Low Memory Footprint CNN Models for end-to-end Driving of Autonomous Ground Vehicle and Custom Adaptation to Various Road Conditions.

Bishwajit Pal, Samitha Khaiyum

Abstract*: The Idea presented in this paper is to establish that different types of Convolution Neural Network (CNN) model are required to be used for different types of terrains, to optimally drive the vehicle. The Layers and the trainable variables required to be configured as per the road conditions. For simpler roads, lighter and simpler networks and for complicated road, a CNN model with heavier network is required. Existing CNN models were used to build the performance baseline with respect to simple and complicated road conditions. In the proposed CNN models, layers and trainable parameters are adjusted according to the road conditions. The objective of the proposed solution is to successfully drive with minimum number of convolution layers and trainable variables, fit for deployment on computer consuming less than 100 watts, without GPU, for moderate speed autonomous vehicle. These new sets of proposed CNN models are either equal or smaller in network size and trainable parameters, memory footprint and refresh rate compared to Alexnet and Nvidia model. The overall requirement of computational power, cost, and size is also reduced. In this paper, we recommend designing AV software with multiple CNN models to address different road conditions instead of on one Model for all road conditions. Finally, we establish that when the terrain is more complex and requires more features to be extracted then the CNN layers need to be tweaked to extract more features and the speed of the vehicle needs to be reduced.

***Keywords* : Autonomous driving, camera, convolution neural network, deep neural network, machine learning.**

I. INTRODUCTION

Innovative work in the field of AI and all the more absolutely profound learning lead to numerous revelations and commonsense applications in various areas. The area where AI has an enormous effect is the car business and the improvement of completely self-governing vehicles. AI arrangements are utilized in a few self-governing vehicle subsystems, as the observation, sensor combination, concurrent limitation and mapping, and way arranging. In parallel with work on full self-rule of business vehicles, the improvement of different car stages is the present pattern.

The proposed solution fits well for self-driving vehicles with constrained equipment assets, computation power, and memory size. Having those equipment confinements at the top of the priority list, a light deep neural system (DNN), model is designed to execute on low-performance processor platforms. The high level architecture of self driving vehicle has typically 4 connected layers namely Sensors, Perception, Planning-decision and Control-Actuation as shown in fig 1. The Sensors can be used for sensing the external environment and Sensing vehicle centric sensors. LiDAR and Camera are used to sense the external environment whereas Critical vehicle metrics like odometer, tachometer, incremental encoders, etc are used to sense vehicle metrics. In the perception phase all the information is processed into data formats which the computers can process. This phase also involves filtering any erroneous data. The "Planning and decision" system is responsible for generation path at real time to make the vehicle drive on road. The control and actuation system is responsible to execute planning commands into vehicular commands resulting in motion.

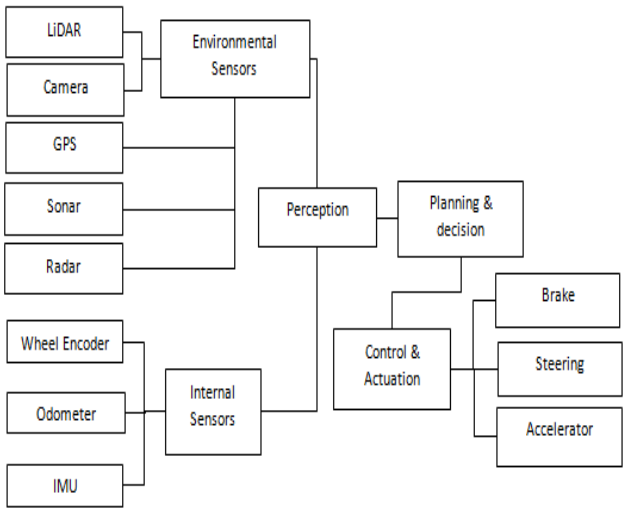

Fig. 1. System block Diagram of Autonomous vehicle

Revised Manuscript Received on 14, October 2019.

Bishwajit Pal, Department of MCA, Dayananda Sagar College Of Engineering, Bangalore, India. (Email: bishwajit.pal@gmail.com)

Dr. Samitha Khaiyum, Department of MCA, Dayananda Sagar College Of Engineering, Bangalore, India. (Email: hod-mcavtu@dayanandasagar.edu)

Retrieval Number: A9165119119/2019©BEIESP
DOI: 10.35940/ijitee.A9165.119119

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

In the proposed CNN networks is designed to accept 3 stream of camera and produce steering, brakes and speed control. Finally this CNN has to drive the vehicle through the terrain in which it has been trained.

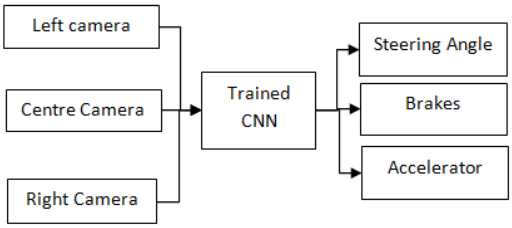

Fig. 2. Block diagram of interface between AI sensor and actuators.

AI is finding its way to small size computing platforms like IoT, Single board computers, embedded systems and even cameras and moble phones. This has lead to development of light weight applications of AI and deploying pre-trained network on low end computers.

The known answers for start to finish learning for self-sufficient driving [4], [5], [14], [9], [30] are grown for the most part for the genuine vehicles, where the AI model utilized for deduction is conveyed on the superior PC, which is typically situated in the storage compartment of the vehicle, or those arrangements utilize profound neural systems that are computationally costly (e.g., utilizing ResNet50 design in) [14].

Existing self driving systems uses heavy computational platforms which are costly and complex like the NVIDIA DRIVE(TM) PX system used in [4], [5] and dataset specific to costly LiDAR [7] with high end computers. Such system is mounted in the trunk of the vehicle and consuming few kilowatts of electricity. The energy requires running such heavy computer comes from the gas engine running the vehicle. This can substantially increase the carbon footprint of the vehicle.

To counter this problem, we propose to use the roof surface of the vehicle to generate power from solar energy and store it in Li-ion batteries and then use this power source to run SBC computers with the proposed CNN models. Based on the road types the appropriate CNN can be used instead of using one very complex, computationally expensive network [14].

In this paper, we present a methodology where the system gets to choose the CNN network which satisfies well with the road conditions. The priority will be to choose a lighter CNN network with higher refresh rate than the heavier one. All the experiments conducted are conducted on the car simulator released by Udacity online course on self driving car [30]. The results of the training and driving performances on the track is presented and compared in this paper.

Our experiments show that a CNN with finer details in the early layers are more precise and can be used in tough toads. Whereas CNN with less feature extraction in the initial layers do well for easy road conditions

II. LITERATURE REVIEW

Deep neural network involves multiple layer [2], [9], [25] where the output presentation of one layer is available to another layer using a kernel. The Output keeps on changing from one layer to another in a unique fashion way, strongly governed by the kernels used. The CNN are useful when the data has some specific patterns or features, like character recognition [13], [22]. ImageNet [18] has also contributed to the early development of CNN. A detailed comparison of DNN used for autonomous vehicles can be found in [28],The most widely known and proven performance CNN networks are Residual Neural Network (ResNet) Microsoft Research [10] , VGG-16 from University of Oxford [23]; ReNet from University of Montreal [31], GoogLeNet from Google [27] and AlexNet from University of Toronto [12]. CNN is widely used for playing games [20], [21], automated machine learning [35], video-to-video synthesis [32].

Some of the other applications of deep learning are, DNN on synthetic data [1], video-to-video synthesis [32], and learning algorithm in NLP [3], Remote sensing [8], Sound analysis [34] etc. The concept of autonomous vehicle is not new. The most Significant development happened when the DARPA announced the AV challenge [29] and then expanded to urban challenge [6].

Autonomous vehicle are very complicated equipment and hence there are challenges in designing systems for sensor fusion [33], higher/low level planning & decisions [17], end-to-end learning for autonomous driving [24], [11], reinforcement learning for autonomous driving [19],

RGB cameras are lower in cost, mainly used to detect patterns. ML algorithms for texture classification rely heavily on Cameras, which requires heavy computation. to achieve higher accuracy, algorithms tends to be complicated and computationally heavy[15]

Developing a light weight machine learning which is accurate, low in hardware cost and solution is difficult to design [15], [16], [26]. In this paper we stared with the Nvidia Model as the baseline, staring with tweaking the features and parameters according to the track on which it's designed to execute.

Overall the contribution of this paper is:

1) We tested a set of CNN network by varying the kernel size, trainable parameters and layers, hence making those different size and computational requirements and tested their behavior on different road conditions.

2) The CNN networks will have at least 15 frames per second or refresh rate as one of the main criteria. These needs to be done without any GPU support and power required should be derived from the solar panels in the car, as much as possible.

3) Minor Modification to existing CNN models to suit the need of self driving.

III. SELF DRIVING CAR SIMULATOR

In our experiments we have used an end to end learning for self driving car, open sourced by Udacity [30]. This self driving car used CNN to drive the scar on a track. The track consist of various road marking features and turns, rams, bridges etc. the CNN training is done using inputs from 3 dashboard camera mounted on the centre left and right side of

the vehicle. It also includes the parameters from speed, throttle and steering angle. To train the model the simulator provides a training mode where a human diver is allowed to drive the vehicle through a track and the camera and video data is captured. This recorded data is then used to train the CNN model. Once the training is done the H5 model files are created by the CNN trainer. These trained models are then used to autonomously drive cars in the simulator.

The simulator environment is designed using Unity engine [30] and Python executable established control over the car using http sockets. The car in the simulator is coded to listen to the socket for steering, speed and throttle variables and at the same time it send the 3 camera picture to the python. The tracks features different types of patterns and road textures resulting in better learning and handle different situations.

IV. DATASET

The simulator provides training mode for capturing a manual driving scenario. This is a supervised learning. In the training mode the image from the three cameras, speed, throttle and steering parameters are recorded into a log file. The images are stored in a separate directory. The parallax effect due 3 different cameras will help keeping the model away from over fitting and during autonomous driving it will reorient the vehicle to the centre of the road.

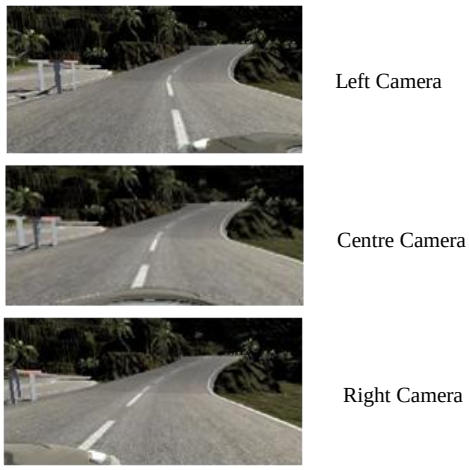

Fig. 3. Example of a image captured by all 3 camera.

To keep the data away from erroneous data the training driving will be done with half the speed, without swaying from the middle of the road. Brakes are applied at every sharp turn and turns are made very smoothly. This is done to relate the curve with the steering angle. This whole process is repeated for the track, the simpler and the mountain track. To make the learning more effective, multiple laps were recorded.

Table- I: Image split ratio for training of CNN Models.

	Total Sample Images	Training	Validation
Valley Track	10,992	8,794	2,198
Mountain Track	22,758	18,206	4,552
Percentage split	100%	80%	20%

The image captured by the simulator is a 320 X 160 24 bit 96 dpi RGB tagged along with the speed, steering and throttle values. The minimum amount of images used for training is 20,000 and for the mountain terrain is 32,000. We have used the 80-20 data split for training validation of the CNN models as shown in Table-1. Cropping of the sky from the image was done to limit unnecessary features into the CNN network.

V. DESIGN

The main idea in designing the Drive-Net set of CNN is to establish the performance and capability of CNN networks in different condition of road features with minimalistic layers and computational requirements. In A CNN network addition of layers and kernels increases the accuracy of the overall network but at the cost of higher computational needs. It also means, lower the number of parameters, layers, kernels or higher striding, the lesser the memory and CPU cycles it will take to execute that CNN model.

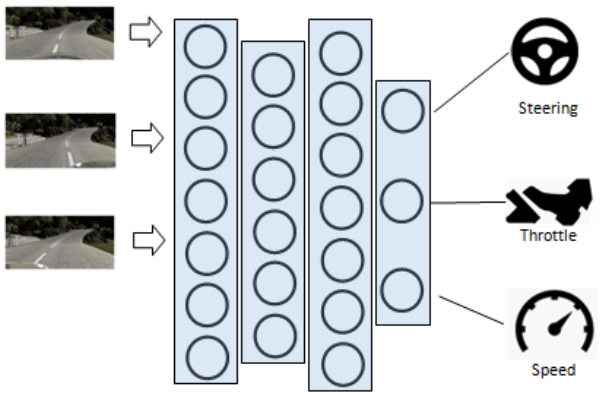

Fig. 4. Architecture of the CNN model used for Autonomous driving

The proposed DNet, abbreviated as Drive-Net is a set of CNN neural network having various patterns in the configuration of the convolution layers. For simpler road, less layer with high striding is used whereas for complicated road extra layer with less striding. By simple and complicated road we mean that a simple road has less or small number of pattern emerging recurrently. Whereas a complicated road means there are many geometric patterns at a lesser number of occurrence. A desert road or a valley road can be considered simpler than sharp curvy road of a forest or mountain. Totally 6 CNN network is designed to address this scenario. The CNN models DNet-1, DNet-2 and DNet-3 are modeled for simpler road where the input layer has less number of kernel filters. The next 3 CNN models, DNet-4, DNet-5 and DNet-6 are for maximizing feature extraction where the first layer is having higher kernel filter values than the previous 3 CNN Models. Max pooling is used to avoid over-fitting and easy on processors as it doesn't add any new parameter.

The first model CNN model, DNet-1 is a 4 convolution layer network. Starting with a kernel of 12 output filters of the size 5 X 5 on a 2 X 2 striding pattern. This layer will filter major features from the image. The DNN structure is shown in fig 5. To reduce over fitting a dropout layer is added after the 4 Convolution layer. Following the dropout layer is the flatten layer which converts the data to a single dimension. 4

set of dense layer is then added in a reducing manner to finally produce the vehicle controls.

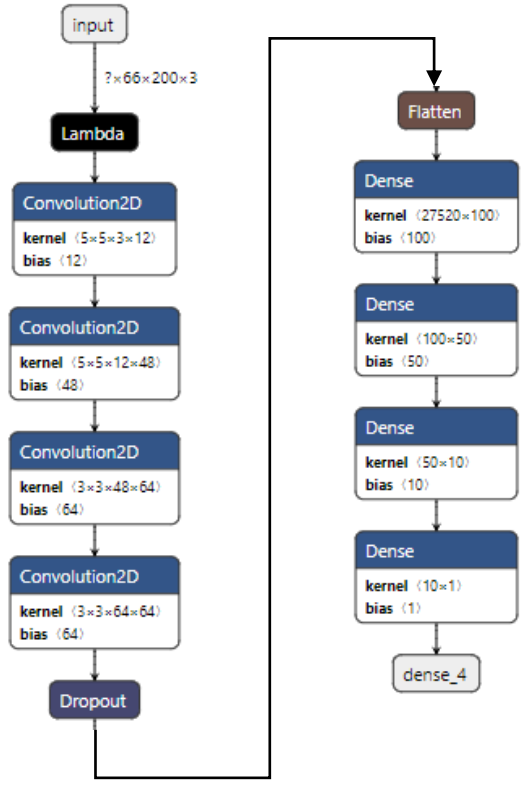

Fig. 5. Neural network design of DNet-1

The DNet-2 in fig 6 has 5 layers instead of 4 starting with 18 feature maps in the first convolution layer which is 50% more than the DNet-1 model. The feature map increases in the following convolution layer till 64 feature map.

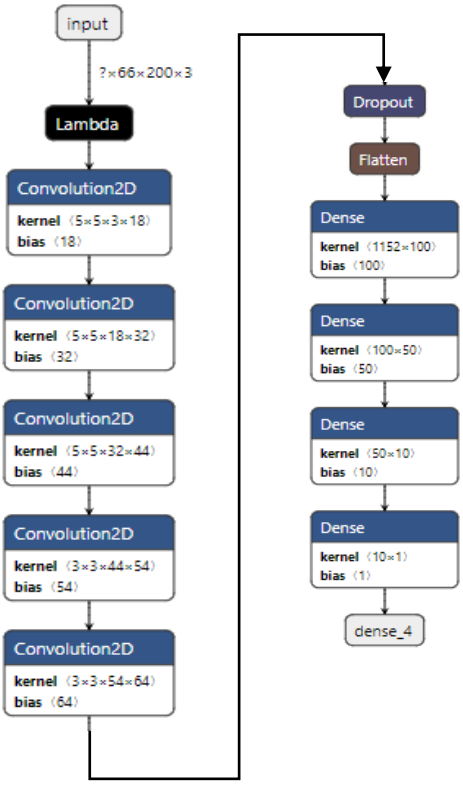

Fig. 6. Neural network design of DNet-2

The DNet-3 in fig 7 has 5 layers starting with 18 feature maps same as the DNet-1 model. The feature map in the following convolution layer increases up to 48 feature maps resulting in a very light weight CNN model.

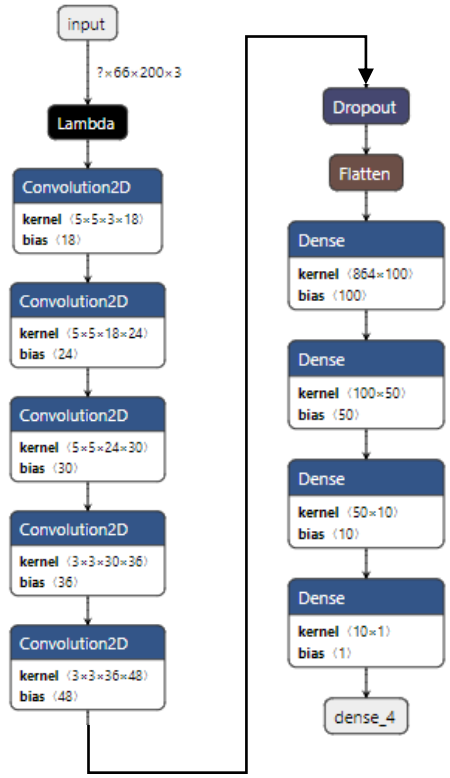

Fig. 7. Neural network design of DNet-3

The DNet-4 is designed for higher number of feature maps. Starting with 2 set of 36 feature map, which is 3 times more than the Nvidia model of 12 feature map. This model is designed for complicated road condition.

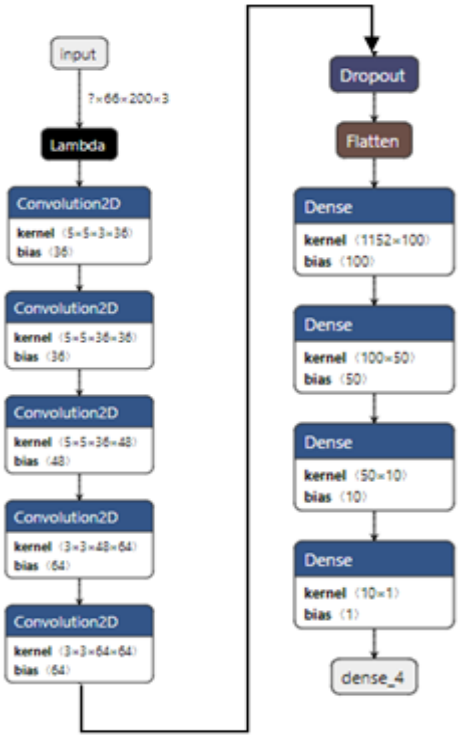

Fig. 8. Neural network design of DNet-4

The DNet-5 is similar to the DNet-4 except that the 5th convolution layer is 12.5% larger, resulting in higher precision of the output.

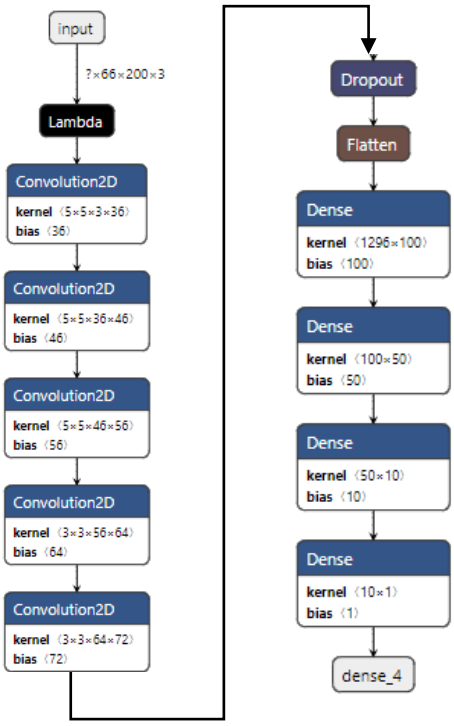

Fig. 9. Neural network design of DNet-5

The DNet-6 is heaviest among all the other 5 Models mentioned earlier. It starts with a convolution layer with 48 feature map followed by 4 other convolution network, increasing gradually and ends with a 92 feature map.

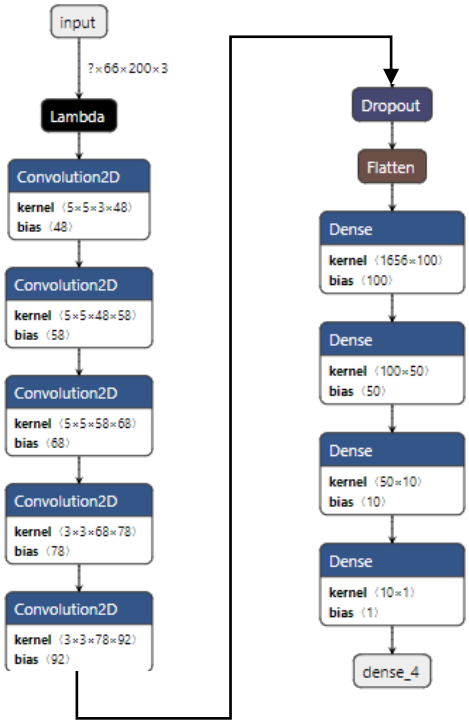

Fig. 10. Neural Network of DNet-6

VI. IMPLEMENTATION

All the CNN models are implemented in Python using Keras. A desktop PC is used for training and interfacing CNN models. This PC was running on Intel i3 a quad core processor on 3.6 GHz 4GB DDR3 RAM. This configuration is also very close to single board computers like Raspberry pi4, but on RISC architecture.

During data pre-processing, images were cropped on the top to take out the sky part and lambda layer used for normalization. Only one data set from the valley track and mountain track was used to train the entire CNN network. This ensures that the results from all the models and both the tracks are based on same conditions and directly validated the performance of the CNN models.

Loss function, mean absolute error, mean square error, acc, value loss, value mean absolute error, value mean square error and val_acc were tracked during the training of the CNN Networks. It is observed that, based on the trainable parameters the models memory space requirement varies. The list of memory requirement based on parameters trained is shown in the Table-2.

Table- II: Trainable parameters by CNN models

Model	Trainable Parameter	Size (MB)
D-Net1	2,837,671	33.20
D-Net2	224,521	2.67
D-Net3	147,649	1.76
D-Net4	263,931	3.13
D-Net5	317,773	3.70
D-Net6	455,735	5.30
Nvidia	252,219	3.02
Alexnet		328.00

All the DNet CNN models and prediction models trains using the 3 images from the camera, steering value, throttle, speed and reverse variables as shown in fig 11.

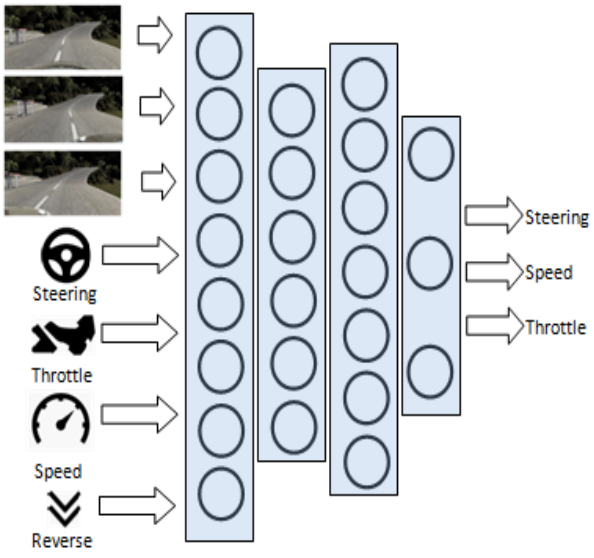

Fig. 11. CNN model for Autonomous Driving

Due to different layer configurations the trainable parameters varies among the layers as shown in Table-3.

Table- III: Trainable parameter for layers

CNN Model	Convolution				
	layer1	*Layer 2*	*Layer 3*	*Layer4*	*Layer 5*
DNet-1	912	14448	27712	36928	NA
DNet-2	1368	14432	35244	21438	31168
DNet-3	1368	10824	18030	9756	15600
DNet-4	2736	32436	43248	27712	36928
DNet-5	2736	41446	64456	32320	41544
DNet-6	3648	69658	98668	47814	64676
Nvidia	1824	21636	43248	27712	36928

VII. RESULTS AND DISCUSSION

The proposed sets of CNN model were tested using same set of data for both the tracks were used for all the models and tested on simulator. The performance and size of a CNN heavily depends on bias, weight and layer depth & type. The results were taken based on performance criteria by driving on both the tracks using the trained model. Scoring is done using the following criteria:

Negative point:

1) Vehicle touching the side shoulder or markings.
2) Exiting the track and re-entering.
3) Swaying after making turn
4) Swaying while entering the turn

Positive point:

1) Breaking before making a sharp turn.
2) Smoothly entering the curve and exiting the curve.
3) Reorienting to the centre of the road after curve.

Un-successful:

1) At least two wheels are outside the road or More than 80% of the car is off the track.
2) Car exiting or entering the track without recovery.

Table- IV: Results of Drivability test of all DNet models.

Model	Drivability			Laps Completed
	Curve In	*Curve Out*	*Breaking*	
DNet-1 Vally	3	3	3	Yes
DNet-1 Mountain	1	2	4	No
DNet-2 Vally	4	4	4	Yes
DNet-2 Mountain	3	3	3	No
DNet-3 Vally	3	3	3	Yes
DNet-3 Mountain	3	3	3	No
DNet-4 Vally	4	4	5	Yes
DNet-4 Mountain	3	3	3	Yes, Speed crash
DNet-5 Vally	5	5	5	Yes
DNet-5 Mountain	4	4	5	Yes
DNet-6 Vally	5	5	5	Yes
DNet-6 Mountain	4	4	5	Yes
Nvidia	5	5	5	Yes
Nvidia	5	5	5	Yes
Alexnet	3	3	3	Yes
Alexnet	3	3	3	Yes

In The proposed sets of CNN model, DNet-4, DNet-5 and DNet-6 were designed to handle complex Roads were able to drive the vehicle successfully. We observed the quality of driving improved as the Kernel filters were increased in one or more layers. This also increased the memory footprint of the CNN model with extra demand for processing power. The result of the drivability test is listed in Table-4.

A. Autonomous Driving Performance Evaluation

The performance evaluation of all the models was done on the Udacity self-driving car simulator. Some of the key criteria like deviation from designated track, breaking before entering the curve or on sliding track etc. are counted for the quality of drivability. Table-5 summarizes the performance of the all the CNN model with Alexnet and Nvidia model.

Table- V: Result of Drive Quality of all DNet models.

Model - Track	Laps Completed	Swaying
DNet-1 Vally	Yes	Mild
DNet-1 Mountain	No	Mild
DNet-2 Vally	Yes	No
DNet-2 Mountain	No	Yes, Less
DNet-3 Vally	Yes	Yes, Mild
DNet-3 Mountain	No	Yes, Heavy
DNet-4 Vally	Yes	Mild
DNet-4 Mountain	Yes, Speed crash	Mild
DNet-5 Vally	Yes	None
DNet-5 Mountain	Yes	None
DNet-6 Vally	Yes	None
DNet-6 Mountain	Yes	None
Nvidia Vally	Yes	Mild
Nvidia Mountain	Yes	Mild
Alexnet Vally	Yes	Average
Alexnet Mountain	Yes	Average

The model DNet-4 did not completed the track successfully at a max speed of 25 miles per hour. But when the speed of the vehicle was bought down to 18 miles per hour it was able to negotiate curves. the DNet-1, DNet-2 and Dnet-3 were able to drive through the Vally track but were unable to do so in the mountain track, which had relatively more complex and sharp curves. this proves that to negotiate road conditions which has more features, required CNN with higher number of Kernal Filters. This also results in an additional cost of higher computational hardware requiremnts.

Finally, all the CNN models were tested for the number of predictions made in a second. careful measuremetns were taken while the vehicle was autonomously driving on the track on autonomous mode to capture the frame rate and unique predictions. the results are listed in the following table.

It is clear from the table-6 that, more complex the CNN models are the rate of prediction reduces. it is also observed that larger the memory footprint of the CNN model has the more computational hardware it requires to execute and also lowers the prediction rate.

Table- VI: Performance

Model / Track	Frames/Sec	Predictions
DNet-1 / Vally	44.16	24
DNet-1 / Mountain	31.25	21
DNet-2 / Vally	45.08	18
DNet-2 / Mountain	34.05	21
DNet-3 / Vally	37.85	18
DNet-3 / Mountain	35.83	24
DNet-4 / Vally	30.06	17
DNet-4 / Mountain	21.91	20
DNet-5 / Vally	37.63	15
DNet-5 / Mountain	30.63	20
DNet-6 / Vally	28.33	16
DNet-6 / Mountain	39.88	20
Nvidia / Vally	32.35	19
Nvidia / Mountain	33.40	21
Alexnet / Vally	22.56	06
Alexnet / Mountain	19.44	04

VIII. CONCLUSION

The availability of high end CPU and GPU has paved the path to faster training of CNN networks. Through the experiments in this paper we have establish that a set of CNN will be required to drive an autonomous vehicle end-to-end instead on one for all types of road conditions. Keeping similar models but with difference in the feature extraction, an appropriate model can be used for a specific road conditions. Of course a large and a complicated CNN can handle all types of road conditions but at the cost of higher CPU requirements.

The speed of the vehicles also allows CNN to perform differently. A CNN model, in a complex road may fail if its driven at higher speed, but when slowed down, the capability to negotiate curves and complex features substantially increases.

All the DNet models mentioned are potential solution for end-to-end autonomous driving. Going forward this work will be implemented on electric golf cart, deployed in large campus as shuttle services, resort, surveillance, border security, delivery bots and warehouse robots with different types of pavements and road conditions.

Using simple CNN models with minimum memory footprint will result in low powered computers to be used in autonomous driving. Which in turn will allow usage of the roof of a vehicle as solar power generators. The carbon emissions of vehicles using gas engines can be reduced to a large extend. This also allows using smaller size computers under the dashboard instead of using the entire storage space at the back of the vehicle. Overall this solution will reduce the cost of autonomous vehicles in terms of computational hardware, sensors, power and mounting space.

REFERENCES

1 Acuna D., Ling H., Kar A., Fidler S. Efficient Interactive Annotation of Segmentation Datasets with Polygon-RNN++; Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition; Salt Lake City, UT, USA. 18–23 June 2018; pp. 859–868

2 Aggarwal C.C. Neural Networks and Deep Learning. Springer International Publishing; Cham, Switzerland: 2018.

3 Amodei D., Ananthanarayanan S., Anubhai R., Bai J., Battenberg E., Case C., Chen J. Deep speech 2: End-to-end speech recognition in English and Mandarin; Proceedings of the 33rd International Conference on Machine Learning; New York, NY, USA. 19–24 June 2016; pp. 173–182.

4 Bojarski M., Del Testa D., Dworakowski D., Firner B., Flepp B., Goyal P., Jackel L., Monfort M., Muller U., Zhang J., et al. End to end learning for self-driving cars. arXiv. 2016. 1604.07316

5 Bojarski M., Yeres P., Choromanska A., Choromanski K., Firner B., Jackel L., Muller U. Explaining how a deep neural network trained with end-to-end learning steers a car. arXiv. 2017. 1704.07911

6 Buehler M., Iagnemma K., Singh S. Springer Tracts in Advanced Robotics. Springer; Berlin/Heidelberg, Germany: 2009. The DARPA Urban Challenge: Autonomous Vehicles in City Traffic.

7 Chen Y., Wang J., Li J., Lu C., Luo Z., Xue H., Wang C. LiDAR-Video Driving Dataset: Learning Driving Policies Effectively; Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition; Salt Lake City, UT, USA. 18–23 June 2018; pp. 5870–5878.

8 Chen Z., Zhang T., Ouyang C. End-to-End Airplane Detection Using Transfer Learning in Remote Sensing Images. Remote Sens. 2018;10:139. doi: 10.3390/rs10010139

9 Chollet F. Deep Learning with Python. Manning Publications; Shelter Island, NY, USA: 2018.

10 He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; Las Vegas, NV, USA. 27–30 June 2016; pp. 770–778

11 Kocić J., Jovičić N., Drndarević V. Driver behavioral cloning using deep learning; Proceedings of the 17th International Symposium INFOTEH-JAHORINA (INFOTEH); East Sarajevo, Republika Srpska. 21–23 March 2018; pp. 1–5

12 Krizhevsky A., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neural networks. In: Pereira F., Burges C.J.C., Bottou L., Weinberger K.Q., editors. Advances in Neural Information Processing Systems. Neural Information Processing Systems Foundation, Inc.; Vancouver, BC, Canada: 2012. pp. 1097–1105.

13 LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-Based Learning Applied to Document Recognition. ; Available online: http://yann.lecun.org/exdb/publis/pdf/lecun-01a.pdf.

14 Mehta A., Adithya S., Anbumani S. Learning end-to-end autonomous driving using guided auxiliary supervision. arXiv. 2018. 1808.10393

15 Orponen P. Computational complexity of neural networks: A survey. Nord. J. Comput. 1994;1994:94–110

16 Raghu M., Poole B., Kleinberg J., Ganguli S., Sohl Dickstein J. On the expressive power of deep neural networks; Proceedings of the 34th International Conference on Machine Learning; Sydney, Australia. 6–11 August 2017; pp. 2847–2854.

17 Ravankar A., Ravankar A.A., Kobayashi Y., Hoshino Y., Peng C.-C. Path Smoothing Techniques in Robot Navigation: State-of-the-Art, Current and Future Challenges. Sensors. 2018;18:3170. doi: 10.3390/s18093170.

18 Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Berg A.C. Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. 2015;115:211–252. doi: 10.1007/s11263-015-0816-y.

19 Shalev-Shwartz S., Shammah S., Shashua A. Safe, multi-agent, reinforcement learning for autonomous driving. arXiv. 2016. 1610.03295

20 Silver D., Hubert T., Schrittwieser J., Antonoglou I., Lai M., Guez A., Lillicrap T. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv. 2017. 1712.01815.

21 Silver D., Schrittwieser J., Simonyan K., Antonoglou J., Huang A., Guez A., Hubert T., Baker L., Lai M., Bolton A. Mastering the game of go without human knowledge. Nature. 2017;550:354. doi: 10.1038/nature24270.

22 Simard D., Steinkraus P.Y., Platt J.C. Best practices for convolutional neural networks applied to visual document analysis; Proceedings of the Seventh International Conference on Document Analysis and Recognition; Edinburgh, UK. 6 August 2003; pp. 958–963.

23 Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv. 2004. 1409.1556

24 Sung Y., Jin Y., Kwak J., Lee S.-G., Cho K. Advanced Camera Image Cropping Approach for CNN-Based End-to-End Controls on Sustainable Computing. Sustainability. 2018;10:816. doi: 10.3390/su10030816.

25 Sutton R.S., Barto A.G. Reinforcement Learning. 2nd ed. The MIT Press; Cambridge, MA, USA: 2018. p. 552.

26 Sze V., Chen Y., Yang T., Emer J.S. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. Proc. IEEE. 2017;105:2295–2329. doi: 10.1109/JPROC.2017.2761740.

27 Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A. Going deeper with convolutions; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; Boston, MA, USA. 7–12 June 2015; pp. 1–9.

28 Teti M., Barenholtz E., Martin S., Hahn W. A Systematic Comparison of Deep Learning Architectures in an Autonomous Vehicle. arXiv. 2018. 1803.09386

29 Thrun S., Montemerlo M., Dahlkamp H., Stavens D., Aron A., Diebel J., Fong P., Gale J., Halpenny M., Hoffmann G., et al. Stanley: The Robot That Won the DARPA Grand Challenge. J. Field Robot. 2006;23:661–692. doi: 10.1002/rob.20147

30 Udacity, Inc. Self-Driving Car Simulator. [(accessed on 5 November 2018)]; Available online: https://github.com/udacity/self-driving-car-sim.

31 Visin F., Kastner K., Cho K., Matteucci M., Courville A., Bengio Y. Renet: A recurrent neural network based alternative to convolutional networks. arXiv. 2015. 1505.00393

32 Wang T.C., Liu M.Y., Zhu J.Y., Liu G., Tao A., Kautz J., Catanzaro B. Video-to-video synthesis; Proceedings of the 32nd International Conference on Neural Information Processing Systems; Montreal, QC, Canada. 3–8 December 2018; pp. 1152–1164.

33 Xu D., Jain A., Anguelov D. PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation; Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition; Salt Lake City, UT, USA. 18–23 June 2018; pp. 244–253.

34 Yao Y., Wang H., Li S., Liu Z., Gui G., Dan Y., Hu J. End-To-End Convolutional Neural Network Model for Gear Fault Diagnosis Based on Sound Signals. Appl. Sci. 2018;8:1584.

35 Zoph B., Vasudevan V., Shlens J., Le Q.V. Learning Transferable Architectures for Scalable Image Recognition; Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition; Salt Lake City, UT, USA. 18–23 June 2018; pp. 8697–8710.

AUTHORS PROFILE

Bishwajit Pal, MCA from Anna University, Chennai is working as research scholar in Dayanada Sagaar college of engineering, Bangalore. He is currently researching in autonomous vehicle and pursuing PhD from VTU, Belagavi.

Dr. Samitha Khaiyum, PhD Currently working as Associate Professor and HOD in department of MCA in Dayananda Sagar college of engineering, Bangalore. Holder of Master of Computer Applications, followed by a M. Phil degree in Computer Science.