Q5. Build a CNN model to identify the expression given the face

Aditi Mittal, 500095138
Ankit Kumar, 500095038

Aditi Mittal	Data Preprocessing	50%
	Model Development	50%
Ankit Kumar	Data Preprocessing	50%
	Model Development	50%
Joint Tasks	Report Writing	100%
	Video Recording	100%
Project Group	Presentation Recording	100%

Import neccessary Libraries

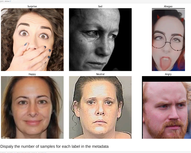
```
import os
import cv2
import numpy as np
import matplotlib
%matplotlib inline
from matplotlib import rcParams, cycler
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
from keras.models import Sequential, Model, load_model
from keras.layers import Input, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D, Dropout, GlobalMaxPooling2D, GlobalAveragePooling2D
from keras.utils.vis_utils import plot_model
from keras.callbacks import ReduceLROnPlate, EarlyStopping, ModelCheckpoint
from keras.preprocessing.image import ImageDataGenerator, load_img
from keras.utils.np_utils import to_categorical
from keras.optimizers import Adam, SGD, RMSprop
from keras.models import load_model
%matplotlib inline
%load_ext tensorboard
%tensorboard --logdir logs
%rm -rf logs
!kill 8000
!ps -ef | grep tensorboard
!ps -ef | grep 8000
```

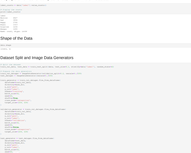
Load the Data

```
!cp -r /kaggle/input/facial-expression-recognition-ferplus/*
!unzip -q -o /kaggle/input/facial-expression-recognition-ferplus.zip
!rm -rf /kaggle/input/*/fer2013.csv
!ls -lrtd /kaggle/input/*
```

Dataset Expression Preview and Summary

```
fig, axs = plt.subplots(1,3,figsize=(12,4))
fig.suptitle('Dataset Expression Preview', fontsize=20)
for ax in axs.ravel():
    ax.axis('off')
axs[0].imshow(cv2.imread('/kaggle/input/fer2013/train/happiness/1.jpg',0),cmap='gray')
axs[0].set_title('Happiness', fontsize=15)
axs[1].imshow(cv2.imread('/kaggle/input/fer2013/train/sadness/1.jpg',0),cmap='gray')
axs[1].set_title('Sad', fontsize=15)
axs[2].imshow(cv2.imread('/kaggle/input/fer2013/train/surprise/1.jpg',0),cmap='gray')
axs[2].set_title('Surprise', fontsize=15)
plt.show()
```





Display the number of samples for each label in the metadata

```
!ls -lrtd /*/*/*/*/*/*
```

```
drwxr-xr-x 2 root root  4096 Jan 1 1970 /kaggle/input/*
drwxr-xr-x 2 root root  4096 Jan 1 1970 /kaggle/input/fer2013
drwxr-xr-x 2 root root  4096 Jan 1 1970 /kaggle/input/fer2013/test
drwxr-xr-x 2 root root  4096 Jan 1 1970 /kaggle/input/fer2013/test/happiness
drwxr-xr-x 2 root root  4096 Jan 1 1970 /kaggle/input/fer2013/test/sadness
drwxr-xr-x 2 root root  4096 Jan 1 1970 /kaggle/input/fer2013/test/surprise
drwxr-xr-x 2 root root  4096 Jan 1 1970 /kaggle/input/fer2013/train
drwxr-xr-x 2 root root  4096 Jan 1 1970 /kaggle/input/fer2013/train/happiness
drwxr-xr-x 2 root root  4096 Jan 1 1970 /kaggle/input/fer2013/train/sadness
drwxr-xr-x 2 root root  4096 Jan 1 1970 /kaggle/input/fer2013/train/surprise
```

Shape of the Data

```
!ls -lrtd
```

```
drwxr-xr-x 2 root root  4096 Jan 1 1970 .
drwxr-xr-x 2 root root  4096 Jan 1 1970 ..
```

Dataset Split and Image Data Generators

```
train_dir = '/kaggle/input/fer2013/train'
test_dir = '/kaggle/input/fer2013/test'
train_datagen = ImageDataGenerator(rescale=1./255,validation_split=0.1)
test_datagen = ImageDataGenerator(rescale=1./255)
print('Training Data Generator:',type(train_datagen))
print('Test Data Generator:',type(test_datagen))
train_generator = train_datagen.flow_from_directory(
    directory=r"/kaggle/input/fer2013/train/",
    target_size=(48,48),
    color_mode="rgb",
    batch_size=32,
    class_mode="categorical",
    shuffle=True,
    seed=42,
    subset='training'
)
print('Training Data Generator:',type(train_generator))
test_generator = test_datagen.flow_from_directory(
    directory=r"/kaggle/input/fer2013/test/",
    target_size=(48,48),
    color_mode="rgb",
    batch_size=32,
    class_mode="categorical",
    shuffle=False,
    seed=42
)
print('Test Data Generator:',type(test_generator))
```

```
Training Data Generator: <class 'keras.preprocessing.image.ImageDataGenerator'>
Test Data Generator: <class 'keras.preprocessing.image.ImageDataGenerator'>
Found 25839 images belong to 3 classes.
Training Data Generator: <class 'keras.preprocessing.image.DirectoryIterator'>
Found 7178 images belong to 3 classes.
Test Data Generator: <class 'keras.preprocessing.image.DirectoryIterator'>
```

CNN Model Definition for Image Classification

```
model = Sequential()
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(48,48,3)))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Flatten)
model.add(Dense(1024,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.0001),metrics=['accuracy'])
model.summary()
```

Model Training (Model 1)

```
model.fit_generator(
    train_generator,
    steps_per_epoch=train_generator.n//32,
    epochs=30,
    validation_data=test_generator,
    validation_steps=test_generator.n//32
)
```

```
Epoch 1/30
807/807 [==============================] - 175s 216ms/step - loss: 0.8966 - accuracy: 0.5811 - val_loss: 0.7966 - val_accuracy: 0.6719
Epoch 2/30
807/807 [==============================] - 170s 211ms/step - loss: 0.7966 - accuracy: 0.6719 - val_loss: 0.7966 - val_accuracy: 0.6719
Epoch 3/30
807/807 [==============================] - 170s 211ms/step - loss: 0.7966 - accuracy: 0.6719 - val_loss: 0.7966 - val_accuracy: 0.6719
Epoch 4/30
807/807 [==============================] - 170s 211ms/step - loss: 0.7966 - accuracy: 0.6719 - val_loss: 0.7966 - val_accuracy: 0.6719
Epoch 5/30
807/807 [==============================] - 170s 211ms/step - loss: 0.7966 - accuracy: 0.6719 - val_loss: 0.7966 - val_accuracy: 0.6719
Epoch 6/30
807/807 [==============================] - 170s 211ms/step - loss: 0.7966 - accuracy: 0.6719 - val_loss: 0.7966 - val_accuracy: 0.6719
Epoch 7/30
807/807 [==============================] - 170s 211ms/step - loss: 0.7966 - accuracy: 0.6719 - val_loss: 0.7966 - val_accuracy: 0.6719
Epoch 8/30
807/807 [==============================] - 170s 211ms/step - loss: 0.7966 - accuracy: 0.6719 - val_loss: 0.7966 - val_accuracy: 0.6719
Epoch 9/30
807/807 [==============================] - 170s 211ms/step - loss: 0.7966 - accuracy: 0.6719 - val_loss: 0.7966 - val_accuracy: 0.6719
Epoch 10/30
807/807 [==============================] - 170s 211ms/step - loss: 0.7966 - accuracy: 0.6719 - val_loss: 0.7966 - val_accuracy: 0.6719
```

Evaluating on test data

```
model.evaluate(test_generator,steps=test_generator.n//32,verbose=1)
```

```
224/224 [==============================] - 14s 63ms/step - loss: 0.7966 - accuracy: 0.6719
```

We observe that our model is overfitting because there is a noticeable difference between the accuracy achieved on the training data and that on the validation data. This discrepancy could be attributed to the imbalance in the dataset.

Confusion matrix and ROC- AUC curve

```
y_pred = model.predict(test_generator)
y_pred = np.argmax(y_pred,axis=1)
print('Confusion Matrix')
print(confusion_matrix(test_generator.classes, y_pred))
print('Classification Report')
target_names = ['Happiness', 'Sad', 'Surprise']
print(classification_report(test_generator.classes, y_pred, target_names=target_names))
y_test = test_generator.classes
y = to_categorical(y_test)
y_pred = model.predict(test_generator)
n_classes = 3
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y[:, i], y_pred[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
colors = cycle(['aqua', 'darkorange', 'cornflowerblue', 'lawnpink', 'lightgreen', 'lavender', 'tan', 'cyan', 'chartreuse', 'deepskyblue', '
```