End-to-end Learning Approach for Autonomous Driving: A Convolutional Neural Network Model

Yaqin Wang, Dongfang Liu, Hyewon Jeon, Zhiwei Chu and Eric T. Matson
Department of Computer and Information Technology, Purdue University, U.S.A.

Keywords: Autonomous Driving, AI, Convolutional Neural Network, End-to-end Approach.

Abstract: End-to-end approach is one of the frequently used approaches for the autonomous driving system. In this study, we adopt the end-to-end approach because this approach has been approved to lead to a distinguished performance with a simpler system. We build a convolutional neural network (CNN) to map raw pixels from cameras of three different angles and to generate steering commands to drive a car in the Udacity simulator. Our proposed model has a promising result, which is more accurate and has lower loss rate comparing to previous models.

1 INTRODUCTION

In 1989, (Pomerleua, 1989) designed a simple structured neural network with three fully-connected layers to steer a car on the CMU campus. This was an initial research to demonstrate that machine learning theory can be implemented to generate commands to guide a car. However, until now, no autonomous cars have been designed which meet the full autonomous driving (Rödel et al., 2014). As the most important component in autonomous driving, different models based on different approaches have been used to generate appropriate commands to steer the car. Progress of autonomous driving is due to the optimization of the model design.

Currently, the mediated perception approach and end-to-end approach are two frequently used approaches for autonomous driving system. Mediated perception approach is widely used in the car industry which requires a large assistive system (lidar, radar, sensor, and central computing systems) for detection, prediction, and decision making (Chen et al., 2015). On the contrary, the end-to-end approach is a much lighter system and utilizes a distinct theory to steer a car (Bojarski et al., 2016)(Xu et al., 2017)(Codevilla et al., 2018)(Pfeiffer et al., 2017). This approach focuses on building an artificial intelligence model which emulates humans driving. For human, driving on the road, steering commands are constantly made based on the instant processing of images (Bojarski et al., 2016)(Xu et al., 2017)(Codevilla et al., 2018)(Pfeiffer et al., 2017). For an end-to-end learning approach, the model is normally trained by learning the humans actions on the road (Bojarski et al., 2016)(Xu et al., 2017)(Codevilla et al., 2018)(Pfeiffer et al., 2017). Here, one end is reading the input image and another end is to respond accordingly to road conditions. Using end-to-end learning approach is easy to maneuver and cost-effective compared to mediated perception approach.

Prior research has explored the end-to-end learning approach for autonomous driving. A pioneering study is from (Muller et al., 2006) who utilized an end-to-end learning system to design an autonomous car. (Muller et al., 2006) used a six-layer convolutional neural network for image processing. This CNN model can process the raw pixels and command steering angels based on the input image. In addition, (Chen et al., 2015) employed the Caffe framework to develop a CNN model to process the dataset from TORCS (The Open Racing Car Simulator). The model has a higher computing speed compared to the model from (Muller et al., 2006) when making commands for steering actions. Another distinguished end-to-end learning model is from (Su et al., 2017). To improve the generalization of their model, (Su et al., 2017) employed a large dataset for training, which included different road conditions. The outcomes of this research obtained a high accuracy pertaining to steering angels compared to its previous counterparts. CNN models have also been widely used in object detection and classification in autonomous driving (Ren et al., 2015)(Girshick et al., 2014).

Wang, Y., Liu, D., Jeon, H., Chu, Z. and Matson, E.
End-to-end Learning Approach for Autonomous Driving: A Convolutional Neural Network Model.
DOI: 10.5220/0007575908330839
In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019)*, pages 833-839
ISBN: 978-989-758-350-6

Aside from the seminal studies discussed above, a widely recognized research is from (Bojarski et al., 2016). The researchers proposed an innovative CNN model based on the end-to-end learning approach. The strength of this research is that the CNN model has a much simpler structure compared to the previous studies. It only has nine layers of networks with five convolutional layers and four fully-connected layers. Hence, the computing speed for this model is faster which can process 30 frame per second. In addition, this model does not require a large dataset for training. The total training dataset for this model only included 100 hours of video clips of different road conditions.

Although the previous research has obtained various achievements for end-to-end learning approach, we identified a number of problems in the process of survey studies of previous research. First, the majority of the previous end-to-end learning approach is based on the CNN model for image processing. CNN is an effective neural framework to process image, but it is also subject to the problems of overfitting and vanishing gradient in the training phase (Srivastava et al., 2014)(Ma et al., 2017)(Klein et al., 2018)(Su et al., 2017). Second, a desirable outcome of the end-to-end learning approach is that the autonomous car can emulate the humans driving and keep on the road with limited human interventions. However, most autonomous cars based on the end-to-end learning approach need a large number of human interventions in previous research (Pomerleua, 1989)(Xu et al., 2017)(Muller et al., 2006)(Chen et al., 2015).

The contributions of this study are in three folds. First, we design a CNN model based on end-to-end learning approaches which has better generalization than that of (Bojarski et al., 2016). We will discuss more details in the proposed model section pertaining to how we improve the noise robustness of our model. This goal is evaluated by testing the generalization of our proposed model. We evaluate the loss rate and accuracy of our model during the training and address the potential problem of overfitting and vanish gradient. Second, our proposed CNN model based on end-to-end learning approach can steer a car longer than the reported time from (Bojarski et al., 2016). We compare the results and report them in the result and discussion section.

The rest of this report is organized as follows: In section 2, the required materials and the proposed methods will be introduced; In section 3, we will present the results and discussions about our project; In section 4, we will conclude the progress of our ongoing project and offer some insights for the future work.

2 METHODOLOGY

2.1 Simulator

There is a large array of autonomous driving simulators available on GitHub, as seen in Table 1. The selection criteria for our project are that: 1. The simulator does not have a high requirement for graphics device; 2. The simulator can easily collect image data and generate CSV file which includes labeling information for the collected images; 3. The simulator can be installed and operated on Mac OS X; 4. The simulator has an autonomous driving mode to test the autonomous model.

We finally decide to use Udacitys Self-Driving simulator, because it fits well with our selection standards. This simulator platform has high graphics requirements and it can be easily installed on Mac OS. It has training mode and autonomous mode. In training mode, the simulator can record images and generate a CSV file simultaneously when we drive a car in its environment. In autonomous mode, we can utilize the designed model to programmatically navigate the car in the environment.

2.2 Data Collection

The data collection is iterative and has two phases. In the first phase, we focused on practicing the driving skills. After the Udacity Self-Driving simulator is installed, we started to use training mode to practice the driving skills. We constantly evaluated our performance of driving in the simulator and improve the driving accuracy pertaining to keeping the car in the middle of the lane. The first phase lasted about one month before we started collecting data for training.

In the second phase, we made strict rubrics for the collection process which determine the usability of the collected data. The usable data only includes images that the car is running inside of two-lane marks and the car does not suffer from drastic steering angle changes. Data which fails to meet the rubrics is not included in the final dataset for training. Figure 1 demonstrates the real data collection process, which simultaneously recorded the left, center, and right picture of a running car in training mode. A CSV file is also automatically generated while we run the car in Udacity. A sample CSV file includes information of labels of center, left, and right pictures as well as a cars current steering angle, accelerating, reserve, and speed, as seen in Figure 2. The second phase of data collection lasted around one month, we run a car for approximately sixty hours on Udacity and we finally decided to include 83,424 pictures for model training.

Table 1: Summary of the autonomous driving simulators.

Name	Graphic requirement	Data collection	OS requirements (Mainly support)	Autonomous driving mode
Udacity	low	easy	Window, Mac OS, Linux	Yes
TORCS	medium	complex	Linux	Yes
Euro Truck Simulator 2	high	complex	Window, Mac OS, Linux	Yes
Gazebo	medium	complex	Linux	Yes
CARLA	high	complex	Window, Linux	Yes

a b c

Figure 1: Left (a), center (b), and right (c) camera of a running car.

2.3 Method

In this study, we build a convolutional neural network (CNN) to map raw pixels from cameras of three different angles which generate steering and acceleration commands to drive a car in the Udacity simulator. The overall framework for this research is demonstrated in Figure 3. An end-to-end approach is employed because this approach has been approved to lead to a distinguished performance with a simpler system (Bojarski et al., 2016). A distinguished performance can be achieved because the end-to-end learning system is trying to emulate humans action, instead of learning human-selected criteria, such as lane detection or lane keeping. In addition, a simpler solution is defined by the fact that an end-to-end system only requires a small number of datasets for training (Bojarski et al., 2016). Theoretically, if the training dataset only includes the actions we want the model to learn, an end-to-end learning system is able to achieve a productive outcome with less effort for training. With the road images associated with steering angles and acceleration as the training inputs, our system automatically learns internal representations of the imperative processing steps pertaining to making proper commands for driving. We do not need to explicitly train our model to detect road features, for instance, the outline of roads or background objects. Our model is able to make command to change steering angles and acceleration when encountering similar road condition.

2.4 The Proposed Model

In this study, we rely on the framework from (Bojarski et al., 2016) to build our model. The study from (Bojarski et al., 2016) is the widely recognized end-to-end learning approach which designed a CNN model for autonomous driving in recent years. However, like all CNN based model, (Bojarski et al., 2016) is subject to the problem of overfitting and vanishing gradient (Srivastava et al., 2014)(Ma et al., 2017)(Klein et al., 2018)(Su et al., 2017). Also, the loss rate and accuracy for (Bojarski et al., 2016) can be further improved.

To address the issue of overfitting and eventually improve the loss rate and accuracy, we purposefully add dropout layers to our model. Dropout is a powerful regularization method for Multi-Layer Perceptrons (MLPs) trained by Stochastic Gradient Descent (SGD) (Szegedy et al., 2016)(Clevert et al., 2015)(Gal and Ghahramani, 2016). During the forward propagation step of the SGD, the dropout layers consist in setting zero to a random subset of the hidden activation. Accordingly, the backward propagation in the SGD is also modified. By this mean, dropout is able to prevent the co-adaptation of learned hidden features which is a source of overfitting. Eventually, the network becomes less sensitive to the specific weights of neurons and has a better capacity of generalization.

Figure 3 demonstrated the overview of the proposed model in the study. For the first five layers, we use convolutional layers to capture the key figures of the input data. The part of the structure is derived from (Bojarski et al., 2016), which has been approved effective to capture road features. After the first five layers of iterative processing, the feature images for input data is well-learned by the system. The output of convolutional layers is connected to four fully-connected layers. At this stage, we try to add a dropout layer between fully-connected layers because a number of research (Szegedy et al., 2015)(Szegedy et al., 2016)(Clevert et al., 2015) suggested that

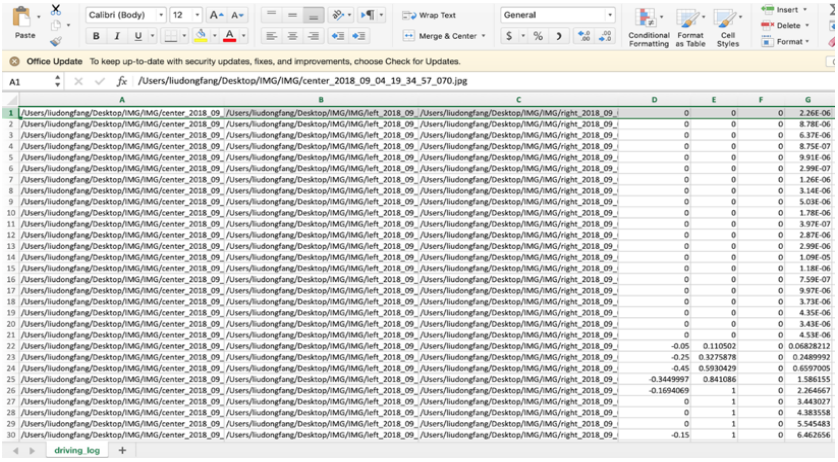

Figure 2: A Sample CSV file.

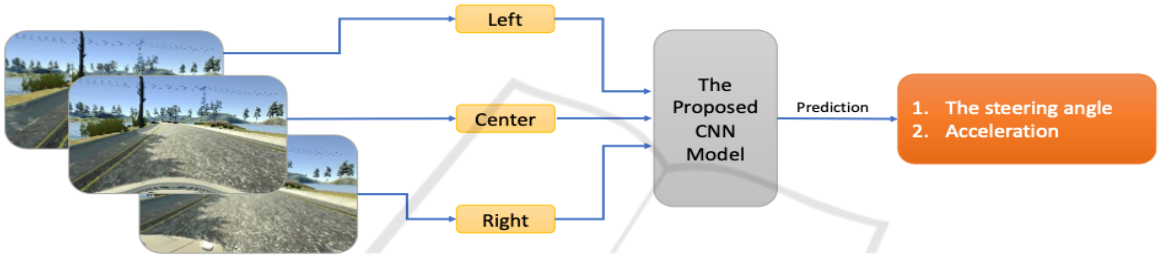

Figure 3: The Overall Framework.

dropout layer should be used between fully-connected layers and not in convolutional layers. However, there is a debate pertaining to where a dropout layer should be added (Chen et al., 2018)(Clevert et al., 2015)(Gal and Ghahramani, 2016)(Bluche et al., 2015). One widely accepted solution is to add dropout layers in different locations in the neural network and test its overall performance (Chen et al., 2018)(Clevert et al., 2015)(Gal and Ghahramani, 2016)(Bluche et al., 2015). Following this rule, we purposefully add the dropout layer in different locations between the fully-connected layers and test the performance for the entire network. In addition, considering the fact that the dataset is relatively small, data augment function is also carefully designed in order to create a new data and prevent overfitting. The augment function we design include flipping (both vertically and horizontally), translating (moving along either of the x, y axis), cropping, and rotating.

In addition, when designing our model, we carefully compare four commonly used activation functions such as ELU, LReLU, ReLU, and SReLU, in order to address the problem of vanishing gradient. After the comparison across the four activation functions, we finally use ELU as our activation function for our model for two main reasons. First, when processing positive values, all of the four activation functions have good performance for solving the problem of vanish gradient. However, if the processed values are negative, RelU will cause bias to the next neural network layer because the output of RelU is zero. The bias will be added cross later layers and amplify the effect of bias weight. Second, although the outputs of LReLU and PReLU is not zero when processing negative inputs, the two activation functions cannot maintain noise robustness when the input value is small. On the contrast, ELU has soft saturation characteristics, so when the input is a small negative value, it improves the noise robustness and has a desirable performance to address the problem of vanish gradient (Szegedy et al., 2015)(Szegedy et al., 2016).

3 RESULTS AND DISCUSSIONS

3.1 Training Results

In the experiment, we utilize 32 as our batch size, 20,000 as our maximum training steps, and 15 as our epochs for each training circle. The trained model is automatically saved after the training. We duplicate the model named NVIDIA and use the same

Table 2: The Results of Validation Loss Rate and Accuracy for Different Models. NVIDIA model is the model from (Bojarski et al., 2016); Model 1 is the proposed model with dropout layer after the first fully-connected layer; Model 2 is the proposed model with dropout layer after the second fully-connected layer; Model 3 is the proposed model with a dropout layer after the third fully-connected layer.

Epoch		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NVIDIA	Loss rate	0.025	0.025	0.025	0.024	0.022	0.022	0.023	0.022	0.024	0.024	0.025	0.023	0.023	0.023	0.023
	Accuracy	0.776	0.78	0.779	0.779	0.784	0.776	0.778	0.783	0.779	0.784	0.783	0.776	0.782	0.776	0.779
Model 1	Loss rate	0.026	0.026	0.026	0.026	0.027	0.028	0.029	0.028	0.028	0.028	0.029	0.028	0.027	0.027	0.027
	Accuracy	0.76	0.76	0.764	0.765	0.765	0.774	0.775	0.76	0.76	0.77	0.75	0.76	0.76	0.76	0.76
Model 2	Loss rate	0.027	0.027	0.028	0.027	0.026	0.025	0.026	0.026	0.027	0.026	0.027	0.026	0.026	0.026	0.026
	Accuracy	0.766	0.762	0.75	0.74	0.765	0.775	0.775	0.774	0.773	0.776	0.776	0.774	0.774	0.774	0.774
Model 3	Loss rate	0.025	0.025	0.025	0.024	0.022	0.022	0.022	0.022	0.022	0.021	0.021	0.021	0.021	0.021	0.022
	Accuracy	0.772	0.778	0.779	0.776	0.781	0.783	0.773	0.778	0.776	0.777	0.775	0.78	0.778	0.78	0.781

dataset to train our models and the NVIDIA model. We also add the dropout layer at different location across fully-connected layers. The results for training dataset are demonstrated in Figure 6. Based on the results, we can see that proposed models with dropout layer have better training outcomes compared to NVIDIA model. The final loss rate for models with the dropout layer is smaller than that of (Bojarski et al., 2016), as seen in Table 2. Also, the final accuracy for models with dropout layer is also higher than that of (Bojarski et al., 2016), as seen in Table 2. The proposed model 3 has best performance pertaining to having lowest loss rate and highest accuracy in training dataset among all presented models. Model 3 is the proposed model with dropout layer after the third fully-connected layer.

Next, we further compare the proposed model 3 with the NVIDIA model in terms of the changing of the loss rate in training and validation process, as seen in Figure 5. The loss rate for model 3 in both training and validation continuously decreases. In contrast, the validation loss rate for the NVIDIA model starts to increase at epoch 12 which is an indicator for overfitting. Based on the results, adding dropout layer effectively addresses the issue of overfitting and improves the noise robustness of the original model.

The main contribution of this study is to build a model that could surpass the performances of previous studies.

3.2 Generalization and Lane Keeping Time

We purposefully evaluate the generalization of our model. The Udacity simulator has two road tracks with totally different road conditions. One road track is a lake road and the other is a mountain road. Lake road is very flat and has no steep curves. On the contrary, the mountain road has challenging road conditions with a lot of steep curves. We use the mountain road dataset to train our model and test the model to

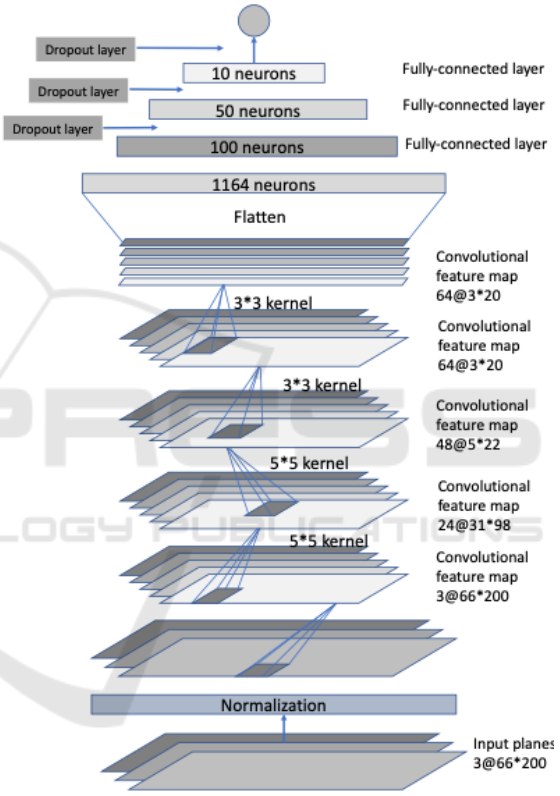

Figure 4: The Comparison of ELU, LReLU, ReLU, and SReLU.

run a car in the lake road track. Since a car running on a mountain road needs frequent turns of steering angle, if the trained model simply clones the behaviors in the dataset, the generalization of the model could be compromised because running on a flat road does not need a large degree of steering angle. However, the result from our experiment demonstrates that our model has a desirable performance and it runs a car on lake track.

In addition, we test the lane keeping time for our model. We test our model 30 times on the lake road. When running on the lake road track, our model can

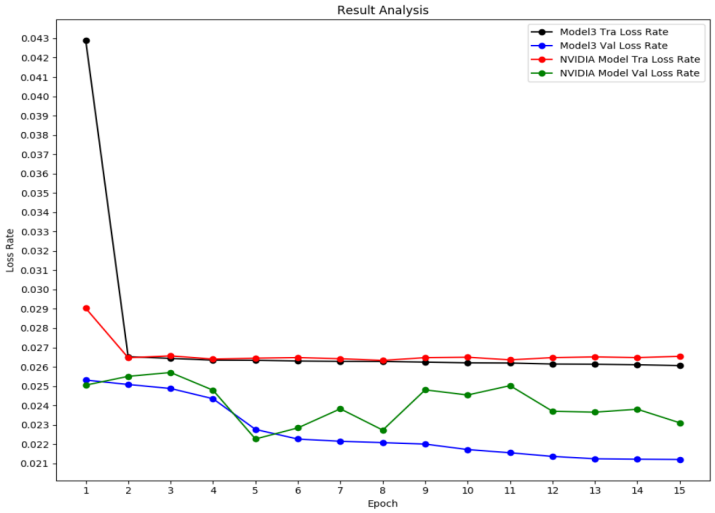

Figure 5: The Overall Framework.

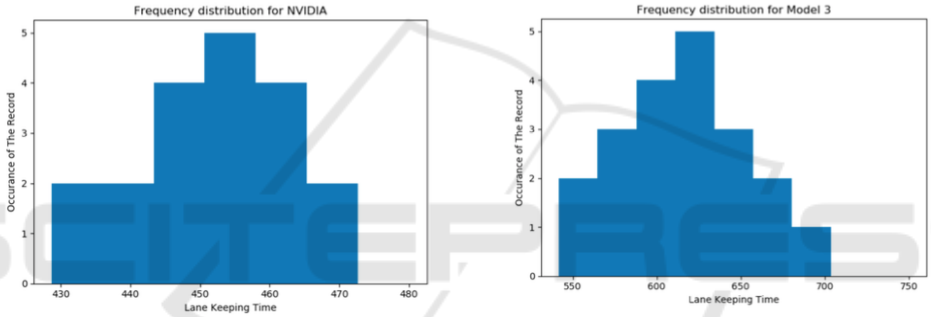

Figure 6: Frequency Distribution of the NVIDIA Model and Model 3.

steer the car on the road and keep the car on the road for more than 617.3 seconds average, as seen in Table 3. In addition, we use the NVIDIA model to steer a car on the lake road track. The NVIDIA model can keep the car in the road without running off the road for 453.7 seconds average, as seen in Table 3. Since we test the two model by using the same car and same road track, we use paired t-test to evaluate the performance difference between two models. We first check the assumption of the paired t-test and ensure that the collected data is normal distributioned, as seen in Figure 6. Then we run the paired t-test for the data. The result is significant ($p<0.0001$) and Model 3 with the dropout layer is better than NVIDIA model without the dropout layer because of longer lane keeping time, as seen in Table 4.

4 CONCLUSION

In this study, we design a CNN model based on an end-to-end learning approach and address the problem of overfitting and vanish gradient. Our proposed model outperforms the previous study which we rely on as our model framework. In addition, we evaluate the generalization of our proposed model. We use a specific road condition to train our model and test it on a different road condition. Finally, we record how much time the car can remain in the lane without human intervention and the result indicates that the car can stay on the road longer than the NVIDIA model. Limitations for the project are a few. First, the action for the end-to-end learning approach is defined as steering angles and change of speed rate in our study. However, actions for real driving scenario is more complicated and should include reverse, brake, and more. In addition, we did not consider obstacles, such as pedestrians and other cars, in our research. Future research can add object detection functions in the model to simulate real road conditions.

Table 3: Lane Keep Time: The report time is in second and rounded up for keeping two decimals.

Test time	Model 3	NVIDIA Model
1	582.3	450.6
2	624.5	472.4
3	541.7	442.3
4	693.1	443.7
5	599.2	458.3
6	612.4	451.4
7	643.8	454.2
8	605.9	451.8
9	564.1	484.9
10	642.4	438.4
11	656.4	460.4
12	604.5	464.1
13	587.1	434.8
14	627.2	469.9
15	623.2	457.5
16	674.4	428.7
17	675.7	449.4
18	611.8	456.3
19	604.2	461.6
20	572.8	443.4
Average	617.3	453.7

Table 4: Paired t-test for Model 3 and NVIDIA Model.

Variable	Mean	Statistic	p-value
Model 3	617.3	16.63	8.84e-11<0.0001
NVIDIA Model	453.7		

REFERENCES

Bluche, T., Kermorvant, C., and Louradour, J. (2015). Where to apply dropout in recurrent neural networks for handwriting recognition? In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 681–685. IEEE.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv, Cornell University:1 604.07316*.

Chen, C., Seff, A., Kornhauser, A., and Xiao, J. (2015). Deepdriving: Learning affordance for direct perception in autonomous driving. pages 2722 – 2730.

Chen, T., Lu, S., and Fan, J. (2018). S-cnn: Subcategory-aware convolutional networks for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2522–2528.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv, Cornell University: 1511.07289*.

Codevilla, F., Miiller, M., López, A., Koltun, V., and Dosovitskiy, A. (2018). End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE.

Gal, Y. and Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Klein, A., Christiansen, E., Murphy, K., and Hutter, F. (2018). Towards reproducible neural architecture and hyperparameter search.

Ma, C., Zhu, Z., Ye, J., Yang, J., Pei, J., Xu, S., Zhou, R., Yu, C., Mo, F., Wen, B., et al. (2017). Deeprt: deep learning for peptide retention time prediction in proteomics. *arXiv, Cornell University: 1705.05368*.

Muller, U., Ben, J., Cosatto, E., Flepp, B., and Cun, Y. L. (2006). Off-road obstacle avoidance through end-to-end learning. In *Advances in neural information processing systems*, pages 739–746.

Pfeiffer, M., Schaeuble, M., Nieto, J., Siegwart, R., and Cadena, C. (2017). From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *2017 ieee international conference on robotics and automation (icra)*, pages 1527–1533. IEEE.

Pomerleua, D. (1989). Alvinn: an autonomous land vehicle in a neural network. *Advances in neural information processing systems*, pages 305–313.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Rödel, C., Stadler, S., Meschtscherjakov, A., and Tscheligi, M. (2014). Towards autonomous cars: the effect of autonomy levels on acceptance and user experience. *Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 1–8.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Su, W., Chen, L., Wu, M., Zhou, M., Liu, Z., and Cao, W. (2017). Nesterov accelerated gradient descent-based convolution neural network with dropout for facial expression recognition. In *Control Conference (ASCC), 2017 11th Asian*, pages 1063–1068. IEEE.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Xu, H., Gao, Y., Yu, F., and Darrell, T. (2017). End-to-end learning of driving models from large-scale video datasets. *arXiv, Cornell University*.