MACHINE LEARNING PROJECT REPORT

Emotion Capture

Ruthwik Ganesh (01FB15ECS248)
Simran Dhinwa (01FB15ECS293)
Satvik Kumar (01FB15ECS269)
Sai Rohit S (01FB15ECS255)

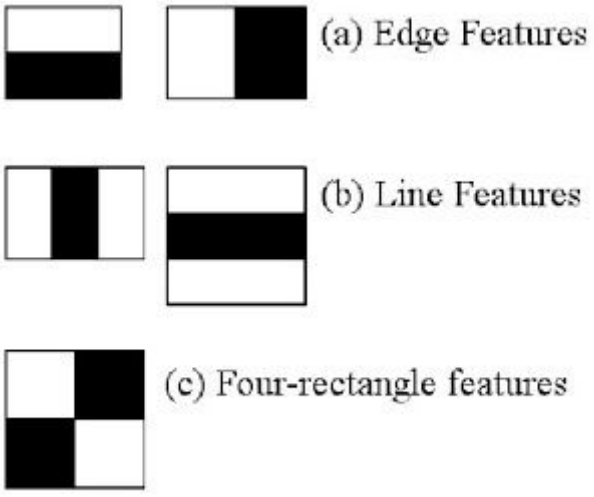
INDEX

Overview

We often have various reactions towards daily life activities. For example, when we watch the trailer of a movie, or when we are watching a sport game. We wish to build a system to analyse someone's reaction during this time. Say for example, we wish to analyse whether the person likes the trailer based on the video of he or she watching the trailer. This can be classified as a form of sentiment analysis problem where we are trying to detect the sentiment of a person towards a video or towards a game. At the base of the problem is the task of image classification. Given an image of a person with an expression on his face, the task would be to classify the expression into a category and then analyse whether this is a positive category, a neutral category or a negative category. Based on the weighted sum of all these, we can obtain a score for the sentiment.

Preprocessing

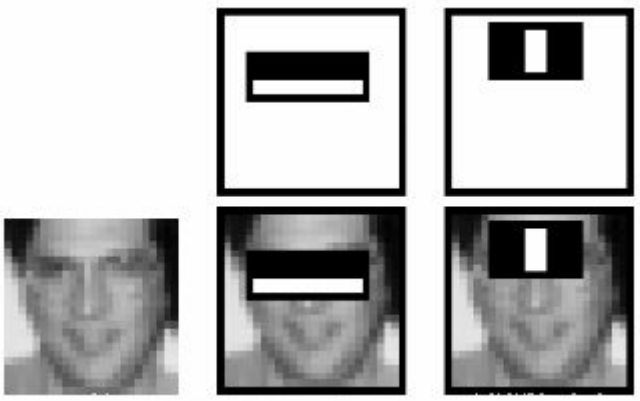
In order to run the images for the appropriate model, we first need to do some preprocessing in order to achieve an efficient prediction.**Object detection** is an important feature of computer science.The algorithm **Haar feature-based cascade classifier** is an efficient algorithm for object detection. So as in input we will require a sufficient amount of negative and positive images.
Initially a lot of positive images(with faces) are required followed my negative images(without faces) to train the classifier.



Some Haar Kernels used for Object Detection

The Haar features that are used are like any other convolution kernel. The way to evaluate the value is by subtracting sum of pixels under the white rectangle in the kernel from the sum of the black rectangle.

So we obtain a lot of features.But many in these would be irrelevant. And in order to overcome this we incorporate the Adaboost algorithm. So we first apply every feature on the training images and then find the best threshold which can classify the face as negative or positive. Finally we select the feature with minimum error rate. The final classifier is a weighted sum of all these weak classifiers.



Haar Example for a Feature Extraction (in this case the **eye**)

This kernel is based on the fact that to detect the eye, we **need to have two dark regions and a white region** in between to match the eye. This is one of the kernels above. Similar kernels are used for object detection based on the details of the objects.

The next challenge comes about applying all the features to each window of the image.And moreover most of the image is a non-face region. So if we could somehow check if a window is a non face region we could discard in a single shot those windows. To achieve this we use the concept of **Cascade of Classifiers**. Instead of applying all the features we first take a group of features categorised into different stages, so if a window fails the first stage itself we discard that window. Otherwise we go ahead with the next stage and continue the process.

The AffectNet Dataset

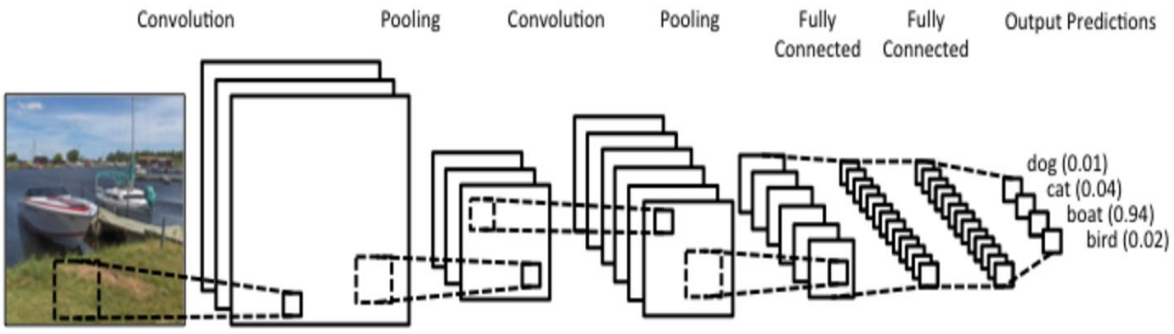
Existing annotated databases of facial expressions in the wild are small and mostly cover discrete emotions (aka the categorical model). There are very limited annotated facial databases for affective computing in the continuous dimensional model (e.g., valence and arousal).

To meet this need, AffectNet, a new database of facial expressions in the wild was created, by **collecting and annotating facial images**. AffectNet contains more than 1M facial images collected from the Internet by querying three major search engines using 1250 emotion related keywords in six different languages. About half of the retrieved images (~440K) were manually annotated for the presence of seven discrete facial expressions (categorial model) and the intensity of valence and arousal (dimensional model). AffectNet is by far the largest database of facial expressions, valence, and arousal in the wild enabling research in automated facial expression recognition in two different emotion models. Two baseline deep neural networks are used to classify images in the categorical model and predict the intensity of valence and arousal.

The following is the list of classes in the database and the number of images under each category.

Neutral	75374
Happy	134915
Sad	25959
Surprise	14590
Fear	6878
Disgust	4303
Anger	25382
Contempt	4250
None	33588
Uncertain	12145
Non-Face	82915
Total	420299

Why CNN?



Convolutional Neural networks allow computers to see, in other words, Convnets are used to recognize images by transforming the original image through layers to a class scores. CNN was inspired by the visual cortex. Every time we see something, a series of layers of neurons gets activated, and each layer will detect a set of features such as lines, edges. The high level of layers will detect more complex features in order to recognize what we saw.
Generally CNN are neural networks stacked over the other, the first layer being convolution neural network followed by pooling layer then a fully connected layer. Depending on the application these layers can be stacked upon each stack to obtain the desired result.

Conv Layer :

The objective of a Conv layer is to extract features of the input volume.
A part of the image is connected to the next Conv layer because if all the pixels of the input is connected to the Conv layer, It will be too computationally expensive. So we are going to apply dot products between a receptive field and a filter on all the dimensions. The outcome of this operation is a single integer of the output volume (feature map**)**. Then we slide the filter over the next receptive field of the same input image by a Stride and compute again the dot products between the new receptive field and the same filter. We repeat this process until we go through the entire input image. The output is going to be the input for the next layer.

Filter, Kernel, or **Feature Detector** is a small matrix used for features detection.
Convolved Feature, **Activation Map** or **Feature Map** is the output volume formed by sliding the filter over the image and computing the dot product.

Receptive field is a local region of the input volume that has the same size as the filter.
Depth is the number of filters.
Depth column (or **fibre**) is the set of neurons that are all pointing to the same receptive field.
Zero-padding adds zeros around the outside of the input volume so that the convolutions end up with the same number of outputs as inputs. If we don't use padding the information at the borders will be lost after each Conv layer, which will reduce the size of the volumes as well as the performance.

ReLU layer :

ReLU Layer applies an elementwise activation function max(0,x), which turns negative values to zeros (thresholding at zero).

POOL layer :

Pool Layer performs a function to reduce the spatial dimensions of the input, and the computational complexity of our model. And it also controls overfitting. It operates independently on every depth slice of the input. There are different functions such as Max pooling, average pooling, or L2-norm pooling. However, Max pooling is the most used type of pooling which only takes the most important part (the value of the brightest pixel) of the input volume.

Fully Connected Layer :

Fully connected layers connect every neuron in one layer to every neuron in another layer. The last fully-connected layer uses a softmax activation function for classifying the generated features of the input image into various classes based on the training dataset.

Our Model

The classifier that we've built using keras (tensorflow background) consists of 4 CNN stack. Input to the model consists of grayscale images of the resolution 200 X 150, which is passed into the 2D Convolution layer of the 1st stack, the output of this layer is 0-thresholded using a relu activation unit and then passed to the next layer which is responsible for max pooling operation. These layers are followed by three such layered stacks which is further responsible for picking out the final details of the image, the output generated by these four stack of layers is flattened and and passed to a fully- connected layer having 128 input neurons activated by relu units having 50% drop out probability, coupled with another fully-connected layer of the whose output dimension being same as that of the number of classes ie 11. This model is optimized by adam optimizer.

```
model=Sequential()

#image size is 200 X 150 ( basically a black and white image )
#change if different
model.add(Convolution2D(64,(3,3),input_shape=(200,150)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Convolution2D(96,(4,4)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Convolution2D(64,(5,5)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Convolution2D(96,(6,6)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(output_dim=10,activation="softmax"))

model.compile(loss='categorical_crossentropy',
                        optimizer='adam',
                        metrics=['accuracy'])

batch_size=16
```

Algorithm

Step 1 : Split the video into images frame wise taking one frame per second.

Step 2 : For each image in the split images :

a. Convert the image to a grayscale image
b. Resize the image to match the input shape of the model
c. Pass the image through the pre trained and stored model.
d. Get the tag output and check the corresponding class output.
e. Increase the count of the corresponding output class.

Step 3 : Calculate sentiment as :

sentiment=((positiveCount*positiveWeight)+(neutralCount*negativeWeight)+ (negativeCount*negativeWeight)) / totalCount

Where :

a. **positiveWeight** is the weight assigned to the positive class (1)
b. **negativeWeight** is the weight assigned to the negative class (0.2)
c. **neutralWeight** is the weight assigned to the neutral class (0.5)

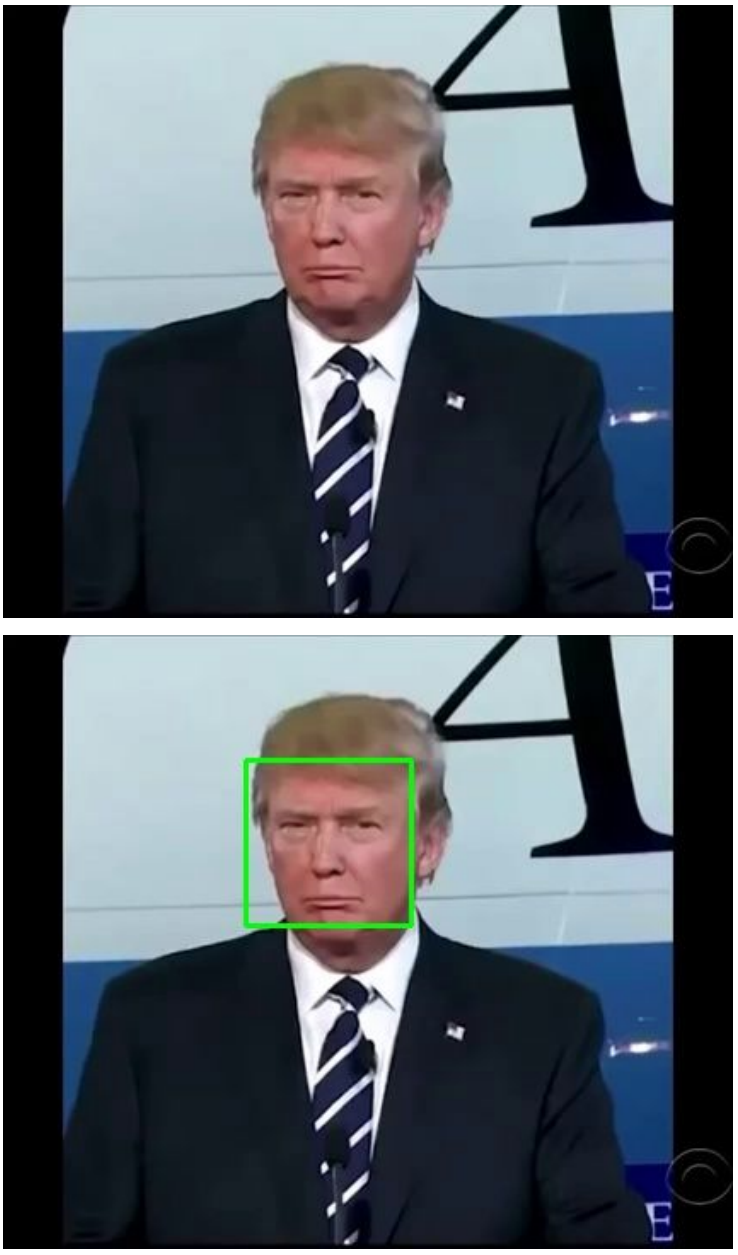
So basically, the entire sentiment is detected as a **weighted average of the number of positive, negative and neutral classified frames**.

Classified Emotions and their Labels with Emotion Category

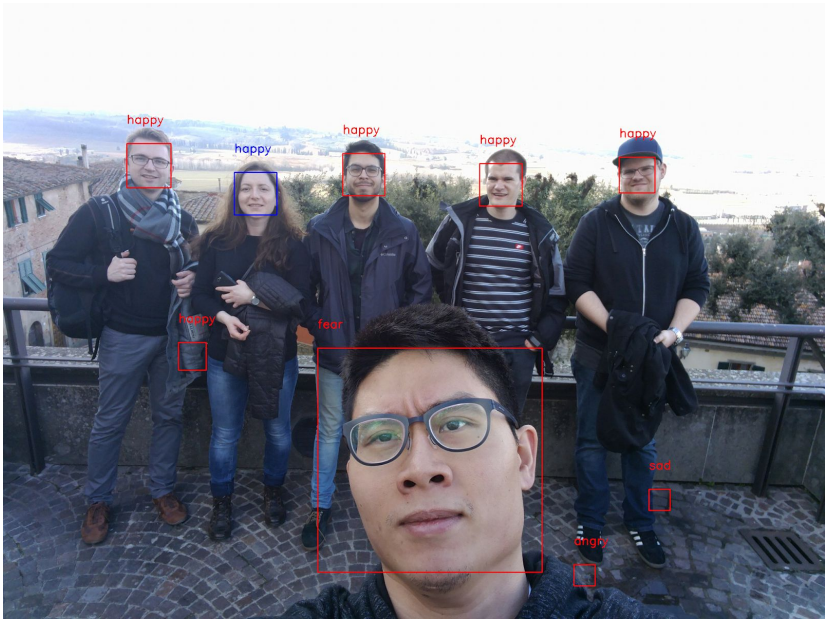
Emotion Name	Classifier Label	Emotion
Angry	0	Negative
Disgust	1	Negative
Fear	2	Negative
Happy	3	Positive
Sad	4	Negative
Surprise	5	Positive
Neutral	6	Neutral

Results

For a particular **input image**, if we apply Haar Face Segmentation using the Cascade Classifier :



Stage 2 is where we apply the **emotion detection algorithm** to each of the face detected in the image.
For example, consider an image with multiple faces :



This is the combined **output** of **two** stages :

1. Stage 1 is the detection of the face using **Haar Cascade Classifier**.
2. Stage 2 is then the **emotion classification** of the face using the pre-trained model.

This is done for a single image, so when we split the video into frames, apply these two stages of processing on the image and then use the algorithm we obtain the sentiment for the video.

Model Accuracy and Conclusion

CNN is the most advanced technology with respect to image classification which has become more and more significant because of the recent improvements in computing machines available. Our model which was run for 50 epochs on the training data on 1000 images for each class with a training to test ratio of 70:30 gave an accuracy of 78%.

Improvements in terms of the classification can be obtained by patch extraction and better preprocessing of the image. This will ensure a much better and a compact input to the neural network and thus will improve the accuracy.

References

1. Object detection using Haar-cascade Classifier, Sander Soo Institute of Computer Science, University of Tartu.