Backseat driving: Using privileged-information auxiliary losses in autonomous racing

Jacques Friedrich Amsel · Supervisor: Dr JC Schoeman · Department of Electrical and Electronic Engineering

Stellenbosch UNIVERSITY IYUNIVESITHI UNIVERSITEIT

The problem

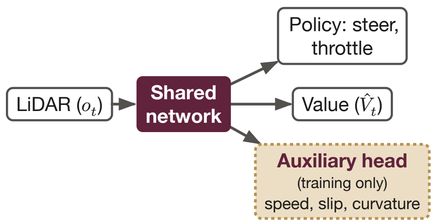
An end-to-end reinforcement-learning agent can learn to race straight from **LiDAR**. To drive well, it must work out things it cannot see: its speed, how much its tyres slip, and how sharply the track **curves ahead**.

In an RL agent, the reward is a weak, noisy teacher for these. In simulation, though, they are known exactly: they are **privileged information**.

> **Question:** can learning to predict privileged information make a LiDAR-only racer learn faster and drive better?

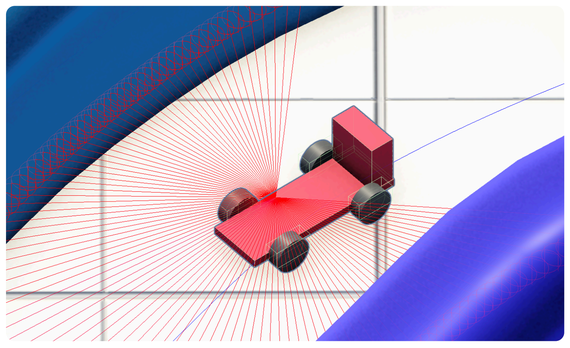
The idea: a backseat driver

During training, an **auxiliary head** predicts the privileged quantities from the agent's LiDAR observations. Its loss is added to the RL loss, so it shapes what the shared network learns. At race time the head is removed and the car drives from LiDAR alone.



$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{PPO}} + \sum_c w_c \, \mathcal{L}^{(c)}_{\text{aux}}$$

The experiment



Custom Unity simulator with physics calibrated to the 1:10-scale RoboRacer (after Samak & Samak) and procedurally generated tracks. Red: LiDAR rays.

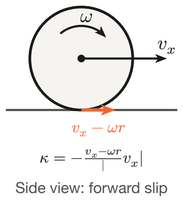
- **Two agents** trained with PPO: **feedforward** (50 stacked scans) and **recurrent** (GRU memory).
- **Each target group switched on alone** (plus all together).
- **3 loss weights** ($w_c = 0.1, 0.3, 1$) × **6 seeds** each, 600 PPO iterations (≈ 7.4 M steps) per run, tested on unseen tracks.
- **Linear probes** measure how readable each quantity is from the network's features (R^2).

What the head predicts

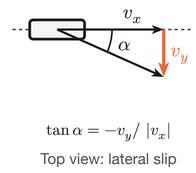
Ten target groups, each normalised to zero mean and unit variance:

Group	Targets	Units
Forward velocity	v_f	m/s
Lateral velocity	v_l	m/s
Yaw rate	ω_y	rad/s
Curvature, close	$c_d,\ d = 0.5, 1, \ldots, 3$ m ahead	1/m
Curvature, medium	$c_d,\ d = 3.5, \ldots, 6$ m	1/m
Curvature, far	$c_d,\ d = 6.5, \ldots, 8$ m	1/m
Front forward slip	κ, front wheel pair	–
Rear forward slip	κ, rear wheel pair	–
Front lateral slip	$\tan\alpha$, front wheel pair	–
Rear lateral slip	$\tan\alpha$, rear wheel pair	–
All	every target above	

Wheel slip measures how far a tyre is from rolling freely. It is 0 for a free-rolling wheel; it grows under wheelspin, braking, or when the wheel is dragged sideways in a corner. A tyre needs some slip to grip, but past a peak it slides and loses grip.



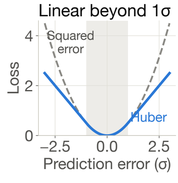
Side view: forward slip

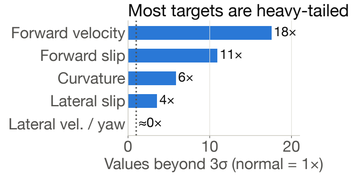


Top view: lateral slip

Why a Huber loss?

Most targets are **heavy-tailed**: forward velocity has 18× more values beyond 3σ than a normal distribution would. A squared-error loss would let these rare extremes dominate the gradient. The **Huber loss** (β = 1) is quadratic for errors within 1σ, where most of the data lies, and linear beyond, so its gradient is bounded.



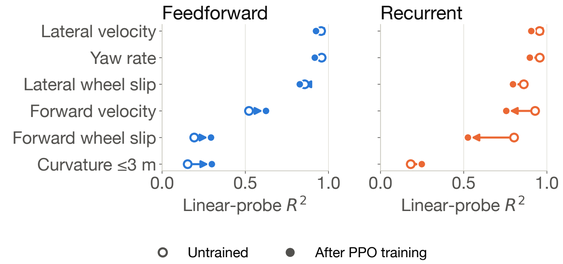


Limitations and future work

- Simulation only, with tracks from a single generator.
- 6 seeds, and PPO settings tuned on the feedforward agent, which may disadvantage the recurrent one.
- Linear probes only see linearly encoded information.
- **Next:** more recurrent seeds; compare against asymmetric critics and teacher–student distillation; test on a physical RoboRacer.

1 Baseline's feature encodings

Even untrained networks encode lateral motion, which follows almost linearly from the steering input. Plain RL training **trades this away** for information about the **upcoming bend**.



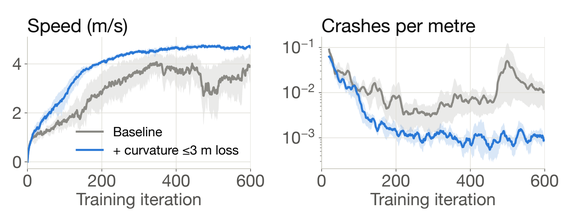
→ The policy learns curvature anyway, so we tried **supervising it directly**.

2 Predicting the next 3 m helps

Close-ahead curvature (≤ 3 m) is the only target that improved **both** agents. A follow-up with 6 fresh seeds confirmed it for the feedforward agent:

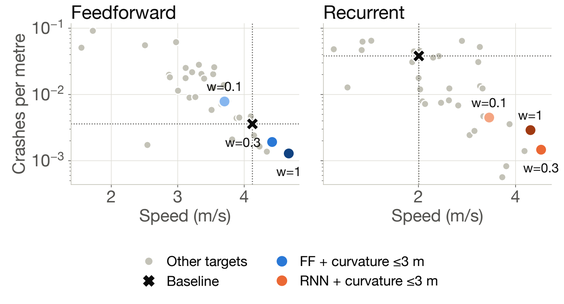
+30% faster (progress per second)

–93% crashes per metre



Feedforward follow-up: mean of 6 seeds, 95% bootstrap interval. Two-sided *t*-test, BH-adjusted $p \approx 0.036$ for both metrics.

The follow-up baseline drew weaker seeds; the screening estimate (**+13%** speed, **–64%** crashes) is likely closer to the true effect. The recurrent agent shows the same trend (+53% speed, –95% crashes) but it is not significant ($p \approx 0.17, 0.20$).



Screening: every configuration, averaged over its last 50 iterations. Below and right of the dotted lines is faster *and* safer than the baseline.

3 RNN vs feedforward

The recurrent baseline is half as fast as the feedforward one (2.0 vs 4.1 m/s) and crashes 11× more often. Auxiliary losses mostly **rescue** this struggling agent, and mostly **hurt** the feedforward one:

26 / 33 recurrent setups learned faster

4 / 33 feedforward setups learned faster

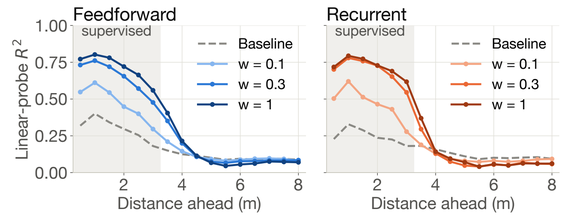
The weight matters too: slip targets help at $w_c = 0.1$ but hurt training at $w_c = 1$.

	Feedforward			Recurrent		
	w = 0.1	w = 0.3	w = 1	w = 0.1	w = 0.3	w = 1
Forward velocity	-21%	-10%	-2%	+35%	+49%	+34%
Lateral velocity	-16%	-14%	-8%	-25%	+38%	+20%
Yaw rate	-7%	-19%	-4%	+50%	+35%	-4%
Curvature ≤3 m	-4%	+14%	+20%	+58%	+113%	+109%
Curvature 3.5–6 m	-6%	-10%	-1%	+15%	+43%	+42%
Curvature 6.5–8 m	-7%	-20%	-10%	+68%	+83%	+81%
Front forward slip	-4%	-28%	-66%	+50%	-38%	-88%
Rear forward slip	-9%	-17%	-71%	+64%	-36%	-88%
Front lateral slip	-18%	-13%	-6%	+32%	+37%	-2%
Rear lateral slip	-15%	-23%	-24%	+18%	-2%	-32%
All targets	+1%	-12%	-59%	+82%	+51%	+6%

Change in learning speed (area under the reward curve) vs each architecture's own baseline. Screening results: trends, not significant after BH correction.

4 Decodable ≠ better driving

Each loss made its own target far more decodable (curvature R^2 up to +0.45 FF, +0.49 RNN), but that alone does not help: **all** targets at $w_c = 1$ gave very decodable features and one of the **worst** feedforward agents.



Conclusions

- Give a LiDAR racer a **close-ahead curvature** auxiliary loss. It is a low-variance signal for something the policy learns anyway, so it is learned **earlier**. And curvature says what the car must do *next*; speed and slip say what it has *already* done.
- **Tune the loss weight**: the same target can help or harm, depending on the weight.
- **Probe a baseline first**: what a plain agent learns by itself made good targets for the feedforward agent.
- Only the feedforward follow-up is statistically significant; the rest are trends.