End-to-End Driving in a Realistic Racing Game with Deep Reinforcement Learning

Etienne Perot
Valeo
etienne.perot@valeo.com

Maximilian Jaritz
Valeo/Inria
maximilian.jaritz@valeo.com

Marin Toromanoff
Valeo
marin.toromanoff@valeo.com

Raoul de Charette
Inria
raoul.de-charette@inria.fr

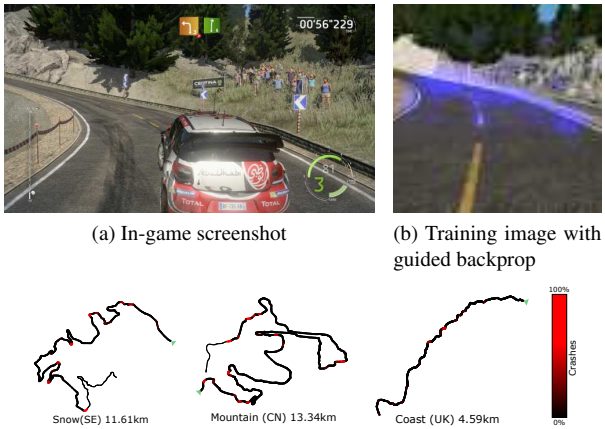
1. Introduction

For autonomous driving the classical paradigm is to use a chain of perception, planning and control; but recent deep learning progress lets foresee an end-to-end alternative to map sensor inputs directly to low-level control of robots [4]. End-to-end driving [5] was showcased in the car racing game TORCS using Reinforcement Learning but its physics and graphics lack realism.

We propose a method benefiting from latest asynchronous learning [5] to train an end-to-end agent in the context of a realistic car racing game - World Rally Championship 6 (WRC6). We do not rely on the in-game score and train solely on image and speed to learn the optimal action while reflecting real driving conditions. Our architecture was trained simultaneously on tracks with different graphics and road structure (cf. fig. 1 and 3). Compared to previous use of TORCS [5, 1, 3], the environment exhibits more realistic physics (grip, drift), graphics (illuminations, animations, etc.), and a variety of environments (sharp turns, slopes, cliffs, snow, etc.). The proposed reward function converges faster than previous ones and offers some generalization capacity. Additionally, the driving style is more comparable to human driving.

2. Method

We used the asynchronous advantage actor-critic (A3C) [5] to train an end-to-end neural network. Every time-step, the algorithm receives the state of the game, acts (acceleration and steering), and gets a reward as supervision signal. This method optimizes driving policy using only RGB image as input (cf. fig. 1b) in order to maximize the cumulated reward. The choice of the A3C baseline is justified by its top performance and because it allows training without any need of experience replay for decorrelation.



(a) In-game screenshot

(b) Training image with guided backprop

(c) Performance after training

Figure 1: The racing environment for our end-to-end driving architecture. (a) The full render in WRC6 game. (b) The 84x84 network input with guided back-propagation (blue overlay). Note, the narrow field of view and removal of indicators (turns signs, scores, etc.). End-to-end performance after training is displayed over tracks in (c).

State encoder. Unlike other computer vision tasks a shallow CNN is sufficient as car racing relies mostly on road detection. Our CNN + LSTM architecture is similar to [2] though using a dense filtering (stride 1). It also uses max pooling to allow more translational invariance and takes advantage of speed and previous actions in the LSTM.

Reward shaping. To help the network to converge to the optimal set of solutions the reward shaping is crucial. The reason not to use the in-game score as reward is that the latter is too sparse to train the agents. In Mnih et al. [5] the reward R is computed with the angle difference θ between the car's heading and the road, and the speed v. Though

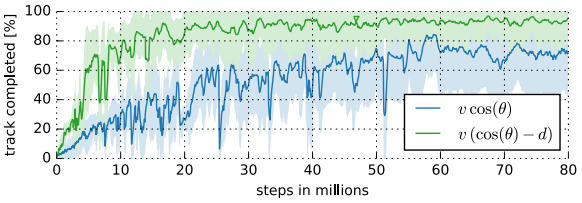


Figure 2: Proposed reward (green) versus reward from [5] (blue). For each reward the rolling mean (dark) and standard deviation (light) are shown (rolling uses 200 steps).

Figure 3: Guided back propagation (blue highlights) of CNN+LSTM architecture after 190 mega steps. Despite the various scene and road appearances the network learned to detect road edges and to rely on the later for control.

efficient, it is limited as it does not prevent the car to slide along the guard rail since the latter follows the road angle. Instead we chose to add the distance from the middle of the road d as a penalty, that is: $R = v(\cos\theta - d)$.

3. Experiments

We ran the algorithm with 15 asynchronous agents. Each agent communicates via TCP with a WRC6 instance through a dedicated API specifically developed. It allows us to retrieve in-game info, compute the reward and send control back to the game. For computational reasons, costly graphics effects were removed and the horizontal field of view reduced. The game engine's clock runs at 30FPS and the physical engine is on hold as it waits for the next action.

Reward Comparison To evaluate the proposed reward against the reward from [5], we trained a network with their architecture and plot the performance in fig. 2. Compared to [5] (blue curve), the proposed reward (green) converges faster (80% track completion after only 15 mega steps) while driving faster and safer. After 80 mega steps, the proposed reward drives at 88.0 km/h (i.e. +5.1km/h) and crashes 0.9 times per km (i.e. -5.3 crashes per km). The explanation is that with the previous reward the car tends to slide along the guard rail which slows it down and is more dangerous. Qualitatively also, the proposed reward leads to a smoother driving style.

Performance and generalization A deeper network was trained for 190 mega steps, with three very different tracks (5 instances of each). The tracks contain sharp curves,

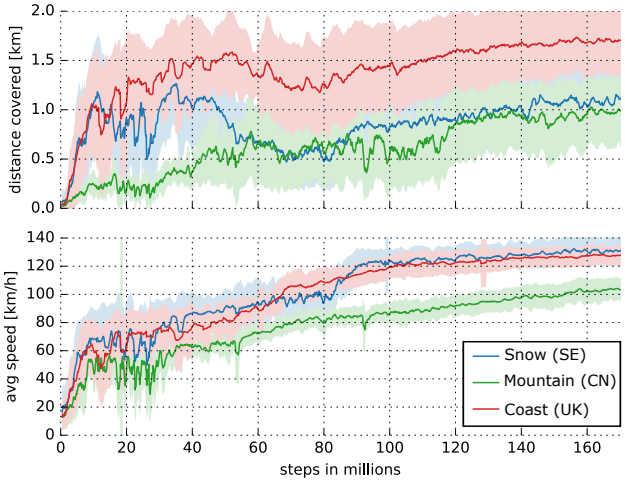


Figure 4: Performance for the challenging training tracks. The agent had more difficulty to progress on the *mountain* track as it exhibits sharp curves and hairpin bends.

cliffs, snow, etc. Physics also differ, especially road adherence. Guided back propagation is displayed in fig. 3 (i.e only positive inner gradients that lead to the chosen action). Despite the various scenes appearance, it learned to detect and use road edges/curvature as a strong control cue. This actually mimics classical approaches that also use lane markings for lateral controls. The training performance is also shown in fig. 4 and demonstrates that it progressed well in each track. After 190 mega steps the agent learned to drive in *mountain*, *snow* and *coast* tracks, to take some sharp turns and even hairpin bends. This is visible in fig. 1c that depicts crash locations along the tracks. We also tested the generalization of this training on unseen test tracks. It exhibits generalization capabilities as the bot was able to drive on new tracks to some extent.

An online video illustrates the performance over trained and new tracks: `https://youtu.be/e9jk-lBWFlw`

References

[1] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of IEEE ICCV*, pages 2722–2730, 2015.

[2] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. *arXiv:1605.02097*, 2016.

[3] B. Lau. Using Keras and Deep Deterministic Policy Gradient to play TORCS, 2016.

[4] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[5] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of ICML*, 2016.