

DEEP LEARNING MODEL FOR SELF DRIVING CARS

GUIDE'S NAME: PROF. JANAKI M.

PLACE OF WORK: VIT, VELLORE

PROJECT BY: KHYATI GUPTA

PROBLEM DEFINITION

- ▶ Take the video of the front scenario using the front camera on the car and process it through a system to determine the output. In the scope of this project, the output is the steering angle. The video is received as a sequence of images and the output is a sequence of steering angle corresponding to the images. The model is to be made in such a way so as to reduce the problem of overfitting and thus making a more generic model.

OBJECTIVES

- ▶ To apply CNN along with 'dropout' to make a more efficient end-to-end learning model to predict the steering angle for a self driving car and overcoming the problem of overfitting.

INTRODUCTION

Self-driving cars, also called 'driverless' cars, are cars that perform the various functions of driving a car by itself. This includes steering, accelerating, braking, amongst others. The input is the images and distances retrieved using the different sensors on the car.

Convolution Neural Networks (CNN) became popular in 2012 and have ever since completely transformed the pattern recognition.

Before CNNs, for image processing, feature extraction and classification was needed to be hand-crafted. However, CNN learns features on its own from the training samples.

NVidia first came up with an algorithm to steer an automated vehicle, after years of research, in 2016.

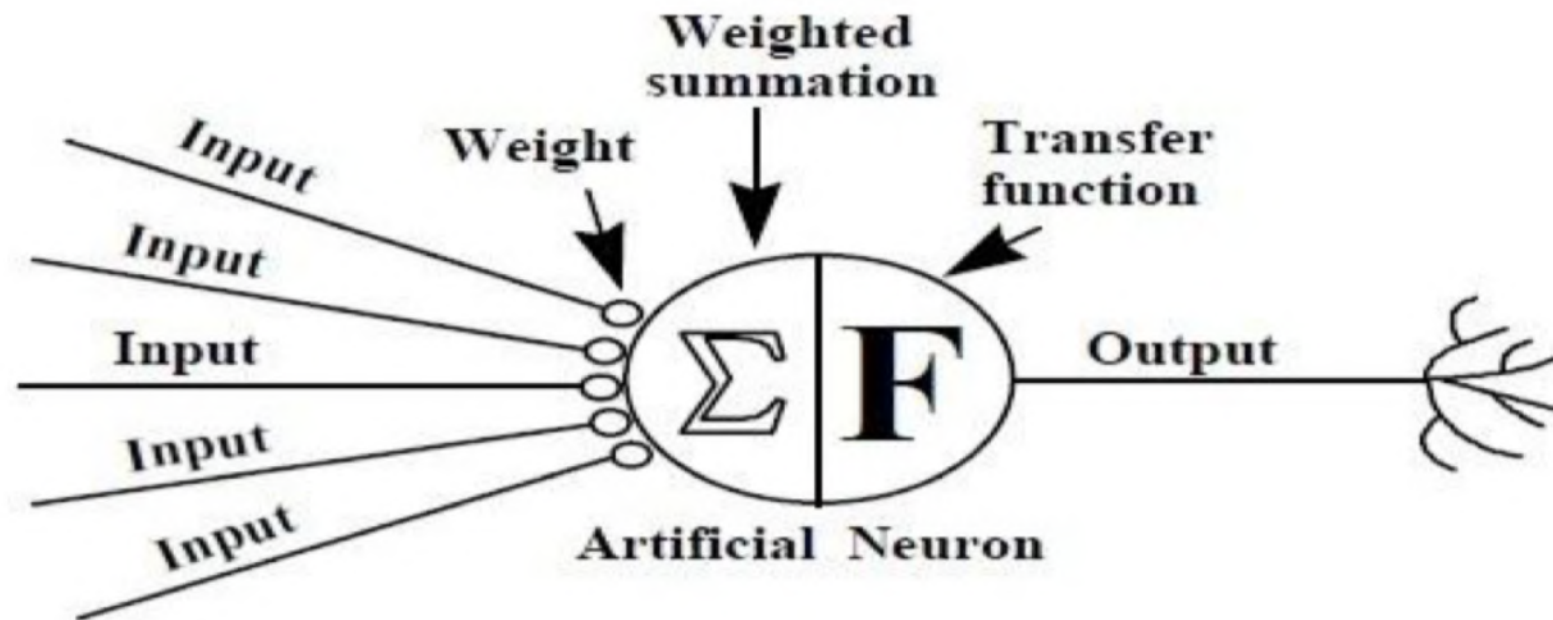
INTRODUCTION

- ▶ CNN is a Deep Learning algorithm. Especially in Deep Learning, there is a huge tendency to 'overfit'. Overfitting is when the model learns too much to adjust to just the training data and thus is not generic enough. There are different techniques to overcome this problem, one being 'dropout'.

WHAT IS ARTIFICIAL NEURAL NETWORK (ANN)?

ANN is like copying the brain's network on the computers. It consists of neurons that have input and output, and are connected by weights. These weights control the importance the connection has. The network is made of 2 or more layers: first being the input layer and the last one being the output layer. All the layers in between the input and the output layers are called as the 'hidden' layers. Every layer has an 'activation function' which is responsible to generate the output from the neurons of that layer. Some of them are step function, sigmoid function, tan hyperbolic function, ReLU, etc.

A TYPICAL ARTIFICIAL NEURON:



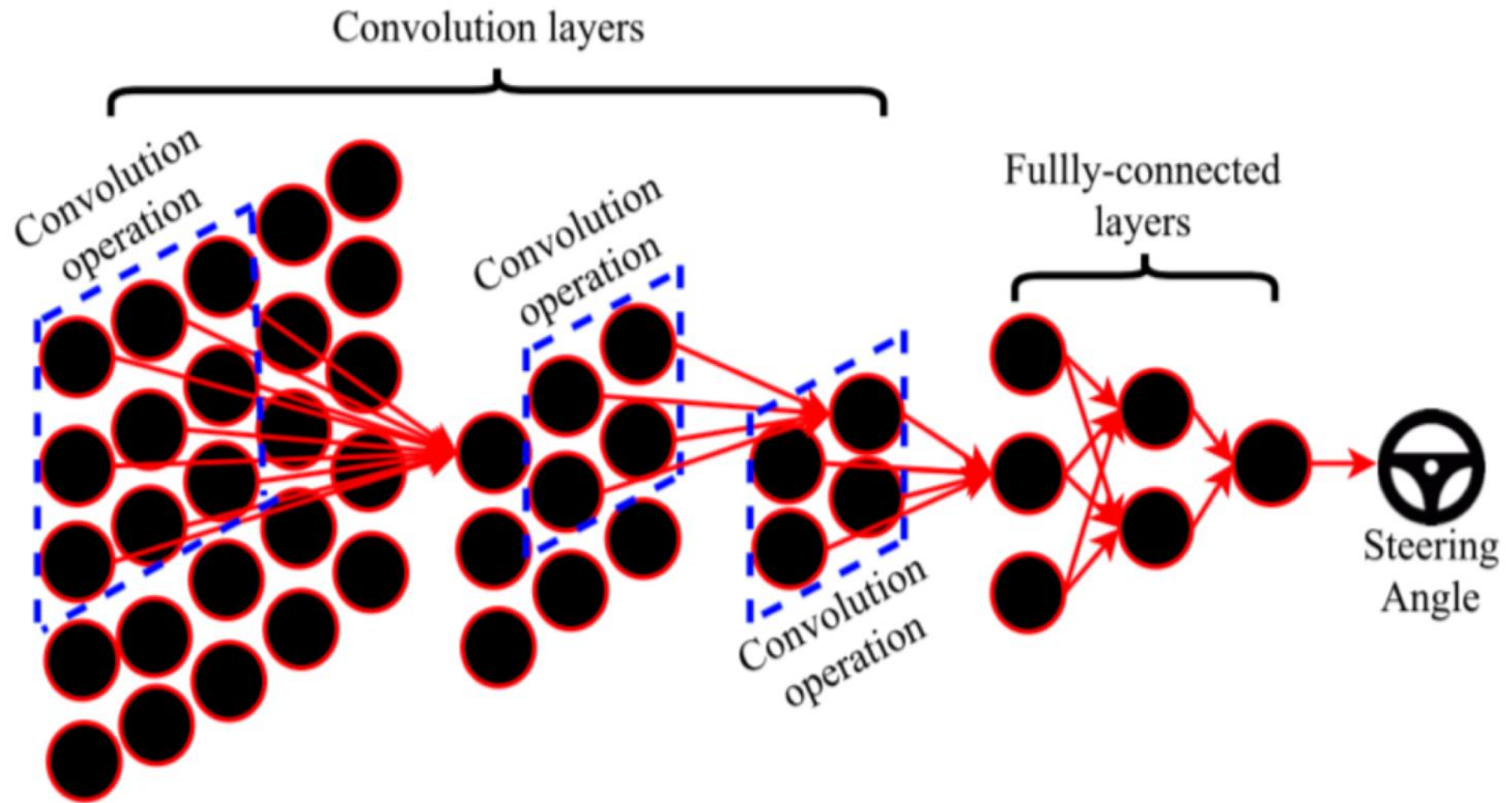
DEEP NEURAL NETWORKS (DNN)

Networks that have multiple hidden layers (generally 2 or more) are called Deep Neural Networks. A hidden layer is any layer in a neural network between the input and the output layer.

WHAT IS CNN?

CNNs are fundamentally different from DNN (deep NN) in that they contain one or more convolution layers. In it, each input image will be passed through a series of convolution layers with filters or Kernels, generally followed by a pooling layer, then by fully connected layers (FC) and a loss layer. Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel/ Feature Map. CNN has revolutionised image processing and pattern recognition.

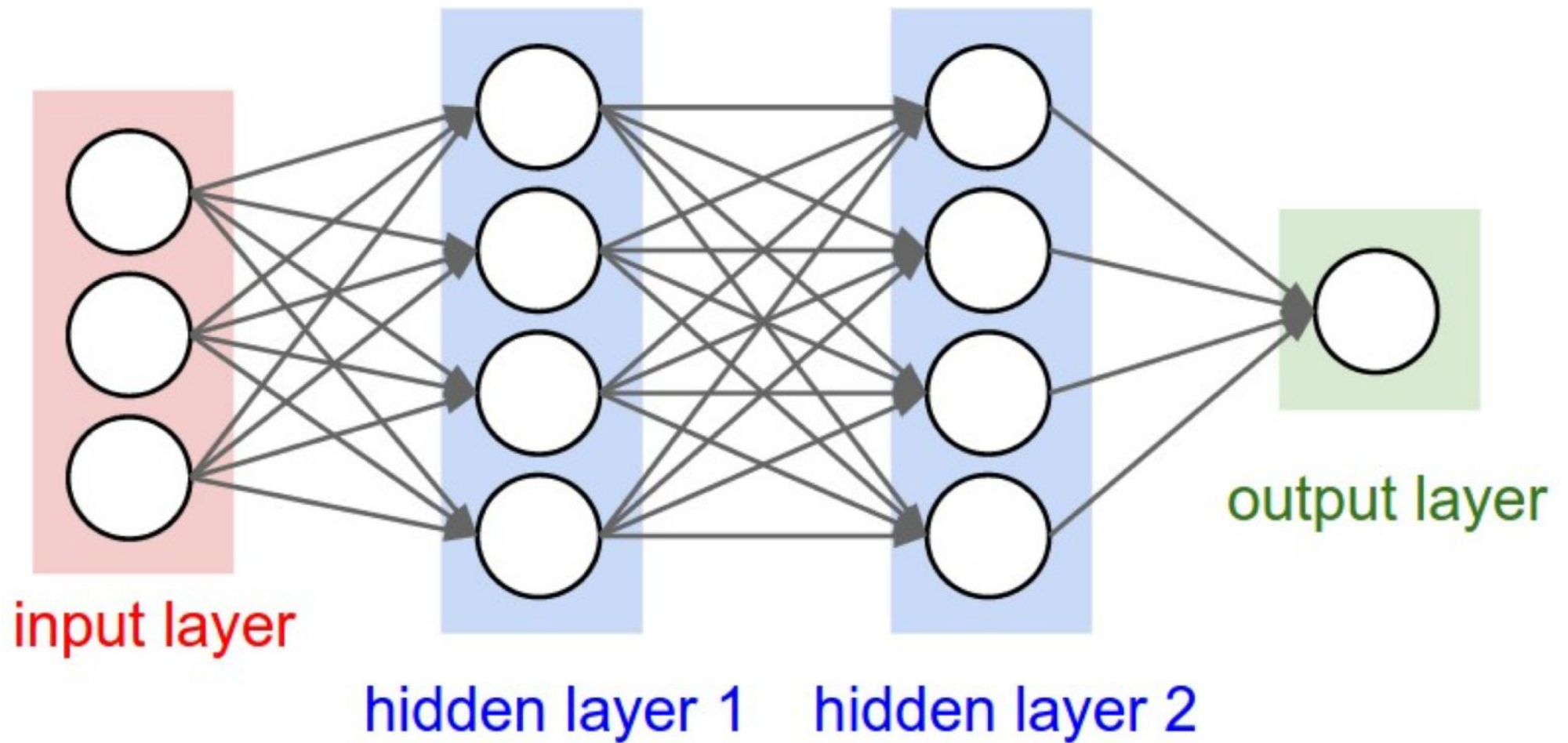
INTRODUCTION



FULLY CONNECTED LAYERS

Fully connected layers are layers in which each of the neurons of one layer are connected to all the neurons of the next layer. These are generally present after the convolution layers in CNN.

FULLY CONNECTED LAYERS

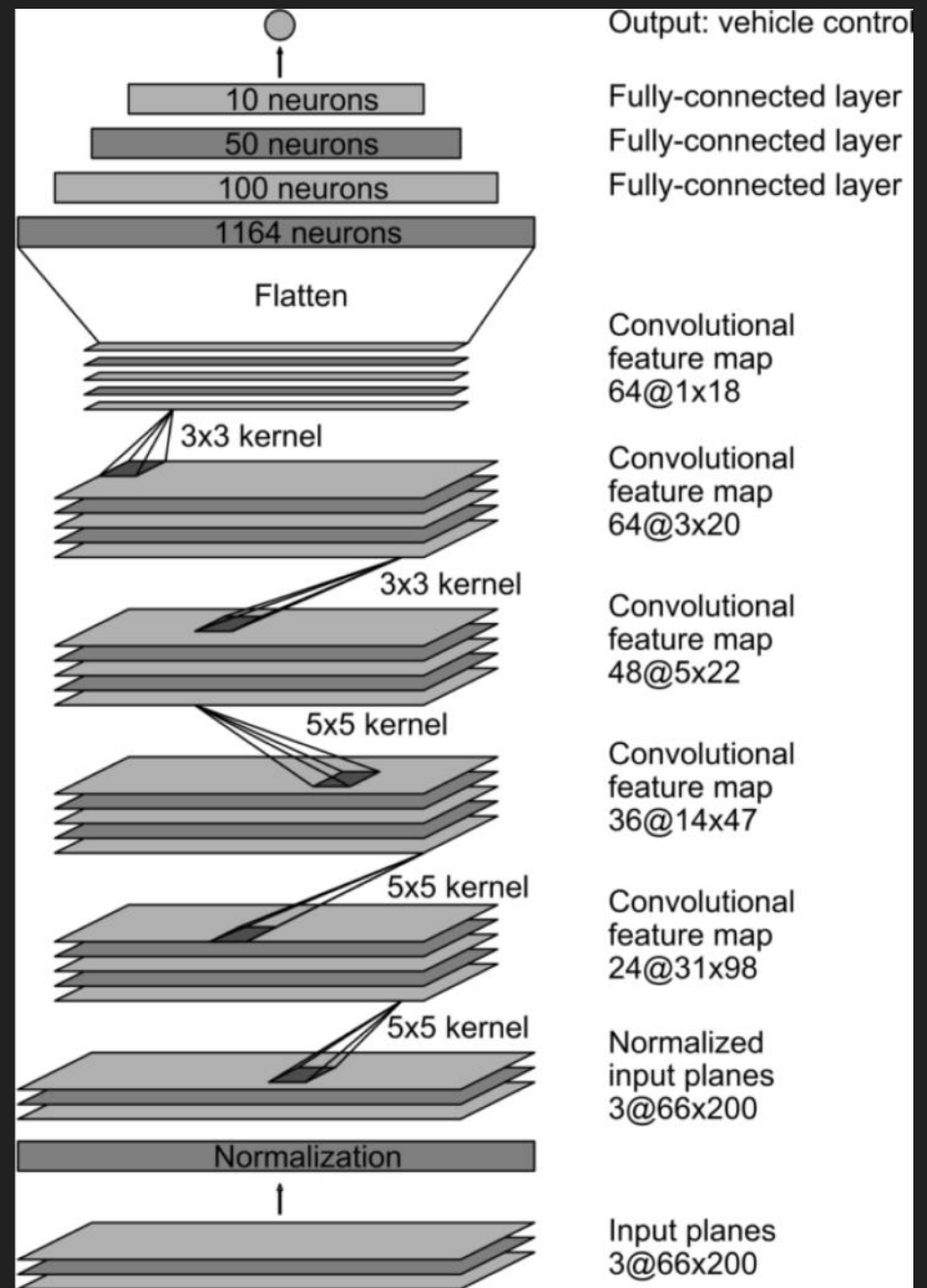


METHODOLOGY

- ▶ Select CNN architecture and training process parameters.
- ▶ Collect data to train and test the CNN (video from front camera and the corresponding steering angles at that time).
- ▶ Make the model and train and test it.
- ▶ Find area of improvement and make possible corrections.

NVIDIA ARCHITECTURE:

CNN architecture. It consists of a normalisation layer, 5 convolution layers, 3 fully connected layers.

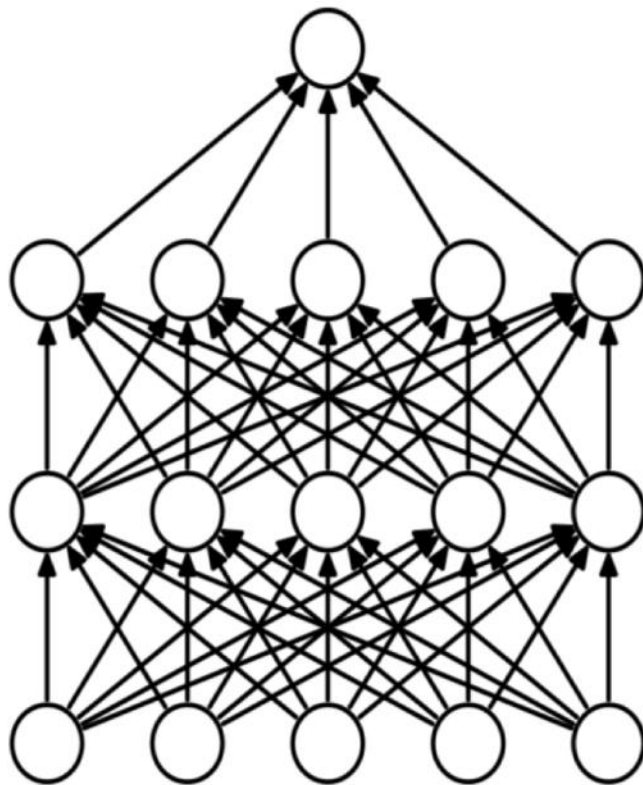


DROPOUT:

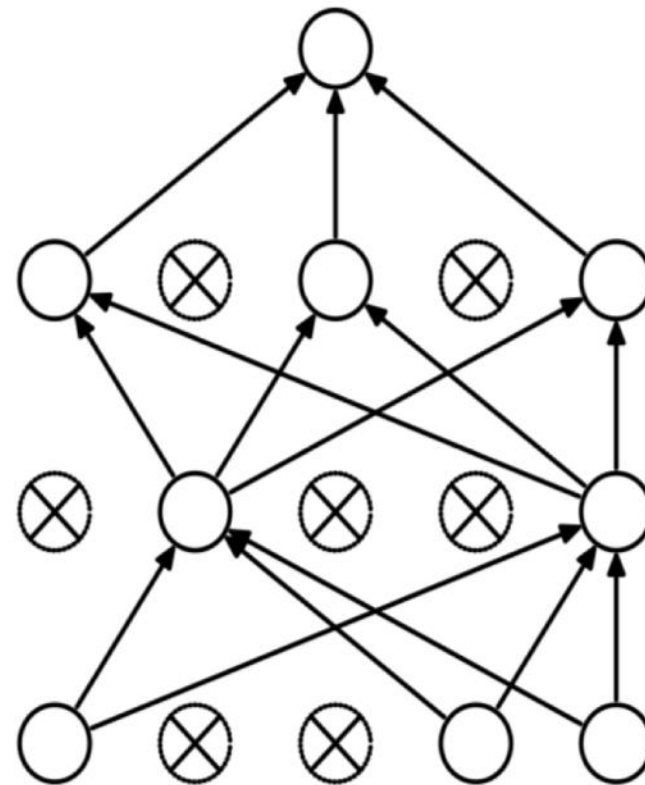
Dropout is a technique used to overcome the problem of overfitting, which is a very common issue, specially in deep neural networks. Dropout means to literally temporarily drop out units from the NN. When we drop a unit, we temporarily remove all its connections, both to and fro from the neuron. The selection of the unit is made at random, thus not partial to any unit.

We decide a 'keep probability' which can be thought of as the percentage of neurons to be kept in the network at any time. This probability is generally closer to 1 and it chosen to be anywhere from 0.5 to 1. So if the keep probability is 0.8, 80% of the neurons are kept in the network while training, or 20% are dropped out. Every time the NN is run, these neurons are chosen at random, thus every neuron has ' p ' (keep probability) probability of staying in the network. This can be thought of as making a 'thinner' network for every run. Dropout is not applied while testing.

NN BEFORE AND AFTER DROPOUT



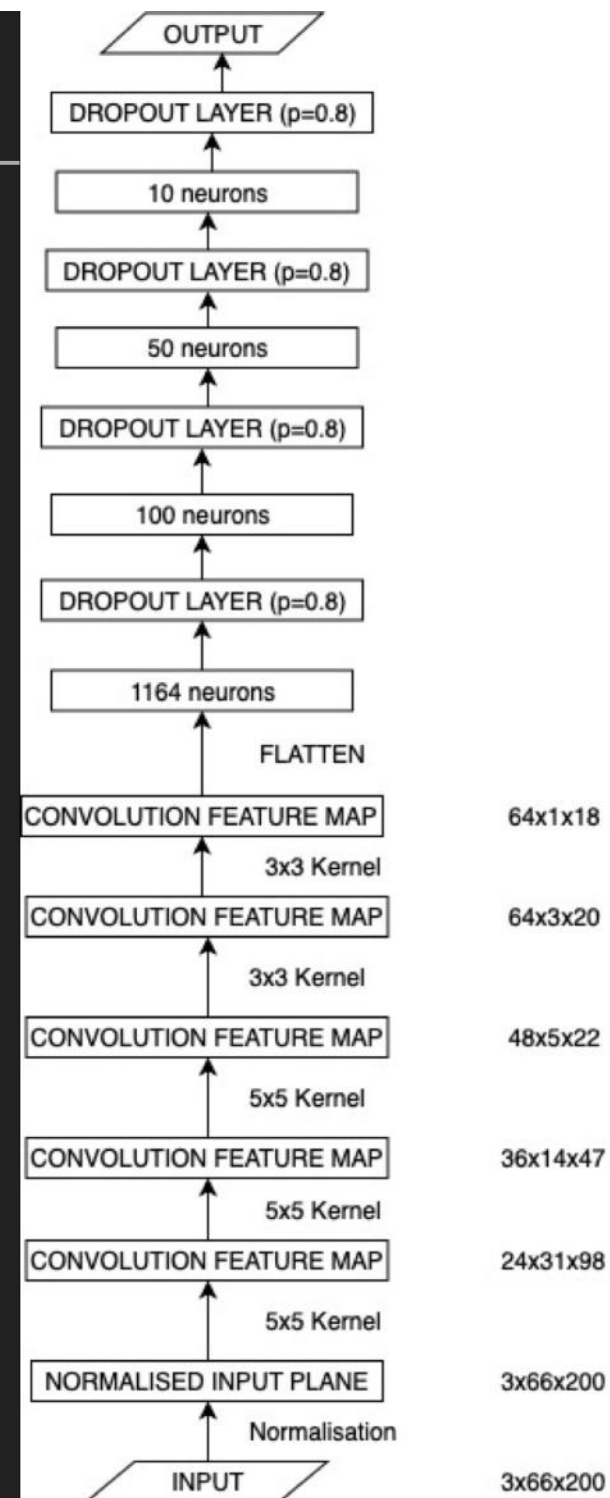
(a) Standard Neural Net



(b) After applying dropout.

PROPOSED ARCHITECTURE:

- Dropout layers with keep probability of 0.8 after every fully connected layer to reduce overfitting



TRAINING PARAMETERS

- ▶ Optimiser used - Adam
- ▶ Activation units - ReLU
- ▶ Loss calculated as Mean Square Error with L2 Norm Regularisation.
- ▶ Training performed using Mini Batch SGD

COLLECTING THE DATA

- ▶ The data collected is from SullyChen's GitHub and consists of 25min long drive across his city. There are a total of 45406 images in the dataset (at the rate of 30 frames per second). The data is divided into training and testing data in 80:20 ratio. So the first 80% of the data (20 mins) is used for training and validation while the last 20% (5 mins) data is used for testing the model.

TRAINING ERRORS FOR MODEL WITHOUT DROPOUT

```
Epoch: 0, Step: 0, Loss: 6.46984
WARNING:tensorflow:*****
WARNING:tensorflow:TensorFlow's V1 checkpoint format has been deprecated.
WARNING:tensorflow:Consider switching to the more efficient V2 format:
WARNING:tensorflow:  `tf.train.Saver(write_version=tf.train.SaverDef.V2)`
WARNING:tensorflow:now on by default.
WARNING:tensorflow:*****
Epoch: 0, Step: 10, Loss: 6.43449
Epoch: 0, Step: 20, Loss: 6.24551
Epoch: 0, Step: 30, Loss: 6.25654
Epoch: 0, Step: 40, Loss: 6.23714
Epoch: 0, Step: 50, Loss: 6.23179
Epoch: 0, Step: 60, Loss: 6.10271
Epoch: 0, Step: 70, Loss: 5.99958
Epoch: 0, Step: 80, Loss: 6.05258
Epoch: 0, Step: 90, Loss: 6.02651
Epoch: 0, Step: 100, Loss: 5.83032
```

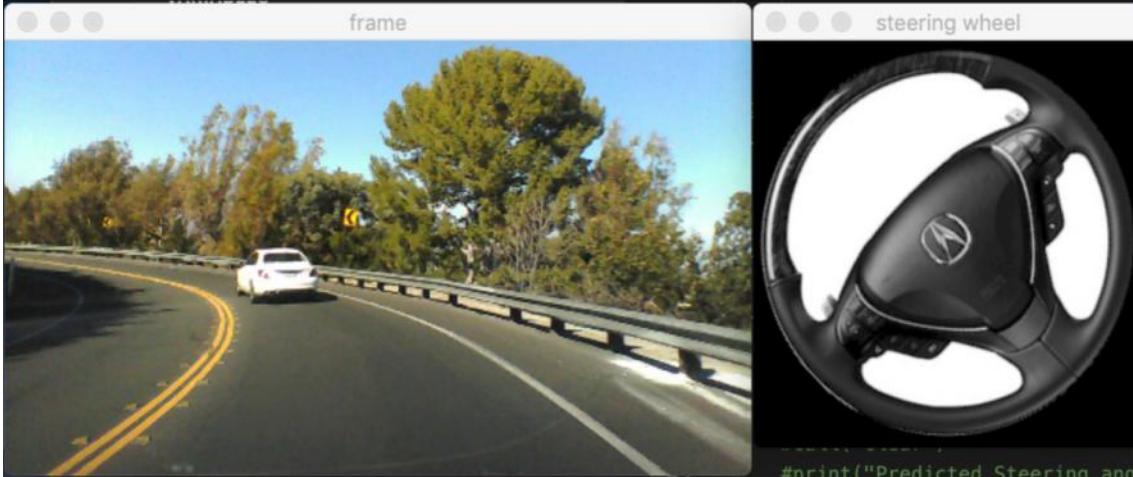
```
Epoch: 29, Step: 3290, Loss: 0.126667
Epoch: 29, Step: 3300, Loss: 0.144321
WARNING:tensorflow:*****
WARNING:tensorflow:TensorFlow's V1 checkpoint format has been deprecated.
WARNING:tensorflow:Consider switching to the more efficient V2 format:
WARNING:tensorflow:  `tf.train.Saver(write_version=tf.train.SaverDef.V2)`
WARNING:tensorflow:now on by default.
WARNING:tensorflow:*****
Epoch: 29, Step: 3310, Loss: 0.193436
Epoch: 29, Step: 3320, Loss: 0.158375
Epoch: 29, Step: 3330, Loss: 0.114582
Epoch: 29, Step: 3340, Loss: 0.126732
Epoch: 29, Step: 3350, Loss: 0.091271
Model saved in file: ./save\model.ckpt
Run the command line:
```

TRAINING ERRORS FOR MODEL WITH DROPOUT

```
Use `tf.global_variables_initializer` instead.  
Epoch: 0, Step: 0, Loss: 8.4389  
WARNING:tensorflow:*****  
WARNING:tensorflow:TensorFlow's V1 checkpoint format has been deprecated.  
WARNING:tensorflow:Consider switching to the more efficient V2 format:  
WARNING:tensorflow:  `tf.train.Saver(write_version=tf.train.SaverDef.V2)`  
WARNING:tensorflow:now on by default.  
WARNING:tensorflow:*****  
Epoch: 0, Step: 10, Loss: 6.23978  
Epoch: 0, Step: 20, Loss: 6.11781  
Epoch: 0, Step: 30, Loss: 6.06508
```

```
WARNING:tensorflow:Consider switching to the more efficient V2 format:
WARNING:tensorflow:  `tf.train.Saver(write_version=tf.train.SaverDef.V2)`
WARNING:tensorflow:now on by default.
WARNING:tensorflow:*****
Epoch: 29, Step: 3310, Loss: 0.287191
Epoch: 29, Step: 3320, Loss: 0.194609
Epoch: 29, Step: 3330, Loss: 0.125513
Epoch: 29, Step: 3340, Loss: 0.150976
Epoch: 29, Step: 3350, Loss: 0.135028
Model saved in file: ./save\model.ckpt
```


TESTING DATA OUTPUT WITHOUT DROPOUT LAYER



The screenshot shows a Python IDE with two windows: 'frame' and 'steering wheel'. The 'frame' window displays a car driving on a road. The 'steering wheel' window displays a steering wheel image. The code in the background is a Python script that processes the frame and steering wheel images to predict the steering angle.

```
def in radians is used as the output
    (i)) * scipy.pi / 180)

(i))

"driving_dataset/" + str(i) + ".jpg", mode="RGB")
l_image[-150:, [66, 200]] / 255.0
t={model.x: [image], model.keep_prob: 1.0}[0][0] * 180.6

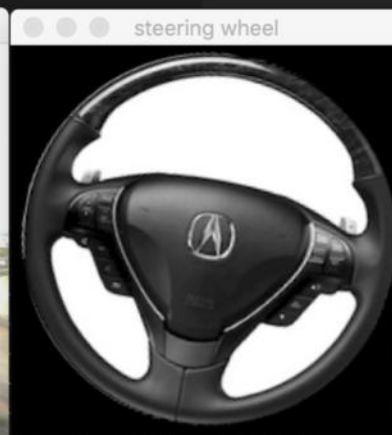
#print("Predicted Steering angle: " + str(degrees))
print("Steering angle: " + str(degrees) + " (pred)\t" + str(ys[i]*180/scipy.pi) + " (act
cv2.imshow("frame", cv2.cvtColor(full_image, cv2.COLOR_RGB2BGR))
#make smooth angle transitions by turning the steering wheel based on the difference of
#and the predicted angle
smoothed_angle += 0.2 * pow(abs((degrees - smoothed_angle)), 2.0 / 3.0) * (degrees - sm
M = cv2.getRotationMatrix2D((cols/2,rows/2),-smoothed_angle,1)
dst = cv2.warpAffine(img,M,(cols,rows))
cv2.imshow("steering wheel", dst)
```

CALL STACK PAUSED ON PAUSE
run_dataset.py 39:1

BREAKPOINTS
☐ Raised Exceptions
☒ Uncaught Exceptions

PROBLEMS **OUTPUT** **DEBUG CONSOLE** **TERMINAL**

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL
		Steering angle: -43.64404477745147 (pred)	-76.44 (actual)
		Steering angle: -45.22256638143853 (pred)	-75.63 (actual)
		Steering angle: -46.04411861028059 (pred)	-74.32 (actual)
		Steering angle: -48.74642489878819 (pred)	-73.61 (actual)
		Steering angle: -46.43006869461592 (pred)	-72.5 (actual)
		Steering angle: -48.90918489161501 (pred)	-71.7 (actual)
		Steering angle: -49.48266464980221 (pred)	-70.08 (actual)
		Steering angle: -47.77504564031273 (pred)	-69.07999999999999 (actual)
		Steering angle: -47.58521419271057 (pred)	-68.97 (actual)
		Steering angle: -45.02008200839892 (pred)	-68.97 (actual)
		Steering angle: -42.86080990985319 (pred)	-68.37 (actual)



```
16
17 smoothed_angle = 0
18
19
20 #read data.txt
21 xs = []
22 ys = []
23 with open("driving_dataset/data.txt") as f:
24     for line in f:
25         #read line and split it into steering angle and speed
```

CALL STACK

<module> run_dataset.py 39:1

BREAKPOINTS

☐ Raised Exceptions

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Steering angle: 8.649172706371989 (pred) 42.959999999999994 (actual)
Steering angle: 8.597778947962228 (pred) 43.869999999999999 (actual)
Steering angle: 8.821724629147667 (pred) 44.77 (actual)
Steering angle: 11.636814416325103 (pred) 45.579999999999999 (actual)
Steering angle: 10.331358652713273 (pred) 45.78 (actual)
Steering angle: 9.795413345336236 (pred) 45.98 (actual)
Steering angle: 11.555712750120371 (pred) 46.29 (actual)
Steering angle: 10.565103094036742 (pred) 46.29 (actual)
Steering angle: 10.989683874544014 (pred) 46.29 (actual)
Steering angle: 11.20397508333764 (pred) 46.29 (actual)
Steering angle: 10.65775375635432 (pred) 46.29 (actual)
```

1: Python Debug Console



```
angle in radians is used as the output
t()[1]) * scipy.pi / 180)

str(i))

):
nd("driving_dataset/" + str(i) + ".jpg", mode="RGB")
full_image[-150:], [66, 200]) / 255.0
dict={model.x: [image], model.keep_prob: 1.0})[0][0] * 180.0

#print("Predicted Steering angle: " + str(degrees))
44 print("Steering angle: " + str(degrees) + " (pred)\t" + str(ys[i]*180/scipy.pi) + " (act
45 cv2.imshow("frame", cv2.cvtColor(full_image, cv2.COLOR_RGB2BGR))
46 #make smooth angle transitions by turning the steering wheel based on the difference of
47 #and the predicted angle
48 smoothed_angle += 0.2 * pow(abs((degrees - smoothed_angle)), 2.0 / 3.0) * (degrees - smc
49 M = cv2.getRotationMatrix2D((cols/2,rows/2),-smoothed_angle,1)
50 dst = cv2.warpAffine(img,M,(cols,rows))
51 cv2.imshow("steering wheel", dst)
```

CALL STACK

PAUSED ON PAUSE

<module> run_dataset.py 39:1

BREAKPOINTS

- ☐ Raised Exceptions
- ☒ Uncaught Exceptions

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Steering angle: 4.037204863197239 (pred) 9.68 (actual)
Steering angle: 4.166629690892988 (pred) 9.979999999999999 (actual)
Steering angle: 3.9845342862995086 (pred) 0.0 (actual)
Steering angle: 3.6869493474643997 (pred) 10.29 (actual)
Steering angle: 3.14003710415785 (pred) 10.59 (actual)
Steering angle: 3.3253593462473363 (pred) 10.79 (actual)
Steering angle: 3.4272179573397366 (pred) 11.089999999999998 (actual)
Steering angle: 3.627068652124075 (pred) 11.290000000000001 (actual)
Steering angle: 4.126951414457091 (pred) 11.39 (actual)
Steering angle: 3.5905057957233506 (pred) 11.6 (actual)
Steering angle: 2.8158265938341507 (pred) 11.699999999999998 (actual)
```

1: Python Debug Console



AVERAGE TEST LOSS WITHOUT DROPOUT LAYERS

- ▶ The average test loss expressed as average deviation in degrees for this model is 13.812

```
Steering angle: 18.62599416851647 (pred)      2.22 (actual)
Steering angle: 18.62599416851647 (pred)      2.22 (actual)
Steering angle: 18.62599416851647 (pred)      2.22 (actual)
Steering angle: 18.62599416851647 (pred)      2.22 (actual)
Steering angle: 18.62599416851647 (pred)      2.22 (actual)
Steering angle: 18.62599416851647 (pred)      2.22 (actual)
Steering angle: 18.62599416851647 (pred)      2.22 (actual)
Steering angle: 18.62599416851647 (pred)      0.0 (actual)
Average loss = 13.811814953226108
Terminated: 15
(hbase) Khyat@MacBook-Pro:Self-Driving-Cars$ khyat@ubuntu$ █
```

TESTING DATA OUTPUT WITH THE DROPOUT LAYERS

The screenshot displays a Python IDE interface with a dark theme. The main editor window shows the code for `run_dataset.py`. The code includes a `Saver` object, image loading, and a loop for processing steering angles. The `CALL STACK` and `BREAKPOINTS` panels are visible on the left. The `TERMINAL` panel at the bottom right shows the output of the program, which is a list of steering angles (predicted and actual values).

Code in `run_dataset.py`:

```
10 saver = tf.train.Saver()
11 saver.save(sess, "save/model.ckpt")
12 img = cv2.imread('steering_wheel_image.jpg',0)
13 ls = img.shape
14 d_angle = 0
15
16 #get number of images
17 num_images = len(xs)
18
19 i = math.ceil(num_images*0.8)
20 print("Starting frameofvideo:" +str(i))
21
22 while True:
23     i = math.ceil(num_images*0.8)
24     print("Starting frameofvideo:" +str(i))
25     i = i + 1
26     i = i % num_images
27     i = i + 1
28     i = i % num_images
29     i = i + 1
30     i = i % num_images
31     i = i + 1
32     i = i % num_images
33     i = i + 1
34     i = i % num_images
35     i = i + 1
36     i = i % num_images
37     i = i + 1
38     i = i % num_images
39     i = i + 1
40     i = i % num_images
41     i = i + 1
42     i = i % num_images
43     i = i + 1
44     i = i % num_images
45     i = i + 1
46     i = i % num_images
47     i = i + 1
48     i = i % num_images
49     i = i + 1
50     i = i % num_images
51     i = i + 1
52     i = i % num_images
53     i = i + 1
54     i = i % num_images
55     i = i + 1
56     i = i % num_images
57     i = i + 1
58     i = i % num_images
59     i = i + 1
60     i = i % num_images
61     i = i + 1
62     i = i % num_images
63     i = i + 1
64     i = i % num_images
65     i = i + 1
66     i = i % num_images
67     i = i + 1
68     i = i % num_images
69     i = i + 1
70     i = i % num_images
71     i = i + 1
72     i = i % num_images
73     i = i + 1
74     i = i % num_images
75     i = i + 1
76     i = i % num_images
77     i = i + 1
78     i = i % num_images
79     i = i + 1
80     i = i % num_images
81     i = i + 1
82     i = i % num_images
83     i = i + 1
84     i = i % num_images
85     i = i + 1
86     i = i % num_images
87     i = i + 1
88     i = i % num_images
89     i = i + 1
90     i = i % num_images
91     i = i + 1
92     i = i % num_images
93     i = i + 1
94     i = i % num_images
95     i = i + 1
96     i = i % num_images
97     i = i + 1
98     i = i % num_images
99     i = i + 1
100    i = i % num_images
```

CALL STACK:

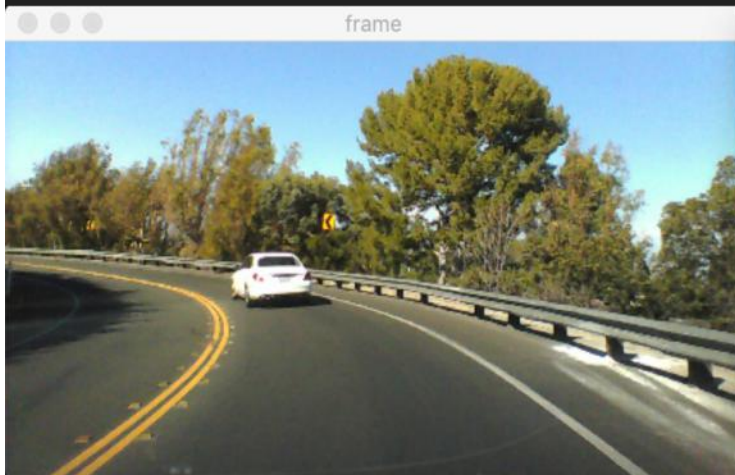
Module	File	Line
<module>	run_dataset.py	39:1
_run_code	runpy.py	85:1
_run_module_code	runpy.py	96:1
run_path	runpy.py	263:1

BREAKPOINTS:

- ☐ Raised Exceptions
- ☒ Uncaught Exceptions

TERMINAL:

```
Steering angle: 21.24995477214944 (pred) 28.34 (actual)
Steering angle: 22.864790784406168 (pred) 28.34 (actual)
Steering angle: 22.77834278762861 (pred) 28.34 (actual)
Steering angle: 21.213749646906496 (pred) 28.34 (actual)
Steering angle: 24.846279889045643 (pred) 28.44 (actual)
Steering angle: 25.00882814600819 (pred) 28.44 (actual)
Steering angle: 25.01902732598631 (pred) 28.44 (actual)
Steering angle: 28.549248164547837 (pred) 28.44 (actual)
Steering angle: 30.55323932740792 (pred) 28.44 (actual)
Steering angle: 29.544707254942175 (pred) 28.54 (actual)
Steering angle: 29.608931523707806 (pred) 28.44 (actual)
```



```
del.ckpt")  
wheel_image.jpg',0)
```

```
data.txt") as f:
```

```
22 |         xs.append("driving_dataset/" + line.split()[0])  
23 |         ys.append(float(line.split()[1]) * scipy.pi / 180)  
24 |  
25 |     #get number of images  
26 |     num_images = len(xs)  
27 |  
28 |  
29 |     i = math.ceil(num_images*0.8)  
30 |     print("Starting frameofvideo:" +str(i))  
31 |
```

CALL STACK

PAUSED ON PAUSE

<module>	run_dataset.py	33:1
_run_code	runpy.py	85:1
_run_module_code	runpy.py	96:1
run_path	runpy.py	263:1

BREAKPOINTS

- ☐ Raised Exceptions
- ☒ Uncaught Exceptions

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

1: Python Debug Console

Steering angle: -50.49293500541172 (pred)	-53.040000000000006 (actual)
Steering angle: -49.35678426339887 (pred)	-53.34 (actual)
Steering angle: -49.56562754255577 (pred)	-53.55 (actual)
Steering angle: -50.72840919214249 (pred)	-54.15 (actual)
Steering angle: -52.06861866579864 (pred)	-54.86 (actual)
Steering angle: -53.77453354309019 (pred)	-55.76 (actual)
Steering angle: -54.93051573935817 (pred)	-56.57 (actual)
Steering angle: -55.908895941732894 (pred)	-56.97 (actual)
Steering angle: -56.730731623425505 (pred)	-57.480000000000004 (actual)
Steering angle: -58.36075626884905 (pred)	-58.290000000000006 (actual)
Steering angle: -57.43451431550362 (pred)	-58.990000000000001 (actual)
Steering angle: -57.64823093463332 (pred)	-59.600000000000001 (actual)

AVERAGE TEST LOSS WITH DROPOUT

- ▶ The average test loss expressed as deviation in degrees for the model with dropout layers is 3.327, 10.5 degrees lesser than that without dropout model.

```
Steering angle: 0.5372797555861281 (pred) 2.22 (actual)
Steering angle: 0.4920055798684341 (pred) 2.22 (actual)
Steering angle: 0.5671486530742698 (pred) 2.22 (actual)
Steering angle: 0.739416962916834 (pred) 2.22 (actual)
Steering angle: 0.5327562493648755 (pred) 2.22 (actual)
Steering angle: 0.483554341348454 (pred) 2.22 (actual)
Steering angle: 0.43468975396358117 (pred) 2.22 (actual)
Steering angle: 0.6795118283058851 (pred) 0.0 (actual)
Average loss = 3.32730183185685
Terminated: 15
(base) Khyatis-MacBook-Pro:Self Driving Cars khyatigupta$
```

THANK YOU!

REFERENCES

1. Manish Mishra, Monika Srivastava, “A View of Artificial Neural Network”, IEEE International Conference on Advances in Engineering & Technology Research (ICAETR - 2014), 2014.
2. Daniel Dworakowski, Mathew Monfort, Urs Muller, Jiakai Zhang, “End to End Learning for Self-Driving Cars”, arXiv:1604.07316v1 [cs.CV], CoRR, vol. abs/1604.07316, April 2016.
3. Neena Aloysius and Geetha M, “A Review on Deep Convolutional Neural Networks”, International Conference on Communication and Signal Processing, April 2017.
4. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, Journal of Machine Learning Research 15, 2014.
5. <https://github.com/SullyChen/Autopilot-TensorFlow>