Emotion Recognition with Deep-Belief Networks

Tom McLaughlin, Mai Le, Naran Bayanbat

Introduction

For our CS229 project, we studied the problem of reliable computerized emotion recognition in images of human faces. First, we performed a preliminary exploration using SVM classifiers, and then developed an approach based on Deep Belief Nets. Deep Belief Nets, or DBNs, are probabilistic generative models composed of multiple layers of stochastic latent variables, where each "building block" layer is a Restricted Boltzmann Machine (RBM). DBNs have a greedy layer-wise unsupervised learning algorithm as well as a discriminative fine-tuning procedure for optimizing performance on classification tasks. [1].

We trained our classifier on three databases: the Cohn-Kanade Extended Database (CK+) [2], the Japanese Female Facial Expression Database (JAFFE) [3], and the Yale Face Database (YALE) [4]. We tested several different database configurations, image pre-processing settings, and DBN parameters, and obtained test errors as low as 20% on a limited subset of the emotion labels.

Finally, we created a real-time system which takes images of a single subject using a computer webcam and classifies the emotion shown by the subject.

Part 1: Exploration of SVM-based approaches

To set a baseline for comparison, we applied an SVM classifier to the emotion images in the CK+ database, using the LIBLINEAR library and its MATLAB interface [5]. This database contains 593 image sequences across 123 human subjects, beginning with a "neutral "expression and showing the progression to one of seven "peak" emotions. When given both a neutral and an expressive face to compare, the SVM obtained accuracy as high as 90%. This section summarizes the implementation of the SVM classifier. For additional details on this stage of the project, please see our Milestone document.

Part 1.1 Choice of labels (emotion numbers vs. FACS features)

The CK+ database offers two sets of emotion features: "emotion numbers" and FACS features. Emotion numbers are integer values representing the main emotion shown in the "peak emotion" image. The emotions are coded as follows: 1=anger, 2=contempt, 3=disgust, 4=fear, 5=happiness, 6=sadness, and 7=surprise.

The other labeling option is called FACS, or the Facial Action Coding System. FACS decomposes every facial emotion into a set of Action Units (AUs), which describe the specific muscle groups involved in forming the emotion. We chose not to use FACS because accurate labeling currently requires trained human experts [8], and we are interesting in creating an automated system.

Part 1.2 Features

Part 1.2.1 Norm of differences between neutral face and full emotion

Each of the CK+ images has been hand-labeled with 68 standard Active Appearance Models (AAM) face landmarks that describe the X and Y position of these landmarks on the image (Figure 1).

Figure 1. AAM Facial Landmarks

We initially trained the SVM on the norm of the vector differences in landmark positions between the neutral and peak expressions. With this approach, the training error was approximately 35% for hold out cross validation (see Figure 2).

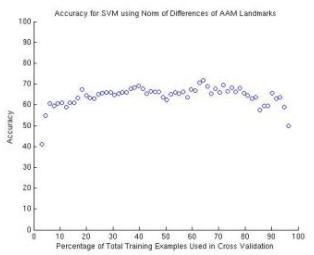

Figure 2. Accuracy of SVM with norm-displacement features.

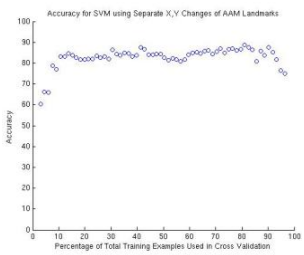

Figure 3. Accuracy of SVM with separate X, Y displacement features.

Part 1.2.2 Separate X and Y differences between neutral face and full emotion

Because the initial approach did not differentiate between displacements of landmarks in different directions, we also provided the differences in the X and Y components of each landmark separately. This doubled the size of our feature vector, and resulting in a significant (about 20%) improvement in accuracy (Figure 3).

Part 1.2.3 Feature Selection

Finally, we visualized which features were the most important for classifying each emotion; the results can be seen in Figure 4. The figure shows the X and Y

displacements for each emotion compared to the average values over all emotions, with average features colored in green and both extremes colored red and blue, respectively. One interesting result is that the Y features are in general more important—there are fewer side-to-side motions of facial features during the formation of expressions. Also, "surprise" showed by far the most extreme displacements for many features. This exercise helped us understand which aspects of our images contain the most information about emotion, and so was valuable for designing our approach to the DBN.

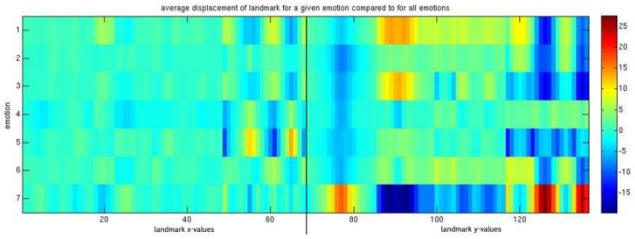

Figure 4. Visualization of relative feature importance.

Part 1.2.5 Limitations of AAM landmarks

While the SVM trained on AAM facial landmark provided promising results, it has a fatal flaw: there is not yet an accurate, automated method for identifying the landmarks on a human face. Because we want to make a fully automated emotion classifier, we had to abandon AAM landmarks. Fortunately, DBNs can be trained on raw images (and actually performed better with these inputs than if they were given the AAM features.)

Part 2: Deep Belief Nets

DBNs are probabilistic generative models composed of hidden stochastic variables and are similar in structure to neural nets. However, unlike other neural nets, DBNs perform learning one layer at a time, computing generative weight matrices that define connections between the nodes of every two adjacent layers. Once the weighted matrices are calculated, the hidden variables at each layer can be inferred from the visible input by reversing the matrices.

Our DBN implementation is based upon a modification of Geoff Hinton's code for classifying the MNIST handwritten digit database [7]. The two problems are actually very similar, since both involve training a DBN on a raw image and sorting the training examples into 5-10 classes. We hoped that, as in the digit recognition case, the DBN algorithm's ability to identify complex patterns in the input would yield high accuracy in our classification task.

This section describes the steps we took in setting up and configuring our DBN-based system, and concludes with test results.

Part 2.1 Face detection and image preprocessing

We used OpenCV's Haar feature detection algorithms to create an image preprocessing program to prepare the images in our databases. There are 3 key steps in the processing chain: 1) identify and crop a face in the image, 2) identify the eyes, and use their location to "letterbox" the left and right sides of the face, and 3) perform histogram equalization. The letterboxing serves to hide areas of the image that don't correlate with emotion, such as the ears and hair. Equalization serves to sparsify the image matrix and emphasize the features of interest, as well as ensure that all our input data has the same contrast.

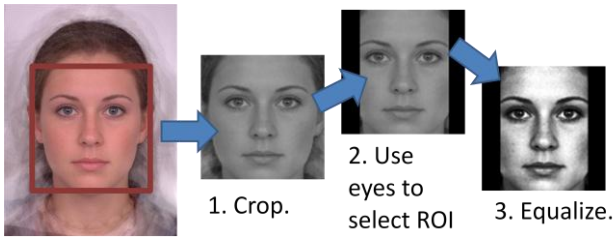

Figure 5. Image pre-processing steps

As it turned out, equalizing the data was extremely important, but the letterboxing technique actually degraded performance (see Results).

An important image preprocessing question is the size to which the input images should be scaled. The Hinton paper on digit recognition used 28 by 28 pixel inputs. However, human faces are much more complex than handwritten digits, so higher resolution is required to avoid losing important details. Excessively high resolution, on the other hand, would greatly increase the running time of the algorithm and could actually degrade accuracy, due to the decreased proportion of "relevant" data in the image. With this in mind, we chose to scale our input images to 100 by 100 pixels.

Part 2.1: Parameter selection

The two most important parameters of a DBN are a) the number of layers and b) the number of hidden variables in each layer. The default settings in the Hinton code are to use 4 layers, containing 500, 500, and 2000 latent variables, with the final hidden layer variables corresponding directly to our classes.

In general, the more layers present in the DBN, the better performance will be. Networks with insufficient depth can require more computational elements than architectures whose depth is well-matched to the task, because deeper networks with fewer hidden variables can provide a simpler, more descriptive model [6]. The problem with deep nets is that they are often harder to optimize, and the best performance is often determined empirically.

While it is agreed that adding more layers to the network will enhance performance, beyond a certain threshold the payoff of adding more layers is no longer worth the additional time and computing resources needed to run the DBN. In addition to the standard 4 layers in the Hinton code, we also tested the DBN with 5 and 6 layers. Since human faces are much more complex than handwritten digit images, we decided not to try decreasing the number of layers.

There are many "rules of thumb" for determining the optimal number of hidden variables per layer. In general, the number of hidden variables required depends in a complex way on the size of the input and output layers, the number of training examples, the amount of noise in the data, and other factors. Increasing the number of hidden units too much would make the network prone to over-fitting, while not enough hidden units could mean under-fitting and the network maybe unable to capture necessary higher order features. We chose to try several settings and determine the best performance empirically.

Part 2.2: Results

Part 2.2.1 Determining convergence/over-fitting

Increasing back-propagation duration may result in over-fitting. When allowed to run for a sufficiently long time, the training error decreases steadily while the test error remains constant or increases (Figure 6). For this reason, it is important to be careful how about when we terminate the back-propagation phase. We initially ran our tests for a constant 200 epochs, but later found that the time at which over-fitting begins is dependent on the specific settings of the test run. In all the results reported in this paper, we allowed each test run to proceed until over-fitting was observed. The number of testing epochs this required was recorded, and repeated runs with the same settings were run for the same number of epochs. In many cases this took as many as 500 epochs.

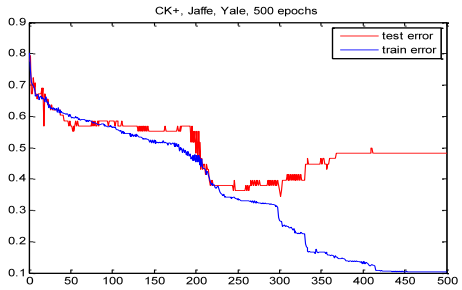

Figure 6. DBN over-fitting.

Part 2.2.1 Testing histogram equalization, letterboxing, limited emotion sets

We began with a series of tests to determine the best way to pre-process our input images. The results show that equalization creates dramatic improvements in performance (up to 20% less training error). See Figure 7 for a comparison of learning curves obtained by training on all three databases together, with equalization both on and off.

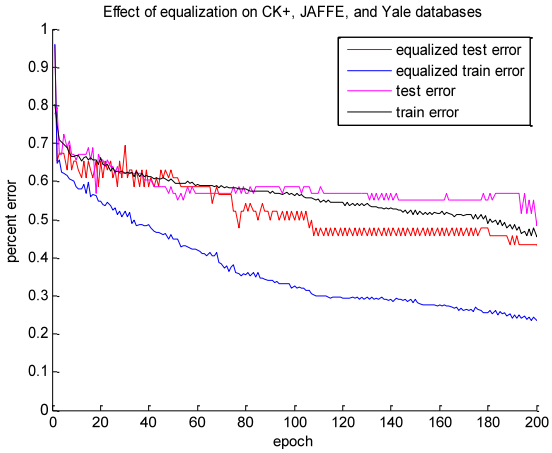

Figure 7. Effect of equalization on test error.

Table 1. Effect of histogram equalization and letterboxing on test accuracy

Databases included	Non-equalized	Equalized	Equalized, letterboxed
CK+	0.39	0.33	0.34
CK+, Yale	0.49	0.30	0.31
CK+, Yale, JAFFE	0.55	0.27	0.28

On the other hand, our "letterboxing" technique of isolating only the main part of the face actually degraded performance somewhat. It's possible that this is because the parts of the face which it excluded (such as the outline of the cheek and cheekbone) were more important than we had thought for classification, although further study is necessary to be sure.

Part 2.2.2 Limited emotion and training sets

We observed both in our SVM explorations and in our early tests of DBNs that certain emotions in our databases, especially contempt and disgust, were a) very difficult for the algorithm to tell apart (difficult for humans as well, in some cases) and b) presented themselves differently in different subjects. We hypothesized that excluding one or more of these emotions would improve our test accuracy. We trained and tested on two different subsets of the emotion labels: the full 7-emotion set, and a limited 4-emotion set containing only the emotions happy, sad, surprised, and angry.

Table 2. Effect of test error when training on a limited emotion set		
	7 Emotions	**4 Emotions**
CK+	0.57	0.22
CK+, Yale	0.29	0.22
CK+, Yale, JAFFE	0.41	0.20

As expected, using the limited emotion set produced improved results. (See Figure 8 for sample learning curves.)

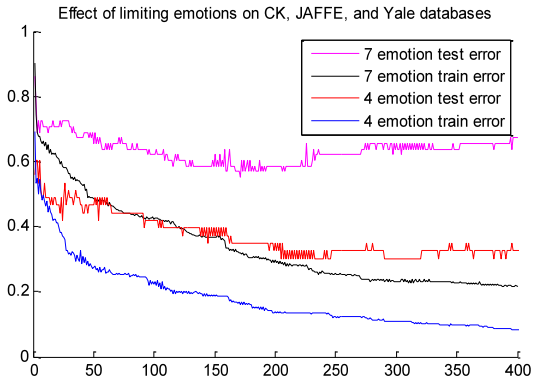

Figure 8. Learning curves when training on limited emotion set.

Part 2.2.3 Varying DBN depth and number of hidden variables

We also experimented with varying the number of hidden layers and latent variables. Figure 9 shows a comparison of the learning curves for a 4-layer, 5-layer, and 6-layer DBNs.

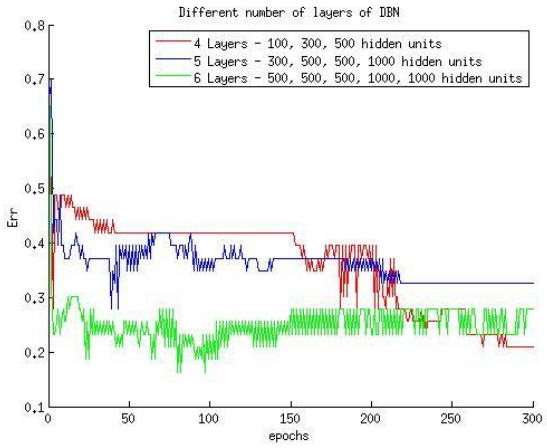

Figure 9. Learning curves with varied DBN depth.

Adding more depth generally improved the performance, with the 6 layer DBN outperforming all other configurations. In particular, the deeper DBN converged much faster to a good solution. However, the 5 and 6 layer configurations took a much longer time to train, so they do have their drawbacks. For this reason, we did not test numbers of layers greater than 6.

Finally, we explored what happens when you vary the number of latent variables in the model. While it is difficult to determine what the optimal configuration is for a given total number of hidden units, it seems clear that increasing the number of latent variables beyond a certain amount decreases performance significantly. This is reasonable, because an excess of hidden variables makes the model more prone to over-fitting. Figure 10 shows the effect of varying the number of hidden variables on a 4-layer DBN.

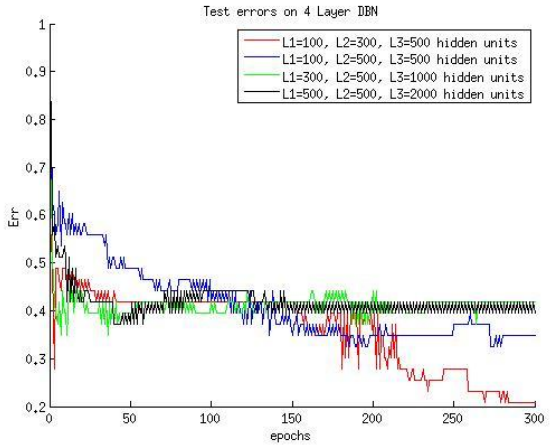

Figure 10. Learning curves with varied number of hidden variables (4-layer DBN)

As might be expected, this over-fitting phenomenon was even more pronounced in a 5- or 6-layer DBN with a large number of hidden variables (Figure 11).

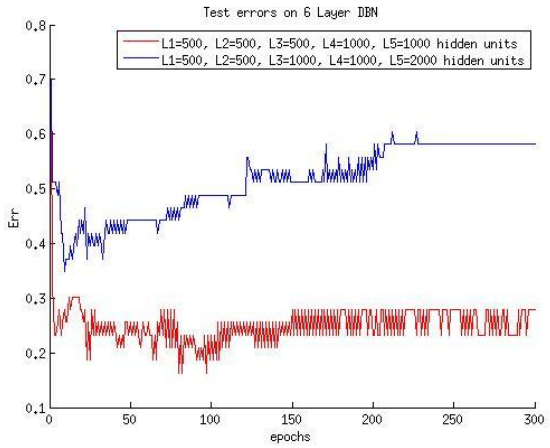

Figure 11. Effect of varying number of hidden variables on a 6-layer DBN.

Part 3 Application: our real-time classifier

To conclude our project, we created a real-time demo system which enables the user to look into a webcam, push a button, and have their picture taken and their emotion classified (See Figure 12). This system used the classifier trained on the 4-emotion dataset, using all three databases. When used in proper lighting conditions, it was surprisingly accurate. (Even in our poorly-lit area of the CS229 poster session, several volunteers had their emotions classified correctly every time.)

This system is a "proof of concept," and improvements of it could lead to applications such as those discussed in our initial project proposal: emotion-aware software, point-and-shoot cameras with "happiness detection," or helping train autistic individuals in recognizing and producing emotional cues.

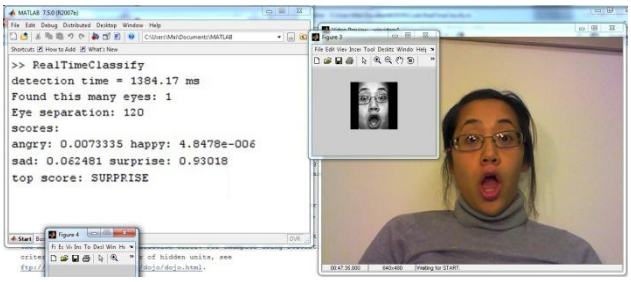

Figure 12. Real-time demo system.

Part 4: Conclusion and future directions

In conclusion, histogram equalization during the image preprocessing step produced a clear improvement in test accuracy. The "letterboxing" technique we attempted, however, did not produce improved results, suggesting that it may be best to give the DBN as much data as possible and not try to guess which parts of the image are most important.

As expected, our accuracy was best (around 20%) when testing on the limited set of 4 emotion classes, where the more ambiguous emotions were eliminated. (We would note that a classifier working with only these 4 emotion classes covers the most common emotions, and would still be useful for many practical applications.)

Increasing the number of DBN layers caused the algorithm to perform better; in particular, it converged to good training error much faster than DBNs with smaller layers. Using more hidden variables, on the other hand, resulted in generally worse performance due to over-fitting. The effect was especially prominent when there were both many layers and many hidden variables.

The learning curves from our data show that combining our different databases results in the best performance. We suspect that further performance gains could probably be realized by increasing the size of our training database. Indeed, the MNIST handwritten digit database contains 60,000 images and 10,000 test images, or 7,000 images (on average) for each of the 10 digit classes. Due to the difficulty of obtaining good emotion images, we had only 525 training examples, or 75 examples per emotion, when using all three databases.

References

[1] Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18(1527-1554).

[2] Lucey, P.; Cohn, J.F.; Kanade, T.; Saragih, J.; Ambadar, Z.; Matthews, I.; , "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* , vol., no., pp.94-101, 13-18 June 2010. doi: 10.1109/CVPRW.2010.5543262.

[3] Michael J. Lyons, Shigeru Akamatsu, Miyuki Kamachi, Jiro Gyoba. Coding Facial Expressions with Gabor Wavelet. *Proceedings, Third IEEE International Conference on Automatic Face and GesturRecognition*, April 14-16 1998, Nara Japan, IEEE Computer Society, pp. 200-205.

[4] Georghiades, A., Belhumeur, P., Kriegman, D.: From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose. *IEEE Trans. Pattern Anal. Mach. Intelligence* 23(6), 643-660 (2001)

[5] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research 9(2008), 1871-1874. Software available at http://www.csie.ntu.edu.tw/~cjlin/liblinear

[6] S. Lee, R. Xiang, S. Cetintas, Y. Fang. Deep Belief Nets: CS590M Paper Presentation. Fall 2008. Department of Computer Science, Purdue University.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998.

[8] Michael J. Lyons, Shigeru Akamatsu, Miyuki Kamachi, Jiro Gyoba. Coding Facial Expressions with Gabor Wavelet. *Proceedings, Third IEEE International Conference on Automatic Face and Gesture Recognition*, April 14-16 1998, Nara Japan, IEEE Computer Society, pp. 200-205.