International Journal of Research in Engineering, Science and Management
Volume 5, Issue 1, January 2022
https://www.ijresm.com | ISSN (Online): 2581-5792

Self-Driving Car Using Image Processing and Deep Learning

Sushaanth Reddy Donthi[1*], Vemulapalli Sandeep[2], Mandadi Goutham Reddy[3], Mehul Bhatt[4], Honne Sharan Kumar[5], Dasamantharao Jayanth[6]

***Abstract*: Self-driving autonomous vehicles are the solution for enhancing mobility intelligence related to driving. This project presents an effective way for implementation of a self-driving car. Proposed work is based on Artificial Intelligence, Computer Vision and Neural Networks. In our project, we are using many features such as mapping, tracking and local planning. We can successfully create a car that can demonstrate proper lane changes, parking, and U-turns on its own. The different innovations we are using are obstacles and curb detection methods, road vehicle tracker, and checking different traffic situations. This will make a robust autonomous self-driven car. It will successfully demonstrate proper parking allotment, lane changes, and automatic U-turns. We can do these using the obstacle and various curb detection method, the vehicle tracker. Self-driving cars combine a variety of sensors to perceive their surroundings, such as radar, lidar, sonar, GPS, odometry and inertial measurement units. Advanced control systems interpret sensory information to identify appropriate navigation paths, as well as obstacles and relevant signage. Long distance trucking is seen as being at the forefront of adopting and implementing the technology. We use Artificial Intelligence for recognizing and proposing the path that the autonomous car should follow for proper working. Additionally, a driverless car can reduce the time taken to reach the destination because it will take the shortest path, avoiding the traffic congestion. Human errors will be avoided thereby allowing disabled people (even blind people) to own their car.**

***Keywords*: Self-driving autonomous vehicles, Computer Vision, Neural Networks, Artificial Intelligence.**

1. Introduction

This paper proposes a model to attain driverless cars. Research is still going on; we hope that this simulated model will be incorporated as software in real-life cars in the near future**.** To build a reliable system to increase the Image Enhancement without loss of quality. The project can be extended and used with different training models to increase the accuracy of detecting turns by the car. Our autonomous vehicle is able to run itself on any generalized track with the same training and validation accuracy. The system should accept input images and optimize for faster processing. To achieve, tracking and local planning. Demonstrate proper lane changes, parking, and U-turns on its own. The different innovations we are using are obstacles and curb detection methods, road vehicle tracker, and checking different traffic situations. Create a robust autonomous self-driven car. It will successfully demonstrate proper parking allotment, lane changes, and automatic U-turns. We can do these using the obstacle and various curb detection methods, the vehicle tracker.

A. Dataset

To gather the data Udacity, has a car simulator in which a track is provided and we can drive the car on the track manually. The simulator has a record button. So, when we click it the option comes to select a folder then when we start driving the car on the track the simulator stores the picture at each instant and also stores the car steering angle corresponding to each image. The car simulator has 3 cameras which store images as left, right and center. For having a proper well-spaced dataset containing all cases and angles, it is recommended to take at least take 3 laps in forward and reverse direction on the track.

B. Data Preprocessing

Deleting high-frequency dataset values which make the model biased: Since most of the time, we drive through the center of the track the dataset contains a high number of 0-degree steering angles which may make the self-driving simulator biased towards predicting 0-degree angle which will lead to the model predicting 0 angles and lead to a crash. So first we drop some 0-degree angle values.

2. Training Set and Validation Set Splitting of Dataset for Better Model Creation and Prevention of Overfitting of Training Data

If we do not make a validation set, then the model will overfit and won't work well for generalized tracks. It will only work well for the track on which the dataset is created.

A. Augmenting Variations of Images

Images from the dataset are taken and variations of all images such as zoomed image, brightness altered image and flipped image are added to the dataset just to make the model more generalized.

B. Preprocessing the Image

The images in the dataset are all RGB images so for ease of training the model the images are converted to YUV format. YUV color-spaces are more efficient coding and reduce the

*Corresponding author: donthisushanth@gmail.com

bandwidth more than RGB capture can. The images are also blurred using Gaussian blur function of openCV and resized so that unimportant parts such as background scenery are cropped out. Then each pixel is divided by 255 so that all pixels get equal priority as pixels with high values get unnecessary priority. Dividing by 255 will reduce all pixel values to 0 or 1.

C. Training the Model

The model is trained for a desired number of epochs so that its output will always strive to eliminate bias in the model and make it efficient so that it can be generalized for any track when put to practical use.

D. Simulating the Model

We simulated our model using an Open-Source simulator that was made with Unity3D editor and we used Flask backend to establish a real-time communication using socket, Io a JavaScript real time common library and were able to note the tilt, angle, coordinate and speed at every instant.

E. Main Principle Behind

The basic idea of the tracking is that the vehicle is a continuous displacement movement process. During the advancement of the vehicle, the change of the lane line is also a continuous change. This change is reflected in the slope of the lane line. The slope of the lane line in the two frames of the front and rear images are not much different from the position of the lane line. Therefore, the two frames before and after the control are compared. The slope of the lane line in the middle is limited near the previously detected lane line area. This is the basic idea of tracking. Finding lane lines within the area of interest can greatly reduce the amount of image processing.

3. Working

When the polar angle of the lane line is within the detection area, the position of the lane line can be quickly and accurately detected. However, when the image is shifted in a turn, lane change or camera position, the lane line easily exceeds the detection area, so that the results appear to be some deviations.

In traditional Hough transform, each point needs to be traversed at each angle, which is time consuming. But, the modified Hough transform is used to perform transformation on the vanishing point and the limited pixels around it. and improve the real time performance of the algorithm.

We trained a custom Convolutional Neural Network. It is a deep neural network. We declare the model to be sequential and add 5 Convolution2D, we also add 4 dropout layers and 4 dense layers to the network. A single flatten layer is added to convert the image matrix to a one-dimensional array. Built basically by following the given characteristics:

We trained a custom Convolutional Neural Network. It is a deep neural network. We declare the model to be sequential and add 5 Convolution2D, we also add 4 dropout layers and 4 dense layers to the network. A single flatten layer is added to convert the image matrix to a one-dimensional array. Built basically by following the given characteristics: We trained a custom Convolutional Neural Network. It is a deep neural network. We declare the model to be sequential and add 5 Convolution2D, we also add 4 dropout layers and 4 dense layers to the network. A single flatten layer is added to convert the image matrix to a one-dimensional array. Built basically by following the given characteristics:

The Hough Transform (HT) is a digital image processing method for the detection of shapes which has multiple uses today. A disadvantage of this method is its sequential computational complexity, particularly when a single processor is used. An optimized algorithm of HT for straight lines detection in an image is presented in this article. Optimization is realized by using a decomposition of the input image recently proposed via central processing unit (CPU), and the technique known as segment decomposition [12].

4. Why we are Relying On Hough Transformation

Most roads on the road are basically straight, and there are few sharp bends in the curve. Therefore, in the lane detection and tracking, the Hough transform is used to detect the line and determine the approximate position and shape of the lane. Then determine the deviation direction of the lane by the slope of the lane, and then find the curve part of the lane. In this way, the accuracy of the detection of the lane line can be ensured, and there is no serious error in the detected curve.

The Hough line detection method is accurate and simple, and the curve detection can be performed after adding the Vanishing point tracking algorithm.

A self-driving car is a vehicle that is able to sense its surroundings and drive without human intervention. Self-driving cars can sense surroundings using many ways like lidar, radar, GPS, camera. Biggest benefit of self-driving cars is the reduced number of accidents. If such cars are widely available and engineered properly can also save fuel which will lead to lesser pollution. In recent period there has been tremendous amount of development in self-driving vehicle space which is attracting wide range of consumers mostly due to the benefits it provides [13].

We can successfully create a car that can demonstrate proper lane changes, parking, and U-turns on its own. The different innovations that feature are obstacles and curb detection methods, road vehicle tracker, and checking different traffic situations. This will make a robust autonomous self-driven car. It will successfully demonstrate proper parking allotment, lane changes, and automatic U-turns. We can do these using the obstacle and various curb detection method, the vehicle tracker. Autonomous cars have various advantages over manual cars like fewer traffic accidents, smart decision making just to name a few [14].

The most commonly occurring failures included the failure to detect lanes and uncomfortable speed changes of the vehicle. Additionally, a majority of the drivers emphasized the importance of being alert while driving with autonomous features and aware of the limitations of the current technology. Our main contribution is to provide a picture of attitudes and experiences towards semi-autonomous driving, revealing that some drivers adopting these features may not perceive autonomous driving as risky, even in an environment with

regular automation failures [15].

For that matter, it is necessary to define what exactly is understood as an autonomous vehicle (AV) in this report. While the technology can be adapted to a great variety of different vehicle types, here road-based transport is considered specifically, covering adaptations of self-driving technology to private cars and public transport solutions. In that regard the terms "autonomous", "self-driving" and "driver-less" are used interchangeably as is often the case in the existing literature in distinction to CVs (conventional vehicles). Furthermore, it needs to be defined on which time scope the review addresses. While the 11 technological development is already making progress, the adoption of self-driving cars is just beginning. There are plenty of scenarios on how the route towards large-scale use of AVs will look like, with technological, societal, legal, and economic barriers [15].

5. Deep Learning

Deep mastering is a department of computer mastering that teaches computer systems to do what comes naturally to human's study from experience. Machine studying algorithms use computational techniques to "learn" facts without delay from information besides relying on a predetermined equation's as a model. Deep gaining knowledge of is mainly suitable for photograph recognition, which is essential for fixing troubles such as facial recognition, action detection, and many superior drivers helps applied sciences such as self-reliant driving, lane detection, pedestrian detection, and self-reliant parking. Deep studying makes use of neural networks to examine beneficial representations of points at once from data. Neural networks mix more than one nonlinear processing layers, the usage of easy factors running in parallel and stimulated by using organic anxious systems.

A. Unity 3D Gaming Engine

Unity is a cross-platform game engine developed by Unity Technologies, first announced and released in June 2005 at Apple Inc.'s Worldwide Developers Conference as a Mac OS X- exclusive game engine. As of 2018, the engine had been extended to support more than 25 platforms.

Unity3D is a powerful cross-platform 3D engine and a user-friendly development environment. Easy enough for the beginner and powerful enough for the expert; Unity should interest anybody who wants to easily create 3D games and applications for mobile, desktop, the web, and consoles.

The generation of the environment in. xodr is done using a free application called OpenRoadEd. Different parts of the roads consisting of straight roads, arcs and spirals are generated using the road settings panel. Later, the roads are connected to each other to form the whole track. In this part, the definition of the environment can be also done using the available textures or by adding the desired textures to the library of images. By saving the geometry, the generated environment is transformed automatically to. xodr format.

B. Supervised Learning

Given a space X that yields a representation of the image space containing the images from the unity framework. We consider a finite subset of training examples XTrain = {x(1), ..., x(n)} ⊆ X. Let further y (i) be the corresponding ground truth data with respect to x(i), e.g., the steering angle, brake and throttle. For supervised learning, the model Mw is trained with the training set T = {(x(i), y (i)) | i =1, ..., n} which includes the training examples and the corresponding ground truth data. Artificial Neural Networks Artificial Neural Networks (ANNs) are a group of models in Machine Learning, which are inspired by the structure and functions of biological neural networks [13]. The central neural system in the mammals' brain contains a huge number of neurons, which are strongly interconnected. With the help of this structure, information is processed and evaluated across neurons by electrochemical interactions. The ANNs are then an attempt at implementing a similar way of information processing on computers. We illustrate below the internal structure of an artificial neuron, which is the constitutive unit of ANNs. The artificial neuron receives one or more inputs xi. All inputs are then multiplied by corresponding weights wi and summed up together with a threshold term θ known as bias. After that, this input is passed through a non-linear function φ known as an activation function to produce a certain output y. We summarize the whole operation applied by an artificial neuron in the following formula:

$$y = \varphi \left(\sum xi \cdot wi + \theta \right)$$

We can represent the ANN as a model Mw, that is composed of learnable weights w, and activation functions φ.

6. Image Processing

Image processing is the method to convert an photo into digital structure and function operations on it to get an greater photograph or extract some beneficial data from it. Changes that take area in snap shots are generally carried out routinely and count on cautiously designed algorithms. Image processing is a multi-discipliner's field, with contribution from unique branches of science which all includes mathematics, physics, optical and electrical engineering.

A. Flask and Socket IO

The model is communicated with the simulator via a Flask backend and suing a real time communication channel established using socket IO.

7. Drawbacks in Existing System

A. Processing Power

First of all, since deep learning requires such a high level of computing power, a very powerful "brain" is needed to handle the big data capabilities and processing requirements. Currently, the most suitable technology is the so-called GPU (graphical processing unit), since it is designed to handle heavy image processing tasks (known from for example the computer gaming industry). Currently the companies Nvidia and Intel are on their way to position themselves as leaders supplying the "brains" for the intelligent vehicle market. However, it is still a challenge to have a low-cost GPU that operates within the energy consumption and other boundaries, such as heat

management, that is required for a market-ready vehicle. Moreover, companies still struggle with bandwidth and synchronization issues.

B. Available training data

As noted before, an end-to-end learning system especially, requires to be fed a huge amount of training data, in order to predict as many driving scenarios as possible and to fulfil a minimum safety requirement. Some claim that at least a billion kilometers of training data from realistic road scenarios are needed in order to make conclusions about safety of the vehicle. Not only that, the data needs to be diverse enough for it to be useful (driving one kilometer a billion times back and forth won't do the job!).

C. Safety

One of the main challenges with safety of deep neural networks is the fact that they are unstable under so-called adversarial perturbations. For example, minimal modifications in camera images, such are resizing, cropping and the change of lighting conditions might cause the system to misclassify the image. Additionally, in general, safety assurance and verification methods for machine learning are poorly studied. The prevailing automotive safety standard of ISO26262, does not have a way to define safety for self-learning algorithms such as deep learning. Hence, there is still no way to standardize the safety aspect yet, due to the fast pace of current technology. A prominent example of a safety failure is the 2016 Tesla auto-pilot accident, where the sensors of the vehicle were blended by the sun and the system failed to recognize the truck coming from the right, leading to the crash [9]. This shows that a lot still needs to be investigated before we can conclude that the current configuration of a (partially) self-driving car is safe.

8. Proposed System

A. Neural Network Regression Algorithms

This kind of algorithm is good at predicting events. The Regression Analysis evaluates the relation between 2 or more variables and collate the effects of variables on distinct scales and are driven mostly by 3 metrics:

- The shape of the regression line.
- The type of dependent variables.
- The number of independent variables.

The images (camera or radar) play a significant role in ADAS in actuation and localization, while for any algorithm, the biggest challenge is to develop an image-based model for feature selection and prediction. The repeatability of the environment is leveraged by regression algorithms to create a statistical model of relation between the given object's position in an image and that image. The statistical model, by allowing the image sampling, provides fast online detection and can be learned offline. It can be extended furthermore to other objects without the requirement of extensive human modeling. An object's position is returned by an algorithm as the online stage's output and a trust on the object's presence. The regression algorithms can also be utilized for short prediction, long learning. This kind of regression algorithms that can be utilized for self-driving cars are decision forest regression, neural network regression and Bayesian regression, among others. The neural networks are utilized for regression, classification or unsupervised learning. They group the data that is not labeled, classify that data or forecast continuous values after supervised training. The neural networks normally use a form of logistic regression in the final layer of the net to change continuous data into variables like 1 or 0.

We will be using Google Collaboratory to write our code in Python3 for this project. Google Collaboratory is a free cloud platform where we can write codes and it also supports GPU which makes it a lot faster than any other expensive PC or laptop without buying it thus making it suitable for Artificial Intelligence and Machine learning

B. Advantages of Proposed System

- Accuracy will be high.
- Picture quality will not be compressed.
- Will be able to rain the car and use the model in any generalized environment.

9. Algorithm's and Approach

The machine learning algorithms are loosely divided into 4 classes: decision matrix algorithms, cluster algorithms, pattern recognition algorithms and regression algorithms. One category of the machine learning algorithms can be utilized to accomplish 2 or more subtasks. For instance, the regression algorithms can be utilized for object localization as well as object detection or prediction of the movement.

A. Decision Matrix Algorithms

The decision matrix algorithm systematically analyzes, identifies and rates the performance of relationships between the sets of information and values. These algorithms are majorly utilized for decision making. Whether a car needs to brake or take a left turn is based on the level of confidence these algorithms have on recognition, classification and prediction of the next movement of objects. The decision matrix algorithms are models composed of various decision models trained independently and, in some way, these predictions are combined to make the overall prediction, while decreasing the possibility of errors in decision making. AdaBoosting is the most commonly used algorithm.

B. AdaBoosting

Adaptive Boosting or AdaBoost is a combination of multiple learning algorithms that can be utilized for regression or classification. It overcomes overfitting when compared with any other machine learning algorithms and is often sensitive to outliers and noisy data. In order to create one composite powerful learner, AdaBoost uses multiple iterations. So, it is termed as adaptive. By adding the weak learners iteratively, AdaBoost creates a strong learner. A new weak learner is appended to the entity and a weighing vector is adjusted in order to pay attention on examples that were classified incorrectly in the prior rounds. A classifier that has much higher accuracy than the classifiers of weak learners is the result.

AdaBoost helps in boosting the weak threshold classifier to strong classifier. The above image depicts the implementation of AdaBoost in one single file with understandable code. The function contains a weak classifier and the boosting component. The weak classifier attempts to locate the ideal threshold in one of data dimensions to segregate the data into 2 classes. The classifier is called by the boosting part iteratively and after each classification step, it changes the weights of misclassified examples. Because of this, a cascade of weak classifiers is created and it behaves like a strong classifier.

C. Clustering Algorithms

Sometimes, the images acquired by the system are not clear and it becomes difficult to locate and detect objects. Sometimes, there is a possibility of classification algorithms missing the object and, in that case, they fail to categorize and report it to the system. The possible reason could be discontinuous data, very few data points or low-resolution images. The clustering algorithm is specialized in discovering the structure from data points. It describes the class of methods and class of problem like regression. The clustering methods are organized typically by modeling the approaches like hierarchical and centroid-based. All methods are concerned with utilizing the inherent structures in data to organize the data perfectly into groups of maximal commonalities.

K- means, Multi-class Neural Network is the most commonly used algorithm.

D. K-Means

K-means Algorithm – The cluster centroids are depicted as crosses and training examples are depicted as dots. (a) Original dataset. (b) Random initial cluster centroids. (c-f) The demonstration of running 2 iterations of k-means. Each training example is assigned in each iteration to the cluster centroid that is closest and then, each cluster centroid is moved to mean of points assigned to it.

E. Pattern Recognition Algorithms (Classification)

The images obtained through sensors in Advanced Driver Assistance Systems (ADAS) consists of all kinds of environmental data; filtering of the images is needed to determine the instances of an object category by ruling out the data points that are irrelevant. Before classifying the objects, the recognition of patterns is an important step in a dataset. This kind of algorithms are defined as data reduction algorithms. The data reduction algorithms are helpful in reducing the dataset edges and polylines (fitting line segments) of an object as well as circular arcs to edges. Till a corner, the line segments are aligned with the edges and a new line segment will begin after this. The circular arcs align with the line segments' sequences that is similar to an arc. In various ways, the features of the image (circular arcs and line segments) are combined to form the features that are utilized for determining an object. With the PCA (Principle Component Analysis) and HOG (Histograms of Oriented Gradients), the SVM (Support Vector Machines) are the commonly used recognition algorithms in ADAS. The K nearest neighbor (KNN) and Bayes decision rule are also used.

F. Support Vector Machines (SVM)

SVM are dependent on the decision planes concept that define the decision boundaries. The decision plane separates the object set consisting of distinct class memberships. A schematic example is illustrated below. In this, the objects belong to either RED or GREEN class. A boundary line of separation separates the RED and GREEN objects. Any new object that falls to the left is labeled as RED and it is labeled as GREEN if it falls to the left.

We will be training our model using the algorithms mentioned below:

1) Regression Algorithms

This kind of algorithm is good at predicting events. The Regression Analysis evaluates the relation between 2 or more variables and collate the effects of variables on distinct scales and are driven mostly by 3 metrics: The shape of regression line. The type of dependent variables. The number of independent variables. The images (camera or radar) play a significant role in ADAS in actuation and localization, while for any algorithm, the biggest challenge is to develop an image-based model for feature selection and prediction. The repeatability of the environment is leveraged by regression algorithms to create a statistical model of relation between the given object's position in an image and that image. The statistical model, by allowing the image sampling, provides fast online detection and can be learned offline. It can be extended furthermore to other objects without the requirement of extensive human modeling. An object's position is returned by an algorithm as the online stage's output and a trust on the object's presence. The regression algorithms can also be utilized for short prediction, long learning. This kind of regression algorithms that can be utilized for self-driving cars are decision forest regression, neural network regression and Bayesian regression, among others.

2) Neural Network Regression

The neural networks are utilized for regression, classification or unsupervised learning. They group the data that is not labeled, classify that data or forecast continuous values after supervised training. The neural networks normally use a form of logistic regression in the final layer of the net to change continuous data into variables like 1 or 0.

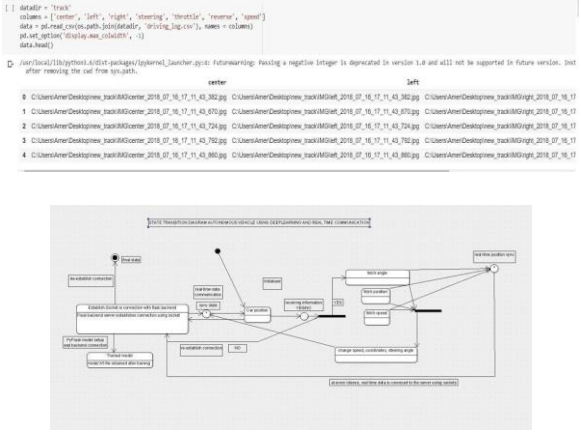


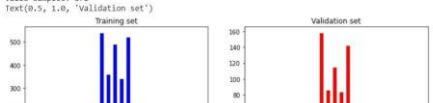
Fig. 1. Sequence diagram

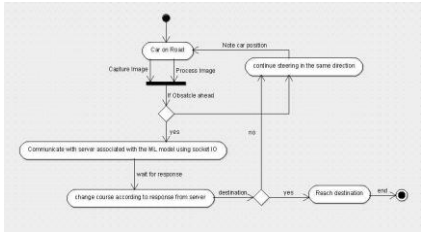
```
[ ] import os
    import numpy as np
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    import keras
    from keras.models import Sequential
    from keras.optimizers import Adam
    from keras.layers import Convolution2D, MaxPooling2D, Dropout, Flatten, Dense
    from sklearn.utils import shuffle
    from sklearn.model_selection import train_test_split
    from imgaug import augmenters as iaa
    import cv2
    import pandas as pd
    import ntpath
    import random

    Using TensorFlow backend.
```

```
[ ] image_paths, steerings = load_img_steering(datadir + '/IMG', data)
    X_train, X_valid, y_train, y_valid = train_test_split(image_paths, steerings, test_size=0.2, random_state=6)
    print('Training Samples: {}\nValid Samples: {}'.format(len(X_train), len(X_valid)))
    fig, axes = plt.subplots(1, 2, figsize=(12, 4))
    axes[0].hist(y_train, bins=num_bins, width=0.05, color='blue')
    axes[0].set_title('Training set')
    axes[1].hist(y_valid, bins=num_bins, width=0.05, color='red')
    axes[1].set_title('Validation set')

    Training Samples: 3511
    Valid Samples: 878
    Text(0.5, 1.0, 'Validation set')
```





Sample Code:

```
[ ] def img_preprocess(img):
        img = img[60:135,:,:]
        img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
        img = cv2.GaussianBlur(img,  (3, 3), 0)
        img = cv2.resize(img, (200, 66))
        img = img/255
        return img
```

```
[ ] image = image_paths[100]
    original_image = mpimg.imread(image)
    preprocessed_image = img_preprocess(original_image)

    fig, axs = plt.subplots(1, 2, figsize=(15, 10))
    fig.tight_layout()
    axs[0].imshow(original_image)
    axs[0].set_title('Original Image')
    axs[1].imshow(preprocessed_image)
    axs[1].set_title('Preprocessed Image')

    Text(0.5, 1.0, 'Preprocessed Image')
```

```
[ ] def batch_generator(image_paths, steering_ang, batch_size, istraining):

      while True:
        batch_img = []
        batch_steering = []

        for i in range(batch_size):
          random_index = random.randint(0, len(image_paths) - 1)

          if istraining:
            im, steering = random_augment(image_paths[random_index], steering_ang[random_index])

          else:
            im = mpimg.imread(image_paths[random_index])
            steering = steering_ang[random_index]

          im = img_preprocess(im)
          batch_img.append(im)
          batch_steering.append(steering)
        yield (np.asarray(batch_img), np.asarray(batch_steering))
    x_train_gen, y_train_gen = next(batch_generator(X_train, y_train, 1, 1))
    x_valid_gen, y_valid_gen = next(batch_generator(X_valid, y_valid, 1, 0))

    fig, axs = plt.subplots(1, 2, figsize=(15, 10))
    fig.tight_layout()

    axs[0].imshow(x_train_gen[0])
    axs[0].set_title('Training Image')

    axs[1].imshow(x_valid_gen[0])
    axs[1].set_title('Validation Image')
```

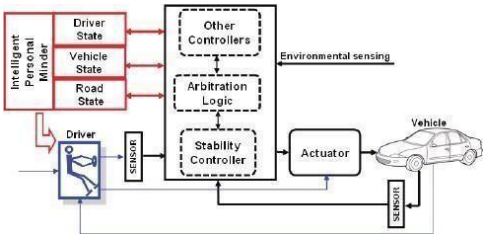
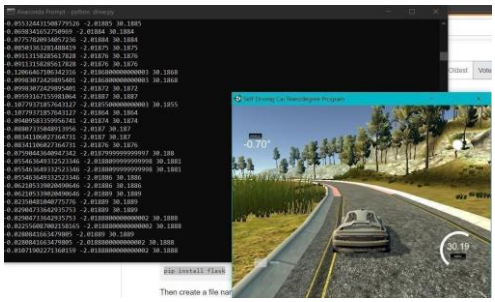


Fig. 2. Batch generator

10. Results

The plot of Loss vs number of epochs shows that there is a sufficient reduction of training and validation loss after 10 epochs. The training loss was obtained as 0.0343 and the validation loss as 0.0275. This proves that our model can also be tested on any simulator track other than the one we used for training.



Drive Coordinates are plotted in real time (-) indicates left turn and (+) right turn:

-0.0488455705344677 -2.0188300000000003 30.1883
-0.0488455705344677 -2.01879 30.1879
-0.06489957123994827 -2.01879 30.1879
-0.05687129124999046 -2.0187999999999997 30.188
-0.05687129124999046 -2.0187999999999997 30.188
-0.05891359969973564 -2.0188099999999998 30.1881
-0.059988927096128464 -2.01875 30.1875
-0.07196944206953049 -2.01878 30.1878
-0.06476251780986786 -2.0188099999999998 30.1881
-0.05087998881936073 -2.0187999999999997 30.188

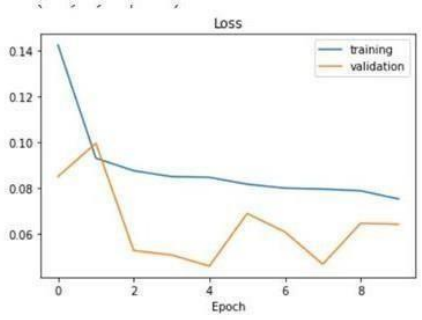
-0.05087998881936073 -2.01878 30.1878
127.0.0.1 - - [06/Jun/2020 07:23:57] "GET /socket.io/?EIO=4&transport=websocket HTTP/1.1" 200 0 100.167020

```
[ ] print(model.summary())

Model: "sequential_1"
```

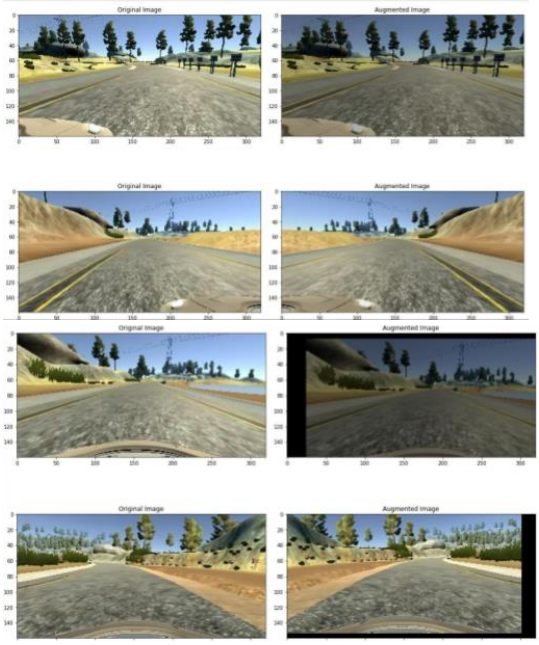
Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 31, 98, 24)	1824
conv2d_2 (Conv2D)	(None, 14, 47, 36)	21636
conv2d_3 (Conv2D)	(None, 5, 22, 48)	43248
conv2d_4 (Conv2D)	(None, 3, 20, 64)	27712
conv2d_5 (Conv2D)	(None, 1, 18, 64)	36928
dropout_1 (Dropout)	(None, 1, 18, 64)	0
flatten_1 (Flatten)	(None, 1152)	0
dense_1 (Dense)	(None, 100)	115300
dropout_2 (Dropout)	(None, 100)	0
dense_2 (Dense)	(None, 50)	5050
dropout_3 (Dropout)	(None, 50)	0
dense_3 (Dense)	(None, 10)	510
dropout_4 (Dropout)	(None, 10)	0
dense_4 (Dense)	(None, 1)	11

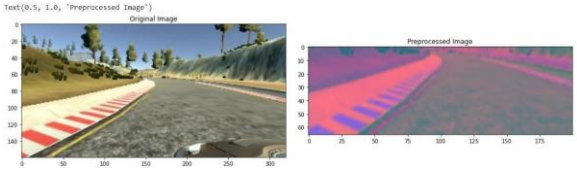
Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0




```
[ ] model.save('model.h5')
```

Loss comparison:





The images in the dataset are all RGB images so for ease of training the model the images are converted to YUV format. YUV color-spaces are a more efficient coding and reduce the bandwidth more than RGB capture can. The images are also blurred using Gaussian blur function of openCV and resized so that unimportant parts such as background scenery are cropped out. Then each pixel is divided by 255 so that all pixels get equal priority as pixels with high values get unnecessary priority. Dividing by 255 will reduce all pixel values to 0 or 1.

```
[ ] history = model.fit_generator(batch_generator(X_train, y_train, 100, 1),
                                  steps_per_epoch=300,
                                  epochs=10,
                                  validation_data=batch_generator(X_valid, y_valid, 100, 0),
                                  validation_steps=200,
                                  verbose=1,
                                  shuffle = 1)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.legend(['training', 'validation'])
    plt.title('Loss')
    plt.xlabel('Epoch')

Epoch 1/10
300/300 [==============================] - 418s 1s/step - loss: 0.1422 - val_loss: 0.0851
Epoch 2/10
300/300 [==============================] - 414s 1s/step - loss: 0.0931 - val_loss: 0.0996
Epoch 3/10
300/300 [==============================] - 414s 1s/step - loss: 0.0876 - val_loss: 0.0530
Epoch 4/10
300/300 [==============================] - 415s 1s/step - loss: 0.0850 - val_loss: 0.0510
Epoch 5/10
300/300 [==============================] - 411s 1s/step - loss: 0.0847 - val_loss: 0.0462
Epoch 6/10
300/300 [==============================] - 414s 1s/step - loss: 0.0817 - val_loss: 0.0690
Epoch 7/10
300/300 [==============================] - 420s 1s/step - loss: 0.0800 - val_loss: 0.0610
Epoch 8/10
300/300 [==============================] - 412s 1s/step - loss: 0.0797 - val_loss: 0.0470
Epoch 9/10
300/300 [==============================] - 418s 1s/step - loss: 0.0789 - val_loss: 0.0648
Epoch 10/10
300/300 [==============================] - 421s 1s/step - loss: 0.0754 - val_loss: 0.0643
Text(0.5, 0, 'Epoch')
```

Connecting the model and establishing real time communication using socketIO and Flask web server:

1. We established a flask web server using SocketIO and PyFlask.
2. The simulator was connected to the server using Sockets.
3. Real time data was communicated between the simulation environment and the web server.
4. The machine learning model imported as model.h5 was therefore used to analyses the received packets and accordingly respond to the changes in the environment, so that the car can drive collision free.

11. Conclusion

This paper proposed a model to attain driverless cars. Research is still going on; we hope that this simulated model will be incorporated as a software in real life cars in the near future. Research is still going on and millions of data scientists and artificial intelligence scientists are working in order to convert this software model into real life application so that cars can become driverless and there would be lesser rule breaks and hopefully lesser road mishaps. Further, the project can be extended and used with different training models to increase the accuracy of detecting turns by the car. We obtained a stable

model which when simulated travelled at a max speed of 30km/h on the track, also the angle of tilt was varying between positive and negative for right and left turns respectively and also made sure that it avoided all possible collisions and we were also able to generate positional and angular data at every point in time based on the polar and Cartesian coordinates of the car in the simulated environment. In this way we were able to obtain data in a CSV format which can later be used for self-driving research and conducting various analytics experiments in future.

References

[1] Morris B, Doshi A, Trivedi M., Lane change intent prediction for driver assistance: On-road design and evaluation, Intelligent Vehicles Symposium. IEEE, 2011:895-901.

[2] Paula M B D, Jung C R., Real-Time Detection and Classification of Road Lane Markings, XXVI Conference on Graphics, Patterns and Images. IEEE Computer Society, 2013:83-90.

[3] Kaur G, Kumar D, Kaur G, et al. Lane Detection Techniques: A Review. International Journal of Computer Applications, 2015, 112(10):4-8.

[4] Dorum O H, Lynch J D, Gnedin M. Creating geometry for advanced driver assistance systems: US, US8762046[P]. 2014.

[5] Bottazzi V S, Borges P V K, Stantic B, et al., Adaptive Regions of Interest Based on HSV Histograms for Lane Marks Detection, Robot Intelligence Technology and Applications. Springer International Publishing, 2014:677-687.

[6] Christos Katrakazas, Mohammed Quddus, Wen-Hua Chen, Lipika Deka, Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions, Transportation Research Part C: Emerging Technologies, vol. 60, 2015, pp. 416-442.

[7] Weiwei Chen, Weixing Wang, Kevin Wang, Zhaoying Li, Huan Li, Sheng Liu, Lane departure warning systems and lane line detection methods based on image processing and semantic segmentation: A review, Journal of Traffic and Transportation Engineering (English Edition), Volume 7, Issue 6, 2020, pp. 748-774.

[8] Narote, Sandipan & Bhujbal, Pradnya & Narote, Abbhilasha & Dhane, Dhiraj. (2017). A Review of Recent Advances in Lane Detection and Departure Warning System. Pattern Recognition. 73.

[9] Duda RO, Hart PE (1975) Use of the Hough transformation to detect lines and curves in pictures. Commun ACM 15(1):11–15.

[10] Muhamad Lazim Talib, Xio Rui, Kamarul Hawari Ghazali, Norulzahrah Mohd. Zainudin, and Suzaimah Ramli. Comparison of Edge Detection Technique for Lane Analysis by Improved Hough Transform.

[11] Satzoda RK, Suchitra S, Srikanthan T (2008) Parallelizing the Hough transform computation. IEEE Signal Process Lett 15:297–300.

[12] Yam-Uicab, R. & López-Martínez, José & Trejo-Sánchez, Joel & Hidalgo-Silva, Hugo & Gonzalez, Sergio. (2017). A fast Hough Transform algorithm for straight lines detection in an image using GPU parallel computing with CUDA-C. The Journal of Supercomputing. 73. 1-20.

[13] Heejun Choi, Travis Teague, and Tanner Luce. 2020. Positive cycle of integrating teaching and research: machine learning self-driving car. J. Comput. Sci. Coll. 35, 7 (April 2020), 74–87.

[14] Rao, Qing & Frtunikj, Jelena. (2018). Deep learning for self-driving cars: chances and challenges. 35-38. 10.1145/3194085.3194087.

[15] Murat Dikmen and Catherine M. Burns. 2016. Autonomous Driving in the Real World: Experiences with Tesla Autopilot and Summon. In Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications. Association for Computing Machinery, New York, NY, USA, 225–228.