

**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 Tunnis

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>
<b>8 Chapter 8</b>	<b>15</b>



# DAFTAR ISI

---

Foreword	xiii
Kata Pengantar	xv
Acknowledgments	xvii
Acronyms	xix
Glossary	xxi
List of Symbols	xxiii
Introduction <i>Rolly Maulana Awangga, S.T., M.T.</i>	xxv
<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>
<b>8</b>	<b>Chapter 8</b>	<b>15</b>
8.1	1174083 - Bakti Qilan Mufid	15
8.1.1	Teori	15
8.1.2	Praktek	23
8.1.3	Penanganan Error	37
8.1.4	Bukti Tidak Plagiat	38
8.1.5	Link Youtube	38

# **FOREWORD**

---

Sepatah kata dari Kaprodi, Kabag Kemahasiswaan dan Mahasiswa



# KATA PENGANTAR

---

Buku ini diciptakan bagi yang awam dengan git sekalipun.

R. M. AWANGGA

*Bandung, Jawa Barat*

*Februari, 2019*



## ACKNOWLEDGMENTS

---

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

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

R. M. A.



## ACRONYMS

---

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



## GLOSSARY

---

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



# SYMBOLS

---

$A$  Amplitude

$\&$  Propositional logic symbol

$a$  Filter Coefficient

$B$  Number of Beats



# INTRODUCTION

---

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

Informatics Research Center  
Bandung, Jawa Barat, Indonesia

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

Rumusnya cuman sebagai contoh aja biar keren[?].

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



# **BAB 1**

---

# **CHAPTER 1**

---



## **BAB 2**

---

## **CHAPTER 2**

---



## **BAB 3**

---

## **CHAPTER 3**

---



## **BAB 4**

---

## **CHAPTER 4**

---



## **BAB 5**

---

## **CHAPTER 5**

---



## **BAB 6**

---

## **CHAPTER 6**

---



## BAB 7

---

# CHAPTER 7

---

### 7.1 1174006 - Kadek Diva Krishna Murti

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

```
1 @inproceedings{awangga2017colenak ,  
2   title={Colenak: GPS tracking model for post-stroke rehabilitation  
3   program using AES-CBC URL encryption and QR-Code},  
4   author={Awangga, Rolly Maulana and Fathonah, Nuraini Siti and  
5   Hasanudin, Trisna Irmayadi},  
6   booktitle={Information Technology, Information Systems and  
7   Electrical Engineering (ICITISEE), 2017 2nd International  
8   conferences on},  
9   pages={255--260},  
10  year={2017},  
11  organization={IEEE}  
12 }
```



**Gambar 7.1** Kecerdasan Buatan.

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
3. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

### 7.1.1 Teori

### 7.1.2 Praktek

### 7.1.3 Penanganan Error

### 7.1.4 Bukti Tidak Plagiat



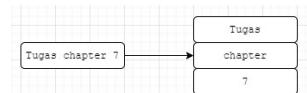
**Gambar 7.2** Kecerdasan Buatan.

## 7.2 1174066 - D.Irga B. Naufal Fakhri

### 7.2.1 Teori

#### 7.2.1.1 *Kenapa file teks harus di lakukan tokenizer*

Karena MTokenizer adalah proses membagi teks yang berupa kalimat, paragraf atau dokumen menjadi kata-kata atau bagian-bagian tertentu dalam kalimat tersebut. Contohnya kalimat "Tugas chapter 7", kalimat itu menjadi beberapa bagian yaitu "Tugas", "chapter", "7". Yang menjadi acuan yakni tanda baca dan spasi.



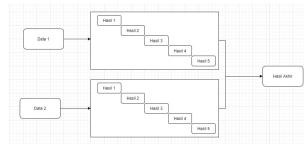
**Gambar 7.3** Teori 1

### 7.2.1.2 konsep dasar K Fold Cross Validation pada dataset komentar Youtube pada kode listing 7.1.

Konsep sederhana dari K Fold Cross Validation ialah Pada code ini:

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

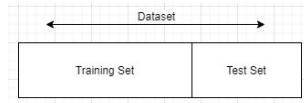
terdapat kfold yang bertujuan untuk melakukan split data menjadi 5 bagian dari dataset komentar Youtube tersebut. Sehingga dari setiap data yang sudah dibagi tersebut akan menghasilkan persentase dari setiap bagiannya, untuk menghasilkan hasil akhir dengan persentase yang cukup baik.



**Gambar 7.4** Teori 2

### 7.2.1.3 Apa maksudnya kode program for train, test in splits

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.



**Gambar 7.5** Teori 3

### 7.2.1.4 Apa maksudnya kode program train content = d['CONTENT'].iloc[train\_idx] dan test content = d['CONTENT'].iloc[test\_idx]

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 diinginkan.

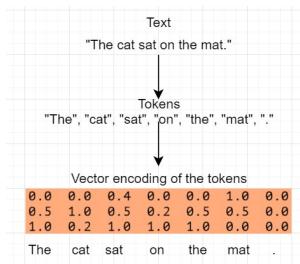
```
1 import pandas as pd
2
3 mydict = [{ 'a': 1, 'b': 2, 'c': 3, 'd': 4},
4           { 'a': 100, 'b': 200, 'c': 300, 'd': 400},
5           { 'a': 1000, 'b': 2000, 'c': 3000, 'd': 4000 }]
6 df = pd.DataFrame(mydict)
7 df
```

```
Out[2]:
a    1
b    2
c    3
d    4
Name: 0, dtype: int64
```

Gambar 7.6 Teori 4

#### 7.2.1.5 Apa maksud dari fungsi `tokenizer = Tokenizer(num_words=2000)` dan `tokenizer.fit_on_texts(train content)`

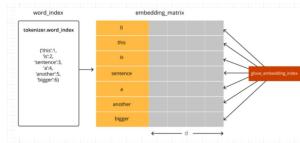
Fungsi tokenizer berfungsi untuk melakukan vektorisasi data kedalam bentuk token sebanyak 2000 kata. Dan selanjutnya akan melakukan fit tokenizer hanya untuk data training saja tidak dengan data testingnya.



Gambar 7.7 Teori 5

#### 7.2.1.6 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')`

Maksud dari baris diatas ialah untuk memasukkan text ke sebuah matrix dengan mode tfidf dan menginputkan data testing untuk di terjemahkan ke sebuah matriks.



Gambar 7.8 Teori 6

#### 7.2.1.7 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))`

Fungsi np.amax adalah nilai Maksimal. Jika sumbu tidak ada, hasilnya adalah nilai skalar. Jika sumbu diberikan, hasilnya adalah array dimensi `a.ndim - 1`.

```

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

```

Gambar 7.9 Teori 7

### 7.2.1.8 Apa 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

Fungsi dari baris kode tersebut ialah membuat train outputs dengan kategori dari class lalu dengan ketentuan iloc train idx. Kemudian membuat keluaran sebagai output.

```

>>> V_train
array([[0., 0., 0.],
       [0., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.], dtype=int64)
>>> V_train.flatten()
array([0., 0., ..., 1., 0., 0.], dtype=int64)
>>> V_train
array([[1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.], dtype=int64)
>>> np_utils.to_categorical(V_train)
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 1.,  0.,  0.],
       [ 0.,  0.,  1.],
       [ 1.,  0.,  0.],
       [ 1.,  0.,  0.]])

```

Gambar 7.10 Teori 8

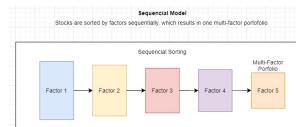
### 7.2.1.9 Apa maksud dari fungsi di listing 7.2.

```

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 7.11 Teori 9

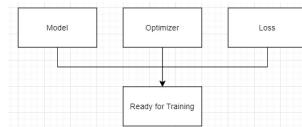
#### **7.2.1.10 Apa maksud dari fungsi di listing 7.3 dengan parameter tersebut**

```

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 7.12 Teori 10

#### **7.2.1.11 Apa itu Deep Learning**

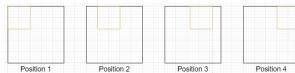
Deep Learning adalah salah satu cabang dari ilmu machine learning yang terdiri dari algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam machine learning dapat digunakan baik untuk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

#### **7.2.1.12 Apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning**

Deep Neural Network 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: Deep Neural Network 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.

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

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:

**Gambar 7.13** Teori 11

## 7.2.2 Praktek

### 7.2.2.1 Nomor 1

```

1 import csv #Import library csv
2 from PIL import Image as pil_image #Import library Image yaitu fungsi
    PIL (Python Imaging Library) yang berguna untuk mengolah data
    berupa gambar
3 import keras.preprocessing.image #Import library keras yang
    menggunakan method preprocessing yang digunakan untuk membuat
    neural network

```

Hasil:

**Gambar 7.14** Hasil No 1

### 7.2.2.2 Nomor 2

```

1 imgs = [] #Membuat variabel imgs
2 classes = [] #Membuat variabel classes dengan variabel kosong
3 with open('N:/HASYv2/hasy-data-labels.csv') as csvfile: #Membuka file
    hasy-data-labels.csv
4     csvreader = csv.reader(csvfile) #Membuat variabel csvreader yang
    berisi method csv.reader untuk membaca csvfile
5     i = 0 # membuat variabel i dengan isi 0
6     for row in csvreader: # Membuat looping pada variabel csvreader
7         if i > 0: #Ketentuannya jika i lebih kecil daripada 0
8             img = keras.preprocessing.image.img_to_array(pil_image.
open("N:/HASYv2/" + row[0])) #Dibuat variabel img dengan isi
9             keras untuk aktivasi neural network fungsi yang membaca data yang
10            berada dalam folder HASYv2 dengan input nilai -1.0 dan 1.0
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 0.0
14            to 1.0 instead of 0-255
15            img /= 255.0 #Membagi data yang ada pada fungsi img
16            sebanyak 255.0
17            imgs.append((row[0], row[2], img)) #Menambah nilai baru
18            pada imgs pada row ke 1 2 dan dilanjutkan dengan variabel img
19            classes.append(row[2]) #Menambahkan nilai pada row ke 2
20            pada variabel classes
21            i += 1 #Menambah nilai satu pada variabel i

```

Hasil:

```
[1] In [1]: import random
random.shuffle(imgs) #Melakukan randomize pada fungsi imgs
split_idx = int(0.8*len(imgs)) #Membuat variabel split_idx dengan nilai integer 80 persen dikali dari pengembalian jumlah dari variabel imgs
train = imgs[:split_idx] #Membuat variabel train dengan isi sebelum split_idx
test = imgs[split_idx:] #Membuat variabel test dengan isi setelah split_idx
```

Gambar 7.15 Hasil No 2

**7.2.2.3 Nomor 3**

```
[1] import random #Mengimport library random
2 random.shuffle(imgs) #Melakukan randomize pada fungsi imgs
3 split_idx = int(0.8*len(imgs)) #Membuat variabel split_idx dengan nilai integer 80 persen dikali dari pengembalian jumlah dari variabel imgs
4 train = imgs[:split_idx] #Membuat variabel train dengan isi sebelum split_idx
5 test = imgs[split_idx:] #Membuat variabel test dengan isi setelah split_idx
```

Hasil:

```
[1] In [1]: import random
random.shuffle(imgs)
split_idx = int(0.8*len(imgs))
train = imgs[:split_idx]
test = imgs[split_idx:]
```

Gambar 7.16 Hasil No 1

**7.2.2.4 Nomor 4**

```
[1] import numpy as np #Mengimport library numpy dengan inisial np
2 train_input = np.asarray(list(map(lambda row: row[2], train))) #
    Membuat variabel train_input dengan np method asarray yang mana membuat array dengan isi row 2 dari data train
3 test_input = np.asarray(list(map(lambda row: row[2], test))) #Membuat test_input input dengan np method asarray yang mana membuat array dengan isi row 2 dari data test
4 train_output = np.asarray(list(map(lambda row: row[1], train))) #
    Membuat variabel train_output dengan np method asarray yang mana membuat array dengan isi row 1 dari data train
5 test_output = np.asarray(list(map(lambda row: row[1], test))) #
    Membuat variabel test_output dengan np method asarray yang mana membuat array dengan isi row 1 dari data test
```

Hasil:

```
[1] In [1]: 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 7.17 Hasil No 4

**7.2.2.5 Nomor 5**

```

1 from sklearn.preprocessing import LabelEncoder #Mengimport library
    LabelEncoder dari sklearn
2 from sklearn.preprocessing import OneHotEncoder #Mengimport library
    OneHotEncoder dari sklearn

```

Hasil:

**Gambar 7.18** Hasil No 5

### 7.2.2.6 Nomor 6

```

1 label_encoder = LabelEncoder() #Membuat variabel label_encoder dengan
    isi LabelEncoder
2 integer_encoded = label_encoder.fit_transform(classes) #Membuat
    variabel integer_encoded yang berfungsi untuk mengkonvert
    variabel classes kedalam bentuk integer

```

Hasil:

**Gambar 7.19** Hasil No 6

### 7.2.2.7 Nomor 7

```

1 onehot_encoder = OneHotEncoder(sparse=False)#Membuat variabel
    onehot_encoder dengan isi OneHotEncoder
2 integer_encoded = integer_encoded.reshape(len(integer_encoded), 1) #
    Mengisi variabel integer_encoded dengan isi integer_encoded yang
    telah di convert pada fungsi sebelumnya
3 onehot_encoder.fit(integer_encoded) #Mengkonvert variabel
    integer_encoded kedalam onehot_encoder

```

Hasil:

**Gambar 7.20** Hasil No 7

### 7.2.2.8 Nomor 8

```

1 train_output_int = label_encoder.transform(train_output) #Mengkonvert
    data train output mengguanakan variabel label_encoder kedalam
    variabel train_output_int
2 train_output = onehot_encoder.transform(train_output_int.reshape(len(
    train_output_int), 1)) #Mengkonvert variabel train_output_int
    kedalam fungsi onehot_encoder

```

```

3 test_output_int = label_encoder.transform(test_output) #Mengkonvert
    data test_output mengguanakan variabel label_encoder kedalam
    variabel test_output_int
4 test_output = onehot_encoder.transform(test_output_int.reshape(len(
    test_output_int), 1)) #Mengkonvert variabel test_output_int
    kedalam fungsi onehot_encoder
5 num_classes = len(label_encoder.classes_) #Membuat variabel
    num_classes dengan isi variabel label_encoder dan classes_
6 print("Number of classes: %d" % num_classes) #Mencetak hasil dari
    nomer Class berupa persen

```

Hasil:

```

[8] train_output_int = label_encoder
train_output = onehot_encoder
test_output_int = label_encoder
test_output = onehot_encoder
num_classes = len(label_encoder)
print("Number of classes: %d" % num_classes)

⇒ Number of classes: 369

```

**Gambar 7.21** Hasil No 8

### 7.2.2.9 Nomor 9

```

1 from keras.models import Sequential #Mengimport library Sequential
    dari Keras
2 from keras.layers import Dense, Dropout, Flatten #Mengimport library
    Dense, Dropout, Flatten dari Keras
3 from keras.layers import Conv2D, MaxPooling2D #Mengimport library
    Conv2D, MaxPooling2D dari Keras

```

Hasil:

```

[9] from keras.models import Sequential #Mengimport library Sequential
from keras.layers import Dense, Dropout, Flatten #
from keras.layers import Conv2D, MaxPooling2D #Mengimport library
#Import tensorflow #Import tensorflow

```

**Gambar 7.22** Hasil No 9

### 7.2.2.10 Nomor 10

```

1 model = Sequential() #Membuat variabel model dengan isian library
    Sequential
2 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
    input_shape=np.shape(train_input[0]))) #Variabel
    model di tambahkan library Conv2D tigapuluhan dua bit dengan ukuran
    kernel 3 x 3 dan fungsi penghitungan relu yang menggunakan data
    train_input
4 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
    tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x 2
    pixel

```

```

5 model.add(Conv2D(32, (3, 3), activation='relu')) #Variabel model di tambahkan dengan library Conv2D 32bit dengan kernel 3 x 3
6 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x 2 pixel
7 model.add(Flatten()) #Variabel model di tambahkan library Flatten
8 model.add(Dense(1024, activation='tanh')) #Variabel model di tambahkan library Dense dengan fungsi tanh
9 model.add(Dropout(0.5)) #Variabel model di tambahkan library dropout untuk memangkas data tree sebesar 50 persen
10 model.add(Dense(num_classes, activation='softmax')) #Variabel model di tambahkan library Dense dengan data dari num_classes dan fungsi softmax
11 model.compile(loss='categorical_crossentropy', optimizer='adam',
12 metrics=['accuracy']) #Mengkompile data model untuk mendapatkan data loss akurasi dan optimasi
13 print(model.summary()) #Mencetak variabel model kemudian memunculkan kesimpulan berupa data total parameter, trainable paremeter dan bukan trainable parameter

```

Hasil:

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)	1188672
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 369)	378225
<hr/>		
total params: 1,569,043		
trainable params: 1,569,043		
non-trainable params: 0		
<hr/>		
None		

Gambar 7.23 Hasil No 10

### 7.2.2.11 Nomor 11

```

1 import keras.callbacks #Mengimport library keras dengan fungsi callbacks
2 tensorboard = keras.callbacks.TensorBoard(log_dir='N:/KB/hasyv2/logs/mnist-style') #Membuat variabel tensorboard dengan isi lib keras

```

Hasil:

```
(11) import keras.callbacks #Mengimport library keras dengan fungsi callbacks
      tensorboard = keras.callbacks.TensorBoard(log_dir='N:/KB/hasyv2/logs/mnist-style')
```

Gambar 7.24 Hasil No 11

### 7.2.2.12 Nomor 12

```

1 model.fit(train_input, train_output, #Fungsi model ditambahkan fungsi fit untuk mengetahui perhitungan dari train_input train_output
2 batch_size=32, #Dengan batch size 32 bit

```

```

3     epochs=10,
4     verbose=2,
5     validation_split=0.2,
6     callbacks=[tensorboard])
7
8 score = model.evaluate(test_input, test_output, verbose=2)
9 print('Test loss:', score[0])
10 print('Test accuracy:', score[1])

```

Hasil:

```

Train on 387608 samples, validate on 20918 samples
Epoch 1/10
  75/75 [==============================] - 1s - loss: 1.5342 - val_loss: 0.9709 - val_accuracy: 0.7226
Epoch 2/10
  75/75 [==============================] - 1s - loss: 0.9688 - accuracy: 0.7330 - val_loss: 0.8765 - val_accuracy: 0.7532
Epoch 3/10
  75/75 [==============================] - 1s - loss: 0.8508 - accuracy: 0.7557 - val_loss: 0.8189 - val_accuracy: 0.7861
Epoch 4/10
  75/75 [==============================] - 1s - loss: 0.7322 - accuracy: 0.7944 - val_loss: 0.6249 - val_accuracy: 0.7664
Epoch 5/10
  75/75 [==============================] - 1s - loss: 0.7318 - accuracy: 0.7946 - val_loss: 0.6420 - val_accuracy: 0.7668
Epoch 6/10
  75/75 [==============================] - 1s - loss: 0.6938 - accuracy: 0.7885 - val_loss: 0.6051 - val_accuracy: 0.7754
Epoch 7/10
  75/75 [==============================] - 1s - loss: 0.6512 - accuracy: 0.7951 - val_loss: 0.6204 - val_accuracy: 0.7753
Epoch 8/10
  75/75 [==============================] - 1s - loss: 0.6092 - accuracy: 0.8021 - val_loss: 0.6460 - val_accuracy: 0.7924
Epoch 9/10
  75/75 [==============================] - 1s - loss: 0.5871 - accuracy: 0.8071 - val_loss: 0.6111 - val_accuracy: 0.7941
Epoch 10/10
  75/75 [==============================] - 1s - loss: 0.5869 - accuracy: 0.8104 - val_loss: 0.6164 - val_accuracy: 0.7968
Test loss: 0.875423807773022
Test accuracy: 0.7649777882457732

```

**Gambar 7.25** Hasil No 12

### 7.2.2.13 Nomor 13

```

1 import time #Mengimport library time
2 results = [] #Membuat variabel result dengan array kosong
3 for conv2d_count in [1, 2]: #Melakukan looping dengan ketentuan
    konvolusi 2 dimensi 1 2
4     for dense_size in [128, 256, 512, 1024, 2048]: #Menentukan ukuran
        besaran fixcel dari data atau konvert 1 fixcel mnjadi data yang
        berada pada codigan dibawah.
5         for dropout in [0.0, 0.25, 0.50, 0.75]: #Membuat looping
            untuk memangkas masing-masing data dengan ketentuan 0 persen 25
            persen 50 persen dan 75 persen.
6             model = Sequential() #Membuat variabel model Sequential
7             for i in range(conv2d_count): #Membuat looping untuk
                variabel i dengan jarak dari hasil konvolusi.
8                 if i == 0: #Syarat jika i samadengan bobotnya 0
9                     model.add(Conv2D(32, kernel_size=(3, 3),
activation='relu', input_shape=np.shape(train_input[0]))) #
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.
10                else: #Jika tidak
11                    model.add(Conv2D(32, kernel_size=(3, 3),
activation='relu')) #Menambahkan method add pada variabel model
dengan konvolusi 2 dimensi 32 bit dengan ukuran kernel 3x3 dan
fungsi aktivasi relu
12                    model.add(MaxPooling2D(pool_size=(2, 2))) #
Menambahkan method add pada variabel model dengan isian method
Max pooling berdimensi 2 dengan ukuran fixcel 2 x 2.
13                    model.add(Flatten()) #Merubah feature gambar menjadi 1
dimensi vektor

```

```

14     model.add(Dense(dense_size , activation='tanh')) #
Menambahkan method dense untuk pemadatan data dengan ukuran dense
di tentukan dengan rumus fungsi tanh.
15     if dropout > 0.0: #Membuat ketentuan jika pemangkasan
lebih besar dari 0 persen
16         model.add(Dropout(dropout)) #Menambahkan method
dropout pada model dengan nilai dari dropout
17         model.add(Dense(num_classes , activation='softmax')) #
Menambahkan method dense dengan fungsi num classs dan rumus
softmax
18         model.compile(loss='categorical_crossentropy' , optimizer=
'adam' , metrics=[ accuracy]) #Mengcompile variabel model dengan
hasi loss optimasi dan akurasi matrix
19         log_dir = 'N:/KB/hasyv2/logs/conv2d_%d-dense_%d-dropout_%
.2f' % (conv2d_count , dense_size , dropout) #Melakukan log
20         tensorboard = keras.callbacks.TensorBoard(log_dir=log_dir
) # membuat variabel tensorboard dengan isian dari library keras
dan nilai dari log-dir
21
22         start = time.time() #Membuat variabel start dengan isian
dari library time menggunakan method time
23         model.fit(train_input , train_output , batch_size=32,
epochs=10,
24             verbose=0, validation_split=0.2, callbacks=[

tensorboard]) #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 tensorboard.
25         score = model.evaluate(test_input , test_output , verbose
=2) #Membuat variabel score dengan nilai evaluasi dari model
menggunakan data tes input dan tes output
26         end = time.time() #Membuat variabel end
27         elapsed = end - start #Membuat variabel elapsed
28         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)) #Mencetak
hasil perhitungan
29         results.append((conv2d_count , dense_size , dropout , score
[0] , score[1] , elapsed))

```

Hasil:

```

Conv2D count: 1, Dense size: 10, Dropout: 0.00 Loss: 1.00, Accuracy: 0.40, Time: 214 sec
Conv2D count: 1, Dense size: 10, Dropout: 0.25 Loss: 0.97, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 10, Dropout: 0.50 Loss: 0.97, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 10, Dropout: 0.75 Loss: 0.91, Accuracy: 0.77, Time: 220 sec
Conv2D count: 1, Dense size: 10, Dropout: 0.90 Loss: 0.91, Accuracy: 0.77, Time: 220 sec
Conv2D count: 1, Dense size: 10, Dropout: 0.95 Loss: 0.91, Accuracy: 0.77, Time: 220 sec
Conv2D count: 1, Dense size: 15, Dropout: 0.00 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 15, Dropout: 0.25 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 15, Dropout: 0.50 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 15, Dropout: 0.75 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 15, Dropout: 0.90 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 15, Dropout: 0.95 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 20, Dropout: 0.00 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 20, Dropout: 0.25 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 20, Dropout: 0.50 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 20, Dropout: 0.75 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 20, Dropout: 0.90 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 20, Dropout: 0.95 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 25, Dropout: 0.00 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 25, Dropout: 0.25 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 25, Dropout: 0.50 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 25, Dropout: 0.75 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 25, Dropout: 0.90 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 25, Dropout: 0.95 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 30, Dropout: 0.00 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 30, Dropout: 0.25 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 30, Dropout: 0.50 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 30, Dropout: 0.75 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 30, Dropout: 0.90 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 30, Dropout: 0.95 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 35, Dropout: 0.00 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 35, Dropout: 0.25 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 35, Dropout: 0.50 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 35, Dropout: 0.75 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 35, Dropout: 0.90 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 35, Dropout: 0.95 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 40, Dropout: 0.00 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 40, Dropout: 0.25 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 40, Dropout: 0.50 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 40, Dropout: 0.75 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 40, Dropout: 0.90 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 1, Dense size: 40, Dropout: 0.95 Loss: 0.93, Accuracy: 0.76, Time: 218 sec
Conv2D count: 2, Dense size: 10, Dropout: 0.00 Loss: 0.89, Accuracy: 0.72, Time: 214 sec
Conv2D count: 2, Dense size: 10, Dropout: 0.25 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 10, Dropout: 0.50 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 10, Dropout: 0.75 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 10, Dropout: 0.90 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 10, Dropout: 0.95 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 15, Dropout: 0.00 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 15, Dropout: 0.25 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 15, Dropout: 0.50 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 15, Dropout: 0.75 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 15, Dropout: 0.90 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 15, Dropout: 0.95 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 20, Dropout: 0.00 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 20, Dropout: 0.25 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 20, Dropout: 0.50 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 20, Dropout: 0.75 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 20, Dropout: 0.90 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 20, Dropout: 0.95 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 25, Dropout: 0.00 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 25, Dropout: 0.25 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 25, Dropout: 0.50 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 25, Dropout: 0.75 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 25, Dropout: 0.90 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 25, Dropout: 0.95 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 30, Dropout: 0.00 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 30, Dropout: 0.25 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 30, Dropout: 0.50 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 30, Dropout: 0.75 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 30, Dropout: 0.90 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 30, Dropout: 0.95 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 35, Dropout: 0.00 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 35, Dropout: 0.25 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 35, Dropout: 0.50 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 35, Dropout: 0.75 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 35, Dropout: 0.90 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 35, Dropout: 0.95 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 40, Dropout: 0.00 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 40, Dropout: 0.25 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 40, Dropout: 0.50 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 40, Dropout: 0.75 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 40, Dropout: 0.90 Loss: 0.89, Accuracy: 0.72, Time: 218 sec
Conv2D count: 2, Dense size: 40, Dropout: 0.95 Loss: 0.89, Accuracy: 0.72, Time: 218 sec

```

Gambar 7.26 Hasil No 13

### 7.2.2.14 Nomor 14

```

1 model = Sequential() #Membuat variabel model dengan isian library
    Sequential
2 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
    input_shape=np.shape(train_input[0])))#Variabel model di
        tambahkan library Conv2D tigapuluhan dua bit dengan ukuran kernel 3
        x 3 dan fungsi penghitungan relu dang menggunakan data
        train_input
3 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
        tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x 2
        pixel
4 model.add(Conv2D(32, (3, 3), activation='relu')) #Variabel model di
        tambahkan dengan library Conv2D 32bit dengan kernel 3 x 3
5 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
        tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x 2
        pixcel
6 model.add(Flatten()) #Variabel model di tambahkan library Flatten
7 model.add(Dense(128, activation='tanh')) #Variabel model di tambahkan
        library Dense dengan fungsi tanh
8 model.add(Dropout(0.5)) #Variabel model di tambahkan library dropout
        untuk memangkas data tree sebesar 50 persen
9 model.add(Dense(num_classes, activation='softmax')) #Variabel model
        di tambahkan library Dense dengan data dari num_classes dan
        fungsi softmax
10 model.compile(loss='categorical_crossentropy', optimizer='adam',
    metrics=['accuracy']) #Mengkompile data model untuk mendapatkan
        data loss akurasi dan optimasi
11 print(model.summary()) #Mencetak variabel model kemudian memunculkan
        kesimpulan berupa data total parameter, trainable paremeter dan
        bukan trainable parameter

```

Hasil:

Model: "sequential_42"		
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, 11, 11, 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)	47081
None		
Total params: 205,329 Trainable params: 205,329 Non-trainable params: 0		

Gambar 7.27 Hasil No 14

### 7.2.2.15 Nomor 15

```

1 model.fit(np.concatenate((train_input, test_input)), #Melakukan
    training yang isi datanya dari join numpy menggunakan data
    train_input test_input
2 np.concatenate((train_output, test_output)), #Kelanjutan
    data yang di gunakan pada join train_output test_output

```

```
batch_size=32, epochs=10, verbose=2) #Menggunakan ukuran 32
bit dan epoch 10
```

Hasil:

```

D:\> Epoch 1/10
Epoch 2/10
- 38s - loss: 1.7842 - accuracy: 0.5857
Epoch 3/10
- 38s - loss: 1.0848 - accuracy: 0.7044
Epoch 4/10
- 38s - loss: 0.9728 - accuracy: 0.7298
Epoch 5/10
- 38s - loss: 0.9141 - accuracy: 0.7411
Epoch 6/10
- 38s - loss: 0.8725 - accuracy: 0.7492
Epoch 7/10
- 38s - loss: 0.8408 - accuracy: 0.7551
Epoch 8/10
- 38s - loss: 0.8258 - accuracy: 0.7597
Epoch 9/10
- 38s - loss: 0.8032 - accuracy: 0.7651
Epoch 10/10
- 38s - loss: 0.7919 - accuracy: 0.7654
<keras.callbacks.callbacks.History at 0x7fee61ec97bb>
```

**Gambar 7.28** Hasil No 15

#### 7.2.2.16 Nomor 16

```
model.save("mathsymbols.model") #Menyimpan model atau mengeksport
model yang telah di jalan tadi
```

Hasil:

```
[1]: model.save("mathsymbols.model")#menyimpan model atau mengeksport model yang telah di jalan tadi
```

**Gambar 7.29** Hasil No 16

#### 7.2.2.17 Nomor 17

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

Hasil:

```
[1]: np.save('classes.npy', label_encoder.classes_())#menyimpan label encoder dengan nama classes.npy
```

**Gambar 7.30** Hasil No 17

#### 7.2.2.18 Nomor 18

```
1 import keras.models #Mengimport library keras model
2 model2 = keras.models.load_model("mathsymbols.model") #Membuat
variabel model2 untuk meload model yang telah di simpan tadi
3 print(model2.summary()) #Mencetak hasil model2
```

Hasil:

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, 15, 15, 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)	47681
Total params: 285,320		
Trainable params: 285,320		
Non-trainable params: 0		
None		

Gambar 7.31 Hasil No 18

**7.2.2.19 Nomor 19**

```

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

```

Hasil:

```

[15] label_encoder2 = LabelEncoder() # membuat variabel label encoder ke 2 dengan isian fungsi label encoder
label_encoder2.classes_ = np.load('content/classes.npy') #Menambahkan method classes
# predict
# prediction = model2.predict(newimg) #membuat variabel prediction dengan model2 dan newimg
newimg = keras.preprocessing.image.img_to_array(pil_image.open(img_path))
newimg /= 255.0 #membagi data yang terdapat pada variabel newimg sebanyak 255
# do the prediction
prediction = model2.predict(newimg.reshape(1, 32, 32, 3)) #membuat variabel prediction dengan model2 dan newimg
# 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 7.32 Hasil No 19

**7.2.2.20 Nomor 20**

```

1 predict("HASYv2/hasy-data/v2-00010.png") #Mencari prediksi
    menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2
    /hasy-data/v2-00010.png
2 predict("HASYv2/hasy-data/v2-00500.png") #Mencari prediksi
    menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2
    /hasy-data/v2-00500.png
3 predict("HASYv2/hasy-data/v2-00700.png") #Mencari prediksi
    menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2
    /hasy-data/v2-00700.png

```

Hasil:

```

    □ Prediction: A, confidence: 0.74
    □ Prediction: \pi, confidence: 0.48
    □ Prediction: \alpha, confidence: 0.98

```

**Gambar 7.33** Hasil No 20

### 7.2.3 Penanganan Error

#### 7.2.3.1 *Error*

- ProfilerNotRunningError

```
ProfilerNotRunningError: Cannot stop profiling. No profiler is running.
```

**Gambar 7.34** ProfilerNotRunningError

#### 7.2.3.2 *Solusi Error*

- ProfilerNotRunningError

tambahkan kode profile=10000000 pada parameter log\_dir

### 7.2.4 Bukti Tidak Plagiat



**Gambar 7.35** Bukti tidak plagiat

### 7.2.5 Link Youtube

[https://youtu.be/Vq\\_LZ89hTpg](https://youtu.be/Vq_LZ89hTpg)

## 7.3 1174080 - Handi Hermawan

### 7.3.1 Soal Teori

- Jelaskan kenapa file teks harus dilakukan tokenization. dilengkapi dengan ilustrasi atau gambar.

Tokenizer untuk memisahkan input text menjadi potongan yang memiliki arti disebut tokenization. Hasil dari proses tokenization disebut token. Token dapat berupa kata, kalimat, paragraph dan lainnya. Untuk ilustrasi, lihat gambar berikut:

Contoh kalimat	Dipecah menjadi :
• This is Andre's text, isn't it?	• This • is • Andre's • text. • isn't • it

Gambar 7.36 Teori 1

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

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

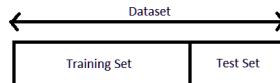
Konsep sederhana dari K Fold Cross Validation ialah Pada code tersebut terdapat kfold yang bertujuan untuk melakukan split data menjadi 5 bagian dari dataset komentar Youtube tersebut. Sehingga dari setiap data yang sudah dibagi tersebut akan menghasilkan persentase dari setiap baginya, untuk menghasilkan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

A	B	C	D	E	F	G	H
1 DATA			HASIL			AKURASI DATA	
2 1						%	
3 2						%	
4 3						%	
5 4						%	
6 5						%	

Gambar 7.37 Teori 2

- 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 7.38** Teori 3

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.

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

```

```

Out[5]:
a    1
b    2
c    3
d    4
Name: 0, dtype: int64

```

**Gambar 7.39** Teori 4

5. 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:

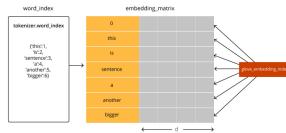
```

    Text
    "The cat sat on the mat."
    ↓
    Tokens
    "the", "cat", "sat", "on", "the", "mat", "."
    ↓
    Vector encoding of the tokens
    0.0 0.0 0.4 0.0 0.0 1.0 0.0
    0.5 1.0 0.5 0.2 0.5 0.5 0.0
    1.0 0.2 1.0 1.0 1.0 0.0 0.0
    the cat sat on the mat .
  
```

**Gambar 7.40** 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 7.41** 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]])
>>> npamax(a)           # Maximum of the flattened array
3
>>> npamax(a, axis=0)   # Maxima along the first axis
array([2, 3])
>>> npamax(a, axis=1)   # Maxima along the second axis
array([1, 3])
>>> npamax(a, where=[False, True], initial=-1, axis=0)
array([-1, 3])
>>> b = np.arange(5, dtype=float)
>>> b[2] = np.nan
>>> npamax(b)
nan
>>> npamax(b, where=~np.isnan(b), initial=-1)
4.0
>>> np.nanmax(b)
4.0
  
```

**Gambar 7.42** 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:

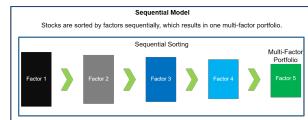
```
>>> V_train
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       ...,
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.], dtype=int64)
>>> V_train.flatten()
array([1., 0., 0., ..., 1., 0., 0.], dtype=int64)
>>> V_train_idx
array([[1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.], dtype=int64)
>>> np_utils.to_categorical(V_train)
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.],
       [ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.],
       [ 1.,  0.,  0.],
```

**Gambar 7.43** 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.



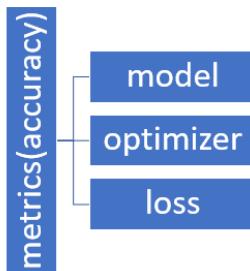
**Gambar 7.44** Teori 9

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 7.45** Teori 10

11. Jelaskan apa itu Deep Learning.

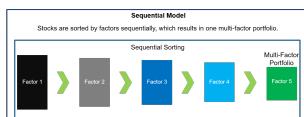
Deep Learning adalah salah satu cabang dari ilmu machine learning yang terdiri dari algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam machine learning dapat digunakan baik untuk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

$1174080 \ mod3+1 \times 1174080 \ mod3+1 = 2 \times 2$ , adapun ilustrasi gambar nya sebagai berikut :



**Gambar 7.46** 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 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 7.47** 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 :

### 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 7.48** 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 [1]: import numpy as np
...
...
...
...
...

```

**Gambar 7.49** 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
...

```

**Gambar 7.50** 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 7.51 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)
... onehot_encoder.fit(integer_encoded)
... OneHotEncoder(categories='auto', drop=None, dtype<class 'numpy.float64'>,
handle_unknown='error', sparse=False)
```

Gambar 7.52 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
test_output_int = np.array([label_encoder.transform(test_output)])
test_output_int = vectorizer_int.transform(test_output_int.reshape(-1, test_output_int.shape[0])))
num_classes = len(label_encoder.classes_)
# print('Number of classes: ', num_classes)
Number of classes: 369
```

Gambar 7.53 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 7.54 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
22 # optimasi
23 model.compile(loss='categorical_crossentropy', optimizer='adam',
24 metrics=['accuracy'])
25 # mencetak variabel model kemudian memunculkan kesimpulan berupa
26 # data total parameter, trainable paremeter dan bukan trainable
27 # parameter
28 print(model.summary())

```

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

Layer (Type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d_6 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_7 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_7 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_5 (Flatten)	(None, 352)	0
dense_9 (Dense)	(None, 28)	147584
dropout_4 (Dropout)	(None, 28)	0
dense_10 (Dense)	(None, 369)	47601

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

**Gambar 7.55** 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 :

**Gambar 7.56** 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 1/10: loss: 1.8815 - val_loss: 0.6283 - val_accuracy: 0.7382
Epoch 2/10: loss: 1.8815 - val_loss: 0.6283 - val_accuracy: 0.7382
Epoch 3/10: loss: 0.8104 - accuracy: 0.7477 - val_loss: 0.6862 - val_accuracy: 0.7602
Epoch 4/10: loss: 0.8228 - accuracy: 0.7312 - val_loss: 0.6350 - val_accuracy: 0.7629
Epoch 5/10: loss: 0.7751 - accuracy: 0.7716 - val_loss: 0.6469 - val_accuracy: 0.7618
Epoch 6/10: loss: 0.7751 - accuracy: 0.7804 - val_loss: 0.6902 - val_accuracy: 0.7642
Epoch 7/10: loss: 0.7934 - accuracy: 0.7859 - val_loss: 0.8566 - val_accuracy: 0.7654
Epoch 8/10: loss: 0.6765 - accuracy: 0.7924 - val_loss: 0.8462 - val_accuracy: 0.7639
Epoch 9/10: loss: 0.6532 - accuracy: 0.7935 - val_loss: 0.8531 - val_accuracy: 0.7681
Epoch 10/10: loss: 0.6532 - accuracy: 0.8021 - val_loss: 0.8792 - val_accuracy: 0.7611
test loss: 0.82084381892023
test accuracy: 0.7645130000000005

```

**Gambar 7.57** Hasil Soal 12.

### 13. Soal 13

```

1 # In[13]:try various model configurations and parameters to find
2     the best
3 # mengimport librari time
4 import time
5 #membuat variabel result dengan array kosong
6 results = []
7 # melakukan looping dengan ketentuan konvolusi 2 dimensi 1 2
8 for conv2d_count in [1, 2]:
9     # menentukan ukuran besaran fixcel dari data atau konvert 1
10    fixcel mnjadi data yang berada pada codigan dibawah.
11    for dense_size in [128, 256, 512, 1024, 2048]:
12        # membuat looping untuk memangkas masing-masing data
13        dengan ketentuan 0 persen 25 persen 50 persen dan 75 persen.
14        for dropout in [0.0, 0.25, 0.50, 0.75]:
15            # membuat variabel model Sequential
16            model = Sequential()
17            #membuat looping untuk variabel i dengan jarak dari
18            hasil konvolusi.
19            for i in range(conv2d_count):
20                # syarat jika i samadengan bobotnya 0
21                if i == 0:
22                    # menambahkan method add pada variabel model
23                    dengan konvolusi 2 dimensi 32 bit didalamnya dan membuat
24                    kernel dengan ukuran 3 x 3 dan rumus aktifasi relu dan data
25                    shape yang di hitung dari data train.
26                    model.add(Conv2D(32, kernel_size=(3, 3),
27 activation='relu', input_shape=np.shape(train_input[0])))

```

```

20             # jika tidak
21         else:
22             # menambahkan method add pada variabel model
23             dengan konvolusi 2 dimensi 32 bit dengan ukuran kernel 3 x3
24             dan fungsi aktivasi relu
25             model.add(Conv2D(32, kernel_size=(3, 3),
26                             activation='relu'))
26                 # menambahkan method add pada variabel model
27                 dengan isian method Max pooling berdimensi 2 dengan ukuran
28                 fixcel 2 x 2.
29                 model.add(MaxPooling2D(pool_size=(2, 2)))
30                 # merubah feature gambar menjadi 1 dimensi vektor
31                 model.add(Flatten())
32                 # menambahkan method dense untuk pemanatan data
33                 dengan ukuran dense di tentukan dengan rumus fungsi tanh.
34                 model.add(Dense(dense_size, activation='tanh'))
35                 # membuat ketentuan jika pemangkasan lebih besar dari
36                 0 persen
37                 if dropout > 0.0:
38                     # menambahkan method dropout pada model dengan
39                     nilai dari dropout
40                     model.add(Dropout(dropout))
41                     # menambahkan method dense dengan fungsi num
42                     classs dan rumus softmax
43                     model.add(Dense(num_classes, activation='softmax'))
44                     # mongkompile variabel model dengan hasi loss
45                     optimasi dan akurasi matrix
46                     model.compile(loss='categorical_crossentropy',
47                         optimizer='adam', metrics=['accuracy'])
47                         # melakukan log pada dir
48                         log_dir = './logs/conv2d-%d-dense_%d-dropout_%.2f' %
49                         (conv2d_count, dense_size, dropout)
50                         # membuat variabel tensorboard dengan isian dari
51                         librari keras dan nilai dari lig dir
52                         tensorflow = keras.callbacks.TensorBoard(log_dir=
53                         log_dir)
54                         # membuat variabel start dengan isian dari librari
55                         time menggunakan method time
56
57             start = time.time()
58             # menambahkan method fit pada model dengan data dari
59             train input train output nilai batch nilai epoch verbose
60             nilai 20 persen validation split dan callback dengan nilai
61             tnsorboard.
62             model.fit(train_input, train_output, batch_size=32,
63             epochs=10,
64             verbose=0, validation_split=0.2, callbacks
65             =[tensorboard])
66             # membuat variabel score dengan nilai evaluasi dari
67             model menggunakan data tes input dan tes output
68             score = model.evaluate(test_input, test_output,
69             verbose=2)
70             # membuat variabel end
71             end = time.time()
72             # membuat variabel elapsed
73             elapsed = end - start

```

```

54     # mencetak hasil perhitungan
55     print("Conv2D count: %d, Dense size: %d, Dropout: %.2f"
56         " f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count,
57             , dense_size, dropout, score[0], score[1], elapsed))
58     results.append((conv2d_count, dense_size, dropout,
59         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 :

**Gambar 7.58** Hasil Soal 13.

#### 14. Soal 14

```

1 # In[14]: rebuild/retrain a model with the best parameters (from
2     # the search) and use all data
3 # membuat variabel model dengan isian librari Sequential
4 model = Sequential()
5 # variabel model di tambahkan librari 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 librari 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 librari Flatten
20 model.add(Flatten())
21 # variabel model di tambahkan librari Dense dengan fungsi tanh
22 model.add(Dense(128, activation='tanh'))
23 # variabel model di tambahkan librari dropout untuk memangkas
24     # data tree sebesar 50 persen
25 model.add(Dropout(0.5))
26 # variabel model di tambahkan librari Dense dengan data dari
27     # num_classes dan fungsi softmax

```

```

19 model.add(Dense(num_classes, activation='softmax'))
20 # mengkompile data model untuk mendapatkan data loss akurasi dan
21 # optimasi
22 model.compile(loss='categorical_crossentropy', optimizer='adam',
23 metrics=['accuracy'])
23 # mencetak variabel model kemudian memunculkan kesimpulan berupa
24 # data total parameter, trainable paremeter dan bukan trainable
25 # parameter
25 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_6 (Conv2D)	(None, 36, 36, 32)	896
max_pooling2d_6 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_7 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_7 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_5 (Flatten)	(None, 1152)	0
dense_9 (Dense)	(None, 128)	147584
dropout_8 (Dropout)	(None, 128)	0
dense_10 (Dense)	(None, 369)	47681

Model: "sequential\_5"

Trainable params: 265,329  
Non-trainable params: 0

**Gambar 7.59** Hasil Soal 14.

## 15. Soal 15

```

1 # In[15]: join train and test data so we train the network on all
2 # data we have available to us
3 # melakukan join numpy menggunakan data train_input test_input
4 model.fit(np.concatenate((train_input, test_input)),
5           # kelanjutan data yang di gunakan pada join
6           train_output test_output
7           np.concatenate((train_output, test_output)),
8           #menggunakan ukuran 32 bit dan epoch 10
9           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: 1.8744 - 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 7.60** 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 7.61** 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 7.62** 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 7.63 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 7.64** 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: 0.58
predict("HASYv2/hasy-data/v2-00700.png")
prediction: V_alpha, confidence: 0.88
```

**Gambar 7.65** Hasil Soal 20.

### 7.3.3 Penanganan Error

#### 1. KeyboardInterrupt

```
file = open('input/48x48x3image.jpg', 'rb')
img = keras.preprocessing.image.load_img(file,image.open("HASYv2/*.*"))
img = np.array(img)
img = np.expand_dims(img, axis=0)
fp = multilabel_confusion_matrix(y_true,y_pred)
KeyboardInterrupt
```

**Gambar 7.66** KeyboardInterrupt

## 2. Cara Penanganan Error

- KeyboardInterrupt

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

### 7.3.4 Bukti Tidak Plagiat



Gambar 7.67 Bukti Tidak Melakukan Plagiat Chapter 7

### 7.3.5 Link Video Youtube

<https://youtu.be/pV9yrZNEZj4>

## 7.4 1174079 - Chandra Kirana Poetra

### 7.4.1 Teori

#### 7.4.1.1 Kenapa file teks harus di lakukan tokenizer

Karena tokenizer berguna untuk memproses atau memisahkan bagian bagian pada teks menjadi beberapa bagian, seperti kalimat atau kata. tokenizer bekerja dengan cara memisahkan kata berdasarkan spasi dan tanda baca



Gambar 7.68 Teori 1

#### 7.4.1.2 konsep dasar K Fold Cross Validation pada dataset komentar Youtube pada kode listing 7.1.

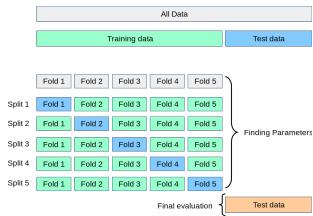
Konsep sederhana dari K Fold Cross Validation ialah Pada code ini:

```

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

```

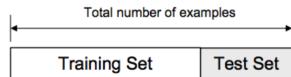
Kfold diatas bertujuan untuk membagi data kedalam 5 bagian yang nantinya akan menampung komentar dari youtube. data yang dibagi ini nanti akan dihitung dan akan menghasilkan presentase yang bisa dibilang cukup baik



Gambar 7.69 Teori 2

#### 7.4.1.3 Apa maksudnya kode program for train, test in splits

Data for train adalah data training set, data ini adalah data yang akan kita uji, sementara test in splits adalah data yang akan kita gunakan untuk validasi data training set, keduanya kemudian dibandingkan dan menghasilkan suatu presentase.



Gambar 7.70 Teori 3

#### 7.4.1.4 Apa maksudnya kode program train content = df['CONTENT'].iloc[train\_idx] dan test content = df['CONTENT'].iloc[test\_idx]

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.

```

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
  
```

```

Out[2]:
a    1
b    2
c    3
d    4
Name: 0, dtype: int64
  
```

Gambar 7.71 Teori 4

#### 7.4.1.5 Apa maksud dari fungsi `tokenizer = Tokenizer(num_words=2000)` dan `tokenizer.fit on texts(train content)`

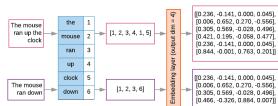
Berfungsi untuk melakukan proses vektorisasi data menjadi token sebanyak 2000 kata



Gambar 7.72 Teori 5

#### 7.4.1.6 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')`

Memasukan teks kedalam suatu matrix dengan menggunakan metode TFIDF dan memasukan data testing yang akan diterjemahkan ke suatu matriks



Gambar 7.73 Teori 6

#### 7.4.1.7 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))`

Fungsi np.amax digunakan untuk mencari nilai maksimal dari suatu array Jika array tidak ada sumbunya, hasilnya adalah nilai skalar. Jika sumbu diberikan, hasilnya adalah array dimensi a.ndim - 1.

```
>>> 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([2, 3])
>>> np.amax(a, axis=1)                         # Maxima along the second axis
array([1, 3])
>>> np.amax(a, where=[False, True], initial=-1, axis=0)
array([-1, 3])
>>> b = np.arange(5, dtype=float)
>>> b[2] = np.Nan
>>> np.amax(b)
nan
>>> np.amax(b, where=~np.isnan(b), initial=-1)
4.0
>>> np.nanmax(b)
4.0
```

Gambar 7.74 Teori 7

#### 7.4.1.8 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

Membuat variable dengan nama train outputs yang diisi dengan kategori dari class dengan menggunakan ketentuan iloc train idx. Kemudian menampungnya.

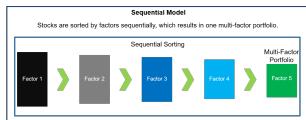
```
>>> V_train
array([[1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.], dtype=int64)
>>> V_train.flatten()
array([1., 0., 0., ..., 1., 0., 0.], dtype=int64)
>>> V_test
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.], dtype=int64)
>>> np_utils.to_categorical(V_train)
array([[ 1.,  0.,  0.],
       [ 1.,  0.,  0.],
       [ 1.,  0.,  0.],
       [ 1.,  0.,  0.],
       [ 1.,  0.,  0.],
       [ 1.,  0.,  0.]])
```

Gambar 7.75 Teori 8

#### 7.4.1.9 Apa maksud dari fungsi di listing 7.2.

```
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'))
```

Fungsinya adalah model perlu mengetahui jenis data input apa yang digunakan atau yang seharusnya. Oleh karena itu, lapisan pertama adalah model Sequential (Karena model sequential ini dapat mengerjakan inferensi secara otomatis) perlu menerima informasi tentang bentuk inputnya.



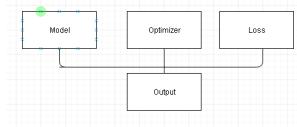
Gambar 7.76 Teori 9

#### 7.4.1.10 Apa maksud dari fungsi di listing 7.3 dengan parameter tersebut

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

Kode diatas merupakan fungsi yang digunakan untuk mendefinisikan loss function, optimizer, dan juga metrics yang akan digunakan pada data yang akan kita train. categoricalcrossentropy merupakan function loss yang kita gunakan untuk melakukan perhitungan, sementara optimizer yang kita gunakan adalah adam, selain adam, ada

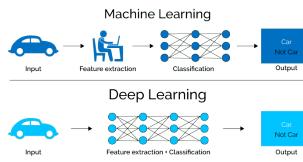
juga SGD,RMSprop, dan lain lain dan metrics accuracy adalah value yang akan kita cari nilainya



**Gambar 7.77** Teori 10

#### 7.4.1.11 Apa itu Deep Learning

Merupakan cabang dari machine learning yang berasal dari cabang lainnya yaitu artificial intelligence, deep learning mempunyai kemampuan jaringan pembelajaran dengan metode unsupervised dari data yang tidak ada strukturnya maupun tidak ada label yang dikenal juga sebagai deep neural network



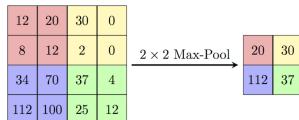
#### 7.4.1.12 Apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning

Komponen Neural netwrok adalah neuron yang dilabeli sebagai  $j$  yang menerima input dari neuron sebelumnya, seringkali dalam bentuk fungsi identifikasi untuk menyediakan output, sementara deep learning menggunakan motherboard, prosesor, ram, dan cpu untuk melakukan prosesnya

Neural netwrok menggunakan metode feed forward neural network atau bisa juga recurrent network sementara deep learning menggunakan unsupervised pretrained network atau convolutional neural network

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

Algoritma Konvolusi merupakan suatu algoritma yang dapat memproses gambar yang nantinya digunakan untuk membedakan satu gambar dengan yang lainnya, strides merupakan salah satu parameter yang ada, yang digunakan untuk mendefinisikan jumlah pergerakan pixel pada input matrix, ketika didefinisikan 1 stridenya, maka satu pixels akan pindah dalam satu waktu, jika 2 maka dua pixel akan pindah secara bersamaan dalam satu waktu

**Gambar 7.78** Teori 11

## 7.4.2 Praktek

### 7.4.2.1 Nomor 1

```

1 import csv #Import library csv
2 from PIL import Image as pil_image #Import library Image yaitu fungsi
   PIL (Python Imaging Library) yang berguna untuk mengolah data
   berupa gambar
3 import keras.preprocessing.image #Import library keras yang
   menggunakan method preprocessing yang digunakan untuk membuat
   neural network

```

### 7.4.2.2 Nomor 2

```

1 imgs = [] #Membuat variabel imgs
2 classes = [] #Membuat variabel classes dengan variabel kosong
3 with open('E:/hasy-data-labels.csv') as csvfile: #Membuka file hasy-
   data-labels.csv
4     csvreader = csv.reader(csvfile) #Membuat variabel csvreader yang
   berisi method csv.reader untuk membaca csvfile
5     i = 0 # membuat variabel i dengan isi 0
6     for row in csvreader: # Membuat looping pada variabel csvreader
7         if i > 0: #Ketentuannya jika i lebih kecil daripada 0
8             img = keras.preprocessing.image.img_to_array(pil_image.
   open("E:/HASYv2/" + row[0])) #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
9             # neuron activation functions behave best when input
   values are between 0.0 and 1.0 (or -1.0 and 1.0),
10            # so we rescale each pixel value to be in the range 0.0
   to 1.0 instead of 0-255
11            img /= 255.0 #Membagi data yang ada pada fungsi img
   sebanyak 255.0
12            imgs.append((row[0], row[2], img)) #Menambah nilai baru
   pada imgs pada row ke 1 2 dan dilanjutkan dengan variabel img
13            classes.append(row[2]) #Menambahkan nilai pada row ke 2
   pada variabel classes
14            i += 1 #Menambah nilai satu pada variabel i

```

### 7.4.2.3 Nomor 3

```

1 import random #Mengimport library random
2 random.shuffle(imgs) #Melakukan randomize pada fungsi imgs

```

```

3 split_idx = int(0.8*len(imgs)) #Membuat variabel split_idx dengan
    nilai integer 80 persen dikali dari pengembalian jumlah dari
    variabel imgs
4 train = imgs[:split_idx] #Membuat variabel train dengan isi sebelum
    split_idx
5 test = imgs[split_idx:] #Membuat variabel test dengan isi setelah
    split_idx

```

#### 7.4.2.4 Nomor 4

```

1 import numpy as np #Mengimport library numpy dengan inisial np
2 train_input = np.asarray(list(map(lambda row: row[2], train))) #
    Membuat variabel train_input dengan np method asarray yang mana
    membuat array dengan isi row 2 dari data train
3 test_input = np.asarray(list(map(lambda row: row[2], test))) #Membuat
    test_input input dengan np method asarray yang mana membuat
    array dengan isi row 2 dari data test
4 train_output = np.asarray(list(map(lambda row: row[1], train))) #
    Membuat variabel train_output dengan np method asarray yang mana
    membuat array dengan isi row 1 dari data train
5 test_output = np.asarray(list(map(lambda row: row[1], test))) #
    Membuat variabel test_output dengan np method asarray yang mana
    membuat array dengan isi row 1 dari data test

```

#### 7.4.2.5 Nomor 5

```

1 from sklearn.preprocessing import LabelEncoder #Mengimport library
    LabelEncoder dari sklearn
2 from sklearn.preprocessing import OneHotEncoder #Mengimport library
    OneHotEncoder dari sklearn

```

#### 7.4.2.6 Nomor 6

```

1 label_encoder = LabelEncoder() #Membuat variabel label_encoder dengan
    isi LabelEncoder
2 integer_encoded = label_encoder.fit_transform(classes) #Membuat
    variabel integer_encoded yang berfungsi untuk mengkonvert
    variabel classes kedalam bentuk integer

```

#### 7.4.2.7 Nomor 7

```

1 onehot_encoder = OneHotEncoder(sparse=False)#Membuat variabel
    onehot_encoder dengan isi OneHotEncoder
2 integer_encoded = integer_encoded.reshape(len(integer_encoded), 1) #
    Mengisi variabel integer_encoded dengan isi integer_encoded yang
    telah di convert pada fungsi sebelumnya
3 onehot_encoder.fit(integer_encoded) #Mengkonvert variabel
    integer_encoded kedalam onehot_encoder

```

#### 7.4.2.8 Nomor 8

```

1 train_output_int = label_encoder.transform(train_output) #Mengkonvert
    data train output mengguanakan variabel label_encoder kedalam
    variabel train_output_int
2 train_output = onehot_encoder.transform(train_output_int.reshape(len(
    train_output_int), 1)) #Mengkonvert variabel train_output_int
    kedalam fungsi onehot_encoder
3 test_output_int = label_encoder.transform(test_output) #Mengkonvert
    data test_output mengguanakan variabel label_encoder kedalam
    variabel test_output_int
4 test_output = onehot_encoder.transform(test_output_int.reshape(len(
    test_output_int), 1)) #Mengkonvert variabel test_output_int
    kedalam fungsi onehot_encoder
5 num_classes = len(label_encoder.classes_) #Membuat variabel
    num_classes dengan isi variabel label_encoder dan classes
6 print("Number of classes: %d" % num_classes) #Mencetak hasil dari
    nomer Class berupa persen

```

#### 7.4.2.9 Nomor 9

```

1 from keras.models import Sequential #Mengimport library Sequential
    dari Keras
2 from keras.layers import Dense, Dropout, Flatten #Mengimport library
    Dense, Dropout, Flatten dari Keras
3 from keras.layers import Conv2D, MaxPooling2D #Mengimport library
    Conv2D, MaxPooling2D dari Keras

```

#### 7.4.2.10 Nomor 10

```

1 model = Sequential() #Membuat variabel model dengan isian library
    Sequential
2 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
    input_shape=np.shape(train_input[0]))) #Variabel
    model di tambahkan library Conv2D tigapuluhan dua bit dengan ukuran
    kernel 3 x 3 dan fungsi penghitungan relu dang menggunakan data
    train_input
4 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
    tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x 2
    pixel
5 model.add(Conv2D(32, (3, 3), activation='relu')) #Variabel model di
    tambahkan dengan library Conv2D 32bit dengan kernel 3 x 3
6 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
    tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x 2
    pixel
7 model.add(Flatten()) #Variabel model di tambahkan library Flatten
8 model.add(Dense(1024, activation='tanh')) #Variabel model di
    tambahkan library Dense dengan fungsi tanh
9 model.add(Dropout(0.5)) #Variabel model di tambahkan library dropout
    untuk memangkas data tree sebesar 50 persen
10 model.add(Dense(num_classes, activation='softmax')) #Variabel model
    di tambahkan library Dense dengan data dari num_classes dan
    fungsi softmax

```

```

11 model.compile(loss='categorical_crossentropy', optimizer='adam',
12                 metrics=['accuracy']) #Mengkompile data model untuk
13 mendapatkan data loss akurasi dan optimasi
14 print(model.summary()) #Mencetak variabel model kemudian memunculkan
15 kesimpulan berupa data total parameter, trainable paremeter dan
16 bukan trainable parameter

```

#### 7.4.2.11 Nomor 11

```

1 import keras.callbacks #Mengimport library keras dengan fungsi
2         callbacks
3 tensorboard = keras.callbacks.TensorBoard(log_dir='E:/KB/hasyv2/logs/
4             mnist-style') #Membuat variabel tensorboard dengan isi lib keras

```

#### 7.4.2.12 Nomor 12

```

1 model.fit(train_input, train_output, #Fungsi model ditambahkan fungsi
2           fit untuk mengetahui perhitungan dari train_input train_output
3           batch_size=32, #Dengan batch size 32 bit
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])

```

#### 7.4.2.13 Nomor 13

```

1 import time #Mengimport library time
2 results = [] #Membuat variabel result dengan array kosong
3 for conv2d_count in [1, 2]: #Melakukan looping dengan ketentuan
4     konvolusi 2 dimensi 1 2
5     for dense_size in [128, 256, 512, 1024, 2048]: #Menentukan ukuran
6         besaran fixcel dari data atau konvert 1 fixcel mnjadi data yang
7         berada pada codigan dibawah.
8         for dropout in [0.0, 0.25, 0.50, 0.75]: #Membuat looping
9             untuk memangkas masing-masing data dengan ketentuan 0 persen 25
10            persen 50 persen dan 75 persen.
11             model = Sequential() #Membuat variabel model Sequential
12             for i in range(conv2d_count): #Membuat looping untuk
13                 variabel i dengan jarak dari hasil konvolusi.
14                 if i == 0: #Syarat jika i samadengan bobotnya 0
15                     model.add(Conv2D(32, kernel_size=(3, 3),
16                                     activation='relu', input_shape=np.shape(train_input[0]))) #
17                     Menambahkan method add pada variabel model dengan konvolusi 2
18                     dimensi 32 bit didalamnya dan membuat kernel dengan ukuran 3 x 3
19                     dan rumus aktifasi relu dan data shape yang di hitung dari data
20                     train .
21                     else: #Jika tidak

```

```

11         model.add(Conv2D(32, kernel_size=(3, 3),
activation='relu')) #Menambahkan method add pada variabel model
dengan konvolusi 2 dimensi 32 bit dengan ukuran kernel 3 x3 dan
fungsi aktivasi relu
12         model.add(MaxPooling2D(pool_size=(2, 2))) #
Menambahkan method add pada variabel model dengan isian method
Max pooling berdimensi 2 dengan ukuran fixcel 2 x 2.
13         model.add(Flatten()) #Merubah feature gambar menjadi 1
dimensi vektor
14         model.add(Dense(dense_size, activation='tanh')) #
Menambahkan method dense untuk pemadatan data dengan ukuran dense
di tentukan dengan rumus fungsi tanh.
15         if dropout > 0.0: #Membuat ketentuan jika pemangkasan
lebih besar dari 0 persen
16             model.add(Dropout(dropout)) #Menambahkan method
dropout pada model dengan nilai dari dropout
17             model.add(Dense(num_classes, activation='softmax')) #
Menambahkan method dense dengan fungsi num classs dan rumus
softmax
18             model.compile(loss='categorical_crossentropy', optimizer=
'adam', metrics=[ accuracy]) #Mengcompile variabel model dengan
hasi loss optimasi dan akurasi matrix
19             log_dir = 'E:/KB/hasyv2/logs/conv2d_%d-dense_%d-dropout_%
.2f' % (conv2d_count, dense_size, dropout) #Melakukan log
20             tensorboard = keras.callbacks.TensorBoard(log_dir=log_dir
) # membuat variabel tensorboard dengan isian dari library keras
dan nilai dari log_dir
21
22             start = time.time() #Membuat variabel start dengan isian
dari library time menggunakan method time
23             model.fit(train_input, train_output, batch_size=32,
epochs=10,
24                         verbose=0, validation_split=0.2, callbacks=[

tensorboard]) #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 tensorboard.
25             score = model.evaluate(test_input, test_output, verbose
=2) #Membuat variabel score dengan nilai evaluasi dari model
menggunakan data tes input dan tes output
26             end = time.time() #Membuat variabel end
27             elapsed = end - start #Membuat variabel elapsed
28             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)) #Mencetak
hasil perhitungan
29             results.append((conv2d_count, dense_size, dropout, score
[0], score[1], elapsed))

```

#### 7.4.2.14 Nomor 14

```

1 model = Sequential() #Membuat variabel model dengan isian library
Sequential
2 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=np.shape(train_input[0])))#Variabel model di
tambahkan library Conv2D tigapuluhan dua bit dengan ukuran kernel 3

```

```

x 3 dan fungsi penghitungan relu yang menggunakan data
train_input
3 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
    tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x 2
    pixel
4 model.add(Conv2D(32, (3, 3), activation='relu')) #Variabel model di
    tambahkan dengan library Conv2D 32bit dengan kernel 3 x 3
5 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
    tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x 2
    pixel
6 model.add(Flatten()) #Variabel model di tambahkan library Flatten
7 model.add(Dense(128, activation='tanh')) #Variabel model di tambahkan
    library Dense dengan fungsi tanh
8 model.add(Dropout(0.5)) #Variabel model di tambahkan library dropout
    untuk memangkas data tree sebesar 50 persen
9 model.add(Dense(num_classes, activation='softmax')) #Variabel model
    di tambahkan library Dense dengan data dari num_classes dan
    fungsi softmax
10 model.compile(loss='categorical_crossentropy', optimizer='adam',
    metrics=['accuracy']) #Mengkompile data model untuk mendapatkan
        data loss akurasi dan optimasi
11 print(model.summary()) #Mencetak variabel model kemudian memunculkan
    kesimpulan berupa data total parameter, trainable paremeter dan
    bukan trainable parameter

```

#### 7.4.2.15 Nomor 15

```

1 model.fit(np.concatenate((train_input, test_input)), #Melakukan
    training yang isi datanya dari join numpy menggunakan data
    train_input test_input
2     np.concatenate((train_output, test_output)), #Kelanjutan
    data yang di gunakan pada join train_output test_output
3         batch_size=32, epochs=10, verbose=2) #Menggunakan ukuran 32
        bit dan epoch 10

```

#### 7.4.2.16 Nomor 16

```

1 model.save("mathsymbols.model") #Menyimpan model atau mengeksport
    model yang telah di jalan tadi

```

#### 7.4.2.17 Nomor 17

```

1 np.save('classes.npy', labelencoder.classes_) #Menyimpan label
    encoder dengan nama classes.npy

```

#### 7.4.2.18 Nomor 18

```

1 import keras.models #Mengimpport library keras model
2 model2 = keras.models.load_model("mathsymbols.model") #Membuat
    variabel model2 untuk meload model yang telah di simpan tadi
3 print(model2.summary()) #Mencetak hasil model2

```

### 7.4.2.19 Nomor 19

```

1 label_encoder2 = LabelEncoder() # membuat variabel label encoder ke 2
2     dengan isian fungsi label encoder.
3 label_encoder2.classes_ = np.load('classes.npy') #Menambahkan method
4     classes dengan data classess yang di eksport tadi
5 def predict(img_path): #Membuat fungsi predict dengan path img
6     newimg = keras.preprocessing.image.img_to_array(pil_image.open(
7         img_path)) #Membuat variabel newimg dengan membuat immage menjadi
8     array dan membuka data berdasarkan img path
9     newimg /= 255.0 #Membagi data yang terdapat pada variabel newimg
10    sebanyak 255
11
12    # do the prediction
13    prediction = model2.predict(newimg.reshape(1, 32, 32, 3)) #
14        Membuat variabel preditvion dengan isian variabel model2
15        menggunakan fungsi predic dengan syarat variabel newimg dengan
16        data reshape
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
21        )]) #Membuat variabel inverted dengan label encoder2 dan
22        menggunakan argmax untuk mencari skor keluaran tertinggi
23    print("Prediction: %s, confidence: %.2f" % (inverted[0], np.max(
24        prediction))) #Mencetak prediksi gambar dan confidence dari
25        gambar.

```

### 7.4.2.20 Nomor 20

```

1 predict("HASYv2/hasy-data/v2-00010.png") #Mencari prediksi
2     menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2
3     /hasy-data/v2-00010.png
4 predict("HASYv2/hasy-data/v2-00500.png") #Mencari prediksi
5     menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2
6     /hasy-data/v2-00500.png
7 predict("HASYv2/hasy-data/v2-00700.png") #Mencari prediksi
8     menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2
9     /hasy-data/v2-00700.png

```

## 7.4.3 Penanganan Error

### 7.4.3.1 Error

- NameError

`NameError: name 'np' is not defined`

Gambar 7.79 NameError

### 7.4.3.2 Solusi Error

- NameError

Pastikan sudah diimport atau cek typo

### 7.4.4 Bukti Tidak Plagiat



Gambar 7.80 Bukti tidak plagiat

### 7.4.5 Link Youtube

<https://youtu.be/AHJOIZJYd9I>

## 7.5 Muhammad Reza Syachrani - 1174084

### 7.5.1 Teori

1. Jelaskan kenapa file teks harus di lakukan tokenizer. dilengkapi dengan ilustrasi atau gambar  
untuk mempermudah mesin dalam mengolah file teks yang dimana bentuk teks tersebut di konversi menjadi urutan integer kata atau vector binary. ilustrasi tokenizer sebagai berikut, saya memiliki teks yaitu "saya makan nasi goreng" dan setelah dilakukan tokenizer berubah menjadi 'saya': 1, 'makan': 2, 'nasi': 3, 'goreng': 4
2. Jelaskan konsep dasar K Fold Cross Validation pada dataset komentar Youtube. pada codingan tersebut terdapat variabel kfolds yang berisi fungsi StratifiedKFold yang memiliki parameter n\_splits=5 yang berarti melakukan pengulangan terhadap data masing-masing lima kali dengan atribut class sebagai acuan pengolahan datanya kemudian akan di hasilkan akurasi dari pengulangan data tersebut sebesar sekian persen tergantung datanya.



```

import numpy as np
from sklearn.model_selection import train_test_split
x, y = np.arange(25).reshape((5, 5)), range(5)
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2, random_state=5252)
x_train, x_test

(array([[20, 21, 22, 23, 24],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [ 0,  1,  2,  3,  4]]), array([[5, 6, 7, 8, 9]))

```

**Gambar 7.82** Teori 3

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.

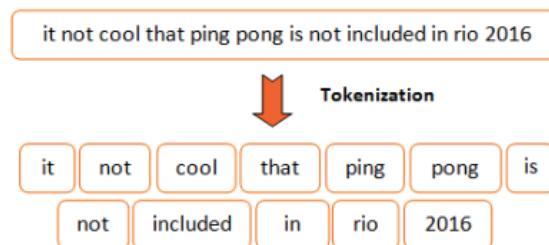
kode program train content tersebut adalah membaca isian kolom pada field yang bernama CONTENT sebagai data training sedangkan kode program test content membaca isi kolom pada field yang bernama CONTENT sebagai data testing.

Index	CONTENT
0	i love this so much. AND als...
1	http://www.billboard...
2	Hey guys! Please join m...
3	http://psnboss.com/?...
4	Hey everyone. Watch this tr...
5	check out my rapping hope ...

**Gambar 7.83** Teori 4

5. Jelaskan apa maksud dari fungsi `tokenizer = Tokenizer(num_words=2000)` dan `tokenizer.fit_on_texts(train_content)`, dilengkapi dengan ilustrasi atau gambar. fungsi `tokenizer = Tokenizer(num_words=2000)` untuk membaca kalimat menjadi token/indeks sebanyak 2000 kata dan fungsi `fit_on_texts(train_konten)` un-

tuk membuat membaca data token/indeks teks yang telah di masukan kedalam data train\_konten.



**Gambar 7.84** Teori 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 variabel `d_train_inputs` melakukan tokenizer dari bentuk teks menjadi bentuk matrix yang berurutan dari data `train_content` dengan mode tf idf begitu juga dengan `d_test_inputs` untuk data test.

Fungsi `texts_to_matrix()` dapat digunakan untuk membuat satu vektor per dokumen, menyediakan skema standar teks encoding bag-of-words melalui mode-mode argumen dari fungsi, yaitu :

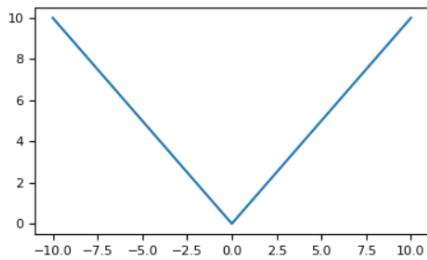
- `binary` : apakah setiap kata ada dalam dokumen atau tidak (default)
- `count` : jumlah setiap kata dalam dokumen
- `tfidf` : skor untuk Text Frequency-Inverse DocumentFrequency (TF-IDF) untuk setiap kata dalam dokumen
- `freq` : frekuensi setiap kata sebagai rasio kata dalam setiap dokumen.

**Gambar 7.85** Teori 6

- 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 tersebut digunakan untuk membagi matrix tfidf untuk menentukan maksimum array yang kemudian hasilnya tersebut dimasukan ke dalam variabel `d_train_input` dan `d_test_input` dengan metode absolute sehingga tanpa adanya bilangan negatif dan nol.

```
>>> import matplotlib.pyplot as plt

>>> x = np.linspace(start=-10, stop=10, num=101)
>>> plt.plot(x, np.absolute(x))
>>> plt.show()
```



**Gambar 7.86** Teori 7

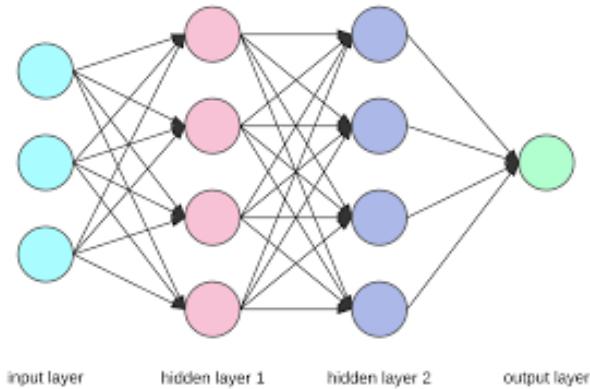
8. Jelaskan apa maksud fungsi dari `d_train_outputs = np_utils.to_categorical(d['CLASS'].iloc` dan `d_test_outputs = np_utils.to_categorical(d['CLASS'].iloc[test_idx])` dalam kode program, dilengkapi dengan ilustrasi atau gambar.  
fungsi untuk melakukan one-hot encoding merubah nilai vektor dengan bentuk integer yang ada pada atribut class menjadi bentuk matrix biner untuk atribut CLASS sehingga hanya ada dua pilihan 0 atau 1.

```
# Consider an array of 5 labels out of a set of 3 classes {0, 1, 2}:
> labels
array([0, 2, 1, 2, 0])
# `to_categorical` converts this into a matrix with as many
# columns as there are classes. The number of rows
# stays the same.
> to_categorical(labels)
array([[ 1.,  0.,  0.],
       [ 0.,  0.,  1.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.],
       [ 1.,  0.,  0.]], dtype=float32)
```

**Gambar 7.87** Teori 8

9. Jelaskan apa maksud dari fungsi di listing 7.2. Gambarkan ilustrasi Neural Network nya dari model kode tersebut.  
fungsi kode tersebut untuk melakukan permodelan sequential yang digunakan untuk mencari data dengan menerima parameter atau argumen kunci. kemudian di tambahkan metod add dengan danse sebanyak 512 neuron inputan dengan input shape 2000 vektor yang sudah dinormalisasi. kemudian di tambahkan lagi fungsi aktivasi dengan fungsi "relu". dan dilakukan overfitting atau pemotongan

bobot sebesar 0.5 atau 50 persen. Lalu pada layer output terdapat 2 neuron output. Kemudian outputan tersebut diaktivasi menggunakan fungsi softmax.



**Gambar 7.88** Teori 9

10. Jelaskan apa maksud dari fungsi di listing 7.3 dengan parameter tersebut.  
fungsi kode yang melakukan compile pada model dengan menggunakan beberapa parameter seperti loss yang akan mengembalikan fungsi nilai loss yang diambil, sedangkan fungsi adamax digunakan untuk mengetahui nilai lossnya, dan matrices = akurasi merupakan akurasi dari nilai matriknya.
11. Jelaskan apa itu Deep Learning  
Merupakan metode pada machine learning yang menggunakan artificial neural networks. algoritma permodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam.
12. Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning  
Deep Neural Network merupakan algoritma jaringan syaraf yang akan melakukan pembobotan terhadap data yang sudah ada sebagai acuan untuk data inputan selanjutnya. kemudian terdiri atas beberapa lapisan/layer atau hiden layer. perbedaan antara deep learning dan Deep Neural Network yaitu Deep Neural Network merupakan algoritma sedangkan deep learning yang menggunakan Deep Neural Network tersebut.
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  
Karena NPM saya 1164084 dan hasil dari  $(NPM \text{ mod } 3)+1 = 2$ , maka saya menggunakan stride 2 dengan ketentuan max pooling 2x2.



Gambar 7.89 Teori 13

### 7.5.2 Praktek

1. Jelaskan kode program pada blok In[1]

```

1 # In [1]: import lib
2 import csv
3 #Mengimport library csv untuk mengimport file ekstensi .csv
4 from PIL import Image as pil_image
5 #Mengimport library Image dari PIL sebagai pil_image yang
   digunakan untuk mengolah data gambar
6 import keras.preprocessing.image
7 #Mengimport library keras dengan metode preprocessing.image yang
   digunakan untuk membuat neural network

```

```

In [1]: import csv
...: #Mengimport library csv untuk mengimport file ekstensi .csv
...: from PIL import Image as pil_image
...: #Mengimport library Image dari PIL sebagai pil_image yang digunakan untuk
mengolah data gambar
...: import keras.preprocessing.image
...: #Mengimport library keras dengan metode preprocessing.image yang digunakan
untuk membuat neural network
Using TensorFlow backend.

```

Gambar 7.90 Hasil Praktek 1

2. Jelaskan kode program pada blok In[2].

```

1 # In [2]: load all images (as numpy arrays) and save their classes
2
3 imgs = []
4 #Membuat variabel imgs dengan variabel kosong
5 classes = []
6 #Membuat variabel imgs dengan variabel kosong
7 with open('D:/New folder/KB3C/src/1174084/7/HASYv2/hasy-data-
   labels.csv') as csvfile:
8     #membuka file csv pada folder HASYv2 yaitu hasy-data-labels.csv
9     #sebagai csvfile
10    csvreader = csv.reader(csvfile)
11    #membuat variabel csvreader yang berisikan metode reader dari
      library csv yang membaca csvfile.
12    i = 0
13    #Membuat varianel i yang berisikan 0

```

```

13     for row in csvreader:
14         #Membuat pengulangan pada variabel csvreader
15         if i > 0:
16             #Dengan ketentuan jika i lebih besar dari 0
17             img = keras.preprocessing.image.img_to_array(
18                 pil_image.open("D:/New folder/KB3C/src/1174084/7/HASYv2/" +
19                 row[0]))
20                 #Membuat variabel img yang berisikan fungsi keras
21                 untuk aktivasi neural network yang membaca data pada folder
22                 HASYv2 yang dibuka dengan row berparameter 0.
23                 # neuron activation functions behave best when input
24                 values are between 0.0 and 1.0 (or -1.0 and 1.0),
25                 # so we rescale each pixel value to be in the range
26                 0.0 to 1.0 instead of 0–255
27                 img /= 255.0
28                 #Membagi data yang berada pada variabel img dengan
29                 255.0
30                 imgs.append((row[0], row[2], img))
31                 #Menambahkan nilai baru pada imgs yaitu row 0,row 2
32                 dilanjutkan dengan variabel img.
33                 classes.append(row[2])
34                 # menambahkan nilai pada row ke 2 pada variabel
35                 classes
36                 i += 1
37                 #Menambahkan nilai 1 pada variabel i

```

Name	Type	Size	Value
classes	list	168233	['A', 'A', ...]
i	int	1	168234
img	float32 (32, 32, 3)		[[[1. 1. 1.]  [1. 1. 1.]  [...]]]
imgs	list	168233	[('hasy-data/v2-00000.png', 'A', Numpy array), ('hasy-data/v2-00001.png', 'A', Numpy array), ...]
row	list	4	['hasy-data/v2-168232.png', '1400', '\guillemotleft', '16925']

Gambar 7.91 Hasil Praktek 2

### 3. Jelaskan kode program pada blok In[3]

```

1 # In[3]: shuffle the data , split into 80% train , 20% test
2
3 import random
4 #Mengimport library random
5 random.shuffle(imgs)
6 #Melakukan shuffle pada variabel imgs
7 split_idx = int(0.8*len(imgs))
8 #Membuat variabel split_idx yang diisi dengan nilai integer dari
8     perkalian 80% dengan jumlah dari variabel imgs
9 train = imgs[:split_idx]
10 #Membuat variabel train yang diisi dengan dengan pemecahan index
10     awal pada data variabel split_idx
11 test = imgs[split_idx:]
12 #Membuat variabel test yang diisi dengan pemecahan index akhir
12     pada data variabel split_idx

```

split_idx	int	1	134586
test	list	33647	[('hasy-data/v2-135728.png', '\vdots', Numpy array), ('hasy-data/v2-84 ...
train	list	134586	[('hasy-data/v2-165563.png', '\sun', Numpy array), ('hasy-data/v2-0377 ...

Gambar 7.92 Hasil Praktek 3

## 4. Jelaskan kode program pada blok In[4]

```

1 # In [4]:
2
3 import numpy as np
4 #Mengimport library numpy sebagai np
5 train_input = np.asarray(list(map(lambda row: row[2], train)))
6 #Membuat variabel train_input dengan np method asarray yang mana
    membuat array dengan fungsi list yang didalamnya diterapkan
    fungsi map untuk mengembalikan interator
7 #dan menggunakan lamba untuk mengecilkan fungsi dari objek yang
    berada pada row 2 dari data train
8 test_input = np.asarray(list(map(lambda row: row[2], test)))
9 #Membuat variabel test_input dengan isi np method asarray yang
    mana membuat array dengan fungsi list yang didalamnya
    diterapkan fungsi map untuk mengembalikan interator
10 #dan menggunakan lamba untuk mengecilkan fungsi dari objek yang
    berada pada row 2 dari data test
11 train_output = np.asarray(list(map(lambda row: row[1], train)))
12 #Membuat variabel train_output dengan np method asarray yang mana
    membuat array dengan fungsi list yang didalamnya diterapkan
    fungsi map untuk mengembalikan interator
13 #dan menggunakan lamba untuk mengecilkan fungsi dari objek yang
    berada pada row 1 dari data train
14 test_output = np.asarray(list(map(lambda row: row[1], test)))
15 #Membuat variabel test_output dengan np method asarray yang mana
    membuat array dengan fungsi list yang didalamnya diterapkan
    fungsi map untuk mengembalikan interator
16 #dan menggunakan lamba untuk mengecilkan fungsi dari objek yang
    berada pada row 1 dari data train

```

test_input	float32	(33647, 32, 32, 3)	[[[1. 1. 1.] [1. 1. 1.]
test_output	str608	(33647,)	ndarray object of numpy module
train	list	134586	[('hasy-data/v2-165563.png', '\sun', Numpy array), ('hasy-data/v2-0377 ...
train_input	float32	(134586, 32, 32, 3)	[[[1. 1. 1.] [1. 1. 1.]
train_output	str608	(134586,)	ndarray object of numpy module

Gambar 7.93 Hasil Praktek 4

## 5. Jelaskan kode program pada blok In[5]

```

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

```

```

3 #Mengimport library LabelEncoder dari sklearn.preprocessing yang
   digunakan untuk mengonversi jenis data teks kategori menjadi
   data numerik
4 from sklearn.preprocessing import OneHotEncoder
5 #Mengimport library OneHotEncoder dari sklearn.preprocessing

```

```

In [7]: from sklearn.preprocessing import LabelEncoder
...: #Mengimport library LabelEncoder dari sklearn.preprocessing yang digunakan
...: untuk mengonversi jenis data teks kategori menjadi data numerik
...: from sklearn.preprocessing import OneHotEncoder
...: #Mengimport library OneHotEncoder dari sklearn.preprocessing

```

**Gambar 7.94** Hasil Praktek 5

## 6. Jelaskan kode program pada blok In[6]

```

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

```

integer_encoded	int64	(168233,)	[ 15 15 15 ... 143 143 143]
-----------------	-------	-----------	-----------------------------

**Gambar 7.95** Hasil Praktek 6

## 7. Jelaskan kode program pada blok In[7]

```

1 # In [7]: then convert integers into one-hot encoding
2 onehot_encoder = OneHotEncoder(sparse=False)
3 #Membuat variabel onehot_encoder dengan isi fungsi OneHotEncoder
   parameter sparse=false
4 integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
5 #Membuat variabel integer_encoder dengan isi fungsi
   integer_encoded yang telah di convert pada fungsi sebelumnya
6 onehot_encoder.fit(integer_encoded)
7 #Onehotencoding melakukan fitting pada variabel integer_encoded

```

integer_encoded	int64	(168233, 1)	[[ 15 ] [ 15 ]]
-----------------	-------	-------------	--------------------

**Gambar 7.96** Hasil Praktek 7

## 8. Jelaskan kode program pada blok In[8]

```

1 # In [8]: convert train and test output to one-hot
2 train_output_int = label_encoder.transform(train_output)

```

```

3 #Membuat variabel train_output_int dengan isi hasil konversi data
   train_output menggunakan label_encoder
4 train_output = onehot_encoder.transform(train_output_int.reshape(
   len(train_output_int), 1))
5 #Mengkonversi data train_output_int menggunakan fungsi
   onehot_encoder
6 test_output_int = label_encoder.transform(test_output)
7 #Membuat variabel test_output_int dengan isi hasil konversi data
   test_output menggunakan label_encoder
8 test_output = onehot_encoder.transform(test_output_int.reshape(
   len(test_output_int), 1))
9 #Mengkonversi data test_output_int menggunakan fungsi
   onehot_encoder
10 num_classes = len(label_encoder.classes_)
11 #Membuat variabel num_classes dengan isi jumlah class pada
   label_encoder
12 print("Number of classes: %d" % num_classes)
13 #Menampilkan hasil dari variabel num_classes

```

---

```

In [10]: train_output_int = label_encoder.transform(train_output)
...: #Membuat variabel train_output_int dengan isi hasil konversi data
train_output menggunakan label_encoder
...: train_output =
onehot_encoder.transform(train_output_int.reshape(len(train_output_int), 1))
...: #Mengkonversi data train_output_int menggunakan fungsi onehot_encoder
...: test_output_int = label_encoder.transform(test_output)
...: #Membuat variabel test_output_int dengan isi hasil konversi data test_output
menggunakan label_encoder
...: test_output =
onehot_encoder.transform(test_output_int.reshape(len(test_output_int), 1))
...: #Mengkonversi data test_output_int menggunakan fungsi onehot_encoder
...: num_classes = len(label_encoder.classes_)
...: #Membuat variabel num_classes dengan isi jumlah class pada label_encoder
...: print("Number of classes: %d" % num_classes)
...: #Menampilkan hasil dari variabel num_classes
Number of classes: 369

```

**Gambar 7.97** Hasil Praktek 8

## 9. Jelaskan kode program pada blok In[9]

```

1 # In [9]: import sequential
2 import tensorflow as tf
3 from keras.models import Sequential
4 #Mengimport Sequential dari library keras
5 from keras.layers import Dense, Dropout, Flatten
6 #Mengimport Dense, Dropout, Flatten dari Library keras
7 from keras.layers import Conv2D, MaxPooling2D
8 #Mengimport Conv2D dan MaxPoolinf2D dari library Keras

```

```
In [11]: from keras.models import Sequential
...: #Mengimport Sequential dari library keras
...: from keras.layers import Dense, Dropout, Flatten
...: #Mengimport Dense, Dropout, Flatten dari Library keras
...: from keras.layers import Conv2D, MaxPooling2D
...: #Mengimport Conv2D dan MaxPooling2D dari library Keras
```

**Gambar 7.98** Hasil Praktek 9

10. Jelaskan kode program pada blok In[10]

```
1 # In[10]: desain jaringan
2 model = tf.keras.Sequential()
3 #Membuat variabel model dengan isi fungsi Sequential
4 model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
5     activation='relu',
6     input_shape=np.shape(train_input[0])))
7 #Variabel model ditambahkan fungsi Conv2d dengan paramater 32
8 #filter dengan kernel berukuran 3x3
9 #dengan algoritam activation relu mengunakan data train_input
10 #mulai dari baris nol.
11 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
12 #Variabel madel ditambahkan fungsi MaxPooling2D dengan ketentuan
13 #ukuran 2x2
14 model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu'))
15 #Variabel model ditambahkan fungsi Conv2D dengan 32 filter dengan
16 #konvolusi berukuran 3x3, menggunakan algoritam activation
17 #relu
18 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
19 #Variabel madel ditambahkan fungsi MaxPooling2D dengan ketentuan
20 #ukuran 2x2
21 model.add(tf.keras.layers.Flatten())
22 #Variabel model di tambahkan fungsi Flatten
23 model.add(tf.keras.layers.Dense(1024, activation='tanh'))
24 #Variabel model ditambahakan fungsi dense dengan 1024 neuron , dan
25 #menggunakan algoritma tanh untuk activation
26 model.add(tf.keras.layers.Dropout(0.5))
27 #Variabel model di tambahkan fungsi Dropout sebesar 50% untuk
28 #mencegah terjadinya overfitting
29 model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
30 #Variabel model ditambahkan fungsi Dense dengan parameter
31 #variabel num_classes menggunakan activation softmax
32 model.compile(loss='categorical_crossentropy', optimizer='adam',
33                 metrics=['accuracy'])
34 #Variabel model di compile dengan parameter loss , matrik , dan
35 #optimasi
36 print(model.summary())
37 #Menampilkan model yang telah dibuat.
```

```
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, 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 7.99** Hasil Praktek 10

## 11. Jelaskan kode program pada blok In[11]

```
1 # In [11]: import sequential
2
3 import keras.callbacks
4 #mengimport library keras callbacks
5 tensorboard = keras.callbacks.TensorBoard(log_dir='./\\logs\\mnist
   -style')
6 #Membuat variabel tensorboard dengan isi fungsi TensorBoard yang
   ada pada library keras.callback dengan parameter director log
   './logs/mnist-style'
```

```
In [13]: import keras.callbacks
...: #mengimport Library keras callbacks
...: tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-style')
...: #Membuat variabel tensorboard dengan isi fungsi TensorBoard yang ada pada
library keras.callback dengan parameter director log './Logs/mnist-style'
```

**Gambar 7.100** Hasil Praktek 11

## 12. Jelaskan kode program pada blok In[12]

```
1 # In[12]: 5menit kali 10 epoch = 50 menit
2 model.fit(train_input, train_output,
3 #Melakukan fitting pada model dengan paramater train_input ,
   train_output
4           batch_size=32,
5             #dengan menggunakan batch_size sebesar 32,
6             epochs=10,
7               #epoch=10 yang berarti terjadi perulangan sebanyak
10 kali
8             verbose=2,
9               #untuk menghasilkan informasi logging dari data yang
ditentukan dengan nilai 2
```

```

10         validation_split=0.2,
11         #melakukan pemecahan nilai sebesar 0.2 / 20% dari
12         perhitungan validasi
13         callbacks=[tensorboard])
14         #mengeksekusi tensorboard dimana digunakan untuk
15         visualisasikan parameter training , metrik , hiperparameter
16         pada nilai/data yang diproses
17
18 score = model.evaluate(test_input , test_output , verbose=2)
19 #Membuat variabel score dengan isi fungsi evulate dari model
20 #dengan paramater test_input , test_output dan verbose=2
21 #untuk memprediksi output dan input
22 print('Test loss:', score[0])
23 #Menampilkan score optimasi dengan ketentuan nilai parameter 0
24 print('Test accuracy:', score[1])
25 #Mencetak score akurasi dengan ketentuan nilai parameter 1

```

```

Train on 107668 samples, validate on 26918 samples
Epoch 1/10
107668/107668 - 74s - loss: 1.5624 - accuracy: 0.6235 - val_loss: 1.0111 - val_accuracy: 0.7197
Epoch 2/10
107668/107668 - 75s - loss: 0.9902 - accuracy: 0.7270 - val_loss: 0.9148 - val_accuracy: 0.7464
Epoch 3/10
107668/107668 - 80s - loss: 0.8742 - accuracy: 0.7509 - val_loss: 0.8888 - val_accuracy: 0.7557
Epoch 4/10
107668/107668 - 80s - loss: 0.8059 - accuracy: 0.7650 - val_loss: 0.8823 - val_accuracy: 0.7541
Epoch 5/10
107668/107668 - 82s - loss: 0.7606 - accuracy: 0.7744 - val_loss: 0.8537 - val_accuracy: 0.7570
Epoch 6/10
107668/107668 - 81s - loss: 0.7213 - accuracy: 0.7820 - val_loss: 0.8514 - val_accuracy: 0.7582
Epoch 7/10
107668/107668 - 81s - loss: 0.6873 - accuracy: 0.7899 - val_loss: 0.8616 - val_accuracy: 0.7651
Epoch 8/10
107668/107668 - 81s - loss: 0.6641 - accuracy: 0.7929 - val_loss: 0.8498 - val_accuracy: 0.7615
Epoch 9/10
107668/107668 - 80s - loss: 0.6419 - accuracy: 0.7986 - val_loss: 0.8788 - val_accuracy: 0.7571
Epoch 10/10
107668/107668 - 78s - loss: 0.6204 - accuracy: 0.8029 - val_loss: 0.8908 - val_accuracy: 0.7584
33647/33647 - 6s - loss: 0.8712 - accuracy: 0.7624
Test loss: 0.871180891701912
Test accuracy: 0.76238596

```

**Gambar 7.101** Hasil Praktek 12

### 13. Jelaskan kode program pada blok In[13]

```

1 # In[13]:try various model configurations and parameters to find
2     the best
3
4 import time
5 #import library time
6
7 results = []
8 #Membuat variabel result dengan isi array kosong
9 for conv2d_count in [1, 2]:
10    #Melakukan looping menggunakan convd2d_count dengan ketentuan
11        konvolusi 2 dimensi yaitu 1, 2
12        for dense_size in [128, 256, 512, 1024, 2048]:
13            #Melakukan looping menggunakan ukuran dari densenya yaitu
14            123, 256, 512, 1024, 2048.
15            for dropout in [0.0, 0.25, 0.50, 0.75]:
16                #Melakukan looping menggunakan dropout dengan ketentuan
0%, 25%, 50%, 75% untuk memangkas data.
17                model = tf.keras.Sequential()
18                #Membuat variabel model dengan isi fungsi sequential.
19                for i in range(conv2d_count):
20

```

```

#Melakukan looping menggunakan i dengan jarak hasil
konvolusi
    if i == 0:
        #jika nilai i sama dengan 0
        model.add(tf.keras.layers.Conv2D(32,
kernel_size=(3, 3), activation='relu', input_shape=np.shape(
train_input[0])))
            #Variabel model ditambahkan fungsi Conv2d
dengan paramater 32 filter dengan karnel berukuran 3x3
            #dengan algoritam activation relu menggunakan
data train_input mulai dari baris nol.
    else:
        #jika tidak
        model.add(tf.keras.layers.Conv2D(32,
kernel_size=(3, 3), activation='relu'))
            #Variabel model akan ditambahkan fungsi
Conv2d dengan paramater 32 filter dengan karnel berukuran 3x3
            #dengan algoritam activation relu tanpa ada
parameter input_shape .
        model.add(tf.keras.layers.MaxPooling2D(pool_size
=(2, 2)))
            #Variabel madel ditambahkan fungsi MaxPooling2D
dengan ketentuan ukuran 2x2
        model.add(tf.keras.layers.Flatten())
            #Variabel model di tambahkan fungsi Flatten
        model.add(tf.keras.layers.Dense(dense_size,
activation='tanh'))
            #Variabel model ditambahakan fungsi dense dengan
jumlah dense yang digunakan , dan menggunakan algoritma tanh
untuk activation
        if dropout > 0.0:
            #jika nilai dari dropout lebih besar dari 0.0
            model.add(tf.keras.layers.Dropout(dropout))
                #Variabel model di tambahkan fungsi Dropout
sebesar dropout yang digunakan untuk mencegah terjadinya
overfitting
            model.add(tf.keras.layers.Dense(num_classes ,
activation='softmax'))
                #Variabel model ditambahkan fungsi Dense dengan
parameter variabel num_classes menggunakan activation softmax
            model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
                #Variabel model di compile dengan parameter loss ,
matrik , dan optimasi
                log_dir = '.\logs\conv2d_%d-dense_%d-dropout_.%2f'
% (conv2d_count, dense_size, dropout)
                    #Melakukan log pada dir
                    tensorboard = keras.callbacks.TensorBoard(log_dir=
log_dir)
                        #Mengisi variabel tensorboard dengan isian dari
library keras dan nilai dari log dir
                    start = time.time()
                    #Membuat variabel start dengan isi fungsi time dari
library time

```

```

49         model.fit(train_input, train_output, batch_size=32,
50                     epochs=10,
51                     verbose=0, validation_split=0.2, callbacks
52                     =[tensorboard])
53             #Melakukan fitting pada model dengan parameter
54             test_input, test_output, batch_size, epochs, verbose,
55             validation_split dan callbacks
56             score = model.evaluate(test_input, test_output,
57             verbose=2)
58             #Membuat variabel score dengan nilai evaluasi dari
59             model menggunakan data tes input dan tes output dengan
60             verbose adalah 2
61             end = time.time()
62             #Membuat variabel end dengan isi fungsi time dari
63             library time
64             elapsed = end - start
65             #Membuat variabel elapse yang diisi dengan nilai
66             hasil waktu end dikurangi start
67             print("Conv2D count: %d, Dense size: %d, Dropout: %.2
f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count
68             , dense_size, dropout, score[0], score[1], elapsed))
69             results.append((conv2d_count, dense_size, dropout,
70             score[0], score[1], elapsed))
71             #Menampilkan hasil perhitungan.

```

```

33647/33647  4s - loss: 1.1405 - accuracy: 0.7386
Conv2D count: 1, Dense size: 128, Dropout: 0.00 - Loss: 1.14, Accuracy: 0.74, Time: 462 sec
33647/33647  5s - loss: 0.9020 - accuracy: 0.7652
Conv2D count: 1, Dense size: 128, Dropout: 0.25 - Loss: 0.92, Accuracy: 0.77, Time: 488 sec
33647/33647  6s - loss: 0.8037 - accuracy: 0.7775
Conv2D count: 1, Dense size: 128, Dropout: 0.50 - Loss: 0.80, Accuracy: 0.78, Time: 550 sec
WARNING:tensorflow:Large dropout rate: 0.75 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate
instead of keep_prob. Please ensure that this is intended.
WARNING:tensorflow:Large dropout rate: 0.75 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate
instead of keep_prob. Please ensure that this is intended.
WARNING:tensorflow:Large dropout rate: 0.75 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate
instead of keep_prob. Please ensure that this is intended.
33647/33647  5s - loss: 0.8012 - accuracy: 0.7706
Conv2D count: 1, Dense size: 128, Dropout: 0.75 - Loss: 0.80, Accuracy: 0.77, Time: 564 sec
33647/33647  6s - loss: 1.3329 - accuracy: 0.7401
Conv2D count: 1, Dense size: 256, Dropout: 0.00 - Loss: 1.33, Accuracy: 0.74, Time: 695 sec
33647/33647  6s - loss: 1.1072 - accuracy: 0.7603
Conv2D count: 1, Dense size: 256, Dropout: 0.25 - Loss: 1.11, Accuracy: 0.76, Time: 738 sec
33647/33647  5s - loss: 0.9145 - accuracy: 0.7757
Conv2D count: 1, Dense size: 256, Dropout: 0.50 - Loss: 0.91, Accuracy: 0.78, Time: 731 sec
WARNING:tensorflow:Large dropout rate: 0.75 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate

```

**Gambar 7.102** Hasil Praktek 13

#### 14. Jelaskan kode program pada blok In[14]

```

1 # In[14]: rebuild/retrain a model with the best parameters (from
2   the search) and use all data
3 model = tf.keras.Sequential()
4 #Membuat variabel model dengan isi fungsi Sequential
5 model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
6   activation='relu', input_shape=np.shape(train_input[0])))
5 #Variabel model ditambahkan fungsi Conv2d dengan paramater 32
7   filter dengan karnel berukuran 3x3
8 #dengan algoritam activation relu mengunakan data train_input
9   mulai dari baris nol.
7 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
8 #Variabel model ditambahkan fungsi MaxPooling2D dengan ketentuan
9   ukuran 2x2
9 model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu'))

```

```

10 #Variabel model ditambahkan fungsi Conv2D dengan 32 filter dengan
   konvolusi berukuran 3x3, menggunakan algoritam activation
   relu
11 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
12 #Variabel model ditambahkan fungsi MaxPooling2D dengan ketentuan
   ukuran 2x2
13 model.add(tf.keras.layers.Flatten())
14 #Variabel model di tambahkan fungsi Flatten
15 model.add(tf.keras.layers.Dense(128, activation='tanh'))
16 #Variabel model ditambahakan fungsi dense dengan 128 neuron , dan
   menggunakan algoritma tanh untuk activation
17 model.add(tf.keras.layers.Dropout(0.5))
18 #Variabel model di tambahkan fungsi Dropout sebesar 50% untuk
   mencegah terjadinya overfitting
19 model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
20 #Variabel model ditambahkan fungsi Dense dengan parameter
   variabel num_classes menggunakan activation softmax
21 model.compile(loss='categorical_crossentropy', optimizer='adam',
   metrics=['accuracy'])
22 #Variabel model di compile dengan parameter loss , matrik , dan
   optimasi
23 print(model.summary())
24 #Menampilkan ringkasan model yang telah dibuat.

```

Model: "sequential_19"		
Layer (type)	Output Shape	Param #
conv2d_24 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_23 (MaxPooling)	(None, 15, 15, 32)	0
conv2d_25 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_24 (MaxPooling)	(None, 6, 6, 32)	0
flatten_19 (Flatten)	(None, 1152)	0
dense_36 (Dense)	(None, 128)	147584
dropout_13 (Dropout)	(None, 128)	0
dense_37 (Dense)	(None, 369)	47601
<hr/>		
Total params: 205,329		
Trainable params: 205,329		
Non-trainable params: 0		
<hr/>		
None		

Gambar 7.103 Hasil Praktek 14

## 15. Jelaskan kode program pada 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 #Melakukan fitting pada model melakukan join numpy menggunakan
   data train_input test_input
               np.concatenate((train_output, test_output)),
4

```

```

5         #kemudian join numpy menggunakan data train_output
6 test_output
7         batch_size=32, epochs=10, verbose=2)
#menggunakan batch ukuran 32, epochs 10 dan verbose 2

```

```

Train on 168233 samples
Epoch 1/10
168233/168233 - 69s - loss: 1.7824 - accuracy: 0.5859
Epoch 2/10
168233/168233 - 69s - loss: 1.0818 - accuracy: 0.7077
Epoch 3/10
168233/168233 - 71s - loss: 0.9625 - accuracy: 0.7305
Epoch 4/10
168233/168233 - 70s - loss: 0.9041 - accuracy: 0.7435
Epoch 5/10
168233/168233 - 69s - loss: 0.8627 - accuracy: 0.7521
Epoch 6/10
168233/168233 - 71s - loss: 0.8364 - accuracy: 0.7582
Epoch 7/10
168233/168233 - 70s - loss: 0.8129 - accuracy: 0.7635
Epoch 8/10
168233/168233 - 71s - loss: 0.7948 - accuracy: 0.7666
Epoch 9/10
168233/168233 - 70s - loss: 0.7816 - accuracy: 0.7687
Epoch 10/10
168233/168233 - 70s - loss: 0.7676 - accuracy: 0.7722

```

**Gambar 7.104** Hasil Praktek 15

16. Jelaskan kode program pada blok In[16]

```

1 # In[16]: save the trained model
2 model.save("mathsymbols.model")
3 #Menyimpan model dengan nama mathsymbols.model

```

```

In [22]: runcell('/16]: save the trained model', 'D:/New folder/KB3C/src/1174084/7/1174084.py')
INFO:tensorflow:Assets written to: mathsymbols.model\assets

```

**Gambar 7.105** Hasil Praktek 16

17. Jelaskan kode program pada blok In[17]

```

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

```

```

In [23]: runcell('/17]: save Label encoder (to reverse one-hot encoding)', 'D:/New folder/KB3C/src/
1174084/7/1174084.py')

```

**Gambar 7.106** Hasil Praktek 17

18. Jelaskan kode program pada blok In[18]

```

1 # In[18]: load the pre-trained model and predict the math symbol
2 # for an arbitrary image;
3 # the code below could be placed in a separate file
4
5 import keras.models
6 #Mengimport library keras.models
7 model2 = keras.models.load_model("mathsymbols.model")
8 #Membuat variabel model2 dengan isi hasil load model dari
9 #mathsymbols.model
10 print(model2.summary())
11 #Menampilkan ringkasan dari model

```

Layer (type)	Output Shape	Param #
conv2d_24 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_23 (MaxPooling)	(None, 15, 15, 32)	0
conv2d_25 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_24 (MaxPooling)	(None, 6, 6, 32)	0
flatten_19 (Flatten)	(None, 1152)	0
dense_36 (Dense)	(None, 128)	147584
dropout_13 (Dropout)	(None, 128)	0
dense_37 (Dense)	(None, 369)	47601
<hr/>		
Total params: 205,329		
Trainable params: 205,329		
Non-trainable params: 0		
<hr/>		
None		

Gambar 7.107 Hasil Praktek 18

### 19. Jelaskan kode program pada blok In[19]

```

1 # In[19]: restore the class name to integer encoder
2 label_encoder2 = LabelEncoder()
3 #Membuat variabel label_encoder ke 2 dengan isi fungsi
4 #Label_Encoder
5 label_encoder2.classes_ = np.load('classes.npy')
6 #Menggunakan method classes dengan data classes.npy yang di
7 eksport.
8
9 def predict(img_path):
10     #Membuat fungsi predict dengan path img
11     newimg = keras.preprocessing.image.img_to_array(pil_image.
12         open(img_path))
13     #Membuat variable newping dengan isi mengubah bentuk image
14     #menjadi array dan membuka data berdasarkan img path
15     newimg /= 255.0
16     #Membagi data yang terdapat pada newimg dengan 255.0
17
18     # do the prediction
19     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))

```

```

16 #Membuat variabel prediction dengan isian variabel model2
17 menggunakan fungsi predict dengan syarat variabel newimg
18 dengan data reshape
19
20 # figure out which output neuron had the highest score , and
21 reverse the one-hot encoding
22 inverted = label_encoder2.inverse_transform([ np.argmax(
23 prediction)]) # argmax finds highest-scoring output
#Membuat variabel inverted dengan label encoder2 dan
#menggunakan argmax untuk mencari skor luaran tertinggi
24 print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
25 max(prediction)))
#Menampilkan prediksi gambar dan confidence dari gambar.

```

```
in [30]: runcell('[19]:restore the class name to integer encoder', 'D:/New folder/KB3C/src/1174084/7/1174084.py')
```

**Gambar 7.108** Hasil Praktek 19

## 20. Jelaskan kode program pada blok In[20]

```

1 # In[20]: grab an image (we'll just use a random training image
2 # for demonstration purposes)
3 predict("D:/New folder/KB3C/src/1174084/7/HASYv2/hasy-data/v2
4 #Melakukan prediksi dari pelatihan dari gambar v2-00010.png
5 predict("D:/New folder/KB3C/src/1174084/7/HASYv2/hasy-data/v2
6 #Melakukan prediksi dari pelatihan dari gambar v2-00500.png
7 predict("D:/New folder/KB3C/src/1174084/7/HASYv2/hasy-data/v2
7 #Melakukan prediksi dari pelatihan dari gambar v2-00700.png

```

```
in [36]: runcell("[20]: grab an image (we'll just use a random training image for
demonstration purposes)", 'D:/New folder/KB3C/src/1174084/7/1174084.py')
Prediction: A, confidence: 0.48
Prediction: \pi, confidence: 0.54
Prediction: \alpha, confidence: 0.87
```

**Gambar 7.109** Hasil Praktek 20

### 7.5.3 Bukti Tidak Plagiat



**Gambar 7.110** plagiarism

### 7.5.4 Link Video Youtube

<https://youtu.be/djRkJBfOFJs>

## 7.6 1174070 - Arrizal Furqona Gifary

### 7.6.1 Soal Teori

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

Tokenizer untuk memisahkan input text menjadi potongan yang memiliki arti disebut tokenization. Hasil dari proses tokenization disebut token. Token dapat berupa kata, kalimat, paragraph dan lainnya. Untuk ilustrasi, lihat gambar berikut:

Contoh kalimat	Disebabkan menjadi
This is Andre's text, isn't it?	<ul style="list-style-type: none"> <li>• This</li> <li>• is</li> <li>• Andre's</li> <li>• text,</li> <li>• isn't</li> <li>• it?</li> </ul>

Gambar 7.111 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
2 kfold = StratifiedKFold(n_splits=5)

```

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:

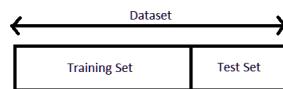
A	B	C	D	E	F	G	H
1 DATA			HASIL			AKURASI DATA	
2 1						%	
3 2						%	
4 3						%	
5 4						%	
6 5						%	

Gambar 7.112 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 7.113** Teori 3

- Jelaskan apa maksudnya kode program `train content = df['CONTENT'].iloc[train_idx]` dan `test content = df['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

```

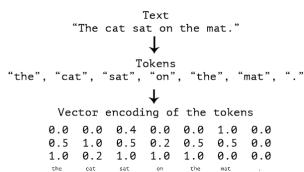
```

Out[5]:
a    1
b    2
c    3
d    4
Name: 0, dtype: int64

```

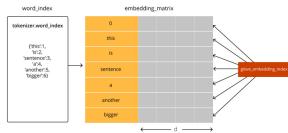
**Gambar 7.114** Teori 4

- 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 7.115** 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 7.116** 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]])
>>> npamax(a)           # Maximum of the flattened array
3
>>> npamax(a, axis=0)  # Maxima along the first axis
array([2, 3])
>>> npamax(a, axis=1)  # Maxima along the second axis
array([1, 3])
>>> npamax(a, where=[False, True], initial=-1, axis=0)
array([-1, 3])
>>> b = np.arange(5, dtype=float)
>>> b[2] = np.nan
>>> npamax(b)
nan
>>> npamax(b, where=~np.isnan(b), initial=-1)
4.0
>>> np.nanmax(b)
4.0
```

**Gambar 7.117** 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:

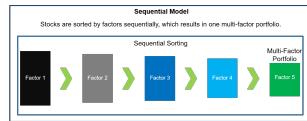
```
>>> V_train
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1],
       ...
       [0, 1, 0],
       [1, 0, 0],
       [1, 0, 0], dtype=int64)
>>> V_train.flatten()
array([1, 0, 0, ..., 1, 0, 0], dtype=int64)
>>> V_train
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1],
       ...
       [0, 1, 0],
       [1, 0, 0],
       [1, 0, 0], dtype=int64)
>>> np_utils.to_categorical(V_train)
array([[ 1.,  0.],
       [ 0.,  1.],
       [ 1.,  0.],
       ...
       [ 1.,  0.],
       [ 0.,  1.],
       [ 0.,  1.]], dtype=float32)
```

**Gambar 7.118** Teori 8

- 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.



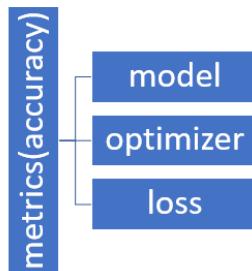
**Gambar 7.119** Teori 9

- 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 7.120** Teori 10

11. Jelaskan apa itu Deep Learning.

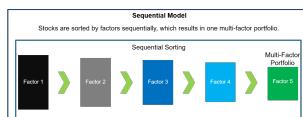
Deep Learning adalah salah satu cabang dari ilmu machine learning yang terdiri dari algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam machine learning dapat digunakan baik untuk 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 ( $NPM \text{ mod}3+1$ ) x ( $NPM \text{ mod}3+1$ ) yang terdapat max pooling.

$1174070 \text{ mod}3+1 \times 1174070 \text{ mod}3+1 = 2 \times 2$ , adapun ilustrasi gambar nya sebagai berikut :



**Gambar 7.121** Teori 9

## 7.6.2 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 yang berguna untuk mengolah data berupa gambar
      # kemudian mengimport librari keras yang menggunakan method preprocessing
      # yang digunakan untuk membuat neutal network
      #import keras.preprocessing.image
Using TensorFlow backend.

```

**Gambar 7.122** 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 :

### 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 7.123** 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 [1]: import numpy as np
...
...
...
...
...
...

```

**Gambar 7.124** 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
...

```

**Gambar 7.125** 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 7.126 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)
... onehot_encoder.fit(integer_encoded)
... OneHotEncoder(categories='auto', drop=None, dtype<class 'numpy.float64'>,
handle_unknown='error', sparse=False)
```

Gambar 7.127 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
test_output_int = label_encoder.transform(test_output)
test_output_int
test_output_int = vectorizer.inverse_transform(test_output_int.reshape(-1, test_output_int.shape[1]))
num_classes = len(label_encoder.classes_)
num_classes
Number of classes: 369
```

**Gambar 7.128** 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 7.129** 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
22 # optimasi
23 model.compile(loss='categorical_crossentropy', optimizer='adam',
24 metrics=['accuracy'])
25 # mencetak variabel model kemudian memunculkan kesimpulan berupa
26 # data total parameter, trainable paremeter dan bukan trainable
27 # parameter
28 print(model.summary())

```

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

Layer (Type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 16, 16, 32)	896
max_pooling2d_6 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_7 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_7 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_5 (Flatten)	(None, 352)	0
dense_9 (Dense)	(None, 28)	147584
dropout_4 (Dropout)	(None, 28)	0
dense_10 (Dense)	(None, 369)	47601

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

**Gambar 7.130** 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 [11]: import keras.callbacks
        tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-style')

```

**Gambar 7.131** 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 1/10: loss: 1.8515 - val_loss: 0.6283 - val_accuracy: 0.7382
Epoch 2/10: loss: 0.8196 - val_loss: 0.6032 - val_accuracy: 0.7382
Epoch 3/10: loss: 0.8196 - val_loss: 0.6032 - val_accuracy: 0.7382
Epoch 4/10: loss: 0.8196 - val_loss: 0.6032 - val_accuracy: 0.7382
Epoch 5/10: loss: 0.8196 - val_loss: 0.6032 - val_accuracy: 0.7382
Epoch 6/10: loss: 0.8196 - val_loss: 0.6032 - val_accuracy: 0.7382
Epoch 7/10: loss: 0.8196 - val_loss: 0.6032 - val_accuracy: 0.7382
Epoch 8/10: loss: 0.8196 - val_loss: 0.6032 - val_accuracy: 0.7382
Epoch 9/10: loss: 0.8196 - val_loss: 0.6032 - val_accuracy: 0.7382
Epoch 10/10: loss: 0.8196 - val_loss: 0.6032 - val_accuracy: 0.7382
Epoch 1/10: loss: 0.8091 - accuracy: 0.7248 - val_loss: 0.6069 - val_accuracy: 0.7323
Epoch 2/10: loss: 0.8091 - accuracy: 0.7248 - val_loss: 0.6069 - val_accuracy: 0.7323
Epoch 3/10: loss: 0.8091 - accuracy: 0.7247 - val_loss: 0.6062 - val_accuracy: 0.7362
Epoch 4/10: loss: 0.8091 - accuracy: 0.7242 - val_loss: 0.6058 - val_accuracy: 0.7329
Epoch 5/10: loss: 0.8091 - accuracy: 0.7232 - val_loss: 0.6059 - val_accuracy: 0.7329
Epoch 6/10: loss: 0.8091 - accuracy: 0.7215 - val_loss: 0.6049 - val_accuracy: 0.7318
Epoch 7/10: loss: 0.8091 - accuracy: 0.7204 - val_loss: 0.6052 - val_accuracy: 0.7313
Epoch 8/10: loss: 0.8091 - accuracy: 0.7204 - val_loss: 0.6052 - val_accuracy: 0.7313
Epoch 9/10: loss: 0.8091 - accuracy: 0.7204 - val_loss: 0.6054 - val_accuracy: 0.7354
Epoch 10/10: loss: 0.8091 - accuracy: 0.7204 - val_loss: 0.6056 - val_accuracy: 0.7354
Epoch 1/10: loss: 0.8176 - accuracy: 0.7294 - val_loss: 0.6042 - val_accuracy: 0.7389
Epoch 2/10: loss: 0.8176 - accuracy: 0.7294 - val_loss: 0.6042 - val_accuracy: 0.7389
Epoch 3/10: loss: 0.8152 - accuracy: 0.7293 - val_loss: 0.6033 - val_accuracy: 0.7381
Epoch 4/10: loss: 0.8152 - accuracy: 0.7293 - val_loss: 0.6033 - val_accuracy: 0.7381
Epoch 5/10: loss: 0.8152 - accuracy: 0.7293 - val_loss: 0.6032 - val_accuracy: 0.7381
Epoch 6/10: loss: 0.8152 - accuracy: 0.7293 - val_loss: 0.6032 - val_accuracy: 0.7381
Epoch 7/10: loss: 0.8152 - accuracy: 0.7293 - val_loss: 0.6032 - val_accuracy: 0.7381
Epoch 8/10: loss: 0.8152 - accuracy: 0.7293 - val_loss: 0.6032 - val_accuracy: 0.7381
Epoch 9/10: loss: 0.8152 - accuracy: 0.7293 - val_loss: 0.6032 - val_accuracy: 0.7381
Epoch 10/10: loss: 0.8152 - accuracy: 0.7293 - val_loss: 0.6032 - val_accuracy: 0.7381
Test loss: 0.820044381891203
Test accuracy: 0.7345130000000001

```

Gambar 7.132 Hasil Soal 12.

### 13. Soal 13

```

1 # In[13]:try various model configurations and parameters to find
2     the best
3
4 # mengimport librari time
5 import time
6 #membuat variabel result dengan array kosong
7 results = []
8 # melakukan looping dengan ketentuan konvolusi 2 dimensi 1 2
9 for conv2d_count in [1, 2]:
10     # menentukan ukuran besaran fixcel dari data atau konvert 1
11     # fixcel mnjadi data yang berada pada codigan dibawah.
12     for dense_size in [128, 256, 512, 1024, 2048]:
13         # membuat looping untuk memangkas masing-masing data
14         # dengan ketentuan 0 persen 25 persen 50 persen dan 75 persen.
15         for dropout in [0.0, 0.25, 0.50, 0.75]:
16             # membuat variabel model Sequential
17             model = Sequential()
18             #membuat looping untuk variabel i dengan jarak dari
19             # hasil konvolusi.
20             for i in range(conv2d_count):
21                 # syarat jika i samadengan bobotnya 0
22                 if i == 0:
23                     # menambahkan method add pada variabel model
24                     # dengan konvolusi 2 dimensi 32 bit didalamnya dan membuat
25                     # kernel dengan ukuran 3 x 3 dan rumus aktifasi relu dan data
26                     # shape yang di hitung dari data train.
27                     model.add(Conv2D(32, kernel_size=(3, 3),
28                                     activation='relu', input_shape=np.shape(train_input[0])))

```

```

20             # jika tidak
21         else:
22             # menambahkan method add pada variabel model
23             dengan konvolusi 2 dimensi 32 bit dengan ukuran kernel 3 x3
24             dan fungsi aktivasi relu
25             model.add(Conv2D(32, kernel_size=(3, 3),
26                             activation='relu'))
26                 # menambahkan method add pada variabel model
27                 dengan isian method Max pooling berdimensi 2 dengan ukuran
28                 fixcel 2 x 2.
29                 model.add(MaxPooling2D(pool_size=(2, 2)))
30                 # merubah feature gambar menjadi 1 dimensi vektor
31                 model.add(Flatten())
32                 # menambahkan method dense untuk pemanatan data
33                 dengan ukuran dense di tentukan dengan rumus fungsi tanh.
34                 model.add(Dense(dense_size, activation='tanh'))
35                 # membuat ketentuan jika pemangkasan lebih besar dari
36                 0 persen
37                 if dropout > 0.0:
38                     # menambahkan method dropout pada model dengan
39                     nilai dari dropout
40                     model.add(Dropout(dropout))
41                     # menambahkan method dense dengan fungsi num
42                     classs dan rumus softmax
43                     model.add(Dense(num_classes, activation='softmax'))
44                     # mongkompile variabel model dengan hasi loss
45                     optimasi dan akurasi matrix
46                     model.compile(loss='categorical_crossentropy',
47                         optimizer='adam', metrics=['accuracy'])
47                         # melakukan log pada dir
48                         log_dir = './logs/conv2d_%d-dense_%d-dropout_%.2f' %
49                         (conv2d_count, dense_size, dropout)
50                         # membuat variabel tensorboard dengan isian dari
51                         librari keras dan nilai dari lig dir
52                         tensorflow = keras.callbacks.TensorBoard(log_dir=
53                         log_dir)
54                         # membuat variabel start dengan isian dari librari
55                         time menggunakan method time
56
57             start = time.time()
58             # menambahkan method fit pada model dengan data dari
59             train input train output nilai batch nilai epoch verbose
60             nilai 20 persen validation split dan callback dengan nilai
61             tnsorboard.
62             model.fit(train_input, train_output, batch_size=32,
63             epochs=10,
64             verbose=0, validation_split=0.2, callbacks
65             =[tensorboard])
66             # membuat variabel score dengan nilai evaluasi dari
67             model menggunakan data tes input dan tes output
68             score = model.evaluate(test_input, test_output,
69             verbose=2)
70             # membuat variabel end
71             end = time.time()
72             # membuat variabel elapsed
73             elapsed = end - start

```

```

54     # mencetak hasil perhitungan
55     print("Conv2D count: %d, Dense size: %d, Dropout: %.2f"
56     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))

```

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

**Gambar 7.133** 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 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',
      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
21 # optimasi
22 model.compile(loss='categorical_crossentropy', optimizer='adam',
23 metrics=['accuracy'])
23 # mencetak variabel model kemudian memunculkan kesimpulan berupa
24 # data total parameter, trainable paremeter dan bukan trainable
25 # parameter
25 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_6 (Conv2D)	(None, 36, 36, 32)	896
max_pooling2d_6 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_7 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_7 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_5 (Flatten)	(None, 1152)	0
dense_9 (Dense)	(None, 128)	147584
dropout_8 (Dropout)	(None, 128)	0
dense_10 (Dense)	(None, 169)	47681
<hr/>		
Total params:	265,329	
Non-trainable params:	0	
None		

Gambar 7.134 Hasil Soal 14.

## 15. Soal 15

```

1 # In[15]: join train and test data so we train the network on all
2 # data we have available to us
3 # melakukan join numpy menggunakan data train_input test_input
4 model.fit(np.concatenate((train_input, test_input)),
5           # kelanjutan data yang di gunakan pada join
6           train_output test_output
7           np.concatenate((train_output, test_output)),
8           #menggunakan ukuran 32 bit dan epoch 10
9           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 [10]: 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
<keras.callbacks.History at 0x21637ebc3d8>

```

**Gambar 7.135** 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 7.136** 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 7.137** 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 7.138 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 7.139 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 7.140 Hasil Soal 20.

### 7.6.3 Penanganan Error

#### 1. KeyboardInterrupt

```
file = open('input.txt', 'w')
for i in range(10000):
    file.write(str(i))
file.close()
log = keras.preprocessing.image.ImageDataGenerator(rescale=1./255)
train_generator = log.flow_from_directory('HASYv2', target_size=(150, 150),
                                         batch_size=32,
                                         class_mode='categorical')
fp = Path('HASYv2').absolute().parent / 'HASYv2'
fp.mkdir(exist_ok=True)
KeyboardInterrupt
```

Gambar 7.141 KeyboardInterrupt

## 2. Cara Penanganan Error

- KeyboardInterrupt

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

### 7.6.4 Bukti Tidak Plagiat



**Gambar 7.142**   Bukti Tidak Melakukan Plagiat Chapter 7

## 7.7 1174087 - Ilham Muhammad Ariq

### 7.7.1 Teori

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

Untuk memudahkan mesin memahami maksud dari apa yang kita inginkan dalam machine learning, kata pada teks disebut token, dan proses vektorisasi dari bentuk kata ke dalam token tersebut disebut tokenzier dan tokenzier akan merubah sebuah teks menjadi simbol, kata, ataupun biner dan bentuk lainnya kedalam token. Ilustrasinya misalkan saya mempunyai sebuah kalimat yaitu "Nama Saya Ilham Muhammad Ariq" maka ketika kita lakukan proses tokenizer maka akan berubah menjadi ['Nama', 'Saya', 'Ilham', 'Muhammad', 'Ariq'].

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

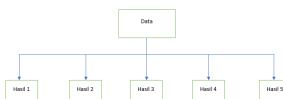
```

1 kfold = StratifiedKFold(n_splits=5)
2 splits = kfold.split(d, d['CLASS'])
3

```

**Listing 7.1**   K Fold Cross Validation

Pada koding diatas terdapat variabel kfolds 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 7.143** Illustrasi K Fold Cross Validation

3. Jelaskan apa maksudnya kode program *for train, test in splits.dilengkapi dengan ilustrasi atau gambar.*  
Maksudnya yaitu untuk menguji apakah setiap data pada dataset sudah di split dan tidak terjadi penumpukan. Yang dimana maksudnya disetiap class tidak akan muncul id yang sama. Ilustrasinya misalkan kita memiliki 5 sepatu dengan model yang berbeda. Kemudian kita bagikan ke dua orang , tentunya setiap orang yang menerima sepatu tidak memiliki model sepatu yang sama.
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.  
Maksudnya yaitu mengambil data pada kolom atau index CONTENT yang merupakan bagian dari train idx dan test idx. Ilustrasinya, ketika data telah diubah menjadi train dan test maka kita dapat memilihnya untuk ditampilkan pada kolom yang diinginkan.
5. 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 7.144** Illustrasi 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')*, 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 7.145** Illustrasi d train inputs = tokenizer.texts to matrix

7. Jelaskan apa maksud dari fungsi  $d\_train\_inputs = d\_train\_inputs / np.amax(np.absolute(d\_train\_inputs))$  dan  $d\_test\_inputs = d\_test\_inputs / np.amax(np.absolute(d\_test\_inputs))$ , 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 metode absolute. Yang berarti tanpa bilangan negatif.
8. Jelaskan apa maksud fungsi dari  $d\_train\_outputs = np_utils.to_categorical(d['CLASS'].iloc[:])$  dan  $d\_test\_outputs = np_utils.to_categorical(d['CLASS'].iloc[test_idx:])$  dalam kode program, dilengkapi dengan ilustrasi atau gambar.  
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.
9. Jelaskan apa maksud dari fungsi di listing ?? . Gambarkan ilustrasi Neural Network nya dari model kode tersebut.

```

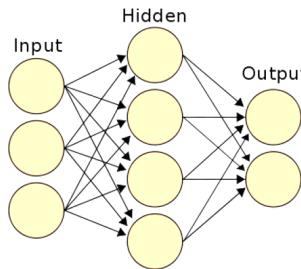
1 model = Sequential()
2 model.add(Dense(512, input_shape=(2000,)))
3 model.add(Activation('relu'))
4 model.add(Dropout(0.5))
5 model.add(Dense(2))
6 model.add(Activation('softmax'))
7
  
```

**Listing 7.2** Membuat model Neural Network

Penjelasannya sebagai berikut:

- Melakukan pemodelan Sequential
- Layer pertama dense dari 512 neuron untuk inputan dengan inputan tadi yang sudah dijadikan matriks sebanyak 2000
- Activationnya menggunakan fungsi relu yaitu jika ada inputan dengan nilai maksimum maka inputan itu yang akan terpilih.
- Dropout ini untuk melakukan pembobotan, dimana pembobotan hanya dilakukan 50 persen saja agar tidak terjadi penumpukan data dari dense inputan tadi

- Dense 2 mengkategorikan 2 neuron untuk output nya yaitu 1 dan 0.
- Untuk dense diatas aktivasinya menggunakan fungsi Softmax.



**Gambar 7.146** Illustrasi Neural Network Pemodelan

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

```

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

**Listing 7.3** Compile model

Melakukan peng compile-an dari model Sequential tadi dengan Loss yandan-gang merupakan fungsi optimisasi skor menggunakan categorical crossentropy,dan menggunakan algoritma adam sebagai optimizer. Adam yaitu algoritma pengoptimalan yang-dapatdigunakansebagai gantidari prosedur penurunang gradien stokastik klasik untuk memperbarui bobot jaringan yang berulang berdasarkan data training.Dengan metrik yaitu fungsi yang digunakan untuk menilai kinerja mode Anda disini menggunakan fungsi accuracy.

11. Jelaskan apa itu Deep Learning

Deep Learning adalah subbidang machine learning yang berkaitan dengan algoritma yang terinspirasi oleh struktur dan fungsi otak yang disebut jaringan saraf tiruan atau Artificial Neural Networks. Jaringan saraf tiruan, algoritma yang terinspirasi oleh otak manusia, belajar dari sejumlah besar data. Demikian pula dengan bagaimana kita belajar dari pengalaman, algoritma pembelajaran yang mendalam akan melakukan tugas berulang kali, setiap kali sedikit mengubahnya untuk meningkatkan hasilnya.

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

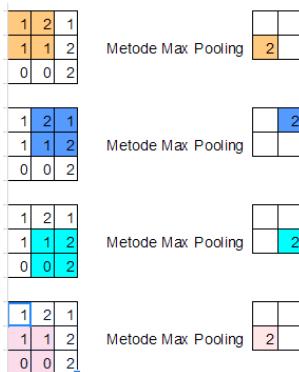
Deep Neural Network adalah jaringan syaraf tiruan (JST) dengan beberapa lapisan antara lapisan input dan output. DNN menemukan manipulasi matematis yang benar untuk mengubah input menjadi output, apakah itu hubungan lin-

ear atau hubungan non-linear. Merupakan jaringan syaraf dengan tingkat kompleksitas tertentu, jaringan syaraf dengan lebih dari dua lapisan. Deep Neural Network menggunakan pemodelan matematika yang canggih untuk memproses data dengan cara yang kompleks.

DNN hanya terdiri dari dua laipsan yaitu input dan output, sedangkan dalam Deep learning kita dapat mendefinisikan layer sebanyak yang kita inginkan atau butuhkan.

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

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 vertikal 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 7.147 Illustrasi perhitungan stride 1 max pooling

### 7.7.2 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 # menimport 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
6     yang di inisialisasi menjadi csvfile
7 with open('HASYv2/hasy-data-labels.csv') as csvfile:
8     #Menginisiasi variabel csvreader yang berisi method csv.
9     reader = csv.reader(csvfile)
10    # Menginisiasi 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 0
15        if i > 0:
16            # dibuat variabel img dengan isi keras untuk aktivasi
17            neural network fungsi yang membaca data yang berada dalam
18            folder HASYv2 dengan input nilai -1.0 dan 1.0
19            img = keras.preprocessing.image.img_to_array(
20                pil_image.open("HASYv2/" + row[0]))
21            #Pembagian data yang ada pada fungsi img sebanyak
22            255.0
23            img /= 255.0
24            # Penambahan nilai baru pada imgs pada row ke 1 2 dan
dilanjutkan dengan variabel img
25            imgs.append((row[0], row[2], img))
# Penambahan nilai pada row ke 2 pada variabel
26            classes
27            classes.append(row[2])
# penambahan nilai satu pada variabel i
28            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]
# Menginisiasi variabel test dengan isi lebih kecil split idx
10 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 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 mengguanakan 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 mengguanakan 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)

```

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
8 import tensorflow as tf

```

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 = tf.keras.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(tf.keras.layers.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 pixel
8 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

```

```

9 # variabel model di tambahkan dengan library Conv2D 32 bit dengan
  kernel 3 x 3
10 model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu'))
11 # variabel model di tambahkan dengan lib MaxPooling2D dengan
  ketentuan ukuran 2 x 2 pixel
12 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
13 # variabel model di tambahkan library Flatten
14 model.add(tf.keras.layers.Flatten())
15 # variabel model di tambahkan library Dense dengan fungsi tanh
16 model.add(tf.keras.layers.Dense(1024, activation='tanh'))
17 # variabel model di tambahkan library dropout untuk memangkas
  data tree sebesar 50 persen
18 model.add(tf.keras.layers.Dropout(0.5))
19 # variabel model di tambahkan library Dense dengan data dari
  num_classes dan fungsi softmax
20 model.add(tf.keras.layers.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())

```

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
2     the best
3 # Melakukan import library time
4 import time
5 #Menginisiasi variabel result dengan array kosong
6 results = []
7 for conv2d_count in [1, 2]:
8     # menentukan ukuran besaran fixcel dari data atau konvert 1
9         fixcel mnjadi data yang berada pada codigan dibawah.
10        for dense_size in [128, 256, 512, 1024, 2048]:
11            # membuat looping untuk memangkas masing-masing data
12            dengan ketentuan 0 persen 25 persen 50 persen dan 75 persen.
13            for dropout in [0.0, 0.25, 0.50, 0.75]:
14                # Menginisiasi variabel model Sequential
15                model = tf.keras.Sequential()
16                #membuat looping untuk variabel i dengan jarak dari
17                hasil konvolusi.
18                for i in range(conv2d_count):
19                    # syarat jika i samadengan bobotnya 0
20                    if i == 0:
21                        # Penambahan method add pada variabel model
22                        dengan konvolusi 2 dimensi 32 bit didalamnya dan membuat
23                        kernel dengan ukuran 3 x 3 dan rumus aktifasi relu dan data
24                        shape yang di hitung dari data train.
25                        model.add(tf.keras.layers.Conv2D(32,
26                            kernel_size=(3, 3), activation='relu', input_shape=np.shape(
27                                train_input[0])))
28                            # jika tidak
29                            else:
30                                # Penambahan method add pada variabel model
31                                dengan konvolusi 2 dimensi 32 bit dengan ukuran kernel 3 x3
32                                dan fungsi aktivasi relu
33                                model.add(tf.keras.layers.Conv2D(32,
34                                    kernel_size=(3, 3), activation='relu'))
35                                # Penambahan method add pada variabel model
36                                dengan isian method Max pooling berdimensi 2 dengan ukuran
37                                fixcel 2 x 2.
38                                model.add(tf.keras.layers.MaxPooling2D(pool_size
39                                =(2, 2)))
40                                # merubah feature gambar menjadi 1 dimensi vektor
41                                model.add(tf.keras.layers.Flatten())
42                                # Penambahan method dense untuk pemadatan data dengan
43                                ukuran dense di tentukan dengan rumus fungsi tanh.
44                                model.add(tf.keras.layers.Dense(dense_size ,
45                                    activation='tanh'))
46                                # membuat ketentuan jika pemangkasan lebih besar dari
47                                0 persen
48                                if dropout > 0.0:
49                                    # Penambahan method dropout pada model dengan
50                                    nilai dari dropout
51                                    model.add(tf.keras.layers.Dropout(dropout))
```

```

34         # Penambahan method dense dengan fungsi num
35         classes dan rumus softmax
36             model.add(tf.keras.layers.Dense(num_classes,
37             activation='softmax'))
38             # mongkompile variabel model dengan hasi loss
39             optimasi dan akurasi matrix
40             model.compile(loss='categorical_crossentropy',
41             optimizer='adam', metrics=['accuracy'])
42             # melakukan log pada dir
43             log_dir = '.\\logs\\conv2d_%d-dense_%d-dropout_.%2f'
44             % (conv2d_count, dense_size, dropout)
45             # Menginisiasi variabel tensorboard dengan isian dari
46             library keras dan nilai dari lig dir
47             tensorboard = keras.callbacks.TensorBoard(log_dir=
48             log_dir)
49             # Menginisiasi variabel start dengan isian dari
50             library time menggunakan method time
51
52             start = time.time()
53             # Penambahan method fit pada model dengan data dari
54             train input train output nilai batch nilai epoch verbose
55             nilai 20 persen validation split dan callback dengan nilai
56             tnsorboard .
57             model.fit(train_input, train_output, batch_size=32,
58             epochs=10,
59             verbose=0, validation_split=0.2, callbacks
60             =[tensorboard])
61             # Menginisiasi variabel score dengan nilai evaluasi
62             dari model menggunakan data tes input dan tes output
63             score = model.evaluate(test_input, test_output,
64             verbose=2)
65             # Menginisiasi variabel end
66             end = time.time()
67             # Menginisiasi variabel elapsed
68             elapsed = end - start
69             # mencetak hasil perhitungan
70             print("Conv2D count: %d, Dense size: %d, Dropout: %.2
71 f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count
72 , dense_size, dropout, score[0], score[1], elapsed))
73             results.append((conv2d_count, dense_size, dropout,
74             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 = tf.keras.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(tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
9     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(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
8 # variabel model di tambahkan dengan library Conv2D 32bit dengan
    kernel 3 x 3
9 model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu'))
10 # variabel model di tambahkan dengan lib MaxPooling2D dengan
        ketentuan ukuran 2 x 2 pixcel
11 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
12 # variabel model di tambahkan library Flatten
13 model.add(tf.keras.layers.Flatten())
14 # variabel model di tambahkan library Dense dengan fungsi tanh
15 model.add(tf.keras.layers.Dense(128, activation='tanh'))
16 # variabel model di tambahkan library dropout untuk memangkas
        data tree sebesar 50 persen
17 model.add(tf.keras.layers.Dropout(0.5))
18 # variabel model di tambahkan library Dense dengan data dari
        num_classes dan fungsi softmax
19 model.add(tf.keras.layers.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())

```

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 mengekspor 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
2 # for an arbitrary image;
3 # the code below could be placed in a separate file
4 # mengimpport library keras model
5 import keras.models
6 # Menginisiasi variabel model2 untuk meload model yang telah di
7 # simpan tadi
8 model2 = keras.models.load_model("mathsymbols.model")
9 # mencetak hasil model2
10 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
3 # label encoder.
4 label_encoder2 = LabelEncoder()
5 # Penambahan method classess dengan data classess yang di eksport
6 # tadi
7 label_encoder2.classes_ = np.load('classes.npy')
8 # membuat fumgsi predict dengan path img
9 def predict(img_path):
10     # Menginisiasi variabel newimg dengan membuat immage menjadi
11     # array dan membuka data berdasarkan img path
12     newimg = keras.preprocessing.image.img_to_array(pil_image.
13         open(img_path))
14     # membagi data yang terdapat pada variabel newimg sebanyak
15     # 255
16     newimg /= 255.0
17
18     # do the prediction
19     # Menginisiasi variabel predivtion dengan isian variabel
20     # model2 menggunakan fungsi predic dengan syarat variabel
21     # newimg dengan data reshape
22     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
23
24     # figure out which output neuron had the highest score, and
25     # reverse the one-hot encoding
26     # Menginisiasi variabel inverted dengan label encoder2 dan
27     # menggunakan argmax untuk mencari skor luaran tertinggi
28     inverted = label_encoder2.inverse_transform([np.argmax(
29         prediction)])
30     # mencetak prediksi gambar dan confidence dari gambar.
31     print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
32         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")

```

### 7.7.3 Penanganan Error

#### 1. SS Error

```

file <ipython-input-1-373905fa9ed>, line 4, in module>
      with open('HASYv2/hasy-data-labels.csv') as csvfile:
FileNotFoundError: [Errno 2] No such file or directory: 'HASYv2/hasy-data-labels.csv'

```

**Gambar 7.148** File Not Found error

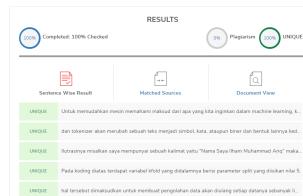
#### 2. Jenis Error

- File Not Found Error

#### 3. Cara Penanganan

Dengan cara menyesuaikan letak file yang ingin dibaca/ digunakan

### 7.7.4 Bukti Tidak Plagiat



**Gambar 7.149** Tidak Melakukan Plagiat Pada Ch 7

### 7.7.5 Link Video Youtube

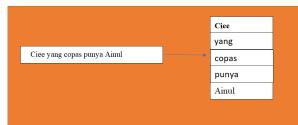
<https://youtu.be/XPdKuIEMeu8>

## 7.8 1174073 - Ainul Filiani

### 7.8.1 Soal Teori

- 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 "Cieee yang copas punya ainul", kalimat itu menjadi beberapa bagian yaitu "cieee", "yang", "copas", "punya", "ainul". Yang menjadi acuan yakni tanda baca dan spasi. Untuk ilustrasi, lihat gambar berikut:



**Gambar 7.150** Teori 1

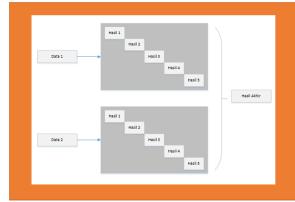
- Jelaskan konsep dasar K Fold Cross Validation pada dataset komentar Youtube pada kode listing 7.1. dilengkapi dengan ilustrasi atau gambar.

```

1 model.add(Dense(2))
2 model.add(Activation('softmax'))

```

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 7.151** Teori 2

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

Untuk penjelasan nya 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 7.152** Teori 3

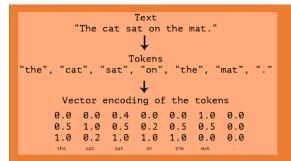
- Jelaskan apa maksudnya kode program `train content = d['CONTENT'].iloc[train_idx]` dan `test content = d['CONTENT'].iloc[test_idx]`. dilengkapi dengan ilustrasi atau gambar.

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:

a	1
b	2
c	3
d	4
<b>Name: 0, dtype: int64</b>	

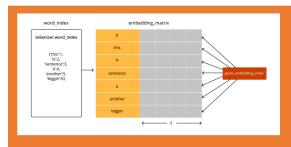
**Gambar 7.153** Teori 4

- 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 7.154** 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 7.155** 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, where=[False, True], initial=-1, axis=0)
array([-1, 3])
>>> b = np.arange(5, dtype=float)
>>> b[2] = np.nan
>>> b
nan
>>> np.sum(b, where=np.isnan(b), initial=-1)
4.0
>>> np.sum(b)
4.0

```

**Gambar 7.156** 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:

```

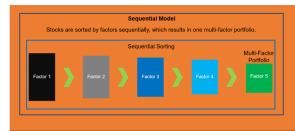
>>> X_train
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
>>> Y_train
array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0], dtype=int64)
>>> Y_train = np_utils.to_categorical(Y_train)
>>> X_train, Y_train, X_val, Y_val = train_test_split(X, Y, test_size=0.2, random_state=42)
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
>>> np_utils.to_categorical(Y_train)
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])

```

**Gambar 7.157** Teori 8

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

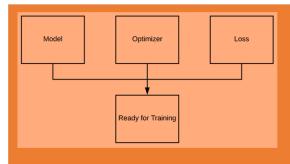
Fungsi dari baris kode tersebut ialah model perlu mengetahui bentuk input apa yang harus diharapkan. Untuk alasan ini, lapisan pertama dalam model Sequential (dan hanya yang pertama, karena lapisan berikut dapat melakukan inferensi bentuk otomatis) perlu menerima informasi tentang bentuk inputnya.



**Gambar 7.158** Teori 9

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

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 7.159** Teori 10

11. Jelaskan apa itu Deep Learning.

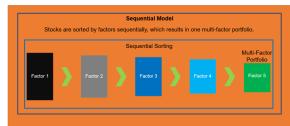
Deep Learning adalah salah satu cabang dari ilmu machine learning yang terdiri dari algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam machine learning dapat digunakan baik untuk 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 ( $NPM \text{ mod}3+1$ ) x ( $NPM \text{ mod}3+1$ ) yang terdapat max pooling.

$1174073 \text{ mod}3+1 \times 1174073 \text{ mod}3+1 = 2 \times 2$ , adapun ilustrasi gambar nya sebagai berikut :

**Gambar 7.160** Teori 9

## 7.8.2 Praktek Program

1. Soal 1

```

1 # In[1]:import lib
2 import keras.models
3 import time
4 import keras.callbacks
5 from keras.layers import Conv2D, MaxPooling2D
6 from keras.layers import Dense, Dropout, Flatten
7 from keras.models import Sequential
8 from sklearn.preprocessing import OneHotEncoder
9 from sklearn.preprocessing import LabelEncoder
10 import numpy as np
11 import random
12 import csv
13 import tensorflow as tf
14 from PIL import Image as pil_image
15 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:



**Gambar 7.161** Hasil Soal 1.

## 2. Soal 2

```

1 # In[2]:load all images (as numpy arrays) and save their classes
2
3 imgs = []
4 classes = []
5 with open('HASYv2/hasy-data-labels.csv') as csvfile:
6     csvreader = csv.reader(csvfile)
7     i = 0
8     for row in csvreader:
9         if i > 0:
10             img = keras.preprocessing.image.img_to_array(
11                 pil_image.open("HASYv2/" + row[0]))
12             # neuron activation functions behave best when input
13             # values are between 0.0 and 1.0 (or -1.0 and 1.0),
14             # so we rescale each pixel value to be in the range
15             # 0.0 to 1.0 instead of 0-255
16             img /= 255.0
17             imgs.append((row[0], row[2], img))
18             classes.append(row[2])
19             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>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 7.162** Hasil Soal 2.

### 3. Soal 3

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

**Gambar 7.163** Hasil Soal 3.

### 4. Soal 4

```
1 # In[4]:
2
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)))
```

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 keluaran nya sebagai output, . Hasilnya adalah sebagai berikut :

**Gambar 7.164** Hasil Soal 4.

### 5. Soal 5

```
1 # In[5]: import encoder and one hot
```

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 7.165** Hasil Soal 5.

## 6. Soal 6

```
1 # In[6]: convert class names into one-hot encoding  
2  
3 # first, convert class names into integers  
4 label_encoder = LabelEncoder()  
5 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 7.166** Hasil Soal 6.

## 7. Soal 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)
```

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

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

**Gambar 7.167** Hasil Soal 7.

## 8. Soal 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(
4     train_output_int.reshape(len(train_output_int), 1))
5 test_output_int = label_encoder.transform(test_output)
6 test_output = onehot_encoder.transform(
7     test_output_int.reshape(len(test_output_int), 1))
8
9 num_classes = len(label_encoder.classes_)
10 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 [8]: train_output_int = label_encoder.transform(train_output)
onehot_encoder.transform(train_output_int.reshape(len(train_output_int), 1))
test_output_int = label_encoder.transform(test_output)
onehot_encoder.transform(test_output_int.reshape(len(test_output_int), 1))
num_classes = len(label_encoder.classes_)
Number of classes: 399

```

**Gambar 7.168** Hasil Soal 8.

## 9. Soal 9

```

1 # In[9]: import sequential

```

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 7.169** Hasil Soal 9.

## 10. Soal 10

```

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

```

```
15
16 print(model.summary())
```

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

```
15 model.add(conv2D_1, kernel_size=(3, 3), activation='relu',
16     input_shape=(train_input[1]))
17     conv2D_1 = model.add(conv2D(32, (3, 3), activation='relu'))
18     model.add(maxPooling2D((2, 2), pool_size=(2, 2)))
19     model.add(conv2D(64, (3, 3), activation='relu'))
20     model.add(maxPooling2D((2, 2), pool_size=(2, 2)))
21     model.add(Flatten())
22     model.add(Dense(128, activation='tanh'))
23     model.add(Dropout(0.2))
24     model.add(Dense(10, activation='softmax'))
25     model.compile(loss='categorical_crossentropy', optimizer='adam',
26     metrics=['accuracy'])
27 print(model.summary())
Model: "sequential_5"
Layer (type)                 Output Shape              Param #
conv2D_1 (Conv2D)            (None, 30, 10, 32)      896
max_pooling2d_1 (MaxPooling2D) (None, 15, 5, 32)       0
conv2D_2 (Conv2D)            (None, 13, 13, 32)      9248
max_pooling2d_2 (MaxPooling2D) (None, 6, 6, 32)       0
flatten_1 (Flatten)          (None, 312)             0
dense_9 (Dense)              (None, 128)            147584
dropout_5 (Dropout)          (None, 128)            0
dense_10 (Dense)             (None, 369)            47601
Total params: 205,329
Trainable params: 205,329
Non-trainable params: 0
```

**Gambar 7.170** Hasil Soal 10.

## 11. Soal 11

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

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

**Gambar 7.171** Hasil Soal 11.

## 12. Soal 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])
```

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

```

Epoch 3/10 - loss: 1.5816 - accuracy: 0.6283 - val_loss: 1.0932 - val_accuracy: 0.7932
Epoch 4/10 - loss: 1.0895 - accuracy: 0.7248 - val_loss: 0.9899 - val_accuracy: 0.8021
Epoch 5/10 - loss: 1.0895 - accuracy: 0.7248 - val_loss: 0.9899 - val_accuracy: 0.8021
Epoch 6/10 - loss: 0.8985 - accuracy: 0.7477 - val_loss: 0.8985 - val_accuracy: 0.7952
Epoch 7/10 - loss: 0.8828 - accuracy: 0.7612 - val_loss: 0.8828 - val_accuracy: 0.7929
Epoch 8/10 - loss: 0.7735 - accuracy: 0.7735 - val_loss: 0.8479 - val_accuracy: 0.7938
Epoch 9/10 - loss: 0.7735 - accuracy: 0.7735 - val_loss: 0.8492 - val_accuracy: 0.7942
Epoch 10/10 - loss: 0.7036 - accuracy: 0.7935 - val_loss: 0.8556 - val_accuracy: 0.7954
Epoch 11/10 - loss: 0.6705 - accuracy: 0.7924 - val_loss: 0.8492 - val_accuracy: 0.7939
Epoch 12/10 - loss: 0.6552 - accuracy: 0.7985 - val_loss: 0.8531 - val_accuracy: 0.7961
Epoch 13/10 - loss: 0.6312 - accuracy: 0.8021 - val_loss: 0.8792 - val_accuracy: 0.8011
test loss: 0.82004181001016
test accuracy: 0.79541880003845

```

**Gambar 7.172** Hasil Soal 12.

### 13. Soal 13

```

1 model.add(tf.keras.layers.Flatten())
2 model.add(tf.keras.layers.Dense(128, activation='tanh'))
3 model.add(tf.keras.layers.Dropout(0.5))
4 model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
5 model.compile(loss='categorical_crossentropy',
6                 optimizer='adam', metrics=[accuracy])
7 print(model.summary())
8 # In[15]:join train and test data so we train the network on all
9 # data we have available to us
10 model.fit(np.concatenate((train_input, test_input)),
11           np.concatenate((train_output, test_output)),
12           batch_size=32, epochs=10, verbose=2)
13
14 # In[16]:save the trained model
15 model.save("mathsymbols.model")
16
17 # In[17]:save label encoder (to reverse one-hot encoding)
18 np.save('classes.npy', label_encoder.classes_)
19
20 # In[18]:load the pre-trained model and predict the math symbol
21 # for an arbitrary image;
22 # the code below could be placed in a separate file
23
24 model2 = keras.models.load_model("mathsymbols.model")
25 print(model2.summary())
26
27 # In[19]:restore the class name to integer encoder
28 label_encoder2 = LabelEncoder()
29 label_encoder2.classes_ = np.load('classes.npy')
30
31 def predict(img_path):
32     newimg = keras.preprocessing.image.img_to_array(pil_image.
33         open(img_path))
34     newimg /= 255.0

```

```

35 # do the prediction
36 prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
37
38 # figure out which output neuron had the highest score, and
39 # reverse the one-hot encoding
40 inverted = label_encoder2.inverse_transform(
41     [np.argmax(prediction)]) # argmax finds highest-scoring
42     # output
43 print("Prediction: %s, confidence: %.2f" %
44       (inverted[0], np.max(prediction)))
45
46 # In[20]: grab an image (we'll just use a random training image
47 # for demonstration purposes)
48 predict("HASYv2/hasy-data/v2-00010.png")
49
50 predict("HASYv2/hasy-data/v2-00500.png")
51
52 predict("HASYv2/hasy-data/v2-00700.png")

```

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

```

train count: 1, Dense slice 128, Dropout: 0.00 loss: 1.19, Accuracy: 0.74, Time: 412 sec
train count: 1, Dense slice 128, Dropout: 0.25 loss: 0.95, Accuracy: 0.79, Time: 451 sec
train count: 1, Dense slice 128, Dropout: 0.50 loss: 0.86, Accuracy: 0.81, Time: 451 sec
train count: 1, Dense slice 128, Dropout: 0.75 loss: 0.86, Accuracy: 0.77, Time: 438 sec
train count: 1, Dense slice 128, Dropout: 0.90 loss: 0.86, Accuracy: 0.77, Time: 461 sec
train count: 1, Dense slice 256, Dropout: 0.00 loss: 0.95, Accuracy: 0.77, Time: 451 sec
train count: 1, Dense slice 256, Dropout: 0.25 loss: 0.86, Accuracy: 0.81, Time: 451 sec
train count: 1, Dense slice 256, Dropout: 0.50 loss: 0.86, Accuracy: 0.77, Time: 461 sec
train count: 1, Dense slice 256, Dropout: 0.75 loss: 0.86, Accuracy: 0.77, Time: 461 sec
train count: 1, Dense slice 256, Dropout: 0.90 loss: 0.86, Accuracy: 0.77, Time: 461 sec
train count: 1, Dense slice 512, Dropout: 0.00 loss: 1.09, Accuracy: 0.74, Time: 468 sec
train count: 1, Dense slice 512, Dropout: 0.25 loss: 0.95, Accuracy: 0.79, Time: 468 sec
train count: 1, Dense slice 512, Dropout: 0.50 loss: 0.95, Accuracy: 0.79, Time: 471 sec
train count: 1, Dense slice 512, Dropout: 0.75 loss: 0.95, Accuracy: 0.79, Time: 471 sec
train count: 1, Dense slice 512, Dropout: 0.90 loss: 0.95, Accuracy: 0.79, Time: 471 sec
train count: 1, Dense slice 1024, Dropout: 0.00 loss: 2.13, Accuracy: 0.75, Time: 488 sec
train count: 1, Dense slice 1024, Dropout: 0.25 loss: 1.19, Accuracy: 0.75, Time: 488 sec
train count: 1, Dense slice 1024, Dropout: 0.50 loss: 1.09, Accuracy: 0.75, Time: 516 sec
train count: 1, Dense slice 1024, Dropout: 0.75 loss: 1.09, Accuracy: 0.75, Time: 516 sec
train count: 1, Dense slice 1024, Dropout: 0.90 loss: 1.09, Accuracy: 0.75, Time: 516 sec
train count: 1, Dense slice 2048, Dropout: 0.00 loss: 2.09, Accuracy: 0.76, Time: 587 sec
train count: 1, Dense slice 2048, Dropout: 0.25 loss: 1.19, Accuracy: 0.77, Time: 587 sec
train count: 1, Dense slice 2048, Dropout: 0.50 loss: 1.09, Accuracy: 0.77, Time: 613 sec
train count: 1, Dense slice 2048, Dropout: 0.75 loss: 1.09, Accuracy: 0.77, Time: 613 sec
train count: 1, Dense slice 2048, Dropout: 0.90 loss: 1.09, Accuracy: 0.77, Time: 613 sec
train count: 2, Dense slice 128, Dropout: 0.00 loss: 0.87, Accuracy: 0.79, Time: 533 sec
train count: 2, Dense slice 128, Dropout: 0.25 loss: 0.86, Accuracy: 0.79, Time: 544 sec
train count: 2, Dense slice 128, Dropout: 0.50 loss: 0.86, Accuracy: 0.77, Time: 544 sec
train count: 2, Dense slice 128, Dropout: 0.75 loss: 0.86, Accuracy: 0.77, Time: 544 sec
train count: 2, Dense slice 128, Dropout: 0.90 loss: 0.86, Accuracy: 0.77, Time: 544 sec
train count: 2, Dense slice 256, Dropout: 0.00 loss: 0.95, Accuracy: 0.77, Time: 588 sec
train count: 2, Dense slice 256, Dropout: 0.25 loss: 0.86, Accuracy: 0.79, Time: 588 sec
train count: 2, Dense slice 256, Dropout: 0.50 loss: 0.86, Accuracy: 0.79, Time: 588 sec
train count: 2, Dense slice 256, Dropout: 0.75 loss: 0.86, Accuracy: 0.79, Time: 588 sec
train count: 2, Dense slice 256, Dropout: 0.90 loss: 0.86, Accuracy: 0.79, Time: 588 sec
train count: 2, Dense slice 512, Dropout: 0.00 loss: 1.09, Accuracy: 0.75, Time: 581 sec
train count: 2, Dense slice 512, Dropout: 0.25 loss: 0.75, Accuracy: 0.75, Time: 581 sec
train count: 2, Dense slice 512, Dropout: 0.50 loss: 0.75, Accuracy: 0.75, Time: 581 sec
train count: 2, Dense slice 512, Dropout: 0.75 loss: 0.75, Accuracy: 0.75, Time: 581 sec
train count: 2, Dense slice 512, Dropout: 0.90 loss: 0.75, Accuracy: 0.75, Time: 581 sec
train count: 2, Dense slice 1024, Dropout: 0.00 loss: 0.77, Accuracy: 0.76, Time: 587 sec
train count: 2, Dense slice 1024, Dropout: 0.25 loss: 0.68, Accuracy: 0.76, Time: 587 sec
train count: 2, Dense slice 1024, Dropout: 0.50 loss: 0.64, Accuracy: 0.76, Time: 594 sec
train count: 2, Dense slice 1024, Dropout: 0.75 loss: 0.64, Accuracy: 0.76, Time: 594 sec
train count: 2, Dense slice 1024, Dropout: 0.90 loss: 0.64, Accuracy: 0.76, Time: 594 sec
train count: 2, Dense slice 2048, Dropout: 0.00 loss: 0.81, Accuracy: 0.77, Time: 603 sec
train count: 2, Dense slice 2048, Dropout: 0.25 loss: 0.73, Accuracy: 0.77, Time: 603 sec
train count: 2, Dense slice 2048, Dropout: 0.50 loss: 0.69, Accuracy: 0.76, Time: 612 sec
train count: 2, Dense slice 2048, Dropout: 0.75 loss: 0.69, Accuracy: 0.76, Time: 612 sec
train count: 2, Dense slice 2048, Dropout: 0.90 loss: 0.69, Accuracy: 0.76, Time: 612 sec

```

**Gambar 7.173** Hasil Soal 13.

## 14. Soal 14

```

1 # In[14]: rebuild/retrain a model with the best parameters (from
2     # the search) and use all data
3 model = tf.keras.Sequential()
4 model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
5     activation='relu',
6         input_shape=np.shape(train_input[0])))
7 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
8 model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu'))
9 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
10 model.add(tf.keras.layers.Flatten())
11 model.add(tf.keras.layers.Dense(128, activation='tanh'))
12 model.add(tf.keras.layers.Dropout(0.5))
13 model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
14

```

```

12 model.compile(loss='categorical_crossentropy',
13                 optimizer='adam', metrics=['accuracy'])
14 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 :

```

>>> model.add(Flatten())
>>> model.add(Dense(32, activation='relu'))
>>> model.add(Dense(16, activation='relu'))
>>> model.add(Dense(8, activation='relu'))
>>> model.add(Dense(4, activation='softmax'))
>>> model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
>>> print(model.summary())
Model: "sequential_5"
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_6 (Conv2D)            (None, 36, 36, 32)       896      
max_pooling2d_6 (MaxPooling2D) (None, 15, 15, 32)      0        
conv2d_7 (Conv2D)            (None, 11, 11, 32)       9248    
max_pooling2d_7 (MaxPooling2D) (None, 6, 6, 32)        0        
flatten_5 (Flatten)          (None, 1152)             0        
dense_9 (Dense)              (None, 128)             147584   
dropout_6 (Dropout)          (None, 128)             0        
dense_10 (Dense)             (None, 369)             47681    
===== 
total params: 265,229
Trainable params: 265,229
Non-trainable params: 0
None

```

Gambar 7.174 Hasil Soal 14.

## 15. Soal 15

```

1 # In[15]: join train and test data so we train the network on all
2 # data we have available to us
3 model.fit(np.concatenate((train_input, test_input)),
4           np.concatenate((train_output, test_output)),
5           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 :

```

In [16]: model.fit(np.concatenate((train_input, test_input)),
np.concatenate((train_output, test_output)),
batch_size=32, epochs=10, verbose=2)
Epoch 1/10
100/100 - loss: 1.7795 - accuracy: 0.5871
Epoch 2/10
100/100 - loss: 1.6744 - accuracy: 0.7074
Epoch 3/10
100/100 - loss: 0.9564 - accuracy: 0.7320
Epoch 4/10
100/100 - loss: 0.8977 - accuracy: 0.7458
Epoch 5/10
100/100 - loss: 0.8579 - accuracy: 0.7527
Epoch 6/10
100/100 - loss: 0.8297 - accuracy: 0.7586
Epoch 7/10
100/100 - loss: 0.8079 - accuracy: 0.7632
Epoch 8/10
100/100 - loss: 0.7917 - accuracy: 0.7663
Epoch 9/10
100/100 - loss: 0.7765 - accuracy: 0.7707
Epoch 10/10
100/100 - loss: 0.7637 - accuracy: 0.7718
<ipython-input-16>: keras.callbacks.History at 0x21637ebc348>

```

Gambar 7.175 Hasil Soal 15.

## 16. Soal 16

```

1 # In[16]: save the trained model
2 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 7.176** Hasil Soal 16.

## 17. Soal 17

```
1 # In[17]: save label encoder (to reverse one-hot encoding)
2 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 7.177** 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
4 model2 = keras.models.load_model("mathsymbols.model")
5 print(model2.summary())
```

Kode di atas befungsi untuk melakukan load data yang sudah di train dan juga memprediksi 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, 1152)	0
dense_89 (Dense)	(None, 128)	147584
dropout_34 (Dropout)	(None, 128)	0
dense_90 (Dense)	(None, 369)	47601
Total params: 269,229		
Trainable params: 269,229		
Non-trainable params: 0		
None		

**Gambar 7.178** Hasil Soal 18.

## 19. Soal 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
6 def predict(img_path):
7     newimg = keras.preprocessing.image.img_to_array(pil_image.
8         open(img_path))
9     newimg /= 255.0
10
11     # do the prediction
12     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
13
14     # figure out which output neuron had the highest score, and
15     # reverse the one-hot encoding
16     inverted = label_encoder2.inverse_transform(
17         [np.argmax(prediction)]) # argmax finds highest-scoring
18     output
19     print("Prediction: %s, confidence: %.2f" %
20           (inverted[0], np.max(prediction)))
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117

```

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 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_69 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_69 (MaxPooling2D)	(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)	47681
<hr/>		
Total params: 205,329		
Trainable params: 205,329		
Non-trainable params: 0		
<hr/>		
None		

**Gambar 7.179** Hasil Soal 19.

## 20. Soal 20

```

1 # In[20]: grab an image (we'll just use a random training image
2     # for demonstration purposes)
3 predict("HASYv2/hasy-data/v2-00010.png")
4
5 predict("HASYv2/hasy-data/v2-00500.png")
6 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('mnist/hasy-data/v2-m0001.png')
predictions[0], confidence: 0.87
predict('mnist/hasy-data/v2-m0002.png')
predictions[0], confidence: 0.58
predict('mnist/hasy-data/v2-m0003.png')
predictions[0], confidence: 0.88
```

Gambar 7.180 Hasil Soal 20.

### 7.8.3 Penanganan Error

#### 1. KeyboardInterrupt

```
File "C:\Users\Haryo\PycharmProjects\image\mnist\mnist.py", line 1, in <module>
    img = keras.preprocessing.image_img_to_array(open("mnist/v2/m0001.png") + raw[0])
  File "C:\Users\Haryo\PycharmProjects\image\mnist\mnist.py", line 1, in <module>
    fp = builtins.open(filename, "rb")
KeyboardInterrupt
```

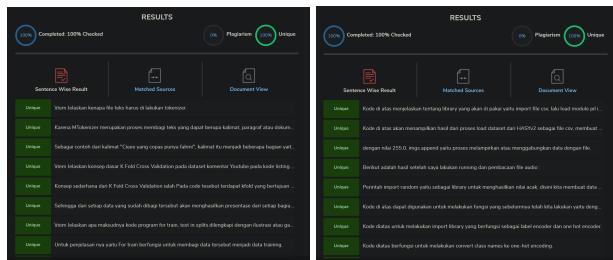
Gambar 7.181 KeyboardInterrupt

#### 2. Cara Penanganan Error

- KeyboardInterrupt

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

### 7.8.4 Bukti Tidak Plagiat



Gambar 7.182 Bukti Tidak Melakukan Plagiat Chapter 7

## 7.9 1174077 - Alvan Alvanzah

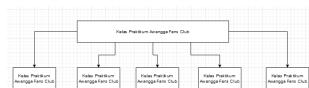
### 7.9.1 Soal Teori

#### 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 dini-lai 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 7.183** Ilustrasi Tokenizer

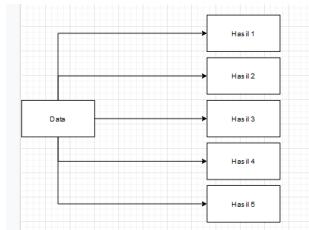
2. Jelaskan konsep dasar K Fold Cross Validation pada dataset komentar Youtube pada kode listing 7.1 ??

```

1 kfold = StratifiedKFold(n_splits=5)
2 splits = kfold.split(d, d['CLASS'])
3
  
```

**Listing 7.4** 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 7.184** 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 [5]: import numpy as np
from sklearn.model_selection import train_test_split
x = np.arange(0,100).reshape((10, 2))
y = np.sin(x[:, 0])
[[1], [2], [3], [4], [5], [6], [7], [8], [9], [10]]
In [6]: train,y_train,x_text,y_text = train_test_split(x,y,test_size=0.3,random_state=42)
[[1], [2], [3], [4], [5], [6], [7], [8], [9], [10]]
[[10], [11], [12], [13], [14], [15], [16], [17], [18], [19]]
  
```

**Gambar 7.185** 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].

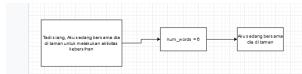
Maksud dari kode program tersebut adalah membaca isian kolom pada field yang bernama CONTENT sebagai data training dan data testing untuk program

No	Nama	Kendaraan darat yang biasanya memiliki roda 4
1	Bus	Kendaraan darat yang biasanya memiliki roda 4
2	Motor	Kendaraan darat yang biasanya memiliki roda 2
3	Traktor	Kendaraan darat yang dipakai untuk membajak sawah
4	Helikopter	Kendaraan udara yang memiliki bang - baling untuk terbang

Gambar 7.186 ilustrasi penggunaan kolom content

5. Jelaskan apa maksud dari fungsi tokenizer = Tokenizer(num words=2000) dan tokenizer.fit on texts(train content).

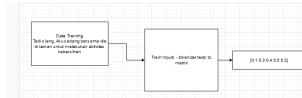
- tokenizer = Tokenizer(num\_words=2000) digunakan untuk membaca kalimat yang telah dibuat menjadi token sebanyak 2000 kata
- fit\_on\_texts digunakan untuk membuat membaca data token teks yang telah dimasukan kedalam fungsi yaitu fungsi train\_konten



Gambar 7.187 Ilustrasi Fit Tokenizer dan num\_word=2000

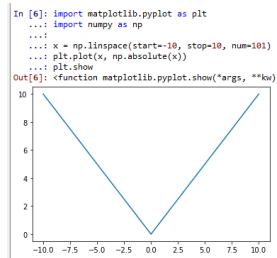
6. Jelaskan apa maksud dari fungsi d train inputs = tokenizer.texts to matrix(train content, mode='tfidf') dan d test inputs = tokenizer.texts to matrix(test content, mode='tfidf'),

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



Gambar 7.188 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 methode absolute. Yang berarti tanpa bilangan negatif.

**Gambar 7.189** fungsi d train inputs

8. Jelaskan apa maksud fungsi dari d train outputs = np utils.to categorical(d['CLASS'].iloc[0]) 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 = [0,2,1,1,2,0,1]
...:
...: array([[1., 0., 0.],
...:        [0., 0., 1.],
...:        [0., 1., 0.],
...:        [0., 0., 1.],
...:        [1., 0., 0.],
...:        [0., 1., 0.]]), dtype=float32)
```

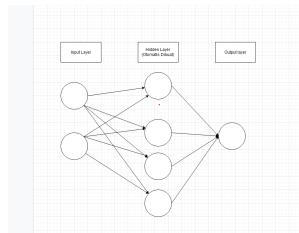
**Gambar 7.190** fungsi train outputs = np utils.to categorical

9. Jelaskan apa maksud dari fungsi di listing 7.2 ??.

```
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

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 7.191** ilustrasi neural network

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

```

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

```

**Listing 7.6** 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.

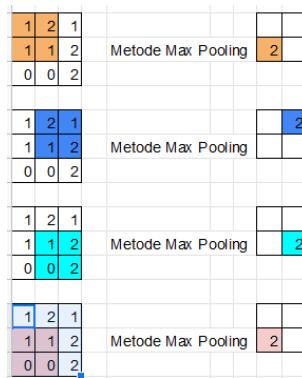
11. Jelaskan apa itu Deep Learning

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 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 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 7.192 ilustrasi perhitungan stride 1 max pooling

### 7.9.2 Praktek Program

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

```

```

In [14]: import csv
...#melakukan Melakukan import library Image yang berguna untuk dari
PIL atau Python Imaging Library yang berguna untuk mengolah data berupa
gambar
...from PIL import Image as pil_image
...# kemudian Melakukan import library keras yang menggunakan method
    preprocessing yang digunakan untuk membuat neural network
...import keras.preprocessing.image

```

Gambar 7.193 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
    kosong
3 imgs = []
4 classes = []

```

```

5 #membuka file hasy-data-labels.csv yang berada di foleder HASYv2
6     yang di inisialisasi menjadi csvfile
7 with open('HASYv2/hasy-data-labels.csv') as csvfile:
8     #Menginisiasi variabel csvreader yang berisi method csv.
9     reader = csv.reader(csvfile)
10    # Menginisiasi 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
17            neural network fungsi yang membaca data yang berada dalam
18            folder HASYv2 dengan input nilai -1.0 dan 1.0
19            img = keras.preprocessing.image.img_to_array(
20                pil_image.open("HASYv2/" + row[0]))
21            #Pembagian data yang ada pada fungsi img sebanyak
22            255.0
23            img /= 255.0
24            # Penambahan nilai baru pada imgs pada row ke 1 2 dan
25            dilanjutkan dengan variabel img
26            imgs.append((row[0], row[2], img))
27            # Penambahan nilai pada row ke 2 pada variabel
28            classes
29            classes.append(row[2])
30            # penambahan nilai satu pada variabel i
31            i += 1
32
33 # In [3]: shuffle the data, split into 80% train, 20% test
34 # Melakukan import library random
35 import random

```

```

In [3]: imgs = []
        classes = []
        # membuka file hasy-data-labels.csv yang berada di foleder HASYv2
        with open('hasy-data-labels.csv') as csvfile:
            #menginisiasi variabel csvreader yang berisi method
            reader = csv.reader(csvfile)
            # menginisiasi variabel i dengan isi 0
            i = 0
            # membuat looping pada variabel csvreader
            for row in csvreader:
                # dengan ketentuan jika i lebihkecil daripada o
                if i > 0:
                    # 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
                    img =
                        keras.preprocessing.image.img_to_array(pil_image.open("HASYv2/" +
                    row[0]))
                    #Pembagian data yang ada pada fungsi img sebanyak
                    255.0
                    img /= 255.0
                    # Penambahan nilai baru pada imgs pada row ke 1 2 dan
                    imgs.append((row[0], row[2], img))
                    # Penambahan nilai pada row ke 2 pada variabel
                    classes
                    classes.append(row[2])
                    # penambahan nilai satu pada variabel i
                    i += 1

```

Gambar 7.194 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  
4 # dikali dari pengembalian jumlah dari variabel imgs  
5 split_idx = int(0.8*len(imgs))  
6 # Menginisiasi variabel train dengan isi lebih besar split_idx  
7 train = imgs[:split_idx]  
8 # Menginisiasi variabel test dengan isi lebih kecil split_idx  
9 test = imgs[split_idx :]  
10  
11 # In [4]:  
12 # Melakukan import library numpy dengan inisial np  
13 import numpy as np
```

**Gambar 7.195** 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  
2     mana membuat array dengan isi row 2 dari data train  
3 train_input = np.asarray(list(map(lambda row: row[2], train)))  
4 # membuat test_input input dengan np method asarray yang mana  
5     membuat array dengan isi row 2 dari data test  
6 test_input = np.asarray(list(map(lambda row: row[2], test)))  
7 # Menginisiasi variabel train_output dengan np method asarray  
8     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  
11     mana membuat array dengan isi row 1 dari data test  
12 test_output = np.asarray(list(map(lambda row: row[1], test)))  
13  
14 # In [5]: import encoder and one hot  
15 # Melakukan import library LabelEncode dari sklearn  
16 from sklearn.preprocessing import LabelEncoder
```

### Gambar 7.196 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 kom-

puter 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
...: from sklearn.preprocessing import OneHotEncoder
```

**Gambar 7.197** 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()
...: integer_encoded = label_encoder.fit_transform(classes)
...: integer_encoded = label_encoder.fit_transform(classes)
```

**Gambar 7.198** 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 [1]: onehot_encoder = OneHotEncoder(sparse=False)
... # mongt variable Integer_encoded into integer_encoded
... long to list
... integer_encoded = Integer_encoded.reshape(len(Integer_encoded),
1)
... # onehot encoder
... onehot_encoder.fit(integer_encoded)
D:\Users\anand\Downloads\libsite-packages\sklearn\preprocessing\_encoders.py:105: UserWarning: This class 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 by the categories parameter. If you want the future behaviour and silence this warning, you can specify 'categories='auto' before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.
  warnings.warn("This class 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 by the categories parameter. If you want the future behaviour and silence this warning, you can specify 'categories='auto' before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.", FutureWarning)
Out[1]: OneHotEncoder(categorical_features='one-hot', categories='one-hot', drop=None,
                   encoding='one-hot', handle_unknown='error',
                   n_values=None, sparse=False)
```

**Gambar 7.199** 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_int)
        ... # Mengandalkan variabel tersembunyi. Train_output_int tidak diperlukan lagi
        ...
        train_output = onehot_encoder.transform(train_output_int).reshape((n_train, output_dim))
        ...
        # Mengandalkan data test, output menggunakan variabel. Label, encode menggunakan variabel
        ...
        test_output_int = label_encoder.transform(test_output)
        ...
        test_output = onehot_encoder.transform(test_output).test_output_int tidak diperlukan lagi
        ...
        onehot_encoder = OneHotEncoder()
        ...
        test_output = onehot_encoder.transform(test_output).int.reshape((n_test, output_dim))
        ...
        # Mengandalkan variabel. num_classes dengan isi variabel
        label_encoder = LabelEncoder()
        ...
        num_classes = len(label_encoder.classes_)
        ...
        print(f'Number of classes: {num_classes} \t Number of classes: {num_classes}'
```

Gambar 7.200 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 7.201** 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 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())

```

...: print(model.summary())		
Model: sequential_1	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)	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 7.202 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 [15]: import keras.callbacks
...: # Menginisiasi variabel tensorboard dengan isi lib keras
...: tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-
    style')

```

Gambar 7.203 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: 0.7280
Epoch 8/10
187668/187668 - 81s - loss: 0.9793 - accuracy: 0.7387 - val_loss: 0.8903
- val_accuracy: 0.7518
Epoch 9/10
187668/187668 - 78s - loss: 0.8683 - accuracy: 0.7539 - val_loss: 0.8577
- val_accuracy: 0.7545
Epoch 10/10
187668/187668 - 82s - loss: 0.7983 - accuracy: 0.7676 - val_loss: 0.8427
- val_accuracy: 0.7629
Epoch 5/10
187668/187668 - 81s - loss: 0.7511 - accuracy: 0.7771 - val_loss: 0.8347
- val_accuracy: 0.7796
Epoch 6/10
187668/187668 - 81s - loss: 0.7061 - accuracy: 0.7872 - val_loss: 0.8329
- val_accuracy: 0.7876
Epoch 7/10
187668/187668 - 85s - loss: 0.6712 - accuracy: 0.7991 - val_loss: 0.8389
- val_accuracy: 0.7957
Epoch 8/10
187668/187668 - 86s - loss: 0.6437 - accuracy: 0.8014 - val_loss: 0.8562
- val_accuracy: 0.8155
Epoch 9/10
187668/187668 - 86s - loss: 0.6216 - accuracy: 0.8985 - val_loss: 0.8719
- val_accuracy: 0.7072
Epoch 10/10
187668/187668 - 86s - loss: 0.5980 - accuracy: 0.8185 - val_loss: 0.8638
- val_accuracy: 0.7284
33447/33447 - loss: 0.6059 - accuracy: 0.7634
Test loss: 0.86598172929628
Test accuracy: 0.76559644

```

**Gambar 7.204** 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
           hasil konvolusi.
15            for i in range(conv2d_count):
16                # syarat jika i samadengan bobotnya 0
17                if i == 0:
18                    # Penambahan 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.
19                model.add(Conv2D(32, kernel_size=(3, 3),
20                               activation='relu', input_shape=np.shape(train_input[0])))
21                               # jika tidak
22                               else:
23                               # Penambahan method add pada variabel model
           dengan konvolusi 2 dimensi 32 bit dengan ukuran kernel 3 x3
           dan fungsi aktivasi relu
24                               model.add(Conv2D(32, kernel_size=(3, 3),
25                               activation='relu')))
```

```

24          # Penambahan method add pada variabel model
25          dengan isian method Max pooling berdimensi 2 dengan ukuran
26          fixcel 2 x 2.
27          model.add(MaxPooling2D(pool_size=(2, 2)))
28          # merubah feature gambar menjadi 1 dimensi vektor
29          model.add(Flatten())
30          # Penambahan method dense untuk pemadatan data dengan
31          ukuran dense di tentukan dengan rumus fungsi tanh.
32          model.add(Dense(dense_size, activation='tanh'))
33          # membuat ketentuan jika pemangkasan lebih besar dari
34          0 persen
35          if dropout > 0.0:
36              # Penambahan method dropout pada model dengan
37              nilai dari dropout
38              model.add(Dropout(dropout))
39              # Penambahan method dense dengan fungsi num
40              classs dan rumus softmax
41              model.add(Dense(num_classes, activation='softmax'))
42              # mongkompile variabel model dengan hasi loss
43              optimasi dan akurasi matrix
44              model.compile(loss='categorical_crossentropy',
45              optimizer='adam', metrics=['accuracy'])
45              # melakukan log pada dir
46              log_dir = './logs/conv2d_%d-dense_%d-dropout_%.2f' %
47              (conv2d_count, dense_size, dropout)
48              # Menginisiasi variabel tensorboard dengan isian dari
49              library keras dan nilai dari lig dir
50              tensorboard = keras.callbacks.TensorBoard(log_dir=
51              log_dir)
52              # Menginisiasi variabel start dengan isian dari
53              library time menggunakan method time
54
55              start = time.time()
56              # Penambahan method fit pada model dengan data dari
57              train input train output nilai batch nilai epoch verbose
58              nilai 20 persen validation split dan callback dengan nilai
59              tnsorboard .
60              model.fit(train_input, train_output, batch_size=32,
61              epochs=10,
62                  verbose=0, validation_split=0.2, callbacks
63                  =[tensorboard])
64                  # Menginisiasi variabel score dengan nilai evaluasi
65                  dari model menggunakan data tes input dan tes output
66                  score = model.evaluate(test_input, test_output,
67                  verbose=2)
68                  # Menginisiasi variabel end
69                  end = time.time()
70                  # Menginisiasi variabel elapsed
71                  elapsed = end - start
72                  # mencetak hasil perhitungan
73                  print("Conv2D count: %d, Dense size: %d, Dropout: %.2
74                  f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count
75                  , dense_size, dropout, score[0], score[1], elapsed))
76                  results.append((conv2d_count, dense_size, dropout,
77                  score[0], score[1], elapsed))

```

```
% (conv2d_count, dense_size, dropout)
...
# Menginisiasi variabel tensorboard dengan isian
dari library keras.callbacks import TensorBoard
tensorboard = TensorBoard(log_dir='log_dir')
keras.callbacks.TensorBoard(log_dir=log_dir)
...
# Menginisiasi variabel start dengan istan dari
time.time() mengingatkan waktu awal
...
start = time.time()
...
# Pada method fit pada model dengan data dan
train_input train_output nilai batch nilai epoch verbose nilai 20 persen
validation_split dari data yang tersedia
...
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 test_input dan test_output
...
score = model.evaluate(test_input, test_output,
verbose=2)
...
# Menginisiasi variabel end
end = time.time()
...
print("Total training time elapsed")
elapsed = end - start
...
# mencetak hasil perhitungan
...
print("conv2d count: %d, dense size: %d, Dropout: %f, Accuracy: %f, Time: %f" % (conv2d_count,
dense_size, dropout, score[0], score[1], elapsed))
...
score[0].append(conv2d_count, dense_size, dropout,
score[0], score[1], elapsed)
```

Gambar 7.205 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
the search) and use all data
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',
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 library Conv2D 32 bit 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 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 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_11 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_11 (MaxPooling2D)	(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)	425601
<hr/>		
Total params:	295,350	
Trainable params:	295,329	
Non-trainable params:	0	
<hr/>		
	None	

Gambar 7.206 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: 0.9910 - 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 7.207 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")
/usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/resource_variable_ops.py:1786: calling BaseResourceVariable._(root_init) on a variable (e.g., tf.Variable) with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass "constraint" arguments to layers.
INFO:tensordeserialize: written to: mathsymbols.model/assets

```

Gambar 7.208 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 7.209** 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 [80]: model2.load("mathsymbols.model")
WARNING:tensorflow:From C:\Users\TITICENDIX\AI\appData\Roaming\Python
\Python37\site-packages\tensorflow_core\python\ops
\resource_variable_ops.py:1786: calling BaseResourceVariable._init_
(from tensorflow.python.ops.resource_variable_ops) with constraint is
deprecated and will be removed in a future version.
Instead, use the update_op argument of the constraint.
If using keras pass "constraint" arguments to layers.
INFO:tensorflow:Assets written to: mathsymbols.model.assets
```

**Gambar 7.210** 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 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
9 array dan membuka data berdasarkan img path
10 newimg = keras.preprocessing.image.img_to_array(pil_image.
11 open(img_path))
12 # membagi data yang terdapat pada variabel newimg sebanyak
13 255
14 newimg /= 255.0
15
16
17 # do the prediction
18 # Menginisiasi variabel predivtion dengan isian variabel
19 model2 menggunakan fungsi predic dengan syarat variabel
20 newimg dengan data reshape
21 prediction = model2.predict(newimg.reshape(1, 32, 32, 3))

22 # figure out which output neuron had the highest score, and
23 reverse the one-hot encoding
24 # Menginisiasi variabel inverted dengan label encoder2 dan
25 menggunakan argmax untuk mencari skor luaran tertinggi
26 inverted = label_encoder2.inverse_transform([np.argmax(
27 prediction)])
28 # mencetak prediksi gambar dan confidence dari gambar.
29 print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
30 max(prediction)))

```

```

In [83]: label_encoder2 = LabelEncoder()
...# Pada blok ini dibuatkan sebuah classes dengan data classes yang di
eksport dari ...
...# label_encoder2.classes_ = np.load('classes.npy')
...# ... def __init__(self, path):
...# ...     self.classes_ = np.load(path)
...# ...     # Menginisiasi variabel reading dengan membuat Immage menjadi
array dan membuka data berdasarkan img path
...# ...     reading =
keras.preprocessing.image.img_to_array(pil_image.open(img_path))
255
...# ...     reading /= 255.0
...# ...     # do the prediction
...# ...     # Menginisiasi variabel inverted dengan label encoder2 di
model2 menggunakan fungsi predic dengan syarat variabel reading dengan
data reshape
...# ...     prediction = model2.predict(reading.reshape(1, 32, 32, 3))
...# ...     # Figure out which output neuron had the highest score, am
reverse the one-hot encoding
...# ...     # Menginisiasi variabel inverted dengan label encoder2 di
menggunakan fungsi mencari skor luaran tertinggi
...# ...     inverted =
label_encoder2.inverse_transform([np.argmax(prediction)])
...# ...     print("Prediction: %s, confidence: %.2f" % (inverted[0],
np.max(prediction)))

```

**Gambar 7.211** 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
2 for demonstration purposes)
3 # mencari prediksi menggunakan fungsi prediksi yang di buat tadi
4 dari data di HASYv2/hasy-data/v2-00010.png
5 predict("HASYv2/hasy-data/v2-00010.png")
6 # mencari prediksi menggunakan fungsi prediksi yang di buat tadi
7 dari data di HASYv2/hasy-data/v2-00500.png
8 predict("HASYv2/hasy-data/v2-00500.png")
9 # mencari prediksi menggunakan fungsi prediksi yang di buat tadi
10 dari data di HASYv2/hasy-data/v2-00700.png
11 predict("HASYv2/hasy-data/v2-00700.png")

```

```
In [86]: predict("D:/v2-890500.png")
...: # mencari prediksi menggunakan fungsi prediksi yang di
dari data di D:/mydata/v2-890500.png
...: predict("D:/v2-895500.png")
...: # mencari prediksi menggunakan fungsi prediksi yang di
dari data di D:/mydata/v2-895500.png
...: predict("D:/v2-897500.png")
```

**Gambar 7.212** Hasil Soal 20.

### 7.9.3 Penanganan Error

- ModuleNotFoundError: No module named 'keras'

```
In [2]: import keras.preprocessing.image
ModuleNotFoundError: No module named 'keras'.
```

**Gambar 7.213** NameError

- Tuliskan Kode Error dan Jenis Error

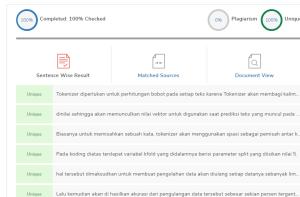
- ModuleNotFoundError

- 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.

### 7.9.4 Bukti Tidak Plagiat



**Gambar 7.214** Bukti Tidak Melakukan Plagiat Chapter 7

## 7.10 1174054 - Aulyardha Anindita

### 7.10.1 Teori

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

Kata pada suatu teks disebut token. Proses vektorisasi dari bentuk kata kedalam token biasa disebut dengan tokenizer dan tokenizer tersebut akan merubah suatu

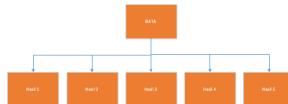
teks tersebut menjadi simbol, kata, ataupun biner dan bentuk lainnya kedalam token.



**Gambar 7.215** Tokenizer

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

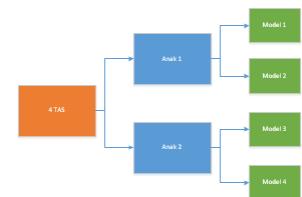
StartifieldKFold berisi presentasi sampel dalam setiap kelas, dimana pada ilustrasi ini sampel tersebut dibagi menjadi 5 bagian dalam setiap classnya, lalu sampel tersebut dimasukkan kedalam class dari dataset youtube tersebut.



**Gambar 7.216** K-Fold Cross Validation

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

Maksud dari kode program for train, test in splits, yaitu menguji apakah setiap data pada dataset sudah displit dan tidak terjadi penumpukan, yang dimana setiap class tidak akan muncul dengan id yang sama. misalnya ketika kita akan mempunyai 4 tas dengan model yang berbeda, lalu kita membaginya menjadi dua anak, tentunya setiap anak tersebut akan menerima tas yang modelnya tidak sama.



**Gambar 7.217** Maksud Kode dor train, test in splits

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 mengambil data pada kolom atau index CONTENT yang merupakan bagian dari train idx dan test idx. Ilustrasinya, ketika data telah diubah menjadi train dan test maka kita dapat memilihnya untuk ditampilkan pada kolom yang diinginkan.

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 membajak sawah
4	Helikopter	Kendaraan udara yang memiliki baling-baling untuk berbang

**Gambar 7.218** Maksud Kode train content

5. Jelaskan apa maksud dari fungsi tokenizer = Tokenizer(num words=2000) dan tokenizer.fit on texts(train content), dilengkapi dengan ilustrasi atau gambar  
 Tokenizer(num words=2000) digunakan untuk membaca kalimat yang telah dibuat menjadi token sebanyak 2000 kata tokenizer.fit on texts(train content) digunakan untuk membuat membaca data token teks yang telah dimasukkan kedalam fungsi yaitu fungsi train konten



**Gambar 7.219** Maksud dari fungsi Tokenizer

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, mod='tfidf'), dilengkapi dengan ilustrasi kode atau gambar

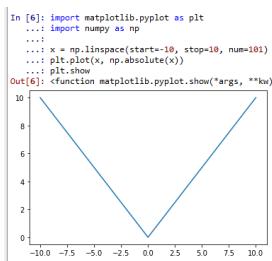
Maksudnya yaitu untuk variabel d train inputs akan melakukan tokenizer dari bentuk teks kematrix dari data train content dengan mode matriksnya yaitu tfidf begitu juga dengan variabel d test inputs untuk data test. berikut gambar ilustrasinya:



**Gambar 7.220** Maksud Kode train inputs

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))), dilengkapi dengan ilustrasi atau gambar.

Fungsi tersebut akan membagi matrix tfidf tadi dengan amax yaitu mengembalikan maksimum array atau maksimum sepanjang sumbu. Yang hasilnya akan dimasukkan kedalam variabel d train inputs untuk data train dan d test inputs untuk data test dengan nominal absolut atau tanpa ada bilangan negatif dan koma.



**Gambar 7.221** Maksud Kode train inputs

8. Jelaskan apa maksud fungsi dari d train outputs = np utils.to categorical(d['CLASS'].iloc[0]) dan d test outputs = np utils.to categorical(d['CLASS'].iloc[test\_idx]) dalam kode program, dilengkapi dengan ilustrasi atau gambar.

Dalam variabe l'd train output dan d test outputs akan dilakukan one hot encoding,dimana np utils akan mengubah vektor dengan bentuk integer ke matriks kelas biner untuk kolom CLASS dimana nantinya hanya akan ada dua pilihan yaitu 1 atau 0. 1 untuk spam 0 untuk non spam atau sebaliknya. Berikut gambar ilustrasinya :

```
In [10]: labels = [0,2,1,2,0,1]
...
kenns.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 7.222** Maksud Kode train outputs

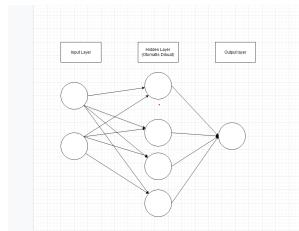
9. Jelaskan apa maksud dari fungsi di listing 7.2. Gambarkan ilustrasi Neural Network nya dari model kode tersebut

Maksud dari listing 7.2 yaitu :

- melakukan pemodelan sequential
- Layer pertama dense dari 512 neuron untuk inputan dengan inputan tadi yang sudah dijadikan matriks sebanyak 2000
- Activationnya menggunakan fungsi relu yaitu jika ada inputan dengan nilai maksimum maka inputan itu yang akan terpilih.

- Dropout ini untuk melakukan pembobotan, dimana pembobotan hanya dilakukan 50 persen saja agar tidak terjadi penumpukan data dari dense inputan tadi
- Dense 2 mengkategorikan 2 neuron untuk output nya yaitu 1 dan 0.
- Untuk dense diatas aktivasinya menggunakan fungsi Softmax.

Untuk ilustrasinya bisa dilihat pada gambar berikut :

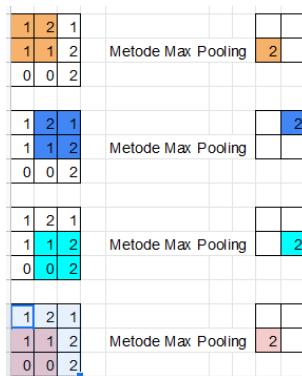


**Gambar 7.223** Ilustrasi Neural Network

10. Jelaskan apa maksud dari fungsi di listing 7.3 dengan parameter tersebut. Melakukan peng compile-an dari model Sequential tadi dengan Loss yan dengan merupakan fungsi optimisasi skor menggunakan categorical crossentropy, dan menggunakan algoritma adam sebagai optimizer. Adam yaitu algoritma pen-optimalan yang dapat digunakan sebagai ganti dari prosedur penurunan gradien stokastik klasik untuk memperbarui bobot jaringan yang berulang berdasarkan data training. Dengan metrik yaitu fungsi yang digunakan untuk menilai kinerja mode Anda disini menggunakan fungsi accuracy.
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 adalah jaringan syaraf tiruan (JST) dengan beberapa lapisan antara lapisan input dan output. DNN menemukan manipulasi matematis yang benar untuk mengubah input menjadi output, apakah itu hubungan linear Deep Neural Network adalah jaringan syaraf tiruan (JST) dengan beberapa lapisan antara lapisan input dan output. DNN menemukan manipulasi matematis yang benar untuk mengubah input menjadi ouput, apakah itu hubungan linear atau hubungan non-linear. Merupakan jaringan syaraf dengan tingkat kompleksitas tertentu, jaringan syaraf dengan lebih dari dua lapisan. Deep Neural Network menggunakan pemodelan matematika yang canggih untuk memproses data dengan cara yang kompleks. DNN hanya terdiri dari dua lapisan yaitu input dan output, sedangkan dalam Deep learning kita dapat mendefinisikan layer sebanyak yang kita inginkan atau butuhkan.

13. Jelaskan dengan ilustrasi gambar buatan sendiri (langkah per langkah) bagaimana perhitungan algoritma kobvolasi dengan ukuran stride ( $\text{NPM mod}3+1$ ) x ( $\text{NPM mod}3+1$ ) yang terdapat pada max pooling

Karena hasil dari  $(\text{NPM mod}3+1) \times (\text{NPM mod}3+1) = 2 \times 2 = 4$ , berarti ada 4 metode 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  $3 \times 3$  dengan stride 1 yang berarti setiap pergeseran 1 pixcel akan diambil nilai terbesar dari pixcel  $3 \times 3$  tersebut.



**Gambar 7.224** Illustrasi perhitungan stride 1 max pooling

### 7.10.2 Praktek

#### 1. Nomor 1

```

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

```

Dan hasilnya bisa dilihat pada gambar berikut :

```

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

```

**Gambar 7.225** Hasil Nomor 1

## 2. Nomor 2

```

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 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
32            i += 1

```

Untuk hasil plotnya bisa dilihat pada gambar berikut:

```

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]))
...:             #A neuron activation functions before best when input values are
...:             between -1.0 and 1.0.
...:             # e.g. we rescale each pixel value to be in the range 0.0 to 1.0
...:             # instead of -1.0 to 1.0.
...:             img /= 255.0
...:             imgs.append((row[0], row[2], img))
...:             classes.append(row[2])
...:             i += 1

```

**Gambar 7.226** Hasil Nomor

## 3. Nomor 3

```

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:]

```

Hasilnya dapat dilihat pada gambar berikut:

```

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

```

**Gambar 7.227** Hasil Nomor 3

#### 4. Nomor 4

```

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 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)))

```

Untuk hasilnya bisa dilihat pada gambar berikut :

```

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 7.228** Hasil Nomor 4

#### 5. Nomor 5

```

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
```

Untuk hasilnya bisa dilihat pada gambar berikut :

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

**Gambar 7.229** Hasil Nomor 5

## 6. Nomor 6

```
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)
```

Hasilnya bisa dilihat pada gambar berikut :

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

**Gambar 7.230** Hasil Nomor 6

## 7. Nomor 7

```
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)
```

Hasilnya bisa dilihat pada gambar berikut :

```
In [7]: onehot_encoder = OneHotEncoder(sparse=False)
...: integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
...: onehot_encoder.fit(integer_encoded)
...: /usr/local/lib/python3.6/dist-packages/sklearn/preprocessing/_encoders.py:191: FutureWarning:
The handling of integer categories changed 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 behavior and silence this warning, you can specify
"category='auto'" or "category='all'". See the documentation for more details.
In case you used a LabelEncoder before this OneHotEncoder to convert the categories
to integers, you need to use a LabelEncoder and a OneHotEncoder directly.
warnings.warn(msg, FutureWarning)
Out[7]:
OneHotEncoder(categorical_features='one', categories='auto',
              dtype='float64', handle_unknown='error',
              n_values='auto', sparse=False)
```

**Gambar 7.231** Hasil Nomor 7

## 8. Nomor 8

```

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)

```

Untuk hasilnya bisa dilihat pada gambar berikut :

```

In [8]: train_output_int = label_encoder.transform(train_output)
...: ...
...: onehot_encoder.transform(train_output_int.reshape(len(train_output_int), 1))
...: ...
...: test_output_int = label_encoder.transform(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 7.232** Hasil Nomor 8

## 9. Nomor 9

```

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

```

Hasilnya bisa dilihat pada gambar berikut :

```

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

```

**Gambar 7.233** Hasil Nomor 9

## 10. Nomor 10

```

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 dengan
      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 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())

```

Hasilnya bisa dilihat pada gambar berikut :

```

...| Model.add(Flatten())
...| model.add(Dense(1024, activation='tanh'))
...| model.add(Dense(1024))
...| 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\framework
op_and_grad.py:130: Using a deprecated API: tf.gradient(). This op will be removed in a future version.
Instructions for updating:
Call gradient() directly by placing
tf.GradientTape() around the code that calls gradient(). (From tensorflow.python.ops.moments_ops)
WARNING:tensorflow:From D:\Anaconda\lib\site-packages\tensorflow\backend
version_utils.py:100: Using a deprecated API: calling dropout_1 (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, 15, 15, 32)    9248
max_pooling2d_2 (MaxPooling2D) (None, 6, 6, 32)    0
flatten_1 (Flatten)   (None, 1152)        0
dense_1 (Dense)       (None, 1024)        1180072
dropout_2 (Dropout)   (None, 1024)        0
dense_2 (Dense)       (None, 300)         378225
...
Total params: 1,569,841
Trainable params: 1,569,841
Non-trainable params: 0

```

Gambar 7.234 Hasil Nomor 10

## 11. Nomor 11

```

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')

```

Hasilnya bisa dilihat pada gambar berikut :

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

**Gambar 7.235** Hasil Nomor 11

## 12. Nomor 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])

```

Hasilnya bisa dilihat pada gambar berikut :

```

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])
Warning: /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_ops.py:3860: DeprecationWarning: tf.int32 (from tensorflow/python/ops/math_ops) is deprecated and will be removed in TensorFlow 2.0 version.
Instructions for updating:
Use tf.int instead.
Train on 60000 samples, validate on 20000 samples
Epoch 1/10
Epoch 1/10: loss: 1.5538 - acc: 0.4294 - val_loss: 0.0824 - val_acc: 0.7225
Epoch 2/10: loss: 0.9694 - acc: 0.7321 - val_loss: 0.8162 - val_acc: 0.7494
Epoch 3/10: loss: 0.8543 - acc: 0.7556 - val_loss: 0.8883 - val_acc: 0.7530
Epoch 4/10: loss: 0.7954 - acc: 0.7999 - val_loss: 0.8441 - val_acc: 0.7686
Epoch 5/10: loss: 0.7364 - acc: 0.8299 - val_loss: 0.8491 - val_acc: 0.7765
Epoch 6/10: loss: 0.6936 - acc: 0.8599 - val_loss: 0.8322 - val_acc: 0.7692
Epoch 7/10: loss: 0.6595 - acc: 0.8762 - val_loss: 0.8250 - val_acc: 0.7656
Epoch 8/10: loss: 0.6344 - acc: 0.8837 - val_loss: 0.8496 - val_acc: 0.7654
Epoch 9/10: loss: 0.6087 - acc: 0.8921 - val_loss: 0.8597 - val_acc: 0.7638
Epoch 10/10: loss: 0.5905 - acc: 0.9016 - val_loss: 0.8889 - val_acc: 0.7637
Test loss: 0.743494672399488
Test accuracy: 0.75238137529488

```

**Gambar 7.236** Hasil Nomor 12

## 13. Nomor 13

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

```

38         # melakukan log pada dir
39         log_dir = './logs/conv2d_%d-dense_%d-dropout_.%2f' %
40 (conv2d_count, dense_size, dropout)
41         # Menginisiasi variabel tensorboard dengan isian dari
42 library keras dan nilai dari lig dir
43         tensorflow = keras.callbacks.TensorBoard(log_dir=log_dir)
44         # Menginisiasi variabel start dengan isian dari
45 library time menggunakan method time
46
47         start = time.time()
48         # Penambahan method fit pada model dengan data dari
49 train input train output nilai batch nilai epoch verbose
50 nilai 20 persen validation split dan callback dengan nilai
51 tnsorboard .
52         model.fit(train_input, train_output, batch_size=32,
53 epochs=10,
54             verbose=0, validation_split=0.2, callbacks
55 =[tensorboard])
56         # Menginisiasi variabel score dengan nilai evaluasi
57 dari model menggunakan data tes input dan tes output
58         score = model.evaluate(test_input, test_output,
59 verbose=2)
60         # Menginisiasi variabel end
61         end = time.time()
62         # Menginisiasi variabel elapsed
63         elapsed = end - start
64         # mencetak hasil perhitungan
65         print("Conv2D count: %d, Dense size: %d, Dropout: %.2
66 f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count,
67 , dense_size, dropout, score[0], score[1], elapsed))
68         results.append((conv2d_count, dense_size, dropout,
69 score[0], score[1], elapsed))

```

Hasilnya bisa dilihat pada gambar berikut :

**Gambar 7.237** Hasil Nomor 13

14. Nomor 14

```
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  
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',
     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 library Conv2D 32 bit 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 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())

```

Hasilnya bisa dilihat pada gambar berikut :

```

In [14]: model = Sequential()
...:     model.add(Conv2D(32,
...:         input_shape=np.shape(train_input[0])))
...:     model.add(Activation('relu'))
...:     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: 295,329		
Trainable params: 295,329		
Non-trainable params: 0		

**Gambar 7.238** Hasil Nomor 14

```

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

```

Hasilnya bisa dilihat pada gambar berikut :

```

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: 1.7049 - acc: 0.5843
Epoch 2/10
- 239s - loss: 1.0797 - acc: 0.7064
Epoch 3/10
- 235s - loss: 0.9648 - acc: 0.7308
Epoch 4/10
- 237s - loss: 0.9960 - 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
- 236s - 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 0x14c1041f5fb>

```

**Gambar 7.239** Hasil Nomor 15

## 16. Nomor 16

```

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

```

Hasilnya bisa dilihat pada gambar berikut :

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

**Gambar 7.240** Hasil Nomor 16

## 17. Nomor 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_)

```

Hasilnya bisa dilihat pada gambar berikut :

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

**Gambar 7.241** Hasil Nomor 17

## 18. Nomor 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 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())

```

Hasilnya bisa dilihat pada gambar berikut :

```

In [24]: import keras.models
... model2 = keras.models.load_model("mathsymbols.model")
... print(model2.summary())
Layer (Type)          Output Shape       Param #
=================================================================
conv2d_3 (Conv2D)     (None, 38, 38, 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: 289,328
Trainable params: 285,329
Non-trainable params: 0
None

```

**Gambar 7.242** Hasil Nomor 18

## 19. Nomor 19

```

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 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     # 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
      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
18     # reverse the one-hot encoding
19     # Menginisiasi variabel inverted dengan label encoder2 dan
20     # menggunakan argmax untuk mencari skor luaran tertinggi
21     inverted = label_encoder2.inverse_transform([np.argmax(
22         prediction)])
23     # mencetak prediksi gambar dan confidence dari gambar.
24     print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
25         max(prediction)))

```

Hasilnya bisa dilihat pada gambar berikut :

```

In [25]: label_encoder2 = LabelEncoder()
...: classes = np.load('classes.npy')
...: img_path = 'HASYv2/hasy-data/v2-00010.png'
...: img = Image.open(img_path)
...: ksize=3
...: kernel = np.ones((ksize,ksize),np.float32)/ksize
...: blurred = cv2.filter2D(np.array(img),-1,kernel)
...: blurred = np.array(blurred)
...: blurred = blurred/255.0
...: # do the prediction
...: prediction = model2.predict(blurred.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)])
...: np.max(prediction)
...: print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
...: max(prediction)))

```

**Gambar 7.243** Hasil Nomor 19

20. Nomor 20

```

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

```

Hasilnya bisa dilihat pada gambar berikut :

```

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 7.244** Hasil Nomor 20

### 7.10.3 Penanganan Error

#### 1. ScreenShoot Error

```

File "C:\ProgramData\Anaconda3\lib\site-packages\keras\backend
\tensorflow_backend.py", line 5, in <module>
    import tensorflow as tf
ModuleNotFoundError: No module named 'tensorflow'

```

**Gambar 7.245** Module Not Found Error

## 2. Tuliskan Kode Error dan Jenis Error

- Module Not Found Error

## 3. Cara Penanganan Error

- Module Not Found Error

Error terjadi karena belum menginstal package atau library tensorflow Untuk mengatasinya dengan mengisntal library tensorflow pada anaconda

### 7.10.4 Bukti Tidak Plagiat

**Gambar 7.246** Bukti Plagiarisme

### 7.10.5 Link Youtube

<https://youtu.be/5jv7jHYu7pg>

## 7.11 1174083 - Bakti Qilan Mufid

Chapter 7 - CNN

### 7.11.1 Teori

#### 7.11.1.1 Jelaskan kenapa file teks harus dilakukan tokenizer. dilengkapi dengan ilustrasi atau gambar.

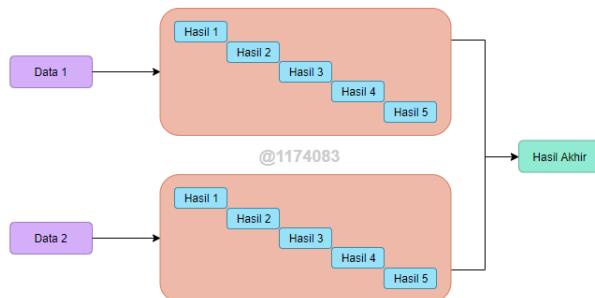
Karena untuk memudahkan mesin memahami maksud dari apa yang user inginkan. Dalam teks file, kata disebut token, dan proses vektorisasi dari bentuk kata kedalam bentuk token tersebut disebut tokenizer dan tokenizer akan merubah sebuah teks menjadi simbol, kata, ataupun biner dan bentuk lainnya kedalam token. Untuk ilustrasinya bisa seperti berikut, terdapat sebuah kalimat a=yang akan dilakukan tokenizer kalimatnya: "jangan panggil aku anak kecil paman" maka ketika dilakukan tokenizer hasilnya akan menjadi ['jangan','panggil','aku','anak','kecil','paman'].

### 7.11.1.2 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 splits = kfold.split(d, d['CLASS'])
```

**Listing 7.7** K-fold Cross Validation

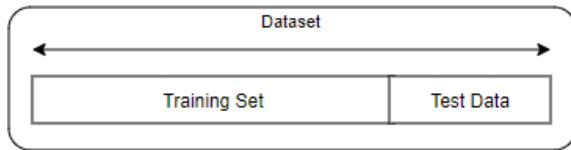
Konsep sederhana dari K-Fold Cross Validation pada code tersebut ialah terdapat kfold yang bertujuan untuk melakukan split data menjadi 5 bagian dari dataset komentar Youtube tersebut. Sehingga dari setiap data yang sudah dibagi tersebut akan menghasilkan presentase dari setiap bagiannya, untuk menghasilkan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



**Gambar 7.247** gambaran penjelasan no. 2

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

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.



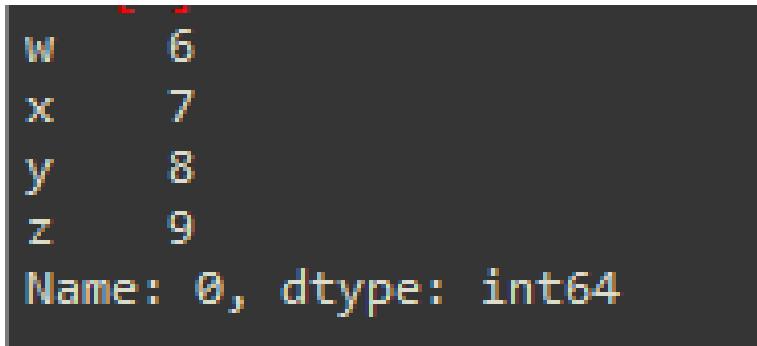
**Gambar 7.248** gambaran penjelasan no. 3

### 7.11.1.4 jelaskan maksudnya kode program

Maksudnya yaitu mengambil data pada kolom atau index CONTENT yang merupakan bagian dari train\_idx dan test\_idx. Ilustrasinya, ketika data telah diubah menjadi train dan test maka kita dapat memil-

ihnya untuk ditampilkan pada kolom yang diinginkan. Untuk ilustrasinya ada pada kode berikut:

```
1 import pandas as pd
2
3 mydict = [{ 'w': 6, 'x': 7, 'y': 8, 'z': 9},
4             { 'w': 600, 'x': 700, 'y': 800, 'z': 900},
5             { 'w': 6000, 'x': 7000, 'y': 8000, 'z': 9000 }]
6 df = pd.DataFrame(mydict)
7 df
8
9 type(df.iloc[0])
10 df.iloc[0]
```

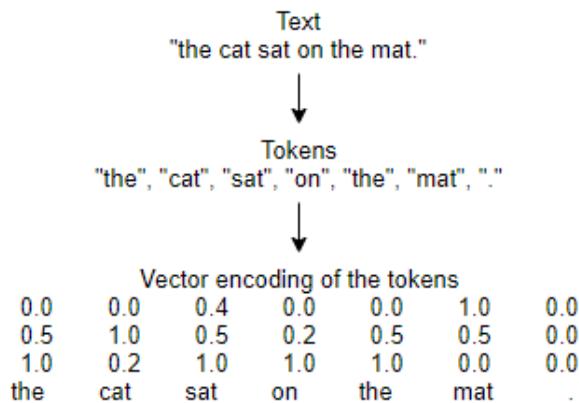


```
w    6
x    7
y    8
z    9
Name: 0, dtype: int64
```

Gambar 7.249 gambaran penjelasan no. 4

#### 7.11.1.5 Jelaskan apa maksud dari fungsi dilengkapi dengan ilustrasi atau gambar

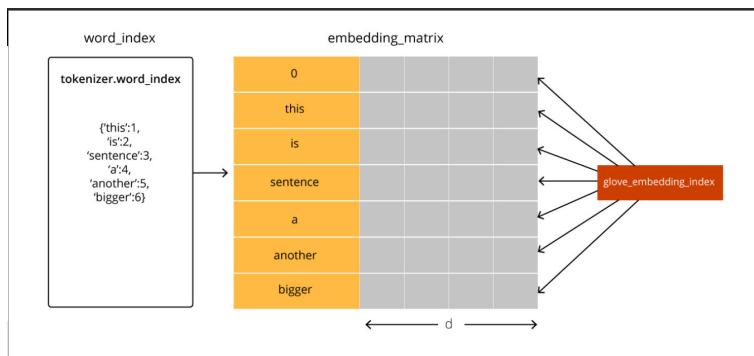
Fungsi tokenizer ini berfungsi untuk melakukan vektorisasi data kedalam bentuk token sebanyak 2000 kata. Dan selanjutnya akan melakukan fit tokenizer hanya untuk data training saja tidak dengan data testingnya. Untuk ilustrasi lihat gambar berikut:



Gambar 7.250 gambaran penjelasan no. 5

#### 7.11.1.6 Jelaskan apa maksud dari dilengkapi dengan ilustrasi kode dan atau gambar

Maksudnya yaitu untuk variabel d\_train\_inputs akan melakukan tokenizer dari bentuk teks ke matrix dari data train\_content dengan mode matriksnya yaitu tfidf begitu juga dengan variabel d\_test\_inputs untuk data test.



Gambar 7.251 gambaran penjelasan no. 6

#### 7.11.1.7 Jelaskan apa maksud dari fungsi dilengkapi dengan ilustrasi atau gambar

Fungsi tersebut akan membagi matrix tfidf tadi dengan np.amax yaitu mengembalikan maksimum array atau maksimum sepanjang sumbu. Jika sumbu tidak ada, hasilnya adalah nilai skalar. Jika sumbu diberikan, hasilnya adalah array dimensi a.ndim - 1. Yang hasilnya akan dimasukan kedalam variabel d\_train\_inputs untuk data train dan d\_test\_inputs untuk data test dengan nominal absolut atau tanpa ada bilangan negatif dan koma.

```
>>> import numpy as np
>>> x = np.array([-5, 6, 1.2])
>>> np.amax(np.absolute(x))
6.0
>>> x = np.array([-5, 6, 1.2, -8])
>>> np.amax(np.absolute(x))
8.0
```

Gambar 7.252 gambaran penjelasan no. 7

**7.11.1.8 Jelaskan apa maksud dari dilengkapi dengan ilustrasi atau gambar**

Dalam variabel d\_train\_output dan d\_test\_outputs akan dilakukan one-hot-encoding, dimana np\_utils akan mengubah vektor dengan bentuk integer ke matriks kelas biner untuk kolom CLASS dimana nantinya hanya akan ada dua pilihan yaitu 1 atau 0. 1 untuk spam 0 untuk non spam atau sebaliknya.

```
>>> bakti = np.array([0,2,1,2,0])
>>> to_categorical(bakti)
array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.]], dtype=float32)
```

Gambar 7.253 gambaran penjelasan no. 8

**7.11.1.9 Jelaskan apa maksud dari fungsi di listing 7.2. Gambar Neural Networknya dari model kode tersebut**

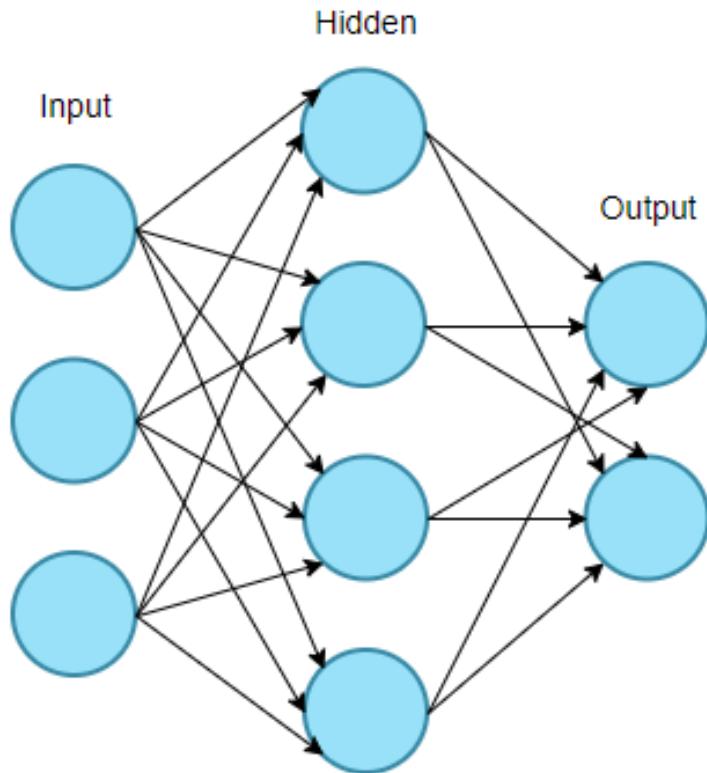
```
1 model = Sequential()
2 model.add(Dense(512, input_shape=(2000,)))
3 model.add(Activation('relu'))
4 model.add(Dropout(0.5))
5 model.add(Dense(2))
6 model.add(Activation('softmax'))
```

Listing 7.8 model Neural Network

Penjelasannya sebagai berikut :

- Melakukan pemodelan Sequential

- Layer pertama dense dari 512 neuron untuk inputan dengan inputan tadi yang sudah dijadikan matriks sebanyak 2000
- Activationnya menggunakan fungsi 'relu' yaitu jika ada inputan dengan nilai maksimum maka inputan itu yang akan terpilih.
- Dropout ini untuk melakukan pembobotan, dimana pembobotan hanya dilakukan 50% saja agar tidak terjadi penumpukan data dari dense inputan tadi
- Dense 2 mengkategorikan 2 neuron untuk output nya yaitu 1 dan 0.
- Untuk dense diatas aktivasinya menggunakan fungsi Softmax.



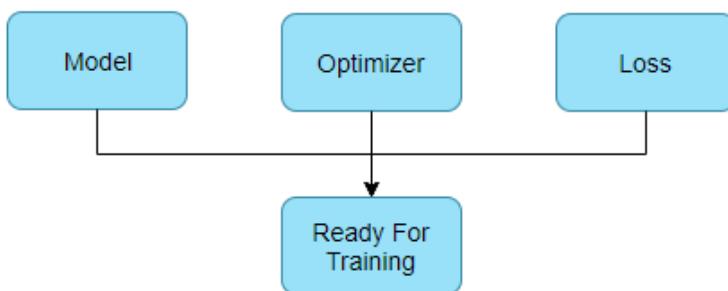
**Gambar 7.254** gambaran penjelasan no. 9

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

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

**Listing 7.9** compile model

Melakukan peng compile-an dari model Sequential tadi dengan Loss yang dengang merupakan fungsi optimisasi skor menggunakan categorical crossentropy , dan menggunakan algoritma adam sebagai optimizer. Adam yaitu algoritma pengoptimalan yang dapat digunakan sebagai ganti dari prosedur penurunan gradien stokastik klasik untuk memperbarui bobot jaringan yang berulang berdasarkan data training.Dengan metrik yaitu fungsi yang digunakan untuk menilai kinerja mode Anda disini menggunakan fungsi accuracy.

**Gambar 7.255** gambaran penjelasan no. 9

#### **7.11.1.11 Jelaskan apa itu Deep Learning**

Deep Learning adalah subbidang machine learning yang berkaitan dengan algoritma yang terinspirasi oleh struktur dan fungsi otak yang disebut jaringan saraf tiruan atau Artificial Neural Networks. Jaringan saraf tiruan, algoritma yang terinspirasi oleh otak manusia, belajar dari sejumlah besar data. Demikian pula dengan bagaimana kita belajar dari pengalaman, algoritma pembelajaran yang mendalam akan melakukan tugas berulang kali, setiap kali sedikit mengubahnya untuk meningkatkan hasilnya.

#### **7.11.1.12 Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning**

Deep Neural Network adalah jaringan syaraf tiruan (JST) dengan beberapa lapisan antara lapisan input dan output. DNN menemukan manipulasi matematis yang benar untuk mengubah input menjadi output, apakah itu hubungan linear atau hubungan non-linear. Merupakan jaringan syaraf dengan tingkat kompleksitas tertentu, jaringan syaraf dengan lebih dari dua lapisan. Deep Neural Network menggunakan pemodelan matematika yang canggih untuk memproses data dengan cara yang kompleks.

DNN hanya terdiri dari dua laipsan yaitu input dan output, sedangkan dalam Deep learning kita dapat mendefiniskan layer sebanyak yang kita inginkan atau butuhkan.

**7.11.1.13 Jelaskan dengan ilustrasi gambar buatan sendiri(langkah per langkah) bagaimana perhitungan algoritma konvolusi dengan ukuran stride (NPM mod 3+1) x (NPM mod 3+1) yang terdapat max pooling**

Stride adalah parameter yang menentukan berapa jumlah pergeseran filter. stride = (NPM mod 3+1) x (NPM mod 3+1) = 1

- disini saya mempunyai data seperti berikut:

1	2	5	2	1	4			
3	2	5	1	1	5			
3	2	4	1	5	3			
5	3	5	1	3	5			
5	1	1	5	5	1			
5	5	4	3	3	1			

0	2	-1						
-1	0	2						
0	1	1						

filter

**Gambar 7.256** gambaran penjelasan no. 9

- Kemudian hitung konvolusi untuk setiap matriksnya seperti berikut :

- pertama

1	2	5	2	1	4	x	0	2	-1	=	12		
3	2	5	1	1	5		-1	0	2				
3	2	4	1	5	3								
5	3	5	1	3	5		0	1	1				
5	1	1	5	5	1								
5	5	4	3	3	1								

**Gambar 7.257** Algoritma Konvolusi

- kedua

1	2	5	2	1	4	x	0	2	-1	=	13		
3	2	5	1	1	5		-1	0	2				
3	2	4	1	5	3								
5	3	5	1	3	5		0	1	1				
5	1	1	5	5	1								
5	5	4	3	3	1								

**Gambar 7.258** Algoritma Konvolusi

- ketiga

A diagram illustrating a convolution step. On the left is a 6x6 input matrix with values ranging from 1 to 5. In the center is a 3x3 kernel with values 0, 2, -1; -1, 0, 2; and 0, 1, 1. To the right of the multiplication symbol (x) is an equals sign (=). On the far right is a 4x4 feature map. The bottom-right cell of the feature map contains the value 6, which is highlighted in orange. The other cells in the feature map are colored green, orange, and grey.

Gambar 7.259 Algoritma Konvolusi

- keenam

A diagram illustrating a convolution step. On the left is a 6x6 input matrix with values ranging from 1 to 5. In the center is a 3x3 kernel with values 0, 2, -1; -1, 0, 2; and 0, 1, 1. To the right of the multiplication symbol (x) is an equals sign (=). On the far right is a 4x4 feature map. The bottom-right cell of the feature map contains the value 15, which is highlighted in orange. The other cells in the feature map are colored green, orange, and grey.

Gambar 7.260 Algoritma Konvolusi

- kelima

A diagram illustrating a convolution step. On the left is a 6x6 input matrix with values ranging from 1 to 5. In the center is a 3x3 kernel with values 0, 2, -1; -1, 0, 2; and 0, 1, 1. To the right of the multiplication symbol (x) is an equals sign (=). On the far right is a 4x4 feature map. The bottom-right cell of the feature map contains the value 12, which is highlighted in orange. The other cells in the feature map are colored green, orange, and grey.

Gambar 7.261 Algoritma Konvolusi

- keenam

$\times$  =

0	2	-1
-1	0	2
0	1	1

15		
	11	

Gambar 7.262 Algoritma Konvolusi

- ketujuh

$\times$  =

0	2	-1
-1	0	2
0	1	1

11		

Gambar 7.263 Algoritma Konvolusi

- kedelapan

$\times$  =

0	2	-1
-1	0	2
0	1	1

10		

Gambar 7.264 Algoritma Konvolusi

- kesembilan

$$\begin{array}{|c|c|c|c|c|c|c|} \hline
 1 & 2 & 5 & 2 & 1 & 4 \\ \hline
 3 & 2 & 5 & 1 & 1 & 5 \\ \hline
 3 & 2 & 4 & 1 & 5 & 3 \\ \hline
 5 & 3 & 5 & 1 & 3 & 5 \\ \hline
 5 & 1 & 1 & 5 & 5 & 1 \\ \hline
 5 & 5 & 4 & 3 & 3 & 1 \\ \hline
 \end{array}
 \times
 \begin{array}{|c|c|c|} \hline
 0 & 2 & -1 \\ \hline
 -1 & 0 & 2 \\ \hline
 0 & 1 & 1 \\ \hline
 \end{array}
 =
 \begin{array}{|c|c|c|} \hline
 \text{green} & \text{green} & \text{yellow} \\ \hline
 \text{orange} & 7 & \text{grey} \\ \hline
 \text{grey} & \text{grey} & \text{grey} \\ \hline
 \end{array}$$

Gambar 7.265 Algoritma Konvolusi

- kesepuluh

$$\begin{array}{|c|c|c|c|c|c|c|} \hline
 1 & 2 & 5 & 2 & 1 & 4 \\ \hline
 3 & 2 & 5 & 1 & 1 & 5 \\ \hline
 3 & 2 & 4 & 1 & 5 & 3 \\ \hline
 5 & 3 & 5 & 1 & 3 & 5 \\ \hline
 5 & 1 & 1 & 5 & 5 & 1 \\ \hline
 5 & 5 & 4 & 3 & 3 & 1 \\ \hline
 \end{array}
 \times
 \begin{array}{|c|c|c|} \hline
 0 & 2 & -1 \\ \hline
 -1 & 0 & 2 \\ \hline
 0 & 1 & 1 \\ \hline
 \end{array}
 =
 \begin{array}{|c|c|c|} \hline
 \text{green} & \text{green} & \text{yellow} \\ \hline
 \text{orange} & 12 & \text{grey} \\ \hline
 \text{grey} & \text{grey} & \text{grey} \\ \hline
 \end{array}$$

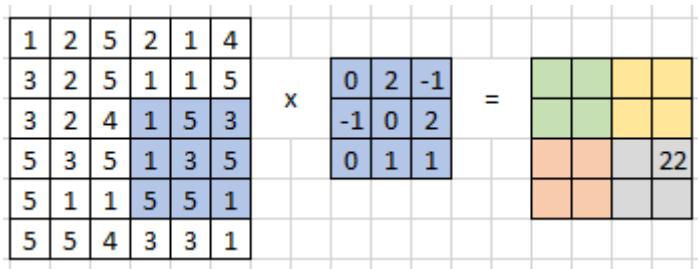
Gambar 7.266 Algoritma Konvolusi

- kesebelas

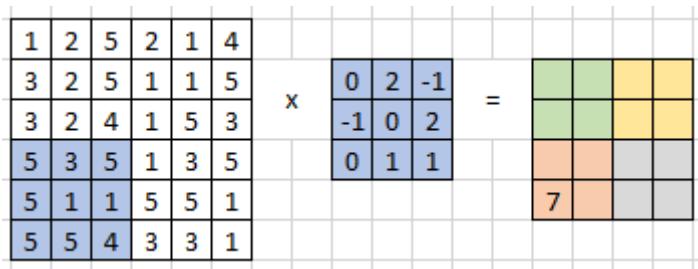
$$\begin{array}{|c|c|c|c|c|c|c|} \hline
 1 & 2 & 5 & 2 & 1 & 4 \\ \hline
 3 & 2 & 5 & 1 & 1 & 5 \\ \hline
 3 & 2 & 4 & 1 & 5 & 3 \\ \hline
 5 & 3 & 5 & 1 & 3 & 5 \\ \hline
 5 & 1 & 1 & 5 & 5 & 1 \\ \hline
 5 & 5 & 4 & 3 & 3 & 1 \\ \hline
 \end{array}
 \times
 \begin{array}{|c|c|c|} \hline
 0 & 2 & -1 \\ \hline
 -1 & 0 & 2 \\ \hline
 0 & 1 & 1 \\ \hline
 \end{array}
 =
 \begin{array}{|c|c|c|} \hline
 \text{green} & \text{green} & \text{yellow} \\ \hline
 \text{orange} & \text{orange} & 8 \\ \hline
 \text{grey} & \text{grey} & \text{grey} \\ \hline
 \end{array}$$

Gambar 7.267 Algoritma Konvolusi

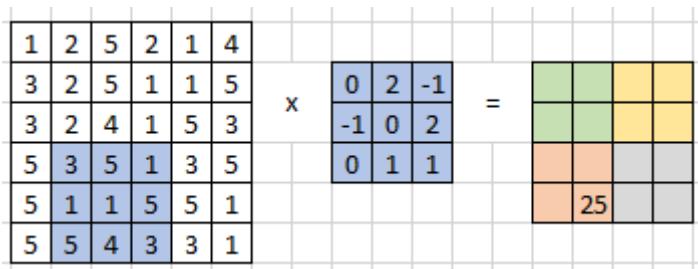
- keduabelas

**Gambar 7.268** Algoritma Konvolusi

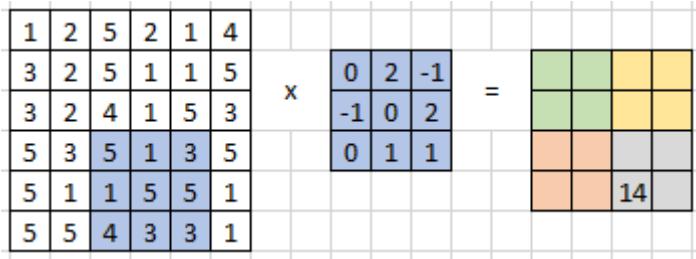
- ketigabelas

**Gambar 7.269** Algoritma Konvolusi

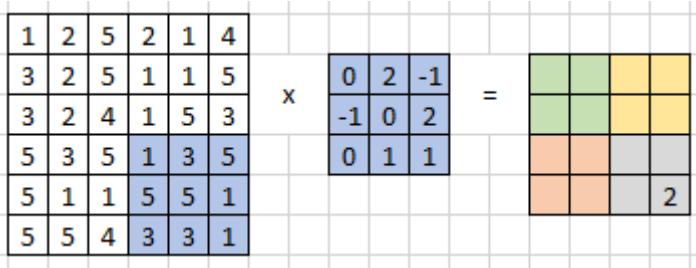
- keempatbelas

**Gambar 7.270** Algoritma Konvolusi

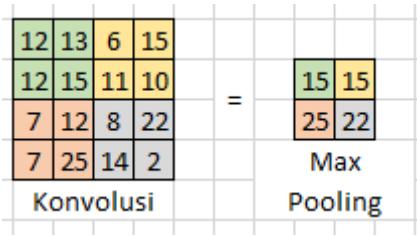
- kelimabelas

**Gambar 7.271** Algoritma Konvolusi

- keenambelas

**Gambar 7.272** Algoritma Konvolusi

- Didapatkan hasil akhir nilai konvolusi dan juga max poolingnya seperti berikut:

**Gambar 7.273** Hasil akhir nilai konvolusi dan max pooling

## 7.11.2 Praktek

**7.11.2.1 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**

Penjelasan kode:

```
1 # In[1]: import lib
2 import csv #Import library csv
```

```

3 from PIL import Image as pil_image #Import library Image yaitu fungsi
   PIL (Python Imaging Library) yang berguna untuk mengolah data
   berupa gambar
4 import keras.preprocessing.image #Import library keras yang
   menggunakan method preprocessing yang digunakan untuk membuat
   neural network

```

**Listing 7.10** Kode program blok(1)

Hasil Skrinsut:

```

[ ] # In[1]:import lib
      import keras.models
      import time
      import keras.callbacks
      from keras.layers import Conv2D, MaxPooling2D
      from keras.layers import Dense, Dropout, Flatten
      from keras.models import Sequential
      from sklearn.preprocessing import OneHotEncoder
      from sklearn.preprocessing import LabelEncoder
      import numpy as np
      import random
      import csv
      import tensorflow as tf
      from PIL import Image as pil_image
      import keras.preprocessing.image

⇒ Using TensorFlow backend.

```

**Gambar 7.274** Hasil Skrinsut

#### 7.11.2.2 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

Penjelasan kode:

```

1 # In[2]:load all images (as numpy arrays) and save their classes
2 imgs = [] #Membuat variabel imgs
3 classes = [] #Membuat variabel classes dengan variabel kosong
4 with open('HASYv2/hasy-data-labels.csv') as csvfile: #Membuka file
      hasy-data-labels.csv
5     csvreader = csv.reader(csvfile) #Membuat variabel csvreader yang
      berisi method csv.reader untuk membaca csvfile
6     i = 0 # membuat variabel i dengan isi 0
7     for row in csvreader: # Membuat looping pada variabel csvreader
8         if i > 0: #Ketentuannya jika i lebih kecil daripada 0
9             img = keras.preprocessing.image.img_to_array(pil_image.
open("N:/HASYv2/" + row[0])) #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
10            # neuron activation functions behave best when input
values are between 0.0 and 1.0 (or -1.0 and 1.0),
11            # so we rescale each pixel value to be in the range 0.0
to 1.0 instead of 0-255
12            img /= 255.0 #Membagi data yang ada pada fungsi img
sebanyak 255.0
13            imgs.append((row[0], row[2], img)) #Menambah nilai baru
pada imgs pada row ke 1 2 dan dilanjutkan dengan variabel img

```

```

14     classes.append(row[2]) #Menambahkan nilai pada row ke 2
    pada variabel classes
15     i += 1 #Menambah nilai satu pada variabel i

```

**Listing 7.11** Kode program blok(2)

Hasil Skrinsut:

```
[ ] # In[2]:load all images (as numpy arrays) and save their classes

imgs = []
classes = []
with open('HASV2/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("HASV2/" + 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 7.275** Hasil Skrinsut

**7.11.2.3 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**

Penjelasan kode:

```

1 # In [3]: shuffle the data , split into 80% train , 20% test
2 import random #Mengimport library random
3 random.shuffle(imgs) #Melakukan randomize pada fungsi imgs
4 split_idx = int(0.8*len(imgs)) #Membuat variabel split_idx dengan
    nilai integer 80 persen dikali dari pengembalian jumlah dari
    variabel imgs
5 train = imgs[:split_idx] #Membuat variabel train dengan isi sebelum
    split_idx
6 test = imgs[split_idx:] #Membuat variabel test dengan isi setelah
    split_idx
```

**Listing 7.12** Kode program blok(3)

Hasil Skrinsut:

```
[ ] # In[3]:shuffle the data, split into 80% train, 20% test

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

**Gambar 7.276** Hasil Skrinsut

**7.11.2.4 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***

Penjelasan kode:

```

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

```

**Listing 7.13** Kode program blok(4)

Hasil Skrinsut:

```

[ ] # In[4]:
    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 7.277** Hasil Skrinsut

#### ***7.11.2.5 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***

Penjelasan kode:

```

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

```

**Listing 7.14** Kode program blok(5)

Hasil Skrinsut:

```

[5] from sklearn.preprocessing import LabelEncoder #Mengimport library LabelEncoder dari sklearn
    from sklearn.preprocessing import OneHotEncoder #Mengimport library OneHotEncoder dari sklearn

```

**Gambar 7.278** Hasil Skrinsut

#### ***7.11.2.6 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***

Penjelasan kode:

```

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

```

**Listing 7.15** Kode program blok(6)

Hasil Skrinsut:

```

[ ] # In[6]:convert class names into one-hot encoding
# first, convert class names into integers
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(classes)

```

**Gambar 7.279** Hasil Skrinsut

**7.11.2.7 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**

Penjelasan kode:

```

1 # In[4]:
2 import numpy as np #Mengimport library numpy dengan inisial np
3 train_input = np.asarray(list(map(lambda row: row[2], train))) #
    Membuat variabel train_input dengan np method asarray yang mana
    membuat array dengan isi row 2 dari data train
4 test_input = np.asarray(list(map(lambda row: row[2], test))) #Membuat
    test_input input dengan np method asarray yang mana membuat
    array dengan isi row 2 dari data test
5 train_output = np.asarray(list(map(lambda row: row[1], train))) #
    Membuat variabel train_output dengan np method asarray yang mana
    membuat array dengan isi row 1 dari data train
6 test_output = np.asarray(list(map(lambda row: row[1], test))) #
    Membuat variabel test_output dengan np method asarray yang mana
    membuat array dengan isi row 1 dari data test

```

**Listing 7.16** Kode program blok(7)

Hasil Skrinsut:

```

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

[ ] OneHotEncoder(categories='auto', drop=None, dtype<class 'numpy.float64'>,
    handle_unknown='error', sparse=False)

```

**Gambar 7.280** Hasil Skrinsut

**7.11.2.8 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**

Penjelasan kode:

```

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

```

**Listing 7.17** Kode program blok(8)

Hasil Skrinsut:

```

[ ] # In[8]:convert train and test output to one-hot
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 7.281** Hasil Skrinsut

**7.11.2.9 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**

Penjelasan kode:

```

1 # In [9]: import sequential
2 from keras.models import Sequential #Mengimport library Sequential
    dari Keras
3 from keras.layers import Dense, Dropout, Flatten #Mengimport library
    Dense, Dropout, Flatten dari Keras
4 from keras.layers import Conv2D, MaxPooling2D #Mengimport library
    Conv2D, MaxPooling2D dari Keras
5 #import tensorflow #Import tensorflow

```

**Listing 7.18** Kode program blok(9)

Hasil Skrinsut:

```
[9] from keras.models import Sequential #Mengimport library Sequential
from keras.layers import Dense, Dropout, Flatten #
from keras.layers import Conv2D, MaxPooling2D #Menambahkan library Conv2D dan MaxPooling2D
#Import tensorflow #Import tensorflow
```

Gambar 7.282 Hasil Skrinsut

**7.11.2.10 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**

Penjelasan kode:

```
1 # In [10]: desain jaringan
2 model = Sequential() #Membuat variabel model dengan isian library Sequential
3 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
4                 input_shape=np.shape(train_input[0]))) #Variabel model di tambahkan library Conv2D tigapuluhan dua bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan relu yang menggunakan data train_input
5 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x 2 pixel
6 model.add(Conv2D(32, (3, 3), activation='relu')) #Variabel model di tambahkan dengan library Conv2D 32bit dengan kernel 3 x 3
7 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x 2 pixel
8 model.add(Flatten()) #Variabel model di tambahkan library Flatten
9 model.add(Dense(1024, activation='tanh')) #Variabel model di tambahkan library Dense dengan fungsi tanh
10 model.add(Dropout(0.5)) #Variabel model di tambahkan library dropout untuk memangkas data tree sebesar 50 persen
11 model.add(Dense(num_classes, activation='softmax')) #Variabel model di tambahkan library Dense dengan data dari num_classes dan fungsi softmax
12 model.compile(loss='categorical_crossentropy', optimizer='adam',
13                 metrics=['accuracy']) #Mengkompile data model untuk mendapatkan data loss akurasi dan optimasi
14 print(model.summary()) #Mencetak variabel model kemudian memunculkan kesimpulan berupa data total parameter, trainable parameter dan bukan trainable parameter
```

Listing 7.19 Kode program blok(10)

Hasil Skrinsut:

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_1 (MaxPooling2)	(None, 6, 6, 32)	0
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 1024)	1180672
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 369)	378225
<hr/>		
Total params: 1,569,041		
Trainable params: 1,569,041		
Non-trainable params: 0		
<hr/>		
None		

Gambar 7.283 Hasil Skrinsut

**7.11.2.11 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**

Penjelasan kode:

```
1 # In[11]: import sequential
2 import keras.callbacks #Mengimport library keras dengan fungsi
3           callbacks
4 tensorboard = keras.callbacks.TensorBoard(log_dir='hasyyv2/logs/mnist-
5           style') #Membuat variabel tensorboard dengan isi lib keras
```

Listing 7.20 Kode program blok(11)

Hasil Skrinsut:

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

Gambar 7.284 Hasil Skrinsut

**7.11.2.12 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**

Penjelasan kode:

```
1 # In[12]: 5menit kali 10 epoch = 50 menit
2 model.fit(train_input, train_output, #Fungsi model ditambahkan fungsi
3           fit untuk mengetahui perhitungan dari train_input train_output
4           batch_size=32, #Dengan batch size 32 bit
5           epochs=10,
6           verbose=2,
7           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])

```

**Listing 7.21** Kode program blok(12)

Hasil Skrinsut:

```

Epoch 1/10
3365/3365 - 100s - loss: 1.5393 - accuracy: 0.6264 - val_loss: 0.9766 - val_accuracy: 0.7349
Epoch 2/10
3365/3365 - 102s - loss: 0.9562 - accuracy: 0.7345 - val_loss: 0.8892 - val_accuracy: 0.7492
Epoch 3/10
3365/3365 - 103s - loss: 0.8443 - accuracy: 0.7575 - val_loss: 0.8475 - val_accuracy: 0.7618
Epoch 4/10
3365/3365 - 103s - loss: 0.7695 - accuracy: 0.7730 - val_loss: 0.8456 - val_accuracy: 0.7639
Epoch 5/10
3365/3365 - 103s - loss: 0.7142 - accuracy: 0.7864 - val_loss: 0.8427 - val_accuracy: 0.7693
Epoch 6/10
3365/3365 - 104s - loss: 0.6754 - accuracy: 0.7931 - val_loss: 0.8405 - val_accuracy: 0.7782
Epoch 7/10
3365/3365 - 103s - loss: 0.6396 - accuracy: 0.8013 - val_loss: 0.8521 - val_accuracy: 0.7634
Epoch 8/10
3365/3365 - 103s - loss: 0.6130 - accuracy: 0.8073 - val_loss: 0.8747 - val_accuracy: 0.7610
Epoch 9/10
3365/3365 - 104s - loss: 0.5872 - accuracy: 0.8117 - val_loss: 0.8675 - val_accuracy: 0.7655
Epoch 10/10
3365/3365 - 103s - loss: 0.5673 - accuracy: 0.8176 - val_loss: 0.8994 - val_accuracy: 0.7681
1052/1052 - 9s - loss: 0.8918 - accuracy: 0.7703
Test loss: 0.8918401002883911
Test accuracy: 0.770321249961853

```

**Gambar 7.285** Hasil Skrinsut

**7.11.2.13 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**

Penjelasan kode:

```

1 # In[13]: try various model configurations and parameters to find the
2 # best
3 import time #Mengimport library time
4 results = [] #Membuat variabel result dengan array kosong
5 for conv2d_count in [1, 2]: #Melakukan looping dengan ketentuan
6     konvolusi 2 dimensi 1
7     for dense_size in [128, 256, 512, 1024, 2048]: #Menentukan ukuran
8         besaran fixcel dari data atau konvert 1 fixcel mnjadi data yang
9         berada pada codigan dibawah.
10        for dropout in [0.0, 0.25, 0.50, 0.75]: #Membuat looping
11            untuk memangkas masing-masing data dengan ketentuan 0 persen 25
12            persen 50 persen dan 75 persen.
13            model = Sequential() #Membuat variabel model Sequential
14            for i in range(conv2d_count): #Membuat looping untuk
15                variabel i dengan jarak dari hasil konvolusi.
16                if i == 0: #Syarat jika i samadengan bobotnya 0
17                    model.add(Conv2D(32, kernel_size=(3, 3),
18                                     activation='relu', input_shape=np.shape(train_input[0])))
19 Menambahkan method add pada variabel model dengan konvolusi 2
20 dimensi 32 bit didalamnya dan membuat kernel dengan ukuran 3 x 3
21 dan rumus aktifasi relu dan data shape yang di hitung dari data
22 train .

```

```

11         else: #Jika tidak
12             model.add(Conv2D(32, kernel_size=(3, 3),
13                               activation='relu')) #Menambahkan method add pada variabel model
14             dengan konvolusi 2 dimensi 32 bit dengan ukuran kernel 3x3 dan
15             fungsi aktivasi relu
16             model.add(MaxPooling2D(pool_size=(2, 2))) #
17             Menambahkan method add pada variabel model dengan isian method
18             Max pooling berdimensi 2 dengan ukuran fixcel 2 x 2.
19             model.add(Flatten()) #Merubah feature gambar menjadi 1
20             dimensi vektor
21             model.add(Dense(dense_size, activation='tanh')) #
22             Menambahkan method dense untuk pemadatan data dengan ukuran dense
23             di tentukan dengan rumus fungsi tanh.
24             if dropout > 0.0: #Membuat ketentuan jika pemangkasan
25             lebih besar dari 0 persen
26                 model.add(Dropout(dropout)) #Menambahkan method
27             dropout pada model dengan nilai dari dropout
28             model.add(Dense(num_classes, activation='softmax')) #
29             Menambahkan method dense dengan fungsi num classs dan rumus
30             softmax
31             model.compile(loss='categorical_crossentropy', optimizer=
32                           'adam', metrics=['accuracy']) #Mengcompile variabel model dengan
33             hasi loss optimasi dan akurasi matrix
34             log_dir = 'hasyv2/logs/conv2d_%d-dense_%d-dropout_.2f' %
35             (conv2d_count, dense_size, dropout) #Melakukan log
36             tensorboard = keras.callbacks.TensorBoard(log_dir=log_dir)
37             # membuat variabel tensorboard dengan isian dari library keras
38             dan nilai dari log_dir
39
40             start = time.time() #Membuat variabel start dengan isian
41             dari library time menggunakan method time
42             model.fit(train_input, train_output, batch_size=32,
43             epochs=10,
44             verbose=0, validation_split=0.2, callbacks=[tensorboard]) #Menambahkan method fit pada model dengan data dari
45             train input train output nilai batch nilai epoch verbose nilai
46             20 persen validation split dan callback dengan nilai tensorboard.
47             score = model.evaluate(test_input, test_output, verbose
48             =2) #Membuat variabel score dengan nilai evaluasi dari model
49             menggunakan data tes input dan tes output
50             end = time.time() #Membuat variabel end
51             elapsed = end - start #Membuat variabel elapsed
52             print("Conv2D count: %d, Dense size: %d, Dropout: %.2f -
53             Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (conv2d_count,
54             dense_size, dropout, score[0], score[1], elapsed)) #Mencetak
55             hasil perhitungan
56             results.append((conv2d_count, dense_size, dropout, score
57             [0], score[1], elapsed))

```

**Listing 7.22** Kode program blok(13)

Hasil Skrinsut:

```
[ ] 1052/1052 - 6s - loss: 1.2295 - accuracy: 0.7344
Conv2D count: 1, Dense size: 128, Dropout: 0.00 - Loss: 1.23, Accuracy: 0.73, Time: 556 sec
1052/1052 - 6s - loss: 0.9560 - accuracy: 0.7614
Conv2D count: 1, Dense size: 128, Dropout: 0.25 - Loss: 0.96, Accuracy: 0.76, Time: 574 sec
1052/1052 - 7s - loss: 0.8298 - accuracy: 0.7737
Conv2D count: 1, Dense size: 128, Dropout: 0.50 - Loss: 0.83, Accuracy: 0.77, Time: 567 sec
1052/1052 - 7s - loss: 0.8140 - accuracy: 0.7693
Conv2D count: 1, Dense size: 128, Dropout: 0.75 - Loss: 0.81, Accuracy: 0.77, Time: 592 sec
1052/1052 - 7s - loss: 1.3661 - accuracy: 0.7432
Conv2D count: 1, Dense size: 256, Dropout: 0.00 - Loss: 1.37, Accuracy: 0.74, Time: 722 sec
1052/1052 - 7s - loss: 1.1462 - accuracy: 0.7532
Conv2D count: 1, Dense size: 256, Dropout: 0.25 - Loss: 1.15, Accuracy: 0.75, Time: 726 sec
1052/1052 - 7s - loss: 0.9360 - accuracy: 0.7685
Conv2D count: 1, Dense size: 256, Dropout: 0.50 - Loss: 0.94, Accuracy: 0.77, Time: 728 sec
1052/1052 - 7s - loss: 0.7998 - accuracy: 0.7783
Conv2D count: 1, Dense size: 256, Dropout: 0.75 - Loss: 0.80, Accuracy: 0.78, Time: 734 sec
1052/1052 - 9s - loss: 1.6735 - accuracy: 0.7346
Conv2D count: 1, Dense size: 512, Dropout: 0.00 - Loss: 1.67, Accuracy: 0.73, Time: 1057 sec
1052/1052 - 8s - loss: 1.5223 - accuracy: 0.7476
Conv2D count: 1, Dense size: 512, Dropout: 0.25 - Loss: 1.52, Accuracy: 0.75, Time: 1049 sec
1052/1052 - 9s - loss: 1.3185 - accuracy: 0.7458
Conv2D count: 1, Dense size: 512, Dropout: 0.50 - Loss: 1.32, Accuracy: 0.75, Time: 1091 sec
1052/1052 - 9s - loss: 0.8951 - accuracy: 0.7719
Conv2D count: 1, Dense size: 512, Dropout: 0.75 - Loss: 0.90, Accuracy: 0.77, Time: 1046 sec
1052/1052 - 13s - loss: 2.5109 - accuracy: 0.7134
Conv2D count: 1, Dense size: 1024, Dropout: 0.00 - Loss: 2.51, Accuracy: 0.71, Time: 2283 sec
1052/1052 - 14s - loss: 2.2605 - accuracy: 0.7204
Conv2D count: 1, Dense size: 1024, Dropout: 0.25 - Loss: 2.26, Accuracy: 0.72, Time: 2213 sec
1052/1052 - 13s - loss: 1.6548 - accuracy: 0.7309
Conv2D count: 1, Dense size: 1024, Dropout: 0.50 - Loss: 1.65, Accuracy: 0.73, Time: 2222 sec
1052/1052 - 13s - loss: 1.0859 - accuracy: 0.7448
Conv2D count: 1, Dense size: 1024, Dropout: 0.75 - Loss: 1.09, Accuracy: 0.74, Time: 2182 sec
1052/1052 - 19s - loss: 2.2024 - accuracy: 0.7027
Conv2D count: 1, Dense size: 2048, Dropout: 0.00 - Loss: 2.20, Accuracy: 0.70, Time: 3958 sec
1052/1052 - 19s - loss: 2.2918 - accuracy: 0.6986
Conv2D count: 1, Dense size: 2048, Dropout: 0.25 - Loss: 2.29, Accuracy: 0.70, Time: 3974 sec
1052/1052 - 20s - loss: 1.7761 - accuracy: 0.7164
Conv2D count: 1, Dense size: 2048, Dropout: 0.50 - Loss: 1.78, Accuracy: 0.72, Time: 4153 sec
1052/1052 - 19s - loss: 1.2051 - accuracy: 0.7334
Conv2D count: 1, Dense size: 2048, Dropout: 0.75 - Loss: 1.21, Accuracy: 0.73, Time: 4160 sec
1052/1052 - 8s - loss: 0.8711 - accuracy: 0.7643
Conv2D count: 2, Dense size: 128, Dropout: 0.00 - Loss: 0.87, Accuracy: 0.76, Time: 704 sec
1052/1052 - 8s - loss: 0.7883 - accuracy: 0.7782
Conv2D count: 2, Dense size: 128, Dropout: 0.25 - Loss: 0.79, Accuracy: 0.78, Time: 709 sec
1052/1052 - 8s - loss: 0.7600 - accuracy: 0.7822
Conv2D count: 2, Dense size: 128, Dropout: 0.50 - Loss: 0.76, Accuracy: 0.78, Time: 718 sec
1052/1052 - 8s - loss: 0.8216 - accuracy: 0.7634
Conv2D count: 2, Dense size: 128, Dropout: 0.75 - Loss: 0.82, Accuracy: 0.76, Time: 730 sec
1052/1052 - 8s - loss: 1.0374 - accuracy: 0.7486
Conv2D count: 2, Dense size: 256, Dropout: 0.00 - Loss: 1.04, Accuracy: 0.75, Time: 746 sec
1052/1052 - 8s - loss: 0.8642 - accuracy: 0.7711
Conv2D count: 2, Dense size: 256, Dropout: 0.25 - Loss: 0.86, Accuracy: 0.77, Time: 754 sec
1052/1052 - 8s - loss: 0.7819 - accuracy: 0.7816
Conv2D count: 2, Dense size: 256, Dropout: 0.50 - Loss: 0.78, Accuracy: 0.78, Time: 755 sec
1052/1052 - 8s - loss: 0.7786 - accuracy: 0.7750
Conv2D count: 2, Dense size: 256, Dropout: 0.75 - Loss: 0.78, Accuracy: 0.78, Time: 755 sec
```

Gambar 7.286 Hasil Skrinsut

**7.11.2.14 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**

Penjelasan kode:

```
1 # In[14]: rebuild/retrain a model with the best parameters (from the
  search) and use all data
2 model = Sequential() #Membuat variabel model dengan isian library
  Sequential
3 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
  input_shape=np.shape(train_input[0])))#Variabel model di
  tambahkan library Conv2D tigapuluhan dua bit dengan ukuran kernel 3
  x 3 dan fungsi penghitungan relu dang menggunakan data
  train_input
```

```

4 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x 2 pixel
5 model.add(Conv2D(32, (3, 3), activation='relu')) #Variabel model di tambahkan dengan library Conv2D 32bit dengan kernel 3 x 3
6 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x 2 pixel
7 model.add(Flatten()) #Variabel model di tambahkan library Flatten
8 model.add(Dense(128, activation='tanh')) #Variabel model di tambahkan library Dense dengan fungsi tanh
9 model.add(Dropout(0.5)) #Variabel model di tambahkan library dropout untuk memangkas data tree sebesar 50 persen
10 model.add(Dense(num_classes, activation='softmax')) #Variabel model di tambahkan library Dense dengan data dari num_classes dan fungsi softmax
11 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy']) #Mengkompile data model untuk mendapatkan data loss akurasi dan optimasi
12 print(model.summary()) #Mencetak variabel model kemudian memunculkan kesimpulan berupa data total parameter, trainable paremeter dan bukan trainable parameter

```

**Listing 7.23** Kode program blok(14)

Hasil Skrinsut:

Model: "sequential_42"		
Layer (type)	Output Shape	Param #
conv2d_63 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_63 (MaxPooling)	(None, 15, 15, 32)	0
conv2d_64 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_64 (MaxPooling)	(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)	47601
Total params: 205,329		
Trainable params: 205,329		
Non-trainable params: 0		
None		

**Gambar 7.287** Hasil Skrinsut

**7.11.2.15 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**

Penjelasan kode:

```

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)), #Melakukan
   training yang isi datanya dari join numpy menggunakan data
   train_input test_input
3           np.concatenate((train_output, test_output)), #Kelanjutan
   data yang di gunakan pada join train_output test_output
4           batch_size=32, epochs=10, verbose=2) #Menggunakan ukuran 32
   bit dan epoch 10

```

**Listing 7.24** Kode program blok(15)

Hasil Skrinsut:

```

[+] Epoch 1/10
- 38s - loss: 1.7842 - accuracy: 0.5857
Epoch 2/10
- 39s - loss: 1.0848 - accuracy: 0.7044
Epoch 3/10
- 38s - loss: 0.9728 - accuracy: 0.7290
Epoch 4/10
- 38s - loss: 0.9141 - accuracy: 0.7411
Epoch 5/10
- 38s - loss: 0.8725 - accuracy: 0.7492
Epoch 6/10
- 38s - loss: 0.8460 - accuracy: 0.7551
Epoch 7/10
- 38s - loss: 0.8250 - accuracy: 0.7597
Epoch 8/10
- 38s - loss: 0.8032 - accuracy: 0.7651
Epoch 9/10
- 37s - loss: 0.7919 - accuracy: 0.7654
Epoch 10/10
- 38s - loss: 0.7786 - accuracy: 0.7686
<keras.callbacks.callbacks.History at 0x7fee61ec97b8>

```

**Gambar 7.288** Hasil Skrinsut

**7.11.2.16 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**

Penjelasan kode:

```

1 # In[16]: save the trained model
2 model.save("mathsymbols.model") #Menyimpan model atau mengeksport
   model yang telah di jalan tadi

```

**Listing 7.25** Kode program blok(16)

Hasil Skrinsut:

```
[16] model.save("/content/mathsymbols.model") #Menyimpan model atau mengeksport model yang telah di jalan tadi
```

**Gambar 7.289** Hasil Skrinsut

**7.11.2.17 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**

Penjelasan kode:

```

1 # In[17]: save label encoder (to reverse one-hot encoding)

```

```
2 np.save('classes.npy', label_encoder.classes_) #Menyimpan label
encoder dengan nama classes.npy
```

**Listing 7.26** Kode program blok(17)

Hasil Skrinsut:

```
[17] np.save('/content/classes.npy', label_encoder.classes_) #Menyimpan label encoder dengan nama classes.npy
```

**Gambar 7.290** Hasil Skrinsut

**7.11.2.18 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**

Penjelasan kode:

```
1 # In[18]: load the pre-trained model and predict the math symbol for
2 # an arbitrary image;
3 # the code below could be placed in a separate file
4 import keras.models #Mengimpport library keras model
5 model2 = keras.models.load_model("mathsymbols.model") #Membuat
# variabel model2 untuk meload model yang telah di simpan tadi
5 print(model2.summary()) #Mencetak hasil model2
```

**Listing 7.27** Kode program blok(18)

Hasil Skrinsut:

Model: "sequential_42"		
Layer (type)	Output Shape	Param #
conv2d_63 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_63 (MaxPooling)	(None, 15, 15, 32)	0
conv2d_64 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_64 (MaxPooling)	(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)	47601
<hr/>		
Total params: 205,329		
Trainable params: 205,329		
Non-trainable params: 0		
<hr/>		
None		

**Gambar 7.291** Hasil Skrinsut

**7.11.2.19 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**

Penjelasan kode:

```

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

```

**Listing 7.28** Kode program blok(19)

Hasil Skrinsut:

```

[19] label_encoder2 = LabelEncoder() # membuat variabel label encoder ke 2 dengan isian
label_encoder2.classes_ = np.load('/content/classes.npy') #Menambahkan method clas
def predict(img_path): #Membuat fungsi predict dengan path img
    newimg = keras.preprocessing.image.img_to_array(pil_image.open(img_path)) #Memb
    newimg /= 255.0 #Membagi data yang terdapat pada variabel newimg sebanyak 255

    # do the prediction
    prediction = model2.predict(newimg.reshape(1, 32, 32, 3)) #Membuat variabel pr

    # figure out which output neuron had the highest score, and reverse the one-ho
    inverted = label_encoder2.inverse_transform([np.argmax(prediction)]) #Membuat
    print("Prediction: %s, confidence: %.2f" % (inverted[0], np.max(prediction)))

```

**Gambar 7.292** Hasil Skrinsut

**7.11.2.20 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**

Penjelasan kode:

```

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") #Mencari prediksi
   menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2
   /hasy-data/v2-00010.png

```

```

3 predict("HASYv2/hasy-data/v2-00500.png") #Mencari prediksi
    menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2
    /hasy-data/v2-00500.png
4 predict("HASYv2/hasy-data/v2-00700.png") #Mencari prediksi
    menggunakan fungsi prediksi yang di buat tadi dari data di HASYv2
    /hasy-data/v2-00700.png

```

**Listing 7.29** Kode program blok(20)

Hasil Skrinsut:

```

C> Prediction: A, confidence: 0.74
Prediction: \pi, confidence: 0.48
Prediction: \alpha, confidence: 0.90

```

**Gambar 7.293** Hasil Skrinsut

### 7.11.3 Penanganan Error

#### 7.11.3.1 Terjadi error

- terjadi error invalid argument, seperti pada gambar berikut:

```

OSError: [Errno 22] Invalid argument: 'HASYv2/hasy-data/v2-152218.png'

```

**Gambar 7.294** terjadi Error 1

- terjadi error Sequential has no attribute, seperti pada gambar berikut:

```

super(TensorBoard, self).set_model(model)
File "C:\ProgramData\Anaconda3\lib\site-packages\tensorflow_core\python\keras\callbacks.py", line
1552, in set_model
    self.log_dir, self.model._get_distribution_strategy()) # pylint: disable=protected-access
AttributeError: 'Sequential' object has no attribute '_get_distribution_strategy'

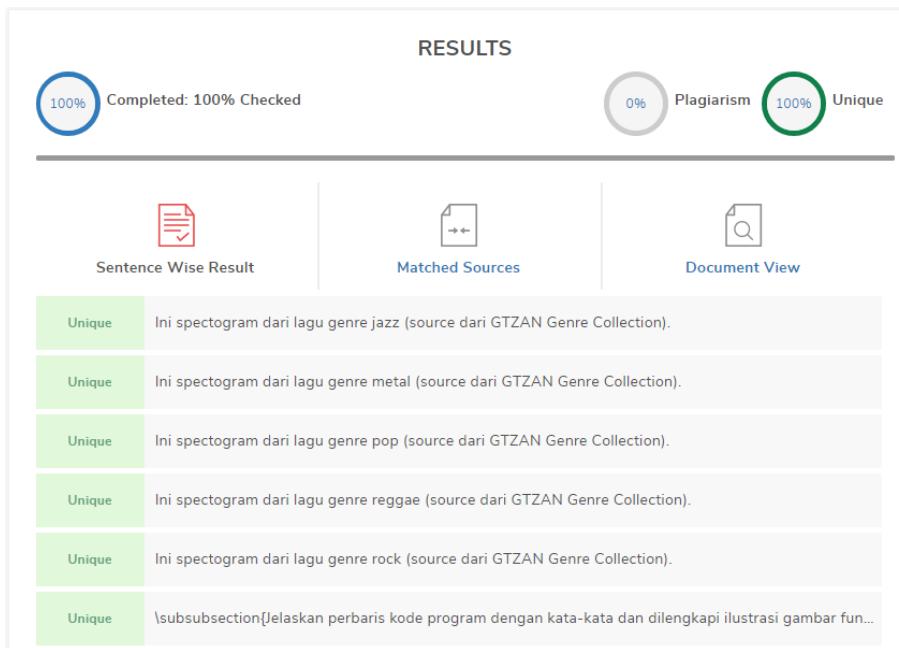
```

**Gambar 7.295** terjadi Error 2

#### 7.11.3.2 Solusi

- solusi dari error 1 ialah: membenarkan argumen, karena terdapat typo
- solusi dari error 2 ialah: proses running yang tidak sempurna pada blok sebelumnya, sehingga harus dijalankan lagi dengan benar.

### 7.11.4 Bukti Tidak Plagiat



**Gambar 7.296** Bukti tidak plagiat

### 7.11.5 Link Youtube

<https://bit.ly/baktiListVideo>



## BAB 8

---

# CHAPTER 8

---

### 8.1 1174083 - Bakti Qilan Mufid

Chapter 8 - Perkenalan Generative Adversarial Network

#### 8.1.1 Teori

##### 8.1.1.1 *Jelaskan dengan ilustrasi gambar sendiri apa itu generator dengan perumpamaan anda sebagai mahasiswa sebagai generatornya.*

Menurut KBBI, Generator adalah pembangkit. Arti lainnya dari generator adalah alat pembangkit tenaga listrik. atau jika pada kecerdasan buatan bisa diartikan sebagai pemubuat. misalnya pembuat gambar, pembuat musik, dll. jika perumpamaan mahasiswa sebagai generator maka dapat digambarkan seperti berikut:



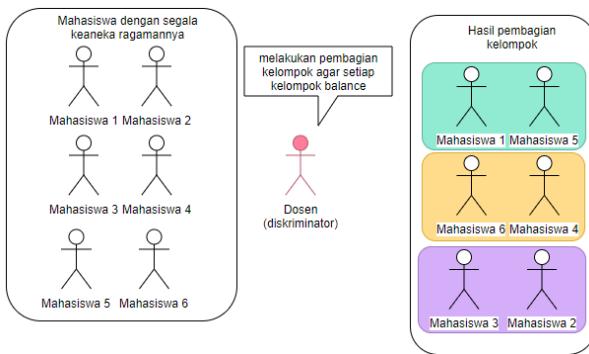
Gambar 8.1 gambaran penjelasan no. 1

#### **8.1.1.2 Jelaskan dengan ilustrasi gambar sendiri apa itu diskriminator dengan perumpamaan dosen anda sebagai diskriminatorya.**

arti kata diskriminator adalah rangkaian dalam berbagai alat. berasal dari kata diskriminasi yang merujuk pada pelayanan yang tidak adil terhadap individu tertentu, di mana layanan ini dibuat berdasarkan karakteristik yang diwakili oleh individu tersebut. dalam machine learning, Discrimination itu bisa ada dua, yaitu

1. Disparate Treatment(perawatan berbeda) atau membedakan/ mengklasifikasikan sesuatu atau seseorang
2. Disparate Impact (dampak berbeda) ialah melihat konsekuensi klasifikasi/ pengambilan keputusan pada kelompok tertentu.

berikut ilustrasi jika dosen sebagai diskriminator:



Gambar 8.2 gambaran penjelasan no. 2

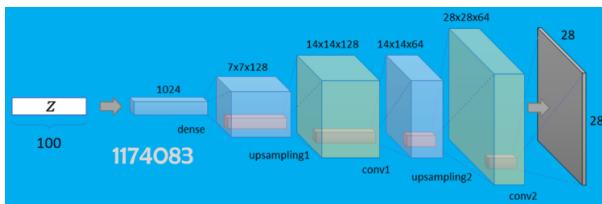
#### **8.1.1.3 Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur generator dibuat**

arsitektur generator didalam GAN terdiri dari lima lapisan, yaitu: satu input layer, tiga dense layer dan satu output layer. bisa dilihat arsitektur gererator pada GAN-Project seperti gambar berikut:

	13970549349376	
dense_1:Dense	Input:	(None, None, 100)
	Output:	(None, None, 500)
dense_2:Dense	Input:	(None, None, 500)
	Output:	(None, None, 500)
dense_3:Dense	Input:	(None, None, 500)
	Output:	(None, None, 784)
reshape_1:Reshape	Input:	(None, None, 784)
	Output:	(None, 28, 28)

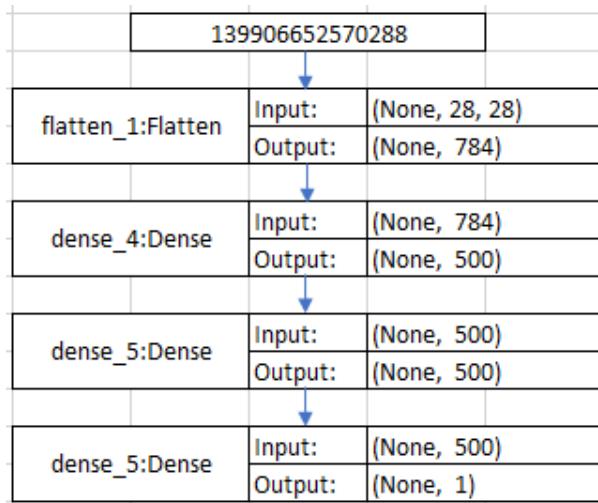
**Gambar 8.3** gambar arsitektur generator pada GAN-Project

untuk ilustrasinya seperti berikut:



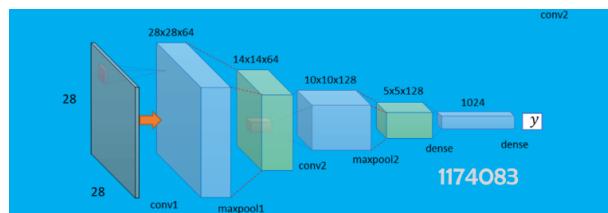
**Gambar 8.4** gambaran penjelasan no. 3

**8.1.1.4 Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur diskriminatore dibuat** arsitektur diskriminator didalam GAN terdiri dari lima lapisan, yaitu: satu input layer, tiga dense layer dan satu output layer. bisa dilihat arsitektur diskriminator pada GAN-Project seperti gambar berikut:



**Gambar 8.5** gambar arsitektur generator pada GAN-Project

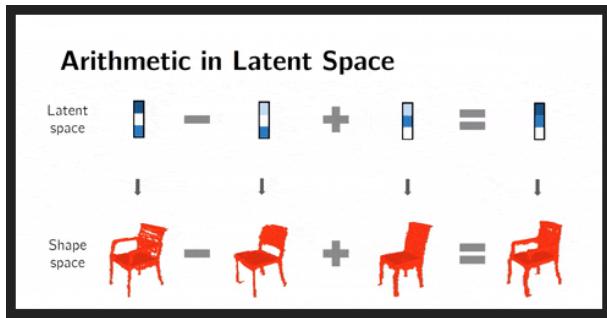
untuk ilustrasinya seperti berikut:



**Gambar 8.6** gambaran penjelasan no. 4

#### 8.1.1.5 Jelaskan dengan ilustrasi gambar apa itu latent space.

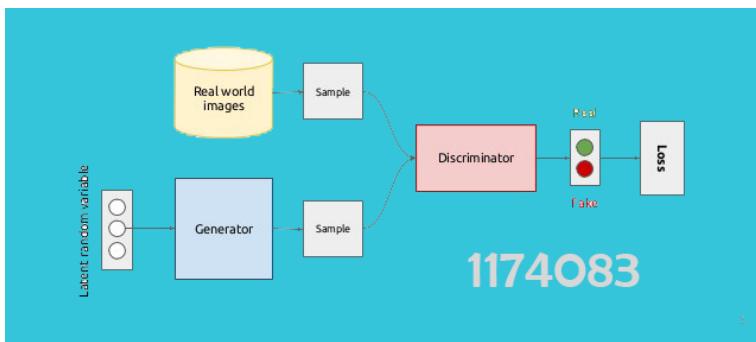
latent space adalah space atau spot yang tersembunyi, sehingga akan mengambil titik data yang dekat dan yang mirip. ilustrasinya seperti pada gambar berikut:



Gambar 8.7 gambaran penjelasan no. 5

#### 8.1.1.6 Jelaskan dengan ilustrasi gambar apa itu adversarial play

Dalam GAN adversarial play ialah persaingan antara generator dan diskriminator. bisa dilihat pada ilustrasi gambar berikut:



Gambar 8.8 gambaran penjelasan no. 6

#### 8.1.1.7 Jelaskan dengan ilustrasi gambar apa itu Nash equilibrium

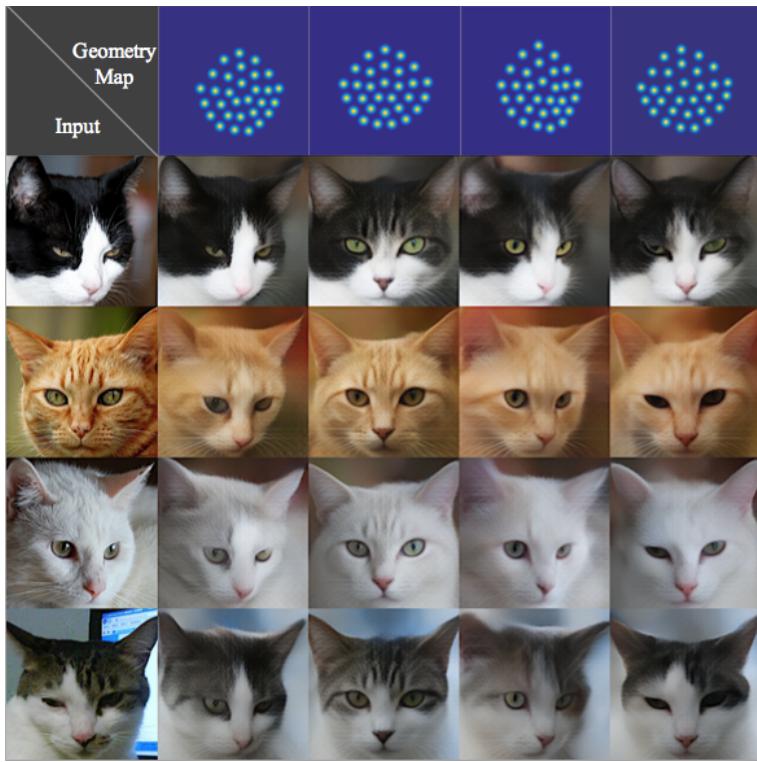
Nash equilibrium adalah strategi yang dipilih oleh masing-masing pemain, dengan strategi pemain lain tertentu. Strategi ini merupakan strategi terbaik dari masing-masing pemain dengan mempertimbangkan strategi tertentu yang diambil oleh pemain lainnya, atau yang disebut dengan Best Respons.

		TOM	
		A	B
SAM	A	1,1	1,-1
	B	-1,1	0,0

**Gambar 8.9** gambaran penjelasan no. 7

#### 8.1.1.8 **Sebutkan dan jelaskan contoh-contoh implementasi dari GAN**

Contoh implementasi dari GAN salah satunya adalah meng-generate wajah kucing, seperti pada gambar berikut:



**Gambar 8.10** gambaran penjelasan no. 8

**8.1.1.9 Berikan contoh dengan penjelasan kode program beserta gambar arsitektur untuk membuat generator(neural network) dengan sebuah input layer, tiga hidden layer(dense layer), dan satu output layer(reshape layer)**  
untuk gambar arsitektur nya ada pada gambar no. 3 sedangkan untuk kode programnya seperti berikut:

```

1 def generator_model():
2     model = Sequential([
3         Dense(1024, input_dim=100, activation='tanh'),
4         Dense(128*7*7),
5         Reshape((7, 7, 128)),
6         UpSampling2D(size=(2, 2)),
7         Conv2D(64, (5, 5), padding='same', activation='tanh'),
8         UpSampling2D(size=(2, 2)),
9         Conv2D(1, (5, 5), padding='same', activation='tanh')
10    ])
11    return model
12
13 generator_model().summary()

```

**Listing 8.1** Kode program arsitektur generator

### **8.1.1.10 Berikan contoh dengan ilustrasi dari arsitektur dikriminator dengan sebuah input layer, 3 buah hidden layer, dan satu output layer.**

untuk gambar arsitektur nya ada pada gambar no. 4 sedangkan untuk kode programnya seperti berikut:

```

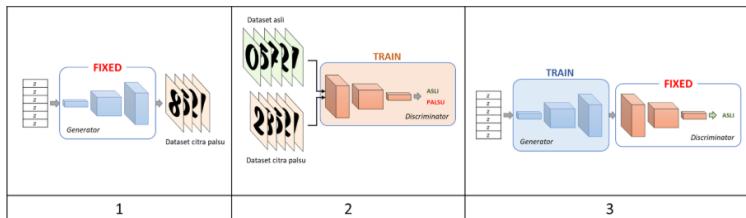
1 def discriminator_model():
2     model = Sequential([
3         Conv2D(64, (5, 5), padding='same', input_shape=(28, 28, 1),
4         activation='tanh'),
5         MaxPooling2D(pool_size=(2, 2)),
6         Conv2D(128, (5, 5), activation='tanh'),
7         MaxPooling2D(pool_size=(2, 2)),
8         Flatten(),
9         Dense(1024, activation='tanh'),
10        Dense(1, activation='sigmoid')
11    ])
12    return model
13
14 discriminator_model().summary()

```

**Listing 8.2** Kode program arsitektur diskriminator

### **8.1.1.11 Jelaskan bagaimana kaitan output dan input antara generator dan diskriminator tersebut. Jelaskan kenapa inputan dan outputan seperti itu.**

kaitan antara output pada generator dan inputan pada diskriminator adalah suatu keterhubungan yang akan menghasilkan data yang benar, karena output dari generator merupakan inoutan pada diskriminator, berikut gambarannya:



**Gambar 8.11** gambaran penjelasan no. 9

### **8.1.1.12 Jelaskan apa perbedaan antara Kullback-Leibler divergence (KL divergence)/relative entropy, Jensen-Shannon(JS) divergence / information radius(iRaD) / total divergence to the average dalam mengukur kualitas dari model.**

diantara keduanya yang paling terkenal adalah KL divergence, karena KL divergence memiliki beberapa sifat/rumus yang bagus .salah satunya adalah  $KL[q; p]$  jenis daerah yang di mana  $q(x)$  memiliki massa non-null dan  $p(x)$  memiliki massa null. Ini mungkin terlihat seperti bug, tetapi sebenarnya ini merupakan fitur dalam situasi tertentu. sedangkan JS tidak memiliki

### **8.1.1.13 Jelaskan apa itu fungsi objektif yang berfungsi untuk mengukur kesamaan antara gambar yang dibuat dengan yang asli.**

fungsi objektif merupakan fungsi yang akan mengambil parameter data dan model sebagai argumen, dan dapat dievaluasi untuk mengembalikan angka sehingga mampu membedakan gambar yang asli dan yang di buat/fake.

#### ***8.1.1.14 Jelaskan apa itu scoring algoritma selain mean square error atau cross entropy seperti The Inception Score dan The Frechet Inception distance.***

Inception Score digunakan untuk mengukur seberapa realistik output dari GAN, dimana ada dua parameter, yaitu : gambarnya punya variasi dan setiap gambar jelas terlihat seperti sesuatu. Frechet Inception Distance adalah ukuran kesamaan antara dua dataset gambar. Itu terbukti berkorelasi baik dengan penilaian manusia terhadap kualitas visual dan paling sering digunakan untuk mengevaluasi kualitas sampel Generative Adversarial Networks. FID dihitung dengan menghitung jarak Fréchet antara dua Gaussians dipasang ke representasi fitur dari jaringan Inception.

#### ***8.1.1.15 Jelaskan kelebihan dan kekurangan GAN***

- Kelebihan

1. GAN Menghasilkan data baru yang bisa hampir mirip dengan data asli. Karena hasil pelatihannya, GAN dapat menghasilkan data gambar, teks, audio, dan video yang dapat dibilang hampir mirip dengan yang aslinya. Berkat hal tersebut, GAN dapat digunakan dalam sistem marketing, e-commerce, games, iklan, dan industri lainnya
2. GAN mempelajari representasi data secara internal sehingga beberapa masalah pada machine learning dapat diatasi dengan mudah
3. Discriminator yang sudah dilatih dapat menjadi sebuah classifier atau pendekripsi jika data sudah sesuai. Karena Discriminator yang akan menjadi tidak efisien berkat seringkali dilatih
4. GAN dapat dilatih menggunakan data yang belum dilabeled

- Kekurangan

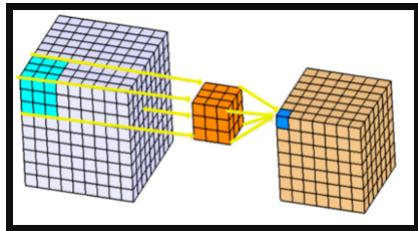
1. Data saat diproses oleh metode gan tidak konvergensi
2. Jenis sampel yang dihasilkan oleh generator terbatas karena modanya terbatas
3. Ketidak seimbangnya antara generator dan discriminator dapat menyebabkan overfitting atau terlalu dekat dengan hasil sampel
4. Sangat sensitif dengan data yang sudah diinisiasi sebelumnya

#### **8.1.2 Praktek**

##### ***8.1.2.1 Jelaskan apa itu 3D convolutions***

3D convolutions merupakan operasi konvolusi 3D yang menerapkan filter 3D ke data

input dengan tiga arah, yaitu x,y, dan z. fitur ini menciptakan sebuah daftar peta yang ditumpuk. agar lebih mudah saya menggunakan sebuah ilustrasi seperti berikut:



**Gambar 8.12** gambaran 3D Convolutions

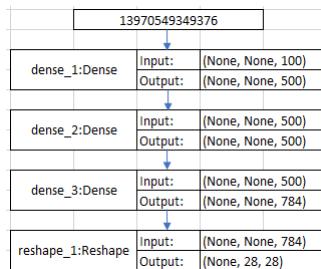
#### 8.1.2.2 Jelaskan dengan kode program arsitektur dari generator networknya, beserta penjelasan input dan output dari generator network.

```

1 def build_generator():#nama kelasnya
2     """
3         Create a Generator Model with hyperparameters values defined as
4         follows
5     """
6     z_size = 200
7     gen_filters = [512, 256, 128, 64, 1]
8     gen_kernel_sizes = [4, 4, 4, 4, 4]
9     gen_strides = [1, 2, 2, 2, 2]
10    gen_input_shape = (1, 1, 1, z_size)
11    gen_activations = ['relu', 'relu', 'relu', 'relu', 'sigmoid']
12    gen_convolutional_blocks = 5
13
14    input_layer = Input(shape=gen_input_shape)
15
16    # First 3D transpose convolution(or 3D deconvolution) block
17    a = Deconv3D(filters=gen_filters[0],
18                  kernel_size=gen_kernel_sizes[0],
19                  strides=gen_strides[0])(input_layer)
20    a = BatchNormalization()(a, training=True)
21    a = Activation(activation='relu')(a)
22
23    # Next 4 3D transpose convolution(or 3D deconvolution) blocks
24    for i in range(gen_convolutional_blocks - 1):
25        a = Deconv3D(filters=gen_filters[i + 1],
26                      kernel_size=gen_kernel_sizes[i + 1],
27                      strides=gen_strides[i + 1], padding='same')(a)
28        a = BatchNormalization()(a, training=True)
29        a = Activation(activation=gen_activations[i + 1])(a)
30
31    gen_model = Model(inputs=[input_layer], outputs=[a])
32    return gen_model

```

kode diatas merupakan generator network yang terdiri dari lima layer, satu input layer, tiga hidden layer, dan satu output layer.Dan prosesnya dapat digambarkan seperti berikut:



**Gambar 8.13** gambaran proses generator network

penjelasan

- Input layer mengambil 100-dimensi sampel dari distribusi Gaussian(normal) dan meneruskan tensor ke. hidden layer pertama tanpa ada modifikasi
- ketiga hidden layer adalah dense layer dengan unit masing-masing 500, 500, dan 784. dimana dense layer pertama mengkonversi bentuk tensor (batch\_size, 100) ke bentuk tensor(batch\_size, 500). sedangkan pada layer dense kedua menghasilkan bentuk tensor (batch\_size, 500), dan layer dense ketiga menghasilkan (batch\_size, 784)
- Output layer, tensor akan dibentuk kembali dari bentuk tensor (batch\_size, 784) menjadi (batch\_size, 28, 28) artinya hasil dari generator ini akan menghasilkan banyak gambar, dimana setiap gambarnya memiliki ukuran (28, 28).

#### **8.1.2.3 Jelaskan dengan kode program arsitektur dari diskriminator network, beserta penjelasan input dan outputnya.**

```

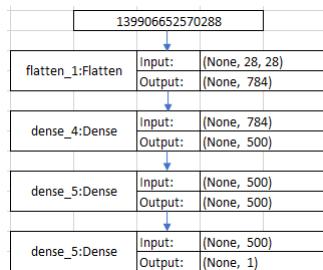
1 def build_discriminator():
2     """
3         Create a Discriminator Model using hyperparameters values defined
4             as follows
5         """
6     dis_input_shape = (64, 64, 64, 1)
7     dis_filters = [64, 128, 256, 512, 1]
8     dis_kernel_sizes = [4, 4, 4, 4, 4]
9     dis_strides = [2, 2, 2, 2, 1]
10    dis_paddings = ['same', 'same', 'same', 'same', 'valid']
11    dis_alphas = [0.2, 0.2, 0.2, 0.2, 0.2]
12    dis_activations = ['leaky_relu', 'leaky_relu', 'leaky_relu',
13                      'leaky_relu', 'sigmoid']
14    dis_convolutional_blocks = 5
15
16    dis_input_layer = Input(shape=dis_input_shape)
17
18    # The first 3D Convolutional block
19    a = Conv3D(filters=dis_filters[0],
20               kernel_size=dis_kernel_sizes[0],
21               
```

```

20             strides=dis_strides[0],
21             padding=dis_paddings[0])(dis_input_layer)
22 # a = BatchNormalization()(a, training=True)
23 a = LeakyReLU(dis_alphas[0])(a)
24
25 # Next 4 3D Convolutional Blocks
26 for i in range(dis_convolutional_blocks - 1):
27     a = Conv3D(filters=dis_filters[i + 1],
28                 kernel_size=dis_kernel_sizes[i + 1],
29                 strides=dis_strides[i + 1],
30                 padding=dis_paddings[i + 1])(a)
31     a = BatchNormalization()(a, training=True)
32     if dis_activations[i + 1] == 'leaky_relu':
33         a = LeakyReLU(dis_alphas[i + 1])(a)
34     elif dis_activations[i + 1] == 'sigmoid':
35         a = Activation(activation='sigmoid')(a)
36
37 dis_model = Model(inputs=[dis_input_layer], outputs=[a])
38 return dis_model

```

kode diatas merupakan diskriminatior network, dimana prosesnya dapat digambarkan seperti berikut:



**Gambar 8.14** gambaran proses diskriminatior network

penjelasan

- awalnya diskriminatior menerima input dengan bentuk 28x28.
- Input layer mengambil tensor input dan meneruskannya ke hidden layer pertama tanpa modifikasi apapun.
- lalu flattens layer akan meratakan tensor menjadi 784-dimensi vektor, yang akan diteruskan ke hidden layer (dense layer) pertama. hidden layer pertama dan kedua aka memodifikasinya menjadi vektor 500-dimensi.
- Output layer mesuk kedalam dense layer, dengan satu unit neuron dan sigmoid sebagai fungsi aktivasi. sigmoid akan menghasilkan nilai tunggal, 0 atau 1. Nilai 0 akan menunjukan bahwa gambar yang diberikan palsu. sebaliknya jika nilai 1 maka gambar asli.

#### **8.1.2.4 Jelaskan proses training 3D-GANs**

Proses training 3D-GANs dilakukan seperti langkah-langkah berikut:

- Terdapat sebuah vektor noise dengan dimensi 200 dari distribusi Gaussian(normal).
- Meng-generate gambar palsu menggunakan model generator.
- Melatih jaringan generator dengan gambar yang asli(sampel dari data yang real) dan dengan gambar palsu yang dihasilkan oleh generator.
- Gunakan adversarial model untuk melatih generator model, jangan melatih diskriminatore model.
- Ulangi langkah ini dengan jumlah epoch tertentu.

#### **8.1.2.5 Jelaskan bagaimana melakukan settingan awal chapter 02 untuk memenuhi semua kebutuhan sebelum melanjutkan ke tahapan persiapan data.**

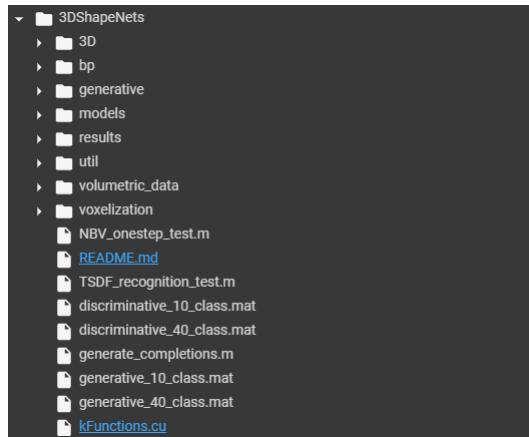
Sebelum melakukan tahapan persiapan data, kita harus melakukan beberapa hal seperti berikut:

- membaca buku panduan atau keterangan dari Generative-Adversarial-Network project ini. bisa didownload di portal kampus keren atau di github <https://github.com/PacktPublishing/Generative-Adversarial-Networks-Projects>
- persiapkan laptop/pc dengan spek yang cukup bagus(tidak kentang). jika tidak ada, bisa menggunakan google colab(cara penggunaanya akan dijelaskan di video).
- usahakan versi yang terinstall sama dengan versi yang ada pada requirement.txt agar terhindar dari error.

#### **8.1.2.6 Jelaskan tentang dataset yang digunakan, dari mulai tempat unduh, cara membuka dan melihat data. sampai deskripsi dari isi dataset dengan detail penjelasan setiap folder/file yang membuat orang awam paham.**

Penjelasan dataset

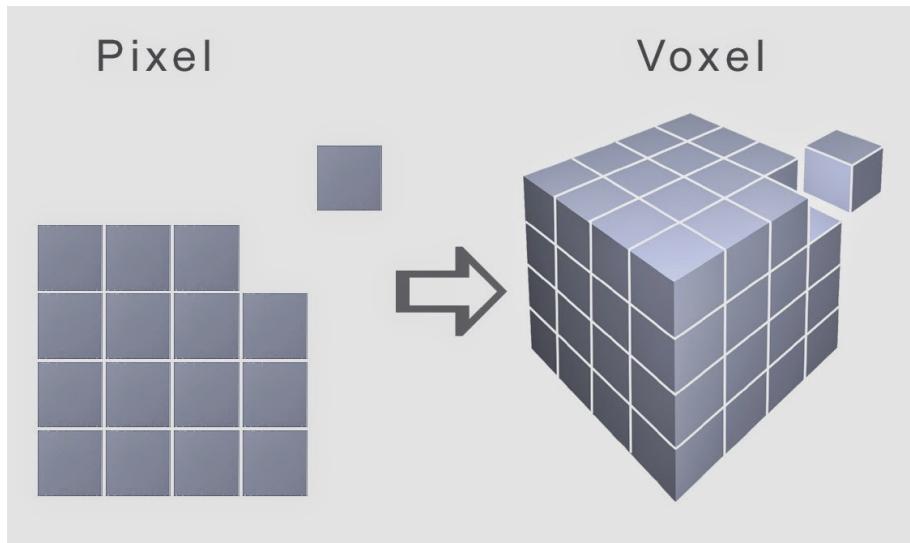
- dataset bisa didownload pada halaman: <http://3dshapenets.cs.princeton.edu/3DShapeNets>
- Untuk membuka data, setelah didownload cukup di eksrak saja menggunakan winrar/ 7zip
- Untuk mengetahui lebih jelasnya tentang dataset ini bisa dibuka saja file README.md, disana akan dijelaskan semuanya.



**Gambar 8.15** Isi dari dataset.zip

#### **8.1.2.7 Jelaskan apa itu voxel dengan ilustrasi dan bahasa paling awam**

Secara singkat, voxel dapat diartikan sebagai 3D pixel atau pixel dalam bentuk 3D. seperti pada ilustrasi berikut:



**Gambar 8.16** Pixel VS Voxel

#### **8.1.2.8 Visualisasikan dataset tersebut dalam tampilan visual plot, jelaskan cara melakukan visualisasinya**

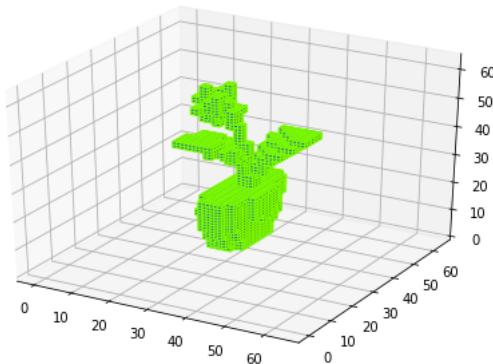
```
1 import scipy.io as io
```

```

2 import matplotlib.pyplot as plt
3 import scipy.ndimage as nd
4 import numpy as np
5 from mpl_toolkits.mplot3d import Axes3D
6
7 voxels = io.loadmat("/content/3DShapeNets/volumetric_data/flower_pot
8 /30/test/flower_pot_000000010_1.mat")['instance']
9 voxels = np.pad(voxels, (1, 1), 'constant', constant_values=(0, 0))
9 voxels = nd.zoom(voxels, (2, 2, 2), mode='constant', order=0)
10
11 fig = plt.figure()
12 ax = Axes3D(fig)
13 ax.voxels(voxels, edgecolor="chartreuse")
14
15 plt.show()
16 plt.savefig('flower_pot')

```

Kode diatas berfungsi untuk melakukan visualisasi dataset dalam tampilan plot. dengan tahapan import library, load data file .mat, dan lakukan read memakai matplotlib. dan hasilnya seperti berikut



**Gambar 8.17** Visualisasi dataset(contoh)

#### 8.1.2.9 Buka file run.py jelaskan perbaris kode pada fungsi untuk membuat generator yaitu build generator

```

1 z_size = 200
2 gen_filters = [512, 256, 128, 64, 1]
3 gen_kernel_sizes = [4, 4, 4, 4, 4]
4 gen_strides = [1, 2, 2, 2, 2]
5 gen_input_shape = (1, 1, 1, z_size)
6 gen_activations = ['relu', 'relu', 'relu', 'relu', 'sigmoid']
7 gen_convolutional_blocks = 5

```

potongan kode diatas berfungsi sebagai penentuan nilai untuk hyperparameters yang berbeda.

```
1 input_layer = Input(shape=gen_input_shape)
```

potongan kode diatas berfungsi untuk membuat input layer

```
1 # First 3D transpose convolution(or 3D deconvolution) block
2 a = Deconv3D(filters=gen_filters[0],
3               kernel_size=gen_kernel_sizes[0],
4               strides=gen_strides[0])(input_layer)
5 a = BatchNormalization()(a, training=True)
6 a = Activation(activation='relu')(a)
```

potongan kode diatas berfungsi untuk menambahkan 3D transpose pertama.

```
1 # Next 4 3D transpose convolution(or 3D deconvolution) blocks
2 for i in range(gen_convolutional_blocks - 1):
3     a = Deconv3D(filters=gen_filters[i + 1],
4                   kernel_size=gen_kernel_sizes[i + 1],
5                   strides=gen_strides[i + 1], padding='same')(a)
6     a = BatchNormalization()(a, training=True)
7     a = Activation(activation=gen_activations[i + 1])(a)
```

potongan kode diatas berfungsi untuk menambahkan empat 3D transpose lainnya.

```
1 gen_model = Model(inputs=[input_layer], outputs=[a])
```

potongan kode diatas berfungsi untuk membuat model keras dan menentukan input dan output untuk network generator.

#### **8.1.2.10 jelaskan juga fungsi untuk membangun diskriminator pada fungsi build discriminator.**

```
1 dis_input_shape = (64, 64, 64, 1)
2 dis_filters = [64, 128, 256, 512, 1]
3 dis_kernel_sizes = [4, 4, 4, 4, 4]
4 dis_strides = [2, 2, 2, 2, 1]
5 dis_paddings = ['same', 'same', 'same', 'same', 'valid']
6 dis_alphas = [0.2, 0.2, 0.2, 0.2, 0.2]
7 dis_activations = ['leaky_relu', 'leaky_relu', 'leaky_relu',
```

potongan kode diatas berfungsi sebagai penentuan nilai untuk hyperparameters yang berbeda.

```
1 dis_input_layer = Input(shape=dis_input_shape)
```

potongan kode diatas berfungsi untuk membuat input layer, berupa gambar 3D dengan dimensi 64x64x64x1

```
1 # The first 3D Convolutional block
2 a = Conv3D(filters=dis_filters[0],
3             kernel_size=dis_kernel_sizes[0],
4             strides=dis_strides[0],
5             padding=dis_paddings[0])(dis_input_layer)
6 # a = BatchNormalization()(a, training=True)
7 a = LeakyReLU(dis_alphas[0])(a)
```

potongan kode diatas berfungsi untuk menambahkan 3D transpose pertama.

```

1 # Next 4 3D Convolutional Blocks
2 for i in range(dis_convolutional_blocks - 1):
3     a = Conv3D(filters=dis_filters[i + 1],
4                 kernel_size=dis_kernel_sizes[i + 1],
5                 strides=dis_strides[i + 1],
6                 padding=dis_paddings[i + 1])(a)
7     a = BatchNormalization()(a, training=True)
8     if dis_activations[i + 1] == 'leaky_relu':
9         a = LeakyReLU(dis_alphas[i + 1])(a)
10    elif dis_activations[i + 1] == 'sigmoid':
11        a = Activation(activation='sigmoid')(a)

```

potongan kode diatas berfungsi untuk menambahkan empat 3D transpose lainnya.

```
1 dis_model = Model(inputs=[dis_input_layer], outputs=[a])
```

potongan kode diatas berfungsi untuk membuat model keras dan menentukan input dan output untuk network generator.

#### **8.1.2.11 jelaskan apa maksud dari kode program name == 'main'**

```
1 if __name__ == '__main__':
```

**Listing 8.3** Kode program run.py

maksudnya kode tersebut memiliki fungsi jika interpreter python menjalankan kode tersebut, maka akan menetapkan variable name untuk memiliki nilai main. jika file ini di import dari modul lain, maka name akan ditetapkan ke nama modul tersebut.

#### **8.1.2.12 jelaskan secara detil perbaris dan per parameter apa arti dari kode program :**

Penjelasan kode:

```

1 # In[ soal_12]
2     object_name = "airplane" #membuat variabel
3     data_dir = "data/3DShapeNets/volumetric_data/" \
4                 "{}/30/train/*.mat".format(object_name) #menunjukkan
5     direktori dari variable sebelumnya
6     gen_learning_rate = 0.0025 #membuat rate pada generator
7     dis_learning_rate = 1e-5 #membuat rate pada diskriminatot
8     beta = 0.5 #
9     batch_size = 1 #menentukan ukuran kumpulan gambar
10    z_size = 200 #
11    epochs = 1 #menetukkan masa periode. setiap epoch memiliki semua
12      batch
13    MODE = "train" #membuat traon mode

```

**Listing 8.4** Kode program run.py

#### **8.1.2.13 Jelaskan secara detil dari kode program pembuatan dan kompilasi arsitektur berikut :**

```

1  gen_optimizer = Adam(lr=gen_learning_rate, beta_1=beta)
2  dis_optimizer = Adam(lr=dis_learning_rate, beta_1=beta)
3
4  discriminator = build_discriminator()
5  discriminator.compile(loss='binary_crossentropy', optimizer=
6    dis_optimizer)
7
8  generator = build_generator()
9  generator.compile(loss='binary_crossentropy', optimizer=
10   gen_optimizer)

```

**Listing 8.5** Kode program run.py

penjelasanya ialah membuat model dari generator model dan diskriminatormodel,

#### **8.1.2.14 Jelaskan secara detil kode program untuk membuat dan melakukan kompilasi model adversarial berikut:**

```

1 # In[ soal 14]
2 discriminator.trainable = False
3
4 input_layer = Input(shape=(1, 1, 1, z_size))
5 generated_volumes = generator(input_layer)
6 validity = discriminator(generated_volumes)
7 adversarial_model = Model(inputs=[input_layer], outputs=[validity])
8 adversarial_model.compile(loss='binary_crossentropy', optimizer=
9   gen_optimizer)

```

**Listing 8.6** Kode program run.py

penjelasanya ialah membuat model diskriminatormodel tidak di training dan membuat model adversarial.

#### **8.1.2.15 Jelaskan Ekstrak dan load data kursi dengan menggunakan fungsi getVoxelsFormat dan get3DImages yang digunakan pada kode program berikut :**

```

1 # In[ soal 15]
2 print("Loading data...")
3 volumes = get3DImages(data_dir=data_dir)
4 volumes = volumes [..., np.newaxis].astype(np.float)
5 print("Data loaded...")

```

**Listing 8.7** Kode program run.py

penjelasanya ialah melakukan print loading data, membuat variabel volumes untuk mendapatkan data gamabr 3D.

#### **8.1.2.16 Jelaskan maksud dari kode program instansiasi TensorBoard yang menambahkan generator dan diskriminatormodel pada program berikut:**

```

1 # In[ soal_16]
2     tensorboard = TensorBoard(log_dir="logs/{}".format(time.time()))
3     tensorboard.set_model(generator)
4     tensorboard.set_model(discriminator)

```

**Listing 8.8** Kode program run.py

penjelasanya ialah membuat variabel tensorboard untuk melakukan pencatatan log, dan men-set model generator dan model diskriminatot

#### **8.1.2.17 Jelaskan apa fungsi dari np reshape ones zeros pada kode program berikut dengan parameternya:**

Penjelasan kode:

```

1 # In[ soal_17]
2     labels_real = np.reshape(np.ones((batch_size,)), (-1, 1, 1, 1, 1))
3     labels_fake = np.reshape(np.zeros((batch_size,)), (-1, 1, 1, 1, 1))

```

**Listing 8.9** Kode program run.py

membuat label\_real untuk menetapkan bahwa jika output 1 itu menunjukan gambar real, dan label\_fake untuk menunjukan bahwa output 0 untuk data fake/palsu.

#### **8.1.2.18 Jelaskan kenapa harus ada perulangan dalam meraih epoch. Dan jelaskan apa itu epoch terkait kode program berikut:**

```

1 # In[ soal_18]
2     if MODE == 'train':
3         for epoch in range(epochs):
4             print("Epoch:", epoch)
5
6             gen_losses = []
7             dis_losses = []

```

**Listing 8.10** Kode program run.py

melakukan perulangan jika MODE = train maka akan diulangi terus menerus sampai epoch(periode) terpenuhi dengan disertai keterangan generator dan diskriminatot losses.

#### **8.1.2.19 Jelaskan apa itu batches dan kaitannya dengan kode program berikut, dan kenapa berada di dalam epoch:**

```

1 # In[ soal_19]
2     number_of_batches = int(volumes.shape[0] / batch_size)
3     print("Number of batches:", number_of_batches)
4     for index in range(number_of_batches):
5         print("Batch:", index + 1)

```

**Listing 8.11** Kode program run.py

menghitung jumlah batch yang telah ditrain

**8.1.2.20 Berikut adalah kode program pengambilan gambar dan noise. Jelaskan apa fungsi `np.random.normal` serta `astype`, serta jelaskan apa arti parameter titik dua dan jelaskan isi dari `z sample` dan `volumes batch`:**

```

1 # In[ soal 20]
2             z_sample = np.random.normal(0, 0.33, size=[batch_size
3 , 1, 1, 1, z_size]).astype(np.float32)
4             volumes_batch = volumes[index * batch_size:(index +
5 1) * batch_size, :, :, :]

```

**Listing 8.12** Kode program run.py

untuk membersihkan gambar dari noise dan menyesuaikan shape.

**8.1.2.21 Berikut adalah kode program generator gambar palsu. Jelaskan apa fungsi `generator.predict_on_batch`, serta jelaskan apa arti parameter `z sample`:**

```

1 # In[ soal 21]
2             # Next, generate volumes using the generate network
3             gen_volumes = generator.predict_on_batch(z_sample)

```

**Listing 8.13** Kode program run.py

untuk meng-generate volume menggunakan model jaringan generator.

**8.1.2.22 Berikut adalah kode program training diskriminator dengan gambar palsu dari generator dan gambar asli. Jelaskan apa maksudnya harus dilakukan training diskriminator secara demikian dan jelaskan apa isi `loss fake` dan `loss real` serta `d loss` dan fungsi `train on batch`.**

```

1 # In[ soal 22]
2             discriminator.trainable = True
3             if index % 2 == 0:
4                 loss_real = discriminator.train_on_batch(
5                     volumes_batch, labels_real)
6                     loss_fake = discriminator.train_on_batch(
7                     gen_volumes, labels_fake)
8
9                     d_loss = 0.5 * np.add(loss_real, loss_fake)
10                    print("d_loss:{:.4f}".format(d_loss))
11

```

**Listing 8.14** Kode program run.py

untuk membuat diskriminator bisa meload gambar fake dan real dari model generator. dengan disertai generatot dan diskriminatot loss, agar terlihat seberapa baik kualitas yang dihasilkan.

**8.1.2.23 Berikut adalah kode program training model adversarial yang terdapat generator dan diskriminatot. Jelaskan apa bagaimana proses terbentuknya**

**parameter z dan g loss:**

```

1 # In[ soal_23]
2         discriminator.trainable = False
3         """
4             Train the generator network
5             """
6         z = np.random.normal(0, 0.33, size=[batch_size, 1, 1,
7             1, z_size]).astype(np.float32)
8         g_loss = adversarial_model.train_on_batch(z,
9             labels_real)
10        print("g_loss:{}".format(g_loss))
11        gen_losses.append(g_loss)
12        dis_losses.append(d_loss)

```

**Listing 8.15** Kode program run.py

untuk mentrain model generator dan variabel g\_loss untuk melakukan perbandingan antara label gambar yang asli.

**8.1.2.24 Berikut adalah kode program generate dan menyimpan gambar 3D setelah beberapa saat setiap epoch. Jelaskan mengapa ada perulangan dengan parameter tersebut, serta jelaskan arti setiap variabel beserta perlihatkan isinya dan artikan isinya :**

```

1 # In[ soal_24]           # Every 10th mini-batch, generate volumes and save
2         them
3         if index % 10 == 0:
4             z_sample2 = np.random.normal(0, 0.33, size=[batch_size, 1, 1, 1, z_size]).astype(np.float32)
5             generated_volumes = generator.predict(z_sample2,
6                 verbose=3)
7             for i, generated_volume in enumerate(
8                 generated_volumes[:5]):
9                 voxels = np.squeeze(generated_volume)
10                voxels[voxels < 0.5] = 0.
11                voxels[voxels >= 0.5] = 1.
12                saveFromVoxels(voxels, "results/img_{:}-{:}-{:}".
13 .format(epoch, index, i))

```

**Listing 8.16** Kode program run.py

untuk melakukan perulangan dimana setiap terdapat 10 mini-batch akan meng-generate volumenya dan akan menyimpannya.

**8.1.2.25 Berikut adalah kode program menyimpan average losses setiap epoch. Jelaskan apa itu tensorboard dan setiap parameter yang digunakan pada kode program ini :**

```

1 # In[ soal_25]

```

```

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

```

**Listing 8.17** Kode program run.py

untuk menuliskan log losses ke Tensorboard

**8.1.2.26 Berikut adalah kode program menyimpan model. Jelaskan apa itu format h5 dan penjelasan dari kode program berikut :**

```

1 # In[ soal_26]
2         generator.save_weights(os.path.join("models" , "generator_weights.h5"))
3         discriminator.save_weights(os.path.join("models" , "discriminator_weights.h5"))

```

**Listing 8.18** Kode program run.py

untuk menyimpan berat model generator dan model diskriminasi kedalam file h5.

H5 adalah Hierarchical Data Format 5 File. File H5 adalah file data yang disimpan dalam Format Data Hirarki (HDF). Ini berisi array multidimensi data ilmiah. File H5 biasanya digunakan di luar angkasa, fisika, teknik, keuangan, penelitian akademis, genomik, astronomi, instrumen elektronik, dan bidang medis.

**8.1.2.27 Berikut adalah kode program testing model. Jelaskan dengan ilustrasi gambar dari mulai meload hingga membuat gambar 3D dengan menggunakan z sample, bisakah parameter z sample tersebut diubah? :**

```

1         generator.load_weights(os.path.join("models" , "generator_weights.h5") , True)
2         discriminator.load_weights(os.path.join("models" , "discriminator_weights.h5") , True)
3
4         # Generate 3D models
5         z_sample = np.random.normal(0 , 1 , size=[batch_size , 1 , 1 , 1 ,
z_size]).astype(np.float32)
6         generated_volumes = generator.predict(z_sample , verbose=3)
7
8         for i , generated_volume in enumerate(generated_volumes [:2]):
9             voxels = np.squeeze(generated_volume)
10            voxels[voxels < 0.5] = 0.
11            voxels[voxels >= 0.5] = 1.
12            saveFromVoxels(voxels , "results/gen_{}`".format(i))

```

**Listing 8.19** Kode program run.py

untuk membuat mode predict/prediksi. dimana dia melakukan pembuatan model generator, model diskriminasi dan meload data h5 yang sudah dibuat tadi dan menggenerate nya kedalam model 3D, lalu menyimpan model voxel nya ke folder results.

## 8.1.3 Penanganan Error

### 8.1.3.1 Terjadi error

1. terjadi error Summary has no attribute, seperti pada gambar berikut:

```

-----
AttributeError                                Traceback (most recent call last)
<ipython-input-4-d9ec96389e6a> in <module>()
  233
  234      # Write losses to Tensorboard
--> 235      write_log(tensorboard, 'g_loss', np.mean(gen_losses), epoch)
  236      write_log(tensorboard, 'd_loss', np.mean(dis_losses), epoch)
  237

<ipython-input-4-d9ec96389e6a> in write_log(callback, name, value, batch_no)
   93
   94     def write_log(callback, name, value, batch_no):
---> 95         summary = tf.Summary()
   96         summary_value = summary.value.add()
   97         summary_value.simple_value = value

AttributeError: module 'tensorflow' has no attribute 'Summary'

```

**Gambar 8.18** terjadi Error 1

### 8.1.3.2 Solusi

1. solusi dari error 1 ialah: dengan menginstall versi tensorflow 1.xx

```
(base) C:\Users\Bakti Qilan>pip install tensorflow==1.2
```

**Gambar 8.19** solusi Error 1

### 8.1.4 Bukti Tidak Plagiat

**RESULTS**


100%
Completed: 100% Checked


4%
Plagiarism


96%
UNIQUE

---


Sentence Wise Result


Matched Sources


Document View

UNIQUE	\subsubsection{Jelaskan dengan ilustrasi gambar sendiri apa itu generator dengan perumpamaan an...
UNIQUE	Arti lainnya dari generator adalah alat pembangkit tenaga listrik.
UNIQUE	atau jika pada kecerdasan buatan bisa diartikan sebagai pembutan.
UNIQUE	jika perumpamaan mahasiswa sebagai generator maka dapat digambarkan seperti berikut:
UNIQUE	\subsubsection{Jelaskan dengan ilustrasi gambar sendiri apa itu diskriminator dengan perumpamaan ...}

**RESULTS**


100%
Completed: 100% Checked


0%
Plagiarism


100%
UNIQUE

---


Sentence Wise Result


Matched Sources


Document View

UNIQUE	\subsubsection{jelaskan apa maksud dari kode program name == ' main '}
UNIQUE	maksudnya kode tersebut memiliki fungsi jika interpreter python menjalankan kode tersebut, maka ak...
UNIQUE	jika file ini di import dari modul lain, maka name akan ditetapkan ke nama modul tersebut.
UNIQUE	\subsubsection{jelaskan secara detil perbaris dan per parameter apa arti dari kode program :]}
UNIQUE	\subsubsection{jelaskan secara detil dari kode program pembuatan dan kompilasi arsitektur berikut :)}

**Gambar 8.20**   Bukti tidak plagiat

### 8.1.5 Link Youtube

<https://bit.ly/baktiListVideo>