Learning On-Road Visual Control for Self-Driving Cars with Auxiliary Tasks

Yilun Chen Katharina Muelling John M. Dolan
Carnegie Mellon University
yiluncl@andrew.cmu.edu, kmuelling@nrec.ri.cmu.edu, jmd@cs.cmu.edu

Praveen Palanisamy Priyantha Mudalige
General Motors
praveen.palanisamy@gm.com, priyantha.mudalige@gm.com

Abstract

Safe and robust on-road navigation system is a crucial component for achieving fully automated vehicles. [3] recently proposed an end-to-end algorithm that can directly learn steering commands from raw pixels of a front camera by using one convolutional neural network. In this paper, we leverage auxiliary information aside from raw images and design a novel network structure to help boost the driving performance, while maintaining the advantage of minimal training data and end-to-end training method. First, we incorporate human common sense into vehicle navigation by transferring features from image recognition tasks. Second, we apply image semantic segmentation as an auxiliary task for navigation. Third, we consider temporal information by introducing an LSTM module to the network. Finally, we combine vehicle kinematics with a sensor fusion step. We show our method can outperform the state-of-the-art visual navigation method both in the Udacity simulation environment and on the real-world comma.ai dataset. Our method also has faster training speed and more stable driving behavior compared to previous methods.

1. Introduction

Perception and control have long been two independent key challenges for learning a driving model in the autonomous driving industry. However, recent advances in deep learning have introduced end-to-end learning as a new method for learning action policies for self-driving cars. Unlike traditional approaches [11] that divide the system into two separate perception and control parts which contain tasks like lane detection, path planning and control logic, end-to-end approaches often directly learn the mapping from raw pixels to vehicle actuation. Recent demonstrations have shown some successful examples of training systems end-to-end to perform simple tasks like lane-keeping [4] or obstacle avoidance [7].

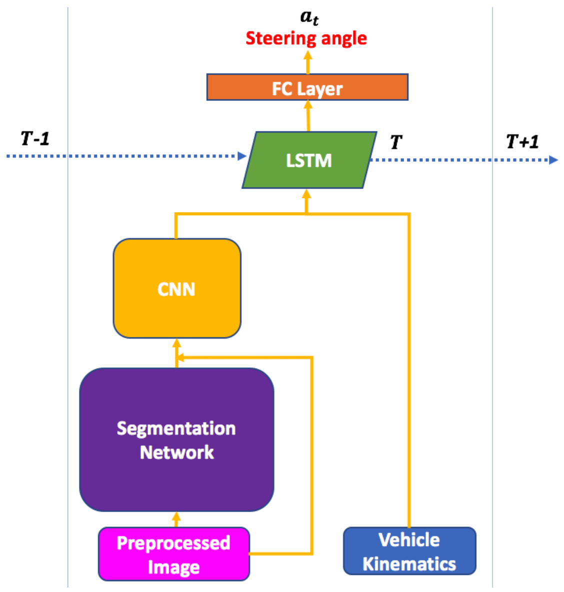

Figure 1. The overall network structure of our proposed end-to-end learning architecture. Compared to the original network structure proposed by [3], our network has additional modules: a segmentation network, an LSTM, and vehicle kinematic input.

The end-to-end methods have the advantage of direct optimization without manually defined rules which result in better performance and less human engineering efforts. However, current end-to-end models use one deep convolutional neural network to map from perception to control [3]. This straightforward representation simplifies the modeling, but has a number of drawbacks. First, it loses a lot of detailed dynamic information for the specific driving task. Speed and map information and vehicle mechanics are all important for the driving task apart from camera images. Second, it lacks human prior knowledge in the network. People obey a lot of basic assumptions like traffic rules when driving, whereas the network is expected to learn from scratch. Third, it only considers current perception information for decision making. In contrast, people memorize history information and process it along with current perception information to drive.

To address these problems, we present an improved end-to-end learning approach with auxiliary tasks. We still target the lane following driving task, but improve the system performance in terms of training efficiency and accuracy. The approach can converge in less time with higher performance.

The new approach is built upon a convolutional neural network, with several variations in the network and trainng procedure. Our proposed network structure is shown in Figure 1. We get raw RGB images from the camera mounted in the front of the autonomous vehicle. First, we do some data preprocessing to increase the diversity of data. Then the preprocessed data are fed into a pre-trained segmentation network. From the segmentation network, we get the segmentation map of the particular image. Then a traditional convolutional network is used to extract necessary features for learning the proper control parameters. After the convolutional layer, we have an LSTM network to incorporate temporal dynamics of the system. The LSTM network here is designed to remember past states of the road configuration. Then finally, we add additional vehicle information, to concatenate the learned features with necessary vehicle kinematics. This concatenated vector is then fed into a fully connected layer to finally learn the continuous steering angle.

Our system differs from the original network structure in four respects: (1) the overall system takes advantage of additional information by first obtaining a segmentation map instead of directly using the implicit raw image; (2) our learning system transfers knowledge from other tasks and hence speeds up training; (3) we consider the temporal information by adding a recurrent module into the network; (4) we use vehicle kinematics information in addition to image input. The final network improves the baseline pure convolutional network by incorporating more information, both from human knowledge and history states.

2. Related Work

With the rapid growth of deep learning technology, end-to-end learning machines have appeared as a common solution to solve complex practical robotic systems. In robotics literature, an end-to-end process refers to a robot or an agent consisting of only one network without modularization from sensors to motors.

The definition of end-to-end learning is in comparison with traditional methods that divide a task into several modules. Traditional methods would break the system into several fundamental building blocks, solve each one separately, and then optimize the pipeline jointly. However, end-to-end learning machines enable a direct mapping from raw sensor input to the desired output only using one network. In an end-to-end system, all parameters are trained at one time jointly, unlike the step-by-step process of traditional methods.

End-to-end learning has shown great success in many fields. Mnih [6] has surpassed human-level performance in Atari game-playing by training an end-to-end deep reinforcement learning agent. Sergey [5] trained an end-to-end policy for a PR2 robot to learn grasp from raw visual input. Deep Speech [1] replaces the entire speech recognition pipeline using hand-designed features with neural networks and results in a more robust model in environments with different noise, accents, and languages.

End-to-end learning has also emerged as a new approach for self-driving cars [3]. In order to learn driving on the road, a traditional approach would involve object and lane detection in an image, path planning to calculate trajectory and PID control for final control output. However, an end-to-end approach would learn a single network to map from sensor perception to steering angles. Despite the diminished effort in engineering hand-crafted features, the end-to-end approach gets better performance than traditional methods, although it relies on large amounts of data.

3. Method

In this section, we introduce our proposed methods for constructing the network in detail. The methods we discuss here include using auxiliary segmentation, transferring from existing tasks, utilizing temporal information and incorporating vehicle information.

3.1. Auxiliary Segmentation

Image segmentation has been widely researched for decades. Image segmentation is the process of partitioning an image into multiple sets of pixels to simplify its representation and derive something more meaningful and easier to analyze. Image segmentation is often used to recognize and locate certain categories of objects by assigning a label to each pixel.

In autonomous driving, image segmentation is often performed to understand the surrounding environment of the ego vehicle, for example, to recognize surrounding vehicles, pedestrians, road boundaries, buildings, etc. This information is crucial for determining the next actions. However, the result of the segmentation process is often used ambiguously and hard to apply directly to drive the car.

Here, we incorporate image segmentation directly into the task of learning to control a car in an end-to-end fashion. We believe the learned segmentation map contains auxiliary information for controlling the car's behavior. So we add a segmentation map as an extra input to the system. The auxiliary information will explicitly tell the network, for example, where the road boundary is and where the surrounding vehicles on the road are. This will decrease the difficulty of learning everything implicitly from the original raw image.

We integrate the image segmentation into our architecture by using the segmentation network proposed by [2].

3.2. Transferring from Existing Tasks

CNN has been applied recently to a significant number of practical and essential tasks. The training results have shown that this approach is very successful in jobs like object recognition. This motivates us to leverage the power of a pre-trained network and apply the concept of transfer learning [8]. Currently, numerous famous network structures in the literature have been proven to be powerful. The most popular task is to learn object recognition on the Imagenet dataset that contains 1.2 million images of approximately 1000 different classes. The resulting trained model can generalize a generic set of features, and recognize a large variety of objects with high accuracy. The intermediate features learned are found to have universal expressiveness across multiple domains. We hence want to use this characteristic and transfer the pre-trained network from a vast field to the specific task of learning the driving policy.

In this paper, we compare three models: Resnet, the Vgg16 network and our baseline CNN for the CNN module in our overall network. The Resnet and Vgg16 network are pre-trained on Imagenet. For feature extraction purposes, we only use the convolutional layers. The detailed configuration of our CNN network is illustrated in Section 3.3. The three networks differ in depth and number of total parameters. To make the comparison fair and even, we froze some of the parameters in Resnet and Vgg16 net, so that only parts of the Resnet and Vgg16 net are tunable. This makes the total number of adjustable parameters approximately the same for each model tested.

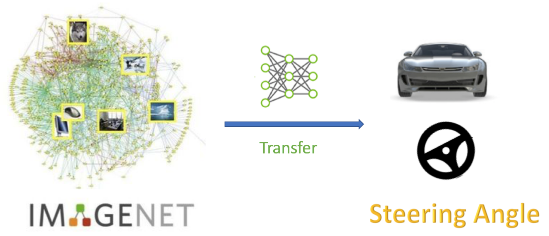

Figure 2. Illustration of transfer learning from object recognition in Imagenet to learning steering angle for the self-driving car.

3.3. Temporal Information

Decision making for an autonomous vehicle is not an independent choice at every time step. A human would consider past environment information and previous actions taken and then make a consistent decision of driving behavior. This requires our system not only to be based on the current state but also incorporate past states. So, apart from the convolutional neural network to capture the spatial information, we introduce recurrent modules into our network architecture.

After getting the spatial features from the CNN layer, we added an LSTM layer to pass in previous information of the environment. The LSTM processes a feature vector v from the CNN in a sliding window of size w. This means the steering action prediction result is dependent on w past input observations $X_{t-w+1} - X_t$. By changing the parameter of w, we can alter how long the system considers to make the decision. Small w leads to shorter-term memory, so it has faster reaction time but is prone to sudden sensor failure. Larger w, on the contrary, will lead to a much smoother and stable behavior. The problem of larger w is that it requires longer training and test time for choosing actions.

With the visual states at each time step, the LSTM fuses all past states and current state into a single state. So the state here is complete, and the autonomous car is theoretically given all the historical information to make the necessary action choice.

3.4. Additional Vehicle Information

We further hypothesize that visual input alone is not good enough to make a good steering angle decision. The vehicle's behavior is better estimated by adding the vehicle's kinematic information. The kinematic information ensures that the car does not execute a driving behavior that is against some physical rules.

It can be speculated that making a U-turn at 10mph and 30 mph are different regarding turning angle and the strategy used. However, the visual observations given are almost the same. Although we can infer the speed of the vehicle by the scene change speed, it remains ambiguous and is not easy to learn from images. That's the reason why we need vehicle kinematic information like vehicle speed.

Limited to the simulation environment and real-world dataset, we select the following kinematic parameters as an extra input to the fully connected layer:

- vehicle acceleration rate
- vehicle speed
- vehicle heading
- vehicle lateral distance to road boundary
- vehicle previous steering angle
- vehicle steering torque

datasets	settings	type	time length	weather diversity	day/night driving
Kitti	city, rural, highway	real world	less than 2 hours	clear weather	day time
CityScape	city	real world	less than 100 hours	multiple weather conditions	day time
Comma.ai	highway	real world	7.5 hours	clear weather	day time and night time
Oxford Robocar	city	real world	214 hours	multiple weather conditions	day time and night time
Torcs	highway	simulation	-	clear weather	day time
Udacity	rural	simulation	-	clear weather	day time

Table 1. Camparison between different benchmark datasets on autonomous driving.

4. Experimental Setup

In this section, we discuss our experimental environment selection both in simulation and on the real dataset. We show how we do data preparation and explain the details of our experimental design and implementation.

4.1. Environment Selection

Recently more and more public datasets and simulation platforms for on-road driving have become available. These datasets contain diverse driving scenarios including cities, highways, towns and rural areas in the US and across the world. Here, we give a comprehensive overview and comparison of available datasets.

Datasets collected in the real world:

- The **Kitti dataset** contains sensor information captured with a car driving around the city of Karlsruhe, Germany in rural areas and on highways. Image data were collected using two high-resolution color and grayscale video cameras. The ground truth is provided by a Velodyne laser scanner and a GPS localization system. The Kitti dataset is suitable for investigating the task of stereo, optical flow, visual odometry, 3D object detection and 3D tracking.

- The **CityScape dataset** is a diverse set of stereo video sequences recorded in street scenes from 50 different cities across the world. It has high-quality pixel-level annotations of 5000 frames in addition to a larger set of 20,000 weakly annotated frames. It can be best used for semantic urban scene understanding.

- The **Comma.ai dataset** contains 7.5 hours of highway driving. The sensor input is recorded at 20 Hz with a camera mounted on the windshield of an Acura ILX 2016. Together with the images, the dataset contains information such as the car's speed, acceleration, steering angle, GPS coordinates and gyroscope angles.

- The **Oxford Robocar dataset** contains 100 repetitions of consistent routes through Oxford, UK, captured over a period of over a year. The dataset captures a combination of weather, illumination, dynamic objects, traffic, and pedestrian information, along with seasonal changes, construction, and roadworks.

Simulation environments:

- The **Torcs simulator** is an open-source racing car simulator written in C++. It is a popular video game as well as a common research platform for studying AI agents. In Torcs, many trials with various environment settings and car models with different behaviors are available.

- The **Udacity simulator** is an open-source simulator developed based on Unity. The simulation gives a realistic 3D visualization of the vehicle driving on three given tracks in the desert, mountain and forest.

A comparison of the different datasets regarding settings, type, time length, scenario and weather diversity is given in Table 1.

For the experiment parts, we used the Udacity simulation environment and the Comma.ai dataset as evaluation platforms for our algorithms. The selected datasets have access to the driver's control actions together with sensor perception information, which met our needs.

4.2. Data Preparation

In the Udacity simulation environment, we use three tracks. The three tracks respectively depict a highway running through a desert, suburb, and mountain. Example screenshots of the different trials are shown in Figure 3. The desert track is used for training purposes, and the suburb and mountain tracks are used for testing.

We collected image data for training by driving the car in the simulation environment. To introduce various driving styles from multiple people, we collected data from six people, each driving the desert track twice. We recorded the steering angle, speed, acceleration rate and braking rate paired with the corresponding images while driving with keyboard input. The system operates at a 10-hertz frequency. Altogether we collected 6245 images, totaling about 1 hour of driving data. We sampled images at 2 Hz to prevent redundant pictures. The images captured simulate the front view from the vehicle via a camera mounted on top of the car.

The images obtained are not directly used for training purposes. Before training, we preprocess and augment the

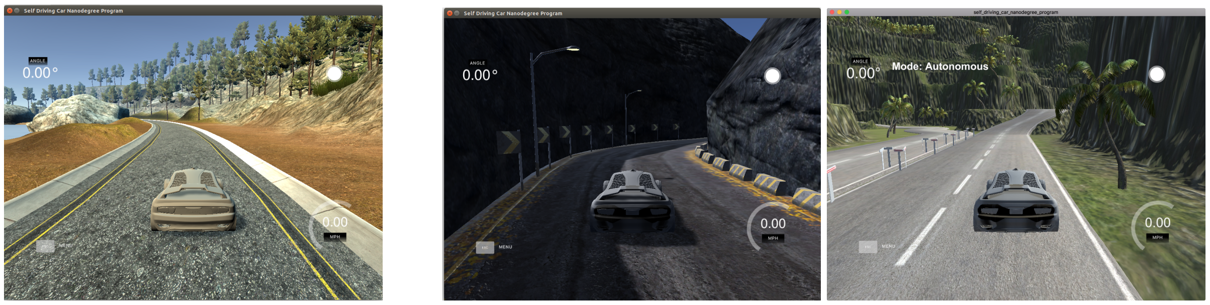
Self Driving Car Nanodegree Program
ANGLE
0.00°
0.00
MPH
Self Driving Car Nanodegree Program
ANGLE
0.00°
0.00
MPH
self_driving_car_nanodegree_program
ANGLE
0.00°
Mode: Autonomous
0.00
MPH

Figure 3. **Sample screenshots** of the environment in the Udacity autonomous driving simulator. The left one shows the training track in the desert, while the two on the right show the test track in suburb and mountain. The test sets are different from the training set regarding **lighting conditions, curvature, inclination, road geometry and off-road scenery** and thus are considered much more difficult.

data using techniques similar to those described in [3]. Data augmentation is used here to increase the size of the training set and also the diversity of training samples. The following operations were performed.

- **Cropping**: The images are cropped to remove extraneous elements. We removed the top of an image which includes a large area of sky or mountain tops and the bottom of the image which contains the car hood.
- **Upsampling**: In the original training set, most scenarios are going straight along the road. Images with a steering angle larger than 15 degrees are scarce compared to a significant number of training samples with a steering angle less than 5 degrees, which means the steering angle distribution is strongly biased towards zero. To overcome the problem of imbalanced data, we manually upsample images with steering angle greater than 10 degrees by ten times and steering angle greater than 5 degrees by five times. The data are then randomly shuffled before training.
- **Brightness changes**: Each frame is first converted to HSV space and the value channel is then multiplied by a random value from 0 to 10. This changes the lighting situation and makes the model adaptive to all kinds of weather.
- **Flipping**: We flip all frames to obtain their mirror in the horizontal direction. This helps to make sure we have exactly the same amount of left and right turning samples. The algorithm won't suffer from any bias in the left or right direction.

For the real data from the Comma.ai driving dataset, there is no need for cropping and brightness preprocessing. We use the same upsampling and flipping techniques to deal with the data balance problem. The dataset includes 11 video clips of 7.5 hours of highway driving at 20 Hz. Here we only want to consider stable highway driving at normal speed in daylight. We further exclude the driving videos at night or in traffic jams with speed under 10mph. The finally selected footage has a length of about 2 hours of driving. We split it by using 130K frames for training and 20K frames for testing.

4.3. Implementation Detail

As a baseline to compare our algorithms against we use a variation of the CNN network structure proposed in [3]. The difference here is that we add batch normalization and dropout layers after each convolutional layer for better convergence and performance.

The CNN network consists of 5 convolutional layers. The first three layers have a kernel size of 5×5, and the last two layers have a kernel size of 3×3. The depth of each feature map is 24, 36, 48, 64, 64. The activation function we use here is ReLu.

Our model has an additional LSTM layer apart from convolutional layers, as shown in Figure 1. The LSTM has 128 hidden units. The output of LSTM and the vehicle kinematic dynamics are concatenated before fed into the FCN module. The FCN module consists of 3 fully connected layers with 256, 128 and 64 units followed by a $Tanh$ activation function.

We use Adam optimization to train all networks. The learning rate is fixed to 0.001 with a momentum decay of 0.9. The batch normalization and dropout layers are used to prevent overfitting.

5. Evaluation and Results

We report our experimental results on the Udacity simulation and the Comma.ai dataset. For the steering angle prediction task, we use Root Mean Square Error (RMSE) as the evaluation metric. RMSE can express average system prediction error on the dataset. The RMSE metric is defined as

$$RMSE = \sqrt{\frac{1}{|D|}\sum_{i=1}^{|D|}(\hat{a}_i - a_i)^2},$$

where $\hat{a}_i$ and a_i are the ground-truth and predicted steering angle for frame i and $|D|$ is the total number of frames in the test set. The angles are measured in angular degrees.

The RMSE can estimate the precision of the system but can neglect the stability of the system. So we also define a stability metric based on the deviation of our prediction. The intuition behind it is that we want our predictions to change smoothly without any sudden bump in the steering angle. We call this metric Mean Continuity Error (MCE).

$$MCE = \sqrt{\frac{1}{|D|-1}\sum_{i=1}^{|D|-1}(a_{i+1} - a_i)^2}$$

We evaluate the influence of the different methods suggested with the baseline method (see Table 2). We first compare the baseline CNN structure with two popular networks Vgg [10] and Resnet [9] for the image recognition tasks. To maintain roughly the same number of weights for training in different models, we only train the last five layers of Vggnet and Resnet. Here transfer learning shows that the pre-knowledge from the image recognition task is beneficial for the job of predicting steering angles. It should be noted that the performance boost in the Comma.ai dataset is much more substantial that Udacity simulation. This is most likely due to the fact that the Comma.ai dataset contains real imagery which has a higher resemblance with Imagenet dataset than the simulation environment.

To evaluate the effect of the segmentation map augmentation, we compare the result of adding the segmentation map as an extra feature map for input to the convolutional layers. We use the segmentation map output categories of the sky, road marking, road, tree, pavement, and vehicle. We construct a binary map for each group and stack them together with the original three channels of the image. As can be seen in the fourth row of Table 4, the precision and stability boost is massive compared to the raw image input. Example segmentation map outputs are shown in Figure 4.

Next, we evaluated the effect of adding temporal information by using the layer of LSTM. The results are shown in the firth and sixth rows of Table 2. The performance is increased both in the Comma.ai dataset and in the Udacity simulation. We conducted grid-search for the optimal window size w for the LSTM and found that $w = 3$ generates the best result, which means we look back for 1.5 seconds. We compare the main difference of the prediction with the baseline model and discover that after using temporal information, absurd outlier predictions are significantly reduced. The cases where two sequential frames make dramatically different steering angle predictions almost disappear. This dramatically improves the stability of the algorithm and also improves the prediction accuracy.

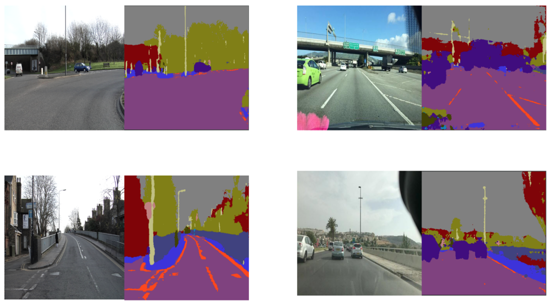
Figure 4. Example intermediate segmentation outputs obtained in the end-to-end learning procedure.

Adding the vehicle kinematics also slightly improves the performance, by about 3%. The ablation test shows the vehicle previous steering angle is the most useful, followed by vehicle speed and vehicle heading. We observe no performance boost in using the vehicle acceleration rate, vehicle lateral distance to road boundary and vehicle steering angle.

Our overall proposed architecture has reduced prediction RMSE error by 44.92% in the Udacity simulation and 48.44% in the Comma.ai dataset. The prediction MCE error was reduced by 41.81% in the Udacity simulation and 49.58% in the Comma.ai dataset.

We also do an empirical test on the Udacity simulation environment to see if the car can successfully drive in a new scenario. The result shows good driving performance on an unseen map with various sceneries in suburb and mountain. A video clip of our trained model is available at `https://youtu.be/reqAHtXtnrI`.

6. Conclusion

We suggested a network architecture that improves the baseline vision-based end-to-end learning of steering angle. The suggested network architecture uses four distinct methods: adding auxiliary segmentation, transferring from existing tasks, utilizing temporal information and incorporating vehicle information. We found that using transfer learning from the Imagenet recognition task can be helpful in learning the task of steering for on-road driving. Using the pre-trained segmentation mask to categorize the image at the pixel level can empower the network with more information and thus result in better prediction accuracy. The incorporation of temporal information of history states indeed helps to make better current decisions, which again proves the concept that driving policy is a long-term decision-making process. Finally, the proper addition of some vehicle kinematics makes the state representation more concrete.

Proposed Network Structure	Udacity Simulation		Comma.ai Dataset	
	RMSE/degrees	MCE/degrees	RMSE/degrees	MCE/degrees
baseline CNN	7.68	2.32	19.84	7.26
Vgg CNN	7.45	2.12	15.86	5.73
Resnet CNN	7.34	2.09	15.23	5.35
Resnet CNN + SegMap	5.23	1.57	12.33	4.21
Resnet CNN + SegMap + LSTM	4.50	1.33	10.72	3.78
Resnet CNN + SegMap + LSTM + vehicle kinematics	**4.23**	**1.32**	**10.23**	**3.66**

Table 2. Comparison between different network structures for vision-based end-to-end learning of steering angle. Our proposed method has the lowest RMSE and MCE both in the Udacity simulation and on the Comma.ai dataset compared to the baseline method.

Acknowledgment

This work is funded by GM.

References

[1] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pages 173–182, 2016. 2

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017. 3

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 1, 2, 5

[4] Z. Chen and X. Huang. End-to-end learning for lane keeping of self-driving cars. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 1856–1860. IEEE, 2017. 1

[5] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016. 2

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. In *NIPS'13 Workshop on Deep Learning*, 2013. 2

[7] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun. Off-road obstacle avoidance through end-to-end learning. In *Advances in neural information processing systems*, pages 739–746, 2006. 1

[8] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010. 3

[9] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani. End-to-end deep reinforcement learning for lane keeping assist. *arXiv preprint arXiv:1612.04340*, 2016. 6

[10] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[11] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi. Towards a viable autonomous driving research platform. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 763–770. IEEE, 2013. 1