

BAB 1

CHAPTER 1

BAB 2

CHAPTER 2

BAB 3

CHAPTER 3

BAB 4

CHAPTER 4

BAB 5

CHAPTER 4

BAB 6

CHAPTER 5

BAB 7

CHAPTER 6

BAB 8

CHAPTER 7

8.1 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
3     rehabilitation program using AES-CBC URL encryption and QR-
4     Code},
5   author={Awangga, Rolly Maulana and Fathonah, Nuraini Siti and
6     Hasanudin, Trisna Irmayadi},
7   booktitle={Information Technology, Information Systems and
8     Electrical Engineering (ICITISEE), 2017 2nd International
9     conferences on},
10  pages={255--260},
11  year={2017},
12  organization={IEEE}
13 }
```



Gambar 8.1 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.

8.1.1 Teori

8.1.2 Praktek

8.1.3 Penanganan Error

8.1.4 Bukti Tidak Plagiat



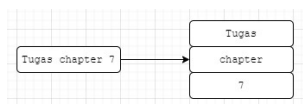
Gambar 8.2 Kecerdasan Buatan.

8.2 1174066 - D.Irga B. Naufal Fakhri

8.2.1 Teori

8.2.1.1 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.



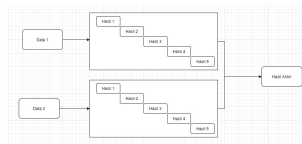
Gambar 8.3 Teori 1

8.2.1.2 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'])
```

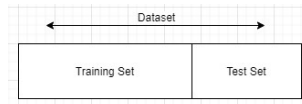
terdapat kfold yang bertujuan untuk melakukan split data menjadi 5 bagian dari dataset komentar Youtube tersebut. Sehingga dari setiap data yang sudah dibagi tersebut akan menghasilkan presentase dari setiap bagiannya, untuk menghasilkan hasil akhir dengan presentase yang cukup baik.



Gambar 8.4 Teori 2

8.2.1.3 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 8.5 Teori 3

8.2.1.4 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 di inginkan.

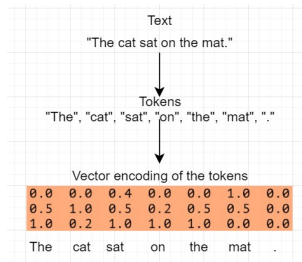
```
1 import pandas as pd
2
3 mydict = [{ 'a': 1, 'b': 2, 'c': 3, 'd': 4},
4           { 'a': 100, 'b': 200, 'c': 300, 'd': 400},
5           { 'a': 1000, 'b': 2000, 'c': 3000, 'd': 4000 } }
6 df = pd.DataFrame(mydict)
7 df
```

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

Gambar 8.6 Teori 4

8.2.1.5 Apa maksud dari fungsi `tokenizer = Tokenizer(num words=2000)` dan `tokenizer.fit on texts(train content)`

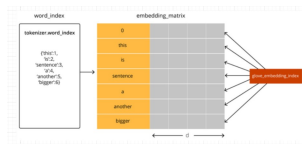
Fungsi tokenizer berfungsi untuk melakukan vektorisasi data kedalam bentuk token sebanyak 2000 kata. Dan selanjutnya akan melakukan fit tokenizer hanya untuk data training saja tidak dengan data testingnya.



Gambar 8.7 Teori 5

8.2.1.6 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 8.8 Teori 6

8.2.1.7 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]])
>>> 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 8.9 Teori 7

8.2.1.8 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([[1., 0., 0.],
       [0., 1., 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_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., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.],
       ...,
       [1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])

```

Gambar 8.10 Teori 8

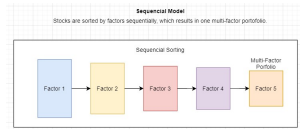
8.2.1.9 Apa maksud dari fungsi di listing 7.2.

```

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.

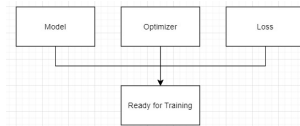


Gambar 8.11 Teori 9

8.2.1.10 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 8.12 Teori 10

8.2.1.11 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.

8.2.1.12 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.

8.2.1.13 Jelaskan dengan ilustrasi gambar buatan sendiri(langkah per langkah) bagaimana perhitungan algoritma konvolusi dengan ukuran stride ($NPM \bmod 3 + 1$) \times ($NPM \bmod 3 + 1$) yang terdapat max pooling.

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:



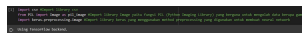
Gambar 8.13 Teori 11

8.2.2 Praktek

8.2.2.1 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 8.14 Hasil No 1

8.2.2.2 Nomor 2

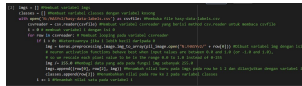
```
1 imgs = [] #Membuat variabel imgs
2 classes = [] #Membuat variabel classes dengan variabel kosong
3 with open('N:/HASYv2/hasy-data-labels.csv') as csvfile: #Membuka
  file hasy-data-labels.csv
4 csvreader = csv.reader(csvfile) #Membuat variabel csvreader
  yang berisi method csv.reader untuk membaca 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 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

```

Hasil:



Gambar 8.15 Hasil No 2

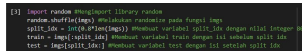
8.2.2.3 Nomor 3

```

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:



Gambar 8.16 Hasil No 1

8.2.2.4 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 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:

```
[4]: import numpy as np #mengimport library numpy dengan inisial np
train_input = np.asarray(list(map(lambda row: row[2], train))) #membuat
test_input = np.asarray(list(map(lambda row: row[2], test))) #membuat
train_output = np.asarray(list(map(lambda row: row[1], train))) #membuat
test_output = np.asarray(list(map(lambda row: row[1], test))) #membuat
```

Gambar 8.17 Hasil No 4

8.2.2.5 Nomor 5

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

Hasil:

```
[2]: from sklearn.preprocessing import LabelEncoder #Mengimport library
from sklearn.preprocessing import OneHotEncoder #Mengimport library
```

Gambar 8.18 Hasil No 5

8.2.2.6 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:

```
[6]: label_encoder = LabelEncoder() #membuat variabel label_encoder dengan
integer_encoded = label_encoder.fit_transform(classes) #membuat variabel
```

Gambar 8.19 Hasil No 6

8.2.2.7 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:

```
[7]: onehot_encoder = OneHotEncoder(sparse=False) #membuat variabel
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1) #mengisi
onehot_encoder.fit(integer_encoded) #mengkonvert variabel
OneHotEncoder(categories='auto', dtype=None, sparse_class='numpy.float64',
handle_unknown='error', sparse=True)
```

Gambar 8.20 Hasil No 7

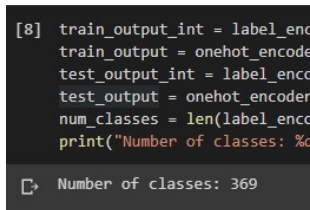
8.2.2.8 Nomor 8

```

1 train_output_int = label_encoder.transform(train_output) #
   Mengkonvert data train output menggunakan 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 menggunakan 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 classess
6 print("Number of classes: %d" % num_classes) #Mencetak hasil dari
   nomer Class berupa persen

```

Hasil:



```

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

```

Number of classes: 369

Gambar 8.21 Hasil No 8

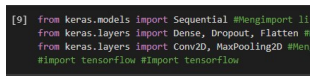
8.2.2.9 Nomor 9

```

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 dari Keras
from keras.layers import Dense, Dropout, Flatten #Mengimport
   library Dense, Dropout, Flatten dari Keras
from keras.layers import Conv2D, MaxPooling2D #Mengimport library
   Conv2D, MaxPooling2D dari Keras

```

Gambar 8.22 Hasil No 9

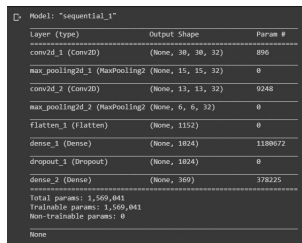
8.2.2.10 Nomor 10


```

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



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)	118672
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 360)	378225
Total params:	1,459,841	
Trainable params:	1,459,841	
Non-trainable params:	0	

Gambar 8.23 Hasil No 10

8.2.2.11 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 module library keras dengan fungsi (callbacks,
tensorboard = keras.callbacks.TensorBoard(log_dir='./tensorboard/logs', write_images=True))
```

Gambar 8.24 Hasil No 11

8.2.2.12 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:

```
C:\train on 107000 samples, validate on 20310 samples
Epoch 1/10: 1.5342 - accuracy: 0.6288 - val_loss: 0.9789 - val_accuracy: 0.7320
Epoch 2/10: 0.9081 - accuracy: 0.7138 - val_loss: 0.8761 - val_accuracy: 0.7512
Epoch 3/10: 0.6786 - accuracy: 0.7570 - val_loss: 0.8186 - val_accuracy: 0.7651
Epoch 4/10: 0.5221 - accuracy: 0.7784 - val_loss: 0.8249 - val_accuracy: 0.7654
Epoch 5/10: 0.7118 - accuracy: 0.7796 - val_loss: 0.8478 - val_accuracy: 0.7658
Epoch 6/10: 0.6938 - accuracy: 0.7885 - val_loss: 0.8351 - val_accuracy: 0.7754
Epoch 7/10: 0.6612 - accuracy: 0.7955 - val_loss: 0.8288 - val_accuracy: 0.7751
Epoch 8/10: 0.6382 - accuracy: 0.8023 - val_loss: 0.8480 - val_accuracy: 0.7674
Epoch 9/10: 0.6081 - accuracy: 0.8071 - val_loss: 0.8311 - val_accuracy: 0.7643
Epoch 10/10: 0.5888 - accuracy: 0.8104 - val_loss: 0.8164 - val_accuracy: 0.7658
Test loss: 0.8164048897773123
Test accuracy: 0.765476884577312
```

Gambar 8.25 Hasil No 12

8.2.2.13 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
                    model.add(Conv2D(32, kernel_size=(3, 3),
9                        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
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 = '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: %.2
f - 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:

[illegible]

Gambar 8.26 Hasil No 13

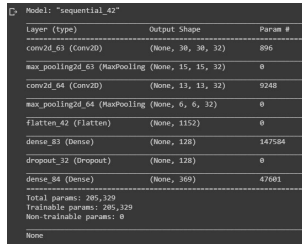
8.2.2.14 Nomor 14

```

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 tigapuluh 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 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
   parameter dan bukan trainable parameter

```

Hasil:



Layer (type)	Output Shape	Param #
conv2d_63 (Conv2D)	(None, 38, 38, 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)	47681
Total params: 285,329		
Trainable params: 285,329		
Non-trainable params: 0		

Gambar 8.27 Hasil No 14

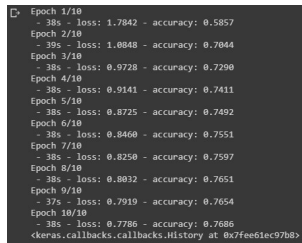
8.2.2.15 Nomor 15

```

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

```

Hasil:



```

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: 0.7298
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: 0.8468 - accuracy: 0.7551
Epoch 7/10
- 38s - loss: 0.8258 - 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 0x7fee5ec97b8>

```

Gambar 8.28 Hasil No 15

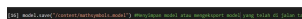
8.2.2.16 Nomor 16

```

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

```

Hasil:



```

model.save("mathsymbols.model")

```

Gambar 8.29 Hasil No 16

8.2.2.17 Nomor 17

```

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

```

Hasil:

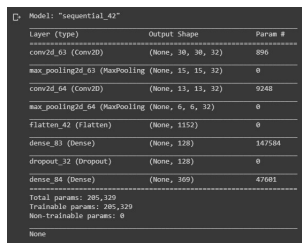


Gambar 8.30 Hasil No 17

8.2.2.18 Nomor 18

```
1 import keras.models #Mengimpor 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_53 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_63 (MaxPooling)	(None, 15, 15, 32)	0
conv2d_54 (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)	147504
dropout_32 (Dropout)	(None, 128)	0
dense_84 (Dense)	(None, 369)	47681
Total params: 265,329		
Trainable params: 265,329		
Non-trainable params: 0		
None		

Gambar 8.31 Hasil No 18

8.2.2.19 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 classess dengan data classess yang di ekspor 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 membay
   immagine 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 predivtion 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
   reverse the one-hot encoding
11  inverted = label_encoder2.inverse_transform([np.argmax(
   prediction)]) #Membuat variabel inverted denagan label
   encoder2 dan menggunakan argmax untuk mencari skor keluaran
   tertinggi
```

```

12 print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
    max(prediction))) #Mencetak prediksi gambar dan confidence
    dari gambar.

```

Hasil:

```

[19] label_encoder = LabelEncoder() # membuat variabel label encoder ke 1 dengan label
    label_encoder.class_ = np.load('centroid/class.npy') #memuat data ke array
    def predict(img_path): #membuat fungsi predict dengan path img
        img = Image.open(img_path).convert('RGB') #membuat gambar dengan path img
        img = ImageProcessing.Image(img).array[0].img.array(img_path) #membuat
        img = img * 255.0 #membuat data yang terdapat pada variabel img menjadi 255
    # di sini prediksi
    prediction = model.predict(img.reshape(1, 32, 32, 3)) #membuat variabel prediction
    # figure out which output neuron had the highest score, and reverse the output
    inverted = label_encoder.inverse_transform(np.argmax(prediction)) #membuat
    print("Prediction: %s, confidence: %.2f" % (inverted[0], np.max(prediction)))

```

Gambar 8.32 Hasil No 19

8.2.2.20 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: 0.90

```

Gambar 8.33 Hasil No 20

8.2.3 Penanganan Error

8.2.3.1 Error

- ProfilerNotRunningError

```

ProfilerNotRunningError: Cannot stop profiling. No profiler is running.

```

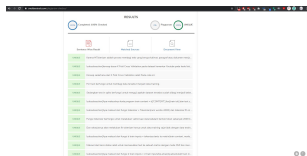
Gambar 8.34 ProfilerNotRunningError

8.2.3.2 Solusi Error

- ProfilerNotRunningError

tambahkan kode `profile=10000000` pada parameter `log_dir`

8.2.4 Bukti Tidak Plagiat



Gambar 8.35 Bukti tidak plagiat

8.2.5 Link Youtube

https://youtu.be/Vq_LZ89hTpg

8.3 1174080 - Handi Hermawan

8.3.1 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	Dipecah menjadi:
• This is Andre's text, isn't it?	<ul style="list-style-type: none"> • This • is • Andre's • text, • isn't • it?

Gambar 8.36 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)

```

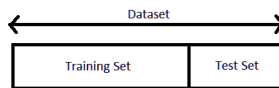
Konsep sederhana dari K Fold Cross Validation ialah Pada code tersebut terdapat kfold yang bertujuan untuk melakukan split data menjadi 5 bagian dari dataset komentar Youtube tersebut. Sehingga dari setiap data yang sudah dibagi tersebut akan menghasilkan presentase dari setiap bagiannya, untuk menghasilkan hasil akhir dengan presentase 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 8.37 Teori 2

3. 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 8.38 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

```

```

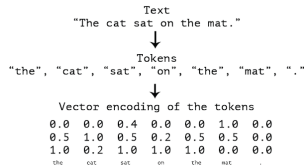
Out[5]:
a      1
b      2
c      3
d      4
Name: 0, dtype: int64

```

Gambar 8.39 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.

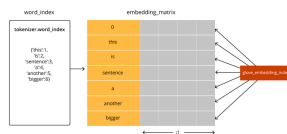
Fungsi tokenizer ini berfungsi untuk melakukan vektorisasi data kedalam bentuk token sebanyak 2000 kata. Dan selanjutnya akan melakukan fit tokenizer hanya untuk data training saja tidak dengan data testingnya. Untuk ilustrasi lihat gambar berikut:



Gambar 8.40 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 8.41 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]])
>>> 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 8.42 Teori 7

8. Jelaskan 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, 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],
       [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)
>>> Y_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(Y_train)
array([[1., 1.],
       [1., 1.],
       [1., 1.],
       ...,
       [1., 1.],
       [1., 1.],
       [1., 1.]])

```

Gambar 8.43 Teori 8

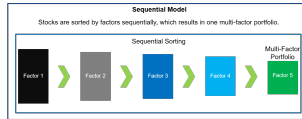
9. Jelaskan apa maksud dari fungsi di listing 7.2. Gambarkan ilustrasi Neural Network nya dari model kode tersebut.

```

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.

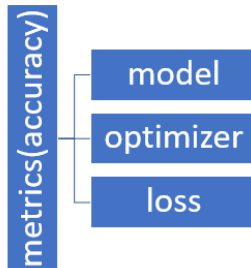


Gambar 8.44 Teori 9

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 8.45 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.

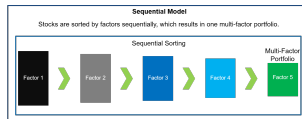
12. Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning.

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.

- Jelaskan dengan ilustrasi gambar buatan sendiri (langkah per langkah) bagaimana perhitungan algoritma konvolusi dengan ukuran stride $(NPM \bmod 3 + 1) \times (NPM \bmod 3 + 1)$ yang terdapat max pooling.

$1174080 \bmod 3 + 1 \times 1174080 \bmod 3 + 1 = 2 \times 2$, adapun ilustrasi gambarnya sebagai berikut :



Gambar 8.46 Teori 9

8.3.2 Praktek Program

1. Soal 1

```
1 # In[1]:import lib
2 # mengimpor library CSV untuk mengolah data ber ekstensi csv
3 import csv
4 #kemudian mengimpor librari Image yang berguna untuk dari
5   PIL atau Python Imaging Library yang berguna untuk
6   mengolah data berupa gambar
7 from PIL import Image as pil_image
8 # kemudian mengimpor librari keras yang menggunakan method
9   preprocessing yang digunakan untuk membuat neural network
10 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
...: # kemudian mengimpor 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 mengimpor librari keras yang menggunakan method preprocessing yang
...:   digunakan untuk membuat neural network
...: import keras.preprocessing.image
...: using tensorflow backend.
```

Gambar 8.47 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 #membuka file hasy-data-labels.csv yang berada di foled
   HASYv2 yang di inialisasi 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(
   pil_image.open("HASYv2/" + row[0]))
18            # neuron activation functions behave best when
   input values are between 0.0 and 1.0 (or -1.0 and 1.0),
19            # so we rescale each pixel value to be in the
   range 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 dilanjutkan dengan variabel img
23            imgs.append((row[0], row[2], img))
24            # menambahkan nilai pada row ke 2 pada variabel
   classes
25            classes.append(row[2])
26            # penambahan nilai satu pada variabel i
27            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 :

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 8.48 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 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 keluaran nya sebagai output, . Hasilnya adalah sebagai berikut :

```

In [5]: 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 8.49 Hasil Soal 4.

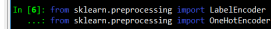
5. Soal 5

```

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

```

Gambar 8.50 Hasil Soal 5.

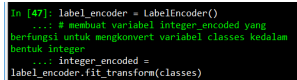
6. Soal 6

```

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 8.51 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 berfungsi untuk melakukan convert integers ke fungsi one hot encoding. Hasilnya adalah sebagai berikut :


```
In [8]: onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape((-1, integer_encoded.shape[1]))
onehot_encoded = onehot_encoder.fit(integer_encoded).transform(integer_encoded)

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

Gambar 8.52 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 classess
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 berfungsi 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
onehot_encoder.transform(train_output_int.reshape((-1, train_output_int.shape[1])))
test_output_int = label_encoder.transform(test_output)
test_output = onehot_encoder.transform(test_output_int.reshape((-1, test_output_int.shape[1])))

num_classes = len(label_encoder.classes_)
print("Number of classes: %d" % num_classes)
Number of classes: 100
```

Gambar 8.53 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 berfungsi 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 8.54 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 tigapuluh dua
   bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
   relu dang menggunakan data train_input
5 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
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 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 pixel
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 # mengkompilae 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())
```

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

```

model.add(Conv2D(kernel_size=(3, 3), activation='relu',
input_shape=train_input_shape))
... model.add(MaxPooling2D(pool_size=(2, 2)))
... model.add(Conv2D(kernel_size=(3, 3), activation='relu'))
... model.add(MaxPooling2D(pool_size=(2, 2)))
... model.add(Flatten())
... model.add(Dense(128), activation='relu')
... model.add(Dropout(0.5))
... model.add(Dense(num_classes, activation='softmax'))
model.compile(loss=keras.losses.categorical_crossentropy, optimizer='adam',
metrics=['accuracy'])
print(model.summary())
Model: "sequential_5"
Layer (type) Output Shape Param #
-----
conv2d_5 (Conv2D) (None, 10, 10, 32) 896
max_pooling2d_5 (MaxPooling2D) (None, 5, 5, 32) 0
conv2d_7 (Conv2D) (None, 10, 10, 32) 9248
max_pooling2d_7 (MaxPooling2D) (None, 5, 5, 32) 0
flatten_5 (Flatten) (None, 1352) 0
dense_9 (Dense) (None, 128) 147344
dropout_4 (Dropout) (None, 128) 0
dense_10 (Dense) (None, 100) 47001
Total params: 265,120
Trainable params: 265,120
Non-trainable params: 0

```

Gambar 8.55 Hasil Soal 10.

11. Soal 11

```

1 # In[11]: import sequential
2 # mengimpor 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 berfungsi untuk melakukan load callbacks pada library keras. Hasilnya adalah sebagai berikut :

```

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

```

Gambar 8.56 Hasil Soal 11.

12. Soal 12

```

1 # In[12]: 5menit kali 10 epoch = 50 menit
2 # fungsi model ditambahkan 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 berfungsi untuk melakukan load epoch yang akan di training dan diberikan hasil selama 10 kali epoch. Hasilnya adalah sebagai berikut :

```
Epoch 1/10: loss: 1.5816 - accuracy: 0.6200 - val_loss: 1.6132 - val_accuracy: 0.7281
WARNING:tensorflow: From C:\Users\Amin\Anaconda11\lib\site-packages\tensorflow\tensorboard\tensorboard_viz.py:140: the name tf.summary is deprecated. Please use tf.compat.v1.summary instead.

Epoch 2/10:
- 170s - loss: 1.0001 - accuracy: 0.7248 - val_loss: 0.8099 - val_accuracy: 0.7521
- 187s - loss: 0.8901 - accuracy: 0.7677 - val_loss: 0.8852 - val_accuracy: 0.7562
Epoch 3/10:
- 220s - loss: 0.8229 - accuracy: 0.7612 - val_loss: 0.8368 - val_accuracy: 0.7629
Epoch 4/10:
- 211s - loss: 0.7793 - accuracy: 0.7718 - val_loss: 0.8435 - val_accuracy: 0.8008
Epoch 5/10:
- 217s - loss: 0.7318 - accuracy: 0.7864 - val_loss: 0.8502 - val_accuracy: 0.8342
Epoch 6/10:
- 189s - loss: 0.7034 - accuracy: 0.7919 - val_loss: 0.8566 - val_accuracy: 0.8014
Epoch 7/10:
- 186s - loss: 0.6705 - accuracy: 0.7924 - val_loss: 0.8402 - val_accuracy: 0.8019
Epoch 8/10:
- 177s - loss: 0.6512 - accuracy: 0.7985 - val_loss: 0.8531 - val_accuracy: 0.7681
Epoch 9/10:
- 172s - loss: 0.6312 - accuracy: 0.8021 - val_loss: 0.8792 - val_accuracy: 0.7611
val_loss: 0.8000000000000000
val accuracy: 0.7600000000000000
```

Gambar 8.57 Hasil Soal 12.

13. Soal 13

```
1 # In[13]:try various model configurations and parameters to
    find the best
2 # mengimpor librari time
3 import time
4 #membuat variabel result dengan array kosong
5 results = []
6 # melakukan looping dengan ketentuan konvolusi 2 dimensi 1 2
7 for conv2d_count in [1, 2]:
8     # menentukan ukuran besaran fixcel dari data atau konvert
9     # 1 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
13        # 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
18            # dari 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
23                    # model dengan konvolusi 2 dimensi 32 bit didalamnya dan
24                    # membuat kernel dengan ukuran 3 x 3 dan rumus aktivasi relu
25                    # dan data 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                    # menambahkan method add pada variabel
31                    # model dengan konvolusi 2 dimensi 32 bit dengan ukuran
32                    # kernel 3 x3 dan fungsi aktivasi relu
33                    model.add(Conv2D(32, kernel_size=(3, 3),
34                                    activation='relu'))
35                # menambahkan method add pada variabel model
36                # dengan isian method Max pooling berdimensi 2 dengan
37                # ukuran fixcel 2 x 2.
38                model.add(MaxPooling2D(pool_size=(2, 2)))
39                # merubah feature gambar menjadi 1 dimensi vektor
40                model.add(Flatten())
```

```

28         # menambahkan method dense untuk pemadatan data
dengan ukuran dense di tentukan dengan rumus fungsi tanh.
29         model.add(Dense(dense_size , activation='tanh'))
30         # membuat ketentuan jika pemangkasan lebih besar
dari 0 persen
31         if dropout > 0.0:
32             # menambahkan method dropout pada model
dengan nilai dari dropout
33             model.add(Dropout(dropout))
34             # menambahkan method dense dengan fungsi num
classs dan rumus softmax
35             model.add(Dense(num_classes , activation='softmax'
))
36             # mongkompile variabel model dengan hasil loss
optimasi dan akurasi matrix
37             model.compile(loss='categorical_crossentropy' ,
optimizer='adam' , metrics=['accuracy'])
38             # melakukan log pada dir
39             log_dir = './logs/conv2d-%d-dense-%d-dropout-%.2f
' % (conv2d_count , dense_size , dropout)
40             # membuat variabel tensorboard dengan isian dari
library keras dan nilai dari log_dir
41             tensorboard = keras.callbacks.TensorBoard(log_dir
=log_dir)
42             # membuat variabel start dengan isian dari
library time menggunakan method time
43
44             start = time.time()
45             # menambahkan method fit pada model dengan data
dari train input train output nilai batch nilai epoch
verbose nilai 20 persen validation split dan callback
dengan nilai tnsorboard.
46             model.fit(train_input , train_output , batch_size
=32, epochs=10,
47                         verbose=0, validation_split=0.2,
callbacks=[tensorboard])
48             # membuat variabel score dengan nilai evaluasi
dari model menggunakan data tes input dan tes output
49             score = model.evaluate(test_input , test_output ,
verbose=2)
50             # membuat variabel end
51             end = time.time()
52             # membuat variabel elapsed
53             elapsed = end - start
54             # mencetak hasil perhitungan
55             print("Conv2D count: %d, Dense size: %d, Dropout:
%.2f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" % (
conv2d_count , dense_size , dropout , score[0] , score[1] ,
elapsed))
56             results.append((conv2d_count , dense_size , dropout
, score[0] , score[1] , elapsed))

```

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

Epoch	Model	Loss	Accuracy	Time
1	128	0.48	0.75	401 sec
2	128	0.47	0.76	401 sec
3	128	0.46	0.77	401 sec
4	128	0.45	0.78	401 sec
5	128	0.44	0.79	401 sec
6	128	0.43	0.80	401 sec
7	128	0.42	0.81	401 sec
8	128	0.41	0.82	401 sec
9	128	0.40	0.83	401 sec
10	128	0.39	0.84	401 sec
11	128	0.38	0.85	401 sec
12	128	0.37	0.86	401 sec
13	128	0.36	0.87	401 sec
14	128	0.35	0.88	401 sec
15	128	0.34	0.89	401 sec
16	128	0.33	0.90	401 sec
17	128	0.32	0.91	401 sec
18	128	0.31	0.92	401 sec
19	128	0.30	0.93	401 sec
20	128	0.29	0.94	401 sec
21	128	0.28	0.95	401 sec
22	128	0.27	0.96	401 sec
23	128	0.26	0.97	401 sec
24	128	0.25	0.98	401 sec

Gambar 8.58 Hasil Soal 13.

14. Soal 14

```

1 # In[14]:rebuild/retrain a model with the best parameters (
    from the search) and use all data
2 # membuat variabel model dengan isian librari Sequential
3 model = Sequential()
4 # variabel model di tambahkan librari Conv2D tigapuluh dua
    bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
    relu dang menggunakan data train_input
5 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
    input_shape=np.shape(train_input[0])))
6 # variabel model di tambahkan dengan lib MaxPooling2D dengan
    ketentuan ukuran 2 x 2 pixel
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 pixel
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 # mengcompile data model untuk mendapatkan data loss akurasi
    dan optimasi
21 model.compile(loss='categorical_crossentropy', optimizer='
    adam', metrics=['accuracy'])
22 # mencetak variabel model kemudian memunculkan kesimpulan
    berupa data total parameter, trainable parameter dan bukan
    trainable parameter
23 print(model.summary())

```

Kode di atas berfungsi 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(MaxPooling2D(pool_size=(2, 2), activation='relu'))
... 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())
Model: "sequential_5"
Layer (type) Output Shape Param #
-----
conv2d_6 (Conv2D) (None, 30, 30, 32) 896
max_pooling2d_6 (MaxPooling2D) (None, 15, 15, 32) 0
conv2d_7 (Conv2D) (None, 13, 13, 32) 824
max_pooling2d_7 (MaxPooling2D) (None, 6, 6, 32) 0
flatten_5 (Flatten) (None, 1152) 0
dense_9 (Dense) (None, 128) 147384
dropout_4 (Dropout) (None, 128) 0
dense_10 (Dense) (None, 100) 4700
Total params: 205,320
Trainable params: 205,320
Non-trainable params: 0
None
```

Gambar 8.59 Hasil Soal 14.

15. Soal 15

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

Kode di atas berfungsi 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=, epochs=, verbose=)
Epoch 1/10
- 116s - loss: 1.7795 - accuracy: 0.5871
Epoch 2/10
- 114s - loss: 1.0744 - accuracy: 0.7074
Epoch 3/10
- 110s - loss: 0.9564 - accuracy: 0.7320
Epoch 4/10
- 115s - loss: 0.8077 - 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
- 116s - loss: 0.8075 - accuracy: 0.7632
Epoch 8/10
- 122s - loss: 0.7917 - accuracy: 0.7663
Epoch 9/10
- 120s - loss: 0.7765 - accuracy: 0.7707
Epoch 10/10
- 118s - loss: 0.7637 - accuracy: 0.7748
Out[16]: keras.callbacks.CallbackHistory at 0x21637eb348
```

Gambar 8.60 Hasil Soal 15.

16. Soal 16

```

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

```

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

```

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

```

Gambar 8.61 Hasil Soal 16.

17. Soal 17

```

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

```

Kode di atas berfungsi 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 8.62 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 # mengimpor 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 berfungsi 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, 30, 30, 32)	896
max_pooling2d_68 (MaxPooling)	(None, 15, 15, 32)	0
conv2d_69 (conv2d)	(None, 13, 13, 32)	9344
max_pooling2d_69 (MaxPooling)	(None, 6, 6, 32)	0
flatten_45 (flatten)	(None, 1152)	0
dense_89 (dense)	(None, 128)	147384
dropout_34 (dropout)	(None, 128)	0
dense_90 (dense)	(None, 309)	47041
Total params: 285,329		
Trainable params: 285,329		
Non-trainable params: 0		
None		

Gambar 8.63 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
  ekspor 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 membuay image 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 denagan label encoder2 dan
      menggunakan argmax untuk mencari skor luaran tertinggi
19    inverted = label_encoder2.inverse_transform([np.argmax(
      prediction)])
20    # mencetak prediksi gambar dan confidence dari gambar.
21    print("Prediction: %s, confidence: %.2f" % (inverted[0],
      np.max(prediction)))

```

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

layer (type)	output shape	params
conv2d_08 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_08 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_09 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_09 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_45 (Flatten)	(None, 1152)	0
dense_80 (Dense)	(None, 128)	147364
dropout_34 (Dropout)	(None, 128)	0
dense_90 (Dense)	(None, 309)	47081
Total params: 266,320		
Trainable params: 266,320		
Non-trainable params: 0		
None		

Gambar 8.64 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 berfungsi untuk menunjukkan hasil akhir prediksi. Hasilnya adalah sebagai berikut :

```
# grab an image (we'll just use a random training image for demonstration purposes)
predict("nsrsv2/hasy-data/v2-00010.png")

Prediction: A, confidence: 0.87

predict("nsrsv2/hasy-data/v2-00780.png")

Prediction: \pi, confidence: 0.58

predict("nsrsv2/hasy-data/v2-00780.png")

Prediction: \alpha, confidence: 0.88
```

Gambar 8.65 Hasil Soal 20.

8.3.3 Penanganan Error

1. KeyboardInterrupt

```
File "C:\python-input-48-d05151581f6a", line 15, in <module>
img = keras.preprocessing.image.img_to_array(pil_image.open("HSV2/" + row[0]))

File "C:\Users\FAHMI-PC\AppData\Local\site-packages\PIL\image.py", line 2389, in open
fp = builtins.open(filename, "rb")

KeyboardInterrupt
```

Gambar 8.66 KeyboardInterrupt

2. Cara Penanganan Error

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

8.3.4 Bukti Tidak Plagiat



Gambar 8.67 Bukti Tidak Melakukan Plagiat Chapter 7

8.3.5 Link Video Youtube

<https://youtu.be/pV9yrZNEZj4>

8.4 1174079 - Chandra Kirana Poetra

8.4.1 Teori

8.4.1.1 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



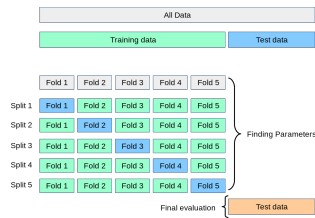
Gambar 8.68 Teori 1

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

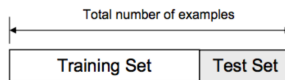
Kfold diatas bertujuan untuk membagi data kedalam 5 bagian yang nantinya akan menampung komentar dari youtube. data yang dibagi ini nanti akan dihitung dan akan menghasilkan presentase yang bisa dibilang cukup baik



Gambar 8.69 Teori 2

8.4.1.3 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 8.70 Teori 3

8.4.1.4 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 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

```

```

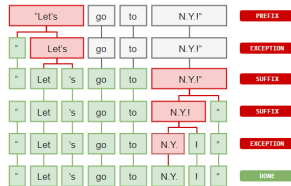
Out[2]:
a      1
b      2
c      3
d      4
Name: 0, dtype: int64

```

Gambar 8.71 Teori 4

8.4.1.5 Apa maksud dari fungsi `tokenizer = Tokenizer(num words=2000)` dan `tokenizer.fit on texts(train content)`

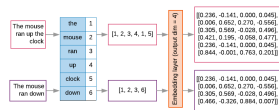
Berfungsi untuk melakukan proses vektorisasi data menjadi token sebanyak 2000 kata



Gambar 8.72 Teori 5

8.4.1.6 Apa maksud dari fungsi $d \text{ train inputs} = \text{tokenizer.texts to matrix}(\text{train content, mode='tfidf'})$ dan $d \text{ test inputs} = \text{tokenizer.texts to matrix}(\text{test content, mode='tfidf'})$

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



Gambar 8.73 Teori 6

8.4.1.7 Apa maksud dari fungsi $d \text{ train inputs} = d \text{ train inputs} / \text{np.amax}(\text{np.absolute}(d \text{ train inputs}))$ dan $d \text{ test inputs} = d \text{ test inputs} / \text{np.amax}(\text{np.absolute}(d \text{ test inputs}))$

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)
3
# Maximum of the flattened array
>>> np.amax(a, axis=0)
array([2, 3])
# Maxima along the first axis
>>> np.amax(a, axis=1)
array([1, 3])
# Maxima along the second axis
>>> 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 8.74 Teori 7

8.4.1.8 Apa maksud fungsi dari $d \text{ train outputs} = \text{np utils.to categorical}(d[\text{'CLASS'}].\text{iloc}[\text{train idx}])$ dan $d \text{ test outputs} = \text{np utils.to categorical}(d[\text{'CLASS'}].\text{iloc}[\text{test idx}])$ dalam

kode program

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

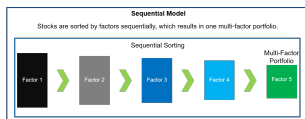
```
>>> V_train
array([[1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.]])
>>> V_train.flatten()
array([1., 0., 0., ..., 1., 0., 0.], dtype=int64)
>>> V_train
array([[1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0.,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

Gambar 8.75 Teori 8

8.4.1.9 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.



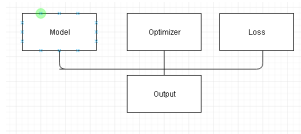
Gambar 8.76 Teori 9

8.4.1.10 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. `categorical_crossentropy` merupakan function loss yang kita gunakan untuk melakukan perhitungan, sementara optimizer yang kita gunakan

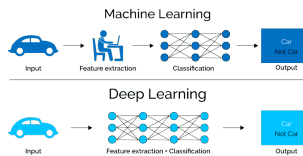
adalah adam, selain adam, ada juga SGD, RMSprop, dan lain lain dan metrics accuracy adalah value yang akan kita cari nilainya



Gambar 8.77 Teori 10

8.4.1.11 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



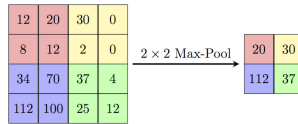
8.4.1.12 Apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning

Komponen Neural network 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, processor, ram, dan cpu untuk melakukan prosesnya

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

8.4.1.13 Jelaskan dengan ilustrasi gambar buatan sendiri(langkah per langkah) bagaimana perhitungan algoritma konvolusi dengan ukuran stride $(NPM \bmod 3 + 1) \times (NPM \bmod 3 + 1)$ yang terdapat max pooling.

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 8.78 Teori 11

8.4.2 Praktek

8.4.2.1 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

```

8.4.2.2 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

```

8.4.2.3 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

```

8.4.2.4 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

```

8.4.2.5 Nomor 5

```

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

```

8.4.2.6 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

```

8.4.2.7 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

```

8.4.2.8 Nomor 8

```

1 train_output_int = label_encoder.transform(train_output) #
  Mengkonvert data train output menggunakan 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 menggunakan 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 classess
6 print("Number of classes: %d" % num_classes) #Mencetak hasil dari
  nomer Class berupa persen

```

8.4.2.9 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

```

8.4.2.10 Nomor 10

```

1 model = Sequential() #Membuat variabel model dengan isian library
  Sequential
2 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
3   input_shape=np.shape(train_input[0]))) #Variabel
  model di tambahkan library Conv2D tigapuluh 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']) #Mengompile data model untuk
    mendapatkan data loss akurasi dan optimasi
13 print(model.summary()) #Mencetak variabel model kemudian
    memunculkan kesimpulan berupa data total parameter, trainable
    parameter dan bukan trainable parameter

```

8.4.2.11 Nomor 11

```

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

```

8.4.2.12 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])

```

8.4.2.13 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
    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
                    model.add(Conv2D(32, kernel_size=(3, 3),
9                                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
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 hasil 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: %.2
f - 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))

```

8.4.2.14 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 tigapuluh 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 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']) #Mengompile data model untuk
  mendapatkan data loss akurasi dan optimasi
11 print(model.summary()) #Mencetak variabel model kemudian
  memunculkan kesimpulan berupa data total parameter, trainable
  parameter dan bukan trainable parameter

```

8.4.2.15 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

```

8.4.2.16 Nomor 16

```

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

```

8.4.2.17 Nomor 17

```

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

```

8.4.2.18 Nomor 18

```

1 import keras.models #Mengimpor 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

```

8.4.2.19 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 classess dengan data classess yang di ekspor 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 membay
   immagine 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 predivtion 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
   reverse the one-hot encoding
11 inverted = label_encoder2.inverse_transform([np.argmax(
   prediction)]) #Membuat variabel inverted denagan label
   encoder2 dan menggunakan argmax untuk mencari skor keluaran
   tertinggi
12 print("Prediction: %s, confidence: %.2f" % (inverted[0], np.
   max(prediction))) #Mencetak prediksi gambar dan confidence
   dari gambar.

```

8.4.2.20 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

```

8.4.3 Penanganan Error

8.4.3.1 Error

- NameError

```
NameError: name 'np' is not defined
```

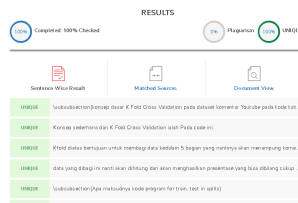
Gambar 8.79 NameError

8.4.3.2 Solusi Error

- NameError

Pastikan sudah diimport atau cek typo

8.4.4 Bukti Tidak Plagiat



Gambar 8.80 Bukti tidak plagiat

8.4.5 Link Youtube

<https://youtu.be/AHJOIZJYd9I>

8.5 Muhammad Reza Syachrani - 1174084

8.5.1 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 kfold yang berisi fungsi StratifiedKFold yang memiliki parameter n_splits=5 yang berarti melakukan

3. Jelaskan apa maksudnya kode program for train, test in splits.dilengkapi dengan ilustrasi atau gambar.
kode program for train fungsinya untuk melakukan training terhadap data yang telah di deklarasikan sebelumnya. sedangkan kode program

test in split fungsinya untuk membatasi jumlah data yang akan digunakan.

```
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 8.82 Teori 3

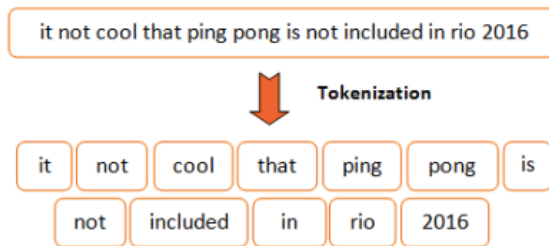
4. Jelaskan apa maksudnya kode program `train_content = d['CONTENT'].iloc[train_idx]` dan `test_content = d['CONTENT'].iloc[test_idx]`. dilengkapi dengan ilustrasi atau gambar.

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 8.83 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)` untuk membuat membaca data token/indeks teks yang telah di masukan kedalam data `train_konten`.



Gambar 8.84 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 variabel `d_train_inputs` melakukan tokenizer dari bentuk teks menjadi bentuk matrix yang berurutan dari data `train_content` dengan mode `tfidf` 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` argument dari fungsi, yaitu :

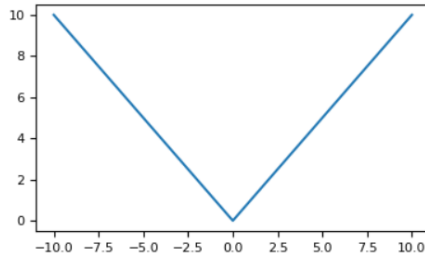
- `binary` : apakah setiap kata ada dalam dokumen atau tidak (default)
- `count` : jumlah setiap kata dalam dokumen
- `tfidf` : skor untuk Text Frequency-Inverse Document frequency (TF-IDF) untuk setiap kata dalam dokumen
- `freq` : frekuensi setiap kata sebagai rasio kata dalam setiap dokumen.

Gambar 8.85 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 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 8.86 Teori 7

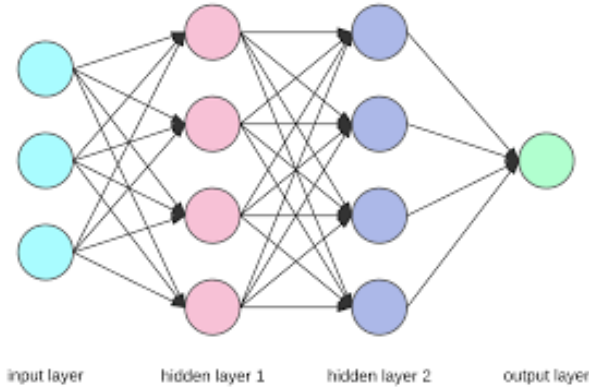
8. Jelaskan apa maksud fungsi dari `d_train_outputs = np_utils.to_categorical(d['CLASS'])` dan `d_test_outputs = np_utils.to_categorical(d['CLASS'].iloc[test_idx])` dalam kode program, dilengkapi dengan ilustrasi atau gambar. fungsi untuk melakukan one-hot encoding merubah nilai vektor dengan bentuk integer yang ada pada atribut class menjadi bentuk matrix biner untuk atribut CLASS sehingga hanya ada dua pilihan yaitu 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 8.87 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".

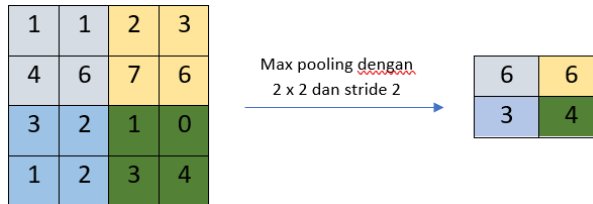
dan dilakukan overfitting atau pemotongan bobot sebesar 0.5 atau 50 persen. Lalu pada layer output terdapat 2 neuron output. Kemudian outputan tersebut diaktivasi menggunakan fungsi softmax.



Gambar 8.88 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 di tata 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/layar 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 $(NPM \bmod 3 + 1) \times (NPM \bmod 3 + 1)$ yang terdapat max pooling

Karena NPM saya 1164084 dan hasil dari $(\text{NPM} \bmod 3) + 1 = 2$, maka saya menggunakan stride 2 dengan ketentuan max pooling 2×2 .



Gambar 8.89 Teori 13

8.5.2 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 8.90 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 sebagai csvfile
```

```

9     csvreader = csv.reader(csvfile)
10    #membuat variabel csvreader yang berisikan metode reader
    dari library csv yang membaca csvfile.
11    i = 0
12    #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(
pil_image.open("D:/New folder/KB3C/src/1174084/7/HASYv2/"
+ row[0]))
18            #Membuat variabel img yang berisikan fungsi keras
    untuk aktivasi neural network yang membaca data pada
    folder HASYv2 yang dibuka dengan row berparameter 0.
19            # neuron activation functions behave best when
    input values are between 0.0 and 1.0 (or -1.0 and 1.0),
20            # so we rescale each pixel value to be in the
    range 0.0 to 1.0 instead of 0-255
21            img /= 255.0
22            #Membagi data yang berada pada variabel img
    dengan 255.0
23            imgs.append((row[0], row[2], img))
24            #Menambahkan nilai baru pada imgs yaitu row 0,row
    2 dilanjutkan dengan variabel img.
25            classes.append(row[2])
26            # menambahkan nilai pada row ke 2 pada variabel
    classes
27            i += 1
28            #Menambahkan nilai 1 pada variabel i

```

Name	Type	Size	Value
classes	list	168233	['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', '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.pn ...
row	list	4	['hasy-data/v2-168232.png', '1400', '\guillemotleft', '16925']

Gambar 8.91 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 perkalian 80% dengan jumlah dari variabel imgs
9    train = imgs[:split_idx]

```

```

10 #Membuat variabel train yang diisi dengan dengan pemecahan
    index awal pada data variabel split_idx
11 test = imgs[split_idx:]
12 #Membuat variabel test yang diisi dengan pemecahan index
    akhir 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 8.92 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 )
7 #Membuat variabel train_input dengan np method asarray yang
    mana membuat array dengan fungsi list yang didalamnya
    diterapkan fungsi map untuk mengembalikan iterator
8 #dan menggunakan lamba untuk mengecilkan fungsi dari objek
    yang berada pada row 2 dari data train
9 test_input = np.asarray(list(map(lambda row: row[2], test)))
10 #Membuat variabel test_input dengan isi np method asarray
    yang mana membuat array dengan fungsi list yang didalamnya
    diterapkan fungsi map untuk mengembalikan iterator
11 #dan menggunakan lamba untuk mengecilkan fungsi dari objek
    yang berada pada row 2 dari data test
12 train_output = np.asarray(list(map(lambda row: row[1], train))
13 )
14 #Membuat variabel train_output dengan np method asarray yang
    mana membuat array dengan fungsi list yang didalamnya
    diterapkan fungsi map untuk mengembalikan iterator
15 #dan menggunakan lamba untuk mengecilkan fungsi dari objek
    yang berada pada row 1 dari data train
16 test_output = np.asarray(list(map(lambda row: row[1], test)))
17 #Membuat variabel test_output dengan np method asarray yang
    mana membuat array dengan fungsi list yang didalamnya
    diterapkan fungsi map untuk mengembalikan iterator
18 #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 8.93 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 8.94 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
  mengkonversi variabel classes kedalam bentuk integer

```

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

Gambar 8.95 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 8.96 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 8.97 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 MaxPoolinf2D dari Library Keras

```

Gambar 8.98 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),
    activation='relu',
5         input_shape=np.shape(train_input[0])))
6 #Variabel model ditambahkan fungsi Conv2d dengan paramater 32
    filter dengan karnel berukuran 3x3
7 #dengan algoritam activation relu menggunakan data train_input
    mulai dari baris nol.
8 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
9 #Variabel madel ditambahkan fungsi MaxPooling2D dengan
    ketentuan ukuran 2x2
10 model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu'
    ))
11 #Variabel model ditambahkan fungsi Conv2D dengan 32 filter
    dengan konvolusi berukuran 3x3, menggunakan algoritam
    activation relu
12 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
13 #Variabel madel ditambahkan fungsi MaxPooling2D dengan
    ketentuan ukuran 2x2
14 model.add(tf.keras.layers.Flatten())
15 #Variabel model di tambahkan fungsi Flatten
16 model.add(tf.keras.layers.Dense(1024, activation='tanh'))
17 #Variabel model ditambahkan fungsi dense dengan 1024 neuron,
    dan menggunakan algoritma tanh untuk activation
18 model.add(tf.keras.layers.Dropout(0.5))
19 #Variabel model di tambahkan fungsi Dropout sebesar 50% untuk
    mencegah terjadinya overfitting
20 model.add(tf.keras.layers.Dense(num_classes, activation='
    softmax'))
21 #Variabel model ditambahkan fungsi Dense dengan parameter
    variabel num_classes menggunakan activation softmax
22 model.compile(loss='categorical_crossentropy', optimizer='
    adam',

```

```

23         metrics=['accuracy'])
24 #Variabel model di compile dengan parameter loss , matrik , dan
    optimasi
25 print(model.summary())
26 #Menampilkan model yang telah dibuat.

```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
=====		
conv2d_1 (Conv2D)	(None, 30, 30, 32)	896
=====		
max_pooling2d_1 (MaxPooling2)	(None, 15, 15, 32)	0
=====		
conv2d_2 (Conv2D)	(None, 13, 13, 32)	9248
=====		
max_pooling2d_2 (MaxPooling2)	(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 8.99 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 8.100 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         #epoche=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
    perhitungan validasi
12        callbacks=[tensorboard])
13        #mengeksekusi tensorboard dimana digunakan untuk
    visualisasikan parameter training, metrik, hiperparameter
    pada nilai/data yang diproses
14
15 score = model.evaluate(test_input, test_output, verbose=2)
16 #Membuat variabel score dengan isi fungsi evulate dari model
    dengan paramater test_input, test_output dan verbose=2
17 #untuk memprediksi output dan input
18 print('Test loss:', score[0])
19 #Menampilkan score optimasi dengan ketentuan nilai parameter
    0
20 print('Test accuracy:', score[1])
21 #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: 0.8780 - 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: 0.76230596

```

Gambar 8.101 Hasil Praktek 12

13. Jelaskan kode program pada blok In[13]

```

1 # In[13]:try various model configurations and parameters to
    find the best
2
3 import time
4 #import library time
5
6 results = []
7 #Membuat variabel result dengan isi array kosong
8 for conv2d_count in [1, 2]:
9 #Melakukan looping menggunakan convd2d_count dengan ketentuan
    konvolusi 2 dimensi yaitu 1, 2

```

```

10     for dense_size in [128, 256, 512, 1024, 2048]:
11         #Melakukan looping menggunakan ukuran dari densenya yaitu
            123, 256, 512, 1024, 2048.
12         for dropout in [0.0, 0.25, 0.50, 0.75]:
13             #Melakukan looping menggunakan dropout dengan
            ketentuan 0%, 25%, 50%, 75% untuk memangkas data.
14             model = tf.keras.Sequential()
15             #Membuat variabel model dengan isi fungsi
            sequential.
16             for i in range(conv2d_count):
17                 #Melakukan looping menggunakan i dengan jarak
            hasil konvolusi
18                 if i == 0:
19                     #jika nilai i sama dengan 0
20                         model.add(tf.keras.layers.Conv2D(32,
            kernel_size=(3, 3), activation='relu', input_shape=np.
            shape(train_input[0])))
21                         #Variabel model ditambahkan fungsi Conv2d
            dengan paramater 32 filter dengan karnel berukuran 3x3
22                         #dengan algoritam activation relu
            menggunakan data train_input mulai dari baris nol.
23                     else:
24                         #jika tidak
25                         model.add(tf.keras.layers.Conv2D(32,
            kernel_size=(3, 3), activation='relu'))
26                         #Variabel model akan ditambahkan fungsi
            Conv2d dengan paramater 32 filter dengan karnel berukuran
            3x3
27                         #dengan algoritam activation relu tanpa
            ada parameter input_shape.
28                         model.add(tf.keras.layers.MaxPooling2D(
            pool_size=(2, 2)))
29                         #Variabel model ditambahkan fungsi
            MaxPooling2D dengan ketentuan ukuran 2x2
30                         model.add(tf.keras.layers.Flatten())
31                         #Variabel model di tambahkan fungsi Flatten
32                         model.add(tf.keras.layers.Dense(dense_size,
            activation='tanh'))
33                         #Variabel model ditambahkan fungsi dense dengan
            jumlah dense yang digunakan, dan menggunakan algoritma
            tanh untuk activation
34                         if dropout > 0.0:
35                             #jika nilai dari dropout lebih besar dari 0.0
36                             model.add(tf.keras.layers.Dropout(dropout))
37                             #Variabel model di tambahkan fungsi Dropout
            sebesar dropout yang digunakan untuk mencegah terjadinya
            overfitting
38                             model.add(tf.keras.layers.Dense(num_classes,
            activation='softmax'))
39                             #Variabel model ditambahkan fungsi Dense dengan
            parameter variabel num_classes menggunakan activation
            softmax
40                             model.compile(loss='categorical_crossentropy',
            optimizer='adam', metrics=['accuracy'])
41                             #Variabel model di compile dengan parameter loss,
            matrik, dan optimasi

```

```

42     log_dir = '.\\logs\\conv2d-%d-dense-%d-dropout-
%.2f' % (conv2d_count, dense_size, dropout)
43     #Melakukan log pada dir
44     tensorboard = keras.callbacks.TensorBoard(log_dir
=log_dir)
45     #Mengisi variabel tensorboard dengan isian dari
library keras dan nilai dari log dir
46
47     start = time.time()
48     #Membuat variabel start dengan isi fungsi time
dari library time
49     model.fit(train_input, train_output, batch_size
=32, epochs=10,
50             verbose=0, validation_split=0.2,
callbacks=[tensorboard])
51     #Melakukan fitting pada model dengan parameter
test_input, test_output, batch_size, epochs, verbose,
validation_split dan callbacks
52     score = model.evaluate(test_input, test_output,
verbose=2)
53     #Membuat variabel score dengan nilai evaluasi
dari model menggunakan data tes input dan tes output
dengan verbose adalah 2
54     end = time.time()
55     #Membuat variabel end dengan isi fungsi time dari
library time
56     elapsed = end - start
57     #Membuat variabel elapse yang diisi dengan nilai
hasil waktu end dikurangi start
58     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))
59     results.append((conv2d_count, dense_size, dropout
, score[0], score[1], elapsed))
60     #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: 0.9202 - 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: 0.8837 - 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: 0.8812 - accuracy: 0.7705
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 8.102 Hasil Praktek 13

14. Jelaskan kode program pada blok In[14]

```

1 # In[14]:rebuild/retrain a model with the best parameters (
from the search) and use all data

```

```
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', input_shape=np.shape(train_input[0])))
6 #Variabel model ditambahkan fungsi Conv2d dengan paramater 32
7     filter dengan karnel berukuran 3x3
8 #dengan algoritam activation relu menggunakan data train_input
9     mulai dari baris nol.
10 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
11 #Variabel model ditambahkan fungsi MaxPooling2D dengan
12     ketentuan ukuran 2x2
13 model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu
14     '))
15 #Variabel model ditambahkan fungsi Conv2D dengan 32 filter
16     dengan konvolusi berukuran 3x3, menggunakan algoritam
17     activation relu
18 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
19 #Variabel model ditambahkan fungsi MaxPooling2D dengan
20     ketentuan ukuran 2x2
21 model.add(tf.keras.layers.Flatten())
22 #Variabel model di tambahkan fungsi Flatten
23 model.add(tf.keras.layers.Dense(128, activation='tanh'))
24 #Variabel model ditambahkan fungsi dense dengan 128 neuron,
25     dan 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='
30     softmax'))
31 #Variabel model ditambahkan fungsi Dense dengan parameter
32     variabel num_classes menggunakan activation softmax
33 model.compile(loss='categorical_crossentropy', optimizer='
34     adam', metrics=['accuracy'])
35 #Variabel model di compile dengan parameter loss, matrik, dan
36     optimasi
37 print(model.summary())
38 #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
Total params: 205,329		
Trainable params: 205,329		
Non-trainable params: 0		
None		

Gambar 8.103 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
4     np.concatenate((train_output, test_output)),
5     #kemudian join numpy menggunakan data train_output
   test_output
6     batch_size=32, epochs=10, verbose=2)
7     #menggunaakn 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 8.104 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('In[16]:save the trained model', 'D:/New folder/KB3C/src/1174084/7/1174084.py')
INFO:tensorflow:Assets written to: mathsymbols.model/assets

```

Gambar 8.105 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('In[17]:save Label encoder (to reverse one-hot encoding)', 'D:/New folder/KB3C/src/1174084/7/1174084.py')

```

Gambar 8.106 Hasil Praktek 17

18. Jelaskan kode program pada blok In[18]

```

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

```

```

5 #Mengimport library keras.models
6 model2 = keras.models.load_model("mathsymbols.model")
7 #Membuat variabel model2 dengan isi hasil load model dari
  mathsymbols.model
8 print(model2.summary())
9 #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
Total params: 205,329		
Trainable params: 205,329		
Non-trainable params: 0		
None		

Gambar 8.107 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
  LabelEncoder
4 label.encoder2.classes_ = np.load('classes.npy')
5 #Menggunakan method classess dengan data classess.npy yang di
  eksport.
6
7 def predict(img_path):
8     #Membuat fungsi predict dengan path img
9     newimg = keras.preprocessing.image.img_to_array(pil_image
  .open(img_path))
10    #Membuat variabel newping dengan isi mengubah bentuk
  image menjadi array dan membuka data berdasarkan img path
11    newimg /= 255.0
12    #Membagi data yang terdapat pada newimg dengan 255.0
13
14    # do the prediction
15    prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
16    #Membuat variabel prediction dengan isian variabel model2
  menggunakan fungsi predict dengan syarat variabel newimg
  dengan data reshape
17
18    # figure out which output neuron had the highest score ,
  and reverse the one-hot encoding

```

```

19 inverted = label_encoder2.inverse_transform([np.argmax(
    prediction)]) # argmax finds highest-scoring output
20 #Membuat variabel inverted dengan label encoder2 dan
    menggunakan argmax untuk mencari skor luaran tertinggi
21 print("Prediction: %s, confidence: %.2f" % (inverted[0],
    np.max(prediction)))
22 #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 8.108 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 for demonstration purposes)
2 predict("D:/New folder/KB3C/src/1174084/7/HASYv2/hasy-data/v2
    -00010.png")
3 #Melakukan prediksi dari pelatihan dari gambar v2-00010.png
4 predict("D:/New folder/KB3C/src/1174084/7/HASYv2/hasy-data/v2
    -00500.png")
5 #Melakukan prediksi dari pelatihan dari gambar v2-00500.png
6 predict("D:/New folder/KB3C/src/1174084/7/HASYv2/hasy-data/v2
    -00700.png")
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 8.109 Hasil Praktek 20

8.5.3 Bukti Tidak Plagiat



Gambar 8.110 plagiarism

8.5.4 Link Video Youtube

<https://youtu.be/djRkJBfOFJs>

