MIDS 207 Final Project - Team 4

Anup Jha
Anish Philip

Our motivation for this project was to understand the concepts of Convolutional Neural Network and its potential applications in Image processing

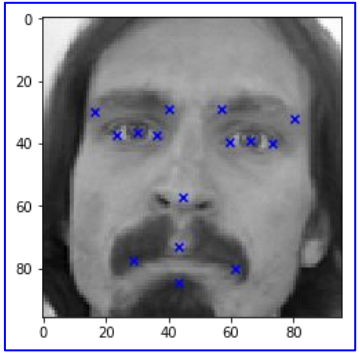
- “Hello World” problem to applications of deep learning in image recognition, and classification

- Problem statement: Given a set of test and training data, get near exact coordinates for 15 key facial features like eyebrow, eye corners, and center, lip center-top and bottom, nose tip...



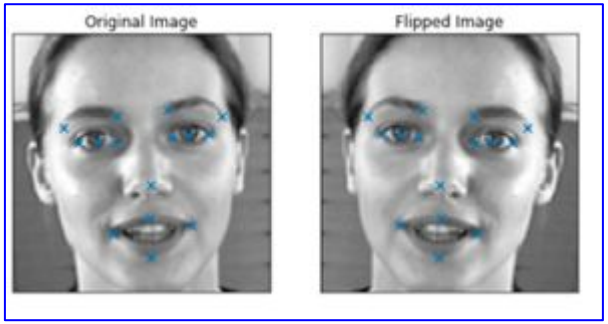
There are 7,049 (96x96 Greyscale) training images with keypoint coordinates and 1,783 test images with No keypoint coordinates

- Only 2,140 images in training data have all 15 features
  - Eyes (Center, Inner and Outer Corners) - 12 features
  - Eyebrows (Corners) - 8 features
  - Nose (Nose tip) - 2 features
  - Lips (Corners, top, bottom) - 8 features



- Expand the size of our data set:

  We flipped the training images on the ‘center-vertical’ axis to double the size of images for which we have keypoint coordinates



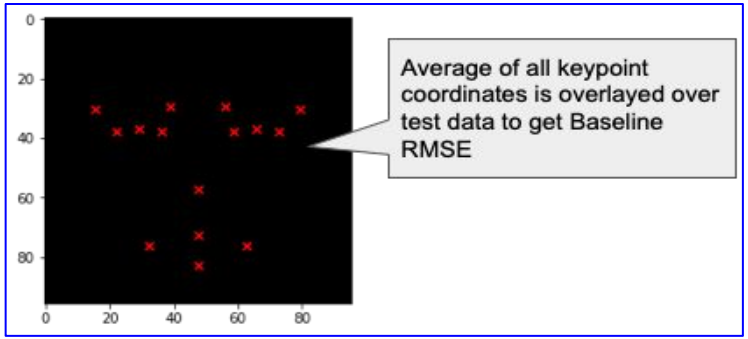
We were able to achieve the objective of the project by configuring a CNN which showed good predictive capability with an RMSE of 1.91, surpassing the baseline RMSE of 4.47 that was set at the onset of the project

- For Baseline performance we took the mean value of location of each keypoint from the training set as the prediction and compared it with the dev data set to arrive at baseline RMSE for our project

Features	RMSE
Left eye Center	2.52
Left eye inner Corner	2.30
Left eye outer Corner	3.19
Left eyebrow inner end	3.28
Left eyebrow outer end	4.06

Features	RMSE
Right eye Center	2.58
Right eye inner Corner	2.39
Right eye outer Corner	3.19
Right eyebrow inner corner	3.39
Right eyebrow outer corner	4.20

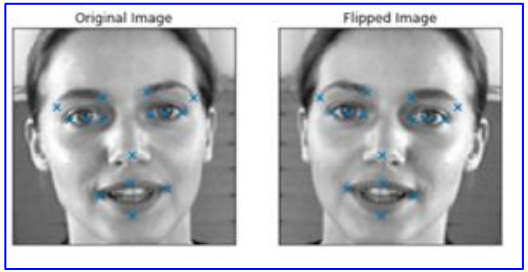
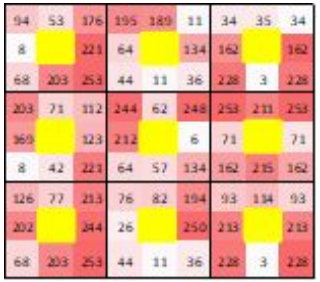
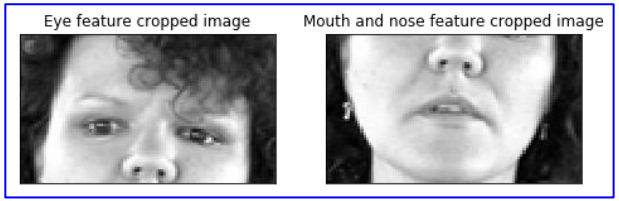
Features	RMSE
Nose tip	4.50
Mouth left corner	4.73
Mouth right corner	4.78
Mouth center top lip	4.70
Mouth center bottom lip	4.74



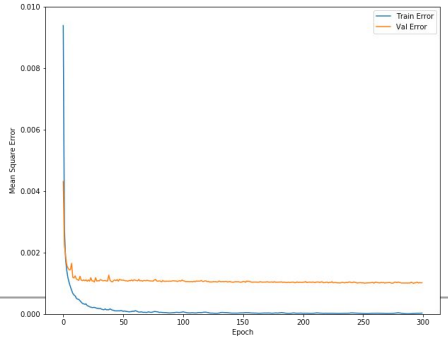
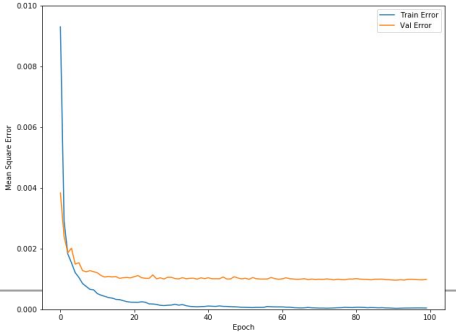
12 Convolutional Neural Net models were configured with varying hyperparameters and image augmentation to achieve the least possible Mean Squared Error

- Following hyperparameters were fine tuned to Optimize RMSE and to avoid overfitting:
  - Number of Convolutional, pooling and fully connected layers
  - Mini Batch Normalization to help contain the disruption in previous layers making learning on the later layers easier (Not optimized for batch size)
  - Drop out to keep overfitting in check
  - Activation function (ReLU, LeakyReLU)
  - Type of pooling and stride (Max, Average)
  - Number of Epochs

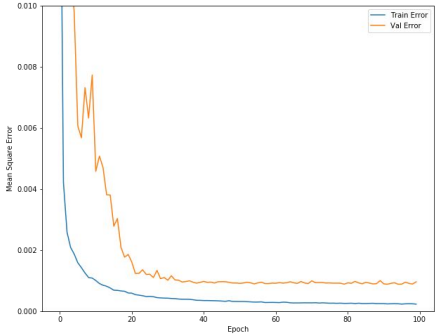
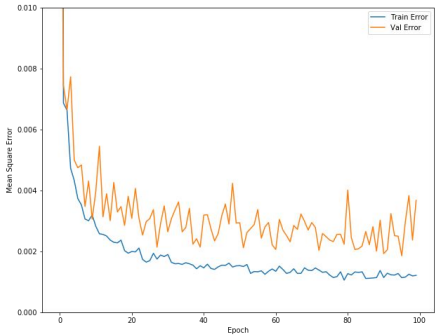
We used different types of Image augmentation to improve sample size, performance and accuracy

Sample size	Processing time	Accuracy
 Flipped Image on Center-Vertical axis	 Take an average of all surrounding pixels to replace the center pixel value. This would shrink NxN image to (N/3)x(N/3)	 Cropped image on center-horizontal axis allowing some overlap in two resulting images

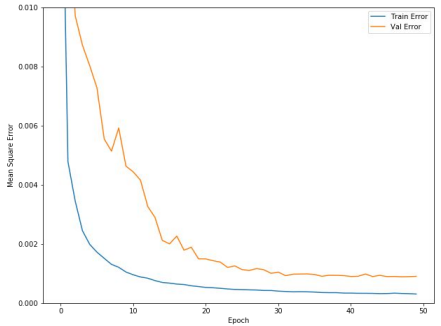
CNN Model 1 and 2 - By using Leaky ReLU, we are observing reduction in the gap of training and validation error and a decrease in validation error

Model	Feature Learning Layers	Classification/ Dense Layers	Training and Validation Error
1	• 32 filter Convolutional Layer (3x3 Kernel) ReLU activation and Max Pooling • 64 filter Convolutional Layer (3x3 Kernel) ReLU activation and Max Pooling • 128 filter Convolutional Layer (3x3 Kernel) ReLU activation and Max Pooling	• Flatten • Dense Layer 1 (500, ReLU) • Dense Layer 2 (500, ReLU) • Output Layer (30, Linear)	
2	• 32 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation and Max Pooling • 64 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation and Max Pooling • 128 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation and Max Pooling	• Flatten • Dense Layer 1 (500, ReLU) • Dense Layer 2 (500, ReLU) • Output Layer (30, Linear)	

CNN Model 3 and 4 - In Model 3, Dropout layers reduced the gap between training & validation error, but Mini Batch normalization layer without dropout in Model 4 increased the overall error

Model	Feature Learning/ Convolutional Layers	Classification/ Dense Layers	Training and Validation Error
3	• 32 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling & Drop out (0.10) • 64 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling & Drop out (0.15) • 128 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling & Drop out (0.20)	• Flatten • Dense Layer 1 (500, ReLU) • Drop out (0.25) • Dense Layer 2 (500, ReLU) • Output Layer (30, Linear)	
4	• 32 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling & MiniBatch Norm • 64 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling & MiniBatch Norm • 128 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling & MiniBatch Norm	• Flatten • Dense Layer 1 (500, ReLU) • MiniBatch Norm • Dense Layer 2 (500, ReLU) • MiniBatch Norm • Output Layer (30, Linear)	

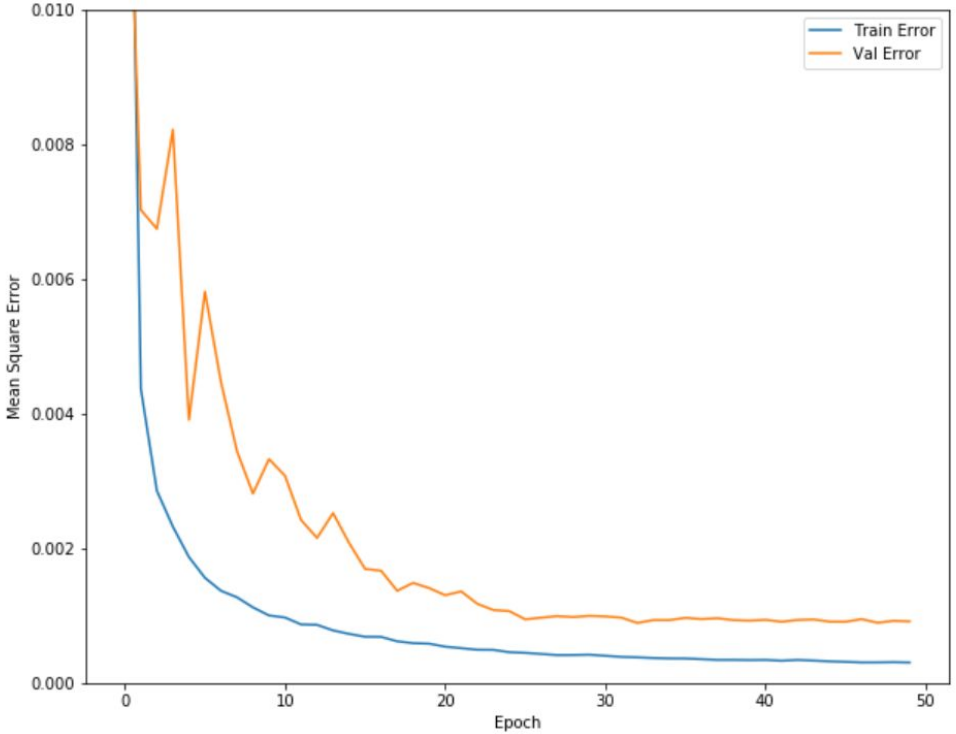
CNN Model 5 and 6 - In model 6, by including drop outs after the 2nd convolutional layer we achieved low error with less gap between training and validation error

Model	Feature Learning/ Convolutional Layers	Classification/ Dense Layers	Training and Validation Error
5	• 96 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling, MiniBatch Norm & Drop out (0.10) • 128 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling, MiniBatch Norm & Drop out (0.15) • 128 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling, MiniBatch Norm & Drop out (0.20)	• Flatten • Dense Layer 1 (1000, ReLU) • MiniBatch Norm • Drop out (0.25) • Dense Layer 2 (1000, ReLU) • MiniBatch Norm • Drop out (0.30) • Output Layer (30, Linear)	*Not optimal for execution on CPU memory*
6	• 32 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling • 64 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling, MiniBatch Norm & Drop out (0.10) • 64 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling, MiniBatch Norm & Drop out (0.15) • 128 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling, MiniBatch Norm & Drop out (0.20)	• Flatten • Dense Layer 1 (500, ReLU) • Drop out (0.25) • Dense Layer 2 (500, ReLU) • Output Layer (30, Linear)	

Model7

Bigger Kernel (5X5) for the first layer CNN

Input
CNN1 -- 32 maps kernel size 5X5 activation leaky-relu
CNN2 -- 64 maps kernel size 3X3 activation leaky-relu
Dropout -- rate 0.15
Maxpool2 -- 2X2 stride 2
CNN3 -- 64 maps kernel size kernel size 3X3 activation leaky-relu
Dropout -- rate 0.15
Maxpool2 -- 2X2 stride 2
CNN4 -- 128 Maps Kernel size 2X2 activation leaky-relu
Dropout -- rate 0.20
Maxpool3 -- 2X2 stride 2
Dense Layer 1 500 neurons activation leaky-relu
Dropout rate 0.25
Dense Layer 2 500 neurons activation leaky-relu
Output layer 30 neurons -- No activation as we have regression problem

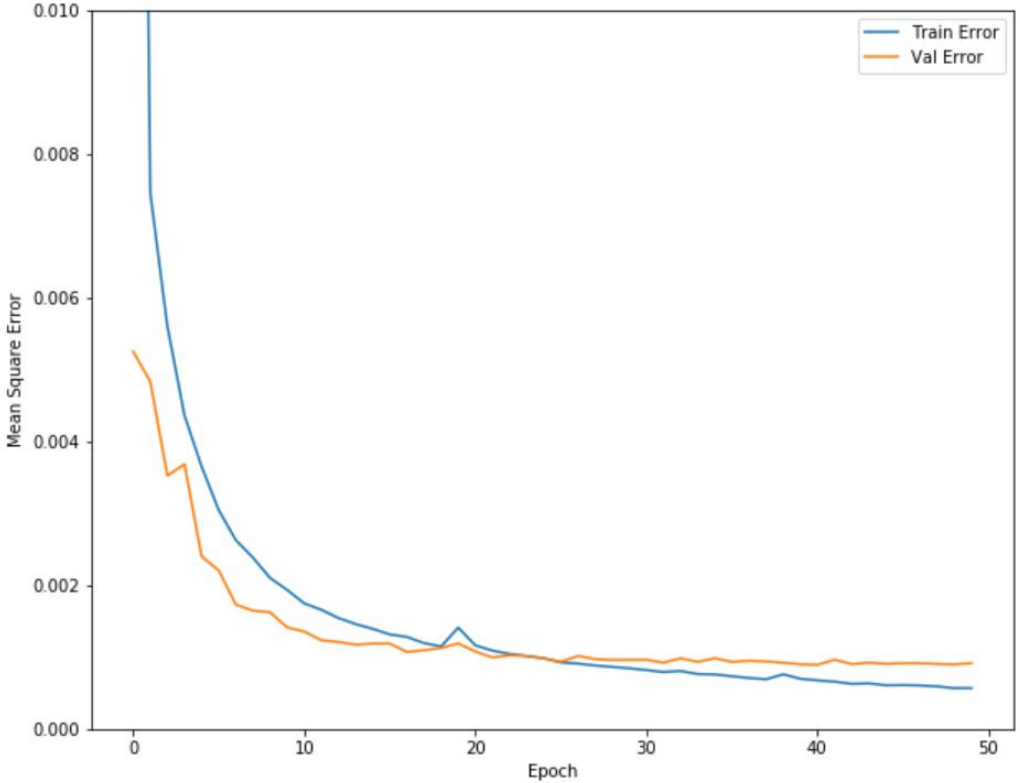


We see that Model7 gives better performance and the train and validation error are quite similar so not much overfitting.

Model8

No dropouts in Convolutional layers and using average pooling instead of max pool

Input
CNN1 -- 32 maps kernel size 5X5 activation leaky-relu
CNN2 -- 64 maps kernel size 3X3 activation leaky-relu
AvgPool -- 2X2 stride 2
CNN3 -- 64 maps kernel size kernel size 3X3 activation leaky-relu
AvgPool -- 2X2 stride 2
CNN4 -- 128 Maps Kernel size 2X2 activation leaky-relu
AvgPool -- 2X2 stride 2
Dense Layer 1 500 neurons activation leaky-relu
Dropout rate 0.25
Dense Layer 2 500 neurons activation leaky-relu
Dropout rate 0.30
Output layer 30 neurons -- No activation as we have regression problem



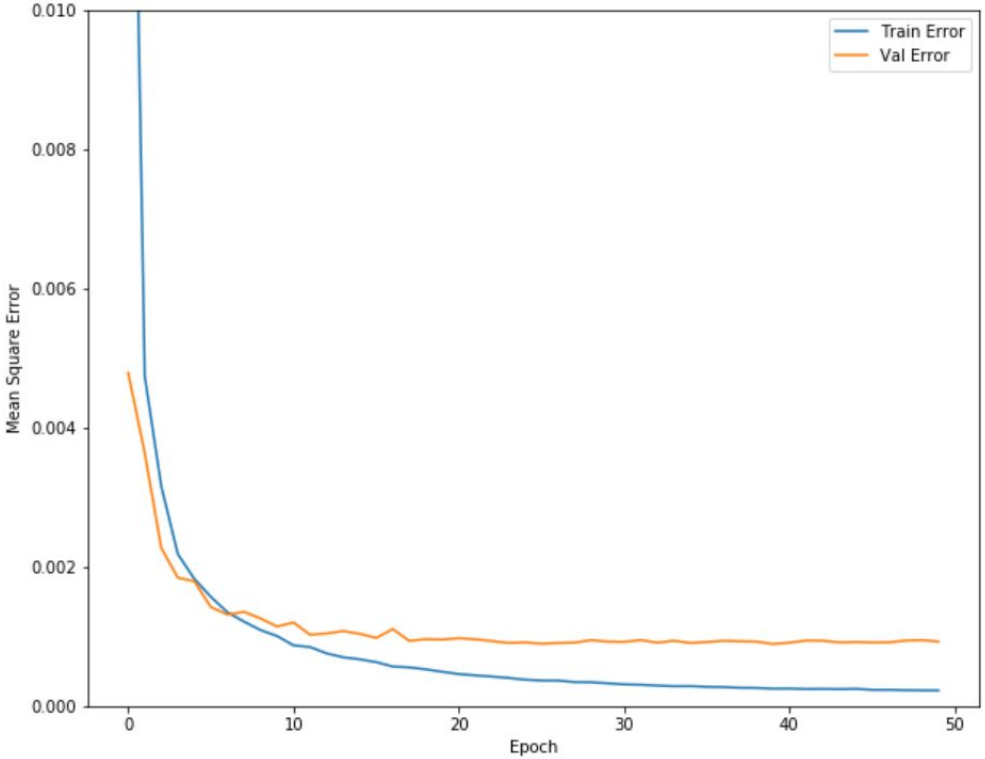
We see that model8 has actually validation loss less than training loss for quite some epochs.Also the validation loss is very close to training loss. This tells us that the model is not quite stable and we actually ran another test with higher number of training cases and the model diverges at certain points. We will try max pool without droputs in the convolution layers this time.

Model9

No dropouts in all Convolutional layers and using max pool

Input
CNN1 -- 32 maps kernel size 5X5 activation leaky-relu
CNN2 -- 64 maps kernel size 3X3 activation leaky-relu
MaxPool -- 2X2 stride 2
CNN3 -- 64 maps kernel size kernel size 3X3 activation leaky-relu
MaxPool -- 2X2 stride 2
CNN4 -- 128 Maps Kernel size 2X2 activation leaky-relu
MaxPool -- 2X2 stride 2
Dense Layer 1 500 neurons activation leaky-relu
Dropout rate 0.25
Dense Layer 2 500 neurons activation leaky-relu
Output layer 30 neurons -- No activation as we have regression problem

We see that Model9 the validation loss and train loss are not very far off which means not overfitting and with 50 epochs pretty much we have stabilized the loss



Model10: Image Cropping

Two Sets of NN

- **Eye Features**
- **Mouth and Nose Features**

Eye feature cropped image

Mouth and nose feature cropped image

Since input is smaller we add one more CNN layer

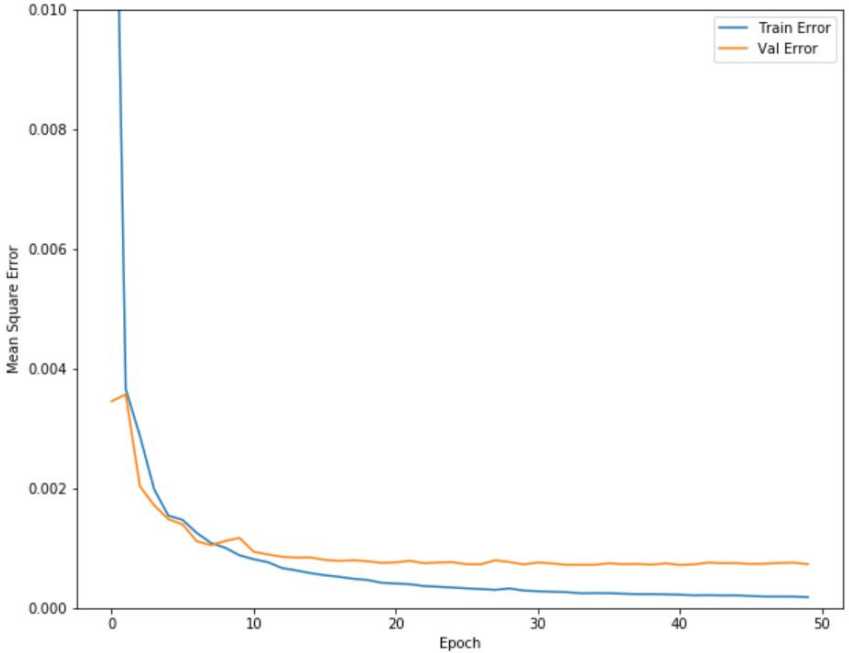
Eye Features Model

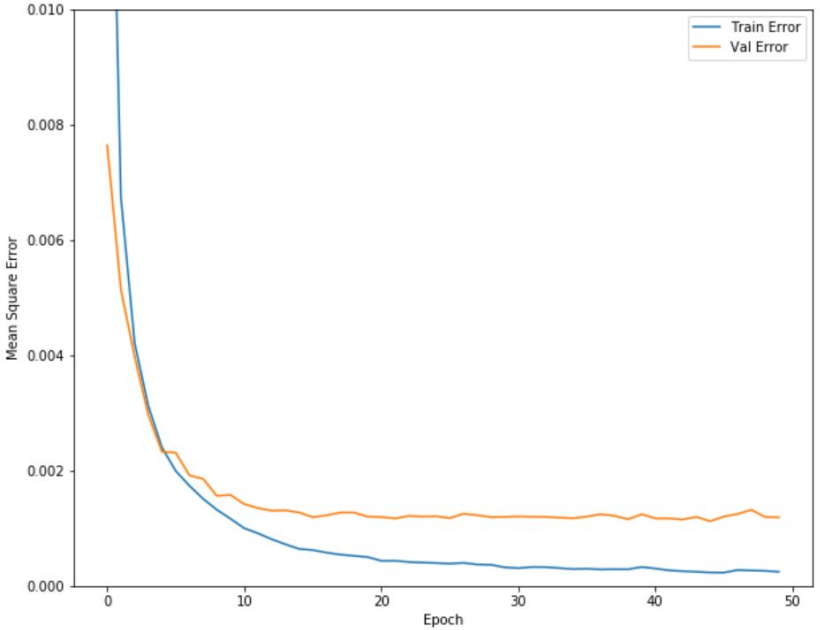
Input (56X96) from top
CNN1 -- 32 maps kernel size 5X5 activation leaky-relu
CNN2 -- 64 maps kernel size 3X3 activation leaky-relu
MaxPool -- 2X2 stride 2
CNN3 -- 64 maps kernel size kernel size 3X3 activation leaky-relu
CNN4 -- 64 maps kernel size kernel size 3X3 activation leaky-relu
MaxPool -- 2X2 stride 2
CNN5 -- 128 Maps Kernel size 2X2 activation leaky-relu
MaxPool -- 2X2 stride 2
Dense Layer 1 500 neurons activation leaky-relu
Dropout rate 0.25
Dense Layer 2 500 neurons activation leaky-relu
Output layer 20 neurons -- No activation as we have regression problem

Mouth and Nose Features Model

Input (56X96) from bottom
CNN1 -- 32 maps kernel size 5X5 activation leaky-relu
CNN2 -- 64 maps kernel size 3X3 activation leaky-relu
MaxPool -- 2X2 stride 2
CNN3 -- 64 maps kernel size kernel size 3X3 activation leaky-relu
CNN4 -- 64 maps kernel size kernel size 3X3 activation leaky-relu
MaxPool -- 2X2 stride 2
CNN5 -- 128 Maps Kernel size 2X2 activation leaky-relu
MaxPool -- 2X2 stride 2
Dense Layer 1 500 neurons activation leaky-relu
Dropout rate 0.25
Dense Layer 2 500 neurons activation leaky-relu
Output layer 10 neurons -- No activation as we have regression problem

Model10 Image Cropping





We see that both models are quite stable and not overfitting and also in general the loss is lower

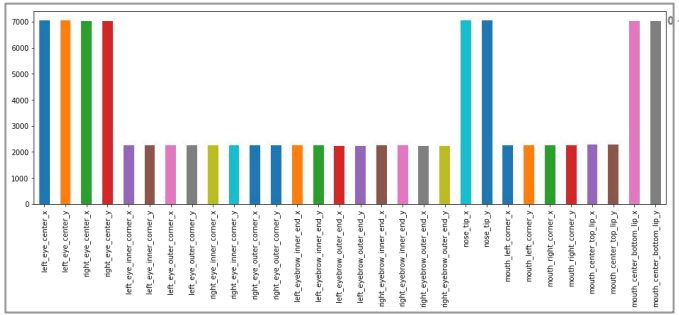
Model11

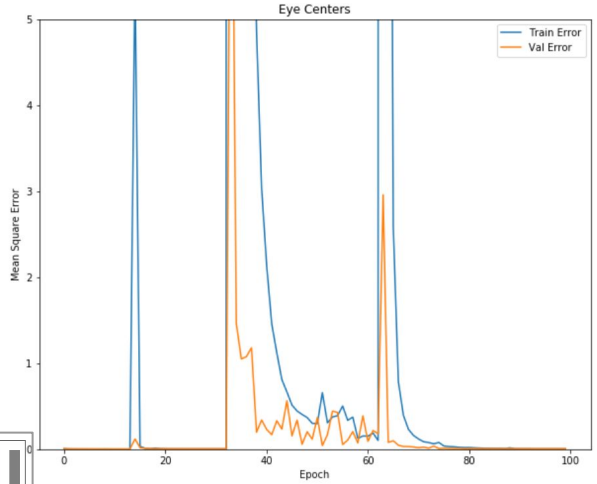
Using all the data . Even the images which didn't have all the key location specified

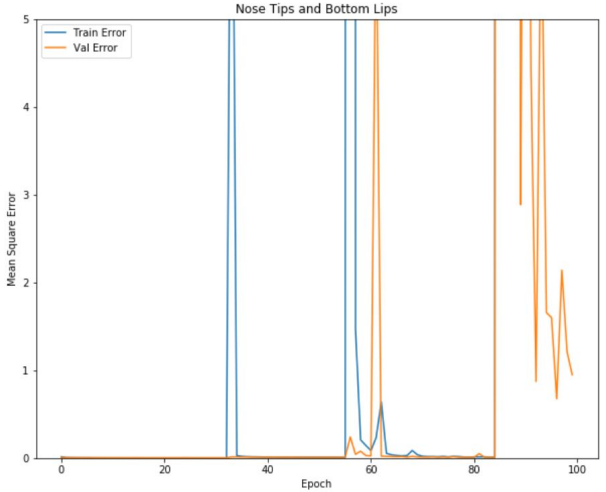
We created specialized CNNs for few locations

- **Eye Centers**
- **Nose and Bottom Lip**

Cropping of images were done and the models similar to model 10 were created







We see that the losses were not stabilizing and even diverging .

Model12

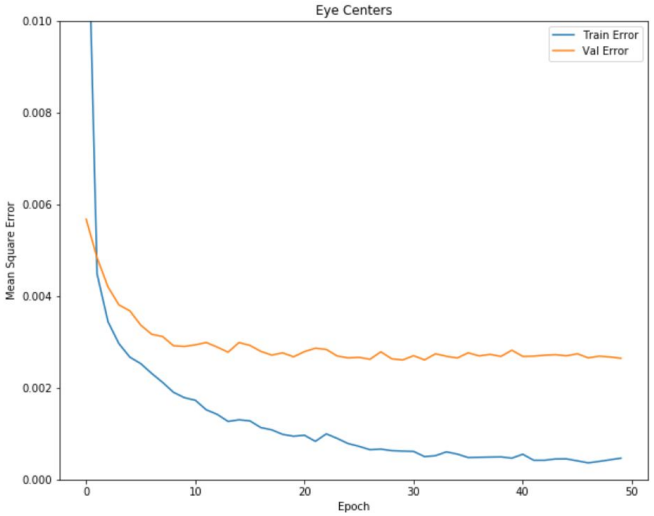
Using all the data . Even the images which didn't have all the key location specified

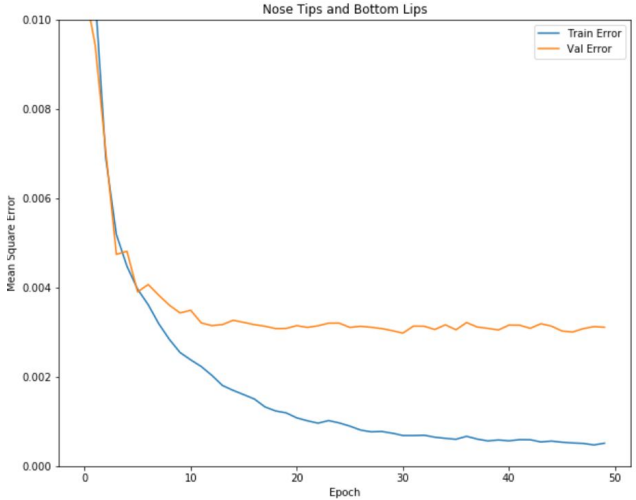
We created specialized CNNs for few locations

- **Eye Centers**
- **Nose and Bottom Lip**

Cropping of images were done and the models similar to model 10 were created

But this time the batch size was increased from 32 to 128



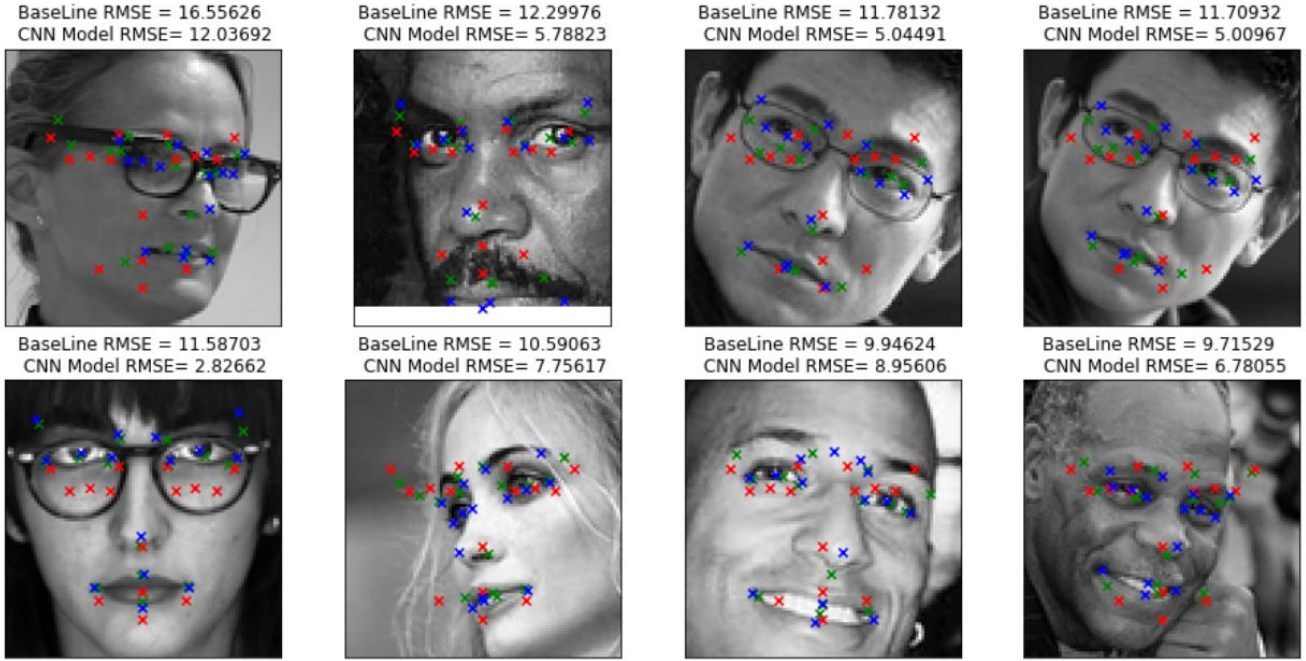


We see that the losses are now stabilizing. Validation loss is bigger than previous models as we have about 4 times more data.

Predicting Dev Set

RMSE for Key Features : 1.912592

	CNN RMSE
left_eye_center	1.221928
right_eye_center	1.288254
left_eye_inner_corner	1.266469
left_eye_outer_corner	1.498243
right_eye_inner_corner	1.338274
right_eye_outer_corner	1.629537
left_eyebrow_inner_end	1.873932
left_eyebrow_outer_end	2.346055
right_eyebrow_inner_end	1.934533
right_eyebrow_outer_end	2.452239
nose_tip	3.086948
mouth_left_corner	1.895786
mouth_right_corner	1.918741
mouth_center_top_lip	1.787617
mouth_center_bottom_lip	2.156498



Kaggle Submission

Submission and Description	Private Score	Public Score
submission.csv 10 hours ago by Anup Jha First Submission	1.95709	2.17532

#	Team Name	Kernel	Team Members	Score	Entries	Last
1	Trump			1.53319	1	2y
2	olegra			1.55021	4	2y
3	enes			1.64096	1	2y
4	helgi			1.73781	24	2y
5	Alastair Breeze			1.85774	1	2y
6	ddwe			1.88312	18	2y
7	Koguma			1.89203	5	2y
8	bigxin			1.95750	2	2y
9	ademenet			1.97115	8	2y
10	CCRS			2.03259	12	2y
11	navidbehazin			2.04092	42	2y
12	RASPUZ			2.08423	5	2y
13	PoojaLalan			2.10009	4	2y
14	Daisuke Hashimoto			2.10824	3	2y
15	TeamName		//	2.13319	1	2y
16	sellena			2.16093	62	2y
17	XuleiYang			2.16330	1	2y
18	Markus Neumann			2.17188	4	2y
19	JamesL			2.17633	1	2y
20	LAV			2.17788	5	2y

If the competition was open we would have been at rank 19 out of 175 teams

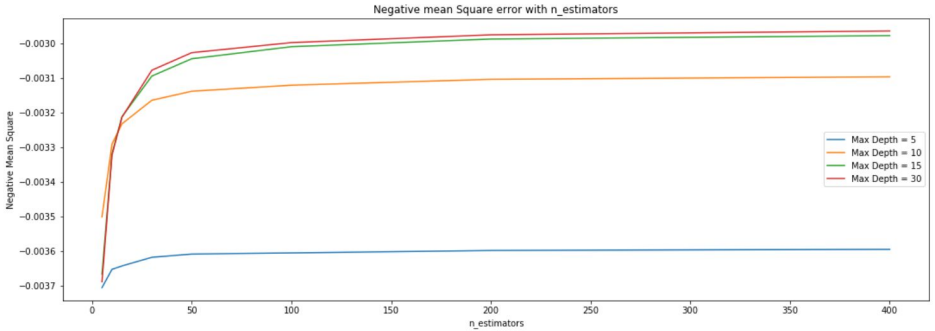
Things we would like to try in future

- Use GPU instead of CPU and tune more hyperparameters and train for more epochs
- Use haar features to isolate right eye area ,left eye area , nose and lips so that we can feed a small image to CNN and get more accurate locations
- Reduce the image size to 32X32 by using the average of 3 pixels.
    - We actually tried this on small data set but didn’t have enough time to run it for all image set using the specialized CNNS for features.
    - Preliminary findings suggest that the loss is similar to what we got in our best CNN

Other Methods

Random Forest :

- PCA reduction 99% variance
- Max Depth : 30
- N_estimators: 500
- Dev Set RMSE : 3.520137



Support Vector Regression:

- PCA reduction 99% variance
- C (Penalty): 1
- e(epsilon): 0.1
- Dev Set RMSE : 3.511949