



Real Time Emotion Classification using User's Facial Expression

Introduction

Facial behaviour is one of the most important cues for sensing human emotion and intentions among people. As computing becomes more human centred, an automatic system for accurate facial expression analysis is relevant in emerging fields such as interactive games (for instance, the games played using Microsoft Kinect), online education, entertainment, autonomous driving, analysis of viewer reaction to advertisements, etc. For example, the reactions of gamers could be used as feedback to improve the gaming experience on systems like the Microsoft Kinect.

The bottom right corner of the slide features a decorative graphic consisting of several overlapping, semi-transparent geometric shapes. These shapes are primarily in shades of teal, dark blue, and light green, creating a modern, abstract design.

Scope of Project Work

- Developing interactive games which can use the user's emotion to keep his interest in the game. Also, the reaction can be used as the feedback for the developer and can help him in improving the gameplay.
 - Analysis of viewer reaction to advertisements.
 - E-Commerce sites can use this tool to study the user's interest on their website and can improve their services.
 - It has been observed that the mental status of a person has an effect on his/her driving which sometimes cause serious accidents. With the proper use of this system we can help in avoiding such accidents.
- 

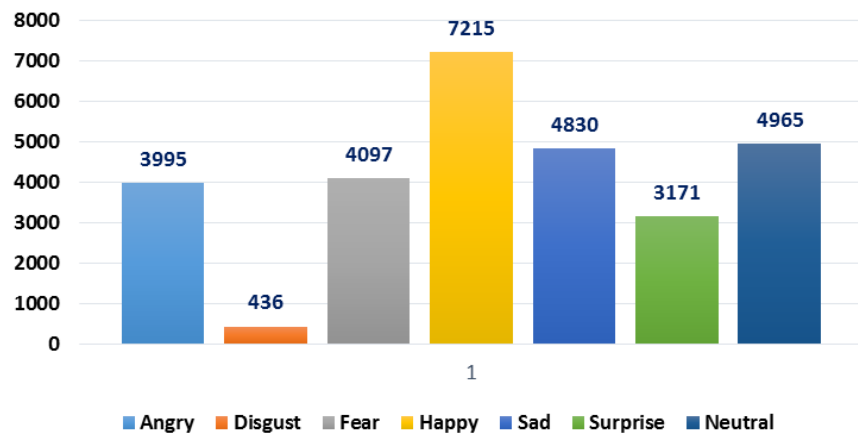
The DataSet

- 32,298 grayscale images
- Each image of 48 x 48 pixel
- 28,709 Training + 3,589 validation
- 7 emotion classes
- Source: FER2013 Dataset

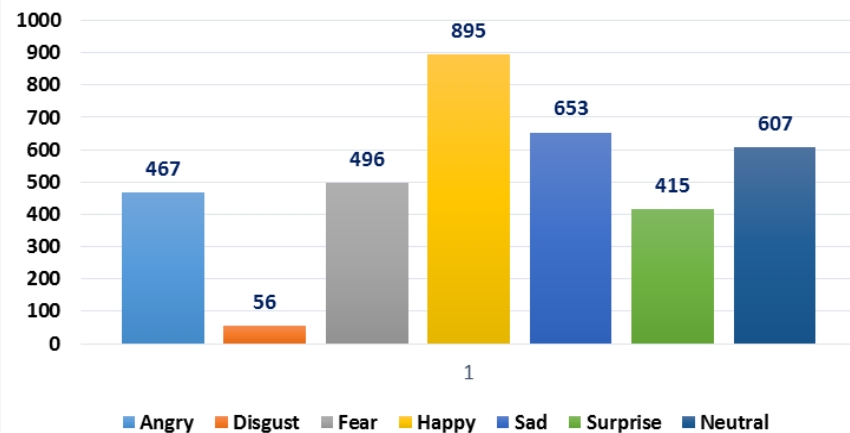
0=Angry
1=Disgust
2=Fear
3=Happy
4=Sad
5=Surprise
6=Neutral.



Training Data

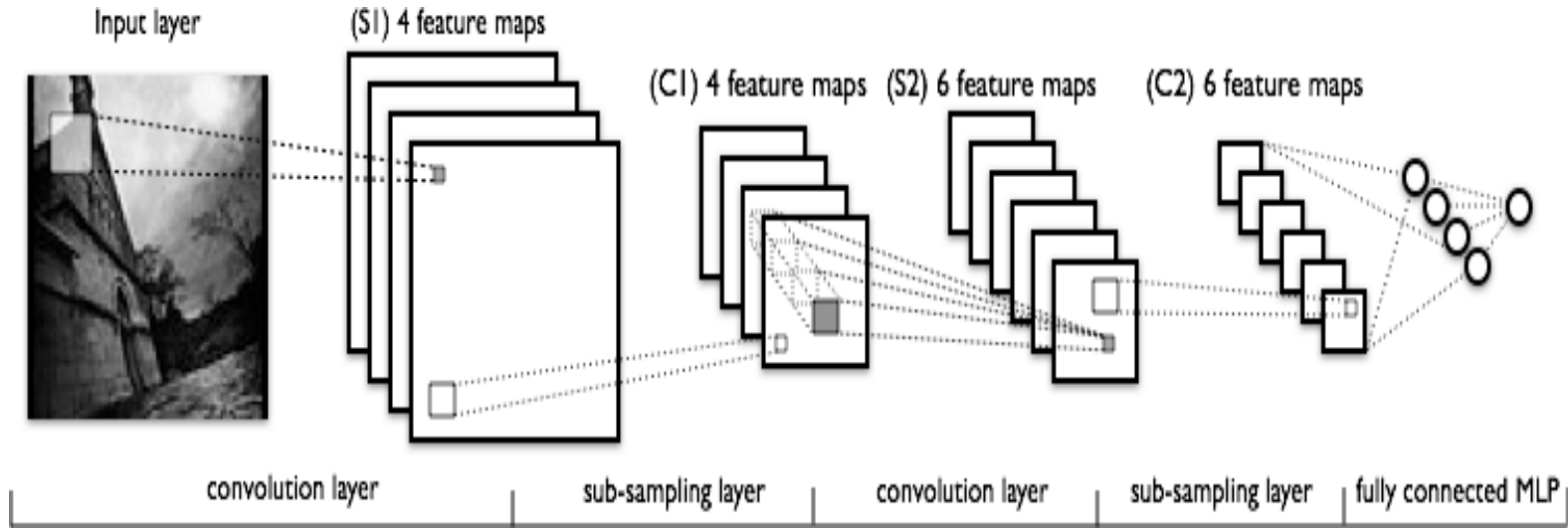


Validation Data



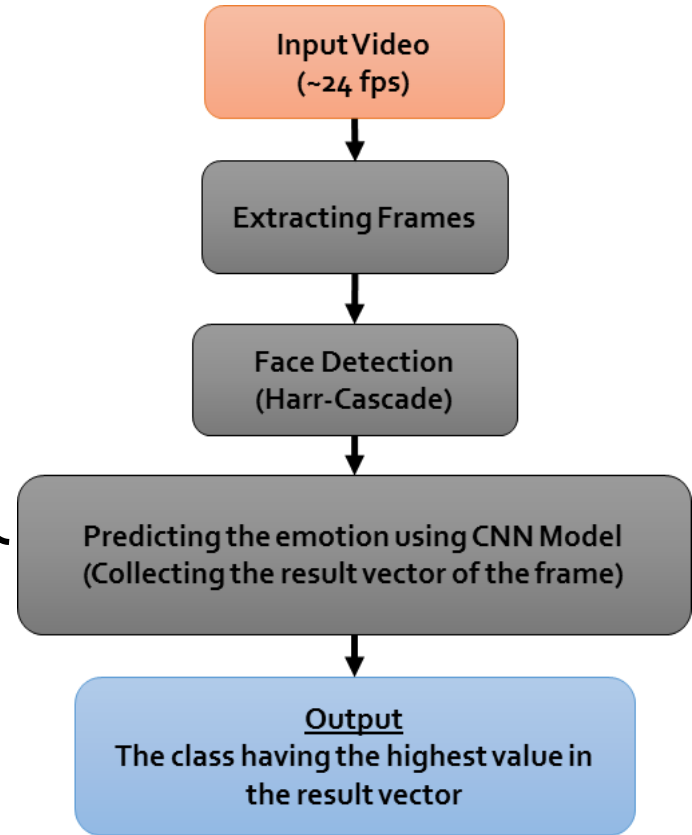
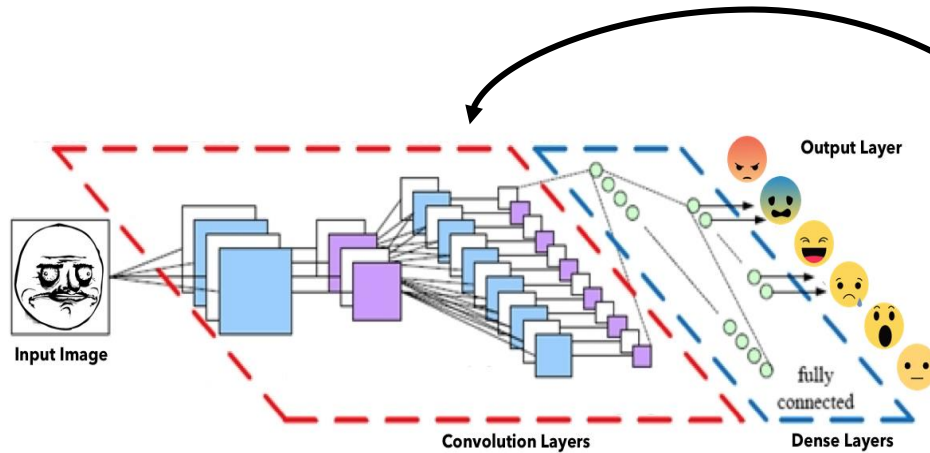
Approach: Deep Learning

- **Convolutional Neural Network(CNN):** CNN take into consideration spatial information in an image
 - **Input Layer** will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B it would have dimensions $32 \times 32 \times 3$.
 - **CONV Layer** will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as $[32 \times 32 \times 12]$ if we decided to use 12 filters.
 - **RELU Layer** will apply an element wise activation function, such as the **max(0,x)** thresholding at zero. This leaves the size of the volume unchanged ($[32 \times 32 \times 12]$) and removes the negative intensities.
 - **POOL layer** will perform a down sampling operation along the spatial dimensions (width, height), resulting in volume such as $[16 \times 16 \times 12]$ if we use a 2×2 pooling filter. Eg. 2×2 max pool filter.
 - **FC Layer**(i.e. fully-connected) layer will compute the class scores, resulting in volume of size $[1 \times 1 \times 10]$, where each of the 10 numbers correspond to a class score, such as among the 10 categories.



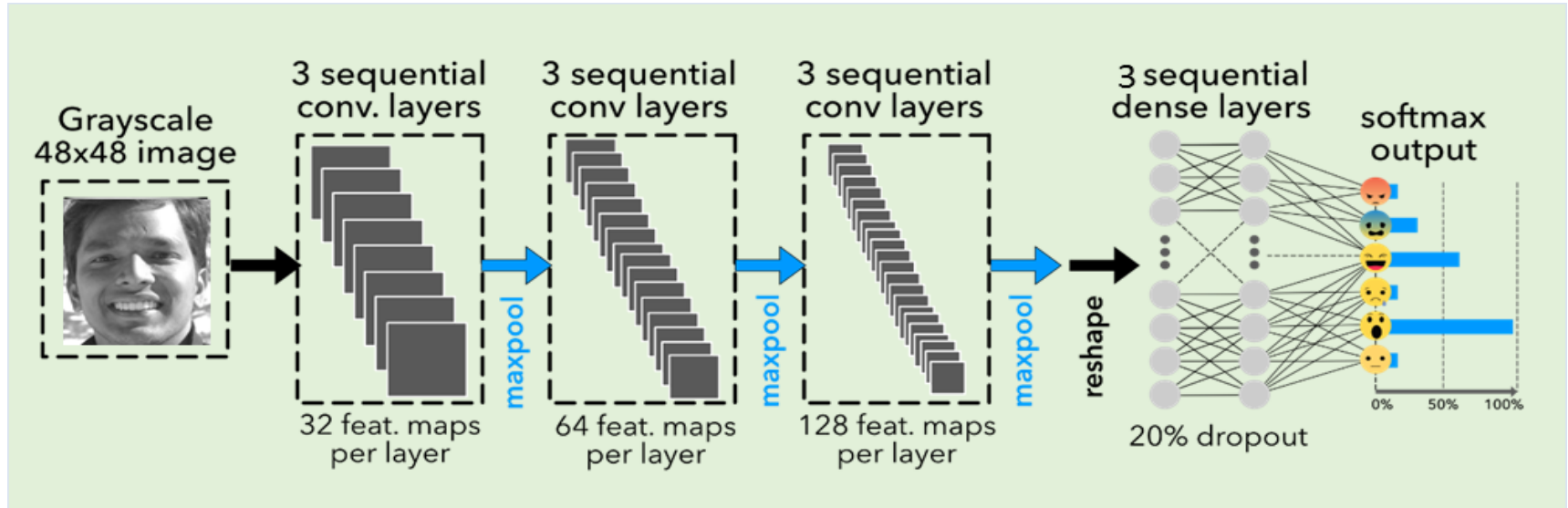
Why CNN??- The basic need of CNN aroused for image recognitions problems as in case of images the no of parameters in input layer become large and in order to make the recognizing system efficient the number of hidden layers in the neural network are also large, due to which the effect to the weights of initial hidden layers is not much during back propagation. This increases the number of iterations needed to adjust the weights in order to obtain good accuracy from the system, thereby increasing the computation power.

Methodology



CNN Model 1

Filter Size: 3x3



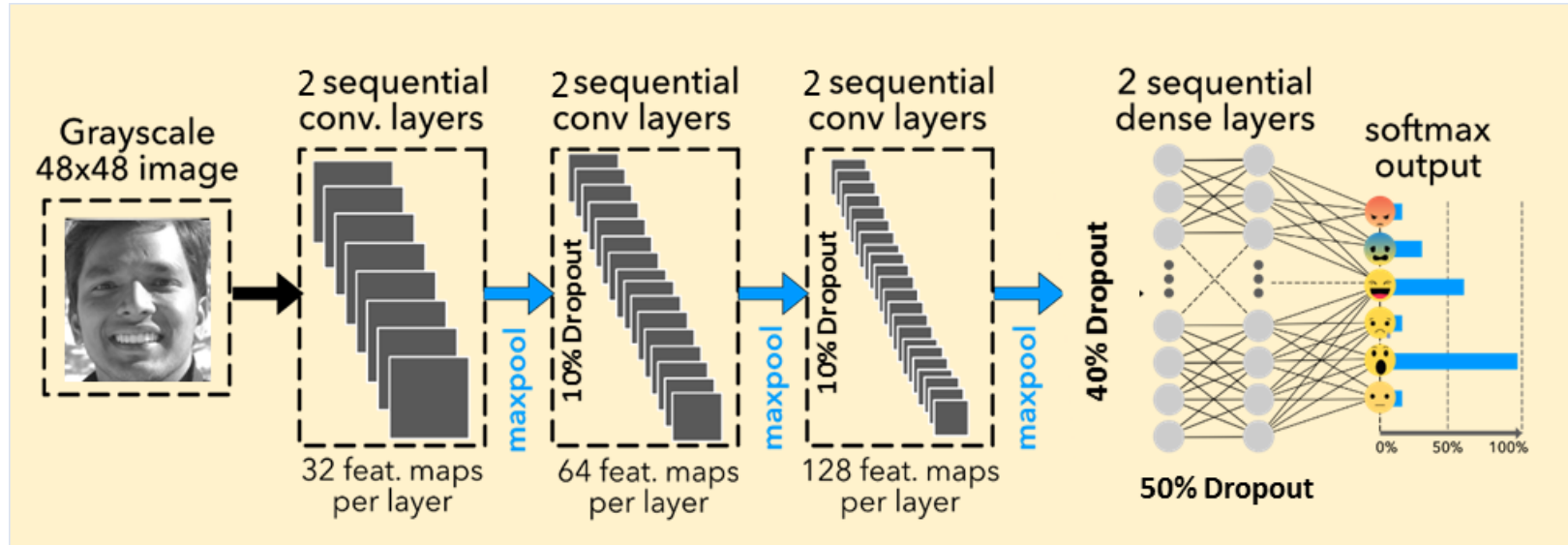
Layer (type)	Output Shape	Param #	Connected to
convolution2d_1 (Convolution2D)	(None, 32, 48, 48)	320	convolution2d_input_1[0][0]
convolution2d_2 (Convolution2D)	(None, 32, 48, 48)	9248	convolution2d_1[0][0]
convolution2d_3 (Convolution2D)	(None, 32, 48, 48)	9248	convolution2d_2[0][0]
maxpooling2d_1 (MaxPooling2D)	(None, 32, 24, 24)	0	convolution2d_3[0][0]
convolution2d_4 (Convolution2D)	(None, 64, 24, 24)	18496	maxpooling2d_1[0][0]
convolution2d_5 (Convolution2D)	(None, 64, 24, 24)	36928	convolution2d_4[0][0]
convolution2d_6 (Convolution2D)	(None, 64, 24, 24)	36928	convolution2d_5[0][0]
maxpooling2d_2 (MaxPooling2D)	(None, 64, 12, 12)	0	convolution2d_6[0][0]
convolution2d_7 (Convolution2D)	(None, 128, 12, 12)	73856	maxpooling2d_2[0][0]
convolution2d_8 (Convolution2D)	(None, 128, 12, 12)	147584	convolution2d_7[0][0]
convolution2d_9 (Convolution2D)	(None, 128, 12, 12)	147584	convolution2d_8[0][0]
maxpooling2d_3 (MaxPooling2D)	(None, 128, 6, 6)	0	convolution2d_9[0][0]
flatten_1 (Flatten)	(None, 4608)	0	maxpooling2d_3[0][0]
dense_1 (Dense)	(None, 64)	294976	flatten_1[0][0]
dense_2 (Dense)	(None, 64)	4160	dense_1[0][0]
dense_3 (Dense)	(None, 7)	455	dense_2[0][0]
Total params: 779783			

	Predicted Label						
	0	1	2	3	4	5	6
0	[215	4	69	25	38	22	94]
1	[24	13	9	3	3	1	3]
2	[66	4	179	16	66	59	106]
3	[49	2	28	618	37	24	137]
4	[85	2	119	40	184	25	198]
5	[17	3	42	20	12	289	32]
6	[55	0	51	44	65	23	369]
Test score: 1.45031974823							
Test accuracy: 0.520200612988							

Confusion Matrix

CNN Model 2

Filter Size: 3x3



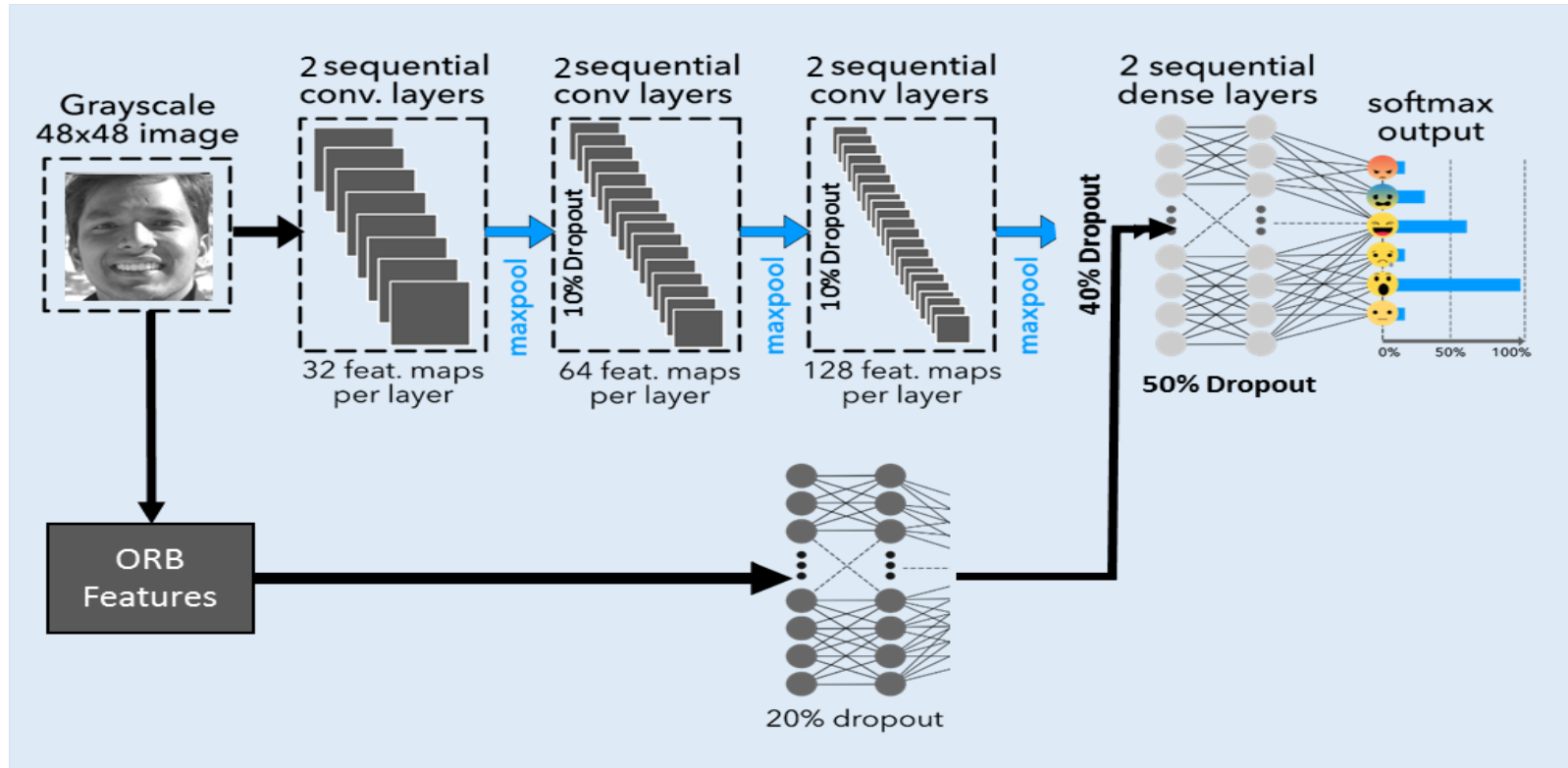
Layer (type)	Output Shape	Param #	Connected to
convolution2d_1 (Convolution2D)	(None, 3, 50, 32)	13856	convolution2d_input_1[0][0]
convolution2d_2 (Convolution2D)	(None, 5, 52, 32)	9248	convolution2d_1[0][0]
maxpooling2d_1 (MaxPooling2D)	(None, 2, 26, 32)	0	convolution2d_2[0][0]
dropout_1 (Dropout)	(None, 2, 26, 32)	0	maxpooling2d_1[0][0]
convolution2d_3 (Convolution2D)	(None, 4, 28, 64)	18496	dropout_1[0][0]
convolution2d_4 (Convolution2D)	(None, 6, 30, 64)	36928	convolution2d_3[0][0]
maxpooling2d_2 (MaxPooling2D)	(None, 3, 15, 64)	0	convolution2d_4[0][0]
dropout_2 (Dropout)	(None, 3, 15, 64)	0	maxpooling2d_2[0][0]
convolution2d_5 (Convolution2D)	(None, 5, 17, 128)	73856	dropout_2[0][0]
convolution2d_6 (Convolution2D)	(None, 7, 19, 128)	147584	convolution2d_5[0][0]
maxpooling2d_3 (MaxPooling2D)	(None, 3, 9, 128)	0	convolution2d_6[0][0]
dropout_3 (Dropout)	(None, 3, 9, 128)	0	maxpooling2d_3[0][0]
flatten_1 (Flatten)	(None, 3456)	0	dropout_3[0][0]
dense_1 (Dense)	(None, 2048)	7079936	flatten_1[0][0]
dropout_4 (Dropout)	(None, 2048)	0	dense_1[0][0]
dense_2 (Dense)	(None, 7)	14343	dropout_4[0][0]
Total params: 7394247			

		<u>Predicted Label</u>						
		0	1	2	3	4	5	6
<u>True Label</u>	0	[[239	3	41	44	70	12	58]
	1	[14	23	6	4	7	0	2]
	2	[32	3	205	30	113	36	77]
	3	[22	1	13	744	30	17	68]
	4	[74	2	55	50	315	8	149]
	5	[11	2	47	23	8	307	17]
	6	[45	1	29	61	99	6	366]]
Test score: 1.08653423576								
Test accuracy: 0.612705489002								

Confusion Matrix

Model 3: CNN + ORB

Filter Size: 3x3

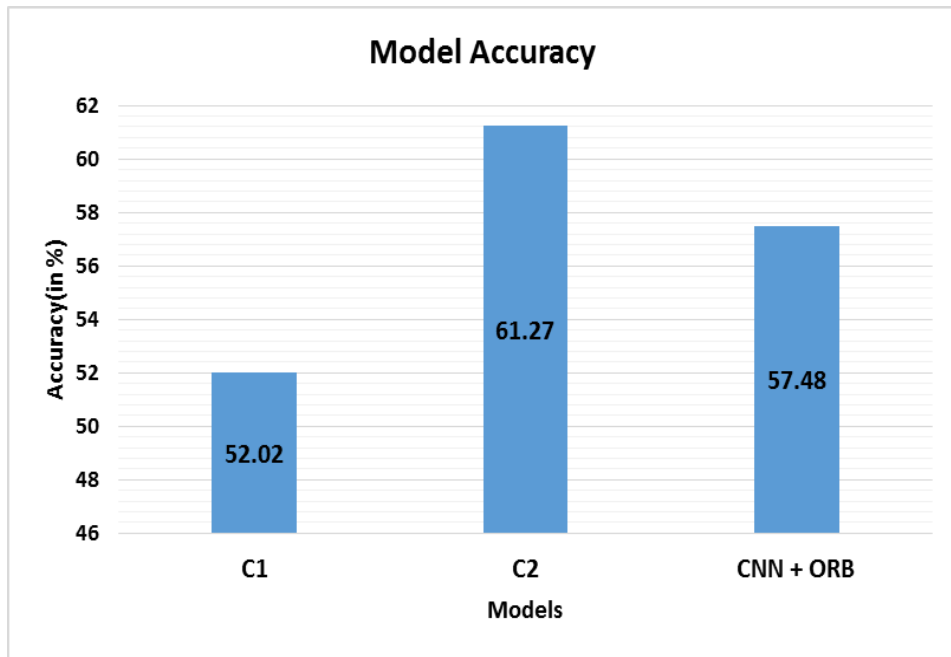


Layer (type)	Output Shape	Param #	Connected to
convolution2d_1 (Convolution2D)	(None, 32, 50, 50)	320	convolution2d_input_1[0][0]
convolution2d_2 (Convolution2D)	(None, 32, 52, 52)	9248	convolution2d_1[0][0]
maxpooling2d_1 (MaxPooling2D)	(None, 32, 26, 26)	0	convolution2d_2[0][0]
dropout_1 (Dropout)	(None, 32, 26, 26)	0	maxpooling2d_1[0][0]
convolution2d_3 (Convolution2D)	(None, 64, 28, 28)	18496	dropout_1[0][0]
convolution2d_4 (Convolution2D)	(None, 64, 30, 30)	36928	convolution2d_3[0][0]
maxpooling2d_2 (MaxPooling2D)	(None, 64, 15, 15)	0	convolution2d_4[0][0]
dropout_2 (Dropout)	(None, 64, 15, 15)	0	maxpooling2d_2[0][0]
convolution2d_5 (Convolution2D)	(None, 128, 17, 17)	73856	dropout_2[0][0]
convolution2d_6 (Convolution2D)	(None, 128, 19, 19)	147584	convolution2d_5[0][0]
maxpooling2d_3 (MaxPooling2D)	(None, 128, 9, 9)	0	convolution2d_6[0][0]
dropout_3 (Dropout)	(None, 128, 9, 9)	0	maxpooling2d_3[0][0]
flatten_1 (Flatten)	(None, 10368)	0	dropout_3[0][0]
dense_1 (Dense)	(None, 4096)	6557696	dense_input_1[0][0]
dropout_4 (Dropout)	(None, 4096)	0	dense_1[0][0]
dense_2 (Dense)	(None, 2048)	29624320	merge_1[0][0]
dropout_5 (Dropout)	(None, 2048)	0	dense_2[0][0]
dense_3 (Dense)	(None, 7)	14343	dropout_5[0][0]
Total params: 36482791			
Training....			

	<u>Predicted Label</u>						
	0	1	2	3	4	5	6
0	[[218	4	54	36	81	24	50]
1	[15	25	2	3	6	1	4]
2	[51	2	187	24	110	60	62]
3	[30	1	28	700	39	29	68]
4	[74	8	73	51	306	17	124]
5	[8	1	41	15	11	330	9]
6	[66	0	45	54	130	15	297]]

Confusion Matrix

Choosing the Best Model



Dashboard

Public Leaderboard - Challenges in Representation Learning: Facial Expression Recognition Challenge

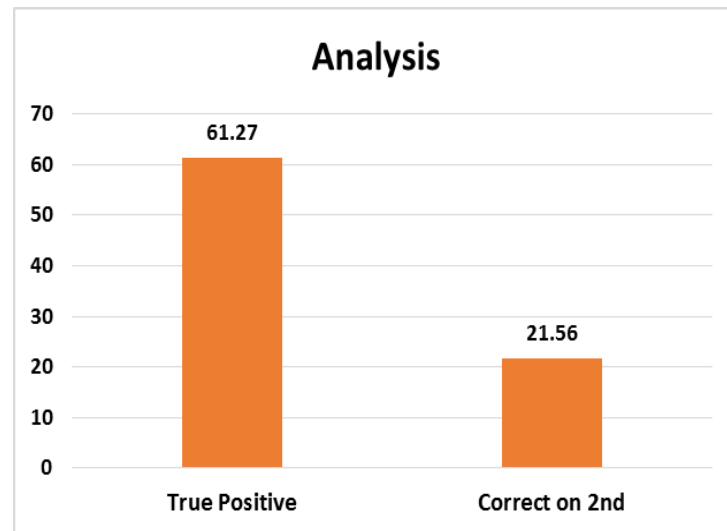
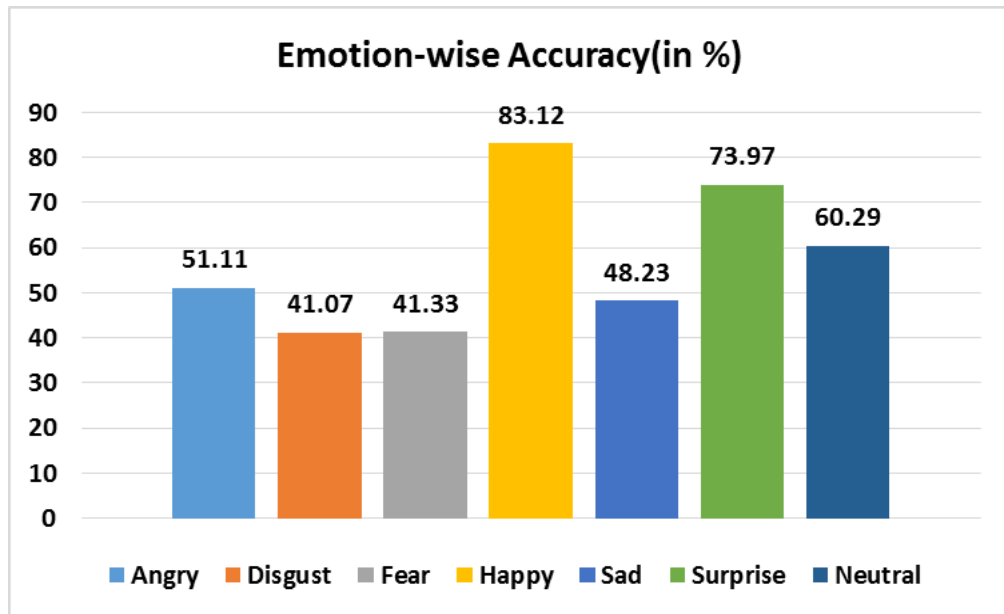
This leaderboard is calculated on approximately 50% of the test data.
The final results will be based on the other 50%, so the final standings may be different.

See someone using multiple accounts?
[Let us know.](#)

#	Δ1w	Team Name <small>• In the money</small>	Score	Entries	Last Submission UTC (Best - Last Submission)
1	new	RBM *	0.69769	5	Thu, 23 May 2013 16:12:26 (-43h)
2	new	Unsupervised	0.69072	8	Fri, 24 May 2013 18:32:35
3	new	Maxim Milakov	0.68153	7	Fri, 24 May 2013 20:07:41
4	new	Radu+Marius+Cristi	0.67317	6	Fri, 24 May 2013 14:35:49 (-2.4d)
5	new	Lor.Voldy	0.64558	2	Tue, 21 May 2013 19:29:59 (-0h)
6	new	Eric Cartman	0.64447	1	Tue, 21 May 2013 19:32:52
7	new	ryank	0.64057	2	Tue, 21 May 2013 21:49:50
8	new	Xavier Bouthillier	0.62775	1	Fri, 24 May 2013 11:23:46
9	new	sayit	0.61911	2	Wed, 22 May 2013 16:52:46 (-14.7h)
10	new	Alejandro Dubrovsky	0.61382	5	Fri, 24 May 2013 23:30:26
11	new	jaberg	0.60797	6	Fri, 24 May 2013 22:09:53 (-30.7h)
12	new	kg	0.59181	4	Thu, 23 May 2013 23:45:22 (-0h)
13	new	bulbuloglu	0.58958	8	Fri, 24 May 2013 12:35:42 (-15.3h)
14	new	Liu	0.58038	8	Fri, 24 May 2013 21:10:48 (-26.2h)

Kaggle Position(8th) with 61.27% accuracy

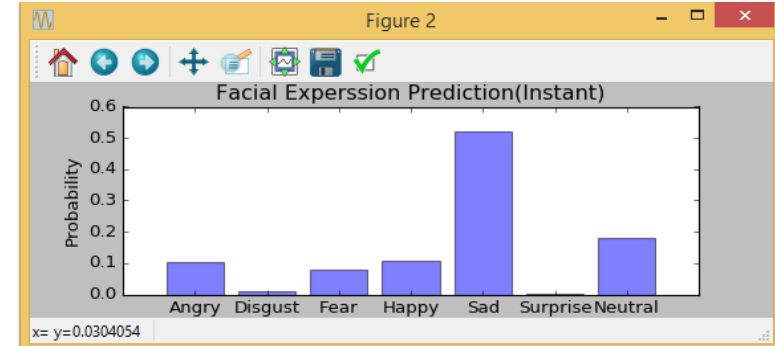
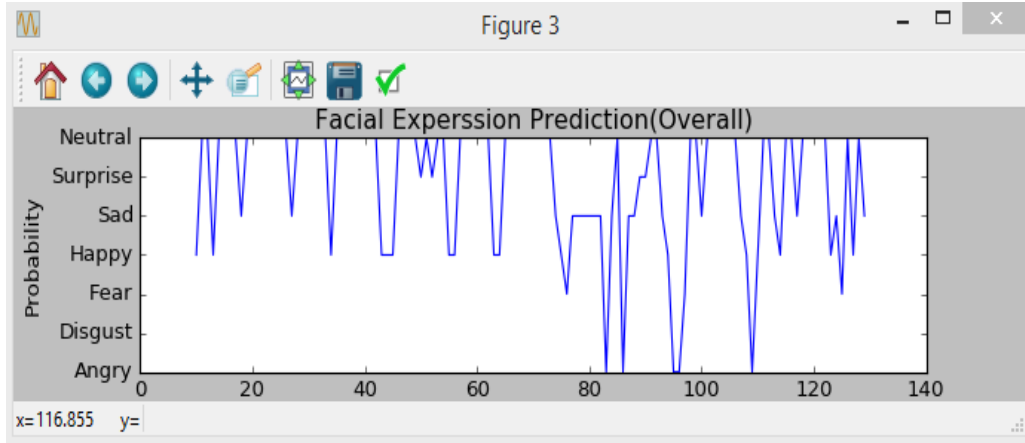
Analysis



On testing the validation data, we analysed that **21.56% misclassified images have second most likely emotion as the correct label.**

Modules Developed

- i. Real Time Webcam Video classification
- ii. Classification of facial expression on Video file
- iii. Classification of facial expression on Image file



```

new_model_1.py - C:\Users\Annu\Documents\Battlefield 4\new_model_1.py (2.7.1...
File Edit Format Run Options Window Help

# model architecture:
model = Sequential()
model.add(Convolution2D(32, 3, 3, border_mode='full', input_shape=(1, 48, 48), a
model.add(Convolution2D(32, 3, 3, border_mode='full', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Convolution2D(64, 3, 3, border_mode='full', activation='relu'))
model.add(Convolution2D(64, 3, 3, border_mode='full', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Convolution2D(128, 3, 3, border_mode='full', activation='relu'))
model.add(Convolution2D(128, 3, 3, border_mode='full', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))

model.add(Flatten())
model.add(Dense(2048, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()

# optimizer:
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accur
model.save('model_221.h5')
#model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accu
print ('Training.....')
model.fit(X_train, Y_train, nb_epoch=nb_epoch, batch_size=batch_size, shuffle=Fa

#####

#model.load_weights('my_model_weights_20_11.h5')
model.save_weights('my_model_221.h5')
score = model.evaluate(X_test, Y_test, show_accuracy=True, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

```

```

*webcam_video_test.py - C:\Users\Annu\Desktop\Sem VII Project\facial_exp\web...
File Edit Format Run Options Window Help
##### LOAD THE MODEL
K.set_image_dim_ordering('th')

model= load_model('model_221.h5')
print ('Model Loaded.....')

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accur
model.load_weights('my_model_221.h5')
print ('Weights Loaded.....')

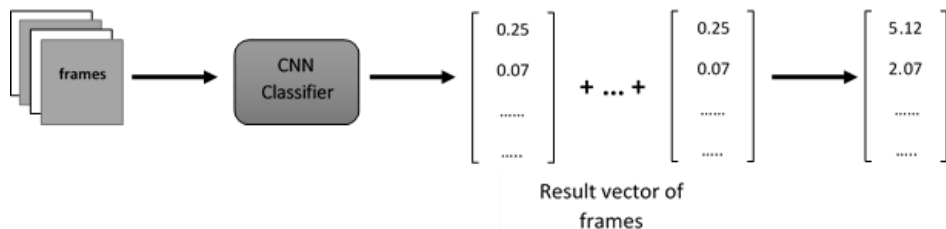
#####WEBCAM DATA

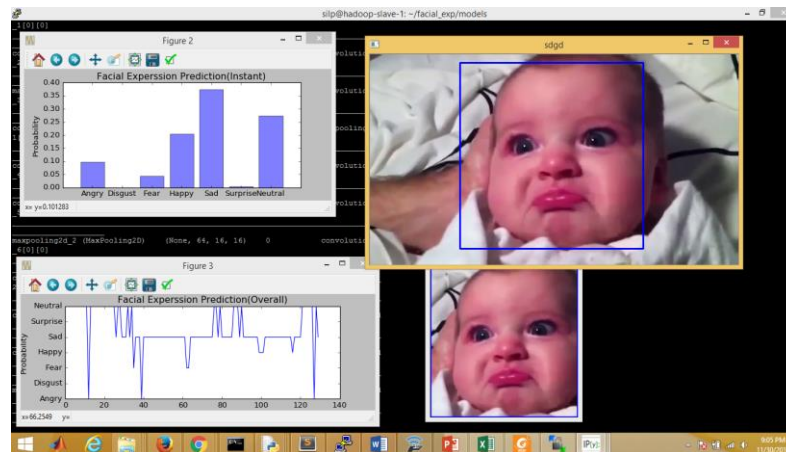
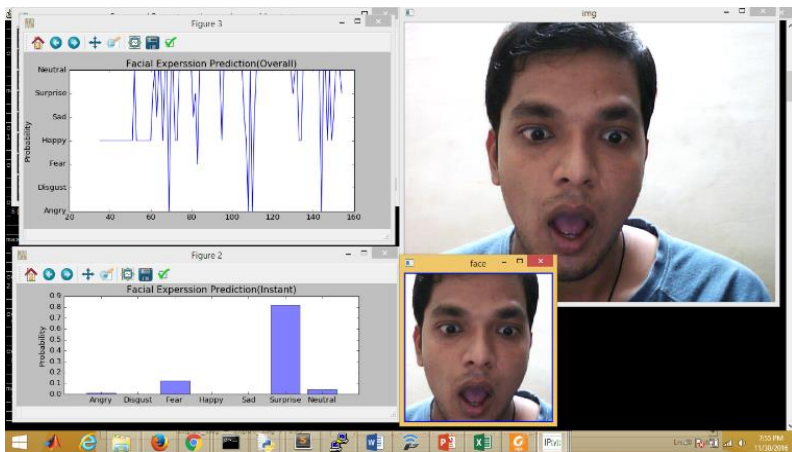
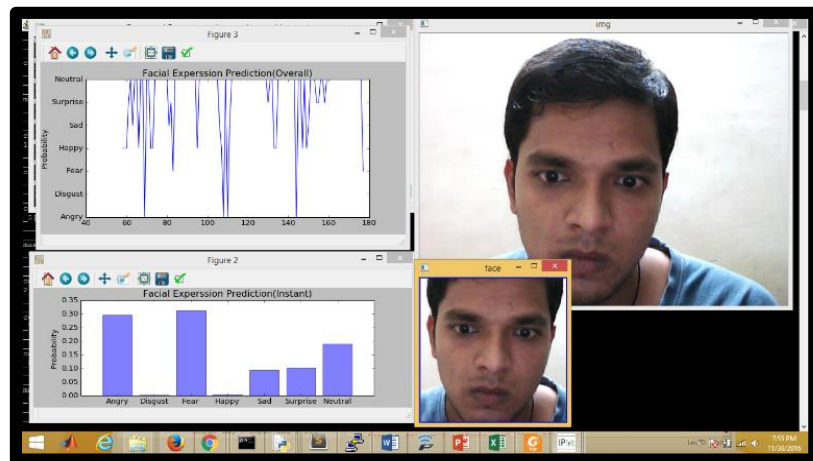
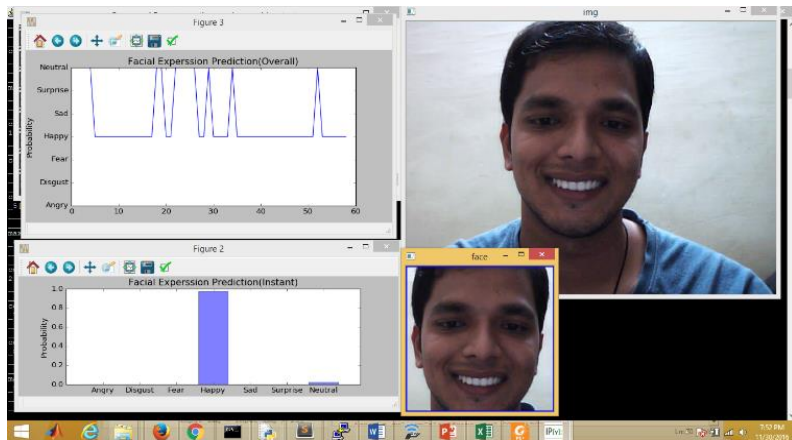
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
ans = []
time = []
t = 0
p = 0
i = 0
c = 0
x_test = np.empty([1,1,48,48])
while(True):
    ret,frame = cap.read()
    #gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = frame;
    cv2.imshow('img',frame)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y), (x+w,y+h), (255,0,0),2)
        i1 = frame[y-p:y+h+p,x-p:x+w+p]
        pts1 = np.float32([[x-p,y-p],[x+w+p,y-p],[x-p,y+p+h],[x+w+p,y+h+p]])
        pts2 = np.float32([[0,0],[255,0],[0,255],[255,255]])

        M = cv2.getPerspectiveTransform(pts1,pts2)
        img = cv2.warpPerspective(frame,M,(256,256))
        input_img = cv2.resize(img, (48,48))


        gray_image = cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)
        input_img = np.array(gray_image, dtype='float32')
        input_img = input_img / 255

```





Technical Requirements

- Nvidia GPU
 - CUDA
 - Python
 - Anaconda
 - Theano
 - Kares
 - PyCharm
- 
- Decorative geometric shapes in the bottom right corner, including a dark teal triangle, a light teal triangle, and a lime green triangle.

The background features a stylized landscape with two mountain peaks. The left peak is dark teal, and the right peak is a lighter green. The foreground is a solid teal color. The text 'Thank You!' is centered in the foreground.

Thank You!

Vaibhav Srivastava (IIT2013027)
Himanshu Tuteja (IIT2013038)
Anirudh Gupta (IIT2013117)