Facial Keypoints Detection

Using CNN to detect facial feature endpoints in facial image library

Legg Yeung
Alberto Melgoza
Jennifer Philippou
Chris Danicic

Agenda

- Introduction
- CNN overview
- Baseline results
- Building specialist models
- Data augmentation techniques & performance
- Model evaluation & final results
- Future refinements

The Kaggle competition

Objective:
To predict 15 keypoint positions on face images

Applications:

- Tracking faces in images and video
- Analysing facial expressions
- Detecting dysmorphic facial signs for medical diagnosis
- Biometrics / face recognition

Challenges:

- Facial features vary greatly
  - 3D pose
  - Size
  - Position
  - Viewing angle
  - Illumination conditions

Technology leveraged for the project

- AWS G2.2x.large EC2 instance
  - vCPU - 8
  - ECU - 26
  - Memory - 15 GB
  - GPU - 4 GB
  - Ubuntu OS with Python 3
  - Cost/hr - $0.65
- W205 security rules + Jupyter port
- EC2 Instance access tools:
  - Windows:
    - Babun (windows shell)
    - pscp (to upload files)
  - OSX/Unix
    - Terminal shell & ssh

	Baseline Model	Specialist Model	Total
Time	1.6 Hrs	81 Hrs	82.6 Hrs
Cost	$1.1	$52	$53.1

GPU Instances - Current Generation

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)
p2.xlarge	4	12	61	EBS Only
p2.8xlarge	32	94	488	EBS Only
p2.16xlarge	64	188	732	EBS Only
p3.2xlarge	8	23.5	61	EBS Only
p3.8xlarge	32	94	244	EBS Only
p3.16xlarge	64	188	488	EBS Only
g2.2xlarge	8	26	15	60 SSD

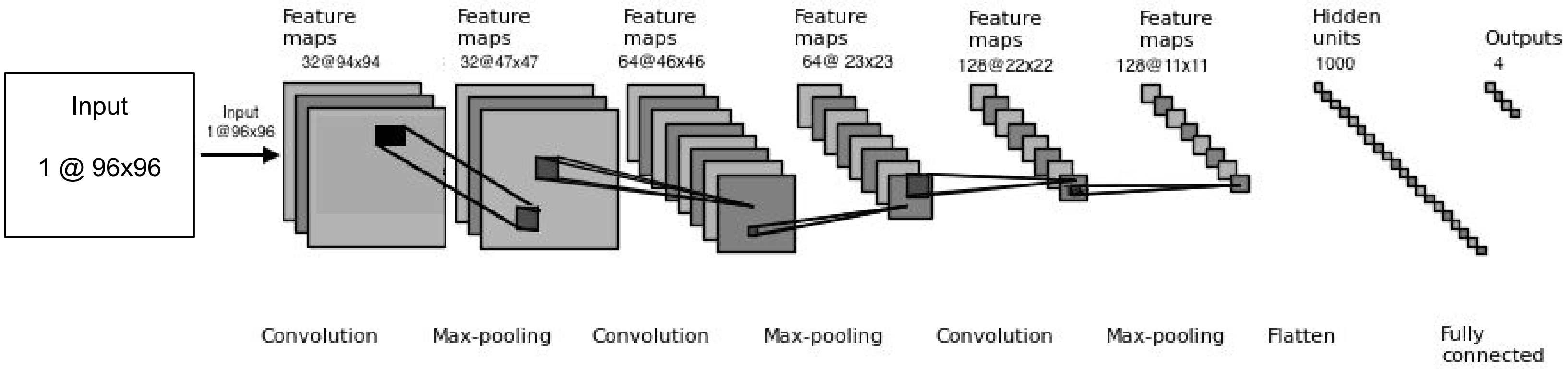
Convolutional Neural Net Overview

Convolution layer
Kernel filters convolve over length and width, computes dot product between filter and input to develop map of activation filters that activate when feature is present

Pooling / downsampling
Reduce spatial size by consolidating larger sampling area to smaller area.

Ex: 2x2 max pooling scans each 2x2 and assigns the max value per each pooling stride

Flattening
Convert feature maps, which process the 2D information, into non-spatially related vector for final classification output



Tuning our CNN’s parameters

```
# Neural Network with 16561502 learnable parameters

## Layer information

  #  name      size
---  --------  ---------
  0  input     1x96x96
  1  conv1     32x94x94
  2  pool1     32x47x47
  3  dropout1  32x47x47
  4  conv2     64x46x46
  5  pool2     64x23x23
  6  dropout2  64x23x23
  7  conv3     128x22x22
  8  pool3     128x11x11
  9  dropout3  128x11x11
 10  hidden4   1000
 11  dropout4  1000
 12  hidden5   1000
 13  output    30
```

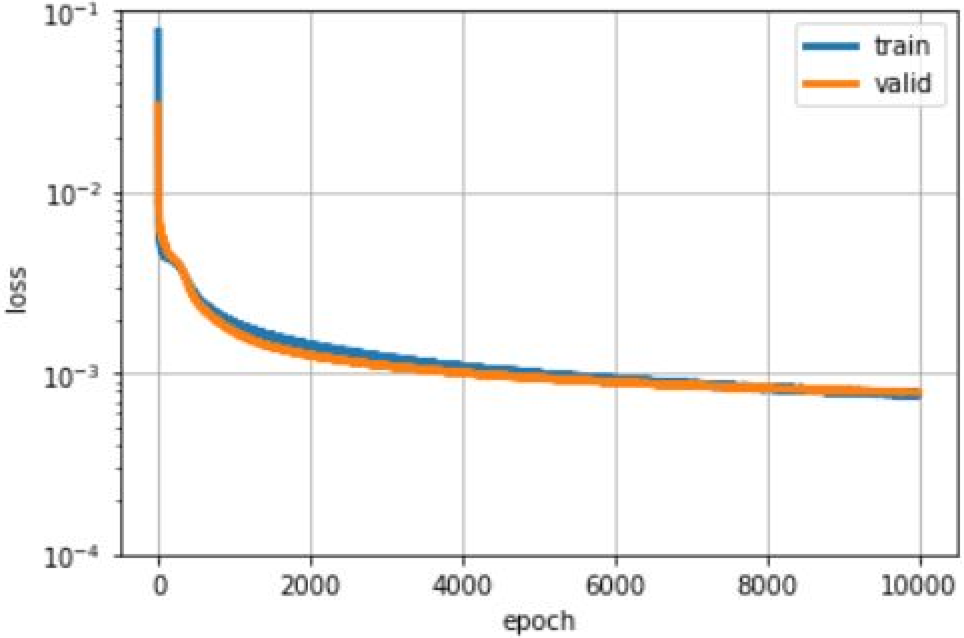
Epoch max: 3,000

Batch size: 128

Early stopping: 200 epochs

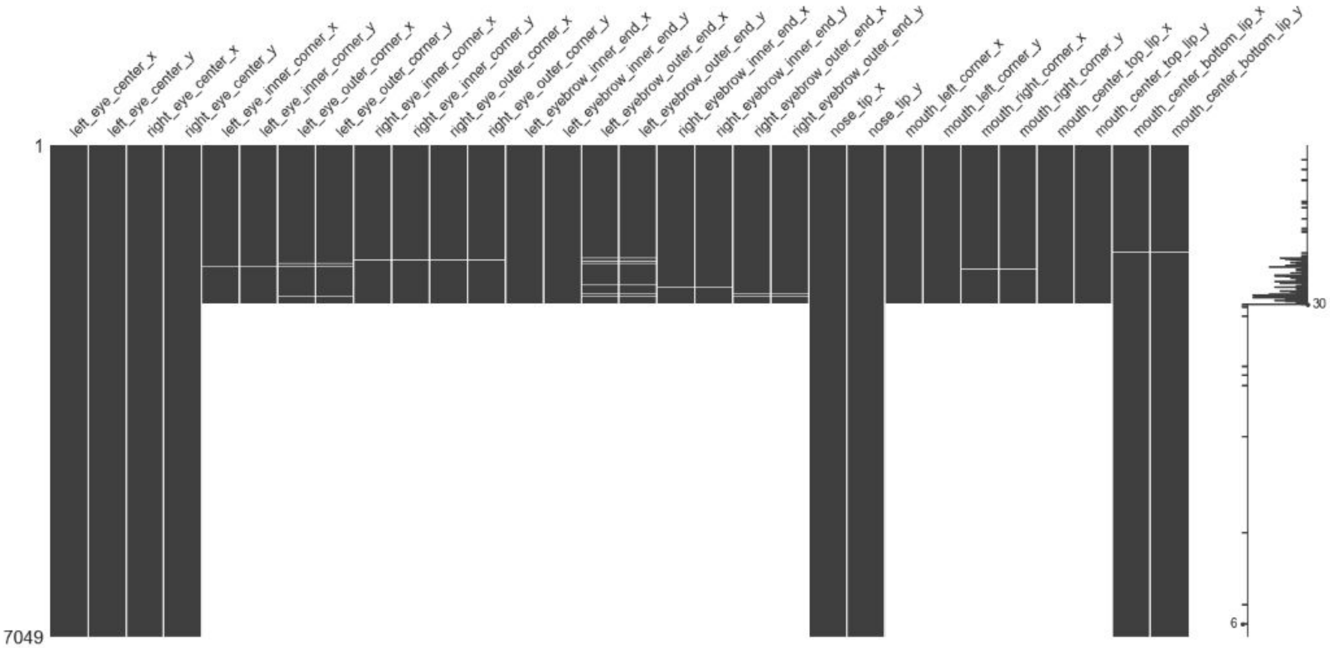
	Start	Stop	Rate
Learning	0.03	0.001	0.03
Momentum	0.90	0.999	0.90

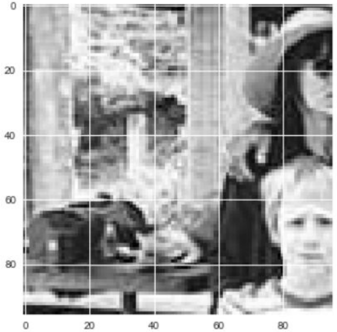
Baseline results

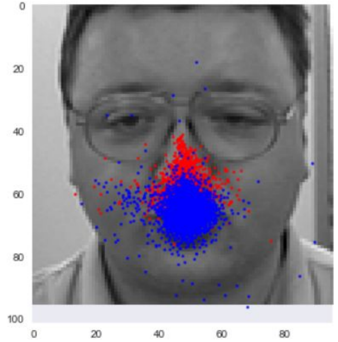


Final Valid Loss
1.3491330549652989

Exploratory Data Analysis

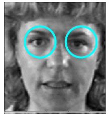
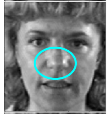
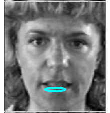






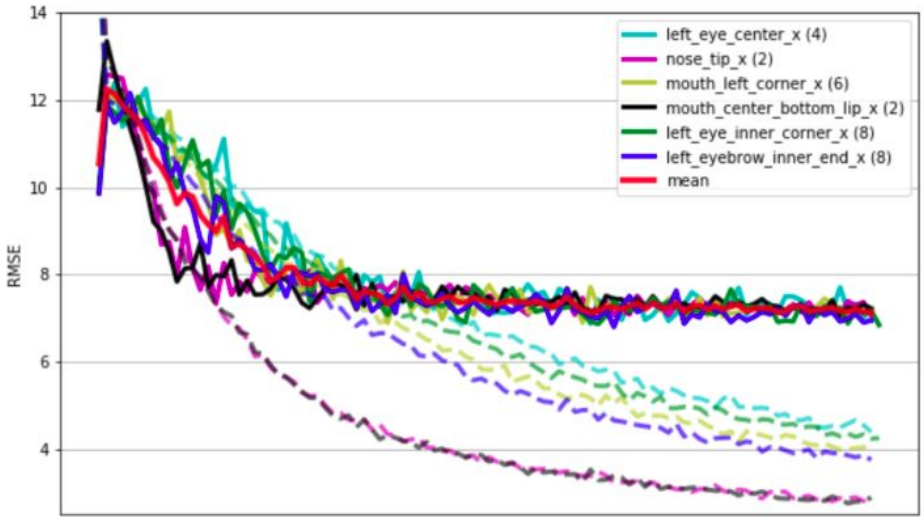
- Majority of training set did not have data for all 30 points
- Filtered images for completeness before training model

Specialist Overview

	Specialist Name	Specialist Number	Time per Epoch	Sample size	Total Seconds	Total Hrs	Best RSME
	L/R eye centers	0	25 sec	6,839	75,000	21	1.95
	Nose tip	1	25 sec	6,849	75,000	21	2.75
	R-mid-L mouth	2	7 sec	2,060	21,000	6	2.10
	Mouth center bottom	3	25 sec	6,816	75,000	21	2.50
	Eye Corner	4	7 sec	2,047	21,000	6	1.85
	L/R eyebrows	5	8 sec	1,990	24,000	7	2.00
		Total	**97 sec**	**7, 049**	**291,000**	**81**	**2.04**

Building the specialist models

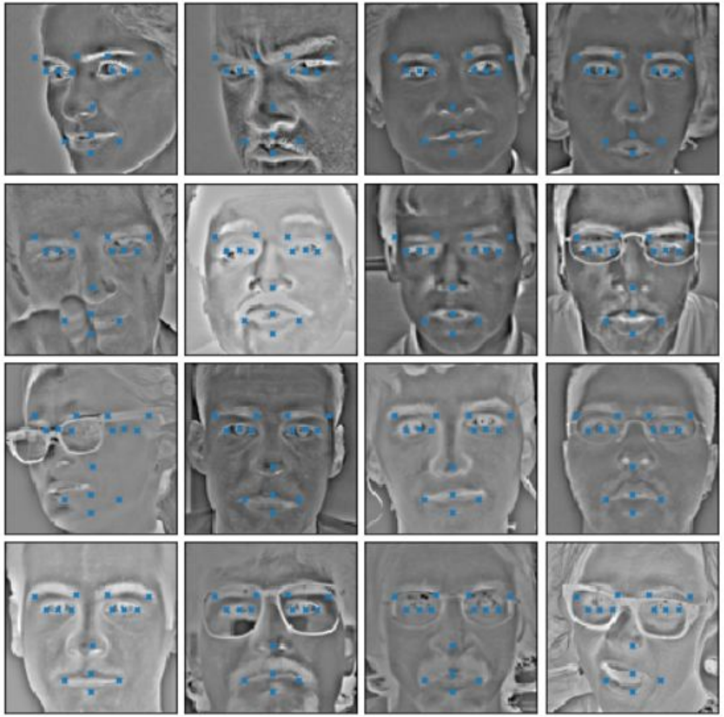
- Pickle update
  - Each specialist model targeted specific points
  - .pickle files for each model
- Debugging datashape for NN code
  - Both the type and shape of input data to model had to be standardized
- Writing a new load function
  - Minor tweaks required for loading data into training set, including reshaping image pixel data
- Trouble shooting specialist model
  - visual correction
  - kaggle update
  - changing the number of inputs/outputs



Data Augmentation Techniques: Part I

Fingerprint motif (Contrast)

Inverse Blur



Data Augmentation Techniques: Part II

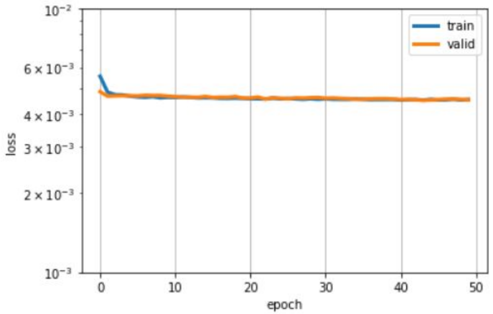
Adaptive Filter (9x9)

Adaptive Filter (3x3)

Performance of Augmentation

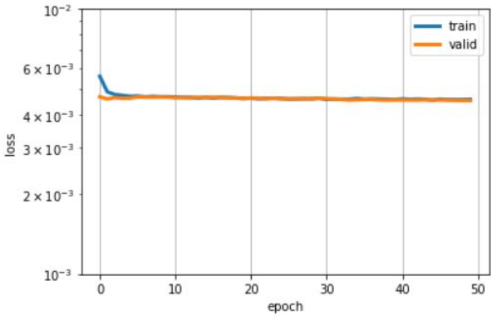
Fingerprint motif (Contrast)

0.00450



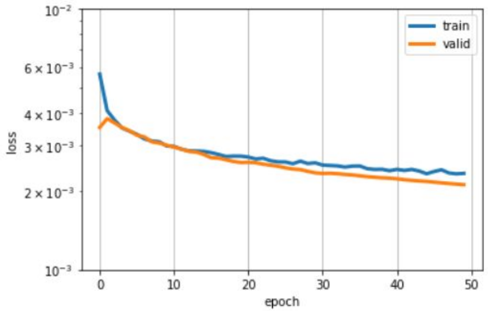
Inverse Blur

0.00452



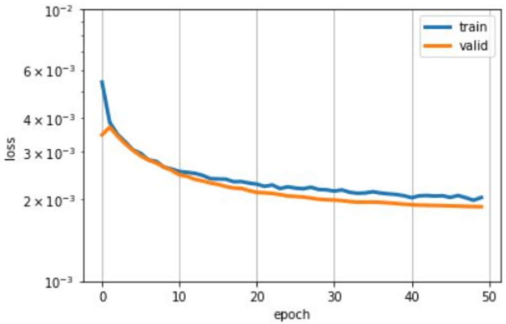
Adaptive Filter (9x9)

0.00212



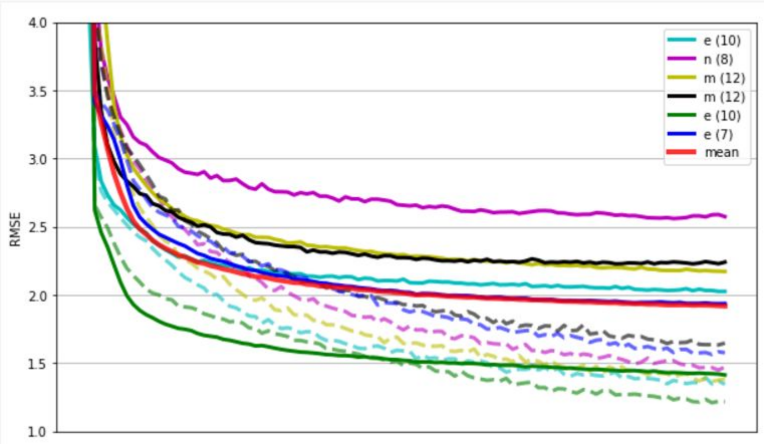
Adaptive Filter (3x3)

0.00188



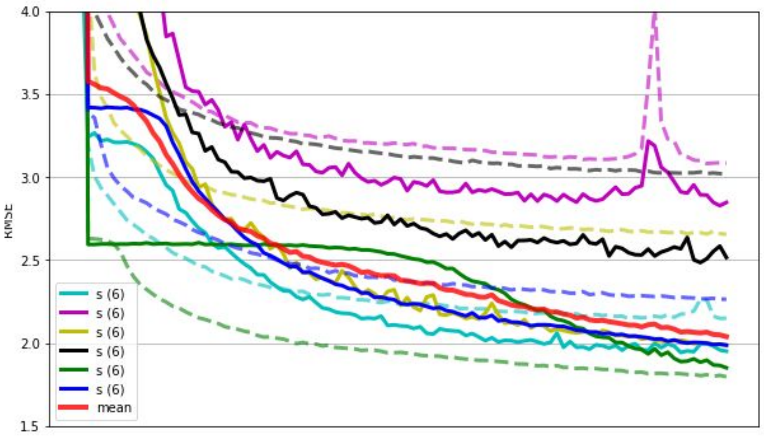
Final Result

Pre Data Augmentation



Mean Validation Loss: 1.92

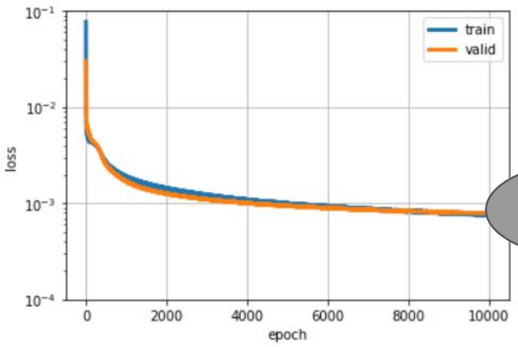
Post Data Augmentation



Mean Validation Loss: 2.04

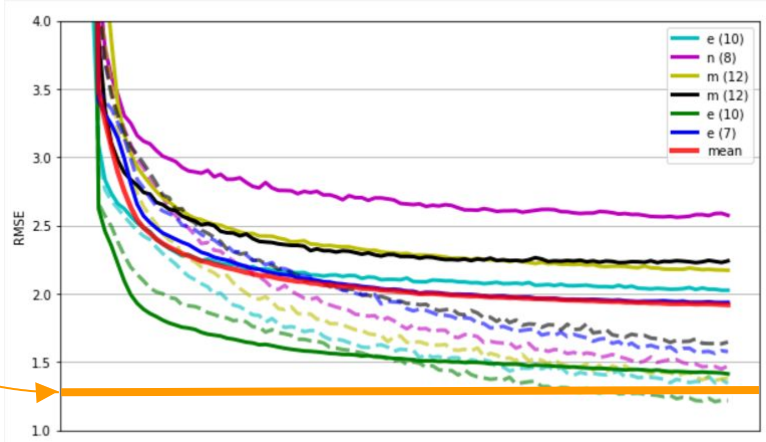
Final Result

Pre Specialist (complete cases)



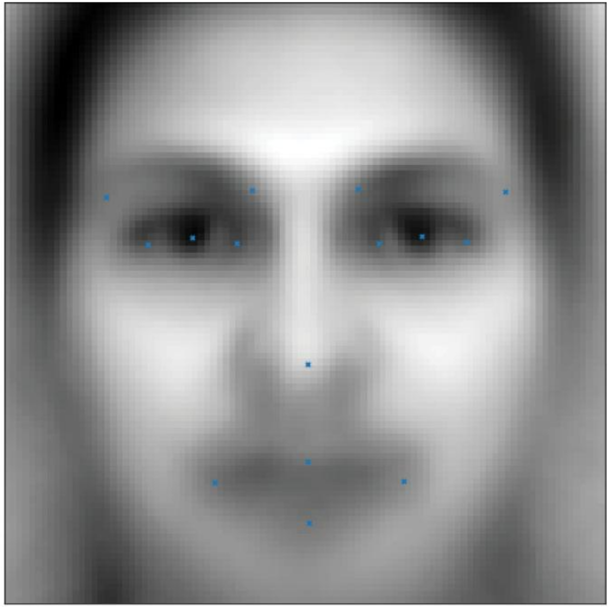
RMSE
1.35

Specialists



Model 0

Model 0: Averaging -- Predict same key points for every image



Model 0

MSE Train: 0.004052768
MSE Dev : 0.0044381749

Worst 4 Labels :
mouth_center_top_lip_y
0.0094488077
mouth_center_bottom_lip_y
0.0090680355
nose_tip_y
0.0089065293
mouth_left_corner_y
0.0076697934

Best 4 Labels :
right_eye_inner_corner_x
0.0012303042
right_eye_inner_corner_y
0.001285556
left_eye_inner_corner_x
0.0013126049
left_eye_center_x
0.0013908964

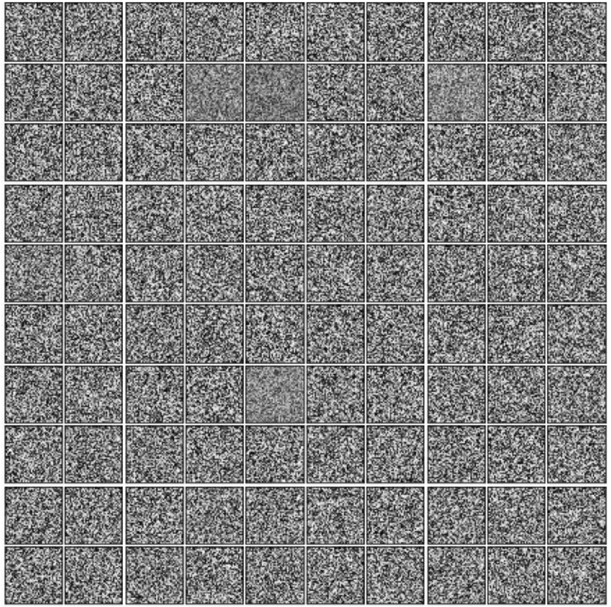
Worst 4 Images

Best 4 Images

Model 1

Model 1: Neural Net
Input Nodes : 96 by 96
Output Nodes : 30
Max Epochs : 1000

Hidden Layer 1 Nodes : 100



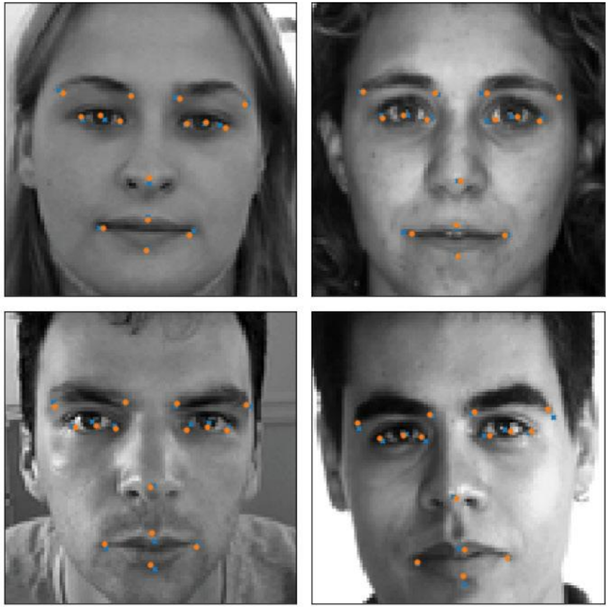
Model 1

MSE Train: 0.0022756304
MSE Dev : 0.0019805911-Leaked

Worst 4 Labels :
mouth_center_bottom_lip_y
0.005132813
nose_tip_y
0.0041158358
mouth_right_corner_x
0.0034587171
nose_tip_x
0.0033688443

Best 4 Labels :
left_eye_inner_corner_y
0.00074552454
right_eye_inner_corner_y
0.00078256853
left_eye_center_y
0.00087295735
right_eye_inner_corner_x
0.00091137283

Worst 4 Images

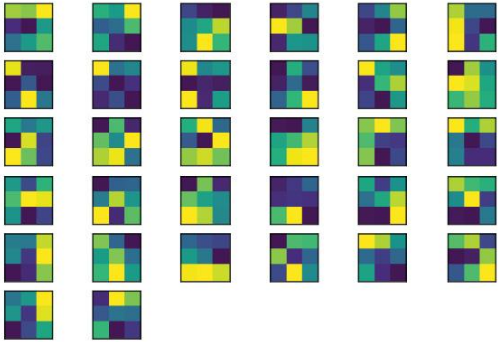


Best 4 Images

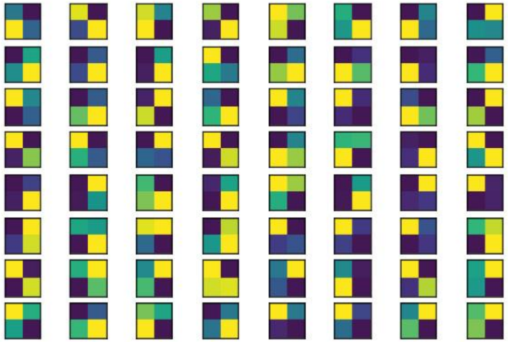
Model 2

Model 2: Convolutional Neural Net
Input Nodes : 96 by 96
Output Nodes : 30
Max Epochs : 1000

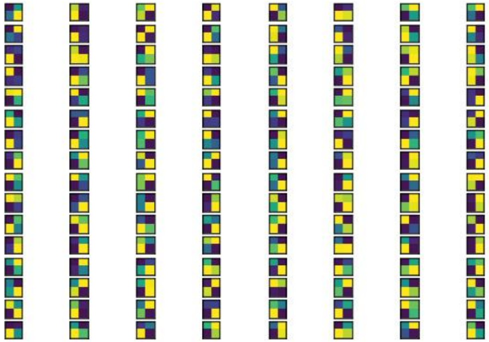
Convolutional Filters : (32, 64,128)
Filter size : (3 by 3, 2 by 2, 2 by 2)
Pool size : 2 by 2
Hidden Layer 1 Nodes : 500



32 Convolution Filters
3 by 3



64 Convolution Filters
2 by 2

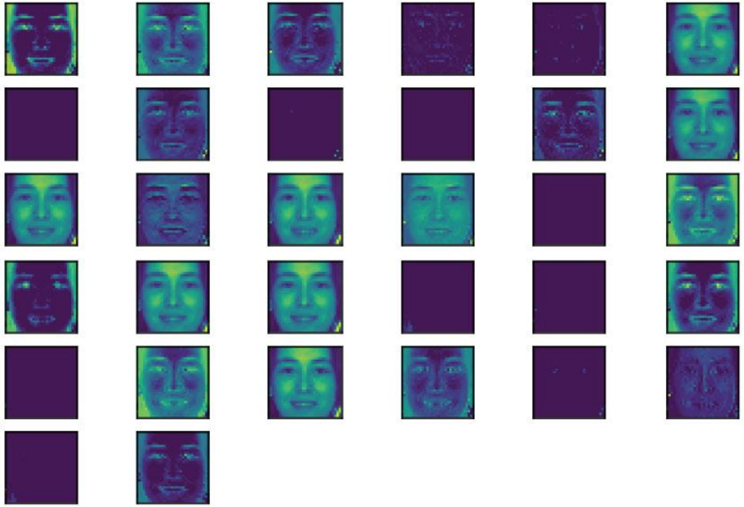


128 Convolution Filters
2 by 2

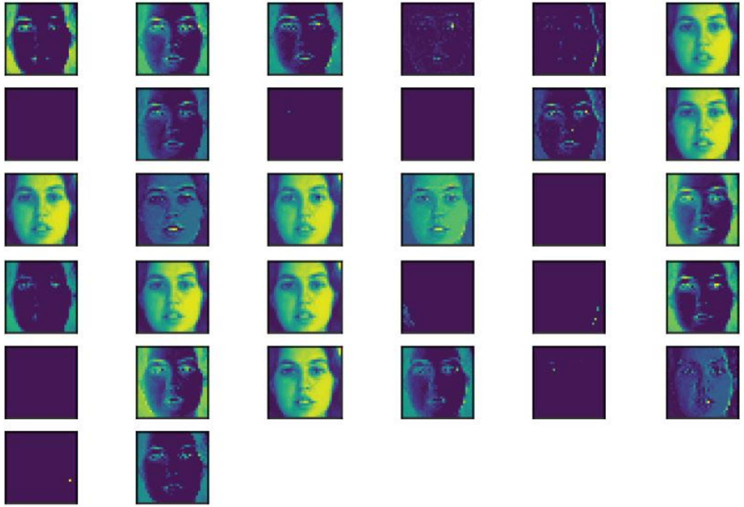
Model 2

Model 2: Convolutional Neural Net
Input Nodes : 96 by 96
Output Nodes : 30
Max Epochs : 1000

Convolutional Filters : (32, 64,128)
Filter size : (3 by 3, 2 by 2, 2 by 2)
Pool size : 2 by 2
Hidden Layer 1 Nodes : 500



Applying 32 convolution filters (5by5) on Image[3]



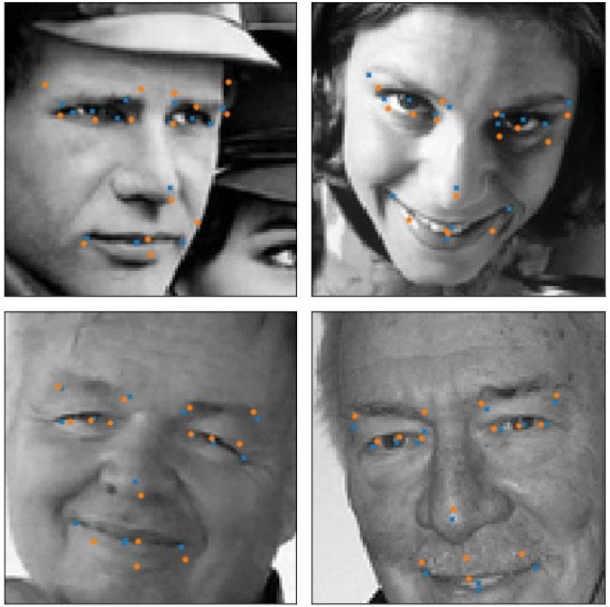
Applying 32 convolution filters (5by5) on Image[100]

Model 2

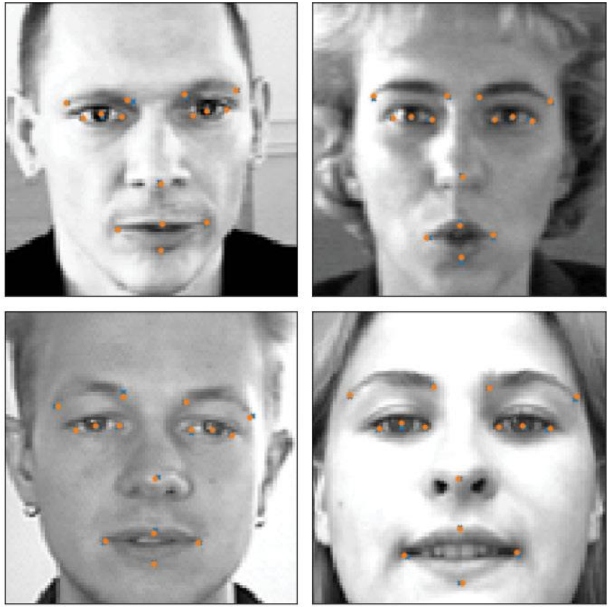
MSE Train: 0.00081224559
MSE Dev : 0.0007239112-Leaked

Worst 4 Labels :
mouth_center_bottom_lip_y
0.0016280584
left_eyebrow_outer_end_y
0.0012390522
right_eyebrow_outer_end_y
0.0012163228
nose_tip_y
0.0011951106

Best 4 Labels :
right_eye_inner_corner_y
0.00032786198
left_eye_inner_corner_y
0.00034015084
left_eye_center_y
0.00037150242
left_eye_inner_corner_x
0.00038773104



Worst 4 Images



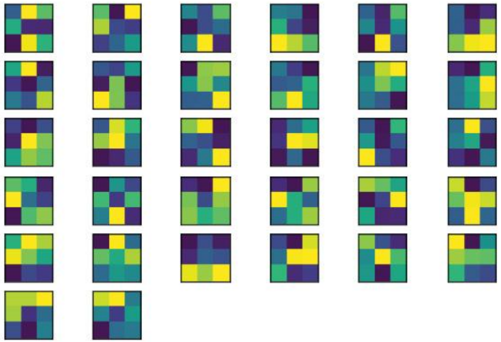
Best 4 Images

Model 3

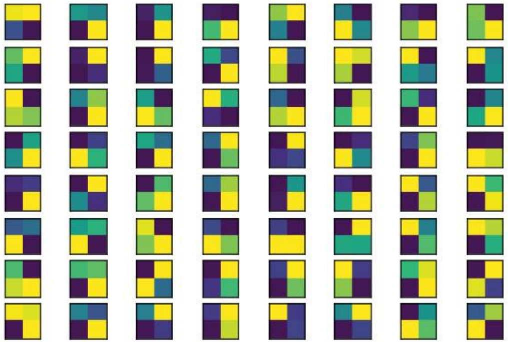
Model 3: Convolutional Neural Net
Input Nodes : 96 by 96
Output Nodes : 30
Max Epochs : 10000

Convolutional Filters : (32, 64,128)
Filter size : (3 by 3, 2 by 2, 2 by 2)
Pool size : 2 by 2
Hidden Layer 1 Nodes : 1000
Hidden Layer 2 Nodes : 1000

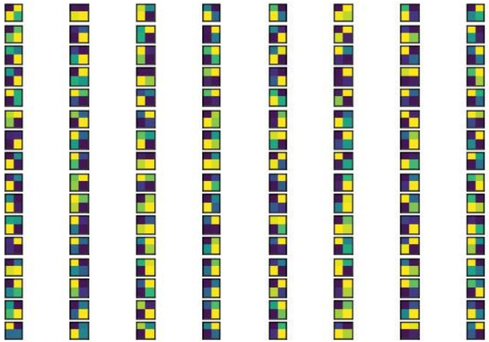
Dropout : (0.1,0.2,0.3,0.5)
update_learning rate
update_moment
flip half of images per batch of 128 images



32 Convolution Filters
3 by 3



64 Convolution Filters
2 by 2



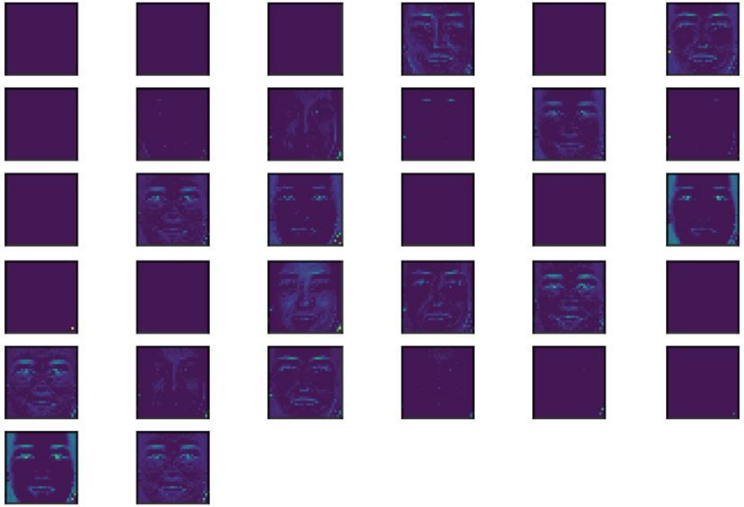
128 Convolution Filters
2 by 2

Model 3

Model 3: Convolutional Neural Net
Input Nodes : 96 by 96
Output Nodes : 30
Max Epochs : 10000

Convolutional Filters : (32, 64,128)
Filter size : (3 by 3, 2 by 2, 2 by 2)
Pool size : 2 by 2
Hidden Layer 1 Nodes : 1000
Hidden Layer 2 Nodes : 1000

Dropout : (0.1,0.2,0.3,0.5)
update_learning rate
update_moment
flip half of images per batch of 128 images



Applying 32 convolution filters (5by5) on Image[3]

Applying 32 convolution filters (5by5) on Image[100]

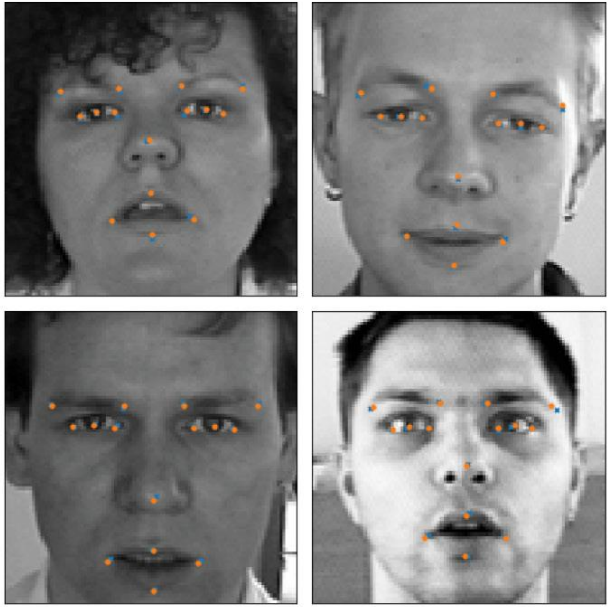
Model 3

MSE Train: 0.00085637317
MSE Dev : 0.00098517188

Worst 4 Labels :
mouth_center_bottom_lip_y
0.005132813
nose_tip_y
0.0041158358
mouth_right_corner_x
0.0034587171
nose_tip_x
0.0033688443

Best 4 Labels :
left_eye_inner_corner_y
0.00074552454
right_eye_inner_corner_y
0.00078256853
left_eye_center_y
0.00087295735
right_eye_inner_corner_x
0.00091137283

Worst 4 Images



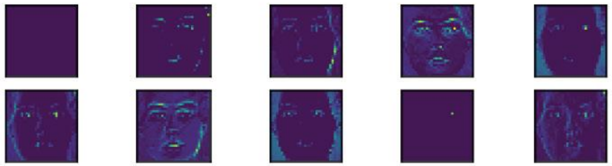
Best 4 Images

Model 4

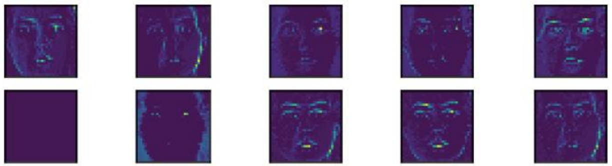
Model 4: Convolutional Neural Net * 6
Data: ~7000
Input Nodes : 96 by 96
Output Nodes : 30
Max Epochs : 10000

Convolutional Filters : (32, 64,128)
Filter size : (3 by 3, 2 by 2, 2 by 2)
Pool size : 2 by 2
Hidden Layer 1 Nodes : 1000
Hidden Layer 2 Nodes : 1000

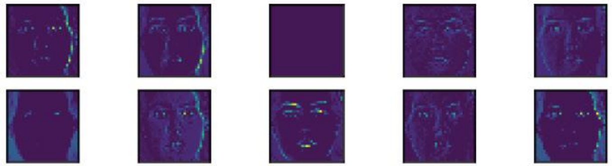
Dropout : (0.1,0.2,0.3,0.5)
update_learning rate
update_moment
flip half of images per batch of 128 images



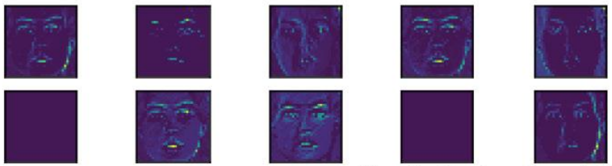
Eye Centers Specialist



Nose tip Specialist



Eye Corners Specialist



Mouth Centers Specialist

Eyebrows Specialist

Mouth Corners Specialist

Model 4

MSE Train: 0.00055848644
MSE Dev : 0.00084981503

Worst 4 Labels :
nose_tip_y
0.0029636011
left_eyebrow_outer_end_y
0.0020160379
mouth_center_bottom_lip_y
0.0019226527
right_eyebrow_outer_end_y
0.0015255155

Best 4 Labels :
left_eye_inner_corner_y
0.00026280512
right_eye_inner_corner_y
0.00026981242
left_eye_center_y
0.00028459064
right_eye_inner_corner_x
0.00035192835

Worst 4 Images

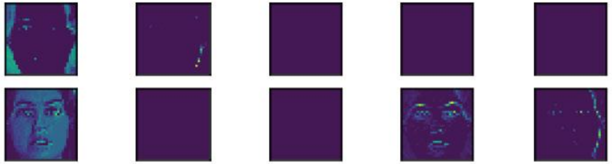
Best 4 Images

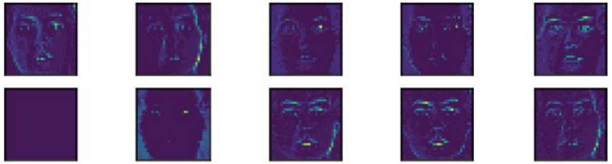
Model 5

Model 5: Convolutional Neural Net * 6
Data: ~28000
Input Nodes : 96 by 96
Output Nodes : 30
Max Epochs : 10000

Convolutional Filters : (32, 64,128)
Filter size : (3 by 3, 2 by 2, 2 by 2)
Pool size : 2 by 2
Hidden Layer 1 Nodes : 1000
Hidden Layer 2 Nodes : 1000

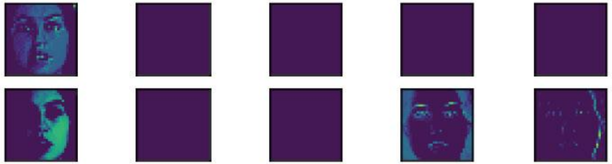
Dropout : (0.1,0.2,0.3,0.5)
update_learning rate
update_moment
flip half of images per batch of 128 images

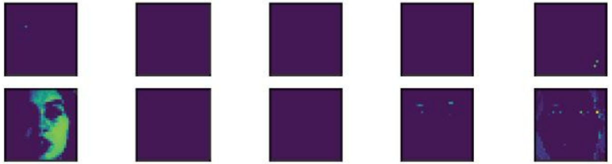

Eye Centers Specialist


Nose tip Specialist


Eye Corners Specialist


Mouth Centers Specialist


Eyebrows Specialist


Mouth Corners Specialist

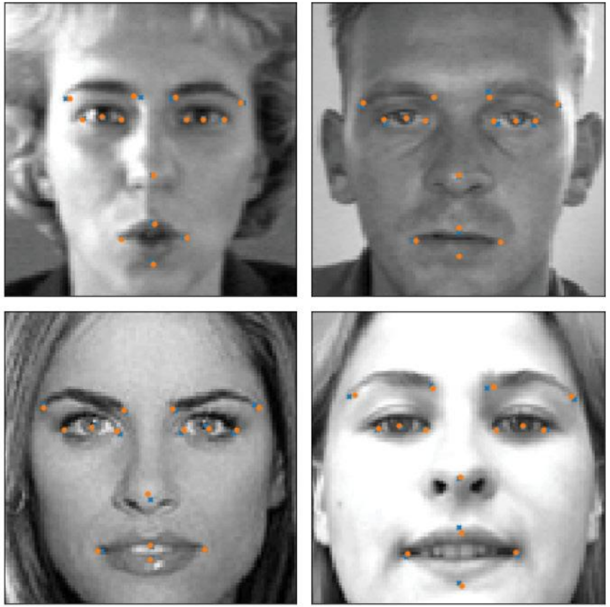
Model 5

MSE Train: 0.0010119774
MSE Dev : 0.0010905139

Worst 4 Labels :
nose_tip_y
0.0029636011
left_eyebrow_outer_end_y
0.0020455536
mouth_center_bottom_lip_y
0.0019226527
right_eyebrow_outer_end_y
0.0017795484

Best 4 Labels :
right_eye_center_y
0.00033021756
left_eye_center_y
0.00047810798
left_eye_center_x
0.0005514645
right_eye_inner_corner_y
0.00058127748

Worst 4 Images



Best 4 Images

Future Refinement

Lessons Learned

- How to split work on local machine vs AWS for cost savings
- CNN architecture
- Save parameters on each model built to hedge against AWS crashes
- Consider run time and efficiently applying transformations for all data augmentations
- The importance of a balanced train|dev split that is representative of the data

Next Steps

- Try Tensorflow|Keras implementation instead of lasagne to increase speed (?)
- Remove fingerprint Motif data transformation
- Align tilted images using linear techniques
- Build an additional model for the cases that have 5 facial features labeled instead of specialist models for each of 6 facial features
- Increase sample size on development dataset for evaluation of data augmentations
- Test accuracy improvements for handling outliers

Questions?