Formula RL

Deep Reinforcement Learning for Autonomous Racing using Telemetry Data

Adrian Remonda, Sarah Krebs, Eduardo Veas, Granit Luzhnica, Roman Kern
arXiv:2104.11106v2, 2022

Notation	Description
θ	Angle between the car direction and the direction of the track axis or racing line.
track	Vector of 19 range finder sensors: each sensor returns the distance between the track edge and the car within a range of 200 m.
trackPos	Distance between the car and the track axis or racing line.
V_x	Speed of the car along its longitudinal axis.
V_y	Speed of the car along its transverse axis.
V_z	Speed of the car along its z-axis.
$\vec{\omega}$	Vector of 4 sensors representing the rotation speed of the wheels.
f_{rot}	Number of rotations per minute of the car engine.
LAC	Look ahead curvature. Vector of 4 curvature measurements from the racing line at 20, 40, 60 and 80 meters ahead. The curvature is recorded from a previous slow lap.

Table 1: *Left:* Telemetry features used as input [11]. *Right:* Tracks used for evaluation [20]: Michigan (Top), Forza (Middle), Aalborg (Bottom). Note that LAC is not used in the 1st and 2nd part of Study 1.

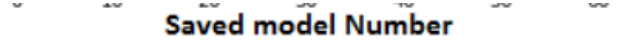


Figure 2: Fastest laps for all PER1M models trained on Aalb
tested on Aalborg (Blue), Michigan (Brown), and Forza (Gr

5.2 Episode Termination

Yoshitaka Otsuka

Paper Presentation

Background: Why Autonomous Racing?

Racing is a useful but difficult vehicle-control problem.

Passenger autonomous driving

Main goal: drive safely and reliably in traffic.

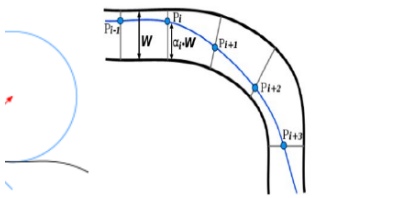
Autonomous racing

Main goal: minimize lap time while staying on track.

Why difficult?

High-speed driving requires precise braking, steering, throttle, and racing-line choice near the physical limits of the car.

- Traditional control methods require heuristics and manual tuning.
- Each track has different corners and therefore different challenges.
- Reinforcement learning can learn by trial and error in simulation.



metrical representation of the radious of curva-
ine representation [3]

ng the track axis.

the **curvature** κ of the curve C at a given
defined by the inverse of the curvature ra-

3 Algorithms

3.1 Deep Deterministic Policy Gr

By combining the insights of DQN wit
terministic policy gradient algorithm, D
solving a wide variety of continuous c
utilizes an actor function $\mu(s|\theta^\mu)$, speci
icy, and a critic function $Q(s, a|\theta^Q)$, b
neural networks. At each step, based o
the agent chooses an action according to
with a noise process $\mathcal{N}$ to allow for exp
a reward r_t and a new state s_{t+1} from t
observed transitions (s_t, a_t, r_t, s_{t+1}) a
buffer. At each step, a minibatch of N tra

Paper Overview

What the authors try to answer

Input

Vehicle telemetry data
—not raw images

Output

Continuous actions:
steering, throttle, brake

Goal

Fast lap time + no damage
across known and unknown tracks

Research Questions

- RQ1: Can a racing driver model be learned from telemetry data only?
- RQ2: Can a model trained on one track generalize to unseen tracks?

Reinforcement Learning Formulation

Agent learns by interacting with TORCS simulator

State

Telemetry features: speed, angle, track position, range sensors, wheel rotation, engine rpm, and optional LAC.

Action

Continuous control values for steering, acceleration/throttle, and brake.

Reward

Encourages forward speed along the racing line and penalizes deviation from track axis / racing line.

Important point

- Driving cannot be represented well by only discrete actions like left / right.
- DDPG is used because it can handle continuous action spaces.
- The car interacts with the simulator every 200 ms.

Telemetry Data as Input

Why the paper avoids raw image input

Notation	Description
θ	Angle between the car direction and the direction of the track axis or racing line.
track	Vector of 19 range finder sensors: each sensor returns the distance between the track edge and the car within a range of 200 m.
trackPos	Distance between the car and the track axis or racing line.
V_x	Speed of the car along its longitudinal axis.
V_y	Speed of the car along its transverse axis.
V_z	Speed of the car along its z-axis.
$\vec{\omega}$	Vector of 4 sensors representing the rotation speed of the wheels.
f_{rot}	Number of rotations per minute of the car engine.
LAC	Look ahead curvature. Vector of 4 curvature measurements from the racing line at 20, 40, 60 and 80 meters ahead. The curvature is recorded from a previous slow lap.

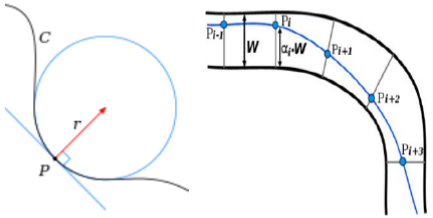
Table 1: *Left:* Telemetry features used as input [11]. *Right:* Tracks used for evaluation [20]: Michigan (Top), Forza (Middle), Aalborg (Bottom). Note that LAC is not used in the 1st and 2nd part of Study 1.

- Images are rich, but expensive to render and train from.
- Telemetry is already available in racing cars and simulators.
- Telemetry directly describes vehicle dynamics: speed, angle, wheel rotation, engine rpm.
- LAC = look-ahead curvature; it gives future information about the racing line.

Takeaway: telemetry makes the task faster and focuses learning on vehicle control.

Racing Line and Look-Ahead Curvature

The model uses a racing line as a loose guide



1: *Left:* Geometrical representation of the radious of curva-
ght: Racing line representation [3]

omputed along the track axis.

ure 1 shows the **curvature** κ of the curve C at a given
P in track is defined by the inverse of the curvature ra-
r) at that point, and it's given by $\kappa = \frac{\partial\theta}{}$ where κ is

3 Algorithms

3.1 Deep Deterministic Policy Gradient (DI

By combining the insights of DQN with the actor-
terministic policy gradient algorithm, DDPG [10] a
solving a wide variety of continuous control tasks
utilizes an actor function $\mu(s|\theta^\mu)$, specifying the cu
icy, and a critic function $Q(s, a|\theta^Q)$, both approxi
neural networks. At each step, based on the current
the agent chooses an action according to $a_t = \mu(s_t|$
with a noise process $\mathcal{N}$ to allow for exploration, an
a reward r_t and a new state s_{t+1} from the environm
observed transitions (s_t, a_t, r_t, s_{t+1}) are stored in
buffer. At each step, a minibatch of N transitions is u

Racing line

Reference trajectory for fast driving. It is not blindly followed; the model can improve it.

Look-ahead curvature (LAC)

Curvature measurements ahead of the car. This helps the agent prepare for upcoming corners.

- Without future track information, braking and corner entry are difficult.
- LAC improves lap times in the experiments.

Experimental Setup

Two studies: learning to drive and generalization

Notation	Description
θ	Angle between the car direction and the direction of the track axis or racing line.
track	Vector of 19 range finder sensors: each sensor returns the distance between the track edge and the car within a range of 200 m.
trackPos	Distance between the car and the track axis or racing line.
V_x	Speed of the car along its longitudinal axis.
V_y	Speed of the car along its transverse axis.
V_z	Speed of the car along its z-axis.
$\vec{\omega}$	Vector of 4 sensors representing the rotation speed of the wheels.
f_{rot}	Number of rotations per minute of the car engine.
LAC	Look ahead curvature. Vector of 4 curvature measurements from the racing line at 20, 40, 60 and 80 meters ahead. The curvature is recorded from a previous slow lap.

Table 1: *Left:* Telemetry features used as input [11]. *Right:* Tracks used for evaluation [20]: Michigan (Top), Forza (Middle), Aalborg (Bottom). Note that LAC is not used in the 1st and 2nd part of Study 1.

Simulator

TORCS: The Open Racing Car Simulator

Tracks

Michigan: simple oval-like
Forza / Aalborg: complex tracks

Study 1

Train and test models to see whether RL can learn fast driving from telemetry.

Study 2

Train on one track and test on unseen tracks to evaluate generalization.

Metrics

Best lap time, average lap time, and damage.

Study 1 Results: Can RL Learn to Race?

PER1M with racing line and LAC performs best

ritic network with window sizes 4 and 8. *MS2, MS3, MS4* ses multi-step targets with 2, 3, and 4 steps. *PER40k, PER1M* tilize PER with buffer sizes 4×10^4 and 10^6.

.2 Study 1: Learning to Drive

his study addresses RQ1 by examining whether RL models an drive a racing car from telemetry data and then comparavely evaluating the performance across different models.

Procedure. This study was split in three parts. To reduce ie training time for hyperparameter selection, part1 uses a mple track, Michigan and trained for 500 episodes. The arned model was *tested* without exploration ($\epsilon = 0$) on the ime track the results were used to select the hyperparametrs or each of the four used algorithms. In part2, the four algothms (with selected hyperparameters) and tested on a techically more complex track, Aalborg, using both MOT and ita bot as racing line references. With the selected best alorithm of part2, in part3 we evaluated the impact of adding ture information of the track. We did so by adding the look ead curvature (LAC) and training in all three tracks. Part 2 nd 3 were trained for 7000 episodes. The results were used select the best algorithm/model suited for the task of au-

	MOT Reference		RC Referen			
	bLT	aLT	bLT	aLT	bLT	aL
WIN1	**26.75**	27.16	-	-	-	-
WIN4	26.77	**26.96**	74.75	75.73	67.56	**69.**
WIN8	26.75	35.61	-	-	-	-
MS2	26.79	30.33	-	-	-	-
MS3	26.80	27.13	-	-	-	-
MS4	26.83	27.04	77.68	78.53	69.94	77.
PER40k	26.84	34.85	-	-	-	-
PER1M	27.06	32.30	71.76	75.22	**67.17**	70.
LSTM4	27.35	27.52	85.87	90.05	-	-
LSTM8	27.50	27.85	-	-	-	-
Tita	28.57	-	-	-	68.11	-
Genetic	33.86	-	-	-	69.92	-

Table 2: Testing results over 10 runs in Michigan and over 5 r in Aalborg. Best lap-time (**bLT**), average best lap-time (**a** (in seconds) for models with 0 damage. The Aalborg track trained/tested using both middle of the line (**MOT**) and racing from Tita (**RC**) bot as racing line references.

Procedure. For this study, we use different variants PER1M (MOT, RC, RC+LAC) as shown in Table 4. models are trained in one track (Michigan or Aalborg) tested in two other unseen tracks. While training, as

	Forza	Michigan	Aalb
Tita	77.39	28.57	68.
PER1M MOT	-	27.060	71.7
PER1M RC Reference	75.306	26.906	67.
PER1M RC Reference + LAC	**73.018**	**26.278**	**63.**

Table 3: Lap times (best) of a comparison between the best of the considered different approaches in Study 1. **Tita** vs l

- RL models outperform the baseline handcrafted bot on Michigan.
- On Aalborg, models using a racing-line reference perform better.
- PER1M is the best algorithm in the complex track.
- Adding LAC gives the best lap times.

Study 2 Results: Generalization to Unseen Tracks

Training on a complex track helps transfer

physics vehicle dynamics, which is important since we intended to drive the car at its physical limits.

The results of Study 1 supports *RQ1*: Deep RL algorithms effectively learn to drive fast from telemetry data and obtain models better than state-of-the-art handcrafted models. The results also showed that PER1M (prioritized experience replay with a replay buffer size of 1M samples) was the best performing algorithm in a complex track. Most importantly, the results evidenced that our proposed look ahead of the curve approach (LAC) improves the performance of models in the self racing scenarios. To the best of our knowledge, these results constitute a contribution to the line optimization

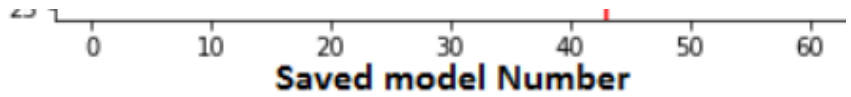


Figure 2: Fastest laps for all PER1M models trained on Aalt
tested on Aalborg (Blue), Michigan (Brown), and Forza (Gr

5.2 Episode Termination

We compared two different settings for good episode te

- Models trained only on simple Michigan did not finish complex unseen tracks.
- Models trained on complex Aalborg could finish unseen tracks.
- However, unseen-track lap times are worse than track-specific models.
- This resembles human racing: even experts practice each specific track.

What Did the Agent Learn?

Learned behavior looks like real driving control

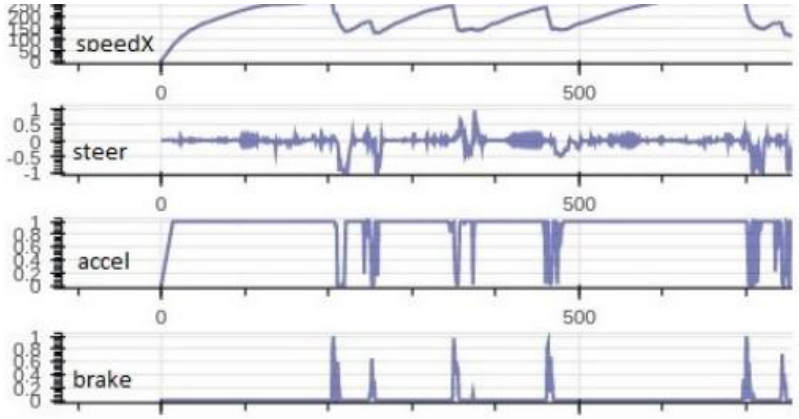


Figure 3: Outputs (steer, acceleration, brake) of a model trained with

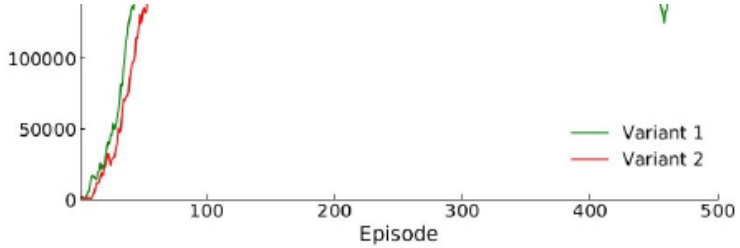


Figure 4: Training episode reward for normal end-of-episode (green) and **AT** (red) transitions. The reward is averaged over 10 training runs with different random seeds, smoothed with a moving average over 5 episodes. **WIN1** with episode terminations caused by reaching the maximum number of steps

- The model learns to avoid pressing throttle and brake at the same time.
- After braking, it waits before strong steering to reduce oversteer.
- The adopted target modification improves training reward.

Contributions and Limitations

What is useful, and what remains difficult

Contribution 1

Shows that deep RL can learn fast autonomous racing from telemetry data.

Contribution 2

Compares DDPG variants and finds PER1M effective in complex tracks.

Contribution 3

Shows that LAC improves performance and that complex-track training helps generalization.

Limitations

- Experiments are simulator-based, not real racing cars.
- The model relies on telemetry and a racing-line reference, not pure vision.
- Generalization is limited; best performance still requires track-specific training.

Conclusion

Main message of the paper

- Autonomous racing is a challenging continuous-control problem.
- Telemetry data provides a practical and lightweight input representation.
- Deep RL models can outperform handcrafted bots in simulation.
- PER1M + racing-line reference + LAC achieved the best results.
- Models can generalize to unseen tracks, but performance is still lower than track-specific training.