

Final Project

# Case Study in Recognition of Emotion from Images

Mason, Carolyn



CSCI E-89 Deep Learning, Spring 2019  
**Harvard University Extension School**  
Prof. Zoran B. Djordjević

# Problem Statement

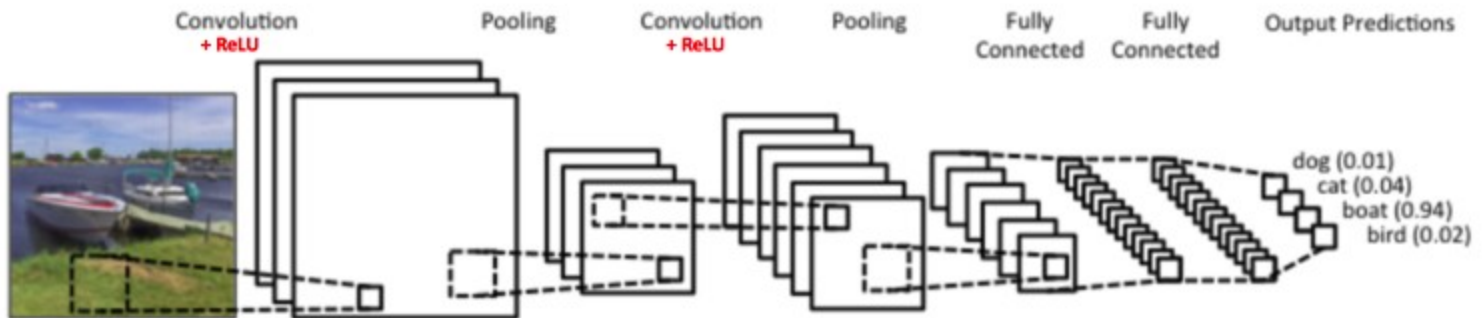
- **Problem:**
- It can take a lot of time to sort through and choose images to share with friends and family. It can also be hard to come back to an album years later and find a specific sets of pictures. Deep learning tools can help speed up this process and pinpoint images of a specific style that you are searching for. This tool is an emotion detector, allowing the user to pick the top 'X' images from their photo album. The user can choose from angry, happy, sad, and neutral.

# Software and Technology

This project demonstrates python, Keras, and CNN (Convolutional Neural Networks).



# Keras

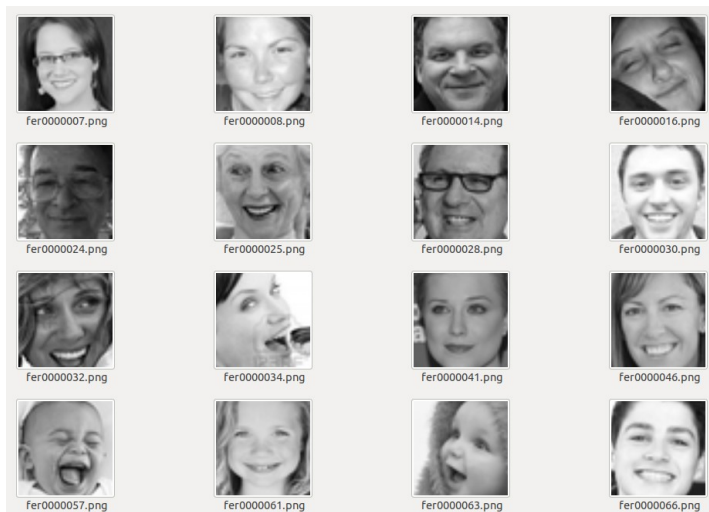


# Data Set

## FER+ data set

- Size of Data Set: 28,709 pictures
- What: Framed faces showing emotion
- Color: Grayscale
- Size: 48x48 pixels
- Labels: Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral
- Location:

<https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data>



	A	
1	emotion	pixels
2	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121 119 115 110 98 91 84 84 90 99 11
3	0	151 150 147 155 148 133 111 140 170 174 182 154 153 164 173 178 185 185 189
4	2	231 212 156 164 174 138 161 173 182 200 106 38 39 74 138 161 164 179 190 201
5	4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 19 43 52 13 26 40 59 65 12 20 63 9
6	6	4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84 115 127 137 142 151 156 155 149 1
7	2	55 55 55 55 55 54 60 68 54 85 151 163 170 179 181 185 188 188 191 196 189 194
8	4	20 17 19 21 25 38 42 42 46 54 56 62 63 66 82 108 118 130 139 134 132 126 113 9
9	3	77 78 79 79 78 75 60 55 47 48 58 73 77 79 57 50 37 44 56 70 80 82 87 91 86 80 7
10	3	85 84 90 121 101 102 133 153 153 169 177 189 195 199 205 207 209 216 221 225
11	2	255 254 255 254 254 179 122 107 95 124 149 150 169 178 179 179 181 181 184 1
12	0	30 24 21 23 25 25 49 67 84 103 120 125 130 139 140 139 148 171 178 175 176 17
13	6	39 75 78 58 58 45 49 48 103 156 81 45 41 38 49 56 60 49 32 31 28 52 83 81 78 7
14	6	219 213 206 202 209 217 216 215 219 218 223 230 227 227 233 235 234 236 237
15	6	148 144 130 129 119 122 129 131 139 153 140 128 139 144 146 143 132 133 134
16	3	4 2 12 41 56 62 67 67 65 62 65 70 80 107 127 140 152 150 165 169 177 187 176

# Overview of Steps

- 1) Download the data from Kaggle
- 2) Setup test and train data sets
- 3) Build the CNN
- 4) Iterate until happy with the model results
- 5) Run on your own images!

# Code Overview:

## 1) Download the data from Kaggle

Note- I created a light version of the data set to look at happy, sad, neutral, and angry expressions

```
# Grab csv
base_dir = '/home/carolyn/Documents/Classes/DeepLearning/week14_finalProjects/
FERPlus_data/data'
data = '/fer2013/fer2013.csv'
fer_path = base_dir+data

df = pd.read_csv(fer_path,
                 header=0,
                 names=['Emotion', 'data', 'etc'])

# Simplify the emotions
l1 = df.loc[df['Emotion'] == 0]
l2 = df.loc[df['Emotion'] == 3]
l3 = df.loc[df['Emotion'] == 4]
l4 = df.loc[df['Emotion'] == 6]

light = pd.concat([l1,l2,l3,l4])

# Fix values
light.Emotion.loc[(light['Emotion'] == 3)] = 1
light.Emotion.loc[(light['Emotion'] == 4)] = 2
light.Emotion.loc[(light['Emotion'] == 6)] = 3

# re-shuffle
light = shuffle(light)

# Save csv
light.to_csv(base_dir+'light.csv')
```

# Code Overview

## 2) Setup test and train data sets

```
# Read csv
count = 0
train_data = []
train_labels = []
test_data = []
test_labels = []

with open(fer_path, 'r') as csvfile:
    dataNum = len(csvfile.readlines())-1
    trainNum = ceil(dataNum*0.8)

with open(fer_path, 'r') as csvfile:
    fer_rows = csv.reader(csvfile, delimiter=',')
    for row in islice(fer_rows, 1, None):
        if(count < trainNum):
            train_data.append([float(s)/255. for s in row[1].split(' ')])
            train_labels.append(row[0])
        else:
            test_data.append([float(s)/255. for s in row[1].split(' ')])
            test_labels.append(row[0])
        count+=1
```

# Code Overview

## 3) Build the CNN

```
model = models.Sequential()  
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(64, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(128, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Dropout(0.25))  
model.add(layers.Conv2D(128, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Flatten())  
model.add(layers.Dropout(0.25))  
model.add(layers.Dense(512, kernel_regularizer=regularizers.l2(11ambda),  
activation='relu'))  
model.add(layers.Dropout(0.4))  
model.add(layers.Dense(num_classes, activation='sigmoid'))
```



# Code Overview

## 4) Iterate until happy with the model results

```
# Create and run model -----  
  
history1 = createAndRun(0.001)  
history2 = createAndRun(0.0005)  
history3 = createAndRun(0.0001)
```

# Code Overview

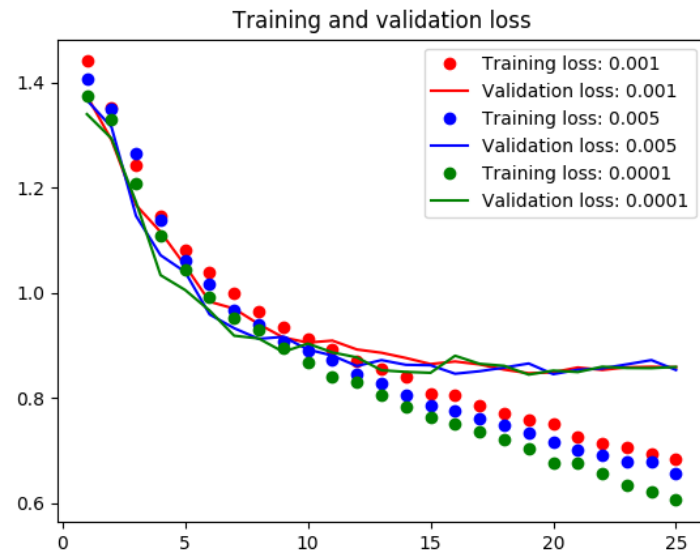
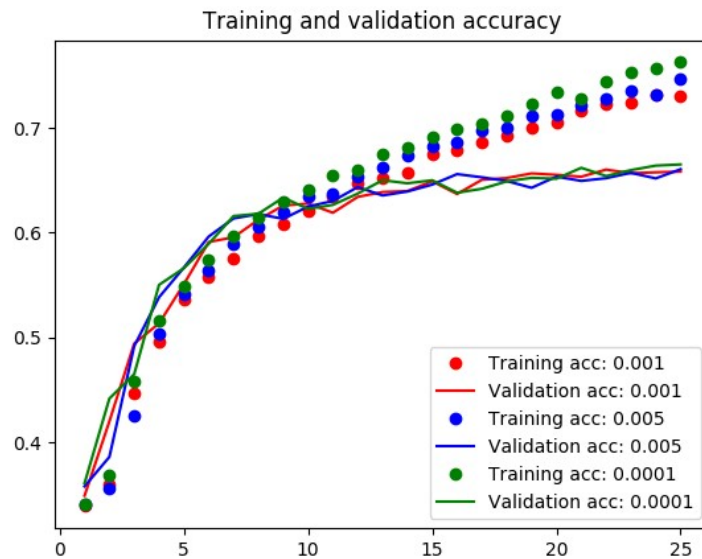
## 5) Run on your own images!

```
model =  
load_model('/home/carolyn/Documents/Classes/DeepLearning/week14_finalProjects/  
FERPlus_data/code/facial_expressions.h5')  
  
# List of labels  
emotion_table = {'0' : 'anger',      # 4953  
                  '1' : 'happy',     # 8989  
                  '2' : 'sad',       # 6077  
                  '3' : 'neutral'}   # 6198
```

# Results







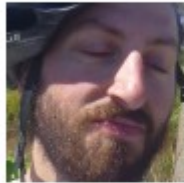






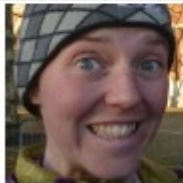






The final accuracy of the model is 66.5% for predicting four emotions of cropped pictures of the face.

The biggest point of concern, for this model, is over fitting. I was able to improve this metric by adding dropout and an L2 llambda value. Here are the plots, showing my final results:



# Results - Demonstration

For my sample data sets angry 4/5, happy 5/5, neutral 4/5, and sad 1/5.

Angry	Happy	Neutral	Sad
 <p>Name: <u>angry0.JPG</u>  anger: 100.0%  happy: 0.0%  sad: 0.0%  neutral: 0.0%</p>	 <p>Name: <u>happy0.png</u>  anger: 0.0%  happy: 100.0%  sad: 0.0%  neutral: 0.0%</p>	 <p>Name: <u>neutral0.png</u>  anger: 0.0%  happy: 0.0%  sad: 0.0%  neutral: 100.0%</p>	 <p>Name: <u>sad0.jpeg</u>  [0. 0. 0.  0.0084666246548295  No face</p>
 <p>Name: <u>angry1.png</u>  anger: 100.0%  happy: 0.0%  sad: 0.0%  neutral: 0.0%</p>	 <p>Name: <u>happy1.JPG</u>  anger: 0.0%  happy: 100.0%  sad: 0.0%  neutral: 0.0%</p>	 <p>Name: <u>neutral1.JPG</u>  [0. 0. 0. 0.]  0.0  No face</p>	 <p>Name: <u>sad1.jpeg</u>  [0.0000000e+00 0.00  1.427988119890005e  No face</p>
 <p>Name: <u>angry2.png</u>  anger: 100.0%  happy: 0.0%  sad: 0.0%  neutral: 0.0%</p>	 <p>Name: <u>happy2.JPG</u>  anger: 0.0%  happy: 100.0%  sad: 0.0%  neutral: 0.0%</p>	 <p>Name: <u>neutral2.jpeg</u>  anger: 0.0%  happy: 0.0%  sad: 0.0%  neutral: 3.0%</p>	 <p>Name: <u>sad2.jpeg</u>  [4.9207607e-09 4.543  4.920760687809889e  No face</p>
 <p>Name: <u>angry3.JPG</u>  anger: 0.0%  happy: 100.0%  sad: 0.0%  neutral: 0.0%</p>	 <p>Name: <u>happy3.JPG</u>  anger: 0.0%  happy: 100.0%  sad: 0.0%  neutral: 0.0%</p>	 <p>Name: <u>neutral3.jpeg</u>  anger: 0.0%  happy: 0.0%  sad: 0.0%  neutral: 100.0%</p>	 <p>Name: <u>sad3.jpeg</u>  [0. 0. 0. 0.]  0.0  No face</p>
 <p>Name: <u>angry4.JPG</u>  anger: 100.0%  happy: 0.0%  sad: 0.0%  neutral: 0.0%</p>	 <p>Name: <u>happy4.png</u>  anger: 0.0%  happy: 100.0%  sad: 0.0%  neutral: 0.0%</p>	 <p>Name: <u>neutral4.jpeg</u>  anger: 0.0%  happy: 0.0%  sad: 0.0%  neutral: 100.0%</p>	 <p>Name: <u>sad5.png</u>  anger: 0.0%  happy: 0.0%  sad: 100.0%  neutral: 0.0%</p>

# Results - Demonstration

I re-ran the model with every 'anger', 'happy', 'sad', or 'neutral' image from the test and training sets to see how the model performed. I found, similar results to my mini test. The table compares correct/ total = percent correct

Anger:  $1329 / 4924 = 27\%$

Happy:  $7356 / 8953 = 82\%$

Sad:  $643 / 6037 = 10.6\%$

Neutral:  $3026 / 6160 = 49\%$

# Lessons Learned

The model has an accuracy of 66.5% for four emotions. As a reference, the Kaggle competition held for this data set had a top accuracy of 71% for seven emotions. There is still work that can be done to improve this model, but the initial results are not bad.

This project taught me how to train a model with categorical data and what knobs could help improve over fitting. Including dropout and L2 regularization were helpful, but not enough to align the training and validation accuracies. I think the key lesson learned is addressing the data and improving the results from categories that do not perform as well as the others.

# Future Work

The next step toward improving this model is determining how to better train the model on the sad dataset. It would be good to get each emotion's results above 50%. Once the 4-emotion model is finished, the next step will be to reach higher accuracies while using 7+ emotions.

In the future, it would also be beneficial to add an automatic face detection/ cropping option in order to find faces in normal, more complicated images. In order to automatically find every vacation photo with 'happy' people, it would be important.

# References

Zoran's class notes

Kaggle Data set:

<https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data>

Deep Facial Expression Recognition: A Survey: <https://arxiv.org/pdf/1804.08348.pdf>

Variety of papers on facial recognition:

<https://paperswithcode.com/task/facial-expression-recognition>

Deep Learning For Smile Recognition: <https://arxiv.org/pdf/1602.00172v2.pdf>



# YouTube URLs

Two minute (short): <https://youtu.be/nrLpXucMcLM>

15 minutes (long): <https://youtu.be/mJR3sblltPs>