Steering In Self Driving Cars Using Deep Learning Algorithm

Guided by
Dr.L.Rasikannan
Assistant Professor
Computer Science Department

Submitted by
Maria Anson.A
1715016
Computer Science Department

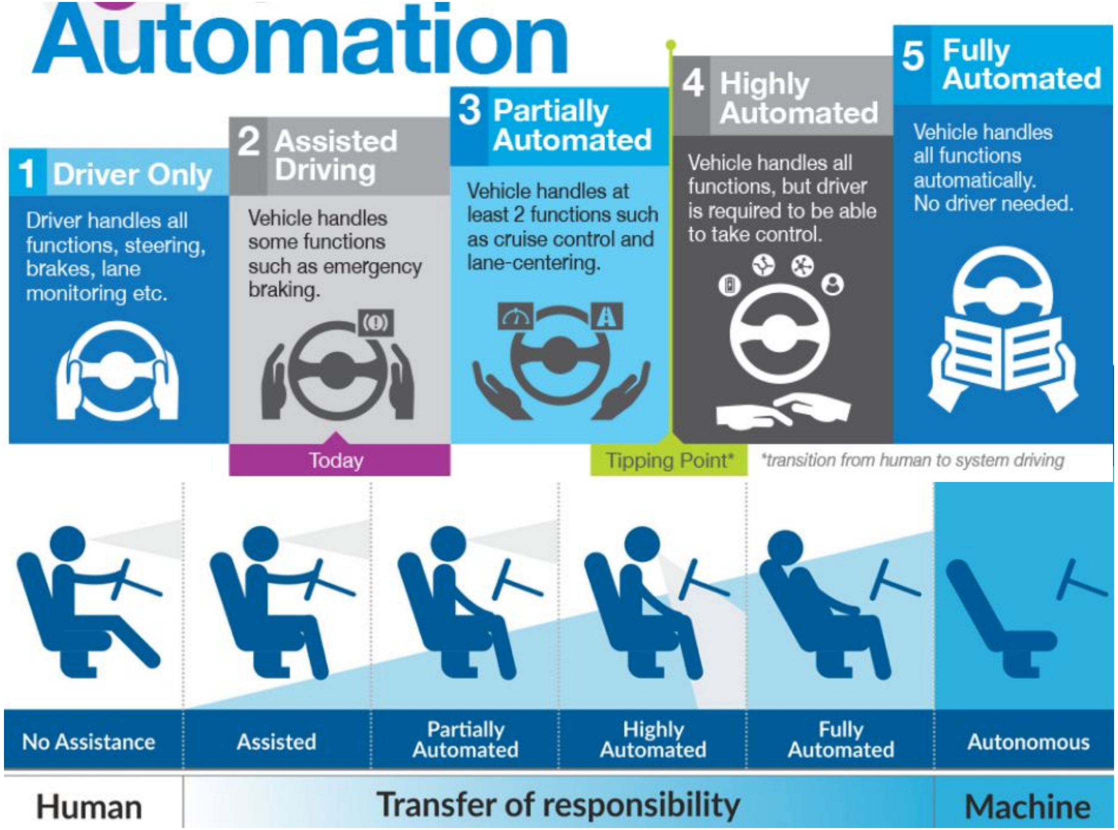
Outline

- ❖ Self Driving Cars
- ❖ Level of Autonomy
- ❖ Machine Learning vs Deep Learning
- ❖ Requirements & Libraries
- ❖ Block Diagram
- ❖ Dataset & Processing
- ❖ Architecture
- ❖ General Working
- ❖ Fine Tuning
- ❖ Training
- ❖ Simulation
- ❖ Conclusion

Self Driving Cars

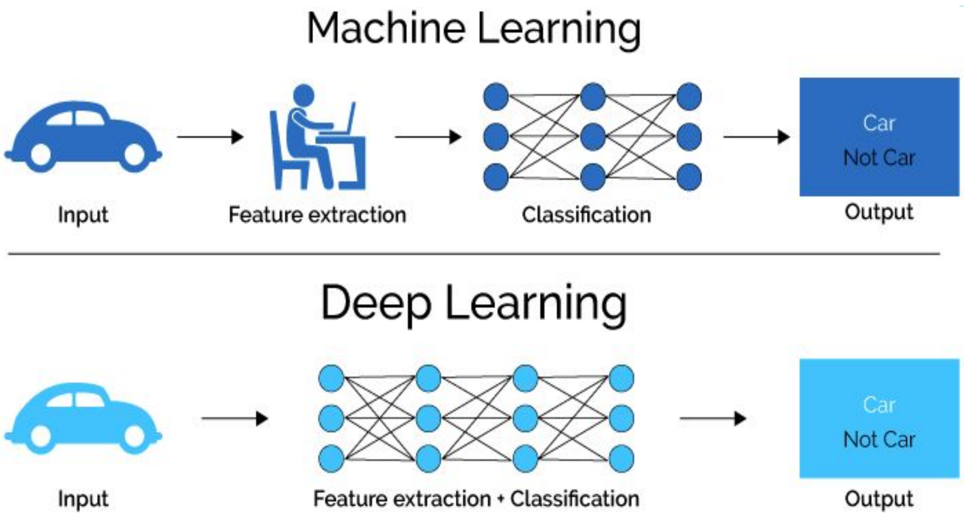
- **Self-driving** vehicles are **cars** or trucks in which human **drivers** are never required to take control to safely operate the vehicle.
- Also known as **autonomous** or “driverless” **cars**, they combine sensors and software to control, navigate, and **drive** the vehicle.
- So to make the vehicle autonomous one of the vital part is steering through different terrains without the intervention of human.
- This can be achieved by using a dash camera placed at the front of the car.
- The video feed is analysed and respective steering angles are generated which are used to actuate the steering wheel.

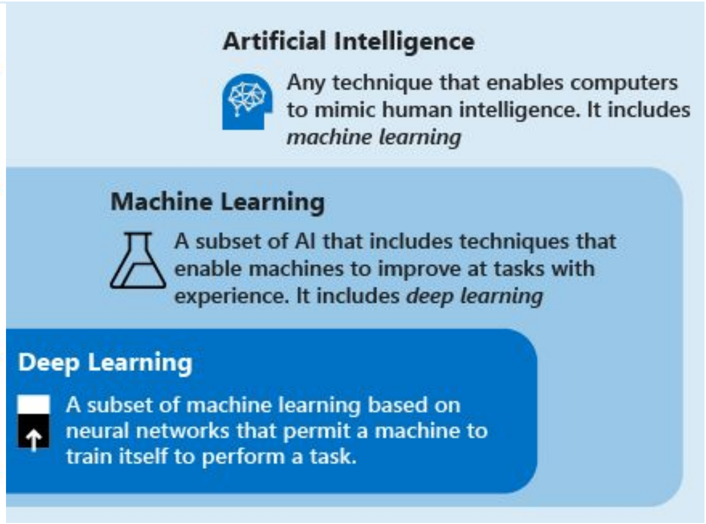
Level of Autonomy



- Level 3 is the current autonomy level in production whereas Tesla cars support level 4 autonomy.
- Level 5 is still in it's conceptual face.

Machine Learning Vs Deep Learning





We adopt the deep learning approach for the project

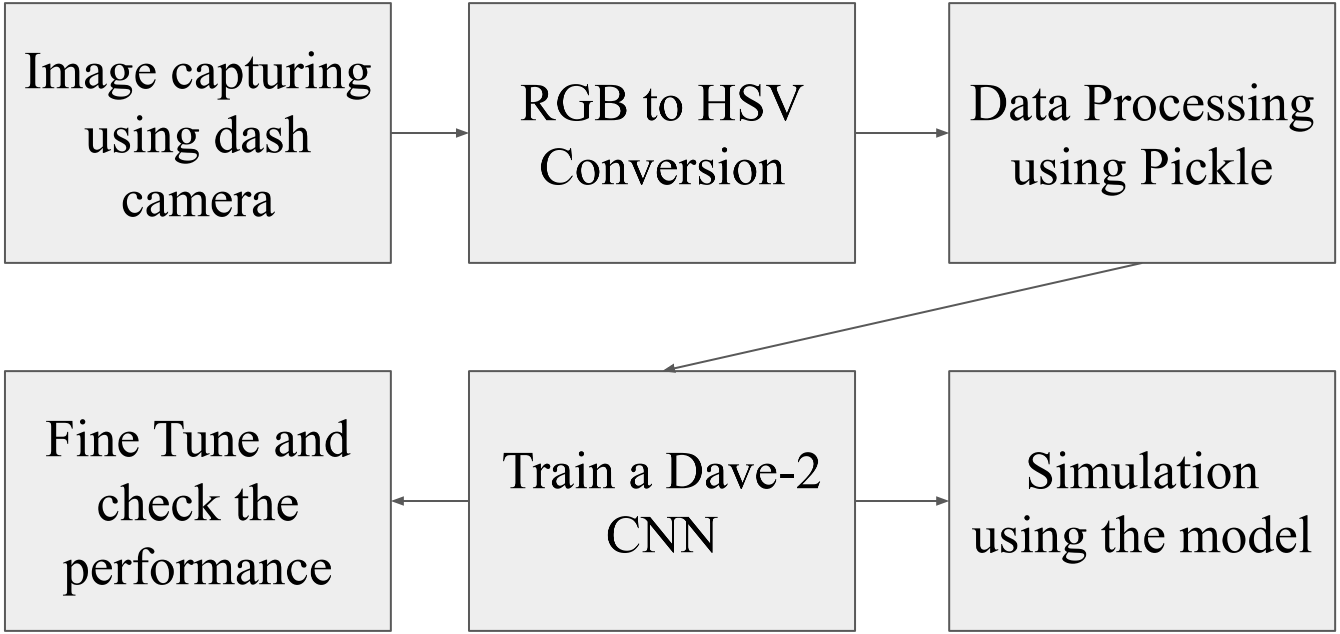
Requirements

- A high quality camera.
- A huge amount of image data from the video recorded by the high quality camera for achieving the accuracy using deep learning approach.
- A high computing gpu enabled system to train the model.
- Prerequisite knowledge of Python programming language.
- Understanding of working of convolutional neural network to perform hyperparameter tuning.
- To achieve the task in real word, we need a mechanical actuation for steering.

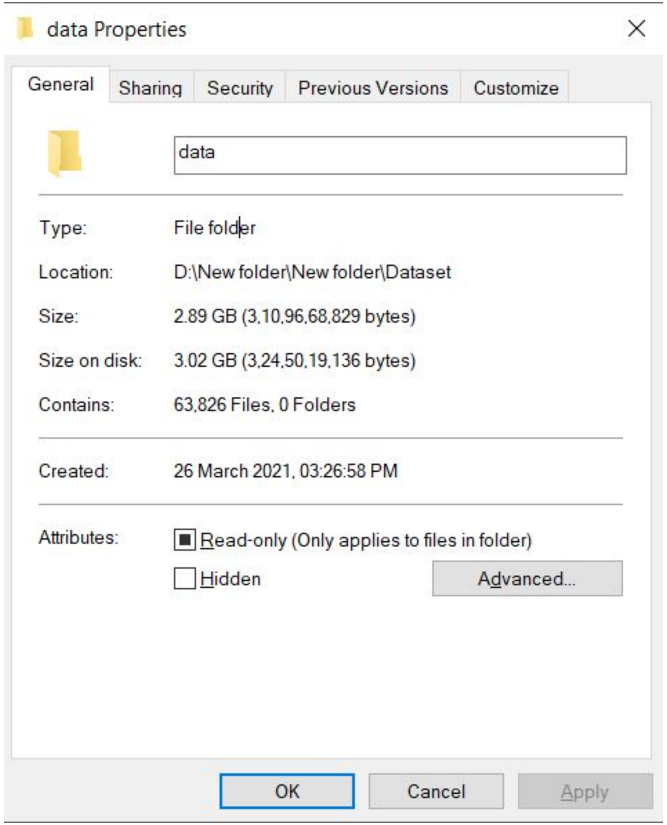
Libraries

- Opencv 4.5.1
- Numpy 1.20.1
- Scipy 1.4.1
- Sklearn 0.24.1
- Keras 2.3.1
- Tensorflow 2.0.0
- Matplotlib 3.3.4
- Pickle
- Os

Block Diagram



Dataset



- The dataset contains 63,826 images which are obtained from the video of a dashcam which was recorded in Rancho Palos Verdes and San Pedro, California and a text file with respective angles for each image.
- Among the images 44677 images are used for training and 19148 images for validation which achieved a minimum loss of 0.22 mean squared error.

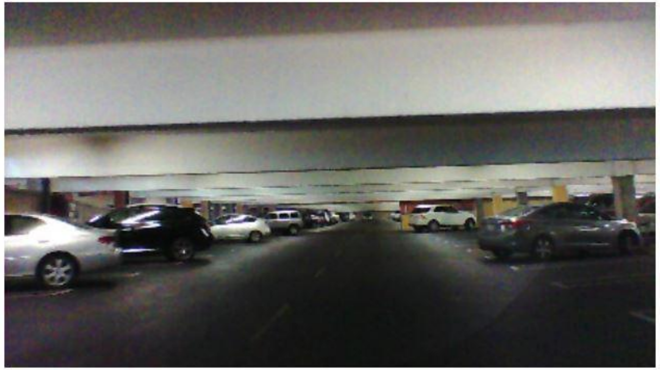
Dataset

```
419.jpg 2.120000,2018-07-01 17:10:06:790
420.jpg 1.920000,2018-07-01 17:10:06:860
421.jpg 1.820000,2018-07-01 17:10:06:889
422.jpg 1.820000,2018-07-01 17:10:06:961
423.jpg 2.020000,2018-07-01 17:10:07:20
424.jpg 2.520000,2018-07-01 17:10:07:58
425.jpg 2.720000,2018-07-01 17:10:07:89
426.jpg 3.030000,2018-07-01 17:10:07:156
```

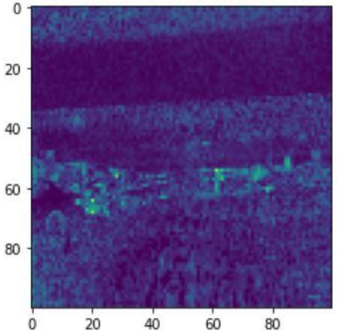
- This is a sample image from the dataset which is 423.jpg
- The label for the image is found in a text file with the respective angle of steering for the particular image.
- The steering angle for each frame is recorded while creating this datset

Data Preprocessing

Raw image : D:\Main Directory\My works\Project\Autonomous car\

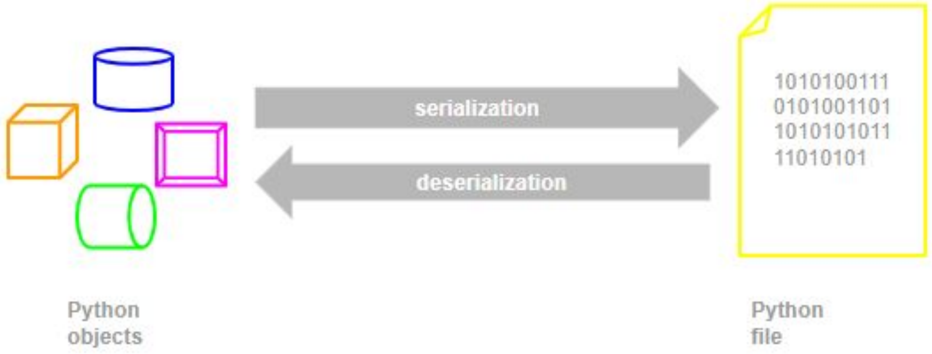


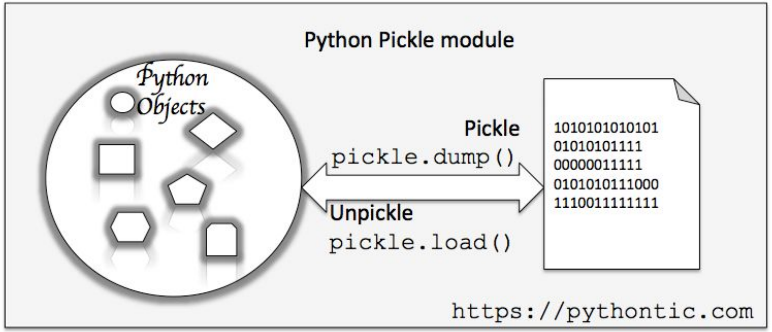
HSV image of raw image D:\Main Directory\My works\Project\Autonomous car\



- The image is converted into a (100,100,3) array so that every image that enters the model follows the same format
- HSV or Hue Saturation Value is used to separate image luminance from color information.
- HSV also used in situations where color description plays an integral role. The images are converted into HSV.

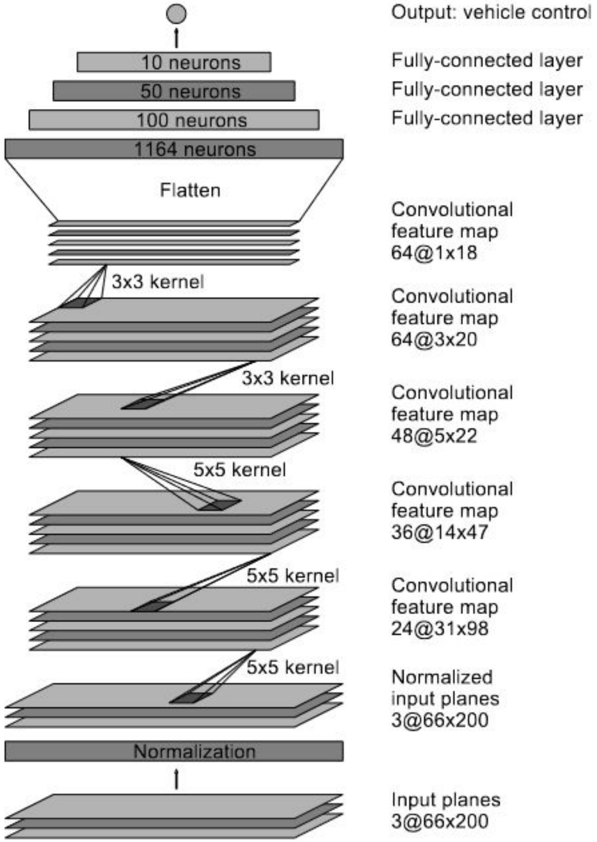
Data Processing





- The transformed data which is image size and in HSV format is stored in a pickle format.
- Pickle format allows processing and accessing the data much faster
- It can also be easily converted into a numpy data type which is converted into tensor data type at the time of model running.

Architecture



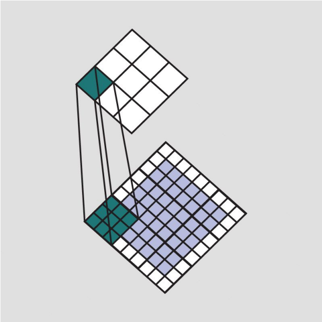
- The architecture of this convolutional neural network is called Dave-2 which is inspired from the paper published by Nvidia corporation.
- It used deep layers to extract more features from the image, so that precise steering angles can be generated
- This architecture has a total of 747,105 parameters.

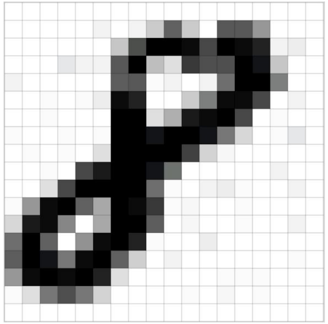
Architecture

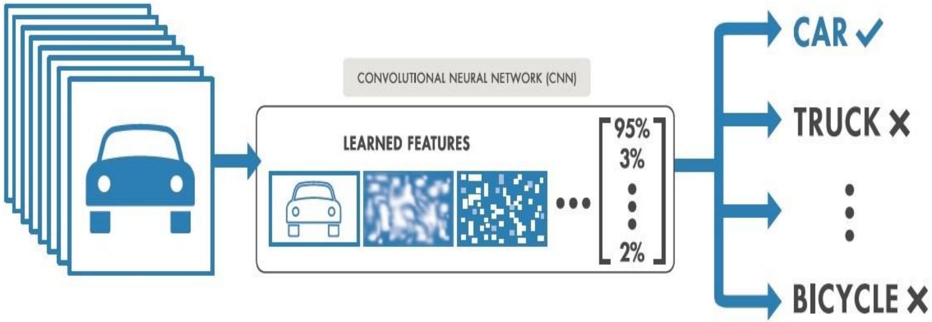
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda (Lambda)              (None, 100, 100, 1)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 100, 100, 32)      832
_________________________________________________________________
activation (Activation)      (None, 100, 100, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 50, 50, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 50, 50, 32)        25632
_________________________________________________________________
activation_1 (Activation)    (None, 50, 50, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 25, 25, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 25, 25, 64)        51264
_________________________________________________________________
activation_2 (Activation)    (None, 25, 25, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
activation_3 (Activation)    (None, 12, 12, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 6, 6, 128)         73856
_________________________________________________________________
activation_4 (Activation)    (None, 6, 6, 128)         0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 3, 3, 128)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 3, 128)         147584
_________________________________________________________________
activation_5 (Activation)    (None, 3, 3, 128)         0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 1024)              132096
_________________________________________________________________
dense_1 (Dense)              (None, 256)               262400
_________________________________________________________________
dense_2 (Dense)              (None, 64)                16448
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
=================================================================
Total params: 747,105
Trainable params: 747,105
Non-trainable params: 0
_________________________________________________________________
None
A total of nearly 7.47 lakh parameters
```

- This architecture has a total of 747,105 parameters which are tuned to generalize for the entire images in the dataset to get optimal solution for every input

General Working

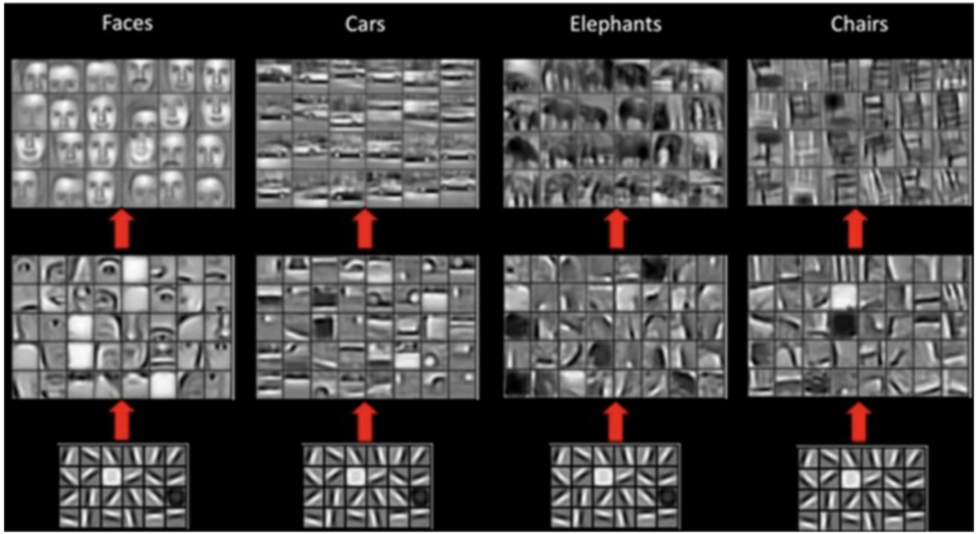






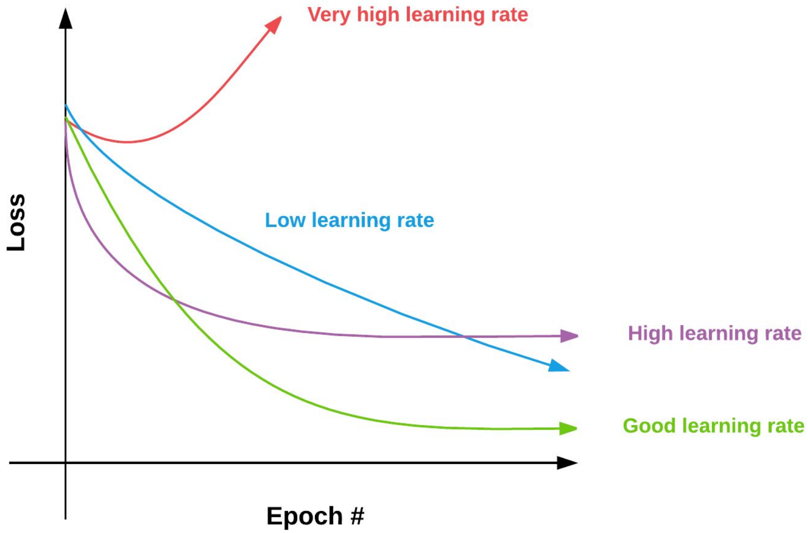
- A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image.

General Working



- Each CNN layer learns filters of increasing complexity.
- The first layers learn basic feature detection filters: edges, corners, etc
- The middle layers learn filters that detect parts of objects. For faces, they might learn to respond to eyes, noses, etc
- The last layers have higher representations: they learn to recognize full objects, in different shapes and positions.

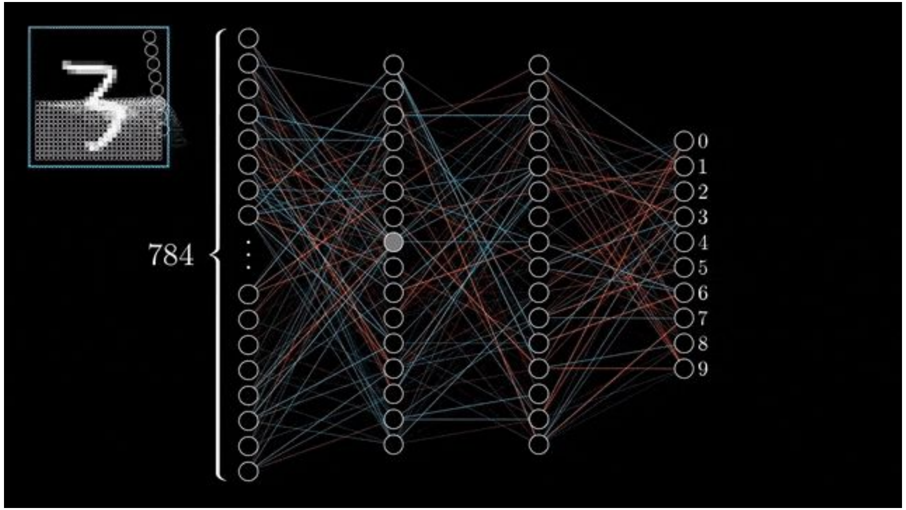
Fine Tuning

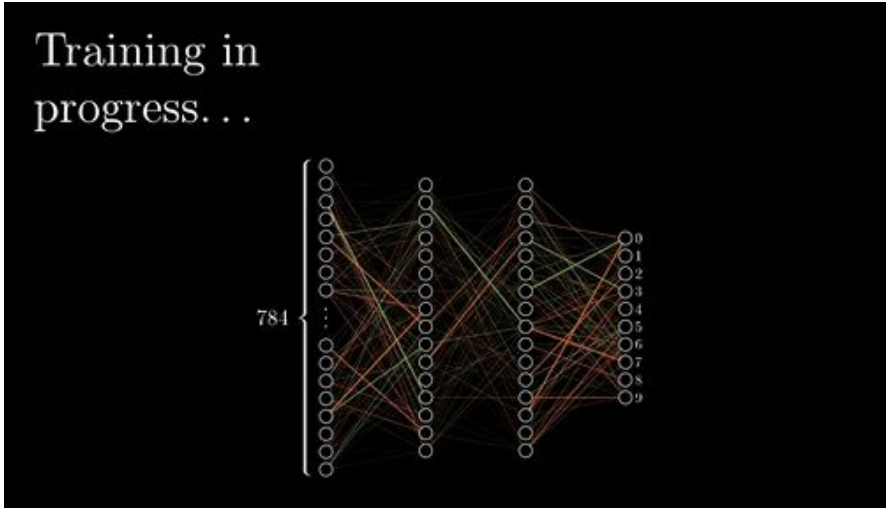



```
1e-08:
Validation loss : 0.22000959515571594
1e-07:
Validation loss : 0.3236052393913269
1e-06:
Validation loss : 0.5467935800552368
1e-05:
Validation loss : 0.33601289987564087
0.0001:
Validation loss : 0.2200288474559784
0.001:
Validation loss : 0.22059911489486694
0.01:
Validation loss : 0.22665756940841675
0.1:
Validation loss : 0.21998214721679688
The most minimum loss is with 0.1 learning rate: 0.21998214721679688
```

- The model is trained for different hyperparameter and we got 0.1 learning rate as the optimal solution for the model.

Training



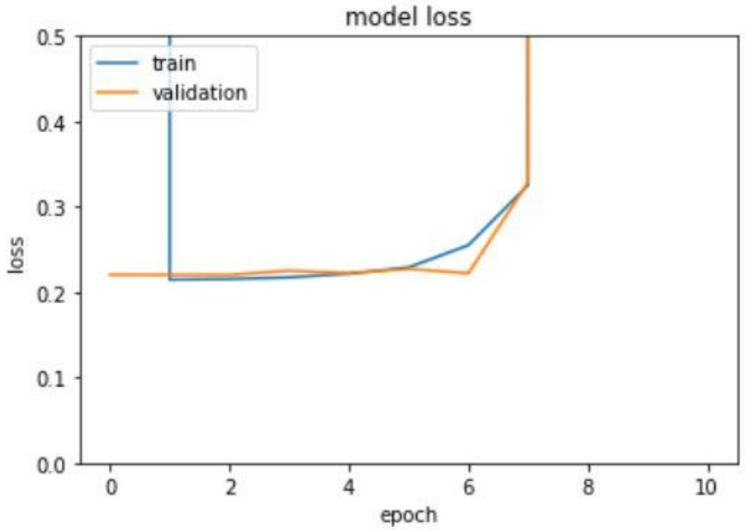


Training

```
Epoch 1/25
1397/1397 [==============================] - 443s 313ms/step - loss: 495554997073553.7500 - accuracy: 0.0279 - val_loss: 0.2203
- val_accuracy: 0.0397
Epoch 2/25
1397/1397 [==============================] - 436s 312ms/step - loss: 0.2118 - accuracy: 0.0383 - val_loss: 0.2202 - val_accurac
y: 0.0397
Epoch 3/25
1397/1397 [==============================] - 425s 304ms/step - loss: 0.2186 - accuracy: 0.0374 - val_loss: 0.2202 - val_accurac
y: 0.0397
Epoch 4/25
1397/1397 [==============================] - 420s 300ms/step - loss: 0.2270 - accuracy: 0.0381 - val_loss: 0.2251 - val_accurac
y: 0.0397
Epoch 5/25
1397/1397 [==============================] - 419s 300ms/step - loss: 0.2235 - accuracy: 0.0378 - val_loss: 0.2221 - val_accurac
y: 0.0397
Epoch 6/25
1397/1397 [==============================] - 418s 299ms/step - loss: 0.2275 - accuracy: 0.0396 - val_loss: 0.2272 - val_accurac
y: 0.0397
Epoch 7/25
1397/1397 [==============================] - 420s 301ms/step - loss: 0.2395 - accuracy: 0.0378 - val_loss: 0.2221 - val_accurac
y: 0.0397
Epoch 8/25
1397/1397 [==============================] - 423s 303ms/step - loss: 0.2987 - accuracy: 0.0364 - val_loss: 0.3288 - val_accurac
y: 0.0397
Epoch 9/25
1397/1397 [==============================] - 423s 303ms/step - loss: 29219.5725 - accuracy: 0.0292 - val_loss: 540.3596 - val_a
ccuracy: 0.0397
Epoch 10/25
1397/1397 [==============================] - 424s 303ms/step - loss: 639426.4395 - accuracy: 0.0211 - val_loss: 239806.0312 - v
al_accuracy: 0.0000e+00
Epoch 11/25
1397/1397 [==============================] - 429s 307ms/step - loss: 574850.2629 - accuracy: 0.0182 - val_loss: 12.0982 - val_a
ccuracy: 0.0000e+00
```

- One epoch is when an entire dataset is passed both forward and backward through the neural network only once.
- One batch is a set of images that is parsed at a time.

Training curve



- The model is trained using early stopping in which the validation loss is monitored to get the optimal solution in which we get 0.21 mean squared error.

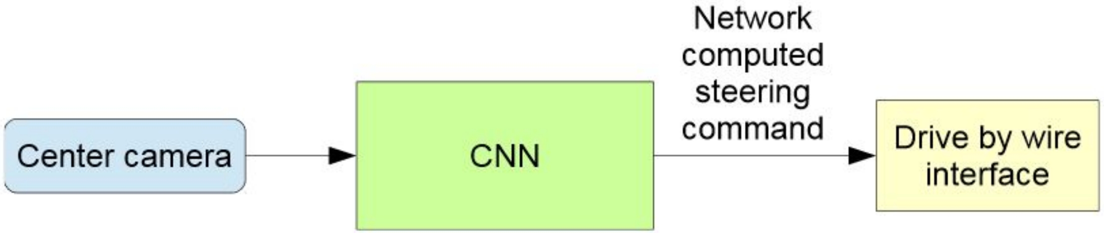
Validation

```
599/599 [==============================] - 22s 37ms/step - loss: 0.2170

0.21695362031459808
```

- The model is validated on data that is not used for training, so that the model does not overfit the data.
- This step is done to know how much the model has generalised to other that has not been used in training.
- We achieved a minimum validation loss of 0.2169

Output



- The trained network is used to generate steering commands for the image from a single front-facing center camera.
- The output is sent to an actuator which actuates the human steering movement automatically

Simulation

- The live feed of a camera is captured and it is converted into images.
- The image is parsed through the convolutional neural network model we have trained.
- These images are parsed at each individual frame level
- The output which is the steering angle is used to rotate the steering wheel image using the cv2 functions.
- Each output is printed to the prompt screen to understand the output of the model
- The live video is played so that we can make sense that the model actually works

Simulation

C:\Windows\py.exe
2.1083325147628784
0.6593618541955948
2.2415920346975327
0.37816448137164116
0.7767069898545742
0.5333627387881279
2.058766633272171
0.9508130326867104
1.2865664809942245
2.0538700371980667
1.8489645794034004
0.7554281316697598
3.4387602657079697
3.277962952852249
2.0314494520425797
3.8453274965286255
3.0534973740577698
3.0496660619974136
0.8195782452821732
1.9164242595434189
2.625654637813568
4.813465476036072
3.4793682396411896
5.063785761594772
5.1011623442173
4.559437483549118
5.802392363548279
4.667009711265564
5.261173099279404
4.267347157001495
6.049876660108566
6.4882828295230865
6.6038694977760315
4.286547303199768
7.797412276268005
7.919551134109497
7.692037224769592
8.026041984558105
7.921048700809479
7.879031002521515
7.475718855857849
9.045480787754059
9.742458164691925
11.168822050094604
11.507451832294464
11.586582362651825
12.043780088424683
11.284188330173492
11.683005094528198
frame
2017-02-04 08:47:32
BLACKVUE DR650GW-2CH/FHD-HD
steering wh...

Conclusion

- Our model is able to learn the entire task of lane and road following without manual decomposition into road or lane marking detection, semantic abstraction,path planning, and control.
- The CNN is able to learn meaningful road features from a very sparse training signal (steering alone). The system learns for example to detect the outline of a road without the need of explicit labels during training.
- The robustness of the method and the internal processing steps in visualization will give a clear idea.

Further improvements

- Often driving steering angles depend upon the previous angle actuated by the human, so feasibility of the models that can remember previous states like LSTM, GRU, etc.
- A separate CNN should be trained for traffic sign classification which will be able to give signals to the autonomous car when to stop, what speed to follow, etc.
- Also some warning signals like steep road ahead, curvy road ahead can be used to alert the driver as well as modify the steering angle which can give a heads up to our end to end approach.

THANK YOU ALL !!!