Emotional Recognition

By Mike Labadie and Matt Wilchek

December 3, 2019

Agenda

- Introduction to the Project
- About the Dataset
- About the Deep Learning Network and Training Algorithm Used
- Experimental Setup
- Results
- Summary and Conclusion

Introduction

Emotional Recognition

- Popular research area involving computer vision, machine learning, and behavioral sciences
- Can be applied in many business areas such as security, human-computer-interaction (HCI), health care, and advertising
- Continues to be a research area that is actively evolving with better methods for data pre-processing and algorithm training
- Motivation: Wanted to apply image classification experience gained from this class to a more complex dataset

The Data

Dataset: Carnegie Mellon University

- Original Goal: develop a good dataset to train emotion detecting models
  - Goal of this research was not to directly build predictive models
- Experiments conducted in 2000 and 2010
- Sequences of images of a person going from no emotion to some “peak” emotional state
- Emotions: Anger, Contempt, Disgust, Fear, Happy, Sadness and Surprise.
- Dataset
  - Thousands of images (varies by person and by sequence)
  - Hundreds of subjects
  - Often several sequence per subject (about 1500 sequences in total)
  - Size, color, facial shapes (wider-narrow faces, short/long hair, glasses/no-glasses)
- Many sequences missing tags - didn’t fit prototype of what was requested
- CMU team produced baseline detection model using SVM

Original Dataset Examples - Peak

Request: Happy

Subject S111

Sequence 002_00000013

Request: Sadness

Subject S108

Sequence 005_00000010

Request: Angry

Subject S053

Sequence 003_00000024

Request: Fear

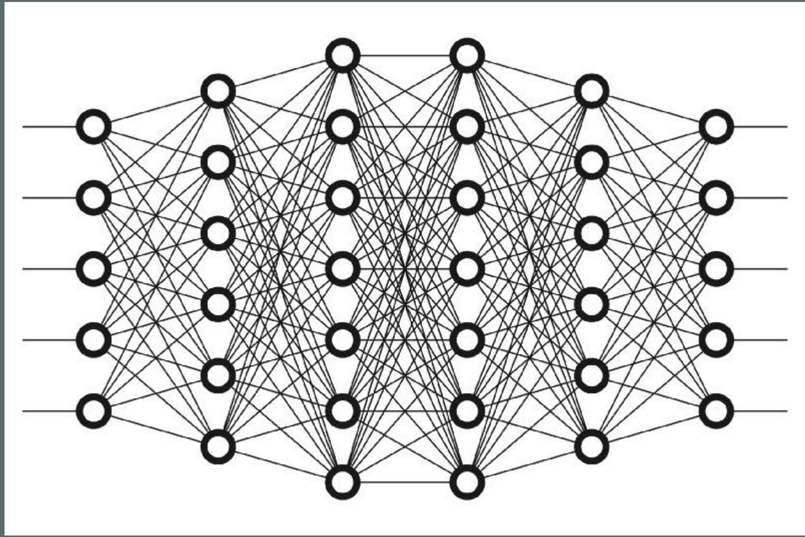
Subject S068

Sequence 004_00000010

Sequence Example

(Request: Sad - S501_006)

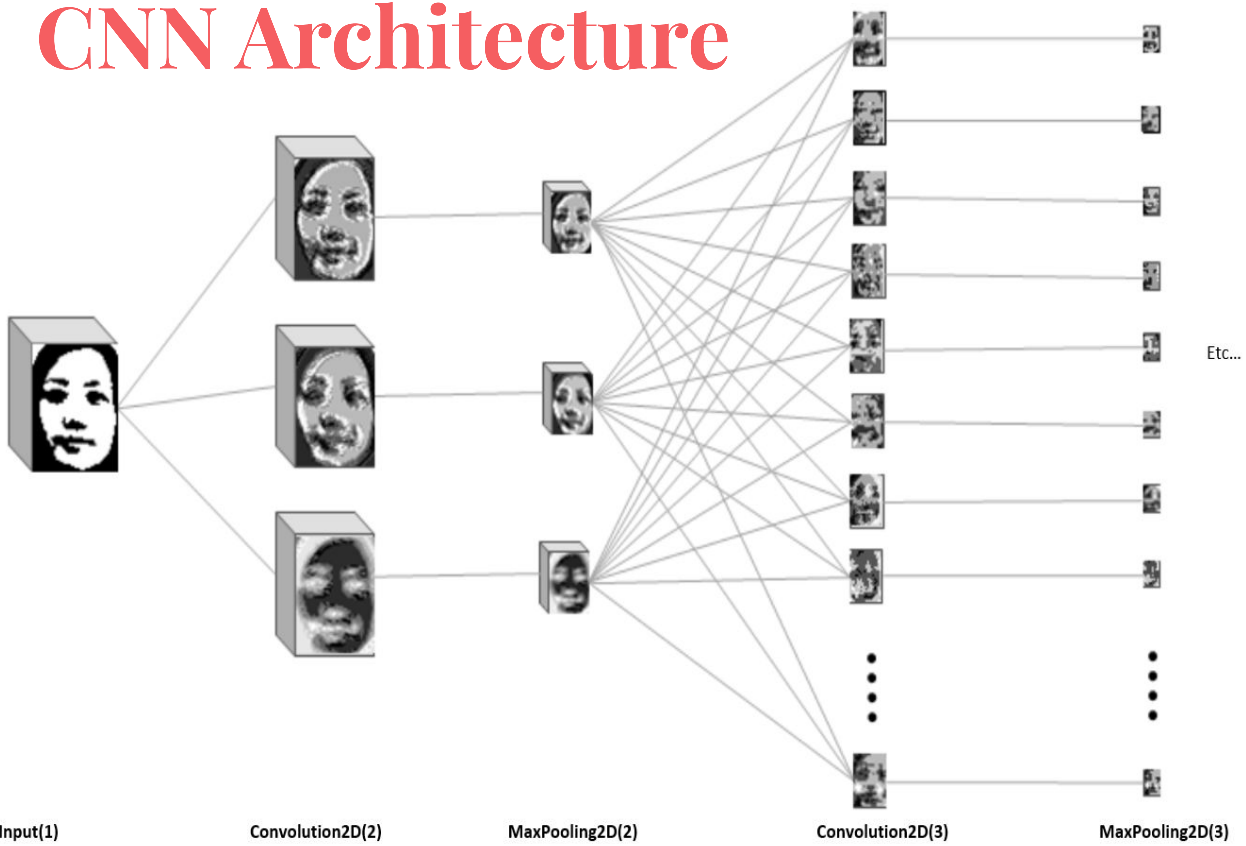
Deep Learning Network & Training Algorithm



Initial Planning

- Multi-classification problem with data that can have key features that define each class
- Best architecture to use: Convolutional Neural Networks (CNN)
- CNNs have been widely used in similar facial recognition studies
    - Requires the ability to pick up small shapes of facial behavior
- Primarily used 3-5 layers with similar parameters and activation functions to have a working concept and identify common issues; then iterate
- Number of Inputs for Initial Modeling was: **2,400**

CNN Architecture



Hidden Layers will increase with neurons to pick up smaller and important features for classification like so...

Setup

Accurately predicting emotion (i.e. facial expression) from a peak image

Data: Lessons Learned

- EDA
    - Original breakdowns depicted
- Manual tagging - sequences existed with no labels
- Data preparation
    - Took peak image from each sequence
    - Took some percentage of images near peak image
    - Augmentation - Left/right flips, brightness adjustments, rotations
- Facial detection
    - MTCNN: Multi-Task Cascaded Convolutional Neural Network
    - Python package: mtcnn from Ivan Da Paz built on Keras and Open CV
    - Total Inputs Increased to: **11,664**
- Looked just at "stills", not the sequence
    - Would be interesting to use RNN on the sequence
- Prepared data for model with Keras - Image Generator

Emotion	**N**
Angry (An)	45
Contempt (Co)	18
Disgust (Di)	59
Fear (Fe)	25
Happy (Ha)	69
Sadness (Sa)	28
Surprise (Su)	83

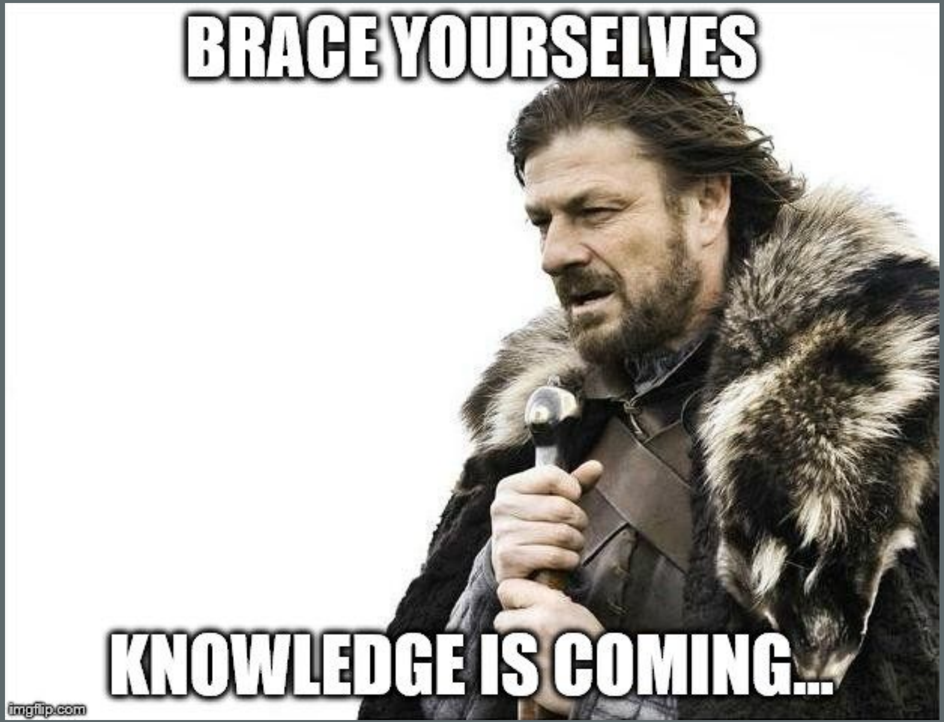
Model Implementation: Lessons Learned

- Modeling frameworks: Keras - CNN
  - Keras Image Generator has pseudo-augmentation: randomly performs transformations each iteration
- Original Model Consisted of only using ‘relu’ as activation for all layers, 2 hidden layers with 32 neurons, and two layers with 64 neurons
- Implemented Hyper-parameter tuning with Talos
  - Executed over 10,000 different convolutional neural networks that tuned parameters for each of the 5 layers throughout the entire project
- Best model now uses updated learning rate, a mix between ‘elu’ and ‘tanh’ for activation functions, and increased neurons for the hidden layers
- Epochs and Batch-sizes were also tuned and adjusted per length of time per epoch (e.g. 1 model could take 2 mins to 20mins based on these constraints, along with Early Stopping set to ‘strict’ or ‘moderate’)

Tuning Output Example

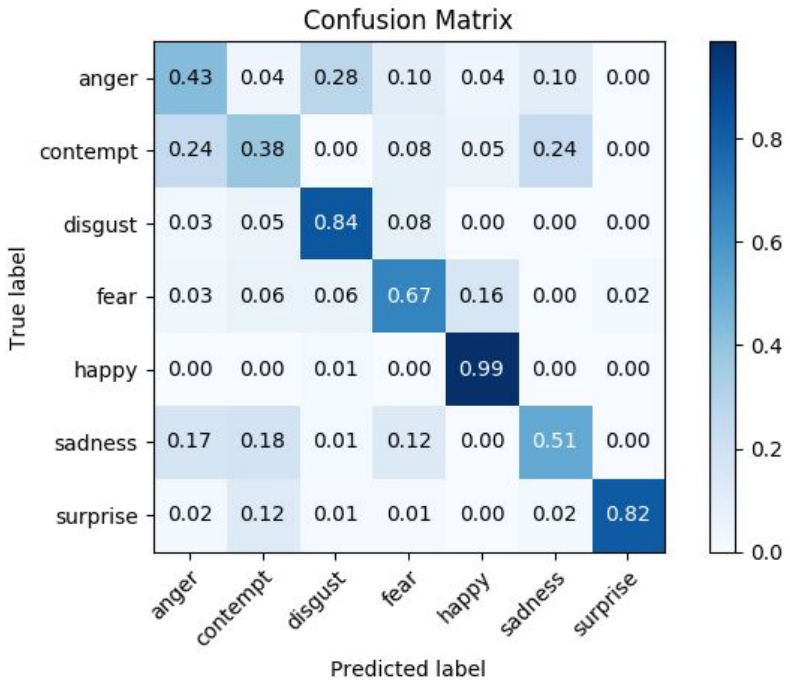
model_id	round_epochs	val_loss	val_accuracy	loss	accuracy	activation_1	activation_2	activation_3	activation_4	activation_5	batch_size	dropout	epochs	kernel_initializer	last_activation	loss2	lr	neurons_layer_2	neurons_layer_3	neurons_layer_4	optimizer
1022	9	4.2684665	0.056224901	4.151104	0.1629927	elu	elu	tanh	tanh	tanh	512	0.45	20	normal	softmax	categorica	9.00001	128	128	64	<class 'keras.optimizers.RMSprop'>
876	4	1.9615306	0.056224901	2.240914	0.1730127	elu	elu	tanh	relu	elu	512	0	20	normal	softmax	categorica	3.00007	32	32	32	<class 'keras.optimizers.Nadam'>
894	7	2.829493	0.056224901	3.277382	0.1629927	tanh	tanh	relu	elu	tanh	512	0.3	30	uniform	softmax	categorica	7.00003	32	128	32	<class 'keras.optimizers.RMSprop'>
913	7	4.1368859	0.056224901	3.49843	0.1569806	tanh	relu	relu	elu	tanh	256	0.3	20	normal	softmax	categorica	8.00002	32	32	32	<class 'keras.optimizers.RMSprop'>
933	8	2.5595546	0.056224901	2.715638	0.1623247	relu	tanh	relu	tanh	tanh	512	0.45	30	uniform	softmax	categorica	5.00005	128	256	128	<class 'keras.optimizers.RMSprop'>
947	4	3.531941	0.056224901	3.376321	0.1496326	tanh	tanh	tanh	relu	tanh	256	0.35	20	normal	softmax	categorica	8.00002	128	64	128	<class 'keras.optimizers.RMSprop'>
969	6	2.0266168	0.056224901	1.97487	0.1997328	elu	relu	tanh	tanh	elu	512	0.15	30	normal	softmax	categorica	3.00007	32	128	64	<class 'keras.optimizers.Adam'>
972	9	2.0064528	0.056224901	2.11129	0.1710087	relu	tanh	tanh	tanh	relu	512	0.15	20	normal	softmax	categorica	4.00006	64	64	64	<class 'keras.optimizers.Nadam'>
1010	30	1.9482636	0.056224901	1.94318	0.1429526	relu	relu	relu	elu	relu	512	0	30	normal	softmax	categorica	0.0001	64	64	64	<class 'keras.optimizers.Nadam'>
853	7	2.3438029	0.056224901	2.505079	0.1643287	elu	relu	relu	tanh	elu	512	0.4	30	uniform	softmax	categorica	8.00002	128	64	32	<class 'keras.optimizers.RMSprop'>
117	14	37.831936	0.056224901	43.13698	0.1683367	elu	tanh	elu	elu	elu	512	0.1	20	normal	softmax	categorica	9.00001	128	256	64	<class 'keras.optimizers.Nadam'>
1044	7	2.0457106	0.056224901	2.021253	0.1503006	relu	elu	tanh	elu	tanh	256	0.3	20	normal	softmax	categorica	4.00006	128	32	256	<class 'keras.optimizers.Adam'>
1049	6	2.3689551	0.056224901	2.654439	0.1576486	tanh	relu	elu	relu	tanh	512	0.1	30	uniform	softmax	categorica	5.00005	64	32	128	<class 'keras.optimizers.Nadam'>
52	5	2.5281706	0.056224901	2.150454	0.1369406	relu	elu	relu	tanh	elu	512	0.35	20	uniform	softmax	categorica	9.00001	32	256	256	<class 'keras.optimizers.Adam'>
1114	7	4.1351982	0.056224901	4.350833	0.1422846	relu	relu	tanh	elu	elu	256	0.05	30	normal	softmax	categorica	8.00002	64	64	128	<class 'keras.optimizers.RMSprop'>
1124	5	4.4868112	0.056224901	3.562364	0.1563126	elu	tanh	relu	tanh	tanh	512	0	30	normal	softmax	categorica	9.00001	128	128	128	<class 'keras.optimizers.RMSprop'>
49	4	2.5861962	0.056224901	2.769263	0.1636607	relu	relu	tanh	relu	tanh	256	0	20	normal	softmax	categorica	6.00004	128	64	64	<class 'keras.optimizers.RMSprop'>
1153	8	2.8128023	0.056224901	2.881657	0.1556446	relu	elu	tanh	relu	tanh	512	0.1	30	normal	softmax	categorica	7.00003	32	64	128	<class 'keras.optimizers.RMSprop'>
1156	5	2.3247428	0.056224901	2.28216	0.1656647	elu	relu	elu	tanh	relu	512	0.25	30	normal	softmax	categorica	7.00003	32	256	32	<class 'keras.optimizers.Adam'>

Results



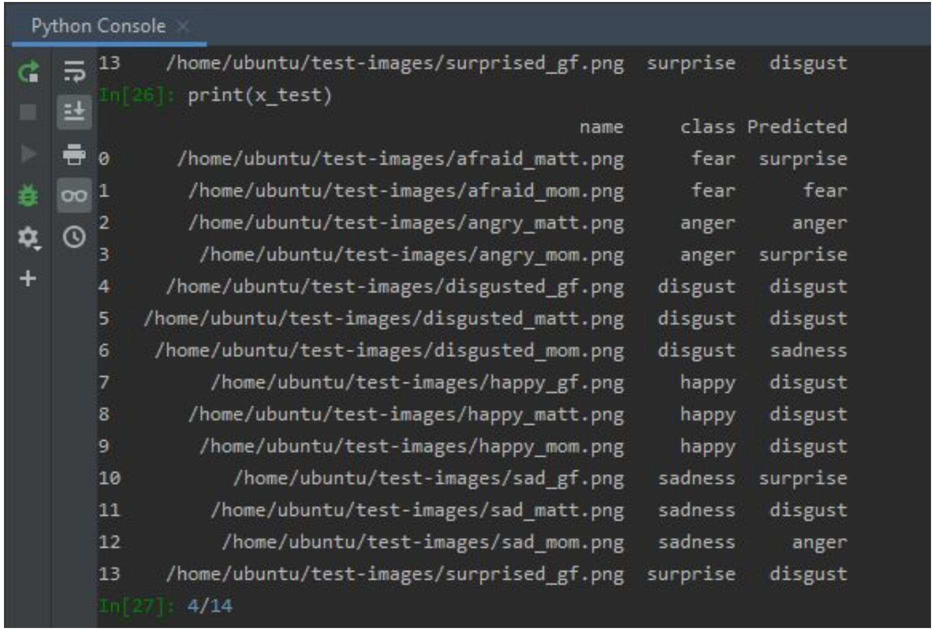
Understanding Results from Performance

- Performance was measured between Loss and Accuracy (both test and validation)
- 21% validation accuracy using Keras CNN example
- 48% validation accuracy without MTCNN face detection data
- **70% validation accuracy while using MTCNN face detection data**
- Struggled with contempt and anger
  - Possibly related to manual tagging misclassification?



Model Results Tested to Us

- The CMU data was quite complex to begin with, but how would the model fair against new data today?
- Asked friends and family for emotional peak images to test against the model
- After using MTCNN, results were somewhat expected...



```
Python Console
13    /home/ubuntu/test-images/surprised_gf.png  surprise   disgust
In[26]: print(x_test)
                                         name     class Predicted
0      /home/ubuntu/test-images/afraid_matt.png      fear  surprise
1       /home/ubuntu/test-images/afraid_mom.png      fear      fear
2       /home/ubuntu/test-images/angry_matt.png     anger     anger
3        /home/ubuntu/test-images/angry_mom.png     anger  surprise
4     /home/ubuntu/test-images/disgusted_gf.png   disgust   disgust
5   /home/ubuntu/test-images/disgusted_matt.png   disgust   disgust
6    /home/ubuntu/test-images/disgusted_mom.png   disgust   sadness
7         /home/ubuntu/test-images/happy_gf.png     happy   disgust
8       /home/ubuntu/test-images/happy_matt.png     happy   disgust
9        /home/ubuntu/test-images/happy_mom.png     happy   disgust
10          /home/ubuntu/test-images/sad_gf.png   sadness  surprise
11        /home/ubuntu/test-images/sad_matt.png   sadness   disgust
12         /home/ubuntu/test-images/sad_mom.png   sadness     anger
13    /home/ubuntu/test-images/surprised_gf.png  surprise   disgust
In[27]: 4/14
```

Identify These Emotions

Predicted: Disgusted

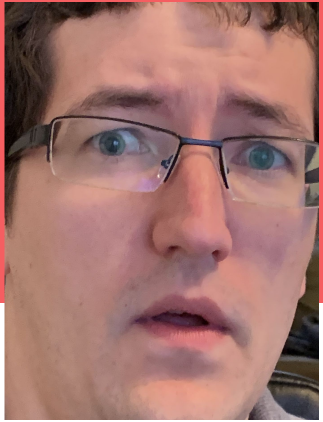
Disgusted Matt

Predicted: Disgusted

Happy Matt

Predicted: Angry

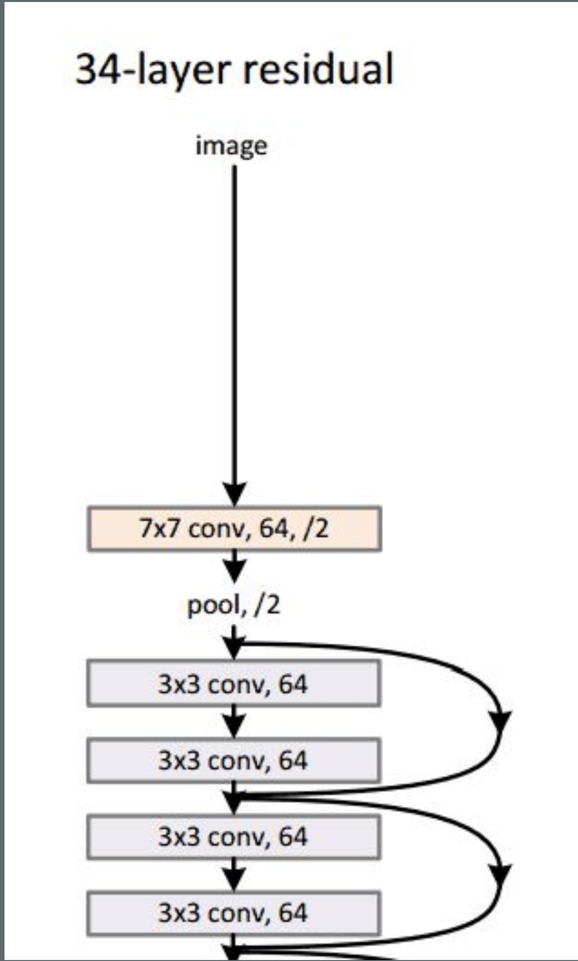
Angry Matt



Predicted: Surprise

Afraid Matt

Summary


34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64

This is a difficult task.

- How can a machine be expected to perfectly detect something humans can’t consistently detect?
    - Trained observers didn’t agree on a non-insignificant number (kappa of 0.82)
    - Our own manual tagging yielded mismatches
- How can a machine be expected to perfectly detect something humans couldn’t execute on?
    - Only 327 of the 593 sequences met CMU criteria for accurately representing requested emotion
    - Perhaps natural observations would be more consistent
- CMU concludes that 10k accurately tagged examples of each emotion are needed for “a fully automatic system to be robust” in practice

Questions?

https://github.com/mikelabadie/Final-Project-Group10