EMOTION DETECTION BY FACIAL EXPRESSIONS

Under the guidance of:

Ms. Kirti Aggarwal

Submitted by:

Rishabh Jaiswal (16103008) B10
Rozel Agrawal (16103038) B2
Maulishri Agrawal (16103047) B10

Jaypee Institute of Information Technology University

(Declared Deemed to the University U/S of UGC Act)

A-10, Sector-62, Noida, India

ABSTRACT

Emotion plays an important role in human life. Interpersonal human communication includes not only language that is spoken, but also non-verbal cues as hand gestures, tone of the voice and most importantly facial expressions, which are used to express feeling and give feedback. Human beings express emotions in day to day interactions. Understanding and knowing how to react to people's expression greatly enriches the interaction.

Human emotion recognition plays an important role in the interpersonal relationship. The automatic recognition of emotions has been an active research topic from early eras. Therefore, there are several advances made in this field. Emotions are reflected from speech, hand and gestures of the body and through facial expressions. Hence extracting and understanding of emotion has a high importance of the interaction between human and machine communication. With this project our main aim is the recognition of the various emotions displayed by a person. We have implemented a real time emotion recognition system.

The human face plays a huge role for automatic recognition of emotion in the field of identification of human emotion and the interaction between human and computer. Facial expression recognition system requires to overcome the human face having multiple variability such as colour, orientation, expression, posture, texture and so on. In our framework we have used neural networks, to detect the emotion- facial attributes extraction by principal component analysis is used and clustering of different facial expression with respective emotions. Finally, to determine facial expressions separately, the processed feature vector is channelled through the already learned pattern classifiers.

Namely seven human emotions have been identified using the following model, they are- anger, disgust, fear, happiness, sadness, surprise and neutral.

OBJECTIVE

Facial expressions are used to reconcile the emotional state of a person. This emotional look (expression on a person’s face) communicates a lot about the person’s internal condition, as emotions are an outer make-up for their specific state of mind. Observers can easily evaluate a person’s emotional state in order to assist him or give feedback. For example, nurses observe such things in case of mental health centre patients, to help doctors in making a well diagnosed treatment plan for them.

Understanding a person’s emotions and expressions is very important in various fields, such as medical institutions, counselling centres etc., as well as in day to day life. With is project our aim is to recognize the expression of a person at a given time from the seven different emotions expressed by them throughout their lifetime. The expressions we have recognized include anger, disgust, fear, happiness, sadness, surprise and neutral.

DIVISION OF WORK AMONG GROUP MEMBERS

Rishabh Jaiswal – Research, CNN Model Development and Experimentation, Error Analysis, Research Paper Write-up

Maulishri Agrawal – Research, Development and Experimentation, Error and Accuracy Analysis, Research Paper Write-up, Final Report Write-up

Rozel Agrawal – Research, Development and Experimentation, Error and Accuracy Analysis, Final Report Write-up

BACKGROUND STUDY AND FINDINGS

PRE PROCESSING AND LOADING OF DATA

1. Loaded the required libraries:

- gplots
- e1071
- keras
- caret

2. The dataset is loaded and analysed by printing the first 5 rows, each data point consists of 2304 pixels, which can be resized into a 48*48 gray scale image.

3. The dataset is then divided into Training, Testing and Validation sets.

4. Further we have visualized a data point by reshaping the 2304 pixels into a 48*48 pixel image, and printed it along with the emotion it displays.

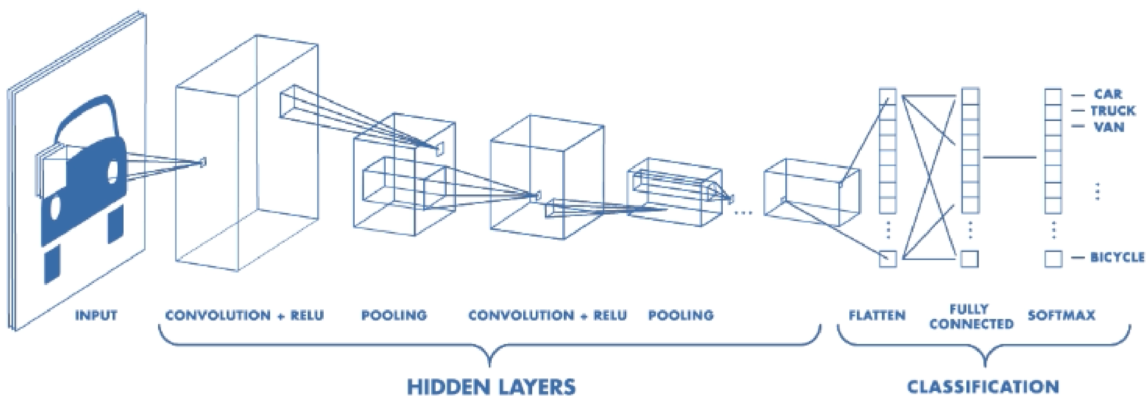
5. After this, we reshaped all the data points for CNN.

CONVOLUTION NEURAL NETWORK (CNN)

The project aims at recognizing the facial expressions of the person whose image is supplied to the system. To achieve this goal, we are using Deep Learning Convolution Neural Network (CNN) Model and Support-Vector Machine(SVM) to train and test the dataset.

CNN is a collection of two types of layers-

1. Hidden Layers/ Feature Extraction Part
    a. Convolution layer
    b. Pooling layer
2. Classifier Part



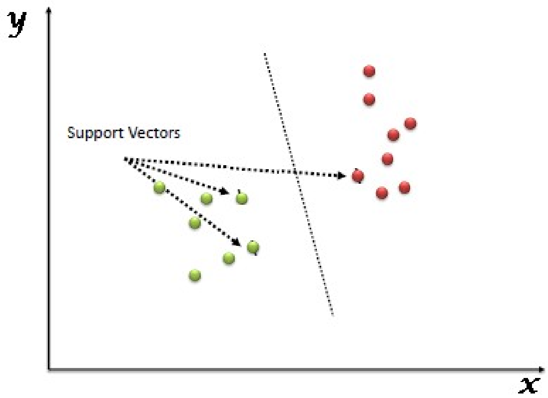
Convolution is a mathematical operation which involves a combination of two functions to produce a third function. In CNN the convolution is performed on the input data with the use of a filter to produce a feature map.

Pooling layer is added after a convolution layer. It performs continuous dimensionality reduction i.e. it reduces the number of parameters and computations thereby shortening training time and controlling overfitting. One such pooling technique is called max-pooling, which takes the maximum value in each window which decreases the feature map size while keeping the significant information.

Dropout is a technique where randomly selected neurons are ignored during the training. They are “dropped out” randomly. This is a great technique which is used to reduce overfitting in our model and to get well-generalized results.

SUPPORT VECTOR MACHINE (SVM)

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.



PROJECT – DESIGN AND STRUCTURE

SVM STRUCTURE

The first model we tried was SVM. To establish a baseline, we first used raw, gray scale pixel values as the features for the SVM. With this combination, we achieved very low accuracy for even the most common emotion every time. Due to the large data set by this method we were only able to train initial 3000 pictures as this method was taking so long.

We then used Principal Component Analysis (PCA) to attempt to isolate the most important components for our analysis. Reducing the dimensionality of the images allowed us to use the full training set. Initially we experimented with 40 components, we achieved an accuracy equally bad as the last experiment. Then we tried by experimenting with 150 components but due to lack of resources we have to reduce to the number of components to 80. By experimenting with 80 components we achieved an accuracy of 26%.

RANDOM FOREST

Next model we tried was Random Forest. We first used raw, gray scale pixel values as feature for it. We initially used 3000 data points by which we achieved very low accuracy of 26%. We then used Principal Component Analysis (PCA) to compress the number of features and retain the important features of our data set in order to achieve a better accuracy. We experimented with 80 components same as we used in SVM by which we achieved an accuracy of 24.1%.

CNN MODEL STRUCTURE

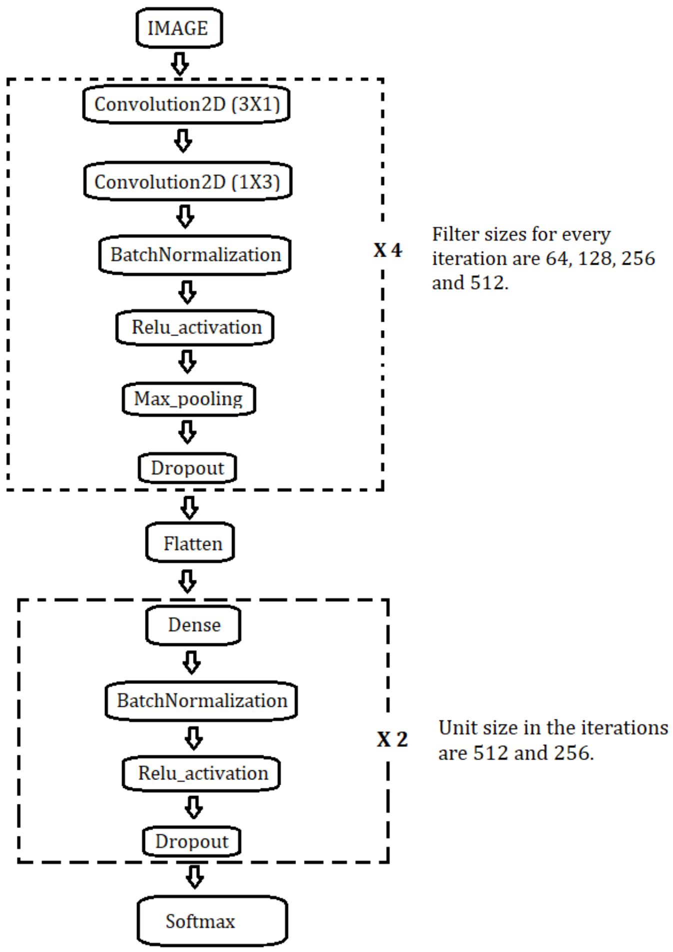
The model takes a 48*48 pixels image as input. It is supplied to the CNN model.

The CNN model consists of multiple convolution layers. Their specifications are:

- The number of filters applied are different. There are 4 filter sizes – 64, 128, 256 and 512.
- Each filter is convoluted using two kernels – 3*1 and 1*3 before being normalized.
- After batch normalization, **Relu** activation function is applied, followed by **max_pooling** and **dropout.**
- All these steps are performed on all the different filters to receive output.

The **flatten()** function is then applied to convert 2D data into 1D, without hampering the batch size.

Finally, **dense()** function uses the features learned using the layers and maps it to the labels. During testing, this layer is responsible for creating the final label for the image being processed.



Dataset Specifications

The data consists of 48*48 pixel gray scale 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 into one of the seven categories:

0 = Angry
1 = Disgust
2 = Fear
3 = Happy
4 = Sad
5 = Surprise
6 = Neutral

The training set consists of 35,888 examples. **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.

We have used FER (Facial Expression Recognition) 2013 dataset.

Training set – 28709

Test set – 3589

Validation – 3589

Accuracy for the CNN Model is coming out to be maximum, equal to 65.3%.

ACCURACY COMPARISON FOR ALL THE THREE MODELS

MODEL	TRAINING ACCURACY	TESTING ACCURACY	AREA UNDER CURVE (AUC)
SVM	35.40%	25.30%	54.79%
RANDOM FOREST	57.00%	24.10%	54.30%
CNN	95.80%	65.30%	74.03%

IMPLEMENTATION

CODE

```
library(randomForest)
library(pROC)
library(FactoMineR)
library(gplots)
library(e1071)
library(keras)
library(caret)
getwd()

data<-read.csv('fer2013.csv')
print(head(data))

print(nrow(data))
print(ncol(data))
summary(data$Usage)

train_set=subset(data,data$Usage=='Training')
valid_set=subset(data,data$Usage=='PublicTest')
test_set=subset(data,data$Usage=='PrivateTest')

emotion_labels<-list("Angry", "Disgust", "Fear", "Happy", "Sad", "Surprise", "Neutral")
num_classes=length(emotion_labels)
print(num_classes)

depth=1
height=48
width=height


a<-as.vector(train_set$pixels[1])
as.numeric(strsplit(a,split=" ")[[1]])->a
a<-as.vector(a)
a<-array_reshape(a,c(48,48))
x <- seq(0, 1, length = nrow(a))
y <- seq(0, 1, length = ncol(a))
print(emotion_labels[train_set$emotion[1]+1])
image(x, y, a, col = grey(seq(0, 0.4, length = 256)))

trainx<-as.vector(train_set$pixels)
testx=as.vector(test_set$pixels)
validx=as.vector(valid_set$pixels)
length(validx)
```

```
testx<-paste(testx,collapse=" ")
as.numeric(strsplit(testx,split=" ")[[1]])->testx
testx<-array_reshape(testx,c(3589,48,48,1))

trainx<-paste(trainx,collapse=" ")
as.numeric(strsplit(trainx,split=" ")[[1]])->trainx
trainx<-array_reshape(trainx,c(28709,48,48,1))

validx<-paste(validx,collapse=" ")
as.numeric(strsplit(validx,split=" ")[[1]])->validx
validx<-array_reshape(validx,c(3589,48,48,1))

tx<-trainx[1:3000,,,]
tx<-array_reshape(tx,c(3000,48,48,1))
dim(trainx)
dim(tx)
dim(testx)
dim(validx)

validy<-valid_set$emotion
validy<-to_categorical(validy,num_classes)
trainy<-train_set$emotion
trainy<-to_categorical(trainy,num_classes)
testy<-test_set$emotion
testy<-to_categorical(testy,num_classes)

ty<-train_set$emotion
ty<-ty[1:3000]
ty<-to_categorical(ty,num_classes)
dim(trainy)
dim(ty)
dim(testy)

checkpoint_dir <- "checkpoints"
dir.create(checkpoint_dir, showWarnings = FALSE)
filepath <- file.path(checkpoint_dir, "model.hdf5")
cp_callback <- callback_model_checkpoint(
  filepath = filepath,
  save_weights_only = TRUE,
  verbose = 1
)
model<-keras_model_sequential()
model %>%
        layer_conv_2d(filters=64,
                      kernel_size=c(3,1),
                      padding='same',
```

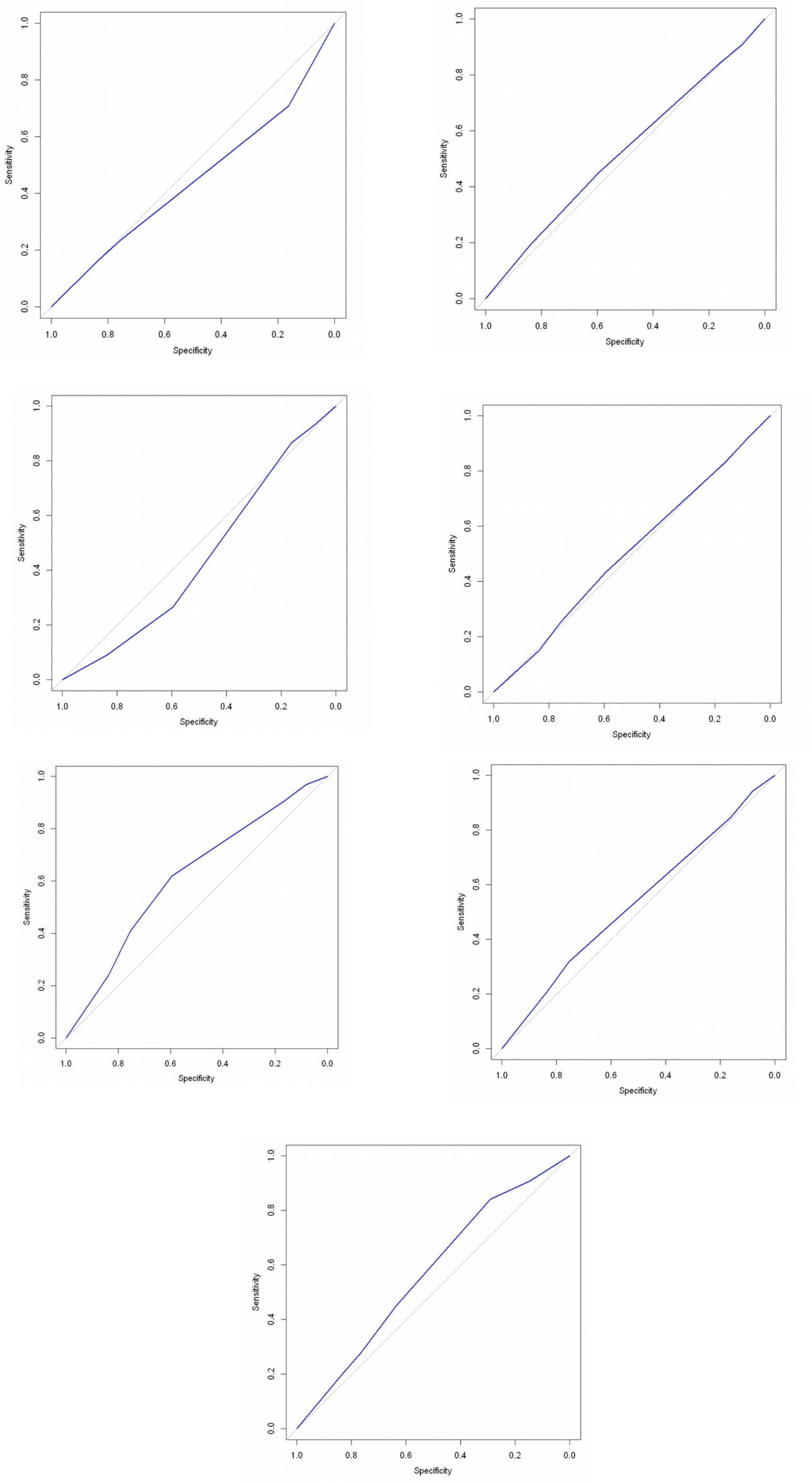
```
            input_shape=c(48,48,1))%>%
  layer_conv_2d(filters=64,
            kernel_size=c(1,3),
            padding='same')%>%
  layer_batch_normalization()%>%
  layer_activation('relu')%>%
  layer_max_pooling_2d(pool_size=c(2,2),
                padding='same')%>%
  layer_dropout(rate=0.25)%>%

  layer_conv_2d(filters=128,
            kernel_size=c(3,1),
            padding='same')%>%
  layer_conv_2d(filters=128,
            kernel_size=c(1,3),
            padding='same')%>%
  layer_batch_normalization()%>%
  layer_activation('relu')%>%
  layer_max_pooling_2d(pool_size=c(2,2),
                padding='same')%>%
  layer_dropout(rate=0.25)%>%

  layer_conv_2d(filters=256,
            kernel_size=c(3,1),
            padding='same')%>%
  layer_conv_2d(filters=256,
            kernel_size=c(1,3),
            padding='same')%>%
  layer_batch_normalization()%>%
  layer_activation('relu')%>%
  layer_max_pooling_2d(pool_size=c(2,2),
                padding='same')%>%
  layer_dropout(rate=0.25)%>%

  layer_conv_2d(filters=512,
            kernel_size=c(3,1),
            padding='same')%>%
  layer_conv_2d(filters=512,
            kernel_size=c(1,3),
            padding='same')%>%
  layer_batch_normalization()%>%
  layer_activation('relu')%>%
  layer_max_pooling_2d(pool_size=c(2,2),
                padding='same')%>%
  layer_dropout(rate=0.25)%>%
```

```
          layer_flatten()%>%

          layer_dense(units=512)%>%
          layer_batch_normalization()%>%
          layer_activation('relu')%>%
          layer_dropout(rate=0.25)%>%

          layer_dense(units=256)%>%
          layer_batch_normalization()%>%
          layer_activation('relu')%>%
          layer_dropout(rate=0.25)%>%

          layer_dense(units=7)%>%
          layer_activation('softmax')%>%
          compile(loss='categorical_crossentropy',
                optimizer=optimizer_adam(),
                metrics=c('accuracy'))

b_size=32
num_epochs=25
history<- model%>%
       fit(trainx,
          trainy,
          verbose=1,
          epochs=num_epochs,
          batch_size=b_size,
          shuffle=TRUE,
         validation_data=list(validx,validy),
         callbacks = list(cp_callback))

#svm model
trainsvmx<-as.vector(train_set$pixels)
trainsvmx<-paste(trainsvmx,collapse=" ")
as.numeric(strsplit(trainsvmx,split=" ")[[1]])->trainsvmx
trainsvmx<-array_reshape(trainsvmx,c(28709,2304))
dim(trainsvmx)

pc<-prcomp(trainsvmx,center=TRUE,scale.=TRUE)
dim(pc$x)
trainnew<-predict(pc,trainsvmx)
trainnew<-trainnew[,1:80]
dim(trainnew)
print(trainnew[1,])

trainsvmy<-train_set$emotion
length(trainsvmy)
```

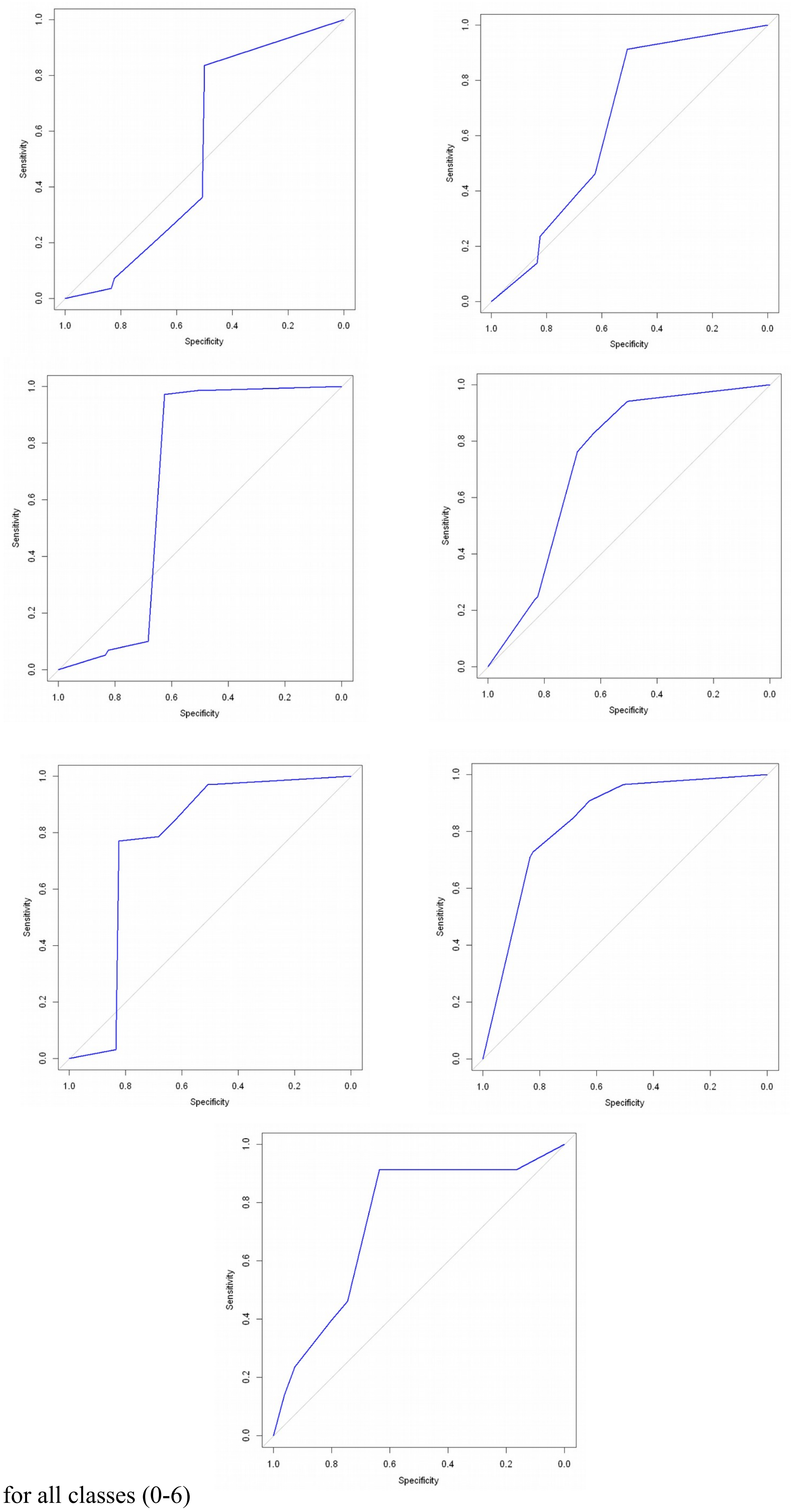
```
testsvmx<-as.vector(test_set$pixels)
testsvmx<-paste(testsvmx,collapse=" ")
as.numeric(strsplit(testsvmx,split=" ")[[1]])->testsvmx
testsvmx<-array_reshape(testsvmx,c(3589,2304))
pct<-prcomp(testsvmx,center=TRUE,scale.=TRUE)
testnew<-predict(pct,testsvmx)
testnew<-testnew[,1:80]
dim(testnew)

testsvmy<-test_set$emotion
length(testsvmy)

svmmodel<-svm(trainnew,trainsvmy,type='C',kernel='linear')
pred<-predict(svmmodel,testnew)
testsvmy<-as.factor(testsvmy)
conf<-confusionMatrix(pred,testsvmy)
conf
t<-table(pred,testsvmy)
h<-
heatmap.2(as.matrix(t),symm=TRUE,scale="column",Rowv=NA,,margin=c(4,4),col=heat.co
lors(256),
           key=FALSE,trace="none",
         main="heatmap for svmmodel",xlab="actual",ylab=("predicted"))
pred<-as.numeric(pred)
m<-multiclass.roc(testsvmy,pred)
print(m$auc)
for(i in (1:7))
{
   plot(m$rocs[[i]],col="blue")
}
#naive bayes
trainnbx<-as.vector(train_set$pixels)
trainnbx<-paste(trainnbx,collapse=" ")
as.numeric(strsplit(trainnbx,split=" ")[[1]])->trainnbx

trainnbx<-array_reshape(trainnbx,c(28709,2304))
dim(trainnbx)
pc<-prcomp(trainnbx,center=TRUE,scale.=TRUE)
trainnew<-predict(pc,trainnbx)
trainnew<-trainnew[,1:80]
dim(trainnew)
trainnby<-train_set$emotion
length(trainnby)
testnbx<-as.vector(test_set$pixels)
testnbx<-paste(testnbx,collapse=" ")
as.numeric(strsplit(testnbx,split=" ")[[1]])->testnbx
```

```
testnbx<-array_reshape(testnbx,c(3589,2304))
pct<-prcomp(testnbx,center=TRUE,scale.=TRUE)
testnew<-predict(pct,testnbx)
testnew<-testnew[,1:80]
dim(testnew)
testnby<-test_set$emotion
length(testnby)
rf<- randomForest(trainnew,trainnby)
prednb<-predict(rf,trainnew)
r<-round(prednb,0)
trainnby<-as.factor(trainnby)
r<-as.factor(r)
conf<-confusionMatrix(r,trainnby)
conf
prednew<-predict(rf,testnew)
r1<-round(prednew,0)
length(r1)
testnby<-as.factor(testnby)
r1<-as.factor(r1)
conf1<-confusionMatrix(r1,testnby)
conf1
r1<-as.numeric(r1)
m<-multiclass.roc(testnby,r1)
m$auc
model %>% load_model_weights_hdf5('weights.h5')
model %>% evaluate(testx,testy)
model %>% evaluate(tx,ty)

pred<-model%>% predict_classes(testx)
t<-table(Predicted=pred,Actual=test_set$emotion)
t
h<-
heatmap.2(as.matrix(t),symm=TRUE,scale="column",Rowv=NA,,margin=c(4,4),col=heat.co
lors(256),
             key=FALSE,trace="none",
          main="heatmap for CNN",xlab="actual",ylab=("predicted"))

pred<-as.numeric(pred)
m<-multiclass.roc(test_set$emotion,pred)
print(m$auc)
for(i in (1:7))
{
   plot(m$rocs[[i]],col="blue")
}
```

SNAPSHOTS



ROC curve for all the classes (0-6) in SVM.



ROC curve for all classes (0-6)
in CNN.

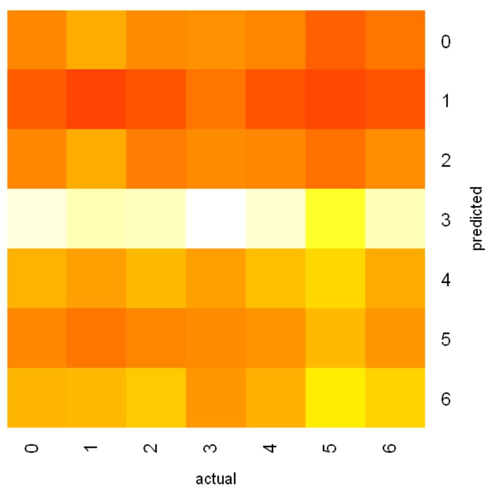
TESTING – OUTPUT

SVM

1. CONFUSION MATRIX

Predicted \ Actual	0	1	2	3	4	5	6
0	246	9	45	11	34	12	21
1	4	26	1	0	2	0	1
2	57	6	238	13	66	53	35
3	29	3	34	766	39	24	38
4	69	7	85	28	305	6	75
5	5	2	51	15	6	308	11
6	81	2	74	46	142	13	445

2. HEAT MAP



RANDOM FOREST

1. CONFUSION MATRIX

Prediction \ Reference	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	3	0	2	2	4	0	0
3	420	48	410	755	479	276	432
4	68	7	116	122	111	140	194
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

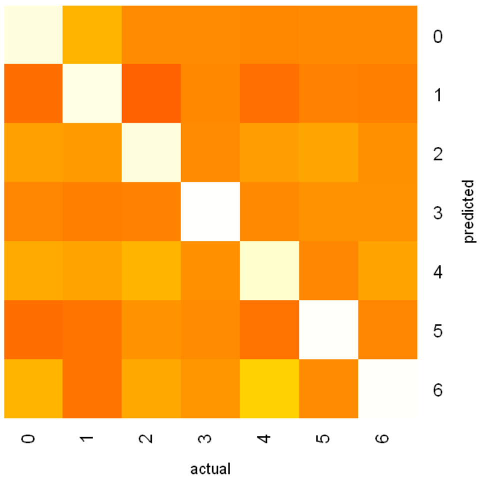
2. AREA UNDER THE CURVE = 54.30%

CNN

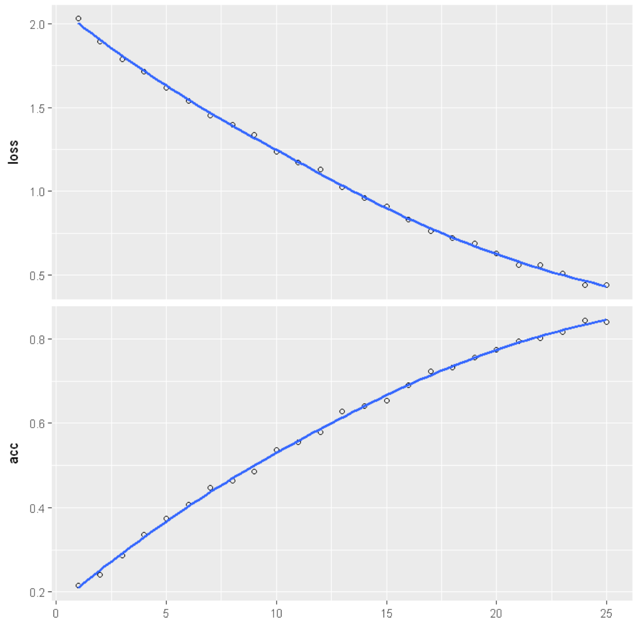
1. CONFUSION MATRIX

Predicted \ Actual	0	1	2	3	4	5	6
0	246	9	45	11	34	12	21
1	4	26	1	0	2	0	1
2	57	6	238	13	66	53	35
3	29	3	34	766	39	24	38
4	69	7	85	28	305	6	75
5	5	2	51	15	6	308	11
6	81	2	74	46	142	13	445

2. HEAT MAP



MODEL ACCURACY PLOT



REFERENCES

- https://www.analyticsvidhya.com/blog/2017/06/hands-on-with-deep-learning-solution-for -age-detection-practice-problem/
- https://www.analyticsvidhya.com/blog/2018/05/24-ultimate-data-science-projects-to-boo st-your-knowledge-and-skills/
- https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
- https://appliedmachinelearning.blog/2018/11/28/demonstration-of-facial-emotion-recognition-on-real-time-video-using-cnn-python-keras/