

## **BAB 1**

---

## **CHAPTER 1**

---



## **BAB 2**

---

## **CHAPTER 2**

---



## **BAB 3**

---

## **CHAPTER 3**

---



## **BAB 4**

---

## **CHAPTER 4**

---



## **BAB 5**

---

## **CHAPTER 4**

---



# BAB 6

---

## CHAPTER 5

---

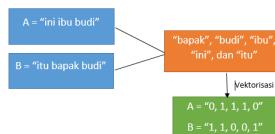
### 6.1 1174054 - Aulyardha Anindita

#### 6.1.1 Teori

1. Mengapa kata-kata harus dilakukan vektorisasi

Kata-kata harus dilakukan vektorisasi karena pada saat kita akan menggunakan machine learning tidak bisa secara langsung menggunakan teks melainkan kita harus mengubah teks tersebut menjadi sebuah angka. vektor tersebut dikonversi menjadi binari (0 dan 1 ) hal itu dilakukan karena komputer adalah alat untuk menghitung. Disamping itu untuk merepresentasikan data teks dalam mengubah kedalam bentuk angka menggunakan algortima prediksi. algoritma tersebut akan mengambil vektor angka sebagai input, maka dari itu kita perlu mengkonversi doku-men menjadi vektor angka dengan panjang tetap atau sama. Contohnya, ada dua frasa sederhana A = ini ibu budi dan B = itu bapak budi , setelah dilakukan proses konversi dan digabungkan hasilnya maka akan diperoleh bapak, budi, ibu, ini, itu. lalu vektorisasi dilakukan kemas-ing masing frase tersebut maka akan diperoleh hasil  $A = 0,1,1,1,0$  untuk

frase pertama dan  $B = 1,1,0,0,1$  untuk frase kedua dengan 0 sebagai representasi tidak ditemukannya dalam teks dan 1 sebagai representasi ditemukannya dalam teks. Untuk lebih jelasnya dapat dilihat pada ilustrasi gambar dibawah ini :



**Gambar 6.1** Vektorisasi Teks

## 2. Mengapa dimensi dari vektor dataset google bisa sampai 300

Dimensi dari vektor dataset google bisa sampai 300 disebabkan karena dalam satu dataset berisi setidaknya 3 miliar kata dan kalimat. Dimensi tersebut berisikan kata-kata yang unik sehingga dimensi pada dataset google bisa mencapai 300. Contohnya, terdapat kata word dan earth pada dataset google tersebut setiap kata tersebut dibuat dimensi vektor 300 untuk kata word dan 300 dimensi vektor juga untuk kata earth lalu kata tersebut dibandingkan bobot kesamaan katanya maka akan muncul akurasi sekitar 70 persen kesamaan bobot dikarenakan kata word dan earth sama-sama digunakan untuk kata benda.



**Gambar 6.2** Dataset Google

## 3. Konsep vektorisasi untuk kata

Vektorisasi untuk kata dengan mengetahui kata tengah dari suatu kalimat atau kata utama atau objek utama pada suatu kalimat. Contohnya seperti: jangan lupa untuk menjaga kesehatan di tengah wabah pandemik ini. kata tengah tersebut merupakan kesehatan yang memiliki bobot sebagai kata tengah dari suatu kalimat atau objek dari suatu kalimat. hal tersebut sangat berkaitan dengan dimensi vektor pada dataset google yang 300 tadi karena untuk mendapatkan nilai atau bobot dari kata tengah tersebut didapatkan dari proses dimensiasi dari kata tersebut.



**Gambar 6.3** Vektorisasi untuk kata

#### 4. Konsep Vektorisasi untuk dokumen

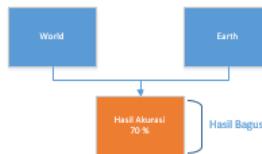
Konsep vektorisasi untuk dokumen hampir sama seperti vektorisasi untuk kata hanya saja pemilihan katanya digunakan untuk mencari kesamaan atau memprediksi seberapa sering kemunculan kata dalam kalimat atau paragraf. contohnya misalnya dalam sebuah artikel kita ingin mencari seberapa banyak kata saya muncul. maka dengan dengan doc2vec dapat diprediksi hasilnya.



**Gambar 6.4** Vektorisasi untuk dokumen

#### 5. Mean dan Standar Deviasi

Mean merupakan suatu petunjuk terhadap kata-kata yang diolah jika kata itu memiliki akurasi yang tinggi berarti kata tersebut sering muncul begitu juga sebaliknya. Standar deviasi adalah standar untuk menimbang suatu kesalahan, sehingga kesalahan tersebut akan dianggap wajar misalnya ketika kita memperkirakan kedalaman dari dataset merupakan 2 atau 3 tapi pada kenyataannya merupakan 5 itu merupakan kesalahan tapi masih bisa dianggap wajar karena masih mendekati perkiraan awal.

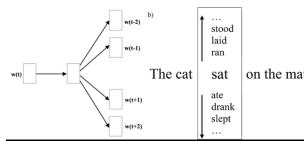


**Gambar 6.5** Mean dan Standar Deviasi

#### 6. Skip-gram

Skip-gram adalah salah satu teknik pembelajaran tanpa pengawasan yang digunakan untuk menemukan kata yang paling terkait untuk kata yang diberikan. Skip-gram digunakan untuk memprediksi kata konteks

untuk kata target yang diberikan. Ini kebalikan dari algoritma CBOW. Di sini, kata target adalah input sedangkan kata konteks adalah output. Karena ada lebih dari satu kata konteks yang dapat diprediksi yang membuat masalah ini sulit.



**Gambar 6.6** Skip-gram

### 6.1.2 Praktek

#### 1. Nomor 1

```
1 import gensim, logging
2 logging.basicConfig(format='%(asctime)s : %(levelname)s : %(
    message)s', level=logging.INFO)
3 dita = gensim.models.KeyedVectors.load_word2vec_format('D:/
    Mata Kuliah/Tingkat 3/Semester 6/Kecerdasan Buatan/Chapter
    5/GoogleNews-vectors-negative300.bin', binary=True, limit
    =500000)
```

Dalam kode diatas berfungsi untuk mengimport library gensim. Gensim berfungsi untuk melakukan pemodelan dengan dataset atau topik yang telah ditentukan. Dalam keluaran 100 logging library opsional karena logging hanya menampilkan berupa log untuk setiap code yang dijalankan. Dan dalam keluaran 101 tersebut hasilnya berupa vektor google. Untuk hasil dari program tersebut dapat dilihat pada gambar dibawah ini :

```
In [78]: import gensim, logging
        logging.basicConfig(format='%(asctime)s : %(levelname)s : %(
            message)s', level=logging.INFO)
        dita = gensim.models.KeyedVectors.load_word2vec_format('D:/Mata Kuliah/
Tingkat 3/Semester 6/Kecerdasan Buatan/Chapter 5/GoogleNews-vectors-
negative300.bin', binary=True, limit=500000)
2020-04-09 13:30:09,756 : INFO : 1 loading projection weights from D:/Mata Kuliah/
Tingkat 3/Semester 6/Kecerdasan Buatan/Chapter 5/GoogleNews-vectors-
negative300.bin
2020-04-09 13:30:18,296 : INFO : loaded (500000, 300) matrix from D:/Mata Kuliah/
Tingkat 3/Semester 6/Kecerdasan Buatan/Chapter 5/GoogleNews-vectors-
negative300.bin
```

**Gambar 6.7** Hasil Nomor 1

```
1 #%%
2 dita[ 'love' ]
```

Dalam kode diatas digunakan untuk menampilkan hasil vektorisasi data dari kata love. Untuk hasilnya dapat dilihat pada gambar berikut :

**Gambar 6.8** Hasil Nomor 1 - Love

1 %%  
2 dita [ 'faith' ]

Dalam kode diatas digunakan untuk menampilkan hasil vektorisasi data dari kata faith. Untuk hasilnya dapat dilihat pada gambar berikut :

**Gambar 6.9** Hasil Nomor 1 - Faith

1 #%%  
2 dita [ 'fall '

Dalam kode diatas digunakan untuk menampilkan hasil vektorisasi data dari kata fall. Untuk hasilnya dapat dilihat pada gambar berikut :

Gambar 6.10 Hasil Nomor 1 Fall

```
1 %%  
2 dita [ 'sick' ]
```

Dalam kode diatas digunakan untuk menampilkan hasil vektorisasi data dari kata sick. Untuk hasilnya dapat dilihat pada gambar berikut :

```
[7]: %%time
[Out[7]:] [dicts['click']]
array([[ 1.4041805e+01, -4.0573745e-02, 1.64607590e+01,
       -5.7956525e+01, 3.2255652e+01, 7.812815e+01, -1.07469385e+01,
       1.01741245e+01, 1.0200000e+01, 1.6690750e+01, -1.08465312e+01,
       1.66615625e+01, 1.7963790e+01, 5.9208160e+01, -2.45117312e+01,
       1.7871935e+01, 9.1931239e+01, 1.0000000e+01, -1.95325080e+01,
       1.08460425e+01, 2.0526500e+01, -4.0571952e+01, 1.09055395e+01,
       1.08460425e+01, 2.0526500e+01, -4.0571952e+01, 1.09055395e+01,
       1.1852544e+01, 1.2364807e+01, -6.3097354e+01, 4.6788000e+01,
       1.1853950e+01, -5.1288000e+01, 1.04578312e+01, -1.51373188e+01,
       1.02171875e+01, 1.02681172e+01, 2.0828321e+01, -1.29705624e+01,
       1.02171875e+01, 1.02681172e+01, 2.0828321e+01, -1.29705624e+01,
       7.1289052e+01, 4.1791172e+01, -0.05078152e+01, 5.71239862e+01,
       4.4872455e+01, 2.2832812e+01, 1.0095125e+01, 7.8062502e+01,
       3.5742175e+01, 1.8212483e+01, 1.0000000e+01, -5.85371894e+01,
       2.1894224e+01, 1.7382152e+01, -0.00878372e+01, 5.93221719e+01,
       1.8834854e+01, 1.5258000e+01, 1.03875422e+01, -2.26525084e+01,
       5.5408000e+01, -0.02550800e+01, 1.03875422e+01, -2.26525084e+01,
       2.01317785e+01, -0.11717750e+01, 1.0375685e+01, -2.17737078e+01,
       1.18936544e+01, -4.18719500e+01, 2.2176790e+01, 1.04464375e+01,
       1.18936544e+01, -4.18719500e+01, 2.2176790e+01, 1.04464375e+01,
       9.4232500e+00, -5.1258000e+01, 6.4756525e+01, -2.47077354e+01]]
```

**Gambar 6.11** Hasil Nomor 1 - Sick

```
1 %%  
2 dita [ 'clear' ]
```

Dalam kode diatas digunakan untuk menampilkan hasil vektorisasi data dari kata clear. Untuk hasilnya dapat dilihat pada gambar berikut :

```
[475]: data['close'].head(10)
```

```
array([ 1.2059731e-04, -1.2059731e-01, -1.4941486e-04, -1.2480868e-02,
```

```
-1.0768795e-01, -1.4648475e-04, -1.7677923e-01, -1.6484675e-01,
```

```
-1.5477923e-01, -1.2532812e-01, -1.3678130e-01, -1.3095594e-01,
```

```
-1.2255265e-01, -1.4451526e-01, -1.1815846e-01, -1.0871256e-01, -1.0871256e-01,
```

```
-0.9676987e-02, -4.1310830e-01, -1.7873938e-01, -1.4038509e-01,
```

```
-0.9874049e-02, -1.3413105e-01, -1.3801523e-01, -1.3447256e-01,
```

```
-0.5559575e-02, -0.9630562e-02, -0.7379025e-01, -0.7379025e-01,
```

```
-0.3191094e-02, -3.7412377e-02, -2.7737248e-01, -0.5484823e-02,
```

```
-0.3191094e-02, -3.7412377e-02, -2.7737248e-01, -0.5484823e-02,
```

```
-1.3732910e-02, -1.3871781e-02, -1.2979626e-01, -4.0084764e-02,
```

```
-1.3732910e-02, -1.3871781e-02, -1.2979626e-01, -4.0084764e-02,
```

```
-1.5574970e-02, -1.3709532e-02, -0.8135981e-01, -1.5722562e-01,
```

```
-1.5574970e-02, -1.3709532e-02, -0.8135981e-01, -1.5722562e-01,
```

```
-1.5573348e-02, -0.6592680e-02, -0.6991181e-01, -1.3941486e-01,
```

```
-1.5573348e-02, -0.6592680e-02, -0.6991181e-01, -1.3941486e-01,
```

```
-0.3161085e-02, -0.1539321e-02, -0.7385622e-01, -1.3941486e-01,
```

```
-0.3161085e-02, -0.1539321e-02, -0.7385622e-01, -1.3941486e-01,
```

```
-0.2161298e-02, -0.8151927e-02, -0.6690918e-01, -1.3895375e-01,
```

```
-0.2161298e-02, -0.8151927e-02, -0.6690918e-01, -1.3895375e-01,
```

```
-0.10375966e-02, -0.3370982e-02, -0.1997326e-01, -2.2740946e-02,
```

**Gambar 6.12** Hasil Nomor 1 - Clear

```
1 %%  
2 dita[ 'shine' ]
```

Dalam kode diatas digunakan untuk menampilkan hasil vektorisasi data dari kata shine. Untuk hasilnya dapat dilihat pada gambar berikut :

**Gambar 6.13** Hasil Nomor 1 - Shine

```
1 %%  
2 dita [ 'bag '
```

Dalam kode diatas digunakan untuk menampilkan hasil vektorisasi data dari kata bag. Untuk hasilnya dapat dilihat pada gambar berikut :

```
In [77]: dtsa['log']['bag']  
array([ 0.15515625,  0.15233475, -0.12402344,  0.13378996, -0.11328125,  
       0.1335667, -0.11512821,  0.14664383, -0.08659398,  0.146625,  
       -0.08605595, -0.34650745,  0.29994899, -0.04543594, -0.21895725,  
       0.18847745, -0.09585322,  0.18847745, -0.09585322,  0.18847745,  
       0.18847745, -0.09585322,  0.18847745, -0.09585322,  0.18847745,  
       0.18847745, -0.09585322,  0.18847745, -0.09585322,  0.18847745,  
       0.05541992,  0.18011384,  0.21410944, -0.21875,  0.07653689,  
       0.17076875,  0.18030825,  0.21432944, -0.21875,  0.07653689,  
       0.17076875,  0.18030825,  0.21432944, -0.21875,  0.07653689,  
       0.13624219,  0.09551758,  0.042684975, -0.18074224,  0.07324194,  
       0.02813824,  0.09551758,  0.042684975, -0.18074224,  0.07324194,  
       0.09552017,  0.07377564,  0.07377564, -0.08955201,  0.07377564,  
       0.05515234, -0.12079851,  0.0682381, -0.0578133, -0.03675,  
       0.2027135,  0.18030825,  0.21432944, -0.21875,  0.07653689,  
       0.20415105,  0.18796822,  0.20822288, -0.39354282, -0.14451235,  
       0.528125,  0.1699512,  0.05495492, -0.12534775, -0.05621691,  
       0.17526556,  0.1395728,  0.07473628, -0.004853284, -0.17262562,  
       0.29729688,  0.09551758,  0.042684975, -0.18074224,  0.07324194,  
       0.03508867,  0.042684975, -0.18074224,  0.07324194, -0.02713864,  
       0.03508867,  0.042684975, -0.18074224,  0.07324194, -0.02713864])
```

**Gambar 6.14** Hasil Nomor 1 - Bag

```
1 %%  
2 dita [ 'car '
```

Dalam kode diatas digunakan untuk menampilkan hasil vektorisasi data dari kata car. Untuk hasilnya dapat dilihat pada gambar berikut :

**Gambar 6.15** Hasil Nomor 1 - Car

```

1 %%%
2 dita [ 'wash' ]

```

Dalam kode diatas digunakan untuk menampilkan hasil vektorisasi data dari kata wash. Untuk hasilnya dapat dilihat pada gambar berikut :

```

In [7]: dita['wash']
Out[7]:
array([ 9.46044932e-05, -1.41001562e-01, -5.46875800e-02, 1.34756525e-01,
       -2.38212159e-01, 3.24219704e-01, -8.44726522e-02, 1.29882812e-01,
       1.32087500e-01, -1.17087500e-01, 1.17087500e-01, -1.32087500e-01,
       -2.79541816e-02, 2.80807812e-01, -4.72268094e-02, 1.05461759e-01,
       -1.05461759e-01, 2.80807812e-01, -4.72268094e-02, 1.05461759e-01,
       -2.79541816e-02, 2.12898625e-01, 1.74568547e-02, 2.02941895e-03,
       6.29828212e-02, 1.62193975e-01, 1.33559375e-01, 2.17283156e-02,
       -2.17283156e-02, 1.62193975e-01, 1.33559375e-01, 2.17283156e-02,
       -8.80781215e-02, 3.08859375e-02, -1.08897556e-01, -1.19645436e-01,
       1.19645436e-01, -3.08859375e-02, 2.30721205e-02, 2.49155123e-01, -6.39827344e-02,
       -1.05468750e-01, 1.62096707e-01, 3.08859375e-01, 5.63731094e-02,
       5.63731094e-02, -1.62096707e-01, -3.08859375e-01, -5.63731094e-02,
       2.84423828e-02, 2.29492118e-01, 5.11054688e-02, 4.15661562e-02,
       -4.15661562e-02, 5.11054688e-02, -2.29492118e-01, -2.84423828e-02,
       -1.18839844e-01, 6.83593757e-02, -2.52685547e-02, -1.27925888e-01,
       4.21875099e-01, 5.32229529e-02, -3.92578132e-01, 1.74886588e-01,
       1.74886588e-01, -5.32229529e-02, 4.21875099e-01, -1.74886588e-01,
       1.88398458e-01, -4.36207852e-03, -2.45117186e-01, -2.08984379e-01,
       3.49884458e-01, -5.47658200e-01, -5.28019531e-02, 2.14468379e-01,
       1.44884458e-01, -3.47658200e-01, -5.28019531e-02, 2.14468379e-01,
       1.69877334e-02, 1.69912875e-01, -1.64843737e-01, 2.16526000e-01,
       2.16526000e-01, -1.69912875e-01, 1.64843737e-01, -2.16526000e-01,
       4.32128986e-02, 1.11604313e-01, 5.37354738e-04, -7.10185162e-02,
       -5.93144531e-02, -5.44471599e-02, 5.13756879e-01, 5.62508644e-01,
       -2.26520000e-03, -5.39597812e-02, 1.13705531e-01, -5.21486944e-02,

```

Gambar 6.16 Hasil Nomor 1 - Wash

```

1 %%%
2 dita [ 'motor' ]

```

Dalam kode diatas digunakan untuk menampilkan hasil vektorisasi data dari kata motor. Untuk hasilnya dapat dilihat pada gambar berikut :

```

In [8]: dita['motor']
Out[8]:
array([ 5.37384699e-02, 1.53094616e-01, 4.51245784e-01, -1.18511608e-01,
       -2.34375000e-02, 2.77334737e-01, 1.74886588e-01, -2.08984379e-01,
       2.34375000e-02, 2.79812986e-01, 1.74886588e-01, 2.44140635e-02,
       -2.51953125e-01, 5.76171875e-01, 8.15429588e-01, 1.86757579e-01,
       -1.86757579e-01, 8.15429588e-01, 5.76171875e-01, -2.51953125e-01,
       1.56250000e-01, -4.24084686e-01, -1.32812598e-01, 2.11914062e-01,
       2.11914062e-01, -1.32812598e-01, -4.24084686e-01, -1.56250000e-01,
       3.8273475e-01, 1.33320312e-01, -1.69921875e-01, -1.01874219e-01,
       -1.01874219e-01, 1.69921875e-01, 1.33320312e-01, 3.8273475e-01,
       2.26530000e-03, 2.29515262e-01, 8.06150000e-02, -7.12380000e-02,
       1.30321000e-01, -1.30321000e-01, 8.06150000e-02, -2.26530000e-03,
       -4.46773344e-02, -3.06646925e-01, 7.42187500e-02, 5.50593750e-01,
       1.10185854e-01, -1.10185854e-01, 7.42187500e-02, 5.50593750e-01,
       1.10185854e-01, -1.10185854e-01, 7.42187500e-02, 5.50593750e-01,
       3.18858945e-02, -6.58024414e-01, 6.34795252e-02, 4.02832011e-02,
       -2.79812986e-01, 1.74886588e-01, 2.14468379e-01, -2.14468379e-01,
       -1.98398625e-01, -1.76893125e-01, 9.36897586e-01, 1.28986250e-01,
       -3.34723506e-02, 1.69812227e-01, -2.14468379e-01, -9.52148438e-02,
       -6.45884458e-02, 2.16526000e-01, 1.64843737e-01, -2.16526000e-01,
       -2.21798600e-01, 2.33594384e-01, 5.03731894e-01, -3.17898051e-01,
       1.14257812e-01, -9.71679588e-02, -2.61718750e-01,
       -8.80781215e-02, 1.14257812e-01, -9.71679588e-02, -2.61718750e-01,
       3.87555259e-01, 1.73980880e-01, 1.13851562e-01, 1.08859594e-01,
       1.08859594e-01, -1.13851562e-01, -1.73980880e-01, -3.87555259e-01,
       -2.4023475e-01, 8.34699393e-02, 4.10921377e-02, -1.91590510e-02,
       9.20312500e-02, -3.12031250e-01, 3.27144433e-02, 5.71289062e-02,
       8.20312500e-02, -3.12031250e-01, 3.27144433e-02, 5.71289062e-02,
       1.77754375e-01, -9.57803125e-01, 2.45117188e-01, 6.08470632e-02,

```

Gambar 6.17 Hasil Nomor 1 - Motor

```

1 %%%
2 dita [ 'cycle' ]

```

Dalam kode diatas digunakan untuk menampilkan hasil vektorisasi data dari kata cycle. Untuk hasilnya dapat dilihat pada gambar berikut :

```
In [81]: dita['cycle']
Out[81]:
array([ 0.84543016,  0.21679668, -0.02789961,  0.12253536, -0.30790325,
       -0.1328125 ,  0.26367188, -0.12898625, -0.125 ,  0.15330381,
       -0.1328125 ,  0.26367188, -0.12898625,  0.12253536, -0.30790325,
       0.82563477, -0.07568359, -0.0625 ,  0.84614258,  0.10956688,
       -0.13280798, -0.11669292, -0.3359375 ,  0.078125 ,  0.86447266,
       0.08727251,  0.02372852,  0.10893359,  0.02498234, -0.18644531,
       0.08727251,  0.02372852,  0.10893359,  0.02498234, -0.18644531,
       0.17771984,  0.01165773, -0.03697773,  0.13671875, -0.15468325,
       0.11425761,  0.20807781, -0.06879182, -0.07275351,  0.15090662,
       0.08872285,  0.03564451, -0.29882012,  0.09998973, -0.1448375 ,
       0.08872285,  0.03564451, -0.29882012,  0.09998973, -0.1448375 ,
       0.03173828, -0.07808078,  0.1464875 , -0.28214884, -0.03393555,
       0.09953231, -0.02038574, -0.08780962, -0.07225652, -0.04243828,
       0.09953231, -0.02038574, -0.08780962, -0.07225652, -0.04243828,
       -0.02636719, -0.03686525, -0.125 , -0.08737899,  0.14297812,
       -0.02636719, -0.03686525, -0.125 , -0.08737899,  0.14297812,
       -0.03694321,  0.14955469,  0.16894531, -0.02858645, -0.18791865, -0.12421875,
       -0.14955469,  0.14955469, -0.07421875,  0.34179869,  0.14802125,
       -0.2578125 ,  0.3671875 ,  0.06483184,  0.20783125,  0.09667969,
       -0.2578125 ,  0.3671875 ,  0.06483184,  0.20783125,  0.09667969,
       0.06689453, -0.1796875 ,  0.02783283,  0.15625 ,  0.02020367,
       0.0324987 , -0.13479562,  0.15927344,  0.11152811, -0.04099868,
```

**Gambar 6.18** Hasil Nomor 1 - Cycle

```
1 %%%
2 dita.similarity('wash', 'clear')
```

Dalam kode diatas digunakan untuk persentase dari kata wash dan clear, persentase nya adalah 9 persen, hasil tersebut tidak terlalu baik. Untuk hasilnya dapat dilihat pada gambar berikut :

```
In [82]: dita.similarity('wash', 'clear')
Out[82]: 0.09019176
```

**Gambar 6.19** Hasil Nomor 1 - Wash dan Clear

```
1 %%%
2 dita.similarity('bag', 'love')
```

Dalam kode diatas digunakan untuk persentase dari kata bag dan love, persentase nya adalah 7 persen, hasil tersebut tidak terlalu baik. Untuk hasilnya dapat dilihat pada gambar berikut :

```
In [83]: dita.similarity('bag', 'love')
Out[83]: 0.07536096
```

**Gambar 6.20** Hasil Nomor 1 - Bag dan Love

```
1 %%%
2 dita.similarity('motor', 'car')
```

Dalam kode diatas digunakan untuk persentase dari kata motor dan car, persentase nya adalah 48 persen, hasil tersebut cukup baik karena mesin dapat membedakan antara motor dan car. Untuk hasilnya dapat dilihat pada gambar berikut :

```
In [84]: dita.similarity('motor', 'car')
Out[84]: 0.4810172
```

**Gambar 6.21** Hasil Nomor 1 - Motor dan Car

```

1 #%%
2 dita.similarity('sick', 'faith')

```

Dalam kode diatas digunakan untuk persentase dari kata sick dan faith, persentase nya adalah 12 persen, hasil tersebut cukup baik. Untuk hasilnya dapat dilihat pada gambar berikut :

```

In [85]: dita.similarity('sick', 'faith')
Out[85]: 0.123073205

```

**Gambar 6.22** Hasil Nomor 1 - Sick dan Faith

```

1 #%%
2 dita.similarity('cycle', 'shine')

```

Dalam kode diatas digunakan untuk persentase dari kata cycle dan shine, persentase nya adalah 6 persen, hasil tersebut kurang baik. Untuk hasilnya dapat dilihat pada gambar berikut :

```

In [86]: dita.similarity('cycle', 'shine')
Out[86]: 0.061617922

```

**Gambar 6.23** Hasil Nomor 1 - Cycle dan Shine

## 2. Nomor 2

```

1 import re
2 test_string = "Aulyardha Anindita, Kuliah di Politeknik
   Pos Indonesia"
3 print ("Biodata : " + test_string)
4 res = re.findall(r'\w+', test_string)
5 print ("List kata nya adalah : " + str(res))

```

Kode diatas digunakan untuk membuat string dengan menggunakan library re, didalamnya dapat digunakan test string sebagai string dan membuat print untuk menambahkan kalimat sebelum test string. Untuk hasilnya dapat dilihat pada gambar berikut :

```

In [87]: defn't re
...: test_string = "Aulyardha Anindita, Kuliah di Politeknik Pos Indonesia"
...: i print ("Biodata : " + test_string)
...: i print ("List kata nya adalah : " + str(res))
...: i print ("List kata nya adalah : [Aulyardha, 'Anindita', 'Kuliah', 'di', 'Politeknik',
...: i print ("List kata nya adalah : [Aulyardha, 'Anindita', 'Kuliah', 'di', 'Politeknik',
...: 'Pos', 'Indonesia']")

```

**Gambar 6.24** Hasil Nomor 2 - 1

```

1 import random
2 sent_matrix = [ ['Aulyardha', 'Anindita'],
3                 ['Kuliah', 'di'],
4                 ['Politeknik', 'Pos'],
5                 ['Indonesia']
6             ]
7 result = ""
8 for elem in sent_matrix:
9     result += random.choice(elem) + " "
10 print (result)

```

Kode diatas digunakan untuk membuat string dengan memakai library random, yaitu dengan memakai sent matrix untuk membuat string dan untuk result sebagai print random yang akan diacak. Hasilnya dapat dilihat pada gambar berikut :

```

In [89]: import random
...: sent_matrix = [['Aulyardha', 'Anindita'],
...:                ['Kuliah', 'di'],
...:                ['Politeknik', 'Pos'],
...:                ['Indonesia']]
...: result = ""
...: for elem in sent_matrix:
...:     result += random.choice(elem) + " "
...: print (result)
Anindita Kuliah Politeknik Indonesia

```

**Gambar 6.25** Hasil Nomor 2 - 2

### 3. Nomor 3

```

1 from gensim.models.doc2vec import Doc2Vec, TaggedDocument
2
3 ## Contoh dokumen
4 doc = ["saya suka membaca",
5        "saya suka menulis",
6        "saya suka menonton",
7        "saya sering tersenyum",
8        "saya sering jalan-jalan",
9        "saya sering makan"]
10 tokenized_doc = ['sering']
11 tokenized_doc
12 print(doc)
13
14 #%%
15 tagged_data = [TaggedDocument(d, [i]) for i, d in enumerate(
16     tokenized_doc)]
16 tagged_data
17 # mengtrain doc2vec model
18 model = Doc2Vec(tagged_data, vector_size=20, window=2,
19                  min_count=1, workers=4, epochs = 100)
20 # menyimpan trained doc2vec model
21 model.save("test_doc2vec.model")
22 # meload doc2vec model
23 model= Doc2Vec.load("test_doc2vec.model")
23 #menampilkan model
24 model.wv.vocab

```

Kode diatas digunakan untuk mengimport library gensim. library gensim biasanya digunakan untuk pemodelan topik tanpa pengawasan dan pemrosesan bahasa alami, atau bisa juga disebut dengan unsupervised. fungsi dari doc2vec adalah untuk membandingkan bobot data yang terdapat pada dokumen lainnya, apakah kata-kata didalamnya sama atau tida. Sedangkan untuk tagged document digunakan untuk memasukkan kata-kata pada setiap dokumen untuk direktorisasi, dam model untuk membuat model dan save file model. Untuk hasilnya dapat dilihat pada gambar berikut :

```
In [92]: from gensim.models.doc2vec import Doc2Vec, TaggedDocument
...: # Contoh dokumen
...: doc = ["saya suka makan",
...: "saya suka menonton",
...: "saya suka menonton",
...: "saya suka menonton",
...: "saya sering jalan-jalan",
...: "saya sering makan"]
...: tokenized_doc = [word_tokenize(doc)]
...: tokenized_doc
...: 
['saya suka makan', 'saya suka menonton', 'saya suka menonton',
 'saya suka menonton', 'saya sering jalan-jalan', 'saya sering makan']

['saya suka makan', 'saya suka menonton', 'saya suka menonton',
 'saya suka menonton', 'saya sering jalan-jalan', 'saya sering makan']
```

**Gambar 6.26** Hasil Nomor 3-1

```
Out[93]:
{'s': <gensim.models.keyedvectors.Vocab at 0x2426c66312b>,
 'e': <gensim.models.keyedvectors.Vocab at 0x2426c6631d0>,
 'r': <gensim.models.keyedvectors.Vocab at 0x2426c66320b>,
 't': <gensim.models.keyedvectors.Vocab at 0x2426c663210>,
 'n': <gensim.models.keyedvectors.Vocab at 0x2426c66327b>,
 'g': <gensim.models.keyedvectors.Vocab at 0x2426c6632b0>}
```

**Gambar 6.27** Hasil Nomor 3-2

#### 4. Nomor 4

```
1 import re
2 import os
3 unsup_sentences = []
4
5 for dirname in ["train/pos", "train/neg", "train/unsup", "test/pos",
6                 "test/neg"]:
    for fname in sorted(os.listdir("aclImdb/" + dirname)):
        if fname[-4:] == '.txt':
            with open("aclImdb/" + dirname + "/" + fname,
4 encoding='UTF-8') as f:
                sent = f.read()
                words = (sent)
                unsup_sentences.append(TaggedDocument(words,
4 [dirname + "/" + fname]))
```

Dalam kode diatas saya menggunakan dataset dari aclImdb. untuk menambahkan data training kita bisa mengimport library os agar os sendiri dapat melakukan interaksi dengan python. kemudian saya membuat variable unsup sentences. Lalu pilih direktori tempat data kita disimpan. Selanjutnya kita bisa menyortir data yang terdapat pada folder aclImdb dan membaca file tersebut dengan ekstensi .txt. Hasilnya dapat dilihat

pada gambar berikut:

**Gambar 6.28** Hasil Nomor 4

## 5. Nomor 5

```
1 mute = (unsup_sentences) #mengacak data
2
3 model.delete_temporary_training_data(keep_inference=True) # membersihkan data
```

Dalam kode diatas pada bagian mengacak data digunakan untuk mengacak data agar ketika data dirunning mampu berjalan dengan baik dan hasil persentase nya juga lebih baik. Dan untuk pembersihan data digunakan untuk memberikan ruang bagi ram laptop setelah melakukan running data sebanyak 3 juta lebih. Untuk hasilnya dapat dilihat pada gambar berikut :

**Gambar 6.29** Hasil Nomor 5

## 6. Nomor 6

```
1 model.save('dita.d2v') #menyimpan data
2
3 model.delete_temporary_training_data(keep_inference=True) # menghapus temporary data
```

Dalam kode diatas terdapat save data yang digunakan untuk menyimpan file hasil dari proses training sebelumnya. Model tersebut dilakukan penyimpanan untuk memberikan keringanan pada ram agar ketika akan melakukan pelatihan lagi, model tersebut tinggal di load saja tanpa harus melakukan pelatihan dari awal sehingga bisa menghemat waktu. Sedangkan pada delete temporart digunakan untuk menghapus data latihan sebelumnya yang sudah dilakukan dan disimpan yang memiliki tujuan untuk memberikan keringanan pada ram. Untuk hasilnya dapat dilihat pada gambar berikut :

```
In [97]: model.save('dita.d2v') #menyimpan data
...
...
...: model.delete_temporary_training_data(keep_inference=True) #menghapus
temporary data
2020-04-05 14:47:29,345 : INFO : saving Doc2Vec object under dita.d2v, separately
None
2020-04-05 14:47:29,358 : INFO : saved dita.d2v
```

**Gambar 6.30** Hasil Nomor 6

## 7. Nomor 7

```
1  
2 model.infer_vector(extract_words("Aulyardha Anindita"))
```

Dalam kode diatas terdapat infer vector yang digunakan untuk membandingkan kata yang tercantum dengan vektor yang mana pada dokumen yang sudah diload pada step sebelumnya. selain itu, infer vector juga digunakan untuk menghitung atau mengkalkulasikan vektor dari kata yang dicantumkan dari model yang telah dibuat. Hasilnya dapat dilihat pada gambar dibawah ini :

```
In [100]: model.infer_vector(extract_words("Aulyardha Anindita"))
Out[100]:
array([-0.01111822, -0.018649 , -0.00204373, 0.00371147, 0.00251893,
       0.02167705, 0.0018844 , 0.01793301, 0.01993932, -0.00896326,
       0.01286298, 0.02230248, 0.00766548, 0.01542173, 0.00844643,
      -0.0041517 , -0.0070347 , -0.02150514, -0.01086319, 0.00711636]
dtypes: float16(12)
```

**Gambar 6.31** Hasil Nomor 7

8. Nomor 8

```
1 from sklearn.metrics.pairwise import cosine_similarity
2 cosine_similarity(
3     [model.infer_vector(extract_words("Dita pulang
4 kampung"))],
5     [model.infer_vector(extract_words("Karena kampus
diliburkan gara-gara corona"))])
```

Pada kode diatas terdapat cosine similary yang biasanya digunakan untuk membandingkan vektorisasi data diantara dua kata yang diinputkan, jika hasil persentase dari keuda data tersebut lebih dari 50 persen, hal itu memungkinkan kata tersebut tidak terdapat dalam satu file. hasilnya dapat didapatkan pada code tersebut hanya 0,8 persen hal itu dikarenakan kata pertama dan kedua tidak memiliki kesamaan vektorisasi dan tidak terdapat pada salah satu dokumen. untuk hasilnya dapat dilihat pada gambar berikut :

```
In [101]: from sklearn.metrics.pairwise import cosine_similarity
..... cosine_similarity(
.....     .... [model.infer_vector(extract_words("Dita pulang kampung"))],
.....     .... [model.infer_vector(extract_words("Karena kampus diliburkan gara
gara corona"))])
Out[101]: array([-0.16165320]), dtype=float32)
```

## 9. Nomor 9

```

1 from sklearn import datasets, linear_model
2 from sklearn.model_selection import cross_val_score
3 aulyta = datasets.load_diabetes()
4 X = aulyta.data[:150]
5 y = aulyta.target[:150]
6 lasso = linear_model.Lasso()
7 print(cross_val_score(lasso, X, y, cv=3))

```

Pada kode diatas digunakan untuk melakukan perhitungan persentase dengan menggunakan cross validation dengan metode kneighborsClassifier. untuk hasilnya dapat dilihat pada gambar berikut :

```

In [102]: from sklearn import datasets, linear_model
...: from sklearn.model_selection import cross_val_score
...: aulyta = datasets.load_diabetes()
...: X = aulyta.data[:150]
...: y = aulyta.target[:150]
...: lasso = linear_model.Lasso()
...: print(cross_val_score(lasso, X, y, cv=3))
[0.35150754 0.08822311 0.03531764]

```

**Gambar 6.33** Hasil Nomor 9

### 6.1.3 Penanganan Error

#### 1. ScreenShoot Error

```
ModuleNotFoundError: No module named 'gensim'
```

**Gambar 6.34** Module Not Found Error

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

**Gambar 6.35** Name Error

#### 2. Tuliskan Kode Error dan Jenis Error

- Module Not Found Error
- Name Error

#### 3. Cara Penanganan Error

- File Not Found Error

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

- Name Error
- Error terjadi karena sentvecs tidak terdefinisi. Untuk mengatasinya yaitu dengan membuat variabel baru dengan nama yang sama.

### 6.1.4 Bukti Tidak Plagiat



**Gambar 6.36**      Bukti Plagiarisme

### 6.1.5 Link Youtube

<https://youtu.be/2YUr0IwEu-w>

# BAB 7

---

# CHAPTER 6

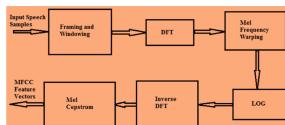
---

## 7.1 1174073 - Ainul Filiani

### 7.1.1 Soal Teori

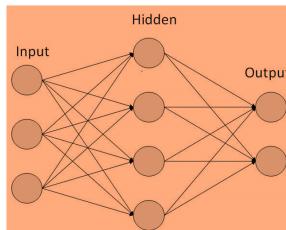
1. Jelaskan kenapa file suara harus di lakukan MFCC. dilengkapi dengan ilustrasi atau gambar.

Karena MFCC adalah koefisien yang mewakili audio. Ekstraksi fitur dalam proses ini ditandai dengan konversi data suara menjadi gambar spektrum gelombang. File audio dilakukan oleh MFCC sehingga objek suara dapat dikonversi menjadi matriks. Suara akan menjadi vektor yang akan diproses sebagai output. Selain mempermudah mesin dalam bahasa ini karena mesin tidak dapat membaca teks, maka MFCC perlu mengubah suara menjadi vektor. Untuk ilustrasi, lihat gambar berikut:

**Gambar 7.1** Teori 1

2. Jelaskan konsep dasar neural network.dilengkapi dengan ilustrasi atau gambar.

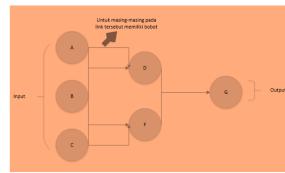
Konsep sederhana dari neural network atau jaringan saraf sederhana dengan proses pembelajaran pada anak-anak dengan memetakan pola-pola baru yang diperoleh dari input untuk membuat pola-pola baru pada output. Contoh sederhana ini menganalogikan kinerja otak manusia. Jaringan saraf itu sendiri terdiri dari unit pemrosesan yang disebut neuron yang berisi fungsi adder dan aktivasi. Fungsi aktivasi itu sendiri untuk publikasi diberikan oleh neuron. Jaringan saraf yang mendukung pemikiran sistem atau aplikasi yang melibatkan otak manusia, baik untuk mendukung berbagai elemen sinyal yang diterima, memeriksa kesalahan, dan juga memproses secara parallel. Karakteristik neural network atau jaringan saraf dilihat dari pola hubungan antar neuron, metode penentuan bobot setiap koneksi, dan fungsi aktivasi mereka.. Untuk ilustrasi, lihat gambar berikut:

**Gambar 7.2** Teori 2

3. Jelaskan konsep pembobotan dalam neural network.dilengkapi dengan ilustrasi atau gambar.

Pembobotan di dalam neural network juga akan menentukan penanda konektivitas. Dalam proses neural network mulai dari input yang diterima oleh neuron bersama dengan nilai bobot masing-masing input. Setelah memasuki neuron, nilai input akan ditambahkan oleh fungsi penerima. Hasil penambahan ini akan diproses oleh masing-masing fungsi neuron, hasil penambahan ini akan dibandingkan dengan nilai ambang tertentu. Jika nilai jumlah ini melebihi nilai ambang batas, aktivasi neuron akan dibatalkan, tetapi sebaliknya jika jumlah hasil di bawah nilai ambang

batas, neuron akan diaktifkan. Setelah neuron aktif maka akan mengirimkan nilai output melalui bobot outputnya ke semua neuron yang terkait. Untuk ilustrasi, lihat gambar berikut:



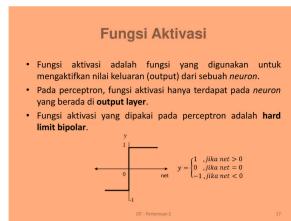
**Gambar 7.3** Teori 3

- Jelaskan konsep fungsi aktifasi dalam neural network. dilengkapi dengan ilustrasi atau gambar.

Fungsi aktivasi dalam neural network ialah merupakan suatu operasi matematik yang dikenakan pada sinyal output. Fungsi ini digunakan untuk mengaktifkan atau menonaktifkan neuron. Fungsi aktivasi ini terbagi setidaknya menjadi 6, adapun sebagai berikut:

- Fungsi Undak Biner Hard Limit, fungsi ini biasanya digunakan oleh jaringan lapiran tunggal untuk mengkonversi nilai input dari suatu variabel yang bernilai kontinu ke suatu nilai output biner 0 atau 1.
- Fungsi Undak Biner Threshold, fungsi ini menggunakan nilai ambang sebagai batasnya.
- Fungsi Bipolar Symetric Hard Limit, fungsi ini memiliki output bernilai 1, 0 atau -1.
- Fungsi Bipolar dengan Threshold, fungsi ini mempunyai output yang bernilai 1, 0 atau -1 untuk batas nilai ambang tertentu.
- Fungsi Linear atau Identitas.

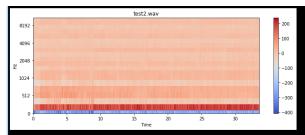
- Namun, untuk ilustrasi lihat gambar berikut:



**Gambar 7.4** Teori 4

- Jelaskan cara membaca hasil plot dari MFCC,dilengkapi dengan ilustrasi atau gambar

Perhatikan gambar dibawah: Gambar menjelaskan bahwa pada waktu ke-5 kekuatan atau layak yang dikeluarkan pada nada ini paling keras pada 20 Hz, selain itu pada 40 -120 Hz daya atau nilai dikeluarkan pada musik yang telah diplot. Demikian juga, warna tersebut adalah kekuatan atau nilai tertinggi dibandingkan dengan warna canggih. Yang merah terdengar di bawah pendengaran manusia, sehingga tidak bisa didengar secara langsung.



**Gambar 7.5** Teori 5

6. Jelaskan apa itu one-hot encoding,dilengkapi dengan ilustrasi kode dan atau gambar.

Sederhananya, one-hot encoding ini adalah untuk mengubah hasil data vektorisasi menjadi angka biner 0 dan 1 dan membuat informasi tentang atribut yang diberi label. Untuk ilustrasi, lihat gambar berikut:



**Gambar 7.6** Teori 6

7. Jelaskan apa fungsi dari np.unique dan to\_categorical dalam kode program,dilengkapi dengan ilustrasi atau gambar.

Fungsi np.unique adalah untuk menemukan berbagai elemen atau array unik, dan dapat mengembalikan elemen unik dari array yang diurutkan. Sebagai ilustrasi, cukup bisa dilihat pada gambar di bawah ini. Gambar tersebut menjelaskan bahwa unik itu sendiri akan mengambil data yang berbeda dari variabel a dalam fungsi array.

```
>>> import numpy as np
>>> x = np.array([[1,2],[1,1],[2,3],[4,5]])
>>> np.unique(x)
array([1, 2, 3, 4, 5])
>>>
```

**Gambar 7.7** Teori 7

Fungsi dari to\_categorical ialah untuk mengubah suatu vektor yang berupa integer menjadi matrix dengan kelas biner. Untuk ilustrasinya bisa dilihat pada gambar ??

```

Categories (5, object): [a, f, h, i, n]
>>> import pandas as pd
>>> df = pd.Categorical(['c', 'i', 'e', 'c', 'o', 'p', 's', 's'])
>>> print(df)
[0: 'c', 1: 'i', 2: 'e', 3: 'c', 4: 'o', 5: 'p', 6: 's', 7: 's']
Categories (7, object): [a, c, e, i, o, p, s]
>>>

```

Gambar 7.8 Teori 7

8. Jelaskan apa fungsi dari Sequential dalam kode program,dilengkapi dengan ilustrasi atau gambar.

Fungsi dari Sequential sebagai salah satu jenis model yang digunakan dalam perhitungan. Sequential ini membangun tumpukan linear yang berurutan. Untuk ilustrasi gambar sebagai berikut :

```

In [20]: model = Sequential([
    ...     Dense(16, input_dim=shape(train_input)[1]),
    ...     Activation('relu'),
    ...     Dense(16),
    ...     Activation('softmax'),
    ...
])

```

Gambar 7.9 Teori 8

### 7.1.2 Praktek Program

- Soal 1

```

1 # In [1]: Soal Nomor 1
2
3 import librosa
4 import librosa.feature
5 import librosa.display
6 import glob
7 import numpy as np
8 import matplotlib.pyplot as plt
9 from keras.layers import Dense, Activation
10 from keras.models import Sequential
11 from keras.utils.np_utils import to_categorical
12
13 def display_mfcc(song):
14     y, _ = librosa.load(song)
15     mfcc = librosa.feature.mfcc(y)
16
17     plt.figure(figsize=(10, 4))
18     librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
19     plt.colorbar()
20     plt.title(song)
21     plt.tight_layout()
22     plt.show()

```

Kode di atas menjelaskan isi data GTZAN. Ini adalah kumpulan data yang berisi 10 genre lagu dengan masing-masing genre memiliki 100 lagu yang akan kami lakukan proses MFCC dan juga freesound yang hanya berisi konten lagu, jika GTZAN memiliki beberapa genre jika freesound

hanya untuk 1 lagu dan disini kita membuat fungsi untuk membaca file audio dan outputnya sebagai plot, hasilnya adalah sebagai berikut:

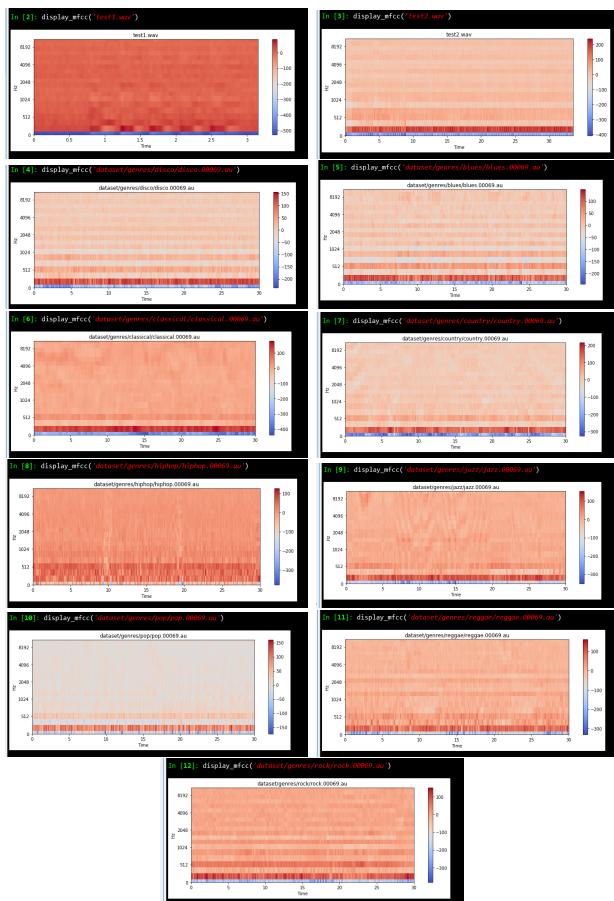
```
[1]: import librosa
import librosa.feature
import librosa.display
import numpy as np
from IPython.display import display, plt
from keras.layers import Dense, Activation
from keras.models import Sequential
from keras.utils import to_categorical
from keras.utils import np_utils
def display_mfcc(song):
    y, sr = librosa.load(song)
    mfcc = librosa.feature.mfcc(y)
    plt.figure(figsize=(10, 4))
    librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
    plt.title(song)
    plt.colorbar()
    plt.show()
```

Gambar 7.10 Hasil Soal 1.

## 2. Soal 2

```
1 # In [2]: Soal Nomor 2
2 display_mfcc('test1.wav')
3 # In [2]: Soal Nomor 2
4 display_mfcc('test2.wav')
5 # In [2]: Soal Nomor 2
6 display_mfcc('dataset/genres/disco/disco.00069.au')
7 # In [2]: Soal Nomor 2
8 display_mfcc('dataset/genres/blues/blues.00069.au')
9 # In [2]: Soal Nomor 2
10 display_mfcc('dataset/genres/classical/classical.00069.au')
11 # In [2]: Soal Nomor 2
12 display_mfcc('dataset/genres/country/country.00069.au')
13 # In [2]: Soal Nomor 2
14 display_mfcc('dataset/genres/hiphop/hiphop.00069.au')
15 # In [2]: Soal Nomor 2
16 display_mfcc('dataset/genres/jazz/jazz.00069.au')
17 # In [2]: Soal Nomor 2
18 display_mfcc('dataset/genres/pop/pop.00069.au')
19 # In [2]: Soal Nomor 2
20 display_mfcc('dataset/genres/reggae/reggae.00069.au')
21 # In [2]: Soal Nomor 2
22 display_mfcc('dataset/genres/rock/rock.00069.au')
```

Kode di atas akan menampilkan hasil dari proses mfcc yang sudah dibuat fungsi pada soal 1, yaitu display\_mfcc() dan akan menampilkan plot dari pembacaan file audio. Berikut adalah hasil setelah saya lakukan running dan pembacaan file audio :



Gambar 7.11 Hasil Soal 2.

## 3. Soal 3

```

1 # In [3]: Soal Nomor 3
2
3 def extract_features_song(f):
4     y, _ = librosa.load(f)
5
6     # get Mel-frequency cepstral coefficients
7     mfcc = librosa.feature.mfcc(y)
8     # normalize values between -1,1 (divide by max)
9     mfcc /= np.amax(np.absolute(mfcc))
10
11    return np.ndarray.flatten(mfcc)[:25000]

```

Baris pertama itu untuk membuat fungsi `extract_features_song(f)`. Pada baris kedua itu akan me-load data inputan dengan menggunakan `librosa`.

Lalu selanjutnya untuk membuat sebuah fitur untuk mfcc dari y atau parameter inputan. Lalu akan me-return menjadi array dan akan mengambil 25000 data saja dari hasil vektorisasi dalam 1 lagu. Hasilnya adalah sebagai berikut :

```

In [13]: def extract_features_song(f):
    y, _ = librosa.load(f)
    ...
    # get Mel-frequency cepstral coefficients
    ...
    mfcc = librosa.feature.mfcc(y)
    ...
    mfcc = np.log10(mfcc) # take log (divide by max)
    mfcc /= np.max(np.abs(mfcc))
    ...
    return np.ndarray.flatten(mfcc)[:1000]

```

Gambar 7.12 Hasil Soal 3.

#### 4. Soal 4

```

1 # In [4]: Soal Nomor 4
2
3 def generate_features_and_labels():
4     all_features = []
5     all_labels = []
6
7     genres = [ 'blues' , 'classical' , 'country' , 'disco' , 'hiphop' ,
8               'jazz' , 'metal' , 'pop' , 'reggae' , 'rock' ]
9     for genre in genres:
10         sound_files = glob.glob('dataset/genres/' + genre + '/*.au')
11         print('Processing %d songs in %s genre...' % (len(
12             sound_files), genre))
13         for f in sound_files:
14             features = extract_features_song(f)
15             all_features.append(features)
16             all_labels.append(genre)
17
18     # convert labels to one-hot encoding cth blues :
19     # 1000000000 classic 0100000000
20     label_uniq_ids, label_row_ids = np.unique(all_labels,
21         return_inverse=True) #ke integer
22     label_row_ids = label_row_ids.astype(np.int32, copy=False)
23     onehot_labels = to_categorical(label_row_ids, len(
24         label_uniq_ids)) #ke one hot
25     return np.stack(all_features), onehot_labels

```

Kode di atas dapat digunakan untuk melakukan fungsi yang sebelumnya telah kita lakukan. Kemudian di bagian genre yang disesuaikan dengan dataset nama folder. Untuk baris berikutnya akan mengulang genre folder dengan ekstensi .au. Maka itu akan memanggil fungsi ekstrak lagu. Setiap file dalam folder itu akan diekstraksi menjadi vektor dan akan ditambahkan ke fitur. Dan fungsi yang ditambahkan adalah untuk menumpuk file yang telah di-vektor-kan. Hasil kode tidak menampilkan output. Hasilnya adalah sebagai berikut :

```
[4]: [In] 100%|██████████| 100/100 [0m0s]
    all_features = []
    all_labels = []

    genres = ['blues', 'classical', 'country', 'disco', 'HipHop', 'jazz', 'metal',
              'pop', 'rock']

    for genre in genres:
        for file in glob.glob(os.path.join('~/data/musdb18', 'genres', genre, '*')):
            print(f'Processing {file} in the genre: {genre}')
            if file in (os.path.join(file, 'onehot'), genres):
                continue
            for song in file:
                features = extract_features(song)
                all_features.append(features)
                all_labels.append(label)

    print(f'Number of tracks: {len(all_labels)}')
    print(f'Number of labels: {len(label)}')

    label_matrix, label_row_ids = np.unique(label, return_inverse=True)
    label_matrix, label_row_ids = np.unique(label, return_inverse=True, copy=False)

    onehot = OneHotEncoder()
    onehot.fit(label_matrix.reshape(-1, 1))

    label_matrix_ids = np.array(label_matrix).reshape(-1, 1)
    onehot_labels = onehot.transform(label_matrix_ids)

    print(f'Number of unique labels: {len(label_matrix)}')
    print(f'Number of unique onehot labels: {len(onehot_labels)}')
    print(f'Number of unique row labels: {len(label_row_ids)}')
    print(f'Number of unique column labels: {len(label_matrix_ids)}')
```

### Gambar 7.13 Hasil Soal 4

### 5. Soal 5

```
1 # In [5]: Soal Nomor 5
2
3 features, labels = generate_features_and_labels()
4 # In [5]: Soal Nomor 5
5 print(np.shape(features))
6 print(np.shape(labels))
```

Kode diatas berfungsi untuk melakukan load variabel features dan labels. Mengapa memakan waktu yang lama ? Karena mesin akan melakukan vektorisasi terhadap semua file yang berada pada setiap foldernya, di sini terdapat 10 folder dengan masing-masing folder terdiri atas 100 buah lagu, setiap lagu tersebut akan dilakukan vektorisasi atau ekstraksi data menggunakan mfcc. Oleh karena itu, proses cukup memakan waktu. Hasilnya adalah sebagai berikut :

```
In [15]: features, labels = generate_features_and_labels()
Processing 100 songs in blues genre...
Processing 100 songs in classical genre...
Processing 100 songs in country genre...
Processing 100 songs in disco genre...
Processing 100 songs in hiphop genre...
processing 100 songs in jazz genre...
processing 100 songs in metal genre...
processing 100 songs in rock genre...
Processing 100 songs in pop genre...
Processing 100 songs in reggae genre...
Processing 100 songs in country genre...
In [16]: print(np.shape(features))
...: print(np.shape(labels))
(1000, 25000)
(1000, 10)
```

### Gambar 7.14 Hasil Soal 5

## 6. Soal 6

```
1 # In [6]: Soal Nomor 6
2 training_split = 0.8
3 # In [6]: Soal Nomor 6
4 alldata = np.column_stack(( features , labels ))
5 # In [6]: Soal Nomor 6
6 np.random.shuffle(alldata)
7 splitidx = int(len(alldata) * training_split)
8 train , test = alldata [:splitidx , : ] , alldata [splitidx : , :]
9 # In [6]: Soal Nomor 6
10 print(np.shape(train))
11 print(np.shape(test))
```

```

12 # In [6]: Soal Nomor 6
13 train_input = train[:, :-10]
14 train_labels = train[:, -10:]
15 # In [6]: Soal Nomor 6
16 test_input = test[:, :-10]
17 test_labels = test[:, -10:]
18 # In [6]: Soal Nomor 6
19 print(np.shape(train_input))
20 print(np.shape(train_labels))

```

Kode diatas berfungsi untuk melakukan training split 80%. Karena supaya mesin dapat terus belajar tentang data baru, jadi ketika prediksi dibuat tentang data yang terlatih itu bisa mendapatkan persentase yang cukup bagus. Hasilnya adalah sebagai berikut :

<b>In [17]:</b> training_split = 0.8	<b>In [18]:</b> alldata = np.column_stack((features, labels))
<b>In [19]:</b> np.random.shuffle(alldata)       ....       splitidx = int(len(alldata) * training_split)       train, test = alldata[:splitidx], alldata[splitidx:, :]	<b>In [20]:</b> print(np.shape(train))       ....       print(np.shape(test))       (800, 25010)       (200, 25010)
<b>In [21]:</b> train_input = train[:, :-10]       ....       train_labels = train[:, -10:]	<b>In [22]:</b> test_input = test[:, :-10]       ....       test_labels = test[:, -10:]
<b>In [23]:</b> print(np.shape(train_input))       ....       print(np.shape(train_labels))       (800, 25000)       (800, 10)	<b>In [23]:</b> print(np.shape(test_input))       ....       print(np.shape(test_labels))       (200, 25000)       (200, 10)

Gambar 7.15 Hasil Soal 6.

## 7. Soal 7

```

1 # In [7]: Soal Nomor 7
2 model = Sequential([
3     Dense(100, input_dim=np.shape(train_input)[1]),
4     Activation('relu'),
5     Dense(10),
6     Activation('softmax'),
7 ])

```

fungsi Sequential() ialah sebuah model untuk menentukan izin pada setiap neuron, di sini adalah 100 dense yang merupakan 100 neuron pertama dari data pelatihan. Fungsi dari relay itu sendiri adalah untuk mengaktifkan neuron atau input yang memiliki nilai maksimum. Sedangkan untuk dense 10 itu adalah output dari hasil neuron yang telah berhasil diaktifkan, untuk dense 10 diaktifkan menggunakan softmax. Hasilnya adalah sebagai berikut :

```
In [24]: model = Sequential([
    ... Dense(100, input_dim=inp.shape[1]),
    ... Activation('relu'),
    ... Dense(10),
    ... Activation('softmax'),
    ...
])
```

Gambar 7.16 Hasil Soal 7.

## 8. Soal 8

```
# In [8]: Soal Nomor 8
1 model.compile(optimizer='adam',
2                 loss='categorical_crossentropy',
3                 metrics=['accuracy'])
4
5 print(model.summary())
```

Model Compile di perjelas dengan gambar dibawah, Hasil output pada kode tersebut seperti gambar menjelaskan bahwa dense pertama itu memiliki 100 neurons dengan parameter sekitar 2 juta lebih dengan aktivasasi 100, jadi untuk setiap neurons memiliki masing-masing 1 aktivasi. Sama halnya seperti dense 2 memiliki jumlah neurons sebanyak 10 dengan parameter 1010 dan jumlah aktivasinya 10 untuk setiap neurons tersebut dan total parameternya sekitar 2.5 juta data yang akan dilatih pada mesin tersebut.

```
In [8]: model.compile(optimizer='adam',
... loss='categorical_crossentropy',
... metrics=['accuracy'])
...
Model: "sequential_1"
Layer (Type)          Output Shape       Params #
dense_1 (Dense)      (None, 100)        2500100
activation_1 (Activation) (None, 100)        0
dense_2 (Dense)      (None, 10)         1010
activation_2 (Activation) (None, 10)        0
...
total params: 2,501,110
Trainable params: 2,501,110
Non-trainable params: 0
None
```

Gambar 7.17 Hasil Soal 8.

## 9. Soal 9

```
# In [9]: Soal Nomor 9
1 model.fit(train_input, train_labels, epochs=10, batch_size
2           =32,
3           validation_split=0.2)
```

Kode tersebut berfungsi untuk melatih mesin dengan data training input dan training label. Epochs ini merupakan iterasi atau pengulangan berapa kali data tersebut akan dilakukan. Batch\_size ini adalah jumlah file yang akan dilakukan pelatihan pada setiap 1 kali pengulangan. Sedangkan validation\_split itu untuk menentukan presentase dari cross validation atau k-fold sebanyak 20% dari masing-masing data pengulangan, hasilnya adalah sebagai berikut :

**Gambar 7.18** Hasil Soal 9.

### 10. Soal 10

```
1 # In [10]: Soal Nomor 10
2 loss , acc = model.evaluate(test_input , test_labels ,
3                             batch_size=32)
4 # In [10]: Soal Nomor 10
5 print("Done!")
6 print("Loss: %.4f, accuracy: %.4f" % (loss , acc))
```

Fungsi evaluate atau evaluasi ini ialah untuk menguji data pengujian setiap file. Di sini ada prediksi yang hilang, artinya mesin memprediksi data, sedangkan untuk keseluruhan perjanjian sekitar 55%, hasilnya adalah sebagai berikut :

```
[in [27]:] loss, acc = model.evaluate(test_input, test_labels, batch_size=1)
200/200 [=====] - 0s 200ms/step
[in [28]:] print("Done!")
Done!
Loss: 1.4637, accuracy: 0.5908
```

**Gambar 7.19** Hasil Soal 10.

### 11. Soal 11

```
1 # In[11]: Soal Nomor 11  
2 model.predict(test_input[:1])
```

Fungsi Predict ialah untuk menghasilkan suatu nilai yang sudah di prediksi dari data training sebelumnya. Gambar dibawah ini menjelaskan file yang di jalankan tersebut termasuk ke dalam genre apa, hasilnya bisa dilihat pada gambar tersebut presentase yang paling besar yakni genre rock. Maka lagu tersebut termasuk ke dalam genre rock dengan perbandingan presentase hasil prediksi.

```
In [29]: model.predict(test_input[:1])
Out[29]:
array([[6.8733118e-02, 8.2320191e-02, 6.34665556e-02, 1.3786874e-02,
       2.5682386e-02, 6.7000211e-01, 2.779774e-04, 2.0203577e-02,
       2.510969e-02, 3.0326517e-02]], dtype=float32)
```

Gambar 7.20 Hasil Soal 11

### 7.1.3 Penanganan Error

#### 1. ScreenShoot Error

```
In [1]: import librosa
... import librosa.feature
... import librosa.display
... import matplotlib.pyplot as plt
Traceback (most recent call last):
File <ipython-input-1-cc3809031994>, line 1, in <module>
    import librosa
ModuleNotFoundError: No module named 'librosa'
```

**Gambar 7.21** ModuleNotFoundError



**Gambar 7.22** An error occurred while starting the kernel

#### 2. Cara Penanganan Error

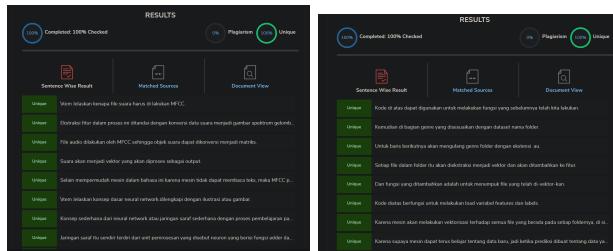
- **ModuleNotFoundError**

Error terdapat pada library yang tidak terbaca, karena library librosa belum di install, solusi nya ialah dengan menginstall library tersebut, pip install librosa.

- **An error occurred while starting the kernel**

Ini error yang tidak biasa karena error terdapat pada kernel, disini solusi nya ialah kita harus menginstall hdf5, yaitu dengan conda unin-stall hdf5 dan conda install hdf5.

### 7.1.4 Bukti Tidak Plagiat



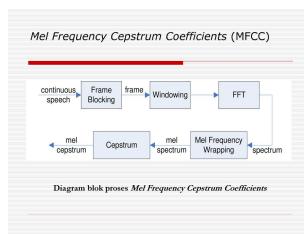
**Gambar 7.23** Bukti Tidak Melakukan Plagiat Chapter 6

## 7.2 1174054 - Aulyardha Anindita

### 7.2.1 Teori

#### 1. Mengapa file suara harus dilakukan MFCC

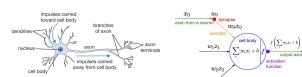
File suara harus dilakukan MFCC karena nilai-nilai pada MFCC meniru pendengaran manusia dan mereka biasanya digunakan dalam aplikasi pengenalan suara serta genre musik yang dideteksi. Nah, nilai-nilai MFCC ini kemudian akan dimasukkan langsung kedalam jaringan saraf agar dapat diubah menjadi bentuk vektor, sehingga dapat digunakan pada machine learning, karena machine learning hanya mengerti bilangan vektor. gambarannya seperti, ketika kita akan menggunakan file suara dalam machine learning, contohnya seperti melihat jam. nah machine learning ini tidak memahami rekaman suara tersebut melainkan vektornya, maka rekaman tersebut akan diubah terlebih dahulu kedalam bentuk vektor kemudian vektor tersebut akan menyesuaikan dengan kata-kata yang telah disediakan.



Gambar 7.24 File suara MFCC

#### 2. Konsep Dasar Neural Network

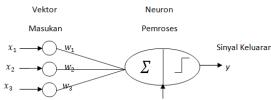
Neural Network adalah suatu model yang terinspirasi tentang bagaimana neuron dalam otak manusia bekerja, nah tiap neuron pada manusia akan saling berhubungan dan informasi tersebut akan mengalir dari setiap neuron tersebut. tiap neuron tersebut akan menerima input dan melakukan operasi dot dengan sebuah weight, kemudian menjumlahkannya dan menambahkan bias, dan hasil dari operasi tersebut akan dijadikan parameter dari activation function yang akan dijadikan output dari neuron tersebut. berikut adalah contoh proses dari neural network :



Gambar 7.25 Neural Network

### 3. Konsep pembobotan dalam neural network

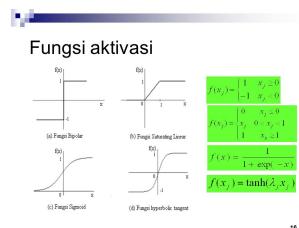
Dalam neural network terdapat jaringan yang saling berhubungan antar node-node atau simpulnya dimana pada tiap-tiap hubungan tersebut memiliki bobot koneksi yang dilatih untuk mencapai respon yang diinginkan. masing-masing bobot koneksi dipropagasi keseluruh simpul atau node. dengan pelatihan terhadap data berdasarkan bobot koneksi tersebut mampu memperoleh output yang diinginkan. struktur artificial neural network terdiri dari 3 lapisan. masing-masing lapisan diberikan pembobot yang akan mentransformasi nilai input menjadi nilai output. proses learning terjadi pada saat pengaturan pembobotan dan bias. Dalam metode yang dipilih, pembobotan diatur untuk meminimalisasi nilai kuadrat beda antara output model dan output taksiran atau secara umum disebut sebagai nilai kuadrat galat atau sum of square error.



**Gambar 7.26** Pembobotan dalam Neural Network

### 4. Konsep fungsi aktifasi dalam neural network

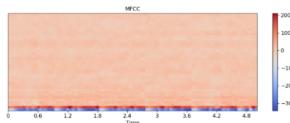
Fungsi aktivasi digunakan untuk menentukan keluaran suatu neuron. dalam fungsi aktivasi sendiri memiliki net masukan atau kombinasi linear masukan dan bobotnya. dalam fungsi aktivasi terdapat fungsi sigmoid yang kemudian dibagi dua bagian yaitu fungsi sigmoid biner dan fungsi sigmoid bipolar. adapun fungsi lain dari aktivasi yaitu memperkenalkan non-linearitas ke jaringan saraf. hal ini dapat menekan nilai dalam rentang yang lebih kecil.



**Gambar 7.27** Fungsi Aktivasi

### 5. Cara Membaca Hasil Plot dari MFCC

Dibawah ini merupakan contoh hasil plot dari rekaman suara:



**Gambar 7.28** Membaca hasil plot dari MFCC

dari gambar diatas dapat diketahui bahwa :

- terdapat 2 dimensi yaitu x sebagai waktu dan y sebagai power atau desibel
- jika berwarna biru maka power dari suara tersebut rendah dan jika merah power dari suara tersebut tinggi
- pada bagian atas terdapat warna pudar yang artinya tidak ada suara sama sekali dalam jangkauan itu.

## 6. One-hot encoding

One-hot encoding adalah suatu proses dimana variabel dikategorikal dan dikonversi menjadi bentuk yang dapat disediakan untuk algoritma ML untuk melakukan pekerjaan yang lebih baik dalam prediksi. nilai kategoris mewakili nilai numerik dari entri dalam dataset. Sebagai contoh: jika ada perusahaan lain dalam dataset, itu akan diberi nilai kategoris sebagai 4. Ketika jumlah entri unik meningkat, nilai kategoris juga meningkat secara proporsional. 0 menunjukkan tidak ada sementara 1 menunjukkan ada

The diagram shows a transformation process. On the left, there is a table with three columns: 'CompanyName', 'CategoricalValue', and 'Price'. The data is as follows:

CompanyName	CategoricalValue	Price
We	1	20
Acura	2	10001
Honda	3	100000
Mazda	3	100000

An arrow labeled "One-hot encoding" points to the right, where another table shows the same data converted into one-hot encoding format. This new table has four columns: 'We', 'Acura', 'Honda', and 'Price'. The 'Price' column remains the same, while the other three columns represent binary values indicating the presence or absence of each company name.

	We	Acura	Honda	Price
1	1	0	0	20
2	0	1	0	10001
3	0	0	1	100000
4	0	0	1	100000

**Gambar 7.29** One-hot encoding

## 7. Fungsi dari np.unique dan to.categorical dalam kode program

Untuk np.unique berfungsi untuk menemukan elemen unik array. mengembalikan elemen unik array yang diurutkan. ada tiap output opsional selain elemen unik yaitu : indeks array input yang memberikan nilai unik, indeks array unik yang merekonstruksi array input, dan berapa kali setiap nilai unik muncul dalam array input.

```
>>> np.unique([1, 1, 2, 2, 3, 3])
array([1, 2, 3])
>>> a = np.array([[1, 1], [2, 3]])
>>> np.unique(a)
array([1, 2, 3])
```

**Gambar 7.30** Numpy Unique

sedangkan untuk to categorical berfungsi untuk mengubah vektor kelas atau integer ke matriks kelas biner.

```
> Consider an array of 5 labels out of a set of 3 classes {0, 1, 2}:
> labels
array([0, 2, 1, 2, 0])
> # This 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., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.]], dtype=float32)
```

**Gambar 7.31** To Categorical

## 8. Fungsi dari Sequential dalam kode program

Sequential berfungsi sebagai tumpukan linear lapisan. Contohnya sebagai berikut :

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(2, input_dim=1))
model.add(Dense(1))
```

**Gambar 7.32** Sequential

### 7.2.2 Praktek

#### 1. Nomor 1

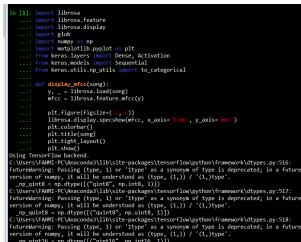
```
1 import librosa
2 import librosa.feature
3 import librosa.display
4 import glob
5 import numpy as np
6 import matplotlib.pyplot as plt
7 from keras.layers import Dense, Activation
8 from keras.models import Sequential
9 from keras.utils import np_utils
10
11 def display_mfcc(song):
12     y, _ = librosa.load(song)
13     mfcc = librosa.feature.mfcc(y)
14
```

```

15 plt.figure(figsize=(10, 4))
16 librosa.display.specshow(mfcc, x_axis='time', y_axis='mel'
17 ')
18 plt.colorbar()
19 plt.title(song)
20 plt.tight_layout()
21 plt.show()

```

Dalam kode diatas menjelaskan isi data GTZAN. GTZAN merupakan suatu kumpulan data yang berisi 10 genre lagu dimana masing-masing genre tersebut memiliki 100 lagu yang akan dilakukan proses MFCC dan freesound yang berisi konten lagu. Dan jika GTZAN memiliki beberapa genre freesound hanya untuk satu lagu sehingga disini letak fungsinya adalah membaca file audio dan outputnya sebagai plot. Dan hasilnya bisa dilihat pada gambar berikut :



**Gambar 7.33** Hasil Nomor 1

## 2. Nomor 2

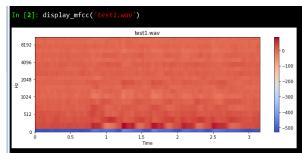
```

1 # In [2]: Nomor 2
2 display_mfcc('test1.wav')
3 # In [2]: Nomor 2
4 display_mfcc('test2.wav')
5 # In [2]: Nomor 2
6 display_mfcc('dataset/genres/disco/disco.00069.au')
7 # In [2]: Nomor 2
8 display_mfcc('dataset/genres/blues/blues.00069.au')
9 # In [2]: Nomor 2
10 display_mfcc('dataset/genres/classical/classical.00069.au')
11 # In [2]: Nomor 2
12 display_mfcc('dataset/genres/country/country.00069.au')
13 # In [2]: Nomor 2
14 display_mfcc('dataset/genres/hiphop/hiphop.00069.au')
15 # In [2]: Nomor 2
16 display_mfcc('dataset/genres/jazz/jazz.00069.au')
17 # In [2]: Nomor 2
18 display_mfcc('dataset/genres/pop/pop.00069.au')
19 # In [2]: Nomor 2
20 display_mfcc('dataset/genres/reggae/reggae.00069.au')
21 # In [2]: Nomor 2

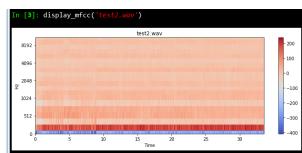
```

```
22 display_mfcc('dataset/genres/rock/rock.00069.au')
```

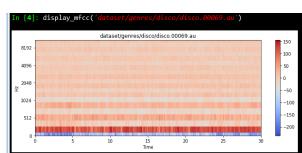
Dalam kode diatas digunakan untuk menampilkan hasil dari proses mfcc yang sudah dibuat pada nomor satu, yaitu display\_mfcc() yang kemudian akan menampilkan plot dari pembacaan file audio. Untuk hasil plotnya bisa dilihat pada gambar berikut:



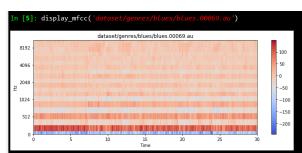
**Gambar 7.34** Hasil Nomor 2 - 1



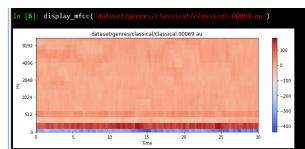
**Gambar 7.35** Hasil Nomor 2 - 2



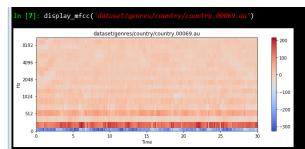
**Gambar 7.36** Hasil Nomor 2 - 3



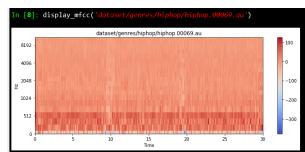
**Gambar 7.37** Hasil Nomor 2 - 4



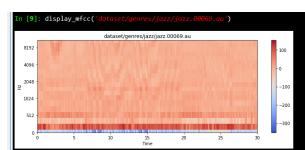
**Gambar 7.38** Hasil Nomor 2 - 5



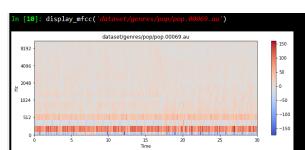
**Gambar 7.39** Hasil Nomor 2 - 6



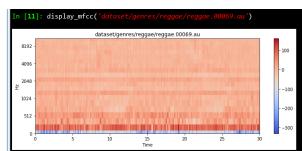
**Gambar 7.40** Hasil Nomor 2 - 7



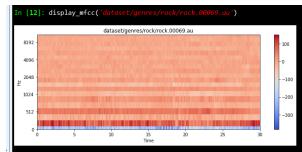
**Gambar 7.41** Hasil Nomor 2 - 8



**Gambar 7.42** Hasil Nomor 2 - 9



**Gambar 7.43** Hasil Nomor 2 - 10



**Gambar 7.44** Hasil Nomor 2 - 11

### 3. Nomor 3

```

1 def extract_features_song(f):
2     y, _ = librosa.load(f)
3
4     # get Mel-frequency cepstral coefficients
5     mfcc = librosa.feature.mfcc(y)
6     # normalize values between -1,1 (divide by max)
7     mfcc /= np.amax(np.absolute(mfcc))
8
9     return np.ndarray.flatten(mfcc)[:25000]

```

Pada kode diatas terdapat beberapa fungsi yang digunakan, nah pada baris pertama itu digunakan untuk membuat fungsi extract features song(f). selanjutnya, pada baris kedua digunakan untuk meload data inputan dengan menggunakan module librosa. Selanjutnya digunakan untuk membuat sebuah fitur untuk mfcc dari y atau parameter inputan. Kemudian akan mereturn dan menjadi array lalu akan mengambil 25000 data dari hasil vektorisasi dalam satu lagu. Hasilnya dapat dilihat pada gambar berikut :

```

In [13]: def extract_features_song():
...:     y, _ = librosa.load(f)
...:     # get Mel-frequency cepstral coefficients
...:     mfcc = librosa.feature.mfcc(y)
...:     # normalize values between -1,1 (divide by max)
...:     mfcc /= np.amax(np.absolute(mfcc))
...:     return np.ndarray.flatten(mfcc)[:25000]

```

**Gambar 7.45** Hasil Nomor 3

### 4. Nomor 4

```

1 def generate_features_and_labels():
2     all_features = []
3     all_labels = []
4
5     genres = [ 'blues' , 'classical' , 'country' , 'disco' , 'hiphop' ,
6               'jazz' , 'metal' , 'pop' , 'reggae' , 'rock' ]
7     for genre in genres:
8         sound_files = glob.glob('dataset/genres/' + genre + '/*.au')
9         print('Processing %d songs in %s genre...' % (len(sound_files), genre))
10        for f in sound_files:
11            features = extract_features_song(f)
12            all_features.append(features)
13            all_labels.append(genre)
14
15    # convert labels to one-hot encoding cth blues :
16    # 1000000000 classic 0100000000
17    label_uniq_ids, label_row_ids = np.unique(all_labels,
18                                              return_inverse=True) #ke integer
19    label_row_ids = label_row_ids.astype(np.int32, copy=False)
20    onehot_labels = to_categorical(label_row_ids, len(label_uniq_ids)) #ke one hot
21
22    return np.stack(all_features), onehot_labels

```

Dalam kode diatas digunakan kembali fungsi sebelumnya yang telah kita lakukan pada proses sebelumnya. lalu dibagian genre disesuaikan dengan dataset nama folder. Dan untuk baris selanjutnya digunakan untuk mengulang genre folder dengan ekstensi .au. lalu dipanggil fungsi ekstrak lagu. sehingga dalam setiap file dalam folder akan diekstraksi menjadi vektor dan akan ditambahkan fitur. lalu fungsi yang ditambahkan digunakan untuk menumpuk file yang telah divektorkan. Untuk hasilnya bisa dilihat pada gambar berikut :

```

In [4]: def generate_features_and_labels():
...:     all_features = []
...:     all_labels = []
...:
...:     genres = ['blues', 'classical', 'country', 'disco', 'hiphop',
...:              'jazz', 'metal', 'pop', 'reggae', 'rock']
...:
...:     for genre in genres:
...:         sound_files = glob.glob('dataset/genres/' + genre + '/*.au')
...:         print('Processing %d songs in %s genre...' % (len(sound_files), genre))
...:         for f in sound_files:
...:             features = extract_features_song(f)
...:             all_features.append(features)
...:             all_labels.append(genre)
...:
...:    # convert labels to one-hot encoding in this commented section because
...:    # label_uniq_ids, label_row_ids = np.unique(all_labels, return_inverse=True)
...:    # label_row_ids = label_row_ids.astype(np.int32, copy=False)
...:    # onehot_labels = to_categorical(label_row_ids, len(label_uniq_ids)) #ke one hot
...:    # np.stack(all_features), onehot_labels

```

Gambar 7.46      Hasil Nomor 4

## 5. Nomor 5

```

1 features, labels = generate_features_and_labels()
2 # In [5]: Nomor 5
3 print(np.shape(features))
4 print(np.shape(labels))

```

Dalam kode diatas digunakan untuk melakukan load variabel features dan labels. Hal ini membutuhkan waktu yang lama, karena mesin akan melakukan vektorisasi terhadap semua file yang berada pada setiap folder, disini terdapat 10 folder dimana pada masing-masing folder terdiri atas 100 buah lagu, dan setiap lagu tersebut akan dilakukan vektorisasi atau ekstraksi data menggunakan mfcc. Untuk hasilnya bisa dilihat pada gambar berikut :

```
In [13]: features, labels = generate_features_and_labels()
Processing 100 songs in blues genre...
Processing 100 songs in classical genre...
Processing 100 songs in country genre...
Processing 100 songs indisi genre...
Processing 100 songs in rock genre...
Processing 100 songs in jazz genre...
Processing 100 songs in metal genre...
Processing 100 songs in pop genre...
Processing 100 songs in reggae genre...
Processing 100 songs in rock genre...
```

**Gambar 7.47** Hasil Nomor 5-1

Name	Type	Size	Value
features	float32	(2000, 2000)	[[-0.81299893 -0.83154599 -0.75134417 ... -0.82318384 0.81827542]
label	float32	(2000, 10)	[1, 0, 0, ..., 0, 0, 0]

**Gambar 7.48** Hasil Nomor 5-2

```
In [16]: print(np.shape(features))
(...,)
(1000, 25000)
(1000, 10)
```

**Gambar 7.49** Hasil Nomor 5-3

## 6. Nomor 6

```
1 # In [6]: Nomor 6
2 training_split = 0.8
3 # In [6]: Nomor 6
4 alldata = np.column_stack(( features , labels ))
5 # In [6]: Nomor 6
6 np.random.shuffle(alldata)
7 splitidx = int(len(alldata) * training_split)
8 train , test = alldata [:splitidx ,:] , alldata [splitidx :,:]
9 # In [6]: Nomor 6
10 print(np.shape(train))
11 print(np.shape(test))
12 # In [6]: Nomor 6
13 train_input = train [:,:-10]
14 train_labels = train [:,-10:]
15 # In [6]: Nomor 6
16 test_input = test [:,:-10]
17 test_labels = test [:,-10:]
18 # In [6]: Nomor 6
```

```
19 print(np.shape(train_input))
20 print(np.shape(train_labels))
```

Dalam kode diatas digunakan untuk melakukan training split 80 persen, hal ini dilakukan agar mesin dapat terus belajar tentang data baru, jadi ketika prediksi dibuat berupa data yang terlatih mesin dapat mendapatkan persentase yang cukup bagus. Hasilnya bisa dilihat pada gambar berikut :

```
In [17]: training_split = 0.8
```

Gambar 7.50 Hasil Nomor 6-1

```
In [18]: alldata = np.column_stack((features, labels))
```

Gambar 7.51 Hasil Nomor 6-2

```
In [19]: np.random.shuffle(alldata)
...: splitidx = int(len(alldata) * training_split)
...: train, test = alldata[splitidx:], alldata[:splitidx]
```

Gambar 7.52 Hasil Nomor 6-3

```
In [20]: print(np.shape(train))
...: print(np.shape(test))
(800, 25010)
(200, 25010)
```

Gambar 7.53 Hasil Nomor 6-4

```
In [21]: train_input = train[:, :-10]
...: train_labels = train[:, -10:]
```

Gambar 7.54 Hasil Nomor 6-5

```
In [22]: test_input = test[:, :-10]
...: test_labels = test[:, -10:]
```

Gambar 7.55 Hasil Nomor 6-6

```
In [23]: print(np.shape(train_input))
...: print(np.shape(train_labels))
(800, 250000)
(800, 10)
```

Gambar 7.56 Hasil Nomor 6-7

**Gambar 7.57** Hasil Nomor 6-8

7. Nomor 7

```
1 # In [7]: Nomor 7
2 model = Sequential([
3     Dense(100, input_dim=np.shape(train_input)[1]),
4     Activation('relu'),
5     Dense(10),
6     Activation('softmax'),
7 ])
```

Dalam kode diatas terdapat fungsi sequential() digunakan untuk menetukan izin pada setiap neuron, yang memiliki 100 dense yang merupakan 100 neuron pertama dari data pelatihan. Fungsi dari relay itu sendiri digunakan untuk mengaktifkan neuron atau input yang memiliki nilai maksimum. sedangkan untuk dense 10 merupakan output dari hasil neuron yang berhasil diaktifkan, dan untuk dense 10 diaktifkan bisa menggunakan softmax. Hasilnya bisa dilihat pada gambar berikut :

```
In [24]: model = Sequential([
    ...     Dense(100, input_dim=np.shape(train_input)[1]),
    ...     Activation('relu'),
    ...     Dense(10),
    ...     Activation('softmax'),
    ... ])
```

**Gambar 7.58** Hasil Nomor 7

## 8. Nomor 8

```
1 # In [8]: Nomor 8
2 model.compile(optimizer='adam',
3                 loss='categorical_crossentropy',
4                 metrics=['accuracy']))
5 print(model.summary())
```

Dalam kode diatas terdapat model compile dimana hasilnya bisa dilihat pada gambar dibawah ini. Hasil output pada kode tersebut seperti gambar yang menjelaskan bahwa dense pertama memiliki 100 neurons dengan parameter sekitar 2 juta lebih dengan aktivasi 100, jadi untuk setiap neurons memiliki masing-masing 1 aktivasi. sama halnya seperti dense 2 yang memiliki jumlah neurons sebanyak 10 dengan parameter 1010 dan

jumlah aktifasinya 10 untuk setiap neurons tersebut dan dimana total parameternya sekitar 2,5 juta data yang akan dilatih pada mesin tersebut.

```
In [8]: model.compile(optimizer='adam',
    loss='categorical_crossentropy',
    metrics=[accuracy])
... print(model.summary())
Model: "sequential_1"
Layer (type)                 Output Shape              Param #   
dense_1 (Dense)              (None, 100)             2500100  
activation_1 (Activation)    (None, 100)             0        
dense_2 (Dense)              (None, 10)              1010    
activation_2 (Activation)    (None, 10)             0        
Total params: 2,501,110
Trainable params: 2,501,110
Non-trainable params: 0
None
```

Gambar 7.59      Hasil Nomor 8

## 9. Nomor 9

```
1 # In [9]: Nomor 9
2 model.fit(train_input , train_labels , epochs=10, batch_size
   =32,
3           validation_split=0.2)
```

Dalam kode diatas digunakan untuk melatih mesin dengan data training input dan data training label. Epochs dalam kode tersebut adalah iterasi atau pengulangan berapa kali data tersebut akan dilakukan. sedangkan batch size yaitu jumlah file yang akan dilakukan pelatihan pada setiap satu kali pengulangan dan terakhir validation split digunakan untuk menentukan persentase dari cross validation atau k-fold sebanyak 20 persen dari masing-masing data pengulangan. Hasilnya bisa dilihat pada gambar berikut :

```
In [10]: model.fit(train_input , train_labels , epochs=10, batch_size=32,
   validation_split=0.2)
WARNING:tensorflow:tf.global_variables is deprecated. Please use tf.Variable instead.
WARNING:tensorflow:tf.global_variables is deprecated. Please use tf.Variable instead.
Train on 1400 samples, validate on 100 samples
Epoch 1/10
1400/1400 [==============================] 1s - loss: 2.1898 - accuracy: 0.2945 -
val_loss: 1.8988 - val_accuracy: 0.2688
Epoch 2/10
1400/1400 [==============================] 1s - loss: 2.0417 - accuracy: 0.4068 -
val_loss: 1.5779 - val_accuracy: 0.4068
Epoch 3/10
1400/1400 [==============================] 1s - loss: 1.6017 - accuracy: 0.4220 -
val_loss: 1.5413 - val_accuracy: 0.4053
Epoch 4/10
1400/1400 [==============================] 1s - loss: 0.8449 - accuracy: 0.7337 -
val_loss: 1.4955 - val_accuracy: 0.4043
Epoch 5/10
1400/1400 [==============================] 1s - loss: 0.7162 - accuracy: 0.8125 -
val_loss: 1.4798 - val_accuracy: 0.4053
Epoch 6/10
1400/1400 [==============================] 1s - loss: 0.5346 - accuracy: 0.8729 -
val_loss: 1.4686 - val_accuracy: 0.4053
Epoch 7/10
1400/1400 [==============================] 1s - loss: 0.4318 - accuracy: 0.9127 -
val_loss: 1.4586 - val_accuracy: 0.4053
Epoch 8/10
1400/1400 [==============================] 1s - loss: 0.3422 - accuracy: 0.9556 -
val_loss: 1.4492 - val_accuracy: 0.4053
```

Gambar 7.60      Hasil Nomor 9

## 10. Nomor 10

```
1 # In [10]: Nomor 10
```

```

2 loss , acc = model.evaluate(test_input , test_labels ,
3                             batch_size=32)
# In [10]: Nomor 10
4 print("Done!")
5 print("Loss: %.4f , accuracy: %.4f" % (loss , acc))

```

dalam kode diatas terdapat fungsi evaluate atau evaluasi yang digunakan untuk menguji data pengujian setiap file. disini terdapat prediksi yang hilang, artinya mesin memprediksi data, sedangkan untuk keseluruhan perjanjian sekitar 55 persen, Hasilnya bisa dilihat pada gambar berikut :

```

In [10]: loss, acc = model.evaluate(test_input, test_labels, batch_size=32)
288/288 [=====] - 0s 280ms/step

```

**Gambar 7.61** Hasil Nomor 10-1

```

In [28]: print("Done!")
Done!
In [28]: print("Loss: %.4f , accuracy: %.4f" % (loss , acc))
Loss: 2.4637, accuracy: 0.4900

```

**Gambar 7.62** Hasil Nomor 10-2

## 11. Nomor 11

```

1 # In [11]: Nomor 11
2 model.predict(test_input [:1])

```

Dalam kode diatas terdapat fungsi predict, fungsi ini berfungsi untuk menghasilkan suatu nilai yang sudah iprediksi dari data training sebelumnya. dalam gambar dibawah ini, menjelaskan file yang dijalankan tersebut termasuk kedalam genre apa, hasilnya bisa dilihat pada gambar tersebut serta persentase yang paling besar yakni genre rock, maka lagu tersebut termasuk kedalam genre rock dengan perbandingan persentase hasil prediksi. Hasilnya bisa dilihat pada gambar berikut :

```

In [29]: model.predict(test_input[:1])
Out[29]:
array([5.873318e-02, 8.232039e-03, 6.366556e-03, 1.3738674e-02,
       5.568286e-02, 6.7089213e-01, 2.779746e-04, 2.0203577e-02,
       2.5181699e-02, 3.0326517e-02]), dtype=float32

```

**Gambar 7.63** Hasil Nomor 11

### 7.2.3 Penanganan Error

#### 1. ScreenShoot Error

```
| File "<ipython-input-1-36bfa33e5383>", line 1, in <module>
|     import librosa
|
| ModuleNotFoundError: No module named 'librosa'
```

**Gambar 7.64** Module Not Found Error

2. Tuliskan Kode Error dan Jenis Error

- Module Not Found Error

3. Cara Penanganan Error

- Module Not Found Error

Error terjadi karena belum menginstal package atau library librosa  
Untuk mengatasinya dengan menginstall library librosa pada anaconda

#### 7.2.4 Bukti Tidak Plagiat



**Gambar 7.65** Bukti Plagiarisme

#### 7.2.5 Link Youtube

<https://youtu.be/2YVDpPMuNG0>

## BAB 8

---

# CHAPTER 7

---

### 8.1 1174006 - Kadek Diva Krishna Murti

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

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



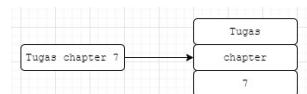
**Gambar 8.2** Kecerdasan Buatan.

### 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.



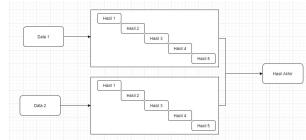
**Gambar 8.3** Teori 1

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 di inginkan.

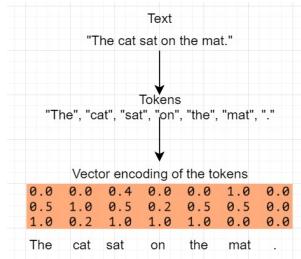
```
1 import pandas as pd
2
3 mydict = [{ 'a': 1, 'b': 2, 'c': 3, 'd': 4},
4           { 'a': 100, 'b': 200, 'c': 300, 'd': 400},
5           { 'a': 1000, 'b': 2000, 'c': 3000, 'd': 4000 }]
6 df = pd.DataFrame(mydict)
7 df
```

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

**Gambar 8.6** Teori 4

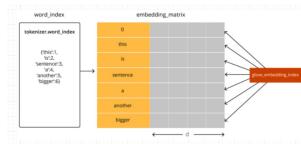
#### 8.2.1.5 Apa maksud dari fungsi `tokenizer = Tokenizer(num_words=2000)` dan `tokenizer.fit_on_texts(train content)`

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

**Gambar 8.7** Teori 5

#### 8.2.1.6 Apa maksud dari fungsi `d train inputs = tokenizer.texts_to_matrix(train content, mode='tfidf')` dan `d test inputs = tokenizer.texts_to_matrix(test content, mode='tfidf')`

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

**Gambar 8.8** Teori 6

#### 8.2.1.7 Apa maksud dari fungsi `d train inputs = d train inputs / np.amax(np.absolute(d train inputs))` dan `d test inputs = d test inputs / np.amax(np.absolute(d test inputs))`

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

```
>>> a = np.arange(4).reshape((2,2))
>>> a
array([[0, 1],
       [2, 3]])
>>> 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.

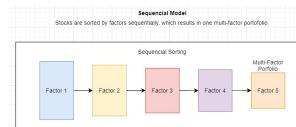
```
>>> v_train
array([[1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.]], dtype=int64)
>>> v_train.flatten()
array([1, 0, ..., 1, 0, 0], dtype=int64)
>>> v_train
array([[1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
      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

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

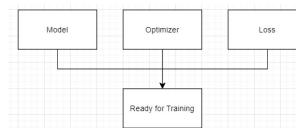
**8.2.1.10 Apa maksud dari fungsi di listing 7.3 dengan parameter tersebut**

```

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**

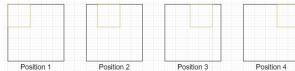
Deep Learning adalah salah satu cabang dari ilmu machine learning yang terdiri dari algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam machine learning dapat digunakan baik untuk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

Gambar 8.15 Hasil No 2

#### 8.2.2.3 Nomor 3

```

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

```

Hasil:

Gambar 8.16 Hasil No 1

#### 8.2.2.4 Nomor 4

```

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

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
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(1024, activation='tanh')) #Variabel model di
        tambahkan library Dense dengan fungsi tanh
8 model.add(Dropout(0.5)) #Variabel model di tambahkan library
        dropout untuk memangkas data tree sebesar 50 persen
9 model.add(Dense(num_classes, activation='softmax')) #Variabel
        model di tambahkan library Dense dengan data dari num_classes
        dan fungsi softmax
10 model.compile(loss='categorical_crossentropy', optimizer='adam',
                 metrics=['accuracy']) #Mengkompile data model untuk
                     mendapatkan data loss akurasi dan optimasi
11 print(model.summary()) #Mencetak variabel model kemudian
        memunculkan kesimpulan berupa data total parameter , trainable
        paremeter dan bukan trainable parameter

```

Hasil:

Model: "sequential_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:

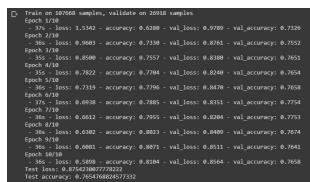
```
[11] import keras.callbacks #Pengimport library keras dengan fungsi callbacks
    tensorboard = keras.callbacks.TensorBoard(log_dir='./content/logs/mnist-style')
```

Gambar 8.24 Hasil No 11

### 8.2.2.12 Nomor 12

```
1 model.fit(train_input, train_output, #Fungsi model ditambahkan  
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])
```

Hasjli



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
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 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: 0.96, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.20	Loss: 0.95, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.30	Loss: 0.95, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.40	Loss: 0.95, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.50	Loss: 0.95, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.60	Loss: 0.95, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.70	Loss: 0.95, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.80	Loss: 0.95, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 0.90	Loss: 0.95, Accuracy: 8.76	Time: 234 sec
0	Conv2D	1. Dense size 128, Dropout 1.00	Loss: 0.95, 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: 1.39, Accuracy: 8.76	Time: 406 sec
0	Conv2D	1. Dense size 128, Dropout 0.20	Loss: 1.34, Accuracy: 8.76	Time: 406 sec
0	Conv2D	1. Dense size 128, Dropout 0.30	Loss: 1.34, Accuracy: 8.76	Time: 406 sec
0	Conv2D	1. Dense size 128, Dropout 0.40	Loss: 1.34, Accuracy: 8.76	Time: 406 sec
0	Conv2D	1. Dense size 128, Dropout 0.50	Loss: 1.34, Accuracy: 8.76	Time: 406 sec
0	Conv2D	1. Dense size 128, Dropout 0.60	Loss: 1.34, Accuracy: 8.76	Time: 406 sec
0	Conv2D	1. Dense size 128, Dropout 0.70	Loss: 1.34, Accuracy: 8.76	Time: 406 sec
0	Conv2D	1. Dense size 128, Dropout 0.80	Loss: 1.34, Accuracy: 8.76	Time: 406 sec
0	Conv2D	1. Dense size 128, Dropout 0.90	Loss: 1.34, Accuracy: 8.76	Time: 406 sec
0	Conv2D	1. Dense size 128, Dropout 1.00	Loss: 1.34, Accuracy: 8.76	Time: 406 sec
0	Conv2D	1. Dense size 128, Dropout 0.00	Loss: 0.95, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.10	Loss: 0.95, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.20	Loss: 0.95, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.30	Loss: 0.95, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.40	Loss: 0.95, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.50	Loss: 0.95, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.60	Loss: 0.95, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.70	Loss: 0.95, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.80	Loss: 0.95, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.90	Loss: 0.95, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 1.00	Loss: 0.95, Accuracy: 8.76	Time: 593 sec
0	Conv2D	1. Dense size 128, Dropout 0.00	Loss: 0.89, Accuracy: 8.76	Time: 288 sec
0	Conv2D	1. Dense size 128, Dropout 0.10	Loss: 0.89, Accuracy: 8.76	Time: 288 sec
0	Conv2D	1. Dense size 128, Dropout 0.20	Loss: 0.89, Accuracy: 8.76	Time: 288 sec
0	Conv2D	1. Dense size 128, Dropout 0.30	Loss: 0.89, Accuracy: 8.76	Time: 288 sec
0	Conv2D	1. Dense size 128, Dropout 0.40	Loss: 0.89, Accuracy: 8.76	Time: 288 sec
0	Conv2D	1. Dense size 128, Dropout 0.50	Loss: 0.89, Accuracy: 8.76	Time: 288 sec
0	Conv2D	1. Dense size 128, Dropout 0.60	Loss: 0.89, Accuracy: 8.76	Time: 288 sec
0	Conv2D	1. Dense size 128, Dropout 0.70	Loss: 0.89, Accuracy: 8.76	Time: 288 sec
0	Conv2D	1. Dense size 128, Dropout 0.80	Loss: 0.89, Accuracy: 8.76	Time: 288 sec
0	Conv2D	1. Dense size 128, Dropout 0.90	Loss: 0.89, Accuracy: 8.76	Time: 288 sec
0	Conv2D	1. Dense size 128, Dropout 1.00	Loss: 0.89, Accuracy: 8.76	Time: 288 sec
0	Conv2D	1. Dense size 128, Dropout 0.00	Loss: 1.49, Accuracy: 8.76	Time: 343 sec
0	Conv2D	1. Dense size 128, Dropout 0.10	Loss: 1.44, Accuracy: 8.76	Time: 343 sec
0	Conv2D	1. Dense size 128, Dropout 0.20	Loss: 1.44, Accuracy: 8.76	Time: 343 sec
0	Conv2D	1. Dense size 128, Dropout 0.30	Loss: 1.44, Accuracy: 8.76	Time: 343 sec
0	Conv2D	1. Dense size 128, Dropout 0.40	Loss: 1.44, Accuracy: 8.76	Time: 343 sec
0	Conv2D	1. Dense size 128, Dropout 0.50	Loss: 1.44, Accuracy: 8.76	Time: 343 sec
0	Conv2D	1. Dense size 128, Dropout 0.60	Loss: 1.44, Accuracy: 8.76	Time: 343 sec
0	Conv2D	1. Dense size 128, Dropout 0.70	Loss: 1.44, Accuracy: 8.76	Time: 343 sec
0	Conv2D	1. Dense size 128, Dropout 0.80	Loss: 1.44, Accuracy: 8.76	Time: 343 sec
0	Conv2D	1. Dense size 128, Dropout 0.90	Loss: 1.44, Accuracy: 8.76	Time: 343 sec
0	Conv2D	1. Dense size 128, Dropout 1.00	Loss: 1.44, Accuracy: 8.76	Time: 343 sec
0	Conv2D	1. Dense size 128, Dropout 0.00	Loss: 1.34, Accuracy: 8.76	Time: 511 sec
0	Conv2D	1. Dense size 128, Dropout 0.10	Loss: 1.34, Accuracy: 8.76	Time: 511 sec
0	Conv2D	1. Dense size 128, Dropout 0.20	Loss: 1.34, Accuracy: 8.76	Time: 511 sec
0	Conv2D	1. Dense size 128, Dropout 0.30	Loss: 1.34, Accuracy: 8.76	Time: 511 sec
0	Conv2D	1. Dense size 128, Dropout 0.40	Loss: 1.34, Accuracy: 8.76	Time: 511 sec
0	Conv2D	1. Dense size 128, Dropout 0.50	Loss: 1.34, Accuracy: 8.76	Time: 511 sec
0	Conv2D	1. Dense size 128, Dropout 0.60	Loss: 1.34, Accuracy: 8.76	Time: 511 sec
0	Conv2D	1. Dense size 128, Dropout 0.70	Loss: 1.34, Accuracy: 8.76	Time: 511 sec
0	Conv2D	1. Dense size 128, Dropout 0.80	Loss: 1.34, Accuracy: 8.76	Time: 511 sec
0	Conv2D	1. Dense size 128, Dropout 0.90	Loss: 1.34, Accuracy: 8.76	Time: 511 sec
0	Conv2D	1. Dense size 128, Dropout 1.00	Loss: 1.34, Accuracy: 8.76	Time: 511 sec
0	Conv2D	1. Dense size 128, Dropout 0.00	Loss: 1.20, Accuracy: 8.76	Time: 617 sec

Gambar 8.26 Hasil No 13

#### 8.2.2.14 Nomor 14

```

1 model = Sequential() #Membuat variabel model dengan isian library Sequential
2 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
      input_shape=np.shape(train_input[0])))#Variabel model di tambahkan library Conv2D 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
[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 immage 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('invertedclasses.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) #membaca gambar
    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
    # inverse it
    print("Prediction: %s, confidence: %.2f" % (label_encoder.inverse_transform([np.argmax(prediction)]), 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?	<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 8.36** 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
  
```

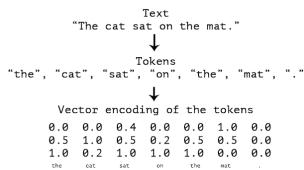
```

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.

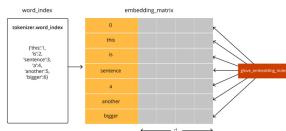
Fungsi tokenizer ini berfungsi untuk melakukan vektorisasi data kedalam bentuk token sebanyak 2000 kata. Dan selanjutnya akan melakukan fit tokenizer hanya untuk data training saja tidak dengan data testingnya. Untuk ilustrasi lihat gambar berikut:



**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.,  1.,  0.],
       [ 1.,  0.,  0.],
       [ 0.,  0.,  1.],
       [ 1.,  0.,  0.],
       [ 0.,  1.,  0.]])

```

Gambar 8.43 Teori 8

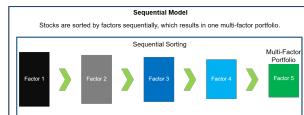
9. Jelaskan apa maksud dari fungsi di listing 7.2. Gambarkan ilustrasi Neural Network nya dari model kode tersebut.

```

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

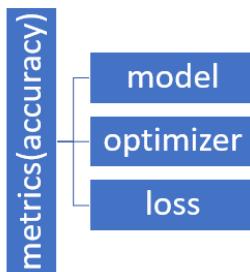
10. Jelaskan apa maksud dari fungsi di listing 7.3 dengan parameter tersebut.

```

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.

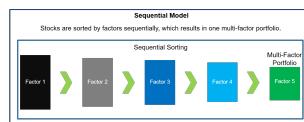
12. Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning.

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 library Image yang berguna untuk dari PIL atau Python
# imaging Library yang berguna untuk mengolah data berupa gambar
from PIL import Image as pil_image
# kemudian 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
    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 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 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 [4]: import numpy as np
.... train_input = np.asarray(list(map(lambda row: row[2], train)))
.... test_input = np.asarray(list(map(lambda row: row[2], test)))
.... train_output = np.asarray(list(map(lambda row: row[1], train)))
.... test_output = np.asarray(list(map(lambda row: row[1], test)))

```

**Gambar 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_)

num_classes
print("Number of classes: %d" % num_classes)
Number of 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',
23                 metrics=['accuracy'])
24 # mencetak variabel model kemudian memunculkan kesimpulan
    berupa data total parameter , trainable paremeter dan bukan
    trainable parameter
25 print(model.summary())
```

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

```

model.add(conv1_1, kernel_size=(3, 3), activation='relu')
input_shape = model.output_shape[1:(model.output_shape[1]+1)]
model.add(conv1_2, input_shape=input_shape, activation='relu')
model.add(conv2D_1, (4, 4), activation='relu')
model.add(maxpool2d_1, pool_size=(2, 2))
model.add(lateral_1)
model.add(dense_1, activation='softmax')
model.add(Dense(100, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
print(model.summary())
Model: <Sequential>
layer (type)          Output Shape       Param #
=====
conv2d_1 (Conv2D)     (None, 10, 32)      896
max_pooling2d_1 (MaxPooling2D) (None, 15, 32)    0
conv2d_2 (Conv2D)     (None, 13, 32)      9248
max_pooling2d_2 (MaxPooling2D) (None, 6, 32)    0
flatten_1 (Flatten)   (None, 1152)        0
dense_1_1 (Dense)    (None, 128)         147584
dropout_1 (Dropout)  (None, 128)         0
dense_1_2 (Dense)    (None, 369)         47601
=====
Total params: 285,299
Trainable params: 285,299
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/mnist-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
17/18 - loss: 1.3816 - accuracy: 0.6081 - val_loss: 1.0032 - val_accuracy: 0.7932
18/18 - loss: 0.8688 - accuracy: 0.7494 - val_loss: 0.8682 - val_accuracy: 0.7922
Epoch 2/10
1/18 - loss: 1.0804 - accuracy: 0.6767 - val_loss: 0.8862 - val_accuracy: 0.7952
2/18 - loss: 0.8688 - accuracy: 0.7494 - val_loss: 0.8862 - val_accuracy: 0.7929
3/18 - loss: 0.8688 - accuracy: 0.7494 - val_loss: 0.8862 - val_accuracy: 0.7929
4/18 - loss: 0.7751 - accuracy: 0.7718 - val_loss: 0.8479 - val_accuracy: 0.7918
5/18 - loss: 0.7751 - accuracy: 0.7718 - val_loss: 0.8479 - val_accuracy: 0.7918
6/18 - loss: 0.7751 - accuracy: 0.7718 - val_loss: 0.8479 - val_accuracy: 0.7918
7/18 - loss: 0.7751 - accuracy: 0.7718 - val_loss: 0.8479 - val_accuracy: 0.7918
8/18 - loss: 0.7751 - accuracy: 0.7718 - val_loss: 0.8479 - val_accuracy: 0.7918
9/18 - loss: 0.7751 - accuracy: 0.7718 - val_loss: 0.8479 - val_accuracy: 0.7918
10/18 - loss: 0.6295 - accuracy: 0.7924 - val_loss: 0.6462 - val_accuracy: 0.7939
11/18 - loss: 0.6295 - accuracy: 0.7924 - val_loss: 0.6462 - val_accuracy: 0.7939
12/18 - loss: 0.6295 - accuracy: 0.7924 - val_loss: 0.6462 - val_accuracy: 0.7939
13/18 - loss: 0.6295 - accuracy: 0.7924 - val_loss: 0.6462 - val_accuracy: 0.7939
14/18 - loss: 0.6295 - accuracy: 0.7924 - val_loss: 0.6462 - val_accuracy: 0.7939
15/18 - loss: 0.6295 - accuracy: 0.7924 - val_loss: 0.6462 - val_accuracy: 0.7939
16/18 - loss: 0.6295 - accuracy: 0.7924 - val_loss: 0.6462 - val_accuracy: 0.7939
17/18 - loss: 0.6295 - accuracy: 0.7924 - val_loss: 0.6462 - val_accuracy: 0.7939
18/18 - loss: 0.6295 - accuracy: 0.7924 - val_loss: 0.6462 - val_accuracy: 0.7939
test_accuracy: 0.79551598898345

```

**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'))
36                    # menambahkan method add pada variabel model
37                    dengan isian method Max pooling berdimensi 2 dengan
38                    ukuran fixcel 2 x 2.
39                    model.add(MaxPooling2D(pool_size=(2, 2)))
40                    # merubah feature gambar menjadi 1 dimensi vektor
41                    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	model	.add	(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=np.shape(train_input[0])))
3	model	.add	(MaxPooling2D(pool_size=(2, 2)))
4	model	.add	(Conv2D(32, (3, 3), activation='relu'))
5	model	.add	(MaxPooling2D(pool_size=(2, 2)))
6	model	.add	(Flatten())
7	model	.add	(Dense(128, activation='tanh'))
8	model	.add	(Dense(num_classes, activation='softmax'))
9	model	.compile	(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
10	print	(model.summary())	

Gambar 8.58 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()
5 # variabel model di tambahkan librari Conv2D tigapuluhan dua
6   bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
7   relu dang menggunakan data train_input
8 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
9   input_shape=np.shape(train_input[0])))
10 # variabel model di tambahkan dengan lib MaxPooling2D dengan
11   ketentuan ukuran 2 x 2 pixcel
12 model.add(MaxPooling2D(pool_size=(2, 2)))
13 # variabel model di tambahkan dengan librari Conv2D 32bit
14   dengan kernel 3 x 3
15 model.add(Conv2D(32, (3, 3), activation='relu'))
16 # variabel model di tambahkan dengan lib MaxPooling2D dengan
17   ketentuan ukuran 2 x 2 pixcel
18 model.add(MaxPooling2D(pool_size=(2, 2)))
19 # variabel model di tambahkan librari Flatten
20 model.add(Flatten())
21 # variabel model di tambahkan librari Dense dengan fungsi
22   tanh
23 model.add(Dense(128, activation='tanh'))
24 # variabel model di tambahkan librari dropout untuk memangkas
25   data tree sebesar 50 persen
26 model.add(Dropout(0.5))
27 # variabel model di tambahkan librari Dense dengan data dari
28   num_classes dan fungsi softmax
29 model.add(Dense(num_classes, activation='softmax'))
30 # mengkompile data model untuk mendapatkan data loss akurasi
31   dan optimasi
32 model.compile(loss='categorical_crossentropy', optimizer='adam',
33   metrics=['accuracy'])
34 # mencetak variabel model kemudian memunculkan kesimpulan
35   berupa data total parameter, trainable paremeter dan bukan
36   trainable parameter
37 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)))
--> model.add(latten())
--> model.add(dense(128, activation='tanh'))
--> model.add(dense(num_classes, activation='softmax'))
--> model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
Model: "sequential_5"
Layer (type)                 Output Shape              Param #
conv2d_6 (Conv2D)            (None, 36, 36, 32)       896
max_pooling2d_6 (MaxPooling2d) (None, 18, 18, 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_4 (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 0x1f637ebc3d8>

```

**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, 10, 30, 32)	896
max_pooling2d_68 (MaxPooling)	(None, 15, 15, 32)	0
conv2d_69 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_69 (MaxPooling)	(None, 9, 6, 32)	0
flatten_45 (Flatten)	(None, 1152)	0
dense_89 (Dense)	(None, 128)	147584
dropout_34 (Dropout)	(None, 128)	0
dense_90 (Dense)	(None, 369)	47681
Total params:	209,320	
Trainable params:	206,329	
Non-trainable params:	0	
None		

**Gambar 8.64** Hasil Soal 19.

20. Soal 20

```
1 # In [20]: grab an image (we'll just use a random training  
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("nsv2/hazy-data/v2-00500.png")

Prediction: A, confidence: 0.87

predict("nsv2/hazy-data/v2-00500.png")

Prediction: \p1, confidence: 0.58

predict("nsv2/hazy-data/v2-00700.png")

Prediction: \alpha\beta, confidence: 0.88
```

**Gambar 8.65** Hasil Soal 20.

### **8.3.3 Penanganan Error**

## 1. KeyboardInterrupt

```
File "C:\Users\lucas\anaconda\lib\site-packages\keras\preprocessing\image.py", line 17, in module
    img = keras.preprocessing.image_img_to_array(pic_image.open("%sGV2/*" + row[0]))  
  
File "C:\Users\lucas\anaconda\lib\site-packages\pillow\Image.py", line 2809, in open
    fp = builtins.open(filename, "rb")  
  
KeyboardInterrupt
```

**Gambar 8.66** KeyboardInterrupt

## 2. Cara Penanganan Error

- KeyboardInterrupt

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

### 8.3.4 Bukti Tidak Plagiat



Gambar 8.67    Bukti Tidak Melakukan Plagiat Chapter 7

### 8.3.5 Link Video Youtube

<https://youtu.be/pV9yrZNEZj4>

## 8.4 1174079 - Chandra Kirana Poetra

### 8.4.1 Teori

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

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



Gambar 8.68    Teori 1

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

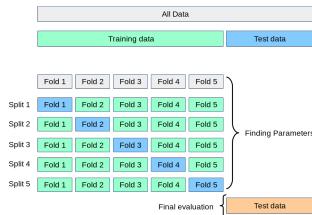
Konsep sederhana dari K Fold Cross Validation ialah Pada code ini:

```

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

```

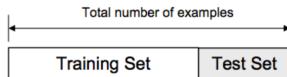
Kfold diatas bertujuan untuk membagi data kedalam 5 bagian yang nantinya akan menampung komentar dari youtube. data yang dibagi ini nanti akan dihitung dan akan menghasilkan persentase yang bisa dibilang cukup baik



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)*

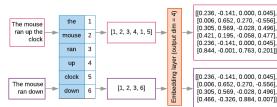
Berfungsi untuk melakukan proses vektorisasi data menjadi token sebanyak 2000 kata



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.

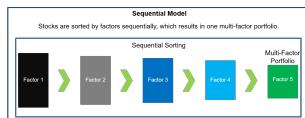
```
>>> V_train
array([[0., 0., 0.],
       [0., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.], dtype=int64)
>>> V_train.flatten()
array([0, 0, 0, ..., 1, 0, 0], dtype=int64)
>>> V_train
array([[1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
    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



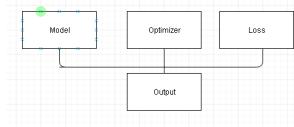
Gambar 8.76 Teori 9

#### *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

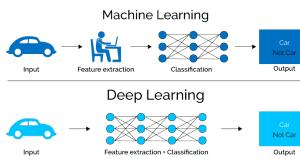
adalah adam, selain adam, ada juga SGD,RMSprop, dan lain lain dan metrics accuracy adalah value yang akan kita cari nilainya



**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



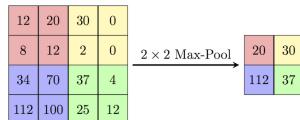
#### 8.4.1.12 Apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning

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 1174070 - Arrizal Furqona Gifary

#### 8.5.1 Soal Teori

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

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

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

**Gambar 8.81** Teori 1

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

```

1 kfold = StratifiedKFold(n_splits=5)
2

```

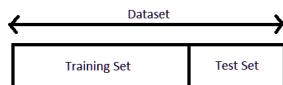
Konsep sederhana dari K Fold Cross Validation ialah Pada code tersebut terdapat kfolds yang bertujuan untuk melakukan split data menjadi 5 bagian dari dataset komentar Youtube tersebut. Sehingga dari setiap data yang sudah dibagi tersebut akan menghasilkan persentase dari setiap bagiannya, untuk menghasilkan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

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

**Gambar 8.82** 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 8.83** 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

```

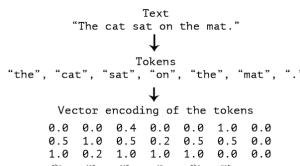
**Out[5]:**

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

**Gambar 8.84** Teori 4

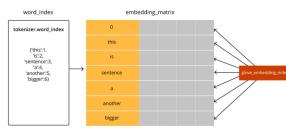
5. Jelaskan apa maksud dari fungsi tokenizer = Tokenizer(num words=2000) dan tokenizer.fit on texts(train content), dilengkapi dengan ilustrasi atau gambar.

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 8.85** 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 8.86** Teori 6

7. Jelaskan apa maksud dari fungsi  $d \text{ train inputs} = d \text{ train inputs}/\text{np.amax}(\text{np.absolut}(d \text{ train inputs}))$  dan  $d \text{ test inputs} = d \text{ test inputs}/\text{np.amax}(\text{np.absolut}(d \text{ test inputs}))$ , dilengkapi dengan ilustrasi atau gambar.

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

```

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

```

Gambar 8.87 Teori 7

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

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

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

**Gambar 8.88** Teori 8

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

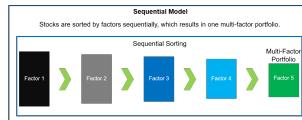
```
1 model = Sequential()  
2 model.add(Dense(512, input_shape=(2000,)))  
3 model.add(Activation('relu'))
```

```

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 8.89** Teori 9

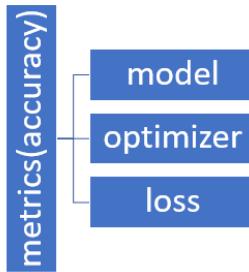
10. Jelaskan apa maksud dari fungsi di listing 7.3 dengan parameter tersebut.

```

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 8.90** 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 meng-

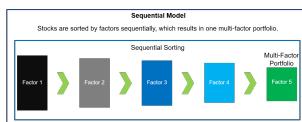
gunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam machine learning dapat digunakan baik untuk supervised learning dan unsupervised learning.

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

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

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

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



Gambar 8.91 Teori 9

### 8.5.2 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]: In[1]: CSV
import numpy as np
import os
from PIL import Image
from keras import Sequential
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import optimizers
from keras import metrics
from keras import preprocessing, image
using TensorFlow backend.
```

Gambar 8.92 Hasil Soal 1.

## 2. Soal 2

```
1 # In[2]:load all images (as numpy arrays) and save their
2 #classes
3 #membuat variabel imgs dengan variabel kosong
4 imgs = []
5 #membuat variabel classes dengan variabel kosong
6 classes = []
7 #membuka file hasy-data-labels.csv yang berada di foleder
8 # HASYv2 yang di inisialisasi menjadi csvfile
9 with open('HASYv2/hasy-data-labels.csv') as csvfile:
10     #membuat variabel csvreader yang berisi method csv.reader
11     # yang membaca variabel csvfile
12     csvreader = csv.reader(csvfile)
13     # membuat variabel i dengan isi 0
14     i = 0
15     # membuat looping pada variabel csvreader
16     for row in csvreader:
17         # dengan ketentuan jika i lebihkecil daripada 0
18         if i > 0:
19             # dibuat variabel img dengan isi keras untuk
20             # aktivasi neural network fungsi yang membaca data yang
21             # berada dalam folder HASYv2 dengan input nilai -1.0 dan 1.0
22             img = keras.preprocessing.image.img_to_array(
23                 pil_image.open("HASYv2/" + row[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             #membagi data yang ada pada fungsi img sebanyak
29             255.0
30             img /= 255.0
31             # menambah nilai baru pada imgs pada row ke 1 2
32             # dan dilanjutkan dengan variabel img
33             imgs.append((row[0], row[2], img))
34             # menambahkan nilai pada row ke 2 pada variabel
35             classes
36             classes.append(row[2])
37             # penambahan nilai satu pada variabel i
38             i += 1
```

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

### 3. Soal 3

```

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

```

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

```

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

```

**Gambar 8.93** 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[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 8.94** 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 8.95** 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 mengonvert variabel classes kedalam
bentuk integer
...: integer_encoded =
label_encoder.fit_transform(classes)
```

Gambar 8.96 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 :

**Gambar 8.97** 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)
      ...
      train_output_int = np.expand_dims(train_output_int.reshape(len(train_output_int), -1), axis=1)
      test_output_int = label_encoder.transform(test_output)
      test_output_int = np.expand_dims(test_output_int.reshape(len(test_output_int), -1), axis=1)
      ...
      num_classes = len(label_encoder.classes_)
      print(f'Number of classes: {len(label_encoder.classes_)}')
      Number of classes: 392
```

**Gambar 8.98** 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 :

Gambar 8.99 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.addConv2D(16, kernel_size=(3, 3), activation='relu',
input_shape=(28, 28, 1), padding='same')
... model.addMaxPooling2D(pool_size=(2, 2))
... model.addConv2D(32, (3, 3), activation='relu')
... model.addMaxPooling2D(pool_size=(2, 2))
... model.add(Flatten())
... model.add(Dense(128, activation='tanh'))
... model.add(Dense(10, activation='softmax'))
metrics=['accuracy'])
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=metrics)
model.summary()
Model: "sequential"
Layer (type)                 Output Shape              Param #
conv2d_6 (Conv2D)            (None, 10, 10, 32)       896
max_pooling2d_6 (MaxPooling2D) (None, 5, 5, 32)        0
conv2d_7 (Conv2D)            (None, 5, 5, 32)        9248
max_pooling2d_7 (MaxPooling2D) (None, 2, 2, 32)        0
flatten_7 (Flatten)          (None, 1152)           0
dense_9 (Dense)              (None, 128)            147584
dropout_8 (Dropout)          (None, 128)            0
dense_10 (Dense)             (None, 10)             100
...
Total params: 205,299
Trainable params: 147,693
Non-trainable params: 0

```

Gambar 8.100 Hasil Soal 10.

## 11. Soal 11

```

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

```

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

Gambar 8.101 Hasil Soal 11.

## 12. Soal 12

```

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

```

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

```
Epoch 0/10
1/10 [loss: 1.5816 - accuracy: 0.689] - val_loss: 1.0312 - val_accuracy: 0.7182
WARNING:tensorFlow:From C:\Users\A971\PycharmProjects\keras\mnist\mnist.py:245: The name tf.Graph is deprecated; please use tf.compat.v1.Graph instead.
Epoch 1/10 [loss: 1.0005 - accuracy: 0.7209] - val_loss: 0.9809 - val_accuracy: 0.7023
Epoch 2/10 [loss: 0.8891 - accuracy: 0.7477] - val_loss: 0.8552 - val_accuracy: 0.7962
Epoch 3/10 [loss: 0.8328 - accuracy: 0.7652] - val_loss: 0.8368 - val_accuracy: 0.7829
Epoch 4/10 [loss: 0.7755 - accuracy: 0.7755] - val_loss: 0.8479 - val_accuracy: 0.7838
Epoch 5/10 [loss: 0.7356 - accuracy: 0.7784] - val_loss: 0.8495 - val_accuracy: 0.7842
Epoch 6/10 [loss: 0.7041 - accuracy: 0.7819] - val_loss: 0.8506 - val_accuracy: 0.7834
Epoch 7/10 [loss: 0.6854 - accuracy: 0.7828] - val_loss: 0.8492 - val_accuracy: 0.7839
Epoch 8/10 [loss: 0.6755 - accuracy: 0.7835] - val_loss: 0.8491 - val_accuracy: 0.7841
Epoch 9/10 [loss: 0.6644 - accuracy: 0.7842] - val_loss: 0.8492 - val_accuracy: 0.7841
Epoch 10/10 [loss: 0.6584 - accuracy: 0.7849] - val_loss: 0.8492 - val_accuracy: 0.7841
Test loss: 0.658413881901203
Test accuracy: 0.784913880903545
```

Gambar 8.102 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 for conv2d_count in [1, 2]:
8     # menentukan ukuran besaran fixcel dari data atau konvert
9         # 1 fixcel mnjadi data yang berada pada codigan dibawah.
10        for dense_size in [128, 256, 512, 1024, 2048]:
11            # membuat looping untuk memangkas masing-masing data
12            # dengan ketentuan 0 persen 25 persen 50 persen dan 75
13            # persen .
14            for dropout in [0.0, 0.25, 0.50, 0.75]:
15                # membuat variabel model Sequential()
16                model = Sequential()
17                #membuat looping untuk variabel i dengan jarak
18                # dari hasil konvolusi .
19                for i in range(conv2d_count):
20                    # syarat jika i samadengan bobotnya 0
21                    if i == 0:
22                        # menambahkan method add pada variabel
23                        model dengan konvolusi 2 dimensi 32 bit didalamnya dan
24                        membuat kernel dengan ukuran 3 x 3 dan rumus aktifasi relu
25                        dan data shape yang di hitung dari data train .
26                        model.add(Conv2D(32, kernel_size=(3, 3),
27 activation='relu', input_shape=np.shape(train_input[0])))
28                        # jika tidak
29                    else:
30                        # menambahkan method add pada variabel
31                        model dengan konvolusi 2 dimensi 32 bit dengan ukuran
32                        kernel 3 x3 dan fungsi aktivasi relu
33                        model.add(Conv2D(32, kernel_size=(3, 3),
34 activation='relu'))
35                        # menambahkan method add pada variabel model
36                        dengan isian method Max pooling berdimensi 2 dengan
37                        ukuran fixcel 2 x 2 .
```

```

25         model.add(MaxPooling2D(pool_size=(2, 2)))
26         # merubah feature gambar menjadi 1 dimensi vektor
27         model.add(Flatten())
28         # menambahkan method dense untuk pemanfaatan 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                 # mengkompile 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_.%2f'
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 :

model count: 1, Dense size: 128, Dropout: 0.05	loss: 1.39, Accuracy: 0.74, Time: 453 sec
model count: 1, Dense size: 128, Dropout: 0.25	loss: 0.79, Accuracy: 0.79, Time: 447 sec
model count: 1, Dense size: 128, Dropout: 0.5	loss: 0.75, Accuracy: 0.77, Time: 454 sec
model count: 1, Dense size: 128, Dropout: 0.75	loss: 0.75, Accuracy: 0.77, Time: 454 sec
model count: 1, Dense size: 128, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 459 sec
model count: 1, Dense size: 256, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 461 sec
model count: 1, Dense size: 256, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 461 sec
model count: 1, Dense size: 256, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 461 sec
model count: 1, Dense size: 256, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 461 sec
model count: 1, Dense size: 512, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 466 sec
model count: 1, Dense size: 512, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 466 sec
model count: 1, Dense size: 512, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 466 sec
model count: 1, Dense size: 512, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 466 sec
model count: 1, Dense size: 1024, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 473 sec
model count: 1, Dense size: 1024, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 473 sec
model count: 1, Dense size: 1024, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 473 sec
model count: 1, Dense size: 1024, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 473 sec
model count: 1, Dense size: 2048, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 512 sec
model count: 1, Dense size: 2048, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 512 sec
model count: 1, Dense size: 2048, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 512 sec
model count: 1, Dense size: 2048, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 512 sec
model count: 1, Dense size: 4096, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 512 sec
model count: 1, Dense size: 4096, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 512 sec
model count: 1, Dense size: 4096, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 512 sec
model count: 1, Dense size: 4096, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 512 sec
model count: 2, Dense size: 128, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 538 sec
model count: 2, Dense size: 128, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 538 sec
model count: 2, Dense size: 128, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 538 sec
model count: 2, Dense size: 128, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 538 sec
model count: 2, Dense size: 256, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 543 sec
model count: 2, Dense size: 256, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 543 sec
model count: 2, Dense size: 256, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 543 sec
model count: 2, Dense size: 256, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 543 sec
model count: 2, Dense size: 512, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 554 sec
model count: 2, Dense size: 512, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 554 sec
model count: 2, Dense size: 512, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 554 sec
model count: 2, Dense size: 512, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 554 sec
model count: 2, Dense size: 1024, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 558 sec
model count: 2, Dense size: 1024, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 558 sec
model count: 2, Dense size: 1024, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 558 sec
model count: 2, Dense size: 1024, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 558 sec
model count: 2, Dense size: 2048, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 568 sec
model count: 2, Dense size: 2048, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 568 sec
model count: 2, Dense size: 2048, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 568 sec
model count: 2, Dense size: 2048, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 568 sec
model count: 2, Dense size: 4096, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 573 sec
model count: 2, Dense size: 4096, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 573 sec
model count: 2, Dense size: 4096, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 573 sec
model count: 2, Dense size: 4096, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 573 sec
model count: 3, Dense size: 128, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 581 sec
model count: 3, Dense size: 128, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 581 sec
model count: 3, Dense size: 128, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 581 sec
model count: 3, Dense size: 128, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 581 sec
model count: 3, Dense size: 256, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 586 sec
model count: 3, Dense size: 256, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 586 sec
model count: 3, Dense size: 256, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 586 sec
model count: 3, Dense size: 256, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 586 sec
model count: 3, Dense size: 512, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 596 sec
model count: 3, Dense size: 512, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 596 sec
model count: 3, Dense size: 512, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 596 sec
model count: 3, Dense size: 512, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 596 sec
model count: 3, Dense size: 1024, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 598 sec
model count: 3, Dense size: 1024, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 598 sec
model count: 3, Dense size: 1024, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 598 sec
model count: 3, Dense size: 1024, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 598 sec
model count: 3, Dense size: 2048, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 614 sec
model count: 3, Dense size: 2048, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 614 sec
model count: 3, Dense size: 2048, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 614 sec
model count: 3, Dense size: 2048, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 614 sec
model count: 3, Dense size: 4096, Dropout: 0.25	loss: 0.75, Accuracy: 0.79, Time: 616 sec
model count: 3, Dense size: 4096, Dropout: 0.5	loss: 0.75, Accuracy: 0.79, Time: 616 sec
model count: 3, Dense size: 4096, Dropout: 0.75	loss: 0.75, Accuracy: 0.79, Time: 616 sec
model count: 3, Dense size: 4096, Dropout: 0.95	loss: 1.12, Accuracy: 0.79, Time: 616 sec

Gambar 8.103 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()
5 # variabel model di tambahkan librari Conv2D tigapuluhan dua
6     bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
7     relu dang menggunakan data train_input
8 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
9     input_shape=np.shape(train_input[0])))
10 # variabel model di tambahkan dengan lib MaxPooling2D dengan
11     ketentuan ukuran 2 x 2 pixcel
12 model.add(MaxPooling2D(pool_size=(2, 2)))
13 # variabel model di tambahkan dengan librari Conv2D 32bit
14     dengan kernel 3 x 3
15 model.add(Conv2D(32, (3, 3), activation='relu'))
16 # variabel model di tambahkan dengan lib MaxPooling2D dengan
17     ketentuan ukuran 2 x 2 pixcel
18 model.add(MaxPooling2D(pool_size=(2, 2)))
19 # variabel model di tambahkan librari Flatten
20 model.add(Flatten())
21 # variabel model di tambahkan librari Dense dengan fungsi
22     tanh
23 model.add(Dense(128, activation='tanh'))
24 # variabel model di tambahkan librari dropout untuk memangkas
25     data tree sebesar 50 persen
26 model.add(Dropout(0.5))
27 # variabel model di tambahkan librari Dense dengan data dari
28     num_classes dan fungsi softmax
29 model.add(Dense(num_classes, activation='softmax'))
30 # mengkompile data model untuk mendapatkan data loss akurasi
31     dan optimasi
32 model.compile(loss='categorical_crossentropy', optimizer='adam',
33     metrics=['accuracy'])

```

```

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

```

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

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 36, 36, 32)	896
max_pooling2d_6 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_7 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_7 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_5 (Flatten)	(None, 1152)	0
dense_9 (Dense)	(None, 128)	147584
dropout_4 (Dropout)	(None, 128)	0
dense_10 (Dense)	(None, 369)	47681
Total params	153,329	
Trainable params:	265,329	
Non-trainable params:	0	
None		

Gambar 8.104 Hasil Soal 14.

## 15. Soal 15

```

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

```

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

```

In [18]: model.fit(np.concatenate((train_input, test_input)),
...             np.concatenate((train_output, test_output)),
...             batch_size= , epochs= , verbose= )
Epoch 1/10
- 12s - loss: 1.7795 - accuracy: 0.5871
Epoch 2/10
- 12s - loss: 1.0744 - accuracy: 0.7674
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: 0.8573 - accuracy: 0.7527
Epoch 6/10
- 11s - loss: 0.8297 - accuracy: 0.7586
Epoch 7/10
- 12s - loss: 0.8075 - accuracy: 0.7632
Epoch 8/10
- 12s - loss: 0.7917 - accuracy: 0.7663
Epoch 9/10
- 11s - loss: 0.7765 - accuracy: 0.7797
Epoch 10/10
- 11s - loss: 0.7637 - accuracy: 0.7718
Out[18]: <keras.callbacks.History at 0x21637ebc148>

```

Gambar 8.105 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 8.106** 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 8.107** 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 8.108 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 8.109 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: 4, confidence: 0.67
predict("HASYv2/hasy-data/v2-00500.png")
prediction: 191, confidence: 0.58
predict("HASYv2/hasy-data/v2-00700.png")
prediction: 19198, confidence: 0.88

```

Gambar 8.110 Hasil Soal 20.

## 8.5.3 Penanganan Error

## 1. KeyboardInterrupt

```

file = keras.preprocessing.image.ImageDataGenerator()
file = file.flow_from_directory('HASYv2/*' + rand(0))
file = file.next()
fp = buildfile('HASYv2/images/' + str(file[0][0]) + '.png', 'image/png', file[0][1])
fp = fp.open('rb')

```

Gambar 8.111 KeyboardInterrupt

## 2. Cara Penanganan Error

## ▪ KeyboardInterrupt

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

#### **8.5.4 Bukti Tidak Plagiat**



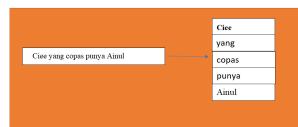
**Gambar 8.112** Bukti Tidak Melakukan Plagiat Chapter 7

## 8.6 1174073 - Ainul Filiani

### 8.6.1 Soal Teori

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

Karena MTOKENIZER merupakan proses membagi teks yang dapat berupa kalimat, paragraf atau dokumen menjadi kata-kata atau bagian-bagian tertentu dalam kalimat tersebut. Sebagai contoh dari kalimat "Cieeee yang copas punya ainul", kalimat itu menjadi beberapa bagian yaitu "cieeee", "yang", "copas", "punya", "ainul". Yang menjadi acuan yakni tanda baca dan spasi. Untuk ilustrasi, lihat gambar berikut:

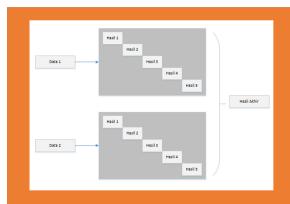


### Gambar 8.113 Teori 1

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

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

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

**Gambar 8.114** 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 8.115** 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:

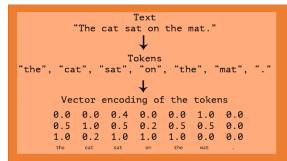
a	1
b	2
c	3
d	4
Name: 0, dtype: int64	

**Gambar 8.116** 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.

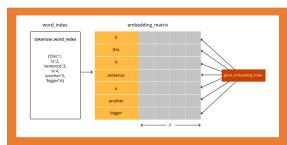
Fungsi tokenizer ini berfungsi untuk melakukan vektorisasi data kedalam bentuk token sebanyak 2000 kata. Dan selanjutnya akan melakukan fit tokenizer hanya untuk data training saja tidak dengan data testingnya. Untuk ilustrasi lihat gambar berikut:



**Gambar 8.117** Teori 5

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

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 8.118** Teori 6

- 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(6).reshape((3,2))
>>> a
array([[0, 1],
       [2, 3]])
>>> a.max() # 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(6, dtype=float)
>>> b
array([ 0.,  1.,  2.,  3.,  4.,  5.])
>>> npamax(b)
5.0
>>> npamax(b, where=(np.isnan(b), initial=-1))
4.0
>>> np.sum(b)
15.0

```

### Gambar 8.119 Teori 7

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

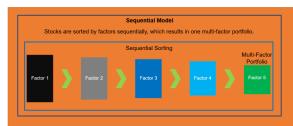
Fungsi dari baris kode tersebut ialah membuat train outputs dengan kategori dari class lalu dengan ketentuan iloc train idx. Kemudian membuat keluaran sebagai output. Untuk ilustrasi, lihat gambar berikut:

```
>>> V_train
array([[ 0.,  0.,  0.],
       [ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 1.,  1.,  0.],
       [ 0.,  0., -int64(4)]])
>>> V_train[[1, 0, 0], [0, 1, 0], [1, 1, 0], [0, 0, -int64(4)]]
>>> V_train[[1, 0, 0], [0, 1, 0], [1, 1, 0], [0, 0, -int64(4)]]
>>> np.allclose(V_train, V_train)
array([ True,  True,  True,  True])
```

### Gambar 8.120 Teori 8

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

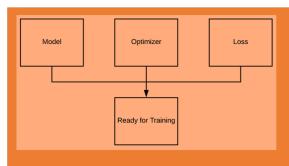
Fungsi dari baris kode tersebut ialah model perlu mengetahui bentuk input apa yang harus diharapkan. Untuk alasan ini, lapisan pertama dalam model Sequential (dan hanya yang pertama, karena lapisan berikut dapat melakukan inferensi bentuk otomatis) perlu menerima informasi tentang bentuk inputnya.



### Gambar 8.121 Teori 9

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

Fungsi dari baris kode tersebut ialah bisa meneruskan nama fungsi loss yang ada, atau melewati fungsi simbolis TensorFlow yang mengembalikan skalar untuk setiap titik data dan mengambil dua argumen `y_true`: True label, dan `y_pred`: Prediksi. Tujuan yang dioptimalkan sebenarnya adalah rata-rata dari array output di semua titik data.



**Gambar 8.122** Teori 10

11. Jelaskan apa itu Deep Learning.

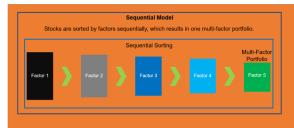
Deep Learning adalah salah satu cabang dari ilmu machine learning yang terdiri dari algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam machine learning dapat digunakan baik untuk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

1174073 mod3+1 x 1174073 mod3+1 = 2 x 2, adapun ilustrasi gambar nya sebagai berikut :



Gambar 8.123 Teori 9

### 8.6.2 Praktek Program

#### 1. Soal 1

```

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

```

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



Gambar 8.124 Hasil Soal 1.

#### 2. Soal 2

```

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

```

```

11         pil_image.open("HASYv2/" + row[0]))
12     # neuron activation functions behave best when
13     input values are between 0.0 and 1.0 (or -1.0 and 1.0),
14     # so we rescale each pixel value to be in the
15     range 0.0 to 1.0 instead of 0-255
16     img /= 255.0
17     imgs.append((row[0], row[2], img))
18     classes.append(row[2])
19     i += 1

```

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

**Gambar 8.125** Hasil Soal 2.

### 3. Soal 3

```

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

```

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

**Gambar 8.126** Hasil Soal 3.

### 4. Soal 4

```

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

```

```

7 train_output = np.asarray(list(map(lambda row: row[1], train)))
8 test_output = np.asarray(list(map(lambda row: row[1], test)))

```

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

```

In [8]: 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 8.127** Hasil Soal 4.

## 5. Soal 5

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

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

```

In [4]: from sklearn.preprocessing import LabelEncoder
.... from sklearn.preprocessing import OneHotEncoder

```

**Gambar 8.128** Hasil Soal 5.

## 6. Soal 6

```

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

```

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

```

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

```

**Gambar 8.129** Hasil Soal 6.

## 7. Soal 7

```

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

```

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

```

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

```

**Gambar 8.130** Hasil Soal 7.

## 8. Soal 8

```

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

```

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

```

In [9]: 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),
num_classes = len(label_encoder.classes_))
Number of classes: 39

```

**Gambar 8.131** Hasil Soal 8.

## 9. Soal 9

```

1 # In [9]: import sequential

```

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

```

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

```

**Gambar 8.132** Hasil Soal 9.

## 10. Soal 10

```

1 # In [10]: desain jaringan
2 model = tf.keras.Sequential()
3 model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
4                               activation='relu',
5                               input_shape=np.shape(train_input[0])))
6 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
7 model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu'))
8 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
9 model.add(tf.keras.layers.Flatten())
10 model.add(tf.keras.layers.Dense(1024, activation='tanh'))
11 model.add(tf.keras.layers.Dropout(0.5))
12 model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
13 model.compile(loss='categorical_crossentropy', optimizer='adam',
14                 metrics=['accuracy'])
15
16 print(model.summary())

```

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

```

model.add(conv2d_1, input_shape=(28, 28, 1), activation='relu')
model.add(maxpooling2d_1(pool_size=(2, 2)))
model.add(conv2d_2, (3, 3), activation='relu')
model.add(maxpooling2d_2(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1024, activation='tanh'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
print(model.summary())
Model: "sequential"
_________________________________________________________________
Layer (Type)                Output Shape             Param #
conv2d_1 (Conv2D)            (None, 14, 14, 32)      896
max_pooling2d_1 (MaxPooling2D) (None, 7, 7, 32)       0
conv2d_2 (Conv2D)            (None, 5, 5, 32)       9248
max_pooling2d_2 (MaxPooling2D) (None, 3, 3, 32)       0
conv2d_3 (Conv2D)            (None, 3, 3, 32)       9248
max_pooling2d_3 (MaxPooling2D) (None, 1, 1, 32)       0
flatten_1 (Flatten)          (None, 32)              0
dense_9 (Dense)              (None, 128)             417584
dropout_4 (Dropout)          (None, 32)              0
dense_10 (Dense)             (None, 69)              47601
_________________________________________________________________
Total params: 295,329
Trainable params: 295,329
Non-trainable params: 0

```

**Gambar 8.133** Hasil Soal 10.

## 11. Soal 11

```

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

```

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

```
(In [12]: keras.callbacks
tensorboard = keras.callbacks.TensorBoard(log_dir='/log/mathsymbols')
)
```

Gambar 8.134 Hasil Soal 11.

## 12. Soal 12

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

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

```
Epoch 1/10
- 375s - loss: 1.5816 - accuracy: 0.6283 - val_loss: 1.0532 - val_accuracy: 0.7932
Epoch 2/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 1.0532 - val_loss: 0.7932]
Epoch 3/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.8552 - val_loss: 0.7932]
Epoch 4/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.8552 - val_loss: 0.7932]
Epoch 5/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.7624 - val_loss: 0.7932]
Epoch 6/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.7624 - val_loss: 0.7932]
Epoch 7/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.7248 - val_loss: 0.7932]
Epoch 8/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.7248 - val_loss: 0.7932]
Epoch 9/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.7056 - val_loss: 0.7932]
Epoch 10/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 11/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 12/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 13/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 14/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 15/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 16/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 17/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 18/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 19/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 20/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 21/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 22/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 23/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 24/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 25/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 26/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 27/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 28/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 29/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 30/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 31/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 32/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 33/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 34/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 35/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 36/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 37/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 38/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 39/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 40/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 41/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 42/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 43/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 44/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 45/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 46/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 47/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 48/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 49/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
Epoch 50/10: 100%|██████████| 375/375 [00:00<00:00, 10.00s/epoch - loss: 0.6552 - val_loss: 0.7932]
```

Gambar 8.135 Hasil Soal 12.

## 13. Soal 13

```
1 model.add(tf.keras.layers.Flatten())
2 model.add(tf.keras.layers.Dense(128, activation='tanh'))
3 model.add(tf.keras.layers.Dropout(0.5))
4 model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
5 model.compile(loss='categorical_crossentropy',
6                 optimizer='adam', metrics=['accuracy'])
7 print(model.summary())
8 # In[15]: join train and test data so we train the network on
9 # all data we have available to us
10 model.fit(np.concatenate((train_input, test_input)),
11           np.concatenate((train_output, test_output)),
12           batch_size=32, epochs=10, verbose=2)
13
14 # In[16]: save the trained model
15 model.save("mathsymbols.model")
```

```

16 # In[17]: save label encoder (to reverse one-hot encoding)
17 np.save('classes.npy', label_encoder.classes_)
18
19
20 # In[18]: load the pre-trained model and predict the math
21     symbol for an arbitrary image;
22 # the code below could be placed in a separate file
23
24 model2 = keras.models.load_model("mathsymbols.model")
25 print(model2.summary())
26
27 # In[19]: restore the class name to integer encoder
28 label_encoder2 = LabelEncoder()
29 label_encoder2.classes_ = np.load('classes.npy')
30
31 def predict(img_path):
32     newimg = keras.preprocessing.image.img_to_array(pil_image
33         .open(img_path))
34     newimg /= 255.0
35
36     # do the prediction
37     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
38
39     # figure out which output neuron had the highest score,
40     # and reverse the one-hot encoding
41     inverted = label_encoder2.inverse_transform(
42         [np.argmax(prediction)]) # argmax finds highest-
43         scoring output
44     print("Prediction: %s, confidence: %.2f" %
45         (inverted[0], np.max(prediction)))
46
47
48 # In[20]: grab an image (we'll just use a random training
49     # image for demonstration purposes)
50 predict("HASYv2/hasy-data/v2-00010.png")
51
52 predict("HASYv2/hasy-data/v2-00500.png")
53
54 predict("HASYv2/hasy-data/v2-00700.png")

```

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

**Gambar 8.136** 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 model = tf.keras.Sequential()
3 model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
4                                 activation='relu',
5                                 input_shape=np.shape(train_input[0])))
6 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
7 model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu'))
8 model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
9 model.add(tf.keras.layers.Flatten())
10 model.add(tf.keras.layers.Dense(128, activation='tanh'))
11 model.add(tf.keras.layers.Dropout(0.5))
12 model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
13 model.compile(loss='categorical_crossentropy',
14                 optimizer='adam', metrics=['accuracy'])
15 print(model.summary())

```

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

```

model.add(Dense(100, input_shape=(100, )))
model.add(Activation('relu'))
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(100, activation='tanh'))
model.add(Dense(100, activation='softmax'))
model.add(MultiClassLoss('categorical_crossentropy', optimizer='adam',
    metrics=['accuracy']))
model.summary()
Model: "sequential_1"

```

layer (Type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 36, 36, 32)	896
max_pooling2d_6 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_7 (Conv2D)	(None, 15, 15, 32)	9488
max_pooling2d_7 (MaxPooling2D)	(None, 15, 15, 32)	0
flatten_3 (Flatten)	(None, 1152)	0
dense_9 (Dense)	(None, 128)	147584
dropout_4 (Dropout)	(None, 118)	0
dense_10 (Dense)	(None, 369)	47681

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

None

Gambar 8.137 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 model.fit(np.concatenate((train_input, test_input)),
3             np.concatenate((train_output, test_output)),
4             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
- 126s - loss: 1.7799 - accuracy: 0.5871
Epoch 2/10
- 126s - loss: 1.0744 - accuracy: 0.7074
Epoch 3/10
- 126s - loss: 0.9564 - accuracy: 0.7320
Epoch 4/10
- 115s - loss: 0.8977 - accuracy: 0.7458
Epoch 5/10
- 118s - loss: 0.8574 - accuracy: 0.7527
Epoch 6/10
- 118s - loss: 0.8299 - accuracy: 0.7586
Epoch 7/10
- 126s - loss: 0.8079 - accuracy: 0.7632
Epoch 8/10
- 122s - loss: 0.7917 - accuracy: 0.7663
Epoch 9/10
- 122s - loss: 0.7765 - accuracy: 0.7707
Epoch 10/10
- 118s - loss: 0.7637 - accuracy: 0.7718
Out[16]: <keras.callbacks.History at 0x21637ebc348>

```

**Gambar 8.138** Hasil Soal 15.

## 16. Soal 16

```

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

```

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

```

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

```

**Gambar 8.139** Hasil Soal 16.

## 17. Soal 17

```

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

```

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

```

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

```

**Gambar 8.140** Hasil Soal 17.

## 18. Soal 18

```

1 # In[18]: load the pre-trained model and predict the math
      symbol for an arbitrary image;
2 # the code below could be placed in a separate file
3
4 model2 = keras.models.load_model("mathsymbols.model")
5 print(model2.summary())

```

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

Layer (type)	Output shape	Param #
conv2d_68 (Conv2D)	(None, 10, 10, 32)	896
max_pooling2d_68 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_69 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_69 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_45 (Flatten)	(None, 1152)	0
dense_89 (Dense)	(None, 128)	147584
dropout_34 (Dropout)	(None, 128)	0
dense_90 (Dense)	(None, 369)	47601
Total params: 205,329		
Trainable params: 205,329		
Non-trainable params: 0		
None		

Gambar 8.141 Hasil Soal 18.

## 19. Soal 19

```

1 # In[19]: restore the class name to integer encoder
2 label_encoder2 = LabelEncoder()
3 label_encoder2.classes_ = np.load('classes.npy')
4
5
6 def predict(img_path):
7     newimg = keras.preprocessing.image.img_to_array(pil_image
8         .open(img_path))
9     newimg /= 255.0
10
11     # do the prediction
12     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
13
14     # figure out which output neuron had the highest score,
15     # and reverse the one-hot encoding
16     inverted = label_encoder2.inverse_transform(
17         [np.argmax(prediction)]) # argmax finds highest-
scoring output
18     print("Prediction: %s, confidence: %.2f" %
19           (inverted[0], np.max(prediction)))

```

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

Layer (type)	Output Shape	Params #
conv2d_68 (Conv2D)	(None, 10, 10, 32)	896
max_pooling2d_68 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_69 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_69 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_45 (Flatten)	(None, 1152)	0
dense_89 (Dense)	(None, 128)	147584
dropout_34 (Dropout)	(None, 128)	0
dense_90 (Dense)	(None, 369)	47681
<b>Total params:</b>	<b>205,129</b>	
<b>Trainable params:</b>	<b>205,129</b>	
<b>Non-trainable params:</b>	<b>0</b>	

**Gambar 8.142** 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 predict("HASYv2/hasy-data/v2-00010.png")  
4 predict("HASYv2/hasy-data/v2-00500.png")  
5 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("mnistv2/hazy-data/v2-00010.png")

Prediction: A, confidence: 0.87

predict("mnistv2/hazy-data/v2-00500.png")

Prediction: A!, confidence: 0.58

predict("mnistv2/hazy-data/v2-00700.png")

Prediction: Alpha, confidence: 0.68
```

**Gambar 8.143** Hasil Soal 20.

### 8.6.3 Penanganan Error

## 1. KeyboardInterrupt

```
File "/c/Users/Alain/Veconda/lib/site-packages/PIL/Image.py", line 2409, in open
    fp = builtins.open(filename, "rb")

KeyboardInterrupt
```

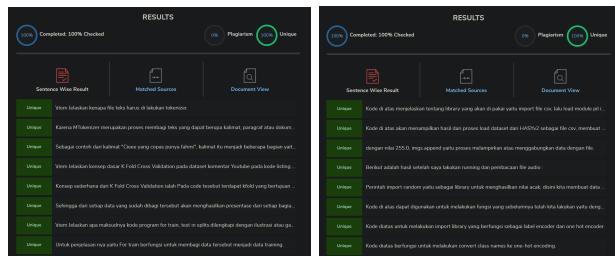
**Gambar 8.144** KeyboardInterrupt

## 2. Cara Penanganan Error

- KeyboardInterrupt

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

## 8.6.4 Bukti Tidak Plagiat



**Gambar 8.145**   Bukti Tidak Melakukan Plagiat Chapter 7

## 8.7 1174054 - Aulyardha Anindita

### 8.7.1 Teori

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

Kata pada suatu teks disebut token. Proses vektorisasi dari bentuk kata kedalam token biasa disebut dengan tokenizer dan tokenizer tersebut akan merubah suatu teks tersebut menjadi simbol, kata, ataupun biner dan bentuk lainnya kedalam token.



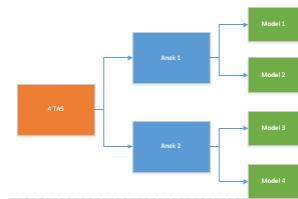
**Gambar 8.146**   Tokenizer

2. Jelaskan konsep dasar K Fold Cross Validation pada dataset komentar Youtube pada kode listing 7.1.dilengkapi dengan ilustrasi atau gambar. StartifieldKFold berisi presentasi sampel dalam setiap kelas, dimana pada ilustrasi ini sampel tersebut dibagi menjadi 5 bagian dalam setiap class-nya, lalu sampel tersebut dimasukkan kedalam class dari dataset youtube tersebut.

**Gambar 8.147** K-Fold Cross Validation

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

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

**Gambar 8.148** Maksud Kode dor train, test in splits

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

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

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

**Gambar 8.149** Maksud Kode train content

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

Tokenizer(num words=2000) digunakan untuk membaca kalimat yang

telah dibuat menjadi token sebanyak 2000 kata tokenizer.fit on texts(train content) digunakan untuk membuat membaca data token teks yang telah dimasukkan kedalam fungsi yaitu fungsi train konten



**Gambar 8.150** Maksud dari fungsi Tokenizer

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

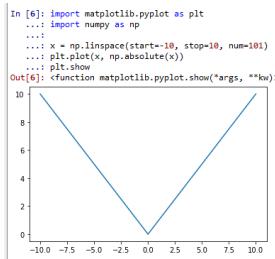
Maksudnya yaitu untuk variabel d train inputs akan melakukan tokenizer dari bentuk teks kematrix dari data train content dengan mode matriksnya yaitu tfidf begitu juga dengan variabel d test inputs untuk data test. berikut gambar ilustrasinya:



**Gambar 8.151** Maksud Kode train inputs

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

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

**Gambar 8.152** Maksud Kode train inputs

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

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

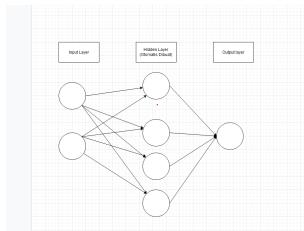
```
In [10]: labels = [0,2,1,2,0,1]
...: keras.utils.to_categorical(labels)Out[11]:
array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.]], dtype=float32)
```

**Gambar 8.153** Maksud Kode train outputs

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

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

Untuk ilustrasinya bisa dilihat pada gambar berikut :

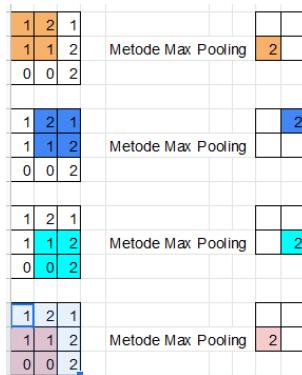


**Gambar 8.154** Ilustrasi Neural Network

10. Jelaskan apa maksud dari fungsi di listing 7.3 dengan parameter tersebut. Melakukan peng compile-an dari model Sequential tadi dengan Loss yang merupakan fungsi optimisasi skor menggunakan categorical crossentropy, dan menggunakan algoritma adam sebagai optimizer. Adam yaitu algoritma pengoptimalan yang dapat digunakan sebagai ganti dari prosedur penurunan gradien stokastik klasik untuk memperbarui bobot jaringan yang berulang berdasarkan data training. Dengan metrik yaitu fungsi yang digunakan untuk menilai kinerja mode Anda disini menggunakan fungsi accuracy.
11. Jelaskan apa itu Deep Learning  
Deep learning merupakan salah satu algoritma yang seperti Neural Network yang menggunakan meta data sebagai inputan dan mengolahnya menggunakan layer layer yang tersembunyi.
12. Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning  
Deep Neural Network adalah jaringan syaraf tiruan (JST) dengan beberapa lapisan antara lapisan input dan output. DNN menemukan manipulasi matematis yang benar untuk mengubah input menjadi output, apakah itu hubungan linear Deep Neural Network adalah jaringan syaraf tiruan (JST) dengan beberapa lapisan antara lapisan input dan output. DNN menemukan manipulasi matematis yang benar untuk mengubah input menjadi 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 lapisan yaitu input dan output, sedangkan dalam Deep learning kita dapat mendefinisikan layer sebanyak yang kita inginkan atau butuhkan.
13. Jelaskan dengan ilustrasi gambar buatan sendiri (langkah per langkah) bagaimana perhitungan algoritma kobvolasi dengan ukuran stride (NPM)

$\text{mod3+1}) \times (\text{NPM mod3+1})$  yang terdapat pada max pooling

Karena hasil dari  $(\text{NPM mod3+1}) \times (\text{NPM mod3+1}) = 2 \times 2 = 4$ , berarti ada 4 metode max pooling, sebelum membuat ilustrasi perlu diketahui apa itu stride, stride adalah acuan atau parameter yang menentukan pergeseran pada filter fixcel, sebagai contoh nilai stride 1 yang berarti filter akan bergeser sebanyak satu fixcel secara vertikal dan horizontal. selanjutnya apa itu max pooling contoh pada suatu gambar ditentukan max pooling dari  $3 \times 3$  dengan stride 1 yang berarti setiap pergeseran 1 pixcel akan diambil nilai terbesar dari pixcel  $3 \times 3$  tersebut.



**Gambar 8.155** Illustrasi perhitungan stride 1 max pooling

### 8.7.2 Praktek

#### 1. Nomor 1

```

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

```

Dan hasilnya bisa dilihat pada gambar berikut :

```

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

```

**Gambar 8.156** Hasil Nomor 1

## 2. Nomor 2

```

1 # In[2]:load all images (as numpy arrays) and save their
   classes
2 #Menginisiasi variabel imgs dan classes dengan variabel array
   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.
8     reader = 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 0
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

```

Untuk hasil plotnya bisa dilihat pada gambar berikut:

```

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

```

**Gambar 8.157** Hasil Nomor

## 3. Nomor 3

```

1 # In[3]:shuffle the data, split into 80% train , 20% test
2 # Melakukan import library random
3 import random

```

```

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

```

Hasilnya dapat dilihat pada gambar berikut:

```

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

```

**Gambar 8.158** Hasil Nomor 3

#### 4. Nomor 4

```

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

```

Untuk hasilnya bisa dilihat pada gambar berikut :

```

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

```

**Gambar 8.159** Hasil Nomor 4

#### 5. Nomor 5

```

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

```

Untuk hasilnya bisa dilihat pada gambar berikut :

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

**Gambar 8.160** Hasil Nomor 5

## 6. Nomor 6

```

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

```

Hasilnya bisa dilihat pada gambar berikut :

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

**Gambar 8.161** Hasil Nomor 6

## 7. Nomor 7

```

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

```

Hasilnya bisa dilihat pada gambar berikut :

```
In [7]: onehot_encoder = OneHotEncoder(handle_unknown='ignore')
...: integer_encoder = IntegerEncoder(handle_unknown='ignore')(integer_encoded), 1)
...: onehot_encoder.fit(integer_encoded)
Out[7]: OneHotEncoder(categories='auto', sparse=False)
Warning: FutureWarning: If integer-encoding, please use encoder.get_dummies(). In future versions of
the library, if integer-encoding, the categories will be determined based on the range [0, max(values)], while in the future they will be
determined based on the unique values in the input.
If you want the future behaviour and silence this warning, you can specify
"categories='auto'" or "categories='all'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories
to integers, you can skip the integer-encoder.
warnings.warn(msg, FutureWarning)
out[7]:
OneHotEncoder(categories='auto', categories=None,
              dtype='class', handle_unknown='error',
              n_values=None, sparse=False)
```

**Gambar 8.162** Hasil Nomor 7

## 8. Nomor 8

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

Untuk hasilnya bisa dilihat pada gambar berikut :

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

**Gambar 8.163** Hasil Nomor 8

## 9. Nomor 9

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

Hasilnya bisa dilihat pada gambar berikut :

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

**Gambar 8.164** Hasil Nomor 9

## 10. Nomor 10

```
1 # In[10]: desain jaringan
2 # Menginisiasi variabel model dengan isian library Sequential
3 model = Sequential()
4 # variabel model di tambahkan library Conv2D tigapuluhan dua
    bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
    relu dan menggunakan data train_input
5 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
6                 input_shape=np.shape(train_input[0])))
7 # variabel model di tambahkan dengan lib MaxPooling2D dengan
    ketentuan ukuran 2 x 2 pixcel
8 model.add(MaxPooling2D(pool_size=(2, 2)))
9 # variabel model di tambahkan dengan library Conv2D 32bit
    dengan kernel 3 x 3
10 model.add(Conv2D(32, (3, 3), activation='relu'))
11 # variabel model di tambahkan dengan lib MaxPooling2D dengan
    ketentuan ukuran 2 x 2 pixcel
12 model.add(MaxPooling2D(pool_size=(2, 2)))
13 # variabel model di tambahkan library Flatten
14 model.add(Flatten())
15 # variabel model di tambahkan library Dense dengan fungsi
    tanh
16 model.add(Dense(1024, activation='tanh'))
17 # variabel model di tambahkan library dropout untuk memangkas
    data tree sebesar 50 persen
18 model.add(Dropout(0.5))
19 # variabel model di tambahkan library Dense dengan data dari
    num_classes dan fungsi softmax
20 model.add(Dense(num_classes, activation='softmax'))
21 # mengkompile data model untuk mendapatkan data loss akurasi
    dan optimasi
22 model.compile(loss='categorical_crossentropy', optimizer='
    adam',
                metrics=['accuracy'])
23 # mencetak variabel model kemudian memunculkan kesimpulan
    berupa data total parameter, trainable paremeter dan bukan
    trainable parameter
24 print(model.summary())
25
```

Hasilnya bisa dilihat pada gambar berikut :

**Gambar 8.165** Hasil Nomor 10

## 11. Nomor 11

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

Hasilnya bisa dilihat pada gambar berikut :

```
In [11]: import keras.callbacks
```

Gambar 8.166 Hasil Nomor 11

## 12. Nomor 12

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

Hasilnya bisa dilihat pada gambar berikut :

```
In [12]: model.fit(train_input, train_output,
...:     epochs=100,
...:     epochs=100,
...:     verbose=2,
...:     validation_split=0.2,
...:     callbacks=[tensorboard])
...:
...: score = model.evaluate(test_input, test_output, verbose=2)
...: print('Test accuracy:', score[1])
WARNING:tensorflow:tf.math.reduce_mean is deprecated and will be removed in a future version.
In [13]: tensorboard --logdir ./logs
Use tf.compat.v1 instead.
Train on 2032 samples, validate on 2032 samples
Epoch 1/18
- 30s - loss: 0.5334 - acc: 0.6294 - val_loss: 0.9824 - val_acc: 0.7285
Epoch 2/18
- 30s - loss: 0.3064 - acc: 0.7322 - val_loss: 0.8182 - val_acc: 0.7494
Epoch 3/18
- 30s - loss: 0.0943 - acc: 0.7596 - val_loss: 0.8493 - val_acc: 0.7599
Epoch 4/18
- 30s - loss: 0.7584 - acc: 0.7790 - val_loss: 0.8441 - val_acc: 0.7666
Epoch 5/18
- 30s - loss: 0.7584 - acc: 0.7790 - val_loss: 0.8441 - val_acc: 0.7665
Epoch 6/18
- 30s - loss: 0.6936 - acc: 0.7899 - val_loss: 0.8522 - val_acc: 0.7692
Epoch 7/18
- 30s - loss: 0.6591 - acc: 0.7942 - val_loss: 0.8522 - val_acc: 0.7698
Epoch 8/18
- 30s - loss: 0.6591 - acc: 0.7942 - val_loss: 0.8500 - val_acc: 0.7654
Epoch 9/18
- 30s - loss: 0.6544 - acc: 0.8027 - val_loss: 0.8496 - val_acc: 0.7638
Epoch 10/18
- 30s - loss: 0.6470 - acc: 0.8072 - val_loss: 0.8597 - val_acc: 0.7638
Epoch 11/18
- 30s - loss: 0.5989 - acc: 0.8118 - val_loss: 0.8380 - val_acc: 0.7637
Test loss: 0.8794284677359739
Test accuracy: 0.793281179238488
```

**Gambar 8.167** Hasil Nomor 12

### 13. Nomor 13

```
1 # In[13]:try various model configurations and parameters to
2 # find the best
3 # Melakukan import library time
4 import time
5 #Menginisiasi variabel result dengan array kosong
6 results = []
7 # melakukan looping dengan ketentuan konvolusi 2 dimensi 1 2
8 for conv2d_count in [1, 2]:
9     # menentukan ukuran besaran fixcel dari data atau konvert
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            # Menginisiasi 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                    # Penambahan 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                    # 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(Conv2D(32, kernel_size=(3, 3),
35 activation='relu'))
36                    # Penambahan method add pada variabel model
37                    dengan isian method Max pooling berdimensi 2 dengan
38                    ukuran fixcel 2 x 2.
```

```

25         model.add(MaxPooling2D(pool_size=(2, 2)))
26         # merubah feature gambar menjadi 1 dimensi vektor
27         model.add(Flatten())
28         # Penambahan 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             # Penambahan method dropout pada model dengan
35             nilai dari dropout
36             model.add(Dropout(dropout))
37             # Penambahan method dense dengan fungsi num
38             classss dan rumus softmax
39             model.add(Dense(num_classes, activation='softmax',
40 ))
41             # mongkompile 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_.%2f'
47             , % (conv2d_count, dense_size, dropout)
48             # Menginisiasi variabel tensorboard dengan isian
49             dari library keras dan nilai dari lig dir
50             tensorboard = keras.callbacks.TensorBoard(log_dir
51 =log_dir)
52             # Menginisiasi variabel start dengan isian dari
53             library time menggunakan method time
54
55             start = time.time()
56             # Penambahan method fit pada model dengan data
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             # Menginisiasi variabel score dengan nilai
65             evaluasi dari model menggunakan data tes input dan tes
66             output
67             score = model.evaluate(test_input, test_output,
68             verbose=2)
69             # Menginisiasi variabel end
70             end = time.time()
71             # Menginisiasi variabel elapsed
72             elapsed = end - start
73             # mencetak hasil perhitungan
74             print("Conv2D count: %d, Dense size: %d, Dropout:
75             %.2f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" %
76             (conv2d_count, dense_size, dropout, score[0], score[1],
77             elapsed))
78             results.append((conv2d_count, dense_size, dropout
79             , score[0], score[1], elapsed))

```

Hasilnya bisa dilihat pada gambar berikut :

```

    ...
    model.compile(loss='categorical_crossentropy', optimizer='adam',
                  metrics=['accuracy'])
    ...
    log_dir = './logs/conv2d_3d-dense_3d-dropout_N_2f' % (conv2d_count,
    tensorboard = keras.callbacks.TensorBoard(log_dir=log_dir))
    ...
    start = time.time()
    model.fit(x_train, y_train, batch_size=32, epochs=10,
    verbose=0, validation_split=0.2, callbacks=[tensorboard])
    ...
    score = model.evaluate(x_test, y_test, verbose=0)
    end = time.time()
    elapsed = end - start
    print('Time: %s sec' % elapsed)
    print('Conv2D count: %d, Dense size: %d, Dropout: %s, Loss: %.2f - Loss: %.2f - Accuracy: %.2f' %
    [conv2d_count, dense_size, dropout, score[0], score[1], elapsed])
    results.append(conv2d_count, dense_size, dropout, score[0],
    score[1], elapsed)
    Conv2D count: 1, Dense size: 128, Dropout: 0.80 - Loss: 1.13, Accuracy: 0.76, Time: 1548
    ...
    Conv2D count: 1, Dense size: 128, Dropout: 0.25 - Loss: 0.93, Accuracy: 0.76, Time: 1579
    ...
    Conv2D count: 1, Dense size: 128, Dropout: 0.50 - Loss: 0.81, Accuracy: 0.77, Time: 1599
    ...
    Conv2D count: 1, Dense size: 128, Dropout: 0.75 - Loss: 0.82, Accuracy: 0.77, Time: 1627

```

Gambar 8.168 Hasil Nomor 13

#### 14. Nomor 14

```

1 # In[14]: rebuild/retrain a model with the best parameters (
2     # from the search) and use all data
3 # Menginisiasi variabel model dengan isian library Sequential
4 model = Sequential()
5 # variabel model di tambahkan library Conv2D tigapuluhan dua
6     # bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
7     # relu dang menggunakan data train_input
8 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
9                 input_shape=np.shape(train_input[0])))
10 # variabel model di tambahkan dengan lib MaxPooling2D dengan
11     # ketentuan ukuran 2 x 2 pixcel
12 model.add(MaxPooling2D(pool_size=(2, 2)))
13 # variabel model di tambahkan dengan library Conv2D 32bit
14     # dengan kernel 3 x 3
15 model.add(Conv2D(32, (3, 3), activation='relu'))
16 # variabel model di tambahkan dengan lib MaxPooling2D dengan
17     # ketentuan ukuran 2 x 2 pixcel
18 model.add(MaxPooling2D(pool_size=(2, 2)))
19 # variabel model di tambahkan library Flatten
20 model.add(Flatten())
21 # variabel model di tambahkan library Dense dengan fungsi
22     # tanh
23 model.add(Dense(128, activation='tanh'))
24 # variabel model di tambahkan library dropout untuk memangkas
25     # data tree sebesar 50 persen
26 model.add(Dropout(0.5))
27 # variabel model di tambahkan library Dense dengan data dari
28     # num_classes dan fungsi softmax
29 model.add(Dense(num_classes, activation='softmax'))
30 # mengkompile data model untuk mendapatkan data loss akurasi
31     # dan optimasi
32 model.compile(loss='categorical_crossentropy', optimizer='adam',
33                 metrics=['accuracy'])
34 # mencetak variabel model kemudian memunculkan kesimpulan
35     # berupa data total parameter, trainable paremeter dan bukan
36     # trainable parameter
37 print(model.summary())

```

Hasilnya bisa dilihat pada gambar berikut :

```
In [16]: model = Sequential()
...:     ...: model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
...: input_shape=(train_input[0].shape[1:2])))
...:     ...: model.add(Flatten())
...:     ...: model.add(Dense(128, activation='relu'))
...:     ...: model.add(Dense(128, activation='tanh'))
...:     ...: model.add(Dropout(0.5))
...:     ...: model.add(Dense(10, activation='softmax'))
...:     ...: model.compile(loss='categorical_crossentropy', optimizer='adam',
...: metrics=['accuracy'])
...:     ...: print(model.summary())
...:
```

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_3 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_4 (Conv2D)	(None, 13, 13, 32)	9248
max_pooling2d_4 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_2 (Flatten)	(None, 1152)	0
dense_3 (Dense)	(None, 128)	147584
dropout_3 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 369)	47681

```
Total params: 265,329
Trainable params: 265,329
Non-trainable params: 0
None
```

Gambar 8.169 Hasil Nomor 14

## 15. Nomor 15

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

Hasilnya bisa dilihat pada gambar berikut :

```
In [26]: model.fit(np.concatenate((train_input, test_input)),
...:               np.concatenate((train_output, test_output)),
...:               batch_size=32, epochs=10, verbose=2)
Epoch 1/10
- 247s - loss: 1.7949 - acc: 0.5843
Epoch 2/10
- 238s - loss: 1.0797 - acc: 0.7064
Epoch 3/10
- 238s - loss: 0.9468 - acc: 0.7388
Epoch 4/10
- 237s - loss: 0.9060 - acc: 0.7434
Epoch 5/10
- 238s - loss: 0.8671 - acc: 0.7519
Epoch 6/10
- 236s - loss: 0.8362 - acc: 0.7573
Epoch 7/10
- 238s - loss: 0.8137 - acc: 0.7631
Epoch 8/10
- 239s - loss: 0.7988 - acc: 0.7652
Epoch 9/10
- 239s - loss: 0.7831 - acc: 0.7692
Epoch 10/10
- 239s - loss: 0.7695 - acc: 0.7724
Out[26]: <keras.callbacks.History at 0x14c1941f5f8>
```

Gambar 8.170 Hasil Nomor 15

## 16. Nomor 16

```
1 # In [16]: save the trained model
```

```

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

```

Hasilnya bisa dilihat pada gambar berikut :

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

**Gambar 8.171** Hasil Nomor 16

17. Nomor 17

```

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

```

Hasilnya bisa dilihat pada gambar berikut :

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

**Gambar 8.172** Hasil Nomor 17

18. Nomor 18

```

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

```

Hasilnya bisa dilihat pada gambar berikut :

```

In [24]: import keras.models
...: model2 = keras.models.load_model("mathsymbols.model")
...: print(model2.summary())
Layer (Type)          Output Shape       Param #
=================================================================
conv2d_3 (Conv2D)     (None, 30, 32)    896
max_pooling2d_3 (MaxPooling2D) (None, 15, 15, 32) 0
conv2d_4 (Conv2D)     (None, 13, 13, 32) 9248
max_pooling2d_4 (MaxPooling2D) (None, 6, 6, 32) 0
flatten_2 (Flatten)  (None, 1152)      0
dense_3 (Dense)      (None, 128)       147584
dropout_2 (Dropout)  (None, 128)       0
dense_4 (Dense)      (None, 369)       47601
=================================================================
Total params: 205,329
Trainable params: 205,329
Non-trainable params: 0
None

```

**Gambar 8.173** Hasil Nomor 18

## 19. Nomor 19

```

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

```

Hasilnya bisa dilihat pada gambar berikut :

```

In [20]: label_encoder2 = LabelEncoder()
...: label_encoder2.classes_ = np.load('classes.npy')
...: def predict(img_path):
...:     newimg =
keras.preprocessing.image.img_to_array(pil_image.open(img_path))
...:     newimg /= 255.0
...:     # do the prediction
...:     prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
...:     # figure out which output neuron had the highest score , and reverse the one-hot encoding
...:     inverted = label_encoder2.inverse_transform([np.argmax(prediction)])
...:     # Inverted = label_encoder2.inverse_transform(np.argmax(prediction)) #
...:     print("Prediction: %s, confidence: %.2f" % (inverted[0],
...: np.max(prediction)))

```

**Gambar 8.174** Hasil Nomor 19

## 20. Nomor 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")

```

Hasilnya bisa dilihat pada gambar berikut :

```

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

```

**Gambar 8.175** Hasil Nomor 20

### 8.7.3 Penanganan Error

#### 1. ScreenShoot Error

```

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

```

**Gambar 8.176** Module Not Found Error

#### 2. Tuliskan Kode Error dan Jenis Error

- Module Not Found Error

#### 3. Cara Penanganan Error

- Module Not Found Error

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

### 8.7.4 Bukti Tidak Plagiat



**Gambar 8.177** Bukti Plagiasrisme

### **8.7.5 Link Youtube**

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

