End-to-End Deep Neural Network Design for Short-term Path Planning

Minh Quan Dao, Davide Lanza, Vincent Frémont

End-to-End Deep Neural Network Design for Short-term Path Planning

Minh Quan Dao[1], Davide Lanza[1] and Vincent Frémont[1]

***Abstract*— Early attempts on imitating human driving behavior with deep learning have been implemented in an reactive navigation scheme which is to directly map the sensory measurement to control signal. Although this approach has successfully delivered the first half of the driving task - predicting steering angles, learning vehicle speed in an end-to-end setting requires significantly large and complex networks as well as the accompanying dataset. Motivated by the rich literature in trajectory planning which timestamps a geometrical path under some dynamic constraints to provide the corresponding velocity profile, we propose an end-to-end architecture for generating a non-parametric path given an image of the environment in front of a vehicle. The level of accuracy of the resulting path is 70%. The first and foremost benefit of our approach is the ability of incorporating deep learning into the navigation pipeline. This is desirable because the neural network can ease the hardness of developing the see-think-act scheme, while the trajectory planning at the end adds a level of safety to the final output by ensuring it obeys static and dynamic constraint.**

I. INTRODUCTION

Motion planning methods for autonomous vehicles are classically developed to sequentially perform path planning, obstacles avoidance, and trajectory optimization [1], [2], [3]. Path planning module, which can be implemented by RRT [4], A* [5], Lattices and motion primitives [6], or function optimization [7], takes into account the geometry characteristic of the environment to produce a collision-free (with regard to static obstacle) non-parametric path. On the other hand, the trajectory optimization module, which subsumes the obstacles avoidance, aims to optimize both way points and the corresponding velocity profile so that they respect vehicle's kinematics and dynamic obstacles [8], [7].

Deep learning can help simplify the path planning and trajectory optimization pipeline above with just a deep neural network which learns human driving behavior in a supervised manner. The first successful example traces back to ALVINN [9], where a shallow network was used to calculate the steering angle directly from images. This work is revised in the deep learning era in [10], the authors went beyond a mere pattern recognition, learning the entire steering angle prediction pipeline for autonomous cars by building a mapping from images obtained by a forward camera to steering angle with a deep CNN. Taking the similar approach but with different input, [11] used visual information obtained by an event-based camera to train their steering model. A more global approach to motion planning using deep learning is to learn a spatial traversability maps [12], [13]. In these works, rather than just predict a single steering angle, the model is designed to learn a cost function with can be later used for path planning. The common shortage of these works is their inability of addressing vehicle speed.

To solve this problem by the end-to-end learning, [14] uses recurrent layers together with CNN network to learn a complete driving policy (steering angle and vehicle speed). Though this work has showed its capability and robustness, it comes with the cost of an extremely large dataset and fairly complex network as well as the training process. Taking a different approach, [15] proposed an integrated solution where a CNN is trained on monocular image data (as in [10]) to output a path, this path is then used as the initial guess for a Particle Swarm Optimization algorithm [16] which in turn transforms the path to a complete trajectory.

Another method to infer vehicle speed is to calculate it proportional to the collision probability [17]. The network designed in this work is made of three consecutive ResNet blocks following by two parallel fully-connected layers, which respectively output the steering angle and a collision probability. The steering prediction is learned through a regression problem, while the collision prediction is addressed as a binary classification problem.

In this paper, we propose and evaluate a deep neural network architecture inspired by DroNet for short-term path planning which is to predict a sequence of steering angles directly from an image obtained by forward camera, hence an end-to-end model. The main difference between [17] and our work is the statement of steering angle prediction problem. In fact, as mentioned in [18], it can be transformed from a regression problem of continuous values to a classification problem where the steering angle range is tessellated into discrete spans with width of 0.01 radians. Such choice of span width is justified by the jitter of steering angle applied by a human driver in straight road. Moreover, the calculated steering sequence is mapped into a non-parameterized path and, once the path is output, any motion planning algorithm can be implemented to timestamp the path.

The rest of this paper is organized as follows. Sec.II adapts the DroNet architecture for learning a single steering angle through a classification problem. In Sec.III, we further modify the resulted architecture and the used dataset to enable the network to learn a geometrical path in an end-to-end setting. The performance of the path planning model derived in the Sec.III is evaluated in Sec.IV. Then Sec.V describes the short-term path generation using the steering angles sequence and car-like vehicles motion constraints.

[1]Authors are with Centrale Nantes, LS2N - UMR 6004 Nantes, France email: vincent.fremont@ec-nantes.fr

Concluding remarks are made in Sec. in VI.

II. LEARNING STEERING ANGLES

Taking a different approach compared to the majority of researches in end-to-end learning for autonomous driving which formulates the prediction of steering angles as a regression problem [9], [10], [17], our model is designed to be a classifier. The reason of our choice stems from the fact that for a regression-based network, there is a continuous range of angles to infer, but only a finite number of samples with which to train against. By tessellating the range of steering angles into discrete bins, the requirement of infinite training samples to fully cover such continuous range is no longer effective. Moreover, Sec.IV shows that with sufficiently small bins, our classifier can outperform the DroNet - a regression-based model.

A. Network Architecture

[17] has showed that their the architecture responds strongly to "line-like" features in the forward images which has a strong relation with the resulted steering angles. Motivate by this work, we design our model out of their ResNet-made body and put a classifier on top of it. This classifier is made of 2 dense layers. The first has 800 neurons activated by ReLU, while the second has 227 neurons (equal to the number of classes) and is activated by Softmax function. The conceptual architecture is shown in Fig.1. Each ResNet block in this figure is comprised of 3 convolutional layers: 2 on the main path and 1 on the shortcut (see Fig. 2).

The hyperparameters of each ResNet block are shown in Tab. I.

Stage	Layer	Number of kernels	Kernel size	Stride Stride	Padding Padding
1	Conv2D_a	32	3	2	same
1	Conv2D_b	32	3	1	same
1	Conv2D_c	32	1	2	same
2	Conv2D_a	64	3	2	same
2	Conv2D_b	64	3	1	same
2	Conv2D_c	64	1	2	same
3	Conv2D_a	128	3	2	same
3	Conv2D_b	128	3	1	same
3	Conv2D_c	128	1	2	same

TABLE I
MODIFIED RESNET CNN BODY PARAMETERS

The intuition behind our model is that the ResNet-made body should learn better how to output useful feature maps which probably contains roads shape and drivable area, while the classifier on its top should learn how to output the steering angle given the provided feature map.

B. Data Preparation

The dataset used to train our model is Udacity dataset challenge 2[1]. This dataset contains several hours of driving on suburban road in good weather and lightning condition. After processing, the dataset is organized as a time-order list. An element of this list contains an image captured by front-facing camera, the associated steering angle, GPS coordinate of the vehicle at this time step and other information.

To decide the class of any steering angle, a histogram of all steering angles in the dataset ranging from -2.051 to 1.903 radian is built (see Fig.4). The width of every bin of this histogram is 1 degree. The class of an angle is the index of the bin it belongs to. As can be seen in Fig.4, the distribution of collected steering angles among these classes is imbalance. To ensure the network does not overlook the less frequent classes during the training process, class i is assigned a weight $w(i)$ based on the median frequency balancing method in [19].

$$w(i) = \frac{\text{median frequency of the dataset}}{\text{frequency of this class}} \tag{1}$$

Here, the numerator is the ratio between the number of samples in class i and the total number of samples in the dataset. The denominator is the median of the all frequencies. The resulted weight $w(i)$ is later used to scale the contribution of every sample in class i to the total loss function.

C. Model Training

The model's weights are initialized randomly and the network is trained by minimizing the cross entropy loss function using Adam optimizer with default parameters. After training for 100 epochs with 1200 batch size, our model's accuracy on validation set peaks at 66% before dropping due to over fitting. With this level of accuracy, its qualitative performance measured by Root Mean Square Error (RMSE) and Explained Variance score (EVA) are respectively 0.1083 and 0.8338. These values are competitive, compared to recent development in end-to-end learning model: DroNet [17] and [11]. The details comparison is carried out in Sec.IV.

III. LEARNING A GEOMETRICAL PATH

As shown in the previous section, a single steering angle can be learned through a classification problem. Nevertheless, knowing the steering angle is just half of the autonomous driving task. The other control signal to be provided, is the vehicle's speed. There are no means of inferring a vehicle speed given a single steering angle at the same time instance. However, the rich trajectory planning literature suggests that a geometrical path can be timestamped to generate a velocity profile [20] such that it can satisfy some dynamic constraint. Therefore, in this section, we modify the resulted architecture from the previous section and the used dataset to enable the network to learn a geometrical path.

A. Network Architecture

Inspired by [15], a path can be encoded as a sequence of steering angles, each of which is applied to a predefined traveling distance. Based on this insight, a network can learn a path by learning a sequence of steering angles. This leads to the replacement of a single classifier on top of the

[1] https://github.com/udacity/self-driving-car/tree/master/datasets/CH2

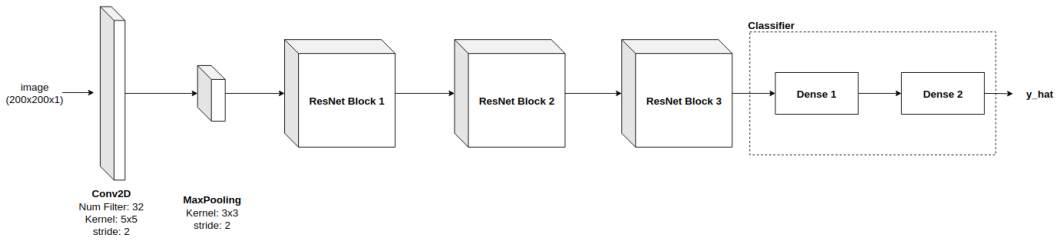

Fig. 1. Modified DroNet architecture [17]

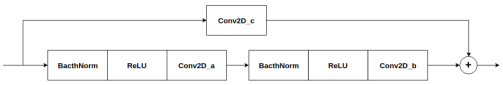

Fig. 2. Components of a ResNet block.

architecture in Sec.II by an array of 5 classifiers having the same number of neurons and activation function. Putting the ResNet-made body and the array of classifiers together, the complete architecture is shown in Fig. 3. Here, a block **Head i** is a 2-dense-layer classifier.

B. Dataset Preparation

The new model's output is interpreted as a sequence of 5 steering angles. Each angle is applied to 2 meters of traveling distance. As a result, a training sample is prepared as following:

- X: an image of the environment in front of the vehicle
- y: a list of one-hot vectors. This first vector denotes the class of the steering angle associates with the frame represented by X. The $i-th$ vector represents the steering angle of the frame $i \times 2$ meters away from X.

Such definition of y suggests that the data set used to train path planning model needs to explicitly contain the distance information between two adjacent labels. This distance can be retrieved from the GPS coordinate of each frame provided in the original Udacity dataset. Upon completely being generated, the whole dataset is divided into training set and validation set with respectively 19000 and 2350 samples.

C. Model Training

Since the ResNet-made body is the same the model which predicts a single steering angle in Sec.II, the weight of the ResNet-made body of the path planning model in this section is trained from the best weight obtained in Sec.II. Conversely, all classifiers' weight are initialized randomly. The loss function is chosen to be the cross-entropy and is minimized by the Adam optimizer. The choice of hyper-parameters is the same as in Sec. II.

After training for 50 epochs, each of 5 five classifiers in our model achieves at least 70% of accuracy.

IV. MODEL PERFORMANCE

A. Quantitative Performance

Using the same approach of [17], [11], the quantitative performance is measured by Root Mean Square Error (RMSE) and the Explained Variance score (EVA). The performance over these metrics of 5 classifiers in our path planning model compared to a constant estimator, which always predicts 0 as steering angle, a random one, the DroNet [17], and the model taking input from an event-based camera [11] is shown in Tab. II. All classifiers in our model outperform the DroNet in both RMSE and EVA, while fall behind event-based model with a small margin in RMSE. This shows by tessellating the range of steering angle into sufficiently small intervals, a classification-based model can deliver a better result, compared to a regression-based one.

Model	RMSE	EVA
Constant baseline	0.2129	0
Random baseline	0.3 ± 0.001	-1.0 ± 0.022
DroNet	0.1090	0.7370
Event-based Model	0.0716	0.8260
Head_0	0.0869	0.8933
Head_1	0.0920	0.8781
Head_2	0.1052	0.8382
Head_3	0.0820	0.9012
Head_4	0.0851	0.8943

TABLE II
PATH PLANNING MODEL QUANTITATIVE PERFORMANCE

B. Qualitative Performance

The comparison between the histograms of predicted angle classes and their ground truth on validation set are shown in Fig.4. This figure indicates a relative match between the predicted distribution and the true distribution.

In addition, the normalized confusion matrix of the first classifier is shown in Fig.5. This matrix features a clear, large magnitude main diagonal. This means the majority of predicted angle classes is actually the true class. Nevertheless, there are a few strong cells in the bottom of Fig.5 implying

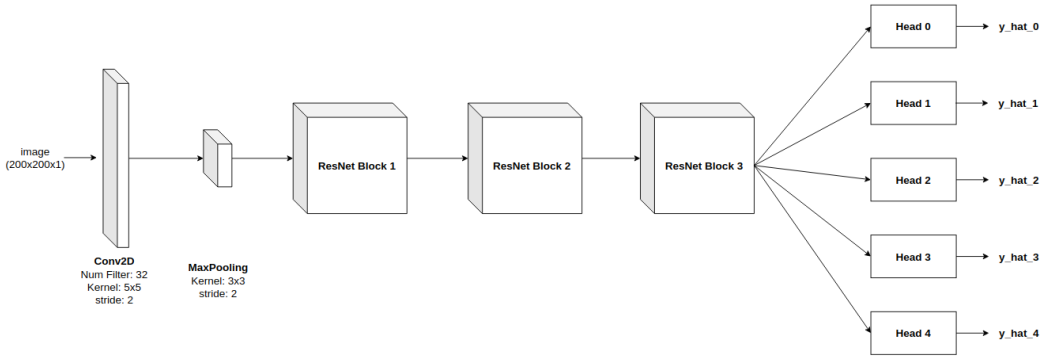

Fig. 3. Path planning architecture (CNN part)

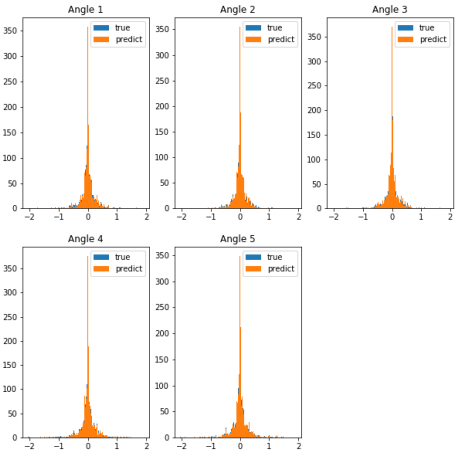

Fig. 4. Predicted angle classes distribution of each classifier compared to their ground truth

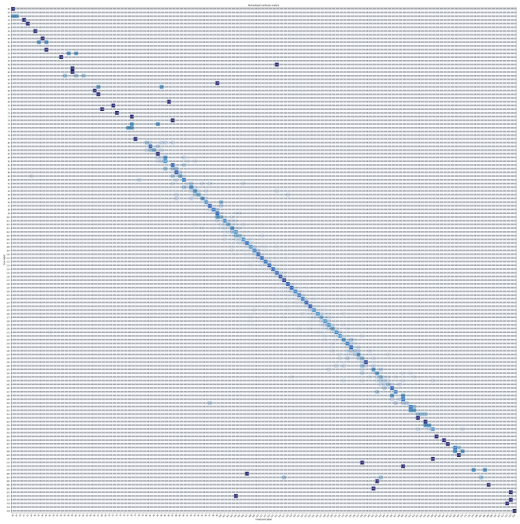

Fig. 5. Normalized confusion matrix of the first classifier.

that the classifier fails to predict the class of extreme right angles.

C. Layer Activation Visualization

In an attempt to understand how our model produces its prediction, the outputs of each ResNet block in the path planning model are displayed in Fig.6. This figure shows that the first block recognizes lane mark and vehicles, while the second block segments the drivable area. The last block learns a down-sampled mapping. Together, these three ResNet blocks learn useful feature maps which contains roads shape and drivable area, while the classifiers on the top learn to calculate the steering angle given those feature maps.

V. INTERPRETING A STEERING SEQUENCE AS A PATH

Since the motion of car-like vehicles is constrained to be circular around its Instantaneous Center of Rotation (ICR) (see Fig. 7), a sequence of steering angles can be interpreted as a geometrical path (i.e. sequence of way points) by applying each angle in the sequence to a predefined traveling distance of s meters.

In Fig.7, L is the distance between the front and rear axle. $\delta_i (i = 0 \dots N)$ is a steering angle ($N+1$ is the length of steering angles sequence). At each time instance, the pose of the vehicle is represented by the pose of the local frame attached to the center of its rear axle - $O_i x_i y_i z_i (i = 0 \dots N)$. x_i goes from the rear axle to the front axle, and is perpendicular to these axles. z_i is orthogonal to the plane of motion, and pointing outward. y_i is defined such that $O_i x_i y_i z_i$ is right-handed. The target is to calculate the position of the center of the front axle relative to the local body frame at the presence - $O_0 x_0 y_0 z_0$. Assuming that the vehicle's motion is planar, the transformation from frame $O_i x_i y_i z_i$ to frame $O_{i+1} x_{i+1} y_{i+1} z_{i+1}$ is described by:

$$ {}^{i}T_{i+1} = \begin{bmatrix} Rot_{z,\phi_i} & {}^{i}t_{i+1} \\ 0_{1\times 2} & 1 \end{bmatrix} \tag{2} $$

Here, ${}^{i}t_{i+1}$ is the coordinate of O_{i+1} in frame i, and

Fig. 6. Output of each ResNet block with respect to different input images. From left to right, the images are respectively the input image, output of the first, second, and third block.

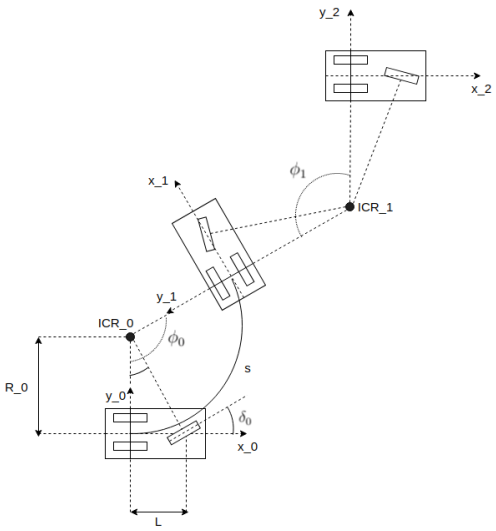

Fig. 7. Car-like vehicle's motion diagram with two different value of steering angles. Each of these angles is applied to an arc distance of s meters.

Rot_{z,ϕ_i} is the matrix represents the rotation around the z-axis by an angle ϕ_i. The radius of the circular motion around ICR_i is calculated as

$$R_i = \frac{L}{\tan \delta_i} \tag{3}$$

Eq.3 implies that when the steering angle is close to zero (i.e. the steering wheel is kept at the neutral position), the radius of motion approach infinity, hence a straight motion. Given R_i, the coordinate of O_{i+1} in frame i is

$$^{i}t_{i+1} = 2R_i \sin\left(\frac{\phi_i}{2}\right) \begin{bmatrix} \cos\left(\frac{\phi_i}{2}\right) \\ \sin\left(\frac{\phi_i}{2}\right) \end{bmatrix} \tag{4}$$

ϕ_i is the angle between x_i and x_{i+1}. As shown in Fig.7, this angle can be calculated by:

$$\phi_i = \frac{s}{R_i} = \frac{s \tan \delta_i}{L} \tag{5}$$

With Eq. 5 and Eq. 4. The transformation from frame i to frame $i+1$ in Eq. 2 is now fully defined. The local position of the center of the front axle in homogeneous form is:

$$^{i}L_i = \begin{bmatrix} L, 0, 1 \end{bmatrix}^T \tag{6}$$

This position is transformed into the local frame at the present time by the following equation

$$^{0}L_i = {}^{0}T_i {}^{i}L_i = \prod_{j=1}^{i} {}^{j-1}T_j {}^{i}L_i \tag{7}$$

To test the quality of the predicted path and its ground truth, the trained path planning model is used to infer path from forward images taken from Udacity dataset. The inference process is implemented on a laptop equipped with an NVIDIA GeForce MX130, an Intel Core i7-8650U (1.90GHz), and 16GB of RAM. The inference time for 1 sample (i.e. 1 image) is 1 millisecond. Examples of path generated by both true and predicted sequence of steering are shown in Fig.8, while the extended video of path planning model's output compared to ground truth can be found in this link: `https://youtu.be/X2fi2xVr2jE`. Fig.8 as well as the video shows a good match between the predicted path and its ground truth, which in turn proves the quality of the prediction of our model.

VI. CONCLUSIONS

In this paper, we explored an end-to-end learning approach to path planning for autonomous vehicles. In details, a neural network made of three ResNet blocks and an array of

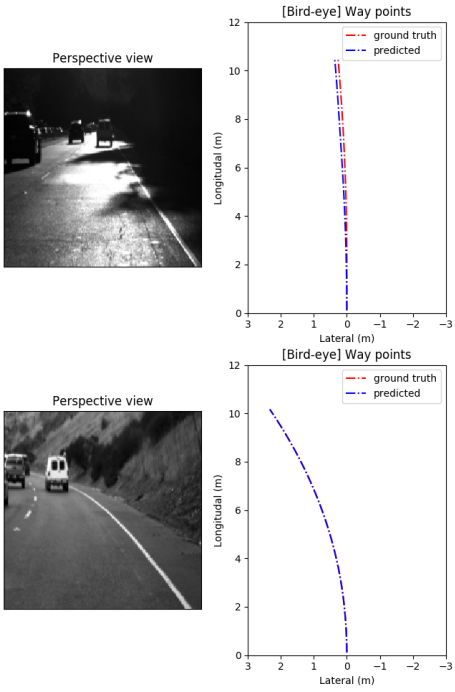

Fig. 8. Display of the predicted path. Left: Images captured by front-facing camera. Right: Path in cartesian coordinates

classifiers is trained to output a sequence of steering angles which is later interpreted into a geometrical path.

The advantage of this approach is that it pays the way for the integration of deep neural network into the motion planning framework. Specifically, the geometrical path learned by the network is then timestamped using trajectory optimization technique to finally produce a parameterized path (a path with a velocity profile). Beside providing two crucial control signals of the driving task (steering angle and velocity), this integrated approach enhances the reliability of the predicted trajectory while ensures it is natural and consistent with vehicles' kinematic.

For the future work, the accuracy of the array of classifiers needs to be improved. Furthermore, the network generalization should be evaluated on other autonomous vehicles datasets with different camera parameters. Since a sequence of steering angles implies a time order, it might be helpful if the network can learn a temporal relation among the steering angles. This can be done by exploring the application of recurrent layers such as LSTM cells to enable the model learning such time relation.

The code used in this paper is hosted on GitHub in Minh-Quan Dao's ECN-E2E repository: `https://github.com/quan-dao/ECN-E2E`.

ACKNOWLEDGMENT

This research has been conducted as part of the HI-ANIC (Human Inspired Autonomous Navigation In Crowds) project, funded by the French Ministry of Education and Research and the French National Research Agency (ANR-17-CE22-0010).

REFERENCES

[1] C. Katrakazas, M. A. Quddus, W. hua Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," 2015.

[2] T. Gu, J. Snider, J. M. Dolan, and J. Lee, "Focused trajectory planning for autonomous on-road driving," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, June 2013, pp. 547–552.

[3] Wenda Xu, Junqing Wei, J. M. Dolan, Huijing Zhao, and Hongbin Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 2061–2067.

[4] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, "Motion planning for urban driving using rrt," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 1681–1686.

[5] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[6] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 1879–1884.

[7] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha a local, continuous method," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, June 2014, pp. 450–457.

[8] C. Liu, W. Zhan, and M. Tomizuka, "Speed profile planning in dynamic environments via temporal optimization," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 154–159.

[9] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313.

[10] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[11] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5419–5427, 2018.

[12] M. Wulfmeier, D. Z. Wang, and I. Posner, "Watch this: Scalable cost-function learning for path planning in urban environments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 2089–2095.

[13] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *I. J. Robotics Res.*, vol. 36, pp. 1073–1087, 2017.

[14] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3530–3538, 2016.

[15] C. Hubschneider, A. Bauer, J. Doll, M. Weber, S. Klemm, F. Kuhnt, and J. M. Zllner, "Integrating end-to-end learned steering into probabilistic autonomous driving," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017, pp. 1–7.

[16] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 5, Oct 1997, pp. 4104–4108 vol.5.

[17] A. Loquercio, A. Maqueda, C. del-Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[18] P. Penkov and V. S. J. Ye, "Applying techniques in supervised deep learning to steering angle prediction in autonomous vehicles," 2016.

[19] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," *eprint arXiv:1411.4734*, 2014.

[20] C. Katrakazas, M. Quddus, W. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416 – 442, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X15003447