CS231N Project: Facial Expression Recognition

Hsiao Chen Chang, Emilien Dupont, William Zhang

March 13, 2016

{hcchang7, edupont, wzhang4}@stanford.edu

Stanford University
CS231N Final Report

Abstract

In this project, we investigated ways to leverage the representational power of convolutional neural networks to distinguish between seven emotions from pictures of facial expressions. We implemented a conv-reLU-pool architecture (with an increasing number of filters for deeper layers) followed by two affine layers and softmax loss. To eke out some extra accuracy, we also drew inspiration from knowledge distillation. Finally, we made use of saliency maps to get a sense of what our neural net managed to/failed to pick up on. Our final validation accuracy on the test was 60.7%.

1 Introduction

Facial expression is an important indicator of a person's emotion. Computers and other electronic devices in our daily lives will become more user-friendly if they can adequately interpret a person's facial expressions, thereby improving human-machine interfaces. Facial expression recognition can be implemented in all computer interfaces, automated psychological research and treatment, robots or even polygraphs. [3]

1.1 Dataset

The data we use is comprised of 48 × 48 pixel grayscale images of faces from the Kaggle competition *Challenges in Representation Learning: Facial Expression Recognition Challenge* [9]. The training set consists of 28,709 examples, while both the test and validation sets are composed of 3,589 examples. This data set contains photos and labels of seven categories of facial expressions/emotions: *Anger, Disgust, Fear, Happy, Sad, Surprise, Neutral.* These images have already been somewhat 'preprocessed': they are mostly centered and adjusted so that the face occupies about the same amount of space in each image.
The class distribution of the dataset can be found in Figure 1:

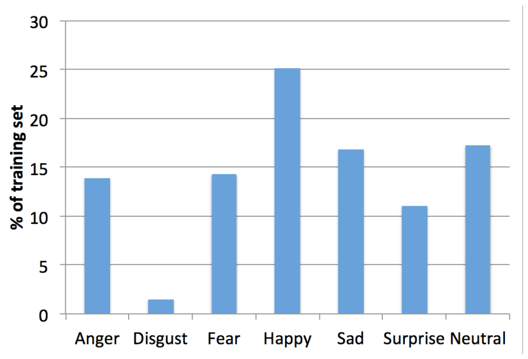


Figure 1: Distribution of the emotions

As can clearly be seen, *Disgust* is heavily underrepresented at roughly 1.5%, and *Happy* the most populated class. This will come into play later (see section 5).

A few example images from the dataset are shown in Figure 2.

1.2 Problem Statement

The goal of this project is to predict, from the grayscale picture of a person's face, which emotion the facial expression conveys. Our evaluation metric will be the accuracy for each emotion (fraction of correctly classified images), supplemented by a confusion matrix which highlights which emotions are better recognized than others. In short,

- **Input** : 48 by 48 grayscale image of a face

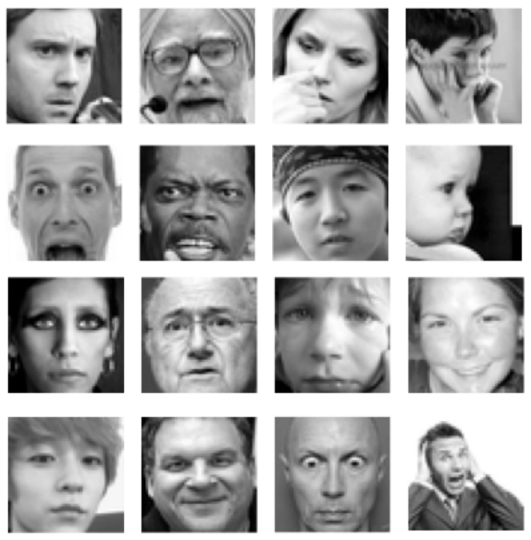


Figure 2: Example images from the dataset.

- **Output** : Emotion conveyed by facial expression

2 Background/Related Work

The problem of classifying emotions from facial expressions in images is widely studied. One of the first papers to apply neural nets to this end is the EMPATH paper from 2002 [4]. It proceeds by performing Gabor filtering on the raw images followed by various transformations and PCA before applying a 3 layer neural net. More recently, papers such as ([10], [12], [16]) have used deep neural and convolutional neural networks to classify emotions. Most of these papers focus on classifying emotions in video footage or based on audiovisual data (mixing speech recognition and video techniques). Many papers seek to recognize and match faces (e.g. [12]), but most papers do *not* use convolutional neural networks to extract emotions from still images. An exception to this is a paper by Kahou et al. which ([10]) actually trains a deep convolutional neural network on a set of static images, but then applies this to video data.

In addition to the Kaggle Competition, there is also another competition for emotion recognition in static images and videos, the *Emotion Recognition In The Wild Challenge* [5]. In particular, the winners of the 2013 competition use convolutional neural networks for classification.

Finally, the winner of the Kaggle competition used a deep neural net (based on CIFAR-10 weights) to extract features and then SVM for classification [17].

3 Technical Approach

3.1 Methodology

We chose Torch 7 [1] as our framework to train neural networks on. Torch is a scientific computing framework with wide support for machine learning algorithms including popular neural network and optimization libraries which are simple to use. It is easy to use, efficient and beginner friendly.
As the Kaggle data came in the form of a .csv file, with each entry as a list of grayscale pixel values, we chose to mean-center and normalize it into a 32-bit array in Python. This array was then saved to an HDF5 file, and subsequently read into Torch.

3.2 Baseline

As a baseline/starting point, we trained a three-layer convolutional network on the data set, using the same architecture as that proposed in Assignment 2 [11], namely: 5×5 Conv - ReLU - 2×2 max-pool - affine - ReLU - affine - softmax.

3.3 Data augmentation

To compensate for the relatively small size of the dataset, we made use of the popular data augmentation technique that consists in flipping images horizontally. As emotions should intuitively not change based on whether or not facial expressions are mirrored, it seemed a sensible choice.

3.4 Architecture

The final architecture retained can be described as follows:

- 3×3 Conv - ReLU - 2×2 Max-Pool with 32 filters
- 3×3 Conv - ReLU - 2×2 Max-Pool with 64 filters
- 3×3 Conv - ReLU - 2×2 Max-Pool with 96 filters
- 3×3 Conv - ReLU - 2×2 Max-Pool with 128 filters
- FC hidden layer with 200 hidden units
- FC layer to 7 class scores

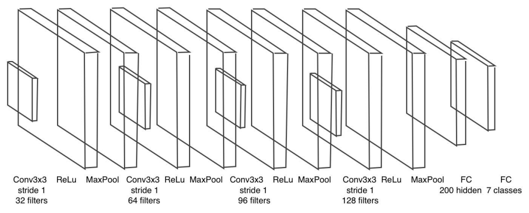


Figure 3: Architecture of the neural net.

A pictorial description of this architecture can be found in Figure 3. The loss function we used was the Softmax loss. The update rule we used was the Adam update.[6] Adam is a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement. We have explored different update rules in our model, including SGD (stochastic gradient descent), Adagrad, RMSProp. It turns out that Adam performs the best among these update rules, slightly better than RMSProp.

4 Experiments/Optimizing the net

4.1 Parameters

To choose the parameters (regularization strength, learning rate and decay), we randomly sampled in log space and retained those that yielded the best validation accuracy. The parameters we ended up using are the following:

- **Batch size**: 50
- **Learning rate**: 0.00127
- **L2 regularization strength**: 0.00172
- **Learning rate decay**: 0.95
- **Hidden layer dimensions**: 200
- **Number of filters**: [32,64,96,128]
- **Size of filters**: 3×3
- **Number of epochs**: 40

4.2 Regularization

One of the common problems we encountered during neural network training is over-fitting. The neural network was driving the error on the training set to a very small value, but this was not the case for the validation error, leading to an increasing gap between the two.
In order to improve neural network generalization and avoid over-fitting, we have experimented with different common regularization techniques listed in Assignment 2[11]: L2 regularization, dropout, and batch normalization. One of the models we trained in this project includes the batch normalization immediately after every convolutional layer. Batch-normalization [14] performs normalization for each training mini batch (which is 50 in our model) so that we can use higher learning rates and make our model significantly more robust to bad initialization. It can alleviate the problem of the change of the distribution of each layer's input. The accuracy improved by a few percentage points as a result. In addition to batch-normalization, we also made use of dropout [13], but unfortunately did not observe significant improvements.

4.3 Knowledge distillation

In order to eke some extra performance out of neural network, we drew inspiration from [8]: the idea behind 'distilling' knowledge into neural networks is that the output of a network (the stage right before an actual prediction on an input is made) is akin to a probability distribution, representing what the network believes to be probabilities of the given input as belonging to each of the seven classes. Intuitively, if a network is very accurate, then these probability vectors will contain much more information, especially for the backward passes, than the typical one-hot vectors. In the case of deep neural networks that are computationally expensive to run on certain architectures (e.g. a mobile phone), the aforementioned probability vectors are used in conjunction with the usual one-hot vectors to feed more useful information to a smaller network, yielding better results than using one-hot vectors alone. Naturally, this requires a separate loss function, which is a weighted average of the softmax loss for one-hot vectors and the Kullback-Leibler Divergence:

$$\text{loss}(x_i) = \alpha \left(-f_{y_i} + \log \textstyle\sum_j e^{f_j}\right) + (1-\alpha) \textstyle\sum_j \tilde{y_j} \left(\log \tilde{y_j} - f_j\right) \quad (1)$$

where α is a number between 0 and 1, f_j the score for class j, and $\tilde{y_i}$ the probability vector output by a previous network.
Note that distillation is usually applied when we have a very large and accurate network and wish to train a smaller network which achieves similar results. In our case, we experimented with using the probability outputs of our network and then we fed the probability dis-

tribution back in to that *same* network to improve the accuracy. The authors of [8] recommend an α in the vicinity of 0.1; however, in our case, since we are not distilling knowledge from a larger network to a smaller one, we chose an α in the vicinity of 0.8.

5 Results

The final validation accuracy we obtained is 60.7%.

An example plot of our accuracy evolution can be found in Figure 4. As can be seen, the training accuracy increases while the test accuracy remains almost constant after about 15 epochs. This means we are slightly overfitting our data.

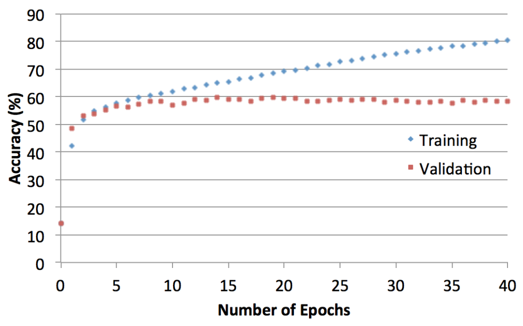


Figure 4: Training and validation accuracy over 40 epochs for a typical network we trained.

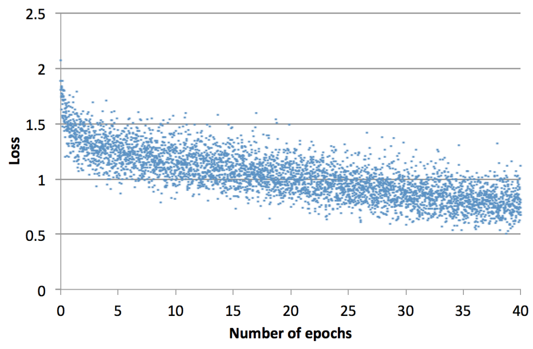
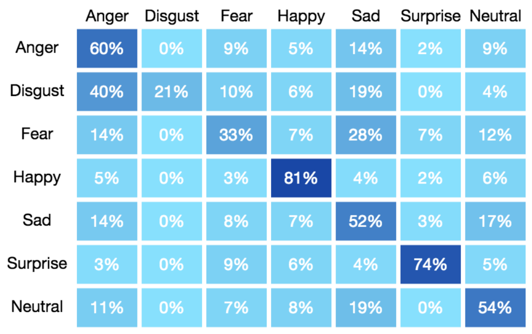


Figure 5: Loss History

The confusion matrix on the validation set is shown in 6.

As can clearly be seen, *Disgust* is the class where our network fares the worst, and *Happy* where it is most suc-



	Anger	Disgust	Fear	Happy	Sad	Surprise	Neutral
Anger	60%	0%	9%	5%	14%	2%	9%
Disgust	40%	21%	10%	6%	19%	0%	4%
Fear	14%	0%	33%	7%	28%	7%	12%
Happy	5%	0%	3%	81%	4%	2%	6%
Sad	14%	0%	8%	7%	52%	3%	17%
Surprise	3%	0%	9%	6%	4%	74%	5%
Neutral	11%	0%	7%	8%	19%	0%	54%

Figure 6: Confusion matrix. The rows correspond to the true values and the columns correspond to our predictions.

cessful - a fact very likely due, at least in part, to the uneven class distribution mentioned in section 1.1.

5.1 Saliency Maps and important facial features

Ekman et al [2] made a study in 1975 of which parts of the face are most important for different emotions. It turns out this depends on different emotions: for instance, anger is often detected on the upper part of the face whereas happiness is often detected in the lower part of the face.

Now, it is interesting to observe which parts of the face the network considers as ”important”. To do this we use *saliency maps* as defined in a paper by Simonyan et al [15]. Saliency maps allow us to see which parts of the image the net ”reacts” to and which it doesn’t. For most images, the saliency map do not show much interesting stuff except that the net reacts to the face and not the background. For some images however interesting behaviour is observed as described in Figure 7.

5.2 Human performance

In an effort to compare our results to human performance, each of us attempted to guess emotions for a 100 randomly chosen faces. On these examples we achieved an accuracy of 63.4% on average. This seems to be quite low, but the images in the dataset are very small and some emotions are very difficult to distinguish. Given larger and clearer images, human accuracy is much higher. The results of a psychological study made by Ekman et al [7] are shown in Table 1.

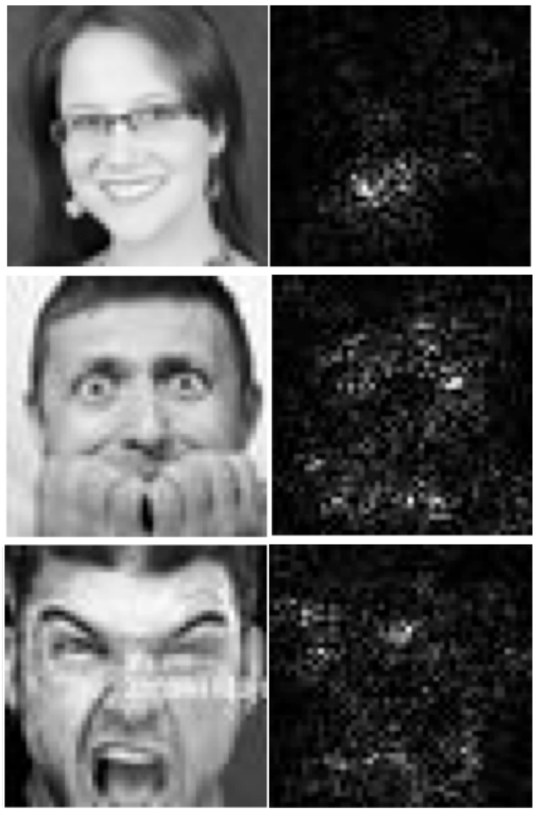


Figure 7: Three images and their corresponding saliency map. In the first image we see a very strong reaction to the smile of the woman, and the net successfully classifies the image as ”Happy”. In the second image, the neural net reacts mostly to the eye of the man and classifies the image as ”Fear”. In the last image, the net reacts to the frown and eyebrow of the man and classifies the image as ”Anger”.

However, it does not make much sense to compare our results to this, since this is emotion recognition for human faces and not grayscale 48 by 48 images with occasionally dubious labeling. Also, in the psychological experiment, the testers could detect a series of facial changes in real time instead of one input image as is the case for our problem.

Finally, it should also be noted that other participants in the Kaggle challenge report human accuracies (around 60%) close to our own.

Emotion	Accuracy (%)
Happiness	98.7
Surprise	92.4
Disgust	92.3
Anger	88.9
Sadness	89.2
Fear	87.7
Neutral	91.6

Table 1: Results of psychological study

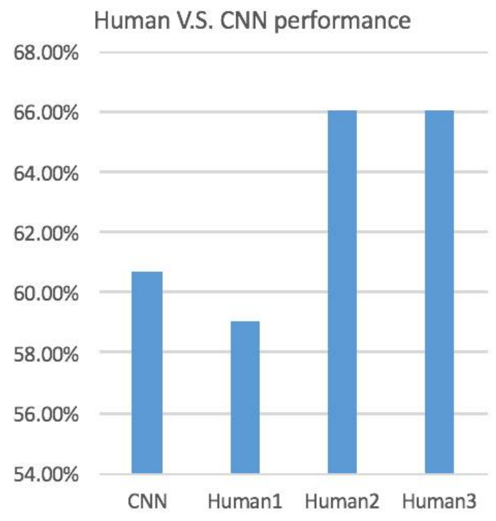


Figure 8: Comparison of human performance and our neural net.

6 Conclusion

In this project, we built a convolutional neural network to recognize emotion from grayscale pictures of faces. We used a Conv-ReLU-Pool architecture and softmax loss. For regularization, we experimented with batch normalization, L2 regularization and dropout, but only retained the first two in our final net. To optimize the net, we used ADAM with learning rate decay.

The final validation accuracy we obtained using this architecture was 60.7%. Some of the difficulties with improving this is that the images are very small and in some cases it is very hard to distinguish which emotion is on each image, even for humans. To understand how the neural net classified different images we used saliency maps, to detect important regions in the images according to the neural net. Even though most results where quite noisy, some images showed convincing results (for example a smile or a frown leading to happiness and anger respectively).

Even though our validation accuracy is fairly high, we believe that adding more layers and more filters would further improve the network. In future work it would be interesting to try this approach as well as building a classifier using pre trained nets (on e.g. CIFAR-10).

Code for this project can be found at the following github repo.

References

[1] Torch: Scientific computing for luajit. 2

[2] J. D. Boucher and P. Ekman. Facial areas and emotional information. 1975. 4

[3] Roberto Cipolla and Alex Pentland. *Computer vision for human-machine interaction*. Cambridge university press, 1998. 1

[4] Matthew N Dailey, Garrison W Cottrell, Curtis Padgett, and Ralph Adolphs. Empath: A neural network that categorizes facial expressions. *Journal of cognitive neuroscience*, 14(8):1158–1173, 2002. 2

[5] Abhinav Dhall, Roland Goecke, Jyoti Joshi, Karan Sikka, and Tom Gedeon. Emotion recognition in the wild challenge 2014: Baseline, data and protocol. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 461–466. ACM, 2014. 2

[6] Jimmy Lei Ba Diederik P. Kingma. Adam: A method for stochastic optimization. 2015. 3

[7] Paul Ekman, Wallace V Friesen, and Consulting Psychologists Press. *Pictures of facial affect*. consulting psychologists press, 1975. 4

[8] Jeff Dean Geoffrey Hinton, Oriol Vinyals. Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning Workshop*. 3, 4

[9] Kaggle. Kaggle competition: Challenges in representation learning: Facial expression recognition challenge, 2013. 1

[10] Samira Ebrahimi Kahou, Christopher Pal, Xavier Bouthillier, Pierre Froumenty, Çaglar Gülçehre, Roland Memisevic, Pascal Vincent, Aaron Courville, Yoshua Bengio, Raul Chandias Ferrari, et al. Combining modality specific deep neural networks for emotion recognition in video. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, pages 543–550. ACM, 2013. 2

[11] Andrej Karpathy and Justin Johnson. Assignment 2: Fully-connected nets, batch normalization, dropout, convolutional nets. 2, 3

[12] Thai Hoang Le. Applying artificial neural networks for face recognition. *Advances in Artificial Neural Systems*, 2011:15, 2011. 2

[13] Alex Krizhevsky Ilya Sutskever Ruslan Salakhutdinov Nitish Srivastava, Geoffrey Hinton. Dropout: A simple way to prevent neural networks from overfitting. 2014. 3

[14] Christian Szegedy Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 3

[15] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013. 4

[16] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1891–1898, 2014. 2

[17] Yichuan Tang. Deep learning using support vector machines. *CoRR, abs/1306.0239*, 2013. 2