FACIAL SENTIMENT ANALYSIS

15IT376L- MINOR PROJECT REPORT

Submitted by

Gaurav Sharma (RA1711008010224)

Aditya Chauhan (RA1711008010204)

for the assessment of 3rd year Minor Project

in the

DEPARTMENT OF INFORMATION TECHNOLOGY

SRM IST

KATTANKULATHUR

May, 2020

SRM UNIVERSITY

KATTANKULATHUR

BONAFIDE CERTIFICATE

Certified that this Minor project report **Facial Sentiment Analysis** is the bonafide work of **Gaurav Sharma and Aditya Chauhan** who carried out the project work under my supervision at SRM University , IT Department, Kattankulathur.

SIGNATURE

(**Ms. A. Helen Victoria)**

SIGNATURE

(Dr. G. Vadivu)

(Head of Department-IT)

DECLARATION

I **Gaurav Sharma (RA1711008010224) and Aditya Chauhan (RA1711008010204)** studying in III year B.Tech Information Technology program at, SRM University, Kattankulathur, Chennai, hereby declare that this project is an original work of mine and I have not verbatim copied / duplicated any material from sources like internet or from print media, excepting some vital company information / statistics and data that is provided by the company itself.

Signature of the Students

Date:

Place:

ACKNOWLEDGEMENT

The success and the final outcome of this project required guidance and assistance from different sources and we feel extremely fortunate to have got this all along the completion of our project. Whatever we have done is largely due to such guidance and assistance and we would not forget to thank them.

We owe our profound gratitude to our project guide **Ms. A. Helen Victoria**, who took keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all the Teaching staff of the Department of Information Technology which helped us in successfully completing our minor project work. Also, we would like to extend our sincere regards to all the non-teaching staff of the department of Information Technology for their timely support.

LIST OF CONTENT

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Every person's facial expressions help other people in understanding a person's state of mind i.e. their emotions like person smiles when he/she is happy or cries when they are sad. Understanding their emotions help us judge a person. For example, an interviewer judges a candidate for how confident they are or a counsellor can detect depression or a lie from their patients. All this is done by judging their facial expressions.

Our project is based on using convolutional neural networks which is a deep learning method to perform face sentimental analysis. We will be employing a two-step process of first detecting face in an image and then detecting emotion on that detected face. Convolution layers learn spatial hierarchical patterns from data, which are also translation invariant, so they are able to learn different aspects of images. For example, the first convolution layer will learn small and local patterns, such as edges and corners, a second convolution layer will learn larger patterns based on the features from the first layers, and so on. Pooling layers helps with down sampling and dimension reduction. There are many different CNN models available that differ in aspects like using which loss function or which normalization function is used.

Our aim is to design a neural network capable of classifying a face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

2. INTRODUCTION AND OVERVIEW

Enabling computers to understand a one's emotions based on audio and visual inputs was an impossible task before the advent of deep learning. It overcame the shortcomings of traditional machine learning algorithms, i.e. it can detect any form of pattern in the data. Thus it enables a computer to learn from complex types of data. Facial expressions strongly express one's emotions and are relatively easier to understand than auditory cues. Facial sentiment analysis is a thriving research area with many applications in Human Computer Interaction. A good model requires good data. So in this project, we are working on the *Challenges in Representation Learning: Facial Expression Recognition Challenge* Dataset which is publically available on Kaggle. The dataset consists of over 35k images and has categorized them in 7 categories, namely (*0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral*).

2.1 CONVOLUTIONAL NEURAL NETWORKS

A convolutional neural network, or CNN, is a subset of deep learning and neural networks most commonly used to analyse visual imagery. Compared to other image classification algorithms, convolutional neural networks use minimal pre-processing, meaning the network learns the filters that typically are hand-engineered in other systems. Because CNNs operate with such independence from human effort, they offer many advantages over alternative algorithms.

In mathematics, a convolution is the integral measurement of how two functions overlap as they interact. Similar to the way one might mix two functions by multiplying them together, a convolution exemplifies the changes functions have upon each other. CNNs use this principle in decoding imagery. CNNs analyze the visual elements of the image and attempt to map out the occurrences of matching features.

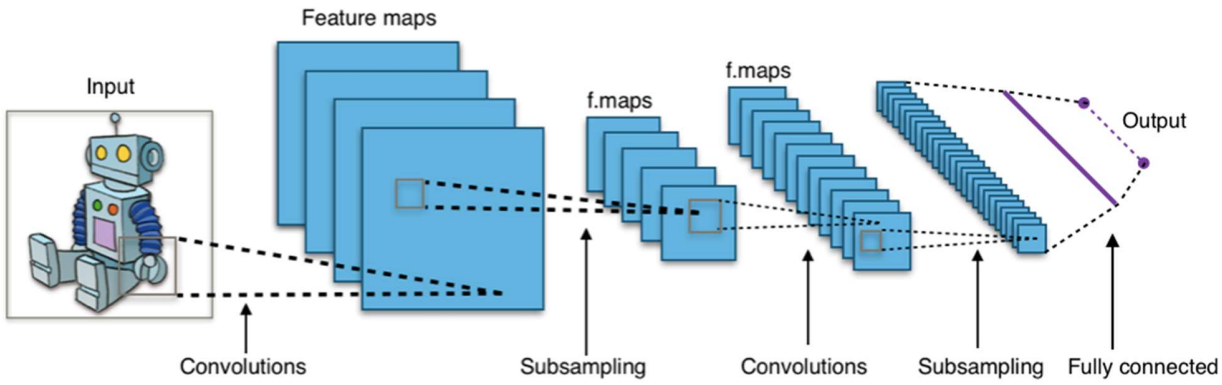

Fig. 1. CNN Basic Architecture

2.2 DEEP LEARNING

Deep learning is a form of machine learning that models patterns in data as complex, multi-layered networks. Because deep learning is the most general way to model a problem, it has the potential to solve difficult problems such as computer vision and natural language processing that outstrip both the conventional programming and other machine learning techniques.

Deep learning not only can produce useful results where other methods fail, but also can build more accurate models than other methods, and can reduce the time needed to

build a useful model. However, training deep learning models requires a great deal of computing power. Another drawback to deep learning is the difficulty of interpreting deep learning models.

2.3 FACIAL EMOTION RECOGNITION

It is the research area with the objective of accurately identifying one's emotions from visual data. FER has many applications, for example, it may be used an organisation to determine customer satisfaction. It can also be used by interviewers to analyse confidence of a candidate. FER is also used in car board systems to track a driver's mentality. It can be used to detect depression and avoiding any incident in the future.

3. ABOUT DATASET

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string a space-separated pixel values in row major order. test.csv contains only the "pixels" column and your task is to predict the emotion column.

The training set consists of 28,709 examples. The public test set used for the leaderboard consists of 3,589 examples. The final test set, which was used to determine the winner of the competition, consists of another 3,589 examples.

This dataset was prepared by Pierre-Luc Carrier and Aaron Courville, as part of an ongoing research project. They have graciously provided the workshop organizers with a preliminary version of their dataset to use for this contest.

4. DESIGN & METHODOLOGY

The dataset was split into 3 parts of 8:1:1 ratio. The highest ratio was used for training while the other two parts were used for testing and validation.

The model was initially designed by following the famous VGG16's [1] trend of pair of convolutional layers followed by a pooling layer. Batch normalization was added between pooling and convolution layers to prevent overfitting. ReLU and LeakyReLU [2] were used as the activation functions. This model consists of 19 hidden layers and over 19 million trainable parameters. To prevent overfitting, dropout and L2 regularizers were also used in the hidden layers. To further prevent overfitting, the above model was implemented using K-Fold Cross Validation. The SciKit-Learn library's KFold method was used to generate 10 folds.

Model: "sequential_1" (Layers)

Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 44, 44, 64)	1664
conv2d_11 (Conv2D)	(None, 40, 40, 64)	102464
batch_normalization_5 (Batch	(None, 40, 40, 64)	256
max_pooling2d_5 (MaxPooling2	(None, 20, 20, 64)	0
dropout_9 (Dropout)	(None, 20, 20, 64)	0
conv2d_12 (Conv2D)	(None, 20, 20, 128)	73856
conv2d_13 (Conv2D)	(None, 20, 20, 128)	147584
batch_normalization_6 (Batch	(None, 20, 20, 128)	512

max_pooling2d_6 (MaxPooling2 (None, 10, 10, 128) 0

dropout_10 (Dropout) (None, 10, 10, 128) 0

conv2d_14 (Conv2D) (None, 10, 10, 256) 295168

conv2d_15 (Conv2D) (None, 10, 10, 256) 590080

batch_normalization_7 (Batch (None, 10, 10, 256) 1024

max_pooling2d_7 (MaxPooling2 (None, 5, 5, 256) 0

dropout_11 (Dropout) (None, 5, 5, 256) 0

conv2d_16 (Conv2D) (None, 5, 5, 512) 1180160

conv2d_17 (Conv2D) (None, 5, 5, 512) 2359808

batch_normalization_8 (Batch (None, 5, 5, 512) 2048

max_pooling2d_8 (MaxPooling2 (None, 2, 2, 512) 0

dropout_12 (Dropout) (None, 2, 2, 512) 0

conv2d_18 (Conv2D) (None, 2, 2, 1024) 4719616

conv2d_19 (Conv2D) (None, 2, 2, 1024) 9438208

batch_normalization_9 (Batch (None, 2, 2, 1024) 4096

max_pooling2d_9 (MaxPooling2 (None, 1, 1, 1024) 0

dropout_13 (Dropout) (None, 1, 1, 1024) 0

flatten_1 (Flatten) (None, 1024) 0

dense_5 (Dense) (None, 512) 524800

dropout_14 (Dropout) (None, 512) 0

dense_6 (Dense) (None, 256) 131328

dropout_15 (Dropout) (None, 256) 0

dense_7 (Dense) (None, 128) 32896

dropout_16 (Dropout) (None, 128) 0

dense_8 (Dense)	(None, 64)	8256
dropout_17 (Dropout)	(None, 64)	0
dense_9 (Dense)	(None, 7)	455

Total params: 19,614,279
Trainable params: 19,610,311
Non-trainable params: 3,968

Table 1. Architecture

The model was run on kaggle's notebook which provides 16 GB RAM and high end processors. The compilation was done on Tensor Processing Units (TPU) which provides high learning speed and are ideal with Tensorflow. TensorflowV2 was used in this project. Each fold was run for 12 epochs, with a total runtime of 20 minutes for the whole notebook.

5. SOURCE CODE

```
import pandas as pd
df=pd.read_csv('fer2013.csv')
print(df)

import numpy as np
train_samples = df[df['Usage']=="Training"]
validation_samples = df[df["Usage"]=="PublicTest"]
test_samples = df[df["Usage"]=="PrivateTest"]

out=df.emotion.astype(np.int32).values

df=df[:35338]
out=out[:35338]
sample=df[35885:35886]
inp=np.array([ np.fromstring(image, np.uint8, sep=" ").reshape((48, 48)) for image in
df.pixels])
inp=inp[:35338]

import tensorflow as tf
from tensorflow.keras import layers, regularizers
from tensorflow.keras.models import Sequential

# detect and init the TPU
tpu = tf.distribute.cluster_resolver.TPUClusterResolver()
tf.config.experimental_connect_to_cluster(tpu)
tf.tpu.experimental.initialize_tpu_system(tpu)

# instantiate a distribution strategy
tpu_strategy = tf.distribute.experimental.TPUStrategy(tpu)

#Achieved training_accuracy=93.85%
with tpu_strategy.scope():
    model=Sequential()

    #Convolution Layers
    #1
    model.add(layers.Conv2D(64, (5, 5), input_shape=(48, 48, 1), strides=1, activati
on=layers.LeakyReLU(alpha=0.3), kernel_regularizer=regularizers.l2(0.02)))
```

```
    model.add(layers.Conv2D(64, (5, 5), strides=1, activation=layers.LeakyReLU(al
pha=0.3)))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D(pool_size=(2,2)))
    model.add(layers.Dropout(0.2))

    #2
    model.add(layers.Conv2D(128, (3, 3), strides=1, padding='same', activation='relu
'))
    model.add(layers.Conv2D(128, (3, 3), strides=1, padding='same', activation='relu
'))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D(pool_size=(2,2), strides=2))
    model.add(layers.Dropout(0.4))

    #3
    model.add(layers.Conv2D(256, (3, 3), strides=1, padding='same', activation='relu
'))
    model.add(layers.Conv2D(256, (3, 3), strides=1, padding='same', activation='relu
'))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D(pool_size=(2,2), strides=2))
    model.add(layers.Dropout(0.4))

    #4
    model.add(layers.Conv2D(512, (3, 3), strides=1, padding='same', activation='relu
'))
    model.add(layers.Conv2D(512, (3, 3), strides=1, padding='same', activation='relu
'))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D(pool_size=(2,2)))
    model.add(layers.Dropout(0.2))

    #5
    model.add(layers.Conv2D(1024, (3, 3), strides=1, padding='same', activation='rel
u'))
    model.add(layers.Conv2D(1024, (3, 3), strides=1, padding='same', activation='rel
u'))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Dropout(0.3))
```

```
        #Fully Connected Layers
        #5
        model.add(layers.Flatten())
        model.add(layers.Dense(512, activation='relu'))
        model.add(layers.Dropout(0.35))
        model.add(layers.Dense(256, activation='relu'))
        model.add(layers.Dropout(0.35))
        model.add(layers.Dense(128, activation='relu'))
        model.add(layers.Dropout(0.3))
        model.add(layers.Dense(64, activation='relu'))
        model.add(layers.Dropout(0.3))
        model.add(layers.Dense(7, activation='softmax'))

        model.compile(loss='sparse_categorical_crossentropy', optimizer='Adam', metric
s=['accuracy'])


model.summary()

inp = inp.reshape((-1, 48, 48, 1)).astype(np.float32)

inp_std=inp/255.

from sklearn.model_selection import KFold
kf = KFold(n_splits=10)
kf.get_n_splits(inp)
fold=1
batch_s=110
for train_index, test_index in kf.split(df):
    print("FOLD: "+str(fold))
    print("TRAIN:", train_index, "TEST:", test_index)
    inp_train_std, inp_test_std = inp_std[train_index], inp_std[test_index]
    out_train, out_test = out[train_index], out[test_index]

    if(len(inp_train_std)!=31800):
        inp_train_std=inp_train_std[:31800]
        out_train=out_train[:31800]
```

```
    history = model.fit(inp_train_std, out_train,
                        batch_size=120,
                        #128 for previous methods without cross validation
                        epochs=12,
                        validation_data=(inp_test_std, out_test),
                        )


    import matplotlib.pyplot as plt
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('model accuracy for FOLD '+str(fold))
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
    fold+=1
    print()

sample=test_samples[0:]

samp_pxls=np.array([np.fromstring(image, np.uint8, sep=" ").reshape((48, 48)) for im
age in sample.pixels])
samp_emot=sample.emotion.astype(np.int32).values

samp_pxls=samp_pxls/255.

#Saving Test Image
tmp_pxls=samp_pxls[5]
tmp_emot=samp_emot[5]
array = np.array(pxls, dtype=np.uint8)
from PIL import Image
img=Image.fromarray(array)
img.save('tmp.png')

samp_pxls = samp_pxls.reshape((-1, 48, 48, 1)).astype(np.float32)


preds=model.predict_classes(samp_pxls) lbls=samp_emot

#Confusion Matrix
```

```
from sklearn.metrics import confusion_matrix as conf_mat
import itertools
import matplotlib.pyplot as plt
labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']
title='Confusion matrix'

cm=conf_mat(lbls, preds)

plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Reds)
plt.title(title)
plt.colorbar()
tick_marks = np.arange(len(labels))
plt.xticks(tick_marks, labels, rotation=45)
plt.yticks(tick_marks, labels)
fmt = 'd'
thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, format(cm[i, j], fmt),
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.tight_layout()
plt.show()
```

6. RESULT & ANALYSIS

A final accuracy of 95.64 and 92.81 were achieved on the training set and validation set respectively. The final losses of 0.1559 and 0.2289 were also noted on the training and validation set respectively. The accuracies were plotted for each of the 10 folds and have been provided below:

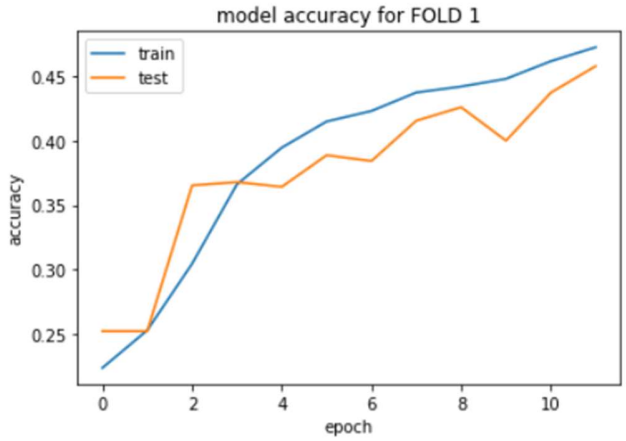

Fig. 2. Fold 1

Fig. 3. Fold 2

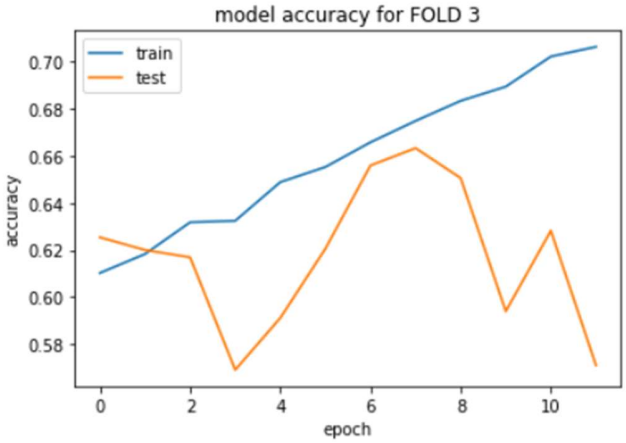

Fig. 4. Fold 3

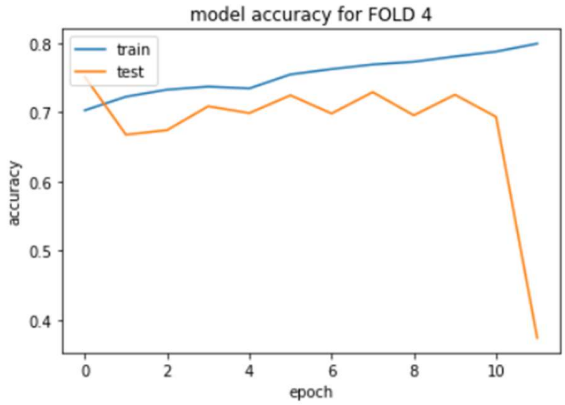

Fig. 5. Fold 4

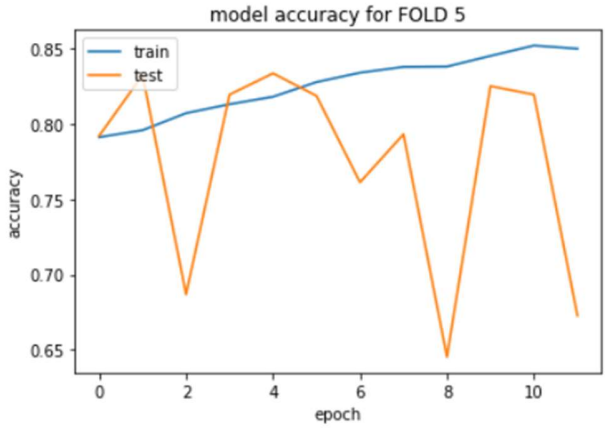

Fig. 6. Fold 5

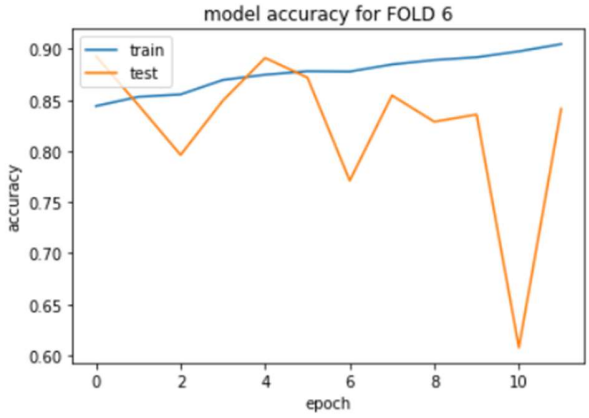

Fig. 7. Fold 6

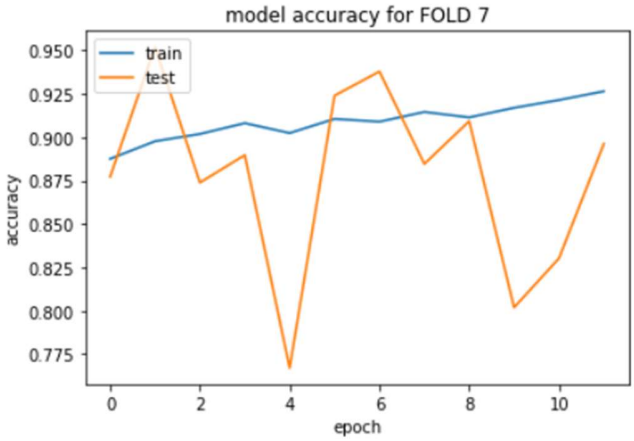

Fig.8. Fold 7

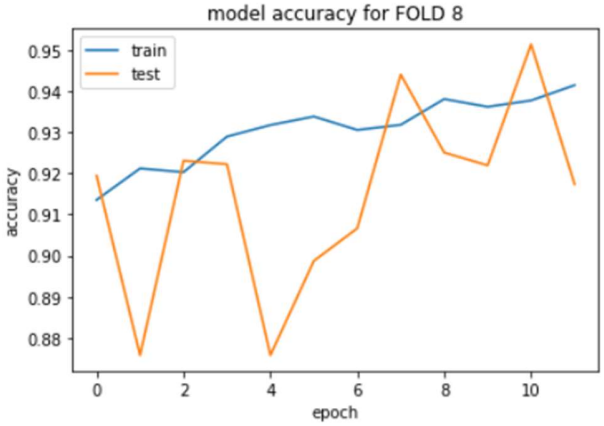

Fig. 9. Fold 8

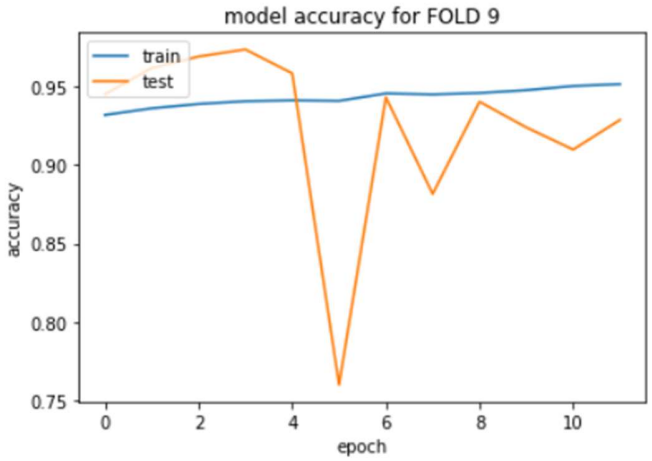

Fig. 10. Fold 9

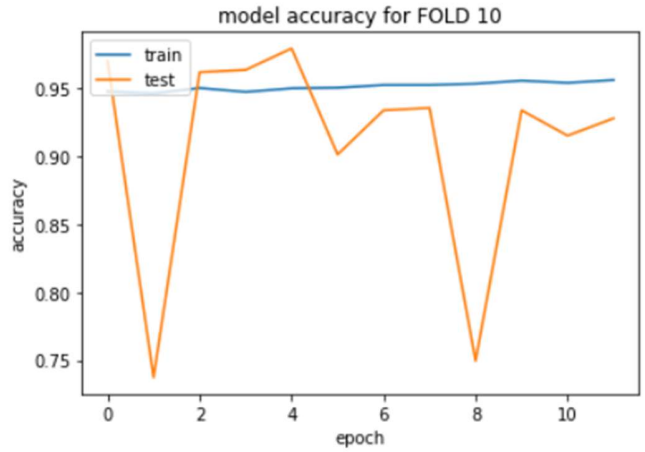

Fig. 11. Fold 10

The graphs show a steady increase in accuracy across 10 folds for training set, which becomes uniform at ~95% in the last fold. The validation accuracy fluctuates a lot in all folds which can be inferred from the fact that the validation set is very small in size (~3k images). However, the final accuracies for both training and validation sets were found to be satisfactory. It was observed that without K-Fold Cross Validation, the

model was always overfitting. Tensorflow was used to train the model. Facial Sentiment Analysis was successfully achieved using Convolutional Neural Networks.

Finally, a confusion matrix was plotted on the basis of the test set which consisted of 3040 images.

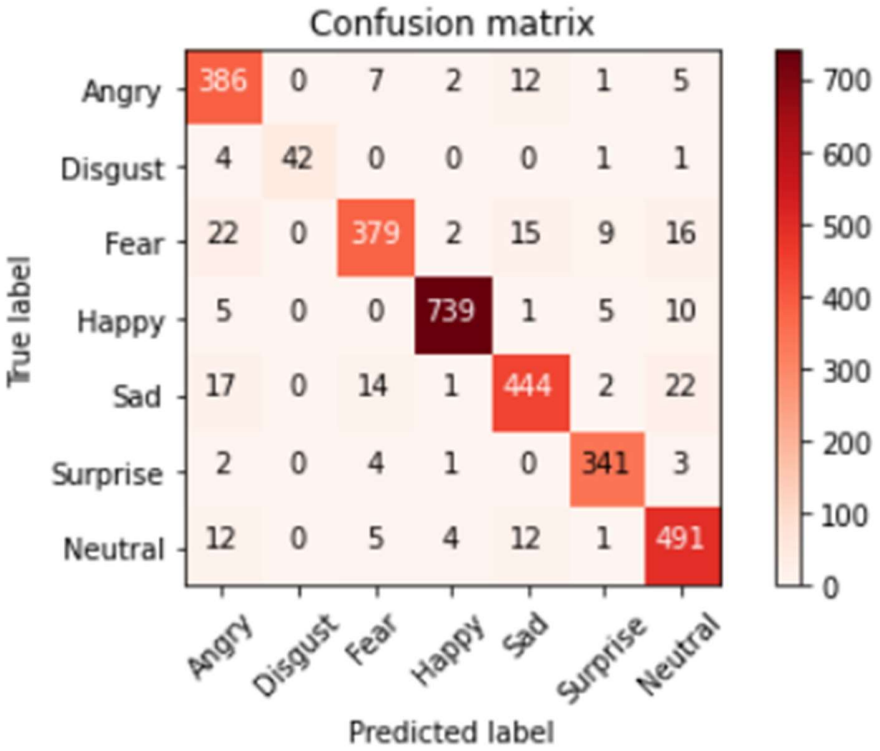

Fig. 12. Confusion Matrix

7. REFERENCES

1. Karen Simonyan and Andrew Zisserman, "*Very Deep Convolutional Networks for Large-Scale Image Recogntion*"

2. Kaiming He et al., "*Delving Deep into Rectifiers*"

3. Nithya Roopa. S, "*Emotion Recognition from Facial Expression using Deep Learning*"