Vision Aided Drone Localization in GPS-denied Indoor Corridor Environments

Submitted by: Shahzad Ahmad (216CS1134)
Under the Supervision Of
Dr. Pankaj K.Sa

Department of Computer Science and Engineering
National Institute of Technology, Rourkela

Table of Contents

Problem Statement

Localizing drone in indoor scenario where GPS signal are absent with computer vision technique.

Problem Motivation

- Unavilabililty of GPS
- Existing methods require much hardwares and are expensive with respect to computation.
- No state of the art with respect to learning algorithm.

Goal

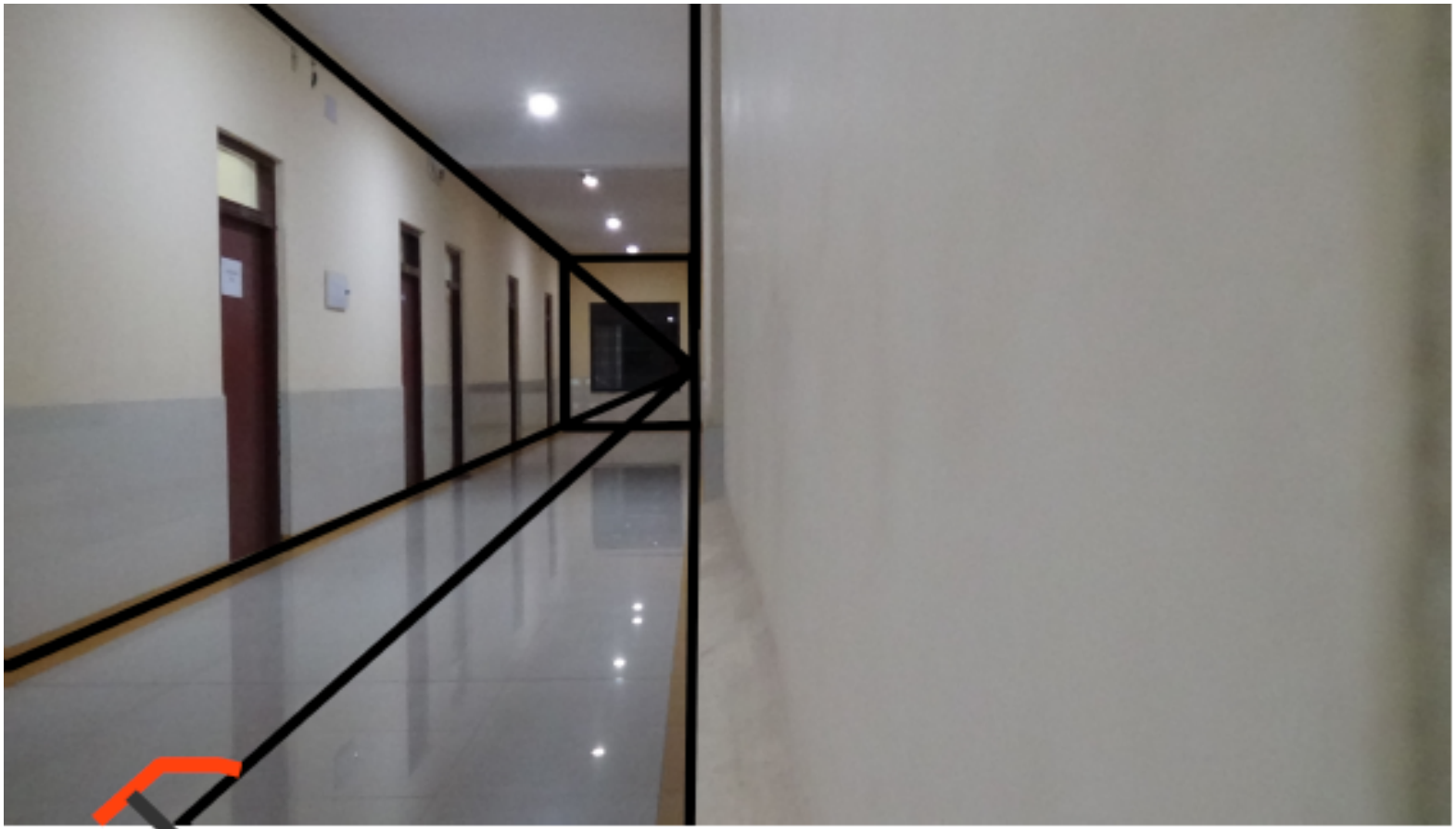
- Enable the drone to localize its position inside indoor environment.
- Remove sophisticated sensor and hardware dependencies.
- To aid drone to navigate autonomously in flat corridor in indoor environment.

Navigation Algorithm

- Proposed algorithm help us to find position of drone in corridor.
- Find position of drone with help bisector of floor.

Navigation Algorithm Cont.

Navigation Algorithm Cont.



Angle > 90

Navigation Algorithm Cont.

Navigation Algorithm Cont.

Navigation Algorithm Cont.

Navigation Algorithm Cont.

Algorithm 1: Algorithm for flight command

Input: image,image.height,image.width

Result: move command for drone

```
1  θ=Algorithm2(image);
2  if θ < 90 then
3  |   right shift drone ;
4  else if θ > 90 then
5  |   left shift drone ;
6  else
7  |   d=Algorithm3(image);
8  if d < image.width/2 then
9  |   rotate anti-clock wise ;
10 else if d > image.width/2 then
11 |   rotate clock wise;
12 else
13 |   move forward ;
```

Navigation Algorithm Cont.

Algorithm 2: Algorithm for finding angle

Input: image

Result: Angle between bisector of plane and horizental axis of image

1 Normalize pixal between 0 to 1;
2 RGB to BGR conversion;
3 Normalize with mean and standard deviation of Image Net dataset.;
4 θ = trained_weights(image);
5 **return** θ;

Navigation Algorithm Cont.

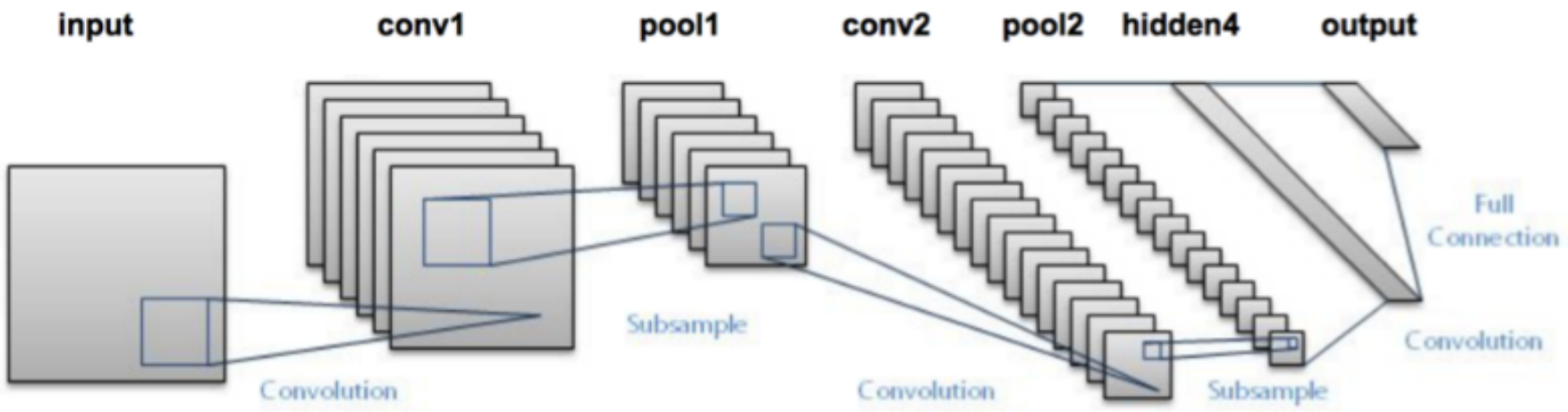
Algorithm 3: Algorithm for finding intersection point

Input: image

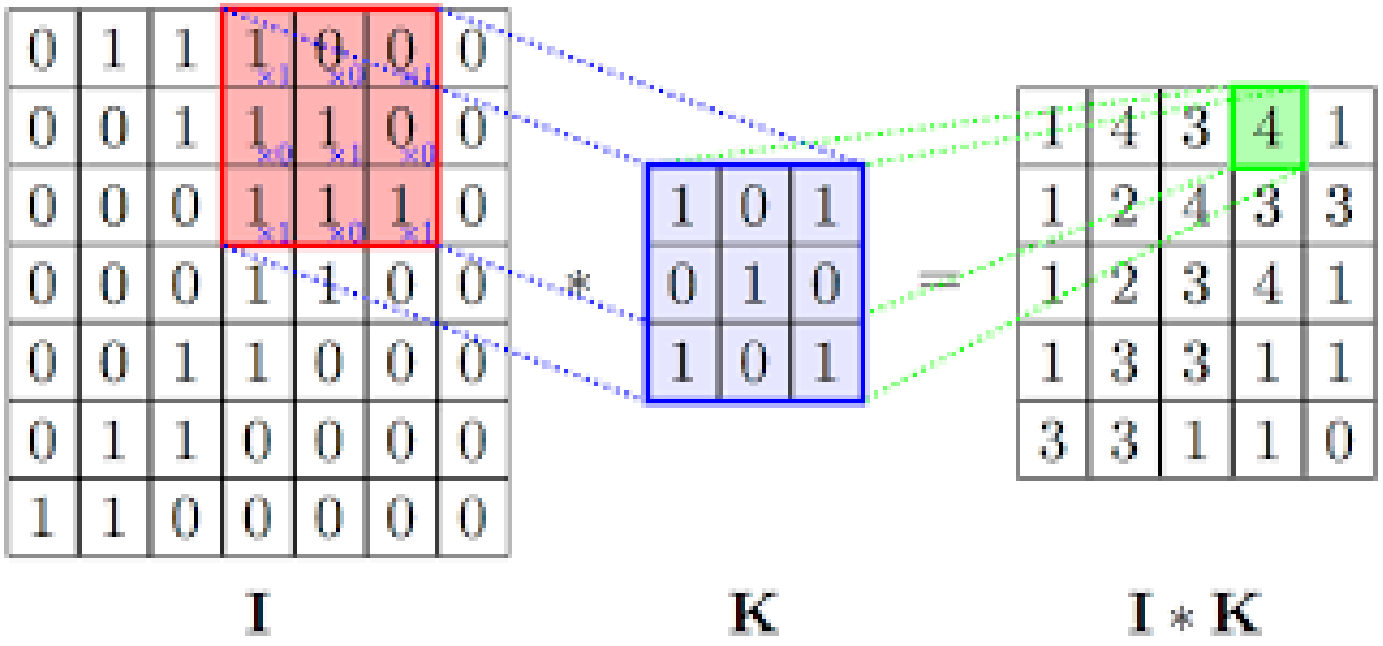
Result: Intersection point of bisector of plane and horizental axis of image

1 Normalize pixel between 0 to 1;
2 RGB to BGR conversion;
3 Normalize with mean and standard deviation of Image Net dataset.;
4 d = trained_weights(image);
5 **return** d;

Deep Learning Architecture[1, 2]

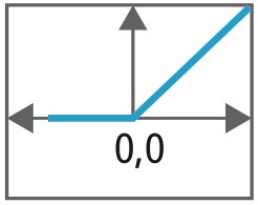


Deep Learning Architecture cont..



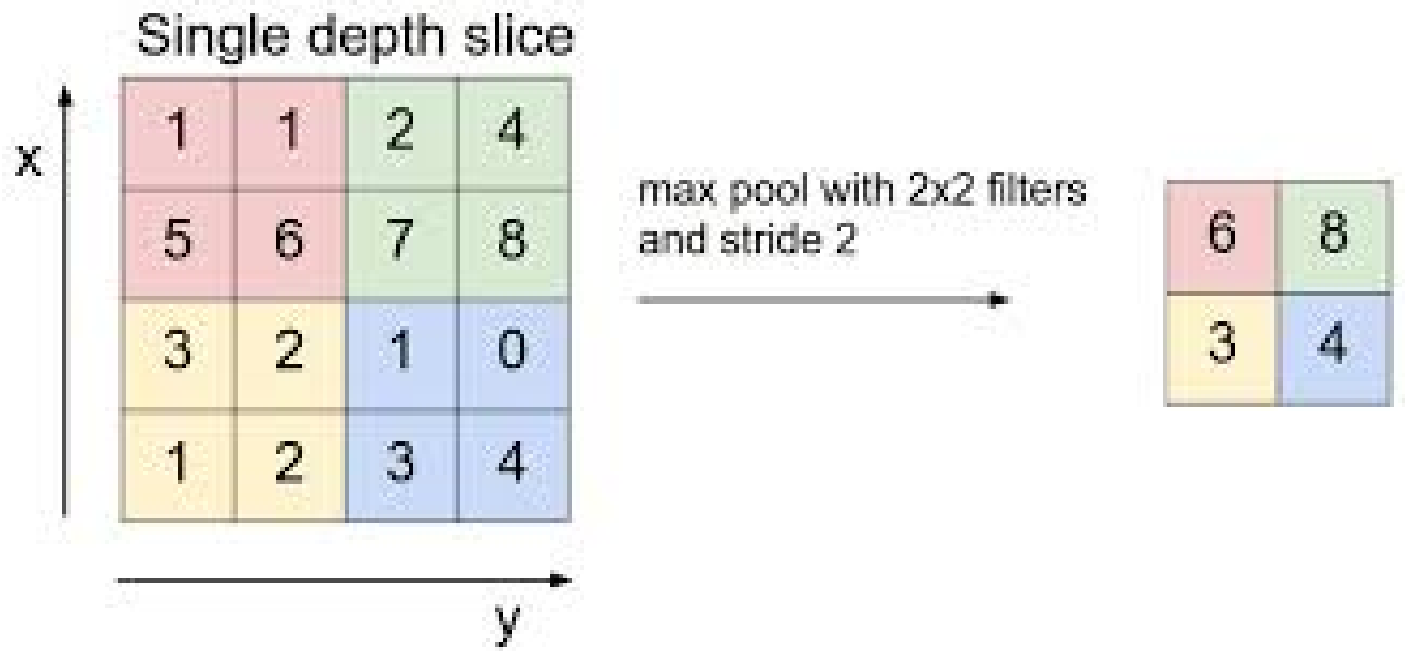
Transfer Function

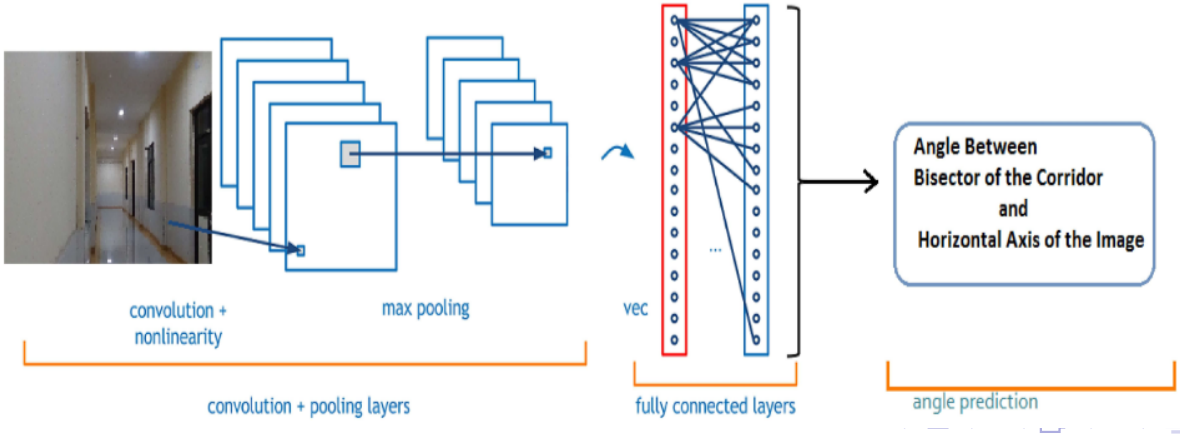
15	20	-10	35
18	-110	25	100
20	-15	25	-10
101	75	18	23



15	20	0	35
18	0	25	100
20	0	25	0
101	75	18	23

Deep Learning Architecture cont..





Why use Deep Learning

- To learn intrinsic features of the image to guide navigation.
- To learn depth information from simple RGB image.

Used Architecture DenseNet-161[4]

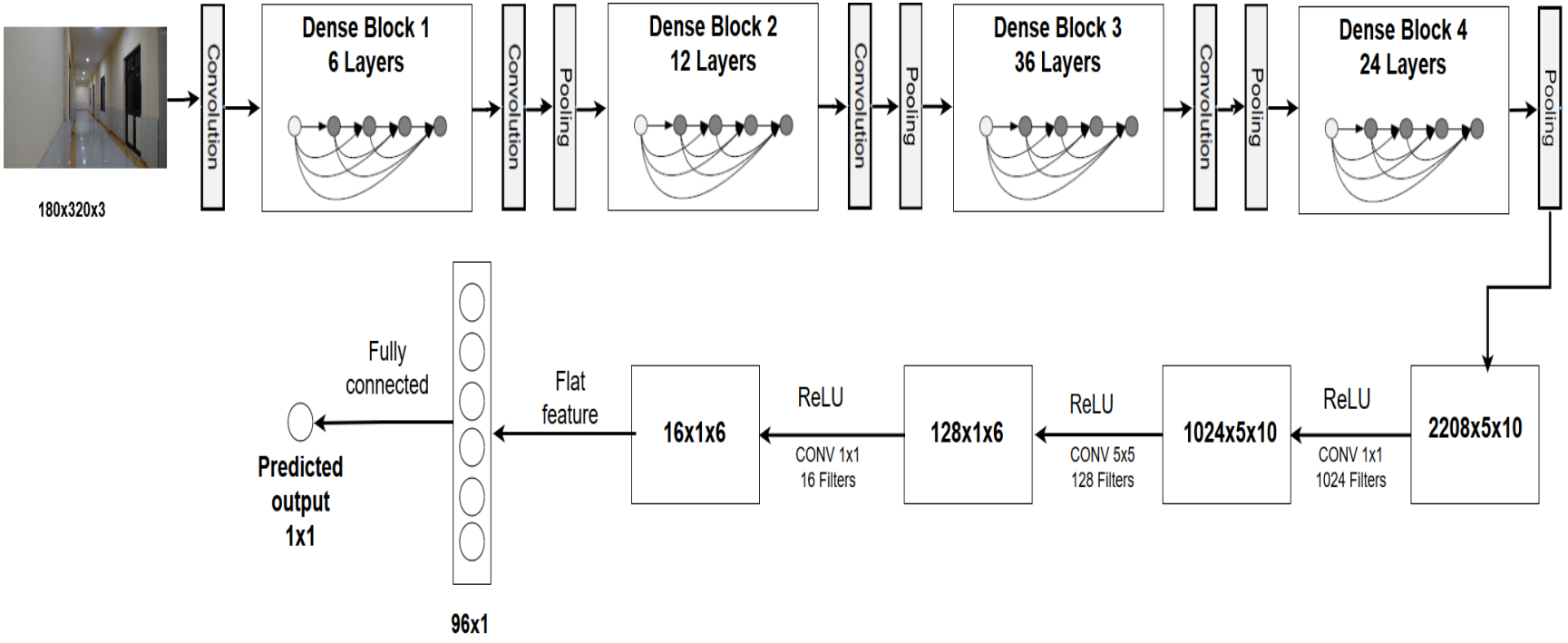


Figure 1: Using DenseNet-161 with Transfer Learning

Architecture (Cont.)

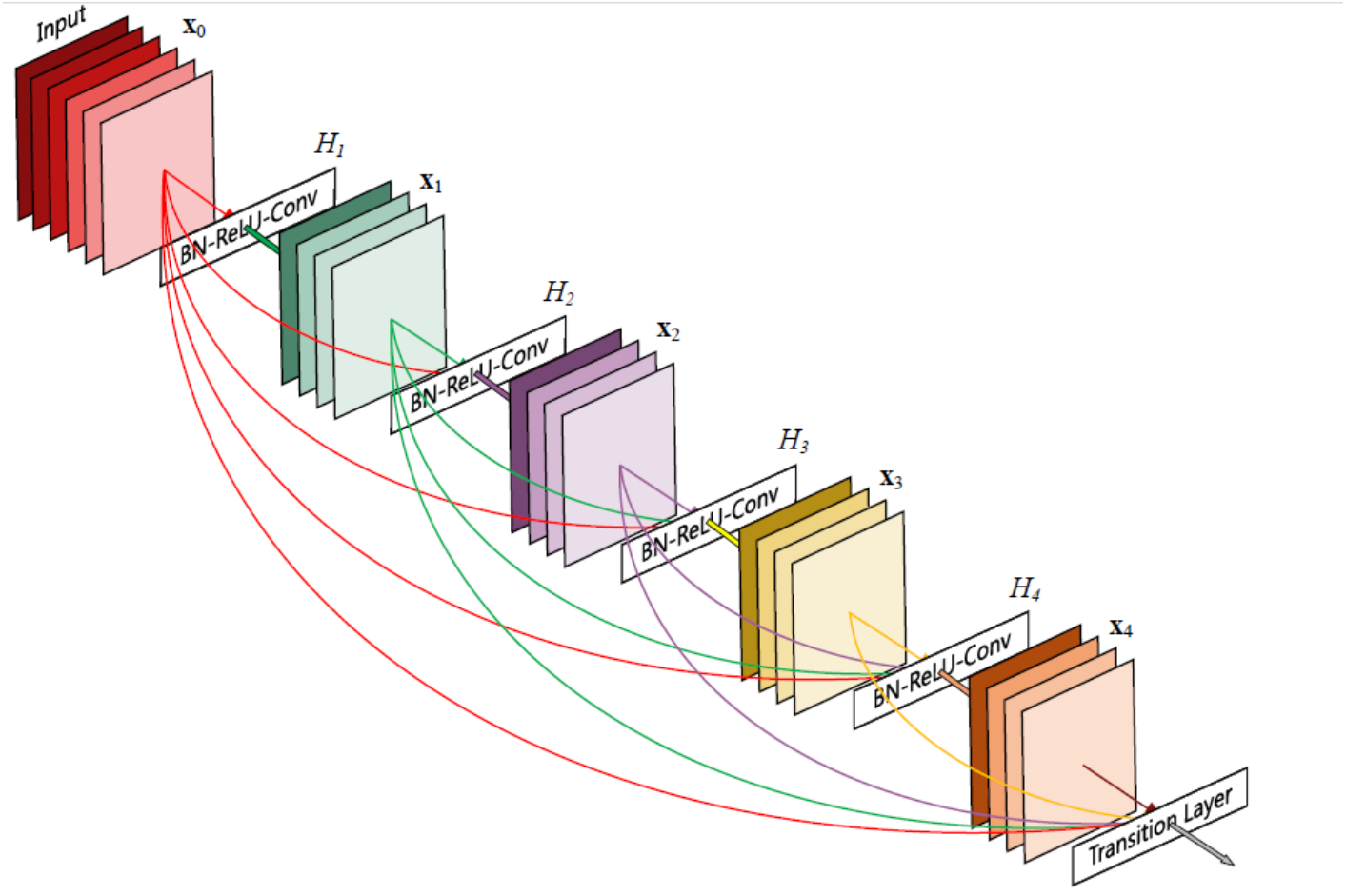


Figure 2: Block of DenseNET

System Setup

- The proposed algorithm is validated using a Parrot AR Drone power edition quadcopter having one front facing camera, one down facing camera.
- Video feed is recorded at 30 fps
- The algorithm is executed on ground station having Ubuntu 14.04 OS,32 GB RAM, and 2.80 GHz XEON processor and NVIDIA GT 1080 11GB memory.
- Pytorch library package is use for deep learning implementation.

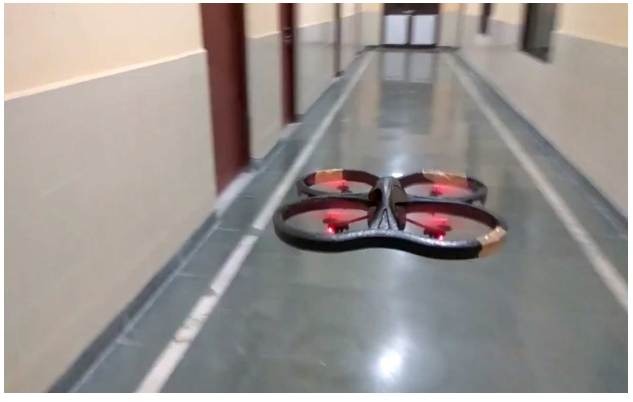


Figure 3: Parrot AR Drone

Data Set Creation (Raw Data)

(a) At Left aligned to Left

(b) At Left aligned to Center

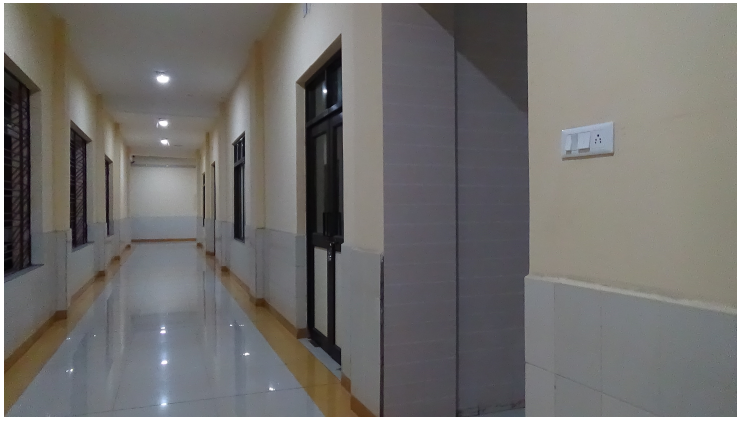
(c) At Left aligned to Right

Figure 4: various instances of indoor corridors

Data Set Creation (Raw Data) (Cont.)

(a) At Center aligned to Left

(b) At Center aligned to Center

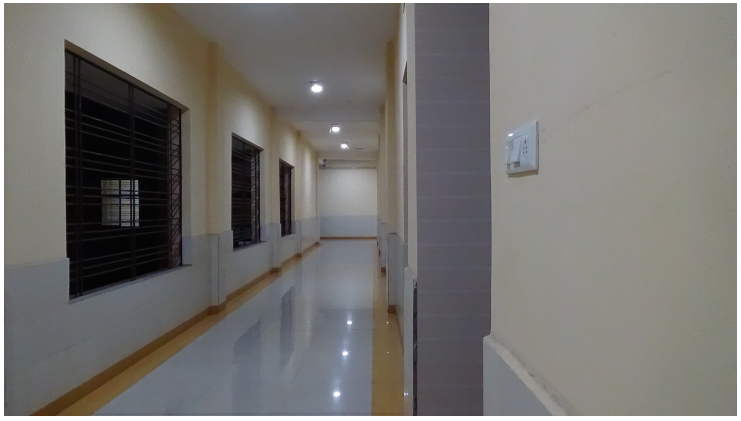


(c) At Center aligned to Right

Figure 5: various instances of indoor corridors

Data Set Creation (Raw Data) (Cont.)

(a) At Right aligned to Left



(b) At Right aligned to Center

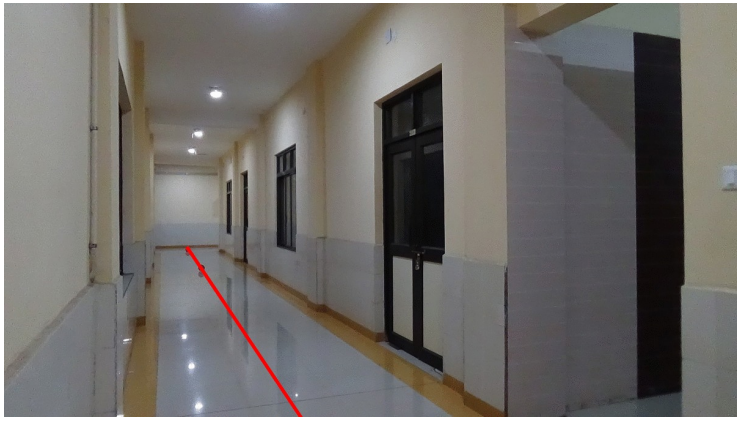
(c) At Right aligned to Right

Figure 6: various instances of indoor corridors

Data Set Creation (Labelled Data)

(a) At Left aligned to Left

(b) At Left aligned to Center



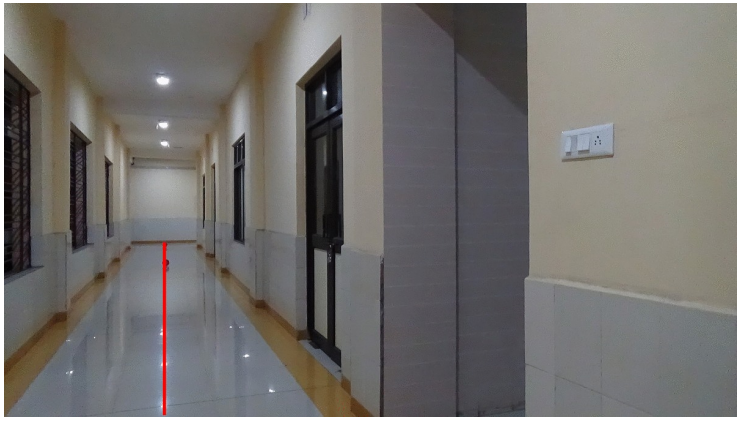
(c) At Left aligned to Right

Figure 7: various instances of indoor corridors

Data Set Creation (Labelled Data) (Cont.)

(a) At Center aligned to Left

(b) At Center aligned to Center



(c) At Center aligned to Right

Figure 8: various instances of indoor corridors

Data Set Creation (Labelled Data) (Cont.)

(a) At Right aligned to Left

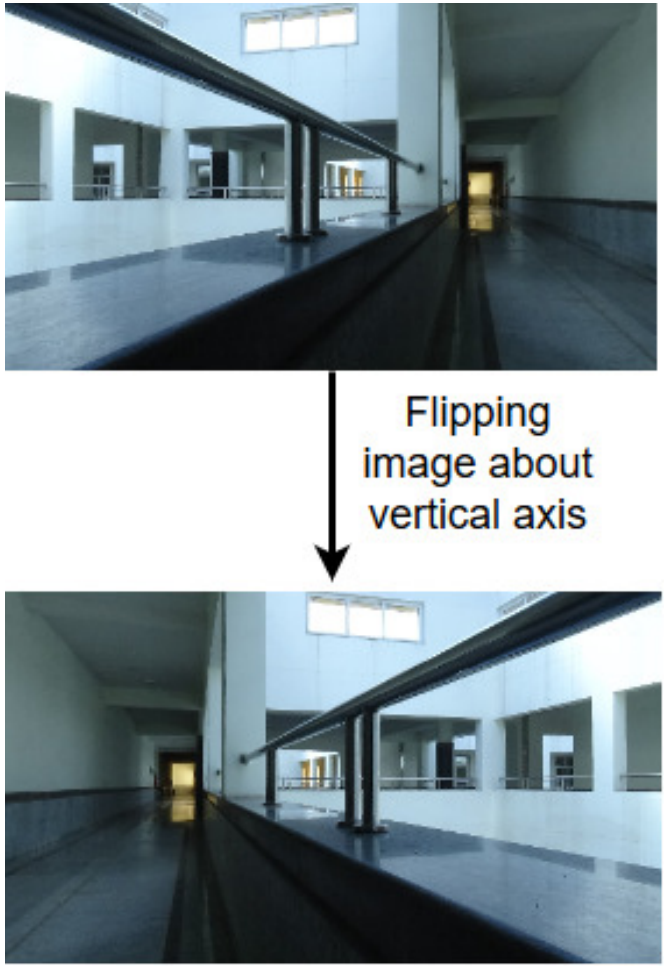
(b) At Right aligned to Center

(c) At Right aligned to Right

Figure 9: various instances of indoor corridors

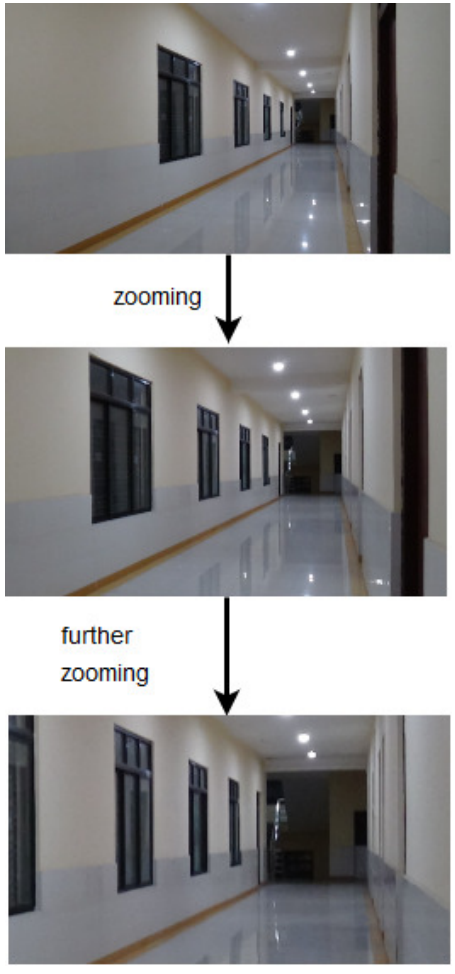
Data Augmentation Technique

- Image Flipping



Data Augmentation Technique (Cont.)

- Image Zooming



Experiment:Finding angle between bisector and horizontal axis

- We use 35000 images for training and 600 images for test
- We use three loss functions for training
- First we use Euclidean Loss function for training of model

$$Euclidean\ Loss(\hat{y}, y) = \frac{1}{2n}\sum_{t=1}^{n}(\hat{y}_t - y_t)^2$$

Where $\hat{y}$ and y are predicted and ground-truth value vector of batch respectively.

Experiment:Finding angle between bisector and horizontal axis

- Second we use berHu Loss[5, 6] function for training of model

$$berHu(\hat{y}, y) = \begin{cases} L_1(\hat{y}, y), & \text{if } |\hat{y} - y| \in [-t, t] \\ \frac{1}{2t}(L_2(\hat{y}, y) + t^2), & \text{otherwise} \end{cases}$$

where

$$L_1(\hat{y}, y) = \sum_{t=1}^{n} |\hat{y}_t - y_t|$$

$$L_2(\hat{y}, y) = \sum_{t=1}^{n} (\hat{y}_t - y_t)^2$$

$$t = 0.2max(|\hat{y}_t - y_t|)$$

Experiment:Finding angle between bisector and horizontal axis

- Third loss we use as Mean Absolute loss function to learn our model

$$Mean\ Absolute\ Loss(\hat{y}, y) = \frac{1}{n}\sum_{t=1}^{n}|\hat{y}_t - y_t|$$

Experiment:Convergence Graph

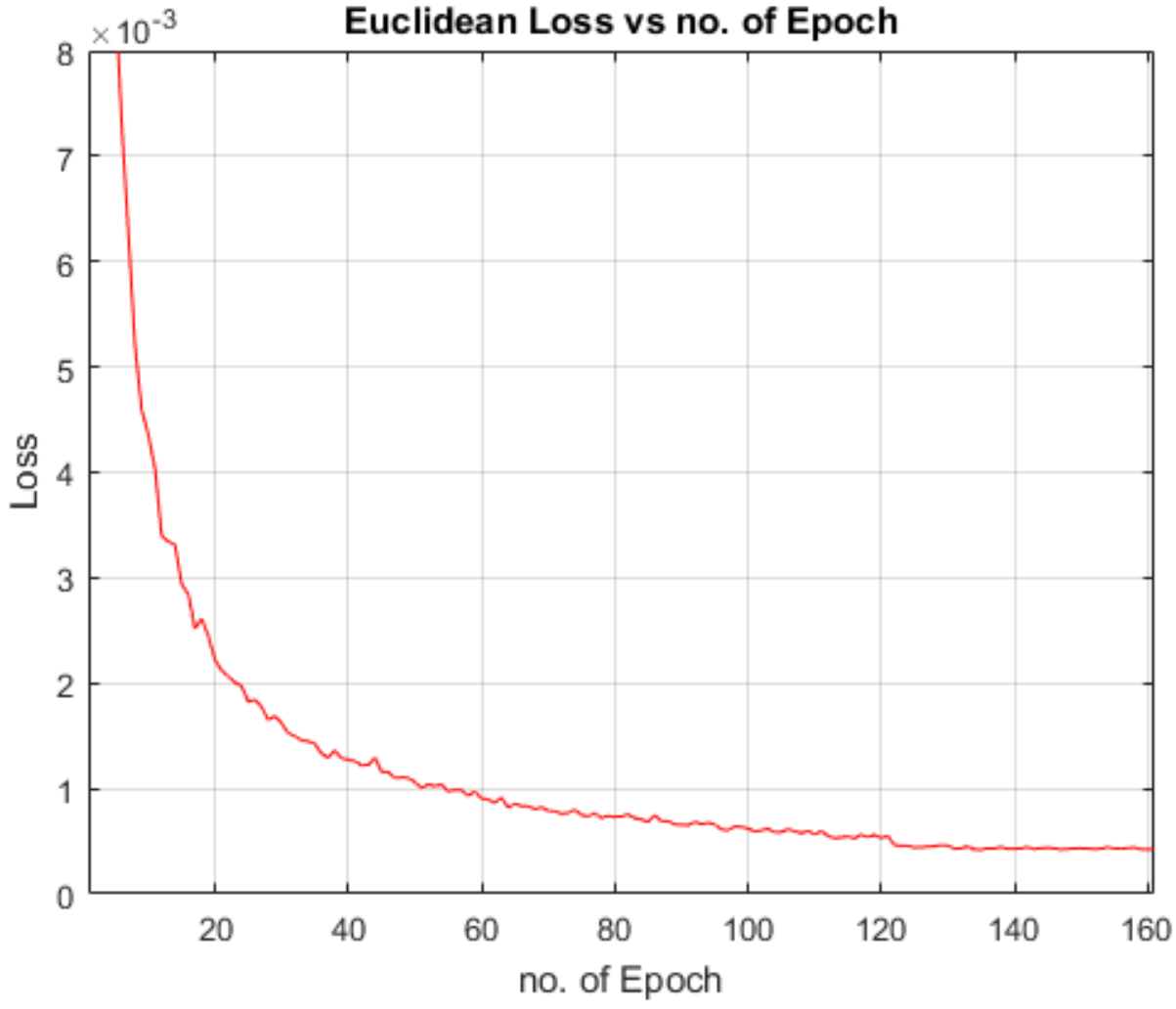


Figure 10: Euclidean Loss vs No. of Epoch

Experiment:Convergence Graph

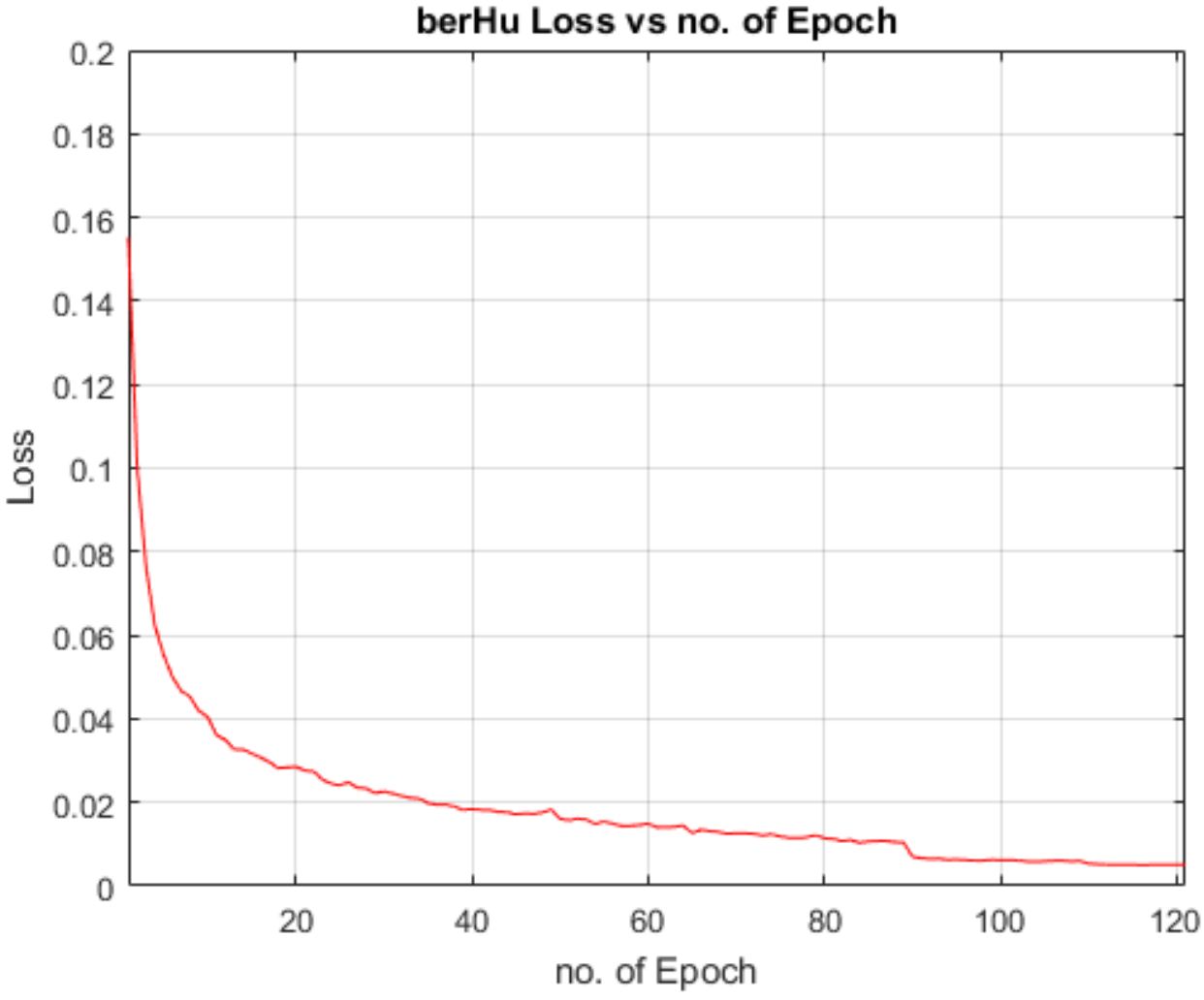


Figure 11: berHu Loss vs No. of Epoch

Experiment:Convergence Graph

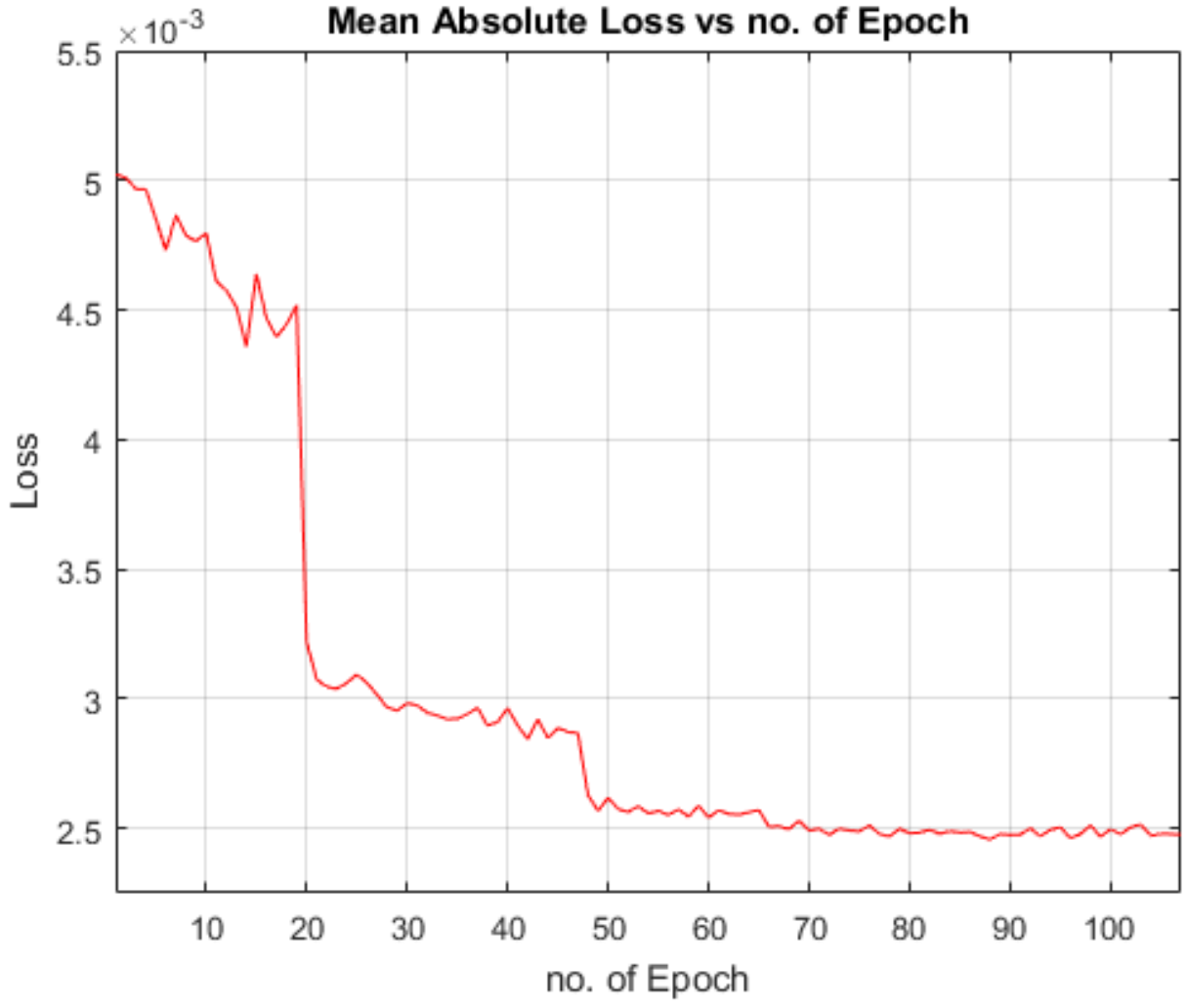


Figure 12: Mean Absolute Loss vs No. of Epoch

Result for angle prediction

- We use three metric as performance measure of model

$$Mean\ Square\ Error(\hat{y}, y) = \frac{1}{n}\sum_{t=1}^{n}(\hat{y}_t - y_t)^2$$

$$Mean\ Absolute\ Error(\hat{y}, y) = \frac{1}{n}\sum_{t=1}^{n}|\hat{y}_t - y_t|$$

$$Mean\ Relative\ Error(\hat{y}, y) = \frac{1}{n}\sum_{t=1}^{n}\frac{|\hat{y}_t - y_t|}{y_t}$$

Test Result for angle prediction in Degree

Result for angle prediction in Degree			
Loss Functions	Mean Square Error	Mean Absolute Error	Mean Relative Error
Euclidean Loss	4.254	11.016	8.978
berHu Loss	2.973	8.275	6.799
Mean Absolute loss	**0.057**	**1.326**	**1.087**

Result:Comparison between results

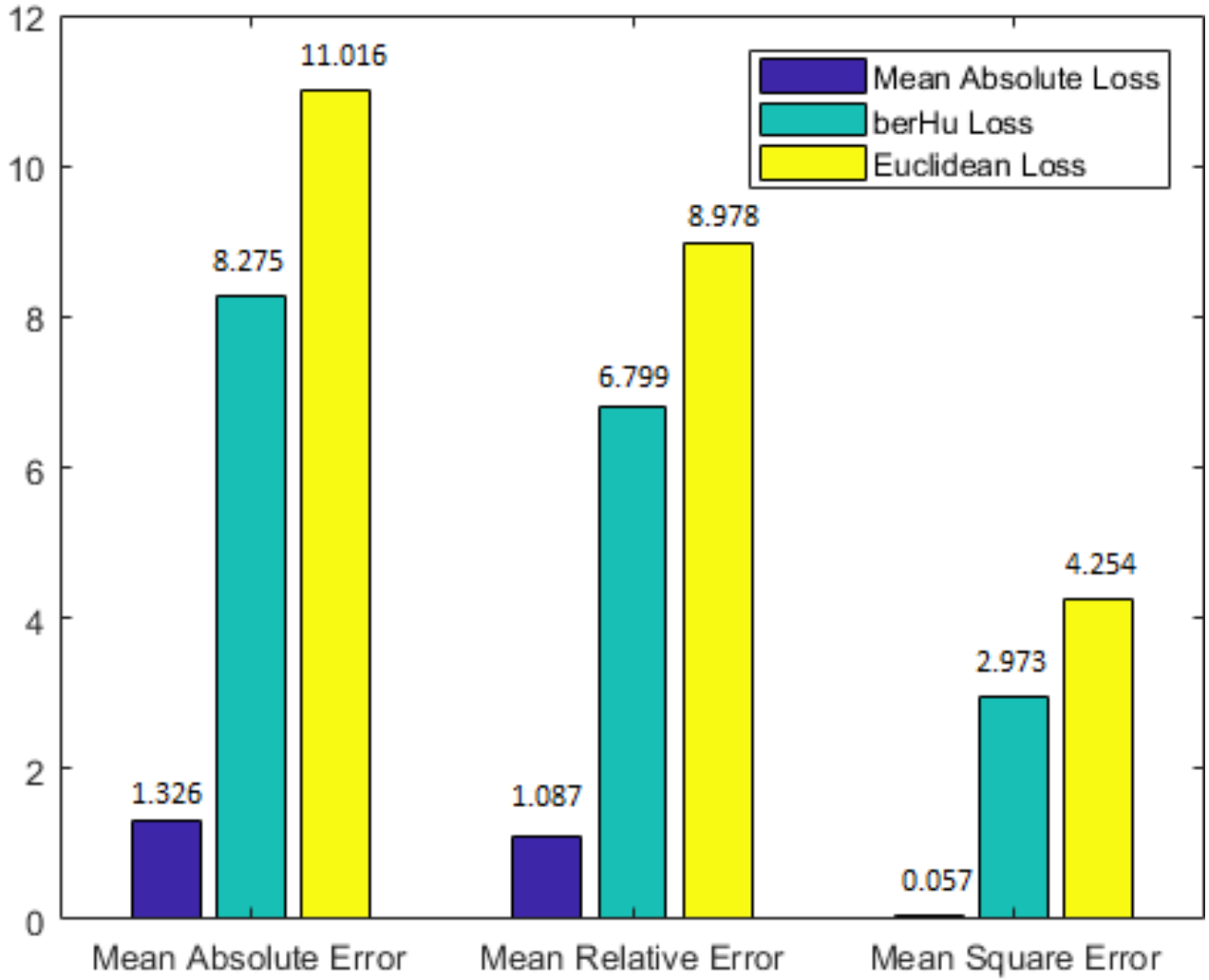


Figure 13: Comparison between results

Experiment:Finding intersection point between bisector and horizontal axis

- We use 35000 images for training and 600 images for test
- We use Mean Absolute loss function to learn our model

$$Mean\ Absolute\ Loss(\hat{y}, y) = \frac{1}{n}\sum_{t=1}^{n}|\hat{y}_t - y_t|$$

Experiment:Convergence Graph

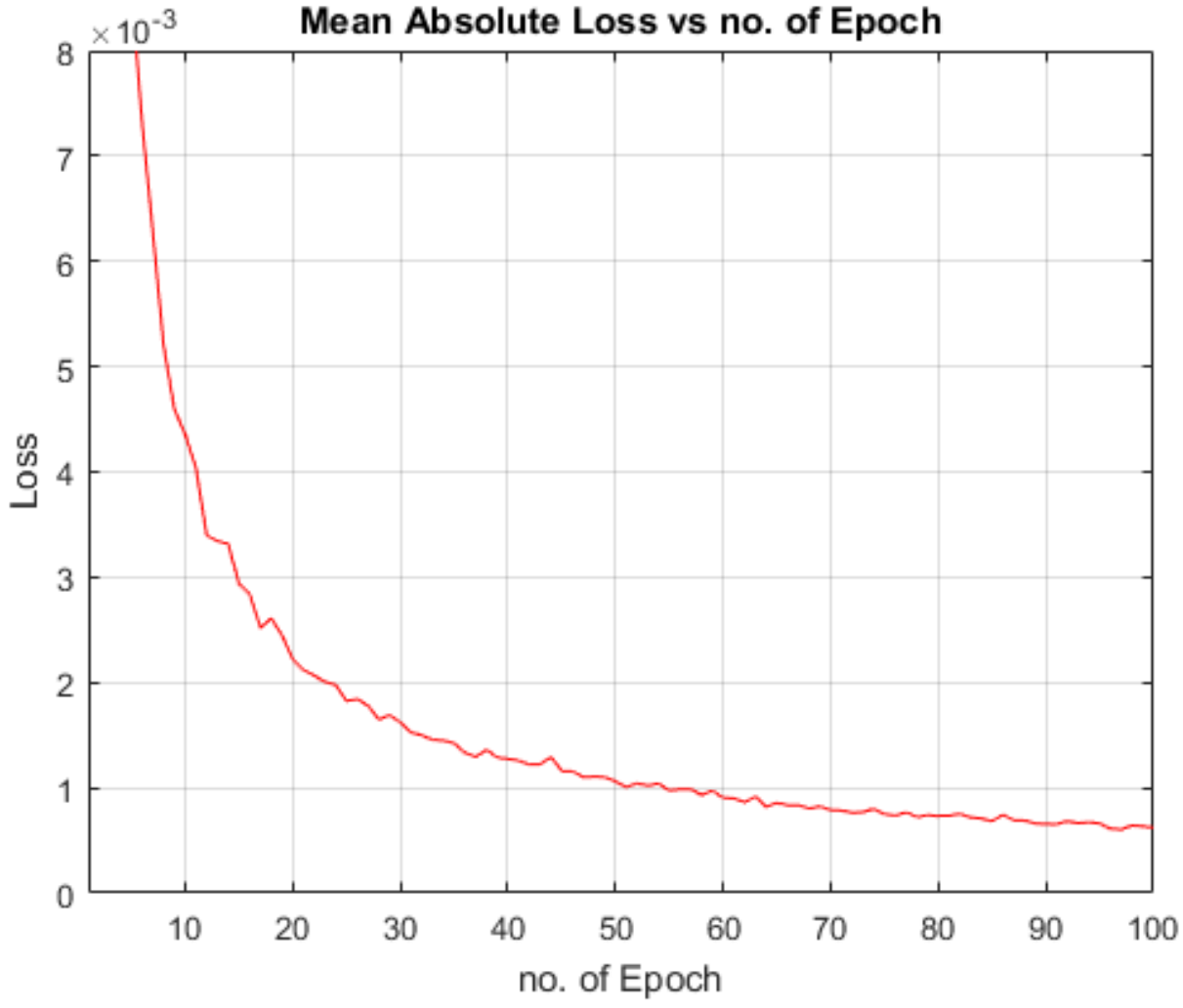


Figure 14: Mean Absolute Loss vs No. of Epoch

Result for intersection point prediction

- We use three metric as performance measure of model

$$Mean\ Square\ Error(\hat{y}, y) = \frac{1}{n}\sum_{t=1}^{n}(\hat{y}_t - y_t)^2$$

$$Mean\ Absolute\ Error(\hat{y}, y) = \frac{1}{n}\sum_{t=1}^{n}|\hat{y}_t - y_t|$$

$$Mean\ Relative\ Error(\hat{y}, y) = \frac{1}{n}\sum_{t=1}^{n}\frac{|\hat{y}_t - y_t|}{y_t}$$

Test Result for intersection point prediction in no. of pixels

Result for intersection point prediction			
Loss Functions	Mean Square Error	Mean Absolute Error	Mean Relative Error
Euclidean Loss	0.032	2.440	1.557

Conclusion

- We have used DenseNet-161[4] along with transfer learning and applied different loss such as Euclidean Loss,berHu Loss and Mean Absolute Loss to compute angle of bisector in image plane as results are shown above.
- We have used DenseNet-161[4] along with transfer learning and applied Mean Absolute Loss to predict intersection point of bisector and horizontal axis in image plane as results are shown above.
- From the above result we can conclude that the Mean Absolute loss work well with modified network.

Future Work

- Deploy the proposed algorithm in drone to autonomously localized and aid drone to navigate in indoor environment.

References I

[1] Matthew D Zeiler and Rob Fergus.
Visualizing and understanding convolutional networks.
In *European conference on computer vision*, pages 818–833. Springer, 2014.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton.
Imagenet classification with deep convolutional neural networks.
In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.
Deep residual learning for image recognition.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

References II

[4] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten.
Densely connected convolutional networks.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[5] Yokila Arora, Ishan Patil, and Thao Nguyen.
Fully convolutional network for depth estimation and semantic segmentation.

[6] Laurent Zwald and Sophie Lambert-Lacroix.
The berhu penalty and the grouped effect.
arXiv preprint arXiv:1207.6868, 2012.

THANK YOU.