

## **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 adipisicing 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
   conferences on},
9   pages={255--260},
10  year={2017},
11  organization={IEEE}
12 }
```



**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



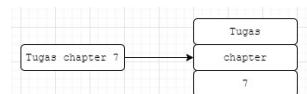
**Gambar 8.2** Kecerdasan Buatan.

### 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.



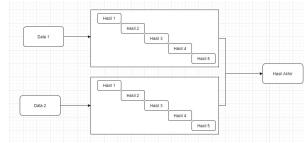
**Gambar 8.3** Teori 1

### 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'])
```

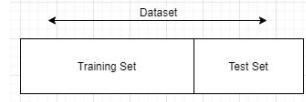
terdapat kfold yang bertujuan untuk melakukan split data menjadi 5 bagian dari dataset komentar Youtube tersebut. Sehingga dari setiap data yang sudah dibagi tersebut akan menghasilkan persentase dari setiap bagiannya, untuk menghasilkan hasil akhir dengan persentase yang cukup baik.



**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 diinginkan.

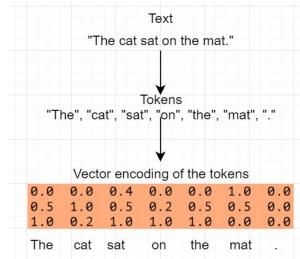
```
1 import pandas as pd
2
3 mydict = [ { 'a': 1, 'b': 2, 'c': 3, 'd': 4 },
4           { 'a': 100, 'b': 200, 'c': 300, 'd': 400 },
5           { 'a': 1000, 'b': 2000, 'c': 3000, 'd': 4000 } ]
6 df = pd.DataFrame(mydict)
7 df
```

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

**Gambar 8.6** Teori 4

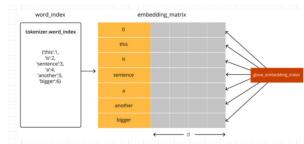
#### 8.2.1.5 Apa maksud dari fungsi `tokenizer = Tokenizer(num_words=2000)` dan `tokenizer.fit_on_texts(train content)`

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.absolute(d train inputs))` dan `d test inputs = d test inputs/np.absolute(d test inputs))`

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

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

Gambar 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.

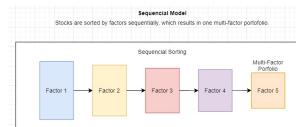
```
>>> v_train
array([[1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.]], dtype=int64)
>>> v_train.flatten()
array([1, 0, 0, ..., 1, 0, 0], dtype=int64)
>>> v_train
array([[1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       ...,
       [0., 1.,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

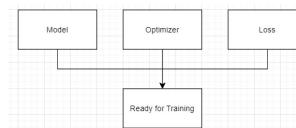
*8.2.1.10 Apa maksud dari fungsi di listing 7.3 dengan parameter tersebut*

```

1 model.compile(loss='categorical_crossentropy', optimizer='adamax'
2 ,
3         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*

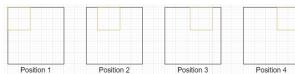
Deep Learning adalah salah satu cabang dari ilmu machine learning yang terdiri dari algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam machine learning dapat digunakan baik untuk 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 ( $NPM \ mod3+1$ )  $\times$  ( $NPM \ mod3+1$ ) yang terdapat max pooling.*

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



**Gambar 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
13     baru pada imgs pada row ke 1 2 dan dilanjutkan dengan
14     variabel img
13         classes.append(row[2]) #Menambahkan nilai pada row ke
12     2 pada variabel classes
14         i += 1 #Menambah nilai satu pada variabel i

```

Hasil:

```

In [1]: def split(img):
    random.shuffle(img)
    split_idx = int(0.8*len(img))
    train = img[:split_idx]
    test = img[split_idx:]
    print(len(train), len(test))

In [2]: split([1, 2, 3, 4, 5])
4 1

```

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:

```

In [1]: import random
In [2]: def split(img):
    random.shuffle(img)
    split_idx = int(0.8*len(img))
    train = img[:split_idx]
    test = img[split_idx:]
    print(len(train), len(test))

In [3]: split([1, 2, 3, 4, 5])
4 1

```

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 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 alias np
train_input = np.asarray([list(map(lambda row: row[2], train))]) #Membuat variabel train_input
test_input = np.asarray([list(map(lambda row: row[2], test))]) #Membuat variabel test_input
train_target = np.asarray([list(map(lambda row: row[1], train))]) #Membuat variabel train_target
test_target = np.asarray([list(map(lambda row: row[1], test))]) #Membuat variabel test_target
```

Gambar 8.17 Hasil No 4

### 8.2.2.5 Nomor 5

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

Hasil:

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

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 isi LabelEncoder
integer_encoded = label_encoder.fit_transform(classes) #Membuat variabel integer_encoded
```

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

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

⇒ Number of classes: 369

```

Gambar 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 li
from keras.layers import Dense, Dropout, Flatten #
from keras.layers import Conv2D, MaxPooling2D #Men
#Import tensorflow #Import tensorflow

```

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',
                 input_shape=np.shape(train_input[0]))) #Variabel
    model di tambahkan library Conv2D tigapuluhan dua bit dengan
    ukuran kernel 3 x 3 dan fungsi penghitungan relu dang
    menggunakan data train_input
4 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
    tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x
    2 pixel
5 model.add(Conv2D(32, (3, 3), activation='relu')) #Variabel model
    di tambahkan dengan library Conv2D 32bit dengan kernel 3 x 3
6 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
    tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x
    2 pixcel
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',
                metrics=['accuracy']) #Mengkompile data model untuk
    mendapatkan data loss akurasi dan optimasi
13 print(model.summary()) #Mencetak variabel model kemudian
    memunculkan kesimpulan berupa data total parameter , trainable
    paremeter dan bukan trainable parameter

```

Hasil:

Model: "sequential_1"		
Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 38, 38, 32)	896
max_pooling2d_1 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_2 (Conv2D)	(None, 15, 15, 32)	9248
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_1 (Flatten)	(None, 1152)	0
dense_1 (Dense)	(None, 1024)	1188072
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 8.23 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) from keras.callbacks import TensorBoard
tensorboard = keras.callbacks.TensorBoard(log_dir='/content/logs/mnist-style')
```

**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
           batch_size=32, #Dengan batch size 32 bit
           epochs=10,
           verbose=2,
           validation_split=0.2,
           callbacks=[tensorboard])

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

Hasil:

```
Train on 187500 samples, validate on 20000 samples
Epoch 1/10
1/10 [00:00:00] - loss: 1.5342 - accuracy: 0.6300 - val_loss: 0.0700 - val_accuracy: 0.7726
2/10 [00:00:00] - loss: 0.9689 - accuracy: 0.7300 - val_loss: 0.0701 - val_accuracy: 0.7952
3/10 [00:00:00] - loss: 0.7959 - accuracy: 0.7559 - val_loss: 0.0700 - val_accuracy: 0.7905
4/10 [00:00:00] - loss: 0.7551 - accuracy: 0.7559 - val_loss: 0.0705 - val_accuracy: 0.7905
5/10 [00:00:00] - loss: 0.7022 - accuracy: 0.7700 - val_loss: 0.0700 - val_accuracy: 0.7904
6/10 [00:00:00] - loss: 0.7159 - accuracy: 0.7790 - val_loss: 0.0700 - val_accuracy: 0.7908
7/10 [00:00:00] - loss: 0.6938 - accuracy: 0.7800 - val_loss: 0.0751 - val_accuracy: 0.7754
8/10 [00:00:00] - loss: 0.6512 - accuracy: 0.7950 - val_loss: 0.0200 - val_accuracy: 0.7713
9/10 [00:00:00] - loss: 0.6023 - accuracy: 0.8023 - val_loss: 0.0400 - val_accuracy: 0.7624
10/10 [00:00:00] - loss: 0.5871 - accuracy: 0.8071 - val_loss: 0.0311 - val_accuracy: 0.7601
Epoch 10/10
1/10 [00:00:00] - loss: 0.5808 - accuracy: 0.8104 - val_loss: 0.0364 - val_accuracy: 0.7608
2/10 [00:00:00] - loss: 0.5750 - accuracy: 0.8114 - val_loss: 0.0364 - val_accuracy: 0.7608
3/10 [00:00:00] - loss: 0.5692 - accuracy: 0.8124 - val_loss: 0.0364 - val_accuracy: 0.7608
4/10 [00:00:00] - loss: 0.5643 - accuracy: 0.8134 - val_loss: 0.0364 - val_accuracy: 0.7608
5/10 [00:00:00] - loss: 0.5604 - accuracy: 0.8144 - val_loss: 0.0364 - val_accuracy: 0.7608
6/10 [00:00:00] - loss: 0.5570 - accuracy: 0.8154 - val_loss: 0.0364 - val_accuracy: 0.7608
7/10 [00:00:00] - loss: 0.5541 - accuracy: 0.8164 - val_loss: 0.0364 - val_accuracy: 0.7608
8/10 [00:00:00] - loss: 0.5516 - accuracy: 0.8174 - val_loss: 0.0364 - val_accuracy: 0.7608
9/10 [00:00:00] - loss: 0.5494 - accuracy: 0.8184 - val_loss: 0.0364 - val_accuracy: 0.7608
10/10 [00:00:00] - loss: 0.5474 - accuracy: 0.8194 - val_loss: 0.0364 - val_accuracy: 0.7608
Test accuracy: 0.76547698267732
```

**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
7     Sequential
8       for i in range(conv2d_count): #Membuat looping untuk
   variabel i dengan jarak dari hasil konvolusi.
9         if i == 0: #Syarat jika i samadengan bobotnya 0
           model.add(Conv2D(32, kernel_size=(3, 3),
activation='relu', input_shape=np.shape(train_input[0]))) #
   Menambahkan method add pada variabel model dengan konvolusi 2
   dimensi 32 bit didalamnya dan membuat kernel dengan ukuran 3
   x 3 dan rumus aktifasi relu dan data shape yang di hitung
   dari data train.
```

```

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

```

Hasil:

Epoch	Count	1. Dense size 128, Dropout 0.00	Loss: 1.20, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.00	Loss: 0.97, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.10	Loss: 0.97, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.20	Loss: 0.97, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.30	Loss: 0.97, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.40	Loss: 0.97, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.50	Loss: 0.97, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.60	Loss: 0.97, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.70	Loss: 0.97, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.80	Loss: 0.97, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.90	Loss: 0.97, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 1.00	Loss: 0.97, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.00	Loss: 1.72, Accuracy: 8.76	Time: 407 sec
0	Conv2D	1. Dense size 128, Dropout 0.10	Loss: 1.72, Accuracy: 8.76	Time: 407 sec
0	Conv2D	1. Dense size 128, Dropout 0.20	Loss: 1.72, Accuracy: 8.76	Time: 407 sec
0	Conv2D	1. Dense size 128, Dropout 0.30	Loss: 1.72, Accuracy: 8.76	Time: 407 sec
0	Conv2D	1. Dense size 128, Dropout 0.40	Loss: 1.72, Accuracy: 8.76	Time: 407 sec
0	Conv2D	1. Dense size 128, Dropout 0.50	Loss: 1.72, Accuracy: 8.76	Time: 407 sec
0	Conv2D	1. Dense size 128, Dropout 0.60	Loss: 1.72, Accuracy: 8.76	Time: 407 sec
0	Conv2D	1. Dense size 128, Dropout 0.70	Loss: 1.72, Accuracy: 8.76	Time: 407 sec
0	Conv2D	1. Dense size 128, Dropout 0.80	Loss: 1.72, Accuracy: 8.76	Time: 407 sec
0	Conv2D	1. Dense size 128, Dropout 0.90	Loss: 1.72, Accuracy: 8.76	Time: 407 sec
0	Conv2D	1. Dense size 128, Dropout 1.00	Loss: 1.72, Accuracy: 8.76	Time: 407 sec
0	Conv2D	1. Dense size 128, Dropout 0.00	Loss: 0.98, Accuracy: 8.76	Time: 516 sec
0	Conv2D	1. Dense size 128, Dropout 0.10	Loss: 0.98, Accuracy: 8.76	Time: 516 sec
0	Conv2D	1. Dense size 128, Dropout 0.20	Loss: 0.98, Accuracy: 8.76	Time: 516 sec
0	Conv2D	1. Dense size 128, Dropout 0.30	Loss: 0.98, Accuracy: 8.76	Time: 516 sec
0	Conv2D	1. Dense size 128, Dropout 0.40	Loss: 0.98, Accuracy: 8.76	Time: 516 sec
0	Conv2D	1. Dense size 128, Dropout 0.50	Loss: 0.98, Accuracy: 8.76	Time: 516 sec
0	Conv2D	1. Dense size 128, Dropout 0.60	Loss: 0.98, Accuracy: 8.76	Time: 516 sec
0	Conv2D	1. Dense size 128, Dropout 0.70	Loss: 0.98, Accuracy: 8.76	Time: 516 sec
0	Conv2D	1. Dense size 128, Dropout 0.80	Loss: 0.98, Accuracy: 8.76	Time: 516 sec
0	Conv2D	1. Dense size 128, Dropout 0.90	Loss: 0.98, Accuracy: 8.76	Time: 516 sec
0	Conv2D	1. Dense size 128, Dropout 1.00	Loss: 0.98, Accuracy: 8.76	Time: 516 sec
0	Conv2D	1. Dense size 128, Dropout 0.00	Loss: 0.94, Accuracy: 8.76	Time: 534 sec
0	Conv2D	1. Dense size 128, Dropout 0.10	Loss: 0.94, Accuracy: 8.76	Time: 534 sec
0	Conv2D	1. Dense size 128, Dropout 0.20	Loss: 0.94, Accuracy: 8.76	Time: 534 sec
0	Conv2D	1. Dense size 128, Dropout 0.30	Loss: 0.94, Accuracy: 8.76	Time: 534 sec
0	Conv2D	1. Dense size 128, Dropout 0.40	Loss: 0.94, Accuracy: 8.76	Time: 534 sec
0	Conv2D	1. Dense size 128, Dropout 0.50	Loss: 0.94, Accuracy: 8.76	Time: 534 sec
0	Conv2D	1. Dense size 128, Dropout 0.60	Loss: 0.94, Accuracy: 8.76	Time: 534 sec
0	Conv2D	1. Dense size 128, Dropout 0.70	Loss: 0.94, Accuracy: 8.76	Time: 534 sec
0	Conv2D	1. Dense size 128, Dropout 0.80	Loss: 0.94, Accuracy: 8.76	Time: 534 sec
0	Conv2D	1. Dense size 128, Dropout 0.90	Loss: 0.94, Accuracy: 8.76	Time: 534 sec
0	Conv2D	1. Dense size 128, Dropout 1.00	Loss: 0.94, Accuracy: 8.76	Time: 534 sec
0	Conv2D	1. Dense size 128, Dropout 0.00	Loss: 0.90, Accuracy: 8.76	Time: 553 sec
0	Conv2D	1. Dense size 128, Dropout 0.10	Loss: 0.90, Accuracy: 8.76	Time: 553 sec
0	Conv2D	1. Dense size 128, Dropout 0.20	Loss: 0.90, Accuracy: 8.76	Time: 553 sec
0	Conv2D	1. Dense size 128, Dropout 0.30	Loss: 0.90, Accuracy: 8.76	Time: 553 sec
0	Conv2D	1. Dense size 128, Dropout 0.40	Loss: 0.90, Accuracy: 8.76	Time: 553 sec
0	Conv2D	1. Dense size 128, Dropout 0.50	Loss: 0.90, Accuracy: 8.76	Time: 553 sec
0	Conv2D	1. Dense size 128, Dropout 0.60	Loss: 0.90, Accuracy: 8.76	Time: 553 sec
0	Conv2D	1. Dense size 128, Dropout 0.70	Loss: 0.90, Accuracy: 8.76	Time: 553 sec
0	Conv2D	1. Dense size 128, Dropout 0.80	Loss: 0.90, Accuracy: 8.76	Time: 553 sec
0	Conv2D	1. Dense size 128, Dropout 0.90	Loss: 0.90, Accuracy: 8.76	Time: 553 sec
0	Conv2D	1. Dense size 128, Dropout 1.00	Loss: 0.90, Accuracy: 8.76	Time: 553 sec
0	Conv2D	1. Dense size 128, Dropout 0.00	Loss: 0.86, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.10	Loss: 0.86, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.20	Loss: 0.86, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.30	Loss: 0.86, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.40	Loss: 0.86, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.50	Loss: 0.86, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.60	Loss: 0.86, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.70	Loss: 0.86, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.80	Loss: 0.86, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.90	Loss: 0.86, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 1.00	Loss: 0.86, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.00	Loss: 0.82, Accuracy: 8.76	Time: 612 sec
0	Conv2D	1. Dense size 128, Dropout 0.10	Loss: 0.82, Accuracy: 8.76	Time: 612 sec
0	Conv2D	1. Dense size 128, Dropout 0.20	Loss: 0.82, Accuracy: 8.76	Time: 612 sec
0	Conv2D	1. Dense size 128, Dropout 0.30	Loss: 0.82, Accuracy: 8.76	Time: 612 sec
0	Conv2D	1. Dense size 128, Dropout 0.40	Loss: 0.82, Accuracy: 8.76	Time: 612 sec
0	Conv2D	1. Dense size 128, Dropout 0.50	Loss: 0.82, Accuracy: 8.76	Time: 612 sec
0	Conv2D	1. Dense size 128, Dropout 0.60	Loss: 0.82, Accuracy: 8.76	Time: 612 sec
0	Conv2D	1. Dense size 128, Dropout 0.70	Loss: 0.82, Accuracy: 8.76	Time: 612 sec
0	Conv2D	1. Dense size 128, Dropout 0.80	Loss: 0.82, Accuracy: 8.76	Time: 612 sec
0	Conv2D	1. Dense size 128, Dropout 0.90	Loss: 0.82, Accuracy: 8.76	Time: 612 sec
0	Conv2D	1. Dense size 128, Dropout 1.00	Loss: 0.82, Accuracy: 8.76	Time: 612 sec

Gambar 8.26 Hasil No 13

```

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

Model: "sequential_42"		
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: 265,329		
Trainable params: 265,329		
Non-trainable params: 0		
None		

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: 0.7290
Epoch 4/10
- 38s - loss: 0.9141 - accuracy: 0.7411
Epoch 5/10
- 38s - loss: 0.8725 - accuracy: 0.7492
Epoch 6/10
- 38s - loss: 0.8468 - accuracy: 0.7551
Epoch 7/10
- 38s - loss: 0.8250 - accuracy: 0.7597
Epoch 8/10
- 38s - loss: 0.8032 - accuracy: 0.7651
Epoch 9/10
- 37s - loss: 0.7919 - accuracy: 0.7654
Epoch 10/10
- 38s - loss: 0.7786 - accuracy: 0.7686
<keras.callbacks.callbacks.History at 0x7fee61ec97b8>

```

Gambar 8.28 Hasil No 15

## 8.2.2.16 Nomor 16

```

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

```

Hasil:

```
(11) model.save("mathsymbols.model") menyimpan model atau mengeksport model yang telah di jalan tadi
```

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:

```
[1]: np.load('classes.npy', allow_pickle=True) # Mengimport library keras model
[2]: model2 = keras.models.load_model("mathsymbols.model") #Membuat variabel model2 untuk meload model yang telah di simpan tadi
[3]: print(model2.summary()) #Mencetak hasil model2
```

Gambar 8.30 Hasil No 17

#### 8.2.2.18 Nomor 18

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

Hasil:

Layer (Type)	Output Shape	Param #
conv2d_63 (Conv2D)	(None, 38, 38, 32)	896
max_pooling2d_63 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_64 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_64 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_42 (Flatten)	(None, 1152)	0
dense_83 (Dense)	(None, 128)	147584
dropout_32 (Dropout)	(None, 128)	0
dense_84 (Dense)	(None, 369)	47081
Total params: 285,329		
Trainable params: 285,329		
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 classes dengan data classes yang di eksport tadi
3 def predict(img_path): #Membuat fungsi predict dengan path img
4     newimg = keras.preprocessing.image.img_to_array(pil_image.
5         open(img_path)) #Membuat variabel newimg dengan membuat imimage menjadi array dan membuka data berdasarkan img path
6     newimg /= 255.0 #Membagi data yang terdapat pada variabel newimg sebanyak 255
7
8     # do the prediction
9     prediction = model2.predict(newimg.reshape(1, 32, 32, 3)) # Membuat variabel predivtion dengan isian variabel model2 menggunakan fungsi predic dengan syarat variabel newimg dengan data reshape
10
11    # figure out which output neuron had the highest score , and
12    # reverse the one-hot encoding
13    inverted = label_encoder2.inverse_transform([np.argmax(
14        prediction)]) #Membuat variabel inverted dengan 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 2 dengan tipe
label_encoder.fit(np.loadtxt('inverted/classes.npy')) #memuatkan method class
def predicting(path): #membuat fungsi untuk memprediksi gambar
    img = cv2.imread(path) #membaca gambar
    reading = keras.preprocessing.image.img_to_array(img) #mengubah gambar menjadi array
    reading = np.expand_dims(reading, axis=0) #menambahkan dimensi
    prediction = model2.predict(reading.reshape(1, 32, 32, 1)) #membuat variabel prediction
    # figure out which output neuron had the highest score, and reverse the one by
    inverted = label_encoder.inverse_transform([np.argmax(prediction)]) #membalik
    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\\_LZ89hTpq](https://youtu.be/Vq_LZ89hTpq)

## 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?	• 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 persentase dari setiap bagiannya, untuk menghasilkan hasil akhir dengan persentase yang cukup baik. Untuk ilustrasi, lihat gambar berikut:

	A	B	C	D	E	F	G	H
1	DATA			HASIL			AKURASI DATA	
2	1						%	
3	2	1					%	
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 penjelasannya yaitu For train berfungsi untuk membagi data tersebut menjadi data training. Sedangkan test in splits berfungsi untuk menguji apakah dataset tersebut sudah dibagi menjadi beberapa bagian atau masih menumpuk. Untuk ilustrasi, lihat gambar berikut:

**Gambar 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 diinginkan. 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
  
```

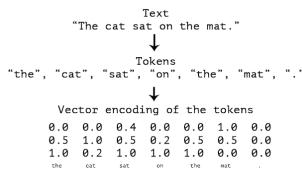
```

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.

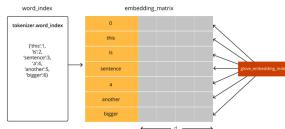
Fungsi tokenizer ini berfungsi untuk melakukan vektorisasi data kedalam bentuk token sebanyak 2000 kata. Dan selanjutnya akan melakukan fit tokenizer hanya untuk data training saja tidak dengan data testingnya. Untuk ilustrasi lihat gambar berikut:



**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.absolutetrain inputs)) dan d test inputs = d test inputs/np.amax(np.absoluted test inputs)), dilengkapi dengan ilustrasi atau gambar.

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

```

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

```

Gambar 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:

```

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

```

Gambar 8.43 Teori 8

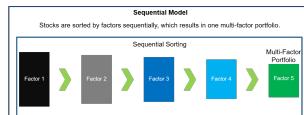
9. Jelaskan apa maksud dari fungsi di listing 7.2. Gambarkan ilustrasi Neural Network nya dari model kode tersebut.

```

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

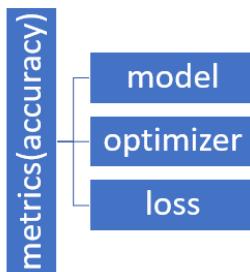
10. Jelaskan apa maksud dari fungsi di listing 7.3 dengan parameter tersebut.

```

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.

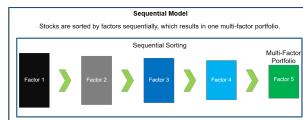
12. Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning.

DNN adalah salah satu algoritma berbasis jaringan saraf tiruan yang memiliki dari 1 lapisan saraf tersembunyi yang dapat digunakan untuk

pengambilan keputun. Perbedaannya dengan deep learning, yakni: DNN dapat menentukan dan mencerna karakteristik tertentu di suatu rangkaian data, kapabilitas lebih kompleks untuk mempelajari, mencerna, dan mengklasifikasikan data, serta dibagi ke dalam berbagai lapisan dengan fungsi yang berbeda-beda.

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

$1174080 \text{ mod}3+1 \times 1174080 \text{ mod}3+1 = 2 \times 2$ , adapun ilustrasi gambar nya sebagai berikut :



Gambar 8.46 Teori 9

### 8.3.2 Praktek Program

#### 1. Soal 1

```

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

```

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

```

In [1]:import csv
# mengimport librari Image yang berguna untuk dari PIL atau Python
# Imaging Library yang berguna untuk mengolah data berupa gambar
from PIL import Image as pil_image
# kemudian mengimport librari keras yang menggunakan method preprocessing yang
# digunakan untuk membuat neutal network
import keras.preprocessing.image

```

Gambar 8.47 Hasil Soal 1.

#### 2. Soal 2

```

1 # In[2]:load all images (as numpy arrays) and save their
2 # classes
3 imgs = []
4 #membuat variabel classes dengan variabel kosong
5 classes = []
6 #membuka file hasy-data-labels.csv yang berada di foleder
7 #HASYv2 yang di inisialisasi menjadi csvfile
8 with open('HASYv2/hasy-data-labels.csv') as csvfile:
9     #membuat variabel csvreader yang berisi method csv.reader
10    yang membaca variabel csvfile
11    csvreader = csv.reader(csvfile)
12    # membuat variabel i dengan isi 0
13    i = 0
14    # membuat looping pada variabel csvreader
15    for row in csvreader:
16        # dengan ketentuan jika i lebihkecil daripada o
17        if i > 0:
18            # dibuat variabel img dengan isi keras untuk
19            # aktivasi neural network fungsi yang membaca data yang
20            # berada dalam folder HASYv2 dengan input nilai -1.0 dan 1.0
21            img = keras.preprocessing.image.img_to_array(
22                pil_image.open("HASYv2/" + row[0]))
23            # neuron activation functions behave best when
24            # input values are between 0.0 and 1.0 (or -1.0 and 1.0),
25            # so we rescale each pixel value to be in the
26            # range 0.0 to 1.0 instead of 0-255
27            #membagi data yang ada pada fungsi img sebanyak
28            255.0
29            img /= 255.0
30            # menambah nilai baru pada imgs pada row ke 1 2
31            # dan dilanjutkan dengan variabel img
32            imgs.append((row[0], row[2], img))
33            # menambahkan nilai pada row ke 2 pada variabel
34            classes
35            classes.append(row[2])
36            # penambahan nilai satu pada variabel i
37            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
7 # 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 [6]: 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 befungsi untuk melakukan convert integers ke fungsi one hot encoding. Hasilnya adalah sebagai berikut :

```
[4]: onehot_encoder = OneHotEncoder(sparse=False)
      ... Integer_encoded = Integer_encoded.reshape((-1, Integer_encoded.shape[1], 1))
onehot_encoder.fit(Integer_encoded)
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 classes
11 num_classes = len(label_encoder.classes_)
12 # mencetak hasil dari nomer Class berupa persen
13 print("Number of classes: %d" % num_classes)
```

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

```
[4]: train_output_int = label_encoder.transform(train_output)
onehot_encoder.transform(train_output_int.reshape((-1,train_output_int.shape[1], 1)))
test_output_int = onehot_encoder.transform(test_output_int.reshape((-1,test_output_int.shape[1], 1)))
... num_classes = (label_encoder.classes_).shape[0] / X.num_classes
num_classes: 399
```

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

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

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

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

```

    .....
    model.add(conv2d_1, kernel_size=(3, 3), activation='relu')
    input_shape = shape('train_imput')[1:-1]
    model.add(conv2d_2, kernel_size=(3, 3), activation='relu')
    model.add(conv2d_3, (4, 4), activation='relu')
    model.add(maxPooling2d_1, pool_size=(2, 2))
    model.add(conv2d_4, activation='relu')
    model.add(conv2d_5, activation='relu')
    model.add(maxPooling2d_2, pool_size=(2, 2))
    model.add(conv2d_6, activation='relu')
    model.add(dense_1, activation='softmax')
    model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=[accuracy])
    print(model.summary())
layer (type)          Output Shape         Params #
=====
conv2d_0 (Conv2D)      (None, 10, 10, 32)     896
max_pooling2d_0 (MaxPooling2d) (None, 5, 5, 32)   0
conv2d_1 (Conv2D)      (None, 5, 5, 32)        9248
max_pooling2d_1 (MaxPooling2d) (None, 2, 2, 32)   0
flatten_0 (Flatten)   (None, 1152)            0
dense_1 (Dense)       (None, 128)             147584
dropout_1 (Dropout)   (None, 128)             0
dense_2 (Dense)       (None, 369)             47601

Total params: 285,329
Trainable params: 285,329
Non-trainable params: 0

```

**Gambar 8.55** Hasil Soal 10.

11. Soal 11

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

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

```
In [12]: import keras.callbacks  
.... tensorboard = keras.callbacks.TensorBoard(log_dir='./Logs/enist-style')
```

**Gambar 8.56** Hasil Soal 11.

12. Soal 12

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

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

```

Epoch 1/10
17/10 - loss: 1.3816 - accuracy: 0.6081 - val_loss: 1.0032 - val_accuracy: 0.7932
18/10 - loss: 0.8862 - accuracy: 0.7481 - val_loss: 0.8862 - val_accuracy: 0.7922
19/10 - loss: 0.8862 - accuracy: 0.7477 - val_loss: 0.8862 - val_accuracy: 0.7922
20/10 - loss: 0.8862 - accuracy: 0.7473 - val_loss: 0.8862 - val_accuracy: 0.7922
21/10 - loss: 0.8828 - accuracy: 0.7612 - val_loss: 0.8828 - val_accuracy: 0.7929
22/10 - loss: 0.8828 - accuracy: 0.7608 - val_loss: 0.8828 - val_accuracy: 0.7929
23/10 - loss: 0.7751 - accuracy: 0.7718 - val_loss: 0.8479 - val_accuracy: 0.7918
24/10 - loss: 0.7751 - accuracy: 0.7714 - val_loss: 0.8479 - val_accuracy: 0.7918
25/10 - loss: 0.7716 - accuracy: 0.7804 - val_loss: 0.9002 - val_accuracy: 0.7642
26/10 - loss: 0.7716 - accuracy: 0.7800 - val_loss: 0.9002 - val_accuracy: 0.7642
27/10 - loss: 0.7704 - accuracy: 0.7815 - val_loss: 0.9360 - val_accuracy: 0.7654
28/10 - loss: 0.7704 - accuracy: 0.7811 - val_loss: 0.9360 - val_accuracy: 0.7654
29/10 - loss: 0.6295 - accuracy: 0.7924 - val_loss: 0.8462 - val_accuracy: 0.7939
30/10 - loss: 0.6295 - accuracy: 0.7920 - val_loss: 0.8462 - val_accuracy: 0.7939
31/10 - loss: 0.6552 - accuracy: 0.7985 - val_loss: 0.8531 - val_accuracy: 0.7981
32/10 - loss: 0.6552 - accuracy: 0.7981 - val_loss: 0.8531 - val_accuracy: 0.7981
33/10 - loss: 0.6512 - accuracy: 0.8023 - val_loss: 0.8792 - val_accuracy: 0.7611
34/10 - loss: 0.6512 - accuracy: 0.8023 - val_loss: 0.8792 - val_accuracy: 0.7611
test_accuracy: 0.76541598898345

```

**Gambar 8.57** Hasil Soal 12.

### 13. Soal 13

```

1 # In[13]:try various model configurations and parameters to
2 # find the best
3 # mengimport librari time
4 import time
5 #membuat variabel result dengan array kosong
6 results = []
7 # melakukan looping dengan ketentuan konvolusi 2 dimensi 1 2
8 for conv2d_count in [1, 2]:
9     # menentukan ukuran besaran fixcel dari data atau konvert
10    # 1 fixcel mnjadi data yang berada pada codigan dibawah.
11    for dense_size in [128, 256, 512, 1024, 2048]:
12        # membuat looping untuk memangkas masing-masing data
13        # dengan ketentuan 0 persen 25 persen 50 persen dan 75
14        # persen .
15        for dropout in [0.0, 0.25, 0.50, 0.75]:
16            # membuat variabel model Sequential
17            model = Sequential()
18            #membuat looping untuk variabel i dengan jarak
19            # dari hasil konvolusi .
20            for i in range(conv2d_count):
21                # syarat jika i samadengan bobotnya 0
22                if i == 0:
23                    # menambahkan method add pada variabel
24                    model dengan konvolusi 2 dimensi 32 bit didalamnya dan
25                    membuat kernel dengan ukuran 3 x 3 dan rumus aktifasi relu
26                    dan data shape yang di hitung dari data train.
27                    model.add(Conv2D(32, kernel_size=(3, 3),
28 activation='relu', input_shape=np.shape(train_input[0])))
29                    # jika tidak
30                else:
31                    # menambahkan method add pada variabel
32                    model dengan konvolusi 2 dimensi 32 bit dengan ukuran
33                    kernel 3 x3 dan fungsi aktivasi relu
34                    model.add(Conv2D(32, kernel_size=(3, 3),
35 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 pemanjangan data
29         dengan ukuran dense di tentukan dengan rumus fungsi tanh.
30         model.add(Dense(dense_size , activation='tanh'))
31         # membuat ketentuan jika pemangkasan lebih besar
32         dari 0 persen
33         if dropout > 0.0:
34             # menambahkan method dropout pada model
35             dengan nilai dari dropout
36             model.add(Dropout(dropout))
37             # menambahkan method dense dengan fungsi num
38             classss dan rumus softmax
39             model.add(Dense(num_classes , activation='softmax',
40             ))
41             # mengompilasi variabel model dengan hasil loss
42             optimasi dan akurasi matrix
43             model.compile(loss='categorical_crossentropy' ,
44             optimizer='adam' , metrics=['accuracy'])
45             # melakukan log pada dir
46             log_dir = './logs/conv2d_%d-dense_%d-dropout_%f'
47             , % (conv2d_count , dense_size , dropout)
48             # membuat variabel tensorboard dengan isian dari
49             librari keras dan nilai dari lig dir
50             tensorboard = keras.callbacks.TensorBoard(log_dir
51             =log_dir)
52             # membuat variabel start dengan isian dari
53             librari time menggunakan method time
54
55             start = time.time()
56             # menambahkan method fit pada model dengan data
57             dari train input train output nilai batch nilai epoch
58             verbose nilai 20 persen validation split dan callback
59             dengan nilai tnsorboard.
60             model.fit(train_input , train_output , batch_size
61             =32, epochs=10,
62             verbose=0, validation_split=0.2,
63             callbacks=[tensorboard])
64             # membuat variabel score dengan nilai evaluasi
65             dari model menggunakan data tes input dan tes output
66             score = model.evaluate(test_input , test_output ,
67             verbose=2)
68             # membuat variabel end
69             end = time.time()
70             # membuat variabel elapsed
71             elapsed = end - start
72             # mencetak hasil perhitungan
73             print("Conv2D count: %d, Dense size: %d, Dropout:
74             %.2f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" %
75             (conv2d_count , dense_size , dropout , score[0] , score[1] ,
76             elapsed))
77             results.append((conv2d_count , dense_size , dropout
78             , score[0] , score[1] , elapsed))

```

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

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

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 tigapuluhan dua
    bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
    relu dang menggunakan data train_input
5 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
    input_shape=np.shape(train_input[0])))
6 # variabel model di tambahkan dengan lib MaxPooling2D dengan
    ketentuan ukuran 2 x 2 pixcel
7 model.add(MaxPooling2D(pool_size=(2, 2)))
8 # variabel model di tambahkan dengan librari Conv2D 32 bit
    dengan kernel 3 x 3
9 model.add(Conv2D(32, (3, 3), activation='relu'))
10 # variabel model di tambahkan dengan lib MaxPooling2D dengan
    ketentuan ukuran 2 x 2 pixcel
11 model.add(MaxPooling2D(pool_size=(2, 2)))
12 # variabel model di tambahkan librari Flatten
13 model.add(Flatten())
14 # variabel model di tambahkan librari Dense dengan fungsi
    tanh
15 model.add(Dense(128, activation='tanh'))
16 # variabel model di tambahkan librari dropout untuk memangkas
    data tree sebesar 50 persen
17 model.add(Dropout(0.5))
18 # variabel model di tambahkan librari Dense dengan data dari
    num_classes dan fungsi softmax
19 model.add(Dense(num_classes, activation='softmax'))
20 # mengkompile data model untuk mendapatkan data loss akurasi
    dan optimasi
21 model.compile(loss='categorical_crossentropy', optimizer='adam',
    metrics=['accuracy'])
22 # mencetak variabel model kemudian memunculkan kesimpulan
    berupa data total parameter, trainable paremeter dan bukan
    trainable parameter
23 print(model.summary())

```

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

```

    ...
    .model.add(maxpooling2d(pool_size=(2, 2)))
    .model.add(conv2d(32, (3, 3), activation='relu'))
    .model.add(maxpooling2d(pool_size=(2, 2), activation='relu'))
    .model.add(fatten())
    .model.add(dense(128, activation='softmax'))
    .model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
Model: "sequential_5"
Layer (type)                 Output Shape              Param #
conv2d_6 (Conv2D)            (None, 38, 38, 32)       896
max_pooling2d_6 (MaxPooling2D) (None, 19, 19, 32)       0
conv2d_7 (Conv2D)            (None, 13, 13, 32)       9248
max_pooling2d_7 (MaxPooling2D) (None, 6, 6, 32)       0
flatten_5 (Flatten)          (None, 1152)             0
dense_9 (Dense)              (None, 128)              147584
dropout_8 (Dropout)          (None, 128)              0
dense_10 (Dense)             (None, 369)              47681
Total params: 265,329
Trainable params: 265,329
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
      all data we have available to us
2 # melakukan join numpy menggunakan data train_input
      test_input
3 model.fit(np.concatenate((train_input, test_input)),
4           # kelanjutan data yang di gunakan pada join
      train_output test_output
5           np.concatenate((train_output, test_output)),
6           #menggunakan ukuran 32 bit dan epoch 10
      batch_size=32, epochs=10, verbose=2)
7

```

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

```

In [15]: model.fit(np.concatenate((train_input, test_input)),
      np.concatenate((train_output, test_output)),
      ...
      batch_size=32, epochs=10, verbose=2)
Epoch 0/10
 100%|██████████| 0/10 [00:00<00:00]
Epoch 1/10
 100%|██████████| 1/10 [00:00<00:00] - loss: 1.7799 - accuracy: 0.5871
Epoch 2/10
 100%|██████████| 2/10 [00:00<00:00] - loss: 1.8744 - accuracy: 0.7074
Epoch 3/10
 100%|██████████| 3/10 [00:00<00:00] - loss: 0.9564 - accuracy: 0.7320
Epoch 4/10
 100%|██████████| 4/10 [00:00<00:00] - loss: 0.8979 - accuracy: 0.7458
Epoch 5/10
 100%|██████████| 5/10 [00:00<00:00] - loss: 0.8573 - accuracy: 0.7527
Epoch 6/10
 100%|██████████| 6/10 [00:00<00:00] - loss: 0.8297 - accuracy: 0.7586
Epoch 7/10
 100%|██████████| 7/10 [00:00<00:00] - loss: 0.8079 - accuracy: 0.7632
Epoch 8/10
 100%|██████████| 8/10 [00:00<00:00] - loss: 0.7917 - accuracy: 0.7663
Epoch 9/10
 100%|██████████| 9/10 [00:00<00:00] - loss: 0.7765 - accuracy: 0.7707
Epoch 10/10
 100%|██████████| 10/10 [00:00<00:00] - loss: 0.7637 - accuracy: 0.7718
Out[16]: <keras.callbacks.callbacks.History at 0x1637ebc3d48>

```

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

```

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

```

**Gambar 8.61** Hasil Soal 16.

### 17. Soal 17

```

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

```

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

```

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

```

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

```

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

Layer (Type)	Output Shape	Param #
conv2d_68 (Conv2D)	(None, 16, 16, 32)	896
max_pooling2d_68 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_69 (Conv2D)	(None, 15, 15, 32)	9248
max_pooling2d_69 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_45 (Flatten)	(None, 1152)	0
dense_89 (Dense)	(None, 128)	147584
dropout_34 (Dropout)	(None, 128)	0
dense_90 (Dense)	(None, 369)	47601
<hr/>		
Total params: 205,329		
Trainable params: 205,329		
Non-trainable params: 0		
<hr/>		
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
3 # label encoder.
4 label_encoder2 = LabelEncoder()
5 # menambahkan method classess dengan data classess yang di
6 # eksport tadi
7 label_encoder2.classes_ = np.load('classes.npy')
8 # membuat fumgsi predict dengan path img
9 def predict(img_path):
10     # membuat variabel newimg dengan membuay immage menjadi
11     # array dan membuka data berdasarkan img path
12     newimg = keras.preprocessing.image.img_to_array(pil_image
13         .open(img_path))
14     # membagi data yang terdapat pada variabel newimg
15     # sebanyak 255
16     newimg /= 255.0
17
18     # do the prediction
19     # membuat variabel predivtion dengan isian variabel
20     # model2 menggunakan fungsi predic dengan syarat variabel
21     # newimg dengan data reshape
22     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
23
24     # figure out which output neuron had the highest score,
25     # and reverse the one-hot encoding
26     # membuat variabel inverted denagan label encoder2 dan
27     # menggunakan argmax untuk mencari skor luaran tertinggi
28     inverted = label_encoder2.inverse_transform([np.argmax(
29         prediction)])
30     # mencetak prediksi gambar dan confidence dari gambar.
31     print("Prediction: %s, confidence: %.2f" % (inverted[0],
32         np.max(prediction)))

```

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

Layer (Type)	Output Shape	Param #
conv2d_68 (Conv2D)	(None, 16, 16, 32)	896
max_pooling2d_68 (MaxPooling)	(None, 15, 15, 32)	0
conv2d_69 (Conv2D)	(None, 15, 15, 32)	9248
max_pooling2d_69 (MaxPooling)	(None, 6, 6, 32)	0
flatten_45 (Flatten)	(None, 1152)	0
dense_89 (Dense)	(None, 128)	147584
dropout_34 (Dropout)	(None, 128)	0
dense_90 (Dense)	(None, 369)	47601
<hr/>		
Total params: 205,329		
Trainable params: 205,329		
Non-trainable params: 0		
<hr/>		
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 befungsi untuk menunjukkan hasil akhir prediski. Hasilnya adalah sebagai berikut :

```

# grab an image (we'll just use a random training image for demonstration purposes)
predict("HASYv2/hasy-data/v2-00010.png")
prediction: a_0, confidence: 0.07
predict("HASYv2/hasy-data/v2-00500.png")
prediction: b_0, confidence: 0.58
predict("HASYv2/hasy-data/v2-00700.png")
prediction: valpha, confidence: 0.88

```

Gambar 8.65 Hasil Soal 20.

## 8.3.3 Penanganan Error

## 1. KeyboardInterrupt

```

file = keras.preprocessing.image.ImageDataGenerator()
file = file.flow_from_directory("HASYv2/*/*", target_size=(150, 150))
fp = open('HASYv2/HASYv2/hasy-data/v2/images.txt', 'r')
for line in fp:
    line = line.strip()
    if line[-1] == '\n':
        line = line[:-1]
    print(file.find_generator(line))

```

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



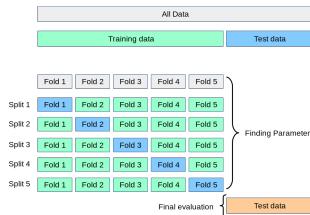
Gambar 8.68    Teori 1

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

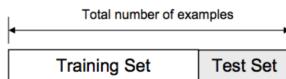
Kfold diatas bertujuan untuk membagi data kedalam 5 bagian yang nantinya akan menampung komentar dari youtube. data yang dibagi ini nanti akan dihitung dan akan menghasilkan persentase yang bisa dibilang cukup baik



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)*

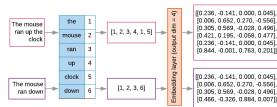
Berfungsi untuk melakukan proses vektorisasi data menjadi token sebanyak 2000 kata



Gambar 8.72 Teori 5

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

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)          # 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 8.74 Teori 7

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

### *kode program*

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

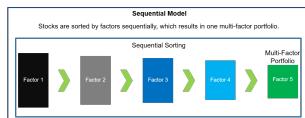
```
>>> V_train
array([[0., 0., 0.],
       [0., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.], dtype=int64)
>>> V_train.flatten()
array([1, 0, 0, ..., 1, 0, 0], dtype=int64)
>>> V_train
array([[1, 0, 0.],
       [0, 1, 0.],
       [1, 0, 0.],
       [0,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

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.



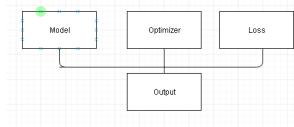
Gambar 8.76 Teori 9

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

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

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

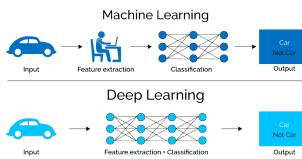
adalah adam, selain adam, ada juga SGD,RMSprop, dan lain lain dan metrics accuracy adalah value yang akan kita cari nilainya



**Gambar 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



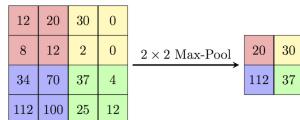
#### 8.4.1.12 Apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning

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

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

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

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



Gambar 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(
9         pil_image.open("E:/HASYv2/" + row[0])) #Dibuat variabel img
10      dengan isi keras untuk aktivasi neural network fungsi yang
11      membaca data yang berada dalam folder HASYv2 dengan input
12      nilai -1.0 dan 1.0
13      # neuron activation functions behave best when input
14      values are between 0.0 and 1.0 (or -1.0 and 1.0),
15      # so we rescale each pixel value to be in the range
16      0.0 to 1.0 instead of 0–255
17      img /= 255.0 #Membagi data yang ada pada fungsi img
18      sebanyak 255.0
19      imgs.append((row[0], row[2], img)) #Menambah nilai
20      baru pada imgs pada row ke 1 2 dan dilanjutkan dengan
21      variabel img
22      classes.append(row[2]) #Menambahkan nilai pada row ke
23      2 pada variabel classes
24      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 LabelEncoder 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 classes
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 tigapuluhan dua bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan relu dengan 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 pixcel
7 model.add(Flatten()) #Variabel model di tambahkan library Flatten
8 model.add(Dense(1024, activation='tanh')) #Variabel model di tambahkan library Dense dengan fungsi tanh
9 model.add(Dropout(0.5)) #Variabel model di tambahkan library dropout untuk memangkas data tree sebesar 50 persen
10 model.add(Dense(num_classes, activation='softmax')) #Variabel model di tambahkan library Dense dengan data dari num_classes dan fungsi softmax

```

```
11 model.compile(loss='categorical_crossentropy', optimizer='adam',  
12                 metrics=['accuracy']) #Mengkompile data model untuk  
13 mendapatkan data loss akurasi dan optimasi  
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  
2         fungsi fit untuk mengetahui perhitungan dari train_input  
3         train_output  
4             batch_size=32, #Dengan batch size 32 bit  
5             epochs=10,  
6             verbose=2,  
7             validation_split=0.2,  
8             callbacks=[tensorboard])  
9  
10 score = model.evaluate(test_input, test_output, verbose=2)  
11 print('Test loss:', score[0])  
12 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
4     konvolusi 2 dimensi 1 2
5     for dense_size in [128, 256, 512, 1024, 2048]: #Menentukan
6         ukuran besaran fixcel dari data atau konvert 1 fixcel mnjadi
7         data yang berada pada codigan dibawah.
8         for dropout in [0.0, 0.25, 0.50, 0.75]: #Membuat looping
9             untuk memangkas masing-masing data dengan ketentuan 0 persen
10            25 persen 50 persen dan 75 persen.
11            model = Sequential() #Membuat variabel model
12            Sequential
13            for i in range(conv2d_count): #Membuat looping untuk
14                variabel i dengan jarak dari hasil konvolusi.
15                if i == 0: #Syarat jika i samadengan bobotnya 0
16                    model.add(Conv2D(32, kernel_size=(3, 3),
17                                     activation='relu', input_shape=np.shape(train_input[0]))) #
18 Menambahkan method add pada variabel model dengan konvolusi 2
19 dimensi 32 bit didalamnya dan membuat kernel dengan ukuran 3
20 x 3 dan rumus aktifasi relu dan data shape yang di hitung
21 dari data train.
```

```

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

```

#### 8.4.2.14 Nomor 14

```

1 model = Sequential() #Membuat variabel model dengan isian library
2 Sequential

```

```

2 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
    input_shape=np.shape(train_input[0]))) #Variabel model di
    tambahkan library Conv2D tigapuluhan dua bit dengan ukuran
    kernel 3 x 3 dan fungsi penghitungan relu yang menggunakan
    data train_input
3 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
    tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x
    2 pixel
4 model.add(Conv2D(32, (3, 3), activation='relu')) #Variabel model
    di tambahkan dengan library Conv2D 32bit dengan kernel 3 x 3
5 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di
    tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x
    2 pixcel
6 model.add(Flatten()) #Variabel model di tambahkan library Flatten
7 model.add(Dense(128, activation='tanh')) #Variabel model di
    tambahkan library Dense dengan fungsi tanh
8 model.add(Dropout(0.5)) #Variabel model di tambahkan library
    dropout untuk memangkas data tree sebesar 50 persen
9 model.add(Dense(num_classes, activation='softmax')) #Variabel
    model di tambahkan library Dense dengan data dari num_classes
    dan fungsi softmax
10 model.compile(loss='categorical_crossentropy', optimizer='adam',
    metrics=['accuracy']) #Mengkompile data model untuk
    mendapatkan data loss akurasi dan optimasi
11 print(model.summary()) #Mencetak variabel model kemudian
    memunculkan kesimpulan berupa data total parameter, trainable
    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 mengeksport
    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 #Mengimpport library keras model
2 model2 = keras.models.load_model("mathsymbols.model") #Membuat
   variabel model2 untuk meload model yang telah di simpan tadi
3 print(model2.summary()) #Mencetak hasil model2

```

#### 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 classes dengan data classess yang di eksport tadi
3 def predict(img_path): #Membuat fumgsi predict dengan path img
4   newimg = keras.preprocessing.image.img_to_array(pil_image.
   open(img_path)) #Membuat variabel newimg dengan membuat
   immage menjadi array dan membuka data berdasarkan img path
5   newimg /= 255.0 #Membagi data yang terdapat pada variabel
   newimg sebanyak 255
6
7 # do the prediction
8 prediction = model2.predict(newimg.reshape(1, 32, 32, 3)) #
   Membuat variabel 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
```

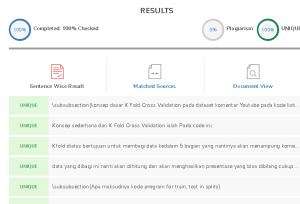
**Gambar 8.79** NameError

#### 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 kfolds yang berisi fungsi StratifiedKFold yang memiliki parameter n\_splits=5 yang berarti melakukan

pengulangan terhadap data masing-masing lima kali dengan atribut class sebagai acuan pengolahan datanya kemudian akan di hasilkan akurasi dari pengulangan data tersebut sebesar sekian persen tergantung datanya.

Gambar 1 – Skema 10 fold CV

### Gambar 8.81 Teori 2

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=525)
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.

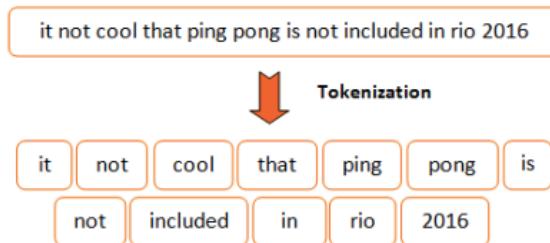
Kode program `train_content` tersebut adalah membaca isi kolom pada field yang bernama `CONTENT` sebagai data training sedangkan kode program `test_content` membaca isi kolom pada field yang bernama `CONTENT` sebagai data testing.

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\_content) 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 tf idf begitu juga dengan d\_test\_inputs untuk data test.

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

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

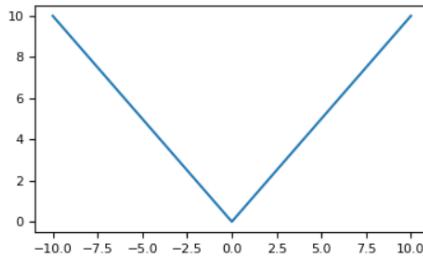
**Gambar 8.85** Teori 6

7. Jelaskan apa maksud dari fungsi d\_train\_inputs = d\_train\_inputs/np.abs(d\_train\_inputs).max() dan d\_test\_inputs = d\_test\_inputs/np.abs(d\_test\_inputs).max(), 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

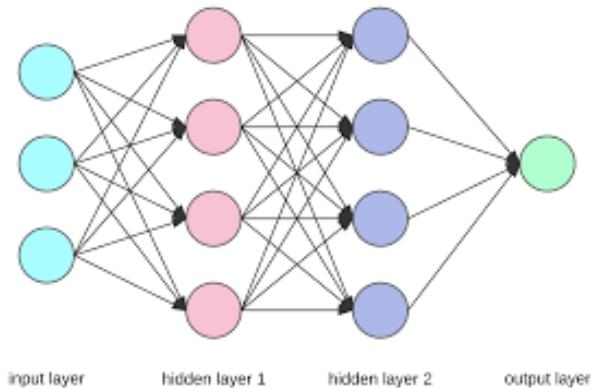
8. Jelaskan apa maksud fungsi dari `d.train_outputs = np_utils.to_categorical(d['CLASS'])` dan `d_test_outputs = np_utils.to_categorical(d['CLASS'].iloc[test_idx])` dalam kode program, dilengkapi dengan ilustrasi atau gambar.  
 fungsi untuk melakukan one-hot encoding merubah nilai vektor dengan bentuk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

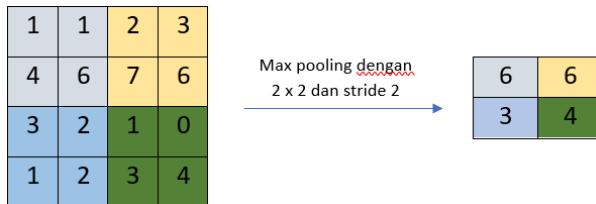
dan dilakukan overfitting atau pemotongan bobot sebesar 0.5 atau 50 persen. Lalu pada layer output terdapat 2 neuron output. Kemudian outputan tersebut diaktivasi menggunakan fungsi softmax.



**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 matrics = akurasi merupakan akurasi dari nilai matriknya.
11. Jelaskan apa itu Deep Learning  
Merupakan metode pada machine learning yang menggunakan artificial neural networks. algoritma permodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam.
12. Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning  
Deep Neural Network merupakan algoritma jaringan syaraf yang akan melakukan pembobotan terhadap data yang sudah ada sebagai acuan untuk data inputan selanjutnya. kemudian terdiri atas beberapa lapisan/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 mod3+1) x (NPM mod3+1) yang terdapat max pooling

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



Gambar 8.89 Teori 13

### 8.5.2 Praktek

- 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

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

```

Name	Type	Size	Value
classes	list	168233	['A', 'A', ...]
i	int	1	168234
img	float32 (32, 32, 3)		[[[1. 1. 1.]  [1. 1. 1.]  [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', '\guillemeotleft', '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 #Membuat variabel train_input dengan np method asarray yang
   mana membuat array dengan fungsi list yang didalamnya
   diterapkan fungsi map untuk mengembalikan interator
7 #dan menggunakan lamba untuk mengecilkan fungsi dari objek
   yang berada pada row 2 dari data train
8 test_input = np.asarray(list(map(lambda row: row[2], test)))
9 #Membuat variabel test_input dengan isi np method asarray
   yang mana membuat array dengan fungsi list yang didalamnya
   diterapkan fungsi map untuk mengembalikan interator
10 #dan menggunakan lamba untuk mengecilkan fungsi dari objek
    yang berada pada row 2 dari data test
11 train_output = np.asarray(list(map(lambda row: row[1], train)))
12 #Membuat variabel train_output dengan np method asarray yang
   mana membuat array dengan fungsi list yang didalamnya
   diterapkan fungsi map untuk mengembalikan interator
13 #dan menggunakan lamba untuk mengecilkan fungsi dari objek
   yang berada pada row 1 dari data train
14 test_output = np.asarray(list(map(lambda row: row[1], test)))
15 #Membuat variabel test_output dengan np method asarray yang
   mana membuat array dengan fungsi list yang didalamnya
   diterapkan fungsi map untuk mengembalikan interator
16 #dan menggunakan lamba untuk mengecilkan fungsi dari objek
   yang berada pada row 1 dari data train

```

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

Gambar 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
4     yang digunakan untuk mengonversi jenis data teks kategori
5     menjadi data numerik
6 from sklearn.preprocessing import OneHotEncoder
7 #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
8     mengonversi 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
4     OneHotEncoder parameter sparse=false
5 integer_encoded = integer_encoded.reshape(len(integer_encoded),
6                                         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),
5                                activation='relu',
6                                input_shape=np.shape(train_input[0])))
#Variabel model ditambahkan fungsi Conv2d dengan paramater 32
#dengan filter dengan karnel berukuran 3x3
7 #dengan algoritam activation relu menggunakan data train_input
#mulai dari baris no1.
8 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
#Variabel model ditambahkan fungsi MaxPooling2D dengan
#ketentuan ukuran 2x2
10 model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu'))
#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)))
#Variabel model ditambahkan fungsi MaxPooling2D dengan
#ketentuan ukuran 2x2
14 model.add(tf.keras.layers.Flatten())
#Variabel model di tambahkan fungsi Flatten
16 model.add(tf.keras.layers.Dense(1024, activation='tanh'))
#Variabel model ditambahakan fungsi dense dengan 1024 neuron,
#dan menggunakan algoritma tanh untuk activation
18 model.add(tf.keras.layers.Dropout(0.5))
#Variabel model di tambahkan fungsi Dropout sebesar 50% untuk
#mencegah terjadinya overfitting
20 model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
#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
25 optimasi
26 print(model.summary())
27 #Menampilkan model yang telah dibuat.

```

```

Model: "sequential_1"
-----  

Layer (type)          Output Shape       Param #
conv2d_1 (Conv2D)     (None, 30, 30, 32)    896
-----  

max_pooling2d_1 (MaxPooling2D) (None, 15, 15, 32) 0
-----  

conv2d_2 (Conv2D)     (None, 13, 13, 32)    9248
-----  

max_pooling2d_2 (MaxPooling2D) (None, 6, 6, 32) 0
-----  

flatten_1 (Flatten)   (None, 1152)        0
-----  

dense_1 (Dense)       (None, 1024)        1180672
-----  

dropout_1 (Dropout)   (None, 1024)        0
-----  

dense_2 (Dense)       (None, 369)         378225
-----  

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

None

```

**Gambar 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 TenserBoard
    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 TenserBoard 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          #epoch=10 yang berarti terjadi perulangan
8          sebanyak 10 kali
9          verbose=2,
10         #untuk menghasilkan informasi logging dari data
11         yang ditentukan dengan nilai 2
12         validation_split=0.2,
13         #melakukan pemecahan nilai sebesar 0.2 / 20% dari
14         perhitungan validasi
15         callbacks=[tensorboard])
16         #mengeksekusi tensorboard dimana digunakan untuk
17         visualisasikan parameter training, metrik, hiperparameter
18         pada nilai/data yang diproses
19
20 score = model.evaluate(test_input, test_output, verbose=2)
21 #Membuat variabel score dengan isi fungsi evluate dari model
22         dengan paramater test_input, test_output dan verbose=2
23 #untuk memprediksi output dan input
24 print('Test loss:', score[0])
25 #Menampilkan score optimasi dengan ketentuan nilai parameter
26         0
27 print('Test accuracy:', score[1])
28 #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: 0.6284 - accuracy: 0.8029 - val_loss: 0.8908 - val_accuracy: 0.7584
33647/33647 - 6s - loss: 0.8712 - accuracy: 0.7624
Test loss: 0.8711880891701912
Test accuracy: 0.76238596

```

Gambar 8.101 Hasil Praktek 12

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

```

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

```



```

42     log_dir = '.\logs\conv2d_%d-dense_%d-dropout_%
43         %.2f' % (conv2d_count, dense_size, dropout)
44         #Melakukan log pada dir
45         tensorboard = keras.callbacks.TensorBoard(log_dir
46             =log_dir)
47             #Mengisi variabel tensorboard dengan isian dari
48             library keras dan nilai dari log dir
49
50             start = time.time()
51             #Membuat variabel start dengan isi fungsi time
52             dari library time
53             model.fit(train_input, train_output, batch_size
54                 =32, epochs=10,
55                     verbose=0, validation_split=0.2,
56                     callbacks=[tensorboard])
57             #Melakukan fitting pada model dengan parameter
58             test_input, test_output, batch_size, epochs, verbose,
59             validation_split dan callbacks
60             score = model.evaluate(test_input, test_output,
61             verbose=2)
62             #Membuat variabel score dengan nilai evaluasi
63             dari model menggunakan data tes input dan tes output
64             dengan verbose adalah 2
65             end = time.time()
66             #Membuat variabel end dengan isi fungsi time dari
67             library time
68             elapsed = end - start
69             #Membuat variabel elapse yang diisi dengan nilai
70             hasil waktu end dikurangi start
71             print("Conv2D count: %d, Dense size: %d, Dropout:
72                 %.2f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" %
73                 (conv2d_count, dense_size, dropout, score[0], score[1],
74                 elapsed))
75             results.append((conv2d_count, dense_size, dropout
76                 , score[0], score[1], elapsed))
77             #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: 0.8037 - accuracy: 0.7775
Conv2D count: 1, Dense size: 128, Dropout: 0.50 - Loss: 0.80, Accuracy: 0.78, Time: 550 sec
WARNING:tensorflow:Large dropout rate: 0.75 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate
instead of keep_prob. Please ensure that this is intended.
WARNING:tensorflow:Large dropout rate: 0.75 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate
instead of keep_prob. Please ensure that this is intended.
WARNING:tensorflow:Large dropout rate: 0.75 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate
instead of keep_prob. Please ensure that this is intended.
33647/33647 7s - Loss: 0.8012 - accuracy: 0.7706
Conv2D count: 1, Dense size: 128, Dropout: 0.75 - Loss: 0.80, Accuracy: 0.77, Time: 564 sec
33647/33647 8s - 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 9s - Loss: 1.0000 - accuracy: 0.7503
Conv2D count: 1, Dense size: 256, Dropout: 0.25 - Loss: 1.11, Accuracy: 0.76, Time: 738 sec
33647/33647 10s - 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])))
#Variabel model ditambahkan fungsi Conv2d dengan paramater 32
    filter dengan karnel berukuran 3x3
6 #dengan algoritam activation relu menggunakan data train_input
    mulai dari baris nol.
7 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
#Variabel model ditambahkan fungsi MaxPooling2D dengan
    ketentuan ukuran 2x2
8 model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
9     ))
#Variabel model ditambahkan fungsi Conv2D dengan 32 filter
    dengan konvolusi berukuran 3x3, menggunakan algoritam
    activation relu
10 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
#Variabel model ditambahkan fungsi MaxPooling2D dengan
    ketentuan ukuran 2x2
11 model.add(tf.keras.layers.Flatten())
#Variabel model di tambahkan fungsi Flatten
12 model.add(tf.keras.layers.Dense(128, activation='tanh'))
#Variabel model ditambahakan fungsi dense dengan 128 neuron ,
    dan menggunakan algoritma tanh untuk activation
13 model.add(tf.keras.layers.Dropout(0.5))
#Variabel model di tambahkan fungsi Dropout sebesar 50% untuk
    mencegah terjadinya overfitting
14 model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
#Variabel model ditambahkan fungsi Dense dengan parameter
    variabel num_classes menggunakan activation softmax
15 model.compile(loss='categorical_crossentropy', optimizer='adam',
    metrics=['accuracy'])
#Variabel model di compile dengan parameter loss , matrik , dan
    optimasi
16 print(model.summary())
#Menampilkan ringkasan model yang telah dibuat.
```

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

Gambar 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
2           all data we have available to us
3 model.fit(np.concatenate((train_input, test_input)),
4 #Melakukan fitting pada model melakukan join numpy
5           menggunakan data train_input test_input
6           np.concatenate((train_output, test_output)),
7           #kemudian join numpy menggunakan data train_output
8           test_output
9           batch_size=32, epochs=10, verbose=2)
10          #menggunaakan batch ukuran 32, epochs 10 dan verbose
11          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 [21]: runcell(['/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(['/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
<hr/>		
Total params: 205,329		
Trainable params: 205,329		
Non-trainable params: 0		
<hr/>		
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
     Label_Encoder
4 label_encoder2.classes_ = np.load('classes.npy')
5 #Menggunakan method classes dengan data classes.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 variable 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(
20         prediction) ]) # argmax finds highest-scoring output
#Membuat variabel inverted dengan label encoder2 dan
21     menggunakan argmax untuk mencari skor luaran tertinggi
22     print("Prediction: %s, confidence: %.2f" % (inverted[0] ,
23         np.max(prediction)))
#Menampilkan prediksi gambar dan confidence dari gambar.

```

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

**Gambar 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
2     image for demonstration purposes)
predict("D:/New folder/KB3C/src/1174084/7/HASYv2/hasy-data/v2
3     -00010.png")
#Melakukan prediksi dari pelatihan dari gambar v2-00010.png
4 predict("D:/New folder/KB3C/src/1174084/7/HASYv2/hasy-data/v2
5     -00500.png")
#Melakukan prediksi dari pelatihan dari gambar v2-00500.png
6 predict("D:/New folder/KB3C/src/1174084/7/HASYv2/hasy-data/v2
7     -00700.png")
#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>

## 8.6 1174070 - Arrizal Furqona Gifary

### 8.6.1 Soal Teori

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

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

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

Gambar 8.111 Teori 1

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

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

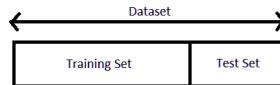
Konsep sederhana dari K Fold Cross Validation ialah Pada code tersebut terdapat kfolds yang bertujuan untuk melakukan split data menjadi 5 bagian dari dataset komentar Youtube tersebut. Sehingga dari setiap data yang sudah dibagi tersebut akan menghasilkan persentase dari setiap bagiannya, untuk menghasilkan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

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

Gambar 8.112 Teori 2

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

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



Gambar 8.113 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
  
```

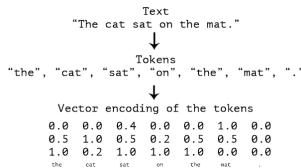
```

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

Gambar 8.114 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.

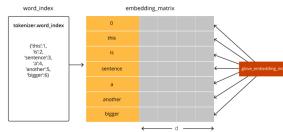
Fungsi tokenizer ini berfungsi untuk melakukan vektorisasi data kedalam bentuk token sebanyak 2000 kata. Dan selanjutnya akan melakukan fit tokenizer hanya untuk data training saja tidak dengan data testingnya. Untuk ilustrasi lihat gambar berikut:



Gambar 8.115 Teori 5

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

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



Gambar 8.116 Teori 6

7. Jelaskan apa maksud dari fungsi `d train inputs = d train inputs/np.amax(np.absolutetrain 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 8.117 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])`

ical(d['CLASS'].iloc[test\_idx]) dalam kode program, dilengkapi dengan ilustrasi atau gambar.

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

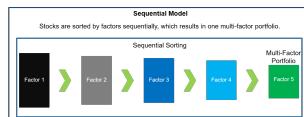
```
>>> Y_train
array([[1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.], dtype=int64)
>>> u_train = flatten(Y_train)
array([1., 0., ..., 1., 0., 0.], dtype=int64)
>>> Y_train
array([[1., 0., 0.],
       [0., 1., 0.],
       [1., 0.,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

Gambar 8.118 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.

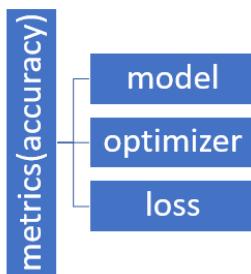


Gambar 8.119 Teori 9

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 8.120** Teori 10

11. Jelaskan apa itu Deep Learning.

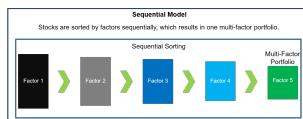
Deep Learning adalah salah satu cabang dari ilmu machine learning yang terdiri dari algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam machine learning dapat digunakan baik untuk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

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

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

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

1174070 mod3+1 x 1174070 mod3+1 = 2 x 2, adapun ilustrasi gambar nya sebagai berikut :



Gambar 8.121 Teori 9

### 8.6.2 Praktek Program

#### 1. Soal 1

```

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

```

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



Gambar 8.122 Hasil Soal 1.

#### 2. Soal 2

```

1 # In [2]:load all images (as numpy arrays) and save their
    classes
2 #membuat variabel imgs dengan variabel kosong
3 imgs = []
4 #membuat variabel classes dengan variabel kosong
5 classes = []
6 #membuka file hasy-data-labels.csv yang berada di foleder
    HASYv2 yang di inisialisasi menjadi csvfile
7 with open('HASYv2/hasy-data-labels.csv') as csvfile:
8     #membuat variabel csvreader yang berisi method csv.reader
        yang membaca variabel csvfile
9     csvreader = csv.reader(csvfile)
10    # membuat variabel i dengan isi 0
11    i = 0
12    # membuat looping pada variabel csvreader
13    for row in csvreader:
14        # dengan ketentuan jika i lebihkecil daripada 0

```

```

15     if i > 0:
16         # dibuat variabel img dengan isi keras untuk
17         # aktivasi neural network fungsi yang membaca data yang
18         # berada dalam folder HASYv2 dengan input nilai -1.0 dan 1.0
19         img = keras.preprocessing.image.img_to_array(
20             pil_image.open("HASYv2/" + row[0]))
21         # neuron activation functions behave best when
22         # input values are between 0.0 and 1.0 (or -1.0 and 1.0),
23         # so we rescale each pixel value to be in the
24         # range 0.0 to 1.0 instead of 0-255
25         #membagi data yang ada pada fungsi img sebanyak
26         255.0
27         img /= 255.0
28         # menambah nilai baru pada imgs pada row ke 1 2
29         # dan dilanjutkan dengan variabel img
30         imgs.append((row[0], row[2], img))
31         # menambahkan nilai pada row ke 2 pada variabel
32         classes
33         classes.append(row[2])
34         # penambahan nilai satu pada variabel i
35         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  $i \geq 0$ , 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
6 # 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 8.123 Hasil Soal 3.

#### 4. Soal 4

```

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

```

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

```

In [9]: import numpy as np
... train_input = np.asarray(list(map(lambda row: row[1], train)))
... test_input = np.asarray(list(map(lambda row: row[1], test)))
... train_output = np.asarray(list(map(lambda row: row[1], train)))
... test_output = np.asarray(list(map(lambda row: row[1], test)))

```

**Gambar 8.124** Hasil Soal 4.

#### 5. Soal 5

```

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

```

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

```

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

```

**Gambar 8.125** Hasil Soal 5.

#### 6. Soal 6

```

1 # In [6]: convert class names into one-hot encoding
2
3 # membuat variabel label_encoder dengan isi LabelEncoder
4 label_encoder = LabelEncoder()
5 # membuat variabel integer_encoded yang berfungsi untuk
6     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 8.126** Hasil Soal 6.

## 7. Soal 7

```

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

```

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

```

In [48]: onehot_encoder = OneHotEncoder(sparse=False)
... integer_encoded = integer_encoded.reshape(-1, (integer_encoded))
... onehot_encoder.fit(integer_encoded)
OneHotEncoder(categories='auto', drop=None, dtype<class 'numpy.float64'>,
handle_unknown='error', sparse=False)

```

**Gambar 8.127** Hasil Soal 7.

## 8. Soal 8

```

1 # In [8]: convert train and test output to one-hot
2 # mengkonvert data train output menggunakan variabel
3     label_encoder kedalam variabel train_output_int
3 train_output_int = label_encoder.transform(train_output)
4 # mengkonvert variabel train_output_int kedalam fungsi
5     onehot_encoder
5 train_output = onehot_encoder.transform(train_output_int.
6     reshape(len(train_output_int), 1))
6 # mengkonvert data test_output menggunakan variabel
7     label_encoder kedalam variabel test_output_int
7 test_output_int = label_encoder.transform(test_output)

```

```

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

```

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

```

In [9]: train_output_int = label_encoder.transform(train_output)
onehot_encoder.transform(test_output_int.reshape(len(test_output_int), 1))
... test_output = onehot_encoder.transform(test_output_int.reshape(len(test_output_int),
))
... num_classes = len(label_encoder.classes_)
... print("Number of classes: %d" % num_classes)
Value of classes: 39

```

**Gambar 8.128** Hasil Soal 8.

## 9. Soal 9

```

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

```

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

```

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

```

**Gambar 8.129** Hasil Soal 9.

## 10. Soal 10

```

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

```

```

9 # variabel model di tambahkan dengan librari Conv2D 32bit
  dengan kernel 3 x 3
10 model.add(Conv2D(32, (3, 3), activation='relu'))
11 # variabel model di tambahkan dengan lib MaxPooling2D dengan
  ketentuan ukuran 2 x 2 pixcel
12 model.add(MaxPooling2D(pool_size=(2, 2)))
13 # variabel model di tambahkan librari Flatten
14 model.add(Flatten())
15 # variabel model di tambahkan librari Dense dengan fungsi
  tanh
16 model.add(Dense(1024, activation='tanh'))
17 # variabel model di tambahkan librari dropout untuk memangkas
  data tree sebesar 50 persen
18 model.add(Dropout(0.5))
19 # variabel model di tambahkan librari Dense dengan data dari
  num_classes dan fungsi softmax
20 model.add(Dense(num_classes, activation='softmax'))
21 # mengkompile data model untuk mendapatkan data loss akurasi
  dan optimasi
22 model.compile(loss='categorical_crossentropy', optimizer='adam',
               metrics=['accuracy'])
23 # mencetak variabel model kemudian memunculkan kesimpulan
  berupa data total parameter, trainable paremeter dan bukan
  trainable parameter
24 print(model.summary())
25

```

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

```

 1   model.add(conv2d_0, kernel_size=(3, 3), activation='relu')
 2   input_shape=np.shape(xtrain_input[1])
 3   input_shape=(input_shape[0], input_shape[1], input_shape[2], 1)
 4   model.add(conv2d_0, input_shape=input_shape, activation='relu')
 5   model.add(conv2d_1, (3, 3), activation='relu')
 6   model.add(maxpooling2d_0, pool_size=(2, 2))
 7   model.add(conv2d_2, (3, 3), activation='relu')
 8   model.add(conv2d_3, (3, 3), activation='relu')
 9   model.add(maxpooling2d_1, pool_size=(2, 2))
10   model.add(dense_4, activation='tanh')
11   model.add(dense_5, activation='softmax')
12   model.compile(loss='categorical_crossentropy', optimizer='adam',
13                 metrics=['accuracy'])
14   print(model.summary())
15
Model: sequential_5
Layer (Type)          Output Shape         Param #
conv2d_0 (Conv2D)      (None, 10, 10, 32)      896
max_pooling2d_0 (MaxPooling2D) (None, 5, 5, 32)      0
conv2d_1 (Conv2D)      (None, 5, 5, 32)      9248
max_pooling2d_1 (MaxPooling2D) (None, 2, 2, 32)      0
conv2d_2 (Conv2D)      (None, 2, 2, 32)      9248
conv2d_3 (Conv2D)      (None, 2, 2, 32)      9248
max_pooling2d_2 (MaxPooling2D) (None, 1, 1, 32)      0
flatten_0 (Flatten)    (None, 32)           0
dense_4 (Dense)        (None, 128)          147584
dropout_4 (Dropout)    (None, 128)          0
dense_5 (Dense)        (None, 699)          47691
Total params: 205,299
Trainable params: 205,299
Non-trainable params: 0

```

Gambar 8.130 Hasil Soal 10.

## 11. Soal 11

```

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

```

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

```
(In [1]): tensorboard = keras.callbacks.TensorBoard(log_dir='/tmp/mnist_logs')
```

**Gambar 8.131** Hasil Soal 11.

## 12. Soal 12

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

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

```
Epoch 1/10: loss: 1.5916 - accuracy: 0.6000 - val_loss: 1.8732 - val_accuracy: 0.7013
WARNING:tensorflow:From C:\Users\A991\PycharmProjects\Keras\keras\models\train.py:10: in eager_tensorflow_scope (as part of eager execution):
  raise RuntimeError("TensorFlow operations can't be captured by tf.function. Instead, use tf.function with eager=True or use tf.compat.v1.Session instead of tf.Session")
Epoch 2/10: loss: 1.0861 - accuracy: 0.7248 - val_loss: 0.3669 - val_accuracy: 0.9523
Epoch 3/10: loss: 0.8491 - accuracy: 0.7707 - val_loss: 0.8052 - val_accuracy: 0.9592
Epoch 4/10: loss: 0.6220 - accuracy: 0.7612 - val_loss: 0.8466 - val_accuracy: 0.929
Epoch 5/10: loss: 0.7753 - accuracy: 0.7718 - val_loss: 0.8479 - val_accuracy: 0.9318
Epoch 6/10: loss: 0.7216 - accuracy: 0.7784 - val_loss: 0.8492 - val_accuracy: 0.9461
Epoch 7/10: loss: 0.7044 - accuracy: 0.7859 - val_loss: 0.8566 - val_accuracy: 0.9354
Epoch 8/10: loss: 0.6795 - accuracy: 0.7924 - val_loss: 0.8460 - val_accuracy: 0.9339
Epoch 9/10: loss: 0.6552 - accuracy: 0.7993 - val_loss: 0.8513 - val_accuracy: 0.9461
Epoch 10/10: loss: 0.6412 - accuracy: 0.8021 - val_loss: 0.8792 - val_accuracy: 0.9511
test loss: 0.70051298003845
test accuracy: 0.90541298003845
```

**Gambar 8.132** Hasil Soal 12.

## 13. Soal 13

```
1 # In[13]: try various model configurations and parameters to
   find the best
2 # mengimport librari time
3 import time
4 #membuat variabel result dengan array kosong
5 results = []
6 # melakukan looping dengan ketentuan konvolusi 2 dimensi 1 2
7 for conv2d_count in [1, 2]:
8     # menentukan ukuran besaran fixcel dari data atau konvert
       1 fixcel mnjadi data yang berada pada codigan dibawah.
```

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

```

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

```

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

Gambar 8.133    Hasil Soal 13.

## 14. Soal 14

```

1 # In [14]: rebuild/retrain a model with the best parameters (
2   # from the search) and use all data
3 # membuat variabel model dengan isian librari Sequential
4 model = Sequential()

```

```

4 # variabel model di tambahkan librari Conv2D tigapuluhan dua
    bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
    relu dang menggunakan data train_input
5 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
    input_shape=np.shape(train_input[0])))
6 # variabel model di tambahkan dengan lib MaxPooling2D dengan
    ketentuan ukuran 2 x 2 pixcel
7 model.add(MaxPooling2D(pool_size=(2, 2)))
8 # variabel model di tambahkan dengan librari Conv2D 32bit
    dengan kernel 3 x 3
9 model.add(Conv2D(32, (3, 3), activation='relu'))
10 # variabel model di tambahkan dengan lib MaxPooling2D dengan
    ketentuan ukuran 2 x 2 pixcel
11 model.add(MaxPooling2D(pool_size=(2, 2)))
12 # variabel model di tambahkan librari Flatten
13 model.add(Flatten())
14 # variabel model di tambahkan librari Dense dengan fungsi
    tanh
15 model.add(Dense(128, activation='tanh'))
16 # variabel model di tambahkan librari dropout untuk memangkas
    data tree sebesar 50 persen
17 model.add(Dropout(0.5))
18 # variabel model di tambahkan librari Dense dengan data dari
    num_classes dan fungsi softmax
19 model.add(Dense(num_classes, activation='softmax'))
20 # mengkompile data model untuk mendapatkan data loss akurasi
    dan optimasi
21 model.compile(loss='categorical_crossentropy', optimizer='adam',
    metrics=['accuracy'])
22 # mencetak variabel model kemudian memunculkan kesimpulan
    berupa data total parameter , trainable paremeter dan bukan
    trainable parameter
23 print(model.summary())

```

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

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 16, 36, 32)	896
max_pooling2d_6 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_7 (Conv2D)	(None, 15, 15, 32)	9248
max_pooling2d_7 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_5 (Flatten)	(None, 1152)	0
dense_9 (Dense)	(None, 128)	147584
dropout_4 (Dropout)	(None, 128)	0
dense_10 (Dense)	(None, 369)	47681
<hr/>		
Total params:	269,329	
Trainable params:	269,329	
Non-trainable params:	0	
<hr/>		

Gambar 8.134 Hasil Soal 14.

## 15. Soal 15

```

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

```

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

```

In [16]: model.fit(np.concatenate((train_input, test_input)),
                  np.concatenate((train_output, test_output)),
                  batch_size= , epochs= , verbose=2)
Epoch 1/10
  10s - loss: 1.7795 - accuracy: 0.5971
Epoch 2/10
  11s - loss: 1.6744 - accuracy: 0.7074
Epoch 3/10
  11s - loss: 0.9564 - accuracy: 0.7320
Epoch 4/10
  11s - loss: 0.8977 - accuracy: 0.7458
Epoch 5/10
  11s - loss: 0.8571 - accuracy: 0.7577
Epoch 6/10
  11s - loss: 0.8297 - accuracy: 0.7586
Epoch 7/10
  11s - loss: 0.8079 - accuracy: 0.7586
Epoch 8/10
  12s - loss: 0.7917 - accuracy: 0.7663
Epoch 9/10
  12s - loss: 0.7765 - accuracy: 0.7707
Epoch 10/10
  11s - loss: 0.7639 - accuracy: 0.7718
[111/111] - 1s - 100% |-----| done.
<ipython-input-16-2d2e383a2a3c>:10 in <module>()
      1 # save the trained model
      2 # menyimpan model atau mengekspor model yang telah di
         jalantadi
      3 model.save("mathsymbols.model")

```

**Gambar 8.135** Hasil Soal 15.

## 16. Soal 16

```

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

```

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

```

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

```

**Gambar 8.136** Hasil Soal 16.

## 17. Soal 17

```

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

```

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

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

**Gambar 8.137** Hasil Soal 17.

## 18. Soal 18

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

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

Layer (type)	Output Shape	Param #
conv2d_68 (Conv2D)	(None, 10, 10, 32)	896
max_pooling2d_68 (MaxPooling2D)	(None, 5, 5, 32)	0
conv2d_69 (Conv2D)	(None, 5, 5, 32)	9248
max_pooling2d_69 (MaxPooling2D)	(None, 2, 2, 32)	0
flatten_45 (Flatten)	(None, 1152)	0
dense_89 (Dense)	(None, 128)	147584
dropout_34 (Dropout)	(None, 128)	0
dense_90 (Dense)	(None, 369)	47601
<hr/>		
total params	205,329	
Trainable params	195,329	
Non-trainable params	0	
<hr/>		
None		

**Gambar 8.138** Hasil Soal 18.

## 19. Soal 19

```
1 # In[19]: restore the class name to integer encoder
2 # membuat variabel label encoder ke 2 dengan isian fungsi
      label encoder.
3 label_encoder2 = LabelEncoder()
4 # menambahkan method classess dengan data classess yang di
      eksport tadi
5 label_encoder2.classes_ = np.load('classes.npy')
6 # membuat fungsi predict dengan path img
7 def predict(img_path):
8     # membuat variabel newimg dengan membuat immage menjadi
      array dan membuka data berdasarkan img path
9     newimg = keras.preprocessing.image.img_to_array(pil_image
      .open(img_path))
```

```

10 # membagi data yang terdapat pada variabel newimg
11      sebanyak 255
12      newimg /= 255.0
13
14      # do the prediction
15      # membuat variabel predivtion dengan isian variabel
16      # model2 menggunakan fungsi predic dengan syarat variabel
17      # newimg dengan data reshape
18      prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
19
20      # figure out which output neuron had the highest score,
21      # and reverse the one-hot encoding
22      # membuat variabel inverted denagan label encoder2 dan
23      # menggunakan argmax untuk mencari skor luaran tertinggi
24      inverted = label_encoder2.inverse_transform([np.argmax(
25          prediction)])
26      # mencetak prediksi gambar dan confidence dari gambar.
27      print("Prediction: %s, confidence: %.2f" % (inverted[0] ,
28          np.max(prediction)))

```

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

Layer (type)	Output Shape	Param #
conv2d_68 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_68 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_69 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_69 (MaxPooling2D)	(None, 6, 6, 32)	0
Flatten_45 (Flatten)	(None, 1152)	0
dense_89 (Dense)	(None, 128)	147384
dropout_34 (Dropout)	(None, 128)	0
dense_90 (Dense)	(None, 369)	47601
<hr/>		
Total params: 205,229		
Trainable params: 205,229		
Non-trainable params: 0		
<hr/>		
None		

Gambar 8.139    Hasil Soal 19.

## 20. Soal 20

```

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

```

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

```
# grab an image (we'll just use a random training image for demonstration purposes)
predict('mnist2/mnist-data/v2-00001.png')
prediction: 0, confidence: 0.87
predict('mnist2/mnist-data/v2-00002.png')
prediction: 1, confidence: 0.58
predict('mnist2/mnist-data/v2-00003.png')
prediction: 0, confidence: 0.88
```

Gambar 8.140 Hasil Soal 20.

### 8.6.3 Penanganan Error

#### 1. KeyboardInterrupt

```
File "c:\Users\user\PycharmProjects\untitled1\main.py", line 1, in <module>
    img = keras.preprocessing.image.load_img("MNISTV2\train\00001.png")
  File "C:\Users\user\anaconda3\envs\py36\lib\site-packages\pillow\Image.py", line 200, in open
    fp = builtins.openfilename("r")
```

Gambar 8.141 KeyboardInterrupt

#### 2. Cara Penanganan Error

- KeyboardInterrupt

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

### 8.6.4 Bukti Tidak Plagiat



Gambar 8.142 Bukti Tidak Melakukan Plagiat Chapter 7

## 8.7 1174087 - Ilham Muhammad Ariq

### 8.7.1 Teori

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

Untuk memudahkan mesin memahami maksud dari apa yang kita inginkan dalam machine learning, kata pada teks disebut token, dan proses vektorisasi dari bentuk kata ke dalam token tersebut disebut tokenizer dan tokenizer akan merubah sebuah teks menjadi simbol, kata, ataupun

biner dan bentuk lainnya kedalam token. Ilustrasinya misalkan saya mempunyai sebuah kalimat yaitu "Nama Saya Ilham Muhammad Ariq" maka ketika kita lakukan proses tokenizer maka akan berubah menjadi ['Nama', 'Saya', 'Ilham', 'Muhammad', 'Ariq'].

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

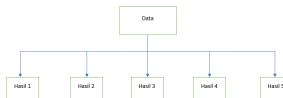
```

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

```

**Listing 8.1** K Fold Cross Validation

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



**Gambar 8.143** Illustrasi K Fold Cross Validation

- Jelaskan apa maksudnya kode program *for train, test in splits*.dilengkapi dengan ilustrasi atau gambar.  
Maksudnya yaitu untuk menguji apakah setiap data pada dataset sudah di split dan tidak terjadi penumpukan. Yang dimana maksudnya disetiap class tidak akan muncul id yang sama. Ilustrasinya misalkan kita memiliki 5 sepatu dengan model yang berbeda. Kemudian kita bagikan ke kedua orang , tentunya setiap orang yang menerima sepatu tidak memiliki model sepatu yang sama.
- Jelaskan apa maksudnya kode program *train\_content = d['CONTENT'].iloc[train\_idx]* dan *test\_content = d['CONTENT'].iloc[test\_idx]*. dilengkapi dengan ilustrasi atau gambar.  
Maksudnya yaitu mengambil data pada kolom atau index CONTENT yang merupakan bagian dari train idx dan test idx. Ilustrasinya, ketika data telah diubah menjadi train dan test maka kita dapat memilihnya untuk ditampilkan pada kolom yang diinginkan.
- Jelaskan apa maksud dari fungsi *tokenizer = Tokenizer(num\_words=2000)* dan *tokenizer.fit\_on\_texts(train\_content)*, dilengkapi dengan ilustrasi atau gambar.

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



**Gambar 8.144** Illustrasi fit tokenizer dan num\_word=2000

6. Jelaskan apa maksud dari fungsi `d.train_inputs = tokenizer.texts_to_matrix(train_content, mode='tfidf')` dan `d.test_inputs = tokenizer.texts_to_matrix(test_content, mode='tfidf')`, dilengkapi dengan ilustrasi kode dan atau gambar. Untuk digunakan sebagai pengubah urutan teks yang tadi telah dilakukan tkoenizer menjadi matriks yang berurutan seperti tf idf



**Gambar 8.145** Illustrasi d train inputs = tokenizer.texts to matrix

7. Jelaskan apa maksud dari fungsi `d.train_inputs = d_train_inputs/npamax(np.abs(d_train_inputs))` dan `d.test_inputs = d_test_inputs/npamax(np.absolute(d_test_inputs))`, dilengkapi dengan ilustrasi atau gambar. Fungsi tersebut digunakan untuk membagi matriks tfidf dnegan penentuan maksimum array sepanjang sumbu sehingga akan menimbulkan garis ke bawah dan ke atas yang membentuk gambar v. Lalu hasil tersebut akan dimasukkan ke variabel d train input dan d test input dengan methode absolute. Yang berarti tanpa bilangan negatif.
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. Maksud dari fungsi tersebut yaitu untuk merubah nilai vektor yang ada pada atribut class menjadi bentuk matrix dengan pengurutan berdasarkan data index training dan testing.
9. Jelaskan apa maksud dari fungsi di listing ???. Gambarkan ilustrasi Neural Network nya dari model kode tersebut.

```

1     model = Sequential()
2     model.add(Dense(512, input_shape=(2000,)))
3     model.add(Activation('relu'))
  
```

```

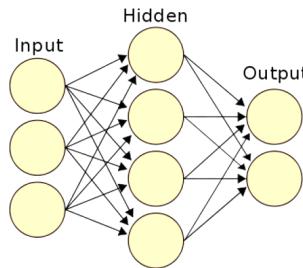
4     model.add(Dropout(0.5))
5     model.add(Dense(2))
6     model.add(Activation('softmax'))
7

```

**Listing 8.2** Membuat model Neural Network

Penjelasannya sebagai berikut:

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

**Gambar 8.146** Illustrasi Neural Network Pemodelan

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

```

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

```

**Listing 8.3** Compile model

Melakukan peng compile-an dari model Sequential tadi dengan Loss yang dengang merupakan fungsi optimisasi kormenggunakan categorical crossentropy,danmenggunakan algoritma adam sebagai optimizer. Adam yaitu

algoritma pengoptimalan yang dapat digunakan sebagai gantikan dari proses turunan gradien stokastik klasik untuk memperbarui bobot jaringan yang berulang berdasarkan data training. Dengan metrik yaitu fungsi yang digunakan untuk menilai kinerja mode Anda disini menggunakan fungsi accuracy.

11. Jelaskan apa itu Deep Learning

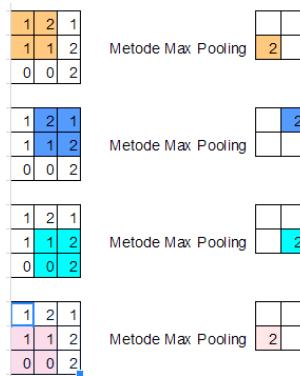
Deep Learning adalah subbidang machine learning yang berkaitan dengan algoritma yang terinspirasi oleh struktur dan fungsi otak yang disebut jaringan saraf tiruan atau Artificial Neural Networks. Jaringan saraf tiruan, algoritma yang terinspirasi oleh otak manusia, belajar dari sejumlah besar data. Demikian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

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

Deep Neural Network adalah jaringan syaraf tiruan (JST) dengan beberapa lapisan antara lapisan input dan output. DNN menemukan manipulasi matematis yang benar untuk mengubah input menjadi output, apakah itu hubungan linear atau hubungan non-linear. Merupakan jaringan syaraf dengan tingkat kompleksitas tertentu, jaringan syaraf dengan lebih dari dua lapisan. Deep Neural Network menggunakan pemodelan matematika yang canggih untuk memproses data dengan cara yang kompleks. DNN hanya terdiri dari dua laipsan yaitu input dan output, sedangkan dalam Deep learning kita dapat mendefinisikan layer sebanyak yang kita inginkan atau butuhkan.

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

Sebelum membuat ilustrasi perlu di ketahui apa itu stride, stride adalah acuan atau parameter yang menentukan pergeseran pada filter fixcel. sebagai contoh nilai stride 1 yang berarti filter akan bergeser sebanyak satu fixcel secara vertikal dan horizontal. selanjutnya apa itu max pooling contoh pada suatu gambar di tentukan Max Pooling dari 3 x 3 dengan stride 1 yang berarti setiap pergeseran 1 pixcel akan diambil nilai terbesar dari pixcel 3 x 3 tersebut.



**Gambar 8.147** Illustrasi perhitungan stride 1 max pooling

### 8.7.2 Praktek

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

```

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

```

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

```

1 # In [2]:load all images (as numpy arrays) and save their
   classes
2 #Menginisiasi variabel imgs dan classes dengan variabel array
   kosong
3 imgs = []
4 classes = []
5 #membuka file hasy-data-labels.csv yang berada di foleder
   HASYv2 yang di inisialisasi menjadi csvfile
6 with open('HASYv2/hasy-data-labels.csv') as csvfile:
7     #Menginisiasi variabel csvreader yang berisi method csv.
   reader yang membaca variabel csvfile
8     csvreader = csv.reader(csvfile)
9     # Menginisiasi variabel i dengan isi 0
10    i = 0

```

```

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

```

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

```

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

```

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

```

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

```

```

9 train_output = np.asarray(list(map(lambda row: row[1], train)))
10 # Menginisiasi variabel test_output dengan np method asarray
    yang mana membuat array dengan isi row 1 dari data test
11 test_output = np.asarray(list(map(lambda row: row[1], test)))

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

1 # In [8]:convert train and test output to one-hot
2 # Menconvert data train output menggunakan variabel
    label_encoder kedalam variabel train_output_int
3 train_output_int = label_encoder.transform(train_output)
4 # Menconvert variabel train_output_int kedalam fungsi
    onehot_encoder

```

```

5 train_output = onehot_encoder.transform(train_output_int.
6     reshape(len(train_output_int), 1))
# Menconvert data test_output menggunakan variabel
7 label_encoder kedalam variabel test_output_int
8 test_output_int = label_encoder.transform(test_output)
# Menconvert variabel test_output_int kedalam fungsi
9     onehot_encoder
test_output = onehot_encoder.transform(test_output_int.
10    reshape(len(test_output_int), 1))
# Menginisiasi variabel num_classes dengan isi variabel
11 label_encoder dan classes
12 num_classes = len(label_encoder.classes_)
# mencetak hasil dari nomer Class berupa persen
13 print("Number of classes: %d" % num_classes)

```

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

```

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

```

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

```

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

```

```

16 model.add(tf.keras.layers.Dense(1024, activation='tanh'))
17 # variabel model di tambahkan library dropout untuk memangkas
    data tree sebesar 50 persen
18 model.add(tf.keras.layers.Dropout(0.5))
19 # variabel model di tambahkan library Dense dengan data dari
    num_classes dan fungsi softmax
20 model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
21 # mengkompile data model untuk mendapatkan data loss akurasi
    dan optimasi
22 model.compile(loss='categorical_crossentropy', optimizer='adam',
                metrics=['accuracy'])
23 # mencetak variabel model kemudian memunculkan kesimpulan
    berupa data total parameter, trainable paremeter dan bukan
    trainable parameter
24 print(model.summary())

```

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

```

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

```

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

```

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

```

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

```

1 # In[13]:try various model configurations and parameters to
    find the best

```

```

2 # Melakukan import library time
3 import time
4 #Menginisiasi variabel result dengan array kosong
5 results = []
6 # melakukan looping dengan ketentuan konvolusi 2 dimensi 1 2
7 for conv2d_count in [1, 2]:
8     # menentukan ukuran besaran fixcel dari data atau konvert
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                # Menginisiasi variabel model Sequential
16                model = tf.keras.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                        # Penambahan method add pada variabel
23                        model dengan konvolusi 2 dimensi 32 bit didalamnya dan
24                        membuat kernel dengan ukuran 3 x 3 dan rumus aktifasi relu
25                        dan data shape yang di hitung dari data train.
26                        model.add(tf.keras.layers.Conv2D(32,
27                        kernel_size=(3, 3), activation='relu', input_shape=np.
28                        shape(train_input[0])))
29                        # jika tidak
30                    else:
31                        # Penambahan method add pada variabel
32                        model dengan konvolusi 2 dimensi 32 bit dengan ukuran
33                        kernel 3 x3 dan fungsi aktivasi relu
34                        model.add(tf.keras.layers.Conv2D(32,
35                        kernel_size=(3, 3), activation='relu'))
36                        # Penambahan method add pada variabel model
37                        dengan isian method Max pooling berdimensi 2 dengan
38                        ukuran fixcel 2 x 2.
39                        model.add(tf.keras.layers.MaxPooling2D(
40                        pool_size=(2, 2)))
41                        # merubah feature gambar menjadi 1 dimensi vektor
42                        model.add(tf.keras.layers.Flatten())
43                        # Penambahan method dense untuk pemanjangan data
44                        dengan ukuran dense di tentukan dengan rumus fungsi tanh.
45                        model.add(tf.keras.layers.Dense(dense_size,
46                        activation='tanh'))
47                        # membuat ketentuan jika pemangkasan lebih besar
48                        dari 0 persen
49                        if dropout > 0.0:
50                            # Penambahan method dropout pada model dengan
51                            nilai dari dropout
52                            model.add(tf.keras.layers.Dropout(dropout))
53                            # Penambahan method dense dengan fungsi num
54                            classs dan rumus softmax
55                            model.add(tf.keras.layers.Dense(num_classes ,
56                            activation='softmax'))

```

```

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

```

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

```

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

```

```

8 # variabel model di tambahkan dengan library Conv2D 32bit
    dengan kernel 3 x 3
9 model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu
    '))
10 # variabel model di tambahkan dengan lib MaxPooling2D dengan
     ketentuan ukuran 2 x 2 pixcel
11 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
12 # variabel model di tambahkan library Flatten
13 model.add(tf.keras.layers.Flatten())
14 # variabel model di tambahkan library Dense dengan fungsi
     tanh
15 model.add(tf.keras.layers.Dense(128, activation='tanh'))
16 # variabel model di tambahkan library dropout untuk memangkas
     data tree sebesar 50 persen
17 model.add(tf.keras.layers.Dropout(0.5))
18 # variabel model di tambahkan library Dense dengan data dari
     num_classes dan fungsi softmax
19 model.add(tf.keras.layers.Dense(num_classes, activation='
     softmax'))
20 # mengkompile data model untuk mendapatkan data loss akurasi
     dan optimasi
21 model.compile(loss='categorical_crossentropy', optimizer='
     adam', metrics=['accuracy'])
22 # mencetak variabel model kemudian memunculkan kesimpulan
     berupa data total parameter, trainable paremeter dan bukan
     trainable parameter
23 print(model.summary())

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

### 8.7.3 Penanganan Error

### 1. SS Error

```
File "cipython-input-1-37390cfadda9", line 4, in <module>
    with open('HASV2/hasy-data-labels.csv') as csvfile:
```

FileNotFoundException: [Errno 2] No such file or directory: 'HASV2/hasy-data-labels.csv'

Gambar 8.148 File Not Found error

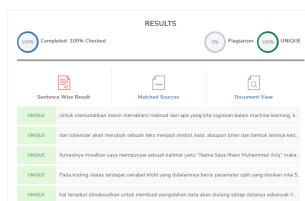
## 2. Jenis Error

- File Not Found Error

### 3. Cara Penanganan

Dengan cara menyesuaikan letak file yang ingin dibaca/ digunakan

#### 8.7.4 Bukti Tidak Plagiat



**Gambar 8.149** Tidak Melakukan Plagiat Pada Ch 7

### 8.7.5 Link Video Youtube

<https://youtu.be/XPdKuIEMeu8>