Deep Learning Algorithm for Autonomous Driving using GoogLeNet

Mohammed Al-Qizwini, Iman Barjasteh, Hothaifa Al-Qassab and Hayder Radha, Fellow, IEEE

***Abstract*— In this paper, we consider the Direct Perception approach for autonomous driving. Previous efforts in this field focused more on feature extraction of the road markings and other vehicles in the scene rather than on the autonomous driving algorithm and its performance under realistic assumptions. Our main contribution in this paper is introducing a new, more robust, and more realistic Direct Perception framework and corresponding algorithm for autonomous driving. First, we compare the top 3 Convolutional Neural Networks (CNN) models in the feature extraction competitions and test their performance for autonomous driving. The experimental results showed that GoogLeNet performs the best in this application. Subsequently, we propose a deep learning based algorithm for autonomous driving, and we refer to our algorithm as GoogLenet for Autonomous Driving (GLAD). Unlike previous efforts, GLAD makes no unrealistic assumptions about the autonomous vehicle or its surroundings, and it uses only five affordance parameters to control the vehicle as compared to the 14 parameters used by prior efforts. Our simulation results show that the proposed GLAD algorithm outperforms previous Direct Perception algorithms both on empty roads and while driving with other surrounding vehicles.**

I. Introduction

Early efforts in autonomous driving research [1][2] and more recent studies [3]–[5] have targeted various facets of perception, control and decision challenges associated with this emerging area. In particular, the rapid development in deep learning methods using Convolutional Neural Networks (CNNs), which have demonstrated very promising capabilities for accurately extracting features from visual data, opens the door for tailoring these methods for self-driving vehicles.

There are three main general approaches that have been proposed for autonomous driving based on the way the driving environment information is processed. First, the *Mediated Perception* approach in which the structure of the environment is assumed to be unknown and different techniques are used to detect the important features in the environment such as lanes, cars, signs and pedestrians. These methods usually assume that there is an Artificial Intelligent (AI) engine that takes this information and make driving decisions [3]–[6].

M. Al-Qizwini, I. Barjasteh, H. Al-Qassab and H. Radha are with the department of Electrical and Computer Engineering, Michigan State University, East Lansing MI 48824 USA (e-mail: {alqizwin, barjaste, alqassab, radha}@msu.edu).

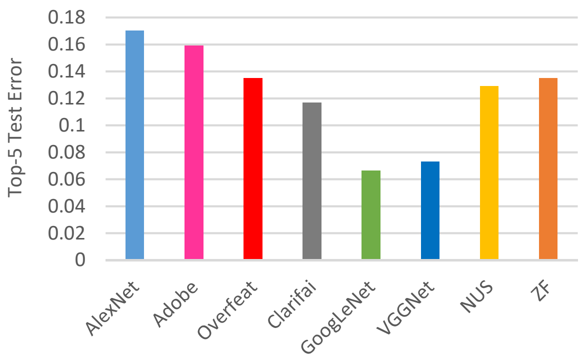


Figure 1: Comparison between state-of-the-art CNNs performance in ImageNet competitions.

The other approach is *Behavior Reflex* in which a neural network model is trained to make driving decisions from monitoring the driving behavior of a human driver in reaction to different driving scenarios [1], [3], [7]. The third approach is called *Direct Perception*, which was proposed in [3]. In this approach, the CNN learns to extract some preselected features from the scene that the authors believe are important to make driving decisions; and subsequently this information is processed by a simple controller to make the corresponding driving decisions.

The Direct Perception approach assumes full knowledge of the road architecture for training purposes, for which the authors employ The Open Racing Car Simulator (TORCS) [8] that has been used by many other efforts as a test driving platform [9], [10]. They also customized it to simulate highway driving conditions. Although the algorithm, which they referred to as ConvNet, is shown to provide superior results when compared to other approaches, the authors in [3] made several assumptions that are unrealistic and inapplicable for real driving scenarios. Firstly, all the affordance parameters in their model, including distances to proceeding cars, are forced to depend on one source of environment sensingwhich is obtained from a frontal view camera that is obtained from the first person driver view of the simulator. However, there are several devices that can be used to provide the controller with more accurate distance measures to other surrounding cars in the environment (e.g. additional cameras, lidar, radar and ultrasonic sensors) [4]. Several automotive companies are already using these sensors for blind spots monitoring and adaptive cruise control systems, and they are proved to be very accurate. Secondly, because their algorithm does not sense other objects next to the autonomous car or behind it, ConvNet assumes that the autonomous car is faster than any other car on the road and by setting a 60 second timer after passing a car, the lane should be clear for a lane change, which makes the algorithm inapplicable for real life driving scenarios. Furthermore, ConvNet is built based on AlexNet [11], which is one of the shallowest CNNs, and it produced the highest test prediction error (top-5) when compared to the other CNNs in the ImageNet competition, as shown in Figure 1 which is plotted using information from the ImageNet website [11].

Our main contribution in this paper is proposing a new, more robust, and more realistic algorithm for Direct Perception autonomous driving. First, we modified the affordance parameters collected for training such that the autonomous car is fully aware of all other vehicles around its surrounding environment. Second, we modified the controller process to provide more accurate decisions based on the new provided information. Third, we built our model based on GoogLeNet which provided the lowest error results when compared to the other CNNs. We refer to the proposed algorithm as GoogLenet for Autonomous Driving (GLAD).

The rest of the paper is organized as follows: In Section 2, the modified affordance parameters and the controller process are explained, Section 3 presents details about the experimental set up including an experiment to choose the CNN that most fits our application. In Section 4, we present the design and architecture of GLAD. Section 5 presents the experimental results in comparison to the previous method with discussions. Finally, in Section 6 we conclude the paper and provide some directions for future work.

II. Affordance Parameters and Controller Process

Similar to ConvNet and other efforts in this area [4], we focus on autonomous driving for highway road conditions. We divide the highway environment into two main objects types; the road markings, i.e. lanes and shoulders markings, and other vehicles on the road. We assume that there is a distance-measuring device (e.g., a radar) that provides accurate distances to other cars within a 60-meter radius around our vehicle.

It is worth highlighting that we made the necessary modifications to the TORCS simulation environment to provide us with such information. This information is provided to the controller process directly to make the appropriate driving decisions. The images obtained from the first person driver view of the TORCS simulator are used to collect the necessary parameters that are required to locate the position and angle of the car on the road by identifying the distances between the car and the road markings as shown in Figure 2. The affordance parameters used in this paper are:

***Angle**:* The angle between the heading of the car and the road.

toMarking_LL: The distance between the center of the car and the left shoulder of the road.

toMarking_ML: The distance between the center of the car and the left marking of the center lane.

toMarking_MR: The distance between the center of the car and the right marking of the center lane.

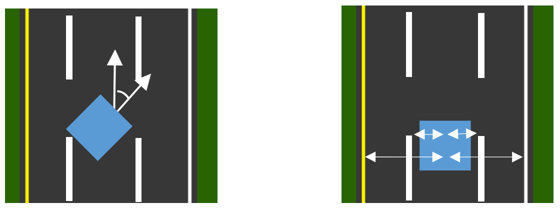


Figure 2: Affordance parameters used for GLAD model training. a- Angle of the car heading relative to the tangent of the road, b- Distances between the center of the car and road markings.

Algorithm1: The controller process for GLAD.

```
Input: output from GLAD model (angle, distances to road
       markings and current lane) and distances to cars within
       60 meters radius.
Steps: If the proceeding car is less than 20 meters ahead then
           If car is not in left lane or on left edge of road and
           left lane clear then
           initiate left lane change procedure
           Else if left lane is occupied and car is not on right
           lane or right edge markings and right lane is clear
           then
           initiate a right lane change.
           Else follow the proceeding car
       If car is not on any road markings, then
           center_lane= center of current lane
       If car is on lane markings (lane change) then
           center_lane= center of target lane
       If car on right road edge, then re-center car in lane
       If car on left road edge, then re-center car in lane
Output: steering, brake and accelerate commands
```

toMarking_RR: The distance between the center of the car and the right shoulder of the road.

For the road marking labels, the authors in [3] used two different sets of variables; four main variables for in lane mode and three other variables for on-road-marking mode. They used all these variables to keep track of the distances to lane markings, which are needed in [3] to maintain correct driving decisions. Instead of these redundant variables we used a combination of the four main variables to keep track of the lane and the lane marking the car is currently driving on. The new GLAD controller process is shown in Algorithm1.

One of the major flaws of the ConvNet algorithm is the assumption that the autonomous car is faster than other cars on the road and within 60 seconds the autonomous car will pass any other car on the road. The assumption is made because the whole model is built on the observed image only, which made the autonomous car incapable of observing the environment next to it and behind it.

As shown in Algorithm1, our controller process has two sources of inputs, the output from the CNN model which is the predicted angle between the car heading and the road and the predicted distances between the car and all the lane markings in addition to the current lane or marking the car is driving on. Hence, if there is a car next or close to the autonomous car on the lane we are trying to move to, the control process will note that and it will not initiate the lane change to prevent a collision, which makes the GLAD algorithm more realistic and more robust for real life implementation.

III. Experimental setup and Selecting the CNN model

One important aspect of using deep learning techniques is selecting the neural network architecture that provides the best accuracy that can run in real time. Before selecting the CNN model for our algorithm, we first ran an experiment to test which of the top three CNNs in Figure 1 performs the best in extracting the road markings. These top three CNNs are GoogLeNet [12], VGGNet [13] and Clarifai [14]. For training, we used 10 out of the 18 tracks in [3], the rest are used for testing the ability of the models to adapt to new environments. We also employ the same 22 cars that were used in [3] after updating the speed of three cars to be faster than our autonomous vehicle. We collected a total of 510,112 images of size 280x210 from driving the label-collecting car for six hours on different empty tracks and eight hours with other cars. The collected data has five labels for the affordance parameters we used, as explained in Section 2. We trained each CNN for 300,000 iterations and selected the model with minimum error. During the test phase, we present 25,000 random "unseen" images from the TORCS simulator and have each model predict the affordance parameters. Since we do nott have discrete labels, we cannot make top-5 guesses similar to the ImageNet competition; instead we used the Root Mean Squared Error (RMSE) to evaluate the performance of the CNNs. This comparison is shown in Figure 3. In Figure 3, one notice that GoogLeNet provided the most accurate predictions for the angle and distances, which is promising for achieving significant performance improvements in the context of autonomous driving. In this paper we focus only on highways with two and three lanes road models similar to most of highway roads.

IV. GLAD Design and Performance Results

As mentioned earlier, the proposed framework is based on the standard GoogLeNet CNN [12]. In particular, we modified the model by adding a sigmoid and Euclidian loss layers instead of each softmax layer in the original model since we are interested in the marginal distribution of each predicted parameter rather than the joint distribution of all the variables. The architecture of the model is shown in Appendix A. We used the Caffe framework to run our deep learning algorithm. As shown in Appendix A, GoogLeNet has 22 convolutional layers, so generally we would expect it to require more training than AlexNet, which has only 8 layers.

However, the experimental results showed that it converges fast. We chose a maximum number of 500,000 iterations. In each training iteration, we used 64 images per training batch. We noticed that the model converges at around 290,000 iterations, so we used that model for testing. During the test phase, we programmed the controller process to drive the autonomous car based on the objects extracted from the frontal view camera and the distances to cars provided by the distance measuring devices.

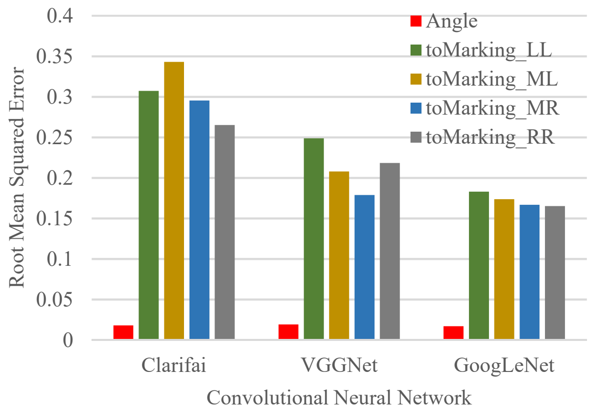


Figure 3: Comparison between state-of-the-art CNNs performance for car heading angle and road marking features extraction.

The frontal view image obtained from TORCS is passed to the model to predict the affordance parameters. The output parameters are passed to the control process in addition to the distances to other cars within 60 meters radius. Subsequently, the control process outputs the three main car control parameters which are: acceleration command or percentage of throttle to be pressed, brake which is also percentage of brake to be pressed and steering which is represented by the angle of tires to minimize the angle between the car and the road (θ). We used the standard steering command from TORCS [8]:

$$S = \frac{\left(\theta - \left(\frac{d}{W}\right)\right)}{s_{lock}} \qquad (1)$$

where s_{lock} is a constant that varies with different cars and it is used to normalize the steering command to the range [-1,1]. d is the distance between the car and the center line of the current lane, or the target lane in case the car changes lanes. W is the lane width.

To prevent the car from oversteering, we also added a function to reduce the speed based on the turning angle of the tires that only applies on high speeds, which is represented as:

$$T = T'(1 - |S|) \qquad (2)$$

T' is the throttle percentage when the steering angle is zero.

V. Results and discussion

As we mentioned earlier, in this paper we focus on the autonomous driving performance rather than feature extraction as in [3]–[5]. For the sake of fair experimental comparison, we ran all algorithms outlined in this section on a desktop computer with GeForce GTX980 Ti. graphics card running Ubuntu 14.04 LTS. Most research papers that target autonomous driving judge the performance of their algorithms based on how accurately the algorithm is capable of predicting obstacles and lane markings only. In addition to road features extractions and obstacles detection performance comparisons which are very crucial to the safety of autonomous driving, we also add another set of evaluation parameters that are related to the autonomous driving experience. The next subsections explain both parameters evaluations and show the related experimental results.

A. Convolutional Neural Networks Performance Evaluation

Since our algorithm uses 5 affordance parameters only compared to the 14* parameters of ConvNet, we found it interesting to study the performance difference between ConvNet5, ConveNet14 and GLAD as training progresses. In this experiment we trained ConvNet5 on the same data set we collected.

For ConvNet14, we used the same model produced in [3]. We trained each model for 300,000 maximum iterations and tested the models generated every 10,000 iterations on a validation set of 30,000 images; half of the validation set was provided to the algorithms during training and the other half was not seen by the algorithms. We recorded the average RMSE of all validation parameters as the prediction error of the CNN. The comparison result is shown in Figure 4.

Since the weights of CNNs are initialized from a random Gaussian distribution, we notice in Figure 4 that the prediction error starts at higher levels and it keeps fluctuating at the beginning of training. However, as the training progresses these fluctuations are reduced and the prediction error is reduced. From the same figure, we notice that using the same CNN model, ConvNet5 produces less training prediction error than ConvNet14. It also shows that GLAD outperforms both ConvNet14 and ConvNet5. To show the performance comparison between GLAD against ConvNet with 14 and 5 parameters, we let the autonomous car drive in four tracks (two 2-lane tracks and two 3-lane tracks) that are not shown to the model during training. We calculated the average error and recorded the results in Figure 5.

Since there is no reason to show the remaining nine parameters of ConvNet14, we will omit them from this comparison and display only the error of the affordance parameters we will be using. The reason for the poor performance of ConvNet relative to GLAD, is due to the small number of convolutional layers in AlexNet model, which are the core of the feature extraction capability of the CNN.

Hence, ConvNet5 is not able to extract the road features as accurately as GLAD. Since ConvNet5 did not provide sufficient accuracy improvement over ConvNet14, we will continue the experiments with ConvNet14 only and we will refer to it as ConvNet.

B. Autonomous Driving Performance Evaluation

For the experiments in this section, we first define some measures for the autonomous driving performance.

*In [3] the authors mentioned that they used 13 affordance parameters. However, their CNN architecture consists of another parameter called "fast" to indicate whether the car is exceeding the desired speed or not.

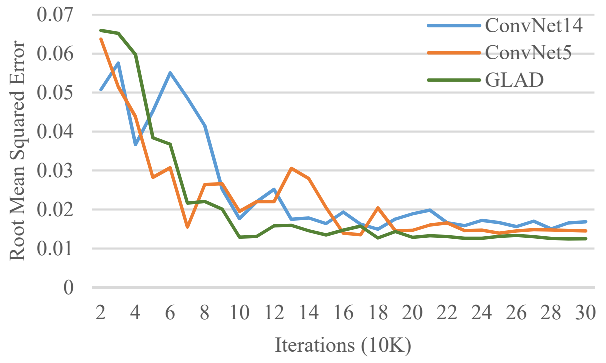


Figure 4: Training error comparison between ConvNet14 and ConvNet5 parameters estimations and GLAD.

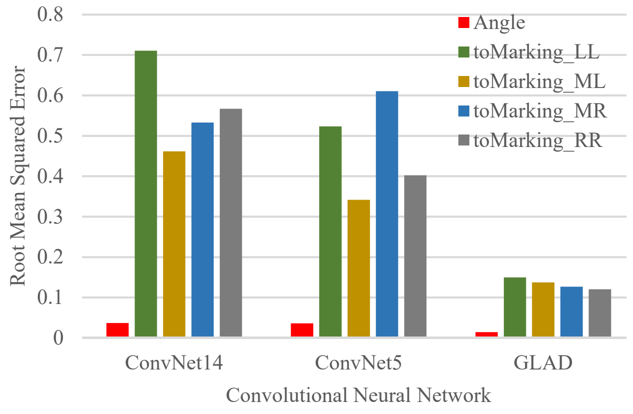


Figure 5: ConvNet14 vs ConvNet5 vs GLAD test phase performance comparison.

In particular, we are interested in characterizing how much the vehicle deviates from the ideal position of driving exactly on the center of the lane that the vehicle is supposed to be following. Thus, the first parameter that we quantify is the sample mean of the deviation from the middle lane (μ); this parameter indicates how close to any lane marking the car is driving. Here, we use the parameter d, which in the TORCS simulator it could take both negative and positive values to indicate whether the car is deviated towards the right (positive) or towards the left (negative) of the middle of the lane. Hence, we evaluate the sample mean of the absolute value of d: $\mu = \frac{\sum|d|}{N}$, where N is the number of collected samples calculated from the number of frames in one lap of each track. This parameter represents the CNN's ability to keep the car centered in the current lane. Larger mean indicates that the car is leaning towards either ends of the lane.

The second parameter is the variance of the distance from the middle of the lane d. $\sigma^2 = \frac{\sum d^2}{N} - \mu^2$. This measure gives an indication on how stable the steering wheel is when driving on a specific lane. In another words, it represents the CNN's ability to keep the car driving in a straight line rather than wondering inside the lane. Here it is important to mention that both μ and σ^2 are measured from the distance between the car and the middle of the lane and not the error in prediction of that distance.

Table 1 shows the mean and variance measures for the 2-lane and 3-lane tracks that are shown in Appendix B. The first six tracks are 2-lane roads and the rest are 3-lane roads. The first three tracks of each set are seen during training and the other three are new to the autonomous car.

From Table 1, we notice that in general GLAD is capable of keeping the autonomous car centered in the lane and without any steering wheel oscillations more accurately than ConvNet. However, for Alpine 2, we noticed that both algorithms showed performance degradation due to the fact that the track simulates a snowy weather as shown in the appendix, so the side roads are white and since both algorithms were not subject to a similar environment during training, the CNNs were not able to control the car as accurately as the rest of the tracks. Also for Aalborg, both algorithms did not perform as good as other tracks due to the wall that surrounds the track, which caused both CNNs to make false road markings predictions. On the other hand, even though CG-Track 2 has different road markings structure with larger gaps between consecutive lanes markings, as shown in the appendix, and such lane markings where never seen during training, GLAD was able to successfully keep the car in the center of the lane as shown by the variance while ConvNet struggled to even keep the car in the center lane. Furthermore, note that although ConvNet has a smaller sample mean for the Wheel 2 track, the variance for ConvNet is significantly larger (close to an order of magnitude) than the variance for GLAD.

Another parameter that we define to judge the autonomous driving performance is the total time that the autonomous car spends driving out of lane and/or crashing measured over two laps of each track. The time of error (t_{error}) in seconds is computed in relative to the number of frames until the car goes back on the road (N_{error}) and the Frame Rate (FR), as shown in (3). The comparison results of evaluating this parameter are shown in Table 2.

$$t_{error} = \frac{N_{error}}{FR} \qquad (3)$$

Since GLAD algorithm runs at 10 Frame Per Second (FPS), we simply divide the number of frames when an error or accident happen by 10.

Figure 6, shows some of the prediction errors caused by depending solely on the frontal view camera in ConvNet. It also shows how using other technologically advanced distance measuring sensors such as radar, lidar or ultrasonic sensors can help improving the experience of autonomous driving.

VI. Conclusions

In this paper we introduced a new Direct Perception deep learning algorithm for autonomous driving. Unlike previous efforts that focused on the feature extraction capabilities of a given CNN, we first studied autonomous driving performance for different deep learning architectures.

TABLE 1: ConvNet vs GLAD, the mean and variance of the distances of the car from the center of the lane on different tracks.

Track	ConvNet		GLAD	
	Mean	Variance	Mean	Variance
Alpine 1	0.21687	0.03519	**0.16500**	**0.03705**
E-Track 2	0.16836	0.06036	**0.02028**	**0.01471**
E-Track 3	0.14689	0.04391	**0.01172**	**0.03831**
Alpine 2	**0.25425**	**0.23766**	0.27180	0.24555
Aalborg	**0.59234**	0.84689	0.59292	**0.67478**
Dirt 3	0.23766	0.20532	**0.04146**	**0.01078**
E-Track 6	0.30655	0.27723	**0.08570**	**0.09244**
E-Road	0.33208	0.49586	**0.10651**	**0.22325**
Street 1	0.13214	0.08061	**0.02834**	**0.07860**
Wheel 2	**0.02568**	0.36832	0.07356	**0.05267**
E-Track 1	0.31971	0.57070	**0.14789**	**0.51295**
CG-Track 2	0.50426	0.99828	**0.24885**	**0.10354**

TABLE 2: ConvNet vs GLAD error time comparison approximated to the nearest tenth of the second.

Track	ConvNet	GLAD
	Error Time (Sec.)	Error Time (Sec.)
Alpine 1	**1.9**	2.7
E-Track 2	17.1	**5.4**
E-Track 3	5.7	**2.1**
Alpine 2	18.5	**7.0**
Aalborg	80.7	**24.4**
Dirt 3	58.7	**7.0**
E-Track 6	10.4	**8.3**
E-Road	5.2	**7.1**
Street 1	6.8	**3.7**
Wheel 2	11.2	**2.3**
E-Track 1	75.2	**10.1**
CG-Track 2	16.7	**16.4**

Furthermore, our proposed framework is not constrained by unrealistic assumptions about other vehicles surrounding the autonomous car. For example, our algorithm works without making assumptions about the speed of the car; instead our algorithm uses a more realistic model by assuming the availability of additional sensors that provides distances to the cars around the autonomous vehicle. We compared the performance of the top three CNNs in road feature extraction; the results showed that GoogLeNet is the most accurate CNN for that task. In addition to the feature extraction performance, we also suggested using additional parameters to evaluate the autonomous driving experience and performance. Finally, we used the suggested parameters to compare our algorithm against the previous Direct Perception algorithm and we showed that the performance of our model is significantly improved relative to the previous effort, which struggles to navigate the whole track. The reason for this improvement is due to removing the overlapped and redundant affordance parameters that are used in [3] in addition to choosing a CNN model that is capable of extracting road features more accurately than the model they used.

For future directions, we are planning to include other parameters to the driving equations, such as human driver fault tolerance for accident avoidance. We are also looking into adapting the TORCS environment to represent city environment driving and also modifying our algorithm to work with pedestrians, bikes, motorcycles and animals detection.

🟨	TORCS real car location.
□	Predicted car location.
🟥	TORCS real Autonomous car location and angle.
□	Predicted Autonomous car location and angle.

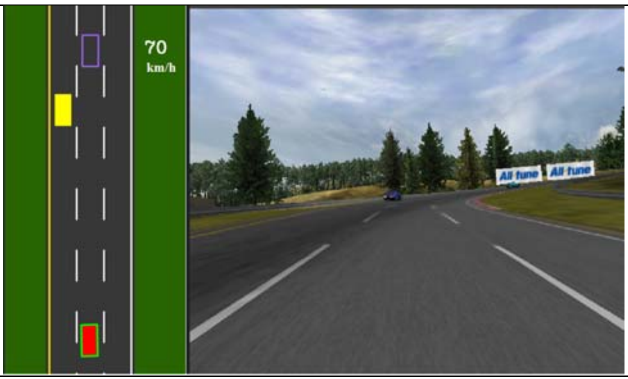
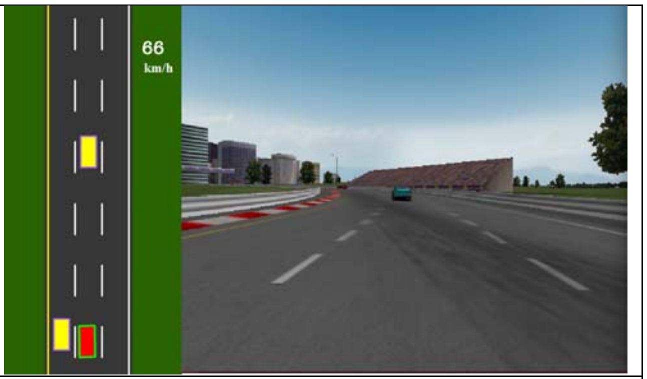
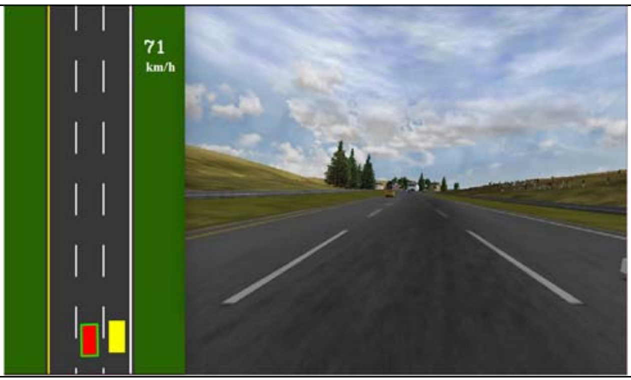
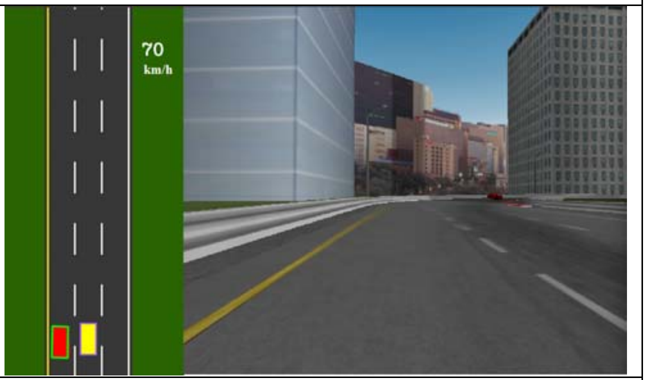
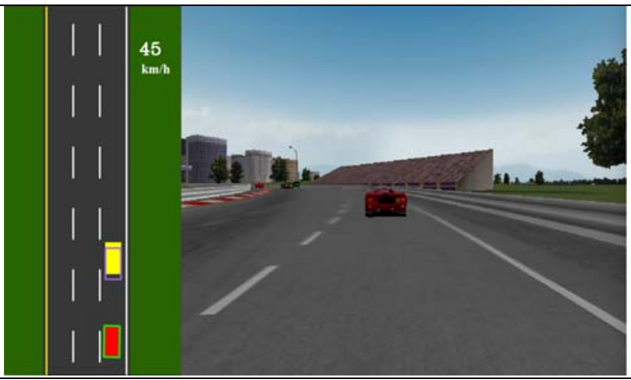
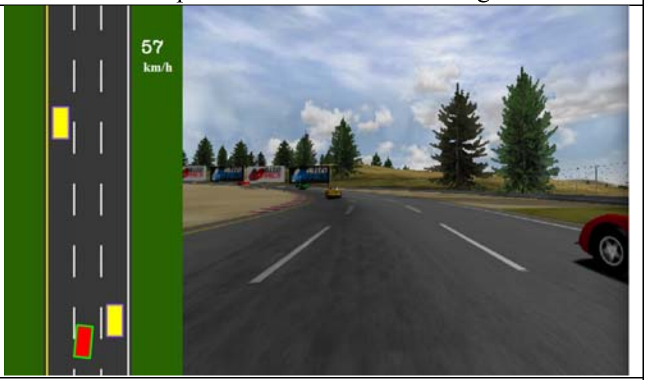
ConvNet	GLAD
	
Car distance and lane error.	Accurate distance is provided by distance measuring sensors.
	
No sense of the environment next or behind the car.	Complete sense of all surroundings.
	
Car goes to following mode after passing another car waiting for the timer to reset	No timer, car goes back to lane after passing another car.

Figure 6: ConvNet vs GLAD predictions error and surroundings sensing.

REFERENCES

[1] D. a Pomerleau, “Alvinn: An autonomous land vehicle in a neural network,” *Adv. Neural Inf. Process. Syst. 1*, pp. 305–313, 1989.

[2] S. Oh, E. Kim, and J. Lee, “Autonomous Intelligent Cruise Control using Scanning Laser Sensor,” pp. 1–7.

[3] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, “DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving,” *2015 IEEE Int. Conf. Comput. Vis.*, pp. 2722–2730, 2015.

[4] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, and A. Y. Ng, “An Empirical Evaluation of Deep Learning on Highway Driving,” *arXiv*, pp. 1–7, 2015.

[5] M. Aly, “Real Time Lane Detection in Urban Streets,” *Intell. Veh. Symp.*, pp. 1–3, 2008.

[6] A. Jazayeri, H. Cai, J. Y. Zheng, and M. Tuceryan, “Vehicle Detection and Tracking in Car Video Based on Motion Model,” *Intell. Transp. Syst. IEEE Trans.*, vol. 12, no. 99, pp. 1–13, 2011.

[7] M. Felsberg, A. Robinson, and K. Ofj, “Visual Autonomous Road Following by Symbiotic Online Learning,” no. Iv, 2016.

[8] B. Wymann, G. Antonio, and A. Corral, “T.O.R.C.S. Manual installation and Robot tutorial.”

[9] Z. Xu, J. Jiang, and Y. Liu, “Experimental Research of Vehicle-Platoon Coordination Control Based on TORCS Platform,” pp. 7404–7409, 2016.

[10] J. Huang, I. Tanev, and K. Shimohara, “Evolving a General Electronic Stability Program for Car Simulated in TORCS,” pp. 446–453, 2015.

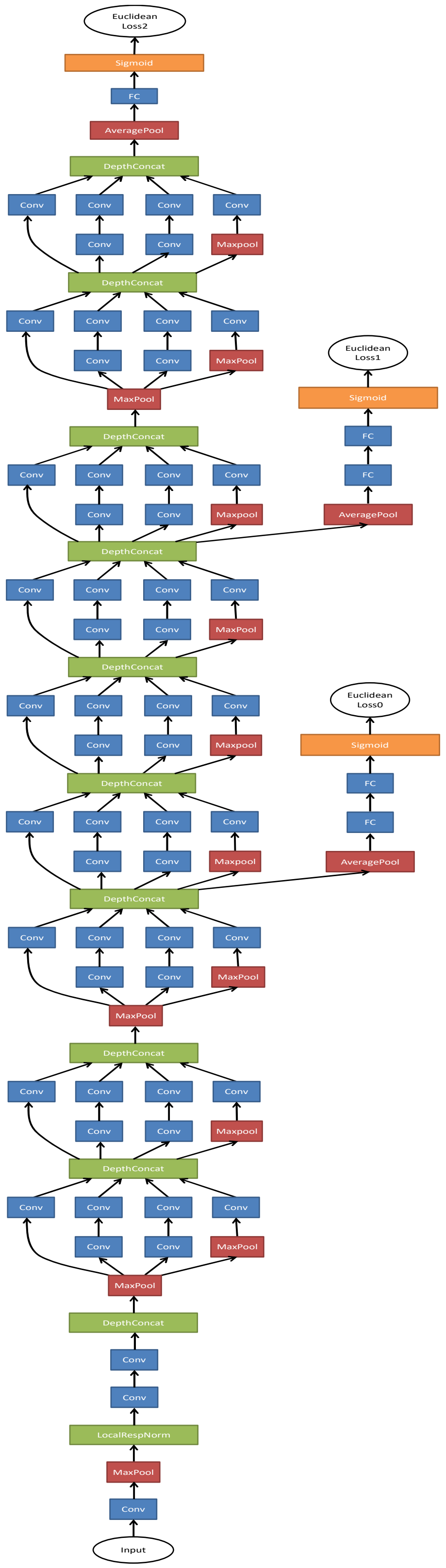
[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2012.

[12] C. Szegedy, W. Liu, Y. Jia, and P. Sermanet, “Going deeper with convolutions,” *arXiv Prepr. arXiv 1409.4842*, 2014.

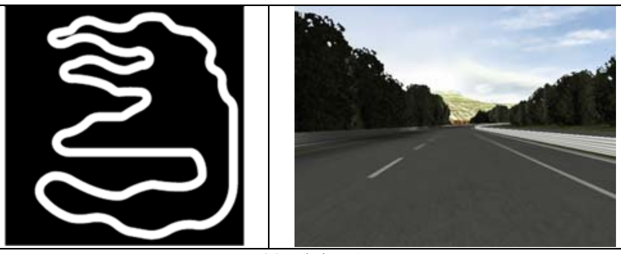
[13] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *Int. Conf. Learn. Represent.*, pp. 1–14, 2015.

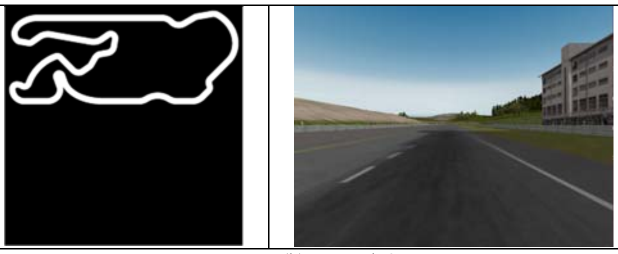
[14] M. Zeiler and Ro. Fergus, ““Visualizing and Understanding Convolutional Networks,” pp. 1–11, 2015.

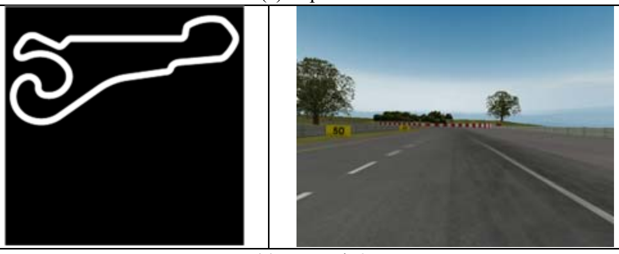
APPENDIX A: GOOGLENET ARCHITECTURE FOR GLAD MODEL.

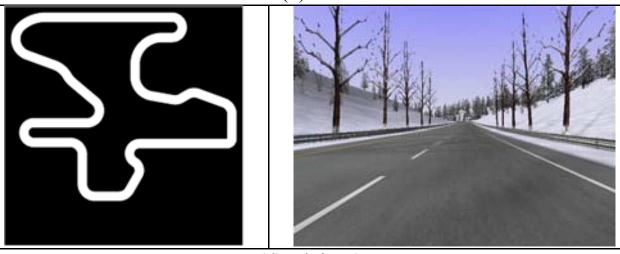


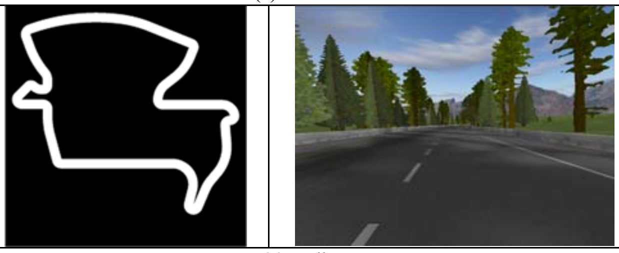
APPENDIX B: TORCS TRACKS THAT ARE USED DURING THE TEST PHASE FOR THE CNNS' PERFORMANCE COMPARISON.

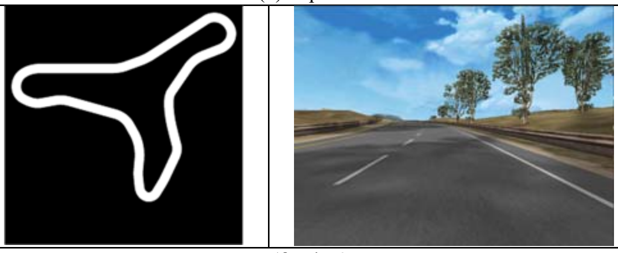

(a) Alpine 1

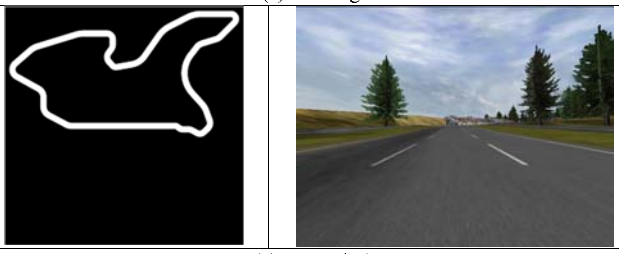

(b) E-Track 2

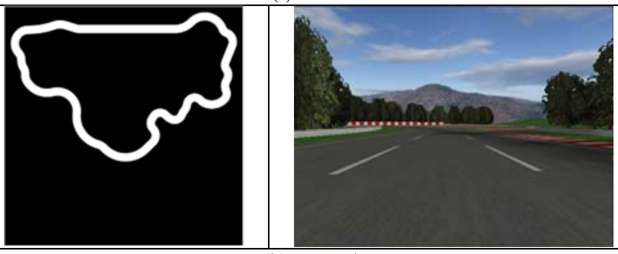

(c) E-Track 3

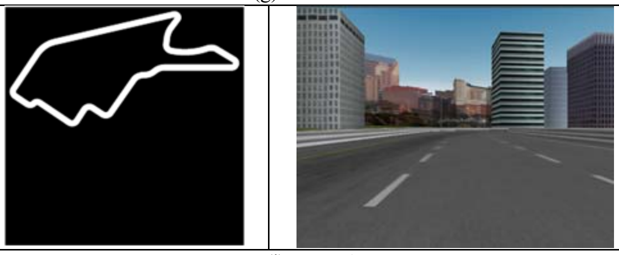

(d) Alpine 2

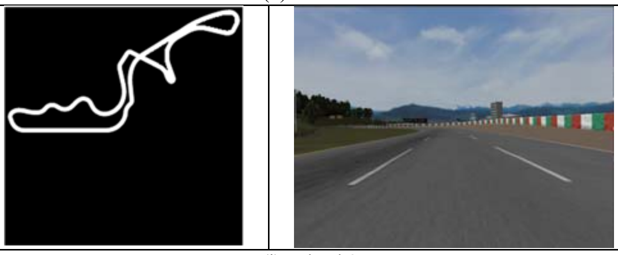

(e) Aalborg

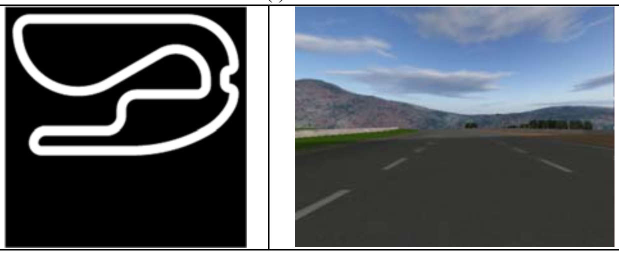

(f) Dirt 3

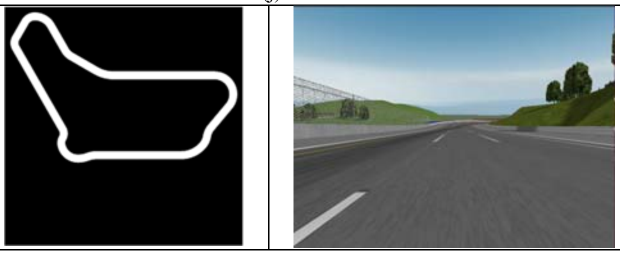

(g) E-Track 6


(h) E-Road


(i) Street 1


(j) Wheel 2


(k) E-Track 1


(l) CG-Track 2