Emotion Detection in Images

Jacob Pollard, Alex Romriell, David Wen

***Abstract*— We implemented a pattern recognition framework to recognize emotion in static facial expressions. Our methods are based on the paper "Hierarchical Committee of Deep CNNs with Exponentially-Weighted Decision Fusion for Static Facial Expression Recognition" [1]. The paper's authors achieved an accuracy of 61.6% classifying all seven basic emotions. We built a convolutional neural network (CNN) and classified photos with an accuracy of 67.67%. In addition to classifying the seven emotions, we built CNNs and support vector machines (SVMs) to classify facial images as expressing one of two emotions. While we trained and tested models for all pairs of emotions, our best models were trained on images labeled as "Happy" or as "Disgust". Our CNN model achieved 96.69% test accuracy and our SVM achieved 94.4% test accuracy.**

I. INTRODUCTION

The primary goal of this project is to train a model to detect and classify emotion from a photograph of a face. We followed the work of Bo-Kyeong Kim, et al. [1] to build our initial model. Kim's team won first prize in the "Third Emotion Recognition in the Wild Challenge (EmotiW 2015)"[4] with a classification accuracy of 61.6%.

We built a convolutional neural network to classify the seven basic emotions: "Anger", "Disgust", "Fear", "Happy", "Unhappy", "Surprise", and "Neutral". Over the course of 30 hours, we achieved 67.67% test accuracy using this model. However, we mainly focused on building models to classify pairs of polarizing emotions such as "Happy" vs. "Unhappy" and "Happy" vs. "Disgust". We built both CNN models and SVMs to classify an image as having one of two polarizing emotions. These models classified pairs of emotions with accuracies as low as 63% and as high as 97%.

II. DATA DESCRIPTION

We retrieved our data from three sources:

- the Static Facial Expression Recognition data set from the EmotiW 2015 Challenge [4]
  - 1,322 images, dimensions: 143×181, format: PNG
- The Japanese Female Facial Expression (JAFFE) Database [5]
  - 213 images, dimensions: 256×256, format: TIFF
- Challenges in Representation Learning: Facial Expression Recognition Challenge [6]
  - 35,887 images, dimensions: 48×48, format: CSV

These three sources comprised a total of 37,422 images. We converted each data set into PNG file format and scaled each picture to 48 x 48 pixels.

Re-scaling the images to 48×48 pixels and converting the TIFF files to PNG files were straightforward tasks. The CSV images, however, required more attention. Each row in the CSV had the following form:

- Emotion: An integer between 0 and 6 corresponding to one of the seven classes of emotions
- Pixels: An array of length 2,304 (48×48) representing each pixel in the image

Since each row contained an array of pixels representing each image, we wrote a script to divide the array after every 48th element to create a 48×48 matrix of pixel values. This matrix was then converted into a PNG image.

We renamed each image file. Every file was prepended with the letter corresponding to the emotion being expressed in that file, followed by a unique six-digit number (e.g."A000791.png").

Convolutional Neural Networks generally need large data sets in order to learn distinguishing features. To increase the size of our data set, we added random perturbations to our set of 37,422 images. Specifically, we applied 3 different perturbations, at random, to each image. These perturbations were flipping the image, blurring the image, and denoising the image. Adding the perturbed images

doubled our data set from 37,422 images to 74,844 images. After this, we sharpened all images to help with edge detection and feature extraction.

III. Exploratory Analysis and Feature Engineering

Fig. 1: Six Images Each With Six Corresponding Example Feature Maps

For the CNN models, we did not perform any explicit feature engineering. As the model is being trained, the kernel filters learn and extract features by design. (Convolutional Neural Networks are intended to reduce the need for manual human engineering.) An example of six of the kernel filters trained in the first convolutional layer of our model applied to six images not in our data set are in Figure 1.

We did, however, do feature engineering for the SVM model. We took the first 15 principal components of each image and used these as features to train our SVM model. We chose the first 15 principal components because, on average, the first 15 components explained more than 95% of the variance in each image (Figure 2). Training the SVM model on principal components, instead of the full image, reduced the run-time by more than half with only a slight decrease in test accuracy. This increase in speed is achieved by reducing the dimensions of the image to one-third of it's original size, while maintaining nearly all of the important data in the image.

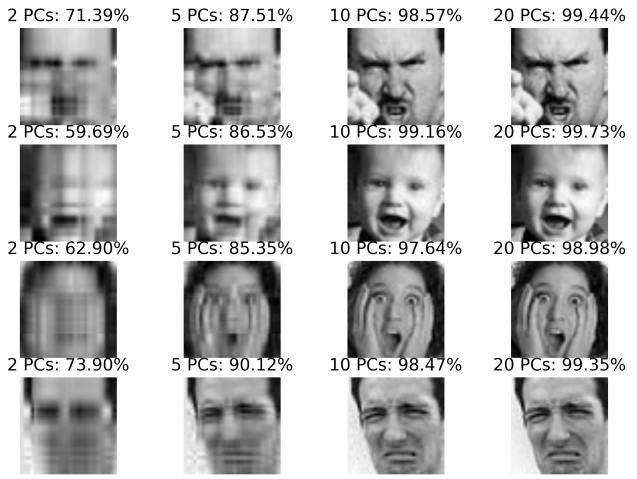

Fig. 2: Example of features learned by principal component analysis and the percentage of variance explained. Emotions: Angry, Happy, Surprise, Disgust

IV. Methods

A. Convolutional Neural Network

The CNN architecture that we decided was best for classifying pairs of emotions comprised of three pairs of convolution and sub-sampling layers fed into 3,072 fully-connected neurons and output to two classification neurons.

The first convolutional layer used 32 7×7 kernels to generate 32 feature maps of size 42×42. Each element in a feature map was a linear convolution of 49 pixels from the input image, passed through a rectified linear unit function. Each of these 32 feature maps were sub-sampled using max-pooling by taking the maximum element of a 2×2 section of a feature map to become an element in the sub-sampled set of feature maps. This reduced the size of the incoming arrays to the next convolutional layer to 21×21 while containing as much information as possible from the original feature map.

In the second convolutional layer, the 32 sub-sampled feature maps were convolved with 32 new 7×7 kernels to generate 32 new feature maps. Again each entry in a feature map is a linear convolution of 49 entries from the incoming feature maps passed through a rectified linear unit function. The feature maps were sub-sampled using max-pooling to reduce the size of the output

Layers	Number of Outputs	Size of Feature Maps	Nonlinear Activation Function
Input Image	1	42×42	None
Convolution 1	32	42×42	Rectified Linear Unit
Max Pool 1	32	21×21	None
Convolution 2	32	19×19	Rectified Linear Unit
Max Pool 2	32	10×10	None
Convolution 3	64	10×10	Rectified Linear Unit
Max Pool 3	64	5×5	None
Fully Connected	3072	N/A	Sigmoid
Classification	2	N/A	Softmax

TABLE I: Convolutional Neural Network Model Layer Specifics

feature maps to 10×10.

The third and final convolutional layer generated 64 feature maps from kernels of size 7×7. Rectified linear units were again the non-linear activation function applied to each linear convolution of 49 entries going into a single entry in the feature maps. These 64 feature maps were sub-sampled again using max pooling to reduce the size of the feature maps to 5×5.

These 64 5×5 feature maps were fed into a fully-connected neural network with 3,072 neurons with a sigmoid function as the activation. The outputs from these 3,072 neurons were then output to two classifying neurons using the softmax function to generate two probabilities, each one representing the probability of an output belonging to one emotion or the other.

The model designed to classify seven emotions is the same as the one used to classify on pairs of emotions with the exception of the final output layer. This output layer has seven output nodes (instead of two) so it can classify the seven emotions. (It uses the softmax activation function just as the CNN used to classify on two emotions.)

Our Convolutional Neural Networks were built using the Theano and Lasagne libraries in Python. Theano is a Deep Learning library that has functions and classes to set up convolutional layers, sub-sampling layers, and fully connected layers, and provides a variety of non-linear activation functions. It also has the ability to use either a CPU or a GPU to perform the matrix computations. Because its API is difficult to use, we chose to use Lasagne. Lasagne is a wrapper for Theano that allowed us to define specific layers line-by-line making it easier to build our CNNs.

Theano expects input as 4-D tensors. We used the Numpy, Scipy, SciKit-Image libraries in Python to load in image data and process it into 4-D arrays where the first dimension is the number of images, the second is fixed at 1, and the remaining 2 dimensions are the image size 42×42.

For training the CNN model, we subset the images so that 80% of our images went into the training set and the remaining 20% went into test set.

We initialized the weights for the model with small, random numbers from a uniform distribution. We used categorical cross entropy for the loss function. 200 training images were sent through the network at a time and classified. The error was then back-propagated through the network and the weights updated. We used stochastic gradient descent as our optimization method. This process is one training epoch.

We trained for 100 epochs and pickled the trained parameters into Numpy *.npz files.

B. Support Vector Machines

We used the Support Vector Machine model to compare against the results and performance of our CNN. As such, we did not seek to optimize the parameters of the SVM model. We did, however, perform some feature engineering (Principal Component Analysis, see "Feature Engineering" Section above) to boost the speed of the SVM model. We used Python's SciKit Learn SVM and PCA libraries to do this. In the SVM model, we held gamma constant at 0.001 and the cost parameter constant at 1. We applied two different kernels for additional comparison: a quadratic kernel and a Gaussian kernel. Again, as we were interested in only having a baseline comparison to our CNN model, this was the extent of the parameter tuning performed on the SVM model.

SciKit Learn's SVM model required that the image be flattened into a one-dimensional array of pixels to perform classification on an image. For non-PCA SVM, this resulted in a 2304-length array (48×48) for each image. For SVM with the first 15 principal components, this resulted in a 720-length array (15×48).

C. Amazon Web Services

Initially, training a CNN model for 100 epochs on our personal computers took on average 8-12 hours. In order to train multiple models efficiently to find the best classifiers, we took advantage of Theano's ability to use GPUs to speed up the internal matrix operations. We set up two Amazon Web Services (AWS) g2.8xlarge instances, each backed by four high-performance NVIDIA GPUs. This reduced model training and testing down to one hour for 100 epochs. We were able to obtain validation curves for 24 models in only a few days (see Appendix). From this we identified our best-performing CNN classifier as the model with three convolution/max pool layers followed by 3,072 fully-connected neurons classifying "Happy" vs. "Disgust".

V. Experimental Results

A. The CNN Model

We trained the CNN model for 100 epochs using stochastic gradient descent and training batch sizes of 200 images. In all, we trained the model on 15,883 images of faces expressing happiness and disgust and validated the model on 3,971 test images. This validation also occurred every epoch on test size batches of 200 images. Figures 3 and 4 both show that our model achieved 96.69% test accuracy classifying "Happy" versus "Disgust", the highest of all the binary classifiers we tried (see Appendix for discussion of the various models we tried). Figure 5 shows the train loss and test loss for the model over 100 epochs. We can see that after about 50 epochs the model begins to overfit the training data and results in a larger test loss than train loss.

For the CNN classifying all seven emotions, we trained the model for 850 epochs using similar methods to the binary CNN classifier. Figure 6 shows the test accuracy of the model versus number of training epochs. While the model did get better test accuracy as it went through successive training epochs, Figure 7 shows that after about 200 epochs the model begins to overfit the training data resulting in larger test loss than training loss.

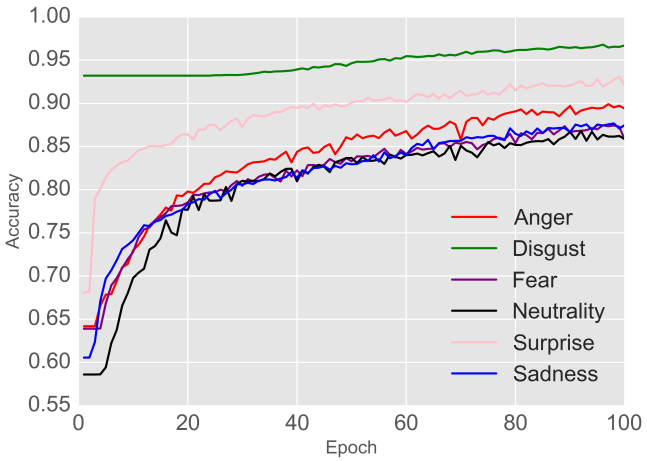

Fig. 3: Accuracy Classifying Faces Expressing Happiness Vs. Other Emotions

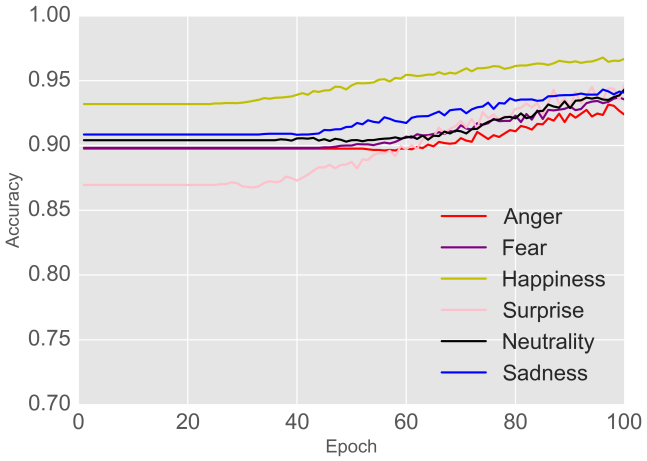

Fig. 4: Accuracy Classifying Faces Expressing Disgust Vs. Other Emotions

B. The SVM Classifier

Upon proper flattening of the images, leave-one-out cross-validation was used to fit the SVM model. We used a 2/3, 1/3 split to split the data into train and test groups, respectively. Under binary classification, the SVM model achieved its highest classification accuracy of 94.4% for "Happy" vs. "Disgust" with a Gaussian kernel. This model was trained on 13,236 images and tested on 6,618 images, for a total of 19,854 images. This model took 40.6 minutes to compute. This same model

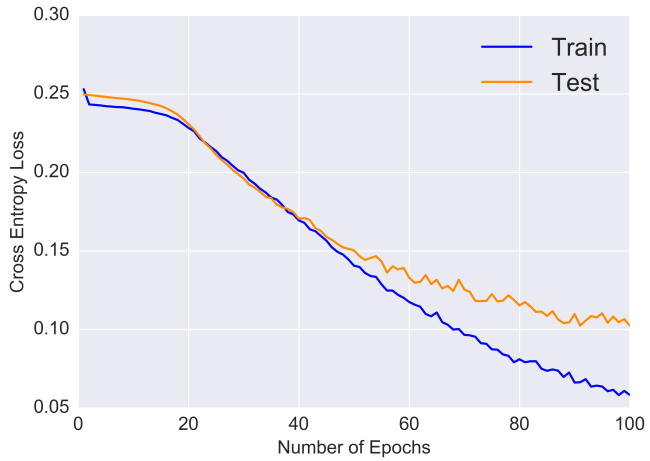

Fig. 5: Model Validation Curves For CNN Model Happy vs Disgust

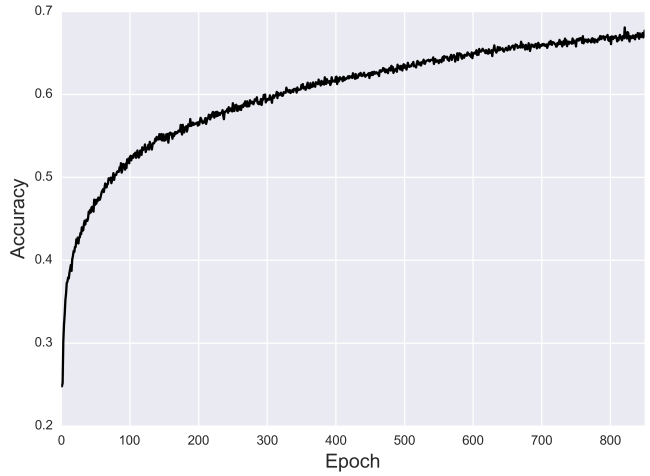

Fig. 6: Accuracy Classifying All Seven Emotions

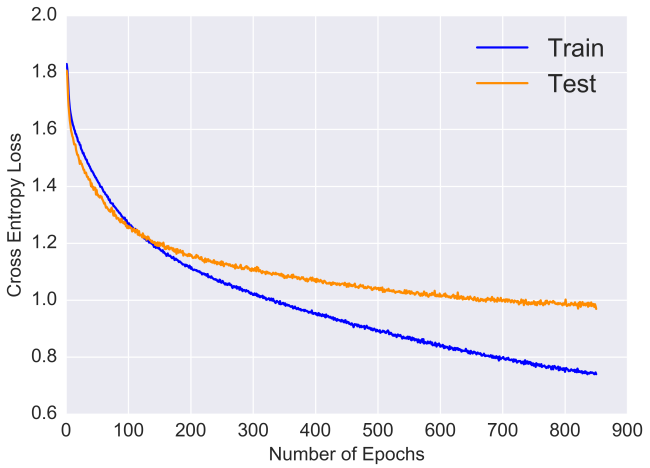

Fig. 7: Model Validation Curves For CNN Model Classifying All Seven Emotions

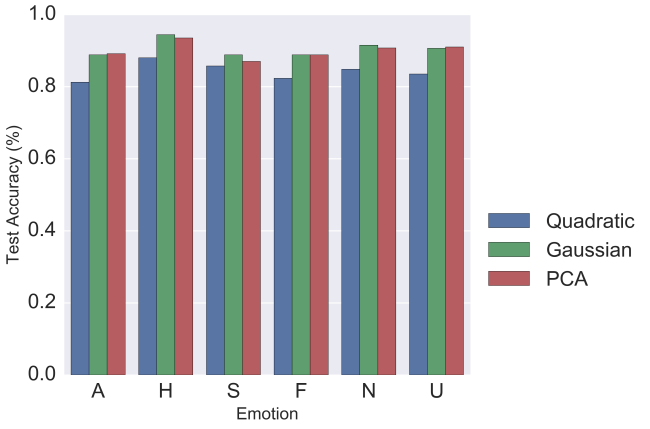

Fig. 8: Classification Accuracy for Disgust versus all Other Emotions

was then trained, but with the first 15 principal components of each image. The principal component model achieved a classification accuracy of 93.5% in only 2.25 minutes. Each of the other emotion combinations performed reasonably well, each giving a classification accuracy greater than 80% (see Fig 8).

Figure 8 shows that the Gaussian Kernel SVM outperformed the quadratic kernel SVM and PCA SVM in nearly all cases. (PCA SVM outperformed the Gaussian performance by only a small margin for "Disgust" vs "Fear.") The quadratic kernel is not as effective at mapping the image pixels into a higher-dimensional, linearly separable space. While the Gaussian kernel seems to be the strongest performer, it suffers in terms of model run-time, see Figure 9. Here, we can see that PCA SVM outperforms all other models, while still maintaining a high degree of classification accuracy as seen in Figure 8. As the Gaussian and Quadratic SVM models get over 10,000 images, the run-times begin to increase drastically. (*NOTE: To get these various run-times, random subsets of 100, 1,000, 10,000, and 20,000 images were created from the full data-set. Each pair of emotions would further subset the total number of images in each subset.*)

We used the SVM model to classify on all 7 emotions. The Gaussian SVM achieved a classification accuracy of 28.6% in 13.2 hours for all 74,844 images. PCA SVM achieved 24.7% in 2.2 hours. We did not attempt to classify all seven

emotions with an SVM quadratic kernel since the results of the Gaussian model were so poor.

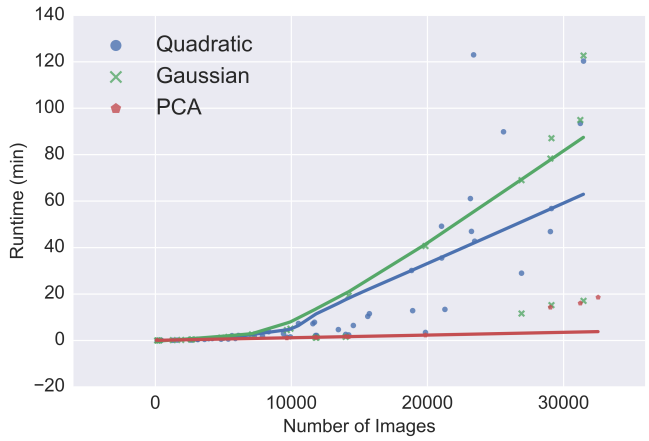

Fig. 9: SVM Model Run-Time as Number of Images Increases

VI. CONCLUSIONS

In this paper, we presented the results of several models to detect and classify emotion from static images. A convolutional neural network with three pairs of convolutional and pooling layers followed by a fully-connected network with 3,072 neurons achieved 96.69% accuracy for the binary classification of "Happy" versus "Disgust" in 1 hour. A similar neural network achieved 67.67% accuracy for all seven base emotions ("Anger", "Disgust", "Fear", "Happy", "Unhappy", "Surprise", and "Neutral") in 30 hours.

While a convolutional neural network was able to achieve higher accuracy than the SVM model, it came at the expense of increased computing time. The neural network required the use of high-powered GPUs while the SVM model can be trained much faster and can be run on a personal computer.

Building a neural network allowed us to extract the features that were learned for classification. This can aid in understanding the learning problem by giving us an idea of what some of the useful features are.

Faces expressing "disgust" and "happiness" are more easily classifiable. This is supported across each of our learning models. For binary classification, the convolutional neural network and all variants of the SVM model performed best when classifying disgust or happiness than it did for the other emotions.

APPENDIX

A. CNN Model Selection

The road to selecting the model that is the main focus of the paper had many siblings before we made our decision. What follows is a brief outline of some of the other models we tried with some accompanying validation curves. Following [1], we built many similar model variants classifying on all possible pairs of emotions. In addition, we attempted two other CNN's with four pairs of convolution/subsampling layers feeding into 4096 fully connected neurons that then classified an image. One classified images as "Fear" versus "Surprise" and the other "Happy" versus "Disgust". See Figure 11 for validation curves for these models. We did not choose the larger CNN for our "Happy" versus "Disgust" model because this model did not classify any better than the model with only three convolution/subsampling layers into 3072 fully connected neurons. Moreover, the time to train and validate on a GPU for this larger model was three hours and did not give much better accuracy. In addition we can also see that these two larger models are beginning to overfit after about 50 epochs.

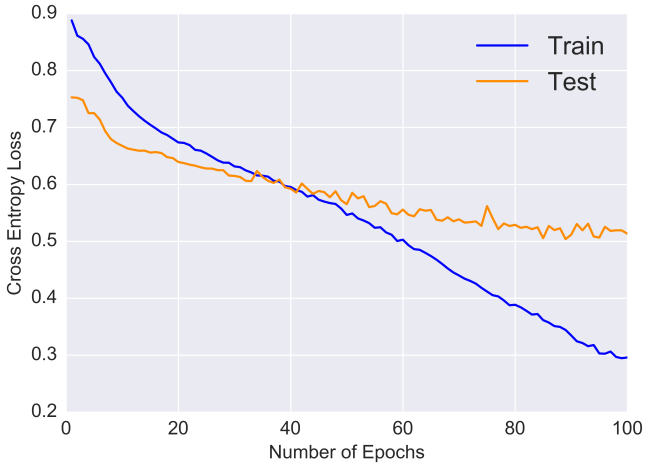

Fig. 10: Validation Curve For Angry vs Fear

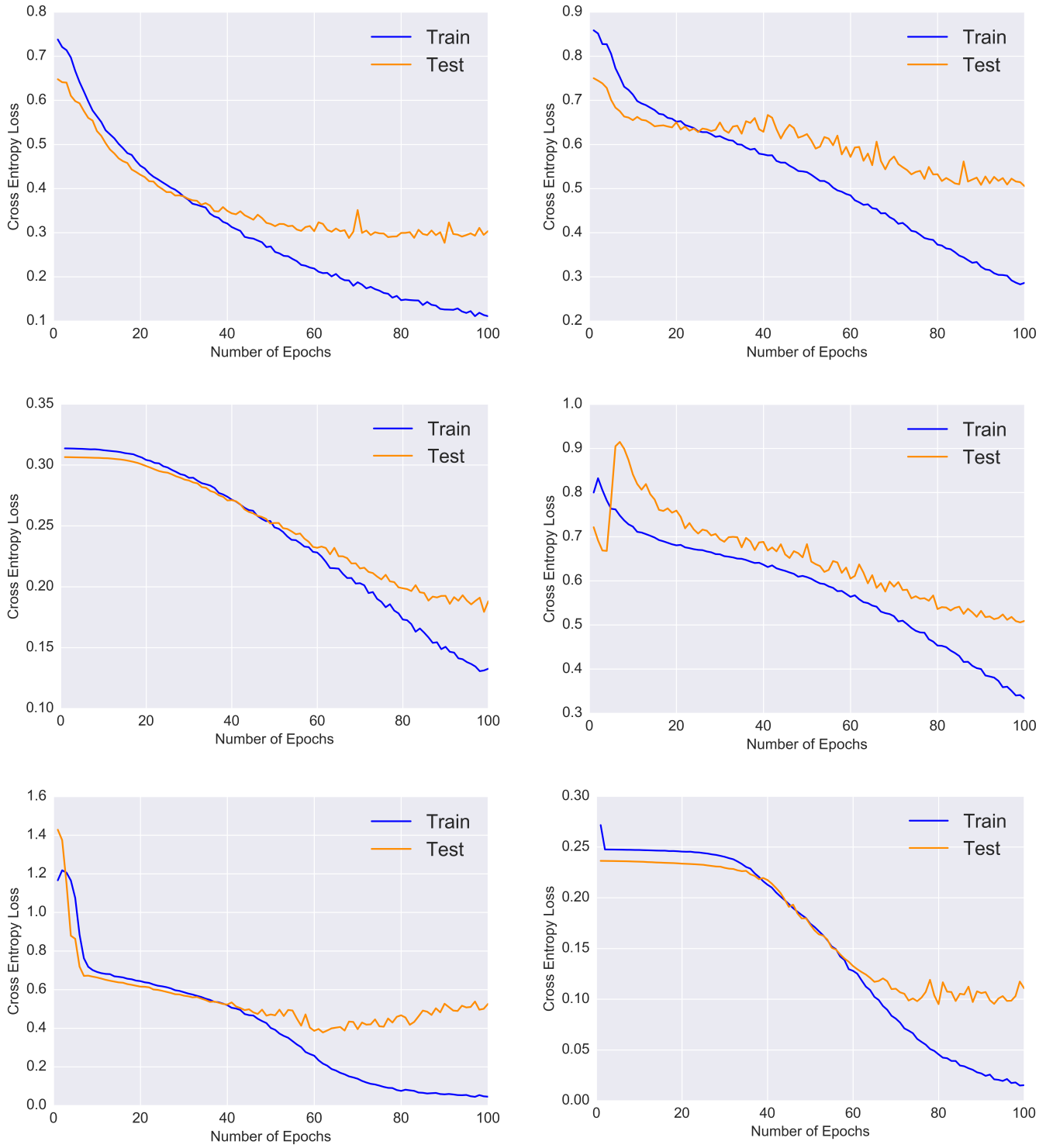

Fig. 11: Validation Curves For (left to right) “Angry” vs “Happy”, “Unhappy” vs “Neutral”, “Unhappy” vs “Disgust”, “Unhappy” vs “Fear”, Four Conv. Layers w/4096 neurons classifying “Fear” vs “Surprise” and similarly classifying “Happy” vs “Disgust”

REFERENCES

[1] Kim, Bo-Kyeong, et al. 2015. Hierarchical Committee of Deep CNNs with Exponentially Weighted Decision Fusion for Static Facial Expression Recognition.

[2] Ng, Hong-Wei, et al. 2015 Deep Learning for Emotion Recognition on Small Datasets Using Transfer Learning.

[3] Yu, Zhiding, et al. 2015. Image-based Static Facial Expression Recognition with Multiple Deep Network Learning.

[4] Dhall, Abhinav. Third Emotion Recognition in the Wild Challenge results @ ACM ICMI March 5, 2016. https://sites.google.com/site/dhallabhinav/home/emotiw2015

[5] Japanese Female Facial Expression (JAFFE) Database. March 5, 2016. http://www.kasrl.org/jaffe.html

[6] Challenges in Representation Learning: Facial Expression Recognition Challenge. March 5, 2016.https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge

[7] Hastie, Trevor, et al. 2013. The Elements of Statistical Learning.

[8] Bishop, Christopher M., et al. 2009. Pattern Recognition and Machine Learning.

[9] Szabo, Roland. 2013. 'Neural Networks in Python. https://rolisz.ro/2013/04/18/neural-networks-in-python/