Recognizing facial expressions from videos using Deep Belief Networks

CS 229 Project
Advisor: Prof. Andrew Ng

Adithya Rao (adithyar@stanford.edu), Narendran Thiagarajan (naren@stanford.edu)

***ABSTRACT:* Deep learning techniques have been shown to perform well for problems such as image classification and handwriting analysis. In this project we aim to apply these deep learning techniques to recognize facial expressions from videos. We employ sparse feature selection to improve the efficiency and accuracy of the classification [5]. We begin by considering images, and extend the algorithms to classify videos. We first train a sparse autoencoder for a known set of images, to verify the correctness of implementation of the encoder. We then pre-train the autoencoder with our dataset of facial expressions, and use the weights to classify test images. We use softmax and logistic regression for classifying. The results show that the individual expressions are classified with high accuracy, and the performance further improves by including FACS and tracking labels.**

I. INTRODUCTION

Deep learning involves learning a hierarchy of internal representations using a stack of multiple modules, each of which is trainable and can implement complex non-linear functions. Deep learning allows us to go from low level representations to high level representations and is more efficient and accurate than shallow architectures such as kernel machines. A deep architecture achieves this efficiency by trading space for time and using sparse feature encoding [5]. Deep learning techniques have been shown to perform significantly better than other techniques for problems such as image classification and handwriting analysis [4].

Here we apply deep learning for recognizing facial expressions in videos. Previous work in [1] and [2] addressed the problem of generating facial expressions with images as inputs. We aim to extend this work to automatic recognition of facial expressions from video frames instead of images. Related work in [3], used AdaBoost and SVMs for feature selection and classification respectively from video sequences, whereas our approach would use DBNs. Using unsupervised deep learning, the sparse autoencoder will automatically choose a set of high level weights, which removes the necessity to perform feature selection separately. Deep belief networks are very suitable to this problem because they have shown promise in machine learning problems that involve human perception such as vision, audio, touch etc.

II. PRELIMINARY STEPS

2.1 Sparse autoencoders

We began by writing the algorithm for the sparse autoencoder using MATLAB and followed the guidelines provided in the CS294 course website. We implemented a neural network with an input layer having 64 units, a hidden layer having 30 units and the output layer with 64 units.

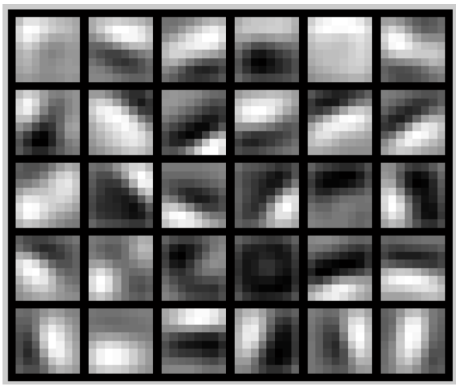


Fig.1. Weights generated by the sparse autoencoder for the data from the CS294 course website.

The algorithm (explained in [9]) performed the following steps:

(i) Run a feedforward pass on our network on input images, to compute all units' activations.
(ii) Perform one step of stochastic gradient descent using backpropagation
(iii) Perform the updates to employ sparsity constraints.

The weights that we obtained using the assignment data are shown in Fig 1. The output weights are seen to closely resemble the expected output. This validates the correctness of our implementation of the autoencoder.

2.2 Facial Image Data

Dataset: As a step towards analyzing facial expression from videos, we first try to classify facial expressions from images. For this purpose we initially used the facial expression database from [7]. This dataset consists of faces showing 8 different classes of emotion, namely -- surprise, fear, disgust, contempt, happiness, sadness, anger and neutral expressions.

Preprocessing: We used partially preprocessed images from [8], where the images have been converted to grayscale and an oval mask has been applied to them, so that only the information that contributes to recognizing the expression remains. Thus, the background and the hair are not considered. The images have also been resized to smaller dimensions. Fig. 2 shows the images before and after preprocessing.

We applied the sparse autoencoder to the facial image data, to obtain the corresponding weights. We tuned our parameters alpha = 0.2, beta = 5, lambda = 0.002 and ran the algorithm for 4 million iterations. Among the different combinations of parameters that we tried, the above values seemed to work best.

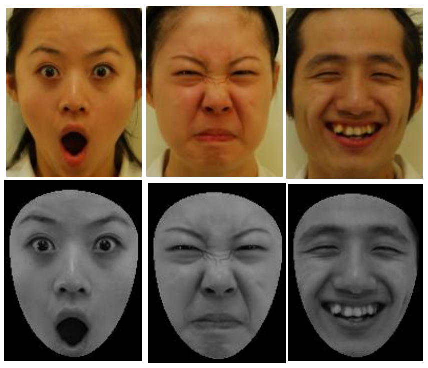


FIG 2. The images above are from the dataset [7], and the images below show the same after preprocessing

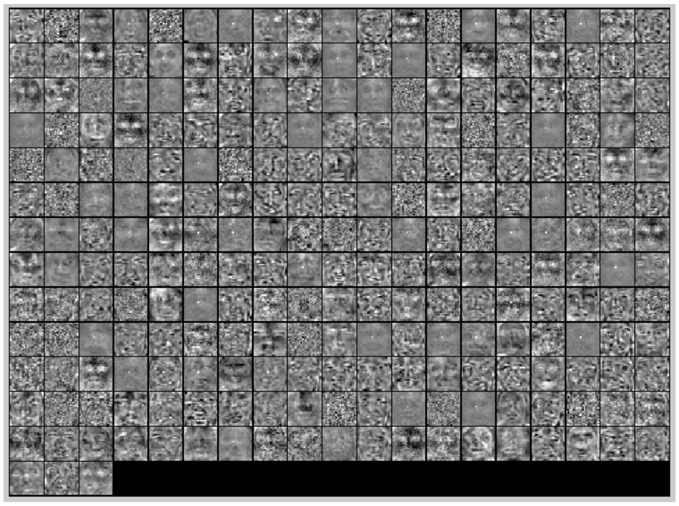


Fig. 3. Weights obtained after running the sparse encoder on preprocessed facial image data

III. METHODOLOGY

3.1 Facial Expression dataset

We use the Cohn Kanade DFAT-504 dataset [11] which is an AU-Coded Facial expression database. It has both posed (about 593 sequences) and non-posed (spontaneous) expressions with validated emotion metadata. A total of 7 expressions are present in the dataset. The target expression for each sequence is fully FACS coded.

3.2 Preprocessing

We performed the following steps for preprocessing the data:
a) Instead of applying an oval mask to the image as in the preliminary step, we extracted face patches from the dataset using the automatic Viola-Jones face detector package [12].
b) The extracted face patches had varying lighting conditions, and needed to be normalized so that we avoid learning lighting features at the expense of face details. To normalize the data, we first resized each image to 24 x 24 image patches using a bilinear transformation. The minimum and maximum pixel intensities for each image were calculated, and each image was scaled accordingly such that the pixel values were now between -1 and +1. Thus all the images in the dataset were resized and normalized, to be provided as input to the autoencoder.

3.3 High level feature extraction

In our model, 576 soft binarized pixel inputs are connected to a hidden layer of 250 logistic units, which are trained by our sparse autoencoder to match the output to the input. We tried different architectures for the neural net, which are explained in more detail in the next section. The feed-forward pass and the back-propagation of errors were performed for each iteration of the stochastic gradient descent, and sparsity constraints were employed. This resulted in the weights shown in Fig 3. Some weights indicate local structure useful for

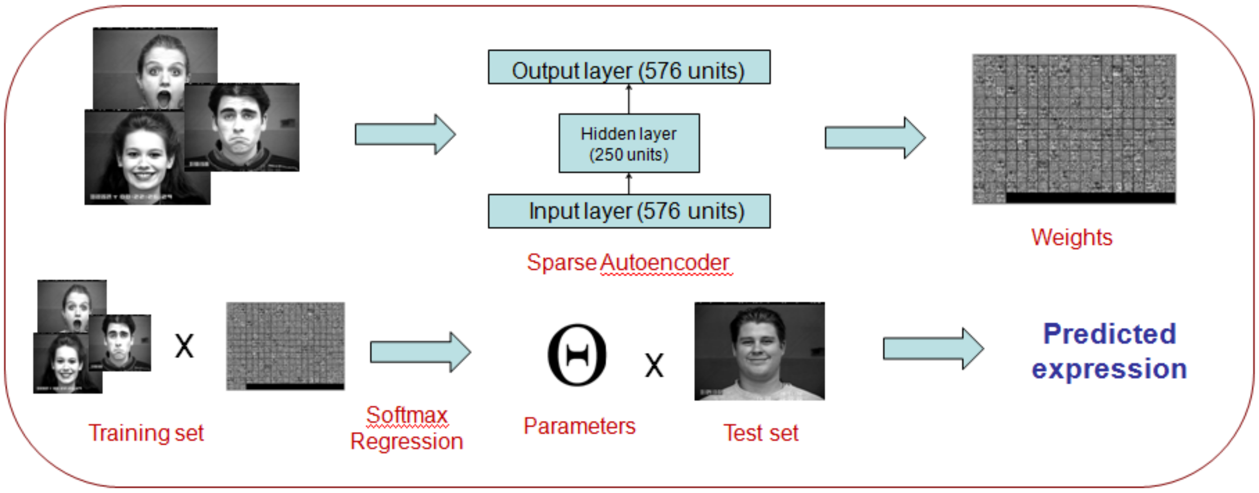


Fig 4. Methodology for feature extraction and classification.

representing distinct features and edges, while others show more global structure.

3.4 Learning the joint distribution of FACS labels and features

For classifying the images, we first use 1-of-K softmax regression to represent the identities of the facial expressions. After obtaining the weights, the feature activities for each image were concatenated with discrete FACS labels and tracking labels. The combined vector is then the input to our softmax classifier. A face image can only be associated with a single identity, and each identity label is set to 1 with probability:

$$p(ID_i = 1) = \frac{\exp(b_i + \sum_{j\in\text{features}} s_j w_{ij})}{\sum_{k\in\text{identities}} \exp(b_k + \sum_{j\in\text{features}} s_j w_{kj})}$$

Where s is our input vector including the FACS labels, and w are the parameters of the softmax regression problem. We split our dataset into training and test data, and train our softmax layer to estimate the parameters using gradient descent. Once the parameters have been estimated, we input a test image to the softmax layer, which will calculate the probabilities for each identity. The expression with the highest probability is chosen as the identity of the test image.

Apart from softmax regression, we also use logistic regression to classify individual expressions. Fig 4. shows the entire methodology described above.

IV. Experiments and Results

4.1 Choosing the right autoencoder

The first step in the pipeline is to identify the features using autoencoders. Our first approach is to use the autoencoder from our preliminary experiments (Sec 2.1) which consists of a single hidden layer with 250 units. The reason for picking 250 is to extract the most important features across images of all expressions. We ran the autoencoder for different sets of alpha, beta parameters for a large number of iterations and we identified the best set of weights (Fig 3).

In other variants of the autoencoder, we tried with 2 hidden layers (containing 250 and 100 units respectively), because for a given DBN adding a new hidden layer could improve the performance. Previous work in [1] used two hidden layers, with the number of hidden units very close to the number of input units. We also set the number of hidden units to 500 and measured the weights. However, we observed that adding a hidden layer or increasing the number of hidden units did not lead to a marked improvement in the resulting weights.

4.2 Feature activations for each expression

The sparse autoencoder activates different neurons for each expression. This is visualized in Fig.5, where the activations for surprise and anger are compared by plotting the difference in their feature activations. It is clearly seen from the figure that some neurons have higher activations for surprise (positive spikes), while some are more active for anger (negative spikes).

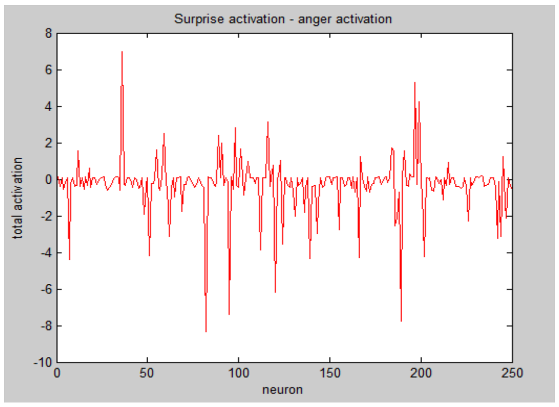


Fig 5. Comparison of neuron activations for Surprise and Anger. Positive spikes indicate neurons active for surprise and negative spikes indicate neurons active for anger.

4.3 Choosing the Classifier

We picked the best set of weights from the autoencoder and then trained a Softmax layer using our dataset. The training set size was approximately 90 percent of the entire dataset size (training set size – 1315, test set size – 150). With just the feature activations as input to the softmax layer, the resulting accuracy is 0.22. The accuracy of a random guessing algorithm is of 0.14 (7 possible output classes). Also the algorithm classified majority of the test cases as disgust and surprise and worked poorly for fear and anger.

In order to improve the accuracy of the predictions, we applied logistic regression to individual expressions. By doing so, the accuracy greatly improved, and the correctly predicted expressions increased significantly. Table 1 shows the resulting values for each expression.

4.4 Choosing the input vector

Given our extracted features and classifier, we experimented with different input vectors to the classifier. We observed that including FACS labels and tracking values along with the feature activations from the autoencoder gave better results than using feature activations alone. This is also consistent with the observations in [1]. Fig. 6. shows the comparison between the accuracy values for both cases. Fig 7. shows the TPR vs FPR scatter plots for both input vectors, where a particular point represents a particular expression. When the FACS data is included, most of the points lie in the upper left corner of the plot, indicating better performance.

4.4 Using video sequences

Extending from images to video sequences, we used our model to predict expressions for sequences of images, showing the peak expression for a particular face. Fig 8 shows the probabilities for a sequence of five images for a particular expression. The model correctly predicts a high probability for one of the expressions and low probability for the others. Classifying sequences of images leads to a better performance of the model, rather than classifying an individual image alone. Including a sequence of images would also help in identifying transitions of expressions.

Table 1

Label	Accuracy	TP	TN	FP	FN
Anger	0.77	11	104	21	14
Disgust	0.91	12	124	1	13
Fear	0.89	14	120	5	11
Happiness	0.90	21	114	11	4
Sadness	0.76	15	99	26	10
Surprise	0.87	17	113	12	8

a) Input: Feature Activations + FACS data

Label	Accuracy	TP	TN	FP	FN
Anger	0.55	18	65	60	7
Disgust	0.81	14	107	18	11
Fear	0.87	10	120	5	15
Happiness	0.82	12	111	14	13
Sadness	0.77	1	114	11	24
Surprise	0.75	19	94	31	6

b) Input: Feature activations only

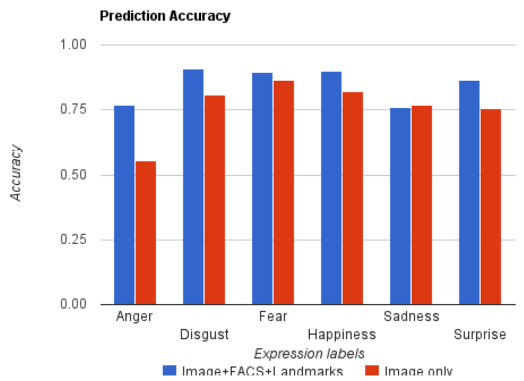


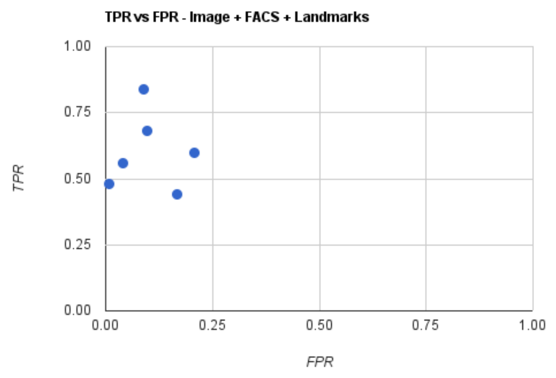
Fig 6. Prediction accuracy for feature activations + FACS labels + tracking data, and for feature activations only.

V. CONCLUSIONS AND FUTURE WORK

Our experiments and results show that deep learning and logistic regression indeed produce good predictions for recognition of facial expressions. Increasing the number of hidden layers and the number of units per hidden layer did not seem to significantly improve performance. An interesting problem in this direction would be to investigate techniques for estimating the required number of hidden layers and units, for a given image recognition dataset.

Further, our experiments also show that including FACS labels along with the feature activations significantly improves performance. Classifying sequences of images leads to better performance than classifying individual images. This indicates that recognizing expressions from videos is more accurate than recognizing them from a single image. These results suggest that for such problems involving facial expressions, it may be helpful to have FACS data available along with the image and video data. Currently, high quality automatic FACS labeling with coded AUs are available publicly only for a small number of datasets. Thus, developing an automated method for FACS labeling is an important challenge that could be pursued in the future.

Another extension to this work would be to attempt to combine audio data with facial expressions, in order to understand the effect of audio-visual cues used in videos to convey emotions.



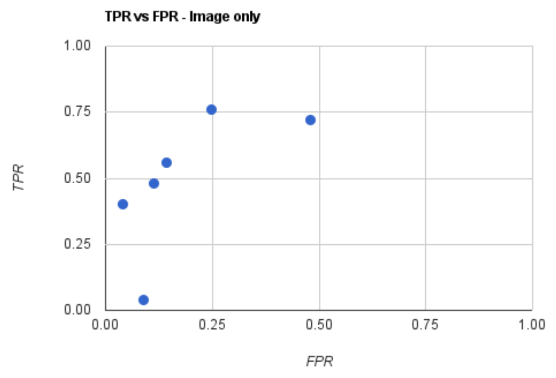


Fig 7. TPR vs FPR

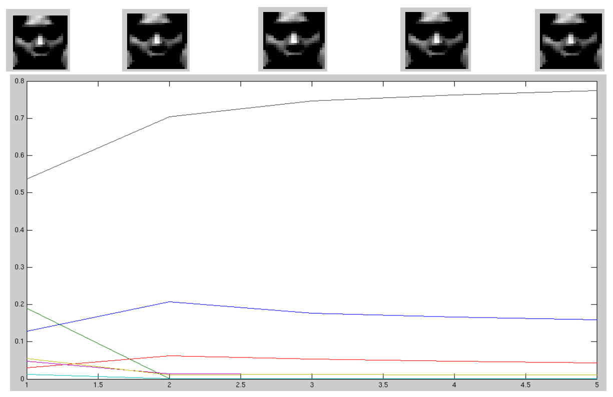


Fig 8. Probabilities of each expression for a sequence of images

VI. References and Links

[1] J.M. Susskind, G.E. Hinton, Javier R. Movellan and Adam K. Anderson, "Generating Facial Expressions with Deep Belief Nets," Affective Computing, 2008, pp. 421-440.

[2] Sabzevari et al., "A Fast and Accurate Facial Expression Synthesis System for Color Face Images Using Face Graph and Deep Belief Network"

[3] Bartlett M.S., Littlewort G., Frank M., Lainscsek C., Fasel I., Javier Movellan "Recognizing Facial Expression: Machine Learning and Application to Spontaneous Behavior"

[4] Lee H., Grosse R, Ranganath R, Ng A., "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations" Proceedings of the Twenty-Sixth International Conference on Machine Learning, 2009.

[5] Ranzato M, Boureau Y., LeCun Y., "Sparse Feature Learning for Deep Belief Networks"

[6] D. Liu, L. Lu, and H.-J. Zhang. Automatic mood detection from acoustic music data. In proceedings of the International Symposium on Music Information Retrieval (ISMIR'03), Baltimore, MD, USA, October 26-30 2003.

[7] Li-Fen Chen and Yu-Shiuan Yen. (2007). Taiwanese Facial Expression Image Database [http://bml.ym.edu.tw/download/html]. Brain Mapping Laboratory, Institute of Brain Science, National Yang-Ming University, Taipei, Taiwan.

[8] Agarwal N, Cosgriff R., Mudur R., Mood Detection: Implementing a facial expression recognition system. (CS229 project, 2009).

[9] Andrew Ng., CS294A Lecture notes – Sparse Autoencoders.

[10] http://vasc.ri.cmu.edu/idb/html/face/facial_expression/

[11] Lucey P., Cohn J.F., Kanade T., Saragih J., Ambadar Z., "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression", Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on, June 2010.

[12] Viola P., Jones M., "Robust Real-time Face Detection", International Journal of Computer Vision, 2004.