SELF DRIVING RASPBERRY PI CAR

RAJESH THANASEELAN

AUTONOMOUS CAR - OVERVIEW

- Autonomous Car:
  - A driverless vehicle with transportation features of a traditional car.
- Scope of this Project:
  - A Simple self driving car which can keep within Road
  - Capable of detecting objects.

POPULAR APROACHES

1) RADAR/LIDAR/MAPS - SENSOR based Approach

Pros – Accurate, Explainable

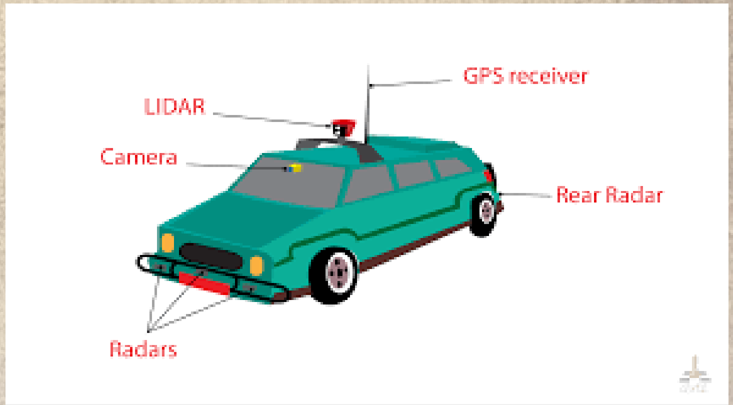
Cons – No learning involved and can't improve over time.

Expensive

2) Image Based Approach (Vision Sensors - Camera and Deep Learning)

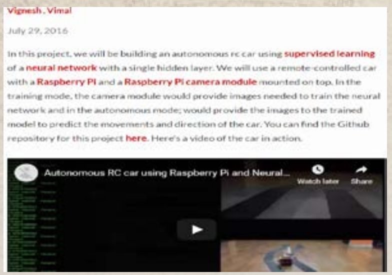
Pros – Images are easy to capture, Data is large, Road are designed for humans eyes and camera image can represent it. Cheap to implement.

Cons – Not Explainable and Consistent



INSPIRATION

https://pythonprogramming.net/ - https://www.youtube.com/user/sentdex


Vignesh . Vimal

July 29, 2016

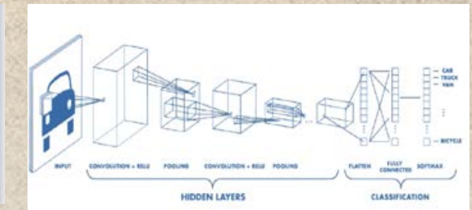
In this project, we will be building an autonomous rc car using supervised learning of a neural network with a single hidden layer. We will use a remote-controlled car with a Raspberry PI and a Raspberry PI camera module mounted on top. In the training mode, the camera module would provide images needed to train the neural network and in the autonomous mode; would provide the images to the trained model to predict the movements and direction of the car. You can find the Github repository for this project here. Here's a video of the car in action.

It uses the following equipment:

- Raspberry Pi
- Pi Camera Module
- PWM Motor Control
- TensorFlow
- OpenCV

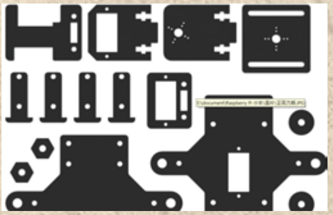
COMPONENTS

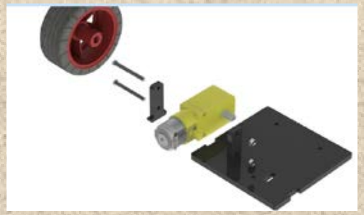
- Sun Founder Smart Video Car Kit V2.0
- Raspberry Pi 3
- Pi Camera V2
- Python 3.4
- Open CV2
- Machine Learning/Deep Learning – Convolution Neural Networks
- Tensorflow(1.4)/ TFLearn
- Laptop - Acer Predator (NVIDIA GTX 1060)

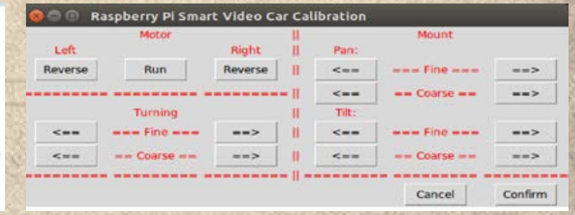


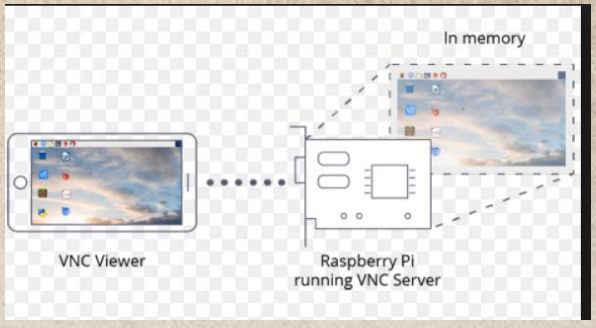
DESIGN PROCESS

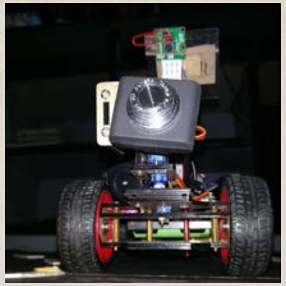
- Assemble the Sun founder Car.
- Setup Raspberry Pi (Jessie)
- Integrate PI, Pi Cam and Configure Sun founder Pi Client and Server for Python 3.4
- Define Strategy for Training ,Testing and Deployment.
- Setup Python Dependencies (3.4)
- Install Open CV on Raspberry Pi (for Image Processing)
- Install Tensor flow Raspberry Pi (for Deep Learning)
- Setup Road and Lanes
- Calibrate the Car

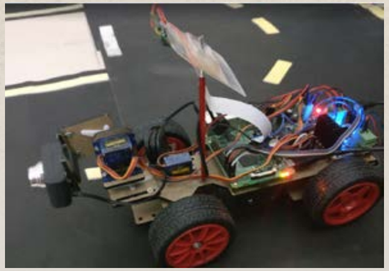










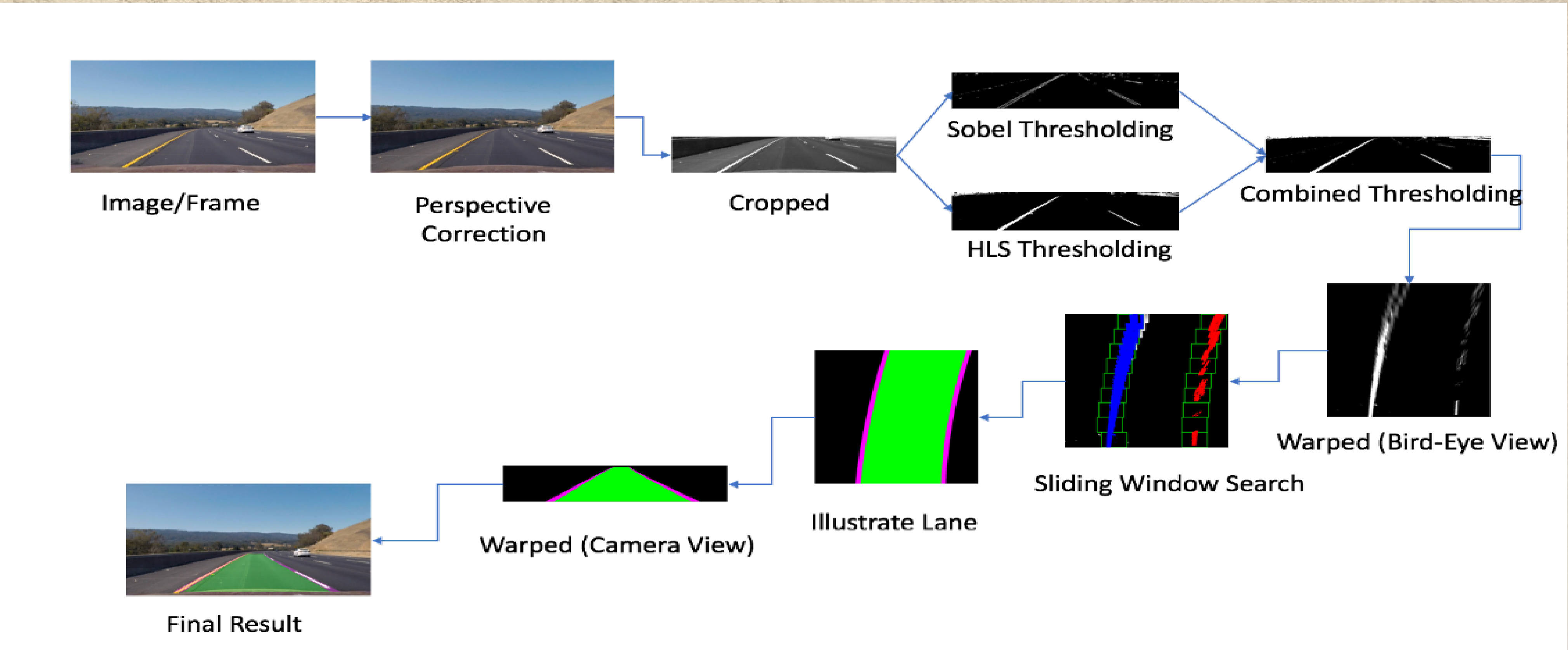


OPEN CV IMAGE PROCESSING

- Sobel Edge Detection / Thresholding (To find the Road and object edges)
- Perspective Transform (To transform the Road Lane to process)
- Sliding Window (Find the consecutive pixels to form Lane)
- Radius of Curvature. (Find the Curving Radius of the Road to determine path)
- Hough Transformation (converting points in the xy space to lines)

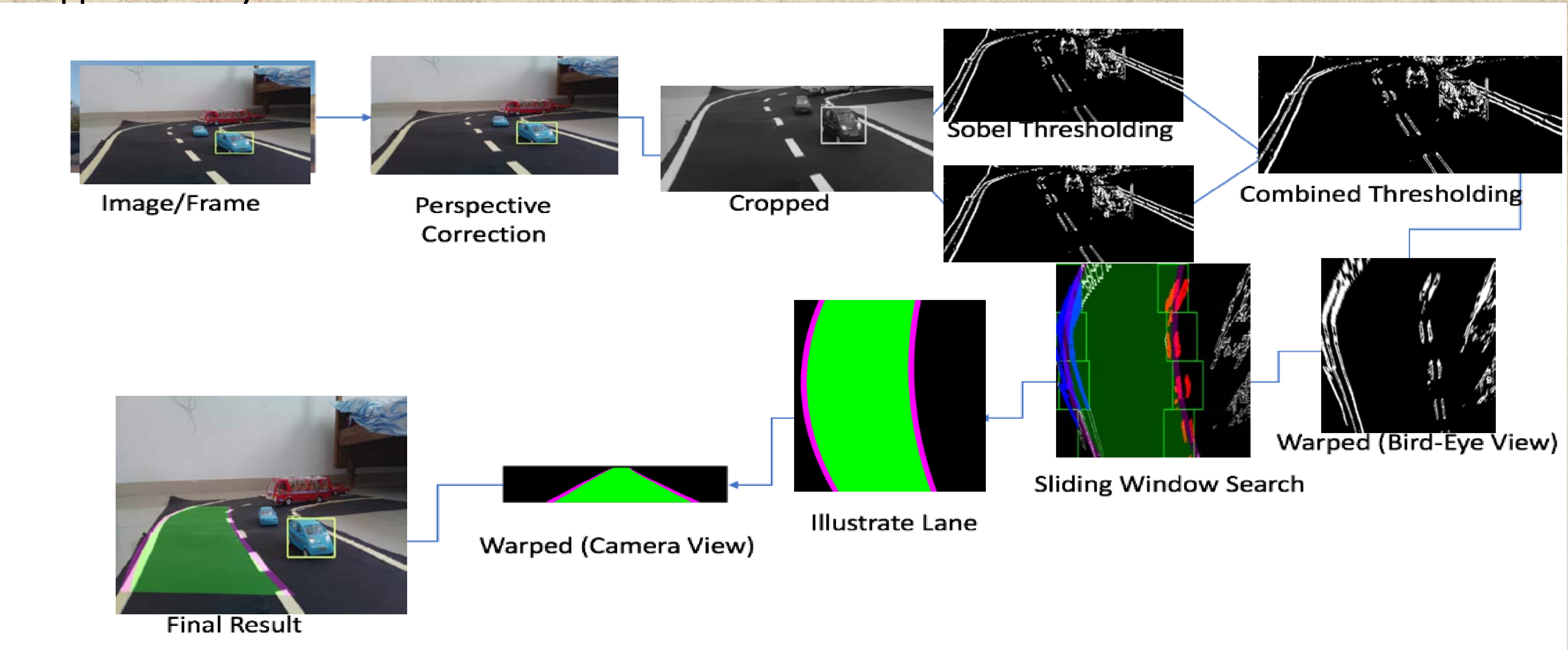
DESIGN PROCESS – LANE DETECTION

https://github.com/maunesh/advanced-lane-detection-for-self-driving-cars



OUTPUT – TOY ROAD

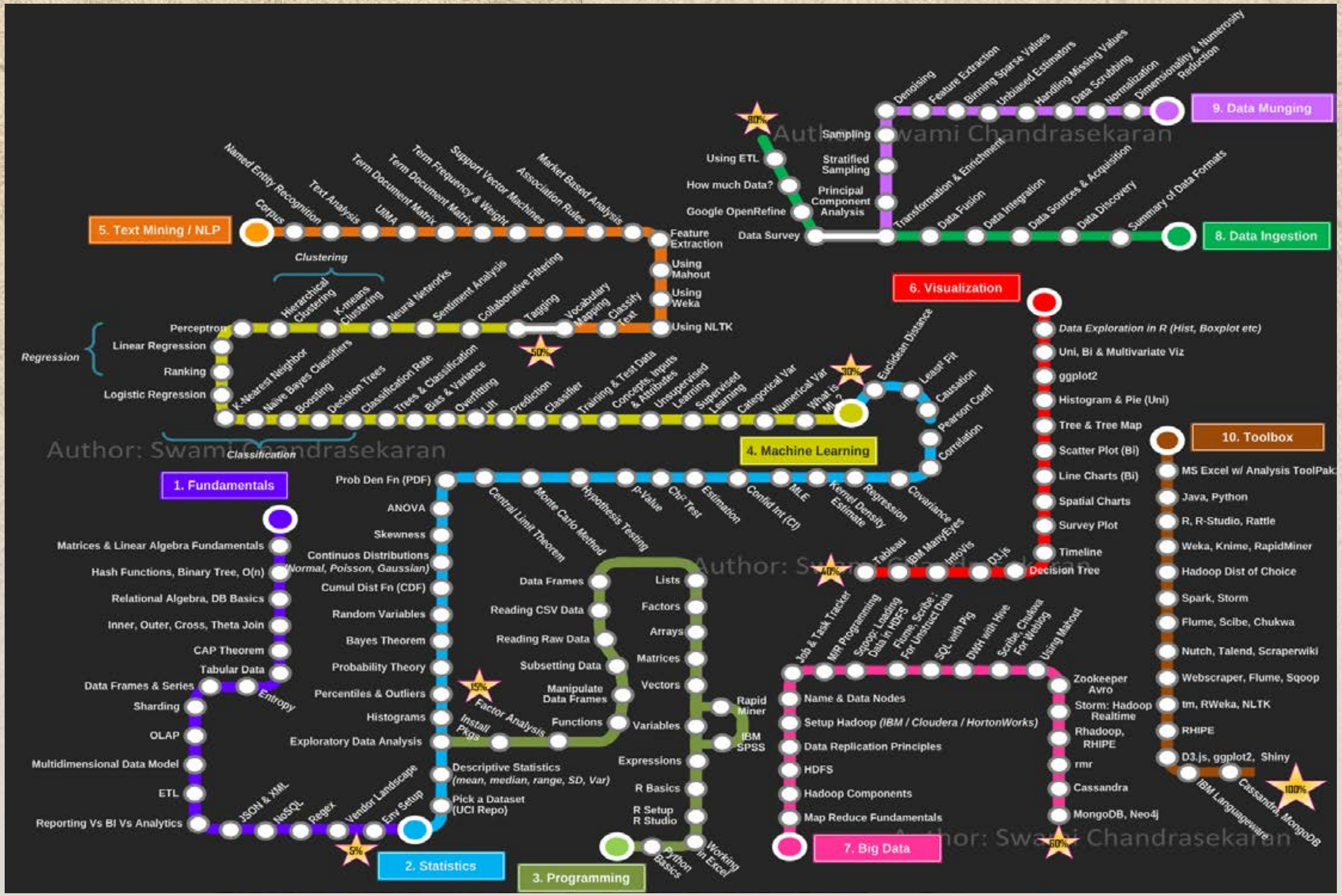
- Applied on Toy Road



LIMITATIONS SO FAR

- Lane Detection Approach on Toy Road didn't work well as the Road Turning Radius, Camera angle capturing the road didn't suit the approach. Though it was able to detect in Parts.
- Existing data available on Internet were for real cars for real road and any model to be trained can only be tested on similar test data. So, for our Toy road we may have to generate our own image data for Training.
- The existing data available on net with Labels, were mostly used to train and validate via simulation, like video games
- Oops! We may need to capture our own images of the toy road, drive and capture steering commands (Left, Straight, Right, Home, Reverse) to be able to train and drive on our toy road.
- We may have to get into Machine Learning!!

MACHINE/DEEP LEARNING – METRO MAP



Metro Map on Technologies Involved in Machine/Deep Learning

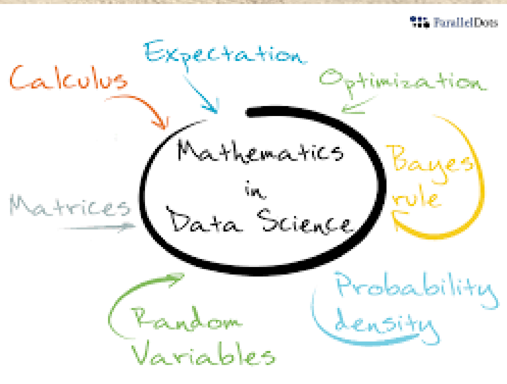
- Swami Chandrasekaran

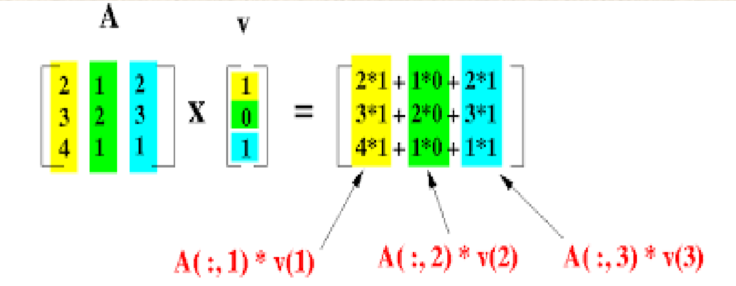
Mind Map – AI/ML

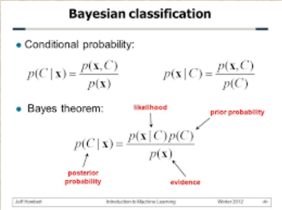
-Karthikeyan Sankaran

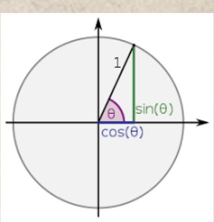
MATH

- Linear Algebra
- Trigonometry
- Calculus
- Geometry
- Probability









Don't miss his Series, a Great Guy!!

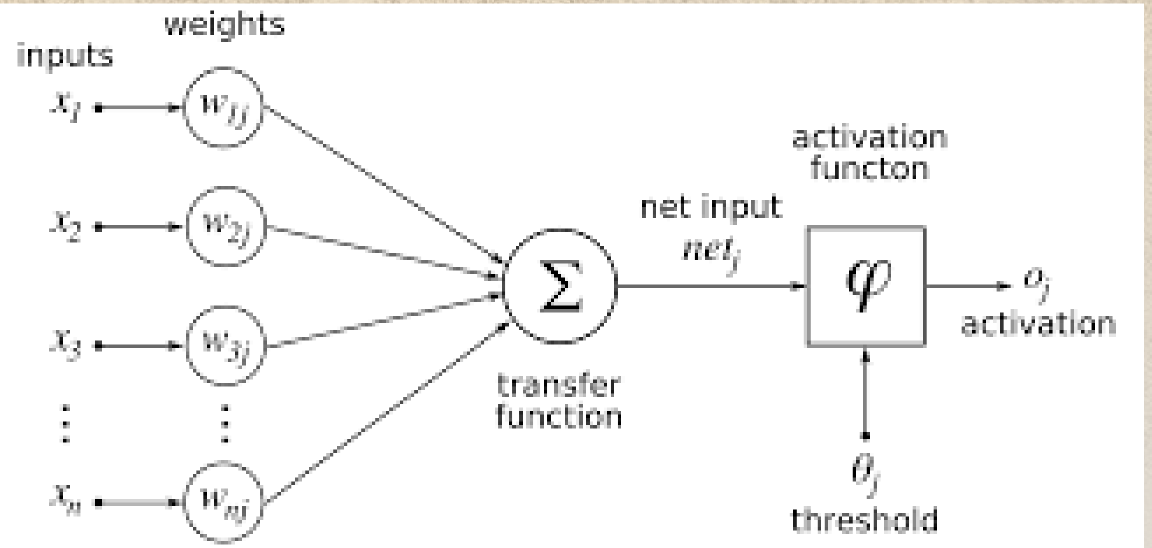
MACHINE/DEEP LEARNING

- Two Problems
  - Regression – e.g Predict the Temperature of a day In Chennai, Predict Stock Prices (Numerical data)
  - Classification – Is it going to be Rainy or Sunny today, is it a cat or a dog (Categorical Data)
- Process
  - Collect Training Data e.g *Images , Text , Numeric*
  - Choose Model – *e.g Linear/Logistic Regression,Neural Network, Decision Trees, Random Forest, Extremt Gradient Boosting*
  - Define Loss/Cost Function – e.g *Lest Squares (Regression), Cross Entropy (Classification)*
  - Define Optimization Objective and Regularization - e.g Maximum Likely hood Estimate, Stochastic Gradient Descent, Adaptive Momentum
  - Define Evaluation Metric – e.g Accuracy, Precision, Recall, F1Measure, Sensitivity, Specificity, Area Under the Curve, Receiver Operator Curve

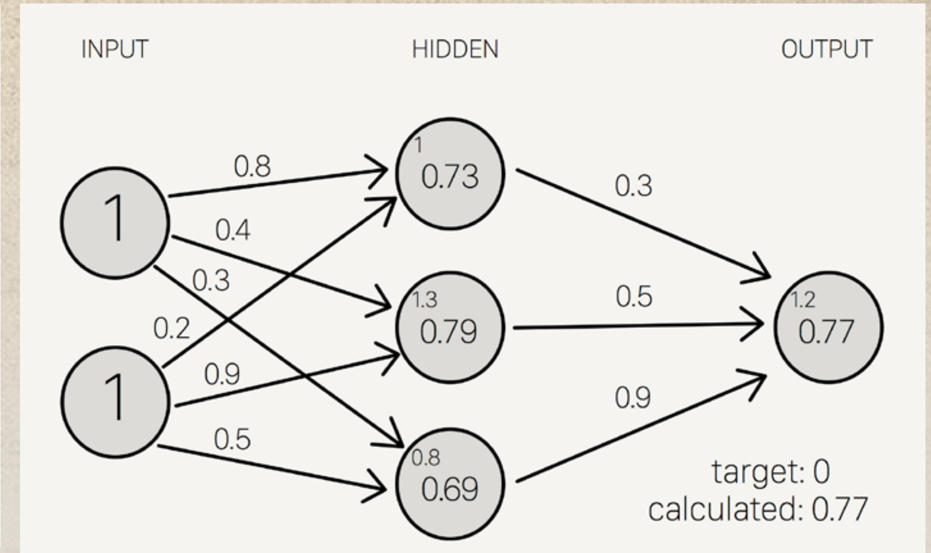
DEEP LEARNING – NEURAL NETWORK

Neural networks process information in a similar way the human brain does. The **network** is composed of a large number of highly interconnected processing elements(neurons) working in parallel to solve a specific problem. Example, we want to predict the command for Car, to turn Right or Left given some input.

[How it Works : Steven Miller]()



Network

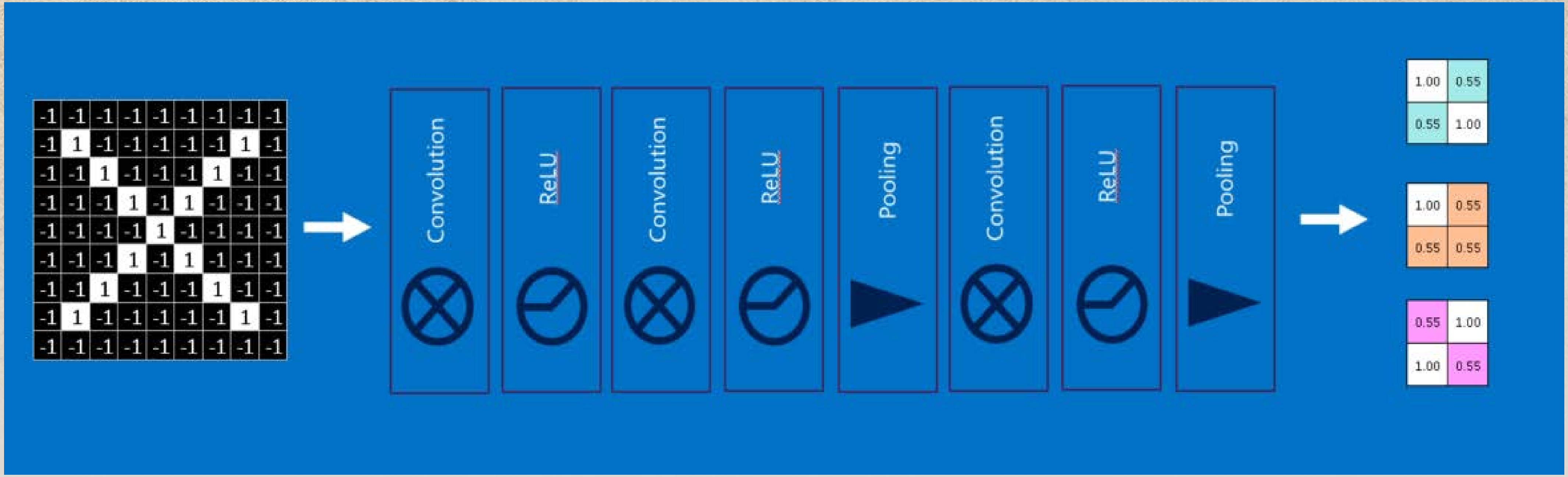


Example Weight, Bias, Target, Error

CONVOLUTION NEURAL NETWORK

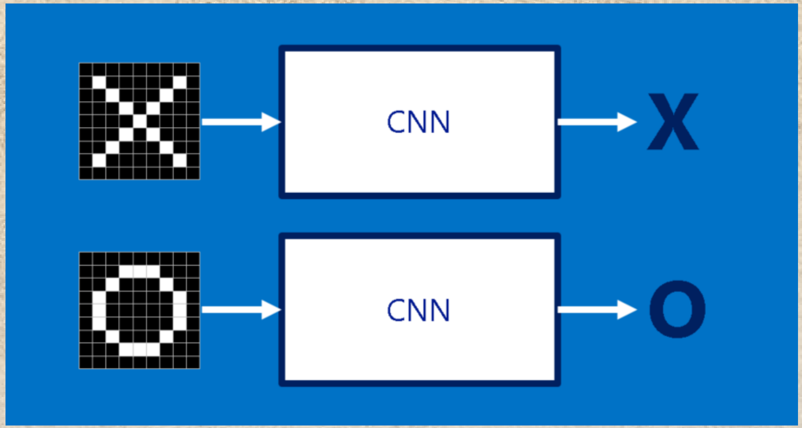
In deep learning, a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery. CNNs use a variation of multilayer perceptron designed to require minimal preprocessing.

Here for Processing our Road Images and Prediction, we use CNN as backbone
How it Works : Brandon Rohrer

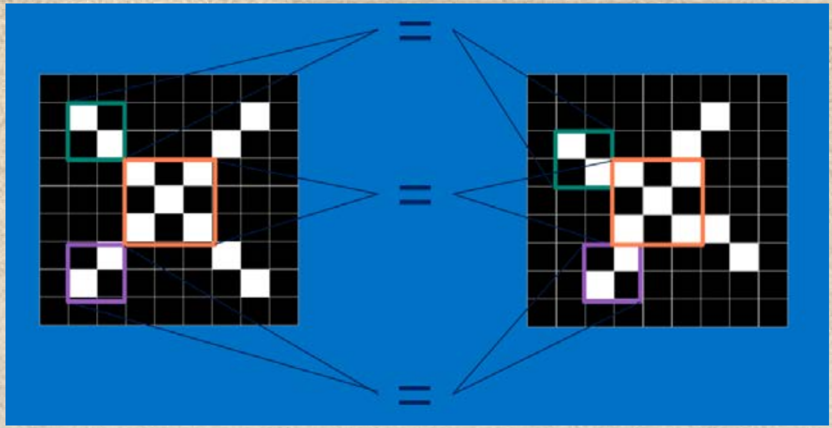


CONVOLUTION NEURAL NETWORK - IMAGE

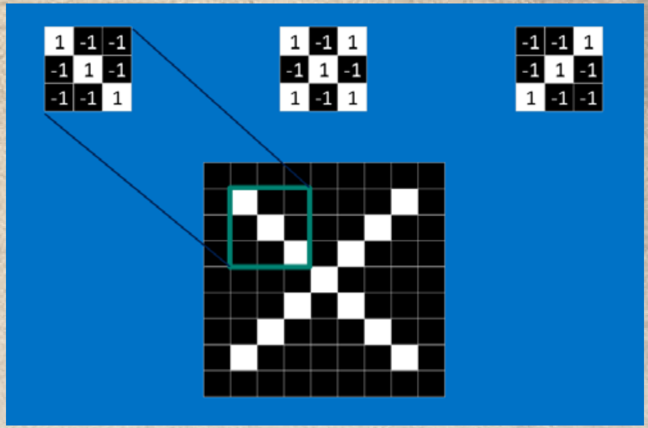
Example– We have Image of O and X . We need to Predict it correct



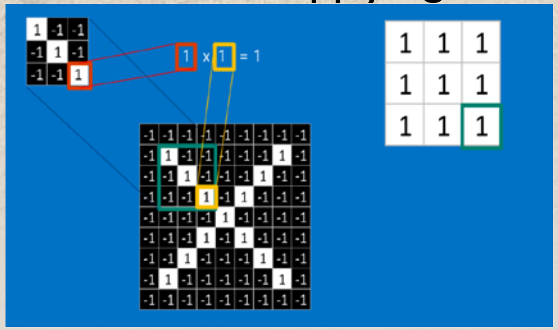
Features – parts of Image

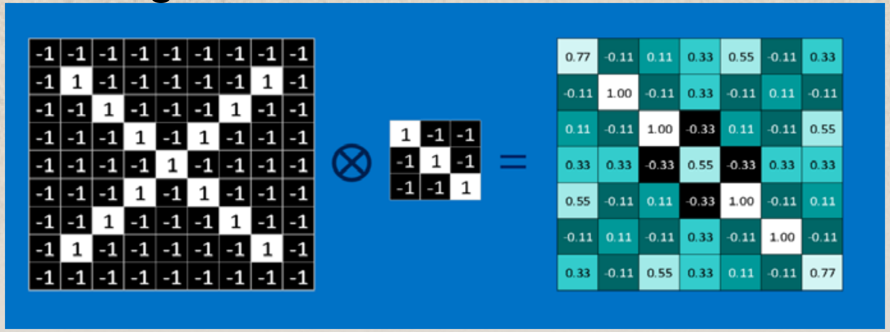


Filters/Kernel – To detect Presence of Features

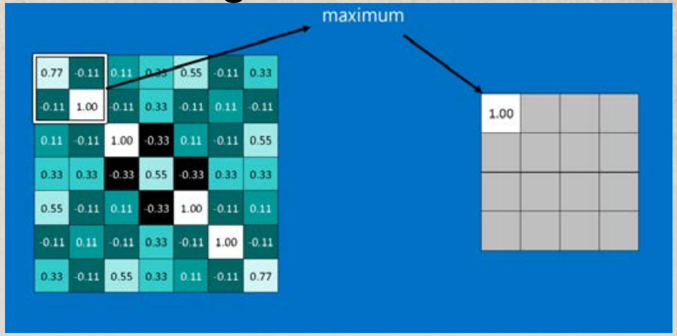


Convolution: Applying Filters to Image



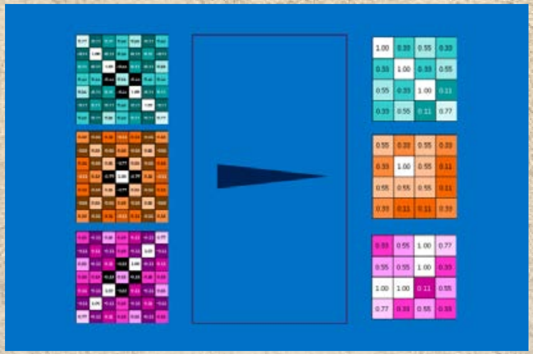


Max Pooling: Dimensionality Reduction

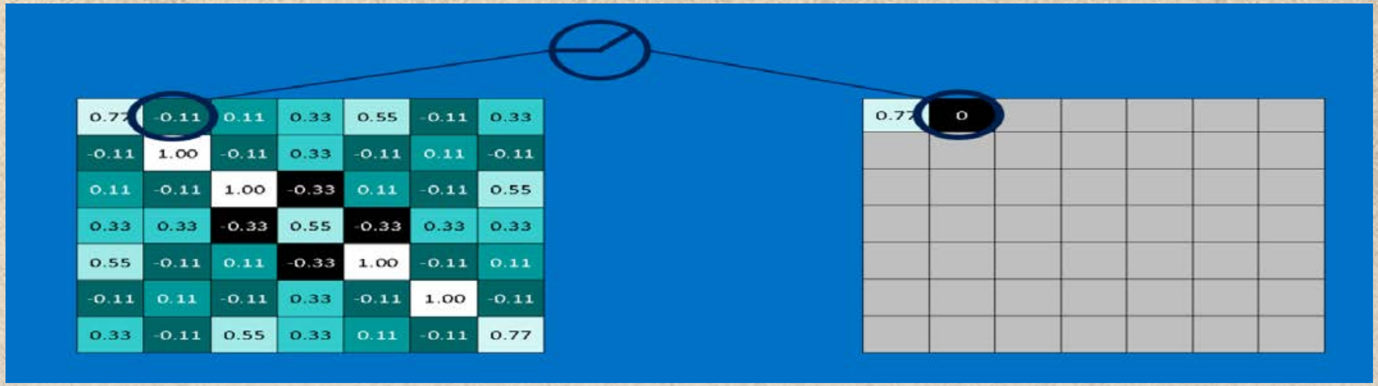


CONVOLUTION NEURAL NETWORK

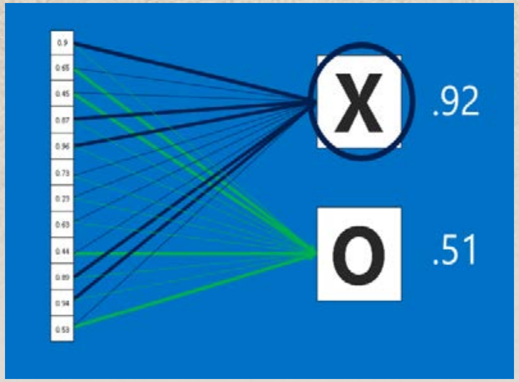
Max Pooling



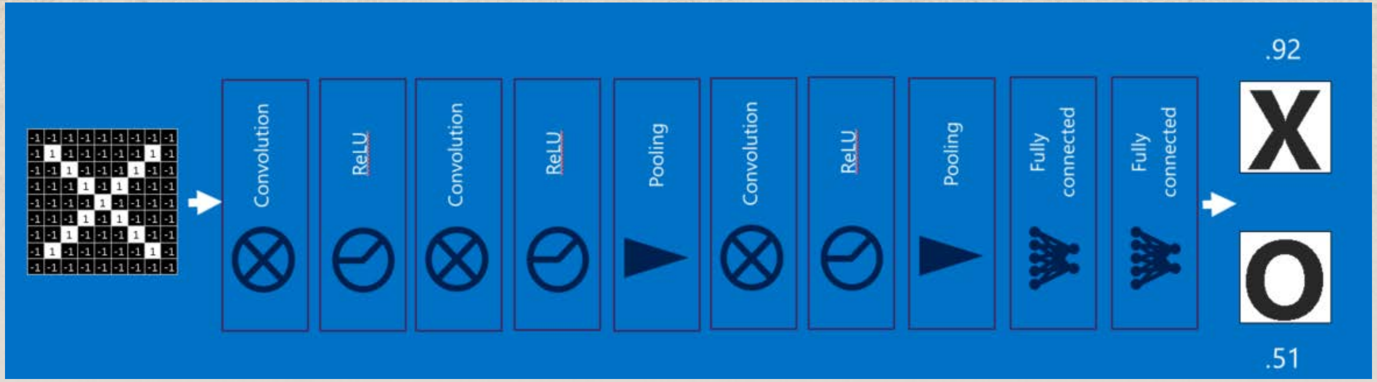
Rectified Linear Unit: Activation function, -ve values are set to Zero



Fully Connected Layers



Output: Predicted X or O with Probability



COST/LOSS FUNCTION – CROSS ENTROPY

Cross Entropy - **Cross-entropy** compares the model's prediction with the label which is the true probability distribution. The cross-entropy goes down as the prediction gets more and more accurate. It becomes zero if the prediction is perfect. i.e If our Model says the car to **Turn Right with 90% Probability** and our Training Label says **Turn Right with 100% Probability**, the Cross entropy is less, witch is good.

- Information Theory – Fundamental of Information
- Information = -log(p), where 'p' if the Probability of the Event
- Rare Event has got more information

Probability of Snow in July in New York (0.1) = -log (0.1) = - * (-3.32) = 3.32 bit of Information (Rare Event, More Information)

Probability of Snow in January in New York (0.9) = -log(0.9) = - *(-0.15) = 0.15 bit of Information (Highly Likely, Less Information)

- Entropy is Average Information : i.e Sum of Probabilities of all events

$$H(X) = \sum_{i=1}^{M} p_i \log_2\left(\frac{1}{p_i}\right)$$

Example : Toss a fair coin

P(Heads)=1/2=0.5

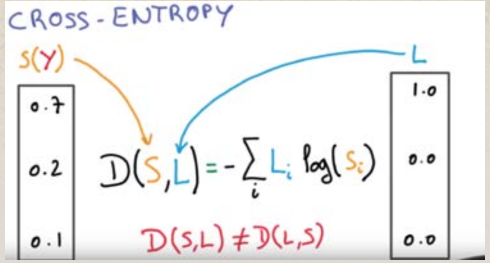
P(Tails) =1/2=0.5

Entropy= ½(log(1/0.5) + ½(log(1/0.5))=1 bit

Cross-entropy – Comparing two Probability distributions and determine it's closeness

Example : Compare Model Prediction [0.1, 0.8, 0.05,0.04,0.01] Vs Actual Label:[0,1,0,0,0]

Note: Direction Labels [left, straight, right, home, reverse]

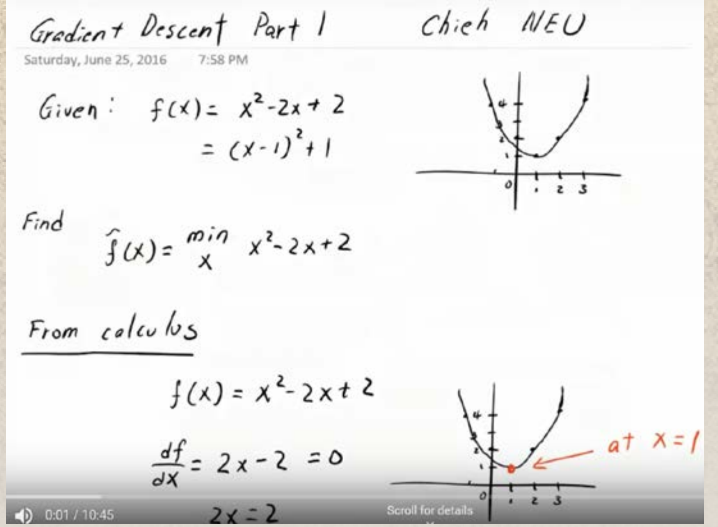


Udacity

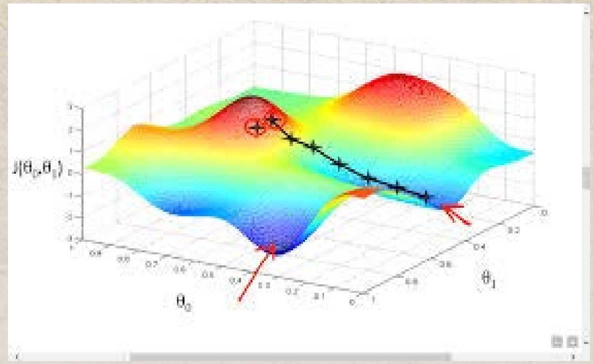
Reference:Information Theory and Coding by Prof. S.N.Merchant, Department of Electrical Engineering, IIT Bombay

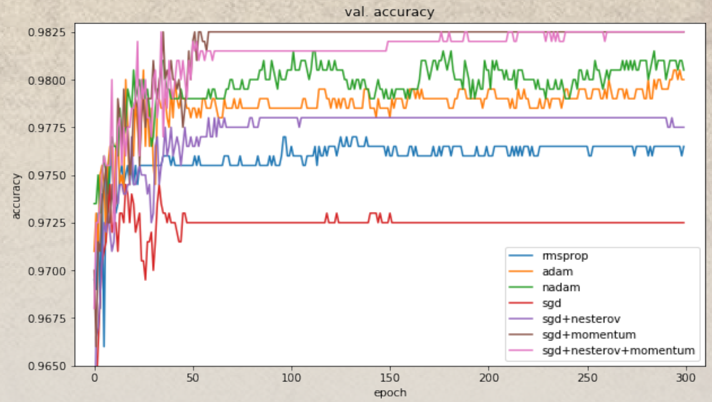
OPTIMIZATION – GRAIDENT DESENT

- Gradient Descent Optimization : Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. Here, we wanted to minimize the loss of function (Cross-Entropy) of Prediction and Actuals and use gradient decent.
- How it works? – Navigate to the bottom of the slope, step by step



In Our Car Training Model, we use ADAM Optimization, is an extension to stochastic gradient descent





- Chieh Wu - from Northeastern University

ARTIFICIAL INTELLIGENCE - FRAMEWORKS

Image Classification
AlexNet, VGG16, GoogLeNet, ResNet, MobileNet, etc.

Object Detection
SSD, Yolo v1/v2/v3, R-FCN, RCNN, Faster RCNN, etc.

Image Segmentation
SegNet, U-Net, FCN, DeepLab v1/v2, etc.

Face Detection / Recognition
MTCNN, DeepFace, Facenet, etc.

Video Classification
RNN, LSTM, etc.

Speech Recognition
DeepVoice, WaveNet, etc

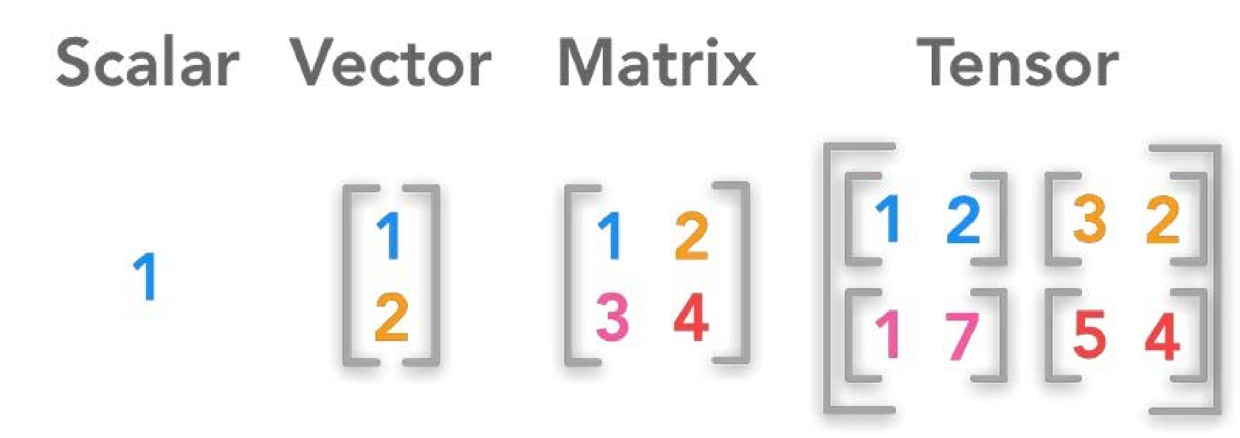
Frameworks
(Caffe, Caffe2, CNTK, MXNet, Neon, PyTorch, Tensorflow …)

Training Platform
Intel® MKL
NVIDIA® CUDA
OpenCL

Inference Platform
CoreML (iOS) , OpenVINO
Tensorflow Lite (Android)
TensorRT

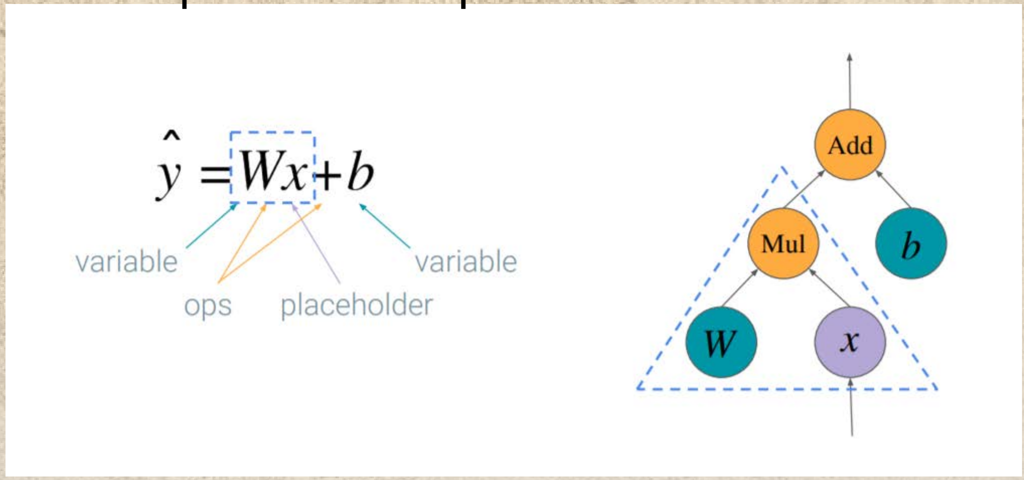
TENSORFLOW

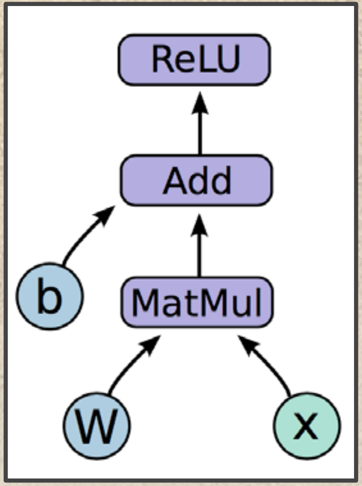
- Open source library for numerical computation using **data flow graphs**
- Developed by Google Brain Team
- Tensor Flow = Tensor (Multi Dimensional Array) + Flow

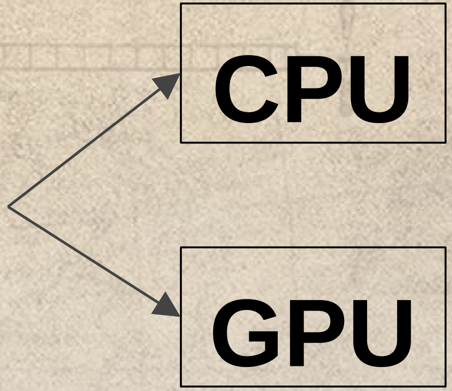


TENSORFLOW

Linear Equation / Perceptron

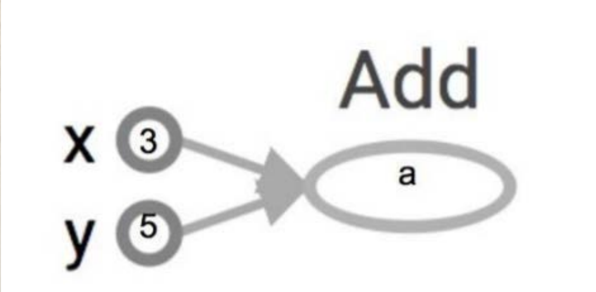






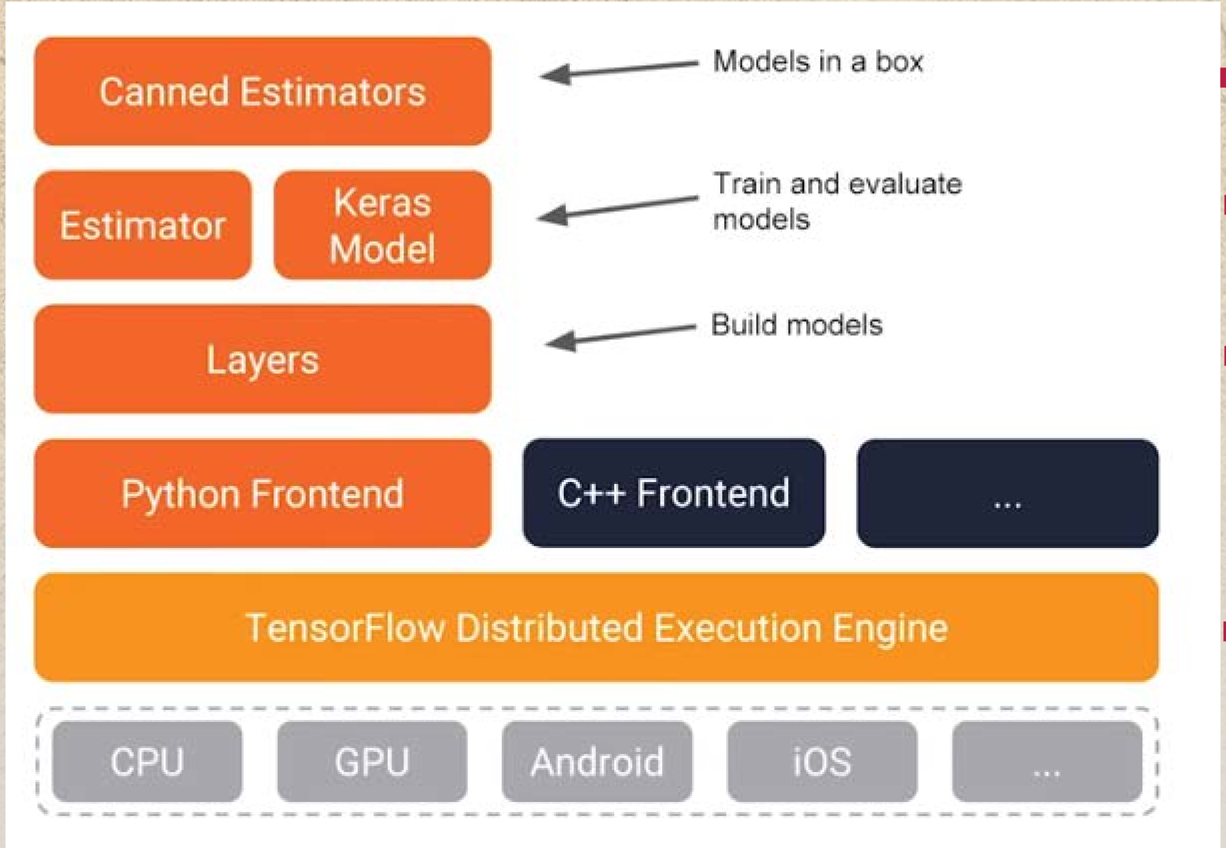
Sample Code: Add two Numbers

```
import tensorflow as tf
a = tf.add(3, 5)
sess = tf.Session()
print sess.run(a)
sess.close()
```



Play here!

TENSORFLOW - LAYERS

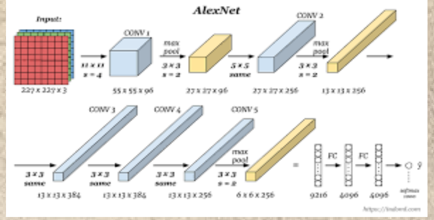


- Built in Linear, Logistic Regression, Neural Network etc
- TFLearn is similar to Keras, High level Abstraction
- Low Level Programming
- Distributed Computing – e. g In SikitLearn, we have to manually take care

CNN BASED – ALEXNET - FRAMEWORK

Alxenet – Tensor flow Flearn Implementation of Alxenet Used to learn from Image and predict the car steering command.

Size / Operation	Filter	Depth	Stride	Padding	Number of Parameters	Forward Computation
3* 227 * 227						
Conv1 + Relu	11 * 11	96	4		(11*11*3 + 1) * 96=34944	(11*11*3 + 1) * 96 * 55 * 55=105705600
96 * 55 * 55						
Max Pooling	3 * 3		2			
96 * 27 * 27						
Norm						
Conv2 + Relu	5 * 5	256	1	2	(5 * 5 * 96 + 1) * 256=614656	(5 * 5 * 96 + 1) * 256 * 27 * 27=448084224
256 * 27 * 27						
Max Pooling	3 * 3		2			
256 * 13 * 13						
Norm						
Conv3 + Relu	3 * 3	384	1	1	(3 * 3 * 256 + 1) * 384=885120	(3 * 3 * 256 + 1) * 384 * 13 * 13=149585280
384 * 13 * 13						
Conv4 + Relu	3 * 3	384	1	1	(3 * 3 * 384 + 1) * 384=1327488	(3 * 3 * 384 + 1) * 384 * 13 * 13=224345472
384 * 13 * 13						
Conv5 + Relu	3 * 3	256	1	1	(3 * 3 * 384 + 1) * 256=884992	(3 * 3 * 384 + 1) * 256 * 13 * 13=149563648
256 * 13 * 13						
Max Pooling	3 * 3		2			
256 * 6 * 6						
Dropout (rate 0.5)						
FC6 + Relu					256 * 6 * 6 * 4096=37748736	256 * 6 * 6 * 4096=37748736
4096						
Dropout (rate 0.5)						
FC7 + Relu					4096 * 4096=16777216	4096 * 4096=16777216
4096						
FC8 + Relu					4096 * 1000=4096000	4096 * 1000=4096000
1000 classes						
Overall					62369152=62.3 million	1135906176=1.1 billion
Conv VS FC					Conv:3.7million (6%) , FC: 58.6 million (94%)	Conv: 1.08 billion (95%) , FC: 58.6 million (5%)



Alxenet - Hao Gao

Input: Road Image
input_data(shape=[None, WIDTH-160, HEIGHT-120, 1], name='input')

Output: Command for Car
[0,1,0,0,0] – One Hot Array

Array Positions- Reference:
0 - Left
1 - Straight
2 - Right
3 – Home
4 - Reverse

CAR – MANUAL DRIVING

- Drive the car manually, capture images, label them and Store as Training data

- Map Keys to Steering [Left, Straight, Right, Home, Reverse]
- Example [0,1,0,0,0]

```python
def keys_to_output(keys):
    '''
    Convert keys to a ...multi-hot... array

     [A,W,D] boolean values.
    '''
    output = [0,0,0,0,0]

    if 'A' in keys:
        output[0] = 1
    elif 'D' in keys:
        output[2] = 1
    elif 'W' in keys:
        output[1] = 1
    elif 'E' in keys:
        output[3] = 1
    elif 'S' in keys:
        output[4] = 1

    return output

file_name = 'training_data_keys_v15.npy'

if os.path.isfile(file_name):
    print('File exists, loading previous data!')
    training_data = list(np.load(file_name))
else:
    print('File does not exist, starting fresh!')
    training_data = []
```

- Captured Road Image + One Hot Encoding of Steering
- Example [[IMAGE PIXELS], [0,1,0,0,0]]

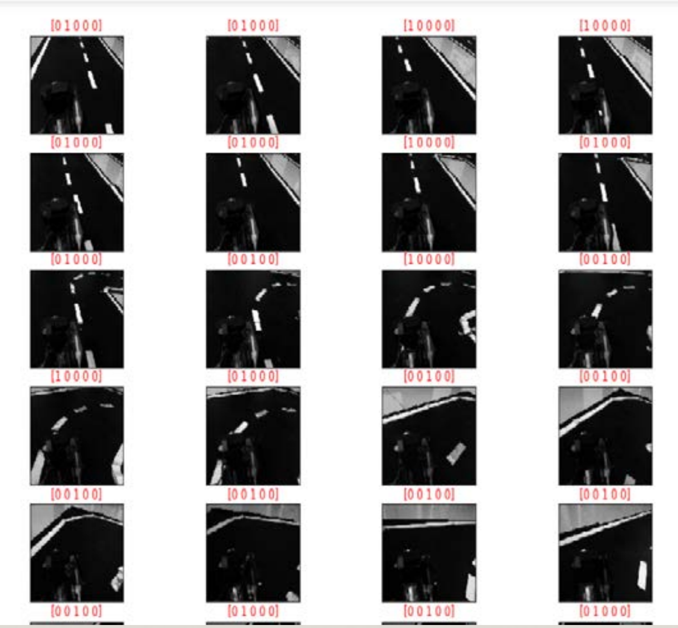
```python
            elif event.key == pygame.K_a:
              left = False
              key_flag=0
            elif event.key == pygame.K_s:
              reverse = False
              key_flag=0
            elif event.key == pygame.K_d:
              right = False
              key_flag=0
            elif event.key == pygame.K_e:
              home = False
              key_flag=0


        time.sleep(1)

        if(keys!='Q'):
          break
        #
    print('Keypress',keys)
    output = keys_to_output(keys)
    print('Keys',output)
    time.sleep(1)
    image_np_without_object=cv2.resize(cv2.imread(os.path.join(dirname, "frame%d.jpg" %count),cv2.IMREAD_G
    if (keys=='D'):
      image_np_front_without_object=cv2.resize(cv2.imread(os.path.join(dirname, "front_cam%d.jpg" %count),
      #training_data.append([np.array(image_np_front_without_object),np.array(output)])

    image_np_without_object_cropped=image_np_without_object[int(HEIGHT/2):HEIGHT , 0:int(WIDTH)]
  #image_np_without_object_cropped = [image_np_without_object.shape[0]/2:image_np_without_object.shape[0]]
  #image_np_without_object_cropped = cv2.resize(image_np_without_object, None, fx=1 / 2, fy=1 / 2, interpo

    training_data.append([np.array(image_np_without_object),np.array(output)])
    print('Training Image shape',image_np_without_object.shape)
    print('Training Image shape -Cropped',image_np_without_object_cropped.shape)
    cv2.imwrite(os.path.join(dirname, "cropped_frame%d.jpg" %count), image_np_without_object_cropped)
    print('Lenght of Traning Data',len(training_data))
    np.save(file_name,training_data)

    #if len(training data) % 1 == 0:
```

- Key Stroke for Steering Captured via python pygme GUI
- Sunfouder TCP Commands Executed on key strokes
- Pi Cam Images are Stored for every move.
- Image Pixels with One hot encoding saved as numpy file
  Ie. [image pixel array],
  [car command]
- Example:
  [[10,155,10], [0,1,0,0,0]]

MODEL TRAINING

- The captured Image Pixel with one hot encoding (Steering)
- Stored as Numpy Array - training_data_keys_v15.npy

Training Data:



TFLearn/Alexnet – Model Training

```python
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
from tflearn.layers.normalization import local_response_normalization
import tensorflow as tf
tf.reset_default_graph()

network = input_data(shape=[None, WIDTH, HEIGHT, 1], name='input')
network = conv_2d(network, 96, 11, strides=4, activation='relu')
network = max_pool_2d(network, 3, strides=2)
network = local_response_normalization(network)
network = conv_2d(network, 256, 5, activation='relu')
network = max_pool_2d(network, 3, strides=2)
network = local_response_normalization(network)
network = conv_2d(network, 384, 3, activation='relu')
network = conv_2d(network, 384, 3, activation='relu')
network = conv_2d(network, 256, 3, activation='relu')
network = max_pool_2d(network, 3, strides=2)
network = local_response_normalization(network)
network = fully_connected(network, 4096, activation='tanh')
network = dropout(network, 0.5)
network = fully_connected(network, 4096, activation='tanh')
network = dropout(network, 0.5)
network = fully_connected(network, 5, activation='softmax')
network = regression(network, optimizer='momentum',
                     loss='categorical_crossentropy',
                     learning_rate=LR, name='targets')

model = tflearn.DNN(network, checkpoint_path='model_alexnet',
                    max_checkpoints=1, tensorboard_verbose=3, tensorboard_dir='log')


#model = tflearn.DNN(network, checkpoint_path='model_alexnet')

#print('inputnode',bundle.Graph.Operation(''))

if os.path.exists('{}.meta'.format(MODEL_NAME)):
    model.load(MODEL_NAME,weights_only=True)
    print('MODEL_NAME',MODEL_NAME)
    #model.load('cardriving-0.001-2conv-basic.model')

    print('model loaded!')
```

TFLearn/Alexnet – Model Training

```python
train = train_data[:50000]
test = train_data[50001:]
print("Training length",len(train))
print("Testing length",len(test))

X = np.array([i[0] for i in train]).reshape(-1,WIDTH,HEIGHT,1)
Y = [i[1] for i in train]

test_x = np.array([i[0] for i in test]).reshape(-1,WIDTH,HEIGHT,1)
test_y = [i[1] for i in test]

model.fit({'input': X}, {'targets': Y}, n_epoch=4, validation_set=({'input': test_x}, {'targets': test_y}),
    snapshot_step=500, show_metric=True, run_id=MODEL_NAME)

model.save(MODEL_NAME)
```

TFLearn/Alexnet – Saved Model

cardriving-0.001-2conv-basicv15_balanced6.model.data-00000-of-00001	1/12/2019 3:15 PM	DATA-00000-OF-0...	324,230 KB
cardriving-0.001-2conv-basicv15 balanced6.model.index	1/12/2019 3:15 PM	INDEX File	2 KB
cardriving-0.001-2conv-basicv15_balanced6.model.meta	1/12/2019 3:15 PM	META File	258 KB

frozen_model.pb

MODEL DEPLOYMENT AND TESTING

- Start the Car, Load the Model, Capture the Image, Pass it to Model and Predict the Steering

Load the Model

```python
network = conv_2d(network, 96, 11, strides=4, activation='relu')
network = max_pool_2d(network, 3, strides=2)
network = local_response_normalization(network)
network = conv_2d(network, 256, 5, activation='relu')
network = max_pool_2d(network, 3, strides=2)
network = local_response_normalization(network)
network = conv_2d(network, 384, 3, activation='relu')
network = conv_2d(network, 384, 3, activation='relu')
network = conv_2d(network, 256, 3, activation='relu')
network = max_pool_2d(network, 3, strides=2)
network = local_response_normalization(network)
network = fully_connected(network, 4096, activation='tanh')
network = dropout(network, 0.5)
network = fully_connected(network, 4096, activation='tanh')
network = dropout(network, 0.5)
network = fully_connected(network, 5, activation='softmax')
network = regression(network, optimizer='momentum',
                     loss='categorical_crossentropy',
                     learning_rate=LR, name='targets')

model = tflearn.DNN(network, checkpoint_path='model_alexnet',
                    max_checkpoints=1, tensorboard_verbose=0, tensorboard_dir='log')


if os.path.exists('{}.meta'.format(MODEL_NAME)):
    model.load(MODEL_NAME,weights_only=True)
    #model.load('cardriving-0.001-2conv-basic.model.meta')
    model.load('cardriving-{}-{}v15_balanced1.model'.format(LR,'2conv-basic'))
    print('model loaded!')
```

TFLearn/Alexnet – Model Prediction

```python
while True:
    key_flag=1
    #for event in pygame.event.get():
    #while True:
    #while True:

    print('Key Flag',key_flag)
    image_np_without_object=cv2.resize(cv2.imread(os.path.join(dirname, "frame%d.jpg" %count),cv2.IMREAD_GRAYSCALE),(WIDTH,HEIGHT))
    img=np.array(image_np_without_object)
    data = img.reshape(WIDTH,HEIGHT,1)
    #print(data[0])
    #testing_data.append([np.array(img),img_num])
    print('Waiting for Model output...')
    model_out = model.predict([data])[0]
    print(model_out)
    print(np.argmax(model_out))
    if np.argmax(model_out) == 1:
        str_label='Forward'
    elif np.argmax(model_out) == 0: str_label='Left'
    elif np.argmax(model_out) == 2: str_label='Right'
    elif np.argmax(model_out) == 3: str_label='Home'
    else: str_label='Reverse'
    #print('event.type',event.type)
    if key_flag==1:
      #print('Waiting for Keypress...')
      #pygame.event.clear()
      #event = pygame.event.wait()
      if str_label == 'Forward':
        print('Forward')
        tcpCliSock.send('home'.encode())
        time.sleep(1)
        tcpCliSock.send('forward'.encode())
```

Drive

OBJECT DETECTION

- The car can detect Common Objects in the Context.
- Users pre trained google SSD Mobile net
- However, Objects aren't kept in the Road as it wasn't trained with.
- It can predict the objects

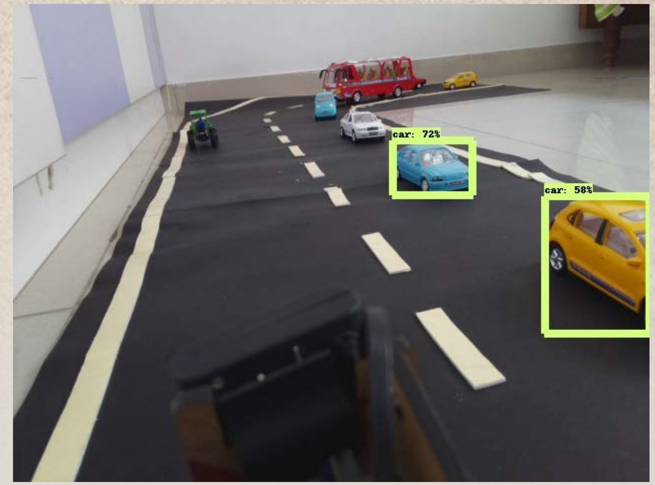
Load the Model Single Shot Detector (Common Objects in Context)

```
NUM_CLASSES = 90


# ## Download Model

# In[5]:

#opener = urllib.request.URLopener()
#opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)
#tar_file = tarfile.open(MODEL_FILE)

tar_file = tarfile.open('/home/pi/tensorflow/ssd_mobilenet_v1_coco_11_06_2017.tar.gz.1')

for file in tar_file.getmembers():
  file_name = os.path.basename(file.name)
  print(file_name)

  if 'frozen_inference_graph.pb' in file_name:
    tar_file.extract(file, os.getcwd())


# ## Load a (frozen) Tensorflow model into memory.

# In[6]:

detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def = tf.GraphDef()
  with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')
    print('Tensorflow Graph imported')
```

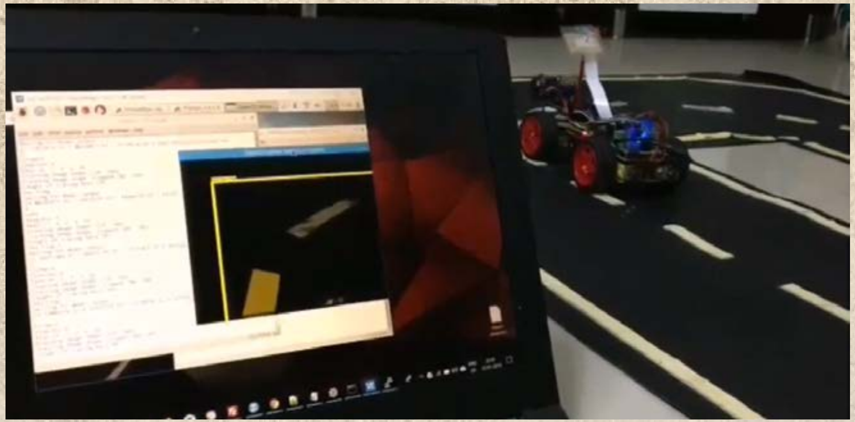
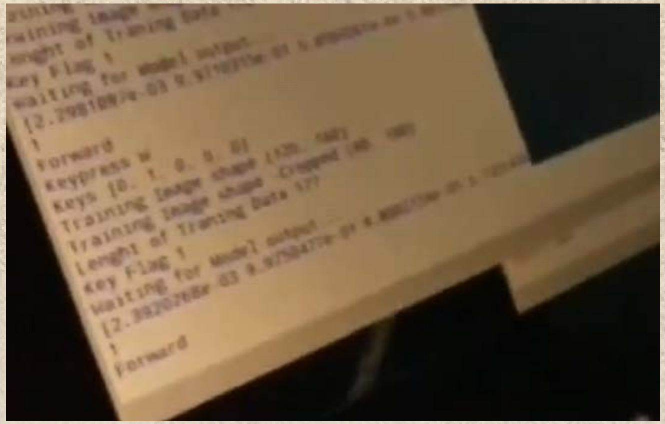
Few other Models

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes

Output



DEMO

https://youtu.be/iV_gfvExhHI

THANK YOU
☺