HEAD POSE DETECTION

CS584 - MACHINE LEARNING

Bharath Bandaru
Himamshu Lakkaraju

Introduction

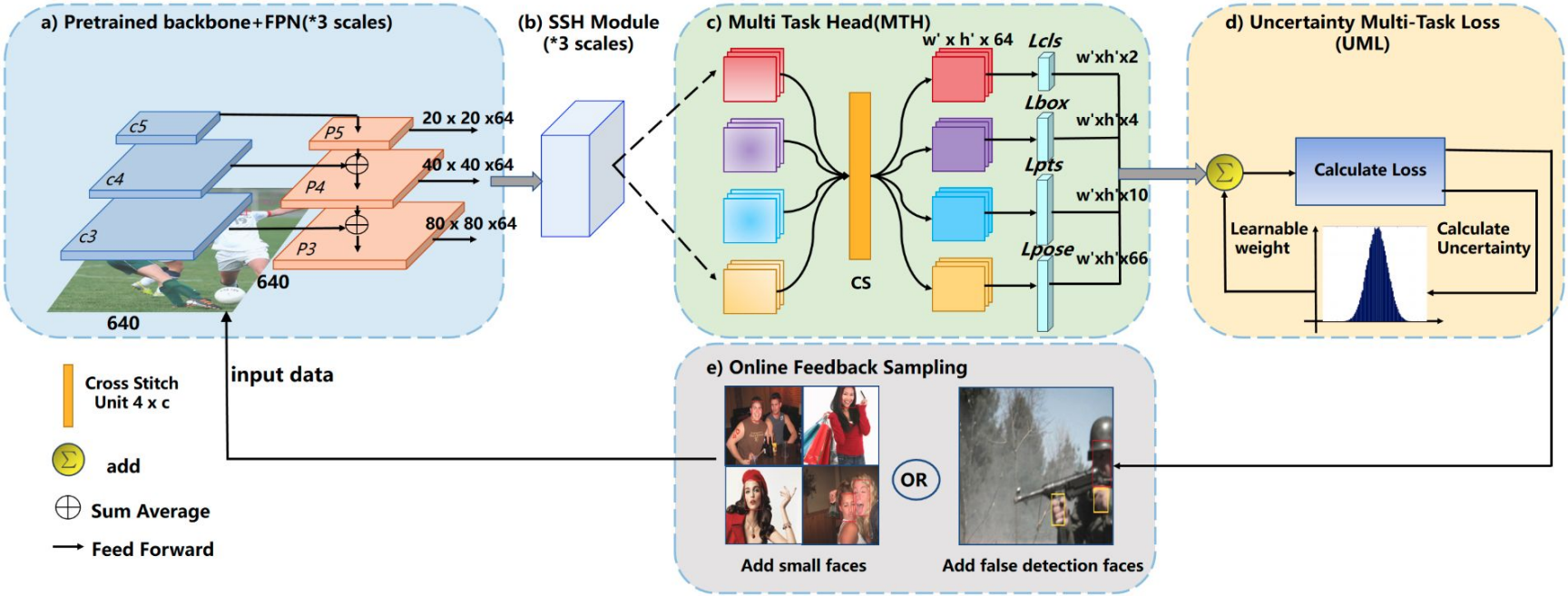
- **Motivation**
- **Use Cases -** HCI, Safety Standards etc
- **Current State -** Head Pose Detection

PAPERS

- MOS: A Low Latency and Lightweight Framework for Face Detection, Landmark Localization, and Head Pose Estimation

- Fine-Grained Head Pose Estimation Without Keypoints

MOS


a) Pretrained backbone+FPN(*3 scales)
c5
c4
c3
P5
P4
P3
20 x 20 x64
40 x 40 x64
80 x 80 x64
640
640
(b) SSH Module
(*3 scales)
c) Multi Task Head(MTH)
w' x h' x 64
Lcls
w'xh'x2
Lbox
w'xh'x4
Lpts
w'xh'x10
Lpose
w'xh'x66
CS
d) Uncertainty Multi-Task Loss
(UML)
Calculate Loss
Learnable
weight
Calculate
Uncertainty
input data
Cross Stitch
Unit 4 x c
add
Sum Average
Feed Forward
e) Online Feedback Sampling
OR
Add small faces
Add false detection faces

HOPENET

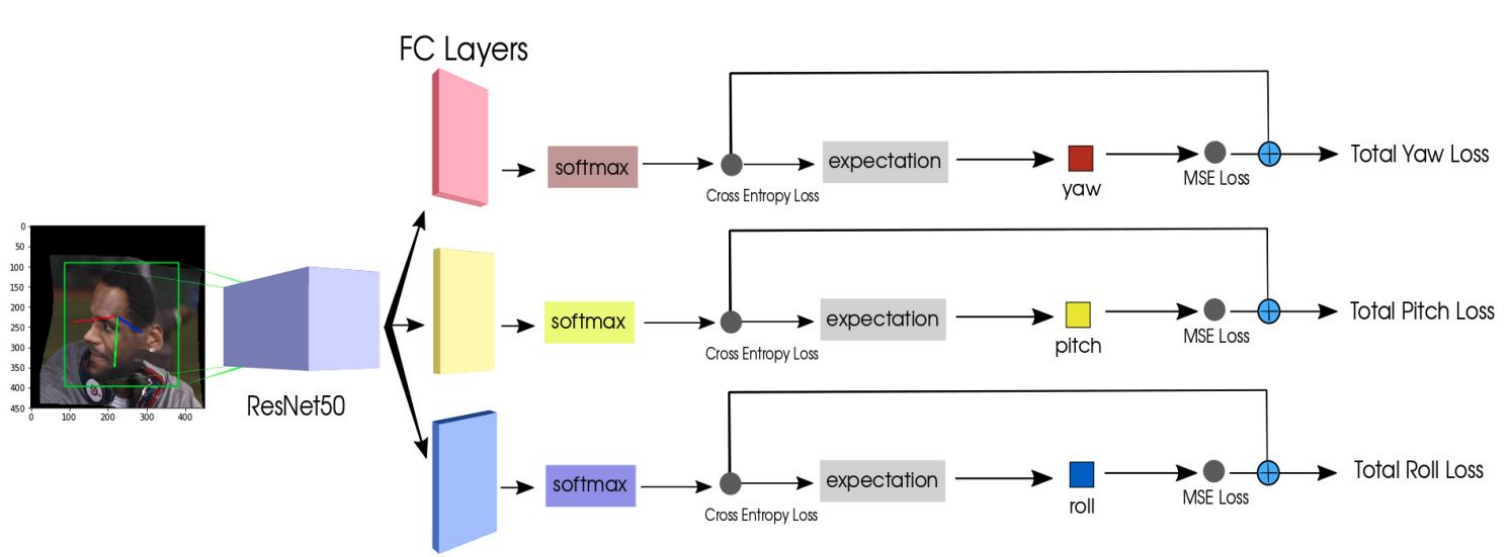


Figure 2. ResNet50 architecture with combined Mean Squared Error and Cross Entropy Losses.

Challenges with Head pose detection

- Amount of training data required
- Number of epochs
- Availability of computational resources

Proposed Solution

- Simple CNN for pose estimation.
- Use Pre-trained model like ResNet to perform Head pose detection
- Use similar network/Loss calculation as HOPENET[2] along with dlib to check the performance.

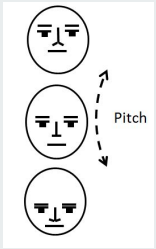
[2] Referring from second paper

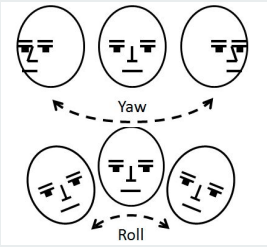
Dataset - 300W_LP
(AFW)

- **Training - 4434**
- **Validation - 554**
- **Testing - 555**

Labels:

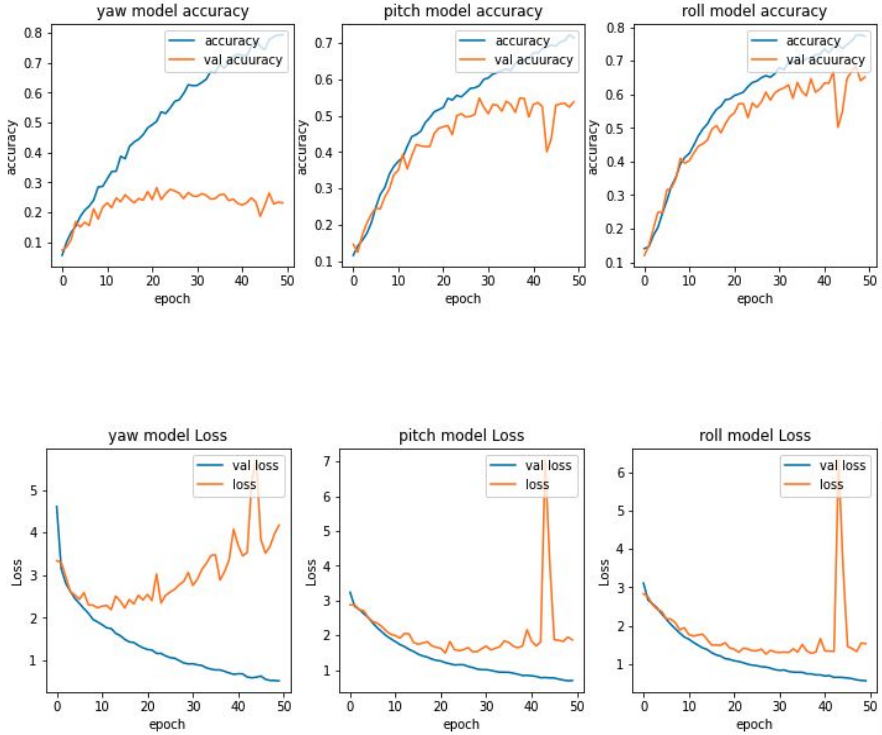
- **Face x,y coordinates**
- **yaw - left, right**
- **pitch - up, down**
- **roll - rotate**



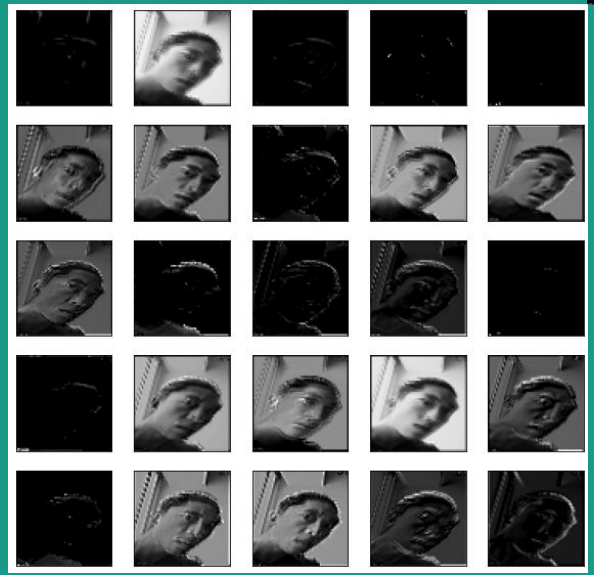


Start

- Basic CNN layers
- Trained with direct images (without cropping)
- Classification
- Loss function - `categorical_crossentropy`
- Observation - Accuracy of *yaw* is under 0.3.



CNN Model

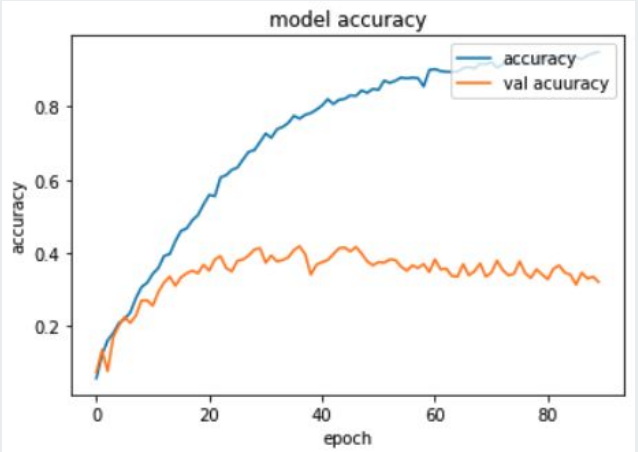


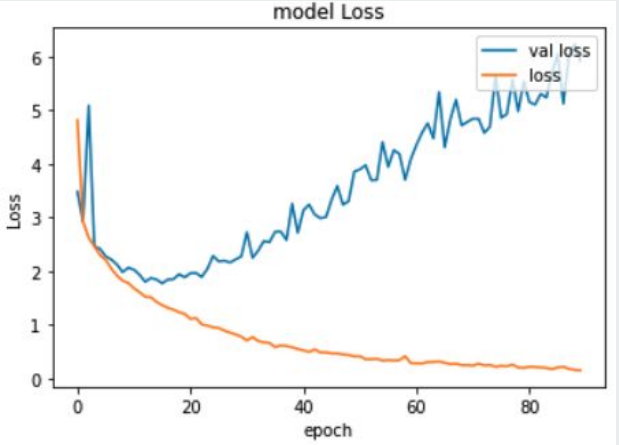
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 240, 240, 3)]	0	[]
conv2d (Conv2D)	(None, 60, 60, 64)	23296	['input_1[0][0]']
max_pooling2d (MaxPooling2D)	(None, 29, 29, 64)	0	['conv2d[0][0]']
conv2d_1 (Conv2D)	(None, 29, 29, 192)	307392	['max_pooling2d[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 192)	0	['conv2d_1[0][0]']
batch_normalization (BatchNorm alization)	(None, 14, 14, 192)	768	['max_pooling2d_1[0][0]']
conv2d_2 (Conv2D)	(None, 14, 14, 384)	663936	['batch_normalization[0][0]']
conv2d_3 (Conv2D)	(None, 14, 14, 256)	884992	['conv2d_2[0][0]']
batch_normalization_1 (BatchNo rmalization)	(None, 14, 14, 256)	1024	['conv2d_3[0][0]']
conv2d_4 (Conv2D)	(None, 14, 14, 256)	590080	['batch_normalization_1[0][0]']
conv2d_5 (Conv2D)	(None, 14, 14, 512)	1180160	['conv2d_4[0][0]']
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 512)	0	['conv2d_5[0][0]']
batch_normalization_2 (BatchNo rmalization)	(None, 6, 6, 512)	2048	['max_pooling2d_2[0][0]']
flatten (Flatten)	(None, 18432)	0	['batch_normalization_2[0][0]']
dropout (Dropout)	(None, 18432)	0	['flatten[0][0]']
dense (Dense)	(None, 4096)	75501568	['dropout[0][0]']
yaw (Dense)	(None, 66)	270402	['dense[0][0]']
pitch (Dense)	(None, 66)	270402	['dense[0][0]']
roll (Dense)	(None, 66)	270402	['dense[0][0]']

Total params: 79,966,470
Trainable params: 79,964,550
Non-trainable params: 1,920

Model considering *yaw* only

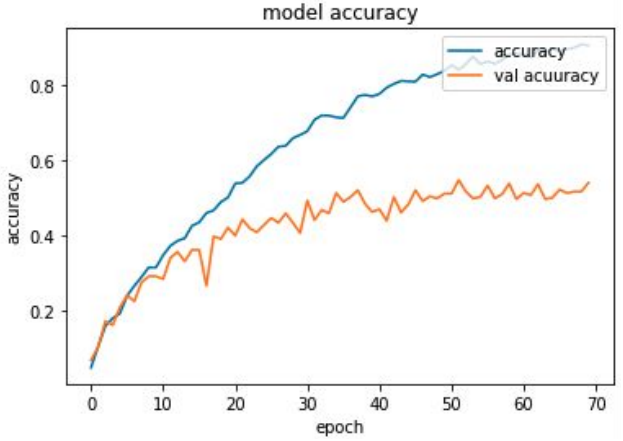
- Basic CNN layers
- Trained with direct images (without cropping)
- Classification
- Loss function - `categorical_crossentropy`
- Observation - Accuracy is sub 0.4.

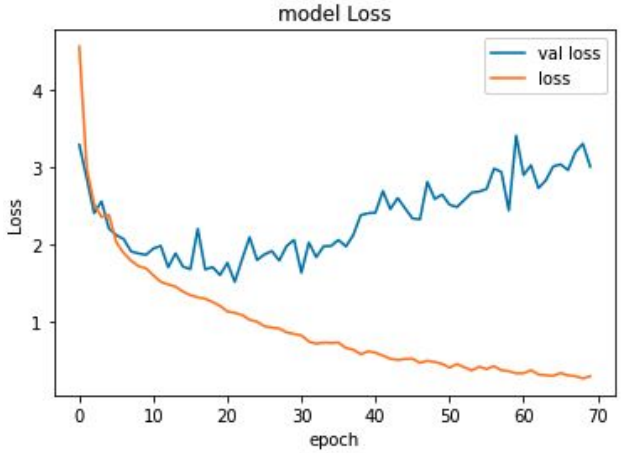




Model with cropped images

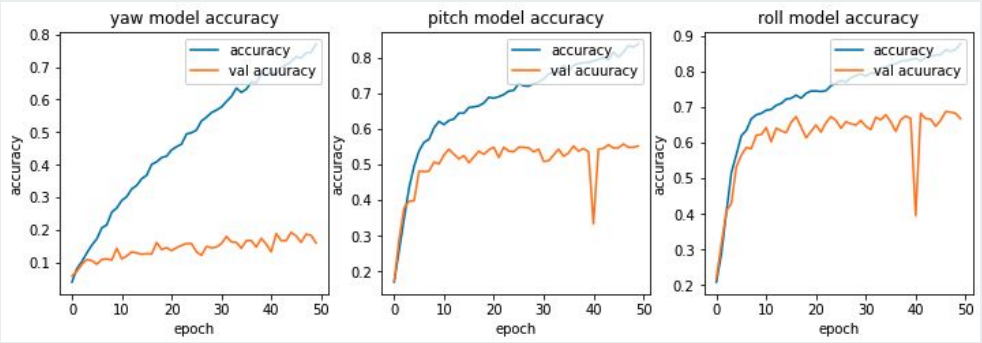
- Basic CNN layers
- Trained with cropped images and previous model (yaw only)
- Observation - Accuracy increased to 0.54

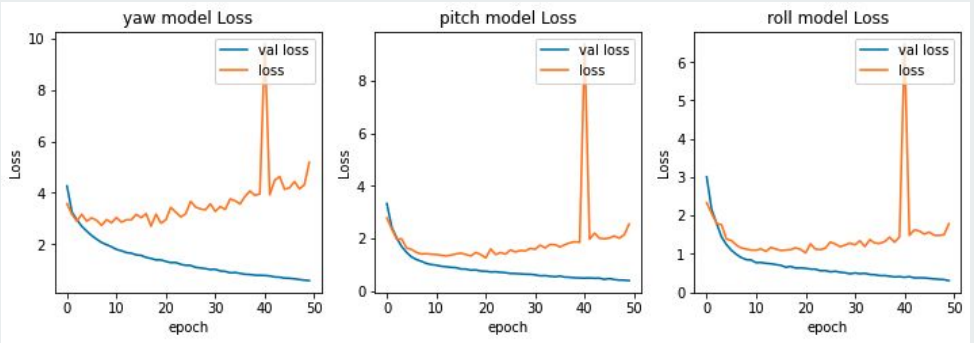




Combining y, p, r

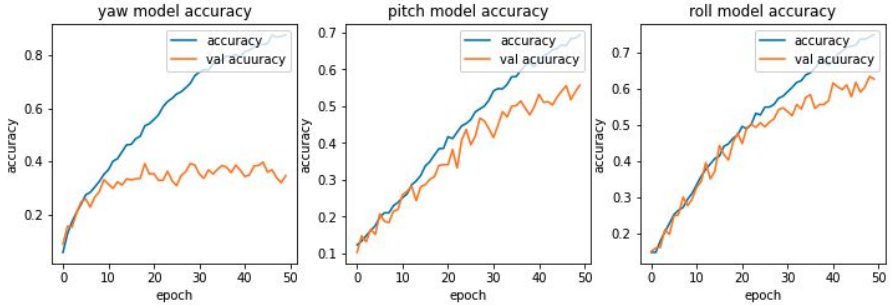
- Basic CNN layers
- Trained with cropped images
- Loss function - `categorical_crossentropy`
- Observation - no improvement in *yaw* accuracy

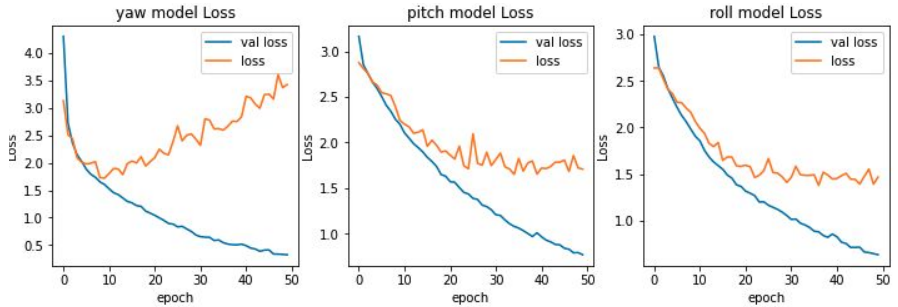




Final Basic Model

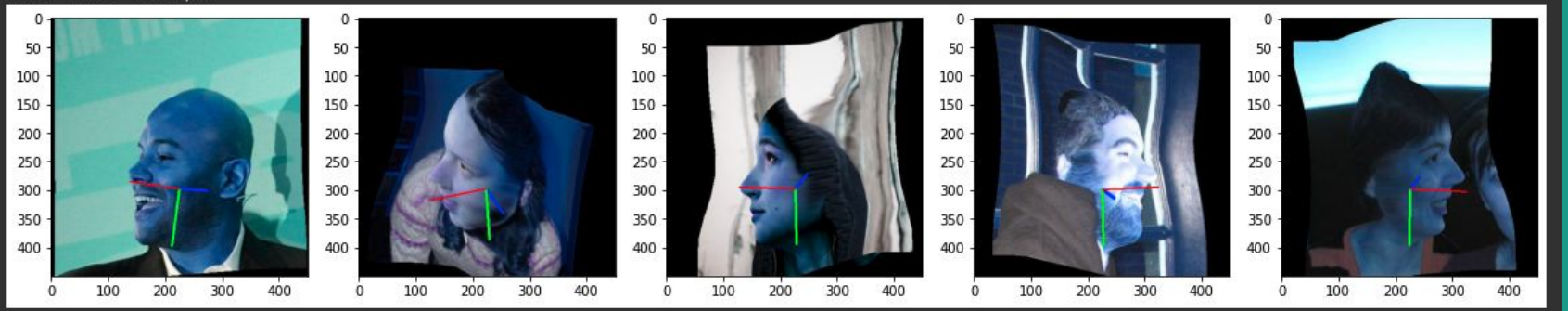
- Basic CNN layers
- Trained with cropped images
- For the previous model we updated loss_weights to (1, 0.3, 0.3)
- Observations: Comparatively the model performed better with yaw accuracy around 0.4





Final result of CNN

```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 7.63 µs
```

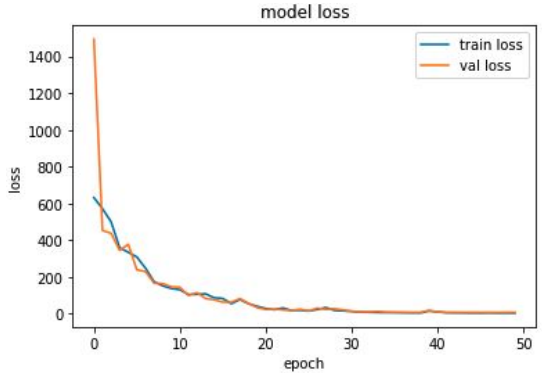


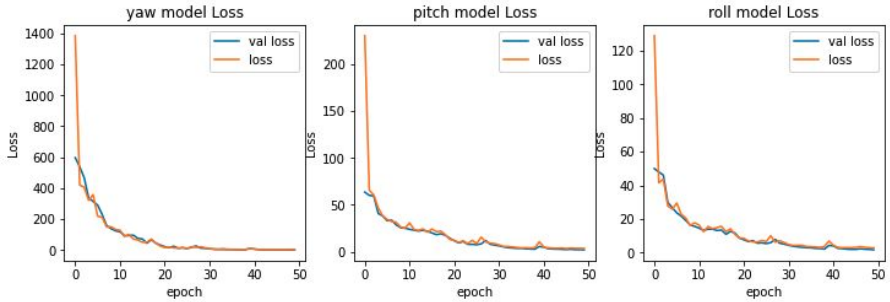
Updated Model

- Basic CNN layers
- Trained with cropped images
- Loss Function -

$$\mathcal{L} = H(y, \hat{y}) + \alpha \cdot MSE(y, \hat{y})$$

- Observations: Nothing much changed (comparing MAE)



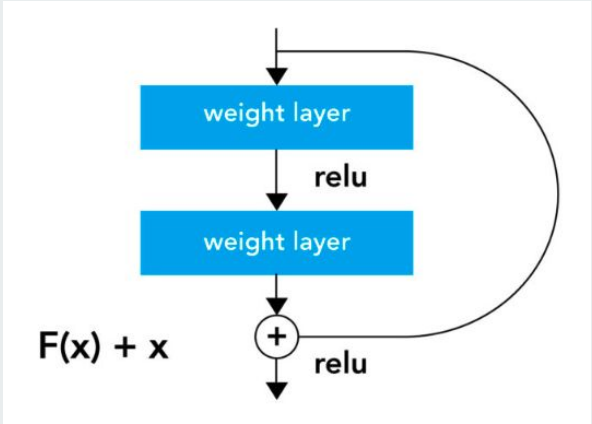


Added Residual net

- Loss Function -

$$\mathcal{L} = H(y, \hat{y}) + \alpha \cdot MSE(y, \hat{y})$$

- Observation - No improvement in MAE



	3 Resnet blocks	Resnet-50
Yaw	49.08	48.71
Pitch	10.82	8.79
Roll	8.03	4.73
Average	22.648	20.74

Demo

THANK YOU