```
Install Jupyter Dash : !pip install jupyter-dash
Download the files : https://drive.google.com/drive/folders/
1i1HjFkLC6dhE0ObPNq-y1Bb3f3CMoUD2?usp=sharing
Upload the WebApp in your Drive.
WebApp folder conatins:
  1)my_model0.h5 - Valence
  2)my_model1.h5 - Arousal
  3)my_model2.h5 - Dominance
  4)my_model3.h5 - Liking

Mount the google drive to your Colab session, so that you can simply write to
google drive as you would to a local file system.
```

Install Jupyter Dash : !pip install jupyter-dash Download the files : https://drive.google.com/drive/folders/1i1HjFkLC6dhE0ObPNq-y1Bb3f3CMoUD2?usp=sharing Upload the WebApp in your Drive. WebApp folder conatins: 1)my_model0.h5 - Valence 2)my_model1.h5 - Arousal 3)my_model2.h5 - Dominance 4)my_model3.h5 - Liking

Mount the google drive to your Colab session, so that you can simply write to google drive as you would to a local file system.

Double-click (or enter) to edit

```
!pip install jupyter-dash
```

```
import plotly.express as px
from jupyter_dash import JupyterDash
from dash.dependencies import Input, Output, State
from dash import html
from dash import dcc
from dash import dash_table
import plotly.express as px

import pandas as pd
""

     ''
```

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

import numpy as np
import pickle as pickle
import pandas as pd
import math

from sklearn import svm
from sklearn.preprocessing import normalize
import os
import time
import pandas as pd
import keras.backend as K
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.models import Sequential
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
#from keras.utils import to_categorical
from keras.layers import Flatten
from keras.layers import Dense
import numpy as np
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from keras import backend as K
from keras.models import Model
import timeit
from keras.models import Sequential
from keras.layers.core import Flatten, Dense, Dropout
from keras.layers.convolutional import Convolution1D, MaxPooling1D, ZeroPadding1D
from keras.models import load_model
# from keras.optimizers import SGD
#import cv2, numpy as np
import warnings
warnings.filterwarnings('ignore')
os.getcwd()
os.chdir('/content/drive/MyDrive')
channel = [1,2,3,4,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,23,24,25,26,27,29,30,31]
band = [4,8,12,16,25,45]
window_size = 256
step_size = 16
sample_rate = 128
subjectList = ['01','02','03','04','05','06','07','08','09','10','11','12','13','14','15',

with open('/content/drive/MyDrive/BTP/data_preprocessed_python/FFTdata_testing.npy', 'rb')
    M  = np.load(fileTrain)

with open('/content/drive/MyDrive/BTP/data_preprocessed_python/FFTlabel_testing.npy', 'rb'
    N  = np.load(fileTrainL)

M = normalize(M)

L0 = np.ravel(N[:, [0]])
L1 = np.ravel(N[:, [1]])
L2 = np.ravel(N[:, [2]])
```

```
L3 = np.ravel(N[:, [3]])

Arousal_Test = np.ravel(N[:, [0]])
Valence_Test = np.ravel(N[:, [1]])
Domain_Test = np.ravel(N[:, [2]])
Like_Test = np.ravel(N[:, [3]])

x_test = np.array(M[:])

from tensorflow.keras.utils import to_categorical
from tensorflow.keras.utils import to_categorical
y0_test = to_categorical(L0)
y1_test = to_categorical(L1)
y2_test = to_categorical(L2)
y3_test = to_categorical(L3)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_test = scaler.fit_transform(x_test)

x_test = x_test.reshape(x_test.shape[0],x_test.shape[1], 1)
from keras import backend as K

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))



dependencies = {
     'f1_m': f1_m,
     'precision_m': precision_m,
     'recall_m': recall_m
}

model0 = load_model('my_model0.h5', custom_objects=dependencies)
model1 = load_model('my_model1.h5', custom_objects=dependencies)
model2 = load_model('my_model2.h5', custom_objects=dependencies)
model3 = load_model('my_model3.h5', custom_objects=dependencies)
```

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

```
# np.save('/content/drive/MyDrive/BTP/data_preprocessed_python/FFTdata_training', np.array
def round3(x):
  if x <= 4:
    return 0
  if x > 4 and x <= 6:
    return 1
  else:
    return 2
def classify(model, i, test):
  q = model.predict( np.array( [test[i],] )  )
  return round3((np.where(q[0]==max(q[0]))[0][0])), (np.where(q[0]==max(q[0]))[0][0])
classify(model0, 255, x_test)
# test values: 29, 48,100, 300, 400
```

```
(2, 8)
```

```
classify(model0, 255, x_test)[0]
```

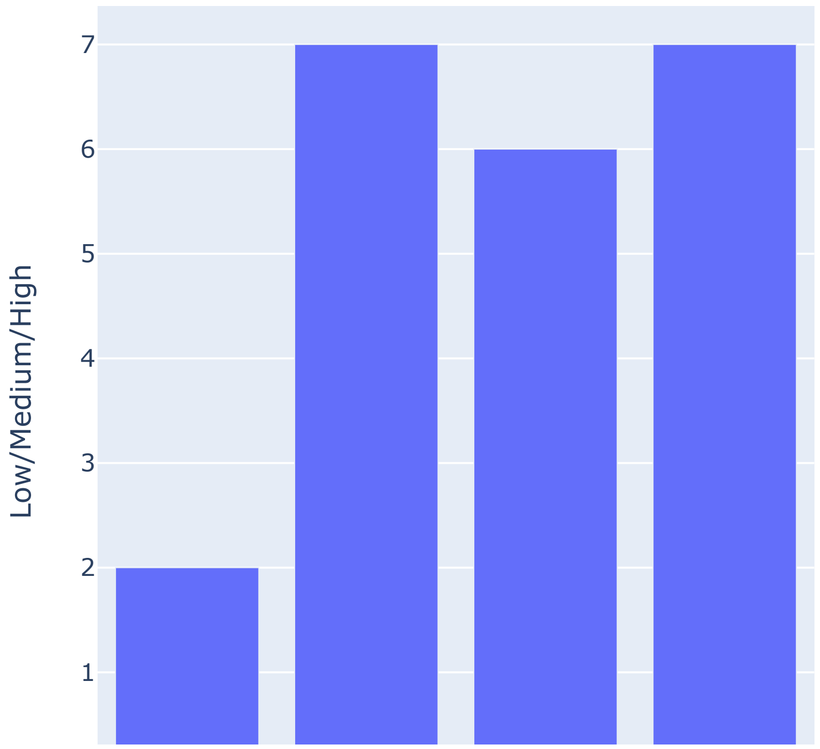
```
2
```

```
i = 48
j = 1
data = [['Arousal',classify(model0, i, x_test)[j] ], ['Valence', classify(model1, i, x_tes

# Create the pandas DataFrame
df = pd.DataFrame(data, columns = ['Label', 'Low/Medium/High'])

# print dataframe.
df
```

	Label	Low/Medium/High
0	Arousal	2
1	Valence	7
2	Dominance	6
3	Liking	7

```
np.save('/content/drive/MyDrive/BTP/data_preprocessed_python/sample_test', x_test[1], allo
```

```
fig = px.bar(df, x='Label', y='Low/Medium/High')
fig.show()
```



```
pip install dash_bootstrap_components
```

```
Collecting dash_bootstrap_components
  Downloading dash_bootstrap_components-1.0.2-py3-none-any.whl (209 kB)
     |████████████████████████████████| 209 kB 28.4 MB/s
Requirement already satisfied: dash>=2.0.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: dash-core-components==2.0.0 in /usr/local/lib/python3
Requirement already satisfied: Flask>=1.0.4 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: dash-html-components==2.0.0 in /usr/local/lib/python3
Requirement already satisfied: dash-table==5.0.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: plotly>=5.0.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: flask-compress in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: Werkzeug<2.0,>=0.15 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: click<8.0,>=5.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: Jinja2<3.0,>=2.10.1 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: itsdangerous<2.0,>=0.24 in /usr/local/lib/python3.7/di
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from pl
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: brotli in /usr/local/lib/python3.7/dist-packages (from
Installing collected packages: dash-bootstrap-components
Successfully installed dash-bootstrap-components-1.0.2
```

```
# Build App
classes={0:'3 Classes',1:'10 Classes'}
import dash_bootstrap_components as dbc

external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = JupyterDash(__name__,external_stylesheets=external_stylesheets)
```

```
app.layout = html.Div(
    children=[
    html.H1("Emotion Recognition using EEG Data",style={'padding-left':700}),
    dbc.Form(
    dbc.Row(
        [
            dbc.Label("Sample", style={'width':"auto",'font-size':30}),
            dbc.Col(
                dcc.Input(type="number",placeholder="Enter Sample No",id="input1",value=0)
                className="me-3",
            ),
            html.Hr(),
            dbc.Label("Classification", style={'width':"auto",'font-size':30}),
            dcc.RadioItems(id='input2', value=0, options=[{'label': classes[c], 'value': c
            html.Hr(),
            dbc.Col(dbc.Button("Submit", color="primary"), style={'width':120,'padding':15
        ],
        className="g-2",style={'padding':70,'margin-left':500,'padding-bottom':100}
    ),style={'backgroundColor':'lightBlue'}
),
   html.Hr(),
   dcc.Upload([
        'Drag and Drop or ',
        html.A('Select a File')

    ], style={
        'width': '100%',
        'height': '60px',
        'lineHeight': '60px',
        'borderWidth': '1px',
        'borderStyle': 'dashed',
        'borderRadius': '5px',
        'textAlign': 'center'
    }),
    html.Div(children=[dcc.Graph(id="graph",figure={})],style = {'display': 'inline-block'
    html.Div(children=[html.Img(src='/content/drive/MyDrive/labelledClasses.jpeg',alt='lab
    ]
)
# Define callback to update graph
@app.callback(
    Output(component_id="graph", component_property='figure'),
    [Input(component_id="input1", component_property="value"),
    Input(component_id="input2",component_property="value")],
    prevent_initial_call=False
)
def update_figure(sampleNo,classW):
  data = [['Arousal',classify(model0,sampleNo,x_test)[classW]], ['Valence', classify(model
  df = pd.DataFrame(data, columns = ['Label', 'Low/Medium/High'])
  fig=px.bar(df, x='Label', y='Low/Medium/High')
  return fig
# Run app and display result inline in the notebook
```

```
app.run_server(debug=True, port=8051,mode='external')
```

```
Dash app running on:
http://127.0.0.1:8051/
```

```
! pip install pyngrok
```

```
Collecting pyngrok
  Downloading pyngrok-5.1.0.tar.gz (745 kB)
     |████████████████████████████████| 745 kB 10.8 MB/s
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-packages (fro
Building wheels for collected packages: pyngrok
  Building wheel for pyngrok (setup.py) ... done
  Created wheel for pyngrok: filename=pyngrok-5.1.0-py3-none-any.whl size=19006 sha25
  Stored in directory: /root/.cache/pip/wheels/bf/e6/af/ccf6598ecefecd44104069371795
Successfully built pyngrok
Installing collected packages: pyngrok
Successfully installed pyngrok-5.1.0
```

```
!ngrok authtoken 20qsZ1gFol3wGsjLsO56scTk0Ks_exmZXUcds4UXNrnhZNjz
```

```
ngrok.kill()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-32-582e608e03de> in <module>()
----> 1 ngrok.kill()

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

SEARCH STACK OVERFLOW

```
from pyngrok import ngrok

# Open a HTTP tunnel on the default port 80
public_url = ngrok.connect(addr = '8050')
```

```
public_url
```

```
# count = 0
# for i in range(1001):
#   q = model0.predict( np.array( [x_test[i],] )  )
#   print(np.where(q[0]==max(q[0]))[0][0], i)
#   if round3(np.where(q[0]==max(q[0]))[0][0]) == round3(np.where(y0_test[i]==max(y0_test[
#     count +=1

# print(count, count/97600)
```

```
# score = model0.evaluate(x_test, y0_test, verbose=1)
# print(model0.metrics_names)
# print(score)
# print('Test loss:', score[0])
# print('Test accuracy:', score[1])
# loss, accuracy, f1_score, precision, recall = model0.evaluate(x_test, y0_test, verbose=0

print("toyota_corolla_2010 (0.412)")
```

✓ 0s completed at 23:01 ● ×