W207 Final Project:
Facial Keypoints Detection

Essa, Fan, Hoopengardner, Lovejoy

December 2019

Agenda

1. Problem Overview
2. Exploratory Analysis
3. Data Preparation and Improvement
4. Modeling Overview and Results
5. Lessons Learned
6. Appendices
    a. Model Detail
    b. References

Problem Overview

Facial Keypoints Detection - Overview

- Facial key points is fundamental to facial recognition.
- Objective of the task is to detect keypoint positions on facial images.
- Facial keypoints are x, y coordinates for 15 points including:
  - Center, inner, and outer corner of each eye
  - Inner and outer end of each eyebrow
  - Center of nose
  - Right corner, left corner, upper and lower center of mouth

- There are numerous applications including:
  - Identifying and tracking faces in images and video
  - Biometrics and facial recognition
  - And, of course, Instagram noses and Animoji!
- Competition has been running on Kaggle for > six years, using a two-month rolling leaderboard.

https://www.kaggle.com/c/facial-keypoints-detection

Exploratory Analysis

Initial Data Exploration

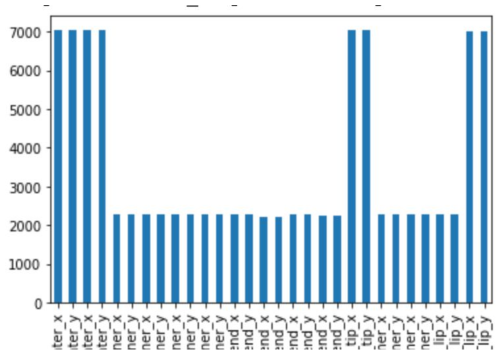
- Data file contains ~7,049 entries.
- Each entry contains 30 coordinates (x and y for 15 keypoints) and a 9,216 digit image that can be re-shaped into a 96x96 pixel image.
- Images display faces, though some are blurry, not centered, cartoons or drawings, etc.

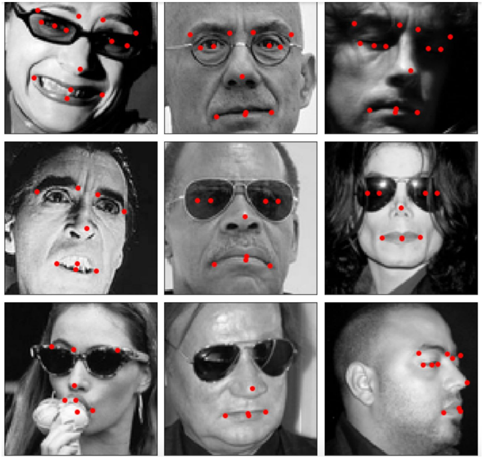
Selection of Images

Selection of Images Showing Keypoints

Missing Keypoints

- Viewing a histogram of the data clearly shows a significant number of the entries has incomplete keypoints.
- Visualizing the images with missing keypoints shows some subjects are wearing sunglasses, are not facing forward, or are obscured by shadows or other items.
- Setting aside incomplete entries leaves a dataset containing only 2,140 entries - a ~70% reduction.





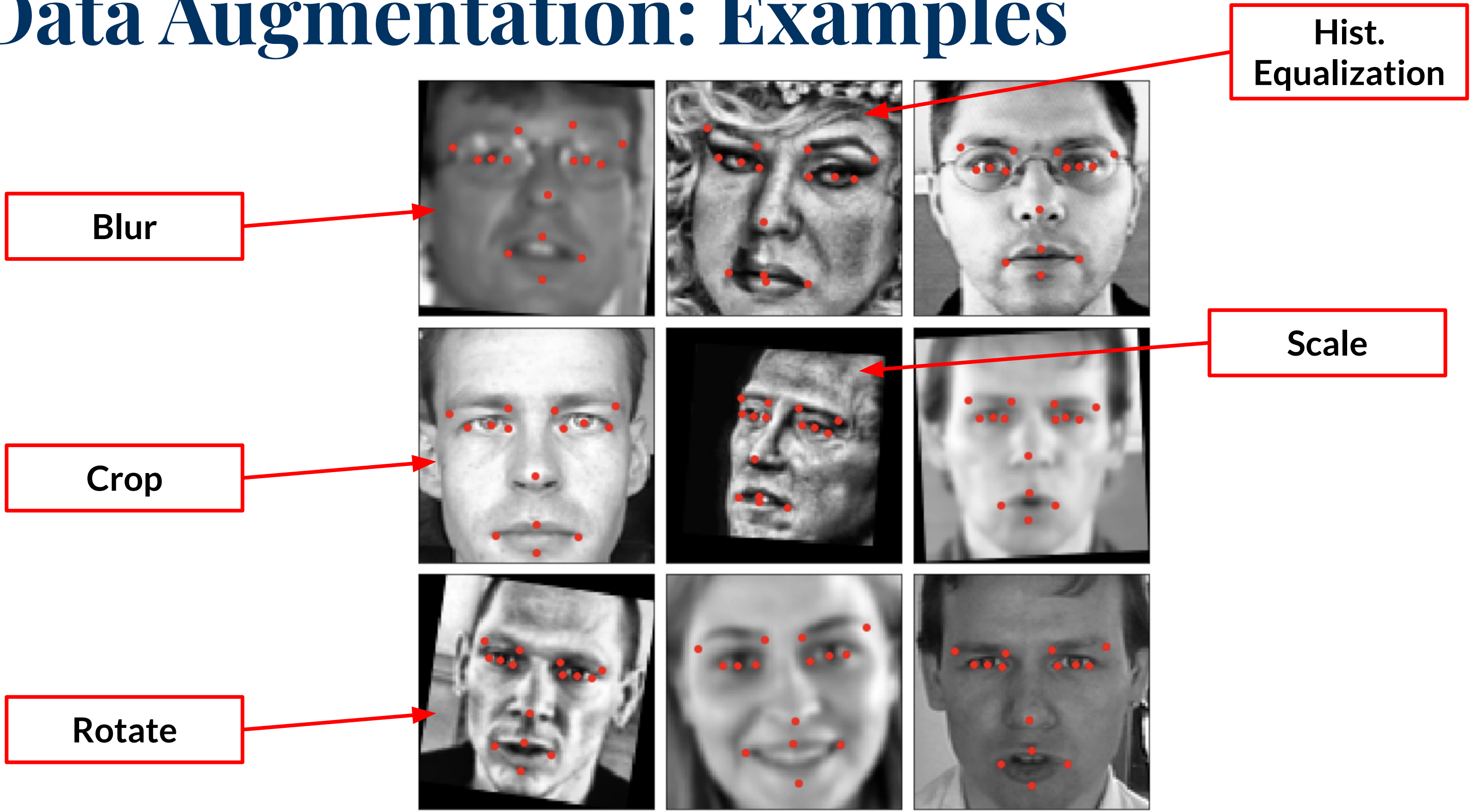
Data Preparation and Augmentation

Data Augmentation: Setup

- **Imgaug library**: Augmentations in random order, parameters define distribution

Augmentation	Prob.	Parameters
Rotate	1.0	(-10, 10)
Scale (zoom)	1.0	(0.8, 1.2)
CLAHE: Histogram equalization	0.8	clip_limit=(1, 10)
Gaussian Blur	1.0	sigma=(0.0, 2.0)
Gaussian Noise	0.5	scale=(0, 0.1*255)
Crop	0.5	percent=(0, 0.1)
Flip (left-right)	0.8	

Data Augmentation: Examples

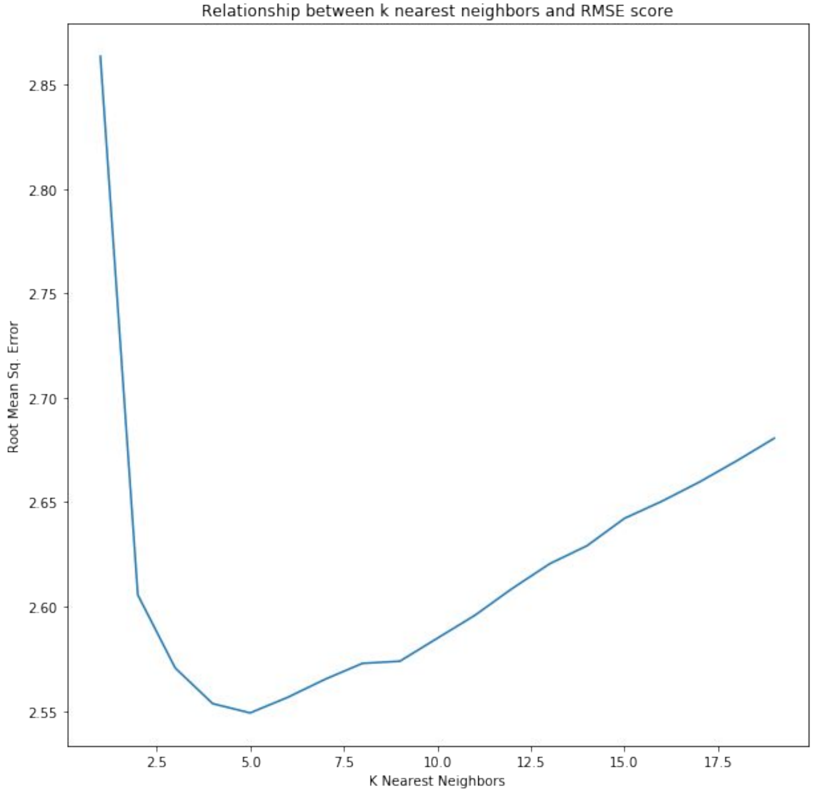


Data Augmentation: Questions & Learnings

- **Trial and Error**: Which augmentations should I use?
- **Problem Goal**: What is the goal of the problem? How can augmentations support this goal?
- **Multiples**: How much augmented data should we use? 2x original dataset?

Modeling Overview and Results

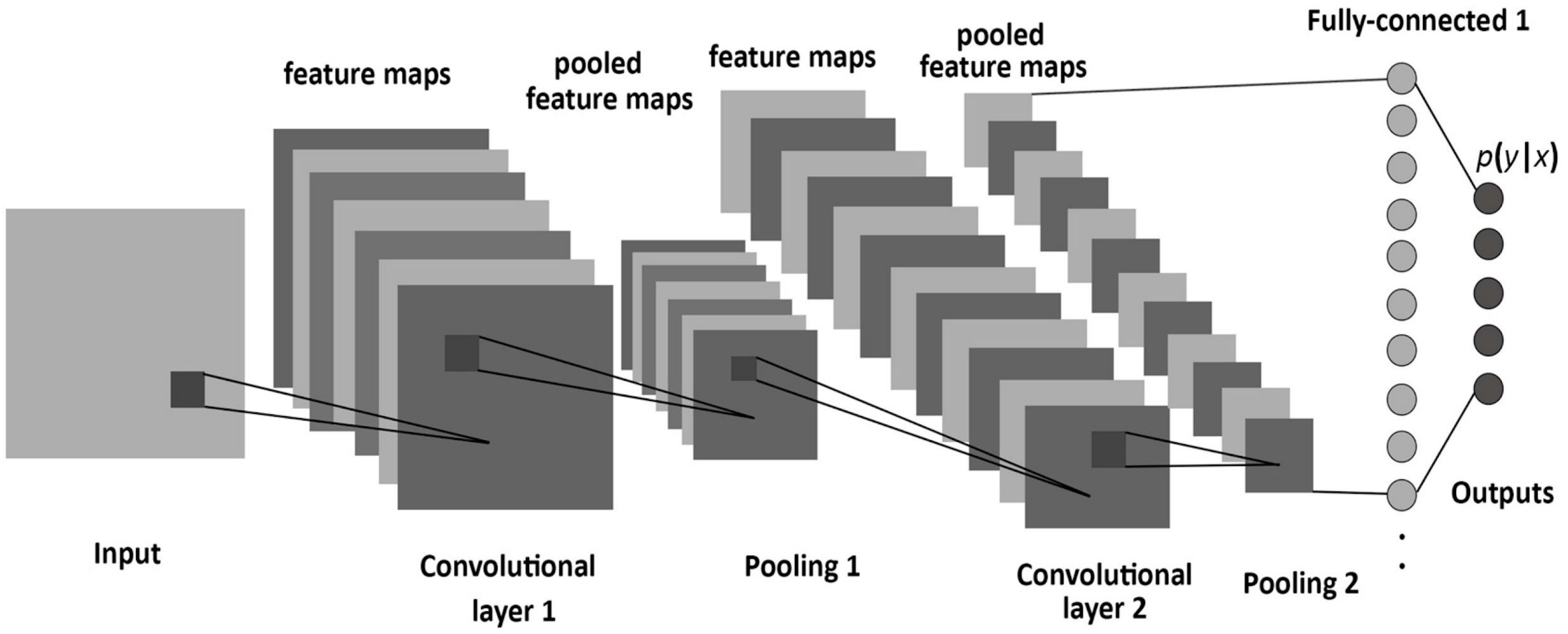
K Nearest Neighbors



- Used a simple KNN model as baseline to compare performance to more optimal Convolutional Neural Networks

- Note that KNN model did not show appreciable improvement after varying the number of nearest neighbors (k). In fact, after RMSE is lowest at a value of k=5, the performance degrades as k is increased

Convolutional Neural Networks



Source: https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90

Convolution Neural Networks Layers and Hyperparameters

- Optimizer
- Kernel
- Weight Initializer
- Activation Function
- Zero Padding
- Max Pooling
- Normalization

- Dropout
- Flatten
- Dense
- Batch size
- Epochs and Early Stopping

Predicted facial keypoints (Dev Data)

Model 3 - trained on augmented data

Actual

Predicted

How did we do?

Model	Description	Total Parameters	Augmented Data?	RMSE on validation holdout	Kaggle Public Score
Base	K-nearest neighbors model with K=5 (k value determined by search)		N	2.4898	4.03406
			Y	2.5213	3.98281
0	CNN with two Conv2D, MaxPooling, and Batch Normalization layers.	14,177,602	N	1.5338	3.3852
			Y	1.4857	3.3857
1	CNN with three Conv2D, MaxPooling, and three Batch Normalization layers.	7,153,762	N	1.9058	3.5105
			Y	1.4911	3.0930
2	CNN with four Conv2D, three MaxPooling, and five Batch Normalization layers.	543,490	N	3.1621	5.4136
			Y	2.3246	3.5182
3	CNN with six Conv2D, six MaxPooling, seven Batch Normalization, and six dropout layers.	150,914	N	1.4177	2.8694
			Y	**1.0914**	**2.7843**

Predicted facial keypoints (Dev Data)

The less optimal model (e.g. Model 0) had difficulty in predicting facial keypoints on faces tilted to one side rather than looking straight ahead. Our best performing model (Model 3 with augmented data) does slightly better in these cases

Model 0 - Augmented Data

Model 3 - Augmented Data

Given More Time...

...we would have liked to explore:

- Building ensembles of models with the same architecture but trained on different parts of the data.
- Building ensembles of models with different architectures.
- Making use of the incompletely labeled images by ensembling models for both (fully labeled and partially labeled) together.
- Improving hyperparameter tuning through both automated search (grid or random search) and applying machine learning (e.g., Bayesian optimization library like Hyperopt).

Lessons Learned

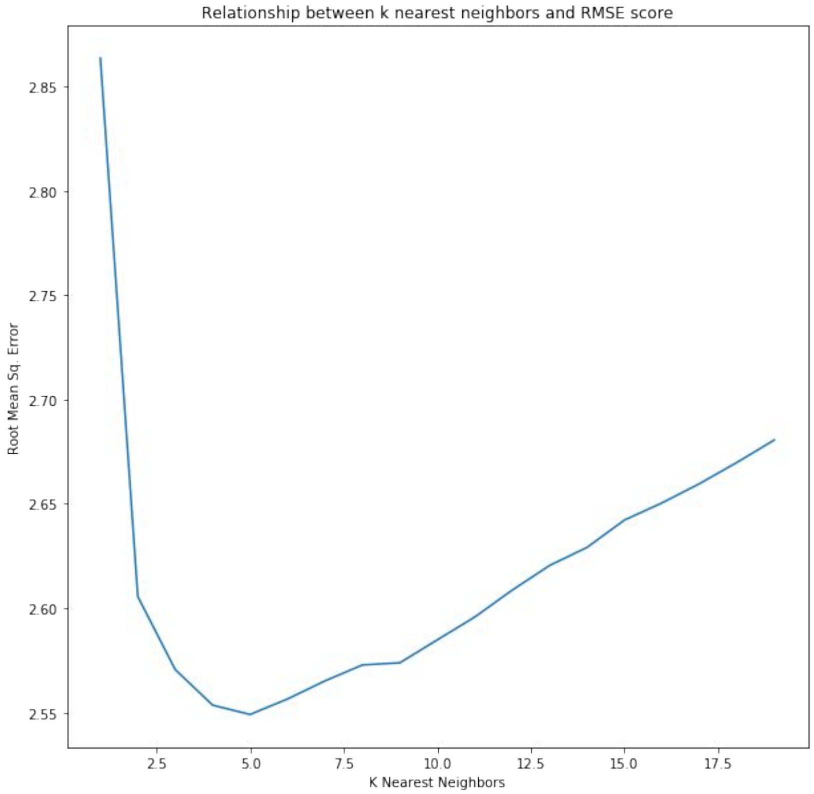
Lessons Learned

- Despite “the unreasonable effectiveness of data”, sub-optimal augmentations can significantly degrade performance.
- Build a complete pipeline early in the process.
- The Google Colab environment worked well (free GPUs are really helpful!)*.
- The “grad student” method of hyperparameter tuning through trial and error is costly in terms of labor hours.
- Careful consideration is required to manage workflow, division of labor, integration of code into master notebook, and finalizing model tuning.

Questions?

Appendix: Model Detail

K Nearest Neighbor

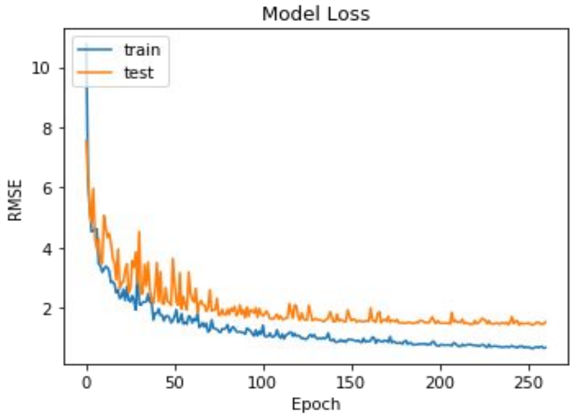


We noted that KNN model did not show appreciable improvement after varying the number of nearest neighbors (k). In fact, after RMSE is lowest at a value of k=5, the performance degrades as k is increased

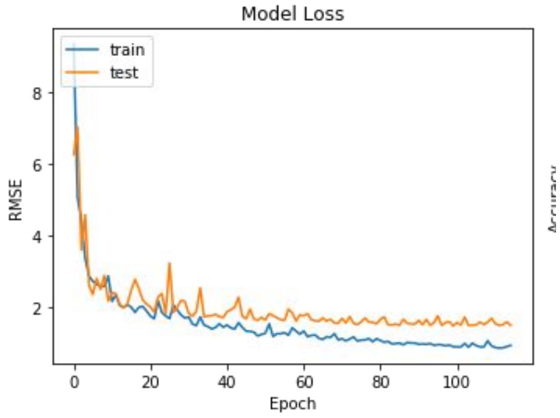
Model o

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization_1 (Batch (None, 96, 96, 1)         4
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 96, 96, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 96, 96, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 48, 48, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 48, 48, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 147456)            0
_________________________________________________________________
dense_1 (Dense)              (None, 96)                14155872
_________________________________________________________________
dense_2 (Dense)              (None, 30)                2910
=================================================================
Total params: 14,177,602
Trainable params: 14,177,600
Non-trainable params: 2
_________________________________________________________________
```

Training History - Base Data



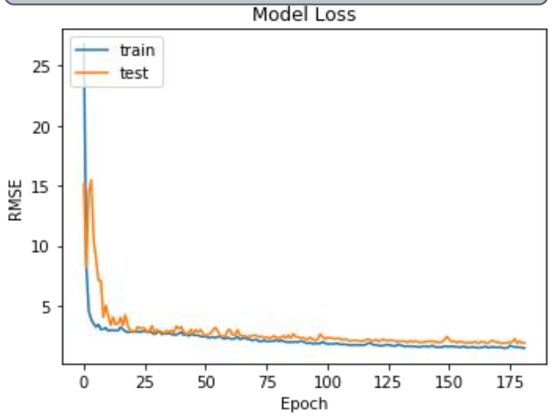
Training History - Augmented Data



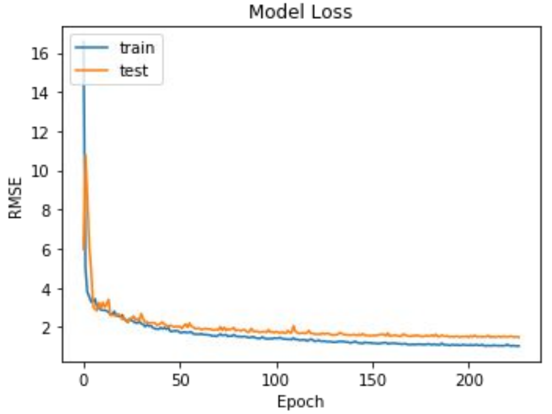
Model 1

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization_6 (Batch (None, 96, 96, 1)         4
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 96, 96, 32)        320
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 96, 96, 64)        18496
_________________________________________________________________
batch_normalization_7 (Batch (None, 96, 96, 64)        256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 48, 48, 64)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 48, 48, 96)        55392
_________________________________________________________________
batch_normalization_8 (Batch (None, 48, 48, 96)        384
_________________________________________________________________
flatten_3 (Flatten)          (None, 221184)            0
_________________________________________________________________
dense_5 (Dense)              (None, 32)                7077920
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_6 (Dense)              (None, 30)                990
=================================================================
Total params: 7,153,762
Trainable params: 7,153,440
Non-trainable params: 322
```

Training History - Base Data



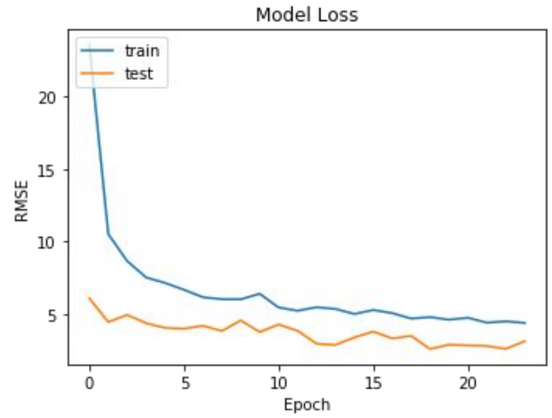
Training History - Augmented Data



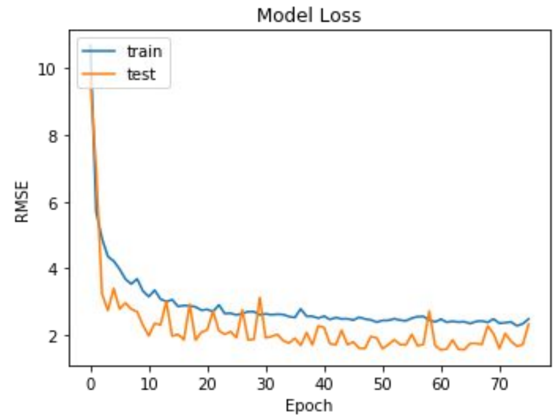
Model 2

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
batch_normalization_12 (Batc (None, 96, 96, 1)         4         
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 96, 96, 16)        160       
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 48, 48, 16)        0         
_________________________________________________________________
batch_normalization_13 (Batc (None, 48, 48, 16)        64        
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 48, 48, 32)        4640      
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 24, 24, 32)        0         
_________________________________________________________________
batch_normalization_14 (Batc (None, 24, 24, 32)        128       
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 24, 24, 64)        18496     
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 12, 12, 64)        0         
_________________________________________________________________
batch_normalization_15 (Batc (None, 12, 12, 64)        256       
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 12, 12, 128)       73856     
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 6, 6, 128)         0         
_________________________________________________________________
batch_normalization_16 (Batc (None, 6, 6, 128)         512       
_________________________________________________________________
flatten_5 (Flatten)          (None, 4608)              0         
_________________________________________________________________
dense_9 (Dense)              (None, 96)                442464    
_________________________________________________________________
dropout_5 (Dropout)          (None, 96)                0         
_________________________________________________________________
dense_10 (Dense)             (None, 30)                2910      
=================================================================
Total params: 543,490
Trainable params: 543,008
Non-trainable params: 482
```

Training History - Base Data



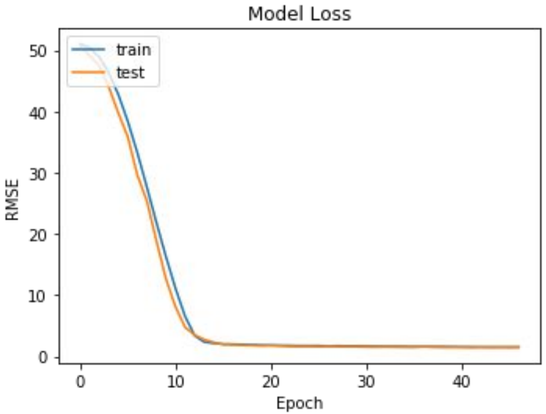
Training History - Augmented Data



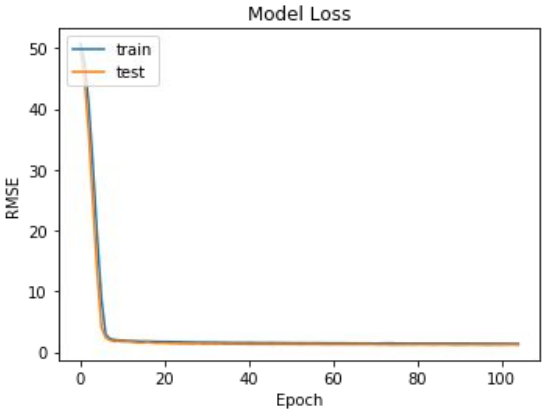
Model 3

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization_22 (Batc (None, 96, 96, 1)         4
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 96, 96, 32)        832
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 48, 48, 32)        0
_________________________________________________________________
batch_normalization_23 (Batc (None, 48, 48, 32)        128
_________________________________________________________________
dropout_7 (Dropout)          (None, 48, 48, 32)        0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 48, 48, 32)        25632
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 24, 24, 32)        0
_________________________________________________________________
batch_normalization_24 (Batc (None, 24, 24, 32)        128
_________________________________________________________________
dropout_8 (Dropout)          (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 24, 24, 48)        38448
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 12, 12, 48)        0
_________________________________________________________________
batch_normalization_25 (Batc (None, 12, 12, 48)        192
_________________________________________________________________
dropout_9 (Dropout)          (None, 12, 12, 48)        0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 12, 12, 64)        27712
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
batch_normalization_26 (Batc (None, 6, 6, 64)          256
_________________________________________________________________
dropout_10 (Dropout)         (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 6, 6, 64)          36928
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 3, 3, 64)          0
_________________________________________________________________
batch_normalization_27 (Batc (None, 3, 3, 64)          256
_________________________________________________________________
flatten_7 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_13 (Dense)             (None, 128)               73856
_________________________________________________________________
dense_14 (Dense)             (None, 96)                12384
_________________________________________________________________
batch_normalization_28 (Batc (None, 96)                384
_________________________________________________________________
dense_15 (Dense)             (None, 30)                2910
=================================================================
Total params: 220,050
Trainable params: 219,376
Non-trainable params: 674
```

Training History - Base Data



Training History - Augmented Data

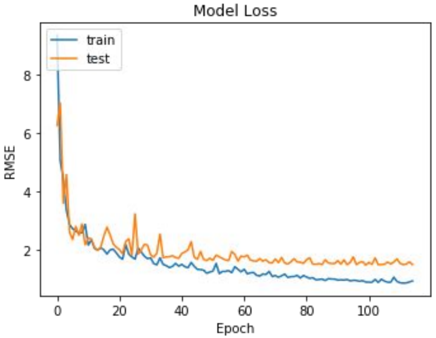


How did we do?

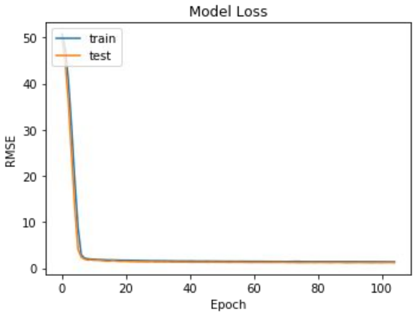
K Nearest Neighbor (k=5)
Base Data

No iterative training process

Training History - Model 0
Augmented Data



Training History - Model 3
Augmented Data



Appendix: References

References

Books
http://www.deeplearningbook.org/contents/convnets.html
https://www.academia.edu/37010160/Hands-On_Machine_Learning_with_Scikit-Learn_and_TensorFlow**General**

CNN Guides
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/
https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90
https://blog.xrds.acm.org/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/
https://heartbeat.fritz.ai/a-beginners-guide-to-convolutional-neural-networks-cnn-cf26c5ee17ed

Different types of Layers
https://forums.fast.ai/t/dense-vs-convolutional-vs-fully-connected-layers/191

Batch Normalization
https://towardsdatascience.com/batch-normalization-theory-and-how-to-use-it-with-tensorflow-1892ca0173ad
https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/
https://arxiv.org/abs/1502.03167
https://machinelearningmastery.com/how-to-accelerate-learning-of-deep-neural-networks-with-batch-normalization/

Pooling
https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

Initialize weights and bias
https://stackoverflow.com/questions/40708169/how-to-initialize-biases-in-a-keras-model
https://medium.com/usf-msds/deep-learning-best-practices-1-weight-initialization-14e5c0295b94
https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78
https://stats.stackexchange.com/questions/27112/danger-of-setting-all-initial-weights-to-zero-in-backpropagation

Padding
https://www.quora.com/What-does-the-same-padding-parameter-in-convolution-mean-in-TensorFlow

Optimizers
CNN optimizers in Keras
https://www.dlology.com/blog/quick-notes-on-how-to-choose-optimizer-in-keras/

Adam Optimizer
https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f
https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

Dropout
https://www.quora.com/In-Keras-what-is-a-dense-and-a-dropout-layer

Hyperparameter Tuning
https://towardsdatascience.com/automated-machine-learning-hyperparameter-tuning-in-python-dfda59b72f8a

Stanford CS231n: Convolutional Neural Networks for Visual Recognition course material
http://cs231n.github.io/
http://cs231n.github.io/convolutional-networks/#overview

Model Architectures:
https://machinelearningmastery.com/stacking-ensemble-for-deep-learning-neural-networks/
https://towardsdatascience.com/ensembling-convnets-using-keras-237d429157eb
https://benanne.github.io/2015/03/17/plankton.html
https://arxiv.org/pdf/1409.1556.pdf