END-TO-END LEARNING FOR SELF DRIVING CARS

Vaidehi Som

Indian Institute of Technology Jammu

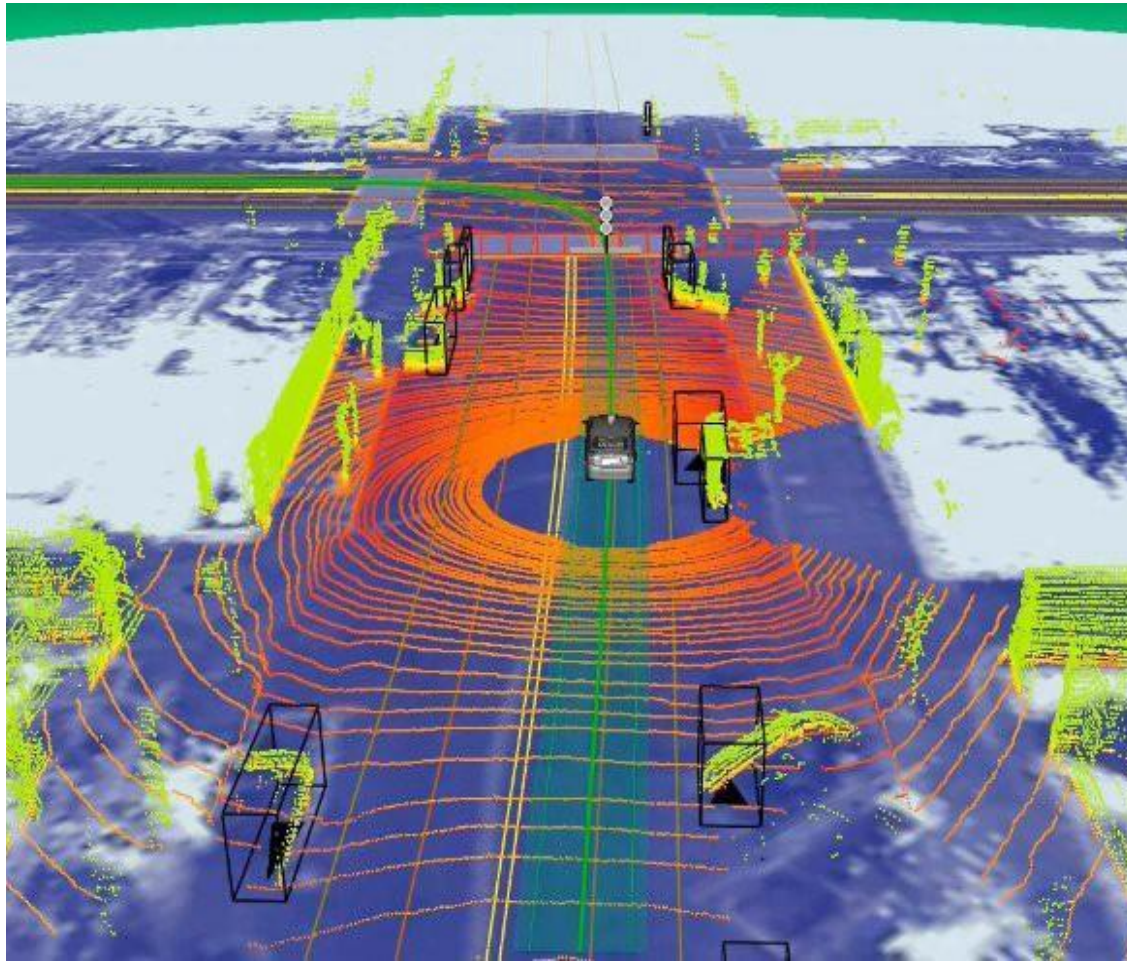
Dept of Mechanical

Internship report

Under the guidance of

DR. VIRENDRA SINGH,
Professor, IIT Bombay, Dept. of CSE

INTRODUCTION



Autonomous Cars Self driven cars that can perceive and drive without human intervention.

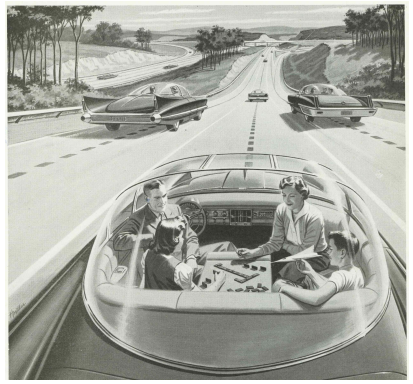
- Sense
- Process (model the outside world and take decisions)
- React, with appropriate movement.

Object avoidance & Path following Detect objects and follow path from sensor data.

Deep End-to-End Learning:

- Machine learning method based on learning data representations.
- E2E learning - Direct Input to Output mapping without data-process decomposition.
- Deep learning – Neural nets with multiple layers of hidden representations.

MOTIVATION

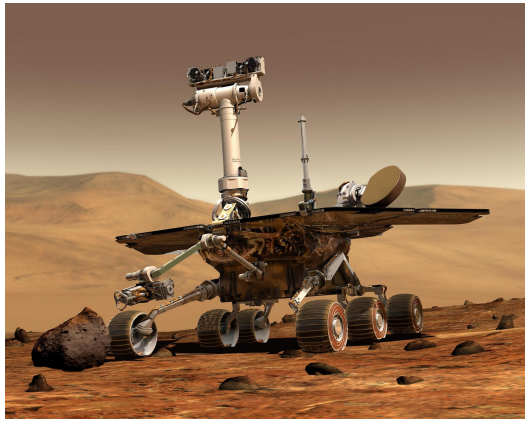


An artist's impression of an autonomous car, 1936

Heavy vehicle can operate for long distance without driver fatigue.

Shared communication on urban/city streets

Surveillance and Maintenance bot



Autonomous planetary expedition rover

Military Surveillance Vehicle

OBJECTIVE

Objective : Exercise and implement Deep learning methodology for autonomous control of the car.

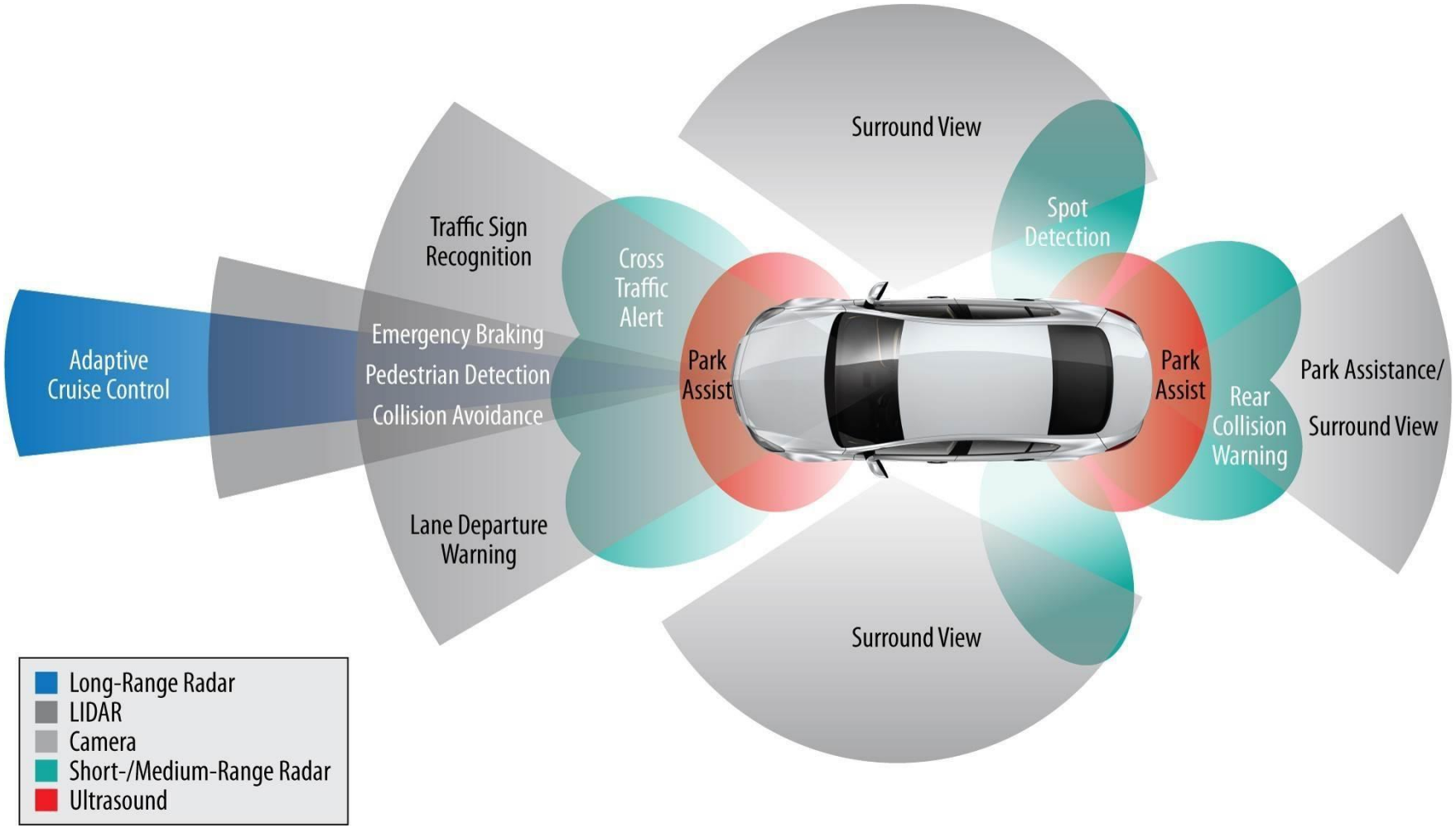
- Study of various deep learning approaches for autonomous vehicle.
- To develop deep learning based model which performs object detection and avoidance in autonomous vehicle.
- Testing and validation of the proposed model.

Various sensors equipped to a car.

LITERATURE REVIEW

ADAS

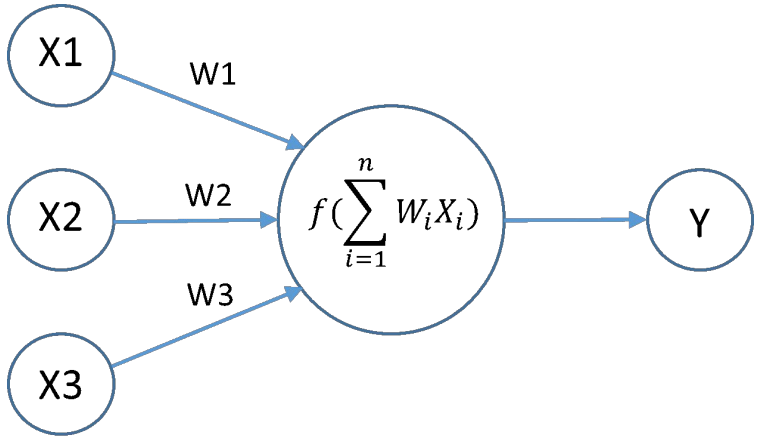
- Vehicle detection,
- Lane detection
- Object detection
- Traffic sign indicator
- Parking assistance
- Adaptive Cruise control
- Surround view



LITERATURE REVIEW

Neural Networks :

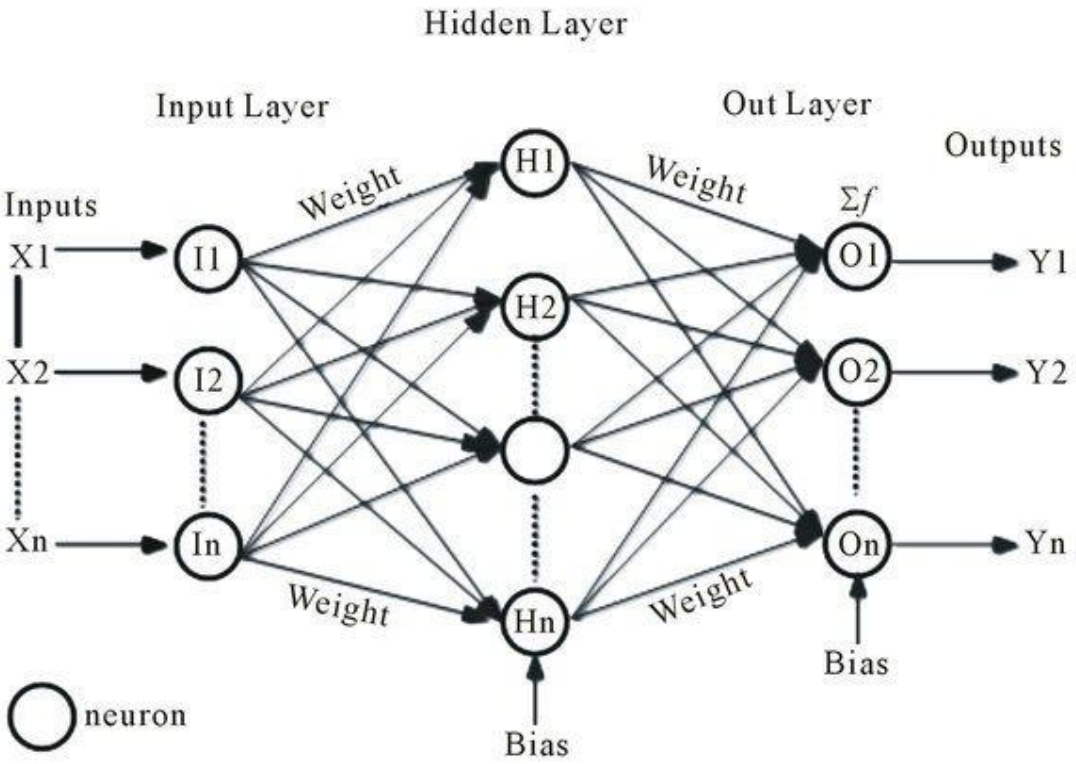
- Distributed parallel processing unit made up of perceptrons.
- Ability to learn from experience.
- Works for a wide range of problems.



Perceptron with weights and activation function.

LITERATURE REVIEW

Backpropagation



Effectiveness:

- Works for any number of hidden layer
- Deeper the network -> complexity increases
- Using different update rule:
    - Adagrad
    - RMSprop
    - SGD
    - Adam
- Tuning Hyperparameter:
    - Weight values
    - Learning rate
    - Bias

LITERATURE REVIEW

Backpropagation

$$E = \frac{1}{2}\sum(d-y)^2$$

$$\frac{dE}{dw_2} = \frac{d}{dw_2}\left[\frac{1}{2}(d-y)^2\right]$$

$$\frac{dE}{dw_2} = -(d-y).y.(1-y)z_h$$

$$\Delta w_2 = \alpha * \frac{dE}{dw_2}$$

$$w_2 = w_2 - \Delta w_2$$

$$\frac{dE}{dw_1} = \frac{d}{dw_1}\left[\frac{1}{2}(d-y)^2\right]$$

$$\frac{dE}{dw_1} = -(d-y).y.(1-y)\frac{d}{dw_1}(z_h * w_2)$$

$$\frac{dE}{dw_1} = -(d-y).y.(1-y).w_2.z_h(1-z_h).x_{conv}$$

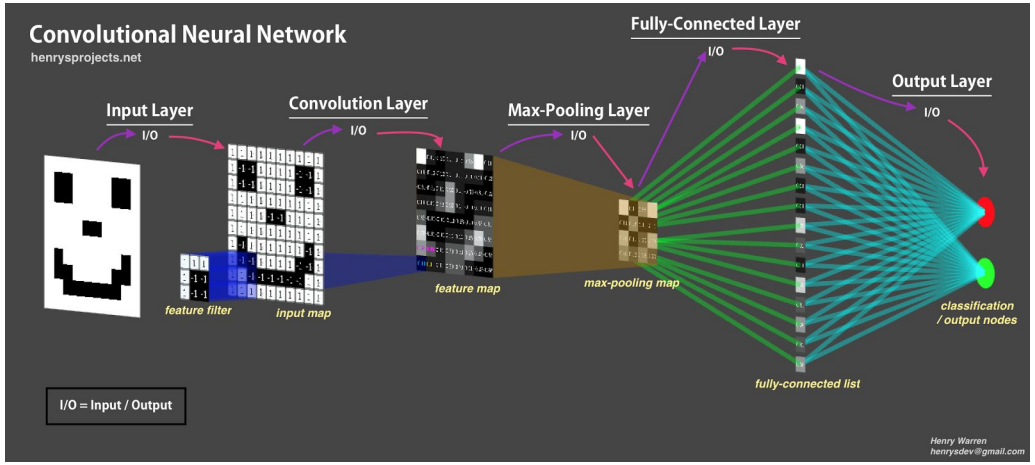
$$\Delta w_1 = \alpha * \frac{dE}{dw_1}$$

$$w_1 = w_1 - \Delta w_1$$

Parameter update: Stochastic gradient descent
Updated weights to be used in next iteration.

w_2 = *hidden-output weights*
w_1 = *input-hidden weights*

LITERATURE REVIEW



Simple CNN architecture

Convolution Neural Networks

- Kernels, instead of weights to extract features.
- Every kernel creates a different representation of feature maps.
- Pooling is done to reduce redundant data.
- 3 basic layers : convolution, pooling, flatten, fully connected layer.

LITERATURE REVIEW

Convolution Neural network

Convolution:

- Input array size = 90
- Kernel array size = 5
- Number of Filters = varies layer to layer
- Number of layers = Also varies based on problem. (hyperparameters)

Activation function:

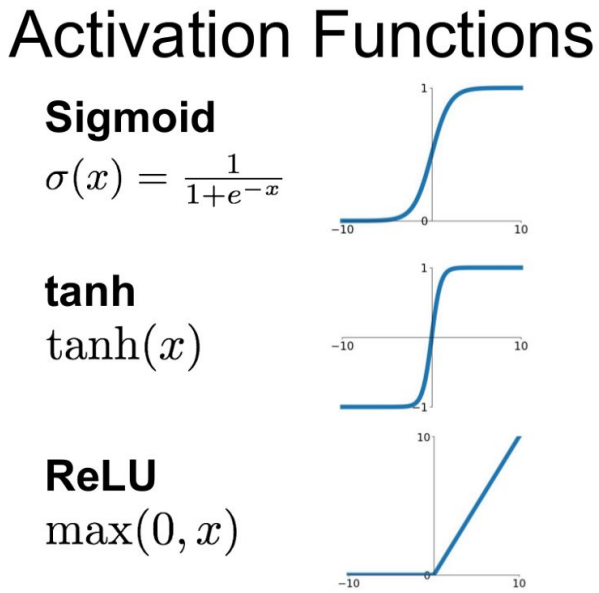
- Bounds the output values between range.
- Provides gradient to the inputs.
- Types -> *sigmoid, tanh, SoftMax, ReLU*

Fully Connected Layer:

- Reshapes the convolution tensors
- Decision making layer
- Forms the output layer with activation function.

Pooling:

- Max pooling, Average pooling,
- Reduces the size of the input, bringing out crisp features.



LITERATURE REVIEW

Neural Networks	Convolution neural Networks
Can perform both Supervised and Unsupervised Learning	Can perform both Supervised and Unsupervised Learning
Inputs are shuffled along the layers	Inputs adjacency is preserved along the layers.
Less computational complexity	Computationally more complex
Large networks are time consuming	Comparatively more time consuming

LITERATURE REVIEW

[4] *Deep Learning for Automated Driving:*

- Addresses 2 detection problem: Vehicle and Lane
- Uses MATLAB's deep learning and GPU acceleration modules.
- Uses a convolution neural network for learning both tasks.

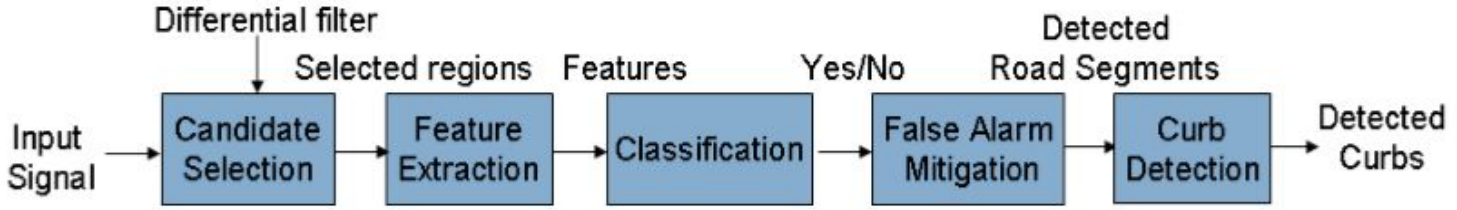
[1] *ALVINN: An Autonomous Land Vehicle in a Neural Network:*

- 3 layer backpropagation neural network for road following.
- Uses camera and laser range finder as inputs
- Trained using simulated road images.

LITERATURE REVIEW

[13] *LiDAR- based road and road-edge detection:*

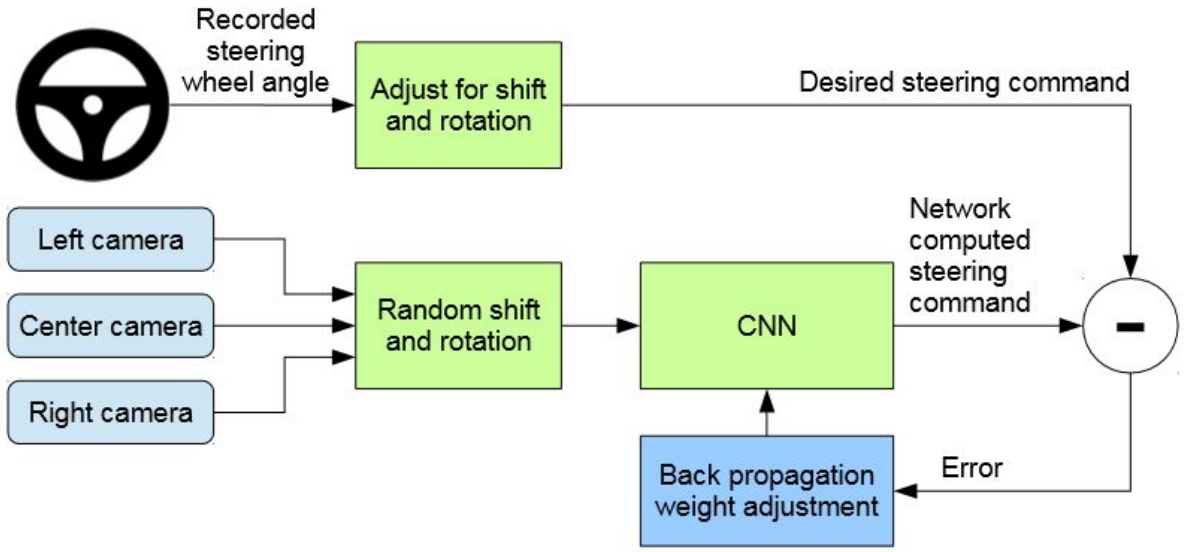
- Uses only LiDAR range data as input.
- Pre-processing of LiDAR data to form elevation maps and line representation of ground plane.
- Trains the classifier using SVM.



LITERATURE REVIEW

[8]*End to End learning for Self-Driving cars:*

- Uses 3 mounted cameras.
- Inputs raw image data from 3 cameras and outputs steering angle.
- Trained by a human driver, who provides steering angles as training signals.
- CNN for classification
- Backpropagation for Weight adjustment



Schematic of DAVE 2

LITERATURE REVIEW

End to End learning:

- Variant of deep learning for complete input to output mapping.
- Learns complex representation of the system, without decomposing them into independent modules.
- Uses gradient descent methods to guide a random initial state to a highly non-trivial solution.
- Amount of training data increases exponentially with the number of modules used.
- Every learning step is directed to the final output, by the overall objective function.
- Optimization is based on variants of Stochastic Gradient Descent methods.

"Dave" - DARPA's seedling project, - Yann LeCun and Urs Muller, uses this model for end-to-end learning

PROPOSED WORK

Progress of Project

- Implementation of End-to-End learning
  - Neural Net
  - Convolution Neural networks
  - Training the network (Data?)
- Designing a 1D-distance acquisition sensor
- Designing a robot for driving data.
- Training the network on data
  - Direction classification using Distance acquisition sensor data for object avoidance
  - Actuator data prediction using Images for path following

PROPOSED WORK

Neural networks And Convolutional neural networks:

- Both are state-of-the-art self learning mathematical model
- Can be used for both classification and regression tasks
- Consists of hidden layers, containing large number of neurons
- Both models can perform on gradient based optimizers
- Hyperparameters are : number of layers (pooling and fully-connected), number of neurons in each layer, optimizer, loss function, activation function, kernel size and strides (for convolutionalNN).
- Also known as deep learning algorithms

PROPOSED WORK

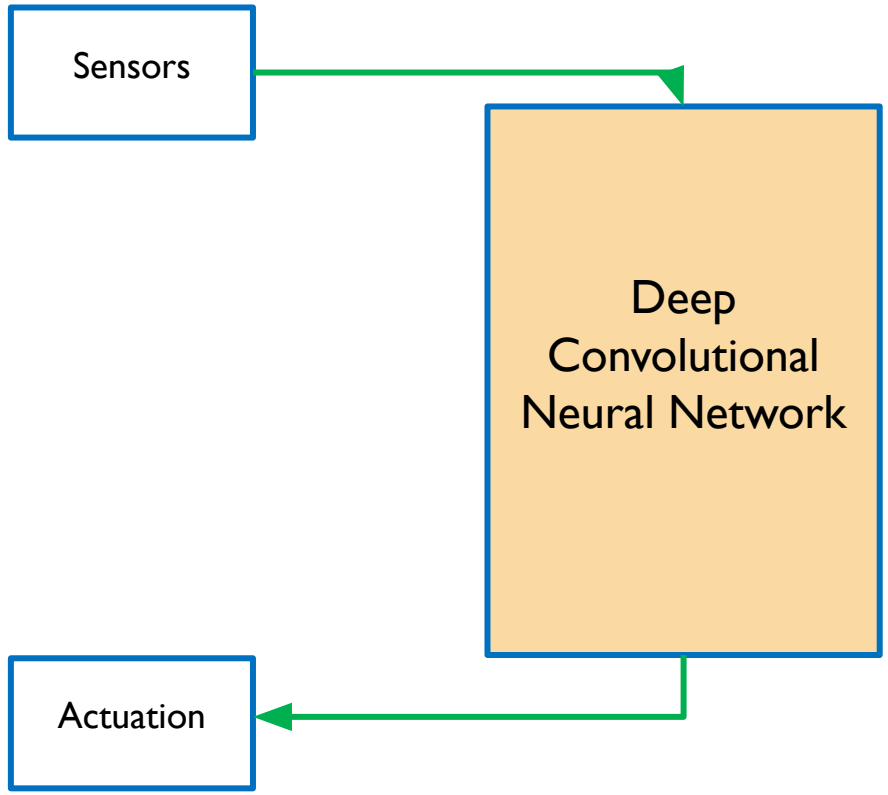
End-to-End learning:

Traditional Approach

- Developing image and different sensor feature extractors.
- Hand labelling a large set of object on the road.
- Decisions made of hand-crafted rules

End-to-End Approach

- Learns to steer by observing a human driving pattern
- No hand-crafted rules.
- No labelling of objects

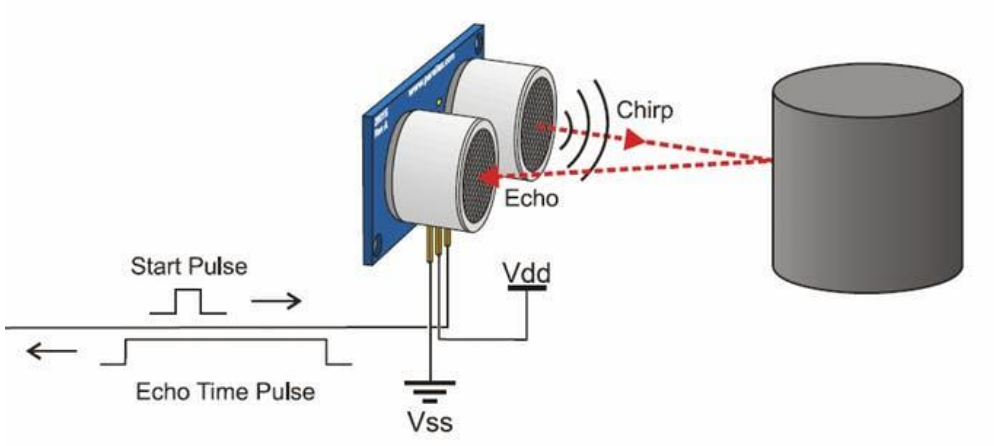


PROPOSED WORK

Component Overview:

Ultrasonic Sensor:

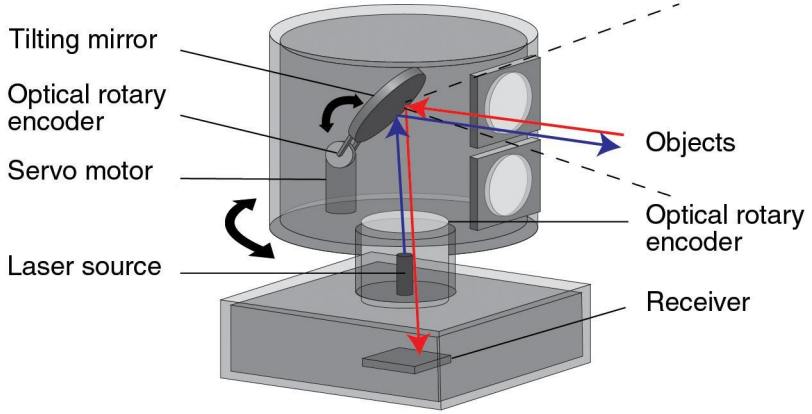
- Uses sound wave to measure distance.
- Min – Max range -> 0.02m – 4m
- Affected by temperature and humidity of the air.



Working of an ultrasonic sensor

LiDAR:

- Light imaging Detection And Ranging
- Min – Max range -> 0.5m – 20m
- Low response time
- Can detect almost any object.
- Uses pulsed laser light to measure distance.



Working of a LiDAR

PROPOSED WORK

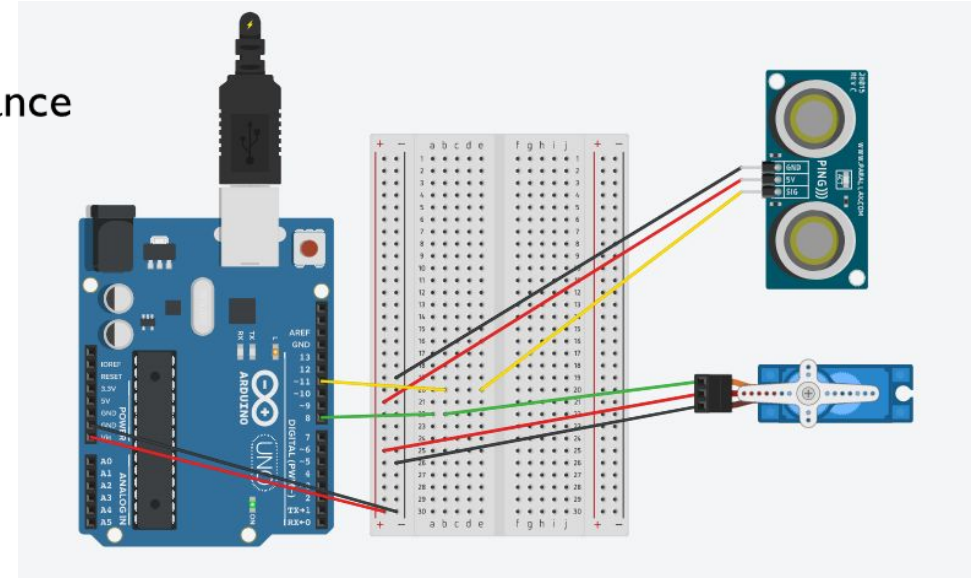
Collecting inputs:

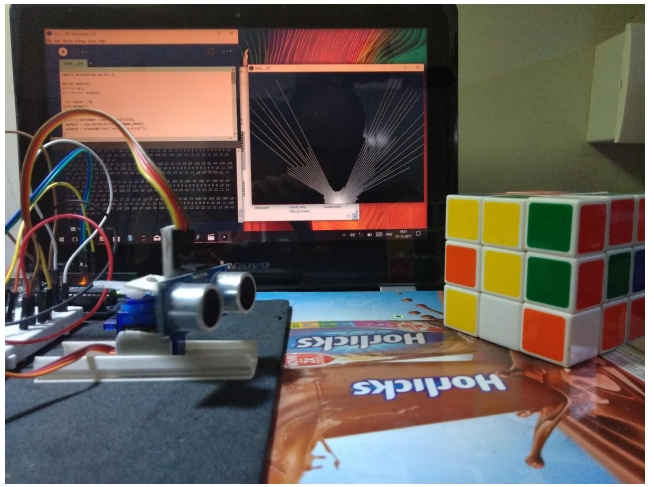
- Ultrasonic sensor scans 180°
- Resolution = 2°.
- Size of the input vector = 90
- Every element value is a distance from sensor to objects.

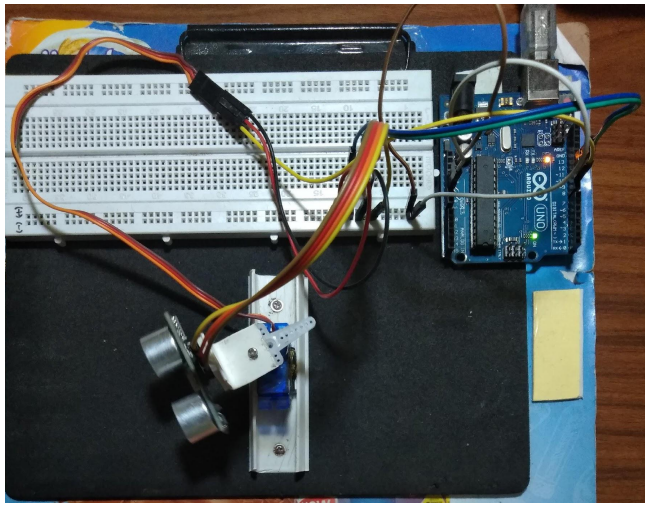
Output Label:

- Straight -> [1 0 0 0]
- Left -> [0 1 0 0]
- Right -> [0 0 1 0]
- Back -> [0 0 0 1]

Scan of the surrounding, with visualization







Setup of the prototype

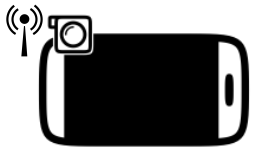
PROPOSED WORK

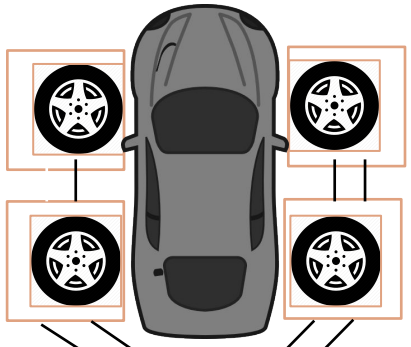
Component Overview:

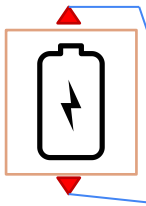
Microcontroller	Joystick	Bluetooth	Motor Shield	Camera	Chassis & Motor	Remote System
Arduino UNO R3	RedGear	HC-05	L298N	Android Phone	Assembled	Laptop
• Operating Volt = 5V • Recommended = 7 – 12V • Atmega 328 microprocessor	• 2 axis controller	• OV – 3.3 V– 5V • PIO control • 9600 baud rate • Connects with last device on power up	• Input = 12V • Output = 5V • 16 pin IC	• Resolution – 240 x 320 • Frame rate = 10 fps • Accessible through IPWebcam App • Integrated Wi-fi.	• Dim = 32x27 cm • 2 motor wheel , one supporting wheel • Stages microcontroller, camera, bluetooth, motor shield.	• Wi – fi connectivity. • Data storage • Control signal transmission unit. • Processing unit
Control of all components	Controlling the bot	Control receiving unit.	Controls 2 set of DC-Motors	Capture Images	Robot platform	Robot control unit.

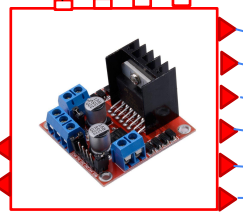
PROPOSED WORK

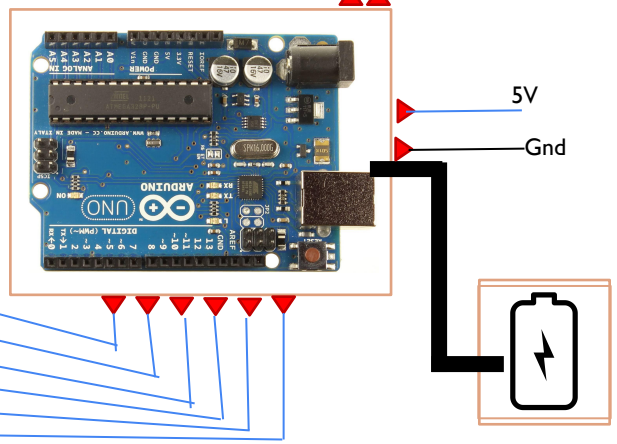
Component Assembly









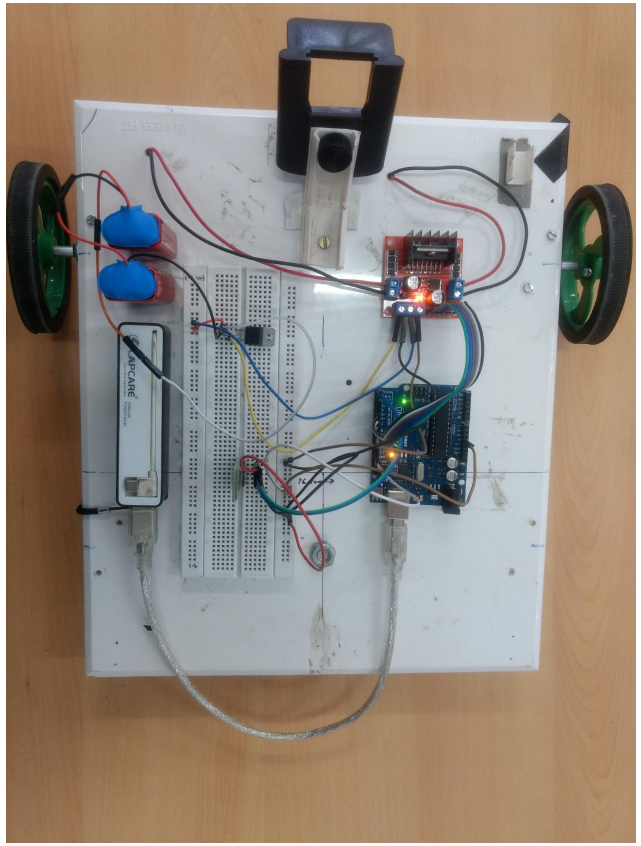


Record saved data .csv

W_set1	W_set2
-0.992	0
-0.992	0
-0.992	0
-0.992	0
-0.992	-0.992
-0.992	-0.992
-0.992	-0.992
-0.992	-0.992
-0.992	-0.992
-0.992	-0.086
-0.992	-0.086
-0.992	-0.086
-0.992	0

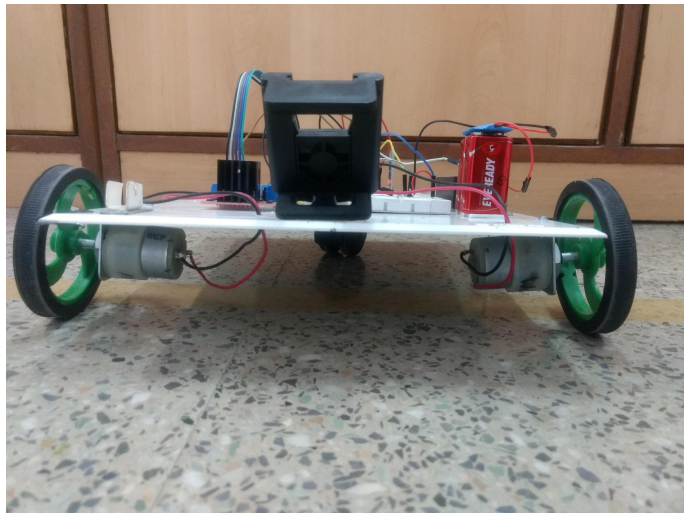
PROPOSED WORK

Data acquisition robot for end-to-end learning

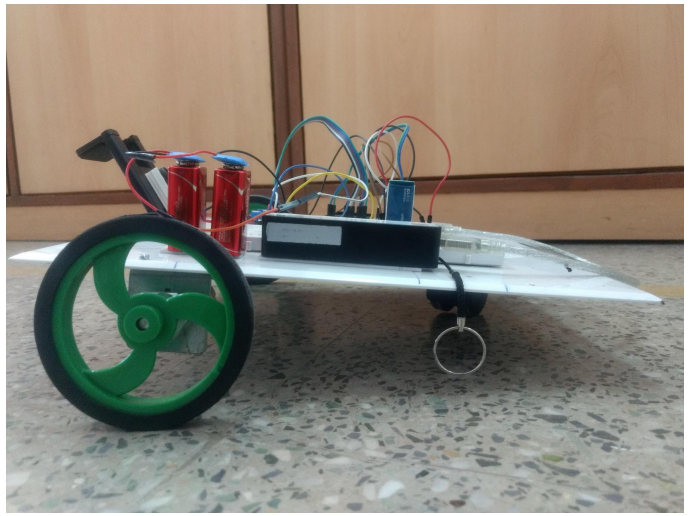


Top view

Front View

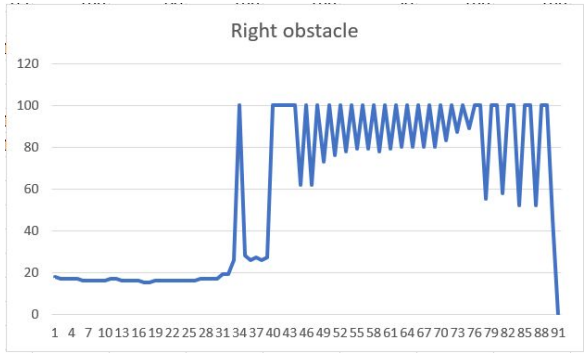


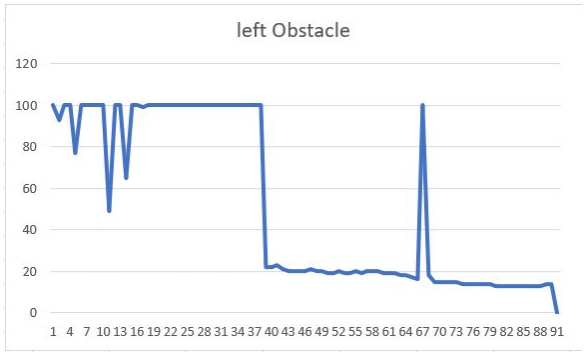
Side View

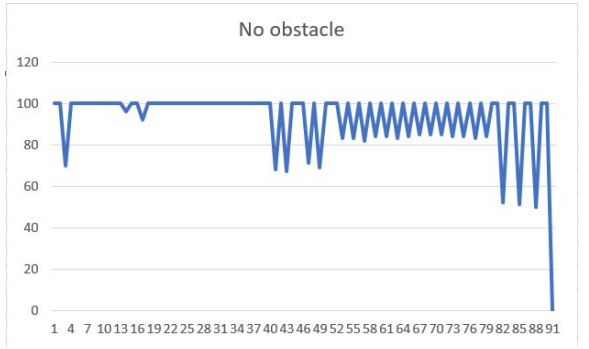


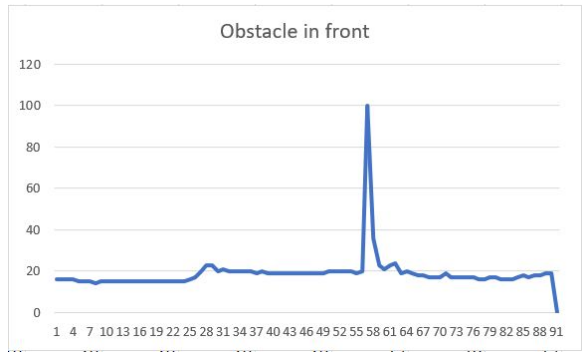
DATA VISUALIZATION

Data of 1D Distance Acquisition Sensor

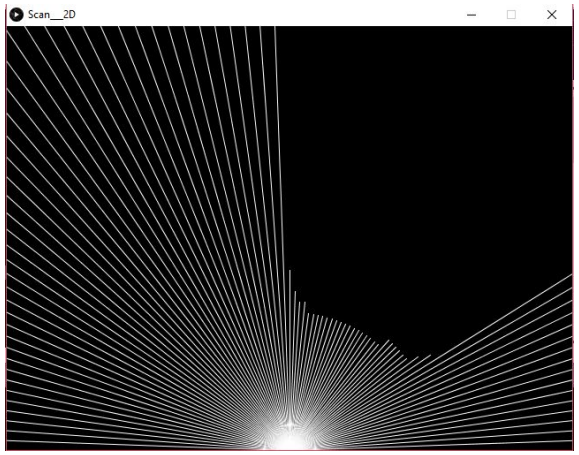


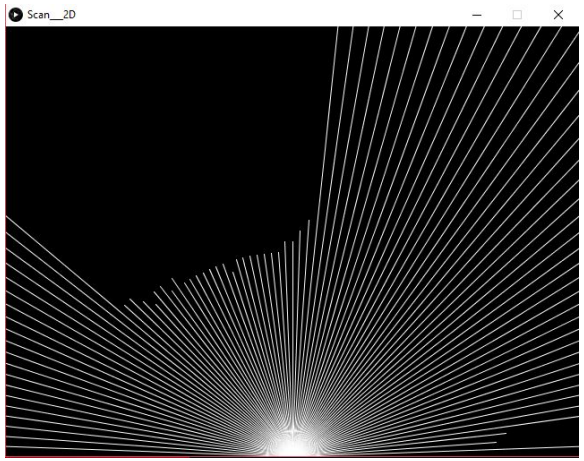


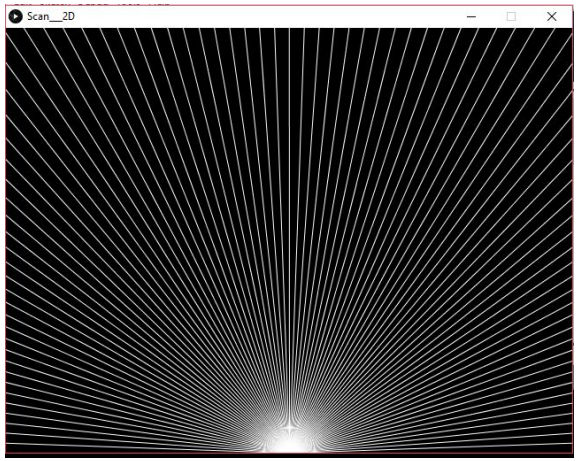


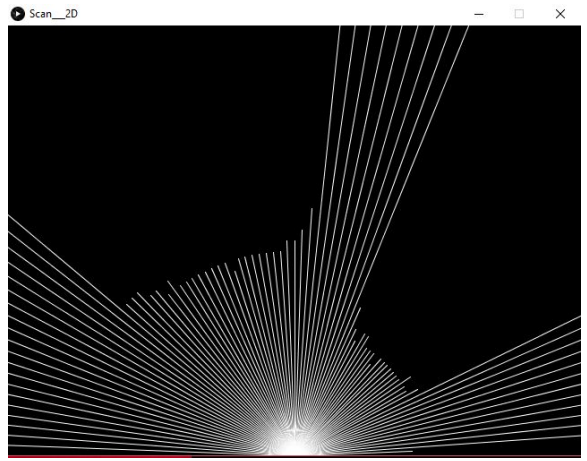


Data visualisation in MS Excel









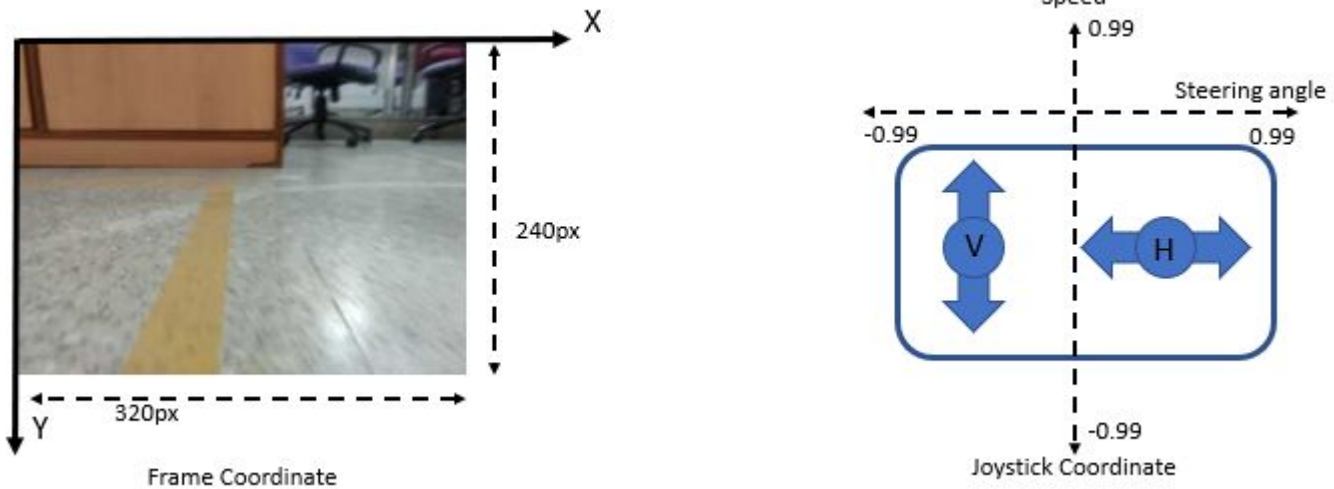
Data visualisation in Processing

DATA VISUALIZATION

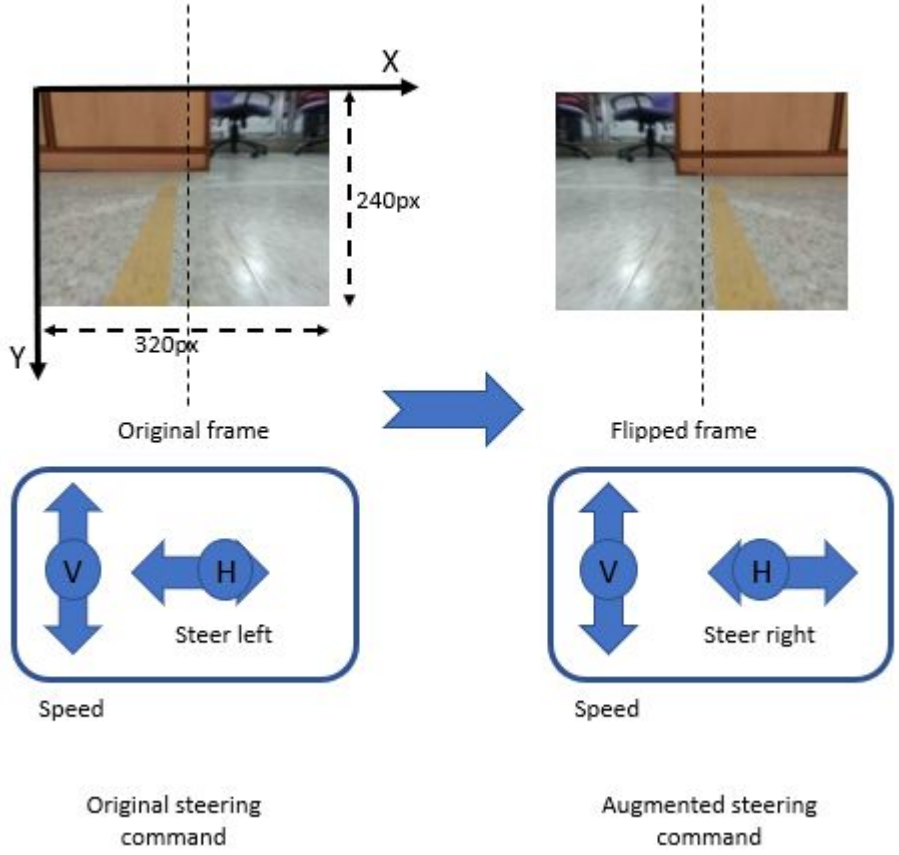
Data Augmentation for Images

Importance of data augmentation:

- Technique for increasing training accuracy
- Brings variability of data by random transformations, which needs the steering data to be adjusted
- Increases the data set
- Three performed transformations – flip, crop and translate



Frame and joystick coordinate system

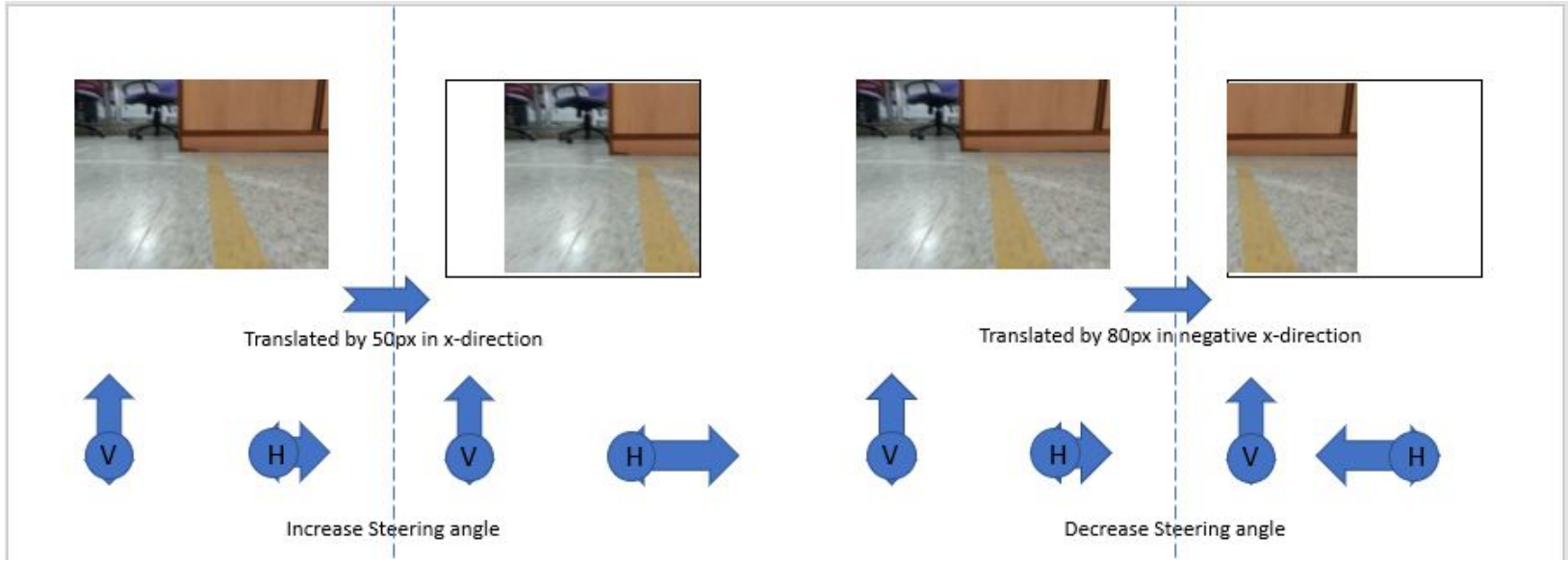


Flipped image and adjust steering angle

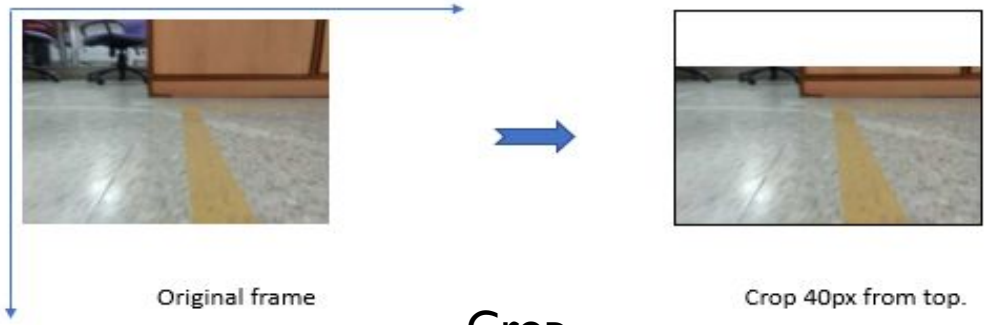
DATA VISUALIZATION

- Cropping is done to reduce unwanted data, so reduces the size of the image.
- Translation of image in this case is helpful for adjusting to the track.

Data Augmentation



Random translation



Crop

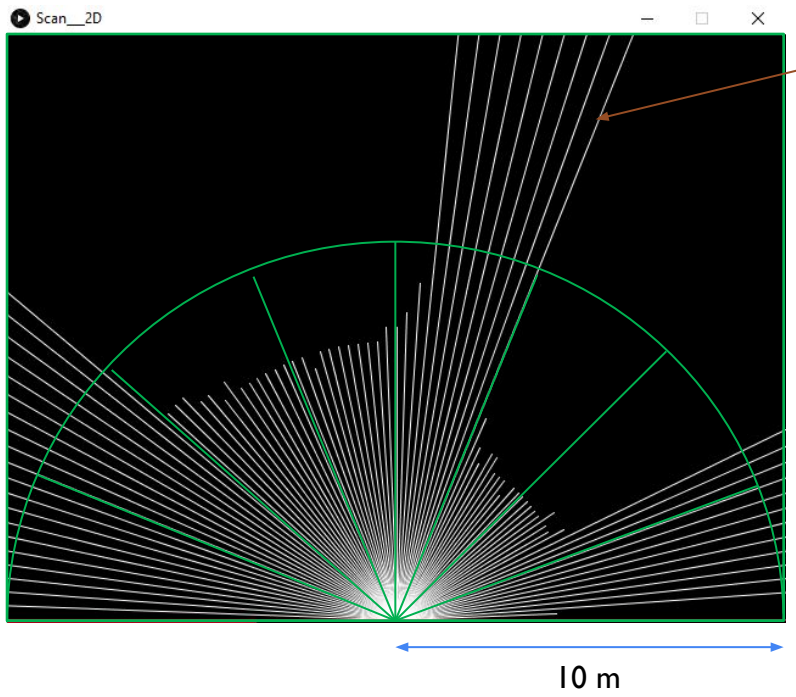
METHODOLOGY

Data Description:

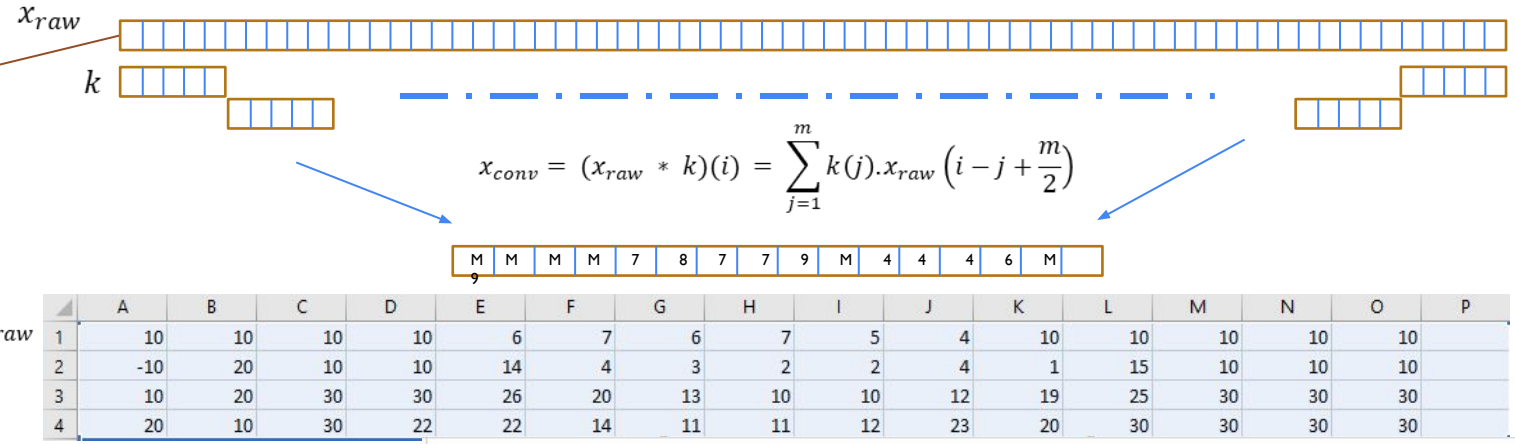
- 1D-Distance Acquisition Sensor
  - Each pixel value indicates positional data.
  - The data is dependent on its neighbouring pixels.
  - 1-Dimensional data
  - Loss of relevant information if data is shuffled.
- Robot data acquisition:
  - Camera images of dimension = 240x320
  - Frame rate = 10 fps
  - The pixel values are dependent on neighbouring pixel values.
  - Loss of relevant information if data is shuffled.

METHODOLOGY

Using convolution for 1D Array of data:



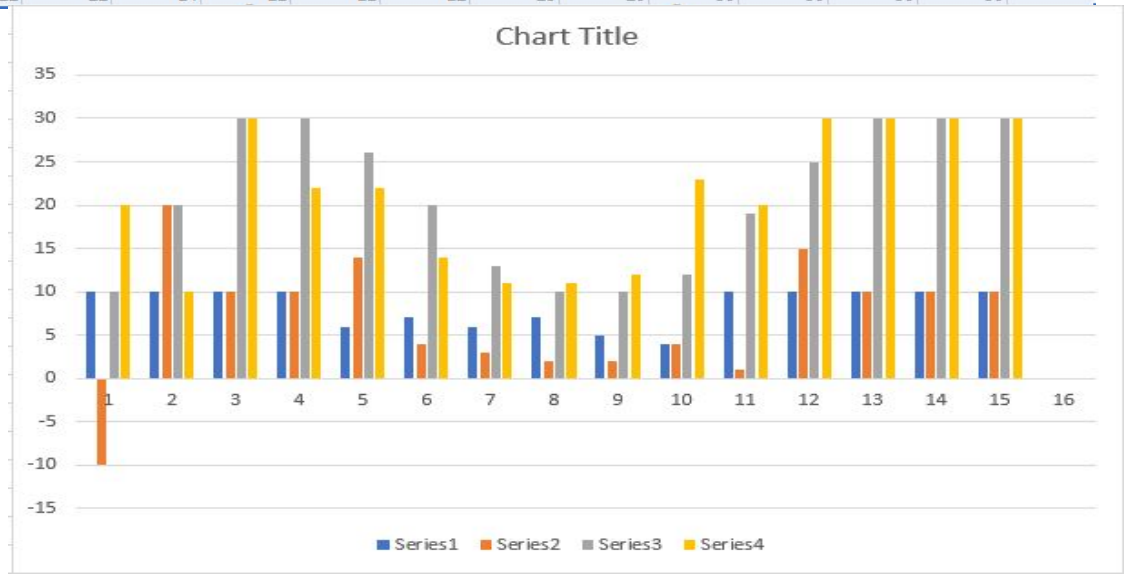
Monitor displaying objects and spaces as sensor interprets – viewed in Processing



x_{raw}

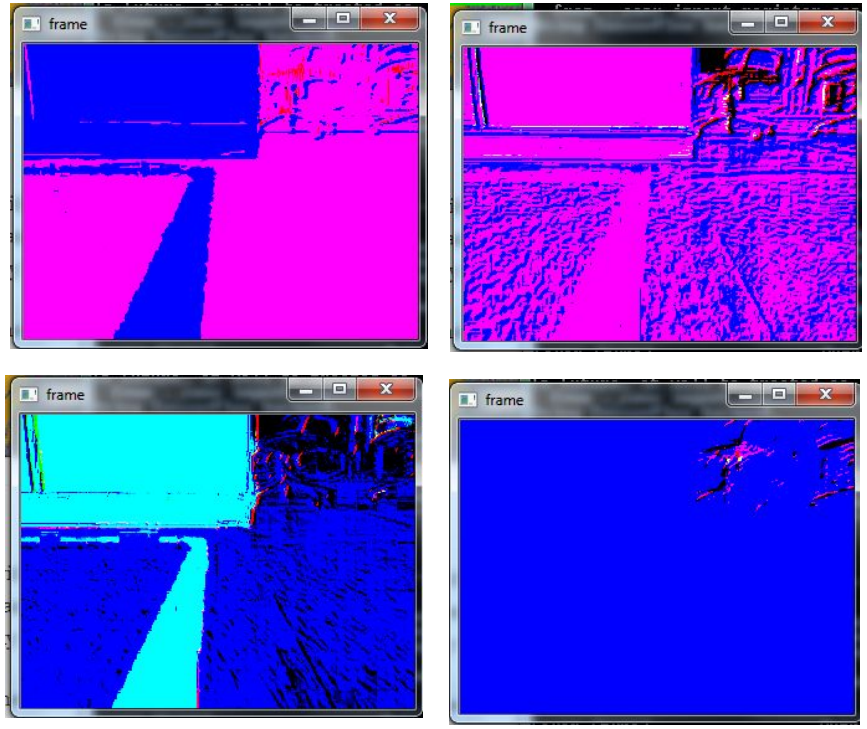
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	10	10	10	10	6	7	6	7	5	4	10	10	10	10	10	
2	-10	20	10	10	14	4	3	2	2	4	1	15	10	10	10	
3	10	20	30	30	26	20	13	10	10	12	19	25	30	30	30	
4	20	10	30	22	22	14	11	11	12	23	20	30	30	30	30	

- Series 1 = x_{raw}
- Series 2 = conv(x_{raw},k), k=[-1, 3, 1]
- Series 3 = conv(x_{raw},k), k=[1, 3, 1]
- Series 3 = conv(x_{raw},k), k=[2, -1, 2]



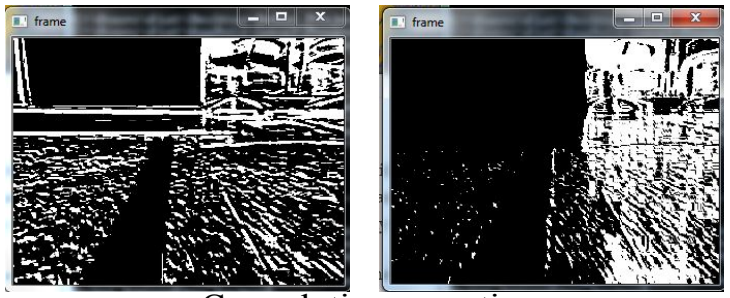
METHODOLOGY

Image representation based on number of kernels and layers

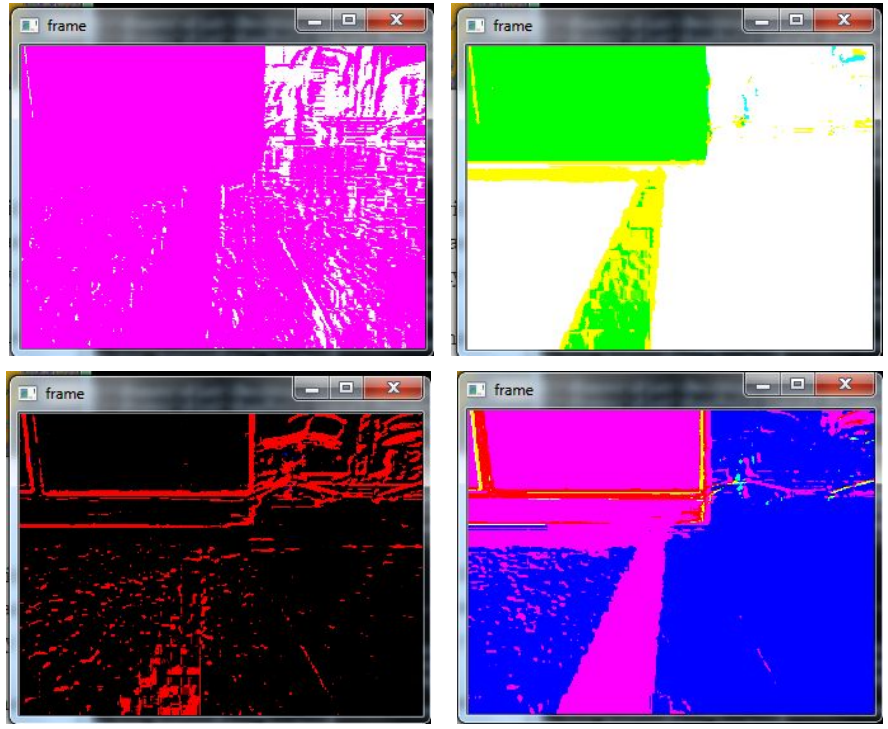


Convolution operation
No. of filter : 3
No. of layers : 1

Original Image



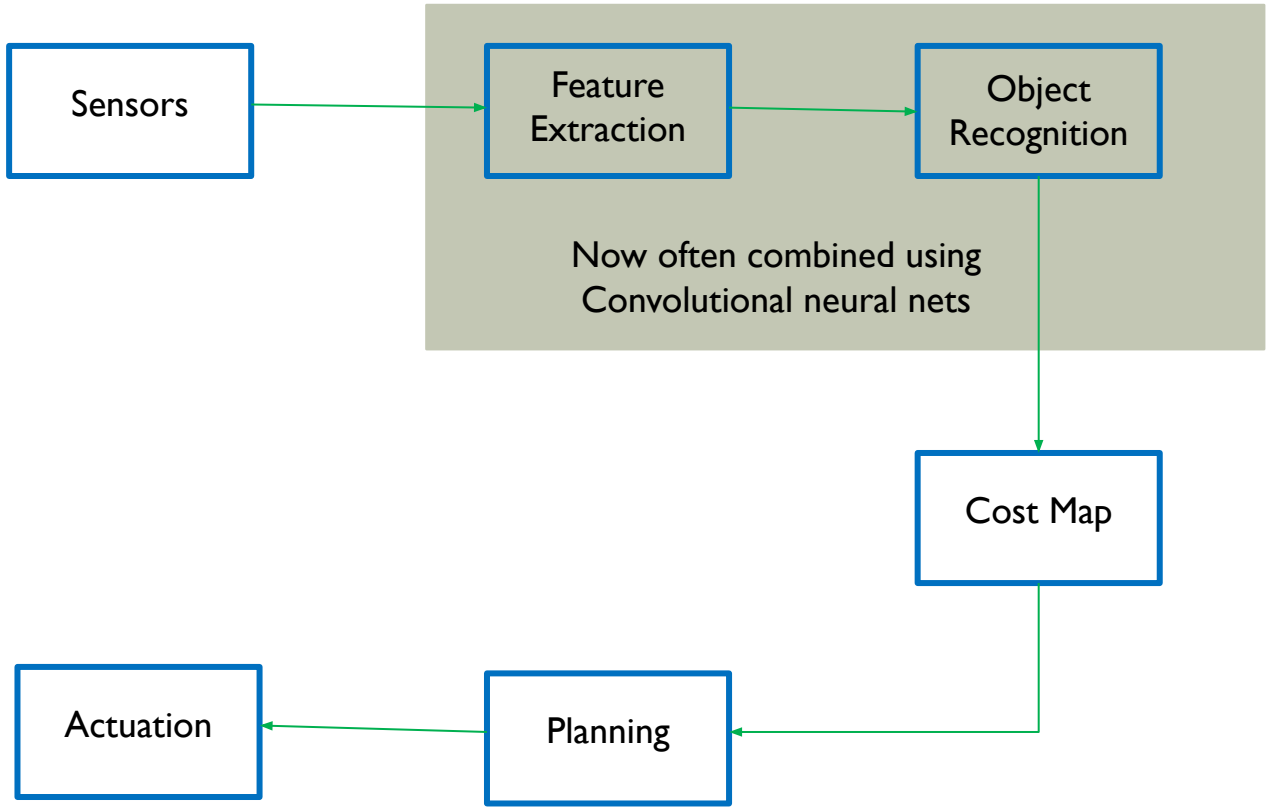
Convolution operation
No. of filter : 1
No. of layers : 1



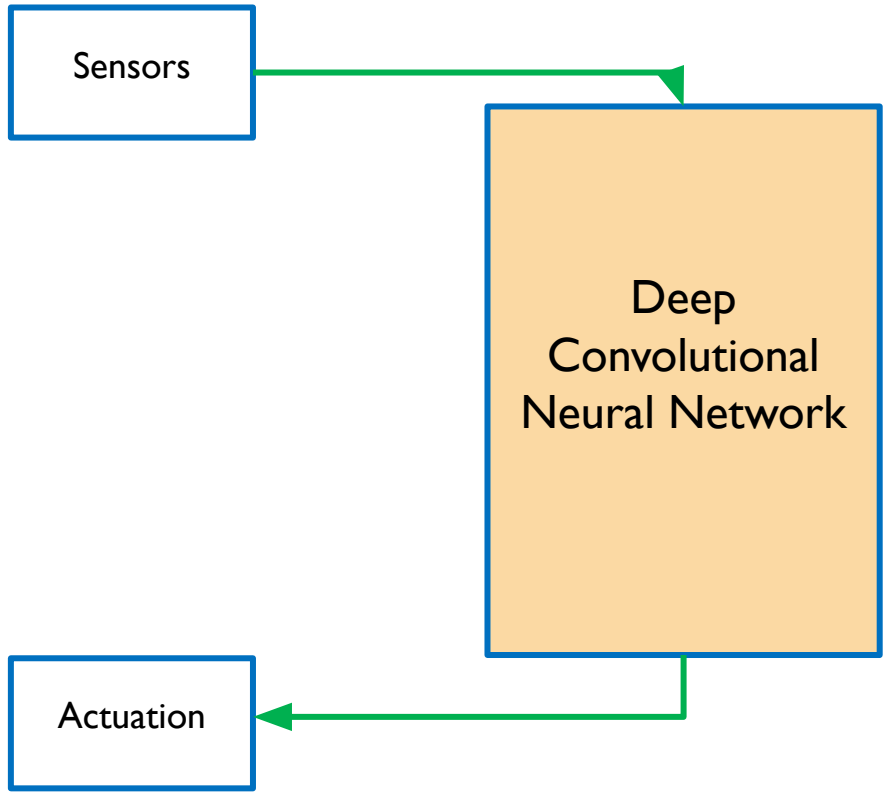
Convolution operation
No. of filter : 2
No. of layers : 2

METHODOLOGY

End-to-End learning:



Usual Approach for self-driving car



End-to-End Approach

METHODOLOGY

Model Description

Direction classification

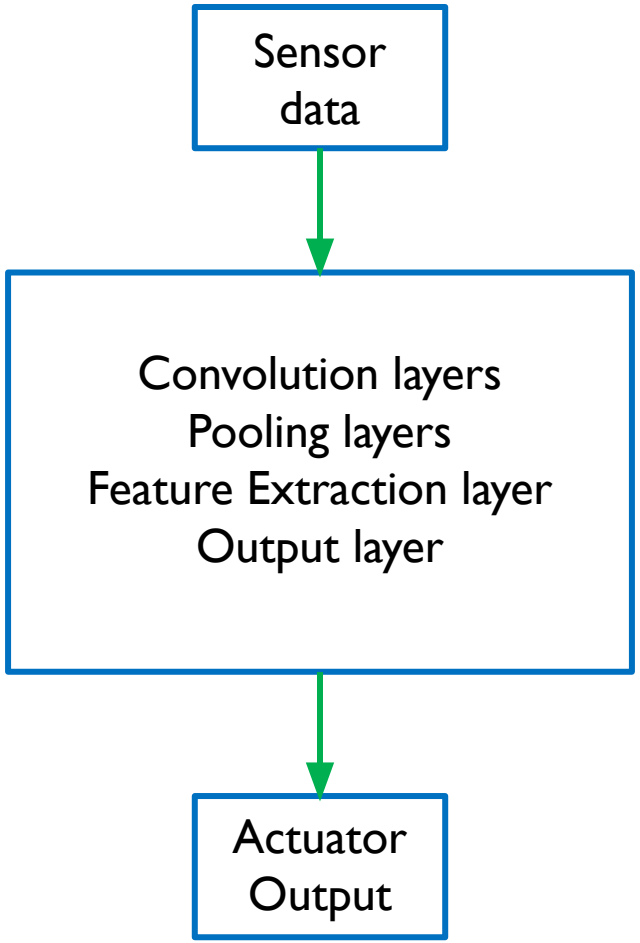
- Inputs – 1 dimensional
- Outputs – Straight , Left, Right, Stop
- Number of parameters –

$$k * (w * h * c + 1)$$

Here, w = 3/5/7
h = 1
c = 1,

$$k * (w + 1)$$

- Output Activation function – SoftMax
- Rest of the hyperparameters are variables and cannot be predefined.



Actuators value regression

- Inputs – 3 dimensional
- Outputs – Left & Right Motor values
- Number of parameters –

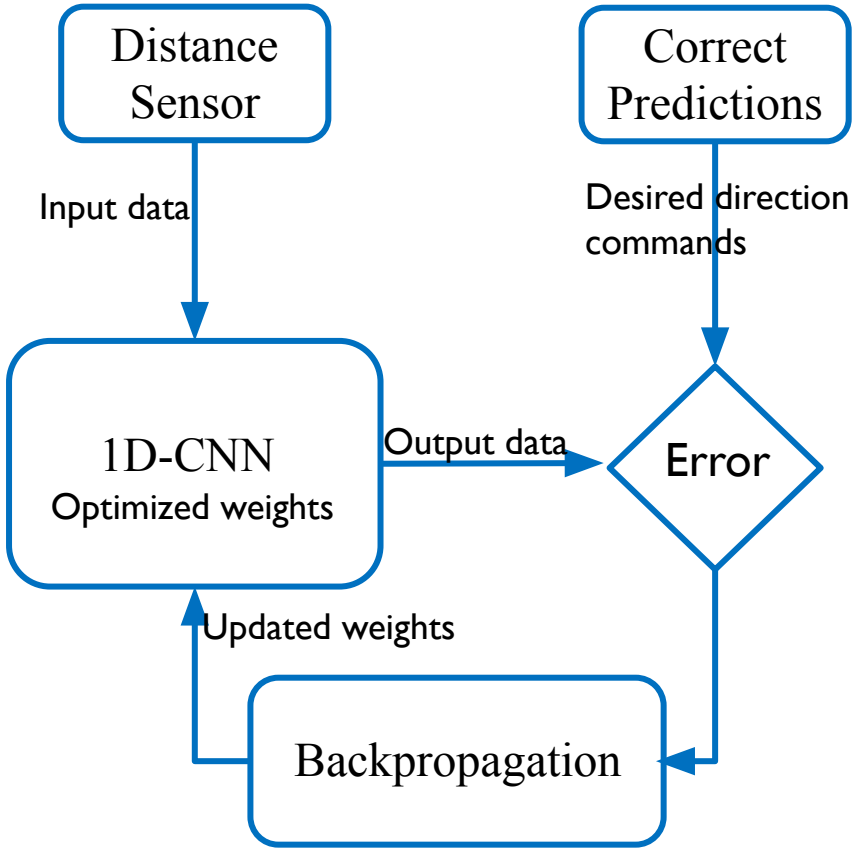
$$k * (w * h * c + 1)$$

Here, w = 3/5/7
h = 3/5/7
c = 3,
k = filters

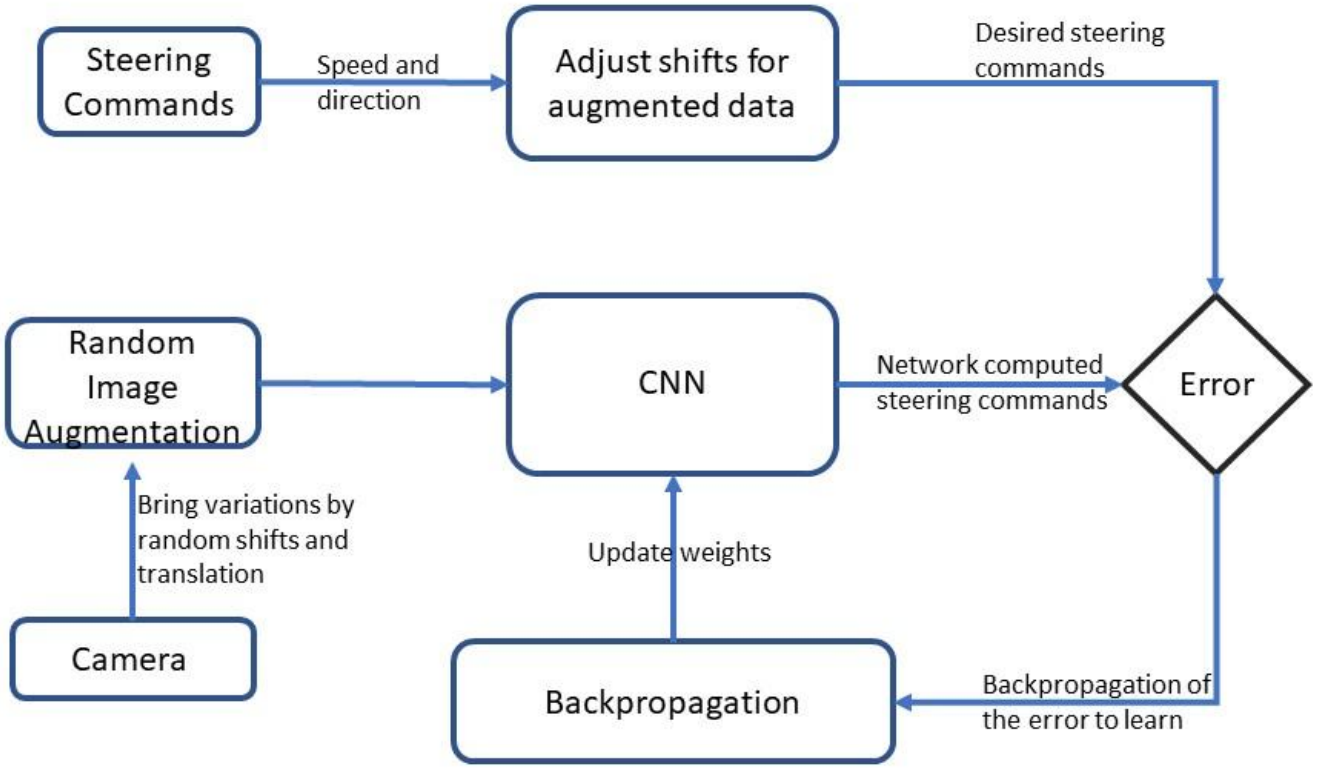
- Output Activation Function – Tanh
- Rest of the hyperparameters are variables and cannot be predefined.

METHODOLOGY

Direction classification

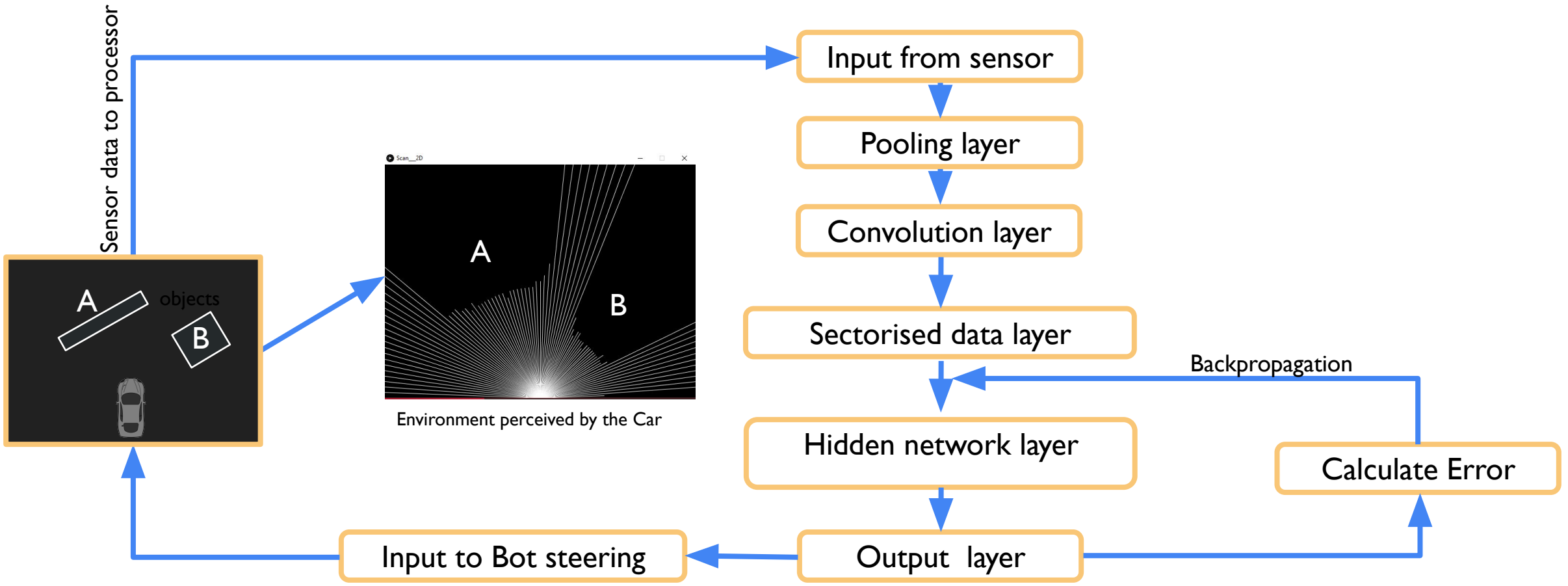


Actuators value regression



TRAINING AND RESULT

Direction Classification for object avoidance



TRAINING AND RESULT

Direction Classification for object avoidance

```
------------------------------
parameters
------------------------------
learning rate = 0.0001
batch_size = 10
training_epochs = 80
loss = categorical_crossentropy
optimizer = SGD
n_dim = 90
```

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 88, 64)	256
max_pooling1d_1 (MaxPooling1	(None, 44, 64)	0
conv1d_2 (Conv1D)	(None, 42, 16)	3088
max_pooling1d_2 (MaxPooling1	(None, 21, 16)	0
flatten_1 (Flatten)	(None, 336)	0
dense_1 (Dense)	(None, 600)	202200
dense_2 (Dense)	(None, 4)	2404

Total params: 207,948
Trainable params: 207,948
Non-trainable params: 0

Summary of the Model

TRAINING AND RESULT

Direction Classification for object avoidance

Training parameters in different network architectures

Loss	Acc	CL	F	PL	Act_CL	FCL	N	Act_FCL	Opt
0.67	0.34	3	8 – 16 - 32	3	ReLU – Relu - Relu	3	160 -30 - 4	Relu - softmax	SGD
0.58	0.64	1	8	2	Relu	3	400-150 - 4	tanh - tanh softmax	SGD
0.37	0.74	3	8 – 16 - 32	3	Relu – relu - relu	2	160 - 4	Relu - softmax	SGD
0.45	0.87	3	48 -16 - 8	3	Relu – relu - relu	2	50-4	tanh - softmax	Adam
0.32	0.82	3	32 -16 8	3	Relu – relu - relu	2	40 - 4	tanh - softmax	Adam
0.24	0.93	1	32	1	Relu	2	60 - 4	tanh - softmax	SGD
0.18	0.92	2	48 -16	2	Relu- tanh	2	80 -4	tanh - softmax	SGD
0.07	0.97	3	64 - 16	2	Relu - tanh	2	600-4	Tanh - softmax	SGD

Test result on a subset of data

```
Test data
[[0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

```
Left
Left
Straight
Left
Right
Stop
Straight
Stop
Right
Stop
```

```
Predicted data
[[0.00905735 0.8950606  0.02062939 0.07525276]
 [0.00686694 0.9555358  0.00848328 0.02911392]
 [0.95618474 0.01378191 0.01650803 0.0135252 ]
 [0.00770855 0.9456203  0.00437578 0.0422953 ]
 [0.02266534 0.01135937 0.95822126 0.00775406]
 [0.01628553 0.0461893  0.00551128 0.9320139 ]
 [0.9722697  0.00899625 0.00758969 0.01114437]
 [0.0228079  0.02680148 0.00558146 0.94480914]
 [0.02447056 0.01274472 0.95447665 0.00830801]
 [0.01570828 0.04959484 0.01206484 0.922632  ]]
```

```
Left
Left
Straight
Left
Right
Stop
Straight
Stop
Right
Stop
```

TRAINING AND RESULT

Actuator value prediction for path following

```
test_size           := 0.1
keep_prob           := 0.5
nb_epoch            := 20
samples_per_epoch   := 250
batch_size          := 40
save_best_only      := True
learning_rate       := 0.001
--------------------------------
```

Layer (type)	Output Shape	Param #
lambda_1 (Lambda)	(None, 120, 160, 3)	0
conv2d_1 (Conv2D)	(None, 116, 156, 16)	1216
max_pooling2d_1 (MaxPooling2	(None, 58, 78, 16)	0
conv2d_2 (Conv2D)	(None, 56, 76, 32)	4640
max_pooling2d_2 (MaxPooling2	(None, 28, 38, 32)	0
conv2d_3 (Conv2D)	(None, 26, 36, 32)	9248
max_pooling2d_3 (MaxPooling2	(None, 13, 18, 32)	0
conv2d_4 (Conv2D)	(None, 11, 16, 48)	13872
max_pooling2d_4 (MaxPooling2	(None, 5, 8, 48)	0
flatten_1 (Flatten)	(None, 1920)	0
dense_1 (Dense)	(None, 600)	1152600
dense_2 (Dense)	(None, 300)	180300
dense_3 (Dense)	(None, 2)	602

```
Total params: 1,362,478
Trainable params: 1,362,478
Non-trainable params: 0
```

```
Loading model and weights
Model weights loaded....
```

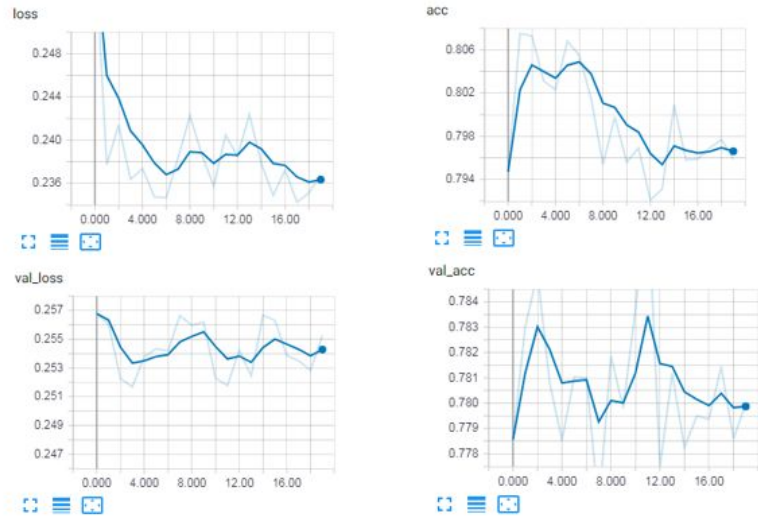
Best trained network parameters and architecture

TRAINING AND RESULT

Actuator value prediction for path following

Loss	Accuracy	CL	PL	FC-layer	Optimizer	Learning rate	Epochs	Data size	
0.46	0.51	11	6	4	Adam	0.001	10	2500	
0.38	0.55	6	4	4	SGD	0.001	10	4500	
0.28	0.58	5	4	4	SGD	0.001	10	5000	
0.38	0.68	4	4	3	SGD	0.0001	10	7000	
0.25	0.74	4(with lambda)	4	3	SGD	0.005	20	7000	

Training parameters in different network architectures



Loss and accuracy graphs, 25% validation loss, and 74% validation accuracy till 20th epoch

CONCLUSION

- Compacts the working of different modules in a system into a single neural net.
- End-to-end learning works well for comparatively smaller size of data, where data complexity and number of steps for traditional recognition is less.
- For image based problems, where number of representational features are more, the network architecture may grow in size.
- Used 2 different sensors for data acquisition for different tasks, and applied end-to-end learning for prediction.
- Using 2 different datasets for achieving the correct predictions, end-to-end learning shows better accuracy for object avoidance than path following.
- Ongoing research and studies prove that, learning systems consisting of less number of modules can show consistency and improve accuracy than training a larger system.

REFERENCES

- Dean A. Pomerleau, “ALVINN: An Autonomous Land Vehicle In a Neural Network.” January 1989. CMU-CS-89-107.
- Junqing Wei, Jarrod M. Snider, Junsung Kim, John M. Dolan, Raj Rajkumar and Bakhtiar Litkouhi, “Towards a Viable Autonomous Driving Research Platform”, 2013 IEEE Intelligent Vehicles Symposium (IV).
- Jesse Levinson, Jake Askelan, et al. “Towards Fully Autonomous Driving: Systems and Algorithms.” 2011 IEEE Intelligent Vehicles Symposium (IV).
- Avinash Nehemiah, Arvind Jayaraman. “Deep Learning for Automated Driving with MATLAB.” Accelerated Computing (https://devblogs.nvidia.com/accelerated-computing).
- Luca Caltagirone, Mauro Bellone, Lennart Svensson, Mattias Wahde.” LIDAR-based Driving Path Generation Using Fully Convolutional Neural Network.” arXiv: 1703.08987v2, April, 2017.
- Matthew Browne, Saeed Shiry Ghidary, “Convolutional neural networks for image processing: an application in robot vision”. The 16th Australian Joint Conference on Artificial Intelligence-AI’03 Dec 2003,Perth,Australia.
- C. Chen, A. Seff, A. Kornhauser and J. Xiao, “Deepdriving: Learning affordance for direct perception in autonomous driving”, in Proceedings of the IEEE International Conference on Computer Vision, 2015.
- M. Bojarski, D. Tel Testa, D. Dworakowski et al., “End-to-End learning for self- driving cars.” arXiv preprint arXiv:1604.07316, 2016.
- Andrej Karpathy, Justin Johnson, Li Fei-Fei .” Visualising and Understanding Recurrent Networks”. arXiv: 1506.02078v2, 17 Nov 2015.
- S. Hochreiter, J. Schmidhuber,” Long Short-Term Memory”. Neural Computation 9(8): 1735-1780,1997.
- Dan C. Ciresen, Ueli Meier, J. Schmidhuber et al.” Flexible, High Performance Convolutional Neural Network for Image Classification”. Proceedings of the 22nd International Joint Conference on Artificial Intelligence.
- Luca Caltagirone, Samuel Scheidegger, L.Svensson, M. Wahde. “Fast-LIDAR-based Road Detection Using Fully Convolutional Neural Networks”.arXiv:/1703.03613 29th March,2017.
- Wende Zhang. “Lidar based Road and Road-Edge Detection”, IEEE.Tobias Glasmachers Institute for Neural Computation, Ruhr-University Bochum, Germany. arXiv:1704.08305v1 [cs.LG] , 26 April 2017.