Autonomous Braking and End to End Learning using Single Shot Detection Model and Convolutional Neural Network

Marwan Elkholy, Kirollos Nagy, Mario Magdy and Hesham H. Ibrahim[a]
Mechatronics Department, German University in Cairo, New Cairo, Egypt

Keywords: DQN Deep Q-Network, CNN Convolutional Neural Network, Relu Rectified Linear Unit, MSE Mean Square Error.

Abstract: Safety issues concerning autonomous vehicles are becoming increasingly striking. Therefore, taking security issues of autonomous driving into account such as detection and identification of the vehicle in the surrounding is necessary to apply warning messages and braking based on the state of the vehicle. This paper develops an end to end deep learning, using different recognition algorithms, to promote the safety of autonomous vehicles in terms of controlling the steering and speed of a self-driving car. Two convolutional neural network architectures are presented with different number of filters in their layers. The networks were trained to take images as input data and scan the raw pixels and convert them directly into steering angle command and speed value. Also, an object recognition algorithm is provided which detects and determines the objects and their distances from the controlled car to have a collision warning system by using a pre-trained single shot detector model. All predicted speed values and steering angles, alongside the object detection model, are then translated into throttle and braking values while evaluating the models using a simulator and real road videos.

1 INTRODUCTION

Autonomous cars are vehicles that are able to drive using digital technologies without having human interventions. Also, less space on road is employed by autonomous vehicles, so they reduce traffic jams and decrease accidents (Szikora, 2017). Traditionally, the implementation of self-driving cars requires going through several stages such as low-level perception, scene parsing, path planning, and vehicle control, but end-to-end learning contributed in reducing the number of stages into one stage which is a deep neural network. End to End learning gives the neural network the chance to transform the raw pixels of images into steering commands automatically using convolution neural networks (CNN), as the important processing steps such as detecting useful road features are learned by the model by only having the steering angle and the model develops all the processing steps together (Bojarski, 2016). Although the control of steering angle has shown great performance as an end to end application in controlling the vehicle, only the steering angle is not enough for the vehicle control. In the current work, it is proposed to control the steering angle, speed, and braking values simultaneously by using the produced models to predict these values and also apply object detection and classification on the road to help in controlling the vehicle while driving.

Several researches were done in order to improve the system of self-driving cars using end to end learning. (Pomerleau, 1988) was the first one to introduce autonomous vehicles using the neural network by inventing Autonomous Land Vehicle in a Neural Network (ALVINN) using fully connected layers and it was the first trial of end to end learning. (Bojarski, 2016) proposed one of the first successful end-to-end learning for self-driving cars network that train convolution neural network to transform pixels from the front-facing camera to steering angle without passing through other processes by detecting useful road features with only steering angle as training signal with great accuracy. (Farag, 2019) described an end-to-end CNN architecture model that is used for detecting useful road features to control the steering wheel angle instead of going through the

[a] https://orcid.org/0000-0002-1388-5739

Elkholy, M., Nagy, K., Magdy, M. and Ibrahim, H.
Autonomous Braking and End to End Learning using Single Shot Detection Model and Convolutional Neural Network.
DOI: 10.5220/0010398003090316
In *Proceedings of the 7th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2021)*, pages 309-316
ISBN: 978-989-758-513-5

process of lane marking detection and path planning. The architecture consists of seventeen layers named BCNet containing convolution layers and fully connected layers that produce the output steering angle. (Zhao, 2019) suggested a new deep model for achieving autopilot technology by predicting the coming model path from the vehicle's state by using a deep learning algorithm which is an end to end learning, as it designed a CNN model that has a simpler structure in order to decrease the model number of parameters which consists of seven layers. (Heylen, 2018) introduced a neural network to anticipate steering angle command depending on the images that were fed into the network. The architecture used for the model was based on the AlexNet, however by applying some variation for the original architecture to improve performance. (Fujiyoshi, 2019) proposed a model that will be able to control the steering of the vehicle and. the throttle in different driving scenarios by getting dashboard images and the vehicle's speed which is given to the network which consists of five layers of convolution layers that went through the pooling process and then flattened and connected to three fully connected layers. (Eneh, 2014) discussed acceleration reduction of the car once an obstacle is detected 250m ahead by an autonomous braking control system using an artificial neural network. Deep reinforcement learning was used to develop a new autonomous braking system that determines automatically whether to apply the brake at each step when a collision is about to happen using the information acquired by the sensors, the brake control strategy is taught using the reinforcement learning approach known as deep Q-network (DQN) (Chae, 2017). (Bamigboye, 2016) used artificial neural networks which consist of a pool of simple processing units that communicate with each other by sending signals over a large number of weighted connections to reduce the vehicle acceleration once an obstacle is detected 100m ahead. Digital image processing and computer vision methods were used to detect rear tail lights of front cars, and to detect and classify various objects (Bamigboye, 2016). In (Liu, 2015), the method differs from the literature (Bamigboye, 2016) in the detection task as literature (Liu, 2015) combines between the vehicle detection and RGB color difference to define the state of moving cars brake lights by recognizing the specified vehicle from the area of interest. Then, consecutive frames are compared to detect the brake light state by using the color difference threshold of RGB color space to develop a rear-end collision warning about the front vehicle. (Fujiyoshi, 2019) illustrated how deep learning is used in the image processing field and how it is related to autonomous driving because it is not easy to detect the object from the input image directly as it needs to identify the number of tasks first such as image identification, object detection, image classification, Scene understanding, and specific object detection.

So far, the previous papers have introduced the idea of controlling the speed and the steering of the car using end to end learning and different types of deep neural networks. Also, digital image processing method was used to detect and recognize traffic lights of the front vehicles.

To the author's knowledge, no previous work has combined the control of speed and steering angle using end to end learning with object detection to achieve braking.

The proposed work is implementing a model that can be used for controlling the main factors of driving a vehicle which are steering commands, speed, and brakes. NVidia and DNet-3 networks will be used for achieving two models for controlling the steering wheel and also two-speed models to predict speed values, so they can be compared. Moreover, the speed values and steering angles will be used for calculating the throttle and braking values for the vehicle. The pre-trained model SSD will be employed in detecting the objects that surround the vehicle in its environment for increasing the safety.

2 AUTONOMOUS VEHICLE CONTROL

2.1 Steering and Speed Control using End to End Learning

This proposed work is using two different deep neural networks which are NVidia and DNet-3 for end to end learning for steering prediction and also for speed values prediction which contains convolution neural layers that differ in the number of filters for each network, therefore two different models will be produced for each parameter to be compared with each other to see which is better for controlling the steering wheel and the speed of the vehicle. Moreover, the predicted speed values and steering angles will be used in calculating throttle and braking values. The pre-trained model which is SSD MobileNet V1 coco is implemented for detecting and classifying the objects that surround the car to add more safety for the self-driving car on the road and also it affects the value of braking and speed based on the closeness of the object to the controlled self-driving car. Figure 2.1 is showing a flowchart that describes the whole process for the proposed work.

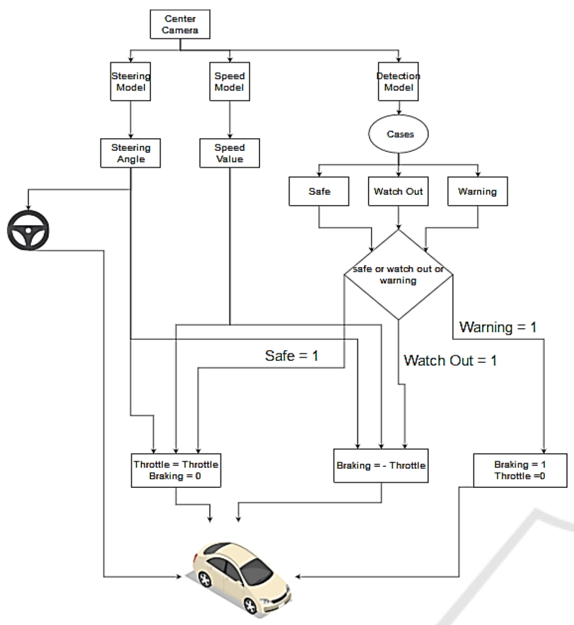


Figure 2.1: Processes implemented to control autonomous car flowchart.

The network used to produce the first model for steering and the speed model is based on the NVidia model with having some adjustments which are adding batch normalization after each layer to avoid overfitting (Bojarski, 2016).The network architecture of the NVidia model starts with the normalization layer and then 5 convolution layers for extracting features from the images starting with 24 feature maps and ending with 64 feature maps in the last layer which have 2x2 strides and 5x5 kernels in the first three convolution layers and non-strided convolution layer in the last two layers. Moreover, the output of the convolution layers is flattened and connected into three fully connected layers to produce the predicted steering angle and speed value.

The second network used for producing the second steering model is based on the drive network-3 (DNet-3) (Pal, 2019). The DNet-3 network architecture also starts by normalizing the raw data and has five convolution layers and four fully connected layers. The convolution layers start with having 18 feature maps and end with 48 feature maps. The 2x2 strides and 5x5 kernels were used in the first three layers and the non-strided convolution layer with 3x3 kernels was used in the last two layers. Besides, the output is flattened and connected to the fully connected layers for producing the steering angle.

In the current work, steering angle commands and speed values are calculated by sending the images recorded from center, right and left cameras into the CNN model. The calculated steering angle and speed value are compared with the desired steering angle and speed for this image and produce an error that is the difference between the two steering angles and the two-speed values. Mean squared error (MSE) is the loss function used to reduce the error between the predicted and desired values to make them closer to each other as MSE is always used for regression networks (Kocic, 2019). MSE loss function is calculating the average of the sum of the squared differences between predicted and desired steering angles and speed values. The equation of MSE is

$$(1/N * \sum (y_j - y \text{ hat}_j)^2) \text{ (Kocic, 2019).}$$

Adam optimizer is used for optimizing the loss for all the models since it is always used for training deep learning applications and it outstands other traditional stochastic gradient descent methods. Each batch includes 20,032 images and the batch size used is 64 through the training process and the optimum total number of epochs is 50.

All the models implemented are done in python using Keras which is the Tensorflow backend. The training for the CNN models was done on the desktop laptop that is running on Intel i5 and CPU was used for running the training code. Dropout was used after the last convolution layer with a value of 0.5 to avoid overfitting. Furthermore, batch normalization was used for all the network architectures after each layer to reduce overfitting. Relu was selected as the activation function for the entire convolution and fully connected layers for all the networks except the final fully connected layer uses softmax activation function as it is usually used for producing one output.

In the current work, an open-sourced simulator was used which is made by Udacity and is built in the Unity game development environment, as it is used to simulate end-to-end learning for self-driving cars (Kocic, 2019). It includes two tracks which are the valley track and mountain track that contains different and many road features such as right angle curves, bridges, rams, and lakes trying to represent real road features to improve the learning process for the vehicle (Pal, 2019). There are two types of modes included in the simulator that is the Training mode and the Autonomous mode. The training mode is used for collecting datasets, as the human driver drives the vehicle using a keyboard in the simulator through the track while having three cameras mounted on the vehicle for capturing video from all the three views (Kocic, 2019). The autonomous mode is used for testing and evaluating the model after training. The vehicle in the simulator is controlled by python code using http sockets, as it sends steering angle, speed, and throttle values to the simulator to drive the car (Pal, 2019).

Data collection was done in the training mode of the simulator while driving the car through the whole track several times. Images are captured using the center, left and right cameras which are mounted on the car, therefore there will be three frames for every moment and this will give us the chance to increase the number of frames in the dataset and cover more road features for the better learning process. After finishing the training mode, the frames with their image title will be stored and a CSV file will be created that contains the image title of all the frames with their corresponding steering command, throttle, brake, and speed values. The steering command is represented by 1/r as r is the turning radius in meters, and 1/r is used instead of r to avoid singularity while moving straight (Bojarski, 2016). The value of 1/r will be positive while turning the steering wheel in the right direction and will have a negative value while turning the steering wheel in the left direction and will be zero while moving straight (Bojarski, 2016). The human driver should try to keep the vehicle driving in the center lane while the data is being collected, so the car will imitate the driver's way in autonomous driving (Kocic, 2019).

Many laps were done in the training mode of the simulator to create the dataset, but even after completing a lot of laps, the dataset is relatively small for training a deep neural network that contains a huge number of parameters, therefore the model will be capable of overfitting the data which will increase the error for autonomous driving (Kocic, 2019). In addition, end-to-end learning also requires having a large dataset to make the vehicle experience various different environmental conditions that could happen on the road. Data augmentation is the most used technique for increasing the dataset and reduce overfitting, as it helps in generating different artificial conditions. There are many ways to implement data augmentation on the images such as horizontal and vertical shifts, brightness augmentation, shadow augmentation, and rotation augmentation which are used for augmenting images used for controlling steering angle, but only brightness augmentation and shadow augmentation are used for augmenting images used for controlling speed, as the rotation of images or applying shifts won't affect the value of speed.

Dataset is collected using the training mode in the simulator by capturing video of manual driving. Besides, the video is cut into several images which are recorded with its corresponding speed, throttle, braking, and steering angle in a log file and the images are stored in a separate file. In the current work, the dataset that includes 5 laps of driving the car through the 2 tracks combined together was used. The total number of the images recorded from the simulator was 50,190 with a resolution of 320x160x3 as the height is 320, width is 160, and has 3 deep channels. The steering angles are normalized between -1 and 1 and the speed values are divided by 31 to be normalized between 0 and 1. The data was split into 80% for training samples which are 40,152 images and into 20% validation samples which are 10,038 images as shown in the following table 2.1

Table 2.1: Image split ratio for training the model.

	Total Sample Images	Training Samples	Validation Samples
Valley Track	19,338	15,470	3,868
Mountain Track	30,852	24,682	6,170
Percentage split	100%	80%	20%

Images have gone through data pre-processing techniques before training, as images have been resized into 300x120x3 for training the model. In addition, the sky at the top of the image and the car front at the bottom were cropped to remove insignificant information from the image to speed up the training process, and also images are converted from RGB to YUV. Figure 2.2 represents the image after preprocessing.

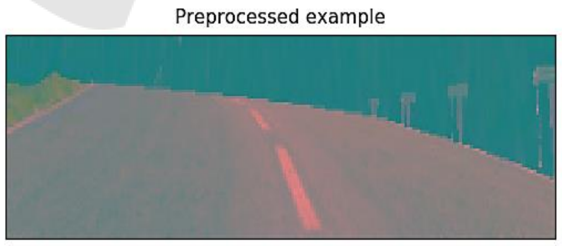


Figure 2.2: Preprocessed Image.

All the images are normalized from a range value of (0,255) to (-1, 1) by dividing the pixels by 127.5 and subtracting 1 from each pixel.

2.2 Autonomous Braking based on the Measured Data

Detecting objects in images could have been done for a very long time. But detecting objects in images that could be streamed like on videos at 15 or 20 frames per second, especially if decent accuracy is required, was a big challenge. So TensorFlow is used to make the detection task easier.

To build an autonomous braking system, the surrounding objects need to be detected and also calculate their distances from the car. If they are far away, then they are not a problem, but if they are really close then action needs to be taken. The action will be a displayed warning to avoid a collision.

First, a pre-trained model (ssd mobilenet v1 coco), which is a single shot detector, is downloaded and used in detecting and classifying the various objects in the image by generating a big number of bounding boxes covering the full image. Next, visual features are extracted for each of the bounding boxes. They are evaluated and it is determined whether and which objects are present in the boxes based on visual features. Then, overlapping boxes are joined together into a single bounding box, where each bounding box represents a part of the image where a particular object is detected and including the score which indicates how the level of confidence for each of the objects.

Then, iteration will be performed through boxes in each frame and check the classes of those boxes if they are cars, buses, or trucks. After that, the score of each class needs to be greater than 50% to make sure that the object is detected correctly.

Now, an approximate distance will be measured by calculating the number of pixels the object is taken widthwise on the screen.

Furthermore, the middle point of each bounding box that contains the detected object needs to be calculated to know where the object is to determine if the object is in the lane, far off to the right, or far off to the left. And even if the detected vehicles were not in the lane of the car and about to change their lane into the car lane, that would trigger a warning message due to their bounding box size. Based on that, two levels of warning will be displayed depending on how the obstacle is far away from the camera.

The value of throttle is needed for applying brakes as it will be calculated using the predicted steering angle and speed values which are the outputs of steering and speed models and the equation will be

throttle = 1 - (predicted steering angle)**2 – (predicted speed)**2.

The value of the throttle will be negative if the brakes should be used to slow down the car.

There are three cases that represent how the obstacle is far away: the first case is (Safe) which means that the obstacle is far away and there is no need to decelerate. As the object is taking less than 40% of pixels widthwise on the screen. The second case is (Watch Out) which means that the object is taking between 40% and 60% of pixels widthwise on the screen which means the obstacle is getting closer and the car needs to decelerate which will be done by applying the brake. The braking value will be estimated by taking the modulus of the throttle value. The third case is (Warning) which means that the object is taking more than 60% of pixels widthwise on the screen which means the obstacle is very close and a collision is about to happen, so the car has to brake and the braking value will be changed to 1 which is the maximum value for braking and also the value of throttle will be 0.

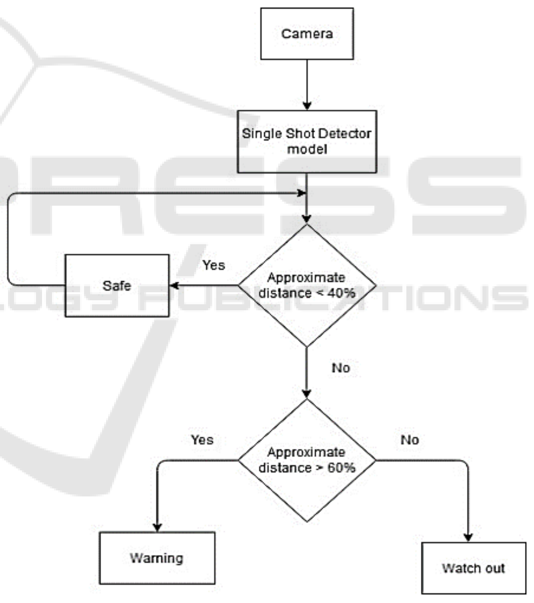

Camera
Single Shot Detector model
Yes
Safe
Approximate distance < 40%
No
Yes
No
Approximate distance > 60%
Warning
Watch out

Figure 2.3: Processes implemented to display a collision warning flowchart.

3 RESULTS AND DISCUSSION

After finishing training for controlling the steering angle and speed for the two different architectures networks, the two networks will be compared with each other based on the network parameters, training, and validation loss for speed and steering models and the computational time taken for training. Models were produced from the training of the networks which will be evaluated in the simulator to compare their performances on the two tracks. Moreover, speed, steering, and detection models will be evaluated on a video from a real road.

3.1 Deep Neural Networks Comparative Study

The two deep neural networks are compared to determine the best-produced model to use in our simulation. The number of trainable parameters depends on the network, connection between the nodes, number of layers, and type of layers which have a huge effect on the computational time and size of the trained model. DNet-3 has 1,214,513 parameters which are divided into 1,213,881 trainable parameters and 632 non-trainable parameters, NVidia network has 1,292,113 parameters that are split into 1,291,321 trainable parameters and 792 non-trainable parameters. Table 3.1 will show the number of parameters for convolution layers in each model as it will be different as a result of the different number of filters.

Table 3.1: Parameters for each layer.

CNN Models	DNet-3	NVidia
Layer 1	1368	1824
Layer 2	1084	21636
Layer 3	18030	43248
Layer 4	9756	27712
Layer 5	15600	36928
Layer 6	1152100	1153610
BN	1264	1584

After seeing the values of table 1, it showed that increasing the number of feature maps and decreasing the number of strides will increase the number of total parameters included in the convolution neural layers. Moreover, there are several non-trainable parameters as a result of using batch normalization to avoid having overfitting. Also, the size of the trained model and computational time were compared to evaluate how the number of parameters affects them. DNet-3 model has a memory size of 14 MB and has taken 900 s for completing one epoch, while the NVidia model has a memory size of 19 MB and has taken 1100 s for completing one epoch which is shown in table 3.2. These results show that the memory size of the model and computational time take for training increases by increasing the number of total parameters for the network. NVidia model has a higher number of parameters which help in extracting more features from the images to increase its performance in driving, but it led to an increase in the memory size and computational time.

Table 3.2: Model size and Computational time.

CNN Model	Memory Size (MB)	Steering Time per epoch (S)
DNet-3	14 MB	900 s
NVidia	19 MB	1100

3.1.1 Steering and Speed Models Training and Validation Loss

Training and validation loss is tracked through the whole training process. Table 3.3 shows the training and the validation loss values for the two models after completing the 50 epochs using the MSE loss function.

Table 3.3: Loss values for CNN models.

	DNet-3	NVidia
Steering Training loss	0.068	0.0624
Steering Validation loss	0.0541	0.0521
Speed Training Loss	0.0076	0.0070
Speed Validation Loss	0.0115	0.0112

3.1.2 Evaluation and Simulator

The steering and speed models of each network are tested together on the Udacity self-driving car simulation to evaluate their performance on the two tracks in the autonomous mode. Table 3.4 presents whether the CNN model was capable of completing the entire lap or not.

Table 3.4: Performance Evaluation for CNN models.

CNN Model	Finished lap on Valley track	Finished lap on Mountain track
DNet-3	Yes	No
NVidia	Yes	Yes

The table showed that increasing the number of filters in the convolution layers helped in making the NVidia model capable of successfully driving and completing the entire lap in the mountain track which has a lot of complex features by predicting accurate steering angles before the right angle curves and controlling the speed values based on the type of the road, while DNet-3 didn't finish the lap in the mountain track as the predicted steering angle and speed value wasn't accurate enough which led into making an accident. Furthermore, the models successfully drove through the entire lap in the valley track as it is much simpler in the road features than the other track. After presenting the results, the NVidia model is the better model to be used for speed and steering prediction.

The next figure is showing the car while entering on a left sharp turn, so the steering angle became negative to turn the steering wheel left to continue driving on the road. Moreover, brakes are applied to decrease the speed of the car before entering the sharp turn in order to drive smoothly through the sharp turn.

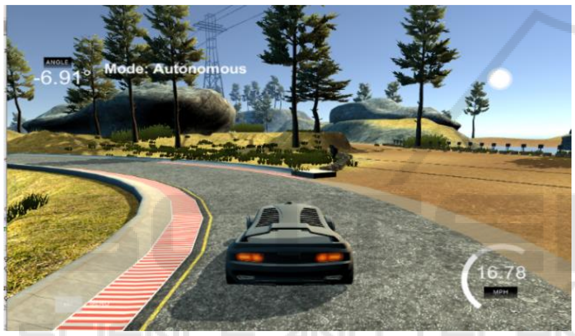


Figure 3.1: Autonomous driving in the simulator track.

3.2 Detection Model

After being able to detect and classify the multiple objects, now the approximate distance between these objects and the vehicle is determined by calculating the number of pixels the object is taken widthwise on the screen. After that, the middle point of each object is calculated to know where the object is to determine if the object is in the lane, far off to the right, or far off to the left. Based on that, two levels of warning are displayed depending on how the obstacle is far away from the camera. There are three cases that represent how distant the obstacle is:

Case 1 (Safe): It means the obstacle is far away and there is no need to decelerate. As the object is taking less than 40% of pixels widthwise on the screen.

Case 2 (Watch Out): The object is taking between 40% and 60% of pixels widthwise on the screen which means the obstacle is getting closer and the car needs to decelerate.

Case 3 (Warning): The object is taking more than 60% of pixels widthwise on the screen which means the obstacle is very close and a collision is about to happen, so the car has to brake.

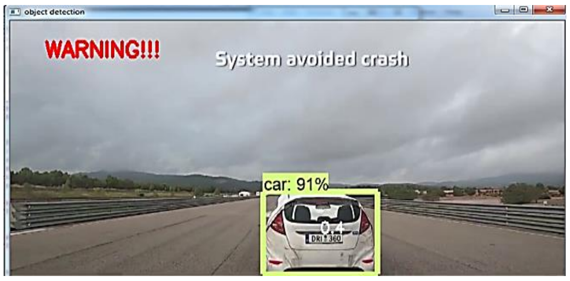
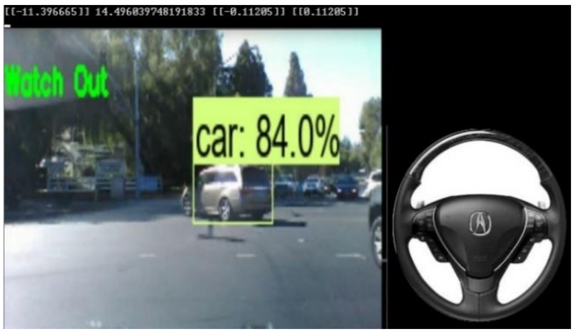


Figure 3.2: Second warning displayed when there is a close car.

3.3 Real Road Video Evaluation

The last results have shown that NVidia models are more accurate in predicting steering angles and speed values as they have got the smaller number for the losses, therefore they were used for the evaluation. The steering and speed models are loaded to the code and connected with the mobile SSD model used for detecting the objects that surround the controlled car and also the braking value will be calculated using steering angles and speed values and also based on the case the car is encountering. These models will be tested on the video of the real road and the predicted steering angle, speed and throttle, and brakes will be shown respectively in the figure.

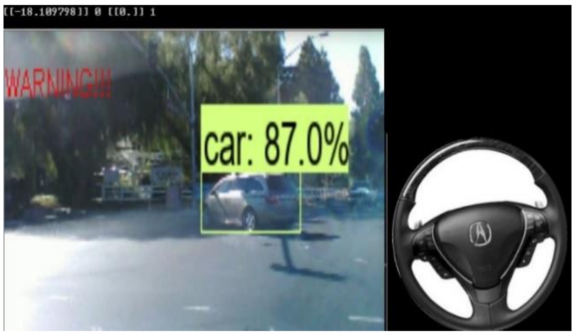
While the car was moving in the road, it was turning left and also there was a car in front of it in the watch out stage, therefore the steering angle became negative to turn the steering wheel to the left and the throttle value became negative to apply brakes and slow down the car to avoid hitting the car in front of it as shown in the next figure



```
[[-11.396665]] 14.496039748191833 [[-0.11205]] [[0.11205]]
```

Figure 3.3: Watch out case in real road video.

The car is still turning left, so the steering angle is negative, but the distance between the controlled car and the one in front of it decreased, therefore the warning stage was entered as the car became very close. Maximum brakes have been applied to force the car to stop before hitting the other car and the speed became zero as shown in the following figure.



```
[[-18.109798]] 0 [[0.]] 1
```

Figure 3.4: Warning case in real road video.

4 CONCLUSIONS

This research paper aims to build an autonomous car system using deep learning techniques by controlling steering, speed, and braking. The work done was implementing two deep neural networks that differ in the number of filters used for the convolution layers which were DNet-3 and NVidia with adding adjustments using CNN to control the steering angle and speed for the vehicle. Steering angles and speed values are used to calculate the throttle and braking values which also depend on the three warning cases applied by the detection model which is a pre-trained SSD model that detects objects surrounding the vehicle and calculate the distance between these objects and the controlled vehicle to choose the warning case accordingly. Increasing the number of filters plays a great role in improving road feature extraction allowing the trained models to drive on complex roads with great performance, but it comes with the expenses of increasing the model memory size and computational time for the training process.

REFERENCES

Szikora, P. and Madarász N., *2017*"Self-driving cars — The human side," *IEEE 14th International Scientific Conference on Informatics*, Poprad, 2017, pp. 383-387.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B, Flepp, B., Goyal, P. , Jackel, L. D., Monfort, M., Muller, U., Zhang, J. , Zhang, X., Zhao J., and Zieba. K. 2016. "End to End Learning for Self-Driving Cars". ArXiv preprints arXiv: 1604.07316v1.

Pomerleau D. A. 1988." ALVINN: AN Autonomous Land vehicle in a Neural Network". In Proceedings of the 1st International Conference on Neural Information Processing Systems.

Farag. W. 2019 "Cloning Safe Driving Behavior for Self-Driving Cars using Convolutional Neural Networks". Recent Patents on Computer Science.

Zhao Y. And Chen. Y. 2019. "End-to-end autonomous driving based on the Convolution neural network model". In Proceedings of APSIPA Annual Summit and Conf.

Heylen, J, Iven, S., De Brabandere, B.,Van Gool J. O. M., L. and Tuytelaars, T. 2018."From Pixels to Actions: Learning to Drive a Car with Deep Neural Networks," *2018 IEEE Winter Conf. on Applications of Computer Vision (WACV)*, Lake Tahoe, NV, pp. 606-615 .

Fujiyoshi, H., Hirakawa T., and Yamashita. T. 2019" Deep learning-based image recognition for autonomous driving". Article in 2019 International Association of Traffic and Safety Sciences.

Eneh, I.I. and Okafor, P.U. 2014 "Design of an automatic brake control system using artificial neural network." Article *in* International Journal of Scientific and Engineering Research, Volume 5, Issue 4.

Chae, H., Kang, C.M., Kim, B., Kim J., Chung, C., and Choi, J.W. 2017 "Autonomous Braking System via Deep Reinforcement Learning." arXiv:1702.02302v2.

Bamigboye, O. O., and Obaje, S.E. 2016 "Intelligent Automatic Car Braking Control System Using Neural Network Classifier." *International Journal of Engineering Inventions Volume 5, Issue 06, PP: 51-56.*

Liu, W., Bao, H., Zhang J., and Cheng, Xu. 2015. "Vision-Based Method for Forward Vehicle Brake Lights Recognition." International Journal of Signal Processing, Image Processing and Pattern Recognition Vol.8, No.6, pp.167-180.

Pal, B., Khaiyum. S. November 2019 "Low Memory Footprint CNN Models for end-to-end Driving of Autonomous Ground Vehicle and Custom Adaptation to Various Road Conditions." In Proceedings of International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-9, Issue-1.

Kocic, J., Jovicic, N. and Drndarevic, V. 2019" An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms". Sensors Book.