***EMOTION DETECTION USING CNN***

***SOURCE CODE:***

***1.Importing necessary libraries***

pip install scikit-plot

import pandas as pd

import numpy as np

import scikitplot

import random

import seaborn as sns

import keras

import os

from matplotlib import pyplot

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.utils import to\_categorical

import warnings

from tensorflow.keras.models import Sequential

from keras.callbacks import EarlyStopping

from keras import regularizers

from keras.callbacks import ModelCheckpoint,EarlyStopping

from tensorflow.keras.optimizers import Adam,RMSprop,SGD,Adamax

from keras.preprocessing.image import ImageDataGenerator,load\_img

from keras.utils.vis\_utils import plot\_model

from keras.layers import Conv2D, MaxPool2D, Flatten,Dense,Dropout,BatchNormalization,MaxPooling2D,Activation,Input

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

warnings.simplefilter("ignore")

from keras.models import Model

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

from keras.regularizers import l1, l2

import plotly.express as px

from matplotlib import pyplot as plt

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

***2.Loading the data***

data = pd.read\_csv("../input/fer2013/fer2013.csv")

data.shape

data.isnull().sum()

data.head()

***3. Data pre-processing***

CLASS\_LABELS = ['Anger', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sadness', "Surprise"]

fig = px.bar(x = CLASS\_LABELS,

y = [list(data['emotion']).count(i) for i **in** np.unique(data['emotion'])] ,

color = np.unique(data['emotion']) ,

color\_continuous\_scale="Emrld")

fig.update\_xaxes(title="Emotions")

fig.update\_yaxes(title = "Number of Images")

fig.update\_layout(showlegend = True,

title = {

'text': 'Train Data Distribution ',

'y':0.95,

'x':0.5,

'xanchor': 'center',

'yanchor': 'top'})

fig.show()

## 

## *4.Train test validation split*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(pixels, labels, test\_size=0.1, shuffle=False)

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.1, shuffle=False)

print(X\_train.shape)

print(X\_test.shape)

print(X\_val.shape)

plt.figure(figsize=(15,23))

label\_dict = {0 : 'Angry', 1 : 'Disgust', 2 : 'Fear', 3 : 'Happiness', 4 : 'Sad', 5 : 'Surprise', 6 : 'Neutral'}

i = 1

for i **in** range (7):

img = np.squeeze(X\_train[i])

plt.subplot(1,7,i+1)

plt.imshow(img)

index = np.argmax(y\_train[i])

plt.title(label\_dict[index])

plt.axis('off')

i += 1

plt.show()

***5. Design a model***

def cnn\_model():

model= tf.keras.models.Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3), padding='same', activation='relu', input\_shape=(48, 48,1)))

model.add(Conv2D(64,(3,3), padding='same', activation='relu' ))

model.add(BatchNormalization())

model.add(MaxPool2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(128,(5,5), padding='same', activation='relu'))

model.add(BatchNormalization())

model.add(MaxPool2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding='same', activation='relu', kernel\_regularizer=regularizers.l2(0.01)))

model.add(BatchNormalization())

model.add(MaxPool2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding='same', activation='relu', kernel\_regularizer=regularizers.l2(0.01)))

model.add(BatchNormalization())

model.add(MaxPool2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding='same', activation='relu', kernel\_regularizer=regularizers.l2(0.01)))

model.add(BatchNormalization())

model.add(MaxPool2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(256,activation = 'relu'))

model.add(BatchNormalization())

model.add(Dropout(0.25))

model.add(Dense(512,activation = 'relu'))

model.add(BatchNormalization())

model.add(Dropout(0.25))

model.add(Dense(7, activation='softmax'))

model.compile(

optimizer = Adam(lr=0.0001), loss='categorical\_crossentropy',metrics=['accuracy'])

return model

## *6. Visualizing results:*

plt.plot(history.history["loss"],'r', label="Training Loss")

plt.plot(history.history["val\_loss"],'b', label="Validation Loss")

plt.legend()

plt.plot(history.history["accuracy"],'r',label="Training Accuracy")

plt.plot(history.history["val\_accuracy"],'b',label="Validation Accuracy")

plt.legend()

loss = model.evaluate(X\_test,y\_test)

print("Test Acc: " + str(loss[1]))

preds = model.predict(X\_test)

y\_pred = np.argmax(preds , axis = 1 )

label\_dict = {0 : 'Angry', 1 : 'Disgust', 2 : 'Fear', 3 : 'Happiness', 4 : 'Sad', 5 : 'Surprise', 6 : 'Neutral'}

figure = plt.figure(figsize=(20, 8))

for i, index **in** enumerate(np.random.choice(X\_test.shape[0], size=24, replace=False)):

ax = figure.add\_subplot(4, 6, i + 1, xticks=[], yticks=[])

ax.imshow(np.squeeze(X\_test[index]))

predict\_index = label\_dict[(y\_pred[index])]

true\_index = label\_dict[np.argmax(y\_test,axis=1)[index]]

ax.set\_title("**{}** (**{}**)".format((predict\_index),

(true\_index)),

color=("green" if predict\_index == true\_index else "red"))

CLASS\_LABELS = ['Anger', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sadness', "Surprise"]

cm\_data = confusion\_matrix(np.argmax(y\_test, axis = 1 ), y\_pred)

cm = pd.DataFrame(cm\_data, columns=CLASS\_LABELS, index = CLASS\_LABELS)

cm.index.name = 'Actual'

cm.columns.name = 'Predicted'

plt.figure(figsize = (15,10))

plt.title('Confusion Matrix', fontsize = 20)

sns.set(font\_scale=1.2)

ax = sns.heatmap(cm, cbar=False, cmap="Blues", annot=True, annot\_kws={"size": 16}, fmt='g')

from sklearn.metrics import classification\_report

print(classification\_report(np.argmax(y\_test, axis = 1 ),y\_pred,digits=3))

***7. Fine tuning:***

model = cnn\_model()

model.compile(optimizer=tf.keras.optimizers.SGD(0.001),

loss='categorical\_crossentropy',

metrics = ['accuracy'])

history = model.fit(train\_generator,

epochs=30,

batch\_size=64,

verbose=1,

callbacks=[checkpointer],

validation\_data=val\_generator)

loss = model.evaluate(X\_test,y\_test)

print("Test Acc: " + str(loss[1]))

plt.plot(history.history["loss"],'r', label="Training Loss")

plt.plot(history.history["val\_loss"],'b', label="Validation Loss")

plt.legend()

plt.plot(history.history["accuracy"],'r',label="Training Accuracy")

plt.plot(history.history["val\_accuracy"],'b',label="Validation Accuracy")

plt.legend()

## *8. Changing epoch number:*

model = cnn\_model()

model.compile(

optimizer = Adam(lr=0.0001),

loss='categorical\_crossentropy',

metrics=['accuracy'])

checkpointer = [EarlyStopping(monitor = 'val\_accuracy', verbose = 1,

restore\_best\_weights=True,mode="max",patience = 10),

ModelCheckpoint('best\_model.h5',monitor="val\_accuracy",verbose=1,

save\_best\_only=True,mode="max")]

history = model.fit(train\_generator,

epochs=50,

batch\_size=64,

verbose=1,

callbacks=[checkpointer],

validation\_data=val\_generator)

loss = model.evaluate(X\_test,y\_test)

print("Test Acc: " + str(loss[1]))

preds = model.predict(X\_test)

y\_pred = np.argmax(preds , axis = 1 )

CLASS\_LABELS = ['Anger', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sadness', "Surprise"]

cm\_data = confusion\_matrix(np.argmax(y\_test, axis = 1 ), y\_pred)

cm = pd.DataFrame(cm\_data, columns=CLASS\_LABELS, index = CLASS\_LABELS)

cm.index.name = 'Actual'

cm.columns.name = 'Predicted'

plt.figure(figsize = (20,10))

plt.title('Confusion Matrix', fontsize = 20)

sns.set(font\_scale=1.2)

ax = sns.heatmap(cm, cbar=False, cmap="Blues", annot=True, annot\_kws={"size": 16}, fmt='g')