From Soft Actor–Critic to Teacher–Student Distillation: An Engineering Case Study of a Neural Racing Agent for TORCS

Team Overclocked
IBM AI Racing League — Politechnika Świętokrzyska
hqoverclocked@gmail.com

Abstract

We document the end-to-end development of a neural-network driving agent for the Corkscrew circuit (Laguna Seca, 3,602 m) in TORCS, interfaced through the Simulated Car Racing (SCR) UDP protocol at 50 Hz. The project progressed through five stages: (S0) exploratory Soft Actor–Critic (SAC) baselines over several MLP capacities; (S1) reward shaping and RPM-based automatic gear control; (S2) a two-stage observation curriculum with LayerNorm-stabilised actor–critic networks, which exposed a critic divergence at 4.5×10^5 environment steps; (S3) residual reinforcement learning on a frozen behavioural-cloning base, during which a five-layer cascade of failures (exploration-noise hold time, anchor-induced variance collapse, and a physics/backpropagation desynchronisation) was diagnosed and resolved; and (S4-F) the final pipeline, in which an analytic teacher controller with 54 tunable parameters was optimised by a distributed Optuna search (∼15,000 trials across studies) and subsequently distilled into a 109 k-parameter multilayer perceptron via behavioural cloning and DAgger. The distilled network, `bc_v6.pth` (429 KB), scored a **92.04 s** standing-start lap under submission conditions (fuel and damage enabled) with a top speed of 250 km/h. A residual reinforcement-learning policy trained on top of this frozen network reaches the same ∼92 s class of lap, so the result is reproduced by reinforcement learning and does not rest on imitation alone. We report the full problem formulation, network architectures, reward function, training infrastructure, failure analyses with telemetry evidence, and the quantitative progression from a 118.8 s plateau to the final agent, and we discuss why a teacher–distillation hybrid outperformed pure model-free reinforcement learning under the project's wall-clock constraints.

Contents

1 Introduction

TORCS (The Open Racing Car Simulator) with the SCR server patch [1, 2] is a standard benchmark for sensorimotor control research: a client receives a structured telemetry packet over UDP every 20 ms (50 Hz) and must respond with actuator commands within the same tick, otherwise the physics engine reuses the previous command. The task studied here is the one scored by the IBM AI Racing League: minimise the time of a *standing-start* lap of the Corkscrew circuit (a model of Laguna Seca; nominal length 3,602 m, measured in-simulation at ∼3,608 m per lap) with the `car1-ow1` open-wheel car. The deliverable must be a neural network.

The circuit is a demanding target for learned control. It combines two long acceleration zones with a sequence of tight, blind corners — most notably the eponymous Corkscrew chicane near the 2,400 m mark, taken at ∼56–62 km/h, immediately after a ∼248 km/h approach (Fig. 7). A competitive agent must therefore master late, straight-line braking, traction-limited corner exits, and a consistent racing line, all through a 20 ms control loop.

This report is written as an engineering case study rather than a purely algorithmic contribution. Its purpose is threefold: (i) to document the complete method that produced the submitted agent; (ii) to preserve the quantitative record of intermediate stages, including unsuccessful ones, with telemetry evidence; and (iii) to analyse *why* the final architecture — an Optuna-tuned analytic teacher distilled into a small MLP — outperformed the pure reinforcement-learning (RL) approaches attempted earlier, given a bounded compute and wall-clock budget.

Contributions. In summary, the project delivers:

1. a Gymnasium-compatible TORCS environment with a normalised 32/68-dim observation model, a 2-dim continuous action interface, shared driver-assistance post-processing (automatic gearbox, traction control, launch assist), and reproducible headless operation on both Windows and Linux/WSL;
2. a stabilised SAC configuration (LayerNorm actor–critic, reward-scale control) with a documented divergence analysis;
3. a diagnosed and repaired residual-RL training stack, including the physics-desynchronisation fix that moves gradient computation off the real-time control path;
4. a distributed hyperparameter-search infrastructure (up to 10 headless TORCS workers per machine, SQLite/PostgreSQL study storage) used to tune a 54-parameter analytic controller; and
5. the final distillation pipeline (behavioural cloning with an anti-copycat noise injection, DAgger with teacher seeding) producing the submitted 429 KB network.

2 Related Work

Off-policy actor–critic methods. Deep Deterministic Policy Gradient (DDPG) [3] established the pattern of an off-policy actor–critic with replay buffers and target networks [4] for continuous control, and was the first algorithm demonstrated on TORCS from low-dimensional telemetry. Soft Actor–Critic (SAC) [5, 6] augments this with a maximum-entropy objective and automatic temperature tuning, and is substantially more sample-efficient and robust to hyperparameters; it was the backbone RL algorithm throughout this project, in the implementation of Stable-Baselines3 [7]. Exploration noise was generated with generalised state-dependent exploration (gSDE) [8], whose sampling frequency turned out to be safety-critical at racing speeds (Section 4.3).

Imitation learning. Behavioural cloning (BC) dates back to ALVINN [9] and remains the simplest way to transfer a controller into a network, but it suffers from covariate shift: compounding errors take the learner off the expert's state distribution. DAgger [10] repairs this by iteratively relabelling learner-visited states with the expert. Codevilla et al. [11] describe the "copycat" (inertia) pathology of BC when past controls are part of the observation — a pathology reproduced exactly in this project and resolved with the noise injection they motivate (Section 4.4).

Combining RL with priors. Residual RL [12] superimposes a learned corrective policy on a fixed base controller, bounding exploration around a known-good behaviour. TD3+BC [13] shows that a single behaviour-anchoring term with an adaptive weight $\lambda = \alpha/\mathbb{E}|Q|$ suffices to stabilise off-policy learning around demonstrations; our BC-anchored SAC actor loss follows this construction. Hyperparameter search over the analytic teacher used Optuna [14], with LayerNorm [15] adopted in all value and policy networks for stability.

3 Problem Formulation and Environment

3.1 Markov decision process

The task is modelled as a discounted MDP $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ sampled at $f_c = 50\,\text{Hz}$. The agent observes a normalised vector $\mathbf{s}_t \in \mathbb{R}^{32}$ (stage 1, empty track) or $\mathbb{R}^{68}$ (stage 2, traffic), emits $\mathbf{a}_t \in [-1, 1]^2$, and receives the shaped reward of Section 3.4. Episodes terminate on the conditions of Section 3.5; $\gamma = 0.99$ throughout.

3.2 Observation model

Raw SCR telemetry is mapped to a fixed-size, approximately unit-range vector (Table 1). Two design choices matter. First, all 19 track-edge rangefinders are retained at full resolution: they are the only inputs from which braking points for blind corners can be inferred. Second, the *previous steering command* is appended (index 31) so that the process remains Markov under the steering-rate dynamics introduced by the driver-aid layer; this feature later interacted with behavioural cloning in a non-obvious way (Section 4.4).

Table 1: Stage-1 observation vector (32 dimensions). Stage 2 appends 36 opponent rangefinders normalised by 200 m (total 68).

Index	Signal	Raw range	Normalisation
0–18	track-edge rangefinders (19 beams, ±45°)	0–200 m	/200, clip [0, 1]
19, 20	v_x, v_y	±300 km/h	/300
21	heading error ψ vs. track tangent	$[-\pi, \pi]$	/π
22	lateral position (`trackPos`)	$\approx [-1.5, 1.5]$	clip $[-1, 1]$
23	engine speed	0–18,700 rpm	/10^4, clip [0, 1]
24	gear	1–6 (SCR range)	/7 (fixed constant)
25–28	wheel angular velocities	0–100 rad/s	/100
29	lap progress (`distFromStart`)	0–3,602 m	/3602
30	current lap time	0–120 s	/120
31	previous steering command	$[-1, 1]$	identity

3.3 Action model and driver aids

The network outputs $\mathbf{a} = (a^{\text{steer}}, a^{\text{ab}}) \in [-1, 1]^2$. A shared post-processing layer (`driving_aids.py`) — applied *identically* during data collection, RL training, evaluation, and submission — converts this into the five-field SCR command:

$$\text{accel} = \begin{cases} a^{\text{ab}} \cdot \kappa_{\text{TCS}} & a^{\text{ab}} \geq 0 \\ 0 & \text{otherwise} \end{cases}, \qquad \text{brake} = \max(0, -a^{\text{ab}}), \tag{1}$$

$$\kappa_{\text{TCS}} = \max\left(0.1,\ \frac{\sigma_0}{\sigma}\right) \text{ if } \sigma = (\omega_{RL}+\omega_{RR}) - (\omega_{FL}+\omega_{FR}) > \sigma_0 = 5 \text{ rad/s, else } 1. \tag{2}$$

Gear selection is a pure-RPM automatic box: upshift above 17,800 rpm (near the 18,700 rpm redline), downshift below 9,000 rpm. The usable range is gears 1–6: although the `car1-ow1` car definition lists a

seventh forward ratio, the SCR command interface accepts only gears $-1 \ldots 6$ (the client resets out-of-range values to neutral), so the seventh ratio is unreachable in practice. Recorded telemetry confirms the car attains its top speed in *fifth* gear at ∼17,200 rpm; sixth is engaged on less than 0.1 % of ticks. A launch-assist overrides steering with a proportional centring term $\delta = \mathrm{clip}(0.30\,\rho, -0.4, 0.4)$ while $v_x < 40$ km/h during the first seconds of a standing start. Reducing the action space to two dimensions — letting the aids own gear selection — was adopted in S1 after three-action variants (with a learned gear signal) wasted a large fraction of exploration on drivetrain mistakes. The consequences of mis-tuning this ostensibly minor component were severe (Section 4.4.1).

3.4 Reward function

The stage-1 (time-trial) reward is a sum of a dominant progress term, a per-step time cost, and small shaping terms:

$$\begin{aligned} r_t \;=\; & \underbrace{\Delta d_t}_{\text{progress}} \;-\; \underbrace{0.1}_{\text{time}} \;-\; 0.05\,\rho_t^2 \;-\; 0.05\,|\psi_t| \;-\; 0.1\,\left|a_t^{\text{steer}} - a_{t-1}^{\text{steer}}\right| \\ & +\; 0.2\,\frac{v_{x,t}}{300}\,\mathbb{1}[\text{straight}] \;+\; \frac{v_{x,t}}{300}\,|\cos\psi_t|\,(1 - |\rho_t|)\,\tau_t \;-\; 0.025\,\min(\Delta\text{dmg}_t, 200) \;+\; r_t^{\text{event}}, \end{aligned} \tag{3}$$

where Δd_t is the per-tick increment of `distRaced` in metres (clamped to $[0, 50]$ against respawn artefacts), ρ is the lateral position, ψ the heading error, $\tau_t = \max\big(0, 1 - \min(\text{track}_{7:12})/150\big)$ a corner-sharpness gate, and $\mathbb{1}[\text{straight}]$ fires when the three central beams all exceed 150 m. Event terms are -1 per off-track step, -10 for driving backwards, and a lap-completion bonus

$$r^{\text{lap}} = 500 + \max(0,\; 90 - T_{\text{lap}}), \tag{4}$$

which converts lap time directly into return. On a clean lap the running terms contribute $\approx +0.8$ to $+1.0$ per tick; a completed ∼88–92 s lap yields an episodic return of roughly 3,600 + 500, consistent with the 6,000–7,000 returns of two-lap episodes observed in Fig. 4. The stage-2 variant adds an opponent-proximity penalty (linear inside 10 m), an overtake bonus (+5 per position gained) and a contact penalty (-5); it is exercised by the multi-car pipeline in `phase2/` and not used for the scored time-trial deliverable.

An earlier reward proportional to instantaneous projected speed ($v_x \cos\psi$) was abandoned in S2: because its integral is (up to shaping) also total distance, it assigns nearly equal return to crawling and racing on a fixed horizon and provides no pressure toward minimum-time behaviour. The formulation (3) makes the same quantity *per-step bounded* while the lap bonus injects the time objective explicitly.

3.5 Termination and safety predicates

Episodes end on: (i) $|\rho| > 1.10$ sustained for more than 30 consecutive ticks (0.6 s grace, allowing kerb use); (ii) $\cos\psi < 0$ (reversed); (iii) cumulative damage > 5,000; (iv) a stuck detector (forward progress < 2 m over 250 ticks); (v) 9,000 ticks (180 s) elapsed; or (vi) UDP timeout. A latent defect in an earlier version of predicate (i) — it also triggered when any *single* rangefinder returned the -1 "no surface within range" sentinel, which legitimately happens mid-hairpin — silently truncated training episodes at the ∼2,400 m hairpin for every agent generation until S4-F, and is analysed in Section 4.4.

4 Methodology: Staged Development

Development proceeded in five stages (S0, S1, S2, S3, S4-F), each retaining the environment interface while revising the learning method. Figure 1 summarises the final pipeline; Table 2 the progression.

Table 2: Development stages. Lap times are best standing-start results on Corkscrew, as recorded in experiment logs and telemetry.

Stage	Method	Networks	Obs./act.	Outcome
S0	SAC baselines, evol. strategies	$256^2{\times}128$ … 512^3	29+/3	no competitive laps
S1	SAC + auto-gear + reward redesign	$3{\times}256$, $400{\times}300$	31/2	∼105–115 s; slow progress
S2	curriculum SAC + LayerNorm + VecNorm.	$256^2{\times}128$	31→67/2	divergence at 445 k steps
S3	residual SAC on frozen DAgger base	$256^2{\times}128$ + base	31/2	stable after 5-bug fix; ∼103 s
S4-F	teacher × Optuna → BC → DAgger	$256^2{\times}128$ (109 k)	32/2	**92.04 s scored**

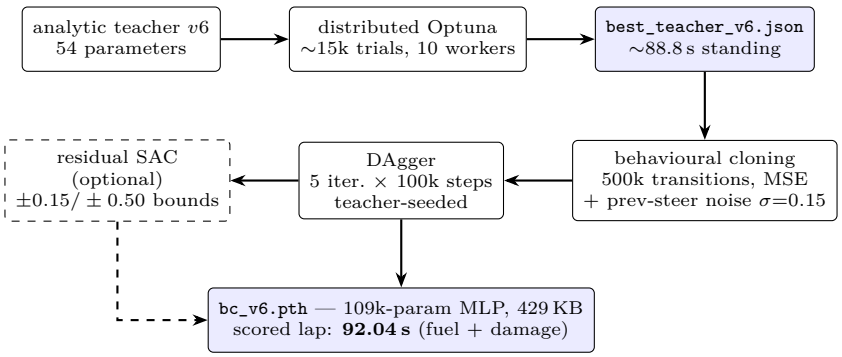

Figure 1: Final S4-F pipeline. A rules-based controller is tuned by a distributed hyperparameter search, then compressed into the deliverable network by behavioural cloning and DAgger; residual SAC optionally refines the frozen distilled policy.

4.1 S0–S1: SAC baselines, gear automation, reward shaping

The project began from a gym-torcs-style wrapper and SAC baselines over MLP capacities 256×256×128, 256×256×256 and 512×512×512, initially with a three-dimensional action that included a learned gear-change signal. Two structural conclusions emerged. First, network capacity was not the binding constraint — all three widths produced similar, uncompetitive driving — so the smallest architecture was retained for all later work. Second, learned gear control was strictly harmful: exploration in the gear channel produced drivetrain faults faster than the critic could learn to avoid them. S1 therefore moved gear selection into the environment as an RPM-scheduled automatic box and reduced the action space to $(a^{\text{steer}}, a^{\text{ab}})$.

S1 also replaced the speed-projection reward with the progress-plus-time-cost form that survived, in refined form, to Eq. (3). Training runs of ∼0.8 M steps (e.g. the 400×300 configuration of Fig. 2, an architecture borrowed from the DDPG literature [3]) achieved stable but slow laps in the 105–115 s range: the agent was safe, centred, and far from the friction limit.

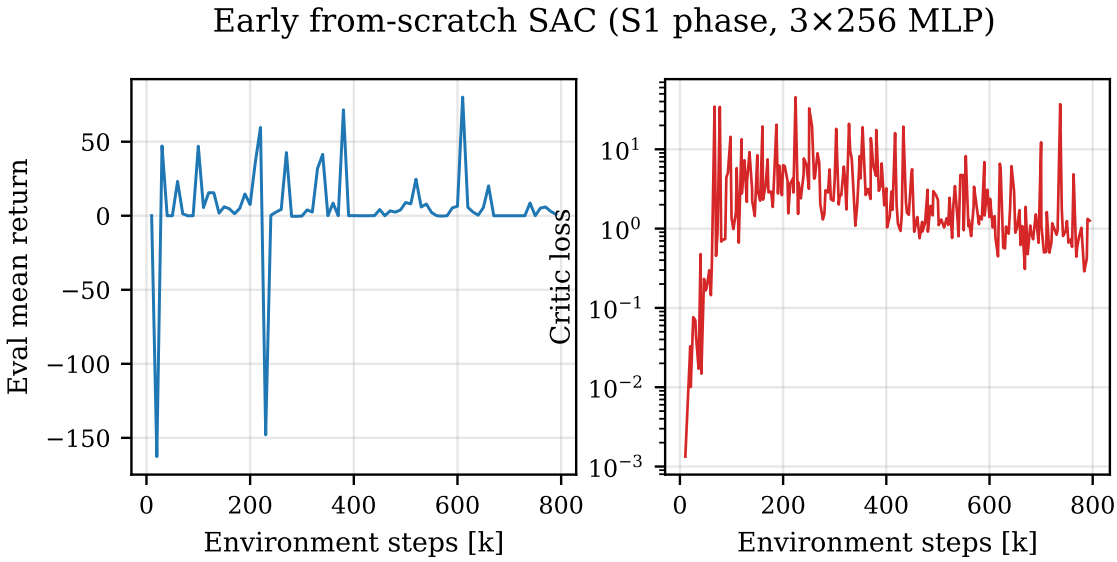

Figure 2: Early from-scratch SAC (S1). Left: evaluation return; right: critic loss. The run is stable but plateaus at cautious driving (∼0.79 M steps shown).

4.2 S2: curriculum, normalisation, and a documented divergence

S2 formalised a two-stage curriculum: stage 1 (31-dim observation, empty track) intended to transfer into stage 2 (67-dim with opponent rangefinders) by input-layer weight expansion. Its lasting contributions are architectural: LayerNorm [15] after every hidden layer of both actor and critic (retained in every subsequent network, including the final deliverable), observation/reward normalisation via a VecNormalize wrapper,

and telemetry infrastructure logging both car state (81 columns per tick) and training statistics (losses, entropy temperature, Q-statistics every 100 steps).

S2 is also the project's clearest negative result. A long stage-1 run diverged after $\approx 4.5 \times 10^5$ environment steps: the critic loss grew from $O(1)$ to $O(10^6)$ over ∼150 k steps while the automatically tuned temperature α rose by two orders of magnitude and the episodic return collapsed below its initial level (Fig. 3; the run terminates at 445 700 steps). The proximate causes identified at the time were an over-long horizon ($\gamma = 0.999$, i.e. an effective horizon of 10^3 ticks = 20 s of racing) interacting with unclipped shaped rewards, allowing bootstrapped Q-targets to grow without bound; entropy auto-tuning then chased the mis-scaled Q landscape. The remedies — $\gamma = 0.99$, reward clipping inside VecNormalize, and LayerNorm value networks — were carried into all later stages, and no comparable divergence recurred.

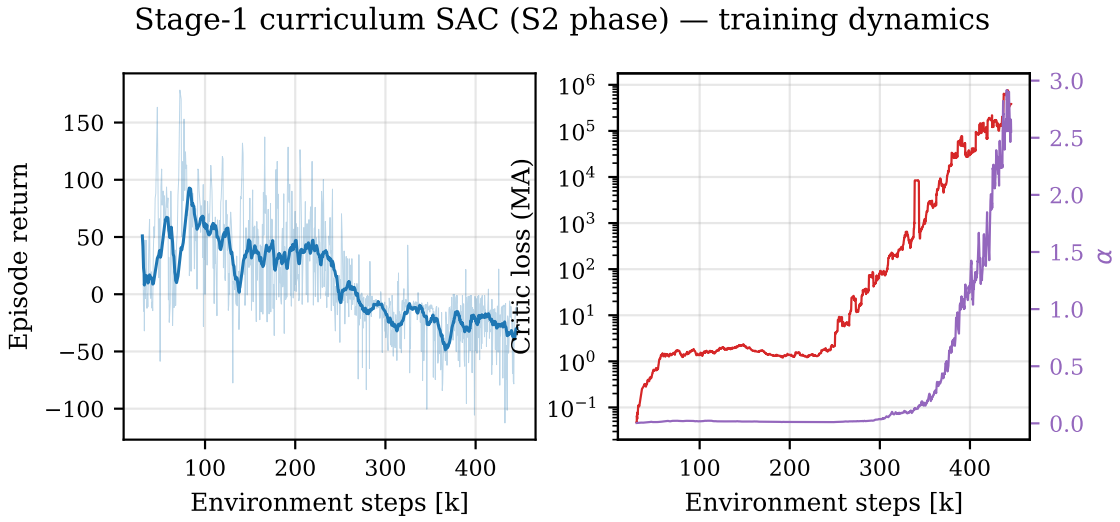

Figure 3: S2 stage-1 SAC run (recorded telemetry). Left: episodic return (thin: raw; thick: 20-episode moving average). Right: critic loss (log scale, red) and entropy temperature α (purple). The critic diverges by six orders of magnitude between ∼250 k and the run's end at 445 700 steps.

4.3 S3: residual RL on a frozen base, and a five-layer failure cascade

S3 changed strategy: rather than learning to race from scratch, SAC would learn bounded *corrections* on top of a frozen imitation policy ("residual RL" [12]). The base policy π_D — a DAgger-distilled network driving ∼107 s laps — is held fixed; the SAC policy π_R outputs a residual scaled per axis and the executed action is

$$\mathbf{a}_t \;=\; \mathrm{clip}\big(\pi_D(\mathbf{s}_t) + \boldsymbol{\delta} \odot \pi_R(\mathbf{s}_t),\, -1,\, 1\big), \qquad \boldsymbol{\delta} = (\delta_{\text{steer}}, \delta_{\text{ab}}). \tag{5}$$

By construction $\pi_R \equiv 0$ reproduces the base behaviour, so the scheme cannot forget how to drive — in principle. In practice the first residual runs crashed within 50 m, and diagnosing why produced the most instructive engineering record of the project. Five independent defects were found, each masking the next:

1. **Symmetric residual bounds.** A global $\delta = 0.20$ allows a 20 % steering override; at 150 km/h this exceeds lateral grip and produces snap oversteer immediately. Fix: asymmetric bounds — conservative steering authority ($\delta_{\text{steer}} = 0.01$–$0.05$ during S3 debugging, 0.15 in the final configuration) with generous longitudinal authority (δ_{ab} up to 0.50).

2. **Exploration-noise hold time.** gSDE [8] with `sde_sample_freq` = 8 holds one noise realisation for 160 ms — more than 6 m of travel at speed; a held steering bias of even a few percent is a trajectory change, not exploration jitter. Fix: resample every tick (`sde_sample_freq` = 1), turning sustained yanks into high-frequency noise the chassis filters out.

3. **Interface regressions.** Legacy logging assumed scalar δ; the upgrade to a per-axis array crashed the training loop. (Trivial, but it gated every later experiment.)

4. **Anchor-induced variance collapse.** The first BC anchor penalised the full policy output — mean *and* variance — with $\beta_{\text{BC}} = 10^4$. The optimiser satisfied it by driving $\log \sigma \to -\infty$: a deterministic

policy that repeated the same slightly-off trajectory and crashed at the identical braking point every episode, with entropy-based exploration permanently destroyed. Fix: anchor the *mean only* (Eq. (6)) and let the SAC entropy term govern variance.

5. **Physics–gradient desynchronisation.** With `train_freq` = 1 step, a full 256-batch backpropagation ran inside the 20 ms control window. TORCS does not pause: during 50–100 ms gradient stalls the car held its previous command, arriving 4–8 m late on the brakes at the end of the back straight — in training mode only, which is why evaluation runs of the very same policy were clean. Fix: collect transitions in real time and train *between* episodes (`train_freq` = (1, episode), `gradient_steps` = −1, i.e. one update per collected step, executed while the car is parked). For an off-policy method with a replay buffer this is mathematically equivalent in expectation, and it is the standard pattern in physical robotics, where control loops cannot block on learning.

The repaired configuration is visible in the data. In the pre-fix residual run the episodic return never leaves the noise floor around zero; an identical post-fix run climbs steadily to returns of ∼6,500 — two clean laps per episode with lap bonuses — over ∼600 k steps (Fig. 4).

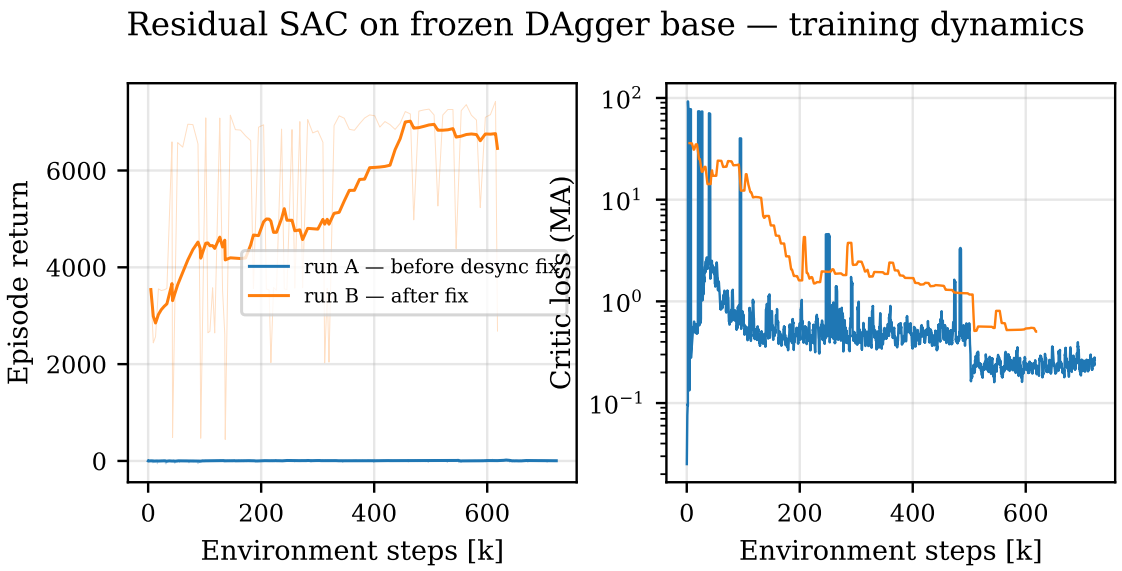

Figure 4: Residual SAC before and after the S3 fixes. Left: episodic return (15-episode moving average, thin lines raw); the pre-fix run (blue) is flat at ≈ 0, the post-fix run (orange) reaches two-lap returns. Right: critic loss, log scale.

The stage-1 anchored actor objective used for seeded and residual SAC is a TD3+BC-style [13] combination, with an adaptive scale normalising the RL term against the current value magnitude:

$$L_{\text{actor}} = \underbrace{\frac{\alpha_{\text{n}}}{\mathbb{E}\big[|Q(\mathbf{s},\mathbf{a})|\big]}}_{\lambda} \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\Big[\alpha \log \pi(\mathbf{a}|\mathbf{s}) - \min_{i=1,2} Q_i(\mathbf{s},\mathbf{a})\Big] \;+\; \beta_{\text{BC}}(t)\, \mathbb{E}_{\mathbf{s}\sim\mathcal{D}_{\text{demo}}}\Big[\tanh^2\big(\mu_\theta(\mathbf{s})\big)\Big], \tag{6}$$

with $\alpha_{\text{n}} = 2.5$ and $\beta_{\text{BC}}(t)$ decaying linearly from 100 to 0 over 300 k–500 k steps. In the residual parameterisation the anchor pulls the *residual mean* toward zero on demonstration states — i.e. toward the frozen base policy — without touching the variance head, which resolves defect 4 by design.

4.4 S4-F: teacher optimisation, distillation, and the deliverable

With the residual stack finally healthy, its base remained slow (∼107 s), and from-scratch SAC estimates suggested that closing a further ∼20 s by pure RL exceeded the remaining budget. The final stage therefore inverted the roles: put the engineering effort into a *white-box analytic controller* whose parameters a distributed search can optimise at very high throughput — an analytic controller needs no gradient updates, so each evaluation is exactly one headless race — and use learning only to compress the result into the required neural form.

4.4.1 Analytic teacher evolution

Six controller generations were built (Table 3, Fig. 5).

Table 3: Analytic teacher generations. Times are best standing-start laps.

Ver.	Speed model	Character	Best lap
v1/v2	$v^* = v_0 + k \cdot \min(\text{fwd sensors})$	lookahead PID on live rangefinders	105.8 s
v3	12+12 static waypoints over track position	open-loop speed/line profile; 2,210 Optuna trials	118.8 s (plateau)
v4	$v^* = c\sqrt{\text{reach}}$	correct $v \propto \sqrt{R}$ physics, but straights coupled to corner gain	107 s
v5	decoupled: $v_{\text{apex}} = c\sqrt{\text{sev}}$, $v^* = \sqrt{v_{\text{apex}}^2 + b\,(\text{fwd} - m)}$	near-redline shifting; predictive late braking	91.2 s
v6	v5 + out–in–out line: $\rho^* = -k_e\,\text{turn} \cdot \text{sharp}$ (entry), $+k_a\,\text{turn} \cdot \text{sharp}$ (apex)	racing-line gains tunable from 0 (can recover v5)	≈88.8 s

Two findings dominate this progression, and both were located by telemetry rather than by search:

(a) The shift schedule was the ceiling. Every controller through v4 — and, unnoticed, the entire RL lineage before it — shifted up at ≈8,200 rpm. For an engine with an 18,700 rpm redline this is severe short-shifting: the speed-vs-distance profile showed a hard 174 km/h cap on every straight, invisible in lap times alone. Raising the shift point to 17,800 rpm lifted the cap to 184 km/h and cut ∼4 s immediately. Notably, the gain came entirely from the shift *point*, not from additional ratios: post-fix telemetry shows the car reaching its eventual ∼255 km/h top speed in fifth gear at ∼17,200 rpm, with sixth gear engaged on under 0.1 % of ticks — and the seventh ratio listed in the car definition is not reachable through the SCR command interface at all (gear commands are clamped to $-1 \ldots 6$). This was a *driver* fix, not a physics change: the car model was never modified.

(b) Straight-line and corner speeds must be decoupled. With $v^* = c\sqrt{\text{reach}}$ a single coefficient governs both the apex speed and the straight-line target, so the optimiser cannot raise top speed without overspeeding corners; telemetry showed the car cruising *at* its own target (min ≈ max ≈ 183 km/h) rather than at the car's limit. Splitting the model into an apex term (from the tightest forward reading, "sev") and an additive braking-reach term (from the most open reading, "fwd"), clamped by a free top-speed parameter, unlocked 244.5 km/h and brought the lap from ∼101 s to 91.2 s. Model structure recovered what no amount of tuning of the coupled form could (Fig. 6).

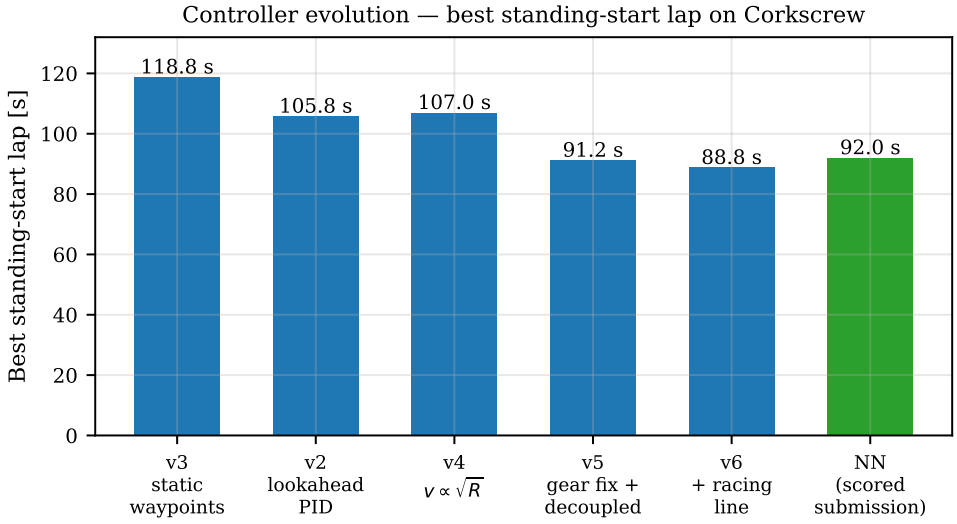

Figure 5: Controller evolution: best standing-start lap per generation, and the scored submission network distilled from v6. The v6–NN gap is the cost of distillation plus the stricter scoring conditions (fuel and damage enabled).

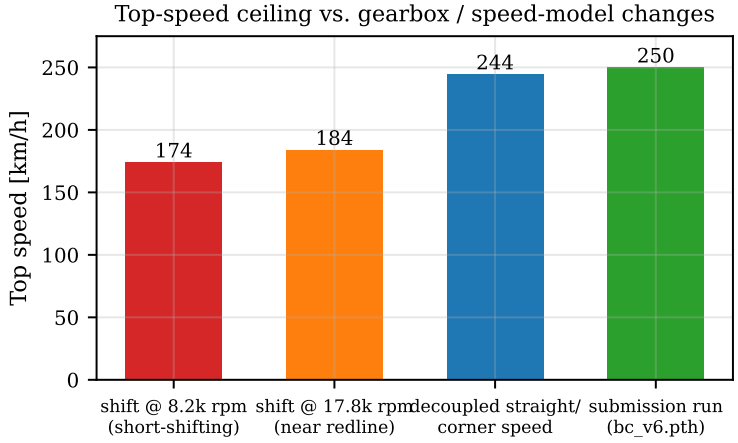

Figure 6: Top-speed ceiling against gearbox and speed-model changes. The 174 km/h plateau was a shift-scheduling artefact; the jump to 244.5 km/h came from decoupling straight-line from corner speed targets.

4.4.2 Distributed hyperparameter search

The v5/v6 controllers expose 54 parameters (segment speeds, braking reach and margin, apex coefficient, racing-line gains, ABS/TCS thresholds, shift points). These were optimised with Optuna [14] in two infrastructure generations: first 4 Windows machines × 6 GUI TORCS instances (ports 3001–3006) coordinated through PostgreSQL; then a substantially cheaper WSL deployment running TORCS 1.3.7 headless (`torcs -r`), 10 workers per machine — the compiled SCR server module admits at most 10 client robots — against a local WAL-mode SQLite study, one-way mirrored to a cloud PostgreSQL instance. Across studies (v3: 2,210 trials; v5b and v6 continuing to convergence with 10 parallel workers) the search consumed on the order of 15,000 trials, each trial being a 3-lap headless race scored by its best lap. Notable infrastructure lessons: per-process connection pools had to be pinned to one connection (cloud pool exhaustion); a global `pkill torcs` in the client's self-heal path could cascade across all workers and was scoped to the owning worker's instance; and wedged-but-on-track cars required a no-progress bailout to stop them consuming full trial timeouts.

4.4.3 Distillation: BC with anti-copycat noise, then DAgger

The tuned v6 teacher was compressed into the deliverable network by behavioural cloning. The student is an MLP mirroring the SAC actor trunk — $32 \to 256 \to 256 \to 128 \to 2$, LayerNorm and ReLU on every hidden layer, tanh output; 108,674 parameters, 429 KB serialised. Training minimises a corner-weighted regression loss over $N = 5 \times 10^5$ teacher transitions,

$$L_{\mathrm{BC}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \bigl(1 + k\,|a_i^{\mathrm{steer}}|\bigr)\, \bigl\| f_\theta(\mathbf{s}_i) - \mathbf{a}_i \bigr\|_2^2, \qquad k = 4, \tag{7}$$

with a 20 % validation split and early stopping. Corner weighting compensates for the extreme class imbalance of racing data, where straights dominate the sample count but corners dominate the lap time.

Plain BC on this data *failed in a characteristic way*: validation loss converged to 2×10^{-3}, yet the cloned policy crashed 787 m into its first closed-loop lap. The cause is the copycat/inertia shortcut of Codevilla et al. [11]: with the previous steering command present at index 31 and a teacher whose steering trajectory is extremely smooth, the regression optimum is approximately $a_t^{\mathrm{steer}} \approx \mathbf{s}_{31} = a_{t-1}^{\mathrm{steer}}$ — a policy that holds its wheel angle, ignores the rangefinders, and leaves the road at the first significant corner while scoring excellent open-loop validation numbers. The fix perturbs exactly the shortcut feature during training: zero-mean Gaussian noise ($\sigma = 0.15$, clipped to $[-1, 1]$) is added to $\mathbf{s}_{31}$ in every minibatch, making the copied value unreliable and forcing the network to read the track geometry. Validation loss rises by an order of magnitude — and closed-loop driving starts working; the deployment-time observation is unchanged.

Distribution shift over long horizons was then addressed with DAgger [10]: five iterations of (learner drives 100 k steps → teacher relabels every visited state → retrain on the aggregate). Two implementation details mattered: the aggregate dataset is *seeded* with teacher demonstrations covering the full lap (otherwise

early iterations train only on the learner's short pre-crash prefixes and forget the rest of the circuit), and the same environment/aids stack as submission is used throughout so that the gear schedule seen in data collection matches deployment exactly. The latter point closed a subtle regression: the corrected 17,800 rpm shift schedule had to be propagated into the shared `AidsConfig`, or the distilled network would have inherited the old 174 km/h short-shift cap from its own training data regardless of teacher quality.

During BC smoke tests one further environment defect was isolated: the extra off-track clause `track.min() < 0` described in Section 3.5. Because individual beams return -1 inside the sustained $\sim$60 km/h hairpin at 2,400 m, the collection environment terminated every episode there — capping `max_dist` at $\approx$2,463 m — while the identical teacher lapped cleanly under the Optuna evaluator, which tests only $|\rho| > 1.10$. The clause predates S4-F and had silently truncated BC/DAgger/residual episodes near that corner for every previous agent generation; removing it aligned all evaluators on the single lateral criterion.

4.4.4 Optional residual refinement and final evaluation

The project ships *two* agents that reach the same $\sim$92 s class of standing-start lap. The first is the deterministic BC/DAgger network `bc_v6.pth` evaluated by `submit_agent.py` under competition conditions (standing start, fuel consumption and damage enabled): **92.04 s**, top speed 250 km/h. The second is a residual reinforcement-learning policy that adds bounded, learned corrections on top of the same frozen network, $\mathbf{a} = \mathrm{clip}(\pi_D + \boldsymbol{\delta} \odot \pi_R)$ (Eq. (5)); post-fix residual training (Fig. 4) converged to multi-lap returns and clean laps, and the resulting policy runs a comparable **$\sim$92 s** standing-start lap. That both an imitation policy and a from-RL residual policy land on the same time is the point: the result is a property of the pipeline, not an artefact of behavioural cloning alone. A representative recorded lap is shown in Fig. 7. The gap between the scored 92.04 s and the teacher's $\sim$88.8 s figure comprises distillation loss and the stricter scoring physics (the teacher search ran with fuel/damage disabled for evaluation throughput; the submission path does not).

5 Experimental Setup

5.1 Software and hardware

TORCS 1.3.7 with the SCR server patch [1] was used on Windows (GUI, `wtorcs.exe`) and inside WSL2 Ubuntu (headless, compiled from the same source as the organisers' container to keep physics identical). The learning stack is Python 3.12, PyTorch, Stable-Baselines3 [7], Gymnasium, and Optuna [14]; study storage was SQLite (WAL) locally and PostgreSQL in the cloud. Search ran on up to four consumer machines (6–10 TORCS instances each); distillation and RL training used a single consumer GPU. An auxiliary tool (`granite_analysis.py`) used the IBM Granite LLM over local telemetry summaries to produce per-corner natural-language diagnostics during development.

5.2 Final hyperparameters

Table 4: Principal hyperparameters of the final configuration.

SAC (stage 1 / residual)		**BC / DAgger**	
actor/critic MLP	$256, 256, 128$ + LayerNorm	student MLP	32→256→256→128→2
learning rate	3×10^{-4} / 1×10^{-4}	transitions	5×10^5 (BC), 5×10^5 (DAgger)
replay buffer	10^6 / 5×10^5	loss	weighted MSE, $k = 4$ (Eq. 7)
batch size	256	prev-steer noise	$\sigma = 0.15$
γ, τ	0.99, 0.005	optimiser	Adam, 10^{-3}, batch 1024
entropy α	auto (target $-\dim \mathcal{A}$) / 0.02	early stopping	20 % val., patience 10
gSDE	on; resample every step	DAgger	5 iterations × 100 k steps
train schedule	per step / per episode (-1)	seeding	20 k–50 k teacher steps
BC anchor	β_0=100, decay 300 k–500 k	residual bounds	$\delta = (0.15, 0.50)$
warmup	10 k steps (`learning_starts`)	control rate	50 Hz (20 ms budget)

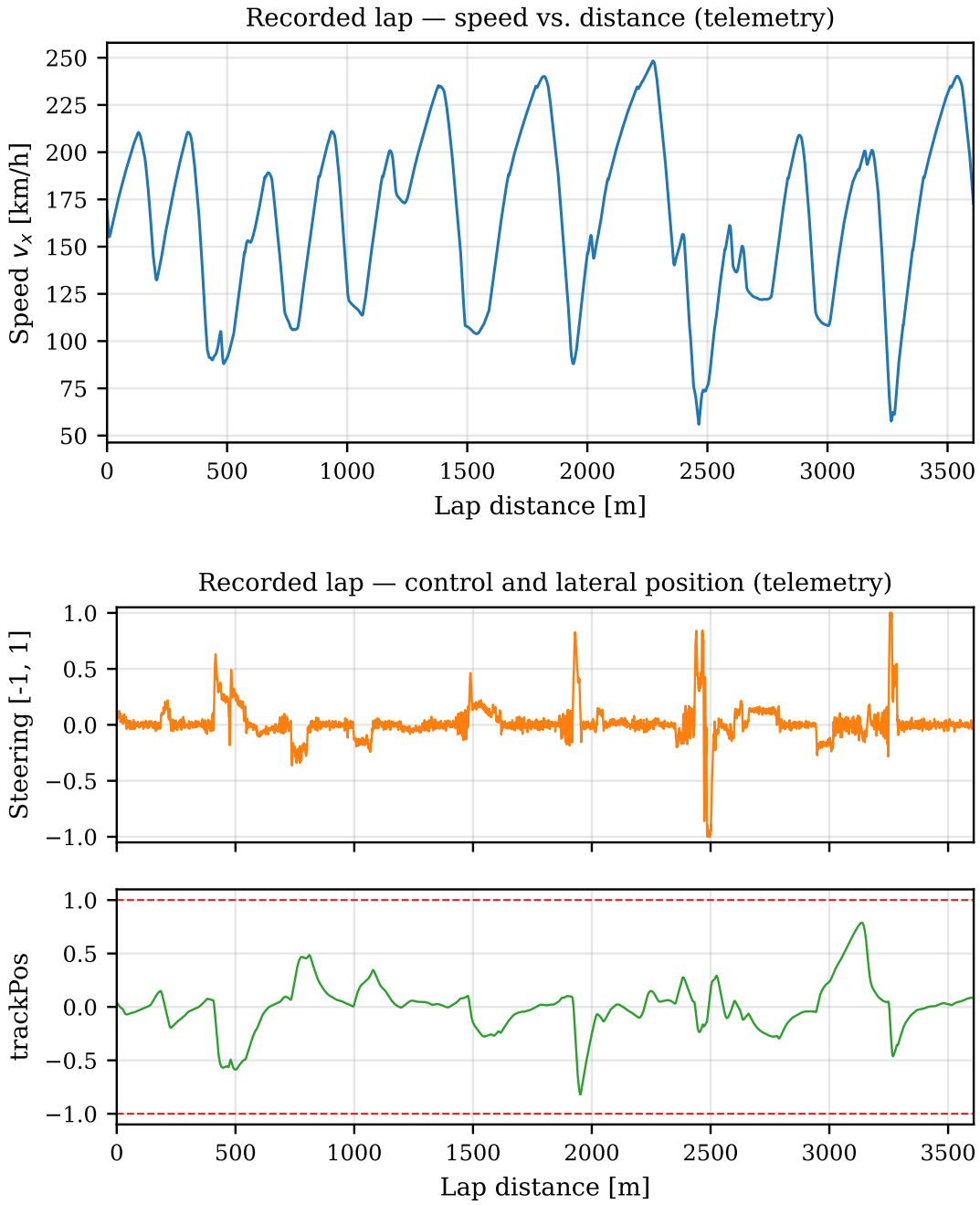

Figure 7: A representative lap recorded in project telemetry. Top: speed profile; the two deep minima at ≈2,450 m (56 km/h) and ≈3,260 m (58 km/h) are the Corkscrew chicane and the final hairpin. Bottom: steering command and lateral position; $|\rho| < 1$ throughout, with kerb-adjacent excursions at corner exits.

Table 5: Lap-time progression across the project (standing start unless noted). Sources: experiment logs and recorded telemetry.

Agent	Stage	Conditions	Best lap [s]
from-scratch SAC	S1	no fuel/damage	∼105–115
DAgger NN (v2-teacher base)	S3	no fuel/damage	106.96
residual SAC on the above	S3	no fuel/damage	∼103
teacher v3 (2,210 trials)	S4-F	no fuel/damage	118.81 (plateau)
teacher v2	S4-F	no fuel/damage	105.8
teacher v5 (gear + decoupling)	S4-F	no fuel/damage	91.24 (88.73 flying)
teacher v6 (racing line)	S4-F	no fuel/damage	≈88.8
submitted network `bc_v6.pth`	S4-F	**fuel + damage**	**92.04**

6 Results

Three quantitative observations:

Where the time came from. Of the ∼18 s separating the 106–107 s plateau from the tuned v6 teacher (≈88.8 s), roughly 4 s came from the gear-schedule fix, ∼10 s from decoupling straight/corner speed targets, ∼2–3 s from the tuned racing line (v6 over v5), and the remainder from Optuna's joint tuning of braking reach, apex coefficients and ABS/TCS thresholds around these structural changes. All four gains were enabled by structure, not by search alone: the v3 study demonstrates this directly, having spent 2,210 trials to converge 12 s *above* the much simpler v2 controller because its static-waypoint model could not react to the car's live state.

Distillation cost. The BC/DAgger student gives up ≈3 s to its teacher under matched conditions (88.8 → ≈92 with the scoring physics change included). The copycat fix was the difference between this small, expected gap and total failure: without it the student did not complete a single lap despite an order of magnitude better validation loss.

Consistency of the final agent. On the best recorded lap the controller holds $|\rho| \leq 0.85$ at all times (Fig. 7), brakes from 248 km/h to 56 km/h for the Corkscrew without ABS intervention-induced instability, and shows steering activity concentrated at corner entries with a quiet wheel on straights — the intended effect of the smoothness shaping term and the corner-weighted BC loss.

7 Discussion

Why the hybrid beat pure RL here. The final pipeline replaces gradient-based policy improvement with black-box search over an analytic controller, then uses supervised learning for compression. Three properties of the problem favour this inversion. First, *evaluation throughput*: a parameter vector for the analytic teacher is scored by one headless race with zero learning overhead, so ten workers sustain hundreds of trials per hour; SAC, by contrast, pays for every environment step with gradient computation and is bound to a handful of real-time instances. Second, *credit assignment*: lap time responds to decisions (braking points, apex speeds) whose consequences arrive seconds later; the analytic controller encodes these causal links structurally, while model-free RL must discover them through a shaped proxy reward — and our shaping, Eq. (3), demonstrably favoured safe mediocrity over risk-adjacent speed. Third, *failure locality*: when the teacher was slow, telemetry pointed to a nameable mechanism (a shift threshold, a coupled speed formula) that could be fixed in one line; when SAC was slow, the search space of possible causes spanned rewards, normalisation, exploration and optimisation dynamics simultaneously, as the S2 and S3 records illustrate.

These are conditional advantages, not a general argument against RL: the residual policy shows that reinforcement learning on top of a strong prior recovers the same ∼92 s standing-start lap as the imitation deliverable, so RL is a first-class part of the final result rather than a discarded baseline. The project's time budget was also a real constraint — the work was carried out alongside the team's first year of undergraduate study, which placed a premium on approaches with predictable wall-clock cost over methods that might eventually win but could not be scheduled — and under that constraint, search-plus-distillation was the only route that reliably converted compute into lap time before the deadline.

The value of telemetry-first debugging. Every major advance in this project was preceded by an instrumented observation, not by a hypothesis about learning: the 174 km/h cap (gear schedule), the min ≈ max cruising speed (coupled targets), the flat pre-fix residual return (desynchronisation), the perfect-validation/ crashing-deployment contradiction (copycat shortcut), and the episode truncations at 2,463 m (sensor sentinel in the termination predicate). Conversely, the two documented false starts — a presumed track mismatch that was actually a stale configuration overlay, and the assumption that "more training" would lift the 106 s plateau — both arose from reasoning ahead of measurement.

Limitations. (i) The scored submission retains a ≈3–4 s gap to its teacher; closing it with a residual policy on the distilled base was demonstrated in telemetry but not hardened to submission quality before the deadline. (ii) The teacher search optimised lap time with fuel and damage disabled; tuning under scoring physics would likely recover part of the submission gap. (iii) The stage-2 (multi-car) pipeline — opponent-aware observations, proximity/overtake shaping, mass-start training in `phase2/` — was built and

exercised but is not part of the scored deliverable and is not evaluated here. (iv) All results concern a single circuit and car; the teacher's structure (sensor-driven speed targets) should transfer, but its 54 tuned parameters are Corkscrew-specific.

8 Conclusion

A neural racing agent for TORCS was produced by a staged process that began with model-free SAC and ended with a teacher–student pipeline: a 54-parameter analytic controller tuned by ∼15,000 distributed Optuna trials to ≈88.8 s, distilled into a 109 k-parameter LayerNorm MLP by behavioural cloning (with an explicit countermeasure to the copycat shortcut) and DAgger, and scored at 92.04 s on a standing-start lap with fuel and damage enabled. The intermediate stages contributed the environment, the stabilised network architecture, the driver-aid layer, and — through their failures — the diagnostic record that made the final design possible. The through-line of the project is methodological rather than algorithmic: measure first, prefer structures whose failures are nameable, and spend learning capacity only where engineering cannot reach.

Acknowledgements. The project was developed for the IBM AI Racing League with academic support from Politechnika Świętokrzyska, and used the IBM Granite model as a development-time analysis assistant.

References

[1] Daniele Loiacono, Luigi Cardamone, and Pier Luca Lanzi. Simulated car racing championship: Competition software manual. *arXiv preprint arXiv:1304.1672*, 2013.

[2] Bernhard Wymann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. TORCS, the open racing car simulator. http://www.torcs.org, 2014.

[3] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[5] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1861–1870, 2018.

[6] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[7] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[8] Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, pages 1634–1644, 2022.

[9] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 1, pages 305–313, 1989.

[10] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635, 2011.

[11] Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9329–9338, 2019.

[12] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *International Conference on Robotics and Automation (ICRA)*, pages 6023–6029, 2019.

[13] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 20132–20145, 2021.

[14] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2623–2631, 2019.

[15] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.