

#### 7.5.3.1 Penanganan Error

- Screenshoot:

```

Traceback (most recent call last):
  File "<ipython-input-3-fd9abfd31556>", line 1, in <module>
    model.fit(np.concatenate((train_input, test_input)),
NameError: name 'model' is not defined

```

**Gambar 7.138** Error

- Code Error:

NameError: name 'model' is not defined

- Penanganan Error:

Berdasarkan error maka penyelesaiannya ialah melakukan pendefinisian variabel model sehingga code dapat dijalankan. Lakukan running pada code yang berada diatasnya dimana mendefinisikan variabel model.

## 7.6 1174009 - Dwi Yulianingsih

### 7.6.1 Teori

#### 1. Kenapa file Text harus di Tokenizer

Tokenizer adalah langkah pertama yang diperlukan dalam tugas pemrosesan bahasa, seperti penghitungan kata, penguraian, pemeriksaan ejaan, pembuatan corpus, dan analisis statistik teks. Itu mengkonversi string teks Python menjadi aliran objek token, di mana setiap objek token adalah kata yang terpisah, tanda baca, nomor / jumlah, tanggal, email, URL / URI, dll. Ini juga mengelompokkan aliran token menjadi kalimat, dengan mempertimbangkan sudut kasus-kasus seperti singkatan dan tanggal di tengah kalimat. ilustrasi dapat dilihat pada gambar



**Gambar 7.139** Ilustrasi Tokenisasi

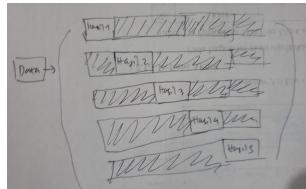
#### 2. Konsep dasar K Fold Cross Validation pada dataset Komentar Youtube

```

1 kfolds = StratifiedKFold(n_splits=5)
2 splits = kfolds.split(d, d['CLASS'])
  
```

**Listing 7.4** K Fold Cross Validation

pada Code pada Listing 7.4 terdapat penggunaan K Fold dengan metode yang akan membagi data dalam 5 hasil pengulangan dari pembagian data yang akan dilakukan pada Attribute CLASS. ilustrasinya dapat dilihat pada Gambar



**Gambar 7.140** Ilustrasi K Fold Cross Validation pada dataset Komentar Youtube

3. Jelaskan apa maksudnya kode program *for train, test in splits*

Membagi susunan atau matriks menjadi rangkaian acak, dimana data yang dimasukan akan dibagi dengan porsi yang sesuai dengan Train dan Test datanya. Untuk lebih jelas tentang bagaimana penggunaannya dapat dilihat pada Gambar

```
Examples
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> X, y = np.arange(10).reshape((5, 2)), range(5)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
>>> list(y)
[0, 1, 2, 3, 4]

>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.33, random_state=42)
...
>>> X_train
array([[4, 5],
       [0, 1],
       [6, 7]])
>>> y_train
[2, 0, 3]
>>> X_test
array([[2, 3],
       [8, 9]])
>>> y_test
[1, 4]

>>> train_test_split(y, shuffle=False)
[[0, 1, 2], [3, 4]]
```

Gambar 7.141 Ilustrasi penggunaan Train dan Test Split

4. Maksud Code program *train\_content = d['CONTENT'].iloc[train\_idx]* dan *test\_content = d['CONTENT'].iloc[train\_idx]*

Pada code tersebut dimaksudkan untuk membaca data isian yang terdapat pada kolom yang bernama attributnya CONTENT sebagai data TRAIN dan data TEST, untuk ilustrasinya dapat dilihat pada Gambar

```
Examples
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> X, y = np.arange(10).reshape((5, 2)), range(5)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
>>> list(y)
[0, 1, 2, 3, 4]

>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.33, random_state=42)
...
>>> X_train
array([[4, 5],
       [0, 1],
       [6, 7]])
>>> y_train
[2, 0, 3]
>>> X_test
array([[2, 3],
       [8, 9]])
>>> y_test
[1, 4]

>>> train_test_split(y, shuffle=False)
[[0, 1, 2], [3, 4]]
```

Gambar 7.142 Ilustrasi penggunaan Train dan Test Split

5. Maksud dari fungsi *tokenizer = Tokenizer(num\_words=2000)* dan *tokenizer.fit\_on\_texts(train\_content)*

Pada code tersebut dimana penggunaan Tokenizer berguna untuk membaca data teks yang berformat kalimat untuk disusun sebanyak 2000 kata, fungsi *fit\_on\_texts(train\_content)* untuk membaca data Teks Tokenizer tadi untuk dimasukan kedalam kolom

CONTENT pada dataset. untuk contoh ilustrasi data yang digunakan dapat dilihat pada Gambar

| Document ID | Author        | Date                    | Content         | Date |
|-------------|---------------|-------------------------|-----------------|------|
| cl10793718  | Lina Merita   | 2021-09-29T03:26:45Z000 | NYC NYC NYC NYC | -    |
| cl10793719  | Maryam        | 2021-09-29T03:26:45Z000 | NYC NYC NYC NYC | -    |
| cl10793720  | Aditya Hana   | 2021-09-29T03:26:45Z000 | NYC NYC NYC NYC | -    |
| cl10793721  | Shabir Naseer | 2021-09-29T03:26:45Z000 | NYC NYC NYC NYC | -    |
| cl10793722  | Ugochukwu Obi | 2021-09-29T03:26:45Z000 | NYC NYC NYC NYC | -    |
| cl10793723  | Reyhan El     | 2021-09-29T03:26:45Z000 | NYC NYC NYC NYC | -    |

Gambar 7.143 data Content yang dimaksud

6. maksud dari fungsi  $d\_train\_inputs = tokenizer.texts\_to\_matrix(train\_content, mode='tfidf')$  dan  $d\_test\_inputs = tokenizer.texts\_to\_matrix(test\_content, mode='tfidf')$

pada code yang pertama terdapat fungsi yang digunakan untuk mengolah data variable d\_train\_inputs dan d\_test\_inputs dengan menggunakan tokernizer dengan fungsi yang merubah data teks menjadi data matrix dengan pemrosesan datanya sesuai dengan data Variable train\_content dan test\_content dengan metodenya adalah TF-IDF. untuk contoh ilustrasi dari penjelasan tersebut dapat dilihat pada Gambar

```
In [34]: from keras.preprocessing.text import Tokenizer
...: # define 2 documents
...: docs = ["Well done!",
...:        "Good work"]
...: # create a tokenizer
...: t = Tokenizer()
...: # fit the tokenizer on the documents
...: t.fit_on_texts(docs)
...: # encode the documents
...: encoded_docs = t.texts_to_matrix(docs, mode='tfidf')
...: print(encoded_docs)
[[0.          0.          0.          0.          ],
 [0.          0.          0.          0.69314718 0.69314718]]
```

Gambar 7.144 Contoh code tentang penggunaan texts\_to\_matrix

7. maksud dari fungsi  $d\_train\_inputs = d\_train\_inputs/npamax(np.absolute(d\_train\_inputs))$  dan  $d\_test\_inputs = d\_test\_inputs/npamax(np.absolute(d\_test\_inputs))$

pada code tersebut berfungsi sebagai pengelolaan data Variable d\_train\_inputs dan d\_test\_inputs dengan menggunakan perintah metode yang terdapat pada Library Numpy metode yang digunakan adalah AMAX yang berguna untuk mengambil nilai maksimum yang terdapat pada data arraynya atau mencari nilai maksimum dari panjang sumbu yang terproses dan ABSOLUTE yang berguna untuk untuk mengkalkulasi nilai pasti dari setiap data elemennya. untuk contoh dari penggunaan AMAX dan ABSOLUTE dapat dilihat pada Gambar

berikut ini adalah contoh Code yang digunakan untuk menjelaskan penggunaan AMAX :

```
In [24]: import numpy as np
...
... arr1 = [1, -3, 15, -466]
... print("Absolute Value of arr1 : \n",
...       np.absolute(arr1))
...
... arr2 = [23 , -56]
... print ("\nAbsolute Value of arr2 : \n",
...       np.absolute(arr2))
Absolute Value of arr1 :
[ 1  3 15 466]
Absolute Value of arr2 :
[23 56]
```

**Gambar 7.145** penggunaan Numpy AMAX

berikut ini adalah contoh Code yang digunakan untuk menjelaskan penggunaan ABSOLUTE :

```
In [29]: import numpy as geek
...
... arr = geek.arange(8)
... print("Max of arr : ", geek.max(arr))
... arr = geek.arange(10).reshape(2, 5)
... print("Max of arr : ", arr)
... print("Value of arr, axis = None : ", geek.max(arr))
... print("Max of arr, axis = 0 : ", geek.max(arr, axis = 0))
... print("Max of arr, axis = 1 : ", geek.max(arr, axis = 1))
arr : [0 1 2 3 4 5 6 7]
Max of arr : 7
arr : [[0 1 2 3 4]
[5 6 7 8 9]]
Max of arr, axis = None : 9
Max of arr, axis = 0 : [5 6 7 8 9]
Max of arr, axis = 1 : [4 9]
```

**Gambar 7.146** penggunaan Numpy ABSOLUTE

8. maksud fungsi dari `d_train_outputs = np_utils.to_categorical(d['CLASS'].iloc[train_idx])` dan `d_test_outputs = np_utils.to_categorical(d['CLASS'].iloc[test_idx])` dalam kode program

pada code tersebut menjelaskan penggunaan Library Numpy yang akan mengelola data d\_train\_outputs dan d\_test\_outputs dengan menggunakan perintah atau metode TO\_CATEGORICAL dimana data yang diambil adalah data CLASS dengan index data yang dituju adalah train\_idx dan test\_idx, dimana to\_categorical berguna untuk mengkonversikan data kelas Vektor(Integer) menjadi data kelas Binary Matrix. contoh penggunaan to categorical dapat dilihat pada Gambar

```
> labels
array([0, 2, 1, 2, 0])
> to_categorical(labels)
array([[ 1.,  0.,  0.,  0.],
[ 0.,  1.,  0.,  0.],
[ 0.,  1.,  0.,  0.],
[ 0.,  0.,  1.,  0.],
[ 1.,  0.,  0.,  0.]], dtype=float32)
> mydict = [{"a": 1, 'b': 2, 'c': 3, 'd': 4},
{'a': 100, 'b': 200, 'c': 300, 'd': 400},
{'a': 1000, 'b': 2000, 'c': 3000, 'd': 4000 }]
> df.iloc[[0, 1]]
   a    b    c    d
0  100  200  300  400
1  100  200  300  400
```

**Gambar 7.147** Contoh Code dan Hasil dari penggunaan to\_categorical dan iloc

9. Penjelasan pada fungsi code dengan Ilustrasi Neural Networknya

```
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(2))
6 model.add(Activation('softmax'))

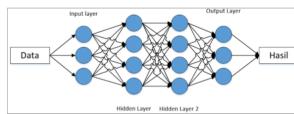
```

**Listing 7.5** Membuat model Neural Network

pada code yang terdapat pada Listing 7.5, dijelaskan bahwa dibuatkannya Variable 'model' untuk menampung model Sequential() atau metode yang berguna untuk menumpuk data Linear. berikut ini adalah penjelasan dari masing - masing fungsi yang terdapat pada Code tersebut :

- Dense (512, input\_shape = (2000,)) dan (2)  
dimana data yang diolah akan berjumlah 512 dengan shape yang diolah dalam 2000 begitu juga dengan yang 2
- Activation ('relu') dan ('softmax')  
Activation adalah untuk melakukan penggunaan fungsi matematika, relu yang berarti fungsi dari metode 'Rectified Linear Unit' dan softmax yang berguna untuk menghasilkan data Vector yang merepresentasikan nilai distribusi probabilitas dari daftar nilai output yang memiliki potensi.
- Dropout(0.5)  
melakukan set sebanyak 50 persen pada output atau nilai hasil yang dikeluaran dari pengelolaan data pada visible layer dan hidden layer.

berikut ini adalah contoh dari struktur Neural Network yang terjadi :

**Gambar 7.148** Contoh dari struktur Neural Network yang terjadi

10. Maksud dari fungsi pada Code berikut :

```

1 model.compile(loss='categorical_crossentropy', optimizer='
2   adamax',
3   metrics=['accuracy'])

```

**Listing 7.6** Compile model

pada code di Listing 7.6 menjelaskan tentang proses yang berlangsung untuk menampilkan nilai LOSS dan ACCURACY yang dihasilkan dari Compile pada program diatas.

11. Jelaskan apa itu Deep Learning

Deep Learning adalah rangkaian metode untuk bisa melatih jaringan saraf buatan multi lapisan atau mempunyai banyak lapisan. Metode ini sangat efektif

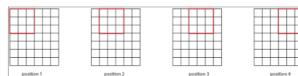
dan lebih mudah dalam mengidentifikasi pola dari data yang dimasukkan. Deep Learning juga dapat meningkatkan bagian AI, mulai dari memproses bahasa alami yang kita katakan sampai dengan mengambil gambar. Oleh karena itu Deep Learning merupakan otak yang lebih baik dan unggul untuk meningkatkan cara kerja pada sistem komputer

12. Jelaskan Apa itu Deep Neural Network dan apa perbedaanya dengan Deep Learning

Deep Neural Network adalah Neural Network dengan tingkat kompleksitas tertentu, Neural Network dengan lebih dari dua lapisan. Neural Network dalam menggunakan pemodelan matematika yang canggih untuk memproses data dengan cara yang kompleks. perbedaan antara Deep Learning dan DNN adalah DNN merupakan sebuah Algoritma yang terdapat dalam Deep Learning itu sendiri sedangkan Deep Learning adalah sebuah pengguna dari algoritma pada DNN.

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

Konvolusi terdapat pada operasi pengolahan citra yang mengalikan sebuah citra dengan sebuah mask atau kernel, Stride adalah parameter yang berfungsi untuk menentukan pergeseran pada filter data pixel yang terjadi. untuk contoh penggunaannya dapat dilihat pada Gambar



**Gambar 7.149** Penggunaan Konvolusi dengan Stridennya adalah  $3 \times 3$

## 7.6.2 Pemrograman

1. No. 1 Kode Program Blok # In 1

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

Keterangannya sebagai berikut :

- Melakukan Import Library CSV
- Melakukan Import Library PIL (Python Imaging Library) yang berguna untuk melakukan pemrosesan data yang bersifat Image. dengan fungsi yang diimport adalah pil\_image
- Melakukan Import data preprocessing dari Library Keras yang akan memproses data Image.

- hasilnya dapat dilihat pada Gambar

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

**Gambar 7.150** kode program pada blok In[1].

## 2. No. 2 Kode Program Blok # In 2

```
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                 # neuron activation functions behave best when input
12                 # values are between 0.0 and 1.0 (or -1.0 and 1.0),
13                 # so we rescale each pixel value to be in the range
14                 # 0.0 to 1.0 instead of 0-255
15                 img /= 255.0
16                 imgs.append((row[0], row[2], img))
17                 classes.append(row[2])
18             i += 1
```

Keterangannya sebagai berikut :

- Membuat Variable 'imgs' dan 'classes' yang memiliki nilai array kosong agar dapat diisi oleh data nantinya.
- Membuka file csv dari Folder HSYv2 dengan nama file hasy-data-labels.csv yang dialiaskan sebagai Variable csvfile.
- Variabel csvreader akan menggunakan fungsi reader pada library csv untuk membaca file pada variable csvfile.
- Membuat Variable i yang bernilai NOL.
- Membuat pemrosesan data yang akan membaca datan dari csvreader dengan membuat Variable 'row', dimasukan dengan fungsi IF jika i lebih besar dari NOL.
- membuat Variable 'img' yang berisi data pemrosesan Library keras dengan fungsi image.im\_to\_array yang akan mmebuat data yang dibuka menjadi format array pada Folder HASYv2 yang dimulai dari NOL.
- Hasil dari variabel img akan dibagi dengan 255.0
- .append akan membuat list array baru untuk baris 0 baris 2 pada img.
- Menyimpan setiap class nya pada baris 2

- Penambahan i sebanyak 1.
- hasilnya dapat dilihat pada Gambar

```
In [2]:
In [2]: imgs = []
...: classes = []
...: with open('mnist/mnist_train.csv') as csvfile:
...:     csvreader = csv.reader(csvfile)
...:     for row in csvreader:
...:         if len(row) == 785:
...:             img = keras.preprocessing.image.img_to_array(pil_image.open("mnist27." +
...:             row[0]))
...:             # inverse activation functions before test when input values are between
...:             # 0 and 1.0 (for 1.0 is white and 0.0 is black). If we want to keep each pixel value to be in the range 0.0 to 1.0 instead of
...:             # 0 to 255.0
...:             img /= 255.0
...:             img = np.reshape(img,[28,28])
...:             classes.append(int(row[0]))
...:             imgs.append(img)
...:     i += 1
```

**Gambar 7.151** kode program pada blok In[2].

### 3. No. 3 Kode Program Blok # In 3

```
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:]
```

Keterangannya sebagai berikut :

- Melakukan Import Library random.
- Membuat parameter dari penggunaan library random yang akan mengacak data dengan fungsi 'shuffle' pada data Variable 'imgs'.
- Membagi index data dari imgs dengan cara mengalikan 80% dengan jumlah data dari imgs.
- Untuk data train mengambil hasil dari perhitungan sebelumnya.
- Untuk data test mengambil sisa dari jumlah yang telah dijadikan data train.
- hasil dapat dilihat pada Gambar'

```
In [5]: import random
...: random.shuffle(imgs)
...: split_idx = int(0.8*len(imgs))
...: train = imgs[:split_idx]
...: test = imgs[split_idx:]
```

**Gambar 7.152** kode program pada blok In[3].

### 4. No. 4 Kode Program Blok # In 4

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

Keterangannya sebagai berikut :

- Melakukan Import dari Library Numpy dialiaskan menjadi np.
- Variabel train\_input mengubah input menjadi sebuah array yang diambil dari baris 2, dari data train.
- Variabel test\_input mengubah input menjadi sebuah array yang diambil dari baris 2, dari data test.
- Variabel train\_output mengubah input menjadi sebuah array yang diambil dari baris 1, dari data train.
- Variabel test\_output mengubah input menjadi sebuah array yang diambil dari baris 1, dari data test.
- hasil dapat dilihat pada Gambar

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

**Gambar 7.153** kode program pada blok In[4].

## 5. No. 5 Kode Program Blok # In 5

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

Keterangannya sebagai berikut :

- Melakukan Import fungsi LabelEncoder dari Library sklearn
- Melakukan Import fungsi OneHotEncoder dari Library sklearn
- hasil dapat dilihat pada Gambar

```
In [7]: from sklearn.preprocessing import LabelEncoder
.... from sklearn.preprocessing import OneHotEncoder
```

**Gambar 7.154** kode program pada blok In[5].

## 6. No. 6 Kode Program Blok # In 6

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

Keterangannya sebagai berikut :

- Membuat Variabel label\_encoder yang akan memanggil fungsi LabelEncoder tadi.

- Membuat Variabel integer\_encoded yang akan menggunakan labelencoder untuk melakukan fit pada classes agar berubah datanya menjadi integer.
- hasil dapat dilihat pada Gambar

```
In [8]: label_encoder = LabelEncoder()
...: integer_encoded = label_encoder.fit_transform(classes)
```

**Gambar 7.155** kode program pada blok In[6].

## 7. No. 7 Kode Program Blok # In 7

```
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)
```

Keterangannya sebagai berikut :

- Membuat Variabel onehot\_encoder akan memanggil fungsi OneHotEncoder dimana tidak berisikan matriks sparse.
- Pada variabel integer\_encoded akan diubah bentuknya dimana setiap nilai integer akan direpresentasikan sebagai vektor binari dengan nilai 0 kecuali index dari integer tersebut ditandai dengan 1.
- Melakukan fit untuk one hot encoder kedalam integer\_encoder.
- hasil dapat dilihat pada Gambar

```
In [11]: onehot_encoder = OneHotEncoder(sparse=False)
...: integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
C:\Users\DELL\OneDrive\Babak\site-packages\sklearn\preprocessing\_encoders.py:160: UserWarning: This class is deprecated, it was kept only for backward compatibility.
Currently, the categories are determined based on the range [0, max(values)], which is not always correct. If you want the future behaviour and silence this warning, you can specify 'categories='auto' or 'categories='all' instead.
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, category=FutureWarning)
Out[11]:
OneHotEncoder(categories='auto', features='one-hot', categories=None,
              dtype=<class 'numpy.float64'>, handle_unknown='error',
              n_values=None, sparse=False)
```

**Gambar 7.156** kode program pada blok In[7].

## 8. No. 8 Kode Program Blok # In 8

```
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
7 num_classes = len(label_encoder.classes_)
8 print("Number of classes: %d" % num_classes)
```

Keterangannya sebagai berikut :

- Membuat Variabel train\_output\_int akan mengubah data dari train\_output menjadi LabelEncoder
- Membuat Variable train\_output dimana setelah diubah labelnya menjadi integer dilakukan onehot\_encoding diambil dari train\_output\_int dan menggunakan .reshape untuk memberikan bentuk baru ke array tanpa mengubah datanya dengan keterangan jika index dari integer tersebut ditandai dengan 1 dan sisanya yang bukan nol.
- Membuat Variabel test\_output\_int akan mengubah data dari test\_output menjadi LabelEncoder
- Membuat Variable test\_output dimana setelah diubah labelnya menjadi integer dilakukan one hot encoding diambil dari test\_output\_int dan menggunakan .reshape untuk memberikan bentuk baru ke array tanpa mengubah datanya dengan keterangan jika index dari integer tersebut ditandai dengan 1 dan sisanya yang bukan nol.
- Membuat Variabel num\_classes akan menampilkan jumlah data dari classes yang telah dilakukan label encoder
- Melakukan print dengan Format tulisannya adalah "Number of classes : %d" dimana mengembalikan nilai integer dari num\_classes.
- hasil dapat dilihat pada Gambar

```
In [8]: train_output_int = label_encoder.transform(train_output)
...:
train_output = label_encoder.transform(train_output_int.reshape(len(train_output_int), 1))
...:
test_output_int = label_encoder.transform(test_output)
...:
num_classes = len(label_encoder.classes_)
...:
print("Number of classes: ", num_classes)
Number of classes: 400
```

**Gambar 7.157** kode program pada blok In[8].

## 9. No. 9 Kode Program Blok # In 9

```
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
```

Keterangannya sebagai berikut :

- Melakukan Import Sequential dari model pada Library Keras.
- Melakukan Import Dense, Dropout, Flatten dari modul Layers pada Library Keras.
- Melakukan Import Conv2D, MaxPooling2D dari modul Layers pada Library Keras.
- hasil dapat dilihat pada Gambar

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

**Gambar 7.158** kode program pada blok In[9].

## 10. No. 10 Kode Program Blok # In 10

```

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
13 model.compile(loss='categorical_crossentropy', optimizer='adam',
14                 metrics=['accuracy'])
15
16 print(model.summary())

```

Keterangannya sebagai berikut :

- Melakukan pemodelan Sequential.
- Menambahkan Konvolusi 2D dengan 32 filter konvolusi masing-masing berukuran 3x3 dengan algoritam activation relu dengan data dari train.input mulai dari baris nol.
- Menambahkan Max Pooling dengan matriks 2x2.
- Dilakukan lagi penambahan Konvolusi 2D dengan 32 filter konvolusi masing-masing berukuran 3x3 dengan algoritam activation relu.
- Menambahkan lagi Max Pooling dengan matriks 2x2.
- Mendefinisikan inputan dengan 1024 neuron dan menggunakan algoritma tanh untuk activationnya.
- Dropout terdiri dari pengaturan secara acak tingkat pecahan unit input ke 0 pada setiap pembaruan selama waktu pelatihan, yang membantu mencegah overfitting sebesar 50% .
- Untuk output layer menggunakan data dari variabel num\_classes dengan fungsi activationnya softmax.
- Mengonfigurasi proses pembelajaran, yang dilakukan melalui metode compile, sebelum melatih suatu model.
- Menampilkan atau mencetak representasi ringkasan model yang telah dibuat.

|                                                                  |              |              |
|------------------------------------------------------------------|--------------|--------------|
| Sequential                                                       | (Sequential) | (Sequential) |
| model = Sequential()                                             |              |              |
| model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',      |              |              |
| input_shape=np.shape(train_input[0]))                            |              |              |
| )                                                                |              |              |
| model.add(MaxPooling2D(pool_size=(2, 2)))                        |              |              |
| )                                                                |              |              |
| model.add(Conv2D(32, (3, 3), activation='relu'))                 |              |              |
| )                                                                |              |              |
| model.add(MaxPooling2D(pool_size=(2, 2)))                        |              |              |
| )                                                                |              |              |
| model.add(Flatten())                                             |              |              |
| )                                                                |              |              |
| model.add(Dense(1024, 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 7.159 kode program pada blok In[10].

## 11. No. 11 Kode Program Blok # In 11

```

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

```

Keterangannya sebagai berikut :

- Impor Modul Callbacks dari Librari Keras.
- Variabel callback mendefinisikan Callback ini untuk menulis log untuk TensorBoard, yang memungkinkan Anda untuk memvisualisasikan grafik dinamis dari pelatihan dan metrik pengujian Anda, serta histogram aktivasi untuk berbagai lapisan dalam model Anda.

```

In [15]: import keras.callbacks
...: tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/
mnist-style')

```

**Gambar 7.160** kode program pada blok In[11].

## 12. No. 12 Kode Program Blok # In 12

```

1 # In[12]: 5menit kali 10 epoch = 50 menit
2 model.fit(train_input, train_output,
3            batch_size=32,
4            epochs=10,
5            verbose=2,
6            validation_split=0.2,
7            callbacks=[tensorboard])
8
9 score = model.evaluate(test_input, test_output, verbose=2)
10 print('Test loss:', score[0])
11 print('Test accuracy:', score[1])

```

Keterangannya sebagai berikut :

- Melakukan fit model dengan 32 ukuran subset dari sampel pelatihan Anda
- Epoch sebanyak 10 kali
- Vebrose=2 maksudnya menampilkan nomor dari epoch yang sedang berjalan atau yang sudah dijalankan.
- Validasi plit sebanyak 20% sebagai fraksi data pelatihan untuk digunakan sebagai data validasi.
- Menggunakan TensorBoard sebagai callback untuk diterapkan selama pelatihan dan validasi.
- Variabel score mengembalikan nilai evaluate untuk menampilkan data lost dan data accuracy dari test
- Menampilkan data loss dengan menghitung jumlah kemunculan nol .
- Menampilkan data accuracy dengan menghitung jumlah kemunculan 1.

```

Instructions for updating:
use tf.estimator instead.
Train on 187668 samples, validate on 20918 samples
Epoch 00000: loss: 1.5519 - acc: 0.6209 - val_loss: 0.9039 - val_acc: 0.7246
  0025 - loss: 0.9666 - acc: 0.7113 - val_loss: 0.8704 - val_acc: 0.7445
  0050 - loss: 0.8594 - acc: 0.7051 - val_loss: 0.8499 - val_acc: 0.7373
  0075 - loss: 0.7770 - acc: 0.7118 - val_loss: 0.8136 - val_acc: 0.7392
  0100 - loss: 0.7730 - acc: 0.7132 - val_loss: 0.8104 - val_acc: 0.7397
  0125 - loss: 0.6612 - acc: 0.7944 - val_loss: 0.8118 - val_acc: 0.7999
  0150 - loss: 0.6543 - acc: 0.7982 - val_loss: 0.8185 - val_acc: 0.7726
  0175 - loss: 0.6206 - acc: 0.8057 - val_loss: 0.8377 - val_acc: 0.7558
  0200 - loss: 0.5976 - acc: 0.8108 - val_loss: 0.8562 - val_acc: 0.7719
  0225 - loss: 0.5778 - acc: 0.8146 - val_loss: 0.8736 - val_acc: 0.7567
Test loss: 0.7577792368258098
Test accuracy: 0.7577792368258098

```

Gambar 7.161 kode program pada blok In[12].

### 13. No. 13 Kode Program Blok # In 13

```

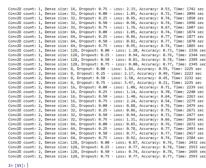
1 # In [13]: try various model configurations and parameters to find
2     the best
3 import time
4
5 results = []
6 for conv2d_count in [1, 2]:
7     for dense_size in [128, 256, 512, 1024, 2048]:
8         for dropout in [0.0, 0.25, 0.50, 0.75]:
9             model = Sequential()
10            for i in range(conv2d_count):
11                if i == 0:
12                    model.add(Conv2D(32, kernel_size=(3, 3),
13 activation='relu', input_shape=np.shape(train_input[0])))
14                else:
15                    model.add(Conv2D(32, kernel_size=(3, 3),
16 activation='relu'))
17                    model.add(MaxPooling2D(pool_size=(2, 2)))
18                    model.add(Flatten())
19                    model.add(Dense(dense_size, activation='tanh'))
20                    if dropout > 0.0:
21                        model.add(Dropout(dropout))
22                    model.add(Dense(num_classes, activation='softmax'))
23
24                    model.compile(loss='categorical_crossentropy',
25 optimizer='adam', metrics=['accuracy'])
26
27                    log_dir = './logs/conv2d_%d-dense_%d-dropout_%.2f' %
28 (conv2d_count, dense_size, dropout)
29                    tensorboard = keras.callbacks.TensorBoard(log_dir=
log_dir)
30
31                    start = time.time()
32                    model.fit(train_input, train_output, batch_size=32,
33 epochs=10,
34                         verbose=0, validation_split=0.2, callbacks
35 =[tensorboard])
36                    score = model.evaluate(test_input, test_output,
37 verbose=2)
38                    end = time.time()
39                    elapsed = end - start
40                    print("Conv2D count: %d, Dense size: %d, Dropout: %.2
f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count
41 , dense_size, dropout, score[0], score[1], elapsed))

```

33 results.append((conv2d\_count, dense\_size, dropout,  
score[0], score[1], elapsed))

Keterangannya sebagai berikut :

- impor modul time dari python anaconda
- Variabel result berisikan array kosong.
- Menggunakan convolution 2D yang dimana akan memiliki 1 atau 2 layer.
- Mendefinisikan dense\_size dengan ukuran 128, 256, 512, 1024, 2048
- Mendefinisikan drop\_out dengan 0, 25%, 50%, dan 75%
- Melakukan pemodelan Sequential
- Jika ini adalah layer pertama, kita perlu memasukkan bentuk input.
- Kalau tidak kita hanya akan menambahkan layer.
- Kemudian, setelah menambahkan layer konvolusi, kita akan melakukan hal yang sama dengan max pooling.
- Lalu, kita akan meratakan atau flatten dan menambahkan dense size ukuran apa pun yang berasal dari dense\_size. Dimana akan selalu menggunakan algoritma tanh
- Jika dropout digunakan, kita akan menambahkan layer dropout. Menyebut dropout ini berarti, katakanlah 50%, bahwa setiap kali ia memperbarui bobot setelah setiap batch, ada peluang 50% untuk setiap bobot yang tidak akan diperbarui
- menempatkan ini di antara dua lapisan padat untuk dihindarkan dari melindunginya dari overfitting.
- Lapisan terakhir akan selalu menjadi jumlah kelas karena itu harus, dan menggunakan softmax. Itu dikompilasi dengan cara yang sama.
- Atur direktori log yang berbeda untuk TensorBoard sehingga dapat memberikan konfigurasi yang berbeda.
- Variabel start akan memanggil modul time atau waktu
- Melakukan fit atau compile
- Melakukan scoring dengan .evaluate yang akan menampilkan data loss dan accuracy dari model
- end merupakan variabel untuk melihat waktu akhir pada saat pemodelan berhasil dilakukan.
- Menampilkan hasil dari run skrip diatas



A screenshot of a Jupyter Notebook cell showing the output of a command. The output consists of several lines of text, each starting with 'Conv2d (conv1, 1, ReLU, class\_10, Dropout, 0.25, loss= 0.35, accuracy= 0.31, time= 1280 ms)' followed by a large number of zeros. This pattern repeats multiple times, indicating a loop or a large dataset being processed.

**Gambar 7.162** kode program pada blok In[13].

#### 14. No. 14 Kode Program Blok # In 14

```

1 # In[14]: rebuild/retrain a model with the best parameters (from
2     the search) and use all data
3 model = Sequential()
4 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
5     input_shape=np.shape(train_input[0])))
6 model.add(MaxPooling2D(pool_size=(2, 2)))
7 model.add(Conv2D(32, (3, 3), activation='relu'))
8 model.add(MaxPooling2D(pool_size=(2, 2)))
9 model.add(Flatten())
10 model.add(Dense(128, activation='tanh'))
11 model.add(Dropout(0.5))
12 model.compile(loss='categorical_crossentropy', optimizer='adam',
13     metrics=['accuracy'])
14 print(model.summary())

```

Keterangannya sebagai berikut :

- Melakukan pemodelan Sequential
- Untuk layer pertama, Menambahkan Convolutio 2D dengan dmensi 32, dan ukuran matriks 3x3 dengan function aktivasi yang digunakan yaitu relu dan menampilkan input\_shape
- Dilakukan Max Pooling 2D dengan ukuran matriks 2x2
- Untuk layer kedua, melakukan Convolusi lagi dengan kriteria yang sama tanpa menambahkan input, ini dilakukan untuk mendapatkan data yang terbaik
- Flatten digunakan ntuk meratakan inputan
- Menambahkan dense input sebanyak 128 neuron dengan menggunakan function aktivasi tanh.
- Dropout sebanyak 50% untuk menghindari overfitting
- Menambahkan dense pada model untuk output dimana layer ini akan menjadi jumlah dari class yang ada.
- Mengcompile model yang didefinisikan diatas
- Menampilkan ringkasan dari pemodelan yang dilakukan

```
In [13]: model = Sequential([
    ...,
    .add(Conv2D(16, kernel_size=(3, 3), activation='relu',
    .add(MaxPooling2D(pool_size=(2, 2)),
    .add(Flatten()),
    .add(Dense(128, activation='tanh'))),
    .add(Dense(10, activation='softmax')),
    .compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
])
print(model.summary())

```

| Layer (type)                   | Output Shape       | Param # |
|--------------------------------|--------------------|---------|
| conv2d_0 (Conv2D)              | (None, 15, 15, 32) | 896     |
| max_pooling2d_0 (MaxPooling2D) | (None, 7, 7, 32)   | 0       |
| conv2d_1 (Conv2D)              | (None, 3, 3, 32)   | 2344    |
| max_pooling2d_1 (MaxPooling2D) | (None, 1, 1, 32)   | 0       |
| flatten_0 (Flatten)            | (None, 32)         | 0       |
| dense_0 (Dense)                | (None, 128)        | 417584  |
| dropout_0 (Dropout)            | (None, 128)        | 0       |
| dense_1 (Dense)                | (None, 10)         | 4700    |

Total params: 285,229  
Trainable params: 285,229  
Non-trainable params: 0

None

**Gambar 7.163** kode program pada blok In[14].

## 15. No. 15 Kode Program Blok # In 15

```
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)
```

Keterangannya sebagai berikut :

- Melakukan fit dengan join data train dan test agar dapat dilakukan pelatihan untuk jaringan pada semua data yang dimiliki.

```
In [16]: model.fit(np.concatenate((train_input, test_input)),
...,
          batch_size=32, epochs=10, verbose=2)
Epoch 1/10
- 383s - loss: 1.7827 - acc: 0.5858
Epoch 2/10
- 387s - loss: 1.0765 - acc: 0.7067
Epoch 3/10
- 364s - loss: 0.9664 - acc: 0.7301
Epoch 4/10
- 367s - loss: 0.9853 - acc: 0.7426
Epoch 5/10
- 367s - loss: 0.8677 - acc: 0.7508
Epoch 6/10
- 365s - loss: 0.8356 - acc: 0.7587
Epoch 7/10
- 355s - loss: 0.8168 - acc: 0.7617
Epoch 8/10
- 367s - loss: 0.7972 - acc: 0.7659
Epoch 9/10
- 379s - loss: 0.7819 - acc: 0.7692
Epoch 10/10
- 473s - loss: 0.7680 - acc: 0.7725
Out[16]: <keras.callbacks.History at 0x22e989c543b>
```

**Gambar 7.164** kode program pada blok In[15].

## 16. No. 16 Kode Program Blok # In 16

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

Keterangannya sebagai berikut :

- Menyimpan atau save model yang telah di latih dengan nama mathsymbols.model

**Gambar 7.165** kode program pada blok In[16].

## 17. No. 17 Kode Program Blok # In 17

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

Keterangannya sebagai berikut :

- Simpan label enkoder (untuk membalikkan one-hot encoder) dengan nama classes.npy

```
In [18]: np.save('classes.npy', label_encoder.classes_)
```

**Gambar 7.166** kode program pada blok In[17].

## 18. No. 18 Kode Program Blok # In 18

```
1 # In[18]: load the pre-trained model and predict the math symbol
   for an arbitrary image;
2 # the code below could be placed in a separate file
3 import keras.models
4 model2 = keras.models.load_model("mathsymbols.model")
5 print(model2.summary())
```

Keterangannya sebagai berikut :

- Impor models dari librari Keras
- Variabel model2 akan memanggil model yang telah disave tadi
- Menampilkan ringkasan dari hasil pemodelan

```
In [19]: import keras.models
...: model2 = keras.models.load_model("mathsymbols.model")
...: print(model2.summary())
Layer (type)                 Output Shape              Param #
conv2d_63 (Conv2D)            (None, 38, 38, 32)      896
max_pooling2d_63 (MaxPooling2D) (None, 15, 15, 32)      0
conv2d_64 (Conv2D)            (None, 13, 13, 32)      9248
max_pooling2d_64 (MaxPooling2D) (None, 6, 6, 32)      0
flatten_42 (Flatten)          (None, 1152)             0
dense_83 (Dense)              (None, 128)              147584
dropout_32 (Dropout)          (None, 128)              0
dense_84 (Dense)              (None, 369)              47691
=====
Total params: 285,329
Trainable params: 285,329
Non-trainable params: 0
None
```

**Gambar 7.167** kode program pada blok In[18].

## 19. No. 19 Kode Program Blok # In 19

```
1 # In[19]: restore the class name to integer encoder
2 label_encoder2 = LabelEncoder()
3 label_encoder2.classes_ = np.load('classes.npy')
4
5 def predict(img_path):
6     newimg = keras.preprocessing.image.img_to_array(pil_image.
open(img_path))
```

```

7 newimg /= 255.0
8
9 # do the prediction
10 prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
11
12 # figure out which output neuron had the highest score, and
13 # reverse the one-hot encoding
14 inverted = label_encoder2.inverse_transform([np.argmax(
    prediction)]) # argmax finds highest-scoring output
    print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
        max(prediction)))

```

Keterangannya sebagai berikut :

- Memanggil fungsi LabelEncoder
- Variabel label\_encoder akan memanggil class yang disave sebelumnya.
- Function Predict akan mengubah gambar kedalam bentuk array
- Variabel prediction akan melakukan prediksi untuk model2 dengan reshape variabel newimg dengan bentukarray 4D.
- Variabel inverted akan mencari nilai tertinggi output dari hasil prediksi tadi
- Menampilkan hasil dari variabel prediction dan inverted

```

In [20]: label_encoder2 = LabelEncoder()
... label_encoder2.classes_ = np.load('classes.npy')
... def predict(img_path):
...     imgimg = Image.open(img_path)
...     imgimg = np.array(imgimg)
...     imgimg = imgimg / 255.0
...     # do the prediction
...     prediction = model2.predict(imgimg.reshape(1, 32, 32, 3))
...     # figure out which output neuron had the highest score, and reverse
the one-hot encoding
...     inverted = label_encoder2.inverse_transform([np.argmax(prediction)])
...     print("Prediction: %s, confidence: %.2f" % (inverted[0],
        np.max(prediction)))

```

**Gambar 7.168** kode program pada blok In[19].

## 20. No. 20 Kode Program Blok # In 20

```

1 # In [20]: grab an image (we'll just use a random training image
      for demonstration purposes)
2 predict("HASYv2/hasy-data/v2-00010.png")
3
4 predict("HASYv2/hasy-data/v2-00500.png")
5
6 predict("HASYv2/hasy-data/v2-00700.png")

```

Keterangannya sebagai berikut :

- Melakukan prediksi dari pelatihan dari gambar v2-00010.png
- Melakukan prediksi dari pelatihan dari gambar v2-00500.png
- Melakukan prediksi dari pelatihan dari gambar v2-00700.png

```
In [23]: predict("HASYv2/hasy-data/v2-00010.png")
...
...: predict("HASYv2/hasy-data/v2-00500.png")
...
...: predict("HASYv2/hasy-data/v2-00700.png")
Prediction: A, confidence: 0.83
Prediction: \pi, confidence: 0.48
Prediction: \alpha, confidence: 0.92

In [24]: predict("HASYv2/hasy-data/v2-00010.png")
...
...: predict("HASYv2/hasy-data/v2-00500.png")
...
...: predict("HASYv2/hasy-data/v2-01000.png")
Prediction: A, confidence: 0.83
Prediction: \pi, confidence: 0.48
Prediction: \sum, confidence: 0.56

In [25]: predict("HASYv2/hasy-data/v2-00010.png")
...
...: predict("HASYv2/hasy-data/v2-00900.png")
...
...: predict("HASYv2/hasy-data/v2-01000.png")
Prediction: A, confidence: 0.83
Prediction: \beta, confidence: 0.84
Prediction: \sum, confidence: 0.56
```

**Gambar 7.169** kode program pada blok In[20].

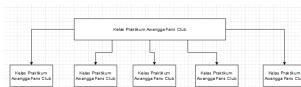
## 7.7 1174096 - Nico Ekklesia Sembiring

### 7.7.1 Soal Teori

1. Jelaskan kenapa file teks harus di lakukan tokenizer

Jelaskan kenapa file teks harus di lakukan tokenizer. dilengkapi dengan ilustrasi atau gambar.

Tokenizer diperlukan untuk perhitungan bobot pada setiap teks karena Tokenizer akan membagi kalimat menjadi beberapa teks sehingga nantinya kalimat yang dimasukkan otomatis langsung diubah menjadi kata - kata dan akan dinilai sehingga akan memunculkan nilai vektor untuk digunakan saat prediksi teks yang muncul pada satu kalimat tersebut. Biasanya untuk memisahkan sebuah kata, tokenizer akan menggunakan spasi sebagai pemisah antar kata.



**Gambar 7.170** Ilustrasi Tokenizer

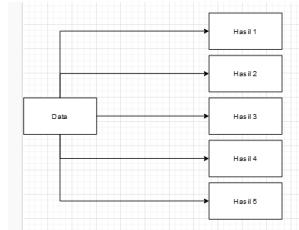
2. Jelaskan konsep dasar K Fold Cross Validation pada dataset komentar Youtube pada kode listing 7.1 [7.7](#)

```
1 kfold = StratifiedKFold(n_splits=5)
2 splits = kfold.split(d, d['CLASS'])
3
```

**Listing 7.7** K Fold Cross Validation

Pada koding diatas terdapat variabel kfold yang didalamnya berisi parameter split yang diisikan nilai 5. hal tersebut dimaksudkan untuk membuat pengolahan data akan diulang setiap datanya sebanyak lima kali dengan atribut class

sebagai acuan pengolahan datanya. Lalu kemudian akan di hasilkan akurasi dari pengulangan data tersebut sebesar sekian persen tergantung datanya



**Gambar 7.171** ilustrasi K-Fold Cross Validation

3. Jelaskan apa maksudnya kode program for train, test in splits.

For train digunakan untuk melakukan training atau pelatihan pada data yang sudah dideklarasikan sebelumnya. Sedangkan test in split digunakan untuk membatasi jumlah data yang akan diinputkan atau data yang akan digunakan.

```

In [1]: import numpy as np
...::: from sklearn.model_selection import train_test_split
...::: from sklearn.datasets import load_iris
...::: y = np.arange(15)
...::: print(y)
[1: 15]
[1: 15]
In [2]: train,test = train_test_split(x_train,y_train,test_size=0.33,random_state=42)
...::: print(train)
[1: 12]
[1: 12]
  
```

**Gambar 7.172** Ilustrasi for train dan test in split

4. Jelaskan apa maksudnya kode program train content = d['CONTENT'].iloc[train idx] dan test content = d['CONTENT'].iloc[test idx].

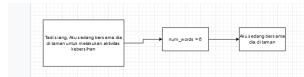
Maksud dari kode program tersebut adalah membaca isian kolom pada field yang bernama CONTENT sebagai data training dan data testing untuk program

| No | Nama       | Content                                                   |
|----|------------|-----------------------------------------------------------|
| 1  | Motor      | Kendaraan darat yang basanya memiliki roda 4              |
| 2  | Motor      | Kendaraan darat yang basanya memiliki roda 2              |
| 3  | Traktor    | Kendaraan darat yang dipakai untuk membajak sawah         |
| 4  | Helikopter | Kendaraan udara yang memiliki baling-baling untuk terbang |

**Gambar 7.173** ilustrasi penggunaan kolom content

5. Jelaskan apa maksud dari fungsi tokenizer = Tokenizer(num words=2000) dan tokenizer.fit on texts(train content).

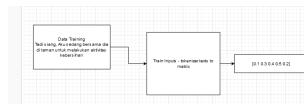
- tokenizer = Tokenizer(num\_words=2000) digunakan untuk membaca kalimat yang telah dibuat menjadi token sebanyak 2000 kata
- fit\_on\_texts digunakan untuk membuat membaca data token teks yang telah dimasukan kedalam fungsi yaitu fungsi train\_konten



**Gambar 7.174** Ilustrasi Fit Tokenizer dan num\_word=2000

6. Jelaskan apa maksud dari fungsi d train inputs = tokenizer.texts to matrix(train content, mode='tfidf') dan d test inputs = tokenizer.texts to matrix(test content, mode='tfidf'),

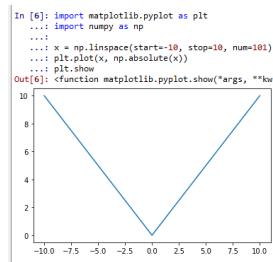
Untuk digunakan sebagai pengubah urutan teks yang tadi telah dilakukan tokenizer menjadi matriks yang berurutan seperti tf idf



**Gambar 7.175** Ilustrasi d train inputs = tokenizer.texts to matrix

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 inputs))

Fungsi tersebut digunakan untuk membagi matriks tfidf dengan penentuan maksimum array sepanjang sumbu sehingga akan menimbulkan garis ke bawah dan ke atas yang membentuk gambar v. Lalu hasil tersebut akan dimasukkan ke variabel d train input dan d test input dengan metode absolute. Yang berarti tanpa bilangan negatif.



**Gambar 7.176** fungsi d train inputs

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

Maksud dari fungsi tersebut yaitu untuk merubah nilai vektor yang ada pada atribut class menjadi bentuk matrix dengan pengurutan berdasarkan data index training dan testing.

```
In [10]: labels = [0,2,1,2,0,1]
...: keras.utils.to_categorical(labels)Out[11]:
array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.]], dtype=float32)
```

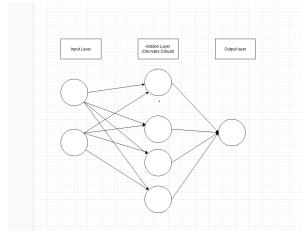
**Gambar 7.177** fungsi train outputs = np utils.to\_categorical

9. Jelaskan apa maksud dari fungsi di listing 7.2 **7.8**.

```
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(2))
6 model.add(Activation('softmax'))
```

**Listing 7.8** Membuat model Neural Network

model = sequential berarti variabel model berisi method sequential yang berguna untuk searching data dengan menerima parameter atau argumen kunci dengan langkah tertentu untuk mencari data yang telah diolah. Kemudian model akan ditambahkan method add dengan dense yang berarti data - data yang diinputkan akan terhubung, dengan data 612 dan 2000 data kata atau word kemudian model tersebut di masukan fungsi activation dengan rumus atau metode relu. setelah itu data akan di dropout 0.5atau dipangkas sebanyak 50 persen dikarenakan pada pohon bobot terlalu akurat terhadap data.



**Gambar 7.178** ilustrasi neural network

10. Jelaskan apa maksud dari fungsi di listing 7.3 **7.9** dengan parameter tersebut.

```
1 model.compile(loss='categorical_crossentropy', optimizer=
2   'adamax',
3   metrics=['accuracy'])
```

**Listing 7.9** Compile model

model tersebut kemudian di compile atau di kembalikan kembali fungsi nilainya yang mana akan mengembalikan fungsi nilai loss nya berapa yang diambil dari

fungsi adamax yang berberguna untuk mengetahui nilai lossnya kemudian metrics = accuracy merupakan akurasi dari nilai matrixnya. kemudian terdiri atas beberapa layer atau hidden layer. perbedaan antara deep learning dan DNN atau Deep Neural Network yaitu deep learning merupakan pemakai algoritma dari DNN dan DNN merupakan algoritma yang ada pada deep learning.

### 11. Jelaskan apa itu Deep Learning

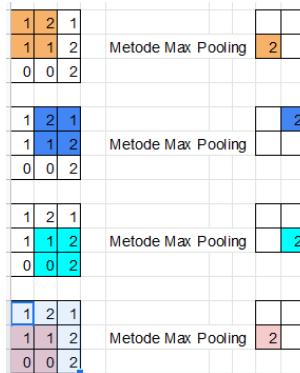
Deep learning merupakan salah satu algoritma yang seperti Neural Network yang menggunakan meta data sebagai inputan dan mengolahnya menggunakan layer layer yang tersembunyi.

### 12. Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning

Deep Neural Network merupakan algoritma jaringan syaraf yang melakukan pembobotan terhadap data yang sudah ada sebagai acuan untuk data inputan selanjutnya.

### 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.

Sebelum membuat ilustrasi perlu diketahui apa itu stride, stride adalah acuan atau parameter yang menentukan pergeseran pada filter fixcel. sebagai contoh nilai stride 1 yang berarti filter akan bergeser sebanyak satu fixcel secara vertikal dan horizontal. selanjutnya apa itu max pooling contoh pada suatu gambar ditentukan Max Pooling dari 3 x 3 dengan stride 1 yang berarti setiap pergeseran 1 pixcel akan diambil nilai terbesar dari pixcel 3 x 3 tersebut.



Gambar 7.179 ilustrasi perhitungan stride 1 max pooling

### 7.7.2 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 # menimpor libtari CSV untuk mengolah data ber ekstensi csv
3 import csv
4 #kemudian Melakukan import library Image yang berguna untuk dari
5 # PIL atau Python Imaging Library yang berguna untuk mengolah
6 # data berupa gambar
7 from PIL import Image as pil_image
8 # kemudian Melakukan import library keras yang menggunakan method
9 # preprocessing yang digunakan untuk membuat neural network
10 import keras.preprocessing.image

```

```

In [14]: import csv
...: #Mendefinisikan Import Library Image yang berguna untuk dari
...: # PIL atau Python Imaging Library yang berguna untuk mengolah data berupa
...: # gambar
...: ...: # Kemudian Melakukan Import Library keras yang menggunakan
...: # method preprocessing yang digunakan untuk membuat neural network
...: ...: # Import keras.preprocessing.image

```

**Gambar 7.180** Hasil Soal 1.

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 #Menginisiasi variabel imgs dan classes dengan variabel array
3 #kosong
4 imgs = []
5 classes = []
6 #membuka file hasy-data-labels.csv yang berada di foleder HASYv2
7 #yang di inisialisasi menjadi csvfile
8 with open('HASYv2/hasy-data-labels.csv') as csvfile:
9     #Menginisiasi variabel csvreader yang berisi method csv.
10    reader = csv.reader(csvfile)
11    # Menginisiasi variabel i dengan isi 0
12    i = 0
13    # membuat looping pada variabel csvreader
14    for row in csvreader:
15        # dengan ketentuan jika i lebihkecil daripada 0
16        if i > 0:
17            # dibuat variabel img dengan isi keras untuk aktivasi
18            # neural network fungsi yang membaca data yang berada dalam
19            # folder HASYv2 dengan input nilai -1.0 dan 1.0
20            img = keras.preprocessing.image.img_to_array(
21                pil_image.open("HASYv2/" + row[0]))
22            #Pembagian data yang ada pada fungsi img sebanyak
23            255.0
24            img /= 255.0
25            # Penambahan nilai baru pada imgs pada row ke 1 2 dan
26            # dilanjutkan dengan variabel img
27            imgs.append((row[0], row[2], img))
28            # Penambahan nilai pada row ke 2 pada variabel
29            classes
30            classes.append(row[2])
31            # penambahan nilai satu pada variabel i

```

```

24 i += 1
25
26 # In [3]: shuffle the data, split into 80% train, 20% test
27 # Melakukan import library random
28 import random

```

```

In [34]: imgs = []
... Classes = []
...     ... membaca file hasy-data-labels.csv yang berada di folder HASY2
... yang di
...     ... dengan open('hasy-data-labels.csv') as csvfile:
...         ... menginisiasi variabel csvreader yang berisi method
...         ... csvreader = csv.reader(csvfile)
...         ... a menginisiasi variabel i dengan isi 0
...         ... i = 0
...         ... # membuat looping pada variabel csvreader
...         ... for row in csvreader:
...             ...     ... membaca file i lebhicccl dari pada o
...             ...     if i > 0:
...                 ... membuat variabel img dengan isi keras untuk
...                 ... aktivitas neural network fungsi yang membuca data yang berada dalam folder
...                 ... HASY2 dengan input nilai -1.0 dan 1.0
...                 ... img =
...                 ... keras.preprocessing.image.img_to_array(pil_image.open("HASY2/" + 
...                 ... row[0]))
...                 ... # membagi data yang ada pada fungsi img sebanyak
253.0
...                 ... img /= 255.0
...                 ... # Penambahan nilai baru pada imgs pada row ke 2
...                 ... dan dilanjutkan dengan penambahan nilai pada row ke 3
...                 ... imgs.append((row[0], row[2], img))
...                 ... # Penambahan nilai pada Row ke 2 pada variabel
... classes
...         ... classes.append(row[1])
...         ... # penambahan nilai satu pada variabel i
...         ...
... i += 1

```

Gambar 7.181 Hasil Soal 2.

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 # melakukan random pada vungsi imgs
2 random.shuffle(imgs)
3 # Menginisiasi variabel split_idx dengan nilai integer 80 persen
# dikali dari pengembalian jumlah dari variabel imgs
4 split_idx = int(0.8*len(imgs))
5 # Menginisiasi variabel train dengan isi lebih besar split_idx
6 train = imgs[:split_idx]
7 # Menginisiasi variabel test dengan isi lebih kecil split_idx
8 test = imgs[split_idx:]
9
10 # In [4]:
11 # Melakukan import library numpy dengan inisial np
12 import numpy as np

```

```

In [2]: import random
... # melakukan random pada vungsi imgs
... i: random.shuffle(imgs)
... # Menginisiasi variabel split_idx dengan nilai integer 80 persen
# dikali dari pengembalian jumlah dari variabel imgs
... split_idx = int(0.8*len(imgs))
... # Menginisiasi variabel train dengan isi lebih besar split_idx
... train = imgs[:split_idx]
... # Menginisiasi variabel test dengan isi lebih kecil split_idx
... i: test = imgs[split_idx:]

```

Gambar 7.182 Hasil Soal 3.

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 # Menginisiasi variabel train_input dengan np method asarray yang
   mana membuat array dengan isi row 2 dari data train
2 train_input = np.asarray(list(map(lambda row: row[2], train)))
3 # membuat test_input input dengan np method asarray yang mana
   membuat array dengan isi row 2 dari data test
4 test_input = np.asarray(list(map(lambda row: row[2], test)))
5 # Menginisiasi variabel train_output dengan np method asarray yang
   mana membuat array dengan isi row 1 dari data train
6 train_output = np.asarray(list(map(lambda row: row[1], train)))
7 # Menginisiasi variabel test_output dengan np method asarray yang
   mana membuat array dengan isi row 1 dari data test
8 test_output = np.asarray(list(map(lambda row: row[1], test)))
9
10 # In [5]: import encoder and one hot
11 # Melakukan import library LabelEncode dari sklearn
12 from sklearn.preprocessing import LabelEncoder

```

```

In [8]: import numpy as np
...: # Membuat variabel train_input dengan np method asarray yang
...: # mana membuat array dengan isi row 2 dari data train
...: train_input = np.asarray(list(map(lambda row: row[2], train)))
...: # Membuat variabel variabel test_input dengan np method asarray yang
...: # mana membuat array dengan isi row 2 dari data test
...: test_input = np.asarray(list(map(lambda row: row[2], test)))
...: # Membuat variabel variabel train_output dengan np method asarray yang
...: # mana membuat array dengan isi row 1 dari data train
...: train_output = np.asarray(list(map(lambda row: row[1], train)))
...: # Membuat variabel variabel test_output dengan np method asarray yang
...: # mana membuat array dengan isi row 1 dari data test
...: test_output = np.asarray(list(map(lambda row: row[1], test)))

```

**Gambar 7.183** Hasil Soal 4.

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 # Melakukan import library OneHotEncoder dari sklearn
2 from sklearn.preprocessing import OneHotEncoder
3
4 # In [6]: convert class names into one-hot encoding
5
6 # Menginisiasi variabel label_encoder dengan isi LabelEncoder

```

```

In [9]: from sklearn.preprocessing import LabelEncoder
...: # Melakukan import library OneHotEncoder dari sklearn
...: from sklearn.preprocessing import OneHotEncoder

```

**Gambar 7.184** Hasil Soal 5.

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 label_encoder = LabelEncoder()
2 # Menginisiasi variabel integer_encoded yang berfungsi untuk
   Menconvert variabel classes kedalam bentuk integer
3 integer_encoded = label_encoder.fit_transform(classes)
4

```

```
5 # In[7]: then convert integers into one-hot encoding
6 # Menginisiasi variabel onehot_encoder dengan isi OneHotEncoder
7 onehot_encoder = OneHotEncoder(sparse=False)
```

```
In [10]: label_encoder = LabelEncoder()
...: # Menginisiasi variabel integer_encoded yang berfungsi untuk
# mengubah kategori menjadi angka
...: integer_encoded = label_encoder.fit_transform(classes)
```

**Gambar 7.185** Hasil Soal 6.

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
2 # In[8]: convert train and test output to one-hot
3 # Menconvert data train output menggunakan variabel
# label_encoder kedalam variabel train_output_int
4 train_output_int = label_encoder.transform(train_output)
```

```
In [11]: onehot_encoder = OneHotEncoder(sparse=False)
...: # Menginisiasi variabel onehot_encoder yang akan di convert pada fungsi reshape
...: integer_encoded = integer_encoded.reshape(len(integer_encoded),
1)
...: # Mengubah kategori menjadi angka
...: onehot_encoder = OneHotEncoder(sparse=False)
...: onehot_encoder.fit(integer_encoded)
# OneHotEncoder is deprecated in favor of OrdinalEncoder. FutureWarning: The handling of integer data will change in a future version of pandas. Currently, the categories are retained based on the range [0, max(values)], while in the future they will be determined based on the unique values.
# You can ignore this warning and silence this warning, you can
# specify "categories='auto'".
# In case you used a LabelEncoder before this OneHotEncoder to convert the
# categories into integers, you can now use the OneHotEncoder directly.
warnings.warn(msg, FutureWarning)
Out[11]:
OneHotEncoder(categorical_features=None, categories=None, drop=None,
              dtype=<class 'numpy.float64'>, handle_unknown='error',
              n_values=None, sparse=False)
```

**Gambar 7.186** Hasil Soal 7.

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 # Menconvert data train output menggunakan variabel
# label_encoder kedalam variabel train_output_int
3 train_output_int = label_encoder.transform(train_output)
4 # Menconvert variabel train_output_int kedalam fungsi
# onehot_encoder
5 train_output = onehot_encoder.transform(train_output_int.reshape(
    len(train_output_int), 1))
6 # Menconvert data test_output menggunakan variabel label_encoder
# kedalam variabel test_output_int
7 test_output_int = label_encoder.transform(test_output)
8 # Menconvert variabel test_output_int kedalam fungsi
# onehot_encoder
9 test_output = onehot_encoder.transform(test_output_int.reshape(
    len(test_output_int), 1))
```

```

10 # Menginisiasi variabel num_classes dengan isi variabel
    label_encoder dan classes
11 num_classes = len(label_encoder.classes_)
12 # mencetak hasil dari nomer Class berupa persen
13 print("Number of classes: %d" % num_classes)

```

```

In [12]: train_output_int = label_encoder.transform(train_output)
... # Mengonversi variabel train_output menjadi variabel integer
onehot_encoder = OneHotEncoder()
... # Mengonversi variabel train_output
onehot_encoder.fit(train_output_int.reshape(len(train_output_int),
1))
... # Mengonversi data test_output menggunakan variabel label_encoder
bedakan variabel test_output_int
... # test_output_int = label_encoder.transform(test_output)
... # Mengonversi variabel test_output_int menggunakan fungsi
onehot_encoder.transform
... # Mengonversi variabel test_output_int.reshape(len(test_output_int),
1))
... # Menginisiasi variabel num_classes dengan isi variabel
label_encoder dan classes
... # mencetak hasil dari nomer Class berupa persen
... # print("Number of classes: %d" % num_classes)
Number of classes: 369

```

**Gambar 7.187** Hasil Soal 8.

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 # Melakukan import library Sequential dari Keras
3 from keras.models import Sequential
4 # Melakukan import library Dense, Dropout, Flatten dari Keras
5 from keras.layers import Dense, Dropout, Flatten
6 # Melakukan import library Conv2D, MaxPooling2D dari Keras
7 from keras.layers import Conv2D, MaxPooling2D

```

```

In [13]: from keras.models import Sequential
... # Melakukan import library Dense, Dropout, Flatten dari Keras
... from keras.layers import Dense, Dropout, Flatten
... # Melakukan import library Conv2D, MaxPooling2D dari Keras
... from keras.layers import Conv2D, MaxPooling2D

```

**Gambar 7.188** Hasil Soal 9.

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 # Menginisiasi variabel model dengan isian library Sequential
3 model = Sequential()
4 # variabel model di tambahkan library Conv2D tigapuluhan dua bit
    dengan ukuran kernel 3 x 3 dan fungsi penghitungan relu dang
    menggunakan data train_input
5 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
6                 input_shape=np.shape(train_input[0])))
7 # variabel model di tambahkan dengan lib MaxPooling2D dengan
    ketentuan ukuran 2 x 2 pixel
8 model.add(MaxPooling2D(pool_size=(2, 2)))

```

```

9 # variabel model di tambahkan dengan library Conv2D 32 bit dengan
  kernel 3 x 3
10 model.add(Conv2D(32, (3, 3), activation='relu'))
11 # variabel model di tambahkan dengan lib MaxPooling2D dengan
  ketentuan ukuran 2 x 2 pixel
12 model.add(MaxPooling2D(pool_size=(2, 2)))
13 # variabel model di tambahkan library Flatten
14 model.add(Flatten())
15 # variabel model di tambahkan library Dense dengan fungsi tanh
16 model.add(Dense(1024, activation='tanh'))
17 # variabel model di tambahkan library dropout untuk memangkas
  data tree sebesar 50 persen
18 model.add(Dropout(0.5))
19 # variabel model di tambahkan library Dense dengan data dari
  num_classes dan fungsi softmax
20 model.add(Dense(num_classes, activation='softmax'))
21 # mengkompile data model untuk mendapatkan data loss akurasi dan
  optimasi
22 model.compile(loss='categorical_crossentropy', optimizer='adam',
  metrics=['accuracy'])
23 # mencetak variabel model kemudian memunculkan kesimpulan berupa
  data total parameter, trainable paremeter dan bukan trainable
  parameter
24 print(model.summary())

```

```

Model: "sequential_1"
Layer (type)          Output Shape       Param # 
===== 
conv2d_1 (Conv2D)      (None, 30, 30, 32)    896    
max_pooling2d_1 (MaxPooling2D) (None, 15, 15, 32)  0      
conv2d_2 (Conv2D)      (None, 15, 15, 32)    9248   
max_pooling2d_2 (MaxPooling2D) (None, 6, 6, 32)   0      
flatten_1 (Flatten)   (None, 1152)        0      
dense_1 (Dense)        (None, 1024)        1186072 
dropout_1 (Dropout)   (None, 1024)        0      
dense_2 (Dense)        (None, 260)         278225  
=====
Total params: 1,569,041
Trainable params: 1,489,041
Non-trainable params: 80
=====
None

```

Gambar 7.189 Hasil Soal 10.

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.

```

1 # In[11]: import sequential
2 # Melakukan import library keras callbacks
3 import keras.callbacks
4 # Menginisiasi variabel tensorboard dengan isi lib keras
5 tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-
  style')

```

```

In [18]: import keras.callbacks
...: # Menginisiasi variabel tensorboard dengan isi lib keras
...: tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-
  style')

```

Gambar 7.190 Hasil Soal 11.

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])

```

```

187668/187668 - 88s - loss: 1.5529 - accuracy: 0.6278 - val_loss: 0.9929
   - val_accuracy: 0.7288
Epoch 0/10
187668/187668 - 81s - loss: 0.9793 - accuracy: 0.7387 - val_loss: 0.8903
   - val_accuracy: 0.7518
Epoch 1/10
187668/187668 - 78s - loss: 0.8683 - accuracy: 0.7933 - val_loss: 0.8577
   - val_accuracy: 0.7545
Epoch 2/10
187668/187668 - 82s - loss: 0.7983 - accuracy: 0.7676 - val_loss: 0.8427
   - val_accuracy: 0.7629
Epoch 3/10
187668/187668 - 81s - loss: 0.7511 - accuracy: 0.7771 - val_loss: 0.8347
   - val_accuracy: 0.7791
Epoch 4/10
187668/187668 - 81s - loss: 0.7661 - accuracy: 0.7872 - val_loss: 0.8329
   - val_accuracy: 0.7676
Epoch 5/10
187668/187668 - 85s - loss: 0.6712 - accuracy: 0.7951 - val_loss: 0.8389
   - val_accuracy: 0.7657
Epoch 6/10
187668/187668 - 86s - loss: 0.6437 - accuracy: 0.8014 - val_loss: 0.8562
   - val_accuracy: 0.7635
Epoch 7/10
187668/187668 - 86s - loss: 0.6216 - accuracy: 0.8058 - val_loss: 0.8719
   - val_accuracy: 0.7672
Epoch 8/10
187668/187668 - 86s - loss: 0.5984 - accuracy: 0.8181 - val_loss: 0.8638
   - val_accuracy: 0.7652
3344/3344 [====================] - ETA: 0s
Test loss: 0.8059817292192628
Test accuracy: 0.76339944

```

**Gambar 7.191** Hasil Soal 12.

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
       fixcel mnjadi data yang berada pada codigan dibawah.
9     for dense_size in [128, 256, 512, 1024, 2048]:
10        # membuat looping untuk memangkas masing-masing data
           dengan ketentuan 0 persen 25 persen 50 persen dan 75 persen.
11        for dropout in [0.0, 0.25, 0.50, 0.75]:

```

```

12         # Menginisiasi variabel model Sequential
13         model = Sequential()
14         #membuat looping untuk variabel i dengan jarak dari
15         hasil konvolusi.
16         for i in range(conv2d_count):
17             # syarat jika i samadengan bobotnya 0
18             if i == 0:
19                 # Penambahan method add pada variabel model
20                 dengan konvolusi 2 dimensi 32 bit didalamnya dan membuat
21                 kernel dengan ukuran 3 x 3 dan rumus aktifasi relu dan data
22                 shape yang di hitung dari data train.
23                 model.add(Conv2D(32, kernel_size=(3, 3),
24                 activation='relu', input_shape=np.shape(train_input[0])))
25                 # jika tidak
26             else:
27                 # Penambahan method add pada variabel model
28                 dengan konvolusi 2 dimensi 32 bit dengan ukuran kernel 3 x3
29                 dan fungsi aktivasi relu
30                 model.add(Conv2D(32, kernel_size=(3, 3),
31                 activation='relu'))
32                 # Penambahan method add pada variabel model
33                 dengan isian method Max pooling berdimensi 2 dengan ukuran
34                 fixcel 2 x 2.
35                 model.add(MaxPooling2D(pool_size=(2, 2)))
36                 # merubah feature gambar menjadi 1 dimensi vektor
37                 model.add(Flatten())
38                 # Penambahan method dense untuk pematatan data dengan
39                 ukuran dense di tentukan dengan rumus fungsi tanh.
40                 model.add(Dense(dense_size , activation='tanh'))
41                 # membuat ketentuan jika pemangkasan lebih besar dari
42                 0 persen
43                 if dropout > 0.0:
44                     # Penambahan method dropout pada model dengan
45                     nilai dari dropout
46                     model.add(Dropout(dropout))
47                     # Penambahan method dense dengan fungsi num
48                     classs dan rumus softmax
49                     model.add(Dense(num_classes , activation='softmax'))
50                     # mongkompile variabel model dengan hasi loss
51                     optimasi dan akurasi matrix
52                     model.compile(loss='categorical_crossentropy',
53                     optimizer='adam' , metrics=['accuracy'])
54                     # melakukan log pada dir
55                     log_dir = './logs/conv2d_%d-dense_%d-dropout_%.2f' %
56 (conv2d_count, dense_size , dropout)
57                     # Menginisiasi variabel tensorflow dengan isian dari
58                     library keras dan nilai dari lig dir
59                     tensorboard = keras.callbacks.TensorBoard(log_dir=
60 log_dir)
61                     # Menginisiasi variabel start dengan isian dari
62                     library time menggunakan method time
63
64                     start = time.time()
65                     # Penambahan method fit pada model dengan data dari
66                     train input train output nilai batch nilai epoch verbose

```

```

    nilai 20 persen validation split dan callback dengan nilai
    tensorboard.
46     model.fit(train_input, train_output, batch_size=32,
47     epochs=10,
48         verbose=0, validation_split=0.2, callbacks
49     =[tensorboard])
50         # Menginisiasi variabel score dengan nilai evaluasi
51     dari model menggunakan data tes input dan tes output
52         score = model.evaluate(test_input, test_output,
53     verbose=2)
54         # Menginisiasi variabel end
55     end = time.time()
56         # Menginisiasi variabel elapsed
57     elapsed = end - start
58         # mencetak hasil perhitungan
59         print("Conv2D count: %d, Dense size: %d, Dropout: %.2
f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count
60     , dense_size, dropout, score[0], score[1], elapsed))
61         results.append((conv2d_count, dense_size, dropout,
62     score[0], score[1], elapsed))

```

```

.....
% (conv2d_count, dense_size, dropout)
.....
# Menginisiasi variabel tensorboard dengan isian
dari library keras menggunakan fungsi dir()
.....
tensorboard =
keras.callbacks.TensorBoard(log_dir=log_dir)
.....
# Menginisiasi variabel start dengan isian dari
library time menggunakan method time
.....
.....
start = time.time()
.....
# Pemanggilan method fit pada model dengan data dan
train_input train_output nilai batch_size 32, 20 persen
validation split dan nilai epoch 10
.....
model.fit(train_input, train_output, batch_size=32
epochs=10,
.....
verbose=0, validation_split=0.2,
callbacks=[tensorboard])
.....
# Menginisiasi variabel score dengan nilai evaluasi
dari model menggunakan data tes input dan tes output
.....
score = model.evaluate(test_input, test_output,
verbose=2)
.....
# Menginisiasi variabel end
.....
end = time.time()
.....
# Menginisiasi variabel elapsed
.....
elapsed = end - start
.....
# mencetak hasil perhitungan
.....
print("Conv2D count: %d, Dense size: %d, Dropout: %.2
f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count,
dense_size, dropout, score[0], score[1], elapsed))
.....
results.append((conv2d_count, dense_size, dropout,
score[0], score[1], elapsed))
.....

```

**Gambar 7.192** Hasil Soal 13.

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

```

```

9 model.add(Conv2D(32, (3, 3), activation='relu'))
10 # variabel model di tambahkan dengan lib MaxPooling2D dengan
     ketentuan ukuran 2 x 2 pixel
11 model.add(MaxPooling2D(pool_size=(2, 2)))
12 # variabel model di tambahkan library Flatten
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 paremeter dan bukan trainable
     parameter
23 print(model.summary())

```

| Layer (Type)                  | Output Shape       | Param # |
|-------------------------------|--------------------|---------|
| conv2d_10 (Conv2D)            | (None, 30, 30, 32) | 896     |
| max_pooling2d_10 (MaxPooling) | (None, 15, 15, 32) | 0       |
| conv2d_11 (Conv2D)            | (None, 15, 15, 32) | 9248    |
| max_pooling2d_11 (MaxPooling) | (None, 6, 6, 32)   | 0       |
| flatten_8 (Flatten)           | (None, 1152)       | 0       |
| dense_16 (Dense)              | (None, 128)        | 147584  |
| dropout_2 (Dropout)           | (None, 128)        | 0       |
| dense_17 (Dense)              | (None, 369)        | 47681   |
| <hr/>                         |                    |         |
| Total params: 295,326         |                    |         |
| Trainable params: 295,326     |                    |         |
| Non-trainable params: 0       |                    |         |
| <hr/>                         |                    |         |
| None                          |                    |         |

Gambar 7.193 Hasil Soal 14.

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 # melakukan join numpy menggunakan data train_input test_input
3 model.fit(np.concatenate((train_input, test_input)),
4           # kelanjutan data yang di gunakan pada join
           train_output test_output
5           np.concatenate((train_output, test_output)),
6           #menggunakan ukuran 32 bit dan epoch 10
7           batch_size=32, epochs=10, verbose=2)

```

```
-----  
Epoch 1/10  
168233/168233 - 83s - loss: 1.8214 - accuracy: 0.5784  
Epoch 2/10  
168233/168233 - 81s - loss: 1.0979 - accuracy: 0.7021  
Epoch 3/10  
168233/168233 - 77s - loss: 0.9918 - accuracy: 0.7246  
Epoch 4/10  
168233/168233 - 76s - loss: 0.9320 - accuracy: 0.7378  
Epoch 5/10  
168233/168233 - 79s - loss: 0.8945 - accuracy: 0.7453  
Epoch 6/10  
168233/168233 - 81s - loss: 0.8649 - accuracy: 0.7511  
Epoch 7/10  
168233/168233 - 78s - loss: 0.8446 - accuracy: 0.7553  
Epoch 8/10  
168233/168233 - 75s - loss: 0.8256 - accuracy: 0.7598  
Epoch 9/10  
168233/168233 - 74s - loss: 0.8111 - accuracy: 0.7616  
Epoch 10/10  
168233/168233 - 79s - loss: 0.8015 - accuracy: 0.7650
```

**Gambar 7.194** Hasil Soal 15.

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 # menyimpan model atau mengeksport model yang telah di jalantadi  
3 model.save("mathsymbols.model")
```

```
In [80]: model.save("mathsymbols.model")  
INFO:tensorflow:From C:\Users\TIANOWIKI\PycharmData\Roaming\Python  
\\Python37\site-packages\tensorflow\concrete\_ops\pywrap_tensorflow.py:1075: _ResourceVariable__init__(  
from tensorflow.python.resource_variable_ops) with constraint is  
deprecated and will be removed in a future version.  
Instructions for updating:  
If using Keras pass "constraint" arguments to layers.  
INFO:tensorflow:Assets written to: mathsymbols.model/assets
```

**Gambar 7.195** Hasil Soal 16.

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 # menyimpan label encoder dengan nama classes.npy  
3 np.save('classes.npy', label_encoder.classes_)
```

```
In [81]: np.save('classes.npy', label_encoder.classes_)
```

**Gambar 7.196** Hasil Soal 17.

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 # the code below could be placed in a separate file  
3 # mengimpport library keras model  
4 import keras.models
```

```

5 # Menginisiasi variabel model2 untuk meload model yang telah di
   simpan tadi
6 model2 = keras.models.load_model("mathsymbols.model")
7 # mencetak hasil model2
8 print(model2.summary())

```

```

In [8]: model.save("mathsymbols.model")
WARNING:tensorflow:From C:\Users\TIGER01\01\appData\Roaming\Python
\Python37\site-packages\tensorflow_core\python\ops
\resource_variable_ops.py:1786: constraint_keras_resource_variable__init__
From tensorflow.python.ops.resource_variable_ops import constraint
DeprecationWarning: Using a constraint with constraint_keras_resource_variable_ops() with constraint is
deprecated and will be removed in a future version.
INFO:tensorflow:If using Keras pass *_constraint arguments to layers.
INFO:tensorflow:Assets written to: mathsymbols.model.assets

```

**Gambar 7.197** Hasil Soal 18.

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 # Menginisiasi variabel label encoder ke 2 dengan isian fungsi
   label encoder.
3 label_encoder2 = LabelEncoder()
4 # Penambahan method classes dengan data classess yang di eksport
   tadi
5 label_encoder2.classes_ = np.load('classes.npy')
6 # membuat fumgsi predict dengan path img
7 def predict(img_path):
8     # Menginisiasi variabel newimg dengam membuay immage menjadi
       array dan membuka data berdasarkan img path
9     newimg = keras.preprocessing.image.img_to_array(pil_image.
       open(img_path))
10    # membagi data yang terdapat pada variabel newimg sebanyak
       255
11    newimg /= 255.0
12
13    # do the prediction
14    # Menginisiasi variabel predivtion dengan isian variabel
       model2 menggunakan fungsi predic dengan syarat variabel
       newimg dengan data reshape
15    prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
16
17    # figure out which output neuron had the highest score , and
       reverse the one-hot encoding
18    # Menginisiasi variabel inverted denagan label encoder2 dan
       menggunakan argmax untuk mencari skor luaran tertinggi
19    inverted = label_encoder2.inverse_transform([np.argmax(
       prediction)])
20    # mencetak prediksi gambar dan confidence dari gambar.
21    print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
       max(prediction)))

```

```
In [83]: label_encoder2 = LabelEncoder()
...: # Pembaruan method classes dengan data classes yang di
...: # buat
...: label_encoder2.classes_ = np.load('classes.npy')
...: # membuat fungsi predict dengan path img
...: def predict(img_path):
...:     # membaca gambar dan memuat label newling dengan membaca image menurut
...:     array dan memuat data berdasarkan img path
...:     newling = keras.preprocessing.image.img_to_array(pil_image.open(img_path))
...:     # membagi data yang terdapat pada variabel newling sebanyak
250
...:     newling /= 255.0
...:     # do the prediction
...:     # menginisiasi variabel predikton dengan isian variabel
...:     model = menggunakan fungsi predict dengan syarat variabel newling dengan
...:     data reshape
...:     prediction = model2.predict(newling.reshape(1, 32, 32, 3))
...:     ... # figure out which output neuron had the highest score, and
...:     reverse the one-hot encoding
...:     ...# mencari variabel inverted berversed dengan label encoder2 di
...:     menggunakan argmax untuk mencari skor luaran tertinggi
...:     label_encoder2.classes_[np.argmax(prediction)]]
...:     ...# mencari prediksi gambar dan confidence dari gambar,
...:     ...print("Prediction: %s, confidence: %.2f" % (inverted[0],
...:     np.max(prediction)))
```

**Gambar 7.198** Hasil Soal 19.

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.

```
1 # In[20]: grab an image (we'll just use a random training image
for demonstration purposes)
2 # mencari prediksi menggunakan fungsi prediksi yang di buat tadi
dari data di HASYv2/hasy-data/v2-00010.png
3 predict("HASYv2/hasy-data/v2-00010.png")
4 # mencari prediksi menggunakan fungsi prediksi yang di buat tadi
dari data di HASYv2/hasy-data/v2-00500.png
5 predict("HASYv2/hasy-data/v2-00500.png")
6 # mencari prediksi menggunakan fungsi prediksi yang di buat tadi
dari data di HASYv2/hasy-data/v2-00700.png
7 predict("HASYv2/hasy-data/v2-00700.png")
```

```
In [86]: predict("D:/v2-00010.png")
...: # mencari prediksi menggunakan fungsi prediksi yang di
...: # buat data di HASYv2/hasy-data/v2-00500.png
...: predict("D:/v2-00500.png")
...: # mencari prediksi menggunakan fungsi prediksi yang di
...: # buat data di HASYv2/hasy-data/v2-00700.png
...: predict("D:/v2-00700.png")
```

**Gambar 7.199** Hasil Soal 20.

### 7.7.3 Penanganan Error

- ModuleNotFoundError: No module named 'keras'

```
File "F:/Nico/Kampus/Artificial Intelligent/Chapter 7/k83A/src/
1174096/7/1174096.py", line 7, in <module>
    import keras.preprocessing.image
ModuleNotFoundError: No module named 'keras'

In [2]:
```

**Gambar 7.200** NameError

- Tuliskan Kode Error dan Jenis Error

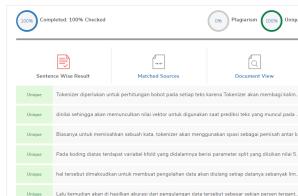
- ModuleNotFoundError

### 3. Cara Penangan Error

#### ▪ ModuleNotFoundError

Error terjadi karena belum dilakukan instalasi keras pada komputer. Sehingga library keras tidak dapat dipanggil. Untuk mengatasinya, lakukan instalasi library keras terlebih dahulu pada komputer.

#### 7.7.4 Bukti Tidak Plagiat



**Gambar 7.201**   Bukti Tidak Melakukan Plagiat Chapter 7

## DAFTAR PUSTAKA

---

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



# Index

---

disruptif, [xxiii](#)  
modern, [xxiii](#)