Steering Angle Prediction in Self Driving Cars Using VGG16 and LSTM Networks

Capstone Project Paper

May 8th, 2018

Arun Reddy Bollam

Sri Manikanta Nukala

Megha Sainath Reddy Vuyyuru

Jaspreet Singh Kohli

Volgenau School of Engineering

Data Analytics and Engineering

DEAN 690 - Spring 2018

Table of Contents

Table of Figures

1 Project Definition

1.1 Problem Space:

In recent years, Autonomous driving vehicles have changed the perception of travel. with the increase in the number new vehicles coming on to roads each year, the question of safety arises. In the timeline of developments in vehicle safety, several technologies have been introduced to ensure safety such as ABS, Blind spot monitoring, Assisted braking, proximity sensors etc. A plethora of sensors and on-board computers in the vehicle ensured that they provided all the necessary assistance required to person who is driving the vehicle. As we progressed through the technology era, Artificial intelligence was introduced into the automobile space and with the data generated from sensors combined with the computing power and state of the art algorithms, complete autonomous vehicles were introduced. There are several research aspects in this space ranging from lane, road detection to prediction of Collison and self-maneuvering. In our project we chose to address the problem of steering angle in autonomous vehicles. In particular self-driving cars with the help of latest state of the are deep learning techniques.

1.2 Primary User stories:

Based on the context of AI powered vehicles, we narrow our project to software aspect of Autonomous driving features and focus on the area of image recognition and steering angle prediction where we will try the concept of steering angle prediction based on data input from various sensors of the car which are in variety of formats. Our project is a unique application of implementing two different state of the art deep leaning models by daisy chaining them.

1.3 Solution Space: Our end goal of the project falls in the domain of features that provides feedback to the driver or make autonomous decisions in real time to avoid Accidents and also, at the same time provide various safety features to the drivers.

1.4 Product Vision: The system can be implemented in all the autonomous vehicles as a standard feature. It can be extended to all categories of vehicles ranging from cars to trucks. This system is relatively inexpensive compared to the alternatives in the market. unlike other systems that uses variety of inputs in a complex structure, this can be implemented relatively easy. The main caveat in using this system is there is a lot riding on the model reliability for various scenarios. The potential to cause harm in case of a small steering angle error is huge when there is

a 2-4-ton vehicle involved in real life situations. Passing all the required regulations in safety is crucial.

1.5 Use Case Scenarios:

1.5.1 Autonomous Parking

An exemplary use case of the autonomous valet parking is depicted in Fig benefit the driving robot parks the vehicle at a remote location after the passengers have exited and cargo has been unloaded. The driving robot drives the vehicle from the parking location to a desired destination. The driving robot re-parks the vehicle. The driver saves the time of finding a parking spot as well as of walking to/from a remote parking spot. In addition, access to the vehicle is eased (spatially and temporally). Additional parking space is used more efficiently and search for parking is used more efficiently. (Walther Wachenfeld, 2016)

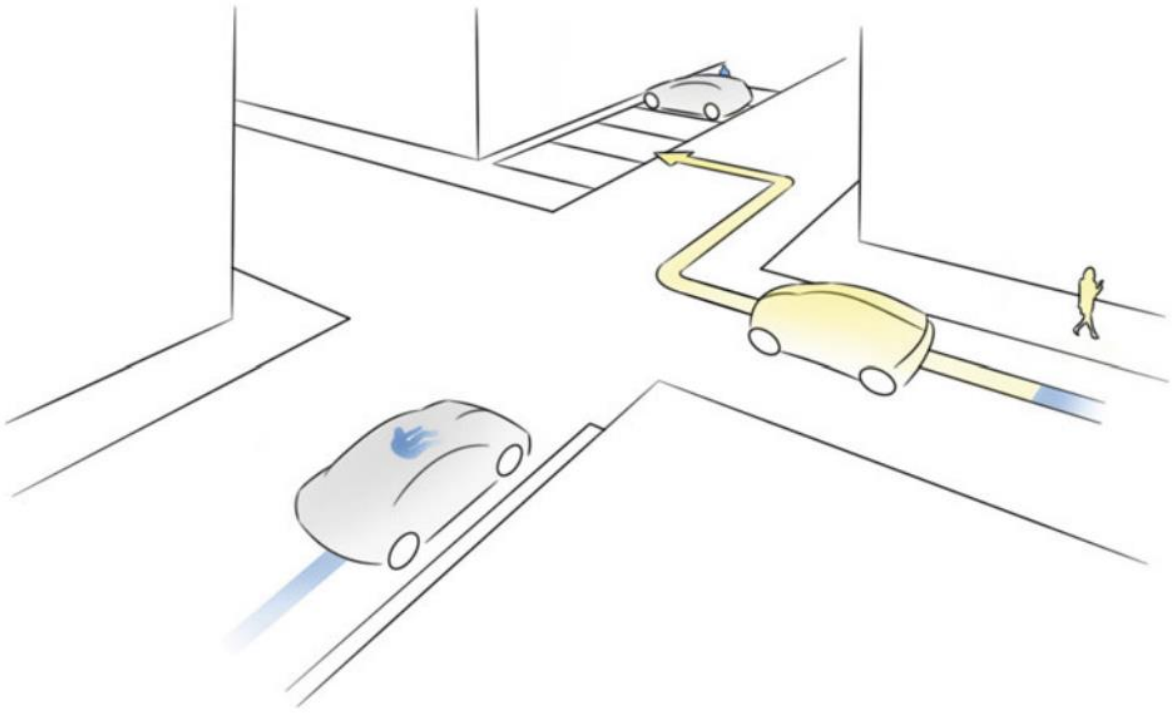

Figure 1: Autonomous Valet Parking

1.5.2 Interstate Pilot Using Driver for Extended Availability:

An exemplary use case of the interstate pilot is depicted in Fig. 2. Benefit The driving robot takes over the driving task of the driver exclusively on interstates or interstate-like expressways. The driver becomes just a passenger during the autonomous journey, can take his/her hands off of the steering wheel and pedals, and can pursue other activities. Description As soon as the driver has entered the interstate, he/she can, if desired, activate the driving robot. This takes place most logically in conjunction with indicating the desired destination. The driving robot takes over navigation, guidance, and control until the exit from or end of the interstate is reached. The driving robot safely coordinates the handover to the driver. If the driver does not meet the requirements for safe handover, e.g. because he/she is asleep or appears to have no situation awareness, the driving robot transfers the vehicle to the risk-minimal state on the emergency lane or shortly after exiting the interstate. During the autonomous journey, no situation awareness is required from the occupant. Because of simple scenery and limited dynamic objects, this use case is considered as an introductory (Walther Wachenfeld, 2016)

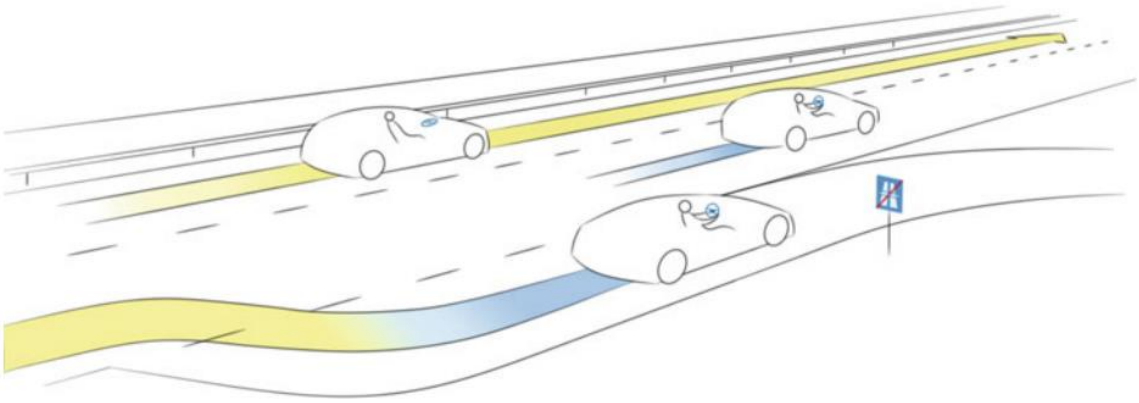

Figure 2 : Interstate Pilot Using Driver for Extended Availability

2 Dataset

2.1 Overview:

The Data for our project was released as a challenge by Udacity open source self-driving car project initiative. The data comes from the installed sensors on a 2016 Lincoln MKZ. sensors and parts: 2 Velodyne VLP-16 LiDARs, 1 Delphi radar, 3 Point Grey Blackfly cameras, an Xsens IMU, an ECU, a power distribution system, and several other peripherals. (Cameron, 2016)

The entire system was configured on the ROS (Robot Operating System). There are two sets of data released, one training set with 4.4 GB of driving images captured in various times of day and driving scenarios. The test set contains 450mb file of 280 sec duration video on which we implement our model.

2.2 Field Description:

The field descriptions for training data is as follows

- HMB_1: 221 seconds, direct sunlight, many lighting changes. Good turns in beginning, discontinuous shoulder lines, ends in lane merge, divided highway
- HMB_2: 791 seconds, two lane road, shadows are prevalent, traffic signal (green), very tight turns where center camera can't see much of the road, direct sunlight, fast elevation changes leading to steep gains/losses over summit. Turns into divided highway around 350s, quickly returns to 2 lanes
- HMB_4: 99 seconds, divided highway segment of return trip over the summit
- HMB_5: 212 seconds, guardrail and two-lane road, shadows in beginning may make training difficult, mostly normalizes towards the end
- HMB_6: 371 seconds, divided multi-lane highway with a fair amount of traffic.

Figure 3 : Data set Images

2.3 Data Conditioning and Quality assessment:

As the data for our project is produced by the Udacity team itself, it is free from noise. Therefore, no need of conditioning required. The only thing that have to be maintained is, as the test data is video file, we must ensure that the processing speed to be greater than or equal to 15 frames per second. This can be done on systems with a GPU, Higher processor and a 16gb memory RAM. We chose to use the AWS deep learning AMI with Linux configuration with NVIDIA CUDA Architecture.

2.4 Other Data sources:

There are several other Data sources that provide data of real time driving conditions. Some famous alternatives are

1. KITTI Vision Benchmark suite: Released by Karlsruhe institute and Toyota motors. 6 hours of traffic scenarios at 10-100 Hz using a variety of sensor modalities such as high-resolution

color and grayscale stereo cameras, a Velodyne 3D laser scanner and a high-precision GPS/IMU inertial navigation system. (Geiger, 2018)

Figure 4: KITTI Vision Benchmark Images

2. Comma.ai - 7 and a quarter hour of largely highway driving. Consists of 10 videos clips of variable size recorded at 20 Hz with a camera mounted on the windshield of an Acura ILX 2016. In parallel to the videos, also recorded some measurements such as car's speed, acceleration, steering angle, GPS coordinates, gyroscope angles. These measurements are transformed into a uniform 100 Hz time base. (Comma.ai, 2016)
3. CSSAD Dataset - Several real-world stereo datasets exist for the development and testing of algorithms in the fields of perception and navigation of autonomous vehicles. However, none of them was recorded in developing countries and therefore they lack the particular characteristics that can be found in their streets and roads, like abundant potholes, speed bumpers and peculiar flows of pedestrians. This stereo dataset was recorded from a moving vehicle and contains high resolution stereo images which are complemented with orientation and acceleration data obtained from an IMU, GPS data, and data from the car computer. (Hayet, 2016).

2.5 Existing Research and Algorithms:

2.5.1 Neural Networks:

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning

process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well (Siganos, 2016).

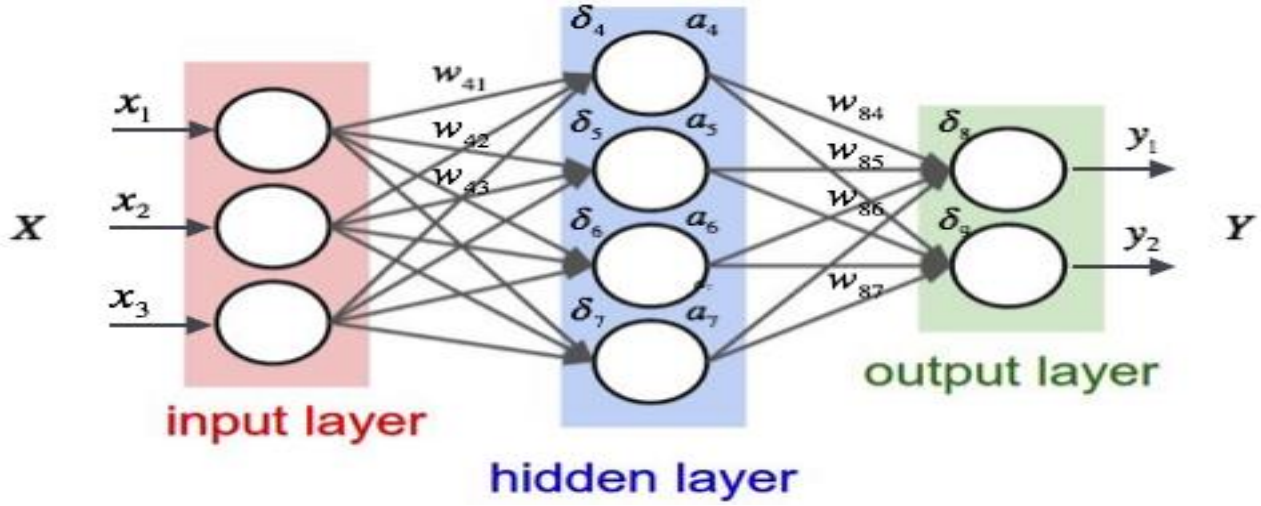

Figure 5: Basic Neural Network

2.6 Convolutional Neural Network:

Convolutional Neural Networks are very similar to ordinary Neural Networks, they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The difference to a normal neural network is, Convolutional Net architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the number of parameters in the network. (Karpathy, 2016)

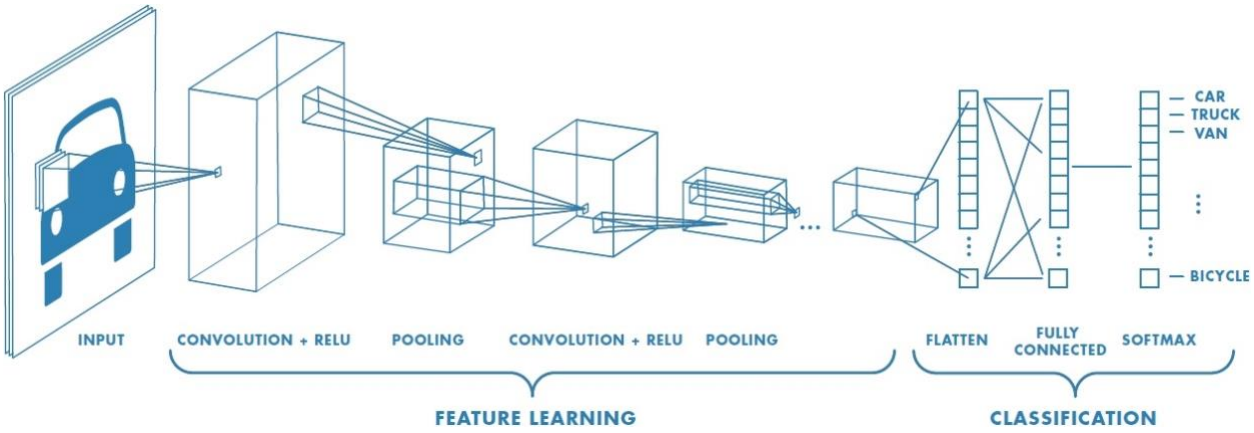

Figure 6: Convolutional Neural Network

YOLO (You only look once):

YOLO is a state-of-the-art, real-time object detection system. A single neural network is applied to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

In the context of self-driving cars, the images of the highway are gathered using an onboard dashcam which would then be linked to a real time object detection system which has the capability of processing images at a rate of 40-90 frames per second with a mean average precision on VOC 2007 of 78.6% and a mean average precision of 48.1% on COCO test-dev. This detection system of "YOLO" (YOU ONLY LOOK ONCE) is a massive Convolutional Neural architecture for object detection and classification based on the confidence scores and the study of box co-ordinates of the object being referred to in the frame. Here, confidence score refers to the probability of any object present in the bounding box whose coordinates are predefined. Some of the output examples are as follows (Redmon, 2016).

Figure 7 : YOLO Working Example

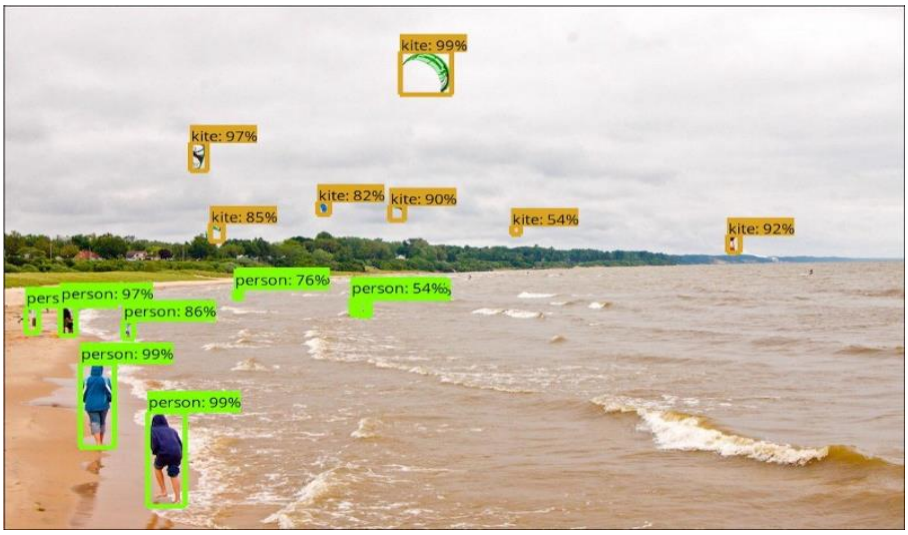

Figure 8 : YOLO Example

There are also various other detection system API's which would be used such as the TensorFlow Object Detection API. This system has an advantage of providing the output without the need of setting up a complex hardware which is the case usually seen while dealing with YOLO. While dealing with YOLO we are required to pre-set a GUI and a controller which would facilitate the flow of information from YOLO and road lane detector into the data model. The road lane detector would consist of four main parts which are warping, filtering, detecting the road lane, and de-warping the images being flown into the system. The width of the highway lane would be pre-fixed, and techniques of neural networks would then be applied after studying the position and speed of various objects on the highway. Once the configuration of YOLO is done the controller would ask for the width of the lane and then the real time information would be projected on the screen. Once the information is obtained the visual data is processed with the help of convolutional neural networks which is a class of deep, feed-forward artificial neural networks. The data containing the steering angle for any given visual image is then processed. This data can be called the training data and through processing various layers of this datatype, CNN allows the vehicle to get adapted to the various "known" situations and predict the steering angle in advance.

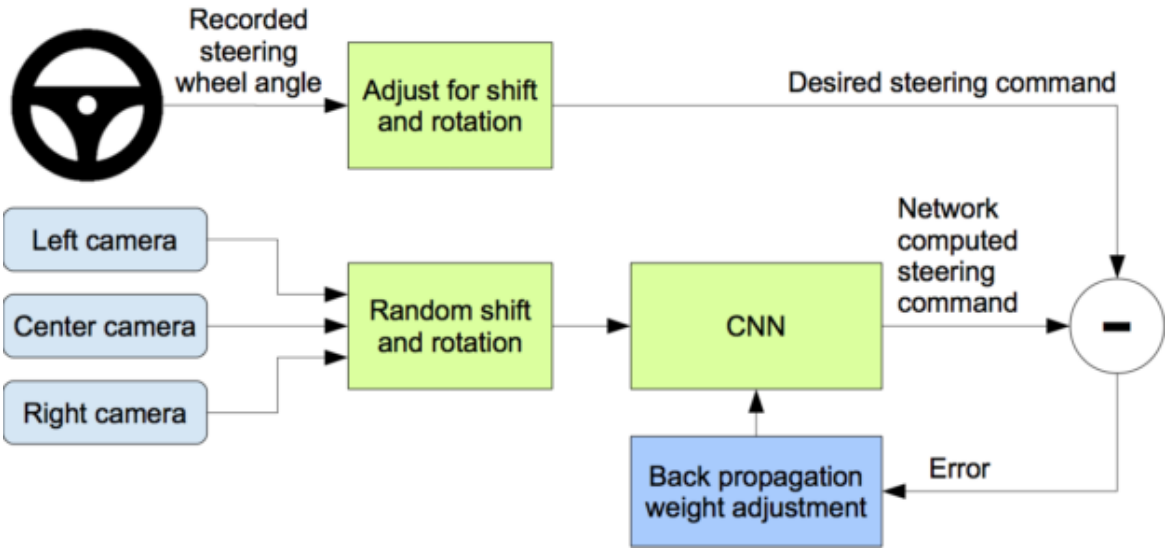

Figure 9 : NVIDIA Steering Angle Outline

Through processing the series of pre-recorded images, the network commands, the simulator to accordingly steer the vehicle. (Mariusz Bojarski, 2016)

2.6.1.1 RESNET:

Res Net or Residual network is the new terminology for residual learning in the field of image classification using Deep convolutional neural networks. In general, in a deep convolutional neural network, several layers are stacked and are trained to the task at hand. The network learns several low/mid/high level features at the end of its layers. In residual learning, instead of trying to learn some features, we try to learn some residual. Residual can be simply understood as subtraction of feature learned from input of that layer. ResNet does this using shortcut connections (directly connecting input of nth layer to some (n+x) th layer. It has proved that training this form of networks is easier than training simple deep convolutional neural networks and also the problem of degrading accuracy is resolved. (Sun, 2015)

Network depth is of crucial importance in neural network architectures, but deeper networks are more difficult to train. The residual learning framework eases the training of these networks and enables them to be substantially deeper—leading to improved performance in both visual and non-visual tasks.

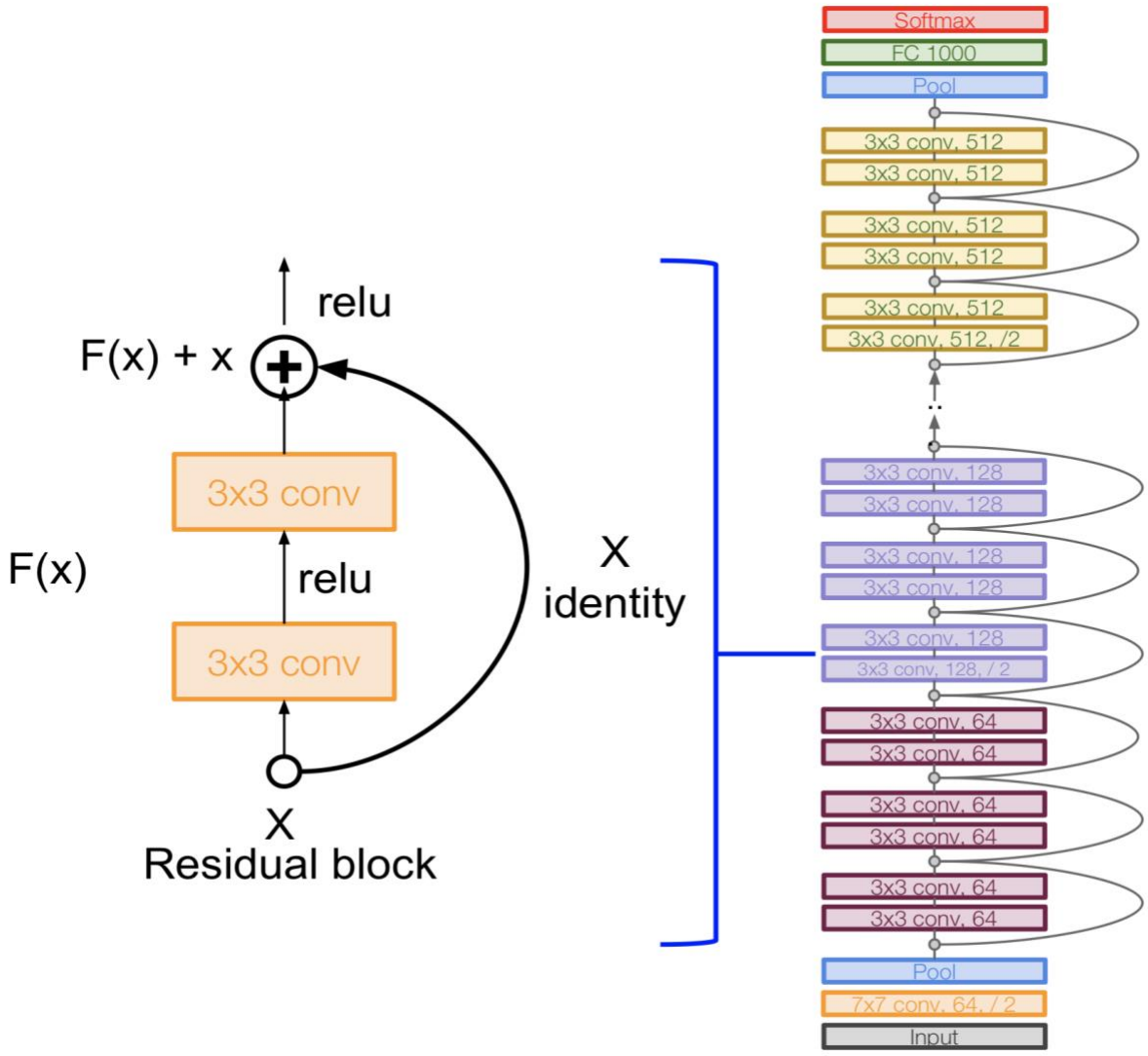

Figure 10 : ResNet Architecture

2.6.1.2 VGG16:

VGG is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.

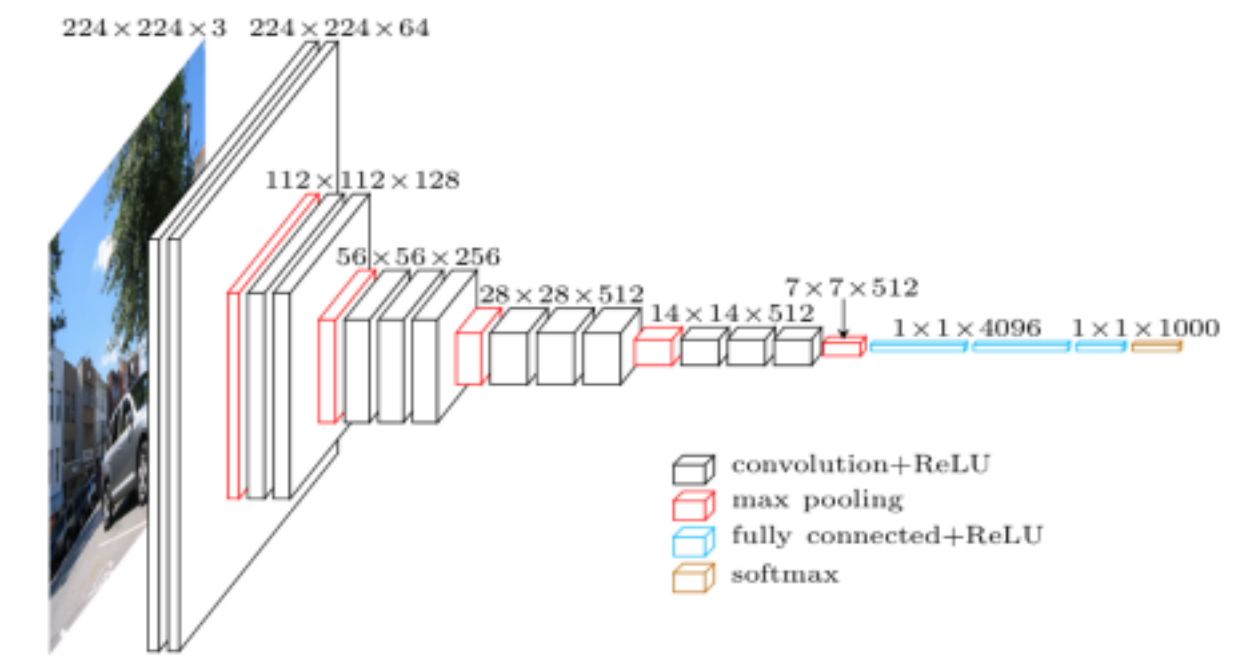

Figure 11 : VGGNet Architecture

2.7 RNN (Recurrent neural network):

RNN or Recurrent Neural Networks are used for image classification and captioning. The classification is usually applied on deeper networks where multiple hidden layers are present. So here, the input layer receives the input, the first hidden layer activations are applied and then these activations are sent to the next hidden layer, and successive activations through the layers to produce the output. Each hidden layer is characterized by its own weights and biases. Here, the weights and bias of these hidden layers are different. And hence each of these layers behave independently and cannot be combined together. To combine these hidden layers together, we shall have the same weights and bias for these hidden layers. (Gupta, 2017)

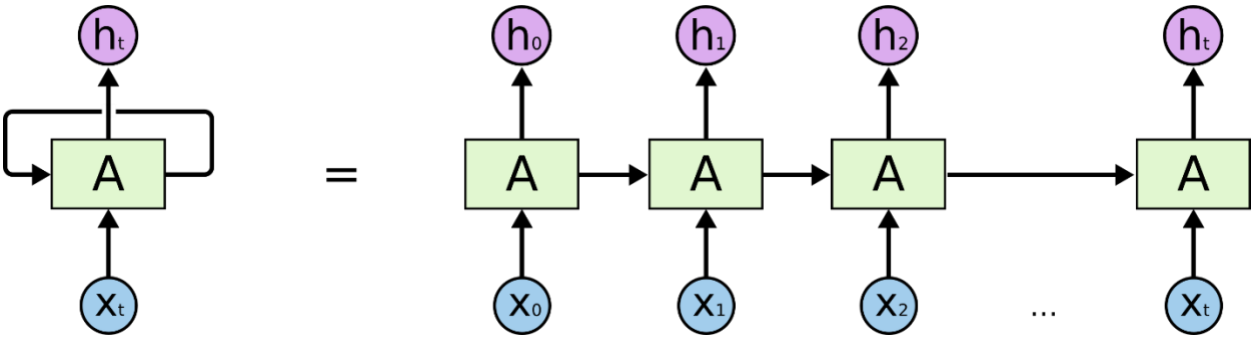

Figure 12: Flow of information in the loop in RNN, where x is input, and h is output

We can now combine these layers together, that the weights and bias of all the hidden layers is the same. All these hidden layers can be rolled in together in a single recurrent layer. So, it's like supplying the input to the hidden layer. At all the time steps weights of the recurrent neuron would be the same since it's a single neuron now. So, a recurrent neuron stores the state of a previous input and combines with the current input thereby preserving some relationship of the current input with the previous input.

Use of RNN in Image classification and labeling:

Whereas CNN is used to generate discriminative features in an image, RNN is used to generate sequential labels. Some of the advantages of using these kinds of networks are:

- Learning things in a hierarchical way is consistent with human perception and concept organization. By predicting the labels in a coarse-to-fine pattern, we can better understand what the images portray.
- It is transferrable. In principle, the framework can be built on top of any CNN architecture which is primarily intended for single-level classification and boost the performance for each hierarchical level.
- The number of the hierarchical labels can be variable. The flexibility of RNN allows us to generate hierarchical labels of different lengths, i.e. more specific categories would have more hierarchical labels.

While CNN has proven to be successful in processing image-like data, RNN is more appropriate in modeling sequential data. For our project we have used the cascaded. This combination process's the CNN and RNN separately, where the RNN takes the output of CNN as input and returns sequential predictions of different timesteps. These frameworks are often intended for different tasks, rather than image classification, such as sequential prediction in hierarchical image frameworks.

2.8 Advantages of LSTM over RNN:

Long Short-Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn. LSTM are generally used to model the sequence

data. That means, one can model dependency with LSTM model, whereas RNN is a class of artificial neural networks where connections between units form a directed cycle, as shown below.

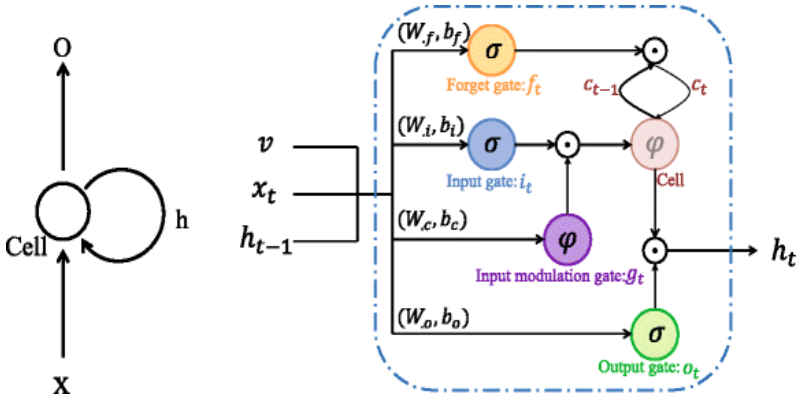

Figure 13 : The pipeline of RNN(left) and LSTM(right)

It can effectively model the dynamic temporal behavior of sequences with arbitrary lengths. However, RNN suffers from the vanishing and exploding gradient problem since the gradients need to propagate down through many layers of the recurrent network. Therefore, it is difficult to model the long-term dynamics. In contrast, Long-Short Term Memory (LSTM) provides a solution by incorporating a memory cell to encode knowledge at each time step. Specifically, the behavior of the cell is controlled by three gates: an input gate, a forget gate and an output gate. These gates are used to control how much it should read its input (input gate i), whether to forget the current cell value (forget gate f) and whether to output the new cell value (output gate o). These gates help the input signal to prorogate through the recurrent hidden states without affecting the output, therefore, LSTM can deal well with exploding and vanishing gradients, and effectively model long-term temporal dynamics that RNN is not capable of learning. (Guo, 2018)

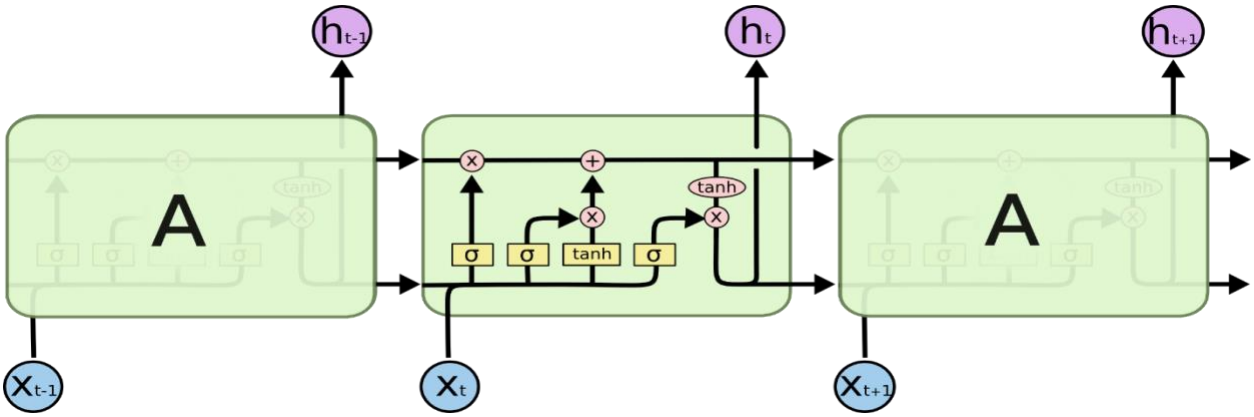

Figure 14: Flow of information in the loop of LSTM module

3 Analytics and Algorithms

3.1 Abstract:

There are variety of Deep learning techniques to Computer vision problems in highway perception and Autonomous vehicle driving scenarios. Application of Computer Vision has become crucial in areas of search, image understanding, apps, mapping, medicine, drones, and in particular self-driving cars. Recent developments in application of deep learning particularly neural networks have greatly improved the performance and efficiency of these state-of-the-art visual recognition systems. In this project, we utilize these recent deep learning advances. Computer vision, combined with deep learning to produce a relatively inexpensive, robust solution to autonomous driving where we combine two different models to perform high quality prediction of steering angle based on images input from the sensors of the car by application of Transfer Learning, LSTM and ResNet. The data we use was released by Udacity open source self-driving car Team. Which consists of 4.4 GB of ROSBAG files with images taken in various lighting conditions and traffic scenarios. We used AWS with deep learning AMI configured for this project.

Keywords: Autonomous vehicles, Recurrent Neural Networks, Transfer Learning, ResNet, LSTM, AWS

3.2 Overview and features:

Incorporating temporal information will play a crucial role in production systems for self-driving cars. We will be designing a 3D Convolution Long Term Short Term Memory architecture to predict the steering angle. This architecture consists of the following aspects:

1) Brightness augmentation and shadow augmentation will be performed using OpenCV to further increase the variability in lighting conditions.
2) 3D convolutional layering will be performed to render a third temporal dimension by view pooling images from 3 different angles i.e., left right and center. Spatial batch normalization will be done after each 3D layer to properly reinitialize this neural network.
3) We will be implementing residual connection to allow more gradients to pass through the network by combing different layers, smoother back propagation, thereby mitigating the vanishing gradient problem.

4) The next step would be to feed these 3D layers into LSTM layers to capture sequential relations. LSTMS have loops in them which allows information to persist. This can be used to understand present frames using previous video frames.

Udacity has released a dataset (Udacity, 2018)of images taken while driving along with the corresponding steering angle data. The system must be capable of driving in many different situations such as varying weather conditions, avoid an obstacle and even going off-road. The MSE (Mean Squared Error) minimization standard is used to benchmark the results with the predicted output against actual output.

We aim to design the model such that it learns the features necessary to predict the steering angle and then learn itself overtime by observing these features.

3.3 Analysis:

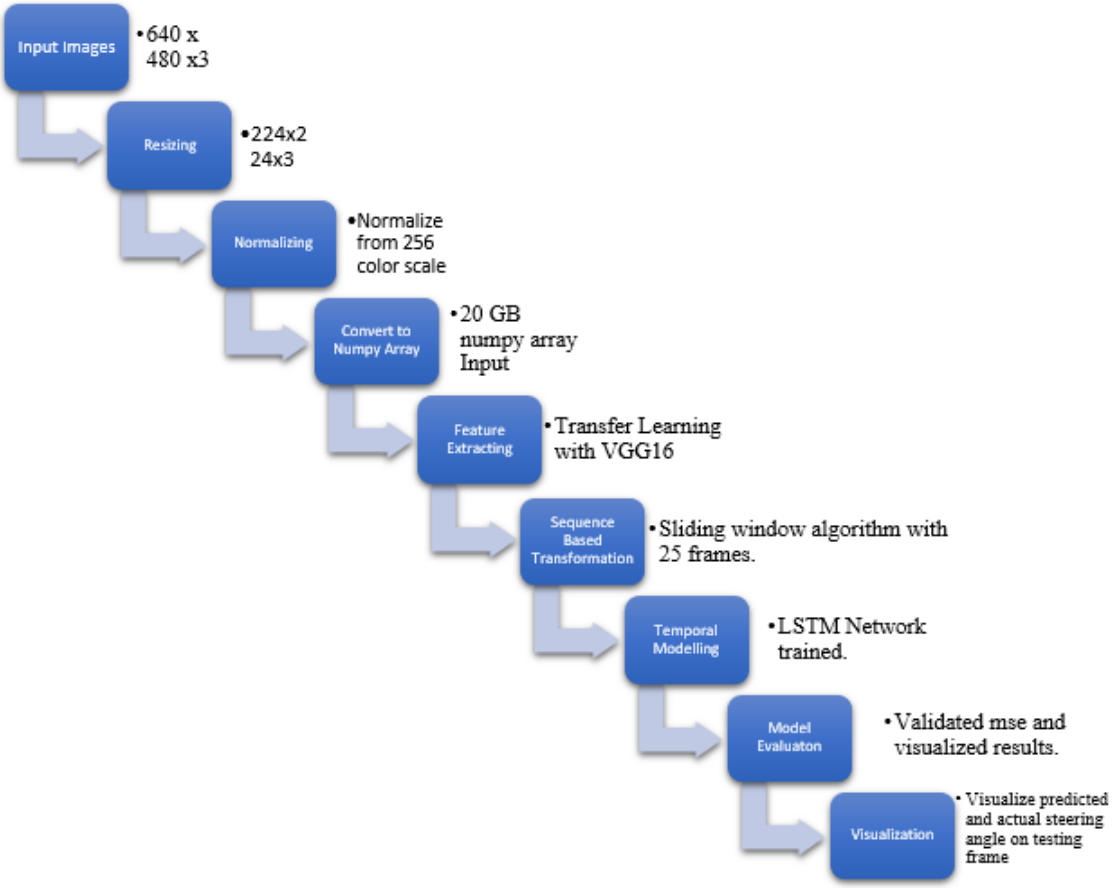

Figure 15 : Analysis Flowchart

Extraction :

Firstly, the extraction process involved installing ROS and running scripts to extract the required data form our bag files. Ross Wightman's repository (Rwightman, 2016) consisted of helpful scripts to speed up this process. The code builds a docker container in which ROS scripting is performed to extract images and other logged driving statistics that were recorded by sensors.

Below is a high-level view of what the recorded driving statistics look like. As we can see there are several statistics available. We will be considering images from the center camera as they tend to capture driving behavior better. Though, it is possible to use statistics like torque and speed to develop out model, we decided to use just filename and angle steering angle (in radians where 1 radian = 57 degrees) to be consistent with the self-driving car challenge rules. This would let us evaluate the results of our model with other competitors. Also, note that the filename is named based on camera and timestamp recorded. Hence, we decided to filter by center camera images and order by ascending time stamp to maintain a sequence. We will be getting into the necessity of the sequence later.

index	timestamp	width	height	frame_id	filename	angle	torque	speed	lat	long	alt
10:15.9	1.47942E+18	640	480	left_came	left/1479424215873184606.jpg	0.00036	0.375	23.00335	37.54527	-122.326	8.116664
10:15.9	1.47942E+18	640	480	right_cam	right/1479424215877284689.jpg	0.000717	0.375	23.00392	37.54527	-122.326	8.12368
10:15.9	1.47942E+18	640	480	center_ca	center/1479424215880976321.j	0.001039	0.375	23.00443	37.54527	-122.326	8.128418
10:15.9	1.47942E+18	640	480	left_came	left/1479424215922817911.jpg	0.003491	0.394737	23.00731	37.54526	-122.326	8.139066
10:15.9	1.47942E+18	640	480	right_cam	right/1479424215927281227.jpg	0.003491	0.380823	23.00607	37.54526	-122.326	8.144778
10:15.9	1.47942E+18	640	480	center_ca	center/1479424215930775951.j	0.003491	0.369925	23.00601	37.54526	-122.326	8.149365
10:16.0	1.47942E+18	640	480	left_came	left/1479424215972815517.jpg	0.003491	0.363861	23.00833	37.54525	-122.327	8.210505
10:16.0	1.47942E+18	640	480	right_cam	right/1479424215977378050.jpg	0.003491	0.349617	23.00833	37.54525	-122.327	8.221686
10:16.0	1.47942E+18	640	480	center_ca	center/1479424215980916687.j	0.003491	0.33857	23.00833	37.54525	-122.327	8.226621
10:16.0	1.47942E+18	640	480	left_came	left/1479424216022820692.jpg	0.00467	0.479716	23.00646	37.54525	-122.327	8.291892
10:16.0	1.47942E+18	640	480	right_cam	right/1479424216027309701.jpg	0.005061	0.493743	23.00583	37.54525	-122.327	8.297856
10:16.0	1.47942E+18	640	480	center_ca	center/1479424216030737492.j	0.005236	0.491111	23.00516	37.54525	-122.327	8.302497
10:16.1	1.47942E+18	640	480	left_came	left/1479424216073039735.jpg	0.005554	0.522755	22.99697	37.54524	-122.327	8.363481
10:16.1	1.47942E+18	640	480	right_cam	right/1479424216077185305.jpg	0.005916	0.548671	22.99985	37.54524	-122.327	8.372955
10:16.1	1.47942E+18	640	480	center_ca	center/1479424216080827865.j	0.006234	0.571443	23.00238	37.54524	-122.327	8.377784
10:16.1	1.47942E+18	640	480	left_came	left/1479424216122856036.jpg	0.012229	0.70862	23.01575	37.54523	-122.327	8.446943
10:16.1	1.47942E+18	640	480	right_cam	right/1479424216127297778.jpg	0.01339	0.736319	23.01636	37.54523	-122.327	8.449328
10:16.1	1.47942E+18	640	480	center_ca	center/1479424216130778849.j	0.014297	0.742006	23.01596	37.54523	-122.327	8.454563
10:16.2	1.47942E+18	640	480	left_came	left/1479424216172794699.jpg	0.021772	0.677618	23.01155	37.54523	-122.327	8.485252
10:16.2	1.47942E+18	640	480	right_cam	right/1479424216177269210.jpg	0.022943	0.663639	23.01217	37.54522	-122.327	8.497931
10:16.2	1.47942E+18	640	480	center_ca	center/1479424216180687241.j	0.023837	0.652962	23.01265	37.54522	-122.327	8.502451
10:16.2	1.47942E+18	640	480	left_came	left/1479424216222706305.jpg	0.031946	0.62778	23.02041	37.54522	-122.327	8.56729
10:16.2	1.47942E+18	640	480	right_cam	right/1479424216227198494.jpg	0.032728	0.585785	23.01979	37.54522	-122.327	8.573053
10:16.2	1.47942E+18	640	480	center_ca	center/1479424216230735818.j	0.033161	0.5625	23.01901	37.54522	-122.327	8.577695
10:16.3	1.47942E+18	640	480	left_came	left/1479424216272749590.jpg	0.032896	0.399492	23.01473	37.54521	-122.327	8.641068
10:16.3	1.47942E+18	640	480	right_cam	right/1479424216277232525.jpg	0.032505	0.343541	23.01598	37.54521	-122.327	8.651324
10:16.3	1.47942E+18	640	480	center_ca	center/1479424216280736012.j	0.0322	0.299817	23.01695	37.54521	-122.327	8.655951

interpolated

Figure 16 : Driving Log File with Timestamps

Initial Glimpse:

Below is a distribution of our steering angles. We can see that the overall driving experience is Moderate. Most of the driving is performed on straight roads.

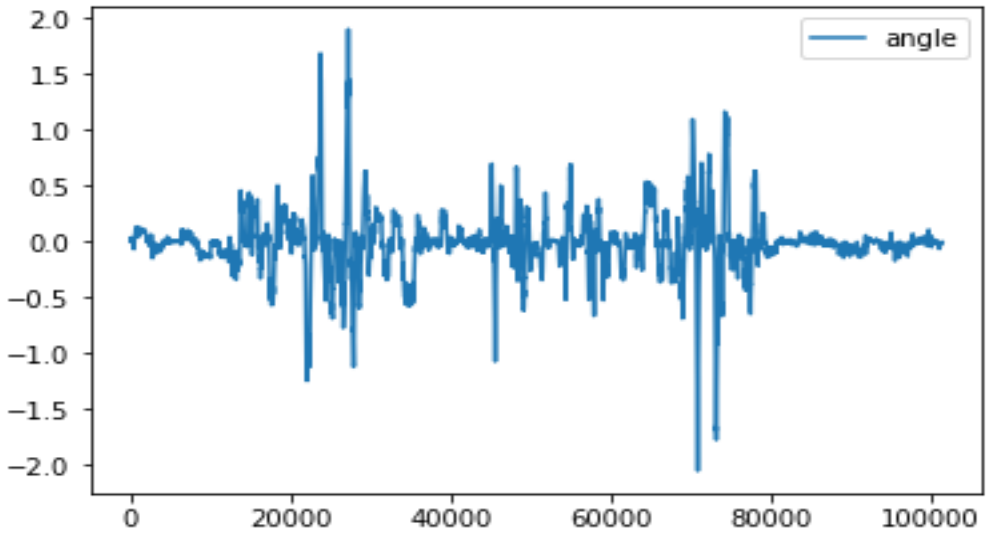

Figure 17 Steering Angle Distribution

Image Processing

Initial images were sized 640 x 480 x3. Because the input to the deep learning network in the next stage requires, resized our images to height of 224 pixels, width of 224 pixel and 3 channels. The images were one a 256-color scale were normalized, and results were converted into a numpy arrays which will be the input to the next step. The size of the resulting numpy array obtained was 20 GB.

Transfer Learning

```
In [21]: model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
=================================================================
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0
_________________________________________________________________
```

Figure 18 VGG16 Summary

We used pre-trained VGG 16 model for extracting relevant features from the above input. This model has been loaded from the Keras Module. The model was frozen from training and only the feed forward functionality was used till the last fully connected layer for feature extraction purposes. The final dense layer which performs classification has been removed and the output of the feedforward, which is a (4096,1) numpy array of all the images have been stored in an array, to be fed as input to the next step.

Creating Sequences Using Sliding Window Algorithm:

The next step was to process our inputs for the LSTM networks. The output from VGG16 model was processed into sequences of 15 frames each using sliding window algorithm with window size of 15. For example, first sequence will have the features of 1 to 15 images, second sequence will have the features of 2 to 16 images. Each sequence has a dimension of (15, 4096).

Training LSTM Network

The Sequence of the features of the images created in the previous step were fed as the inputs to the LSTM. The label for each sequence is the steering angle of the first image in the sequence. The LSTM was a 4 layered network and batch normalization was used after the input layer to normalize existing variations in the input which existed due to variance in driving conditions found in the video frames. The LSTM network was trained in 40 epochs using Adam Optimizer.

```
In [35]: lstm_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization_1 (Batch (None, None, 4096)        16384
_________________________________________________________________
lstm_1 (LSTM)                (None, 256)               4457472
_________________________________________________________________
dense_1 (Dense)              (None, 64)                16448
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 4,492,417
Trainable params: 4,484,225
Non-trainable params: 8,192
_________________________________________________________________
```

Figure 19 LSTM Model Summary

3.4 RESULTS

Training Results:

We achieved an exceptional training mse of 0.00123 radians. From plotting the predicted results against the actual results, we can see that our model has captured all the variations in the data.

```
In [41]: #Train mean squared error
         print('train_mean_squared_error')
         mean_squared_error(y_train_5, train_predictions)

         train_mean_squared_error

Out[41]: 0.0012379788744759416
```

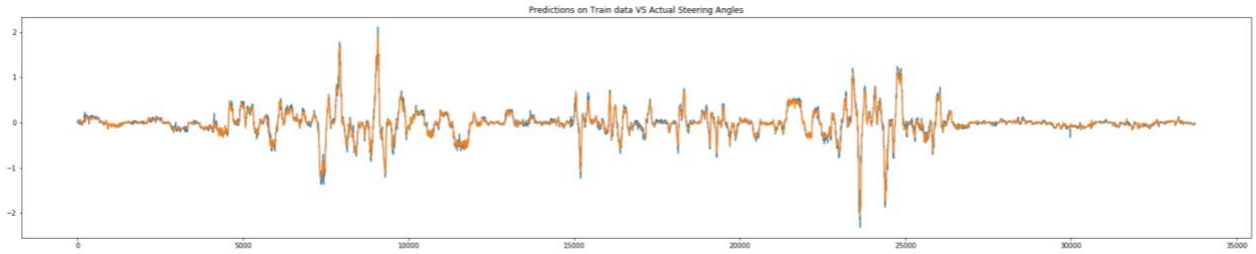

Figure 20: Training Predictions Vs Actual Steering Angles.

Testing Result:

We achieve a testing mean squared error of 0.029 radians. On visualizing our results we see that our model predicts exceptionally well.

```
In [40]: #Test Mean squared error
         from sklearn.metrics import mean_squared_error
         print('test_mean_squared_error')
         mean_squared_error(y_test_5,final_predictions)

         test_mean_squared_error

Out[40]: 0.02958017350833886
```

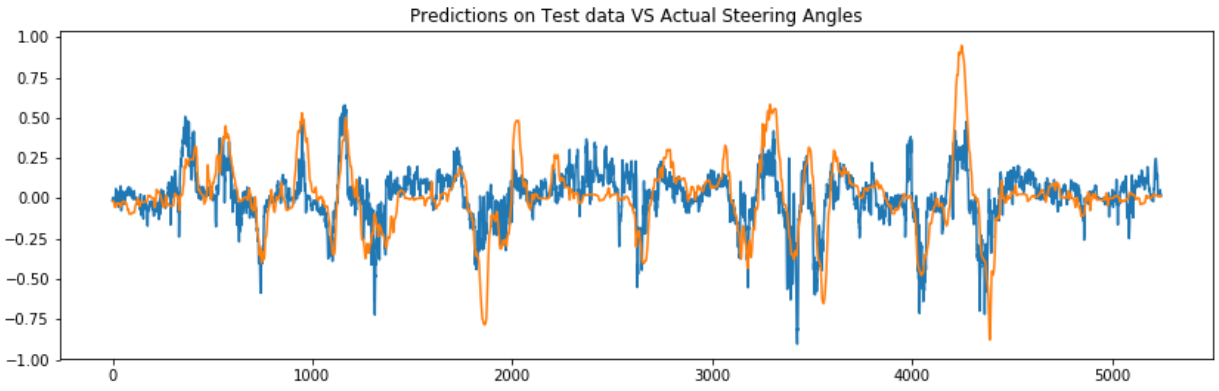

Figure 21: Testing Predictions VS Actual Steering Angles

On comparing with the Self driving car challenge 2's public leaderboard our mse score made its mark in the 8th place.

Rank	Team Name	Public Score	Private Score
1	komanda	0.019354826554271058	0.019000886503484679
2	rambo	0.024043031353416212	0.023139445736868425
3	chauffeur	0.024799567374978741	0.024630170556960426
4	rwightman	0.025382078860711952	0.024652455035466161
5	epoch	0.029204824313005848	0.028311347483058014
6	lookma	0.029812086032704476	0.028358303595130471
7	andre	0.02993788506224929	0.028726047921710424
8	ai-world	0.030237998400137252	0.028816583237209977
9	autumn	0.030146843269163438	0.029024739076791626
10	three-musketeers	0.031684487565107521	0.029599617257261870
11	fsc	0.031239015007820548	0.030011184271148059
12	jeffd23	0.031372997721939133	0.030040979056168659
13	lab	0.032098858758217365	0.031068155110669536
14	team-sf	0.034779221439476628	0.033373129399809916
15	bitas	0.035121636548452885	0.033737833847975851

Figure 22 : LeaderBoard Comparison

3.5 Summary:

The ability to operate a self-driving car would require an education on the driver's part. While the computer takes over once the vehicle is operational, the driver would still be required to maintain some knowledge about how to operate it safely. so, with the developments till date, complete autonomous vehicle standard is not allowed by various countries. Also, by implementing such systems, the cost of the car would significantly go up because the cost of implementing the new technology is very higher and Currently, the engineering, power and computer requirements, software, and sensors add up to more than $100,000.

3.6 Future Work:

With the recent developments in variety state of the art algorithms and data processing power there are more efficient and accurate systems being developed. The amount of GPU's used for computing are bound to increase and also evolve such that they can be placed in the vehicle itself without space constraints. Therefore, the future of this project with respect to the self-driving car domain is to be adaptive and open to all the new technologies. Also, the safety aspect will be the

main issue as there are real lives of persons involved, hence concentrating on developing robust safety regulations can be of help.

4 Appendix A

4.1 Risks:

Less Adoption of technology: The savings in terms of cost, time, and lives is going to come from when more people "opt in" to the service. If self-driving cars are not adopted widely, accidents can and will still happen.

Security Issues: The security behind self-driving cars would be a major obstacle, especially because of the growing technology, the safety systems are not in par with the advancements. This would be of very high interest to hackers to do any kind of unethical things.

Privacy issues: keeping track of every detail and storing the data on the computer on board in to operate a vehicle can be an issue of privacy. Some individuals are concerned about the opportunity for a computer built into the self-driving car to collect personal data.

Unemployment: Self-driving cars would eliminate many jobs in the transportation sector, especially when it comes to freight transportation and taxi drivers. This could have a negative impact on the unemployment rate and the economy.

4.2 Opportunities:

Since 81 percent of car crashes are the result of human error, computers would take a lot of danger out of the equation entirely. Computers use complicated algorithms to determine appropriate stopping distance, distance from another vehicle and other data that decreases the chances of car accidents dramatically. There would be a significant cost savings in many different venues like insurance costs and healthcare costs associated with accident recovery. The productivity of drivers would increase in doing another work instead of being focused on driving. Self-driving would significantly improve traffic conditions and congestion by optimizing the routes based on traffic patterns. This would help to reduce commute times for drivers in high-traffic areas but also to maximize on gasoline usage. Also, Disabled individuals, who have to rely on public transportation or assistance from others to can be hugely benefitted.

References:

Cameron, O. (2016). Open Source Self-Driving Car. Retrieved from Medium Inc.: https://medium.com/udacity/were-building-an-open-source-self-driving-car-ac3e973cd163

Comma.ai. (2016). comma.ai driving dataset. Retrieved from https://archive.org/details/comma-dataset

Geiger, A. (2018). The KITTI vision benchmark suite. Retrieved from http://www.cvlibs.net/datasets/kitti/

Guo, Y. L. (2018). CNN-RNN: a large-scale hierarchical image classification framework. Multimedia Tools and Applications, 10251-10271.

Gupta, D. (2017). Fundamentals of Deep Learning – Introduction to Recurrent Neural Networks. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

Hayet, J.-B. (2016). The CCSAD dataset. Retrieved from Towards Ubiquitous Autonomous Driving: The CCSAD Dataset: http://aplicaciones.cimat.mx/Personal/jbhayet/ccsad-dataset

Karpathy, A. (2016). CS231 stanford. Retrieved from Convolutional Neural Networks For Visual Recognition: http://cs231n.github.io/convolutional-networks/#conv

Mariusz Bojarski, B. F. (2016). End-to-End Deep Learning for Self-Driving Cars. Retrieved from https://devblogs.nvidia.com/deep-learning-self-driving-cars/

Redmon, J. a. (2016). YOLO : Better, Faster, Stronger. Retrieved from Darknet yolo: https://pjreddie.com/darknet/yolo/

Rwightman. (2016). Udacity Driving Reader. Retrieved from githib: https://github.com/rwightman/udacity-driving-reader

Siganos, C. S. (2016). Neural Networks. Retrieved from ic.ac.uk: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#Introduction%20to%20neural%20networks

Sun, K. H. (2015). Deep Residual Learning for Image Recognition. seattle : Microsoft Research.

Udacity. (2018). self driving car data sets. Retrieved from github: https://github.com/udacity/self-driving-car/tree/master/datasets

Walther Wachenfeld, H. W. (2016). Use Cases for Autonomous Driving. Darmstadt: Springer.