Optimization of End-to-End Deep Learning for Obtaining Human-Like Driving Models

Erik Dovgan[1], Jaka Sodnik[2], Bogdan Filipic[3]
[1, 3] Department of Intelligent Systems
Jozef Stefan Institute
Jamova cesta 39, SI-1000 Ljubljana, Slovenia
[2] Faculty of Electrical Engineering
University of Ljubljana
Trzaška cesta 25
SI-1000 Ljubljana, Slovenia
NERVteh, raziskave in razvoj
d.o.o. Kidriceva ulica 118
SI-1236 Trzin, Slovenia
erik.dovgan@ijs.si, jaka.sodnik@fe.uni-lj.si, bogdan.filipic@ijs.si

ABSTRACT: *Modeling human driving with human-like driving models can help companies in the evaluation of human drivers. While a human-like driving model can be tested in various scenarios, this is not feasible for driver evaluation due to time constraints. During the evaluation, only a small set of driving data can be typically collected for each driver, which represents an issue for advanced modeling approaches such as deep learning. To overcome this issue, an optimization approach is proposed, which tunes deep learning when a small learning dataset is available.*

Keywords: Optimization, Deep Learning, Human-like Driving Models

Received: 8 September 2018, Revised 2 November 2018, Accepted 10 November 2018

1. Introduction

Human-like driving models have been learned with several methods, such as ARX models [8], Gaussian processes [11], Gaussian mixture models [1], artificial neural networks [15], support vector regression [13], etc. Recently, Deep Neural Networks (DNN) are being effectively used in learning tasks from various application fields. For example, when driving a vehicle, DNN can be used to recognize the road, other vehicles, pedestrians, etc. from video data [7]. Moreover, DNN has been also applied to directly learn the control actions from video data without firstly reconstructing the scene. This approach is called end-to-end learning and its examples aim to learn steering, throttle and braking control actions, etc. [5, 6, 14].

Unfortunately, deep learning has a significant drawback: it requires a lot of learning data. Existing driving datasets used for training DNN models vary from about 10 hours to up to 10,000 hours [14]. However, in some cases such a large set of driving data is not available. For example, the deep learning approach can be used to assess a driver, e.g., if he/she drives safely, is able to avoid critical situations, etc. [12]. This can be done by building a human-like driver model, i.e., a clone of the driver, and test it in a large number of driving situations. A similar approach has been applied in related domains where the goal was to learn human behavior [10]. This procedure requires only a small set of driving data, i.e., driving data of only a small subset of driving situations. Consequently, the time to collect the driving data is reduced, while the driver or more precisely his/her clone is still evaluated in a large number of situations.

Existing work has demonstrated that end-to-end approach for learning to drive is appropriate when large sets of learning data are available [5, 6, 14]. On the other hand, the problems with small sets of learning data have not been addressed appropriately. This paper aims at tackling this issue by enhancing end-to-end deep learning approach with optimization in order to obtain human-like driving models from small sets of learning data.

The paper is further organized as follows. Section 2 presents the optimization approach for end-to-end deep learning. Experiments and results are described in Section 3. Finally, Section 4 concludes the paper with ideas for future work.

2. Optimization of End-to-end Deep Learning

End-to-end deep learning approach applies deep neural networks to learn the transformation between the input and the output data. The main property of this approach is that a single model is used to obtain this transformation. There exist also other approaches that decompose the problem and apply specific models for each subproblem. For example, one model can be used to recognize the objects, while another model can be used for higher-level reasoning [7]. The end-to-end approach aims at solving all the sub problems at once with a single model [5].

Existing work in the field of end-to-end deep learning for obtaining human-like driving models has shown that the selection of deep learning model and its parameter values is not straightforward [9]. In addition, the data need to be augmented to learn how to recover from poor positions or orientations [3]. We propose to automate the selection of appropriate parameter values and data augmentation functions with an evolutionary algorithm. Evolutionary algorithms are search and optimization algorithms inspired

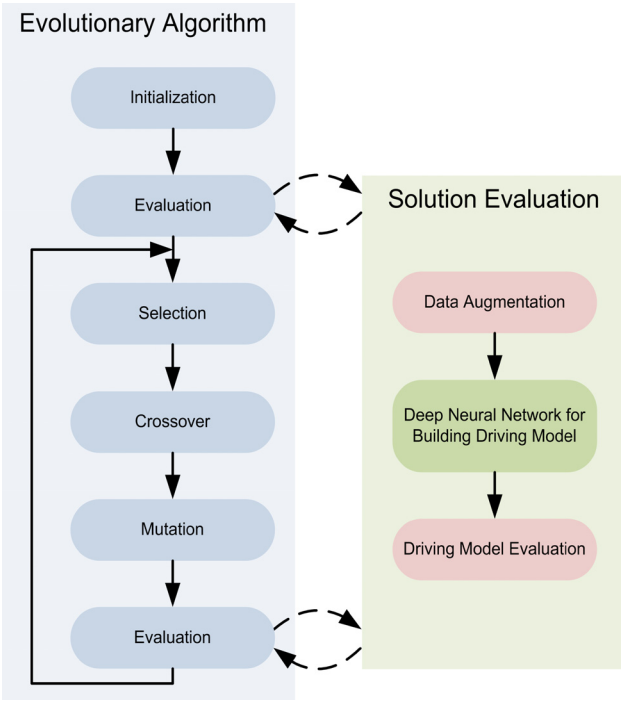


Figure 1. Overview of the algorithm for obtaining human-like driving models

by the principles of biological evolution. They work with a set of solutions that are improved through several generations by applying genetic operators, i.e., selection, crossover and mutation [4].

We propose to discover human-like driving models in two steps. In the initial step, driving models that are able to drive the vehicle along a route are built, while in the final step, these models are enhanced to imitate human driving. The approach presented in this paper focuses on the initial step by applying an evolutionary algorithm to maximize the length of the route that has been traveled by the driving model during the simulation. Each solution (consisting of parameters of model construction) is evaluated by applying the following steps:

1. The learning data are augmented to enable recovery from poor situations or orientations.

2. The deep learning algorithm is used to learn a human driving model.

3. The driving model is evaluated on a route to measure the route length of feasible driving.

The driving simulation stops if the driving becomes infeasible (e.g., the vehicle goes offroad) or when the entire route is traveled. The evolutionary algorithm applies tournament selection (tournament size = 2), two-point crossover (probability = 0.9) and uniform mutation (probability = 0.1) to improve the solutions over generations. An overview of the developed algorithm and its steps, i.e., evolutionary algorithm steps (selection, crossover and mutation) and solution evaluation steps (data augmentation, model building and model evaluation), is shown in Figure 1.

The evolutionary algorithm optimizes the following deep learning and data augmentation parameters:

Batch size: Parameter of the deep learning algorithm. Defines the number of training examples utilized in one learning iteration.

Number of epochs: Parameter of the deep learning algorithm. Defines the number of passes through the training dataset during learning.

Image Multiplier: Data augmentation parameter. Defines how many times an image is multiplied. If it is multiplied, it is divided into overlapping sub images. For example, if the image is multiplied by 3, three images are created containing: 1) left 80 % of the original image; 2) central 80 % of the original image; 3) right 80 % of the original image.

The control actions are also appropriately adapted. For the left images steering is added to simulate turning right, while for the right images steering is subtracted to simulate turning left.

Noise added to output: Data augmentation parameter. Defines the amount of noise a_n to be added to the control actions. The amount of noise is randomly selected at each time step with a uniform distribution $\mathcal{U}(-a_n, a_n)$.

Flip image: Data augmentation parameter. Defines whether randomly selected images should be vertically flipped. If the image is flipped, the control action is also appropriately adapted.

Activation function: Parameter of the neural network model. Defines the activation function of the neural network layers.

Kernel Regularizer: Parameter of the neural network model. Defines the regularization of the neural network layers, which applies penalties on layer weights.

The penalties try to keep the weights small, which reduces the possibility of over weighting a small subset of layer's input data and prevents over fitting.

3. Experiments and Results

The developed approach was tested on two scenarios. Both scenarios did not contain traffic vehicles or pedestrians. For both scenarios, the same architecture of the neural network was used. This architecture is shown in Figure 2 and is based on the architecture presented in [2]. It contains five convolutional layers and three fully connected layers. The convolutional layers

extract features, from simple features such as lines to complex features such as road contour. The fully connected layers implement the vehicle controller, which calculates the control action based on the extracted features.

3.1 First Scenario

The first scenario consisted of a circular route of around 2 km, which is shown in Figure 3a. An example of a route image as input to the neural network is shown in Figure 4a. The learning data were obtained from one driving along the route.

The proposed approach was evaluated by tuning only a subset of the parameters listed in Section 2, which already enabled us to obtain models that drove along the entire route for this scenario and consequently no additional parameters needed to be tuned. The values of tuned and not tuned parameters are shown in Table 1.

The feasible solutions, i.e., those solutions that drove the entire route, are shown in Table 2. These results show that feasible solutions multiply the images by 3 or 5 and flip images, while the noise added to output does not influence the results. In addition, the results also show that a lower number of epochs is needed if the images are multiplied more times.

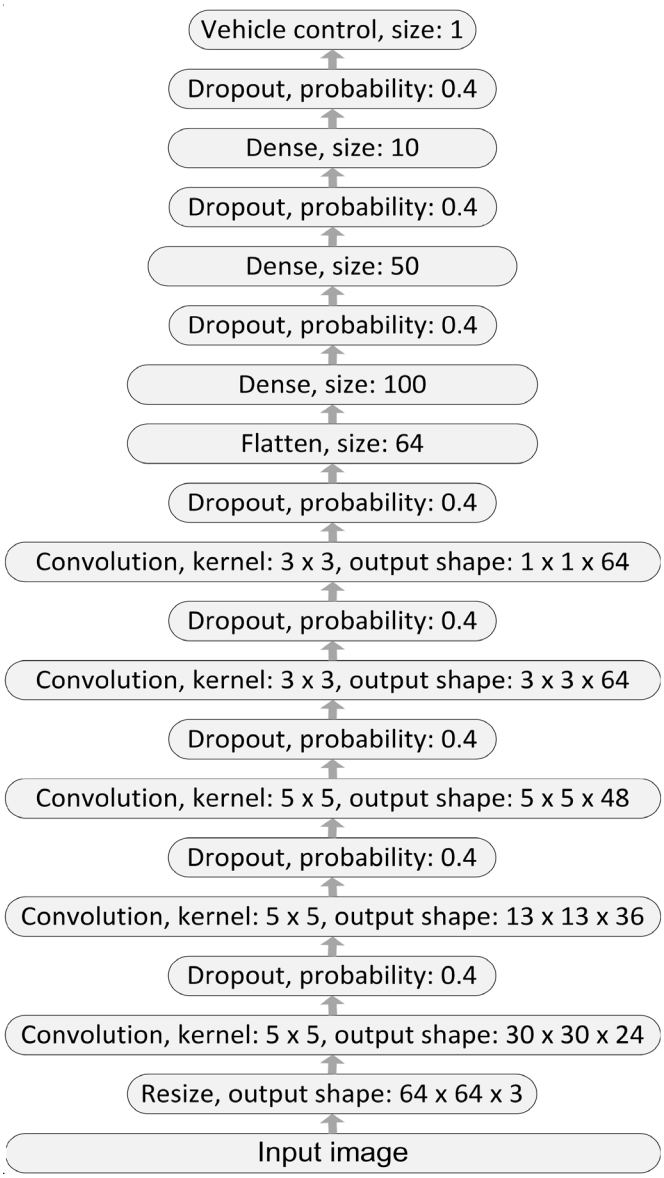


Figure 2. Architecture of the neural network

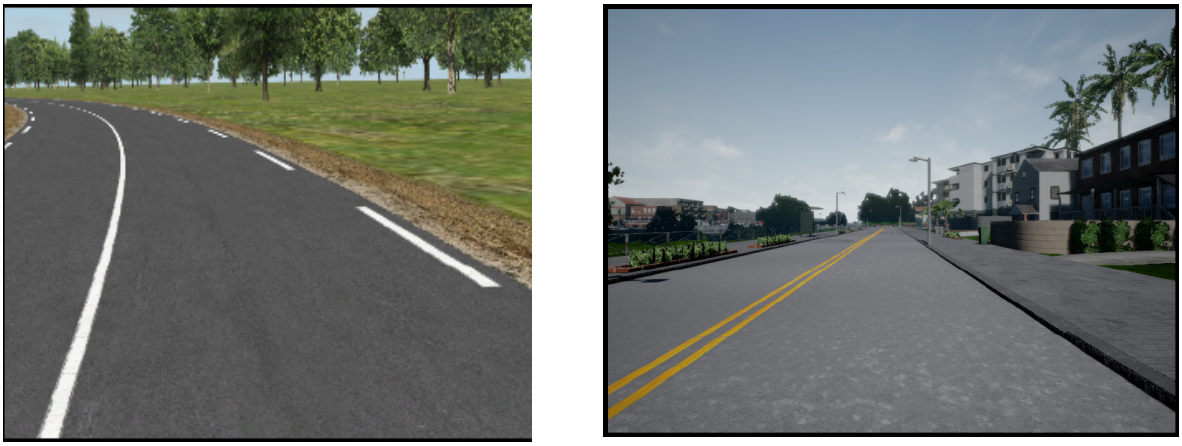
(a) (b)

Figure 3. Maps of the testing routes: (a) first scenario, (b) second scenario

3.2 Second Scenario

The second scenario was related to a city whose map is shown in Figure 3b. Figure 4b shows an example of the city image, which was given as input to the neural network. The learning data were obtained from one driving through several crossroads. In contrast to the first scenario, the second scenario does not predefine the route. Nevertheless, the simulation stops if a distance of more than 2 km has been driven.

The proposed approach was evaluated with the parameter values shown in Table 3. The results show that the built models were able to drive only short routes (see Figure 5). However, it should be noted that due to high time complexity of deep learning, only a small number of generations were executed. More precisely, it took more than 17 days to execute 30 generations on a 3.6 GHz desktop computer with 16 GB RAM. The analysis of the results also shows that the activation function had the most significant effect on the results. It turned out that the majority of models that were able to drive more than 450 m, contained the relu activation function. In addition, the models were able to drive on straight segments, but had issues with crossroads. This is probably due to the relatively simple architecture of neural network. For example, images of the first route (see Figure 4a) are significantly less complex in comparison to the images of the second route (see Figure 4b), since they do not contain any buildings, sidewalks, crossroads, etc. Consequently, more complex architectures of the neural network are needed for the city roads. These can be obtained by optimizing also the topology of the neural network.



(a) (b)

Figure 4. Examples of the input images: (a) first scenario, (b) second scenario

Parameter	Values
image multiplier	{1, 3, 5}
noise added to output	{0, 0, 1}
ip image	{true, false}
batch size	40 (not tuned)
number of epochs	100 (not tuned)
activation function	elu (not tuned)
kernel regulizer	none (not tuned)

Table 1. Parameter values for the first scenario

Noise added to output	Image multiplier	Flip image	Epochs to feasible solution
0	3	*true*	30
0:1	3	*true*	30
0	5	*true*	18
0:1	5	*true*	16

Table 2. Tuned parameter values of feasible solutions for the first scenario

Parameter	Values
batch size	{20, 40, ... , 200}
number of epochs	{10, 20, ... , 50}
image multiplier	{1, 3, 5, 7}
noise added to output	{0, 0.05, ..., 0.20}
ip image	{true, false}
activation function	{linear, elu, relu}
kernel regulizer	{none, l2(0.001)}

Table 3. Parameter values for the second scenario

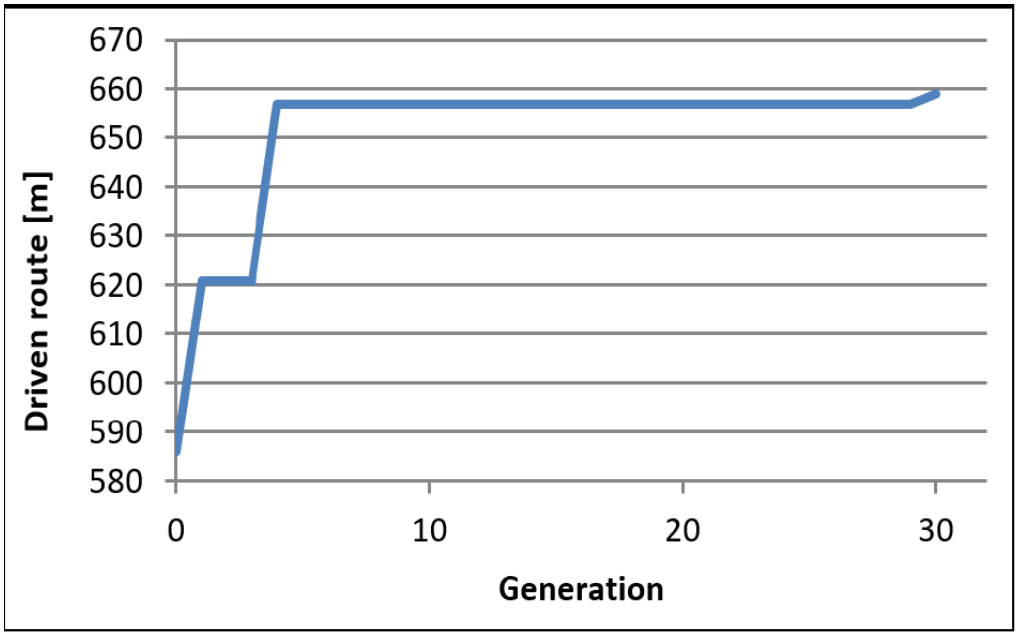


Figure 5. Length of the feasible route through generations for the second scenario

4. Conclusions

This paper presented an optimization approach for tuning end-to-end deep learning that builds human-like driving models. This approach aims at learning good driving models when a low quantity of learning data is available. It was evaluated with one neural network architecture on two routes: a circular route and a city route. The results show that this approach was able to find driving models for the circular route, but did not manage to find driving models for handling crossroads inside the city.

Future work will focus on determining the most appropriate neural network architecture for urban environments. In addition, the efficiency of the evolutionary process needs to be increased by, for example, introducing parallelism in the model learning. Furthermore, the behavior of the obtained driving models will be compared to human driving behavior to determine how well the models reproduce human driving. In case of unacceptable reproduction, these models with be enhanced to obtain driving models that are able to imitate human driving.

5. Acknowledgments

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2- 0209, and research project \Multiobjective discovery of driving strategies for autonomous vehicles", funding No. Z2-7581). This work was also partially funded by NERVteh, raziskave in razvoj, d.o.o., and is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 692286.

References

[1] Angkititrakul, P., Miyajima, C., and Takeda, K. (2011). Modeling and adaptation of stochastic driver-behavior model with application to car following. *In*: Proceedings of the *IEEE Intelligent Vehicles Symposium IV,* p. 814-819, 2011.

[2] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K. (2016). End to end learning for self-driving cars. arXiv, 1604.07316.

[3] Chen, Z., Huang, X. (2017). End-to-end learning for lane keeping of self-driving cars. *In:* Proceedings of the IEEE Intelligent Vehicles Symposium IV, p 1856-1860, 2017.

[4] Eiben, A. E., Smith, J. E. (2015). Introduction to Evolutionary Computing (2nd ed.). Springer, Berlin.

[5] Eraqi, H. M., Moustafa, M. N., Honer, J. (2017). End-to-end deep learning for steering autonomous vehicles considering temporal dependencies. CoRR, abs/1710.03804.

[6] Fernando, T., Denman, S., Sridharan, S., Fookes, C. (2017). Going deeper: Autonomous steering with neural memory networks. *In*: Proceedings of the IEEE International Conference on Computer Vision Workshops, p. 214-221, 2017.

[7] Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Martinez-Gonzalez, P., Garcia-Rodriguez, J. A. (2018). Survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70. 41-65, 2018.

[8] Mikami, K., Okuda, H., Taguchi, S., Tazaki, Y., Suzuki, T. (2010). Model predictive assisting control of vehicle following task based on driver model. *In*: Proceedings of the IEEE Conference on Control Technology and Applications, p. 890-895.

[9] Patterson, J., Gibson, A. (2017). Deep Learning: A Practitioner's Approach. O'Reilly, Sebastopol, 2017.

[10] Tavcar, A., Kaluza, B., Kvassay, M., Schneider, B., Gams, M. (2014). Surrogate-agent modeling for improved training. *In:* Proceedings of the Twenty-First European Conference on Artificial Intelligence (ECAI), p. 1103-1104.

[11] Tran, Q., Firl, J. (2017). Modelling of trac situations at urban intersections with probabilistic non-parametric regression. *In:* Proceedings of the IEEE Intelligent Vehicles Symposium IV, p. 334-339, 2013.

[12] Trontelj, K., Cegovnik, T., Dovgan, E., Sodnik, J. (2017). Evaluating safe driving behavior in a driving simulator. *In:* Proceedings of the 7th International Conference on Information Society and Technology, p 299-302, 2017.

[13] Wei, D., Liu, H. (2013). Analysis of asymmetric driving behavior using a self-learning approach. *Transportation Research Part B: Methodological*, 47:1-14, 2013.

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

[15] Xu, L., Hu, J., Jiang, H., Meng, W. (2015). Establishing style-oriented driver models by imitating human driving behaviors. *IEEE Transactions on Intelligent Transportation Systems*, 16 (5) 2522-2530, 2015.