AUTOMATIC EMOTIONAL DETECTION FROM IMAGE DATA WITH CNN AND SVM

Nicholas Klepp
University of Georgia
Department of Computer Science
nickbk@uga.edu

I-Huei Ho
University of Georgia
Department of Statistics
ihuei.ho25@uga.edu

Abstract

Recognizing emotion through facial expression is a key aspect of human communication. While human beings are capable of accurately assessing emotion through interpreting facial expressions, it is still challenging for artificial intelligence to automatically recognize the emotions of human beings. The task is expressed as image classification along the axis of expressed emotion. This paper explores the use of two different decision frameworks for classifying the images: convolutional neural networks (CNN) and support vector machines (SVM). Images consist of subjects with disparate ethnicity, age, and sex, each expressing one of seven emotions: Anger, Contempt, Disgust, Fear, Happy, Sadness, and Surprise. The SVM method is deemed preferable to the CNN method, though both achieve reasonable results in terms of accuracy of decision. The difficulties of training the CNN in emotion classification is discussed. To further this research, employing more robust CNN or employing an ensemble method including the HOG+SVM are inevitable.

1 Introduction

Human beings communicate in a number of ways, one of which is the expression of emotional state through facial expression. Accurately interpreting and understanding these facial expressions is a critical component of understanding human communication in a variety of settings. Facial expression is used in one-on-one conversation for emphasis and clarification; actors use facial expression to convey motivation or explain plot points; and facial expression is a vital tool for assessing security in crowds, during volatile public gatherings, etc.

It stands to reason that if humans are able to accurately identify the emotion in facial expressions that a computer scientist should be able to train a computer to do so as well. Improving the ability of computers to correctly identify the facial expression of human emotions would be a boon to numerous areas of active research and exploration in computer vision and machine learning, including image captioning, movie synopsis, and crowd control / threat detection.

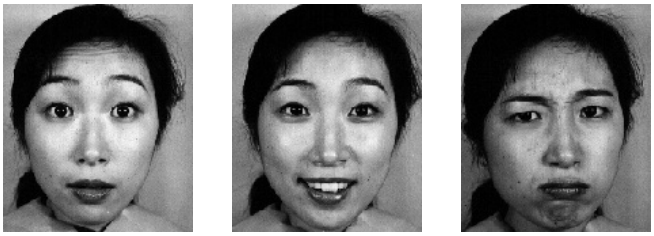
The rest of this paper is organized into five sections: the first section will allow a brief description of the data used in our effort to train the computers, the second section will focus on an introduction to CNNs, the third will briefly introduce HOG, the fourth will discuss the methodology employed, and the last will focus on the results achieved in our experiments as compared to the results achieved by other researchers on this question.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

2 Datasets

2.1 JAFFE

The Japanese Female Facial Expression (JAFFE) dataset contains 213 images of 10 subject and 6 emotions (Anger, Disgust, Fear, Happy, Sadness, and Surprise) with 1 neutral. Each subject repeats the expression for three to four times. This dataset is smaller and less robust than the following dataset, but its small size makes it a good dataset for baseline testing for computationally expensive algorithms.



(a) Surprise (b) Happy (c) Angry

Figure 2.1: JAFFE dataset

2.2 RafD

The Radbound Faces Database (RafD) contains 1068 images and are composed of three ethnicities (Caucasian, Moroccan, Kid), two genders (Male, Female), eight emotions (Angry, Contemptuous, Disgusted, Fearful, Happy, Sad, Surprised, Neutral), and three eyes angles (Frontal, Left, Right) by 67 subjects. Regarding eye angles as replicates, each subject express each emotion three times. Table 2.1 shows the amounts of each emotion and Figure 2.2 shows three cropped example images in this dataset.

(a) Sad-Morrocan-Male (b) Angry-Caucasian-Male (c) Neutral-Caucasian-Female

Figure 2.2: RafD dataset

	Neutral	**Anger**	**Contm**	**Disg**	**Fear**	**Happy**	**Sad**	**Surpr.**	**Total**
Jaffe	30	30	0	29	32	31	31	30	213
RafD	201	201	201	201	201	201	201	201	1068

Table 2.1: Emotion Distribution

3 Convolutional Neural Networks

Convolutional Neural Netwroks (CNNs) are a refinement on the classical, densely connected, feed forwarded artificial neural network. They receive most of their focus in image recognition, though they are adaptable to other problem domains as well.

Consider a binary classification task on image input in the form of a two-dimensional array of pixel data. If the image is $n \times n$ many pixels, then the number of features in the data is n^2. If we are to train a classical, feed forward densely connected ANN with a single hidden layer with m many inputs on this input data, then the number of parameters in the model is $n^2 * m = O(n^3)$, assuming that the hidden layer has a reasonably large number of nodes in it (a reasonable assumption). This is a HUGE number of parameters, and our our net is going to suffer from a number of over-parameterized problems, not to mention excessive training time.

CNNs address problems introduced above by taking advantage of a reasonable assumption: locality. The principle of locality states that pixels which are near to one another in an image should have more in common than pixels which are far away from each other. This is usually a good assumption for image data. To exploit the principle of locality, CNNs use convolutional filters to extract information about regions of the image. Fig. 3.3 is a visual depiction of how a convolutional filter works. A

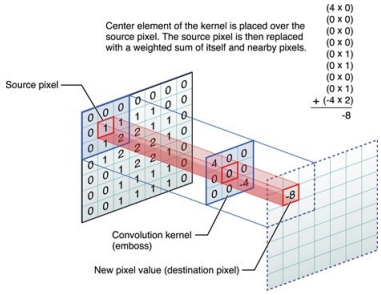


Figure 3.3: Convolutional Filter

convolutional filter is a function which takes as input an $n \times n$ matrix, in the case of Fig. 3.3 a 3×3 matrix, multiplies the input element-wise by a weighting matrix, and returns the sum of the product as its output. By applying the convolutional filter to each $n \times n$ region of the original image, the CNN produces a new, intermediary representation of the image. Filters with different weights naturally produce different intermediary images, and different filters are capable of capturing different features of the image. It is typical for a CNN to employ many filters on each input, some times hundreds. It is the weights of these filters that the CNN learns during back-propagation.

Once the intermediary representation of the image has been produced, most CNNs employ a technique known as "pooling" which helps to reduce the number of parameters the model must fit. Fig. 3.4 is an example of one of the most common forms of pooling, max pooling. Pooling "down

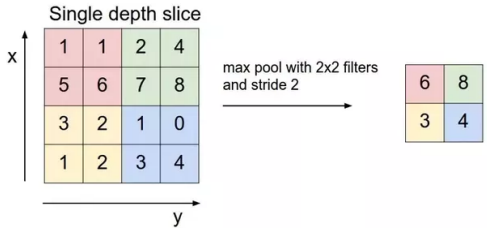


Figure 3.4: 2×2 Max Pooling with Stride 2

samples" the intermediate representation of the image, again utilizing the assumption of locality, in

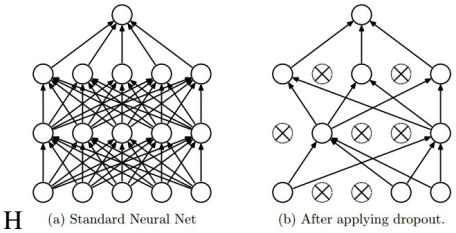


Figure 3.5: Dropout Example

order to reduce the size of each intermediate representation. The output of a 2×2 Max Pooling filter is the maximum pixel value in any 2×2 region of the image. The stride of the pooling filter dictates the number of pixels the pooling filter moves after each region is processed. So, a 2×2 max pooling filter with a stride of 2 run over an image of $n * m$ pixels returns a new image of $n/2 \times m/2$ pixels.

Dropout is another regularization method frequently used to avoid over-fitting a model. At training time, every node in the network is set to output a 0 valued activation with some preset probability. As a result the net never becomes too dependent on a small subset of its nodes and instead is encouraged to utilize all of the nodes in the decision process. Thus, dominant features which are overrepresented in the data are not allowed to dominate the decision process. Fig. 3.5 describes a traditional ANN with a dropout mechanism applied to its hidden layer.

The typical architecture of a CNN employs multiple "Convolutional Layers" consisting of one or more convolutional filters, followed by a max pooling filter applied to each of the intermediate results produced by the convolutional filters, followed by a final dropout layer. We have followed this general guideline in our architectures.

4 Histogram of Oriented Gradients

A feature descriptor is a down-sampled representation of an image that seeks to throw away supplementary information but retain any useful features. A feature descriptor converts an image of 3 dimensions, width $\times$ height $\times$ channels, to a feature vector of length n. For example, a HOG feature descriptor when input an image of size $64 \times 128 \times 3$ will output a feature vector of length 3780.

The feature vector is not useful for the purpose of viewing the image, but it is useful for image recognition and object detection. The feature vectors produce impressive results when feed into image classification such as Support Vector Machines (SVM). Then, determining useful and extraneous features will be critical to these kinds of algorithms. Good features extracted from an image should be able to tell the difference between shapes. In HOG feature descriptor, the distribution of directions of gradients, which is named as histogram of oriented gradients, are used as the features we expected. The magnitude of gradients of an image is large around edges and corners, which contain lots of information about an object's shape.

For example of a pedestrian detection, HOG feature descriptor is calculated on a 64×128 patch of an image. Patches at multiple scales are analyzed at many image locations but restricted to a fixed aspect ratio. Thus, in this pedestrian case, patches can be any size of ratio 1:2 but not 1:3 or others.

To calculate a HOG descriptor, firstly calculate the horizontal and vertical gradients by filtering the kernels in table Fig. 4, then calculate the histogram of gradients.

Figure 4.6 shows the gradient image of absolute x-gradient and y-gradient, and magnitude of gradient. The gradient image removed a lot of non-essential information such as constant colored background, but highlighted outlines. Even though the colors of the images are not correct as original images, we are still able to recognize what is in the image and where the pedestrian is.

-1	0	1

-1
0
1

Table 4.2: HOG horizontal and vertical kernels

(a) Abs.Value of x-gradient (b) Abs.Value of y-gradient (c) Magnitude of gradient

Figure 4.6: Gradient Images

At every pixel, the gradient has a magnitude and a direction. The image is divided into 8×8 cells and a histogram of gradients is calculated for each 8×8 cells. The histogram is essentially a vector of 9 bins corresponding to angles 0, 20, 40, 60, ..., 160. The patch of the image overlaid with arrows shows the gradient—the arrow shows the direction of gradient and the length shows the magnitude. The direction of arrows points to the direction of change in intensity and the magnitude shows how big the difference is.

Gradients of an image are sensitive to overall lighting. To make the feature descriptor be independent of lighting variations, normalize the histograms then they will not affected by lighting variations. We implement 16×16 blocks to concatenate four histograms to a 36×1 element vector. At the end, the final feature vector for the entire image patch, those 36×1 vectors are concatenated into one huge vector with 3780 dimensions. The size comes from 7 horizontal positions times 15 vertical positions in each 16×16 block, times the length of 36×1 vector. Briefly, the size is obtained by the equation $7 \times 15 \times 36 = 3780$.

In the following section, the reason of implementing HOG to extract features from images will be mentioned, and the advantages of HOG will also be explained.

5 Methodology

5.1 Face Detection

The face extraction from the image is done first using a Haar cascade object face detector. The detection framework seeks to identify faces or features of a face by passing feature boxes over an image and computing the difference of summed pixel values between adjacent regions. In a feature-region of interest on the face, it will generally hold that some areas will be lighter or darker than surrounding area. It is noted that Haar-like features are simple and fast, but therefore weak classifier, which requires multiple passes.

Once the faces are detected, the images are cropped around the detected face. Owing to the naturally variability in the size of each subject's face, each image then has a slightly different size. The images

must then be padded with zero padding on the edges to properly conform to the standards expected by the CNN.

In implementing CNN, we will pad the cropped faces to 444x444 pixels and input them into the network. In implementing histogram of oriented gradients, we will resize the cropped faces to 384x384 pixels and deskew the faces.

5.2 Convolutional Neural Network

Both a shallow and (relatively) deeper CNN were trained on the task of emotional recognition. Fig. 5.7 shows the architecture of the more shallow of the configurations which were trained, CNN1. The 444*444 input is first passed through 3 separate 3*3 convolution filters, to which result a 2*2 max pooling filter with stride 2 is applied. This layer is passed through a single 3*3 convolution filter which result is also then passed through a 2*2 max pooling filter with stride of 2. The final image is flattened and passed to a densely connected layer of 256 nodes which are then passed to the output layer with 8 nodes. Drop out regularization at a rate of 0.25 is applied to all but the last layer in the architecture.

Fig. 5.8 shows the architecture of the deeper of the architectures. As opposed to CNN1, CNN2 has five convolutional layers. The 444*444 input is passed first into a layer of (7,7,64) convolutions. The second layer consists of (5,5,2) convolutions, the third and fourth of (3,3,2) convolutions, and the last layer is a (3,3,1) convolution. After each convolution layer is a (2,2) max pooling layer with a stride of 2, as well as a dropout applied at a rate of 0.25. The next layer is again a 256 densely connected layer, and the output layer is again a densely connected layer of 8 nodes.

All of the convolutional layers mentioned above as well as the densely connected 256 node layer employ RELU activation functions, while the final output layer uses a softmax activation. Bias regularization is applied to the convolutional layer outputs, and the "adam" optimizer is used to optimize back-propagation.

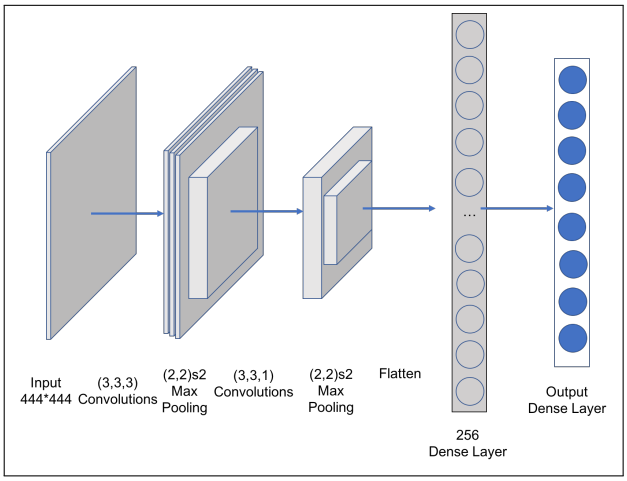


Figure 5.7: Shallow CNN Architecture, CNN1

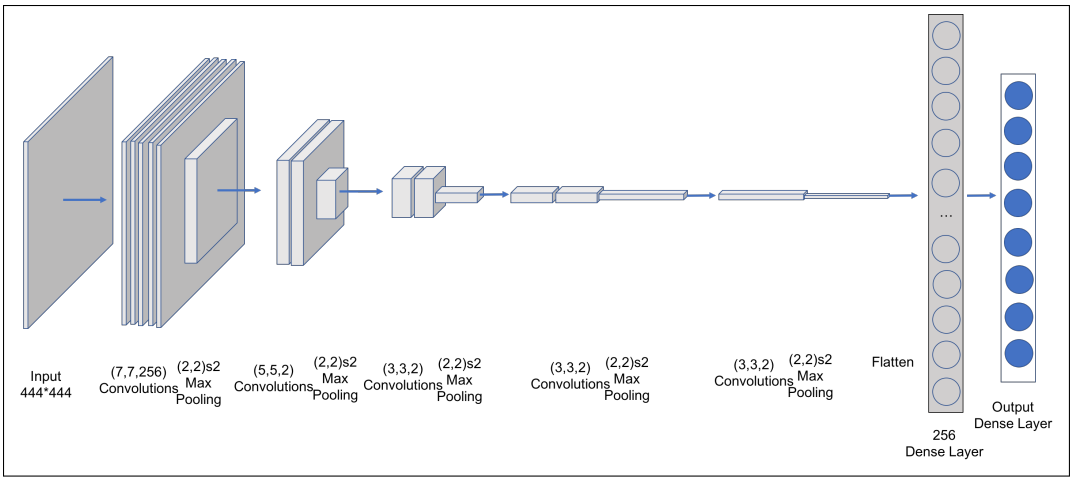


Figure 5.8: Shallow CNN Architecture, CNN2

5.3 Histogram of Oriented Gradient

Other than directly inputting images into neural network, facial recognition is also known to extract features and apply support vector machine to classify the emotions. Histogram of oriented gradients (HOG) features are calculated and extracted from the faces. To determine the HOG features, an image is separated into evenly-sized and spaced grids. Within each grid the orientation of the gradient for each pixel at (x,y) is calculated as:

$$\theta_{x,y} = \tan^{-1} \frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y}$$

where L is the intensity function describing the image. These orientations of gradients are then binned into a histogram for each grid, and every grid within the image is concatenated in a HOG description vector. Figure 5.9 shows the steps of obtaining HOG feature descriptors: (a) original image, (b) gradient vector calculation, (c) pixel grouping in cells, (d) cell histogram calculation, (e) cell grouping in blocks, (f) descriptor assembly. Figure 5.3 shows the calculated HOG features that are plotted on our four emotional images.

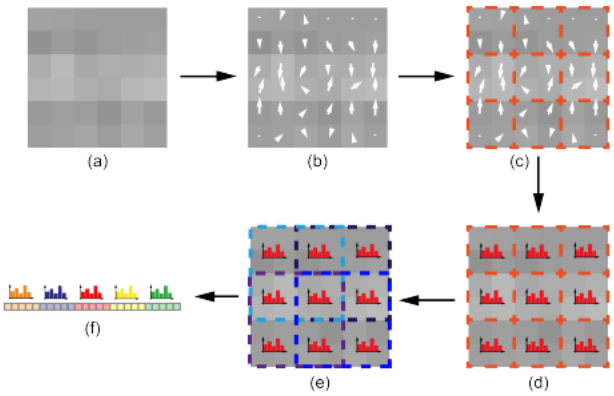
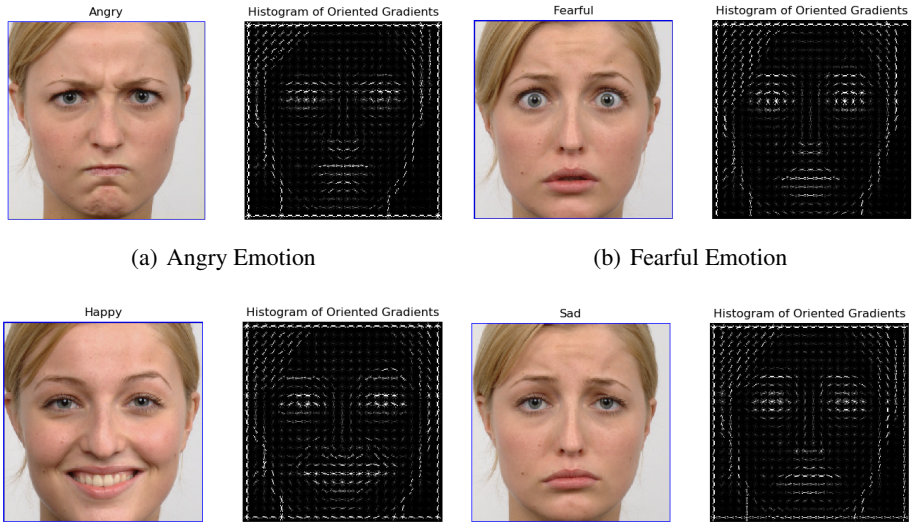


Figure 5.9: HOG Descriptor Extraction Steps



(a) Angry Emotion (b) Fearful Emotion

(c) Happy Emotion (d) Sad Emotion

Figure 5.10: Visualizations of HOG features on four emotions

Facial expressions with different muscular manipulations are expected to vary HOG features. Those extracted, resized, and deskewed faces are consistent in dimension, then the same number of HOG features of each images are extracted and required for classifier training in next stage.

5.4 Support Vector Machine

We use facial HOG feature vector for each training image, and assign a corresponding label. This requires us to know the class that each testing image belongs to. Upon completing HOG extraction for each image, we then train a multi-class support vector machine (SVM) using the HOG vectors. According to the complexity of facial images, those testing accuracy of different combination of C parameter values and γ parameter values are showed in the following results section.

6 Results

The completed training implementation uses Haas cascade detector to detect faces. Detected faces are cropped, padded, or resized, and deskewed, then CNN and SVM with HOG extraction are performed. The results obtained from training the neural net classifiers were not encouraging, though the results obtained through the HOG+SVM classifier were more impressive.

Fig 6.3 is a confusion matrix for the shallow neural net.

	Neutral	**Anger**	**Contm**	**Disg**	**Fear**	**Happy**	**Sad**	**Avg**
Neutral	25	0	0	2	1	1	1	0.833
Anger	5	24	0	0	0	1	0	0.800
Disg	2	1	21	4	0	1	0	0.724
Fear	3	3	0	24	0	1	1	0.750
Happy	2	0	0	1	24	4	0	0.774
Sad	4	0	0	1	1	2	22	0.839

Table 6.3: Shallow CNN - Training and Validation Accuracies

Fig. 6.4 is a confusion matrix for decisions made by the deeper neural net:

	Neutral	**Anger**	**Contm**	**Disg**	**Fear**	**Happy**	**Sad**	**Avg**
Neutral	21	4	0	1	0	1	3	0.700
Anger	0	28	1	0	1	0	0	0.933
Disg	0	4	23	0	0	2	0	0.793
Fear	4	1	0	1	24	1	0	0.844
Happy	1	3	0	0	0	0	26	0.774
Sad	1	0	2	0	1	25	0	0.806
Surpise	3	0	0	0	0	0	26	0.867

Table 6.4: Deep CNN - Training and Validation Accuracies

While the results may look equivalent, a deeper dive reveals a more nuanced picture. The overall accuracy for the shallow net, at 77.9%, is within just 4% of the overall accuracy of the deeper neural at 81.7%. However, in examining the results deeper we find that the shallow neural net was able to achieve a training accuracy of 94.63% but testing accuracy of just 40.6%. For the deeper CNN the results were quite different: a training accuracy of 77.85% was matched by a testing accuracy of 65.5%. With little room for improvement in the shallow net, the testing accuracy is unlikely to improve very much. However, after 500 epochs the deeper neural network was still improving its training accuracy consistently and substantially. If allowed to run for over 1000 epochs there is reason to believe that the deeper neural net could reach much higher levels of both training and testing accuracy.

The primary reason we use HOG+SVM approach is to improve speed and accuracy issues. When we implemented CNN for facial recognition, we have poor accuracy of % as it is only slightly better than human recognition. Moreover, HOG are known as fair approach to apply in image recognition and object detection.

When using the SVM classifier with HOG feature vectors, the accuracy for recognition is at most 98% when setting extreme low C parameter value. Table 6.5 shows the accuracy according to disparate combination of C and γ parameters values. Known that the larger the C parameter, the smoother boundary and the more correct training points we have. The accuracy doesn't change with different c values, on the contrary, the accuracy intensively decreases when the γ parameter increases. γ parameter defines how far the influence of a single training examples reaches. Due to the complication of facial images, the low values of γ describe that the SVM classifier only concern those few points closed to the hyperplane. Though we obtain a perfectly high accuracy, but we are still skeptical to the results and decide to use SVM with C = 12 and γ = 0.1 instead as our final SVM classifier.

γ \ C	C=12	C=12.5	C=13	C=13.5	C=14
γ=0.01	98.76%	98.76%	98.76%	98.76%	98.76%
γ=0.05	94.41%	94.41%	94.41%	94.41%	94.41%
γ=0.10	80.12%	80.12%	80.12%	80.12%	80.12%
γ=0.20	50.31%	50.31%	50.31%	50.31%	50.31%
γ=0.30	24.84%	24.84%	24.84%	24.84%	24.84%
γ=0.40	14.29%	14.29%	14.29%	14.29%	14.29%
γ=0.50	11.80%	11.80%	11.80%	11.80%	11.80%

Table 6.5: Testing Accuracy of SVM classifier

Figure 6.11 expresses the successful and failed emotional recognition when setting C parameter as 12 and γ parameter as 0.1. Those images in red are failed to recognize correctly. It is known that failures mostly happened when recognizing between surprise and fear, and between sadness and disgust. Happy and angry faces are relatively easier to classify due to the dramatic facial difference. Specifically, the confusion matrix of testing set in this SVM classifier are showed in table 6.6.

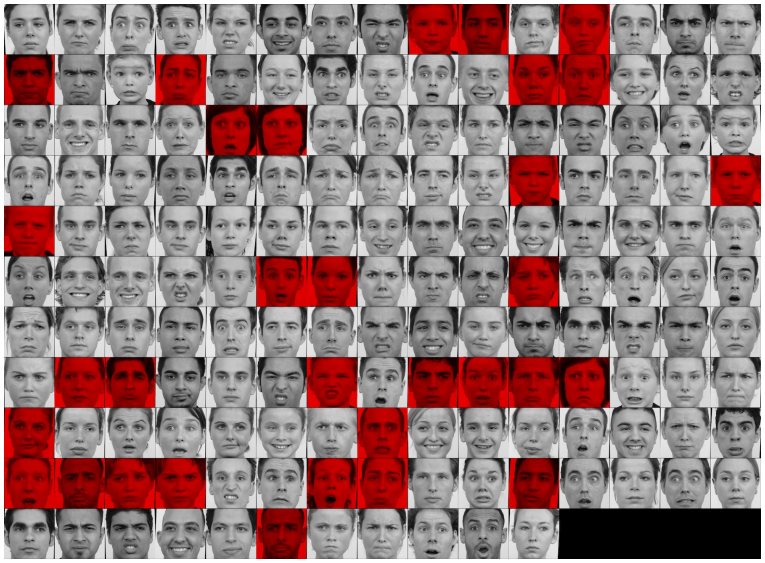


Figure 6.11: SVM Classification (C=12, γ=0.1)

	Neutral	**Anger**	**Contm**	**Disg**	**Fear**	**Happy**	**Sad**	**Surpr.**	**Avg**
Neutral	24	0	2	0	1	0	0	0	0.889
Anger	0	20	0	0	5	0	0	0	0.800
Contm	2	0	14	0	4	0	0	0	0.700
Disg	0	0	0	14	1	0	0	0	0.933
Fear	0	0	0	0	15	0	2	0	0.882
Happy	0	0	0	0	0	17	0	0	1.000
Sad	0	1	1	0	8	0	9	0	0.563
Sad	0	0	0	0	5	0	0	16	0.762

Table 6.6: SVM - Training and Validation Accuracies

7 Conclusion

The recognition of facial expressions of human emotional state has been formulated as an image classification problem over 7 (or 8) different categories. Two separate techniques have been applied to solving the problem: decision by convolutional neural networks and decision by SVM using HOG as a feature encoder for the images. While both techniques show promising results, the preference of these authors is for the SVM technique. The amount of time and computing power required to accurately train and test the CNNs in this task is ENORMOUS. Training the above deep neural net on the relatively small JAFFE dataset required 500 epochs to achieve meaningful accuracy, and the time to complete each epoch was on average 175s. That means that to train the machine for 500 epochs took over 24 hours, and one can assume that the time to train it another 500 epochs would have taken another 24 hours. Considering that returns tend to diminish near the higher accuracy rates for a CNN,

it is not unreasonable to assume that the deeper neural net would have taken up to or more than 72 hours to train to a reasonable accuracy rate, and this on a small dataset. Training the deeper neural net on the larger data set took almost 60 minutes per epoch. Training for 1000 epochs in this case is clearly time prohibitive without the assistance of a GPU machine or a super computer. Without access to high power computing resources, the HOG+SVM technique is much more appropriate.

8 Reference

[1] Langner, O., Dotsch, R., Bijlstra, G., Wigboldus, D.H.J., Hawk, S.T., and van Knippenberg, A. (2010). Presentation and validation of the Radboud Faces Database. Cognition and Emotion, 24(8), 1377—1388. DOI: 10.1080/02699930903485076

[2] The Japanese Female Facial Expression (JAFFEE) dataset, [Online]. Access: http://www.kasrl.org/jaffe.html

[3] J. Pao, "Emotion Detection Through Facial Feature Recognition," Stanford University, 2016.

[4] Histogram of Oriented Gradients [Online]. Access: https://www.learnopencv.com/histogram-of-oriented-gradients/

[5] Juliano E. C. Cruz, Eleio H. Shiguemori, and Lamartine N. F. Guimaraes, "A comparison of Haar-like, LBP and HOG approaches to concrete and asphalt runway detection in high resolution imagery", Sao Jose dos Campos, SP, Brazil, 2015.