DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING,GUINDY

ANNA UNIVERSITY

CHENNAI 600-025

CS6301 – Machine Learning Laboratory

Analysing Emotions From Facial Expressions

NAME	REG. NO	AADHAR NO	E-MAIL	MOBILE NO
S.K.Risheek Rakshit	2018103580	4241 9184 4020	risheekrakshit7@gmail.com	+1 (515) 770 6717
K.S Sumaiya Fathima	2018103607	9367 4395 4139	sumaiya290301@gmail.com	+91 99411 74390

INSTRUCTOR & MENTOR: **Dr AROCKIA XAVIER ANNIE R**

Contents

1. ABSTRACT

We humans tend to show a lot of emotions in our day to day lives which separates us from other creatures. There is a need for having some effective means to find out the emotions of humans and make use of it in an effective manner. Facial expression recognition is an essential ability for good interpersonal relations and a major subject of study in the fields of human development, psychological well-being, and social adjustment. We tend to have a lot of variations in our facial expressions based on the emotion we want to express.So our model is based on predicting the emotions of humans through their facial expressions,which can be used in a variety of ways according to our needs. This could be used to analyse the change in the emotions undergone by an individual just by looking at their video feed.

We convey a lot of nonverbal information in our faces, and we tend to focus on different areas of the face when we try to interpret what each expression might mean. We look at the eyes to determine if someone is sad or angry, for example, and at the mouth to check if someone is happy.At some cases there may exist an ambiguity in determining the emotions expressed only if we focus only on one feature for example if we try to focus only on the mouth we may get struck with angry or happiness.So it is essential that we focus on the entire face.

2. INTRODUCTION

The primary aim of our project is to predict the right emotions expressed by the human (which is classified into angry, disgust, happy, surprise, sad and scared). Hence our project is a multi-class classification problem. This process involves detecting the face in the input image(either an image or a live input feed) with the help of Haar feature-based cascade classifiers. The face detected from this process is the input on which the prediction is made. For in-depth classification we have decided to use Convolutional Neural Networks(CNNs).

Convolutional Neural Network is an artificial neural network used for image analysis to identify patterns and make sense of them. The convolutional layers are the hidden layers which predict the patterns.The number of filters and the dimension of the filter matrices has to be specified for each convolution layer.

CNN’s are used for image,speech classification and recognition because of its high accuracy. The CNN follows a hierarchical model which works on building a network, like a funnel, and finally gives out a fully-connected layer where all the neurons are connected to each other and the output is processed.

The data points extracted from the image files after preprocessing representing different emotions are fed into the Convolution Neural Network and the model is trained.CNN’s usually require a fairly large number of images to train.As any dataset even the ones consisting of around 50,000 samples is considered less,the approach of data augmentation is used to increase the number of samples.Under this approach the same image is horizontally flipped,rescaled,resized such that a single image can be used in a number of variations to train the CNN model more accurately.In simple words,when the training set is not sufficient enough to learn representation more data is generated using the training set by applying transformations.The image data is generated by transforming the actual training images by rotation, crop, shifts, shear, zoom, flip, reflection, normalization etc.

3. LITERATURE SURVEY

1. Octavio Arriaga ,Matias Valdenegro-Toro, Paul Plöger.“ Real-time Convolutional Neural Networks for Emotion and Gender Classification .”Submitted to ICRA 2018

Features:

- A general convolutional neural network (CNN) building framework for designing real-time CNNs.
- The models are validated by creating a real-time vision system which accomplishes the tasks of face detection, gender classification and emotion classification simultaneously in one blended step using our proposed CNN architecture.
- After presenting the details of the training procedure setup we proceed to evaluate on standard benchmark sets.

- Guided back-propagation uncovers the dynamics of the weight changes and evaluates the learned features. The careful implementation of modern CNN architectures, the use of the current regularization methods and the visualization of previously hidden features are necessary in order to reduce the gap between slow performances and real-time architectures.

Improvement:

We tried and experimented many different architectures along with the one proposed by changing/modifying different layers at each and every module. This resulted in increasing the training and test accuracy considerably and the best model trained was used for predictions.

2. Li, Shan and Deng, Weihong. “Deep Facial Expression Recognition: A Survey “.IEEE Transactions on Affective Computing, Institute of Electrical and Electronics Engineers (IEEE) 10.,1109/taffc.2020.2981446, 2020

Features:

- Introduces a few more available datasets that are widely used in the literature and provide accepted data selection and evaluation principles for these datasets.
- It introduces an architecture to pipeline the deepFER and suggestions of applicable implications.
- For the state of the art in deep FER, we review existing novel deep neural networks and related training strategies that are designed for FER based on both static images and dynamic image sequences, and discuss their advantages and limitations.

Improvement:

We used the architecture given in this paper and trained it.We also used transfer learning techniques along with this architecture compared them and also performed ensembling techniques to enhance the performance of the model further.

3. Amil Khanzada ,Charles Bai,Ferhat Turker Celepcikay. "Facial Expression Recognition with Deep Learning. ",2020

Features:

- This paper takes a deep dive, implementing multiple deep learning models for facial expression recognition (FER).
- The two main goals are twofold: To aim not only to maximize accuracy, but also to apply our results to the real-world.
- By leveraging numerous techniques from recent research, we demonstrate a state-of-the-art 75.8% accuracy on the FER2013 test set, outperforming all existing publications.
- Additionally, we showcase a mobile web app which runs our FER models on-device in real time.

4.PROBLEM STATEMENT

To develop a system which recognizes the facial expressions of human faces using deep CNN model and haar cascade classifier and output the emotion label which has the maximum probability out of the seven emotions considered.

Given an image/live video feed as an input expressing any one of the seven emotions specified we are to find out the probability of each emotion and output the label of the emotion with maximum probability. For example, consider the image given below :

As we can see the expression expressed by the person in the image is 'happiness' . Since the emotion expressed is unambiguous(does not express a mix of two or more emotions) the system has to recognize the expression clearly by extracting the face and predict the emotion label 'happy' with the help of the trained model.

5. PROBLEM SOLUTION

5.1. Data Set

The dataset used for our project is fer2013 dataset. The data consists of 48×48 pixel gray scale images of faces. It has been used because it provides a variety of facial data portraying various emotions of both genders. The training set consists of 35,888 examples and it has 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(numeric code given below). The pixels column contains a string surrounded in quotes for each image. The contents of this string are space-separated pixel values in row major order.

The aim is to classify each face based on the emotion expressed in the facial expression into one of seven categories

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

This dataset has a human level accuracy of 65±5%.This is due to the fact that it consists of few confounding(confusing) images, few cartoons and few images wrongly labelled in the dataset which can be quite capricious.

5.2 Input

The face of the person expressing one of the seven emotions is the input to the system

5.3 Approach

The trained model file(in .hdf5 format) for the fer2013 dataset is used to make predictions of the expression in real time video (this is the best model trained out of all the models with variations in architecture). The face of the person who is expressing the emotion is extracted. This is done using a haar feature-based cascade classifier. It detects frontal face at each and every point while capturing the real time video more accurately.

The trained emotion predicting model is loaded into 'emotion model path' and a haar-cascade based classifier is loaded into 'detection model path' which identifies the face of the person.

The face of the person is captured from the live video at each point of time by the detectMultiScale function. This function will return a rectangle with coordinates(x,y,w,h) around the detected face. The extracted features of the face are preprocessed/scaled and fed into the emotion classifier model. The emotion label with maximum probability is returned as result and it is displayed along with the rectangle which captures the face.

6. NOVELTY

Achieving better accuracy on fer2013 dataset has always been an arduous task due to the various anomalies present in the dataset.Anything greater than 60 percent on this dataset is considered a better model as the current highest one stands at around 75%.The base paper we refer has an accuracy of 66% on the validation set. We tried various mutations on the base architecture proposed and we were able to achieve results around the same few of which showed higher validation accuracy as well as in the training set but didn't perform that well when tested with the test dataset. We have compared this model with few other models which use transfer learning techniques such as VGG16,ResNet50 and

SeNet50 with ensembling performed on them,which yielded cogent accuracy levels.

7. ARCHITECTURE

7.1. ABSTRACT ARCHITECTURE

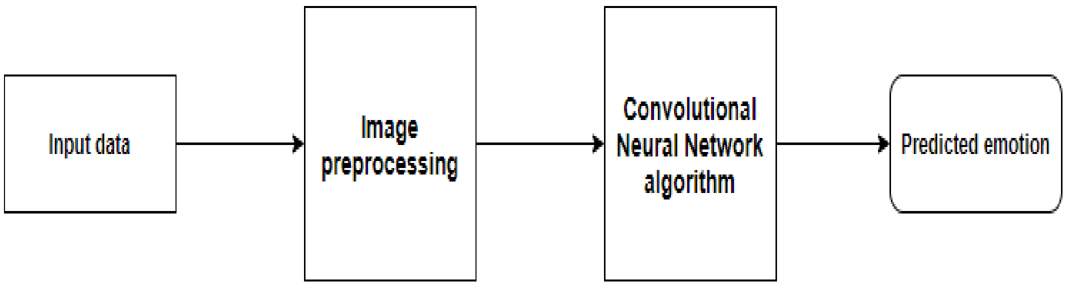


Figure 1

.

This is an abstract view of our project. Given the input data which can either be an image or a frame from the live video feed, we first preprocess the way it matches the input for the model, run the classification process and then use softmax activation function to predict the emotion.

7.2 DETAILED ARCHITECTURE

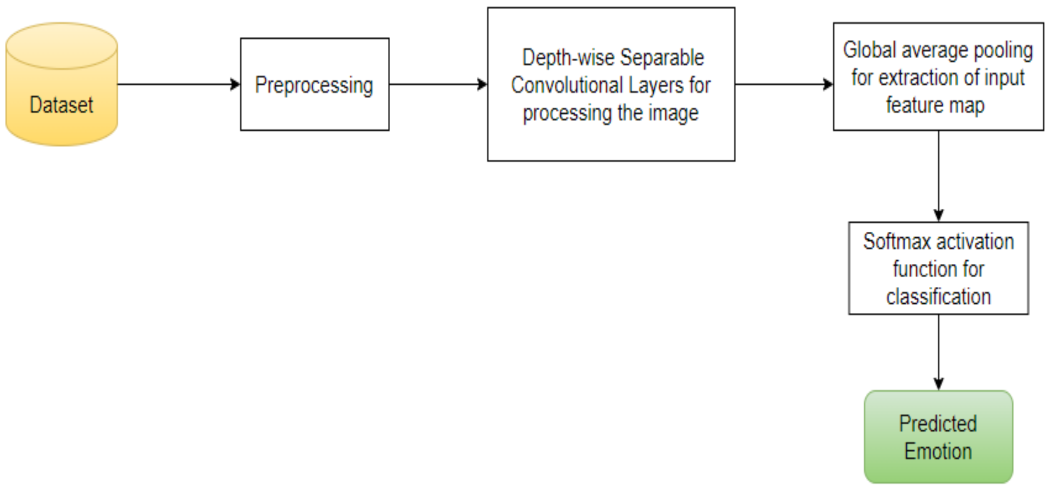

Dataset
Preprocessing
Depth-wise Separable Convolutional Layers for processing the image
Global average pooling for extraction of input feature map
Softmax activation function for classification
Predicted Emotion

Figure 2

7.3 EXPLANATION OF THE ARCHITECTURE

1. <u>Preprocessing data</u>:

Input: gray scaled image from fer-2013 dataset

It is a standard way to pre-process images by scaling them between -1 to 1. Images are scaled to [0,1] by dividing it by 255. This preprocessing process provides a better range for neural network models in computer vision problems.It involves transforming the pixels given in the dataset into an array format.

Output: Preprocessed data after scaling

2. <u>Depth wise separable convolutional layers for processing the image</u>

Input:Preprocessed data

A depthwise separable convolution splits a kernel into 2 separate kernels that do two convolutions: the depthwise convolution and the pointwise convolution.It works well with kernels that cannot be factored into two smaller kernels.The output feature map obtained from one layer is processed by the next layer and subsequently the feature map from the last layer is fed as input to the next module.The CNN model learns the representation features of emotions from the training images.

Output:Feature Map from the last convolutional layer

3. <u>Global average pooling for extraction of input feature map</u>

Input: The output feature of previous module

It reduces each feature map into a scalar value by taking the average over all elements in the feature map. The average operation forces the network to extract global features from the input image.

Output: One feature map for each corresponding emotions

4. __Softmax activation function for classification__

Input: feature map corresponding to the emotions

Softmax activation function normalizes the output of the network to a probability distribution over the required 7 output classes.For a particular input,if a particular class has the highest probability based on this activation,it is considered as the output.

Output: The expressed emotion corresponding to the frontal face is displayed

8. DETAILED MODULE DESIGN

8.1. *Preprocessing Data*:

The dataset consists of three usage types Training,Public Test,Private Test.It is split into three separate csv based on the usage category. The dataset is loaded for the preprocessing to take place. A function is defined to read the fer2013.csv file and convert pixel sequence of each row in image of dimension 48*48. It returns faces and emotion labels. The pixel values of the image are then pre-processed by scaling them between -1 to 1. Images are scaled to [0,1] by dividing it by 255. Further, subtraction by 0.5 and multiplication by 2 changes the range to [-1,1].

Algorithm:

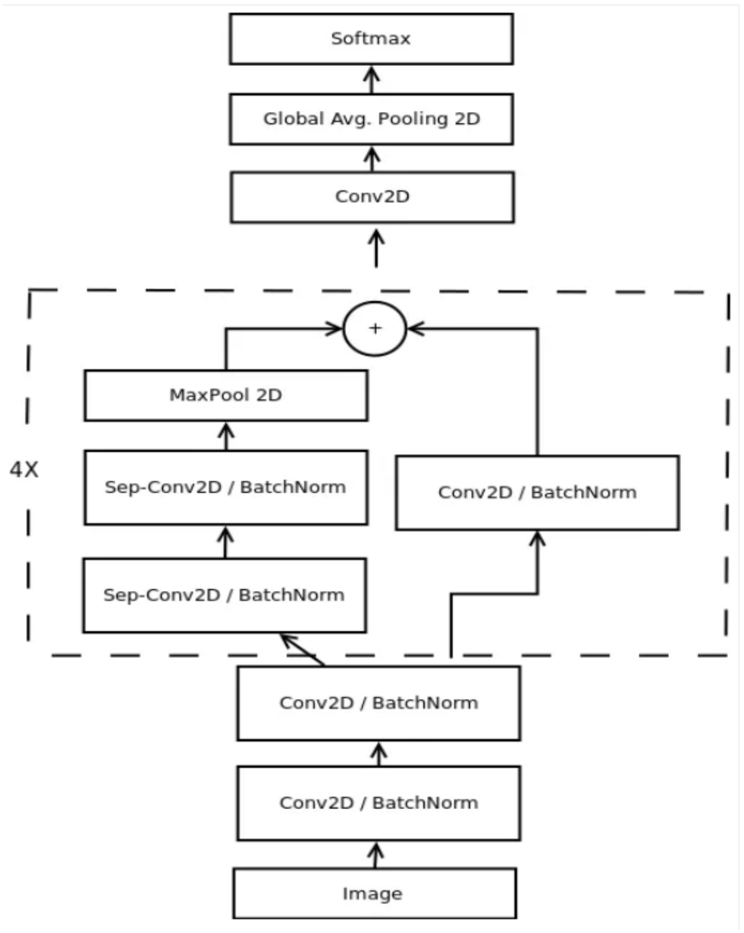
```
Preprocess_input (dataset):
    Extract the pixel column
    Store the contents of the column emotion
    Extract pixel_values from string
    Store the pixel_values as a 48*48 array
    Convert it to float
    Adjust the pixel_values to range [-1,1]
    Return (pixel_values , emotion)
```

8.2. *Model Training*

The following techniques are implemented in training our model.

- **Data Augmentation**: When the training set is not sufficient enough to learn representation more data is generated using the training set by applying transformations.The image data is generated by transforming the actual training images by rotation, crop, shifts, shear, zoom, flip, reflection, normalization etc.
- **Conv2D:** The most common type of convolution that is used is the 2D convolution layer, and is usually abbreviated as conv2D. A filter or a kernel in a conv2D layer has a height and a width. They are generally smaller than the input image and so we move them across the whole image.
- **BatchNormalization**: It normalizes the activation of the previous layer at each batch by applying a transformation which maintains the mean activation close to 0 and the activation standard deviation close to 1. It also acts as a regularizer and helps in speeding up the training process.
- **SeparableConv2D:** Separable convolutions perform a depthwise spatial convolution (which acts on each input channel separately) followed by a pointwise convolution which mixes the resulting output channels.
- **Activation('relu'):** The ReLU layer applies the function f(x) = max(0, x) to all of the values in the input volume. This layer just changes all the negative activations to 0 and increases the nonlinear properties of the model and the overall network without affecting the receptive fields of the conv layer.
- **MaxPooling2D:** Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. The output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.
- **Global Average Pooling:** Global Average Pooling is a pooling operation designed to replace fully connected layers in classical CNNs. The idea is to generate one feature map for each corresponding category of the classification task in the last mlpconv layer.

System Architecture:



Initially, the model's architecture is built using a base layer and 4 intermediate layers.In the base layer, the image is passed to 2D Convolution layer with a number of filters each of the same dimensions and the strides is set to one.The output of the layer is normalized through BatchNormalization and the activation function used is Rectified Linear Unit(relu).This process is again implemented on the output obtained.

The outputs of this base layer are now used in two ways.

- One copy of the output obtained Is again passed through a Convolutional 2D layer with varied filter size.

- Another copy is used as an input for a process nearly the same as the base layer except for the fact that Separable Convolutional layers are used. Both these layers have the same number of filters and filters of the same dimension.**Max-pooling** is used on the output of this process.

- The outputs of both these processes are then combined.

The above mentioned process is performed for variable number of times, and the output is passed to a 2D convolutional layer with 7 filers(The number of output classes in our problem is 7) each of dimension 3*3.Then **Global-Average Pooling** is performed on these different layers obtained,which yields 7 individual values(1 from each filter).These 7 values are used for the process of Softmax Activation to determine the class to which the input belongs.

Algorithm:

```
Model_builder (parameters):
        Build the architecture
        Define parameters for early-stopping and validation
        Compile the model
        Fit the model for the dataset
```

8.3. ***Predicting Expressions on Image***

After the training phase, the trained model file(in .hdf5 format) is used to make predictions of the expression on images. This is done using a haar feature-based cascade classifier. It detects the frontal face in an image in both static and real-time accurately.

As a first step, the path of the image in the local system is specified. The trained emotion predicting model is loaded into ‘emotion_model_path’. Once the specified image is read by cv2 the face of the person is identified through the haar cascade classifier which is loaded through opencv. The detectMultiScale function is used to detect the faces. This function will return a rectangle with coordinates(x,y,w,h) around the detected face. The extracted features of the face are processed and fed into the emotion classifier model. The emotion label with maximum probability is returned as result.

Algorithm:

```
Predict (Image):
    Load the model
    Detect the face
    Preprocess the face section
    Predict using the model
    Return the prediction
```

8.4. *Predicting Expressions on Live video*

The trained emotion predicting model is loaded into 'emotion_model_path' and a haar-cascade based classifier is loaded into 'detection_model_path' which identifies the face of the person. The face of the person is captured from the live video at each point of time by the detectMultiScale function. This function will return a rectangle with coordinates(x,y,w,h) around the detected face. The extracted features of the face are processed and fed into the emotion classifier model. The emotion label with maximum probability is returned as result and it is displayed along with the rectangle which captures the face.

Algorithm:

```
Predict_on_video (Image):
    Load the model
    Access the camera
    Loop over for each frame:
        Detect the face
        Preprocess the face section
        Predict using the model
        Output the prediction
        if(destroy command)Destroy the frame and exit
    Terminate the process
```

9. IMPLEMENTATION

9.1. Initial setup

- Install all the necessary packages - keras,tensorflow etc
- Enabling GPU by creating a virtual environment in the local system.
- Upload the dataset in kaggle account as well as in google drive as Google colaboratory was used on in the later stages.

9.2. Tools

- Jupyter Notebook
- Kaggle
- Google Colab

9.3. Code Snippets

9.3.1. Phase-1: Data Preprocessing

```
dataset_path = "fer2013.csv"

def load_fer2013():
    data = pd.read_csv(dataset_path)
    pixels = data['pixels'].tolist()
    width, height = 48, 48
    faces = []
    for pixel_sequence in pixels:
        face = [int(pixel) for pixel in pixel_sequence.split(' ')]
        face = np.asarray(face).reshape(width, height)
        face = cv2.resize(face.astype('uint8'),image_size)
        faces.append(face.astype('float32'))
    faces = np.asarray(faces)
    faces = np.expand_dims(faces, -1)
    emotions = pd.get_dummies(data['emotion'])
    #.as_matrix()
    #print(emotions)
    return faces, emotions
```

```
def preprocess_input(x, v2=True):
    x = x.astype('float32')
    x = x / 255.0
    if v2:
        x = x - 0.5
        x = x * 2.0
        return x
faces,emotions = load_fer2013()

#preprocessing the data
#to be within the range -1 to +1
faces = preprocess_input(faces)
```

9.3.2. Phase-2: Model Training

```
data_generator = ImageDataGenerator(
                        featurewise_center=False,
                        featurewise_std_normalization=False,
                        rotation_range=10,
                        width_shift_range=0.1,
                        height_shift_range=0.1,
                        zoom_range=.1,
                        horizontal_flip=True)

# base
img_input = Input(input_shape)
x = Conv2D(10, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
use_bias=False)(img_input)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = Conv2D(10, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
print(x)
```

```
# module 5
residual = Conv2D(256, (1, 1), strides=(2, 2),padding='same',
use_bias=False)(x)
residual = BatchNormalization()(residual)
residual2 = Conv2D(256, (1, 1), strides=(2, 2),padding='same',
use_bias=False)(x)
residual2 = BatchNormalization()(residual2)
x = SeparableConv2D(256, (3, 3),
padding='same',kernel_regularizer=regularization,use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(256, (3, 3),
padding='same',kernel_regularizer=regularization,use_bias=False)(x)
x = BatchNormalization()(x)
x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])
x = layers.add([x, residual2])
x = Conv2D(num_classes, (3, 3), padding='same')(x)
x = GlobalAveragePooling2D()(x)
output = Activation('softmax',name='predictions')(x)


faces, emotions = load_fer2013()
faces = preprocess_input(faces)
xtrain, xtest,ytrain,ytest = train_test_split(faces,
emotions,test_size=0.2,shuffle=True)
model.fit(data_generator.flow(xtrain, ytrain,batch_size),
                  steps_per_epoch=len(xtrain) / batch_size,
                  epochs=num_epochs, verbose=1, callbacks=callbacks,
                  validation_data=(xtest,ytest),shuffle = True)
```

9.3.3. Phase-3: Predicting Expressions on Image

```
orig_frame = cv2.imread(img_path)
frame = cv2.imread(img_path,0)
```

```
faces =
face_detection.detectMultiScale(frame,scaleFactor=1.1,minNeighbors=5,minSi
ze=(30,30),flags=cv2.CASCADE_SCALE_IMAGE)
if len(faces) > 0:
   faces = sorted(faces, reverse=True,key=lambda x: (x[2] - x[0]) * (x[3] -
x[1]))[0]
   print(faces)
   (fX, fY, fW, fH) = faces
   roi = frame[fY:fY + fH, fX:fX + fW]
   print(roi)
   roi = cv2.resize(roi, (48, 48))
   print(roi)
   roi = roi.astype("float") / 255.0
   roi = img_to_array(roi)
   roi = np.expand_dims(roi, axis=0)
   preds = emotion_classifier.predict(roi)[0]
   emotion_probability = np.max(preds)
   label = EMOTIONS[preds.argmax()]
   cv2.putText(orig_frame, label, (fX, fY - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.99, (0, 0, 255), 2)
   cv2.rectangle(orig_frame, (fX, fY), (fX + fW, fY + fH),(0, 0, 255), 2)

cv2.imshow('test_face', orig_frame)
cv2.imwrite('test_output/'+img_path.split('/')[-1],orig_frame)
```

9.3.4. Phase-4: Predicting Expressions on Live video

```
cv2.namedWindow('your_face')
camera = cv2.VideoCapture(0)
while True:
   frame = camera.read()[1]
   #reading the frame
   frame = imutils.resize(frame,width=400)
   gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
  faces = 
face_detection.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=5,minSiz
e=(30,30),flags=cv2.CASCADE_SCALE_IMAGE)

  canvas = np.zeros((1250, 3000, 3), dtype="uint8")
  frameClone = frame.copy()
  if len(faces) > 0:
    faces = sorted(faces, reverse=True,
    key=lambda x: (x[2] - x[0]) * (x[3] - x[1]))[0]
    (fX, fY, fW, fH) = faces
      roi = gray[fY:fY + fH, fX:fX + fW]
    roi = cv2.resize(roi, (48, 48))
    roi = roi.astype("float") / 255.0
    roi = img_to_array(roi)
    roi = np.expand_dims(roi, axis=0)
   preds = emotion_classifier.predict(roi)[0]
    emotion_probability = np.max(preds)
    label = EMOTIONS[preds.argmax()]


  for (i, (emotion, prob)) in enumerate(zip(EMOTIONS, preds)):
          # construct the label text
          text = "{}: {:.2f}%".format(emotion, prob * 100)
          w = int(prob * 300)
          cv2.rectangle(canvas, (7, (i * 35) + 5),
          (w, (i * 35) + 35), (0, 0, 255), -1)
          cv2.putText(canvas, text, (10, (i * 35) + 23),
          cv2.FONT_HERSHEY_SIMPLEX, 0.45,
          (255, 255, 255), 2)
          cv2.putText(frameClone, label, (fX, fY - 10),
          cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
          cv2.rectangle(frameClone, (fX, fY), (fX + fW, fY + fH),
                    (0, 0, 255), 2)

  cv2.imshow('your_face', frameClone)
  cv2.imshow("Probabilities", canvas)
```

10. SUPPORT INFORMATION

https://serokell.io/blog/machine-learning-testing

https://keras.io/api/models/model_training_apis/

https://github.com/opencv/opencv/tree/master/data/haarcascades

https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

11. SURVEYED CONTENT

https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

https://www.pyimagesearch.com/2018/06/04/keras-multiple-outputs-and-multiple-losses/

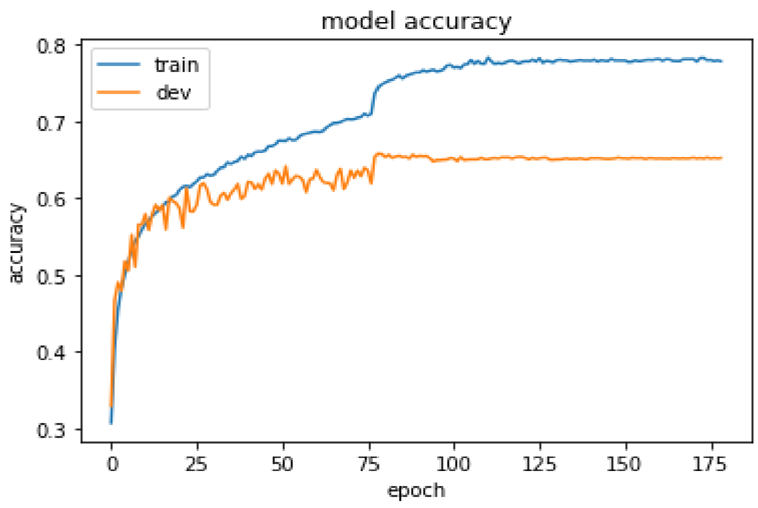
https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/

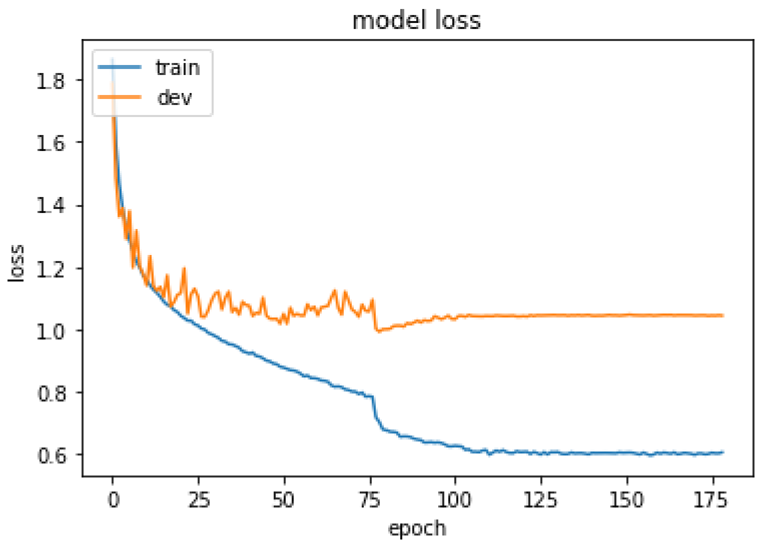
https://www.youtube.com/watch?v=YRhxdVk_sIs

https://www.youtube.com/watch?v=qPKsVAI_W6M

12. RESULTS AND COMPARISON

We tried various architectures apart from the one proposed in the base model with various parameters.We used the concept of Early-Stopping as the means of preventing the model from overfitting the dataset. Initially the early stopping was based on validation set's loss i.e when the patience value was staged at 50,when the model's validation set's loss did not decrease for about 12 iterations,the learning rate of the model was reduced and after 50 iterations of no improvement the training was stopped. Later when it was based on the validation set's accuracy the model trained for more epochs and it resulted in slightly better accuracy. The fluctuations in the training set's accuracy,loss and the same for the Validation set is shown in the below graphs.





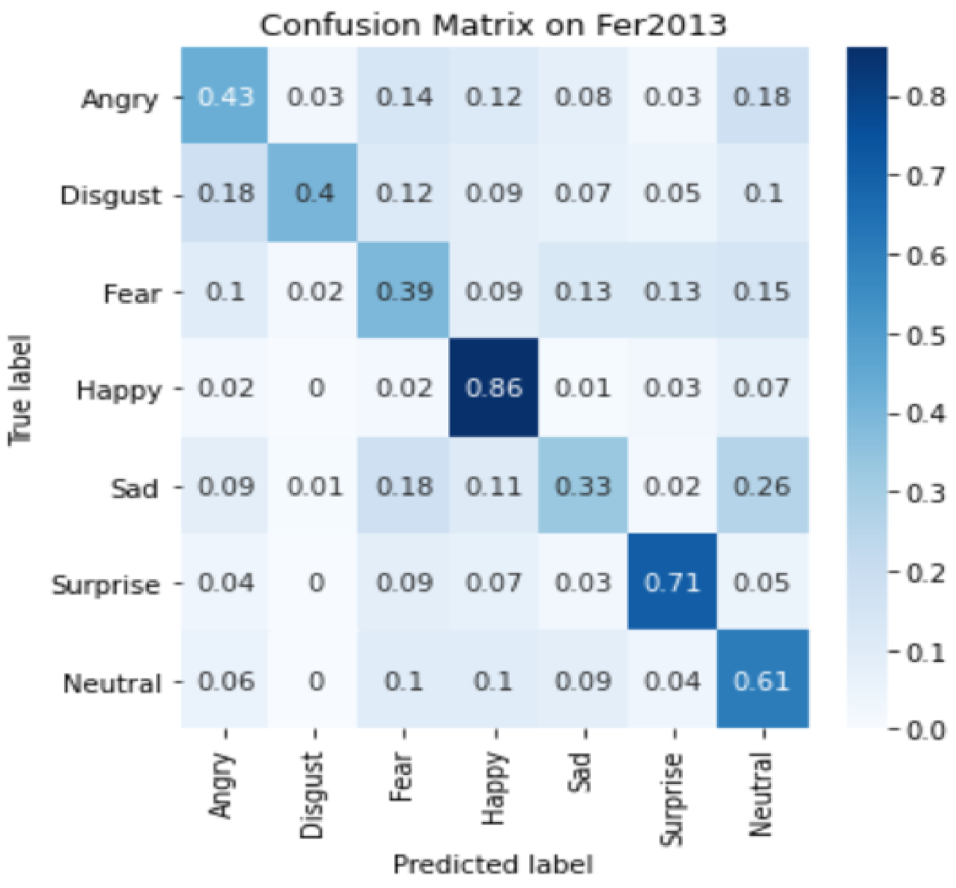
Our final model was better in terms of Validation set's accuracy resulting in 66.16%, the overall classification on the fer2013 dataset yielded the following results.

```
              precision    recall  f1-score   support

     class 0       0.54      0.43      0.48      4953
     class 1       0.38      0.40      0.39       547
     class 2       0.40      0.39      0.40      5121
     class 3       0.74      0.86      0.80      8989
     class 4       0.53      0.33      0.41      6077
     class 5       0.66      0.71      0.69      4002
     class 6       0.48      0.61      0.54      6198

    accuracy                           0.58     35887
   macro avg       0.53      0.53      0.53     35887
weighted avg       0.57      0.58      0.57     35887
```

The confusion matrix for the entire fer2013 dataset:



For comparison, we decided to run ensembling techniques with few transfer learning techniques along with another similar conv2d based approach which will be described in the sections below. This would considerably increase the model's performance in terms of accuracy.

Deep learning neural networks are nonlinear methods.They offer increased flexibility and can scale in proportion to the amount of training data available. A downside of this flexibility is that they learn via a stochastic training algorithm which means that they are sensitive to the specifics of the training data and may find a different set of weights each time they are trained, which in turn produce different predictions.
Generally, this is referred to as neural networks having a high variance and it can be frustrating when trying to develop a final model to use for making predictions.

A successful approach to reducing the variance of neural network models is to train multiple models instead of a single model and to combine the predictions from these models. This is called ensemble learning and not only reduces the variance of predictions but also can result in predictions that are better than any single model.

MODELS USED:

For this approach we use:

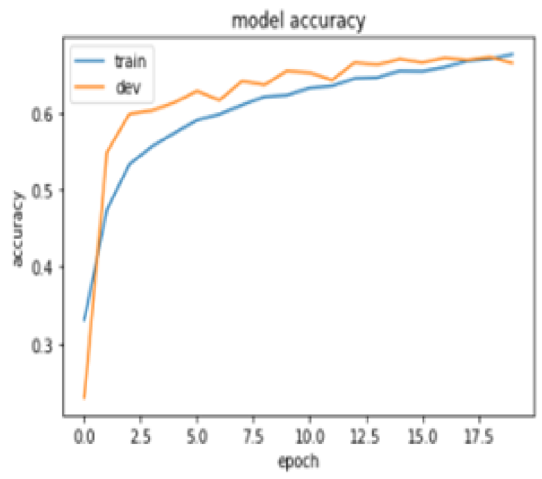
- ResNet50
- SeNet50
- VGG16
- And a model similar to the one discussed above but with a linear flow instead of branching and then combining.(A model proposed in the paper by Pramerdorfer and Kampel's)

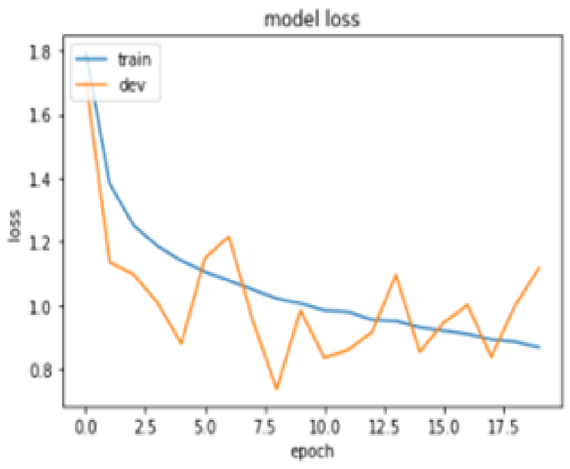
ResNet50:

ResNet-50 is a convolutional neural network that is 50 layers deep. It is defined in Keras with 175 layers. The output layer has two fully connected layers of sizes 4,096 and 1,024 respectively and a softmax output layer of 7 emotion classes.

Keras provides the SGD class that implements the stochastic gradient descent optimizer with a learning rate and momentum.SGD optimizer takes decay value as ‘0.0001’ and update the learning rate by a decreasing factor in each epoch. The learning rate was fixed as 0.01 and a batch size of 32 is fixed. The first 170 layers in ResNet remain undisturbed, and keep the rest of the network trainable.

Graphs:

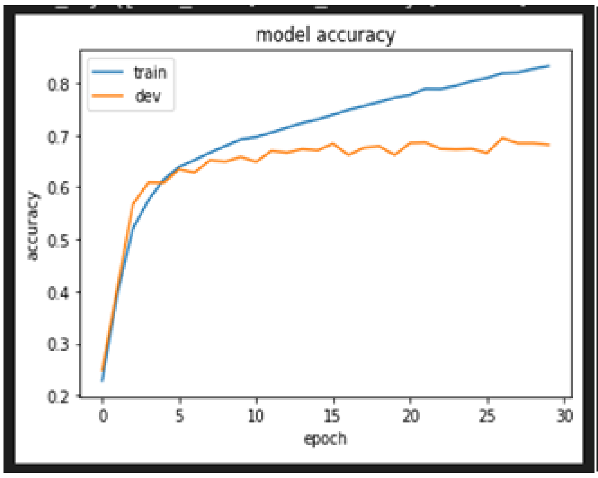


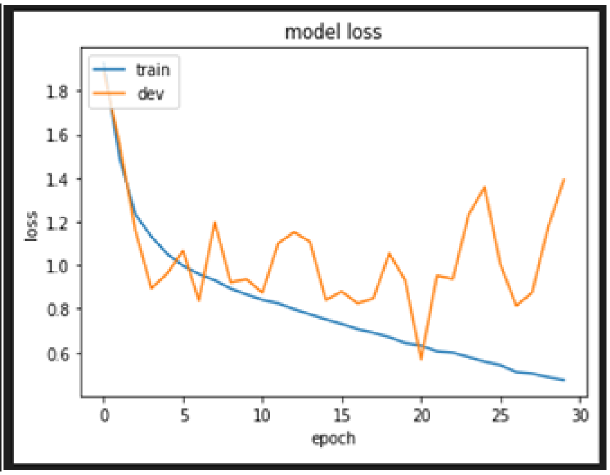


SeNet50:

SeNet50 is a deep residual network with 50 layers. SeNet50 has a similar structure with ResNet50 . The Squeeze-and-Excitation (SE) network is an architectural unit designed to improve the representational power of a network by enabling it to perform dynamic channel-wise feature recalibration.The block has a convolutional block as an input.Each channel is squeezed into a single numeric value using average pooling.A dense layer followed by a ReLU adds non-linearity and output channel complexity is reduced by a ratio.Another dense layer followed by a sigmoid gives each channel a smooth gating function and each feature map of the convolutional block is given a weight based on the side network the excitation.The network was trained on the same set of parameters as used for ResNet50.

Graphs:

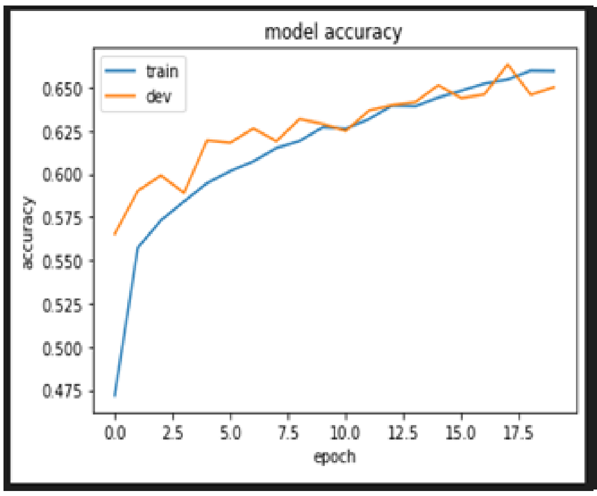


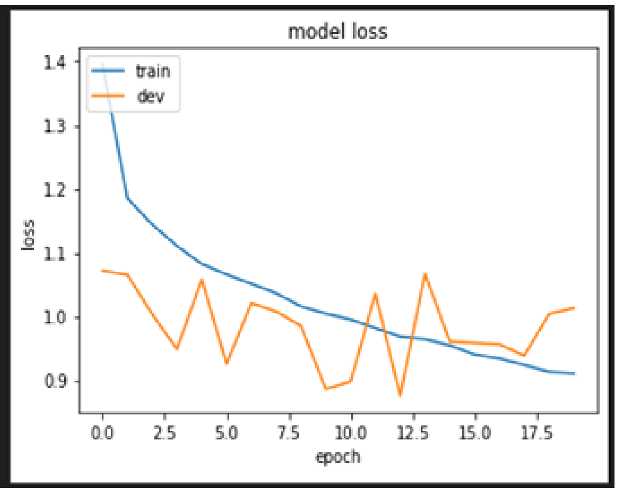


VGG-16:

VGG16 is more complex and has many more parameters compared to senet50 and resnet50. All pre-trained layers are kept frozen and two Fully Connected layers of size 4096 and 1024 are added respectively with 50% dropout. The optimizers used are Adam optimizer with learning rate 0.001 and SGD (stochastic gradient descent) optimizer with learning rate 0.01 and the sgd_decay value is specified as 0.0001 and update the learning rate by a decreasing factor in each epoch. The input data samples are resized and data augmentation is adopted to produce more training samples from the existing ones.

Graphs:

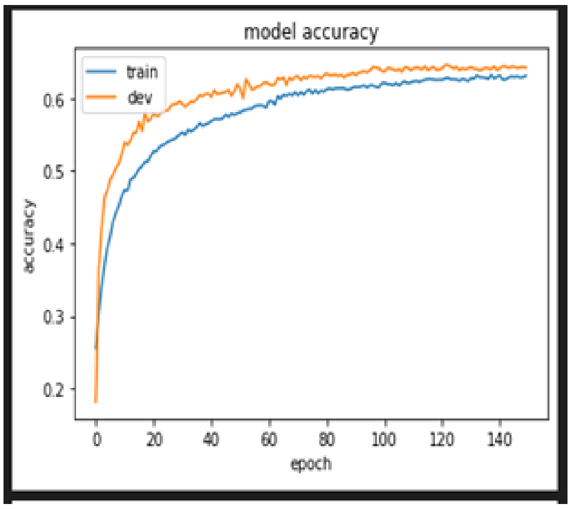


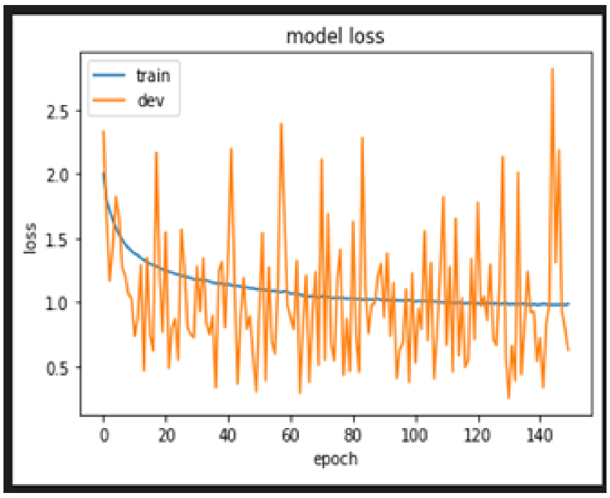


Pramerdorfer and Kampel's model:

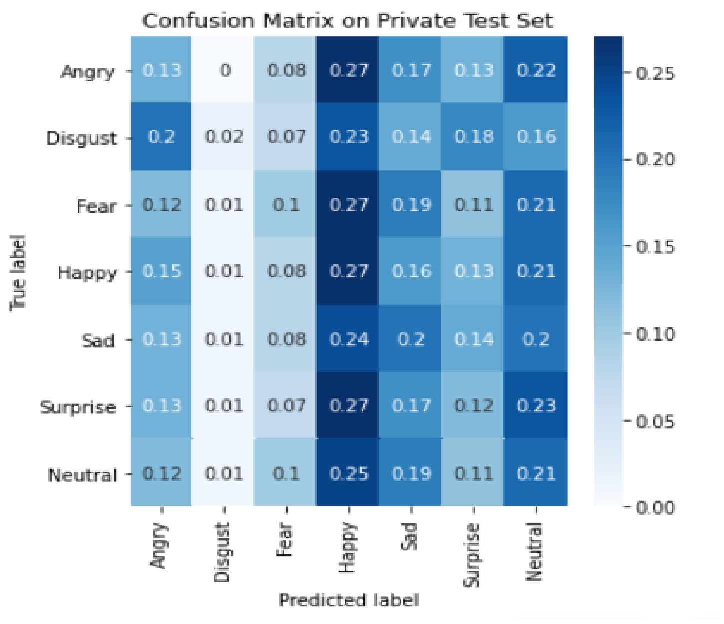
This model consists of four stages of convolutional and max-pooling layers, followed by an FC layer of size 1024 and a softmax output layer. The convolutional layers use 32, 32,64 and 64 filters all of size 3x3, respectively. The max-pooling layers use kernels of size 2x2. ReLU was utilized as the activation function. To improve performance, we also added batchnorm at every layer and 30% dropout after the last FC layer in one model and 40%dropout in another model. We trained the model for 150 epochs, optimizing the cross-entropy loss using stochastic gradient descent. The initial learning rate, batch size, and weight decay are fixed at 0.1, 128, and 0.0001, respectively. The learning rate is halved if the validation accuracy does not improve for 10 epochs.

Graphs:



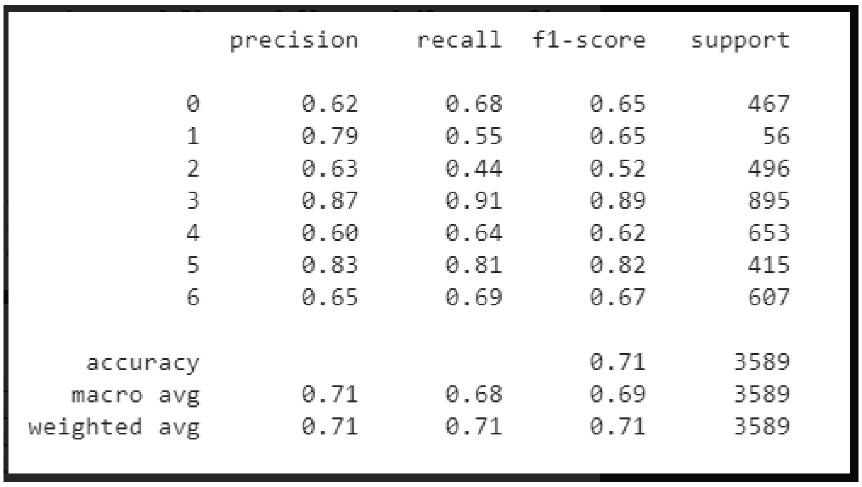


Confusion matrix when tested on the private test dataset of fer2013



PERFORMING ENSEMBLING

We trained 5 models in total as discussed above. The five models are resnet50, senet50, vgg-16, Pramerdorfer and Kampel's model with 2 different dropout values. In order to get better performance and improve the accuracy of the model, ensembling technique is done on these five models.Five models are loaded and given under two variable names. Accuracy obtained from the ensemble model is 71.18%


```
              precision    recall  f1-score   support

           0       0.62      0.68      0.65       467
           1       0.79      0.55      0.65        56
           2       0.63      0.44      0.52       496
           3       0.87      0.91      0.89       895
           4       0.60      0.64      0.62       653
           5       0.83      0.81      0.82       415
           6       0.65      0.69      0.67       607

    accuracy                           0.71      3589
   macro avg       0.71      0.68      0.69      3589
weighted avg       0.71      0.71      0.71      3589
```

COMPARING ACCURACIES

Model	Accuracy on Private Test Set	Accuracy on Public Test Set
Pramerdorfer and Kampel's model-1(dropout=0.3)	67.01%	64.976%
Pramerdorfer and Kampel's model-2(dropout=0.4)	65.728%	64.251%
ResNet50	68.849%	66.982%
SeNet50	67.038%	65.867%
VGG-16	69.796%	67.651%

Model Name	Accuracy
Model Proposed	58.07%
Ensembling Result	71.1%

We were able to infer that slight changes in the architecture of the model can increase the performance of the model.We were able to find out that more the number of Layers,better the accuracy though in some cases it decreased the performance. When compared with another model which uses transfer learning techniques such as ResNet50,SeNet50 and VGG16 and performing ensemble on them yielded much better performance.

13. CONCLUSION AND FUTURE ENHANCEMENTS

As a part of our project, we explored convolutional neural networks and many more architectures related to it. After understanding and fine-tuning various parameters involved with it we were able to come up with a slightly better model and we were able to visually see the differences when we use only a single model and the result obtained by ensembling multiple trained models.

By further looking into the fer2013 dataset we found out that there are a few ambiguities and also few cartoons which when resolved can yield much better results with the same model itself. We can also use some other extra datasets to train the model exposing to a wider variety of ethnic faces like the japanese dataset available.We can also further improve the accuracy by deploying it in real time environments ,store the results, add it to the dataset and compile the model again.We can also incorporate few advanced transfer learning models such as VGG 19,Google Inception Model along with the ones proposed to yield much better accuracies.

14. REFERENCES

1. *Franc¸ois Chollet. Xception: Deep learning with depthwise separable convolutions. CoRR, abs/1610.02357, 2016.*
2. *Yichuan Tang. Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239, 2013.*
3. *Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning, pages 448–456, 2015*
4. *Pramerdorfer, C., Kampel, M.: Facial expression recognition using convolutional neural networks: state of the art. Preprint arXiv:1612.02903v1, 2016.*
5. *S. Li and W. Deng, "Deep facial expression recognition: A survey," arXiv preprint arXiv:1804.08348, 2018.*
6. *Minaee S., Abdolrashidi A., "Deep-Emotion: Facial Expression Recognition Using Attentional Convolutional Network", 2019*
7. *A. Mollahosseini, D. Chan, and M. H. Mahoor, "Going Deeper in Facial Expression Recognition using Deep Neural Networks," CoRR, vol. 1511, 2015.*
8. *E. Barsoum, C. Zhang, C. C. Ferrer, and Z. Zhang, "Training deep networks for facial expression recognition with crowd-sourced label distribution," in Proceedings of the 18th ACM International Conference on Multimodal Interaction. ACM, 2016, pp. 279–283.*
9. *Quinn M., Sivesind G., and Reis G., "Real-time Emotion Recognition From Facial Expressions", 2017*
10. *I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee et al., "Challenges in representation learning: A report on three machine learning contests," in International Conference on Neural Information Processing. Springer, 2013, pp. 117–124.*

15. APPENDIX A

The undersigned acknowledge they have completed implementing the project "*Analysing Emotions From Facial Expressions*" and agree with the approach it presents.

Signature:	[illegible]	Date:	18/5/2021
Name:	S.K.Risheek Rakshit	Roll No:	2018103580

Signature:	K.S. Sumaiya Fathima	Date:	18/5/2021
Name:	K.S.Sumaiya Fathima	Roll No:	2018103607

16. APPENDIX B: REFERENCES

The following table summarizes the research papers referenced in this document.

Document Name	Description	Link
Real-time Convolutional Neural Networks for Emotion and Gender Classification	Introduced real-time enabled guided back-propagation visualization technique. Guided back-propagation uncovers the dynamics of the weight changes and evaluates the learned features.Also suggests an architecture to use for fer	https://arxiv.org/abs/1710.07557v1
Deep Facial Expression Recognition: A Survey	Review existing novel deep neural networks and related training strategies that are designed for FER and discuss their advantages and limitations.	https://arxiv.org/abs/1804.08348
Facial Expression Recognition with Deep Learning	Uses results and models from recent publications to improve accuracy, including transfer learning, data augmentation, class weighting, adding auxiliary data, and ensembling. In addition, we analyzed our models through error analysis and several interpretability techniques	https://arxiv.org/ftp/arxiv/papers/2004/2004.11823.pdf

17.APPENDIX C:KEY TERMS

The following table provides definitions for terms relevant to this document.

Term	Definition
CNN	**Convolutional Neural Network** is an artificial neural network used for image analysis to identify patterns and make sense of them. The convolutional layers are the hidden layers which predict the patterns
FER2013	**F**acial **E**xpression **R**ecognition is the expansion for FER. FER2013 is the dataset used and it consists of 48x48 pixel grayscale images of faces.The training set consists of 28,709 examples and the public test set consists of 3,589 examples.
VGG16	VGG16 (also called OxfordNet) is a convolutional neural network architecture named after the Visual Geometry Group from Oxford, who developed it. It was used to win the ILSVR (ImageNet) competition in 2014. To this day it is still considered to be an excellent vision model, although it has been somewhat outperformed by more revent advances such as Inception and ResNet.
ResNet-50	**ResNet-50** is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories.