Facial Expression Recognition

1. Introduction

Facial recognition has been a research area starting from 1960's. it is an application of identifying a person through images or video frameworks. It has a wide range of application in security system, and social media. Recent years, the approach of deep learning achieve an dramatically improvement of accuracy.

1.1 Project Overview

In this project, we will implement an algorithm to take the task of recognizing facial expression with a given dataset. The dataset is Toronto Faces Dataset (TFD), which contains 4178 images cropped on different people's face. There are 7 categories of facial expression: 1-Anger, 2-Disgust, 3-Fear, 4-Happy, 5-Sad, 6-Surprise, 7-Neutral. One example face of each class is displayed below in Figure 1.

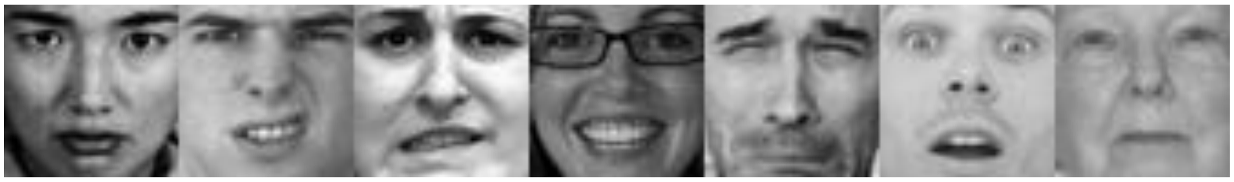


Figure 1

1.2 Problem Statement

The problem defined is to get a classifier that is able to classify a person's emotion from their facial expression given the image of their face. The classifier should work well on unseen pictures. There are multiple approach to solve this problem, such as principal component analysis using eigenfaces, linear discriminant analysis, elastic bunch graph matching using the Fisherface algorithm, the hidden Markov model[1]. However, in this project the approach of convolutional neural network (CNN) will be used for two reasons: (1) With the growth of computational power and the amount of data, deep learning has been showing great strength on image recognition task. (2) Convolutional neural network can capture spatial features therefore are more suitable than regular neural network and recurrence neural network. The strategy for implementing this approach is first preprocessing the given data and build a smaller working CNN as a prototype, and then refines it by tuning relevant hyper-parameters through their performance on validation set. Finally, test the final version model on the test set.

1.3 Metrics

There are two metrics to evaluate the model: accuracy rate and the cross entropy. They will be further explained in section 4.1. In addition, the dataset is divided into three set: training, validation, and test. During the training session, the evaluation is based on the accuracy rate and cross-entropy on the prediction over the validation set. After the training session, the accuracy of predictions on the test set will be evaluated as the final score. They formula of cross-entropy and predicting accuracy are shown below, where y is the predicted value, t is the true label, and i is the index of output class.

$$\text{Cross Entropy: } H(y,t) = -\sum_i t_i \log y_i$$

$$Prediction\ Accuracy = \frac{number\ of\ correct\ predictions}{number\ of\ total\ predictions}$$

2. Analysis

2.1 Dataset Exploration & Visualization

The dataset contains 4178 gray scale images of face that are already properly cropped as well as the corresponding labels. Each image has 48x48 pixel. In this project, the dataset is divided as: 3374 training data, 419 validation data, and 385 test data. The statistics of the training data is showed in figure 2. Although class 7 and class 4 appears at higher frequency than others, all classes can be considered quite well distributed and requires no further techniques for skewness.

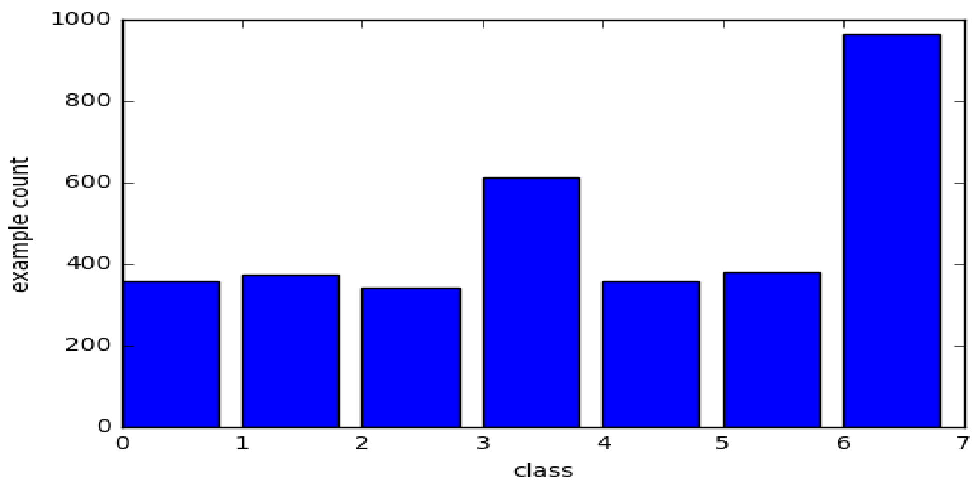


Figure 2

A conventional way to split the existing dataset by machine learning engineers is to divide it into three parts: training set, validation set and test set. Training set is used for training the model, a convolutional neural network in our case. Validation set is used to monitor the model's performance during the training session, but should not be trained by the model. The main purpose of validation set is to check the overfitting. Test set is only used when the model has already been fully trained and tuned, and it should only be used once, which is the final test.

2.2 Algorithms and Techniques

The algorithm chosen is a convolutional neural network (CNN)

2.2.1 Architecture

A convolutional neural network is designed to solve this problem. It consists of 3 convolutional layers followed by 3 pooling layers respectively, and two fully connected layers. The entire structure is showed in figure 3.

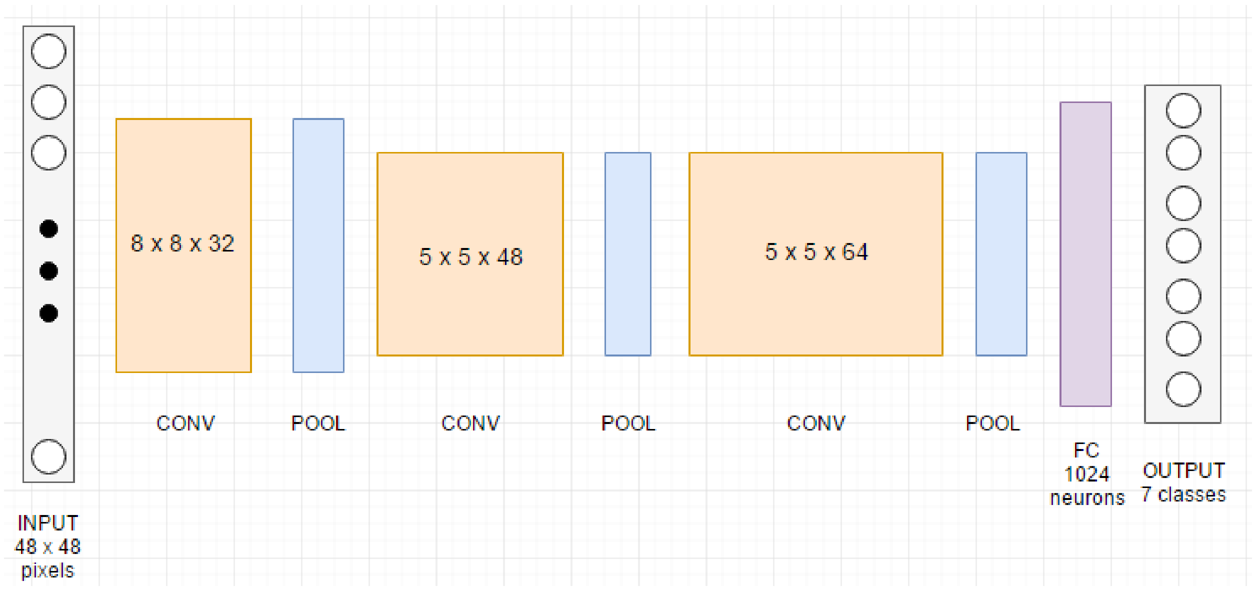


Figure 3

2.2.2 Input layer

The input of the neural network are the images. Each image is 48 x 48 pixels and only one channel (grayscale). Thus, the input layer would consist a batch of 2304 raw pixels as features.

2.2.3 Convolutional Layers

The convolutional layer is the core part of CNN, each filter in a convolutional layer use a window (8 x 8 for my first CONV layer, and 5 x 5 for the other 2 CONV layers) to slide through the entire image. This whole process is same as the spatial convolution in signal processing. By convention, the window slides with stride of 1, and zero padding is added to maintain dimensionality. The weights in a window will be shared during the convolution. The depth of a CONV layer represents how many filters it has, each filter has different set of weights (25 number of weights for 5x5 window size) that can capture different features after the training. An example of this mechanism is showed in figure 4. [2]

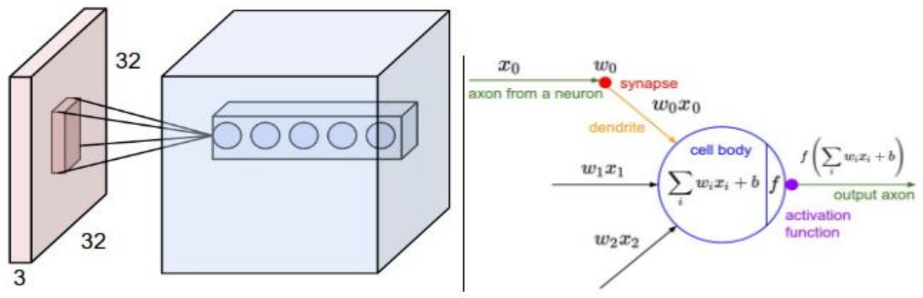
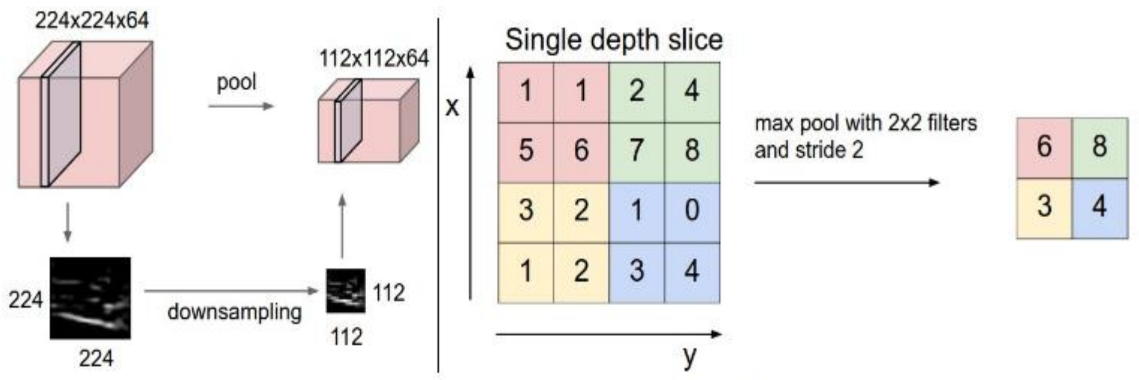


Figure 4

2.2.4 Pooling layer

The pooling layer is doing the job of down-sampling to reduce the computation. 2 by 2 window is used the extract the pixel with maximum value among the 4 pixels. An graphic illustration is showed in figure 5. [2]



Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

Figure 5

2.2.5 Fully Connected layer

Fully connected layers are used in the last two layer of our neural network. Each neuron is connected to every neuron of the next layer. Each neuron to neuron connection has a weight updating at each training step. The final output layer has seven neurons represent the likelihood of 7 classes respectively.

2.2.6 Overfitting Prevention

One of the key point to almost every machine learning application is how to prevent overfitting. In this model, “dropout” is applied in the fully connected layer. During each training step, a randomly selected half of the neurons would be closed, and block their connections toward the next layers. Since we will have thousands of training steps, each neuron is activated with a probability (dropout rate). After the training ended, all the weights will be scaled back. This technique prevents the neural network study over complicated non-linear features and prevent overfitting. Another technique used is early stop. The performance of the model on validation set is monitored entirely, so we can choose to stop the training when validation curve begins dropping.

2.2.7 Back Propagation

At the output layer, the output value will be normalized by "Softmax" function, and compared with the labels of the training image to get the cross entropy, which represent how "confident" we are. The formula of cross entropy and Softmax function are shown below, where y is the predicted value, t is the true label, and i is the index of output class.

$$\text{Cross Entropy: } H(y,t) = -\sum_i t_i \log y_i$$

$$\text{Softmax: } \mathrm{P(class\ i)} = \frac{\mathrm{e}^{\mathrm{y_i}}}{\sum_{\mathrm{i}} \mathrm{e}^{\mathrm{y_i}}}$$

The objective is to minimize the cross entropy. Then the error will then backpropagated. Using partial derivatives and chain rule from calculus, each weight will be assigned the "error" it is responsible for. After the error for each neuron is assigned, using convex optimization technique can make each weight move toward to correct direction and finally reach a local optimum. The optimizer chosen is Adam optimizer, which usually used with a small learning rate.

2.3 Benchmark

Since random guess among one of seven categories would has an accuracy of 1/7, which is 14.3% correctness. Since all images are pre-cropped which reduce the challenge of the task, the benchmark of this project is to achieve an accuracy above 70% over the test set.

3. Methodology

3.1 Data Preprocessing

The only preprocessing for this project is to convert the integer label into one-hot key format, for example, the face of fear which labeled as 3 will be converted to vector [0, 0, 1, 0, 0, 0, 0].

3.2 Implementation

The entire project is run in Ubuntu 14 environment on an Amazon Web Service(AWS) EC2 instance. The EC2 instance used is p2.xlarge which has 1 GPU, four 2.7 GHz CPU units and 61 GB memory. All the coding is implemented by Python 2.7 with Numpy and Tensorflow. Fortunately, During the implementation, no techniques and metrics required to be change before getting the solution. However, in Tensorflow, the concept such as “Graph”, “Placeholder”, and “Session” are quite confusing to new user and cost lots of time to comprehend.

3.3 Refinement

The only refinement for this model is “parameter tuning”. After tuning each hyper-parameter while holding others as constant value, the final value of hyper-parameters is shown in table 1.

Learning rate	1e-4
Activation function	Rectified linear unit (RELU)
Convolutional window size	8 x 8, 5 x 5, 5 x 5 respectively
No. filters in convolutional layers	32, 48, 64 respectively
Pooling window size	2 x 2
No. neuron in FC layer	1024
Batch size	241

Table 1

During the refinement, I found most parameters are quite robust except learning rate and batch size, so others were hold as constant. The parameters setting and the test result of initial, intermediate, and final solutions are shown below. Since our training data has 3374 images, batch size chosen are those that can divide 3374 without remainder.

	Initial solution	One of intermediate solution	Final solution
Learning rate	1e-4	1e-3	1e-4
Batch size	14	482	241
Validation accuracy	61.3%	73.8%	79.2%

Table 2

4. Result & Discussion

4.1 Model Evaluation

The cross-entropy and prediction score for both training set and validation set during the training session are showed in figure 6 and figure 7.

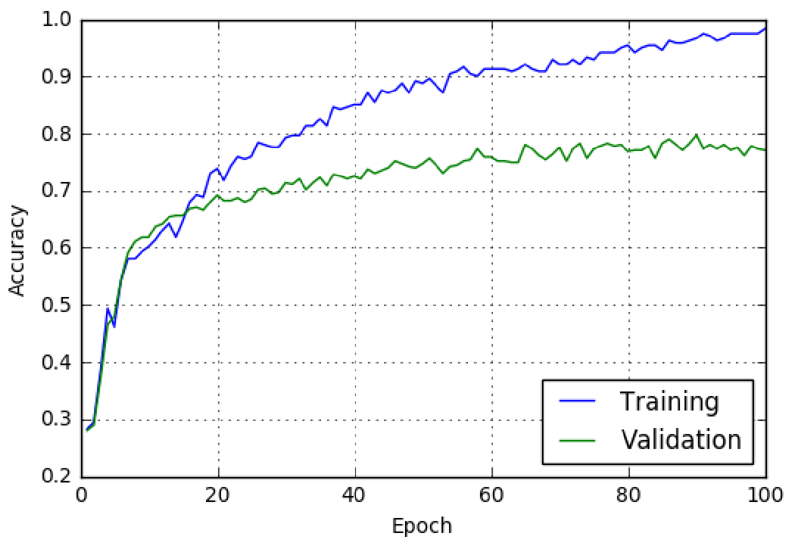


Figure 6

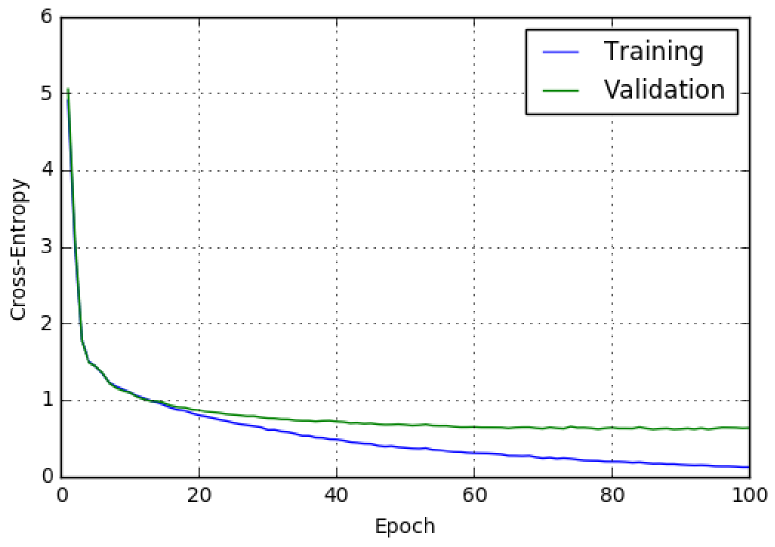


Figure 7

The test set has 385 images which are unseen by the model before to ensure the credibility of the result. The model achieves an accuracy of 79.5 percent correctness on the test set. The model has a good robustness of input data but not the input space, since the neural network is designed for taking input images that are in the format of 48x48 grayscale pixels. Two sample predicting results on test set are shown in table 2.

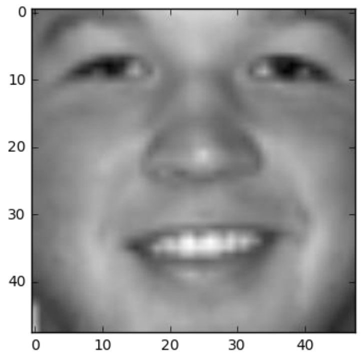
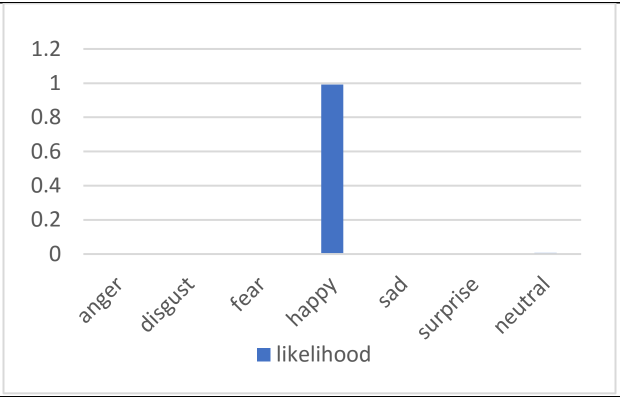
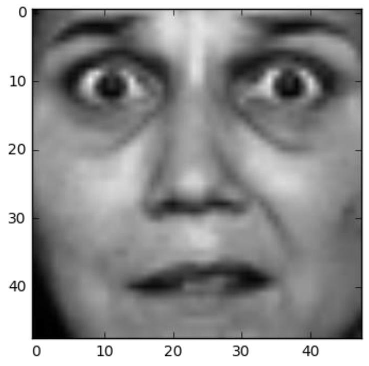
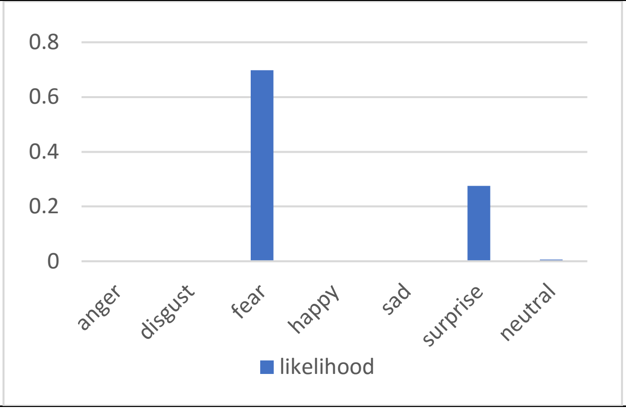
Sample test images	Predicted Likelihood	label
		Happy
		Fear

Table 3

4.2 Justification

In theory, convolution layers can be trained to extract different spatial features of images, and fully connected layer can map those extracted feature toward the likelihood of being each category through the training. The final accuracy 79.4% surpasses the benchmark 70% by almost 10 percent. Therefore, through this project, we verified that the solution of using convolutional neural network is significant enough to handle the facial expression recognition problem.

5. Conclusion

5.1 Visualization

A quantitative visualization of the model performance compare to the benchmark and random guess is showed in figure 8, which illustrate that the performance of our model reaches our goal.

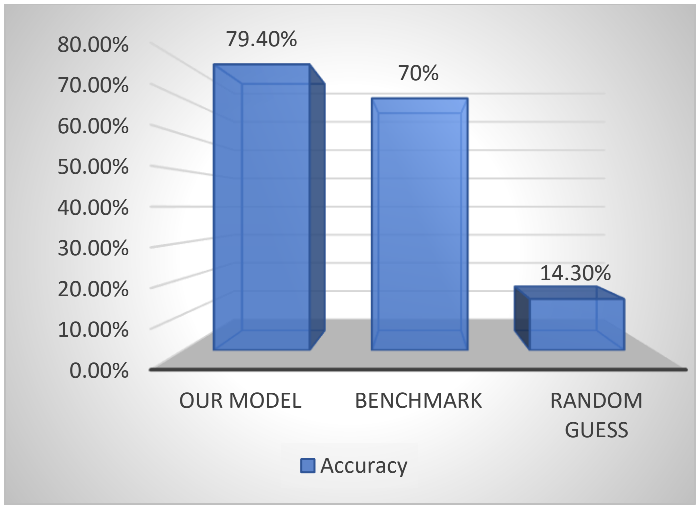


Figure 8

5.2 Reflection

In summary, the solution to this problem is applying a convolutional neural network which consists 3 CONV layers followed by pooling and a fully connected layer. A dropout is applied at the fully connected layer to prevent overfitting. One interesting part is how to choose the number of layers and neurons since grid search of all combination of those parameter setting is not realistic. Even today, it is considered as a black art and we only know that deeper network can solve harder problems. The major difficulty of this project is the computational power, since training a neural network is always time consuming and requires high quality hardware.

5.3 Improvements

There are couple of improvements can be made in the future. Firstly, we can increase the dataset by taking some geometric operation such as rotating by a small angle or flipping horizontally. More training data can always benefit the model. Secondly, ensemble method can be used for further improvement. For example, bagging method. We can train multiple neural network on different subset of the training data and averaging their predicting likelihood in the prediction.

6. References

[1] https://en.wikipedia.org/wiki/Facial_recognition_system

[2] http://cs231n.github.io/convolutional-networks/

[3] https://www.tensorflow.org/tutorials/mnist/pros/

[4] http://research.microsoft.com/en-us/um/people/zhang/Papers/IJPRAI.pdf

[5] http://cs231n.stanford.edu/reports2016/023_Report.pdf