Facial affect detection using convolutional neural networks

Sherrie Wang
Institute of Computational and Mathematical Engineering
Stanford University
sherwang@stanford.edu

Abstract

In this work, I present a convolutional neural network (CNN) to classify human facial affect into seven basic emotions. Classification using a three-layer CNN and rotation-augmented data yields a test set accuracy of 38.1%, significantly higher than the SVM benchmark of 12.1%. I also extract features from the last layers of deeper networks, and at best achieve 30.3% test set accuracy using features from a network trained on data from the 2015 Emotion Recognition in the Wild competition.

1. Introduction

Emotions are an important part of human intelligence, decision making, communication, memory, and more. In order for computers to interact effectively with people, they ought to be able to detect human affect.

Affect detection has historically been under-emphasized in the field of artificial intelligence, but has important ramifications in application. In medicine, it could be used to detect pain during procedures, or monitor depression in patients; in education, teaching could be tailored to the emotional state of the student, *e.g*. making the lesson easier if the student exhibits frustration. In marketing, successful affect detection can be extremely lucrative — last year, advertising giant M&C Saatchi began testing billboards with hidden Microsoft Kinect cameras that read viewers' facial expressions and react accordingly.

In this paper, I explore the detection of emotions from facial expressions using convolutional neural networks (CNNs). The input to my algorithm are images from the Cohn-Kanade AU-Coded Expression Database Version 2 (CK+), which includes both posed and non-posed (spontaneous) expressions, comes labeled with six basic emotions, and was recorded in settings controlled for illumination and angle. I then use an SVM, three-layer CNN, and deeper architectures trained on ImageNet and other databases to output a predicted emotion.

1.1. Classification of Emotions

The classification of emotions can be a difficult problem, as there is no consensus whether emotions are continuous or discrete, on which axes they can be measured, or, if they are discrete, how many categories exist [16]. Recent research classify as few as four emotions [8] and as many as hundreds [17].

Thankfully, we do not need complete understanding of emotions to produce useful results; just knowing whether someone feels good or bad can steer a person's — or computer's — response in vastly different directions. For simplicity and familiarity, I choose to focus on the categorization of facial affect into seven basic emotions as illuminated by the work of Ekman and Friesen and built upon by Matsumoto *et al*.: anger, contempt, disgust, fear, happiness, sadness and surprise [6] [13]. Ekman's categorization has been popular since the 1970s, and seven categories provide enough variety to be useful while remaining tractable.

1.2. Facial Expression

Facial expressions are one of the primary ways humans communicate emotion. When humans communicate, we look at each others' faces; "emoticons" — pictorial representations of facial expressions — play a significant if not primary role in conveying mood and feeling in modern textual communication.

As early as the 1800s, Charles Darwin wrote that "facial expressions of emotion are universal, not learned differently in each culture" [3]. In the 1970s, the Facial Action Coding System (FACS) was created to taxonomize human facial movements by their appearance on the face, and remains a standard to systematically categorize the physical expression of emotions [7].

The universality of facial expressions has been controversial since Darwin's declaration, but for the purposes of this work, we focus on the seven emotions as recognizable via facial expressions by Americans.

I hypothesized that the CNN would learn facial expressions for emotions that match closely with those found by Ekman and articulated through FACS.

Figure 1. The Cohn-Kanade AU-Coded Expression Database Version 2 (CK+) contains 327 labeled recordings of posed and non-posed sequences, and are coded for seven basic emotions. The last row shows the mean image in the dataset for each emotion.

2. Related Work

2.1. Facial Action Unit Detection

Much work has been done to detect the basic action units (AUs) of the FACS. Tian *et al*. developed an automatic face analysis (AFA) system to analyze individual AUs based on both permanent and transient facial features in frontal face image sequences [11]. Their recognition rate is 95.6% on the Cohn-Kanade Database, Version 1 (CK). Donato *et al*. were able to achieve 96.9% recognition using Gabor wavelet decomposition [5]. Bazzo and Lamar invented a pre-processing step based on the neutral face average difference and used a neural-network-based classifier combined with Gabor wavelet to obtain recognition rates of 86.55% and 81.63%, respectively, for the upper and the lower faces [1]. In 2006, Chuang and Shih used independent component analysis for feature extraction and support vector machine for the pattern classifier to attain 100% recognition on the CK dataset using the whole face [2].

2.2. Emotion Recognition in the Wild

Since 2013, the Emotion Recognition in the Wild (EmotiW) challenge has been held at the ACM International Conference on Multimodal Interfaces in Seattle. The challenge consists of two sub challenges, (1) video based emotion recognition and (2) image based static facial expression recognition. The task is to assign a single emotion label to the video clip or static image from Ekman's six basic emotions and a seventh emotional state called "neutral". Static images come from the Static Facial Expressions in the Wild (SFEW) dataset [4].

I will be focusing on static facial expression recognition for this project, and a number of EmotiW entries in the 2015 challenge use deep CNNs with various preprocessing techniques, transfer learning, and ensembles to achieve accuracies of around 50% on the validation set and 55% on the test set [10] [14] (a large improvement over 36% and 39% challenge baseline using SVM).

3. Methods

3.1. Baseline SVM

The model used to benchmark CNN performance is a linear classifier trained with multi-class support vector machine (SVM) loss. The score function is defined as

$$f(x_i, W, b) = Wx_i + b$$

where x_i is an image's pixel data flattened to a $K \times 1$ vector, W is a $C \times K$ weight matrix, and b is a $C \times 1$ bias vector. The output of the function is a $C \times 1$ vector of class scores, where C is the number of classes. The score for a class is the weighted sum of an image's pixel values, so a linear classifier can be interpreted as how much an image matches the "template" for a class.

In training, once the class scores are calculated, we use a loss function to quantify how well the classifier performs. In multi-class SVM loss, the loss for the ith image is calculated as

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta)$$

where y_i is the correct class for x_i. The SVM loss will be non-zero for a class $j \neq y_i$ when the score for class j is not at least Δ lower than the score for the correct class y_i. A commonly used value for Δ, and one adopted here, is $\Delta = 1$.

To discourage the weights from taking on arbitrarily large values, we add an L2 regularization term to the loss function. The complete loss function is

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda \sum_{j=1}^{C} \sum_{k=1}^{D} W_{j,k}^2$$

where $W_{j,k}$ is the (j,k) entry of the weight matrix and λ is a hyper-parameter determined through cross-validation.

The goal of training is to minimize the loss across training data. Each element of the weight and bias is initialized as a Gaussian with mean zero and some small standard deviation. At each iteration, the derivative of the loss is calculated with respect to W and b, and the parameters are updated using stochastic gradient descent.

3.2. Three-layer CNN

Like a linear classifier, convolutional neural networks have learnable weights and biases; however, in a CNN not all of the image is "seen" by the model at once, there are many convolutional layers of weights and biases, and between convolutional layers are non-linear functions that in combination allow the model to approximate much more complicated functions than a linear classifier.

A typical CNN architecture contains all or some of the following layer types.

- **Input layer.** The input layer contains the pixel values of the images that the network is training or testing on. It is of size $N \times D \times W \times H$, where N is the number of samples, D is the dimension of each image (usually 3 for color images), W is the width of the image, and H is the height of the image.

- **Convolutional layer.** Each convolutional layer has many "neurons" — a set of weights and biases — that are each connected to a small region of the layer before it. The filter size F of a neuron determines how many pixels (if the previous layer is the input layer) or elements of the previous layer's output are "seen" by the neuron at a time. For a square filter of width F, a single neuron is comprised of a $F \times F \times D$ weight matrix and a $1 \times 1 \times 1$ bias. Note that each neuron extends through the full depth D of the input volume. The output of the convolutional layer is the dot product between its filters and the input as the filter is "slid" across the height and width of the input. The stride S dictates how many units the filter advances as it convolves with the input, and the zero-padding P describes how many layers of zeros are added around the border of the input volume. Thus each filter of size F requires $\left(\frac{W-F+2P}{S}+1\right)\left(\frac{H-F+2P}{S}+1\right)$ neurons sharing the same weights and biases to view the entire input of width W and height H. Parameters are shared across neurons of a filter under the assumption that if a patch of features is of interest at one location, it is also of interest at another. Thus we can think of each filter as capturing some feature that is of interest across spatial dimensions. The output volume for an input of size $N \times D \times W_{in} \times H_{in}$ has dimension $N \times K \times W_{out} \times H_{out}$, where K is the number of filters in the layer, $W_{out} = \frac{W_{in}-F+2P}{S} + 1$, and $H_{out} = \frac{H_{in}-F+2P}{S} + 1$.

- **ReLU layer.** This layer applies an element-wise activation function, often the zero-thresholding function $\max(0, x_{i,j})$, or rectified linear unit (ReLU). This introduces a non-linearity to the CNN and does not change the input volume dimensions.

- **Pooling layer.** The pooling layer down-samples along spatial dimensions to reduce the number of parameters and help combat overfitting. It does not act across the depth dimension of the input volume. A layer that pools with filters of size F retains the maximum value in a $F \times F$ window, and discards the other elements. Thus for an input of size $N \times D \times W_{in} \times H_{in}$, a pooling layer with spatial extent F and stride S outputs a volume of size $N \times D \times W_{out} \times H_{out}$, where $W_{out} = \frac{W_{in}-F}{S}+1$ and $H_{out} = \frac{H_{in}-F}{S}+1$. A typical filter size is 2×2; any larger and too much information is discarded.

- **Normalization layer.** A normalization layer. Batch normalization forces its input to become a unit Gaussian output, and is usually inserted after convolutional or fully-connected layers. Networks with batch normalization are usually much more robust to bad initialization than those without. The input volume dimension does not change after a normalization layer.

- **Fully-connected layer.** A fully-connected (FC) layer is comprised of "neurons" that each sees the entire input volume and transforms it into a one-dimensional vector. The weight matrix of an FC layer has dimension $D_{flat} \times K$, where D_{flat} is the dimension of flattened input, and K is the number of neurons. The output volume for an input of size $N \times D_{in} \times W_{in} \times H_{in}$ ($D_{flat} = D_{in} \times W_{in} \times H_{in}$) has dimension $N \times K$.

The CNN used in this project has the architecture

```
CONV - BN - ReLU - 2x2 POOL - FC - BN -
        ReLU - FC - softmax
```

That is, images are first processed by a convolutional layer, then normalized, rectified, and pooled, before being seen by a fully-connected layer, normalized again, rectified again, and seen by a second fully-connected layer. The loss is then calculated using a cross-entropy loss for the Softmax classifier; the loss takes the form

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

Unlike the SVM classifier, Softmax classification does not stop at any threshold in its attempt to minimize $f_j, j \neq$

y_i relative to f_{y_i}. In practice, however, the two are comparable in performance.

As discussed previously, an L2 regularization term was also added to the Softmax loss to deter large weights and overfitting in training.

To update the parameters across the network, I use the Adam update, which smoothes the gradient at each iteration and updates parameters at rates inversely proportional to the size of their smoothed gradient. At each iteration,

$$m = \beta_1 m + (1 - \beta_1) dx$$
$$v = \beta_2 v + (1 - \beta_2) dx^2$$
$$x = x - l * \frac{m}{\sqrt{v} + \epsilon}$$

where dx is the gradient at a given iteration, m is the smoothed gradient, v is the smoothed squared gradient, l is the learning rate, and ϵ is some small term that prevents division by zero.

3.3. Transfer Learning

Since deep convolutional neural networks can take weeks to train and require large datasets, a popular technique is to take a CNN that has already been trained and modify it slightly. This approach has the benefit of borrowing basic features — for example, edges in various directions — from the pre-trained network rather than learning them from scratch. This method is particularly appealing given the small size of the CK+ database; rather than training a deep network on so little data and severely overfit, I can extract features from a deeper network and train one or more fully-connected layers atop the features. Here I extract from four deeper networks using Caffe on Amazon Web Services' Elastic Compute Cloud (EC2):

- **BVLC Reference CaffeNet.** [9] This network is the Berkeley Vision and Learning Center's (BVLC) implementation of an AlexNet trained on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 dataset, which contains 1.2 million images of objects belonging to 1000 different classes. It contains five convolutional and three fully-connected layers.

- **BVLC GoogLeNet.** [18] The GoogLeNet is also trained on the ILSVRC 2012 dataset. Its architecture contains two vanilla convolutional layers, three "inception" modules, and a fully-connected layer at the end. Inception modules involve parallel layers of convolutional (sometimes pooling) layers whose filters are then concatenated. Convolutions of filter 1×1 are introduced to reduce dimensions.

- **VGG Face Descriptor.** [15] This network was trained on 2.6 million images of over 2,600 people and used for face detection. It has 13 convolutional layers and three fully-connected layers. The input is a 225-by-225 pixel face image.

- **EmotiW Network.** [10] This network was trained on the SFEW dataset for the 2015 Emotion Recognition in the Wild (EmotiW) contest. It has five convolutional layers and three fully-connected layers, like AlexNet, and outputs class scores for six basic emotions: anger, disgust, fear, sadness, happiness, and surprise — very close to the classes of interest in this project.

Images from the CK+ dataset were convolved with each network, and features were extracted from the last or near-last fully-connected layers of the models. I then added one to three fully-connected layers atop the extracted features to classify them into the seven basic emotions. I expected that networks trained on images of faces would perform better than those trained on ImageNet.

4. Dataset and Features

4.1. Cohn-Kanade Database

The Cohn-Kanade AU-Coded Expression Database Version 2 (CK+) [12] is the preeminent dataset used by researchers in the development of facial action unit and expression recognition systems. The CK+ database contains 593 recordings (10,708 frames) of posed and non-posed sequences, which are coded for action units and the seven basic emotions. The sequences are recorded in a lab setting under controlled conditions of light and head motion, which make them a good starting place but of limited use for detection of facial expressions in the wild. In total, 123 subjects are represented in the dataset.

Of the 593 recordings, only 327 have corresponding emotion labels — the rest were considered by researchers compiling the database to contain unusual expressions of the emotions (further limiting the generalization of any model trained on CK+ data). Each recording contains around 10 images: the first frame shows the subject in a neutral expression, the last frame shows the subject exhibiting a particular emotion, and the frames in between transition between the two states. Upon examining the frames, I concluded that the emotion of interest was discernible on the subjects' faces in last 6 frames, to varying intensities, to the human eye. Therefore rather than take a single frame from each recording, I opted to take the last 6 frames. In aggregate, the dataset was 1,962 images, representing 327 recordings. The dataset is visualized in Figure 1.

Each image was originally of width 640 pixels by height 490 pixels, with a 10-pixel black "header" at the top of each file. In preprocessing I removed these 640-by-10 pixel regions. When training a shallow CNN on my own CPU, I decreased the images to 80 pixels by 60 pixels by sampling every eighth row and eighth column of pixels. To the human eye, emotions are still clearly recognizable at this resolution. When training and testing deeper models on Amazon

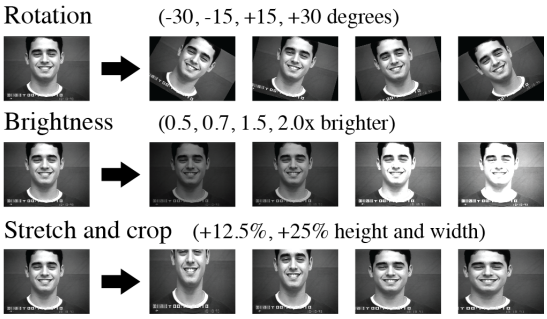

Figure 2. I change the rotation, brightness, and aspect ratio of the images to diversify the features and increase the size of the training set.

Web Services' GPU, I left the images at their original resolution.

Furthermore, of these 1,962 images, 1,758 were black and white, and 204 in color. The color images were added in the Version 2 update to the original Cohn-Kanade data. Rather than throw away the color data in color images, I tripled the grayscale data for black and white images to have the same dimensions across input data. In hindsight, it may have been better to make all images grayscale, as the distribution of classes for grayscale and color images are not identical (see "contempt" in Figure 1).

The distribution of classes across all images is not entirely even, but not terribly skewed: 14% anger, 6% contempt, 18% disgust, 8% fear, 21% happy, 8% sadness, and 25% surprise.

Sixty-four percent of the dataset (1,254 images) was used as training data, 16% (312 images) as validation data, and the remaining 20% (396 images) as test data. Selection of images for each set was randomized, but care was taken to ensure that the 6 frames from each recording were not split among different sets.

4.2. Data Augmentation

Because the dataset is relatively small — in comparison to datasets like ImageNet that are comprised of millions of images — I augmented the data with the following techniques (Figure 2).

- **Rotation.** Each image was rotated by -30, -15, +15, and +30 degrees, and any gaps appearing between the image and the view window were padded with zeros. This procedure should theoretically make the network more robust to head tilts and different angles that features (smiles, eyes) appear on the faces of different subjects.

- **Brightness.** Each image was brightened by a factor of 0.5, 0.7, 1.5, and 2.0. Although the CK+ dataset was recorded in controlled environments, images vary in brightness and subjects vary in skin tone. We want the network to learn facial expressions, not how dark or light images are.

- **Stretch and crop.** Each image was stretched by +12.5% and +25% in width and height, and then cropped to 80 pixels by 60 pixels. We want the network to learn prototypes of each emotion, which should be independent of how thin or wide one's face is.

Figure 3. Visualizations of the three-layer CNN's 64 convolutional filters, trained on 1,254 images in the CK+ dataset.

The augmented data was combined with the 1,254 non-augmented training images to produce a total of 6,270 training images. The validation and test sets were not augmented.

5. Results

I use the methods discussed above to classify images from the CK+ dataset into the seven basic emotions. The results from each model are evaluated on the accuracy of emotion prediction on the test set.

5.1. Baseline SVM

A linear classifier trained on 1,254 images with SVM loss yielded a test set accuracy of 12.1%, no better than random guessing of 7 categories (Table 1). Weights were initialized from a Gaussian with mean zero and standard deviation 0.001. A learning rate of 1e-7 and regularization of 5e4 were used based on coarse hyper-parameter selection. Visualizations of the learned weights clearly resemble faces; interestingly, disgust has high weights around the subjects' eyes, sadness around the mouth, happiness on the upper lip, and surprise around the mouth.

I am a bit surprised that despite templates for each class — especially "surprise" — picking up seemingly useful fea-

Model	N	Filters	Filter size	Augmentation	Train acc	Val acc	Test acc
SVM	1,962	-	-	-	0.242	0.153	0.121
CNN	327	64	7x7	-	0.842	0.192	0.273
CNN	327	64	9x9	-	0.766	0.250	0.303
CNN	327	128	7x7	-	0.766	0.288	0.394
CNN	327	128	7x7	Mean subtraction	0.885	0.288	0.379
CNN	1,962	64	7x7	-	0.775	0.321	0.255
CNN	1,962	64	7x7	Rotation	0.768	0.407	0.381
CNN	1,962	64	7x7	Brightness	0.955	0.256	0.220
CNN	1,962	64	7x7	Stretch	0.687	0.369	0.250

Table 1. A three-layer convolutional neural network (CNN) was trained using a fully connected layer of size 500. After a grid search, the learning rate was set at 1e-4 and the regularization strength at 1e-5.

tures, the classification accuracy is still no better than random.

5.2. Three-layer CNN

Grid search on hyper-parameters yielded a learning rate of 5e-5 when training on 327 images with a batch size of 20 and 1e-4 when training on 1,962 with a batch size of 100, and a regularization strength of 1e-5. Loss curves over iterations were a good shape with these hyper-parameters: steep at first before tapering off. At first models were trained for 20 epochs; after observing that validation accuracy peaked earlier, models were trained for only 6 epochs.

Results from the three-layer CNN are summarized in Table 1. I first experimented with different filter numbers and sizes on a three-layer CNN with no data augmentation, using a small dataset of 327 images. (Note that the images are fairly centered, so one could argue that a CNN may not have a great advantage over a network where parameters are not shared among neurons at different spatial locations. However, since facial features differ in location across individuals, it may nonetheless be better to share parameters in a CNN architecture — given more time, I would also have a vanilla neural network classification benchmark.)

The highest performing three-layer CNN had 128 filters, each of which were of size 7. This is not surprising, as more filters allow the network to learn more features with which to discern an image class. Increasing the size of the filter also seemed to improve classification accuracy, though not by as much as doubling the number of filters. Visualizations of the first convolutional layer filters (Figure 3) show that the network was learning features that look like dots and lines in various directions — parts of facial features oriented in various directions, perhaps.

In the interest of training time, I used 64 rather than 128 filters of size 7 on the full dataset of size 1,962. The network was clearly able to overfit the training data, achieving a training accuracy of 76.6% at peak validation accuracy (and eventually 100%). Validation accuracy was 32.1% and test accuracy was 25.5%, better than random but not by much.

Regularization was not enough to combat overfitting in training. It often happened that validation accuracy would peak at a training accuracy near 80%, and then decrease. I stopped the training early at peak validation accuracy to catch the model before it was too overfit to the training data.

5.3. Data Augmentation

Of the three data augmentation techniques, rotation showed the greatest improvement in accuracy; validation accuracy was 40.7% and test accuracy was 38.1% — significantly better than random. Changing brightness and aspect ratio did not boost accuracy as much; brightness saw a validation accuracy of 25.6% and a test accuracy of 22.0%, and stretching a validation accuracy of 36.9% and test accuracy of 25.0%.

One can speculate why rotation seemed to help the most: perhaps facial expressions appear on individuals' faces at a variety of angles, and rotating faces introduced the network to smiles, furrowed brows, etc. with this variety.

Figure 4 shows the confusion matrix for the CNN trained

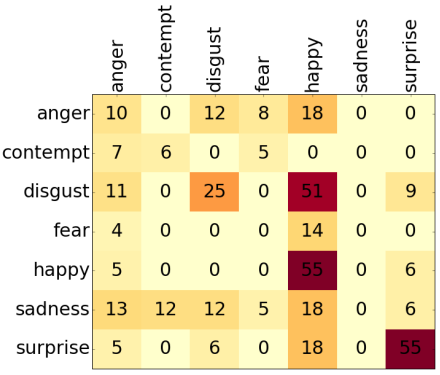

Figure 4. The confusion matrix for the highest performing CNN model. True classes are the rows, and predicted classes are the columns. Happiness and surprise are easiest for the CNN to learn. Disgust and other emotions are often mistaken for happiness. Sadness and fear prove challenging to classify.

Network	Feature layer	Train acc	Val acc	Test acc
BVLC Reference CaffeNet	FC8	0.550	0.250	0.197
BVLC Reference CaffeNet	FC7	0.794	0.212	0.288
BVLC Reference CaffeNet	FC6	0.914	0.212	0.167
BVLC GoogLeNet	Pool5 + FC	0.340	0.269	0.152
VGG Face Descriptor	FC8	0.871	0.212	0.242
VGG Face Descriptor	FC6	0.990	0.231	0.197
EmotiW Network	FC7	0.895	0.269	0.303

Table 2. Fully-connected layers were trained atop features extracted from various layers of deep networks trained on millions of images. The EmotiW network performed the best, while others were no better than random.

on rotation-augmented data. It seems that happiness and surprise are by far the easiest emotions to classify — a look at the mean images for these emotions makes this understandable. Surprise is characterized by an open mouth and raised eyebrows, and happiness a smile showing teeth. Meanwhile, every emotion except contempt is frequently mistaken for happiness, an observation not entirely explained by the representation of the classes in the dataset (surprise is the most common class, with happiness second).

5.4. Transfer Learning

In the interest of training time, I did not construct more than three convolutional and fully-connected layers on my CPU. I experiment with deeper networks only in the context of transfer learning, running deeper networks on a GPU.

Table 2 summarizes the results of extracting features from various fully-connected layers of deeper networks and training fully-connected layers on top to classify them into the seven emotions.

It is expected that networks trained on ImageNet would not perform well classifying images of human faces into emotional categories — ImageNet is comprised of 1000 classes ranging from dogs to cars. By the last fully-connected layers of the networks, the network is close to predicting what class of object the image falls in. These features are understandably not good at differentiating between a sad face and a happy one.

I did however expect the VGG Face and EmotiW networks to perform better than they did. The VGG Face Descriptor did no better than random, and the EmotiW network only slightly so, even though it was trained to recognize six basic emotions and did so at 55% accuracy on its test set. With more time I would like to figure out why the accuracy is not closer to 50% (or higher, since CK+ data is taken under controlled lab conditions). (I have not ruled out a bug in my code.)

6. Conclusion

Convolutional neural networks are a promising method for classifying human facial expressions into the seven basic emotions. On the Cohn-Kanade dataset, a three-layer CNN with rotation-augmented data is able to achieve a test set accuracy of 38%, significantly better than the SVM baseline and random guessing.

Adding fully-connected layers atop extracted features from deeper networks did not yield high classification accuracies. Transfer learning on the EmotiW network created by Levi *et al*. gave the highest test set accuracy of 30%, much lower than the 55% cited in their work.

A logical next step is to not only extract features from deep networks but train their parameters using the CK+ dataset at a low learning rate. I expect this can dramatically increase classification accuracy. Other future directions include trying more data augmentation methods as well — reflection, more rotation angles, cropping, obscuring parts of the image, etc. Given more time I also would have liked to visualize saliency maps and generate images for each emotion class.

References

[1] J. J. Bazzo and M. V. Lamar. Recognizing facial actions using gabor recognizing facial actions using gabor wavelets with neutral face average difference. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, May 2004.

[2] C.-F. Chuang and F. Y. Shih. Recognizing facial action units using independent component analysis and support vector machine. *Pattern Recognition*, 39:1795–1798, 2006.

[3] C. Darwin. *The Expression of the Emotions in Man and Animals*. John Murray, 1872.

[4] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon. Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 2106–2112, November 2011.

[5] G. Donato, M. S. Bartlett, J. C. Hager, P. Ekman, and T. J. Sejnowski. Classifying facial actions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):974–989, October 1999.

[6] P. Ekman and W. V. Friesen. Constants across cultures in the face and emotion. *Journal of Personality and Social Psychology*, 17(2):124–129, 1971.

[7] P. Ekman and E. L. Rosenberg. *What the Face Reveals: Basic and Applied Studies of Spontaneous Expression Using the Facial Action Coding System (FACS)*. Oxford University Press, 1997.

[8] R. E. Jack, O. G. Garrod, and P. G. Schyns. Dynamic facial expressions of emotion transmit an evolving hierarchy of signals over time. *Current Biology*, 24(2):187–192, 2014.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2012.

[10] G. Levi and T. Hassner. Emotion recognition in the wild via convolutional neural networks and mapped binary patterns. *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 503–510, November 2015.

[11] Y. li Tian, T. Kanade, and J. F. Cohn. Recognizing action units for facial expression analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):97–115, February 2001.

[12] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, and Z. Ambadar. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, 2010.

[13] D. Matsumoto, D. Keltner, M. N. Shiota, M. G. Frank, and M. O'Sullivan. *Handbook of emotions*, chapter What's in a face? Facial expressions as signals of discrete emotions, pages 211–234. Guilford Press, New York, 2008.

[14] H.-W. Ng, V. D. Nguyen, V. Vonikakis, and S. Winkler. Deep learning for emotion recognition on small datasets using transfer learning. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 443–449, November 2015.

[15] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *Proceedings of the British Machine Vision Conference*.

[16] R. Picard. *Affective Computing*. MIT Press, 1997.

[17] P. Shaver, J. Schwartz, D. Kirson, and C. O'Connor. Emotion knowledge: Further exploration of a prototype approach. *Journal of Personality and Social Psychology*, 52(6):1061–1086, June 1987.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.