Library & Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
pip install tensorflow.io
```

```
Collecting tensorflow.io
  Downloading tensorflow_io-0.23.1-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64
     |████████████████████████████████| 23.1 MB 1.5 MB/s
Requirement already satisfied: tensorflow-io-gcs-filesystem==0.23.1 in /usr/local/lib/py
Installing collected packages: tensorflow.io
Successfully installed tensorflow.io-0.23.1
```

```
from IPython.display import Audio

import tensorflow as tf
import tensorflow_io as tfio
import matplotlib.pyplot as plt
import numpy as np
import glob


#pip install python_speech_features


import os
import librosa
import pickle
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from tqdm import tqdm
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, Dropout
from keras.models import Sequential
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
#from python_speech_features import mfcc
```

Load Data Gumaman

```
daftar_gdangdut = glob.glob('/content/drive/MyDrive/Dataset01/Dangdut/Gumam/*')
daftar_gkpop = glob.glob('/content/drive/MyDrive/Dataset01/KPop/Gumam/*')


#baca file dan masukin ke list
gumaman_dangdut =[]
for i in range(len(daftar_gdangdut)):
  gumam = tf.io.read_file(daftar_gdangdut[i])
  suara = tfio.audio.decode_mp3(gumam)
  gumaman_dangdut.append(suara)


gumaman_kpop =[]
for i in range(len(daftar_gkpop)):
  gumam = tf.io.read_file(daftar_gkpop[i])
  suara = tfio.audio.decode_mp3(gumam)
  gumaman_kpop.append(suara)


gumaman_kpop
```

```
     [<tf.Tensor: shape=(9231360, 1), dtype=float32, numpy=
      array([[ 0.0000000e+00],
             [ 0.0000000e+00],
             [ 0.0000000e+00],
             ...,
             [-5.4995853e-05],
             [ 6.2783435e-04],
             [ 6.1394507e-04]], dtype=float32)>,
      <tf.Tensor: shape=(8942296, 1), dtype=float32, numpy=
      array([[-1.0203528e-05],
             [-1.5740072e-05],
             [-1.7388518e-05],
             ...,
             [ 3.9938779e-05],
             [ 6.0729850e-05],
             [ 5.0359940e-05]], dtype=float32)>,
      <tf.Tensor: shape=(8942296, 1), dtype=float32, numpy=
      array([[-1.0203528e-05],
             [-1.5740072e-05],
             [-1.7388518e-05],
             ...,
             [ 3.9938779e-05],
             [ 6.0729850e-05],
             [ 5.0359940e-05]], dtype=float32)>,
      <tf.Tensor: shape=(9245656, 1), dtype=float32, numpy=
      array([[-4.4457088e-06],
             [-8.1725630e-06],
             [-7.7637205e-06],
```

```
            ...,
            [ 1.8270534e-03],
            [ 2.3665898e-03],
            [ 1.8961872e-03]], dtype=float32)>,
      <tf.Tensor: shape=(11251096, 1), dtype=float32, numpy=
      array([[-1.08603235e-05],
            [-1.64603007e-05],
            [-1.81411378e-05],
            ...,
            [ 1.25858587e-05],
            [ 1.96480178e-05],
            [ 1.58288731e-05]], dtype=float32)>]


#train
x=[]
y=[]

#mencacah/motong lagu
for j in range(len(daftar_gdangdut)):
  for i in range(0, len(gumaman_dangdut[j])-44100, 11025):
    x.append(gumaman_dangdut[j][i:i+44100])
    y.append(0)

for j in range(len(daftar_gkpop)):
  for i in range(0, len(gumaman_kpop[j])-44100, 11025):
    x.append(gumaman_kpop[j][i:i+44100])
    y.append(1)

     '\nfor j in range(len(daftar_gjpop)):\n  for i in range(0, len(gumaman_jpop[j])-4
     4100, 11025):\n    x.append(gumaman_jpop[j][i:i+44100])\n    y.append(2)\n\nfor j
     in range(len(daftar_gpop)):\n  for i in range(0, len(gumaman_pop[j])-44100, 1102

x = np.asarray(x)
y = np.asarray(y)


y
#y.shape

     array([0, 0, 0, ..., 1, 1, 1])


label_encoder = LabelEncoder()
label_encoded = label_encoder.fit_transform(y)


label_encoded

     array([0, 0, 0, ..., 1, 1, 1])


label_encoded=label_encoded[:,np.newaxis]
```

```
label_encoded
```

```
     array([[0],
            [0],
            [0],
            ...,
            [1],
            [1],
            [1]])
```

```
one_hot_encoder = OneHotEncoder(sparse=False)
one_hot_encoded = one_hot_encoder.fit_transform(label_encoded)
```

```
one_hot_encoded
```

```
     array([[1., 0.],
            [1., 0.],
            [1., 0.],
            ...,
            [0., 1.],
            [0., 1.],
            [0., 1.]])
```

```
y = one_hot_encoded
```

```
maks = np.max(x)
mins = np.min(x)

#normalisasi
x = (x-mins)/(maks-mins)
#y = (y-mins)/(maks-mins)
```

```
x
```

```
     array([[[0.5660316 ],
             [0.5660273 ],
             [0.56602603],
             ...,
             [0.5660046 ],
             [0.56586987],
             [0.56586444]],

            [[0.5619971 ],
             [0.56287265],
             [0.5639755 ],
             ...,
             [0.5665793 ],
             [0.5665167 ],
             [0.5664939 ]],

            [[0.5697608 ],
```

```
             [0.56930095],
             [0.56876004],
             ...,
             [0.5660639 ],
             [0.566091  ],
             [0.5661264 ]],

            ...,

            [[0.5661747 ],
             [0.5656962 ],
             [0.56544554],
             ...,
             [0.5659916 ],
             [0.56598556],
             [0.56599766]],

            [[0.56604743],
             [0.5660389 ],
             [0.56604606],
             ...,
             [0.56604743],
             [0.5660431 ],
             [0.56603867]],

            [[0.56609505],
             [0.5658925 ],
             [0.5657439 ],
             ...,
             [0.566036  ],
             [0.56605107],
             [0.5660517 ]]], dtype=float32)
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

```
print(x.shape)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(8955, 44100, 1)
(7164, 44100, 1)
(7164, 2)
(1791, 44100, 1)
(1791, 2)
```

Model

```
# CNN 1D
```

```
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Activation

#size_ = 8 #inputnya ada 8, dr pclass-embarked

model = Sequential()

model.add(Conv1D(128, kernel_size=3, input_shape = (x.shape[1],1), activation = 'relu'))
model.add(Flatten())

model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(64, activation='relu'))

#model.add(Dense(units = 8, activation = 'relu'))
model.add(Dense(units = 2, activation = 'softmax'))

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


model.fit(x_train,y_train, batch_size=20, epochs=50)
```

```
Epoch 1/50
359/359 [==============================] - 56s 128ms/step - loss: 1.0325 - accuracy: 0.5
Epoch 2/50
  6/359 [..............................] - ETA: 44s - loss: 0.6990 - accuracy: 0.5167
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-24-d5dce5824fb3> in <module>()
----> 1 model.fit(x_train,y_train, batch_size=20, epochs=50)

                                 8 frames
/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/execute.py in
quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     57     ctx.ensure_initialized()
     58     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
---> 59                                         inputs, attrs, num_outputs)
     60   except core._NotOkStatusException as e:
     61     if name is not None:

KeyboardInterrupt:
```

```
#model.save_weights('f1_classifiergumam.h5')
#read file
model.load_weights('/content/drive/MyDrive/Dataset01/f1_classifiergumam.h5')


lur = model.predict(x_test)
```

```
lur
```

```
array([[3.7499070e-16, 1.0000000e+00],
       [7.4571616e-29, 1.0000000e+00],
       [8.1414337e-29, 1.0000000e+00],
       ...,
       [4.9899347e-28, 1.0000000e+00],
       [2.0520323e-29, 1.0000000e+00],
       [8.5616235e-29, 1.0000000e+00]], dtype=float32)
```

```
lur[50]
```

```
array([5.0949437e-18, 1.0000000e+00], dtype=float32)
```

```
predictions = np.argmax(lur, axis=1)
y_test = one_hot_encoder.inverse_transform(y_test)
```

```
predictions
```

```
array([1, 1, 1, ..., 1, 1, 1])
```

```
predict = pd.DataFrame(predictions)
```

```
predict
```

	0
0	1
1	1
2	1
3	1
4	1
...	...
1786	1
1787	1
1788	1
1789	1
1790	1

1791 rows × 1 columns

```
predict[(predict[0]==1)].shape
```

```
(1791, 1)
```

```
print(predictions)
```

```
[1 1 1 ... 1 1 1]
```

```
y_test
```

```
array([[1],
       [1],
       [0],
       ...,
       [0],
       [0],
       [0]])
```

```
accuracy_score(predictions, y_test)
```

```
0.4796203238414294
```

Predict with new file

```
ftest = '/content/drive/MyDrive/Dataset01/KPop/Gumam/Akdong Musician - Give Love.mp3'
```

```
ftest
```

```
'/content/drive/MyDrive/Dataset01/KPop/Gumam/Akdong Musician - Give Love.mp3'
```

```
f_Test = tf.io.read_file(ftest)
s_Test = tfio.audio.decode_mp3(f_Test)
```

```
s_Test
```

```
<tf.Tensor: shape=(8942296, 1), dtype=float32, numpy=
array([[-1.0203528e-05],
       [-1.5740072e-05],
       [-1.7388518e-05],
       ...,
       [ 3.9938779e-05],
       [ 6.0729850e-05],
       [ 5.0359940e-05]], dtype=float32)>
```

```
input = []
```

```
for i in range(0, len(s_Test)-44100, 11025):
    input.append(s_Test[i:i+44100])

input = np.asarray(input)
maks = np.max(input)
min = np.min(input)

input = (input-min)/(maks-min)


input.shape
```

```
(808, 44100, 1)
```

```
hasil = model.predict(input)
hasil = np.argmax(hasil,axis=1)


hasil
```

```
array([1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
```

```
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
            1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])


hasil = pd.DataFrame(hasil)
jmlh_0 = hasil[(hasil[0]==0)].shape[0]
jmlh_1 = hasil[(hasil[0]==1)].shape[0]


if jmlh_0 > jmlh_1 :
  print('dangdut')
elif jmlh_0 < jmlh_1 :
  print('kpop')
else :
  print('not known')
```

```
     kpop
```