

BAB 1

CHAPTER 1

BAB 2

CHAPTER 2

BAB 3

CHAPTER 3

BAB 4

CHAPTER 4

BAB 5

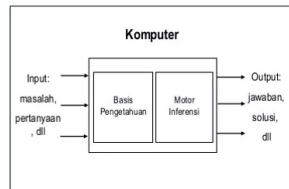
CHAPTER 4

5.1 1174080 Handi Hermawan

5.1.1 Teori

1. Jelaskan apa itu klasifikasi teks, sertakan gambar ilustrasi buatan sendiri.

Klasifikasi teks merupakan sebuah model yang biasa digunakan untuk untuk mengkategorikan sebuah teks ke dalam kelompok-kelompok yang lebih terorganisir. Jadi untuk setiap kalimat yang di masukan ke dalam mesin, mesin tersebut akan menjadikan setiap kata dari kalimat tersebut menjadi sebuah kolom. Untuk ilustrasinya bisa dilihat pada gambar berikut :



Gambar 5.1 Klasifikasi teks.

2. Jelaskan mengapa hal ini bisa terjadi, klasifikasi bunga tidak bisa digunakan untuk machine learning, sertakan ilustrasi gambar sendiri.

Karena machine learning tidak dapat menampilkan inputan sesuai dengan apa yang kita inputkan. Karena inputan tersebut serupa namun mesin memberikan output yang berbeda, biasanya output atau error ini disebut dengan istilah noise. Untuk contoh sederhananya misalkan kita inputkan salah satu label yang terdapat pada bunga, output yang dihasilkan oleh mesin tersebut ialah label yang lain. Itu dikarenakan bunga banyak jenis yang serupa namun tidak sama. Untuk ilustrasinya bisa dilihat pada gambar berikut :



Gambar 5.2 Klasifikasi Bunga.

3. Jelaskan bagaimana yang dimaksud dengan teknik pembelajaran mesin pada teks yang digunakan dan sertakan ilustrasi buatan sendiri.

Teknik yang digunakan pada youtube salah satunya ialah keywords. Dengan keywords tersebut mesin dapat memberikan video sesuai dengan keyword yang kita inputkan pada kolom pencarian. Teknik pembelajarannya tergantung user memberikan input teks seperti apa, karena pada youtube itu sendiri akan menyesuaikan dengan apa yang biasa kita inputkan dan akan memfilter video secara otomatis sesuai dengan keyword yang biasa kita inputkan. Contoh ilustrasi sederhananya seperti berikut :



Gambar 5.3 Klasifikasi teks Youtube.

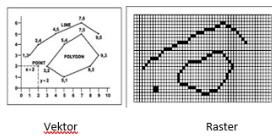
4. Jelaskan apa yang dimaksud vektorisasi data.

Vektorisasi adalah proses konversi data raster menjadi data vektor yang

lebih umum disebut dengan istilah digitalisasi adapun aktifitasnya disebut digitasi. Wujud digitalisasi ini diklasifikasikan secara spesifik dalam tema-tema tertentu yang direpresentasikan oleh bentuk garis, poligon dan titik. Pada akhirnya proses vektorisasi ini menghasilkan suatu wujud peta topografi yang menggambarkan keadaan permukaan bumi atau bentang alam. Sifat data yang geometris menunjukkan ukuran dimensi yang sesungguhnya. contoh sebagai berikut :

5. Jelaskan apa yang dimaksud dengan bag of words dengan ilustrasi sendiri.

Representasi penyederhanaan sebuah kalimat atau perhitungan setiap kata pada suatu kalimat dengan presentase berapa kali muncul kata tersebut untuk setiap kalimatnya. Contoh ilustrasi sederhananya seperti berikut :



Gambar 5.4 Decision Tree.

6. Jelaskan apa yang dimaksud dengan TF-IDF.

TF-IDF merupakan metode untuk menghitung bobot setiap kata pada suatu kalimat yang paling sering digunakan. TF atau term frequency adalah weighting scheme yang digunakan untuk menentukan relevansi dokumen dengan sebuah query (term). TF menentukan bobot relevansi sebuah dokumen dan term berdasarkan frekuensi kemunculan term pada dokumen terkait. Untuk menghitung TF terdapat beberapa jenis fungsi yang dapat digunakan. IDF adalah inverse dari DF, IDF akan melakukan proses scaling pada TF. Term yang memiliki DF yang rendah akan memiliki IDF yang tinggi. Dengan kata lain, sebuah term yang jarang ditemui pada koleksi dokumen atau bisa dikatakan sebagai term khusus akan memiliki nilai IDF yang tinggi. Contohnya :

Contoh TF

Kasus: hitungan weight kata "anak" pada kalimat "anak ini bukan anak kamu"

Kasus: hitungan weight kata "anak" pada dokumen: anak ini bukan anak kamu.

term: anak

weight t pada d: 2

$$tf(t, d) = \begin{cases} x(n), & \text{if } f_{t,d} > 0 \\ x(n-1), & \text{otherwise} \end{cases}$$

Ru

Maka IDF dari term "olahraga" dan "taman"

Term	DF	IDF
olahraga	3	0
taman	1	0.477121

$$idf_t = \log\left(\frac{N}{df_t}\right)$$

Gambar 5.5 TF-IDF.

5.1.2 Praktek Program

1. Soal 1

```
1 %% Soal 1
2 import pandas as pd #digunakan untuk mengimport library
    pandas dengan alias pd
3 pd = pd.read_csv("csv.csv") #membaca file csv
```

Kode di atas digunakan untuk buat aplikasi sederhana menggunakan pandas dengan format csv sebanyak 500 baris, hasilnya ialah sebagai berikut :

Gambar 5.6 Hasil Soal 1.

2. Soal 2

```
1 %% Soal 2
2 love=pd[:450] #membagi data training menjadi 450
3 fixs=pd[450:] #membagi data menjadi 450 atau sisa dari data
    yang tersedia
```

Kode di atas digunakan untuk memecah dataframe tersebut menjadi dua bagian yaitu 450 row pertama dan 50 row kedua. Hasilnya adalah sebagai berikut :

Gambar 5.7 Hasil Soal 2.

3. Soal 3

```
1 %% Soal 3
2 import pandas as hans #untuk import library pandas berguna
    untuk mengelola dataframe
```

```

3 hans = hans.read_csv("Youtube03-LMFAO.csv") #membaca file
   dengan format csv
4
5 spam=hans.query('CLASS == 1') #membagi tabel spam
6 nospam=hans.query('CLASS == 0')#membagi tabel no spam
7
8 from sklearn.feature_extraction.text import CountVectorizer #untuk import countvectorizer berfungsi untuk memecah data tersebut menjadi sebuah kata yang lebih sederhana
9 vectorizer = CountVectorizer () #ntuk menjalankan fungsi tersebut, pada code ini tidak ada hasilnya dikarenakan spyder tidak mendukung hasil dari instasiasi.
10
11 dvec = vectorizer.fit_transform(hans['CONTENT']) #untuk melakukan pemecahan data pada dataframe yang terdapat pada kolom konten
12 dvec #Untuk menampilkan hasil dari code sebelumnya
13
14 Daptarkata= vectorizer.get_feature_names()
15
16 dshuf = hans.sample(frac=1)
17
18 love=dshuf[:300]
19 fixs=dshuf[300:]
20
21 love_att = vectorizer.fit_transform(love['CONTENT'])
22 love_att
23
24 love_label=love['CLASS']
25 fixs_label=fixs['CLASS']

```

Dengan menggunakan $1174080 \bmod 4$ ialah 1, artinya disini saya memakai dataset LMFAO. Hasilnya adalah sebagai berikut :

4. Soal 4

```

1 %% Soal 4
2
3 from sklearn import svm
4 clfsvm = svm.SVR(gamma = 'auto')
5 clfsvm.fit(love_att , love_label)

```

Klasifikasi dari data vektorisasi menggunakan klasifikasi Decision Tree. Hasilnya adalah sebagai berikut :

5. Soal 5

```

1 %%soal 5
2
3 from sklearn import tree
4 clftree = tree.DecisionTreeClassifier()
5 clftree.fit(love_att , love_label)

```

Plotlah confusion matrix dari praktik modul ini menggunakan matplotlib. Hasilnya adalah sebagai berikut :

6. Soal 6

```

1 %%soal 6
2
3 from sklearn.metrics import confusion_matrix
4 pred_labels=clftree.predict(fixs)
5 cm=confusion_matrix(fixs_label,pred_labels)
6
7 %%
8
9 import matplotlib.pyplot as plt
10
11 def plot_confusion_matrix(cm, classes,
12                           normalize=False,
13                           title='Confusion matrix',
14                           cmap=plt.cm.Blues):
15
16     if normalize:
17         cm = cm.astype('float') / cm.sum(axis=1)[:, np.
newaxis]
18         print("Normalized confusion matrix")
19     else:
20         print('Confusion matrix, without normalization')
21
22     print(cm)

```

Menjalankan program cross validation. Hasilnya adalah sebagai berikut :

```

In [11]: import matplotlib.pyplot as plt
...:
...: def plot_confusion_matrix(cm, classes,
...:                           normalize=False,
...:                           title='Confusion matrix',
...:                           cmap=plt.cm.Blues):
...:
...:     if normalize:
...:         cm = cm.astype('float') / cm.sum(axis=1)[:, np.
...:newaxis]
...:         print("Normalized confusion matrix")
...:     else:
...:         print('Confusion matrix, without normalization')
...:
...:     print(cm)
...:     cm

```

Gambar 5.8 Hasil Soal 6.

7. Soal 7

```

1 %%soal 7
2
3 from sklearn.model_selection import cross_val_score
4
5 scores=cross_val_score(clftree,love_att,love_label, cv=5)
6
7 skor_rata2=scores.mean()
8 skoresd=scores.std()

```

Tree.export graphviz merupakan fungsi yang menghasilkan representasi Graphviz dari decision tree, yang kemudian ditulis ke outfile. Disini akan menyimpan classifiernya, akan meng ekspor file student performance jika salah akan mengembalikan nilai fail. Hasilnya adalah sebagai berikut :

```
In [12]: from sklearn.model_selection import cross_val_score
...:
scores=cross_val_score(clftree,d_train_att,d_train_label,cv=5)
...:
skor_rata2=scores.mean()
...:
skoresd=scores.std()
```

Gambar 5.9 Hasil Soal 7.

8. Soal 8

```
1 #%%soal 8
2
3 max_features_opts = range(5, 50, 5) #max_features_opts
4         sebagai variabel untuk membuat range 5,50,5
5 n_estimators_opts = range(10, 200, 20) #n_estimators_opts
6         sebagai variabel untuk membuat range 10,200,20
7 rf_params = hans.empty((len(max_features_opts)*len(
8     n_estimators_opts),4), float) #rf_params sebagai variabel
9         untuk menjumlahkan yang sudah di tentukan sebelumnya
10 i = 0
11 for max_features in max_features_opts: #pengulangan
12     for n_estimators in n_estimators_opts: #pengulangan
13         clftree = RandomForestClassifier(max_features=
14             max_features, n_estimators=n_estimators) #menampilkan
15             variabel csf
16             scores = cross_val_score(clf, df_train_att,
17             df_train_label, cv=5) #scores sebagai variabel training
18             rf_params[i,0] = max_features #index 0
19             rf_params[i,1] = n_estimators #index 1
20             rf_params[i,2] = scores.mean() #index 2
21             rf_params[i,3] = scores.std() * 2 #index 3
22             i += 1 #dengan ketentuan i += 1
23             print("Max features: %d, num estimators: %d, accuracy
24 : %.02f (+/- %.02f)" %(max_features, n_estimators, scores.
25 mean(), scores.std() * 2))
26             #print hasil pengulangan yang sudah ditentukan
```

Buatlah program pengamatan komponen informasi. Jadi disini kita akan memprediksi nilai dari variabel test att dan test pass Hasilnya adalah sebagai berikut :

Gambar 5.10 Hasil Soal 8.

5.1.3 Penanganan Error

1. ScreenShoot Error

```
file "<ipython>-input-14-"
cb9b9919846", line 14
    print("Max features: %d, num
estimators: %d, accuracy: %.2f (+-
%.2f)" %
^
SyntaxError: unexpected character after
line continuation character
```

Gambar 5.11 SyntaxError

2. Cara Penangan Error

- **SyntaxError**

Error terdapat pada codingan yang tidak terdefenisikan, alhasil ialah saya mencoba codingan pada chapter 3 dan hasilnya berhasil.

5.1.4 Bukti Tidak Plagiat



Gambar 5.12 Bukti Tidak Melakukan Plagiat Chapter 4

5.1.5 Link Video Youtube

<https://youtu.be/ZXjjIdXNbMc>

BAB 6

CHAPTER 5

|||||| HEAD

BAB 7

CHAPTER 6

BAB 8

CHAPTER 7

8.1 1174006 - Kadek Diva Krishna Murti

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

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



Gambar 8.1 Kecerdasan Buatan.

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

8.1.1 Teori

8.1.2 Praktek

8.1.3 Penanganan Error

8.1.4 Bukti Tidak Plagiat



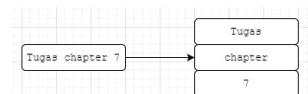
Gambar 8.2 Kecerdasan Buatan.

8.2 1174066 - D.Irga B. Naufal Fakhri

8.2.1 Teori

8.2.1.1 Kenapa file teks harus di lakukan tokenizer

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



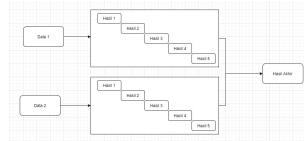
Gambar 8.3 Teori 1

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

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

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

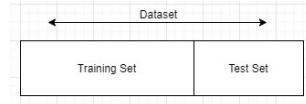
terdapat kfold yang bertujuan untuk melakukan split data menjadi 5 bagian dari dataset komentar Youtube tersebut. Sehingga dari setiap data yang sudah dibagi tersebut akan menghasilkan persentase dari setiap bagiannya, untuk menghasilkan hasil akhir dengan persentase yang cukup baik.



Gambar 8.4 Teori 2

8.2.1.3 Apa maksudnya kode program for train, test in splits

For train berfungsi untuk membagi data tersebut menjadi data training. Sedangkan test in splits berfungsi untuk menguji apakah dataset tersebut sudah dibagi menjadi beberapa bagian atau masih menumpuk.



Gambar 8.5 Teori 3

8.2.1.4 Apa maksudnya kode program train content = d['CONTENT'].iloc[train_idx] dan test content = d['CONTENT'].iloc[test_idx]

Fungsi dalam kode tersebut berfungsi untuk mengambil data pada kolom atau index CONTENT yang merupakan bagian dari train_idx dan test_idx. Contoh sederhananya ketika data telah diubah menjadi data train dan data test maka kita dapat memilihnya untuk ditampilkan pada kolom yang diinginkan.

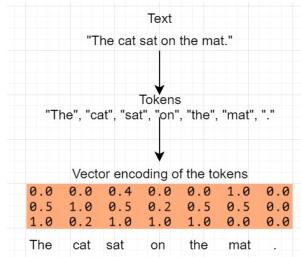
```
1 import pandas as pd
2
3 mydict = [ { 'a': 1, 'b': 2, 'c': 3, 'd': 4 },
4           { 'a': 100, 'b': 200, 'c': 300, 'd': 400 },
5           { 'a': 1000, 'b': 2000, 'c': 3000, 'd': 4000 } ]
6 df = pd.DataFrame(mydict)
7 df
```

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

Gambar 8.6 Teori 4

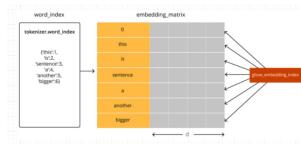
8.2.1.5 Apa maksud dari fungsi $\text{tokenizer} = \text{Tokenizer}(\text{num_words}=2000)$ dan $\text{tokenizer.fit on texts}(\text{train content})$

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

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 $d \text{ train inputs} = d \text{ train inputs}/\text{np.amax}(\text{np.absolute}(d \text{ train inputs}))$ dan $d \text{ test inputs} = d \text{ test inputs}/\text{np.amax}(\text{np.absolute}(d \text{ test inputs}))$

Fungsi np.amax 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.

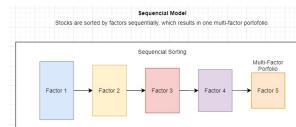
```
>>> v_train
array([[1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.]], dtype=int64)
>>> v_train.flatten()
array([1, 0, 0, ..., 1, 0, 0], dtype=int64)
>>> v_train
array([[1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
   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

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

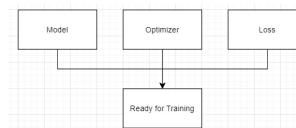
8.2.1.10 Apa maksud dari fungsi di listing 7.3 dengan parameter tersebut

```

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 OneHotEncoder
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
from tensorflow.keras import regularizers
from tensorflow import keras
from tensorflow.keras import callbacks
tensorboard = keras.callbacks.TensorBoard(log_dir='/content/logs/mnist-style')
```

Gambar 8.24 Hasil No 11

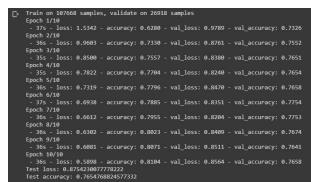
8.2.2.12 Nomor 12

```
1 model.fit(train_input, train_output, #Fungsi model ditambahkan
           fungsi fit untuk mengetahui perhitungan dari train_input
           train_output
           batch_size=32, #Dengan batch size 32 bit
           epochs=10,
           verbose=2,
           validation_split=0.2,
           callbacks=[tensorboard])
```

7

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

Hasil:



Gambar 8.25 Hasil No 12

8.2.2.13 Nomor 13

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

```

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

```

Hasil:

Gambar 8.26 Hasil No 13

8.2.2.14 Nomor 14

```
1 model = Sequential() #Membuat variabel model dengan isian library  
2 Sequential  
3 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',  
4 input_shape=np.shape(train_input[0])))#Variabel model di  
5 tambahkan library Conv2D tigapuluhan dua bit dengan ukuran  
6 kernel 3 x 3 dan fungsi penghitungan relu yang menggunakan  
7 data train_input  
8 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di  
9 tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x  
10 2 pixel  
11 model.add(Conv2D(32, (3, 3), activation='relu')) #Variabel model  
12 di tambahkan dengan library Conv2D 32bit dengan kernel 3 x 3  
13 model.add(MaxPooling2D(pool_size=(2, 2))) #Variabel model di  
14 tambahkan dengan lib MaxPooling2D dengan ketentuan ukuran 2 x  
15 2 pixcel  
16 model.add(Flatten()) #Variabel model di tambahkan library Flatten  
17 model.add(Dense(128, activation='tanh')) #Variabel model di  
18 tambahkan library Dense dengan fungsi tanh  
19 model.add(Dropout(0.5)) #Variabel model di tambahkan library  
20 dropout untuk memangkas data tree sebesar 50 persen  
21 model.add(Dense(num_classes, activation='softmax')) #Variabel  
22 model di tambahkan library Dense dengan data dari num_classes  
23 dan fungsi softmax  
24 model.compile(loss='categorical_crossentropy', optimizer='adam',  
25 metrics=['accuracy']) #Mengcompile data model untuk  
26 mendapatkan data loss akurasi dan optimasi  
27 print(model.summary()) #Mencetak variabel model kemudian  
28 memunculkan kesimpulan berupa data total parameter, trainable  
29 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)
          ┌─────────────────────────────────────────────────────────────────────────┐
          │ 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)         47081    │
          │                                                               ──────────────────────────────────────────────────┘
          ┌─────────────────────────────────────────────────────────────────┐
          │ Total params: 285,370                                     │
          │ Trainable params: 285,370                                │
          │ Non-trainable params: 0                                  │
          └─────────────────────────────────────────────────────────┘
          None
```

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:

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.
   open(img_path)) #Membuat variabel newimg dengan membuat
   imimage 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 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('dataset/class.csv')) #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
    inverted = label_encoder.inverse_transform([np.argmax(prediction)]) #membuat
    print("Prediction: %s, confidence: %.2f" % (inverted[0], np.max(prediction)))
```

Gambar 8.32 Hasil No 19

8.2.2.20 Nomor 20

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

Hasil:

```
► Prediction: A, confidence: 0.74
► Prediction: \pi, confidence: 0.48
► Prediction: \alpha, confidence: 0.90
```

Gambar 8.33 Hasil No 20

8.2.3 Penanganan Error

8.2.3.1 Error

- ProfilerNotRunningError

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

Gambar 8.34 ProfilerNotRunningError

8.2.3.2 Solusi Error

- ProfilerNotRunningError

tambahkan kode profile=10000000 pada parameter log_dir

8.2.4 Bukti Tidak Plagiat



Gambar 8.35 Bukti tidak plagiat

8.2.5 Link Youtube

https://youtu.be/Vq_LZ89hTpq

8.3 1174080 - Handi Hermawan

8.3.1 Soal Teori

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

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

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

Gambar 8.36 Teori 1

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

```

1
2 kfold = StratifiedKFold(n_splits=5)

```

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

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

Gambar 8.37 Teori 2

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

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

**Gambar 8.38** Teori 3

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

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

```

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

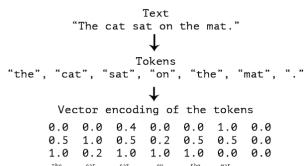
```

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

Gambar 8.39 Teori 4

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

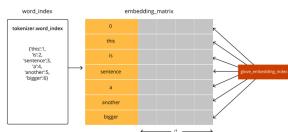
Fungsi tokenizer ini berfungsi untuk melakukan vektorisasi data kedalam bentuk token sebanyak 2000 kata. Dan selanjutnya akan melakukan fit tokenizer hanya untuk data training saja tidak dengan data testingnya. Untuk ilustrasi lihat gambar berikut:



Gambar 8.40 Teori 5

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

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



Gambar 8.41 Teori 6

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

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

```

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

```

Gambar 8.42 Teori 7

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

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

```

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

```

Gambar 8.43 Teori 8

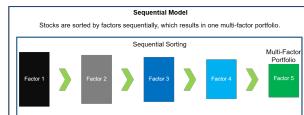
9. Jelaskan apa maksud dari fungsi di listing 7.2. Gambarkan ilustrasi Neural Network nya dari model kode tersebut.

```

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

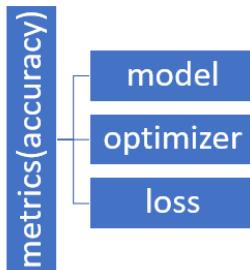
10. Jelaskan apa maksud dari fungsi di listing 7.3 dengan parameter tersebut.

```

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.

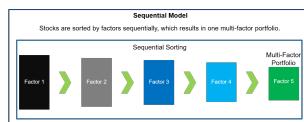
12. Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning.

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 librari Image yang berguna untuk dari PIL atau Python
# imaging Library yang berguna untuk mengolah data berupa gambar
from PIL import Image as pil_image
# kemudian mengimport librari keras yang menggunakan method preprocessing yang
# digunakan untuk membuat neutal network
import keras.preprocessing.image

```

Gambar 8.47 Hasil Soal 1.

2. Soal 2

```

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

```

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

3. Soal 3

```

1 # In[3]: shuffle the data, split into 80% train , 20% test
2 # mengimport library random
3 import random
4 # melakukan random pada vungsi imgs
5 random.shuffle(imgs)
6 # membuat variabel split_idx dengan nilai integer 80 persen
7 # dikali dari pengembalian jumlah dari variabel imgs

```

```

7 split_idx = int(0.8* len(imgs))
8 # membuat variabel train dengan isi lebih besar split_idx
9 train = imgs[:split_idx]
10 # membuat variabel test dengan isi lebih kecil split_idx
11 test = imgs[split_idx:]

```

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

```

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

```

Gambar 8.48 Hasil Soal 3.

4. Soal 4

```

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

```

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

```

In [5]: import numpy as np
.... train_input = np.asarray(list(map(lambda row: row[2], train)))
.... test_input = np.asarray(list(map(lambda row: row[2], test)))
.... train_output = np.asarray(list(map(lambda row: row[1], train)))
.... test_output = np.asarray(list(map(lambda row: row[1], test)))

```

Gambar 8.49 Hasil Soal 4.

5. Soal 5

```

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

```

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

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

Gambar 8.50 Hasil Soal 5.

6. Soal 6

```

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

```

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

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

Gambar 8.51 Hasil Soal 6.

7. Soal 7

```

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

```

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

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

Gambar 8.52 Hasil Soal 7.

8. Soal 8

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

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

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

num_classes
399
399 classes for X 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',
                metrics=['accuracy'])
23 # mencetak variabel model kemudian memunculkan kesimpulan
    berupa data total parameter , trainable paremeter dan bukan
    trainable parameter
24 print(model.summary())
```

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

```

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

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

```

Gambar 8.55 Hasil Soal 10.

11. Soal 11

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

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

```
In [12]: import keras.callbacks  
.... tensorboard = keras.callbacks.TensorBoard(log_dir='./Logs/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: 0.8685 - accuracy: 0.7494 - val_loss: 0.8462 - 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: 0.8685 - accuracy: 0.7494 - val_loss: 0.8462 - val_accuracy: 0.7929
3/18 - loss: 0.8685 - accuracy: 0.7494 - val_loss: 0.8462 - 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: 0.6495 - accuracy: 0.7912 - val_loss: 0.8462 - val_accuracy: 0.7939
11/18 - loss: 0.6495 - accuracy: 0.7912 - val_loss: 0.8462 - val_accuracy: 0.7939
12/18 - loss: 0.6495 - accuracy: 0.7912 - val_loss: 0.8462 - val_accuracy: 0.7939
13/18 - loss: 0.6495 - accuracy: 0.7912 - val_loss: 0.8462 - val_accuracy: 0.7939
14/18 - loss: 0.6495 - accuracy: 0.7912 - val_loss: 0.8462 - val_accuracy: 0.7939
15/18 - loss: 0.6495 - accuracy: 0.7912 - val_loss: 0.8462 - val_accuracy: 0.7939
16/18 - loss: 0.6495 - accuracy: 0.7912 - val_loss: 0.8462 - val_accuracy: 0.7939
17/18 - loss: 0.6495 - accuracy: 0.7912 - val_loss: 0.8462 - val_accuracy: 0.7939
18/18 - loss: 0.6495 - accuracy: 0.7912 - val_loss: 0.8462 - 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'))
35                    # menambahkan method add pada variabel model
36                    dengan isian method Max pooling berdimensi 2 dengan
37                    ukuran fixcel 2 x 2.
38                    model.add(MaxPooling2D(pool_size=(2, 2)))
39                    # merubah feature gambar menjadi 1 dimensi vektor
40                    model.add(Flatten())

```

```

28         # menambahkan method dense untuk pemanjangan data
29         dengan ukuran dense di tentukan dengan rumus fungsi tanh.
30         model.add(Dense(dense_size , activation='tanh'))
31         # membuat ketentuan jika pemangkasan lebih besar
32         dari 0 persen
33         if dropout > 0.0:
34             # menambahkan method dropout pada model
35             dengan nilai dari dropout
36             model.add(Dropout(dropout))
37             # menambahkan method dense dengan fungsi num
38             classss dan rumus softmax
39             model.add(Dense(num_classes , activation='softmax',
40             ))
41             # 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_%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(Flatten())
--> model.add(Dense(128, activation='relu'))
--> 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, 38, 38, 32)       896
max_pooling2d_6 (MaxPooling2D) (None, 19, 19, 32)        0
conv2d_7 (Conv2D)            (None, 13, 13, 32)       9248
max_pooling2d_7 (MaxPooling2D) (None, 6, 6, 32)        0
flatten_5 (Flatten)          (None, 1152)             0
dense_9 (Dense)              (None, 128)              147584
dropout_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: 1.0744 - 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, 16, 16, 32)	896
max_pooling2d_68 (MaxPooling)	(None, 15, 15, 32)	0
conv2d_69 (Conv2D)	(None, 15, 15, 32)	9248
max_pooling2d_69 (MaxPooling)	(None, 6, 6, 32)	0
flatten_45 (Flatten)	(None, 1152)	0
dense_89 (Dense)	(None, 128)	147584
dropout_34 (Dropout)	(None, 128)	0
dense_90 (Dense)	(None, 369)	47601
<hr/>		
Total params: 205,329		
Trainable params: 205,329		
Non-trainable params: 0		
<hr/>		
None		

Gambar 8.64 Hasil Soal 19.

20. Soal 20

```

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

```

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

```

# grab an image (we'll just use a random training image for demonstration purposes)
predict("HASYv2/hasy-data/v2-00010.png")
prediction: 42, confidence: 0.07
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 8.65 Hasil Soal 20.

8.3.3 Penanganan Error

1. KeyboardInterrupt

```

file = keras.preprocessing.image.ImageDataGenerator()
file = file.flow_from_directory("HASYv2/* + rnb(0))
```

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> =====

BAB 9

CHAPTER 6

9.1 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 9.1 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.

9.1.1 Teori

9.1.2 Praktek

9.1.3 Penanganan Error

9.1.4 Bukti Tidak Plagiat



Gambar 9.2 Kecerdasan Buatan.

9.2 1174087 - Ilham Muhammad Ariq

9.2.1 Teori

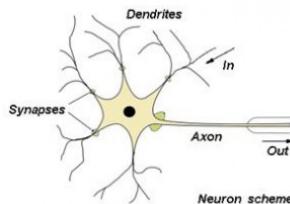
1. Jelaskan kenapa file suara harus di lakukan MFCC. dilengkapi dengan ilustrasi atau gambar.

Nilai-nilai MFCC meniru pendengaran manusia dan mereka biasanya digunakan dalam aplikasi pengenalan suara serta genre musik deteksi. Nilai-nilai MFCC ini akan dimasukkan langsung kejaringan saraf. Agar dapat diubah menjadi bentuk vektor, dan dapat digunakan pada machine learning. Disebabkan machine learning hanya mengerti bilangan vektor saja.

Ilustrasinya, Ketika ingin menggunakan file suara dalam machine learning. Machine learning tidak memahami rekaman suara melainkan vektor. Maka rekaman tersebut akan diubah kedalam bentuk vektor kemudian vektor akan menyesuaikan dengan kata-kata yang sudah disediakan. Jika cocok maka akan mengembalikan waktu yang diinginkan

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

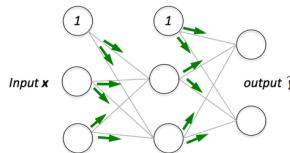
Neural Network ini terinspirasi dari jaringan saraf otak manusia. Dimana setiap neuron terhubung ke setiap neuron di lapisan berikutnya. Lapisan pertama menerima input dan lapisan terakhir memberikan keluaran. Struktur jaringan, yang berarti jumlah neuron dan koneksinya, diputuskan sebelumnya dan tidak dapat berubah, setidaknya tidak selama training. Juga, setiap input harus memiliki jumlah nilai yang sama. Ini berarti bahwa gambar, misalnya, mungkin perlu diubah ukurannya agar sesuai dengan jumlah neuron input.



Gambar 9.3 Contoh Pembobotan Neural Network

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

Bobot mewakili kekuatan koneksi antar unit. Jika bobot dari node 1 ke node 2 memiliki besaran lebih besar, itu berarti bahwa neuron 1 memiliki pengaruh lebih besar terhadap neuron 2. Bobot penting untuk nilai input. Bobot mendekati nol berarti mengubah input ini tidak akan mengubah output. Bobot negatif berarti meningkatkan input ini akan mengurangi output. Bobot menentukan seberapa besar pengaruh input terhadap output. Seperti contoh contoh berikut :



Gambar 9.4 Contoh Pembobotan Neural Network

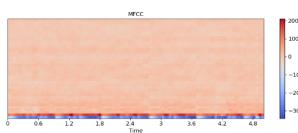
4. Jelaskan konsep fungsi aktifasi dalam neural network. dilengkapi dengan ilustrasi atau gambar

Fungsi aktivasi digunakan untuk memperkenalkan non-linearitas ke jaringan saraf. Ini menekan nilai dalam rentang yang lebih kecil yaitu. fungsi aktivasi Sigmoid memeras nilai antara rentang 0 hingga 1. Ada banyak

fungsi aktivasi yang digunakan dalam industri pembelajaran yang dalam dan ReLU, SELU dan TanH lebih disukai daripada fungsi aktivasi sigmoid. Ilustrasinya, ketika fungsi aktivasi linier, jaringan saraf dua lapis mampu mendekati hampir semua fungsi. Namun, jika fungsi aktivasi identik dengan fungsi aktivasi $F(X) = X$, properti ini tidak puas, dan jika MLP menggunakan fungsi aktivasi yang sama, seluruh jaringan se-tara dengan jaringan saraf lapis tunggal.

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

Berikut merupakan hasil plot dari rekaman suara :



Gambar 9.5 Cara Membaca Hasil Plot MFCC

Dari gambar tersebut dapat diketahui :

- Terdapat 2 dimensi yaitu x sebagai waktu, dan y sebagai power atau desibel.
- Dapat dilihat bahwa jika berwarna biru maka power dari suara tersebut rendah, dan jika merah power dari suara tersebut tinggi
- Dibagian atas terdapat warna pudar yang menandakan bahwa tidak ada suara sama sekali dalam jangkauan tersebut.

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

One-hot encoding adalah representasi variabel kategorikal sebagai vektor biner. Mengharuskan nilai kategorikal dipetakan ke nilai integer. Kemudian, setiap nilai integer direpresentasikan sebagai vektor biner yang semuanya bernilai nol kecuali indeks integer, yang ditandai dengan 1.

Label Encoding			→	One Hot Encoding		
Food Name	Categorical #	Calories		Apple	Chicken	Broccoli
Apple	1	95		1	0	0
Chicken	2	231		0	1	0
Broccoli	3	50		0	0	1

Gambar 9.6 One Hot Encoding

- Jelaskan apa fungsi dari np/.unique dan to categorical dalam kode program,dilengkapi dengan ilustrasi atau gambar.

Untuk np.unique fungsinya yaitu menemukan elemen unik array. Mengembalikan elemen unik array yang diurutkan. Ada tiga output opsional selain elemen unik:

- Indeks array input yang memberikan nilai unik
- Indeks array unik yang merekonstruksi array input
- 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 9.7 Numpy Unique

Untuk To Categorical fungsinya untuk mengubah vektor kelas (integer) ke matriks kelas biner.

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

Gambar 9.8 To Categorical

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

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 9.9 Sequential

9.2.2 Praktek Program

1. 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 cara mengimport library yang dibutuhkan dan membuat fungsi display mfcc untuk melakukan plot pada file audio nanti. Isi data GTZAN adalah datasets lagu atau suara yang terdiri dari 10 genre yang di simpan kedalam 10 folder yaitu folder blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, dan rock ke sepuluh folder tersebut masing-masing berisi 100 data suara sedangkan data freesound merupakan contoh data suara yang akan di gunakan untuk menguji hasil pengolahan data tersebut dengan menggunakan metode mfcc. 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:

```

In [2]: import librosa
...: import librosa.feature
...: import librosa.display
...: import glob
...: import numpy as np
...: import matplotlib.pyplot as plt
...: from keras.layers import Dense, Activation
...: from keras.models import Sequential
...: from keras.utils.np_utils import to_categorical
...:
...: def display_mfcc(song):
...:     y, _ = librosa.load(song)
...:     mfcc = librosa.feature.mfcc(y)
...:
...:     plt.figure(figsize=(10, 4))
...:     librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
...:     plt.colorbar()
...:     plt.title(song)
...:     plt.tight_layout()
...:     plt.show()

```

Gambar 9.10 Hasil Soal 1.

2. Soal 2

```

1 # In [2]: Soal Nomor 2
2 display_mfcc('freesound1.wav')
3 # In [2]: Soal Nomor 2

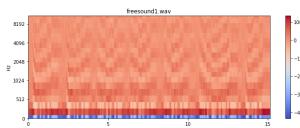
```

```

4 display_mfcc('freesound2.wav')
5 # In [2]: Soal Nomor 2
6 display_mfcc('genres/blues/blues.00023.au')
7 # In [2]: Soal Nomor 2
8 display_mfcc('genres/classical/classical.00023.au')
9 # In [2]: Soal Nomor 2
10 display_mfcc('genres/country/country.00023.au')
11 # In [2]: Soal Nomor 2
12 display_mfcc('genres/disco/disco.00023.au')
13 # In [2]: Soal Nomor 2
14 display_mfcc('genres/hiphop/hiphop.00023.au')
15 # In [2]: Soal Nomor 2
16 display_mfcc('genres/jazz/jazz.00023.au')
17 # In [2]: Soal Nomor 2
18 display_mfcc('genres/metal/metal.00023.au')
19 # In [2]: Soal Nomor 2
20 display_mfcc('genres/pop/pop.00023.au')
21 # In [2]: Soal Nomor 2
22 display_mfcc('genres/reggae/reggae.00023.au')
23 # In [2]: Soal Nomor 2
24 display_mfcc('genres/rock/rock.00023.au')

```

Kode di atas akan menampilkan hasil dari proses mfcc yang sudah dibuat fungsi pada soal 1, yaitu fungsi display mfcc akan menampilkan plot dari pembacaan file audio. Berikut adalah hasil dari salah satu pembacaan file audio :



Gambar 9.11 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]

```

Kode di atas adalah membuat fungsi yang didefinisikan dengan nama extract_features_song yang nantinya akan di gunakan pada fungsi yang lainnya kemudian dibuat variabel y dengan method librosa load setelah

itu dibuat variabel baru mfcc dengan isi librosa features mfcc dengan isi variabel y tadi kemudian dibuat variabel mfcc dengan isian np.max dan variabel mfcc tadi terakhir di buat array dari data tersebut merupakan data 25000 data pertama. kenapa data 25000 pertama yang digunakan dikarenakan data tersebut digunakan sebagai data testing semakin besar data testing yang di gunakan maka semakin akurat hasil AI. tapi sebenarnya data tersebut relatif bisa lebih besar atau lebih kecil tergantung pada komputer masing masing. Hasilnya adalah sebagai berikut :

```
In [8]: def extract_features_song(f):
    ...
    y, sr = librosa.load(f)
    ...
    # get Mel-frequency cepstral coefficients
    ...
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=12) # (divide by max)
    ...
    mfcc /= np.max(np.abs(mfcc))
    ...
    return np.ndarray.flatten(mfcc)[:25000]
```

Gambar 9.12 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' , '
8         hiphop' , 'jazz' , 'metal' , 'pop' , 'reggae' , 'rock' ]
9     for genre in genres:
10        sound_files = glob.glob('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 adalah mendefinisian nama fungsi yaitu generate features and labels kemudian membuat variabel baru dengan array kosong yaitu all_features dan all_labels kemudian mendefinisikan isian label untuk genre dengan cara membuat variabel genres kemudian di isi dengan 10 genre yang tadi setelah itu dilakukan fungsi if else dengan code for dan in setelah itu akan di buat encoding untuk data tiap tiap label contoh

untuk blues 1000000000 dan untuk clasical 0100000000. Hasilnya adalah sebagai berikut :

```
In [8]: def generate_features_and_labels():
    ...
    all_labels = []
    ...
    genres = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz', 'metal', 'pop',
              'reggae', 'rock']
    for genre in genres:
        for file in glob.glob(genre+'/*.mp3'):
            song = Song(file)
            if song.genre == genre:
                for t in song.frames:
                    if t['label'] != None:
                        all_labels.append(t['label'])
    all_features.append(features)
    ...
    # convert list of labels into integer encoding with blues : 10000000000000000000000000000000
    labels_int = []
    for i in range(len(all_labels)):
        labels_int.append(all_labels[i])
    labels_int = np.array(labels_int).reshape(-1, 1)
    return np.array(all_features), labels_int
```

Gambar 9.13 Hasil Soal 4.

5. Soal 5

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

Hal ini menjadi lama dikarenakan mesin membaca satupersatu file yang ada pada folder dan dalam foldert tersebut terdapat 100 file sehingga wajar menjadi lama ditambah lagi mengolah data yang tadinya suara menjadi bentuk vektor. Hasilnya adalah sebagai berikut :

```
In [11]: 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 pop genre...
Processing 100 songs in rock genre...
Processing 100 songs in reggae genre...
In [12]: print(np.shape(features))
...: print(np.shape(labels))
(10000, 25000)
(10000, 10)
```

Name	Type	Size	Value
features	float64	(10000, 25000)	[1.0, 0.12599871, -0.03151005, -0.75154891, ..., -0.01513394, -0.0127242]
labels	float64	(10000, 10)	[1.0, 0.0, ..., 0.0, 0.0]

Gambar 9.14 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 :

```
In [13]: training_split = 0.8
```

Gambar 9.15 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 [20]: model = Sequential([
...:     Dense(100, input_dim=np.shape(train_input)[1]),
...:     Activation('relu'),
...:     Dense(10),
...:     Activation('softmax'),
...: ])

```

Gambar 9.16 Hasil Soal 7.

8. Soal 8

```

1 # In [8]: Soal Nomor 8
2 model.compile(optimizer='adam',
3                 loss='categorical_crossentropy',
4                 metrics=['accuracy'])
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 aktivasinya 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 [21]: model.compile(optimizer='adam',
    ...,
    loss='categorical_crossentropy',
    ...,
    metrics=['accuracy'])
Model: "sequential_1"
Layer (Type)          Output Shape       Param #
dense_1 (Dense)      (None, 100)        1000
dense_2 (Dense)      (None, 10)         100
activation_1 (Activation) (None, 100)        0
dense_3 (Dense)      (None, 10)         100
activation_2 (Activation) (None, 10)         0
total params: 2,500,110
Trainable params: 2,500,110
Non-trainable params: 0
None
```

Gambar 9.17 Hasil Soal 8.

9. Soal 9

```
1 # In [9]: Soal Nomor 9
2 model.fit(train_input, train_labels, epochs=10, batch_size=
3           =32,
4           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 persentase dari cross validation atau k-fold sebanyak 20% dari masing-masing data pengulangan, hasilnya adalah sebagai berikut :

```
In [2]: model.fit(x_train, input_labels, epochs=10, batch_size=32)

Out[2]: 448/448 [448 samples] - 0s - loss: 2.0000 - accuracy: 0.2393 - val_loss: 1.9999 - val_accuracy: 0.4000
Epoch 1/10
448/448 [448 samples] - 0s - loss: 2.0000 - accuracy: 0.2393 - val_loss: 1.9999 - val_accuracy: 0.4000
Epoch 2/10
448/448 [448 samples] - 0s - loss: 1.9999 - accuracy: 0.3993 - val_loss: 1.9999 - val_accuracy: 0.4025
Epoch 3/10
448/448 [448 samples] - 0s - loss: 1.9999 - accuracy: 0.4001 - val_loss: 1.9999 - val_accuracy: 0.4025
Epoch 4/10
448/448 [448 samples] - 0s - loss: 1.9999 - accuracy: 0.4001 - val_loss: 1.9999 - val_accuracy: 0.4025
Epoch 5/10
448/448 [448 samples] - 0s - loss: 1.9999 - accuracy: 0.4001 - val_loss: 1.9999 - val_accuracy: 0.4025
Epoch 6/10
448/448 [448 samples] - 0s - loss: 1.9999 - accuracy: 0.4001 - val_loss: 1.9999 - val_accuracy: 0.4025
Epoch 7/10
448/448 [448 samples] - 0s - loss: 1.9999 - accuracy: 0.4001 - val_loss: 1.9999 - val_accuracy: 0.4025
Epoch 8/10
448/448 [448 samples] - 0s - loss: 1.9999 - accuracy: 0.4001 - val_loss: 1.9999 - val_accuracy: 0.4025
Epoch 9/10
448/448 [448 samples] - 0s - loss: 1.9999 - accuracy: 0.4001 - val_loss: 1.9999 - val_accuracy: 0.4025
Epoch 10/10
448/448 [448 samples] - 0s - loss: 1.9999 - accuracy: 0.4001 - val_loss: 1.9999 - val_accuracy: 0.4025
```

Gambar 9.18 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!")
5 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 [23]: loss, acc = model.evaluate(test_input, test_labels, batch_size=32)
200/200 [=====] - 0s 65ms/Step
```

Gambar 9.19 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 [25]: model.predict(test_input[:1])
Out[25]:
array([[5.2140345e-04, 5.0897381e-06, 5.1087983e-01, 4.7763154e-01,
       1.4319259e-03, 1.4710493e-05, 1.7895336e-06, 9.3976054e-03,
       2.2317986e-05, 9.7474178e-05]], dtype=float32)
```

Gambar 9.20 Hasil Soal 11.

9.2.3 Penanganan Error

1. ScreenShoot Error

```
FileNotFoundError: [Errno 2] No such file or directory: 'E:\Videoclassan Butan\dataset\genres\blues\blues.00022.wv'
```

Gambar 9.21 FileNotFoundError

2. Cara Penanganan Error

- **FileNotFoundException**

Error terdapat pada letak file yang tidak terbaca, karena letak file berbeda dengan pemanggilannya, solusi nya ialah dengan meletakkan direktori file yang dibaca dengan benar.

9.2.4 Bukti Tidak Plagiat



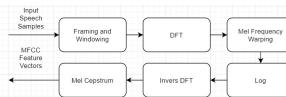
Gambar 9.22 Bukti Tidak Melakukan Plagiat Chapter 6

9.3 1174066 - D.Irga B. Naufal Fakhri

9.3.1 Teori

9.3.1.1 Kenapa file suara harus dilakukan MFCC

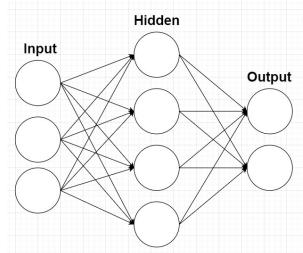
Karena MFCC adalah koefisien yang mewakili audio. Ekstraksi fitur dalam proses ini ditandai dengan konversi data suara menjadi gambar spektrum gelombang. File audio dilakukan oleh MFCC sehingga objek suara dapat dikonversi menjadi matriks. Suara akan menjadi vektor yang akan diproses sebagai output.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



Gambar 9.23 Teori 1

9.3.1.2 Konsep dasar neural network

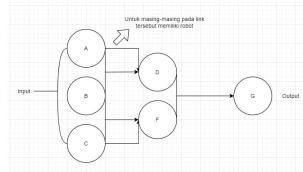
Konsep sederhana dari neural network atau jaringan saraf sederhana dengan proses pembelajaran pada anak-anak dengan memetakan pola-pola baru yang diperoleh dari input untuk membuat pola-pola baru pada output. Contoh sederhana ini menganalogikan kinerja otak manusia. Jaringan saraf itu sendiri terdiri 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 paralel. Karakteristik neural network atau jaringan saraf dilihat dari pola hubungan antar neuron, metode penentuan bobot setiap koneksi, dan fungsi aktivasi mereka.



Gambar 9.24 Teori 2

9.3.1.3 Konsep pembobotan dalam neural network

Pembobotan di dalam neural network juga akan menentukan penanda koneksi. 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.



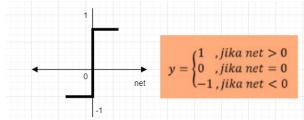
Gambar 9.25 Teori 3

9.3.1.4 Konsep fungsi aktifasi dalam neural network

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:

1. 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.
2. Fungsi Undak Biner Threshold, fungsi ini menggunakan nilai ambang sebagai batasnya.
3. Fungsi Bipolar Symetric Hard Limit, fungsi ini memiliki output bernilai 1, 0 atau -1.

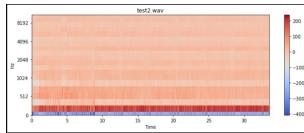
4. Fungsi Bipolar dengan Threshold, fungsi ini mempunyai output yang bernilai 1, 0 atau -1 untuk batas nilai ambang tertentu.
5. Fungsi Linear atau Identitas.



Gambar 9.26 Teori 4

9.3.1.5 Cara membaca hasil plot dari MFCC

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 terseksi adalah kekuatan atau nilai tertinggi dibandingkan dengan warna canggih. Yang merah terdengar di bawah pendengaran manusia, sehingga tidak bisa didengar secara langsung.



Gambar 9.27 Teori 5

9.3.1.6 one-hot encoding

one-hot encoding ini adalah untuk mengubah hasil data vektorisasi menjadi angka biner 0 dan 1 dan membuat informasi tentang atribut yang diberi label.

```
label = np.array([1, 0, 1, 0, 0]) # label row 0: 1, 0, 0, 0, 0
label_row_id = np.unique(label) # return unique integer
label_row_id = label_row_id.astype(np.int32, copy=False)
unique_label = np.unique(label) # return unique integer
label_id = np.where(label == unique_label)[0] # return index
one_hot = np.zeros((len(label), len(unique_label))) # one hot
one_hot[range(len(label)), label_id] = 1 # set value
```

Gambar 9.28 Teori 6-1

No	Color
0	Red
1	Green
2	Blue
3	Red
4	Blue

→

No	Red	Green	Blue
0	1	0	0
1	0	1	0
2	0	0	1
3	1	0	0
4	0	0	1

Gambar 9.29 Teori 6-2

9.3.1.7 Apa fungsi dari np.unique dan to_categorical dalam kode program

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 9.30 Teori 7-1

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

```
>>> import pandas as pd
>>> var = pd.Categorical(['a', 'a', 'y', 'a', 'd', 'i', 'e', 'g', 'a'])
>>> var
[('a', 0), ('a', 0), ('d', 1), ('r', 0), ('g', 0)]
Categories (7, object): [a, d, g, i, r, s, y].labels
```

Gambar 9.31 Teori 7-2

9.3.1.8 Apa fungsi dari Sequential dalam kode program

Fungsi dari Sequential sebagai salah satu jenis model yang digunakan dalam perhitungan. Sequential ini membangun tumpukan linear yang berurutan.

```
In [35]: model = Sequential([
    ...:     Dense(100, input_dim=inp.shape[1]),
    ...:     Activation('relu'),
    ...:     Dense(10),
    ...:     Activation('softmax'),
    ...: ])
```

Gambar 9.32 Teori 8

9.3.2 Praktek

9.3.2.1 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     plt.colorbar()
18     plt.title(song)
```

```

19     plt.tight_layout()
20     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 lagu yang saya pilih, jika GTZAN memiliki beberapa genre jika freesound hanya untuk 1 lagu dan disini kita membuat fungsi untuk membaca file audio dan outputnya sebagai plot

```

In [1]: import librosa
...: import librosa.feature
...: import librosa.display
...: import numpy as np
...: import matplotlib.pyplot as plt
...: from keras.layers import Dense, Activation
...: from keras.models import Sequential
...: from keras.utils import to_categorical
...: import tensorflow as tf
...: tf.logging.set_verbosity(tf.logging.ERROR)
...: %matplotlib inline
...: %config InlineBackend.figure_format = 'retina'
...: def display_mfcc(song):
...:     librosa.load(song)
...:     mfcc = librosa.feature.mfcc(y)
...: 
...:     plt.figure(figsize=(10, 4))
...:     librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
...:     plt.colorbar()
...:     plt.title(song)
...:     plt.tight_layout()
...: 
...: display_mfcc('N:/KB/unravel.wav')
Using TensorFlow backend.

```

Gambar 9.33 Nomor 1

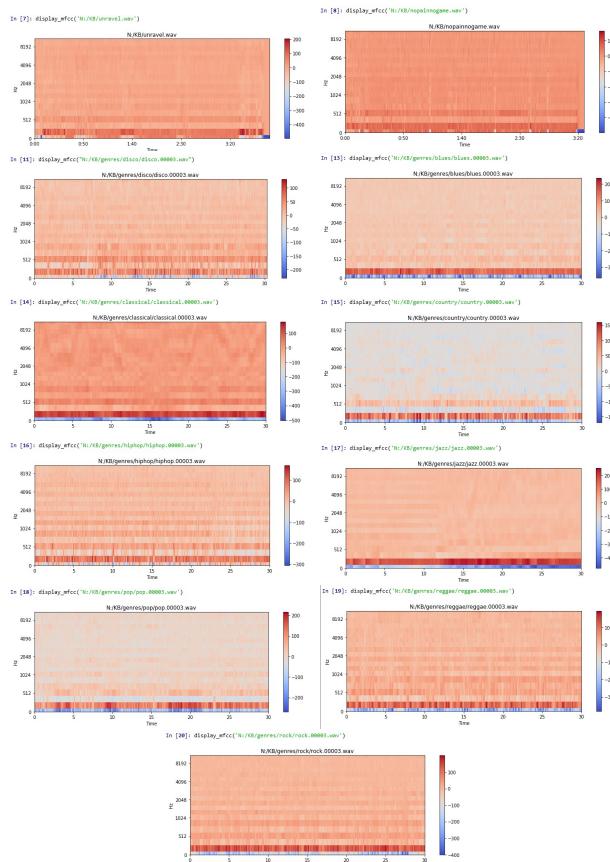
9.3.2.2 Nomor 2

```

1 # In [2]: Soal 2
2 display_mfcc('N:/KB/unravel.wav')
3 # In [2]: Soal 2
4 display_mfcc('N:/KB/nopainnogame.wav')
5 # In [2]: Soal 2
6 display_mfcc("N:/KB/genres/disco/disco.00003.wav")
7 # In [2]: Soal 2
8 display_mfcc('N:/KB/genres/blues/blues.00003.wav')
9 # In [2]: Soal 2
10 display_mfcc('N:/KB/genres/classical/classical.00003.wav')
11 # In [2]: Soal 2
12 display_mfcc('N:/KB/genres/country/country.00003.wav')
13 # In [2]: Soal 2
14 display_mfcc('N:/KB/genres/hiphop/hiphop.00003.wav')
15 # In [2]: Soal 2
16 display_mfcc('N:/KB/genres/jazz/jazz.00003.wav')
17 # In [2]: Soal 2
18 display_mfcc('N:/KB/genres/pop/pop.00003.wav')
19 # In [2]: Soal 2
20 display_mfcc('N:/KB/genres/reggae/reggae.00003.wav')
21 # In [2]: Soal 2
22 display_mfcc('N:/KB/genres/rock/rock.00003.wav')

```

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 9.34 Nomor 2

9.3.2.3 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]

```

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.

```
In [21]: def extract_features_song(f):
    ...
    y, sr = librosa.load(f)
    ...
    mfcc = librosa.feature.mfcc(y)
    ...
    # normalize features between 0,1 (divide by max)
    ...
    mfcc /= np.max(np.abs(mfcc))
    ...
    return np.ndarray.flatten(mfcc)[:25000]
```

Gambar 9.35 Nomor 3

9.3.2.4 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('N:/KB/genres/' + genre + '/*.wav')
9         print('Processing %d songs in %s genre...' % (len(
10            sound_files), genre))
11         for f in sound_files:
12             features = extract_features_song(f)
13             all_features.append(features)
14             all_labels.append(genre)
15
16     # convert labels to one-hot encoding cth blues : 1000000000
17     # classic 0100000000
18     label_uniq_ids, label_row_ids = np.unique(all_labels,
19                                              return_inverse=True) #ke integer
20     label_row_ids = label_row_ids.astype(np.int32, copy=False)
21     onehot_labels = to_categorical(label_row_ids, len(
22     label_uniq_ids)) #ke one hot
23
24     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.

```
In [26]: def generate_features_and_labels():
    ...
    all_features = []
    all_labels = []
    ...
    genres = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz', 'metal', 'pop', 'reggae', 'rock']
    ...
    for genre in genres:
        genre_dir = os.path.join('N:/KB/genres', genre)
        for f in os.listdir(genre_dir):
            if f.endswith('.au'):
                features = extract_features_song(os.path.join(genre_dir, f))
                all_features.append(features)
                all_labels.append(genre)
    ...
    label_uniq_ids, label_row_ids = np.unique(all_labels, return_inverse=True) #ke integer
    onehot_labels = to_categorical(label_row_ids, len(
        label_uniq_ids)) #ke one hot
    return np.stack(all_features), onehot_labels
```

Gambar 9.36 Nomor 4

9.3.2.5 Nomor 5

```
1 features, labels = generate_features_and_labels()
```

```
2 print(np.shape(features))  
3 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.

```
In [27]: features, labels = generate_features_and_labels()
...: print(np.shape(features))
...: print(np.shape(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 pop genre...
Processing 100 songs in reggae genre...
Processing 100 songs in rock genre...
(1000, 25900)
(1000, 10)
```

features	float32	(1000, 25900)	[1. 0. 0. ... 0. 0.]
labels	float32	(1000, 10)	[1. 0. 0. ... 0. 0.]

Gambar 9.37 Nomor 5

9.3.2.6 Nomor 6

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

The screenshot shows a Jupyter Notebook interface with several code cells and their outputs.

- In [28]:** `training_split = 0.8`
- In [29]:** Prints the shape of training data and features.
- In [30]:** Prints the shape of training data, features, and labels.
- In [31]:** Prints the shape of training and test data.
- In [32]:** Prints the shape of training input and labels.
- In [33]:** Prints the shape of test input and labels.
- In [34]:** Prints the shape of training input and labels.

Gambar 9.38 Nomor 6

9.3.2.7 Nomor 7

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

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.

```
In [35]: model = Sequential([
...:     Dense(100, input_dim=np.shape(train_input)[1]),
...:     Activation('relu'),
...:     Dense(10),
...:     Activation('softmax'),
...: ])
```

Gambar 9.39 Nomor 7

9.3.2.8 Nomor 8

```
1 model.compile(optimizer='adam',
```

```
2         loss='categorical_crossentropy',  
3         metrics=['accuracy']))  
4 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 aktivasinya 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 [36]: 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: 2,500,110
Non-trainable params: 0
None
```

Gambar 9.40 Nomor 8

9.3.2.9 Nomor 9

```
1 model.fit(train_input, train_labels, epochs=10, batch_size=32,
2           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 persentase dari cross validation atau k-fold sebanyak 20% dari masing-masing data pengulangan.

Gambar 9.41 Nomor 9

9.3.2.10 Nomor 10

```
1 loss , acc = model.evaluate(test_input , test_labels , batch_size  
=32)
```

```
2 print("Done!")
3 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%.

```
In [38]: loss, acc = model.evaluate(test_input, test_labels, batch_size=32)
...:     print("Done!")
...:     print("Loss: %.4f, accuracy: %.4f" % (loss, acc))
200/200 [=====] - 0s 453us/step
Done!
Loss: 1.3000 accuracy: 0.5152
```

Gambar 9.42 Nomor 10

9.3.2.11 Nomor 11

```
1 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 [39]: model.predict(test_input[:1])
Out[39]:
```

Gambar 9.43 Nomer 11

9.3.3 Penanganan Error

0.3.3.1 Error

- #### ■ NoBackendError

Gambar 9.44 NoBackendError

9.3.3.2 Solusi Error

- #### ■ NoBackendError

Cek kembali file format audionya dipastikan menggunakan wav

9.3.4 Bukti Tidak Plagiat



Gambar 9.45 Bukti tidak plagiat

9.3.5 Link Youtube

https://youtu.be/zA7U-zfvI_w

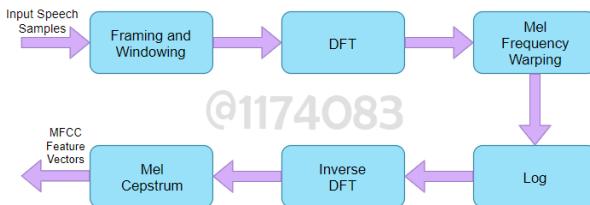
9.4 1174083 - Bakti Qilan Mufid

Chapter 6 - MFCC dan Neural Network

9.4.1 Teori

9.4.1.1 Jelaskan kenapa file suara harus dilakukan MFCC. lengkapilah dengan ilustrasi atau gambar

Mel-Frequency Cepstral Coefficient (MFCC) itu ya cuma koefisien(bisa lebih dari satu) yang disebut Koefisien Mel Frequency. Nah, itu tu buat apa sih??. jadi gini, MFCC itu biasa digunakan dalam speech processing. lebih khususnya lagi dalam pengenalan suara atau yang lebih kerennya kita kenal dengan speech recognition. MFCC adalah metode untuk memproses sinyal suara, agar bisa kelihatan deh tu ciri-cirinya. Dilakukan proses MFCC sedemikian rupa, sehingga suatu sinyal suara bisa diekstraksi ciri2 yang membedakannya. Uniknya lagi, MFCC itu seperti pendengaran kita, karena dia memproses suara itu hampir seperti telinga kita. Tau ga sih, telinga kita itu filternya kan beda2. Kalau tinggi filternya gimana, kalau rendah gimana, nah MFCC itu seperti itu, memprosesnya secara logaritmik. Jadi Mel-Frequency Cepstral Coefficients (MFCC) dapat digunakan sebagai vektor ciri yang baik untuk merepresentasikan suara manusia dan sinyal musik. Lebih khusus lagi, MFCC telah terbukti bermanfaat untuk pengenalan suara.



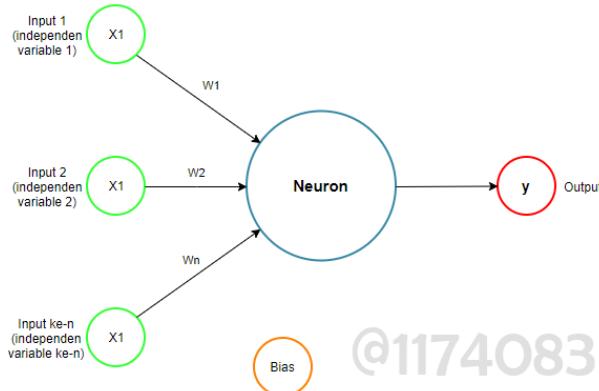
Gambar 9.46 gambaran penjelasan no. 1

9.4.1.2 Jelaskan konsep dasar neural network dilengkapi dengan ilustrasi atau gambar

Melalui ilustrasi di bawah bisa dilihat bahwa secara umum sebuah neural network (NN) terbagi menjadi tiga bagian, yaitu input, neuron (hidden layer) dan output. Dalam konteks deep learning, neuron memiliki istilah lain yaitu perceptron.

Input dari sebuah NN adalah variabel independen yang kita miliki. Output adalah variabel dependen yang kita cari. Misal kita ingin memprediksi harga rumah, maka variabel independennya misalnya luas tanah, luas bangunan, usia bangunan, dan lain-lain. Variabel dependennya adalah harga rumah itu sendiri. Pada contoh ini jenis output (variabel dependen) nya adalah variabel kontinu (continuous variable).

Untuk kasus yang lain, misal kita ingin mendeteksi apakah seorang calon pelanggan masuk ke kriteria target pemasaran atau tidak, maka variabel dependennya adalah berjenis binary (hanya ada 2 pilihan, ya/tidak). Jika terdiri dari lebih dari 2 kategori, maka ia masuk ke jenis kategori (categorical dependent variable). Perlu diperhatikan, jika berjenis kategori, maka outputnya juga lebih dari satu sesuai jumlah kategorinya.

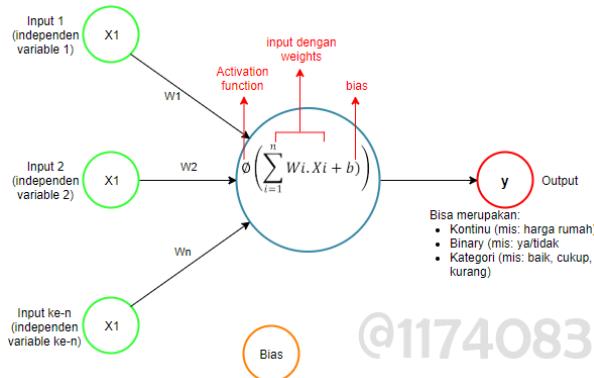


Gambar 9.47 gambaran penjelasan no. 2

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

Setiap input memiliki nilai W yang berbeda. Bobot (weights) sangat krusial bagi neuron, karena ini merupakan bagian agar neuron bisa belajar. Nilai w (simbol untuk weights) bisa sama dan bisa juga tidak untuk setiap input yang diterima dari neuron.

Semakin besar nilai w sebuah input, maka sinyal yang berasal dari input tertentu memiliki prioritas yang semakin besar untuk bisa berkontribusi kepada neuron di depannya. Selain itu, nilai w juga menentukan apakah sinyal dari input tertentu bisa melewati neuron di depannya atau tidak.

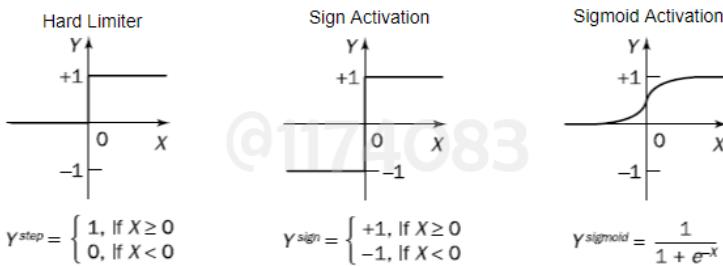


Gambar 9.48 gambaran penjelasan no. 3

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

Fungsi aktivasi (activation function), merupakan fungsi matematis yang digunakan untuk mendapatkan output neuron dari nilai inputnya. Disebut aktifasi karena output akan bernilai jika melampaui nilai threshold-nya. Beberapa fungsi aktivasi yang sering digunakan, yaitu :

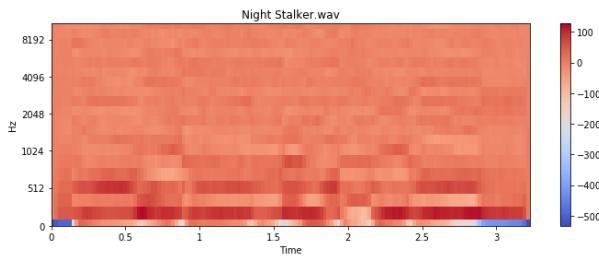
- hard limiter
- signum activation
- sigmoid activation.



Gambar 9.49 gambaran penjelasan no. 4

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

Sumbu y untuk tinggi rendah frekuensi dan sumbu x untuk lama waktu audio. Semakin gelap warnanya, atau semakin dekat ke merah, semakin banyak daya dalam rentang frekuensi pada waktu itu.



Gambar 9.50 gambaran penjelasan no. 5

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

one-hot encoding merupakan proses untuk mengkoversi variable categorical menjadi bentuk yang dapat diterapkan untuk Machine Learning.

The diagram illustrates a mapping from a categorical list to a binary matrix. On the left, a vertical list of categories is shown: Warna, Merah, Merah, Kuning, Hijau, and Kuning. A large blue arrow points from this list to a 6x3 binary matrix on the right. The matrix has columns labeled Merah, Kuning, and Hijau. The rows correspond to the categories in the list. The matrix values are: Row 1 (Warna) contains [1, 0, 0]; Row 2 (Merah) contains [1, 0, 0]; Row 3 (Merah) contains [0, 1, 0]; Row 4 (Kuning) contains [0, 0, 1]; Row 5 (Hijau) contains [0, 1, 0]; Row 6 (Kuning) contains [0, 0, 1].

Warna	Merah	Kuning	Hijau
Merah	1	0	0
Merah	1	0	0
Kuning	0	1	0
Hijau	0	0	1
Kuning	0	1	0

Gambar 9.51 gambaran penjelasan no. 6

9.4.1.7 Jelaskan apa fungsi dari np.unique dan to_categorical dalam kode program, dilengkapi dengan ilustrasi atau gambar

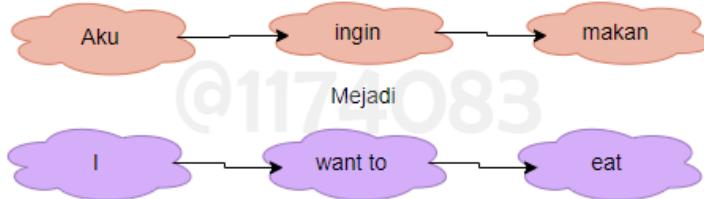
Fungsi np.unique adalah untuk mengembalikan array elemen unik dari inputan array. Contoh sebuah array [5, 2, 6, 2, 7, 5, 6, 8, 2, 9] diterapkan fungsi np.unique hasilnya menjadi [2, 5, 6, 7, 8, 9]. Fungsi to_categorical adalah untuk mengubah nama class menjadi beberapa genre atau mengkategorikan class sesuai jenisnya.

```
>>> import numpy as np
>>> x = np.array([5, 2, 6, 2, 7, 5, 6, 8, 2, 9])
>>> np.unique(x)
array([2, 5, 6, 7, 8, 9])
<<<
>>> import pandas as pd
>>> bakti = pd.Categorical(['n','o','c','o','p','a','s','-','c','o','p','a','s','c','l','u','b'])
>>> print(bakti)
[n, o, c, o, p, ..., s, c, l, u, b]
Length: 17
Categories (10, object): [-, a, b, c, ..., o, p, s, u]
```

Gambar 9.52 gambaran penjelasan no. 7

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

Fungsi dari Sequential adalah untuk memodelkan atau membuat prediksi jenis data yang sequential atau berurutan, seperti audio, teks, dan lain-lain. Contoh program untuk mengambil sepotong teks dalam bahasa Indonesia dan menerjemahkannya ke bahasa Inggris.



Gambar 9.53 gambaran penjelasan no. 9

9.4.2 Praktek

9.4.2.1 *Jelaskan isi dari data GTZAN Genre Collection dan data dari freesound. Buat kode program untuk meload data tersebut untuk digunakan pada MFCC. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas)*

Data GTZAN Genre Collection berisikan 1000 lagu dari 10 genre berbeda. Di-tiap genrenya masing-masing terdapat 100 lagu yang kurang lebih durasinya 30 detik. sedangkan freesound adalah repository yang emmiliki lisensi berisikan sampel audio, contohnya suara burung, suaran drum, suara alam, dll.

Kode program untuk meload data tersebut untuk digunakan pada MFCC.

```

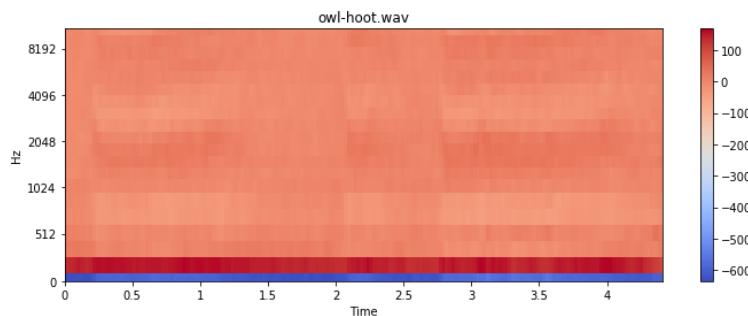
1 # In [ Soal Nomor 1]:
2 import librosa
3 import librosa.feature
4 import librosa.display
5 import glob
6 import numpy as np
7 import matplotlib.pyplot as plt
8 from keras.layers import Dense, Activation
9 from keras.models import Sequential
10 from keras.utils.np_utils import to_categorical
11 from scipy import signal
  
```

Import library librosa untuk mengekstrak fitur dari lagu, import library glob untuk me-list file lagu ke berbagai direktori genre, import library numpy untuk melakukan operasi numerical, import library matplotlib untuk menggambarkan grafik MFCC,import layer dense dari library keras yang memiliki banyak neuron di dalamnya, import layer activation dari library keras untuk menggunakan activation function di tiap layer neuron, import model sequential dari library keras, import to categorical dari library keras untuk mengubah nama class menjadi beberapa genre, dan import signal dari library scipy untuk mengatasi error.

Ini spectogram dari owl-hoot(source dari freesound) yang menampilkan frekuensi rendah.

```

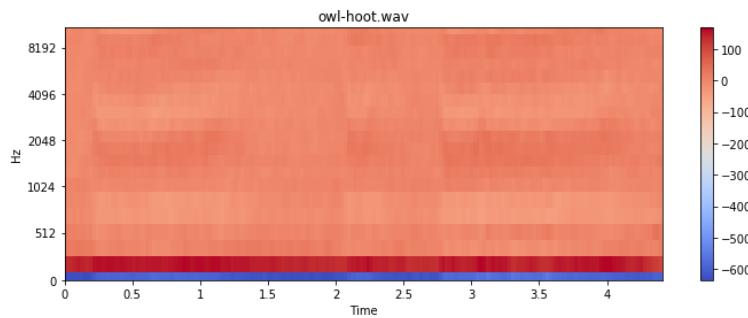
1 display_mfcc('owl-hoot.wav')
  
```



Gambar 9.54 spectogram owl-hoot

Ini spectrogram dari the-canadian-loon(source dari freesound) yang menampilkan frekuensi rendah.

```
1 display_mfcc('the-canadian-loon.wav')
```



Gambar 9.55 spectogram the-canadian-loon

Untuk contoh spectrogram dari GTZAN Genre Collection, bisa dilihat di nomor selanjutnya.

9.4.2.2 Jelaskan perbaris kode program dengan kata-kata dan dilengkapi ilustrasi gambar fungsi dari `display_mfcc()`

```
1 def display_mfcc(song):
2     y, _ = librosa.load(song)
3     mfcc = librosa.feature.mfcc(y)
4
5     plt.figure(figsize=(10, 4))
6     librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
7     plt.colorbar()
8     plt.title(song)
9     plt.tight_layout()
10    plt.show()
```

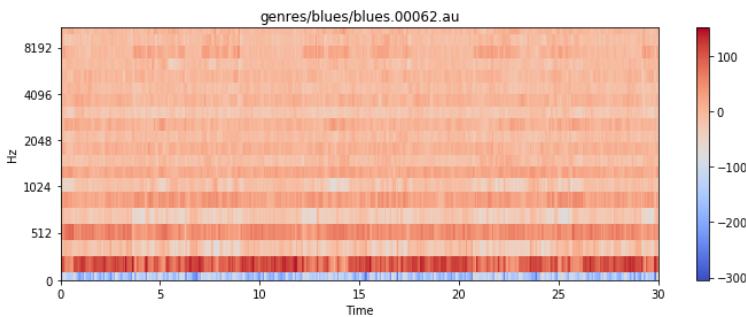
Membuat fungsi untuk menampilkan nilai dari MFCC dari tiap lagu. Pertama load lagunya, lalu ekstrak lagunya dengan fungsi mfcc dan tampung nilainya. Kemudian tampilkan spectogramnya dengan fungsi specshow.

- Nama fungsinya display_mfcc() dengan parameter song untuk lagu yang nantinya diproses.
- Fungsi load() untuk meload lagunya kemudian ditampung nilainya.
- Fungsi mfcc() untuk mengekstrak fitur dari lagunya kemudian ditampung nilainya.
- Fungsi figure() untuk membuat grafiknya.
- Fungsi specshow() untuk membuat spectogramnya.
- Fungsi colorbar() untuk memberi warna pada grafiknya.
- Fungsi title() untuk memberi judul pada grafiknya.
- Fungsi tight layout() untuk menyesuaikan grafiknya sesuai layout.
- Fungsi show() untuk menampilkan grafiknya.

berikut adalah beberapa contoh penggunaan display_mfcc(), beserta contoh data dari GTZAN Genre Collection

Ini spectogram dari lagu genre blues (source dari GTZAN Genre Collection).

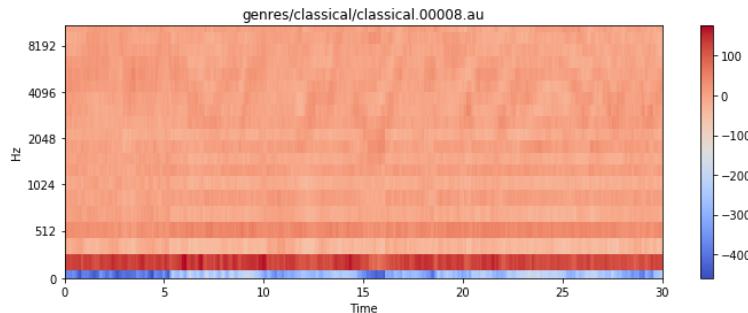
```
1 display_mfcc('genres/blues/blues.00062.au')
```



Gambar 9.56 spectogram dari lagu genre blues

Ini spectogram dari lagu genre classical (source dari GTZAN Genre Collection).

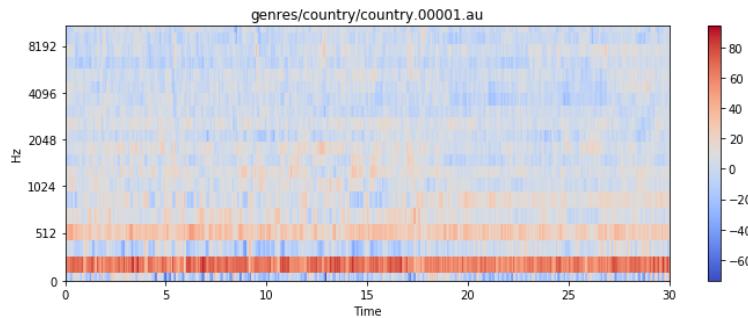
```
1 display_mfcc('genres/classical/classical.00008.au')
```



Gambar 9.57 spectogram dari lagu genre classical

Ini spectogram dari lagu genre country (source dari GTZAN Genre Collection).

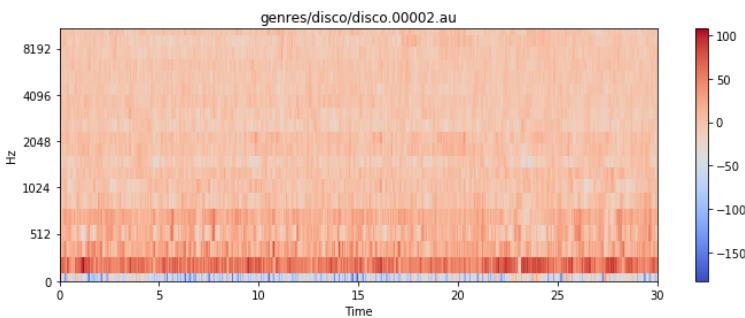
```
1 display_mfcc('genres/country/country.00001.au')
```



Gambar 9.58 spectogram dari lagu genre country

Ini spectogram dari lagu genre disco (source dari GTZAN Genre Collection).

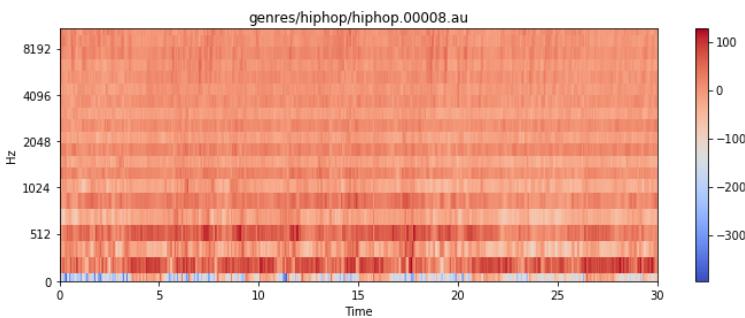
```
1 display_mfcc('genres/disco/disco.00002.au')
```



Gambar 9.59 spectogram dari lagu genre disco

Ini spectogram dari lagu genre hiphop (source dari GTZAN Genre Collection).

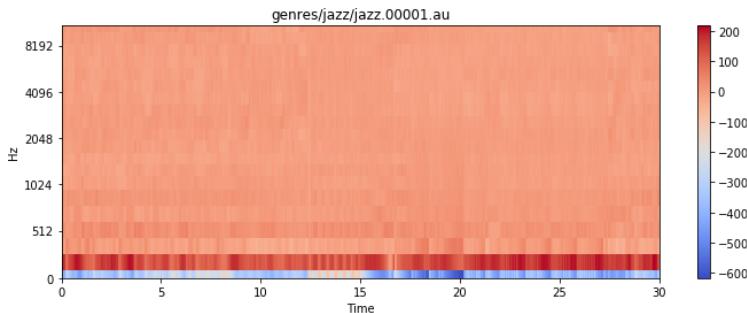
```
1 display_mfcc('genres/hiphop/hiphop.00008.au')
```



Gambar 9.60 spectogram dari lagu genre hiphop

Ini spectogram dari lagu genre jazz (source dari GTZAN Genre Collection).

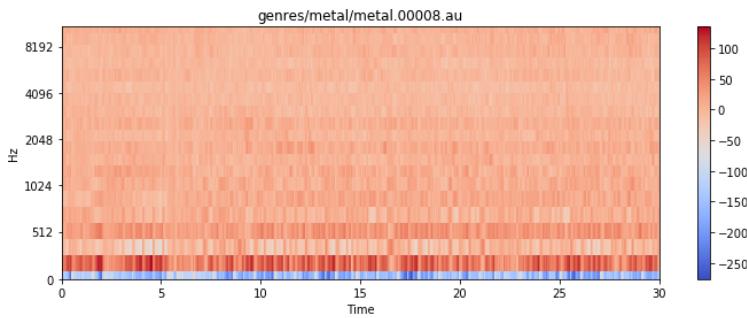
```
1 display_mfcc('genres/jazz/jazz.00001.au')
```



Gambar 9.61 spectrogram dari lagu genre jazz

Ini spectrogram dari lagu genre metal (source dari GTZAN Genre Collection).

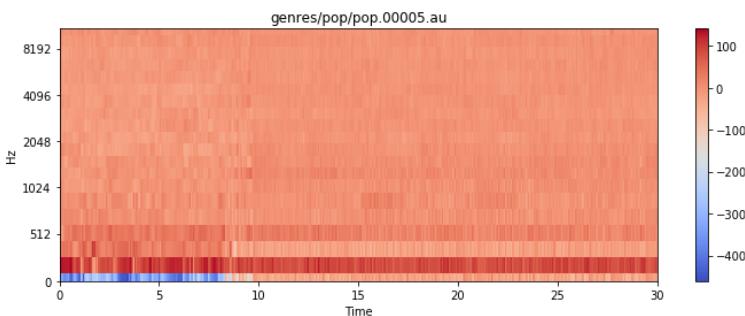
```
1 display_mfcc('genres/metal/metal.00008.au')
```



Gambar 9.62 spectrogram dari lagu genre metal

Ini spectrogram dari lagu genre pop (source dari GTZAN Genre Collection).

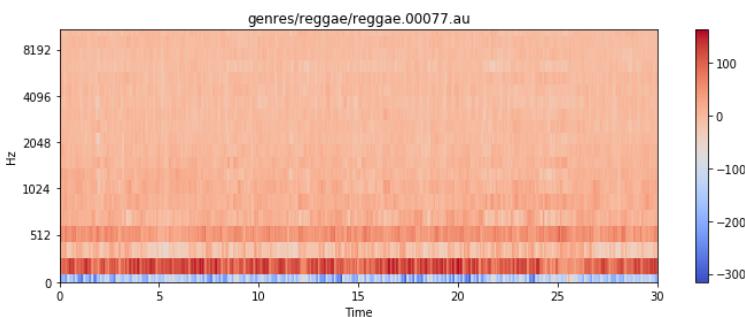
```
1 display_mfcc('genres/pop/pop.00005.au')
```



Gambar 9.63 spectogram dari lagu genre pop

Ini spectogram dari lagu genre reggae (source dari GTZAN Genre Collection).

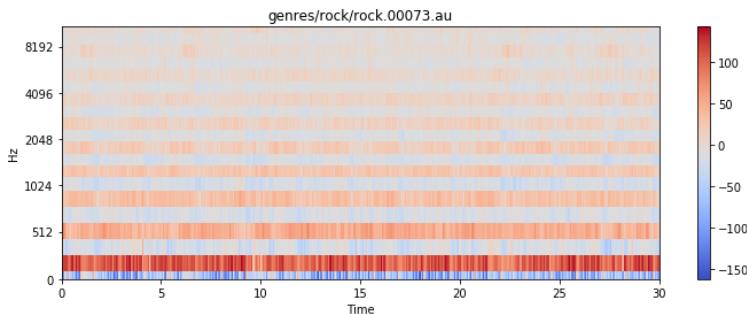
```
1 display_mfcc('genres/reggae/reggae.00077.au')
```



Gambar 9.64 spectogram dari lagu genre reggae

Ini spectogram dari lagu genre rock (source dari GTZAN Genre Collection).

```
1 display_mfcc('genres/rock/rock.00073.au')
```



Gambar 9.65 spectrogram dari lagu genre rock

9.4.2.3 *Jelaskan perbaris kode program dengan kata-kata dan dilengkapi ilustrasi gambar fungsi dari extract_features_song(). Jelaskan juga mengapa yang diambil 25.000 baris pertama?*

Kode :

```

1 # In [Soal Nomor 3]:
2 def extract_features_song(f):
3     y, _ = librosa.load(f)
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]

```

- nama fungsinya adalah extract_features_song() dengan paramete f untuk lagu yang nantinya diproses.
- fungsi load() untuk meload lagunya kemudian ditampung nilainya.
- fungsi mfcc() untuk mengekstrak fitur dari lagunya kemudian ditampung nilainya.
- fungsi absolute() untuk menjadikannya nilai absolut. Panggil fungsi amax untuk mencari nilai max. Kemudian hasil tadi dibagi dengan nilai mfcc sebelumnya.
- lalu mengembalikan hasil convert array ke dalam bentuk satu dimensi dari nilai mfcc dan diambil 25000 baris pertama.

Alasan diambil 25000 baris pertama, karena setiap lagu memiliki panjang nilai MFCC yang berbeda-beda dan yang dimasukkan ke dalam neuron harus memiliki ukuran yang sama.

```
In [27]: def extract_features_song(f):
...:     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 9.66 Hasil dari kode program praktek nomor 3

9.4.2.4 Jelaskan perbaris kode program dengan kata-kata dan dilengkapi ilustrasi gambar fungsi dari generate_features_and_labels()

Kode :

```
1 # In [ Soal Nomor 4]:
2 def generate_features_and_labels():
3     all_features = []
4     all_labels = []
5
6     genres = [ 'blues' , 'classical' , 'country' , 'disco' , 'hiphop' ,
7               'jazz' , 'metal' , 'pop' , 'reggae' , 'rock' ]
8     for genre in genres:
9         sound_files = glob.glob( 'genres/' +genre+ '/*.au' )
10        print('Processing %d songs in %s genre...' % (len(
11            sound_files) , genre))
12        for f in sound_files:
13            features = extract_features_song(f)
14            all_features.append(features)
15            all_labels.append(genre)
16
17        # convert labels to one-hot encoding cth blues : 10000000000000000000
18        # convert labels to one-hot encoding cth blues : 10000000000000000000
19        label_uniq_ids , label_row_ids = np.unique(all_labels ,
20                                         return_inverse=True)#ke integer
21        label_row_ids = label_row_ids.astype(np.int32 , copy=False)
22        onehot_labels = to_categorical(label_row_ids , len(
23            label_uniq_ids))#ke one hot
24
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.

```
In [30]: 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('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 cth blues : 1000000000 classic 0100000000
    label_uniq_ids, label_row_ids = np.unique(all_labels, return_inverse=True) #ke integer
    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
    ...
    return np.stack(all_features), onehot_labels
```

Gambar 9.67 Hasil dari kode program praktek nomor 4

9.4.2.5 Jelaskan dengan kata dan praktek kenapa penggunaan fungsi `generate_features_and_labels()` sangat lama ketika meload dataset genre. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan

Kode :

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

<pre>In [31]: 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 pop genre... Processing 100 songs in reggae genre... Processing 100 songs in rock genre...</pre>	<pre>In [32]: print(np.shape(features)) ... print(np.shape(labels)) (1000, 25000) (1000, 10)</pre>												
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Size</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>features</td> <td>float32</td> <td>(1000, 25000)</td> <td>[[-0.81999075 -0.81014305 -0.75184417 ... -0.01818394 0.01827242 0 ...]]</td> </tr> <tr> <td>labels</td> <td>float32</td> <td>(1000, 10)</td> <td>[[1. 0. 0. ... 0. 0. 0.] [1. 0. 0. ... 0. 0. 0.]</td> </tr> </tbody> </table>	Name	Type	Size	Value	features	float32	(1000, 25000)	[[-0.81999075 -0.81014305 -0.75184417 ... -0.01818394 0.01827242 0 ...]]	labels	float32	(1000, 10)	[[1. 0. 0. ... 0. 0. 0.] [1. 0. 0. ... 0. 0. 0.]	
Name	Type	Size	Value										
features	float32	(1000, 25000)	[[-0.81999075 -0.81014305 -0.75184417 ... -0.01818394 0.01827242 0 ...]]										
labels	float32	(1000, 10)	[[1. 0. 0. ... 0. 0. 0.] [1. 0. 0. ... 0. 0. 0.]										

Gambar 9.68 Hasil dari kode program praktek nomor 5

9.4.2.6 Jelaskan kenapa harus dilakukan pemisahan data training dan data set sebesar 80%? Praktekan dengan kode dan Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan

Kode :

```

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

```

Dilakukannya pemisahan data training dan data set sebesar 80% agar model yang dihasilkan lebih akurat dan meminimalisir kesalahan dalam pembelajaran mesin.

In [33]:	training_split = 0.8	In [36]: print(np.shape(train)) ...: print(np.shape(test)) (800, 25010) (200, 25010)
		In [39]: print(np.shape(train_input)) ...: print(np.shape(train_labels)) (800, 25000) (800, 10)
training_split	float 1	0.8
alldata	float32 (1000, 25010)	[-0.81999075 -0.81014305 -0.75184417 ... 0. 0. ...]
Name	Type	Size
training_split	float 1	0.8
train	float32 (800, 25010)	[[-0.16444726 -0.08589051 0.2347324 ... 1. 0. ...]
test	float32 (200, 25010)	[[-0.3358299 -0.33254293 -0.4001908 ... 0. 1. ...]
splitidx	int 1	800
train_input	float32 (800, 25000)	[[-0.16444726 -0.08589051 0.2347324 ... 0.06372784 0.05331... -0 ...]
train_labels	float32 (800, 10)	[[[0. 0. 0. ... 1. 0. 0.] [0. 0. 1. ... 0. 0. 0.]]
test_input	float32 (200, 25000)	[[-0.3358299 -0.33254293 -0.4001908 ... 0.05416475 0.05123... 0 ...]
test_labels	float32 (200, 10)	[[[0. 0. 0. ... 0. 0. 1.] [0. 0. 0. ... 0. 0. 1.]]

Gambar 9.69 Hasil dari kode program praktek nomor 6

9.4.2.7 Praktekkan dan jelaskan masing-masing parameter dari fungsi Sequential(). Tunjukkan keluaranya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan

Kode :

```
1 # In [ 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() adalah 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.

```
In [40]: model = Sequential([
...:     Dense(100, input_dim=np.shape(train_input)[1]),
...:     Activation('relu'),
...:     Dense(10),
...:     Activation('softmax'),
...: ])
```

Gambar 9.70 Hasil dari kode program praktek nomor 7

9.4.2.8 Praktekkan dan jelaskan masing-masing parameter dari fungsi compile(). Tunjukkan keluaranya dengan fungsi summary dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan

Kode :

```
1 # In [ Soal Nomor 8]:
2 model.compile(optimizer='adam',
3                 loss='categorical_crossentropy',
4                 metrics=['accuracy'])
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 aktivasi 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 [42]: model.compile(optimizer='adam',
...:                     loss='categorical_crossentropy',
...:                     metrics=['accuracy'])
...: print(model.summary())
Model: "sequential_2"

Layer (type)                 Output Shape              Param #
=====
dense_3 (Dense)            (None, 100)             2500100
activation_3 (Activation)   (None, 100)              0
dense_4 (Dense)            (None, 10)               1010
activation_4 (Activation)   (None, 10)              0
=====
Total params: 2,501,110
Trainable params: 2,501,110
Non-trainable params: 0
=====
None
```

Gambar 9.71 Hasil dari kode program praktek nomor 8

9.4.2.9 Praktekkan dan jelaskan masing-masing parameter dari fungsi *fit()*. dan tunjukkan maksud setiap luaran yang didapatkan

Kode :

```
# In [Soal Nomor 9]:
model.fit(train_input, train_labels, epochs=10, batch_size=32,
           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.

```
[43]: model.fit(train_input, train_labels, epochs=10, batch_size=32,
Train on 640 samples, validate on 160 samples
Epoch 1/10
640/640 [=====] - 3s 4ms/step - loss: 1.0998 - accuracy: 0.2703 - val_loss:
1.252 - val_accuracy: 0.3500
Epoch 2/10
640/640 [=====] - 2s 3ms/step - loss: 1.4819 - accuracy: 0.5078 - val_loss:
1.4663 - val_accuracy: 0.4812
Epoch 3/10
640/640 [=====] - 2s 3ms/step - loss: 1.0866 - accuracy: 0.6669 - val_loss:
1.4845 - val_accuracy: 0.5063
Epoch 4/10
640/640 [=====] - 2s 3ms/step - loss: 0.8257 - accuracy: 0.7563 - val_loss:
1.4801 - val_accuracy: 0.8875
Epoch 5/10
640/640 [=====] - 2s 3ms/step - loss: 0.6824 - accuracy: 0.8266 - val_loss:
1.4952 - val_accuracy: 0.4798
Epoch 6/10
640/640 [=====] - 2s 3ms/step - loss: 0.5573 - accuracy: 0.8969 - val_loss:
1.3768 - val_accuracy: 0.5132
Epoch 7/10
640/640 [=====] - 2s 3ms/step - loss: 0.4153 - accuracy: 0.9250 - val_loss:
1.3843 - val_accuracy: 0.5375
Epoch 8/10
640/640 [=====] - 2s 3ms/step - loss: 0.3512 - accuracy: 0.9469 - val_loss:
1.4800 - val_accuracy: 0.4625
Epoch 9/10
640/640 [=====] - 2s 3ms/step - loss: 0.2584 - accuracy: 0.9828 - val_loss:
1.3868 - val_accuracy: 0.5437
Epoch 10/10
640/640 [=====] - 2s 3ms/step - loss: 0.1953 - accuracy: 0.9986 - val_loss:
1.4956 - val_accuracy: 0.5926

```

Gambar 9.72 Hasil dari kode program praktek nomor 9

9.4.2.10 Praktekkan dan jelaskan masing-masing parameter dari fungsi *evaluate()*. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan

Kode :

```

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

```

Fungsi evaluate atau evaluasi ini ialah untuk menguji data pengujian se- tiap file. Di sini ada prediksi yang hilang, artinya mesin memprediksi data, sedangkan untuk keseluruhan perjanjian sekitar 55%.

```

In [45]: print("Done!")
...: print("Loss: %.4f, accuracy: %.4f" % (loss, acc))
Done!
Loss: 1.2897, accuracy: 0.5800

```

Gambar 9.73 Hasil dari kode program praktek nomor 10

9.4.2.11 Praktekkan dan jelaskan masing-masing parameter dari fungsi predict(). Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan

Kode :

```

1 # In [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 [46]: model.predict(test_input[:1])
Out[46]:
array([[2.30070949e-03, 2.77521031e-06, 2.15052031e-02, 3.57471257e-01,
       9.54874605e-03, 1.09723915e-05, 3.56088638e-01, 1.69718251e-09,
      1.03918152e-04, 2.52967805e-01]], dtype=float32)

```

Gambar 9.74 Hasil dari kode program praktek nomor 11

9.4.3 Penanganan Error

9.4.3.1 Terjadi error

1. terjadi error no module named 'librosa', seperti pada gambar berikut:

```

ModuleNotFoundError: No module named 'librosa'

```

Gambar 9.75 terjadi Error 1

2. terjadi error no module named 'keras', seperti pada gambar berikut:

```
ModuleNotFoundError: No module named 'keras'
```

Gambar 9.76 terjadi Error 2

3. terjadi error no module named 'tensorflow', seperti pada gambar berikut:

```
ModuleNotFoundError: No module named 'tensorflow'
```

Gambar 9.77 terjadi Error 3

4. terjadi error module 'scipy' has no attribute 'signal', seperti pada gambar berikut:

```
AttributeError: module 'scipy' has no attribute 'signal'
```

Gambar 9.78 terjadi Error 4

9.4.3.2 Solusi

1. solusi dari error 1 ialah:

```
(base) C:\Windows\system32>pip install librosa
Collecting librosa
  Downloading https://files.pythonhosted.org/packages/77/b5/181786/librosa-0.7.2.tar.gz (1.6MB)
    |██████████| 1.2MB 187kB/s eta 0:00:03
```

Gambar 9.79 solusi error 1

2. solusi dari error 2 ialah:

```
(base) C:\Windows\system32>pip install keras
Collecting keras
  Downloading https://files.pythonhosted.org/p/Keras-2.3.1-py2.py3-none-any.whl (377kB)
```

Gambar 9.80 solusi error 2

3. solusi dari error 3 ialah:

```
(base) C:\Windows\system32>pip install tensorflow
Collecting tensorflow
  Downloading https://files.pythonhosted.org/packages/3
/tensorflow-2.1.0-cp37-cp37m-win_amd64.whl (355.8MB)
    |                                                 1.5MB 91kB/s et
```

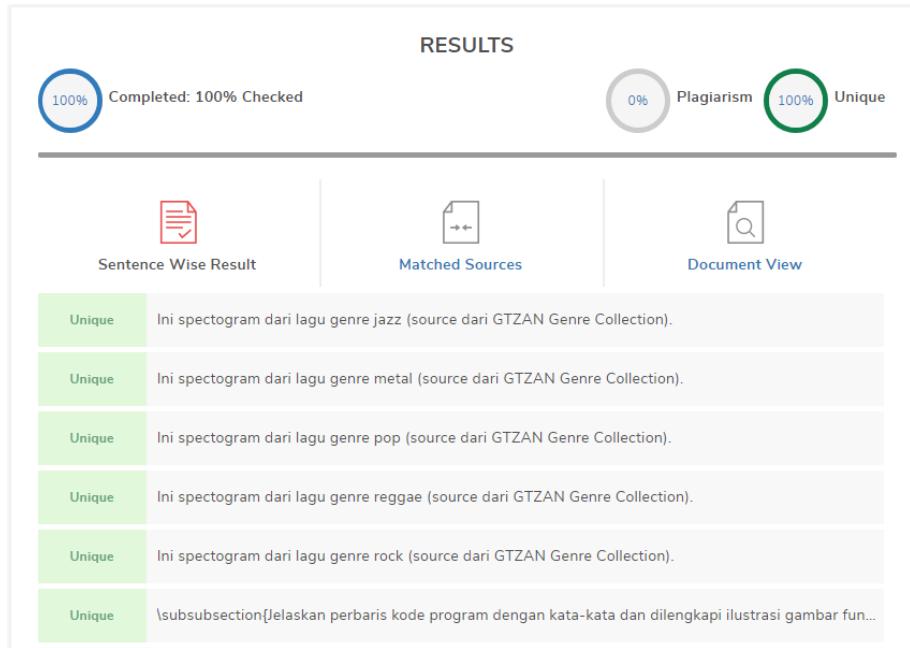
Gambar 9.81 solusi error 3

4. solusi dari error 4 ialah menambahkan kode berikut:

```
from scipy import signal
```

Gambar 9.82 solusi error 4

9.4.4 Bukti Tidak Plagiat

**Gambar 9.83** Bukti tidak plagiat

9.4.5 Link Youtube

<https://youtu.be/RgBkUfYZHgY>

9.5 Arrizal Furqona Gifary / 1174070

9.5.1 Teori

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

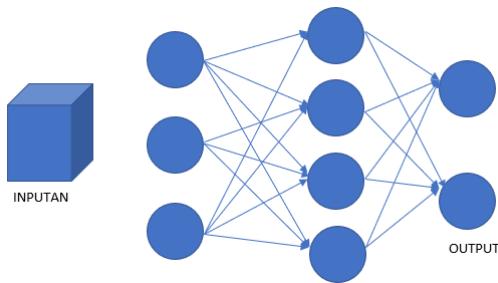
digunakan untuk mengidentifikasi jenis suara misalkan jenis suara gendre lagu jes pop metal dan klasikal atau suara ultra sonic.



Gambar 9.84 Ilustrasi gambar metode MFCC

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

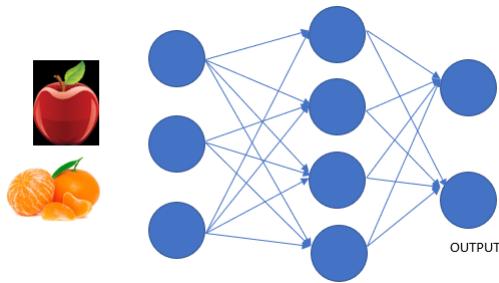
konsep neural network dilakukan ada inputan pasti ada outputan sesuai dengan kategori inputan dan fungsi di dalamnya.



Gambar 9.85 Ilustrasi Konsep dasar neural network

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

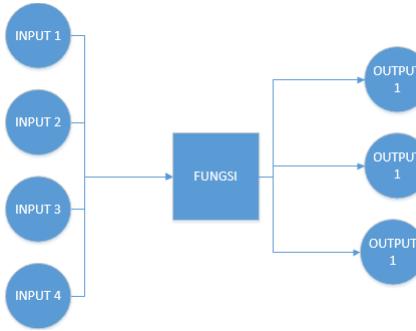
pembobotan dalam neural network yaitu digunakan untuk membedakan objek inputan atau variabel inputan untuk AI misalkan apel dan jeruk digunakan untuk variabel inputan maka dibuat pembobotan antara kedua benda tersebut untuk menentukan output yang pasti dari inputan yang dilakukan.



Gambar 9.86 Ilustrasi Konsep pembobotan pada neural network

4. Jelaskan konsep aktifitas dalam neural network. dilengkapi dengan ilustrasi gambar.

cara aktifitas dalam neural network dilakukan terhadap input pada neural network inputan tersebut dimasukan kepada fungsi pada mesin misalkan fungsi $\tanh(x)$ sehingga di hasilkanlah output yang sesuai dengan fungsi tersebut.

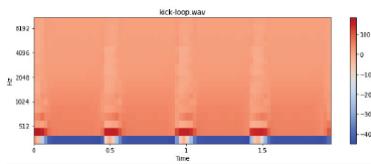


Gambar 9.87 Gambar yang dibaca hasil plotnya

5. Jelaskan cara membaca hasil plot dari MFCC dilengkapi dengan ilustrasi gambar.

cara membaca hasil plotting dari MFCC yaitu tentukan terlebih dahulu batas minimal Hz dari gelombang suara dan batas maksimal dari suara tersebut. kemudian warna yang paling pekat merupakan hasil dari pengolahan data tersebut misalkan muncul warna orange pekat di bagian bawah dan orange muda di bagian atas yang berarti suara tersebut kuat bagian basnya dan biasanya juga antara warna yang pekat tersebut ada jarak.

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



Gambar 9.88 Ilustrasi Cara Membaca Hasil Plot

one-hot encoding merupakan pemberian nilai pada suatu variabel jika nilai itu iya maka nilainya satu dan jika tidak maka nilainya nol.

	pop	rock	blues	rege	clasical
Lagu 1	1	0	0	0	0
Lagu 2	1	0	0	0	0
Lagu 3	0	1	0	0	0
Lagu 4	0	0	1	0	0
Lagu 5	0	0	0	1	0
Lagu 6	0	0	0	0	1
Lagu 7	0	0	0	0	1

Gambar 9.89 Ilustrasi Konsep one-hot encoding

7. Jelaskan apa dari np.unique dan to_categorical dalam kode program, dilengkapi dengan ilustrasi atau gambar.
digunakan untuk membuat array sedangkan to_categorical digunakan untuk membuat matrix bauititu 64 bit atau 32 bit.

```
>>> 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 9.90 Ilustrasi np.unique

to_categorical

```
keras.utils.to_categorical(y, num_classes=None, dtype='float32')
```

Gambar 9.91 Ilustrasi to_categorical

8. Jelaskan apa fungsi dari Sequential dalam kode program, dilengkapi dengan ilustrasi atau gambar.
sequential adalah proses perbandingan setiap elemen satu persatu mulai dari dari objek pertama hingga yang di tuju atau jika mencari angka 100

maka sequential akan membagi bagian misalnya dari satu sampai 20 dan seterusnya sampai mendapat nilai seratus.

Bobot 1	Bobot 2	Bobot 3	Bobot 4	Bobot 5
1-20	21-40	41-60	61-80	81-100

Gambar 9.92 Ilustrasi Konsep pembobotan pada neural network

9.5.2 Praktikum

1. Jelaskan isi dari data GTZAN Genre Collection dan data dari freesound. Buat kode program untuk meload data tersebut untuk digunakan pada MFCC. Jelaskan arti dari perbaris kode yang dibuat (harus beda dengan teman satukelas).

Isi data data merupakan datasets lagu atau suara yang tersirri dari 10 gendre yang di simpan kedalam 10 folder yaitu folder blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, dan rock ke sepu-luh folder tersebut masing-masing berisi 100 data suara sedangkan data freesound merupakan contoh data suara yang akan di gunakan untuk menguji hasil pengolahan data tersebut dengan menggunakan metode mfcc. apakah suara dari freesound termasuk kategori jazz pop atau sebagainya ?.

```

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.np_utils import to_categorical
10
11 # In[1]: buat fungsi mfcc untuk ngetest ajah
12 def display_mfcc(song):
13     y, _ = librosa.load(song)
14     mfcc = librosa.feature.mfcc(y)
15
16     plt.figure(figsize=(10, 4))
17     librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
18     plt.colorbar()
19     plt.title(song)
20     plt.tight_layout()
21     plt.show()
```

dapat dilihat pada kode diatas pada baris kesatu dilakukan import librosa tang digunakan untuk fungsi mfcc pada suara. pada baris kedua dilakukan import librosa feature dan pada baris ke tiga dilakukan librosa

display selanjutnya pada baris ke empat dilakukan import glob kemudian insert numpy untuk pengolahan data menjadi vektor setelah itu dilakukan import matplotlib untuk melakukan plotting setelah itu dilakukan import librari keras.

Selanjutnya yaitu membuat fungsi mfcc dengan nama display_mfcc yang didalamnya terdapat variabel y yang berisi method librosa load kemudian variabel mfcc yang berisi method librosa featurea mfcc. Setelah itu membuat float figure dengan ukuran 10 banding 4 kemudian di isi oleh data librosa display dengan variabel x nya yaitu waktu dan y yaitu mel atau Hz kemudian melakukan plot warna setelah itu melakukan plot judul dan terakhir float di tampilkan.

2. Jelaskan perbaris kode program dengan kata-kata dan di lengkapi ilustrasi gambar fungsi dari display_mfcc().

```

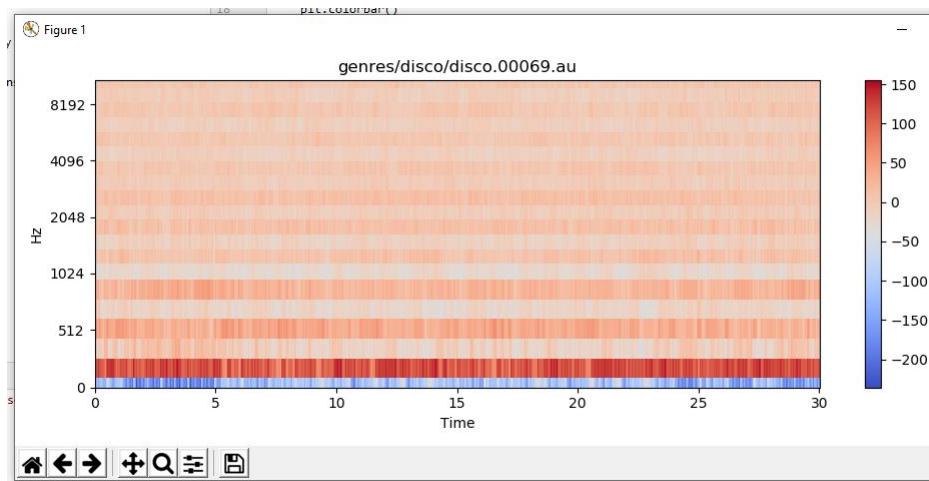
1 # In [2]: cek fungsi
2 display_mfcc ('genres/disco/disco.00069.au')
3 # In [2]: cek fungsi
4 display_mfcc ('genres/blues/blues.00069.au')
5 # In [2]: cek fungsi
6 display_mfcc ('genres/classical/classical.00069.au')
7 # In [2]: cek fungsi
8 display_mfcc ('genres/country/country.00069.au')
9 # In [2]: cek fungsi
10 display_mfcc ('genres/hiphop/hiphop.00069.au')
11 # In [2]: cek fungsi
12 display_mfcc ('genres/jazz/jazz.00069.au')
13 # In [2]: cek fungsi
14 display_mfcc ('genres/pop/pop.00069.au')
15 # In [2]: cek fungsi
16 display_mfcc ('genres/reggae/reggae.00069.au')
17 # In [2]: cek fungsi
18 display_mfcc ('genres/rock/rock.00069.au')
```

pada baris ke dua program diatas digunakan untuk mendisplay tampilan glombang suara dari file 266093_stereo-surgeon_kick-loop-5.wav menggunakan metode mfcc dengan menggunakan fungsi display_mfcc yang telah tadi di buat pada nomer dua begitu juga pada baris ke 4 6 8 sampai ke 22 secara teksis sama menggunakan fungsi display_mfcc hanyasaja beda penyimpanan data yang akan di tampilkan atau di eksekusi. untuk contoh hasilnya dapat dilihat pada gambar ?? berikut:

Gambar tersebut merupakan hasil dari mfcc dari salah satu gender lagu pop yang ada pada datasets yang 1000 atau terdapat dalam sepuluh folder tadi.

3. Jelaskan perbaris dengan kata-kata dan dilengkapi dengan ilustrasi gambar fungsi dari extract_features_song jelaskan kenapa data yangdiambil merupakan data 25.000 baris pertama ?

```
1 def extract_features_song (f):
```



Gambar 9.93 Ilustrasi gambar fungsi dari `display_mfcc()`

```

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 baris ke tiga di definisikan nama `extract_features_song` yang nantinya akan di gunakan pada fungsi yang lainnya kemudian dibuat variabel `y` dengan method `librosa load` setelah itu dibuat variabel baru `mfcc` dengan isi `librosa features mfcc` dengan isi variabel `y` tadi kemudian dibuat variabel `mfcc` dengan isian `np.max` dan variabel `mfcc` tadi terakhir di buat array dari data tersebut merupakan data 25000 data pertama. kenapa data 25000 pertama yang digunakan dikarenakan data tersebut digunakan sebagai data testing semakin besar data testing yang di gunakan maka semakin akurat hasil AI. tapi sebenarnya data tersebut relatif bisa lebih besar atau lebih kecil tergantung pada komputer masing masing.

4. Jelaskan Perbaris kode program dengan kata-kata dan lengkapi ilustrasi gambar fungsi dari `generate_features_and_labels`.

```

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' ]

```

```

6   for genre in genres:
7       sound_files = glob.glob('genres/' + genre + '/*.au')
8       print('Processing %d songs in %s genre...' % (len(
9           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
18      label_row_ids = label_row_ids.astype(np.int32, copy=False)
19      onehot_labels = to_categorical(label_row_ids, len(
20          label_uniq_ids)) #ke one hot
21
22  return np.stack(all_features), onehot_labels

```

pada baris ke tiga merupakan pendefinisian nama fungsi yaitu generate features and labels kemudian membuat variabel baru dengan array kosong yaitu all_features dan all_labels kemudian mendefinisikan isian label untuk gendre dengan cara membuat variabel genres kemudian di isi dengan 10 gendre yang tadi setelah itu dilakukan fungsi if else dengan code for dan in setelah itu akan di buat encoding untuk data tiap tiap label contoh untuk blues 1000000000 dan untuk clasical 0100000000.

- Jelaskan dengan kata dan praktek kenapa penggunaan fungsi generate features and labels sangat lama saat meload dataset gendre tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

halnini menjadi lama dikarenakan mesin membaca satupersatu file yang ada pada folder dan dalam foldertersebut terdapat 100 file sehingga wajar menjadi lama ditambah lagi mengolah data yang tadinya suara menjadi bentuk vektor. berikut merupakan codenya.

```

1 # In [3]: passing parameter dari fitur ekstraksi menggunakan
2                         mfcc
3 features, labels = generate_features_and_labels()

```

- jelaskan kenapa harus dilakukan pemisahan data training dan data testing sebesar 80 persen praktekan dengan kode dan tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut. untuk code nya adalah sebagai berikut yang merupakan code untuk membagi data sebanyak 80 persen untuk data training maka data musik tadi yang total jumlahnya 1000 akan di bagi dua untuk data training sebanyak 800 dan 200 untuk data testing.

```

1 # In [3]: fitur ekstraksi
2 training_split = 0.8

```

```

57     features, labels = generate_features_and_labels()
      generate features and labels()
Run: 1 ×
  Using TensorFlow backend.
  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 pop genre...
  Processing 100 songs in reggae genre...
  Processing 100 songs in rock genre...

Process finished with exit code 0

```

Gambar 9.94 Hasil dari fungsi generate features and labels

- praktekan dan jelaskan masing masing parameter dari fungsi Sequential(). tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

fungsi sequential digunakan untuk mengolah data inputan sesuai dengan fungsi yang ada pada fungsi sequential pada fungsi sequential kali ini menggunakan dua fungsi sequential mengkompile data dari 100 neuron atau dari 1 folder file dengan menggunakan fungsi relu dan softmax untuk menghasilkan outputan yang sesuai dengan keriteria.

```

1 # In [3]: membuat seq NN, layer pertama dense dari 100 neurons
2 model = Sequential([
3     Dense(100, input_dim=np.shape(train_input)[1]),
4     Activation('relu'),
5     Dense(10),
6     Activation('softmax'),
7 ])

```

- praktekan dan jelaskan masing masing parameter dari fungsi compile(). tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

yaitu fungsi kompile yang digunakan untuk mengetahui parameter yang digunakan dari data yang telah diolah untuk caranya dapat menggunakan codingan sebagai berikut, pada gambar tersebut memunculkan parameternya berapasaja dan total parameter yang digunakan.

```

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

```

- praktekan dan jelaskan masing masing parameter dari fungsi fit(). tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

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		

Process finished with exit code 0

Gambar 9.95 Hasil fungsi compile

pada fungsi ini dilakukan pengolahan data dari 10 label tadi atau 10 file data sets tadi kemudian di hitung tingkat akurasi masing masing dan tingkat kegagalan atau loss data darisetiap file tersebut caranya dengan melakukan codingan berikut. pada gambar tersebut menunjukan 10 pengolahan data untuk menentukan nilai akurasi dan loss dari data tersebut dan selanjutnya dilakukan fingsi evaluasi.

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

32/640 [>.....] - ETA: 1s - loss: 0.2331 - accuracy: 1.0000
 64/640 [=>.....] - ETA: 1s - loss: 0.2113 - accuracy: 1.0000
 96/640 [==>.....] - ETA: 1s - loss: 0.2268 - accuracy: 0.9896
 128/640 [====>.....] - ETA: 1s - loss: 0.2172 - accuracy: 0.9922
 160/640 [=====>.....] - ETA: 1s - loss: 0.2371 - accuracy: 0.9812
 192/640 [=====>.....] - ETA: 0s - loss: 0.2391 - accuracy: 0.9844
 224/640 [=====>.....] - ETA: 0s - loss: 0.2484 - accuracy: 0.9821
 256/640 [=====>.....] - ETA: 0s - loss: 0.2529 - accuracy: 0.9727
 288/640 [=====>.....] - ETA: 0s - loss: 0.2549 - accuracy: 0.9722
 320/640 [=====>.....] - ETA: 0s - loss: 0.2469 - accuracy: 0.9719
 352/640 [=====>.....] - ETA: 0s - loss: 0.2512 - accuracy: 0.9688
 384/640 [=====>.....] - ETA: 0s - loss: 0.2525 - accuracy: 0.9661
 416/640 [=====>.....] - ETA: 0s - loss: 0.2508 - accuracy: 0.9688
 448/640 [=====>.....] - ETA: 0s - loss: 0.2511 - accuracy: 0.9710
 480/640 [=====>.....] - ETA: 0s - loss: 0.2503 - accuracy: 0.9708
 512/640 [=====>.....] - ETA: 0s - loss: 0.2519 - accuracy: 0.9727
 544/640 [=====>.....] - ETA: 0s - loss: 0.2543 - accuracy: 0.9724
 576/640 [=====>.....] - ETA: 0s - loss: 0.2503 - accuracy: 0.9740
 608/640 [=====>.....] - ETA: 0s - loss: 0.2602 - accuracy: 0.9720
 640/640 [=====>.....] - 1s 2ms/step - loss: 0.2544 - accuracy: 0.9734 - val_loss: 1.2366 - val_accuracy: 0.5625

Process finished with exit code 0

Gambar 9.96 Hasil fungsi fit

10. praktekan dan jelaskan masing masing parameter dari fungsi evaluate(). tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

pada fungsi ini dilakukan evaluasi terhadap datayang telah di runing sebelumnya untuk lebih jelasnya dapat di lihat codingan tersebut pada codingan tersebut dilakukan evaluasi pada tingkat kegagalan dan akurasi kebenaran maka hasilnya munculkan hasil evaluasi dari 10 proses dari setiap gendre yaitu akurasi sebesar 51 persen dan loss data sebesar 1.4105 data.

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

```
Run: 1 ×
32/200 [====>.....] - ETA: 5s
200/200 [=====] - 1s 6ms/step
Done!
Loss: 2.3352, accuracy: 0.0800
Process finished with exit code 0
```

Gambar 9.97 Hasil fungsi evaluasi

11. praktekan dan jelaskan masing masing parameter dari fungsi predict tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

fungsi predict merupakan fungsi untuk membandingkan tingkat akurasi pada setiap label yang sepuluh tadi maka data akan di sandingkan ke masing masing tingkat akurasinya, yang akurasinya paling tinggi maka itulah jawaban untuk setiap inputan yang dilakukan.

```
1 # In [3]: fitur_ekstraksi
2 model.predict(test_input[:1])
```

```
Run: 1 ×
32/200 [====>.....] - ETA: 5s
200/200 [=====] - 1s 6ms/step
Done!
Loss: 2.3352, accuracy: 0.0800
Process finished with exit code 0
```

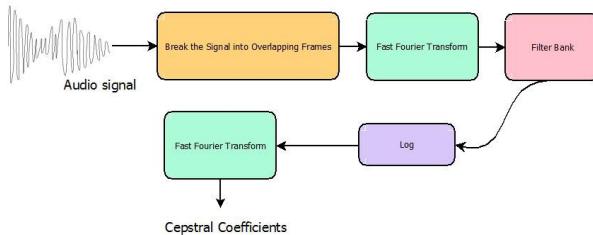
Gambar 9.98 Hasil fungsi prediksi

9.6 1174079 - Chandra Kirana Poetra

9.6.1 Teori

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

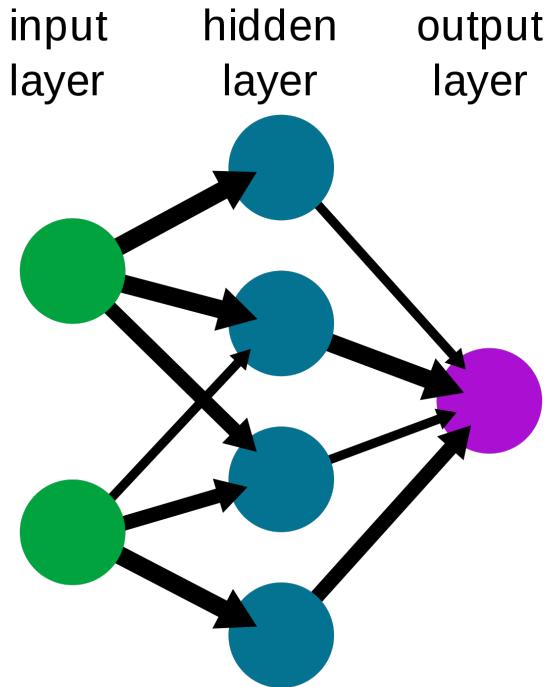
Mel Frequency Cepstral Coefficient (MFCC) berguna untuk melakukan proses ekstraksi data khususnya audio dan juga merupakan suatu fitur yang ada di aplikasi pengenalan suara yang bisa digunakan untuk men- genali kata atau angka yang dikenal.



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

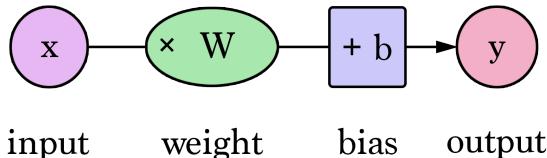
Neural network merupakan suatu set algoritma yang dibuat berdasarkan otak manusia. didesain untuk mengenali pola, neural network meng- intepresentasikan data sensor melalui berbagai macam persepsi mesin, melabeli atau mengelompokkan input. pola yang dapat dikenali dapat berupa angka atau vektor atau bisa juga data real seperti gambar atau suara dan juga teks.

A simple neural network

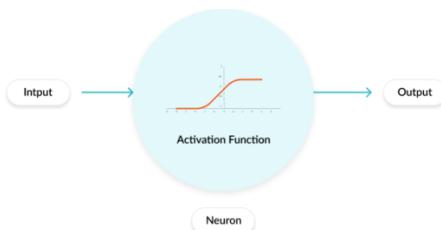


3. Jelaskan konsep pembobotan dalam neural network. dilengkapi dengan ilustrasi atau gambar.

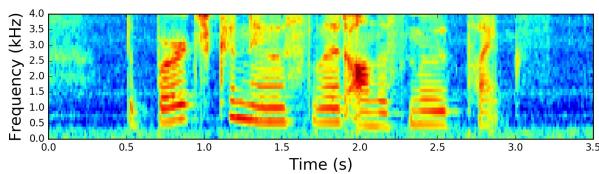
Weight merupakan parameter dalam neural network yang berguna untuk melakukan transformasi data input yang akan masuk kebagian hidden layer dari neural network. suatu neural network merupakan sekelompok dari nodes atau neuron. di dalam tiap node ada beberapa set instruksi, bobot, dan juga bias value. suatu input masuk ke bagian node kemudian di jumlahkan dengan value dari weight dan hasilnya akan di lihat atau di berikan ke layer berikutnya yang ada pada neural network. seringkali bobot yang ada pada neural network sering berada pada bagian hidden layer di neural network



4. Jelaskan konsep fungsi aktifasi dalam neural network. dilengkapi dengan ilustrasi atau gambar.
- function aktifasi merupakan sebuah rumus matematika yang ada diantara input neuron dan output yang akan menuju ke layer berikutnya. Hal ini bisa sesimple seperti langkah langkah function yang berfungsi untuk menyalakan atau mematikan suatu output dari neuron yang tergantung oleh beberapa aturan.



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



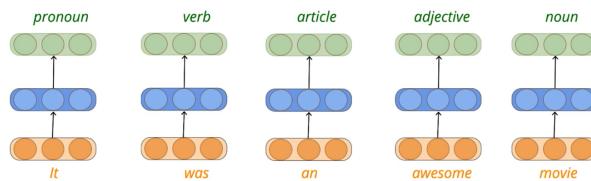
Sumbu y digunakan untuk mengukur tingginya khz, sementara sumbu x digunakan untuk mengukur durasi dari audio tersebut.

6. Jelaskan apa itu one-hot encoding, dilengkapi dengan ilustrasi kode dan atau gambar. One hot encoding digunakan untuk memisahkan columnm

yang berisi kategori berdasarkan angka menjadi ke beberapa column berdasarkan jumlah kategori yang ada. setiap kolumn berisi 0 atau 1. one hot encoding dipakai untuk kasus kasus seperti misalkan pada suatu dataset, kita menemukan kolumn yang berisi angka tanpa susunan. data ini biasanya menggambarkan kategori atau value dari suatu kategori. ini akan membuat machine learning kebingungan sehingga kita butuh yang namanya one hot encoding agar mesin mengerti.

Label Encoding			One Hot Encoding			
Food Name	Categorical #	Calories	Apple	Chicken	Broccoli	Calories
Apple	1	95	1	0	0	95
Chicken	2	231	0	1	0	231
Broccoli	3	50	0	0	1	50

7. Jelaskan apa fungsi dari Sequential dalam kode program,dilengkapi dengan ilustrasi atau gambar.
 Sequential digunakan untuk membuat suatu prediksi dari data yang dimasukkan. contohnya seperti gambar dibawah ini yang digunakan untuk memprediksi adjective, noun,verb dan pronoun dari teks bahasa inggris



9.6.2 Praktek

1. Jelaskan isi dari data GTZAN Genre Collection dan data dari freesound. Buat kode program untuk meload data tersebut untuk digunakan pada MFCC. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas).

GTZAN Genre Collection merupakan sebuah koleksi lagu dari berbagai jenis genre yang durasinya rata rata 30 detik.

Data dari freesound yang chandra gunakan ada 2 yaitu kick.wav dan juga spaceysynth.wa.

Kode program untuk meload data tersebut untuk digunakan pada MFCC.

```

1 #%%Soal1
2 import librosa
3 import librosa.feature
4 import librosa.display
5 import glob
6 import numpy as np

```

```

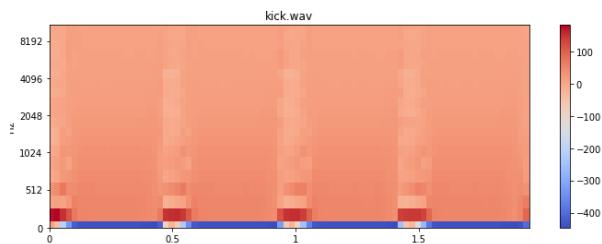
7 import matplotlib.pyplot as plt
8 from keras.models import Sequential
9 from keras.layers import Dense, Activation
10 from keras.utils.np_utils import to_categorical
11
12 def display_mfcc(song):
13     y, _ = librosa.load(song)
14     mfcc = librosa.feature.mfcc(y)
15
16     plt.figure(figsize=(10, 4))
17     librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
18     plt.colorbar()
19     plt.title(song)
20     plt.tight_layout()
21     plt.show()
22
23 display_mfcc('kick.wav')
24
25 display_mfcc('spaceysynth.wav')
26
27 display_mfcc('country.00000.au')
28
29 display_mfcc('blues.00000.au')

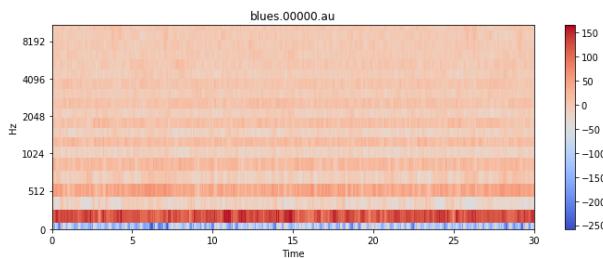
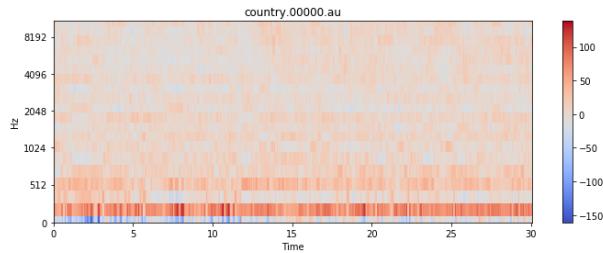
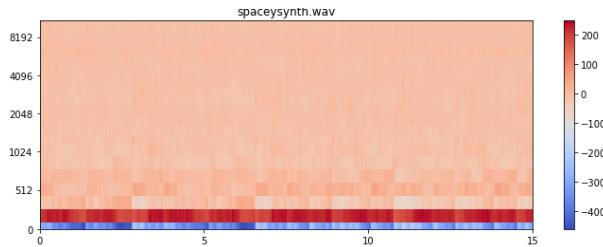
```

Import librarynya terlebih dahulu seperti librosa, glob dan numpy serta matplotlib dan juga keras. Librosa digunakan untuk ekstrak fitur yang ada pada lagu, glob digunakan untuk membuat list file, sedangkan numpy untuk melakukan operasi yang berhubungan dengan angka dan matplotlib untuk membuat grafik. import juga sequential dan lain lain dari library keras untuk penggunaan function activation serta dense yang memiliki banyak sekali neuron didalamnya serta tocategorycal untuk merubah vector menjadi binary

Setelah itu buat class dengan parameter bernama song yang isinya berisi function untuk load lagu dan fitur serta spectrogramnya

Berikut hasilnya





2. Jelaskan perbaris kode program dengan kata-kata dan dilengkapi ilustrasi gambar fungsi dari displaymfcc()

```

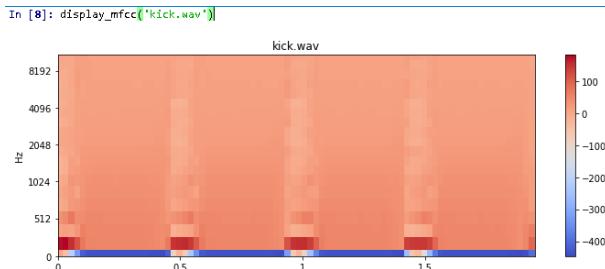
1 #%%Soal2
2 def display_mfcc(song):
3     y, _ = librosa.load(song)
4     mfcc = librosa.feature.mfcc(y)
5
6     plt.figure(figsize=(10, 4))
7     librosa.display.specshow(mfcc, x_axis='time',
8     y_axis='mel')
9     plt.colorbar()
10    plt.title(song)

```

```
10     plt.tight_layout()
11     plt.show()
```

- Nama fungsinya adalah displaymfcc dengan satu parameter bernama song.
- Fungsi load() yang dipanggil digunakan untuk meload lagunya yang kemudian ditampung di variable.
- Fungsi mfcc() dipakai untuk melakukan ekstrak fitur file lagu yang dimuat kemudian ditampung pada variable nilainya.
- Fungsi figure() digunakan untuk membuat grafik.
- Fungsi specshow() untuk membuat spectrogram
- Fungsi colorbar() digunakan untuk memberi warna pada grafik
- Fungsi title() untuk memberikan judul pada grafik.
- Fungsi tightlayout() untuk menyesuaikan layout grafik.
- Fungsi show() menampilkan output grafik.

Jika kita memanggil fungsi displaymfcc() nantinya akan seperti ini.



3. Jelaskan perbaris kode program dengan kata-kata dan dilengkapi ilustrasi gambar fungsi dari extractfeaturesong(). Jelaskan juga mengapa yang diambil 25.000 baris pertama?

```
1 def extract_features_song(f):
2     y, _ = librosa.load(f)
3     # get Mel-frequency cepstral coefficients
4     mfcc = librosa.feature.mfcc(y)
5     # normalize values between -1,1 (divide by max)
6     mfcc /= np.amax(np.absolute(mfcc))
7     return np.ndarray.flatten(mfcc)[:25000]
8
9 extract_features_song('blues.00000.au')
```

- Pertama-tama nama fungsinya adalah extractfeaturesong dengan parameter bernama f
- Fungsi load() dipakai untuk memuat lagunya yang kemudian disimpan di variable.
- Fungsi mfcc() untuk melakukan ekstrak fitur file lagu yang dimuat lalu disimpan nilainya di variable.
- Fungsi absolute() digunakan supaya nilainya menjadi nilai absolut. Sedangkan fungsi amax dipakai untuk mencari nilai max. Lalu hasil yang didapatkan sebelumnya dibagi dengan nilai mfcc.
- Terakhir, hasil yang telah ditampung di variable mfcc di convert ke array 1 dimensi dan diambil 25000 baris pertama saja mengembalikan hasil convert array ke dalam bentuk satu dimensi dari nilai mfcc dan diambil 25000 baris pertama.
- Alasannya mengapa hanya diambil 25000 baris pertama karena hampir setiap lagu memiliki durasi atau panjang yang berbeda beda se mentara data yang harus dimasukan ke machine learning harus ukuran yang sama.

```

....: extract_features_song('blues.
00000.au')
Out[12]:
array([-0.81999071, -0.81014303, -0.75184401,
..., -0.01818394,
         0.01827243,  0.01200242])

```

4. Jelaskan perbaris kode program dengan kata-kata dan dilengkapi ilustrasi gambar fungsi dari generatefeaturesandlabels().

```

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('*.*au')
9         print('Processing %d songs in %s genre...' % (len(
10            sound_files), genre))
11         for f in sound_files:
12             features = extract_features_song(f)
13             all_features.append(features)
14             all_labels.append(genre)
15
16     # convert labels to one-hot encoding

```

```

15     label_uniq_ids , label_row_ids = np.unique( all_labels ,
16         return_inverse=True)
17     label_row_ids = label_row_ids.astype(np.int32 , copy=False)
18     onehot_labels = to_categorical(label_row_ids , len(
19         label_uniq_ids))
    return np.stack(all_features) , onehot_labels
features , labels = generate_features_and_labels()

```

- Buat fungsi dengan nama generatefeaturesandlabels().
- Kemudian buat array list allfeatures
- Kemudian buat array list alllabels
- Lalu buat array list genres untuk menyimpan daftar genre musik
- Gunakan loop untuk melakukan proses secara berulang agar semua file dapat dimuat
- Pakai fungsi glob() untuk mencari file lagu yang ingin dimuat dengan mendefinisikan direktoriya
- Memperlihatkan panjang daftar lagu berdasarkan genre
- looping pada soundfiles
- Gunakan fungsi extractfeaturessong() untuk mengambil fitur yang ada pada file musik yang di load
- tambahkan hasil dari fungsi extractfeaturessong() ke allfeatures
- Isi allfeatures dengan features dan alllabels dengan genrelabels
- Gunakan fungsi unique untuk membuat label unik
- Gunakan fungsi astype untuk mengubah type menjadi type int32
- Gunakan function to_categorical untuk melakukan konversi ke one-hot-encoding
- Return nilainya kemudian gabungkan dengan fungsi stack() dari allfeatures ke matrix
- panggil fungsinya

```

.....
.... features , labels = generate_features_and_labels()
Processing 2 songs in blues genre...
Processing 2 songs in classical genre...
Processing 2 songs in country genre...
Processing 2 songs in disco genre...
Processing 2 songs in hiphop genre...
Processing 2 songs in jazz genre...
Processing 2 songs in metal genre...
Processing 2 songs in pop genre...
Processing 2 songs in reggae genre...
Processing 2 songs in rock genre...

```

5. Jelaskan dengan kata dan praktik kenapa penggunaan fungsi generate-featuresandlabels() sangat lama ketika meload dataset genre. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

Sangat lama karena file yang dimuat bisa banyak yaitu ada 1000 file serta kita juga mencari 25000 nilai pertama dari setiap file yang ada sehingga akibatnya lama, apabila hanya dua file seperti gambar dibawah ini tentunya lebih cepat.

```
.....
....: features, labels = generate_features_and_labels()
Processing 2 songs in blues genre...
Processing 2 songs in classical genre...
Processing 2 songs in country genre...
Processing 2 songs in disco genre...
Processing 2 songs in hiphop genre...
Processing 2 songs in jazz genre...
Processing 2 songs in metal genre...
Processing 2 songs in pop genre...
Processing 2 songs in reggae genre...
Processing 2 songs in rock genre...
```

6. Jelaskan kenapa harus dilakukan pemisahan data training dan data set sebesar 80 persen? Praktekkan dengan kode dan Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

Karena agar nantinya data yang akan kita hitung menghasilkan hasil yang lebih akurat dan juga meminimalisir kemungkinan kesalahan oleh mesinnya itu sendiri.

```
1 print(np.shape(features))
2 print(np.shape(labels))
3 training_split = 0.8
4 # last column has genre, turn it into unique ids
5 alldata = np.column_stack((features, labels))
6 np.random.shuffle(alldata)
7 splitidx = int(len(alldata) * training_split)
8 train, test = alldata[:splitidx, :], alldata[splitidx:, :]
9 print(np.shape(train))
10 print(np.shape(test))
11 train_input = train[:, :-10]
12 train_labels = train[:, -10:]
13 test_input = test[:, :-10]
14 test_labels = test[:, -10:]
15 print(np.shape(train_input))
16 print(np.shape(train_labels))
```

- (20, 25000)
 (20, 10)
 (16, 25010)
 (4, 25010)
 (16, 25000)
 (16, 10)

- Ukuran array atau jumlah data pada data features
- Ukuran array atau jumlah data pada data labels
- Ukuran array atau jumlah data pada data train
- Ukuran array atau jumlah data data test
- Ukuran array atau jumlah data data traininput
- Ukuran array atau jumlah data data trainlabels

7. Praktekkan dan jelaskan masing-masing parameter dari fungsi Sequential(). Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```

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

```

Layer pertama terdapat 100 neuron dengan parameter berjumlah 25000 yang didapat dari variable train input dengan neuron sebanyak 100, layer kedua memiliki 10 neuron dengan parameter sekitar 1010 input ditambah biasnya 10. Dengan total params 250110. Setelah itu data dilakukan training.

- Layer pertama adalah layer dense dengan jumlah 100 neuron, kemudian inputkan datanya
- Eksekusi activation dengan tipe relu
- Lalu ada layer kedua dengan jumlah sebanyak 10 neuron
- Melakukan fungsi activation dengan tipe softmax

```
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
```

8. Praktekkan dan jelaskan masing-masing parameter dari fungsi compile(). Tunjukkan keluarannya dengan fungsi summary dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

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

- Optimizer yang dipakai yaitu adam
- Loss dipakai adalah categoricalexentropy
- Metrics yang dipakai accuracy

```
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
```

Layer pertama terdapat 100 neuron dengan parameter berjumlah 25000 yang didapat dari variable train input dengan neuron sebanyak 100, layer kedua memiliki 10 neuron dengan parameter sekitar 1010 input ditambah biasnya 10. Dengan total params 2501110.

9. Praktekkan dan jelaskan masing-masing parameter dari fungsi fit(). Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```
1 model.fit(train_input, train_labels, epochs=10, batch_size
           =32, validation_split=0.2)
```

- Parameter yang pertama adalah variable train input
- Parameter yang pertama adalah variable train labels
- Jumlah epoch yang dipakai adalah 10
- Batch Size yang dipakai sebesar 32
- Validation Splitnya adalah 0.2 atau 20%

```
In [20]: model.fit(train_input, train_labels, epochs=10,
batch_size=32, validation_split=0.2)
Train on 12 samples, validate on 4 samples
Epoch 1/10
12/12 [=====] - 0s 4ms/step -
loss: 1.9453 - accuracy: 0.1667 - val_loss: 10.9742 -
val_accuracy: 0.0000e+00
Epoch 2/10
12/12 [=====] - 0s 4ms/step -
loss: 1.9359 - accuracy: 0.1667 - val_loss: 11.0833 -
val_accuracy: 0.0000e+00
Epoch 3/10
12/12 [=====] - 0s 3ms/step -
loss: 1.9135 - accuracy: 0.1667 - val_loss: 11.2846 -
val_accuracy: 0.0000e+00
Epoch 4/10
12/12 [=====] - 0s 4ms/step -
loss: 1.8996 - accuracy: 0.1667 - val_loss: 11.5726 -
```

10. Praktekkan dan jelaskan masing-masing parameter dari fungsi evaluate(). Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```
1 print("Done!")
2 print("Loss: %.4f, accuracy: %.4f" % (loss , acc))
```

- Parameter yang pertama adalah variable test input
- Parameter yang kedua adalah variable test labels
- Batch Size yang dibutuhkan 32

```
In [22]: loss, acc = model.evaluate(test_input,
test_labels, batch_size=32)
...: print("Done!")
...: print("Loss: %.4f, accuracy: %.4f" % (loss, acc))
4/4 [=====] - 0s 750us/step
Done!
Loss: 11.3426, accuracy: 0.0000
```

Lossnya 11.3426,dan akurasinya 0.000

11. Praktekkan dan jelaskan masing-masing parameter dari fungsi predict(). Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```
1 print(prediction)
```

- Parameter yang pertama adalah variable test input

- Batch Size yang dibutuhkan 32

```
In [23]: prediction = model.predict(test_input,
batch_size=32)
....: print(prediction)
[[1.87984303e-01 2.17278704e-01 8.68563291e-07
4.76343166e-06
1.12103038e-01 1.13045134e-01 1.48350701e-01
2.88178694e-06
2.21214622e-01 1.49135822e-05]
[2.13587453e-04 1.58349320e-01 3.92913398e-06
1.15934108e-05
2.38298401e-01 1.33187622e-01 2.75805682e-01
2.39947894e-05
7.94716701e-02 1.14634179e-01]
[2.13587453e-04 1.58349320e-01 3.92913398e-06
1.15934108e-05
2.38298401e-01 1.33187622e-01 2.75805682e-01
2.39947894e-05
```

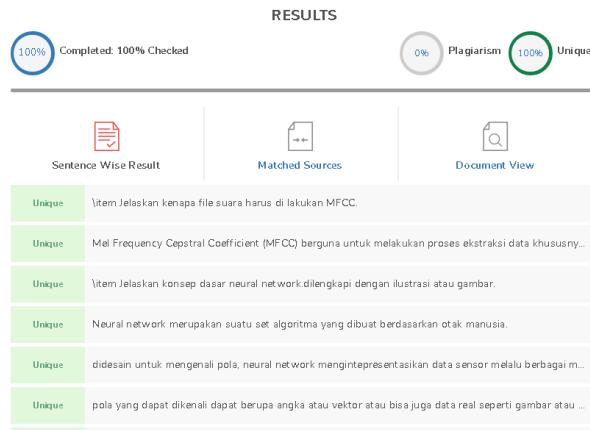
9.6.3 Penanganan Error

No module named librosa

```
C:\Users\acer>pip install librosa
Collecting librosa
  Downloading https://files.pythonhosted.org/packages/77d05/1817862d6447c211afdf15419d8418aeff000742cac275e95c74b219cbccb
/librosa-0.7.2.tar.gz (1.6MB) |████████| 1.6MB 1.7MB/s
Collecting audiostream >=2.0.0
  Downloading https://files.pythonhosted.org/packages/2e/0b/940e7861e0e9840f09dcfd72a0e9ae55f697c17c299a323f0140f913d2
/audiostream-2.0.0.tar.gz (1.4MB) |████████| 1.4MB 1.4MB/s
Requirement already satisfied: numpy >=1.15.0 in c:\users\acer\appdata\local\programs\python\python38-32\lib\site-packages
  (from librosa) (1.18.2)
Requirement already satisfied: scipy >=1.6.0 in c:\users\acer\appdata\local\programs\python\python38-32\lib\site-packages
  (from librosa) (1.4.1)
Requirement already satisfied: scikit-learn >=0.19.0, <0.19.1 in c:\users\acer\appdata\local\programs\python\python38-32\lib\site-packages
  (from librosa) (0.22.2.post1)
Requirement already satisfied: joblib >=0.12 in c:\users\acer\appdata\local\programs\python\python38-32\lib\site-packages
  (from librosa) (0.14.1)
```

9.6.3.1 Solusi Error Install library terlebih dahulu

9.6.4 Bukti Tidak Plagiat



9.6.5 Link Youtube

<https://youtu.be/LPeaSGrQSgA>

9.7 1174077 - Alvan Alvanzah

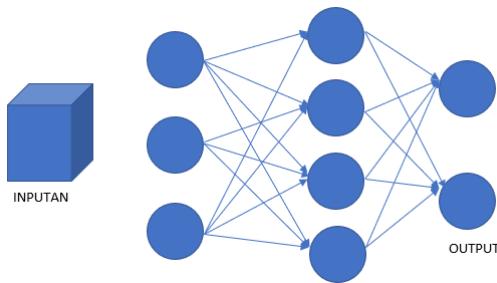
9.7.1 Teori

1. Jelaskan kenapa file suara harus dilakukan MFCC dilengkapi dengan ilustrasi atau gambar.
digunakan untuk mengidentifikasi jenis suara misalkan jenis suara gendre lagu jes pop metal dan klasikal atau suara ultra sonic.



Gambar 9.99 Ilustrasi gambar metode MFCC

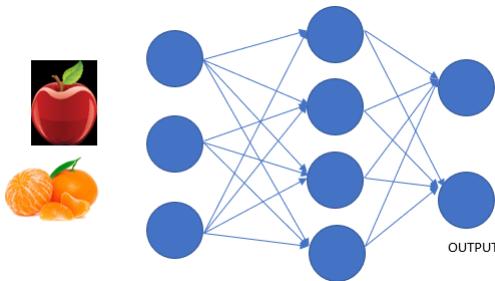
2. Jelaskan konsep dasar neural network. dilengkapi dengan ilustrasi gambar.
konsep neural network dilakukan ada inputan pasti ada outputan sesuai dengan kategori inputan dan fungsi di dalamnya.



Gambar 9.100 Ilustrasi Konsep dasar neural network

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

pembobotan dalam neural network yaitu digunakan untuk membedakan objek inputan atau variabel inputan untuk AI misalkan apel dan jeruk digunakan untuk variabel inputan maka dibuat pembobotan antara kedua benda tersebut untuk menentukan output yang pasti dari inputan yang dilakukan.



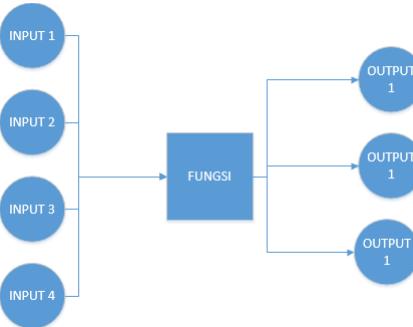
Gambar 9.101 Ilustrasi Konsep pembobotan pada neural network

4. Jelaskan konsep aktifitas dalam neural network. dilengkapi dengan ilustrasi gambar.

cara aktifitas dalam neural network dilakukan terhadap input pada neural network inputan tersebut dimasukan kepada fungsi pada mesin misalkan fungsi $\tanh(x)$ sehingga di hasilkanlah output yang sesuai dengan fungsi tersebut.

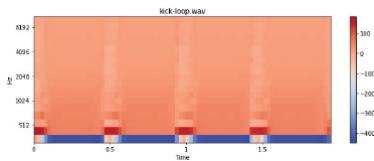
5. Jelaskan cara membaca hasil plot dari MFCC dilengkapi dengan ilustrasi gambar.

cara membaca hasil plotting dari MFCC yaitu tentukan terlebih dahulu batas minimal Hz dari gelombang suara dan batas maksimal dari suara



Gambar 9.102 Gambar yang dibaca hasil plotnya

tersebut. kemudian warna yang paling pekat merupakan hasil dari pengolahan data tersebut misalkan muncul warna orange pekat di bagian bawah dan orange muda di bagian atas yang berarti suara tersebut kuat bagian basnya dan biasanya juga antara warna yang pekat tersebut ada jarak.



Gambar 9.103 Ilustrasi Cara Membaca Hasil Plot

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

one-hot encoding merupakan pemberian nilai pada suatu variabel jika nilai itu iya maka nilainya satu dan jika tidak maka nilainya nol.

	pop	rock	blues	rege	clasical
Lagu 1	1	0	0	0	0
Lagu 2	1	0	0	0	0
Lagu 3	0	1	0	0	0
Lagu 4	0	0	1	0	0
Lagu 5	0	0	0	1	0
Lagu 6	0	0	0	0	1
Lagu 7	0	0	0	0	1

Gambar 9.104 Ilustrasi Konsep one-hot encoding

7. Jelaskan apa dari np.unique dan to_categorical dalam kode program, dilengkapi dengan ilustrasi atau gambar.

digunakan untuk membuat array sedangkan to_categorical digunakan untuk membuat matrix bauit 64 bit atau 32 bit.

```
>>> 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 9.105 Ilustrasi np.unique

to_categorical

```
keras.utils.to_categorical(y, num_classes=None, dtype='float32')
```

Gambar 9.106 Ilustrasi to_categorical

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

sequential adalah proses perbandingan setiap elemen satu persatu mulai dari dari objek pertama hingga yang di tuju atau jika mencari angka 100 maka sequential akan membagi bagian misalnya dari satu sampai 20 dan seterusnya sampai mendapat nilai seratus.

Bobot 1	Bobot 2	Bobot 3	Bobot 4	Bobot 5
1-20	21-40	41-60	61-80	81-100

Gambar 9.107 Ilustrasi Konsep pembobotan pada neural network

9.7.2 Praktek Program

1. 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 cara mengimport library yang dibutuhkan dan membuat fungsi display mfcc untuk melakukan plot pada file audio nanti. Isi data GTZAN adalah datasets lagu atau suara yang terdiri dari 10 genre yang di simpan kedalam 10 folder yaitu folder blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, dan rock ke sepuluh folder tersebut masing-masing berisi 100 data suara sedangkan data freesound merupakan contoh data suara yang akan di gunakan untuk menguji hasil pengolahan data tersebut dengan menggunakan metode mfcc. 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:

```

In [2]: import librosa
...: import librosa.feature
...: import librosa.display
...: import numpy as np
...: import matplotlib.pyplot as plt
...: from keras.layers import Dense, Activation
...: from keras.models import Sequential
...: from keras.utils import to_categorical
...:
...: 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.tight_layout()
...:     plt.show()

```

Gambar 9.108 Hasil Soal 1.

2. Soal 2

```

1 # In [2]: Soal Nomor 2
2 display_mfcc('freesound1.wav')
3 # In [2]: Soal Nomor 2
4 display_mfcc('freesound2.wav')
5 # In [2]: Soal Nomor 2
6 display_mfcc('genres/blues/blues.00023.au')
7 # In [2]: Soal Nomor 2
8 display_mfcc('genres/classical/classical.00023.au')
9 # In [2]: Soal Nomor 2
10 display_mfcc('genres/country/country.00023.au')
11 # In [2]: Soal Nomor 2
12 display_mfcc('genres/disco/disco.00023.au')
13 # In [2]: Soal Nomor 2
14 display_mfcc('genres/hiphop/hiphop.00023.au')
15 # In [2]: Soal Nomor 2

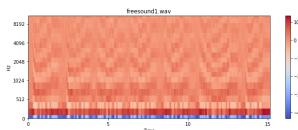
```

```

16 display_mfcc('genres/jazz/jazz.00023.au')
17 # In [2]: Soal Nomor 2
18 display_mfcc('genres/metal/metal.00023.au')
19 # In [2]: Soal Nomor 2
20 display_mfcc('genres/pop/pop.00023.au')
21 # In [2]: Soal Nomor 2
22 display_mfcc('genres/reggae/reggae.00023.au')
23 # In [2]: Soal Nomor 2
24 display_mfcc('genres/rock/rock.00023.au')

```

Kode di atas akan menampilkan hasil dari proses mfcc yang sudah dibuat fungsi pada soal 1, yaitu fungsi display mfcc akan menampilkan plot dari pembacaan file audio. Berikut adalah hasil dari salah satu pembacaan file audio :



Gambar 9.109 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]

```

Kode di atas adalah membuat fungsi yang didefinisikan dengan nama extract_features_song yang nantinya akan digunakan pada fungsi yang lainnya kemudian dibuat variabel y dengan method librosa load setelah itu dibuat variabel baru mfcc dengan isi librosa features mfcc dengan isi variabel y tadi kemudian dibuat variabel mfcc dengan isian np.max dan variabel mfcc tadi terakhir dibuat array dari data tersebut merupakan data 25000 data pertama. kenapa data 25000 pertama yang digunakan dikarenakan data tersebut digunakan sebagai data testing semakin besar data testing yang digunakan maka semakin akurat hasil AI. tapi sebenarnya data tersebut relatif bisa lebih besar atau lebih kecil tergantung pada komputer masing masing. Hasilnya adalah sebagai berikut :

```
In [8]: def extract_features_song(f):
...:     y, sr = 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 9.110 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' , '
8         hiphop' , 'jazz' , 'metal' , 'pop' , 'reggae' , 'rock' ]
9     for genre in genres:
10        sound_files = glob.glob('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    # 10000000000 classic 01000000000
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
26    return np.stack(all_features), onehot_labels
```

Kode di atas adalah mendefinisian nama fungsi yaitu generate features and labels kemudian membuat variabel baru dengan array kosong yaitu all_features dan all_labels kemudian mendefinisikan isian label untuk genre dengan cara membuat variabel genres kemudian di isi dengan 10 genre yang tadi setelah itu dilakukan fungsi if else dengan code for dan in setelah itu akan di buat encoding untuk data tiap tiap label contoh untuk blues 10000000000 dan untuk clasical 01000000000. Hasilnya adalah sebagai berikut :

```
In [8]: 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('genres/' + genre + '/*.au')
...:         for f in sound_files:
...:             features = extract_features_song(f)
...:             all_features.append(features)
...:             all_labels.append(genre)
...: 
...:     # convert labels to one-hot encoding cth blues :
...:     # 10000000000 classic 01000000000
...:     label_uniq_ids, label_row_ids = np.unique(all_labels,
...:         return_inverse=True) #ke integer
...:     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
...: 
...:     return np.stack(all_features), onehot_labels
```

Gambar 9.111 Hasil Soal 4.

5. Soal 5

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

Hal ini menjadi lama dikarenakan mesin membaca satupersatu file yang ada pada folder dan dalam folder tersebut terdapat 100 file sehingga wajar menjadi lama ditambah lagi mengolah data yang tadinya suara menjadi bentuk vektor. Hasilnya adalah sebagai berikut :

```
In [11]: 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 pop genre...
Processing 100 songs in reggae genre...
Processing 100 songs in rock genre...
In [12]: print(np.shape(features))
...: print(np.shape(labels))
(1000, 25000)
(1000, 10)
```

Gambar 9.112 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 :

```
In [13]: training_split = 0.8
```

Gambar 9.113 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() adalah 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 [20]: model = Sequential([
...:     Dense(100, input_dim=np.shape(train_input)[1]),
...:     Activation('relu'),
...:     Dense(10),
...:     Activation('softmax'),
...: ])
```

Gambar 9.114 Hasil Soal 7.

8. Soal 8

```
1 # In [8]: Soal Nomor 8
2 model.compile(optimizer='adam',
3                 loss='categorical_crossentropy',
4                 metrics=['accuracy'])
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 aktivasinya 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 [21]: model.compile(optimizer='adam',
    ...,
    loss='categorical_crossentropy',
    metrics=['accuracy'])
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: 2,581,110
Trainable params: 2,581,110
Non-trainable params: 0
```

Gambar 9.115 Hasil Soal 8.

9. Soal 9

```
1 # In [9]: Soal Nomor 9
2 model.fit(train_input, train_labels, epochs=10, batch_size=
3           =32,
4           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 persentase dari cross validation atau k-fold sebanyak 20% dari masing-masing data pengulangan, hasilnya adalah sebagai berikut :

Gambar 9.116 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 [23]: loss, acc = model.evaluate(test_input, test_labels, batch_size=32)
200/200 [=====] - 0s 656us/step
```

Gambar 9.117 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 [25]: model.predict(test_input[1])
Out[25]:
array([1.2146345e-04, 5.0897301e-06, 5.1087093e-01, 4.7763154e-01,
       1.4319259e-03, 1.4710484e-05, 1.7895336e-05, 9.3976054e-03,
       2.2317965e-05, 9.7474178e-05], dtype=float32)
```

Gambar 9.118 Hasil Soal 11.

9.7.3 Penanganan Error

1. ScreenShoot Error

```
FileNotFoundError: [Errno 2] No such file or directory: 'E:\Koleksi Musik\dataset\genres\blues\blues.00001.wv'
```

Gambar 9.119 FileNotFoundError

2. Cara Penanganan Error

- **FileNotFoundException**

Error terdapat pada letak file yang tidak terbaca, karena letak file berbeda dengan pemanggilannya, solusi nya ialah dengan meletakkan direktori file yang dibaca dengan benar.

9.7.4 Bukti Tidak Plagiat

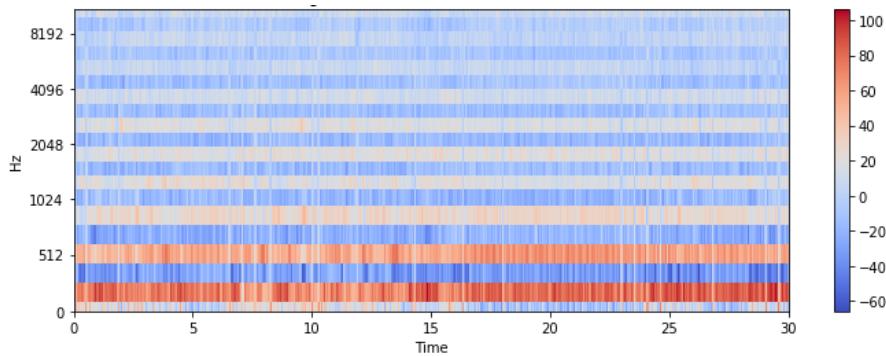


Gambar 9.120 Bukti Tidak Melakukan Plagiat Chapter 6

9.8 Muhammad Reza Syachrani - 1174084

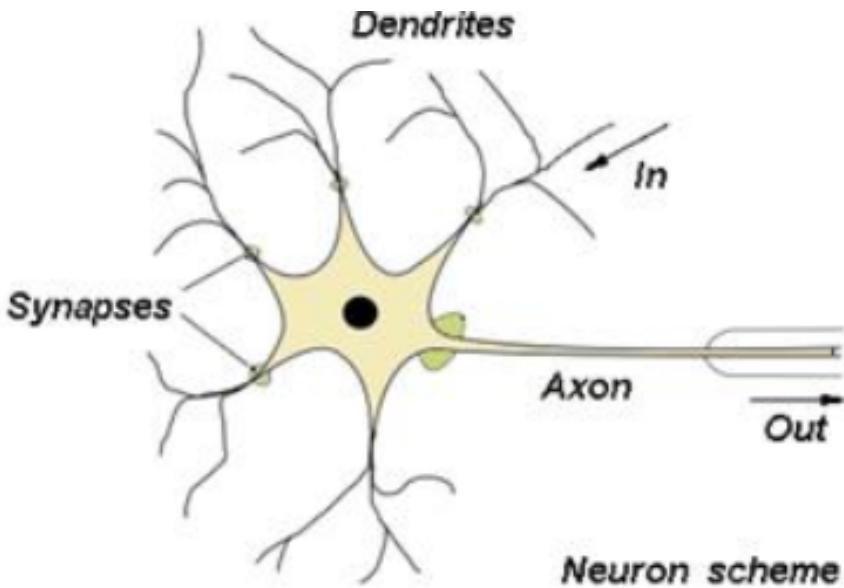
9.8.1 Teori

1. Jelaskan kenapa file suara harus di lakukan MFCC. dilengkapi dengan ilustrasi atau gambar
- digunakan untuk mendapatkan nilai dari vektor suara tersebut yang berguna untuk mendapatkan suatu parameter dan informasi mengenai ciri dari suara.



Gambar 9.121 Teori 1

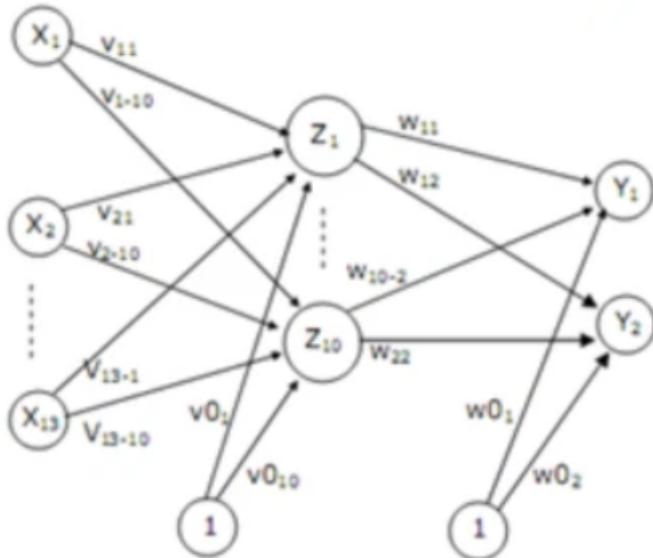
2. Jelaskan konsep dasar neural network.dilengkapi dengan ilustrasi atau gambar.
- Konsep dasar neural network sendiri di mulai dari ide dasar neural network otak manusia, dimana otak terdiri dari neuron, neuron ini berfungsi memproses setiap informasi yang masuk. Satu neuron memiliki satu akson dan minimal satu dendrit. sehingga konsep dasar ini membangun neural network buatan yang disebut (Artificial Neural Network).



Gambar 9.122 Teori 2

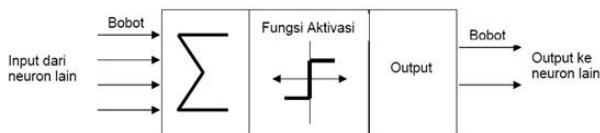
3. Jelaskan konsep pembobotan dalam neural network.dilengkapi dengan ilustrasi atau gambar
Konsep pembobotan dalam neural network menggunakan algoritma back-propagation untuk memperkecil tingkat eror dengan menyesuaikan nilai bobot berdasarkan perbedaan output serta target yang diinginkan.

Input Layer Hidden Layer Output Layer



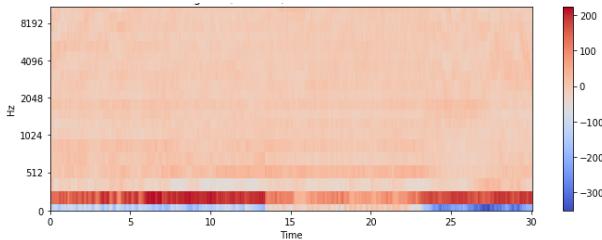
Gambar 9.123 Teori 3

4. Jelaskan konsep fungsi aktifasi dalam neural network. dilengkapi dengan ilustrasi atau gambar
merupakan fungsi matematis yang digunakan untuk mendapatkan output neuron dari nilai inputnya. Disebut suatu aktivasi karena output akan bernilai jika dapat melampaui nilai threshold-nya.



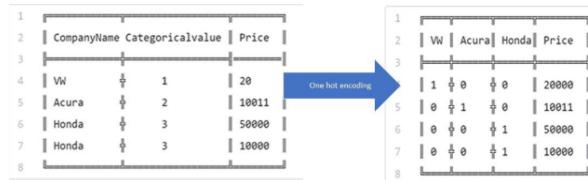
Gambar 9.124 Teori 4

5. Jelaskan cara membaca hasil plot dari MFCC,dilengkapi dengan ilustrasi atau gambar.
Dapat dilakukan dengan sampel suara pengguna diubah dalam bentuk file *.wav yang digunakan sebagai inputan kemudian direpresentasikan menjadi sinyal suara dalam bentuk matriks dengan menggunakan perintah audioread di Matlab R2017a. hasil sampel suara pada gambar dibawah.



Gambar 9.125 Teori 5

6. Jelaskan apa itu one-hot encoding,dilengkapi dengan ilustrasi kode dan atau gambar.
- proses dimana variabel kategorikal dikonversi menjadi bentuk yang dapat disediakan untuk algoritma ML untuk melakukan pekerjaan yang lebih baik dalam prediksi.



Gambar 9.126 Teori 6

7. Jelaskan apa fungsi dari np.unique dan to_categorical dalam kode program,dilengkapi dengan ilustrasi atau gambar.
- np.unique digunakan untuk membuat array, sedangkan to_categorical digunakan untuk mengubah vektor kelas yang berupa integer (number) menjadi matrix kelas biner.

```
In [85]: a = np.array([4,3,4,4,3,2,2,1,1,6,6,6,3,3,2,2,0,0])
...: np.unique(a)
Out[85]: array([0, 1, 2, 3, 4, 6])
```

Gambar 9.127 Teori 7

```
In [90]: a = np.array([4,3,4,4,3,2,2,1,1,6,6,6,3,3,2,2,0,0])
...: to_categorical(np.unique(a))
Out[90]:
array([[1., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0.]]], dtype=float32)
```

Gambar 9.128 Teori 7

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

Sequential berfungsi sebagai tumpukan linear lapisan.

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

model = Sequential([
    Dense(32, input_shape=(784,)),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
])
```

Gambar 9.129 Teori 8

9.8.2 Praktek

1. Jelaskan isi dari data GTZAN Genre Collection dan data dari freesound. GTZAN Genre Collection berisikan klasifikasi genre musik. terdapat 10 genre musik didalamnya yaitu blues, clasisscal, country, disco, hiphop, jazz, metal, pop, reggae, dan rock. Dalam setiap genre tersebut berisikan 100 tracks musik.

```
1 import librosa #import librosa digunakan untuk fungsi mfcc
2 import librosa.feature #import librosa feattuse
3 import librosa.display #import librosa display
4 import glob #import glob
5 import numpy as np #import numpy untuk pengolahan data
       menjadi vektor sebagai np
6 import matplotlib.pyplot as plt #import matplotlib untuk
       melakukan plotting sebagai plt
7 from keras.models import Sequential #import sequential dari
       library keras
8 from keras.layers import Dense, Activation #import dense dan
       activation dari library keras
9 from keras.utils import np_utils #import
       to_categorical dari library keras
10
```

```

11 # In [1]: membuat fungsi mfcc untuk melakukan pengujian
12 def display_mfcc(song): #membuat fungsi display_mfcc
13     y, _ = librosa.load(song) # variabel Y yang berisi method
14         librosa load
15     mfcc = librosa.feature.mfcc(y) # method librosa featurea
16         mfcc
17
18     plt.figure(figsize=(10, 4)) #membuat ot gure dengan
19         ukuran 10 banding 4
20     librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
21         ) #data librosa display dengan variabel x nya yaitu waktu
22         dan y yaitu mel atau Hz
23     plt.colorbar() #plot bar warna
24     plt.title(song) #judul plot
25     plt.tight_layout()
26     plt.show() #plot di tampilkan

```

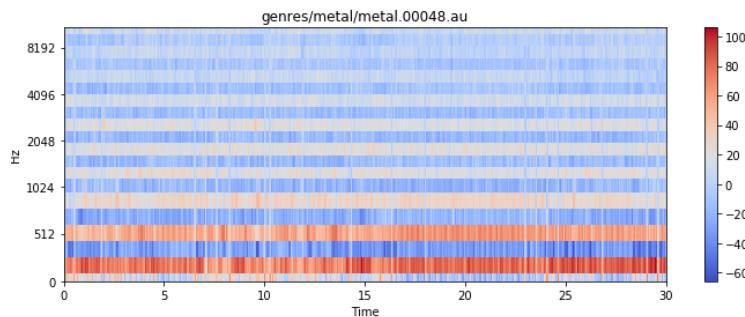
2. Jelaskan perbaris kode program dengan kata-kata dan dilengkapi ilustrasi gambar fungsi dari display_mfcc()

Ini spectogram dari lagu genre metal (source dari GTZAN Genre Collection).

```

1 # In [2]: cek fungsi
2 display_mfcc('genres/metal/metal.00048.au') #mendisplay
        tampilan glombang suara

```



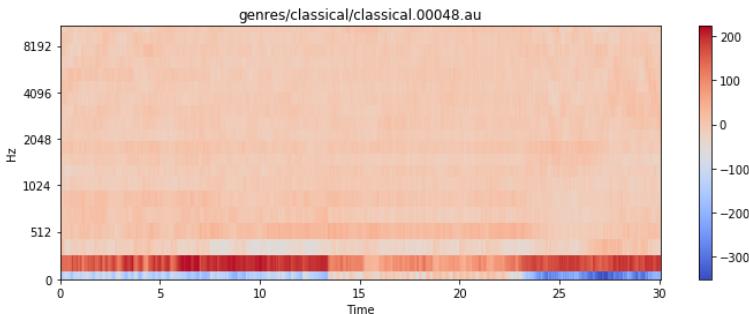
Gambar 9.130 spectrogram dari lagu genre metal

Ini spectogram dari lagu genre classical (source dari GTZAN Genre Collection).

```

1 # In [2]: cek fungsi
2 display_mfcc('genres/classical/classical.00048.au') #
        mendifplay tampilan glombang suara

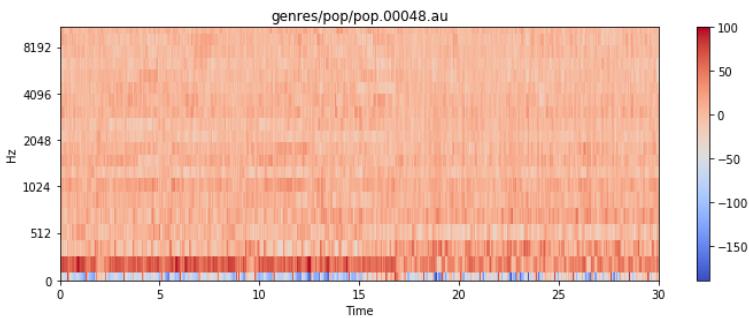
```



Gambar 9.131 spectogram dari lagu genre classical

Ini spectogram dari lagu genre pop (source dari GTZAN Genre Collection).

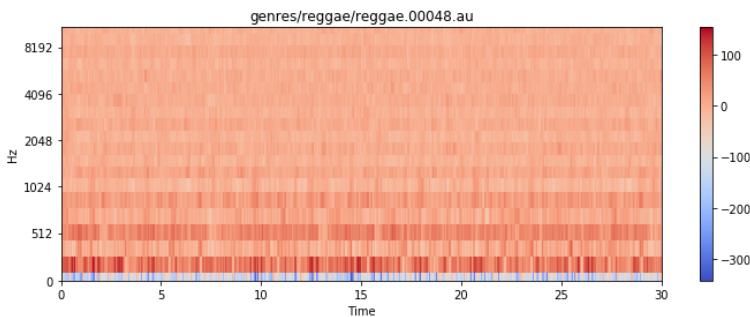
```
1 # In [2]: cek fungsi
2 display_mfcc('genres/pop/pop.00048.au') #mendisplay tampilan
glombang suara
```



Gambar 9.132 spectogram dari lagu genre pop

Ini spectogram dari lagu genre reggae (source dari GTZAN Genre Collection).

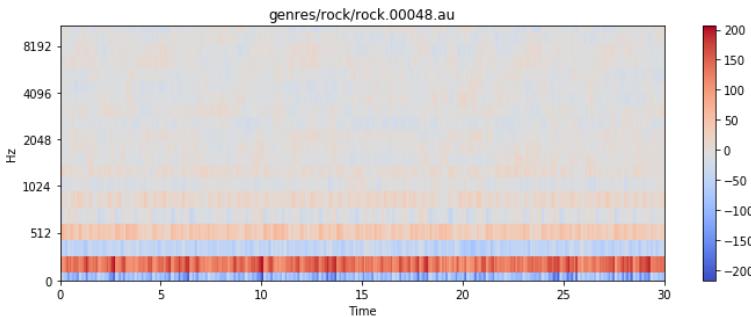
```
1 # In [2]: cek fungsi
2 display_mfcc('genres/reggae/reggae.00048.au') #mendisplay
tampilan glombang suara
```



Gambar 9.133 spectogram dari lagu genre reggae

Ini spectrogram dari lagu genre rock (source dari GTZAN Genre Collection).

```
1 # In [2]: cek fungsi
2 display_mfcc ('genres/rock/rock.00048.au') #mendisplay
    tampilan glombang suara
```



Gambar 9.134 spectogram dari lagu genre rock

3. Jelaskan perbaris kode program dengan kata-kata dan dilengkapi ilustrasi gambar fungsi dari extract features song()
Mengapa mengambil 25000 row pertama? dikarenakan Audio yang terdapat pada dataset ini tidak menentu durasinya. Dan kita harus mengambil data yang memiliki durasi yang sama untuk mempermudah dalam melakukan training.

```
1 # In [3]:
2 def extract_features_song(f): #nama extract features song
    yang nantinya akan di gunakan pada fungsi yang lainnya
3     y, _ = librosa.load(f) #membuat variabel y dengan method
        librosa load
4
```

```

5     mfcc = librosa.feature.mfcc(y) #membuat variabel baru
6     mfcc dengan isi librosa features mfcc dengan isi variabel
7     y
8
9     mfcc /= np.amax(np.absolute(mfcc)) #variabel mfcc dengan
10    isian np.max
11
12    return np.ndarray.flatten(mfcc)[:25000] #membuat array
13    dari data tersebut merupakan data 25000 data pertama

```

```

In [98]: def extract_features_song(f): #nama extract features song yang nantinya akan
di gunakan pada fungsi yang lainnya
...     y, _ = librosa.load(f) #membuat variabel y dengan method Librosa Load
...     ...
...     mfcc = librosa.feature.mfcc(y) #membuat variabel baru mfcc dengan isi
Librosa features mfcc dengan isi variabel y
...     ...
...     mfcc /= np.amax(np.absolute(mfcc)) #variabel mfcc dengan isian np.max
...     ...
...     return np.ndarray.flatten(mfcc)[:25000] #membuat array dari data
tersebut merupakan data 25000 data pertama

```

Gambar 9.135 Praktek 3

4. Jelaskan perbaris kode program dengan kata-kata dan dilengkapi ilustrasi gambar fungsi dari generate features and labels().

```

1 # In [4]:
2 def generate_features_and_labels(): #definisi nama fungsi
3     yaitu generate features and labels
4     all_features = [] #membuat variabel baru dengan array
5     kosong yaitu all\features
6     all_labels = [] ##membuat variabel baru dengan array
7     kosong yaitu all\label
8
9     genres = [ 'blues' , 'classical' , 'country' , 'disco' , '
10    hiphop' , 'jazz' , 'metal' , 'pop' , 'reggae' , 'rock' ] # mendefinisikan isian label untuk genres dengan cara
11    membuat variabel genres kemudian di isi dengan 10 genre
12    for genre in genres:
13        sound_files = glob.glob('genres/' +genre+ '/*.au')
14        print('Processing %d songs in %s genre...' % (len(
15        sound_files), genre))
16        for f in sound_files:
17            features = extract_features_song(f)
18            all_features.append(features)
19            all_labels.append(genre)
20
21
22    # convert labels to one-hot encoding cth blues :
23    1000000000 classic 0100000000
24    label_uniq_ids , label_row_ids = np.unique(all_labels ,
25    return_inverse=True)#ke integer
26    label_row_ids = label_row_ids.astype(np.int32 , copy=False
27    )
28    onehot_labels = to_categorical(label_row_ids , len(
29    label_uniq_ids))#ke one hot
30
31    return np.stack(all_features) , onehot_labels

```

```
In [99]: def generate_features_and_labels(): #definisiakan nama fungsi yaitu generate features and
labels
...:
    all_features = [] ##membuat variabel baru dengan array kosong yaitu all_features
    ...:
    all_labels = [] ##membuat variabel baru dengan array kosong yaitu all_labels
    ...:
    genres = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz', 'metal', 'pop',
'reggae', 'rock'] ##mendefinisikan isian label untuk genres dengan cara membuat variabel genres
kemudian di isi dengan 10 genre
    ...:
    for genre in genres:
        sound_files = glob.glob('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 cth blues : 1000000000 classic 0100000000
    ...:
    label_uniq_ids, label_row_ids = np.unique(all_labels, return_inverse=True)## integer
    label_row_ids = label_row_ids.astype(np.int32, copy=False)
    onehot_labels = to_categorical(label_row_ids, len(label_uniq_ids))##be one hot
    ...:
    return np.stack(all_features), onehot_labels
```

Gambar 9.136 Praktek 4

- Jelaskan dengan kata dan praktek kenapa penggunaan fungsi generate features and labels() sangat lama ketika meload dataset genre.
Karena mesin membaca file satu persatu yang ada pada folder dan dalam folder tersebut dan setiap folder terdapat 100 file sehingga menjadi lama dan mengolah data yang tadinya suara menjadi bentuk vektor.

```
1 # In[5]: passing parameter dari fitur ekstraksi menggunakan
     mfcc
2 features , labels = generate_features_and_labels()
```

```
In [100]: 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 pop genre...
Processing 100 songs in reggae genre...
Processing 100 songs in rock genre...

In [101]: print(np.shape(features))
...: print(np.shape(labels))
(1000, 25000)
(1000, 10)
```

Gambar 9.137 Praktek 5

- Jelaskan kenapa harus dilakukan pemisahan data training dan data set sebesar 80 persen? Praktekkan dengan kode dan Tunjukkan keluarannya
Melakukan training split 80% supaya mesin dapat terus belajar tentang data baru, jadi ketika prediksi dibuat tentang data yang terlatih itu bisa mendapatkan persentase yang cukup bagus.

```
1 training_split = 0.8
2
3 # In []
4 # last column has genre, turn it into unique ids
```

```

5 alldata = np.column_stack((features, labels))
6
7 # In []
8 np.random.shuffle(alldata)
9 splitidx = int(len(alldata) * training_split)
10 train, test = alldata[:splitidx,:], alldata[splitidx,:,:]
11
12 # In []
13 print(np.shape(train))
14 print(np.shape(test))
15
16 # In []
17 train_input = train[:, :-10]
18 train_labels = train[:, -10:]
19
20 test_input = test[:, :-10]
21 test_labels = test[:, -10:]
22
23 # In []
24 print(np.shape(train_input))
25 print(np.shape(train_labels))

```

In [102]: `training_split = 0.8`

In [103]: `alldata = np.column_stack((features, labels))`

In [104]: `np.random.shuffle(alldata)`
`....: splitidx = int(len(alldata) * training_split)`
`....: train, test = alldata[:splitidx,:], alldata[splitidx,:,:]`

In [105]: `print(np.shape(train))`
`....: print(np.shape(test))`
`(800, 25010)`
`(200, 25010)`

In [106]: `train_input = train[:, :-10]`
`....: train_labels = train[:, -10:]`
`....:`
`....: test_input = test[:, :-10]`
`....: test_labels = test[:, -10:]`

In [107]: `print(np.shape(train_input))`
`....: print(np.shape(train_labels))`
`(800, 25000)`
`(800, 10)`

Gambar 9.138 Praktek 6

- Praktekan dan jelaskan masing-masing parameter dari fungsi Sequential().

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.

```

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

```

```

In [108]: model = Sequential([
...:     Dense(100, input_dim=np.shape(train_input)[1]),
...:     Activation('relu'),
...:     Dense(10),
...:     Activation('softmax'),
...: ])

```

Gambar 9.139 Praktek 7

8. Praktekkan dan jelaskan masing-masing parameter dari fungsi compile().

Model Compile dijelaskan 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 aktivasi 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.

```

1 model.compile(optimizer='adam',
2                 loss='categorical_crossentropy',
3                 metrics=['accuracy'])
4 print(model.summary())

```

```
In [109]: model.compile(optimizer='adam',
    ...
    loss='categorical_crossentropy',
    ...
    metrics=['accuracy'])
    ...: print(model.summary())
Model: "sequential_3"

```

Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 100)	2500100
activation_5 (Activation)	(None, 100)	0
dense_6 (Dense)	(None, 10)	1010
activation_6 (Activation)	(None, 10)	0

```
Total params: 2,501,110
Trainable params: 2,501,110
Non-trainable params: 0
```

```
None
```

Gambar 9.140 Praktek 8

9. Praktekkan dan jelaskan masing-masing parameter dari fungsi fit(). 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.

```
1 model.fit(train_input, train_labels, epochs=10, batch_size
2 =32,
3 validation_split=0.2)
```

```
In [110]: model.fit(train_input, train_labels, epochs=10, batch_size=32,
...: validation_split=0.2)
Train on 640 samples, validate on 160 samples
Epoch 1/10
640/640 [=====] - 1s 2.163s/step - loss: 2.1635 - accuracy: 0.3109 - val_loss: 1.8495 - val_accuracy: 0.3375
Epoch 2/10
640/640 [=====] - 1s 891us/step - loss: 1.3906 - accuracy: 0.5125 - val_loss: 1.6384 - val_accuracy: 0.4313
Epoch 3/10
640/640 [=====] - 1s 890us/step - loss: 1.0521 - accuracy: 0.6486 - val_loss: 1.5986 - val_accuracy: 0.4375
Epoch 4/10
640/640 [=====] - 1s 893us/step - loss: 0.8351 - accuracy: 0.7328 - val_loss: 1.5962 - val_accuracy: 0.4563
Epoch 5/10
640/640 [=====] - 1s 890us/step - loss: 0.6233 - accuracy: 0.8313 - val_loss: 1.4628 - val_accuracy: 0.5125
Epoch 6/10
640/640 [=====] - 1s 888us/step - loss: 0.5007 - accuracy: 0.8844 - val_loss: 1.5120 - val_accuracy: 0.4812
Epoch 7/10
640/640 [=====] - 1s 898us/step - loss: 0.3688 - accuracy: 0.9375 - val_loss: 1.5988 - val_accuracy: 0.5000
Epoch 8/10
640/640 [=====] - 1s 899us/step - loss: 0.2823 - accuracy: 0.9594 - val_loss: 1.5113 - val_accuracy: 0.4437
Epoch 9/10
640/640 [=====] - 1s 919us/step - loss: 0.2226 - accuracy: 0.9828 - val_loss: 1.5454 - val_accuracy: 0.5312
Epoch 10/10
640/640 [=====] - 1s 923us/step - loss: 0.1822 - accuracy: 0.9891 - val_loss: 1.5895 - val_accuracy: 0.4625
0x110]: <keras.callbacks.Callback>.History at 0x1f7584d2f00
```

Gambar 9.141 Praktek 9

10. Praktekkan dan jelaskan masing-masing parameter dari fungsi evaluate(). Fungsi evaluate atau evaluasi ini ialah untuk menguji data pengujian setiap file. Di sini ada data yang hilang sebesar 1.6111 data, sedangkan untuk akurasinya sekitar 51%.

```

1 loss , acc = model.evaluate(test_input , test_labels ,
2   batch_size=32)
3
4 print("Done!")
5 print("Loss: %.4f , accuracy: %.4f" % (loss , acc))

```

```

In [122]: loss, acc = model.evaluate(test_input, test_labels, batch_size=32)
...
...
...: print("Done!")
...: print("Loss: %.4f, accuracy: %.4f" % (loss, acc))
200/200 [=====] - 0s 174us/step
Done!
Loss: 1.6111, accuracy: 0.5150

```

Gambar 9.142 Praktek 10

11. Praktekkan dan jelaskan masing-masing parameter dari fungsi predict(). 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 classical. Maka lagu tersebut termasuk ke dalam genre classical dengan perbandingan presentase hasil prediksi.

```
1 model.predict(test_input[:1])
```

```

In [115]: model.predict(test_input[:1])
Out[115]:
array([[1.7469719e-03, 9.4805568e-01, 8.8974694e-04, 4.0234820e-04,
       6.9074599e-07, 4.7569752e-02, 6.0497349e-09, 1.8554716e-04,
       1.0829815e-03, 6.6229710e-05]], dtype=float32)

```

Gambar 9.143 Praktek 11

9.8.3 Bukti Tidak Plagiat



Gambar 9.144 plagiarism

9.8.4 Link Video Youtube

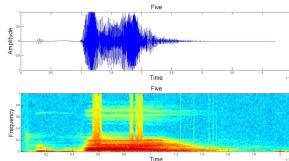
<https://youtu.be/MnrV7s-Tdy0>

9.9 1174080 - Handi Hermawan

9.9.1 Soal Teori

- Jelaskan kenapa file suara harus dilakukan MFCC. dilengkapi dengan ilustrasi atau gambar.

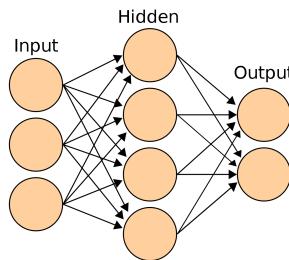
Karena MFCC adalah koefisien yang mewakili audio. Machine learning tidak memahami rekaman suara melainkan vektor. Maka rekaman tersebut akan diubah kedalam bentuk vektor kemudian vektor akan menyesuaikan dengan kata-kata yang sudah disediakan. Ekstraksi fitur dalam proses ini ditandai dengan konversi 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.



Gambar 9.145 Teori 1

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

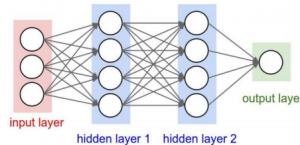
Konsep sederhana dari neural network atau jaringan saraf sederhana dengan proses pembelajaran pada anak-anak dengan memetakan pola-pola baru yang diperoleh dari input untuk membuat pola-pola baru pada output. Contoh sederhana ini menganalogikan kinerja otak manusia. Jaringan saraf itu sendiri terdiri 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 paralel. Karakteristik neural network atau jaringan saraf dilihat dari pola hubungan antar neuron, metode penentuan bobot setiap koneksi, dan fungsi aktivasi mereka..



Gambar 9.146 Teori 2

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

Pembobotan di dalam neural network juga akan menentukan penanda koneksi. 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.



Gambar 9.147 Teori 3

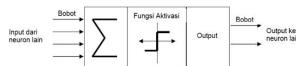
- Jelaskan konsep fungsi aktifasi dalam neural network. dilengkapi dengan ilustrasi atau gambar.

Fungsi aktifasi dalam neural network ialah merupakan suatu operasi matematik yang dikenakan pada sinyal output. Fungsi ini digunakan untuk mengaktifkan atau menonaktifkan neuron. Fungsi aktifasi 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.

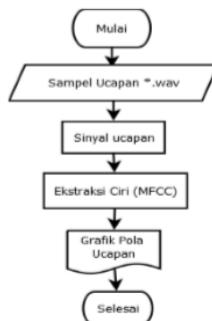
. Namun, untuk ilustrasi lihat gambar berikut:



Gambar 9.148 Teori 4

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

Penelitian ini dilakukan untuk mengatahui hasil dari ekstraksi ciri MFCC dari ucapan emosi manusia. Penelitian ini menggunakan software Matlab R2013a dengan menggunakan sampel dari ucapan aktor di database Berlin Database of Emotional. Bahasa yang digunakan adalah bahasa Berlin. Metode MFCC dalam penelitian ini digunakan untuk mengekstraksi ciri sampel ucapan dari aktor. Berikut adalah langkah-langkah tahapan penelitian:



Gambar 9.149 Teori 5

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

Digunakan dalam pembelajaran mesin sebagai metode untuk mengukur data kategorikal. Singkatnya, metode ini menghasilkan vektor dengan panjang yang sama dengan jumlah kategori dalam kumpulan data. Jika titik data milik kategori engan maka komponen dari vektor ini diberi nilai 0 kecuali untuk komponen engan, yang diberi nilai 1. Dengan cara

ini seseorang dapat melacak kategori dengan cara yang bermakna secara numerik. Mungkin timbul pertanyaan: Apa yang terjadi jika ada banyak 1? Klasifikasi mana yang benar? Ini dengan cepat diperbaiki oleh fakta bahwa sesuatu yang dikodekan satu-panas memiliki tepat satu posisi dalam array yang diberi label sebagai 1. Misalnya, [0,0,0,1,0] akan menjadi satu-panas yang valid pengkodean yang akan memberitahu Anda klasifikasi di posisi 4 (atau 3 dalam pengindeksan array) adalah klasifikasi objek. Sebaliknya, [0,1,0,1,0], dan [1,1,1,1,1] adalah contoh pengkodean, seperti :

```
[0,1,0,0] // female
[0,1,0,0]
[0,1,0,0]
[0,1,0,0]
[1,0,0,0] // male
[1,0,0,0]
[1,0,0,0]
[0,0,1,0] // gender-neutral
[0,0,0,1] // other
[0,0,0,1]
```

Gambar 9.150 Teori 6

- Jelaskan apa fungsi dari np.unique dan to_categorical dalam kode program,dilengkapi dengan ilustrasi atau gambar.
numpy.unique untuk mengembalikan elemen unik array yang diurutkan. Ada tiga output opsional selain elemen unik: indeks array input yang memberikan nilai unik dan indeks array unik yang merekonstruksi array input berapa kali setiap nilai unik muncul dalam array input

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

Gambar 9.151 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 untuk memproses categorical data kolom Country ??

```
dataset$Country = factor(dataset$Country,
                         levels = c("France", "Spain", "Germany"),
                         labels = c(1, 2, 3))
```

Gambar 9.152 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 [29]: model = Sequential([
    Dense(128, input_dim=train_input[1]),
    Activation('relu'),
    Dense(64),
    Activation('relu'),
    Dense(32),
    Activation('softmax'),
])
```

Gambar 9.153 Teori 8

9.9.2 Praktek Program

1. 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 libraries
2 import librosa
3 import librosa.feature
4 import librosa.display
5 import glob
6 import os
7 import matplotlib.pyplot as plt
8
9 from sklearn.model_selection import train_test_split
10 from sklearn.neighbors import KNeighborsClassifier
11 from sklearn.metrics import accuracy_score
12 from keras.models import Sequential
13 from keras.layers import Dense
14
15 def display_mfcc(file):
16     y, sr = librosa.load(file)
17     mfcc = librosa.feature.mfcc(y=y)
18
19     plt.figure(figsize=(10, 4))
20     librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
21     plt.title(file)
22     plt.colorbar()
23     plt.show()

```

Gambar 9.154 Hasil Soal 1.

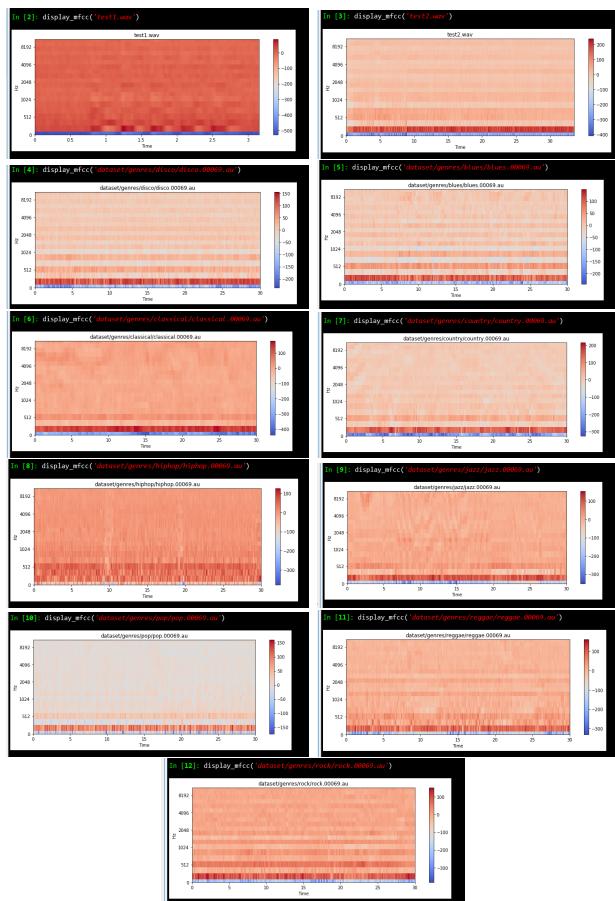
2. Soal 2

```

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 9.155 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, sr = librosa.load(f)
    ...
    # get Mel-frequency cepstral coefficients
    mfcc = librosa.feature.mfcc(y)
    mfcc -= np.mean(mfcc) # centering (divide by max)
    mfcc /= np.max(np.abs(mfcc))
    ...
    return np.ndarray.flatten(mfcc)[:1000]

```

Gambar 9.156 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) Load features, predict_labels();
    all_features = []
    all_labels = []
    genres = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz', 'metal',
              'pop', 'reggae', 'rock']

    for genre in genres:
        for song in os.listdir(os.path.join('genres', genre)):
            if song[-4:] == '.mp3':
                song_file = os.path.join('genres', genre, song)
                song_f = Song(song_file)
                song_f.load()
                features = extract_features(song_f)
                all_features.append(features)
                all_labels.append(label_map[genre])

    # Invert label mapping
    label_map_inv, label_map_rev = np.unique(all_labels, return_inverse=True)
    print(label_map_rev)

    # Create labels
    label_raw_ids = np.array([label_map_rev[int(i)] for i in all_labels])
    onehot_labels = np.zeros((len(all_labels), len(label_map_inv)))
    for i in range(len(all_labels)):
        onehot_labels[i][label_raw_ids[i]] = 1

    # (5) Train model
    model = Sequential()
    model.add(Dense(128, input_dim=100))
    model.add(Activation('relu'))
    model.add(Dropout(0.2))
    model.add(Dense(128))
    model.add(Activation('relu'))
    model.add(Dropout(0.2))
    model.add(Dense(10))
    model.add(Activation('softmax'))

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

    model.fit(all_features, onehot_labels, batch_size=32, epochs=10, validation_split=0.2)

```

Gambar 9.157 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 pop genre...
Processing 100 songs in reggae genre...
Processing 100 songs in rock genre...
In [16]: print(np.shape(features))
...: print(np.shape(labels))
(1000, 25000)
(1000, 10)
```

Gambar 9.158 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 :

In [17]: training_split = 0.8	In [18]: alldata = np.column_stack((features, labels))
In [19]: np.random.shuffle(alldata) ...: splitidx = int(len(alldata) * training_split) ...: train, test = alldata[:splitidx], alldata[splitidx:,:]	In [20]: print(np.shape(train)) ...: print(np.shape(test)) (800, 25010) (200, 25010)
In [21]: train_input = train[:, :-10] ...: train_labels = train[:, -10:]	In [22]: test_input = test[:, :-10] ...: test_labels = test[:, -10:]
In [23]: print(np.shape(train_input)) ...: print(np.shape(train_labels)) (800, 25000) (800, 10)	<pre> Name Type Size Value allata float64 (2000, 25010) [[4.266458 -0.7990807 -0.770504 ... 0. 0. train float64 (800, 25000) [[4.219967 -0.8101095 -0.7156167 ... -0.4101850 0.49377342 labels float32 (800, 10) [[0. 0. ... 0. 0. 0. 0. 0. 0. 0.] splitidx float32 (200,) [[1.0] train float32 (200, 25000) [[4.266458 -0.7990807 -0.770504 ... 0. 0. test float32 (200, 25000) [[4.057048 -0.5668772 -0.4598078 ... 0. 1. test_input float32 (200, 25000) [[4.057048 -0.5668772 -0.4598078 ... 0. 1. test_labels float32 (200, 10) [[0. 0. 0. ... 0. 1. 0. train float32 (800, 25000) [[4.219967 -0.8101095 -0.7156167 ... 0. 0. train_input float32 (800, 25000) [[4.219967 -0.8101095 -0.7156167 ... 0. 0. train_labels float32 (800, 10) [[0. 0. 0. ... 0. 0. 0. 0. 0. 0. 0.] training_split float32 (1) [[0.8] </pre>

Gambar 9.159 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=train_input[1]),
    Activation('relu'),
    ...
    Dense(10),
    Activation('softmax'),
    ...
])
```

Gambar 9.160 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 aktivasasi. 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'])
... print(model.summary())
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 9.161 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 persentase dari cross validation atau k-fold sebanyak 20% dari masing-masing data pengulangan, hasilnya adalah sebagai berikut :

Gambar 9.162 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 :

```
[1]: [27] loss, acc = model.evaluate(test_input, test_labels, batch_size=1)
200/200 [=====] - 0s 200ms/step
[28]: print("Done!")
Done!
[29]: print(f"loss: {loss:.4f}, accuracy: {acc:.4f}" % (loss, acc))
loss: 1.4637, accuracy: 0.4900
```

Gambar 9.163 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, 8.2329191e-02, 6.3466556e-02, 1.3786874e-02,
       2.5682386e-02, 6.7009211e-01, 6.7797744e-02, 2.02035776e-02,
       2.510169e-02, 3.03265517e-02]], dtype=float32)
```

Gambar 9.164 Hasil Soal 11

9.9.3 Penanganan Error

1. ScreenShoot Error

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

Gambar 9.165 ModuleNotFoundError



Gambar 9.166 An error occurred while starting the kernel

2. Cara Penanganan Error

- ModuleNotFoundError

Error terdapat pada library yang tidak terbaca, karena library tensorflow belum di install, solusinya dengan menginstall library tersebut, pip install tensorflow.

- An error occurred while starting the kernel

Seperti pada gambar tersebut untuk mengistall tensorflow

9.9.4 Bukti Tidak Plagiat



Gambar 9.167 Bukti Tidak Melakukan Plagiat Chapter 6

9.9.5 Link Video Youtube

<https://youtu.be/MzxIZ-C-gi0>

..... 8d03468695a613ab2ea6674a0634ca35b347d724

