



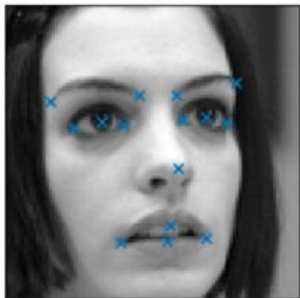
MIDS 207 Final Project - Team 4

Anup Jha

Anish Philip

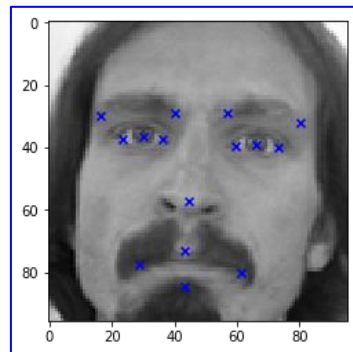
# Our motivation for this project was to understand the concepts of Convolutional Neural Network and its potential applications in Image processing

- “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...



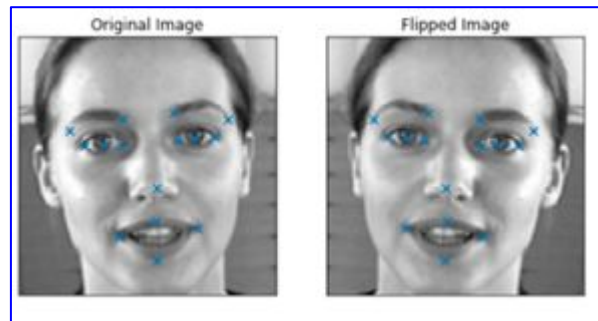
# There are 7,049 (96x96 Greyscale) training images with keypoint coordinates and 1,783 test images with No keypoint coordinates

- 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:

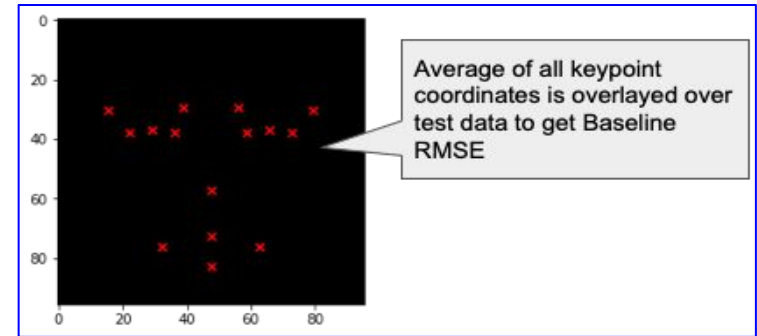
We flipped the training images on the 'center-vertical' axis to double the size of images for which we have keypoint coordinates



**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

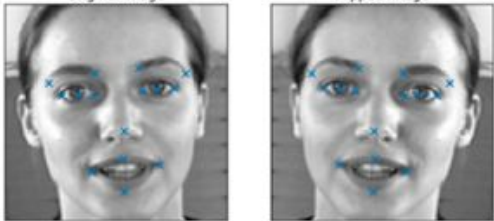

Features	RMSE	Features	RMSE	Features	RMSE
Left eye Center	2.52	Right eye Center	2.58	Nose tip	4.50
Left eye inner Corner	2.30	Right eye inner Corner	2.39	Mouth left corner	4.73
Left eye outer Corner	3.19	Right eye outer Corner	3.19	Mouth right corner	4.78
Left eyebrow inner end	3.28	Right eyebrow inner corner	3.39	Mouth center top lip	4.70
Left eyebrow outer end	4.06	Right eyebrow outer corner	4.20	Mouth center bottom lip	4.74



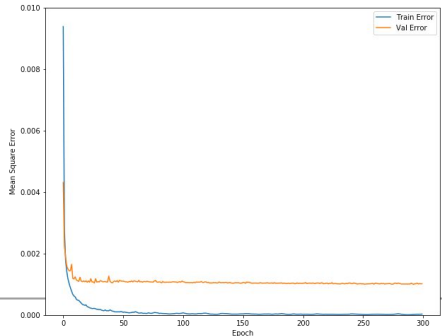
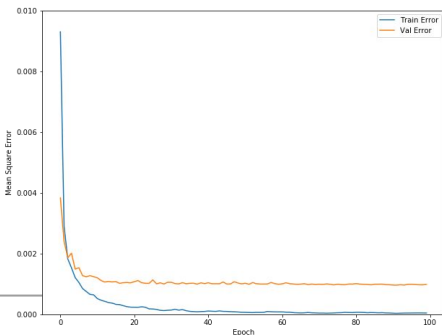
## 12 Convolutional Neural Net models were configured with varying hyperparameters and image augmentation to achieve the least possible Mean Squared Error

- 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**

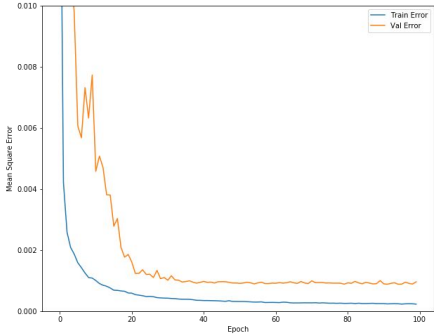
# We used different types of Image augmentation to improve sample size, performance and accuracy

Sample size	Processing time	Accuracy																																																																																	
<div><div>Original Image</div><div>Flipped Image</div><div></div></div> <p>Flipped Image on Center-Vertical axis</p>	<div><table><tr><td>94</td><td>53</td><td>176</td><td>195</td><td>189</td><td>11</td><td>34</td><td>35</td><td>34</td></tr><tr><td>8</td><td></td><td>221</td><td>64</td><td></td><td>134</td><td>162</td><td></td><td>162</td></tr><tr><td>68</td><td>203</td><td>253</td><td>44</td><td>11</td><td>36</td><td>228</td><td>3</td><td>228</td></tr><tr><td>203</td><td>71</td><td>112</td><td>244</td><td>62</td><td>248</td><td>253</td><td>211</td><td>253</td></tr><tr><td>360</td><td></td><td>123</td><td>211</td><td></td><td>6</td><td>71</td><td></td><td>71</td></tr><tr><td>8</td><td>42</td><td>221</td><td>64</td><td>57</td><td>134</td><td>162</td><td>215</td><td>162</td></tr><tr><td>126</td><td>77</td><td>213</td><td>76</td><td>82</td><td>194</td><td>93</td><td>134</td><td>93</td></tr><tr><td>202</td><td></td><td>244</td><td>26</td><td></td><td>250</td><td>213</td><td></td><td>213</td></tr><tr><td>68</td><td>203</td><td>253</td><td>44</td><td>11</td><td>36</td><td>228</td><td>3</td><td>228</td></tr></table></div> <p>Take an average of all surrounding pixels to replace the center pixel value. This would shrink NxN image to (N/3)x(N/3)</p>	94	53	176	195	189	11	34	35	34	8		221	64		134	162		162	68	203	253	44	11	36	228	3	228	203	71	112	244	62	248	253	211	253	360		123	211		6	71		71	8	42	221	64	57	134	162	215	162	126	77	213	76	82	194	93	134	93	202		244	26		250	213		213	68	203	253	44	11	36	228	3	228	<div><div>Eye feature cropped image</div><div>Mouth and nose feature cropped image</div><div></div></div> <p>Cropped image on center-horizontal axis allowing some overlap in two resulting images</p>
94	53	176	195	189	11	34	35	34																																																																											
8		221	64		134	162		162																																																																											
68	203	253	44	11	36	228	3	228																																																																											
203	71	112	244	62	248	253	211	253																																																																											
360		123	211		6	71		71																																																																											
8	42	221	64	57	134	162	215	162																																																																											
126	77	213	76	82	194	93	134	93																																																																											
202		244	26		250	213		213																																																																											
68	203	253	44	11	36	228	3	228																																																																											

# 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

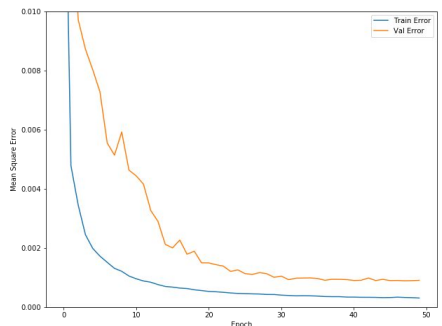
Model	Feature Learning Layers	Classification/ Dense Layers	Training and Validation Error
1	<ul style="list-style-type: none"> <li>32 filter Convolutional Layer (3x3 Kernel) ReLU activation and Max Pooling</li> <li>64 filter Convolutional Layer (3x3 Kernel) ReLU activation and Max Pooling</li> <li>128 filter Convolutional Layer (3x3 Kernel) ReLU activation and Max Pooling</li> </ul>	<ul style="list-style-type: none"> <li>Flatten</li> <li>Dense Layer 1 (500, ReLU)</li> <li>Dense Layer 2 (500, ReLU)</li> <li>Output Layer (30, Linear)</li> </ul>	
2	<ul style="list-style-type: none"> <li>32 filter Convolutional Layer (3x3 Kernel) <b>Leaky</b> ReLU activation and Max Pooling</li> <li>64 filter Convolutional Layer (3x3 Kernel) <b>Leaky</b> ReLU activation and Max Pooling</li> <li>128 filter Convolutional Layer (3x3 Kernel) <b>Leaky</b> ReLU activation and Max Pooling</li> </ul>	<ul style="list-style-type: none"> <li>Flatten</li> <li>Dense Layer 1 (500, ReLU)</li> <li>Dense Layer 2 (500, ReLU)</li> <li>Output Layer (30, Linear)</li> </ul>	

# 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	<ul style="list-style-type: none"> <li>32 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling &amp; Drop out (0.10)</li> <li>64 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling &amp; Drop out (0.15)</li> <li>128 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling &amp; Drop out (0.20)</li> </ul>	<ul style="list-style-type: none"> <li>Flatten</li> <li>Dense Layer 1 (500, ReLU)</li> <li>Drop out (0.25)</li> <li>Dense Layer 2 (500, ReLU)</li> <li>Output Layer (30, Linear)</li> </ul>	
4	<ul style="list-style-type: none"> <li>32 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling &amp; MiniBatch Norm</li> <li>64 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling &amp; MiniBatch Norm</li> <li>128 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling &amp; MiniBatch Norm</li> </ul>	<ul style="list-style-type: none"> <li>Flatten</li> <li>Dense Layer 1 (500, ReLU)</li> <li>MiniBatch Norm</li> <li>Dense Layer 2 (500, ReLU)</li> <li>MiniBatch Norm</li> <li>Output Layer (30, Linear)</li> </ul>	



# 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

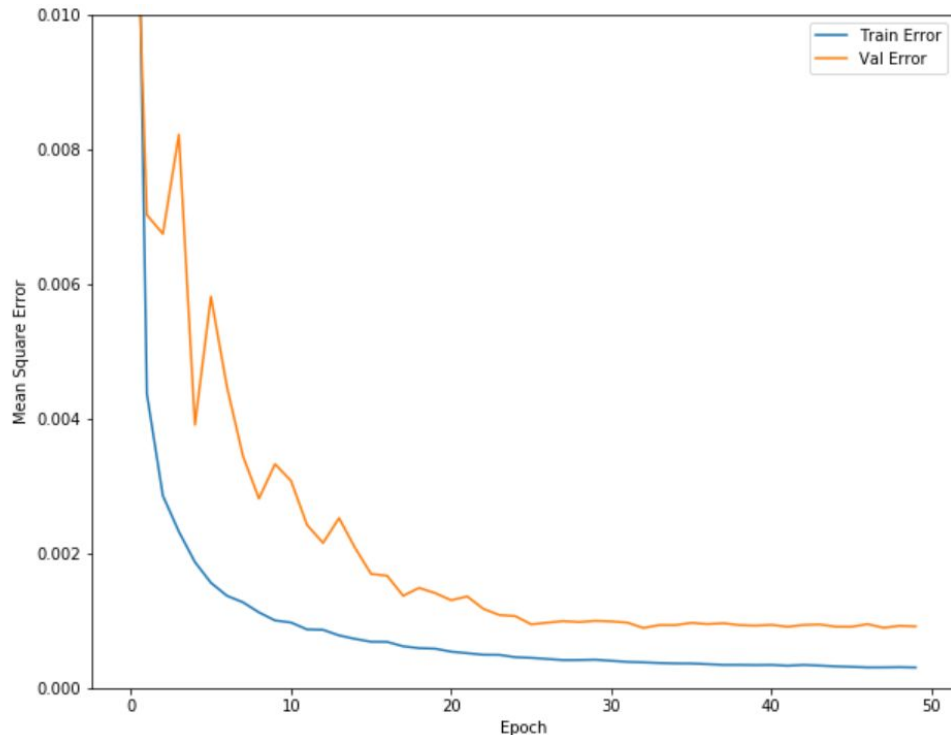
Model	Feature Learning/ Convolutional Layers	Classification/ Dense Layers	Training and Validation Error																								
5	<ul style="list-style-type: none"><li>96 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling, MiniBatch Norm &amp; Drop out (0.10)</li><li>128 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling, MiniBatch Norm &amp; Drop out (0.15)</li><li>128 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling, MiniBatch Norm &amp; Drop out (0.20)</li></ul>	<ul style="list-style-type: none"><li>Flatten</li><li>Dense Layer 1 (1000, ReLU)</li><li>MiniBatch Norm</li><li>Drop out (0.25)</li><li>Dense Layer 2 (1000, ReLU)</li><li>MiniBatch Norm</li><li>Drop out (0.30)</li><li>Output Layer (30, Linear)</li></ul>	<p><i>Not optimal for execution on CPU memory</i></p>																								
6	<ul style="list-style-type: none"><li>32 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling</li><li>64 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling, MiniBatch Norm &amp; Drop out (0.10)</li><li>64 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling, MiniBatch Norm &amp; Drop out (0.15)</li><li>128 filter Convolutional Layer (3x3 Kernel) Leaky ReLU activation, Max Pooling, MiniBatch Norm &amp; Drop out (0.20)</li></ul>	<ul style="list-style-type: none"><li>Flatten</li><li>Dense Layer 1 (500, ReLU)</li><li>Drop out (0.25)</li><li>Dense Layer 2 (500, ReLU)</li><li>Output Layer (30, Linear)</li></ul>	 <table><caption>Approximate Mean Square Error (MSE) values for Model 6</caption><thead><tr><th>Epoch</th><th>Train Error</th><th>Val Error</th></tr></thead><tbody><tr><td>0</td><td>0.010</td><td>0.010</td></tr><tr><td>5</td><td>0.003</td><td>0.008</td></tr><tr><td>10</td><td>0.001</td><td>0.006</td></tr><tr><td>20</td><td>0.0005</td><td>0.002</td></tr><tr><td>30</td><td>0.0002</td><td>0.0015</td></tr><tr><td>40</td><td>0.0001</td><td>0.0012</td></tr><tr><td>50</td><td>0.0001</td><td>0.001</td></tr></tbody></table>	Epoch	Train Error	Val Error	0	0.010	0.010	5	0.003	0.008	10	0.001	0.006	20	0.0005	0.002	30	0.0002	0.0015	40	0.0001	0.0012	50	0.0001	0.001
Epoch	Train Error	Val Error																									
0	0.010	0.010																									
5	0.003	0.008																									
10	0.001	0.006																									
20	0.0005	0.002																									
30	0.0002	0.0015																									
40	0.0001	0.0012																									
50	0.0001	0.001																									

# 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.**

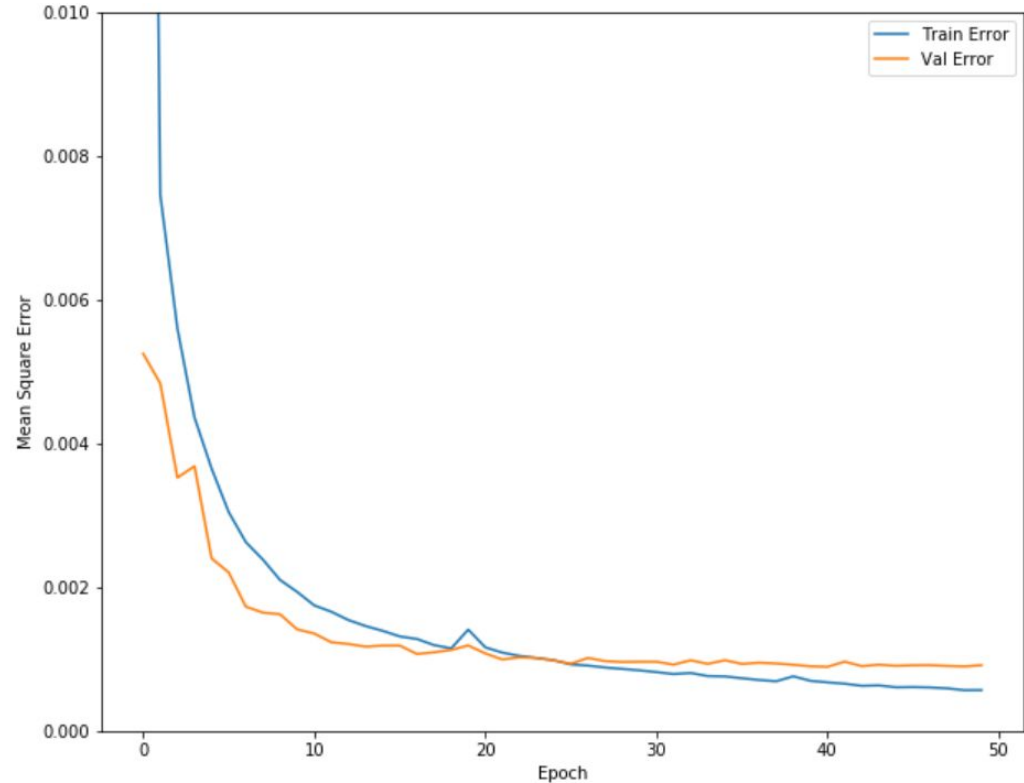


# Model8

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 dropouts in the convolution layers this time.**

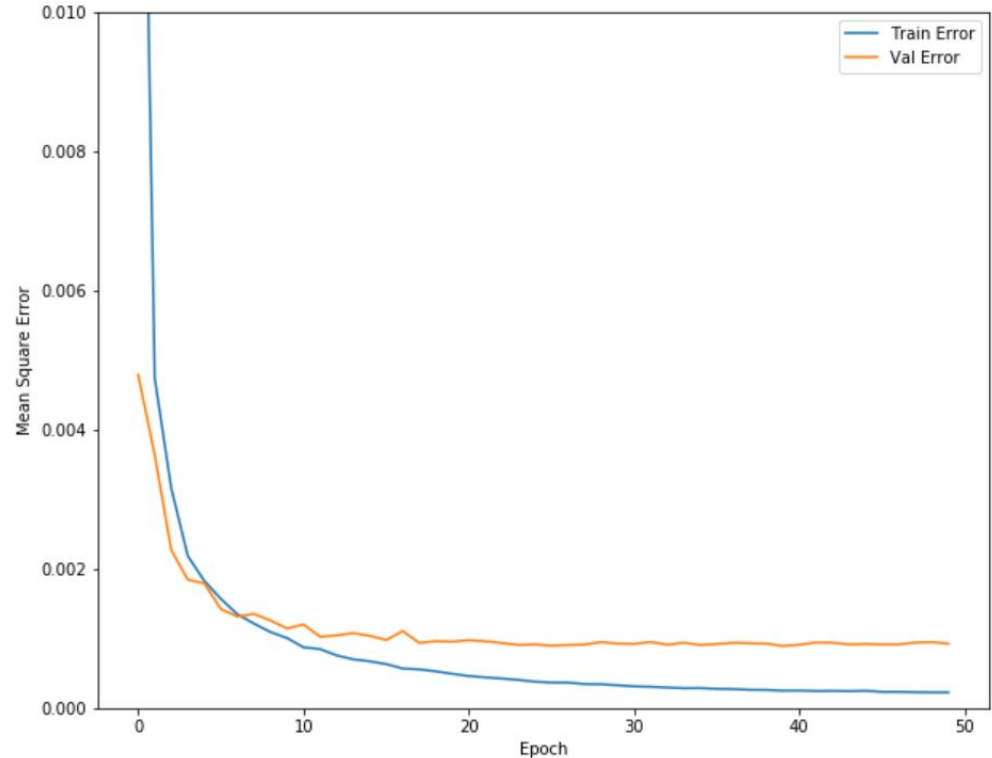


# Model9

## 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**

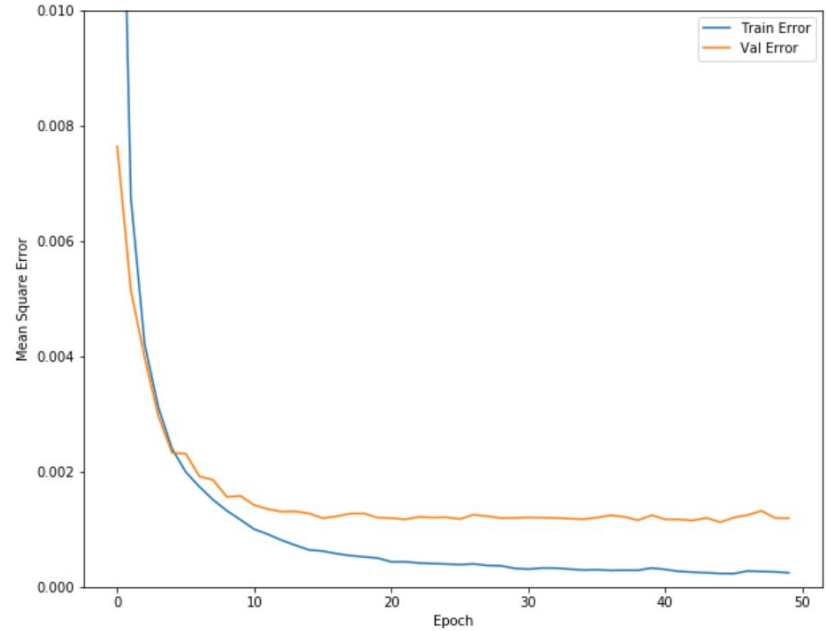
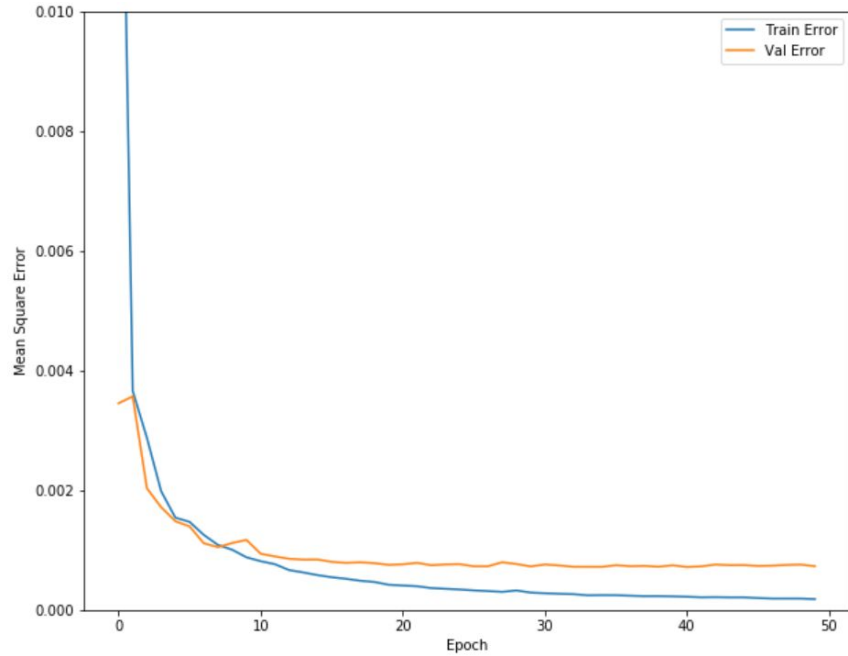
### Eye Features Model

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 3X3 activation leaky-relu  
CNN4 -- 64 maps 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 3X3 activation leaky-relu  
CNN4 -- 64 maps 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**

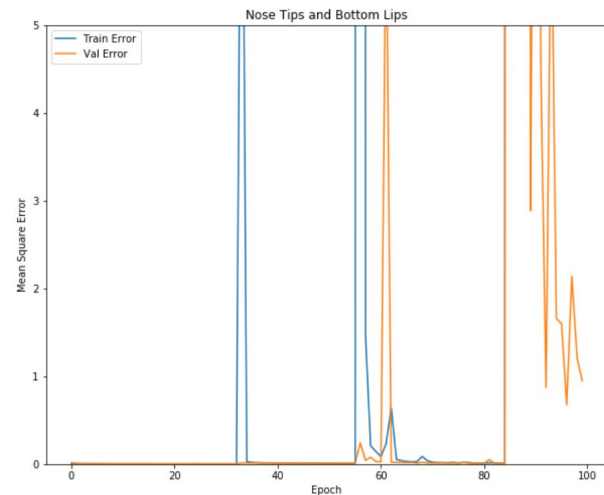
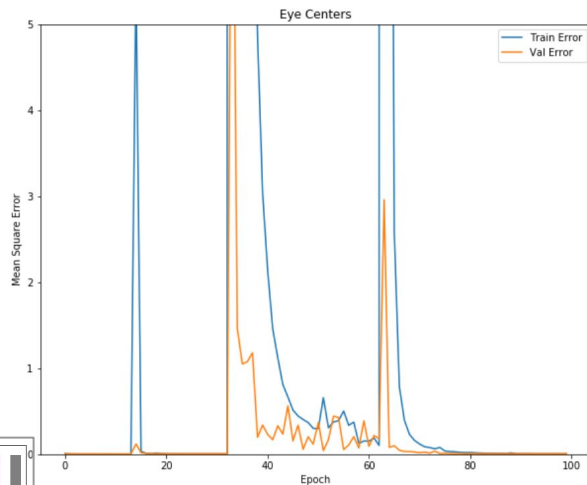
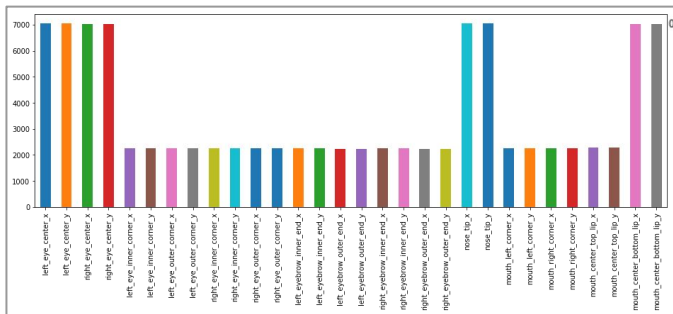
# Model11

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

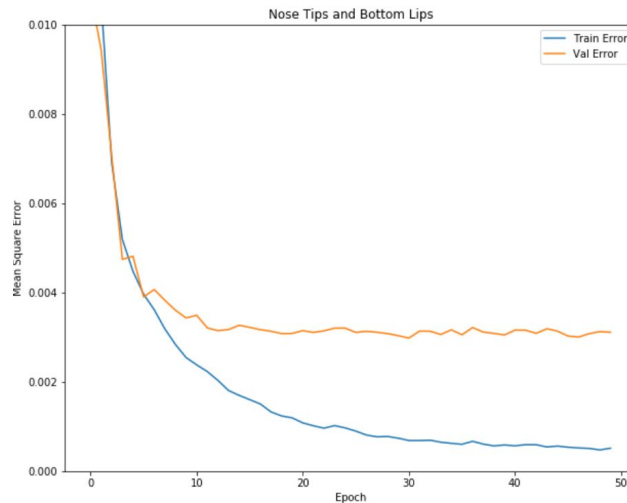
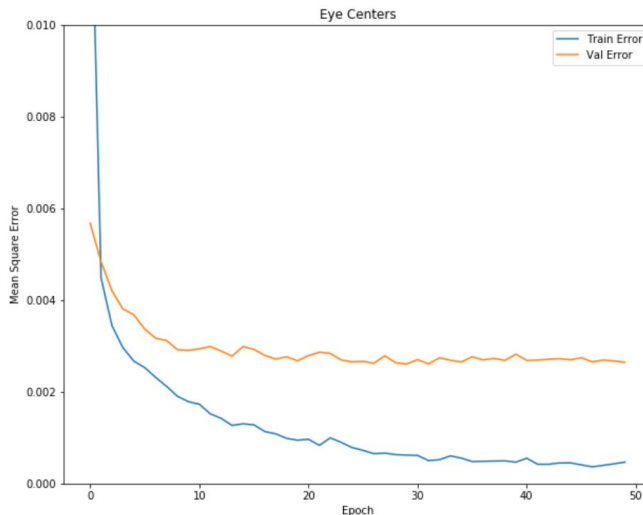
Using all the data . Even the images which didn't have all the key location specified

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.

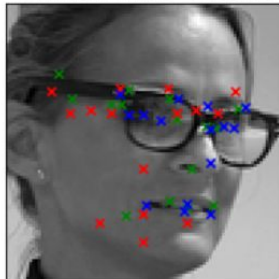


# Predicting Dev Set

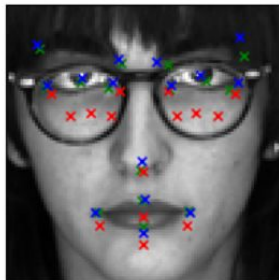
RMSE for Key Features : 1.912592

	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

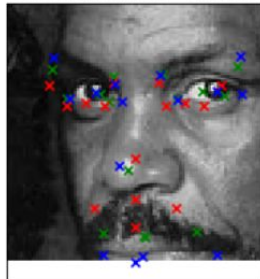
BaseLine RMSE = 16.55626  
CNN Model RMSE= 12.03692



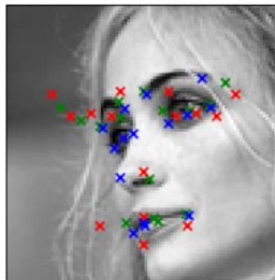
BaseLine RMSE = 11.58703  
CNN Model RMSE= 2.82662



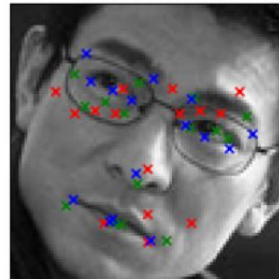
BaseLine RMSE = 12.29976  
CNN Model RMSE= 5.78823



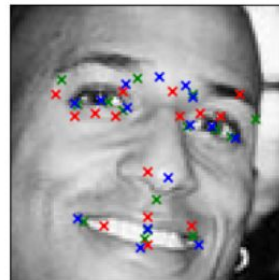
BaseLine RMSE = 10.59063  
CNN Model RMSE= 7.75617



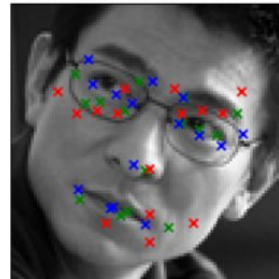
BaseLine RMSE = 11.78132  
CNN Model RMSE= 5.04491



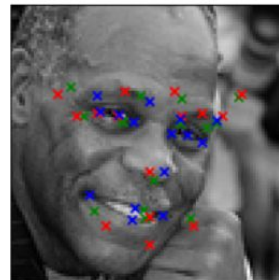
BaseLine RMSE = 9.94624  
CNN Model RMSE= 8.95606



BaseLine RMSE = 11.70932  
CNN Model RMSE= 5.00967



BaseLine RMSE = 9.71529  
CNN Model RMSE= 6.78055



# Kaggle Submission

Submission and Description	Private Score	Public Score
----------------------------	---------------	--------------

submission.csv

1.95709

2.17532

10 hours ago by Anup Jha

First Submission

#	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

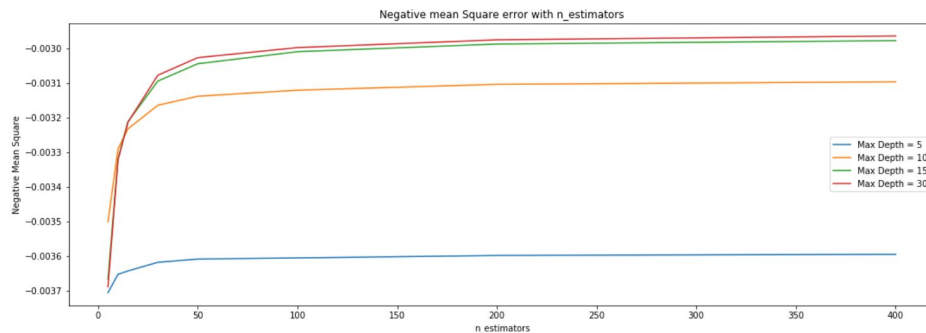
# Things we would like to try in future

- 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

