SELF DRIVING CARS USING CONVOLUTIONAL NEURAL NETWORKS AND DEEP LEARNING

PROJECT REPORT

Submitted for the course: Technical Answers For Real-World Problems (TARP)

(ECE3999)

By

NAME	REG NO.	SLOT
S.ARUN KARTHIK	17BEC0114	TA1
S P SHARVARI	17BEC0458	TA1
ADITYA SHARMA	17BEC0633	TA1
A T RUTHVIK SRINIVAS	17BIS0101	TA1

NAME OF FACULTY: PROF. Arunkumar Chandrasekhar

(SCHOOL OF ELECTRONICS ENGINEERING**)**

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SELF DRIVING CAR

A self-driving car, also known as an autonomous car, driverless car, or a robotic car, is a vehicle that is capable of sensing its environment and moving safely with little or no human input.

ABSTRACT

- Driverless cars stand to solve all sorts of problems, like traffic delays and traffic collisions caused by driver error.
- Autonomous vehicles will bring to market all sorts of new and exciting applications for a variety of industries, like shipping, transportation, and emergency transportation.
- The ultimate goal of self-driving cars is to delegate the responsibility of driving to a machine.
- Here, we use a neural network to clone car driving behavior. It is a supervised regression problem between the car steering angles and the road images in front of a car.

OBJECTIVE

- To train an end-to-end deep learning model that would let a car drive by itself around the track in a driving simulator.
- We'll use Udacity's driving simulator which has two different tracks. One of them will be used for collecting training data, and the other one — never seen by the model — as a substitute for the test set.
- For training our model, we would drive the car using arrow keys in the training track and we collect the data by recording in the simulator.
- Now after we train our model, we would evaluate its performance on a completely different 'testing track' where the car will be made to run autonomously.

LITERATURE SURVEY

1. **Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J. and Zhang, X., 2016. End to end learning for self-driving cars.**

   In the paper, the authors used a CNN architecture to extract features from the driving frames. The network was trained using augmented data, which was found to improve the model's performance. Shifted and rotated images were generated from the training set with corresponding modified steering angles.

This approach was found to work well in simple real-world scenarios, such as highway lane-following and driving in flat, obstacle-free courses. Several research efforts have been undertaken to build more complex perception-action models to tackle the myriad of environments, and unpredictable situations usually encountered in urban environments.

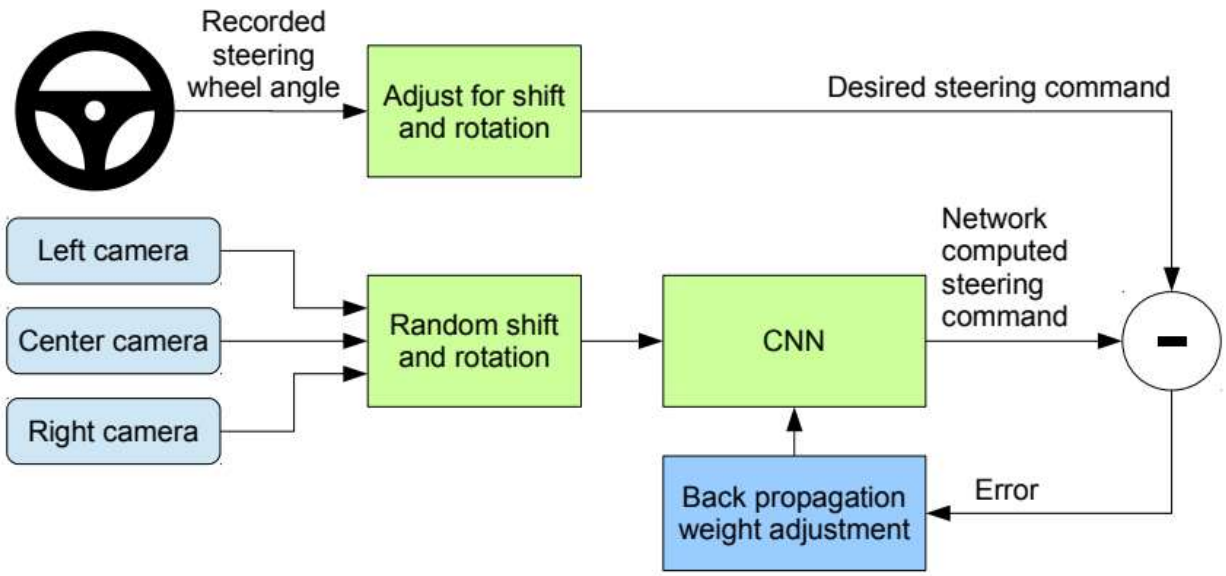
2. **J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In Computer Vision and Pattern Recognition (CVPR), 2017.**

   This work adopts the CIFAR-10 dataset, in which a very wide and deep network architecture is developed, combined with GPU support to decrease training time. On popular datasets, such as the MNIST handwritten digits, Chinese characters, and the CIFAR-10 images, near-human performance is achieved.

   The extremely low error rates beat prior state-of-the-art results significantly. However, it has to be mentioned that the network used for the CIFAR-10 dataset consists of 4 convolutional layers with 300 maps each, 3 max-pooling layers, and 3 fully connected 6 Page output layers. As a result, although a GPU was used, the training time was several days.

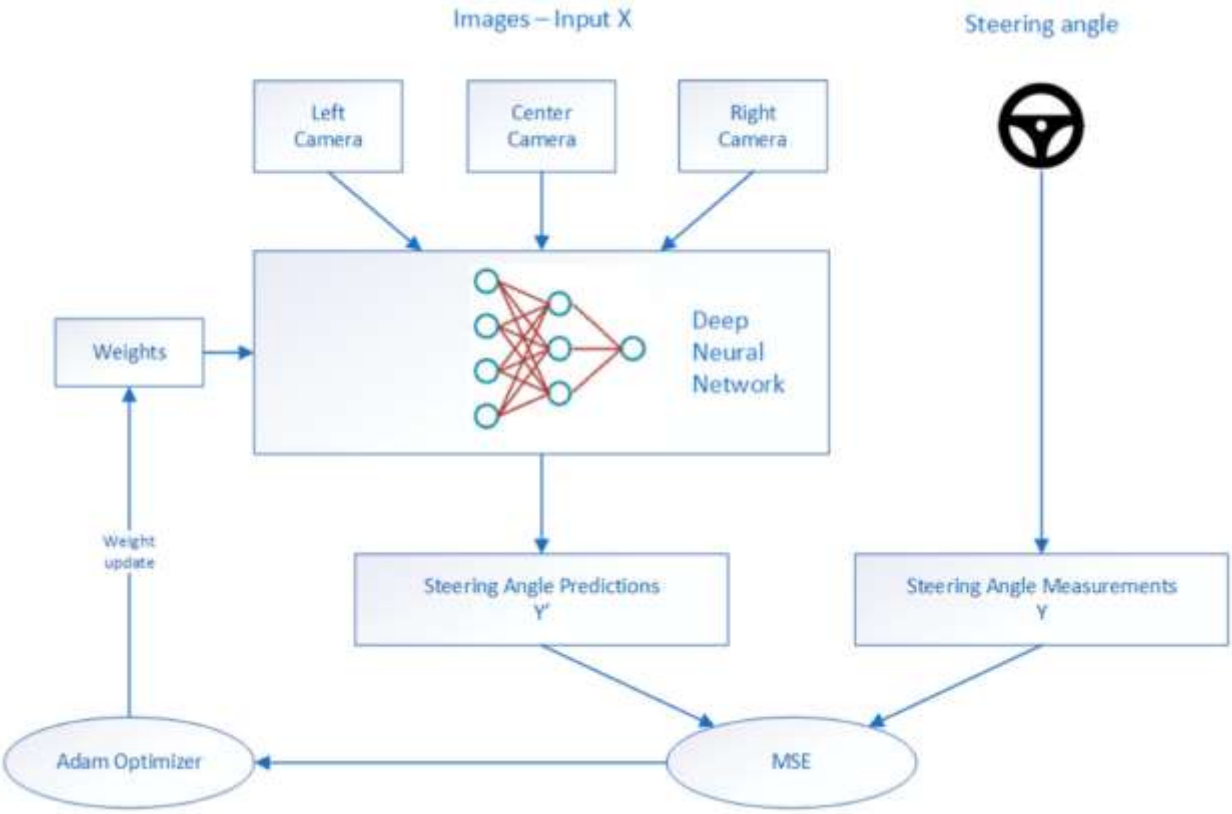
METHODOLOGY

- As we drive the car through the simulator, we are going to be taking images at each instance of the drive. These images are going to represent our training data set and the label for each specific image is going to be the steering angle of the car at that specific instance.
- We will then show all of these images to our Convolutional Neural Network (CNN) and allow it to learn how to drive autonomously by learning from our behavior as a manual driver.
- The driving simulator would save frames from three front-facing “cameras”, recording data from the car’s point of view; as well as various driving statistics like the throttle, speed, and steering angle. We are going to use camera data as model input and expect it to predict the steering angle in the [-1, 1] range.
- Now after we train our model, we are going to evaluate its performance on a completely different ‘testing track’ where the car will be made to run autonomously. If we can train the car properly, it will perform very well on our second track and will drive on its own.
- The Behavioral Cloning Technique is incredibly useful and plays a big role in real-life Self Driving Cars as well.



(developer.nvidia.com**)**

FLOWCHART

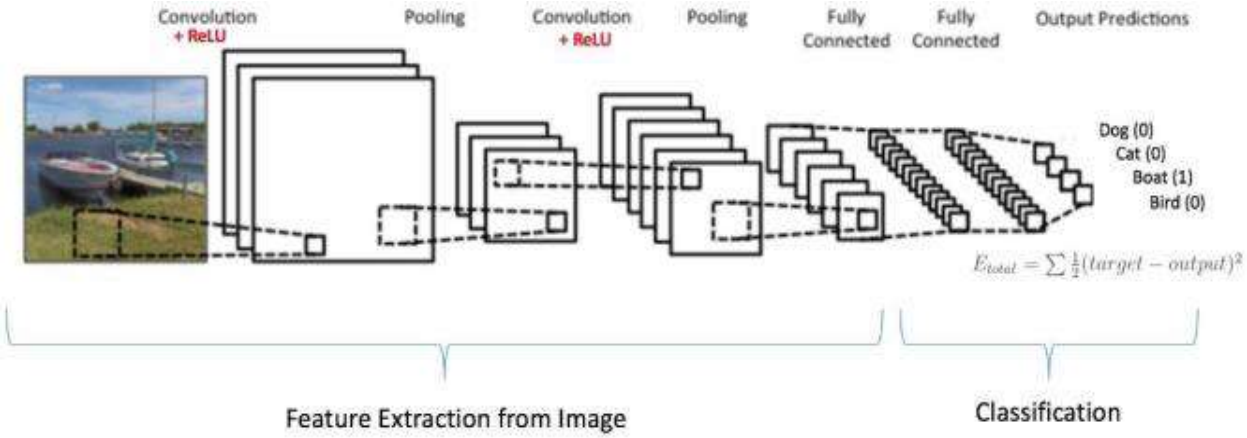


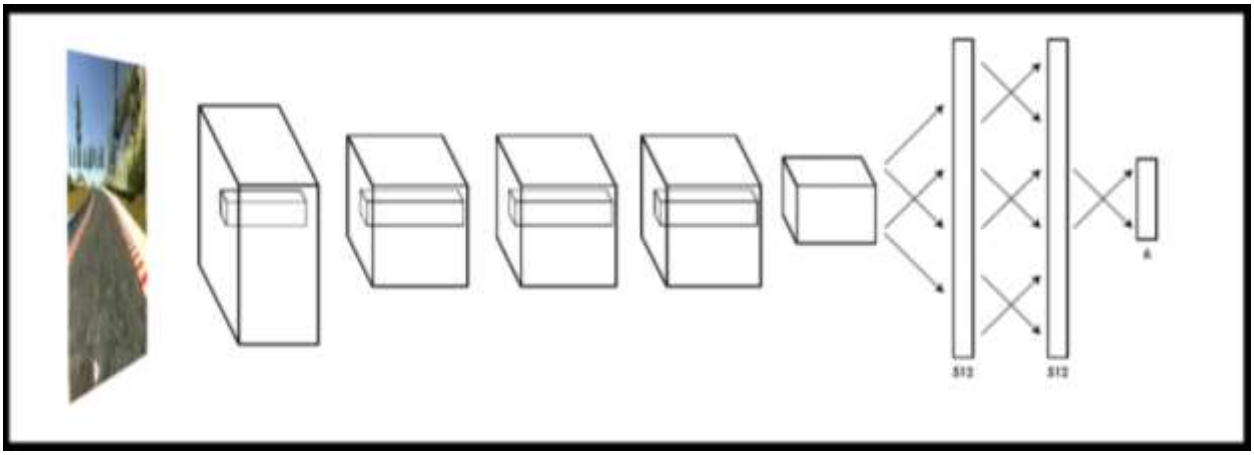
SOFTWARE USED

- Google Colab, a free online cloud-based Jupyter notebook environment is used to train and evaluate our model.
- Atom, a free and open-source text editor is used to connect our model and the simulator.
- We have used an open source software provided by Udacity for our project and python programming language here to develop our model.

- In this driving simulator there are two types of modes. One is Training mode and other is Autonomous mode.

NEURAL NETWORK DESIGN





(towardsdatascience.com)

MODULES

1. **Gathering Data**

First we have to drive the car in the Training track and we should collect the data by recording in the simulator. Normal Arrow Keys are used to drive the car. Once the recording is done we can see our data in the folder that we selected before recording.

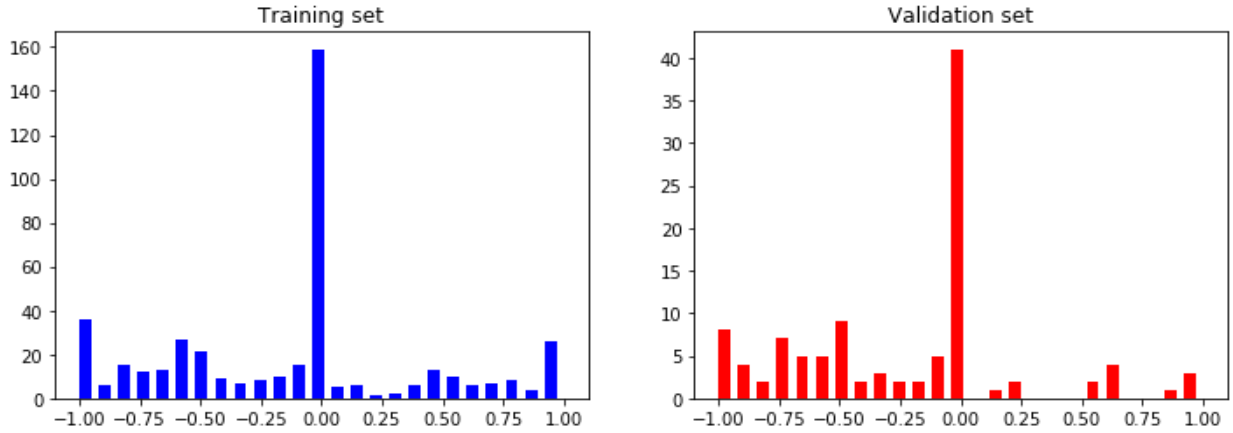
An Excel sheet of Data and Image folder is produced. The excel sheet contains lots of data. A sample of it is shown below.

right	steering	throttle	reverse	speed
right_2019_07_22_20_38_15_382.jpg	0.0	0.0	0	0.000079
right_2019_07_22_20_38_15_526.jpg	0.0	0.0	0	0.000082
right_2019_07_22_20_38_15_669.jpg	0.0	0.0	0	0.000078
right_2019_07_22_20_38_15_802.jpg	0.0	0.0	0	0.000078
right_2019_07_22_20_38_15_937.jpg	0.0	0.0	0	0.000080

2. Training and Validation split

We will then split the images into training and validation set in order to measure the performance at every epoch.

We split the data using the 80–20 rule which means using 80% of the data for training while the rest for testing the model on unseen images.



Steering Angle Distributions

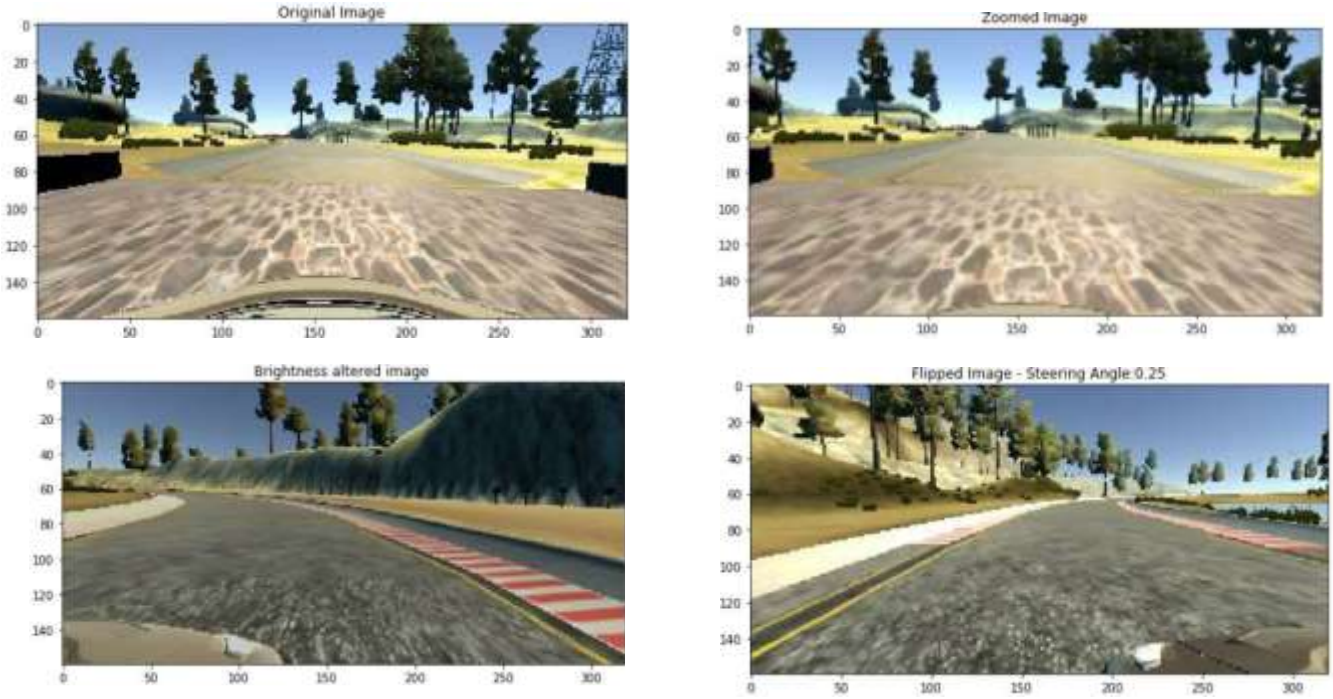
3. Model Training (Image Augmentation)

For training, we use the following augmentation technique along with Python generator to generate unlimited number of images:

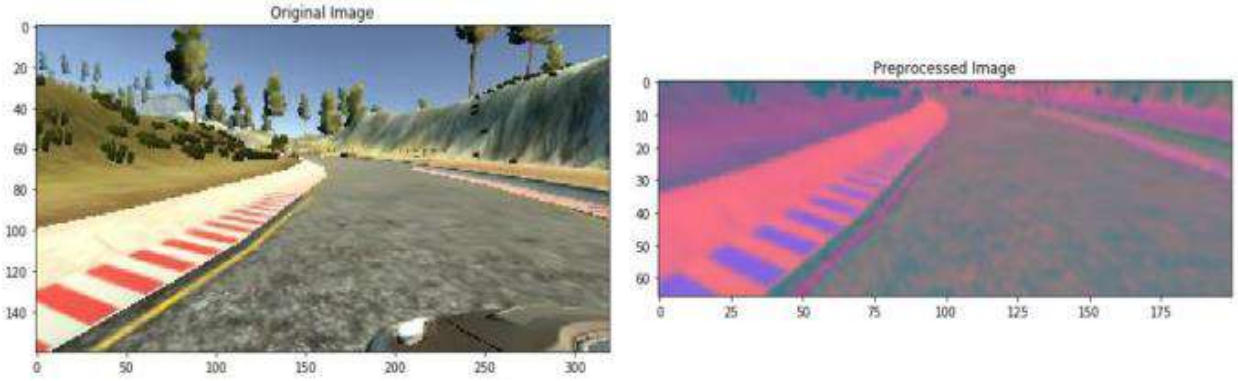
- Randomly choose right, left or center images.
- For left image, steering angle is adjusted by +0.2
- For right image, steering angle is adjusted by -0.2
- Randomly flip image left/right
- Randomly translate image horizontally with steering angle adjustment (0.002 per pixel shift)
- Randomly translate image vertically
- Randomly added shadows
- Randomly altering image brightness (lighter or darker)

Using the left/right images is useful to train the recovery driving scenario. The horizontal translation is useful for difficult curve handling (i.e. the one after the bridge).

Below we show all the Augmenting Images



Below is the preprocessed image which is actually used in training to train the car whether if it's raining or if any climatic changes are there.



4. Image Pre-processing

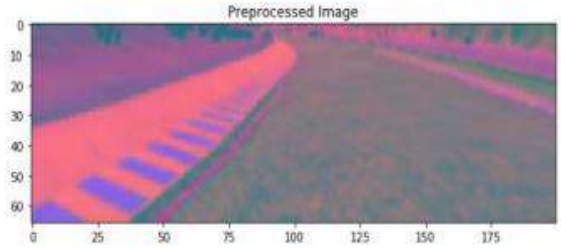
The images are cropped so that the model won't be trained with the sky and the car front parts. The images are then resized as per the NVIDIA model.

The picture is taken by the Car which has three front cameras. They are center, left, right image. Below are the images of all the three cameras at one particular instance.

Center Image

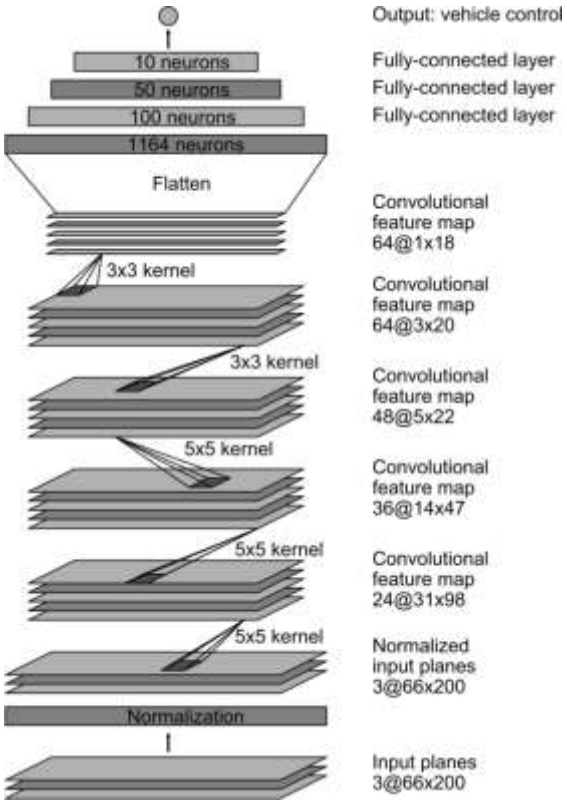
Left Image

Right Image



Pre-processed Image

NVIDIA MODEL ARCHITECTURE



(developer.nvidia.com**)**

The model looks like as follows:

- Image normalization
- Convolution: 5x5, filter: 24, strides: 2x2, activation: ELU
- Convolution: 5x5, filter: 36, strides: 2x2, activation: ELU
- Convolution: 5x5, filter: 48, strides: 2x2, activation: ELU
- Convolution: 3x3, filter: 64, strides: 1x1, activation: ELU
- Convolution: 3x3, filter: 64, strides: 1x1, activation: ELU
- Drop out (0.5)
- Fully connected: neurons: 100, activation: ELU
- Fully connected: neurons: 50, activation: ELU
- Fully connected: neurons: 10, activation: ELU
- Fully connected: neurons: 1 (output)

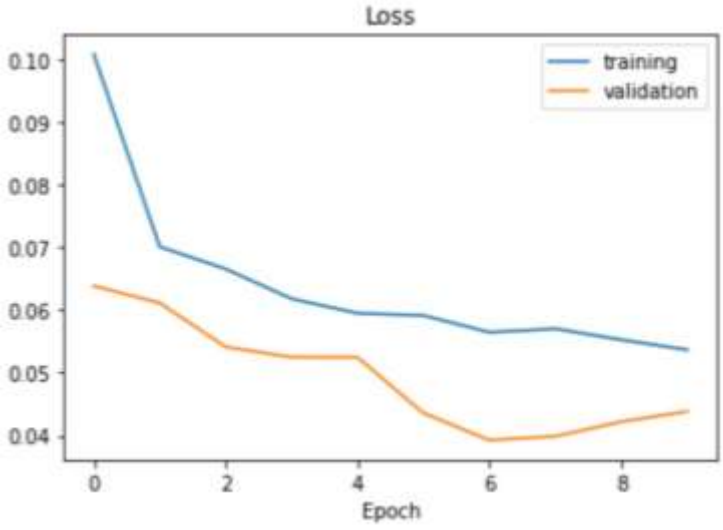
```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 31, 98, 24)        1824
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 47, 36)        21636
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 5, 22, 48)         43248
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 20, 64)         27712
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 1, 18, 64)         36928
_________________________________________________________________
dropout (Dropout)            (None, 1, 18, 64)         0
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense (Dense)                (None, 100)               115300
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_2 (Dense)              (None, 10)                510
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 11
=================================================================
Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0
_________________________________________________________________
```

- The design of the network is based on the NVIDIA model, which has been used by NVIDIA for the end-to-end self driving test.
- It is a deep convolution network which works well with supervised image classification / regression problems.

TRAINING OUR MODEL

- Activation function : Exponential Linear Unit (elu)
- Learning Rate: 1*e-3
- Method Adopted to calculate loss= Mean Square Error Method
- Epochs= 10
- Time taken to train model for each epoch is ~433seconds
- Image input size = 200*66
- 300 training steps per epoch (consisting of 100 images per epoch)
- 200 validating steps per epoch (consisting of 100 images per epoch)

```
Epoch 1/10
300/300 [==============================] - 444s 1s/step - loss: 0.1458 - val_loss: 0.0607
Epoch 2/10
300/300 [==============================] - 441s 1s/step - loss: 0.0804 - val_loss: 0.0559
Epoch 3/10
300/300 [==============================] - 439s 1s/step - loss: 0.0752 - val_loss: 0.0539
Epoch 4/10
300/300 [==============================] - 438s 1s/step - loss: 0.0710 - val_loss: 0.0511
Epoch 5/10
300/300 [==============================] - 440s 1s/step - loss: 0.0691 - val_loss: 0.0413
Epoch 6/10
300/300 [==============================] - 439s 1s/step - loss: 0.0636 - val_loss: 0.0430
Epoch 7/10
300/300 [==============================] - 438s 1s/step - loss: 0.0613 - val_loss: 0.0415
Epoch 8/10
300/300 [==============================] - 438s 1s/step - loss: 0.0602 - val_loss: 0.0367
Epoch 9/10
300/300 [==============================] - 438s 1s/step - loss: 0.0582 - val_loss: 0.0443
Epoch 10/10
300/300 [==============================] - 434s 1s/step - loss: 0.0556 - val_loss: 0.0363
```



OUTCOMES

- Now after we train our model, we are going to evaluate its performance on a completely different ‘testing track’ where the car will be made to run autonomously.
- If we are able to train the car properly, it will perform very well on our second track and will drive on its own.
- The Behavioral Cloning Technique is incredibly useful and plays a big role in real-life Self Driving Cars as well.
- If we have trained our model perfectly, by testing it in new track (autonomous mode) shown below our car will move without hitting the edge of the road. Then we come to know that we have trained our model perfectly.

DEMONSRATION OF RESULTS

Final Video Link:

https://drive.google.com/file/d/1gkvCVv23vR5aJcag9FXGh94RPSK2Fgr7/view?usp=sharing

IMPACT TO SOCIETY

1. Road Safety

With human errors being the cause for around 90% of traffic accidents, delegating the vehicle operation responsibilities to a machine can reduce the accidents and thus increase road safety.

2. Reduced emissions

Autonomous capabilities such as consistent driving speeds and keeping a measured distance between vehicles can reduce unnecessary braking and re-acceleration. They further reduce pollution by eliminating or lessening the use of fuel.

3. Eases Traffic Congestion

Because self-driving cars are rarely involved in accidents, their potential to ease congestion is high. Not only that, because self-driving cars can communicate with each other, they would also eliminate the need for traffic signals. By driving at a slower rate but with fewer stops, better-coordinated traffic would lead to less congestion.

REFERENCES

[1] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In Computer Vision and Pattern Recognition (CVPR), 2017.

[2] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J. and Zhang, X., 2016. End to end learning for self-driving cars.

[3] M. A. A. Babiker, M. A. O. Elawad and A. H. M. Ahmed, "Convolutional Neural Network for a Self-Driving Car in a Virtual Environment," 2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), Khartoum, Sudan, 2019, pp. 1-6.

[4] B. T. Nugraha, S. Su and Fahmizal, "Towards self-driving car using convolutional neural network and road lane detector," 2017 2nd International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT), Jakarta, 2017, pp. 65-69.

[5] Á. Takács, I. Rudas, D. Bösl and T. Haidegger, "Highly Automated Vehicles and Self-Driving Cars [Industry Tutorial]," in IEEE Robotics & Automation Magazine, vol. 25, no. 4, pp. 106-112, Dec. 2018.

[6] Chen, Z. and Huang, X., 2017, June. End-to-end learning for lane keeping of self-driving cars. In 2017 IEEE Intelligent Vehicles Symposium (IV) (pp. 1856-1860). IEEE.

[7] Farag, W., 2019. Cloning safe driving behavior for self-driving cars using convolutional neural networks. Recent Patents on Computer Science, 12(2), pp.120-127.