2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Continual-RL for Generalization in Autonomous Racing on the RoboRacer Platform

Joel Siegert[1,†], Edoardo Ghignone[1,†], Michele Magno[1]

***Abstract*—A key challenge in modern robotics is to adapt to changing environments, a challenge that is exacerbated when simulations cannot encompass every possible real-world configuration, and therefore Reinforcement Learning (RL) in the physical world becomes necessary. Continual Reinforcement Learning (RL) provides the tools to address this challenge; however, both the frameworks and the methods remain underexplored. Autonomous Racing (AR) and in particular the RoboRacer competition provide a testing ground for such methods, as learning to drive on a new track-floor combination with the least amount of new experience naturally frames a continual learning problem. This work tries to address this gap by proposing a continual RL framework based on Continual Backpropagation (CBP) that is able, with only real-world data, to train a generalistic policy on a set of tracks and then fine-tune it within 15 minutes to outperform classical controllers. Furthermore, a comparison method based on offline RL is proposed, and a simulation analysis of the plasticity properties of the methods is conducted.**

I. INTRODUCTION

RL has recently evolved into a practical tool for robotics, achieving superhuman performance in board games [1], quadrupeds [2, 3], and drone racing [4]. Generalization to the physical world remains a central challenge, usually tackled by carefully designing the simulator used for pre-training and adding randomization on key parameters: some examples are legged locomotion on unknown terrain [5] or grasping novel objects [6].

Following a different approach, this paper focuses on direct real-world training, as simulators can be expensive or difficult to set up [7], and generalization via randomization can further add over-conservativeness. Real-world training, though, only allows training of one agent in a single environment at a time (given one robot), and this drastically reduces the amount of data available, highlighting the need for sample-efficient algorithms. Furthermore, the real world might present non-stationary environments, which introduce continual learning challenges: maintaining adaptability without catastrophic forgetting is a requirement [8].

Autonomous Racing (AR) on the RoboRacer platform (formerly F1TENTH) provides an affordable yet demanding testbed for benchmarking State-of-the-Art (SotA) controllers on real hardware in non-stationary environments: firstly, the ever-changing track layout highlights the need for algorithms that can adapt to different settings, such as continual learning methods; furthermore, the tight schedule of the race imposes a strict constraint on the available samples. Algorithms have to adapt with only minutes and not hours or days of extra data. Different previous RL methods have been applied to the RoboRacer platform [9, 10, 11, 12], but no result has been a viable solution able to generalize across different real-world track layouts.

In this context, we present a continual RoboRacer training framework that adopts the off-policy Soft Actor-Critic (SAC) architecture [13], used for its renowned sample efficiency, and adapt it to a continual setup with Continual Backpropagation (CBP) [14].

In particular, we conduct tests assuming multiple tracks can be available for pre-training, but a final track is unknown (see Figure 1 for more details), and we show that the combination of CBP with SAC can surpass one of the leading controllers in the RoboRacer competition within 15 minutes of fine-tuning. Furthermore, we compare this first method to offline RL pre-training by means of Implicit Q-Learning (IQL) [15], showing that even though offline-RL can train a model on the entire dataset, continual RL can adapt quicker to an unseen track layout and tire-floor combination.

Our summarized contributions are:

- A continual RL framework based on CBP that learns generalistic policies with only real-world data and fine-tunes within 15 minutes on an unseen track-floor combination. Previous work showed similar performance after only 82 minutes [12].
- An alternative framework for generalistic policy pre-training based on offline RL, that shows promising signs of plasticity but reduced performance, at parity of training steps.
- An evaluation in simulation of the effect of adding continual learning network modifications (CBP [14], L2 Init [16]) to only buffer management strategies, showing that continual techniques help the performance increase from a fine-tuning stage.
- Tracks, simulation models, and RL frameworks are open-sourced to foster replication and comparisons (github.com/ForzaETH/Continual-RL-ICRA-26)

II. RELATED WORK

A. Real-World Reinforcement Learning

Most of the success of RL in the real world comes from policies pretrained in simulation [17, 4, 18]. However, the sim-to-real gap is a remaining challenge: most successful

*This work was not supported by any organization
[1]Center for Project-Based Learning, D-ITET, ETH Zurich, edoardo.ghignone@pbl.ee.ethz.ch
[†]Equal Contribution

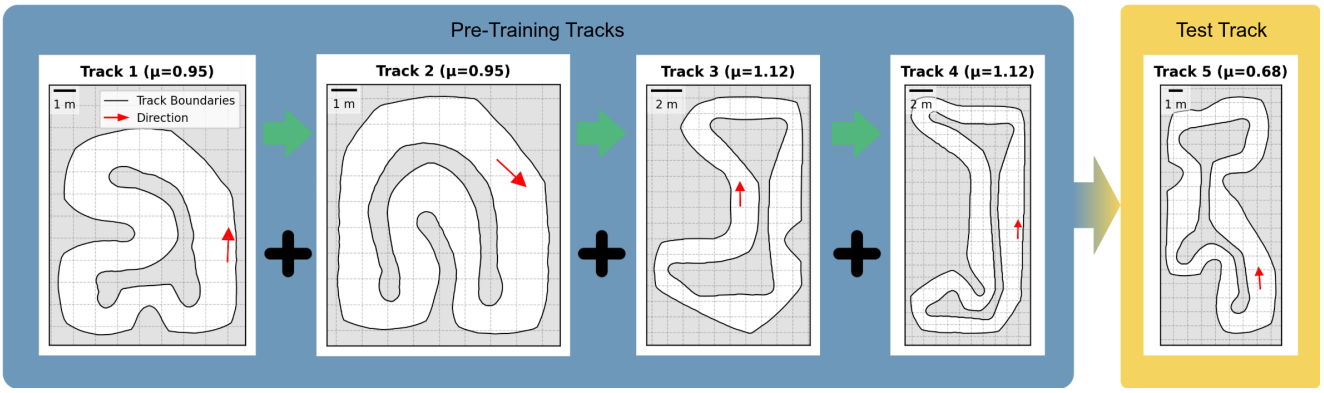

Fig. 1: Set of tracks used for this study, subdivided into four pre-training and one test track. A Newton-meter was used to measure lateral static friction (indicated with μ). The **green** arrows indicate the sequential order in which the tracks are used for continual training, while the **black** addition signs indicate the alternative offline pre-training strategy.

transfers require at least some amount of real data or fine-tuning on the real system [19, 20], zero-shot transfer can often be impractical due to the sim-to-real gap [19, 21], and domain randomization could trade generalizability for performance, and produce over-conservative policies.

For this reason, training with physical robots remains a viable alternative, and this work will therefore focus on real-world learning, where the agent will learn only from real-world interactions, bypassing any type of simulator and the issues that might arise with it. The review by Tang et al. [19] highlight that real-world learning introduces mainly two new challenges, one of them being how to accelerate training with real-world samples, which are clearly limited in number when compared to simulators that can simulate thousands of environments in parallel [22]. This paper tries to tackle the sample efficiency limitation by applying continual learning [23] to the AR challenge. Specifically, we consider the setup where a set of track layouts can be available in a pre-training phase, and an unknown track, with possibly a different floor, will make up the test condition.

A first crucial branch of related work analyzes sample-efficient RL algorithms. SAC [13], one of the most diffused off-policy algorithms, constitutes the core component of this paper. While different extensions improving the sample-efficiency of the original algorithm exist (e.g. [24, 25]), we preferred to base our work on the more widespread open-source implementation of SAC [26], which has been extended successfully with custom features for real-world learning (asynchronous architecture [27], generalized State-Dependent Exploration (SDE) [28]), and leave the experimentation with novel algorithms to future work. Similarly, model-based methods have had great success in terms of sample complexity and have been successfully deployed on physical robots [29, 30]. However, the added training compute can outweigh the benefits when computational resources are tight.

Real-world training further makes it highly impractical to present the learner with substantially different environments and hence the agent usually has to learn sequentially, i.e. by possibly having access to a specific environment only once. Continual learning offers the tools to make this possible.

B. Continual Learning and Reinforcement Learning

Continual RL addresses the degrading performance of neural networks under changing tasks or state distributions [23]. Some of its typical challenges are catastrophic forgetting (reduced performance after task switch) and loss of plasticity (inability to adapt after a task switch). Typical strategies to avoid forgetting include replay buffer management, regularization, or network segregation [31]. CLEAR [32], for instance, combines replay buffer reuse and regularization and shows good performance on the CORA benchmark [33], while methods such as Emphasizing Recent Experiences (ERE) [34] emphasize newer samples to accelerate adaptation. Focusing on loss of plasticity instead, different metrics have been developed [35, 14]. Plasticity has been studied through metrics like dormant neurons (dead units) [36] or average weight magnitudes [16, 14, 35], and mitigated by reinitializing biased network heads [37] or low-utility units, as with CBP [14] and in Reinitialize Dormant Neurons (ReDo) [36]. Many continual learning methods have been proposed and show increasing performance in simulation. However, few ideas are tested on a physical system. This work will focus on the racing setting within the RoboRacer format, and particularly on loss of plasticity.

C. Autonomous Racing and Roboracer

The Roboracer AR competition [38] (formerly F1TENTH) provides a key robotic challenge, as participants have to adapt to unseen track layouts under strict time constraints, demanding efficient generalization, either by zero-shot transfer, rapid fine-tuning, or leveraging pre-trained policies. Classical methods for AR such as Pure Pursuit (PP) [39], Model Predictive Contouring Control (MPCC) [40], and Model- and Acceleration-based Pursuit (MAP) [41] are the most performant on the RoboRacer platform [42], and are also widespread across different AR competition formats (e.g. in [43, 44]). RL methods, however, provide an interesting alternative, as they achieved SotA performance in simulated car racing [45] and real-world drone racing [4].

On the RoboRacer physical platform, different RL methods have been explored without clearly demonstrating high-

performance: domain randomization [9], direct sim-to-real deployment [46, 30], and image pretraining with online finetuning (FastRLap) [10]. Furthermore, Hildisch et al. [12] show that, with a residual policy structure, an RL controller can drive faster than a classical controller, and can learn significantly faster than an end-to-end controller, reaching a higher performance level than classical control methods for AR.

This work addresses the main issues of [12] in generalization and presents a continual learning framework based on CBP that is able to learn on multiple tracks, generalize to unseen tracks, and continually adapt to a new layout, without using a residual policy structure. Furthermore, offline RL offers an alternative [47], enabling pre-training on a dataset of diverse policies, and even outperforming these later in some cases [48]. This work compares the proposed continual learning framework to an offline framework based on IQL [15].

III. METHODOLOGY

A. Evaluation Setup

Physical System - For evaluation, we replicate the platform of [49] and [12], extending the training framework of the latter. Training is conducted on a remote computer with a NVIDIA GeForce RTX 4070 laptop GPU. Note that training directly on the car was also successful, but the increased computational load induced higher variance in the actor's control frequency and reduced the battery lifetime, making training more time-consuming.

Human-Monitored Training - The safety filter used in [12] decides based on the relative heading to the intended race line whether to use the RL controller or the baseline controller. This safety filter does not recognize if the car is about to crash when the car is sliding out in corners but still stays parallel to the race line, which can easily happen at high speeds. Furthermore, tracking progress on a residual RL approach can be less clear, due to the interplay between the RL agent and the baseline controller. For these reasons, we chose to not use the residual RL structure, but to follow the usual structure where full agency is relinquished to the RL agent, following what [12] names an end-to-end controller. Although the method requires some initial human intervention, continual strategies improve sample efficiency. As a result, the agent can learn a basic slow policy after only a few episodes, and after a few minutes, only sporadic operator interventions are needed.

Train for Generalization, Fine-tune for Racing - We train our setup on 4 different tracks with 2 different tire-floor combinations (see Figure 1 for details) for a total of 100'000 gradient steps, which corresponds to 40-60 minutes of training. On the final track, we only train for performance. Practically, this means we keep a maximum of 200'000 samples in the buffer during training and prune overflowing samples at the end of training according to a diversity measure applied trajectory-wise. However, in our tests, this number was only reached after the last training track and therefore will only have an influence for further training. Such a diversity method consists of a dimensionality reduction of the concatenated observation and action vector by PCA, where we keep the 10 main principal components accounting for 95% of the variability. This is followed by k-Nearest Neighbors (k-NN) with k=5. The per sample diversity score is calculated as the average distance s_i to the k nearest neighbors. From this, we get the final sample priority as $P(i) = 1/rank(s)$, where the rank is the sorted position in the replay buffer. We define the trajectory priority as the average score of all samples of the trajectory and keep all samples from the trajectories with the highest priorities. This was inspired by previous approaches that tried to tackle the continual learning problem with buffer management strategies [31]. Keeping older samples during training for generalization prevents forgetting. On the final race track, we start with 5'000 samples from the previous replay buffer and let the agent adapt more, accepting some forgetting. The samples are again obtained by employing the diversity filter method. First experiments with an entirely fresh replay buffer were unsuccessful: after a few moments of training the performance dropped severely and the car started acting as in early training.

B. Soft Actor-Critic

Our setup builds on top of the SAC [13] implementation from Stable Baselines 3 (SB3) [26]. We use Async-SAC-1 from [27] implemented as two separate ROS nodes: an acting node and a learning node. The acting node runs at 20 Hz, while the learning node runs without time constraints. In practice, our setup ran with an Update-To-Data (UTD) ratio between 1 and 1.7, which was primarily influenced by the sampling time for the increasing replay buffer size. Furthermore, as previous work has highlighted [24], a higher UTD ratio would not have been beneficial for the standard SAC algorithm.

To improve exploration efficiency, we use Generalized State-Dependent Exploration (gSDE) [28] from the SB3 implementation. The most important hyperparameters are summarized in Table I. This setup, as well as observation space, action space, and reward function, is shared between all considered methods. Trainings were repeated for three different random initializations of the network in the case of real-world experiments, and five different random initializations for the simulated experiments.

TABLE I: Hyperparameters for the asynchronous SAC implementation.

Hyperparameter	Value
Network dimension	2x256
Activation function	ReLU
Mini-batch size	128
Learning rate	0.003

Hyperparameter	Value
Discount factor	0.99
gSDE sampling rate	20 steps
n-step TD	3 steps

Observation Space - The state information used in the observation space is acquired from the sensor data following [49]. The components and the corresponding dimensions are listed in Table II. The tire forces are estimated using the Pacejka tire model and give the car some information about the grip on a specific floor. The parameters for the

model are estimated before training on a new floor through system identification according to [50]. The observations are flattened before they are passed as inputs to the networks. Therefore, we define the observations as $\boldsymbol{o_t} = [\Delta t, c, s_t, s_{t-s}, L, d, \theta, \boldsymbol{v}, \dot{\psi}, \boldsymbol{F}, \alpha_f, \alpha_r, \boldsymbol{p}_{race}, \boldsymbol{p}_{in}, \boldsymbol{p}_{out}]^T \in \mathcal{O} \subset \mathbb{R}^{138}$, with full definition of the symbols available in Table II.

TABLE II: Observation space.

Symbol	Observation	Range	Dimension
Δt	Time between observations	$(0, 0.1]$ s	1
c	Collision	$\{0, 1\}$	1
s_t	Longitudinal frenet coordinate	$[0, 120]m$	1
s_{t-1}	Previous s	$[0, 120]m$	1
L	Length of current track	$[0, 120]m$	1
d	Lateral deviation from proposed race line	$[-2, 2]m$	1
θ	Relative heading to the proposed race line	$[-\pi, \pi]$	1
$\boldsymbol{v}$	Longitudinal and lateral velocity in car frame	$[v_{min}, v_{max}]$	2
$\dot{\psi}$	Yaw rate	$[-13, 13]/s$	1
$\boldsymbol{F}$	Front and rear tire forces (x,y,z)	$[-40, 40]m$	2x3
α_f, α_r	Front and rear tire slip angles	$[-\pi, \pi]m$	2
$\boldsymbol{p}_{race}$	Points along proposed race line in car frame	$[-136, 136]m$	2x20
$\boldsymbol{p}_{in}$	Lateral velocity in car frame	$[-136, 136]m$	2x20
$\boldsymbol{p}_{out}$	Lateral velocity in car frame	$[-136, 136]m$	2x20

Action Space - Actions at time step t are defined as $\boldsymbol{a_t} \in \mathcal{A} = [-\delta_{\max}, \delta_{\max}] \times [v_{\min}, v_{\max}] \subset \mathbb{R}^2$, where $\delta_{max} = 0.42$ is the steering angle limit and $v_{\min} = 0.5$ and $v_{\max} = 10$ are the velocity limits. The positive lower velocity limit was crucial for fast early learning.

Reward Function - The reward function consists of three parts. The time normalized stepwise advancement reward $r_{sa} = \frac{s_t - s_{t-s}}{v_{\max} \cdot \Delta t}$ and the collision penalty $r_{collision} = -\mathbf{1}_{collision==True}$ are used in simulation and on-car. For the human-monitored on-car learning, we add another penalty $r_{drive} = -\mathbf{1}_{drive-button==False}$ for releasing the safety drive button which has to be pressed by the human operator while the car autonomy is active. The partial rewards are multiplied by a weighting factor, which results in $r_{total} = w_{sa} \cdot r_{sa} + w_{collision} \cdot r_{collision} + \mathbf{1}_{on-car} \cdot w_{drive} \cdot r_{drive}$. We empirically chose the weights to be $w_{sa} = 20$, $w_{collision} = 16$, and $w_{drive} = 10$.

C. Continual Learning extensions for SAC

Our pipeline integrates CBP [14] into the SAC algorithm. In particular, we apply it to all layers of the actor and the critic networks. The original implementation of CBP only includes one-to-one layer transitions. However, the SAC actor network has 2 heads for mean and Standard Deviation (SD). We extend the instantaneous utility function as follows with a summation over the output layers:

$$y_{l,i,t} = \frac{\left|h_{l,i,t} - \hat{f}_{l,i,t}\right|}{\sum_{j=1}^{n_{l-1}} \left|w_{l-1,j,i,t}\right|} \cdot \sum_{m=1}^{H_{l+1}} \sum_{k=1}^{n_{l+1}} \left| w_{l,i,k,t}^{(m)} \right|$$

With H_{l+1} being the cardinality of similarly shaped output layers. $h_{l,i,t}$ is the current activation of i in layer l, $\hat{f}_{l,i,t}$ is the running average of the activation and their difference is the mean-corrected activation. n_{l-1} and n_{l+1} are the number of units in the previous and next layer respectively. $w_{l,i,k,t}^{(m)}$ is the units k-th outgoing weight of the head with index m and $w_{l-1,j,i,t}$ the units j-th incoming weight.

We combine CBP with L2 Init [16], a regularization-based continual learning method, that adds an L2-loss term between the current and the initial network weights to the learning objective. This is applied to the actor and the critic networks and prevents the network weights from growing indefinitely, which is a common phenomenon that induces loss of plasticity.

Lastly, we employ a variation of the ERE [34] sampling strategy, which assumes a fixed size replay buffer and an alternation between a data collection phase and a parameter update phase. In the latter, the training experiences are uniformly sampled from a stepwise shrinking replay buffer. Because we use the asynchronous SAC setup with a continuously growing replay buffer, the algorithm needs to be adapted. Instead of shrinking the replay buffer, we define a priority curve (Figure 2) which is evaluated for the current replay buffer occupation during sampling. The curve has three parts, a high priority part for new samples with priority $P_{high}(i) = 1.0$, an adaptive low priority part, and a transition spline curve in between which follows the quintic smooth step function $P_{transient}(i) = 6 \cdot t^5 - 15 \cdot t^4 + 10 \cdot t^3, i \in [c_{low}, c_{high}]$, with c_{low} and c_{high} being the percentages of low and high priority samples respectively. We let the lower priority be dependent on the buffer occupation according to $P_{low}(i) = 1/(1+\eta \cdot N) > P_{low,min}$, where η is a decay factor and N is the replay buffer occupation. This has the effect that the priority for older samples shrinks with increasing buffer occupation. For all experiments, we set those values to $c_{low} = 0.5$, $c_{high} = 0.95$, $\eta = 0.001$, and $P_{low,min} = 0.001$. This method gives newer samples more weight, allowing the policy to adapt to the current state distribution while being reminded of older experiences. Note that recomputing the priority curve and the sampling probabilities can dominate computation time if the replay buffer increases too much. In this case, a solution could be to recompute the probabilities only every n steps.

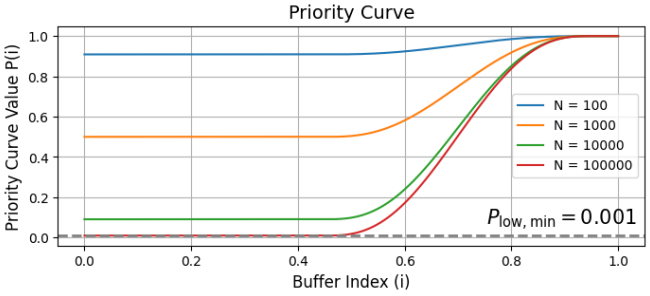

Fig. 2: Priority curve for sampling from the replay buffer with emphasis on recent experiences. Buffer index (i) represents relative positioning in the buffer, where i = 1 indicates the newest sample and i = 0 indicates the oldest sample.

D. Pre-Training with Implicit Q-Learning

As the goal during pre-training is to obtain a policy able to generalize across tracks, we tested an offline RL method, able to train from different trajectories previously obtained by training with CBP-FS (Section IV-A) on the training tracks. A subset of trajectories from all the training sessions, including some from early learning, were kept for the pre-training. During the offline pre-training, we sample uniformly from all experiences instead of relying on the most recent one. We use the IQL algorithm from [15] without modifications and set the expectile for the value loss to 0.7 and the temperature for the advantage weighted behavior cloning weights to 0.1. Furthermore, we use the same network extensions as for the continual variant, CBP and L2 Init, and apply them during pre-training and finetuning.

IV. RESULTS

A. Simulation Experiments

To more thoroughly analyze the different algorithms, an initial experiment was run in simulation. The ROS simulator described by [38] was used, adapted with a custom Pacejka Tire model, identified from real data following [50]. While the model was adapted to closely fit reality, none of the simulation trained models were successfully deployed on the physical platform. The map layout was directly transferred from the one used in the real world.

Five different models were compared in this experiment: a baseline implementation from the end-to-end model of [12] trained from scratch on every map, named **Baseline-FS** (for **F**rom **S**cratch); a continually trained version of the baseline, named **Baseline-C-n**, with n indicating the number of tracks it was trained on; a version of the continual method trained from scratch on every track, named **CBP-FS**; a version trained continually, **CBP-n**; the offline method, **IQL-n**. Pre-training followed the same procedure as described in Figure 1 and in the same order.

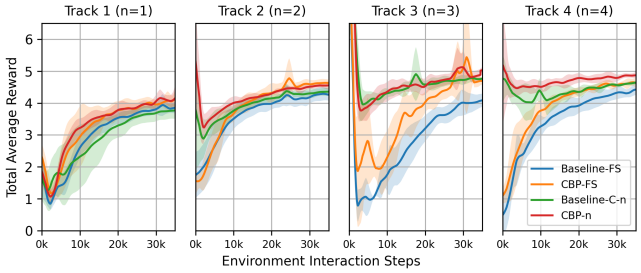

Fig. 3: Average total rewards of training in simulation. All configurations are averaged over 5 runs. Shaded areas indicate one SD distance.

In Figure 3 the average reward curves show the different learning patterns for the continual methods (i.e. IQL-n is omitted as it is not trained on one track at a time). While training on the first two tracks does not isolate clear, distinct patterns, the switch to a larger track layout and a different floor highlights the limitations of the non-continual method. On Track 4, CPB-4 exhibits a marginally higher average reward than Baseline-C-n. Furthermore, CBP-FS shows its ability to train more efficiently compared to the other method from scratch, Baseline-FS. Baseline-C-n also shows a significantly reduced plasticity: looking at the number of dead units of the actor in Figure 4, it can be clearly seen that Baseline-C-n flattens at a higher percentage of dead units, slightly above 70%, while CBP-n stays constantly lower, between 60% and 65%.

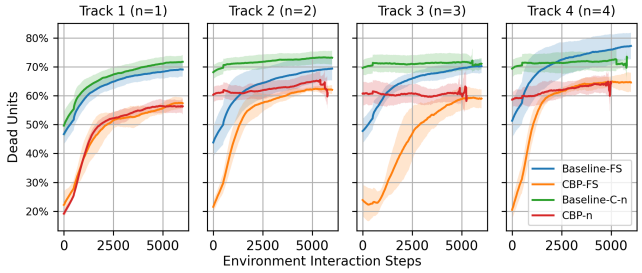

Fig. 4: Percentage of dead units in the actor network during training.

Furthermore, it is interesting to notice that depending on the track, the methods trained from scratch end at consistently different percentages of dead units: while CBP-FS is better than Baseline-FS on every track, both these methods show a higher count of dead units on Track 4 than on the other tracks. This result suggests that training on an arguably more complex track (Track 4) first might be detrimental, as it induces a higher loss of plasticity. Finally, the different methods are

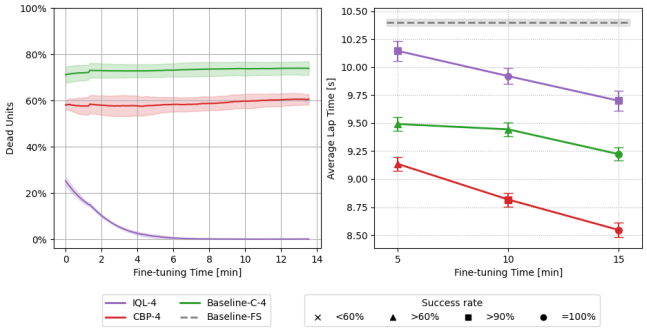

Fig. 5: Simulation results on the test track. Left: Percentage of dead units during the finetuning process. Right: Test lap times after 5, 10, and 15 minutes of training on Track 5.

fine-tuned for the same durations as on the physical platform: 5, 10, and 15 minutes. As can be seen in Figure 5, CBP-n manages to consistently outperform Baseline-C-n, resulting in fewer collisions and always averaging a lower lap time. As expected from the higher amount of dead units shown during pre-training, Baseline-C-n also shows less plasticity. In particular, the lap time reduction is much lower, only a 0.28s improvement (9.50s @ 5 min, 9.22 @ 15 min, 2.9% relative improvement) for Baseline-C-n, while CBP-n improves by 0.59s (9.13s @ 5 min, 8.54 @ 15 min, 6.5% relative improvement): for this reason, Baseline-C-n was not

further tested on the physical platform. IQL-n, on the other hand, reports the slowest lap time at all three testing points, but manages to do so with a safer policy at the earlier stages.

Analyzing the plasticity of the actors via the percentage of active units, the trend remains similar to pretraining, as both the continually trained methods face a relatively moderate increase in dead units. The offline RL method, instead, clearly shows a different pattern, by converging to full activation (0% dead units) and thus promising a highly plastic network. However, the final performance gap to CBP-n turns out to be a shortcoming for the fine-tuning phase.

B. On-car experimental setup

Similarly to the simulation setup, the models used for the real-world evaluation were pre-trained on the four physical tracks presented in Figure 1: the resulting continual policy is named CBP-4, and similarly, the offline pre-trained policy is IQL-4. Following Section III-A, the two policies are then finetuned for 15 minutes on the test track, yielding the final policies CBP-5 and IQL-4-CBP, respectively. All RL policies were trained for three different random seeds, and performance was evaluated after 5, 10, and 15 minutes. The track layout and the tire-floor combination were unseen. The friction was 30% to 39% lower than the training tracks, significantly impeding adaptation. Practically, Tracks 1 and 2 corresponded to a resin-coated floor, Tracks 3 and 4 corresponded to asphalt, and Track 5 was on polished concrete. Additionally, we trained a reference policy from scratch (CBP-FS) for 100'000 gradient steps ($\sim$ 50 minutes) and evaluated its final performance. We also compare our results to the two well-established model-based classical controllers MAP and Model Predictive Control (MPC), following the baseline implementation of [12]. For this comparison, system identification following the procedure of [50] was used, and basic tuning was conducted. All lap times at the end of fine-tuning are reported in Table III.

TABLE III: Lap time results after 15 minutes of fine-tuning on the test track. t_μ is the average lap time, $t_{\min}$ and $t_{\max}$ the minimum and maximum lap time respectively, and σ the standard deviation

Test run	t_μ [s]	$t_{\min}$ [s]	$t_{\max}$ [s]	σ [s]
Classical controllers				
MPC	11.50	11.29	11.68	0.095
MAP	10.97	10.75	11.12	**0.092**
From scratch	10.62	**9.94**	11.74	0.431
Continual RL	**10.44**	9.96	**10.99**	0.199
Offline pre-training	12.13	10.80	14.06	1.199

C. On-car results

Lap time evaluations are available in Figure 6. After 5 minutes of fine-tuning, one of the three continually learned policies reached MAP performance. After 10 minutes, this was the case on average, and after 15 minutes, all the policies beat the MAP controller. The fastest RL policy outperformed the classical baseline by 6.4%, which is a compatible value with the result achieved by [12] after 84 minutes. IQL-4-CBP showed similar results as in simulation, achieving

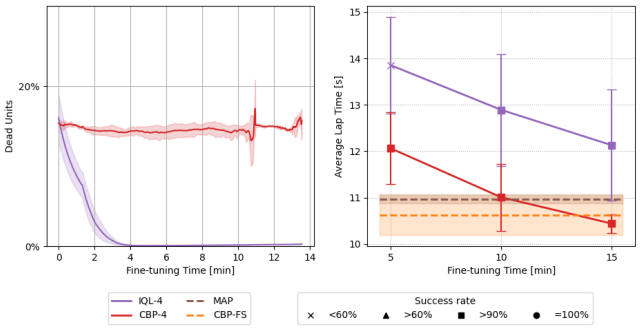

Fig. 6: Physical results on the test track. Left: Percentage of dead units during the finetuning process. Right: Test lap times after 5, 10, and 15 minutes of training on Track 5.

overall a lower performance than the continually trained one, despite its advantage of equal training on all experiences. Dead units metrics, also available in Figure 6, show a similar pattern as in simulation, particularly for IQL-4-CBP. More interestingly, CBP-5 shows a much lower amount of dead units when compared to simulation (around 18% instead of 60%), indicating that the real-world environment requires a much more complex representation. Almost the same performance, and even the overall lowest lap time, was reached by CBP-FS, albeit with a much longer time at disposal.

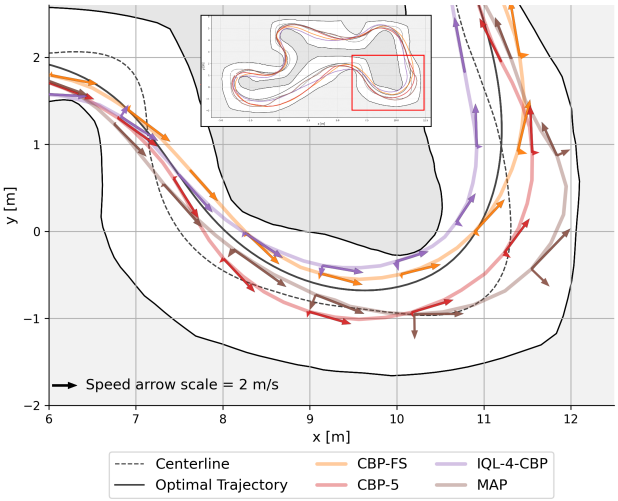

Fig. 7: Minimum lap time trajectories on Track 5. Arrows aligned with the trajectories indicate longitudinal velocity, and arrows perpendicular to the trajectory indicate lateral velocity.

The qualitative comparison of the trajectories in Figure 7 shows that the learned policies that were faster than the MAP controller often chose wider trajectories with lower curvature and managed to carry higher speed in the corners. CBP-FS, in particular, manages to carry the highest speed through the corner by exiting with a line much closer to the boundaries, possibly indicating a better specialization than CBP-5, which instead tackles the left-hander with a safer trajectory. Also,

MAP seems to commit to a too high velocity before the highlighted corners and needs to overcorrect, induce high slip (notice the higher lateral velocity), and overall exit the corner with a much lower speed than the RL policies. Finally, the offline pre-trained policy chooses a trajectory much closer to the interior boundary in tight curves, which is still possible at lower speeds when the car has not started slipping yet.

V. CONCLUSIONS

In this work, we propose a RoboRacer continual RL training framework to obtain a generalistic policy able to be finetuned on unseen terrains. Integrating SAC with CBP, L2 Init, and an adaptation of ERE, the proposed method manages to be finetuned on a track with lower friction than observed in pre-training, within 10-15 minutes (instead of 82 as previously [12]), and outperforms previous classical controllers such as [41]. Additionally, we explore offline pre-training with IQL, presenting a policy with promising plasticity, but reduced performance. Our findings concerning network plasticity strengthen the dead units metric proposed in [14] and further the correlation between it and improved network adaptation capabilities.

Overall, the method presented can be used to train a generalistic policy and to finetune it on a reasonably out-of-distribution task: such an approach could be adapted to other platforms, such as quadruped robots on unseen terrains, dexterous manipulators with unseen materials, etc., where accurate simulation might become impossible and training with real-world samples crucial. Different limitations could be further investigated, such as the loss of performance induced by offline RL pre-training, or the effect of even more sample-efficient RL algorithms (e.g. [25]).

REFERENCES

[1] Y. Tian *et al.*, "ELF opengo: an analysis and open reimplementation of alphazero," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2019, pp. 6244–6253. [Online]. Available: http://proceedings.mlr.press/v97/tian19a.html

[2] F. Shi *et al.*, "Circus anymal: A quadruped learning dexterous manipulation with its limbs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 2316–2323.

[3] K. Zhou *et al.*, "Adaptive Interactive Navigation of Quadruped Robots using Large Language Models," Mar. 2025, arXiv:2503.22942 [cs]. [Online]. Available: http://arxiv.org/abs/2503.22942

[4] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, Aug. 2023. [Online]. Available: https://www.nature.com/articles/s41586-023-06419-4

[5] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning Quadrupedal Locomotion over Challenging Terrain," *Science Robotics*, vol. 5, no. 47, p. eabc5986, Oct. 2020, arXiv:2010.11251 [cs]. [Online]. Available: http://arxiv.org/abs/2010.11251

[6] Y. Qin, B. Huang, Z.-H. Yin, H. Su, and X. Wang, "Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation," in *6th Annual Conference on Robot Learning*, 2022. [Online]. Available: https://openreview.net/forum?id=tJE1Yyi8fUX

[7] A. Afzal, D. S. Katz, C. Le Goues, and C. S. Timperley, "Simulation for robotics test automation: Developer perspectives," in *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, 2021, pp. 263–274.

[8] S. Kumar *et al.*, "Continual learning as computationally constrained reinforcement learning," *Found. Trends Mach. Learn.*, vol. 18, no. 5, p. 913–1053, Aug. 2025. [Online]. Available: https://doi.org/10.1561/2200000116

[9] M. Mammadov, "End-to-end Lidar-Driven Reinforcement Learning for Autonomous Racing," Sep. 2023, arXiv:2309.00296 [cs]. [Online]. Available: http://arxiv.org/abs/2309.00296

[10] K. Stachowicz, D. Shah, A. Bhorkar, I. Kostrikov, and S. Levine, "FastRLAP: A system for learning high-speed driving via deep RL and autonomous practicing," in *7th Annual Conference on Robot Learning*, 2023. [Online]. Available: https://openreview.net/forum?id=FRKBdXhkQE0

[11] M. M. Zarrar, Q. Weng, B. Yerjan, A. Soyyigit, and H. Yun, "TinyLidarNet: 2D LiDAR-based End-to-End Deep Learning Model for F1TENTH Autonomous Racing," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Abu Dhabi, United Arab Emirates: IEEE, Oct. 2024, pp. 2878–2884. [Online]. Available: https://ieeexplore.ieee.org/document/10801430/

[12] B. Hildisch, E. Ghignone, N. Baumann, C. Hu, A. Carron, and M. Magno, "Drive fast, learn faster: On-board RL for high performance autonomous racing," in *Reinforcement Learning Conference*, 2025. [Online]. Available: https://openreview.net/forum?id=4naF7E5J6n

[13] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *International Conference on Machine Learning (ICML)*, 2018.

[14] S. Dohare, J. F. Hernandez-Garcia, Q. Lan, P. Rahman, A. R. Mahmood, and R. S. Sutton, "Loss of plasticity in deep continual learning," *Nature*, vol. 632, no. 8026, pp. 768–774, Aug. 2024. [Online]. Available: https://www.nature.com/articles/s41586-024-07711-7

[15] I. Kostrikov, A. Nair, and S. Levine, "Offline Reinforcement Learning with Implicit Q-Learning," Oct. 2021, arXiv:2110.06169 [cs]. [Online]. Available: http://arxiv.org/abs/2110.06169

[16] S. Kumar, H. Marklund, and B. V. Roy, "Maintaining Plasticity in Continual Learning via Regenerative Regularization," Oct. 2024, arXiv:2308.11958 [cs]. [Online]. Available: http://arxiv.org/abs/2308.11958

[17] N. Rudin, J. He, J. Aurand, and M. Hutter, "Parkour in the wild: Learning a general and extensible agile locomotion policy using multi-expert distillation and rl fine-tuning," 2025. [Online]. Available: https://arxiv.org/abs/2505.11164

[18] J. Degrave *et al.*, "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nat.*, vol. 602, no. 7897, pp. 414–419, 2022. [Online]. Available: https://doi.org/10.1038/s41586-021-04301-9

[19] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, "Deep reinforcement learning for robotics: A survey of real-world successes," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 8, no. Volume 8, 2025, pp. 153–188, 2025. [Online]. Available: https://www.annualreviews.org/content/journals/10.1146/annurev-control-030323-022510

[20] F. Djeumou, M. Thompson, M. Suminaka, and J. Subosits, "Reference-free formula drift with reinforcement learning: From driving data to tire energy-inspired, real-world policies," 2024. [Online]. Available: https://arxiv.org/abs/2410.20990

[21] L. Da, J. Turnau, T. P. Kutralingam, A. Velasquez, P. Shakarian, and H. Wei, "A Survey of Sim-to-Real Methods in RL: Progress, Prospects and Challenges with

Foundation Models," Mar. 2025, arXiv:2502.13187 [cs]. [Online]. Available: http://arxiv.org/abs/2502.13187

[22] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *5th Annual Conference on Robot Learning*, 2021. [Online]. Available: https://openreview.net/forum?id=wK2fDDJ5VcF

[23] D. Abel, A. Barreto, B. V. Roy, D. Precup, H. v. Hasselt, and S. Singh, "A Definition of Continual Reinforcement Learning," Dec. 2023, arXiv:2307.11046 [cs]. [Online]. Available: http://arxiv.org/abs/2307.11046

[24] A. Bhatt *et al.*, "Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=PczQtTsTIX

[25] H. Lee *et al.*, "Simba: Simplicity bias for scaling up parameters in deep reinforcement learning," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=jXLiDKsuDo

[26] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[27] Y. Yuan and A. R. Mahmood, "Asynchronous Reinforcement Learning for Real-Time Control of Physical Robots," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 5546–5552.

[28] A. Raffin, J. Kober, and F. Stulp, "Smooth exploration for robotic reinforcement learning," in *5th Annual Conference on Robot Learning*, 2021. [Online]. Available: https://openreview.net/forum?id=TSuSGVkjuXd

[29] J. Rothfuss, B. Sukhija, L. Treven, F. Dörfler, S. Coros, and A. Krause, "Bridging the sim-to-real gap with bayesian inference," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 10 784–10 791.

[30] A. Brunnbauer *et al.*, "Latent imagination facilitates zero-shot transfer in autonomous racing," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 7513–7520.

[31] C. Pan *et al.*, "A Survey of Continual Reinforcement Learning," Jun. 2025, arXiv:2506.21872 [cs]. [Online]. Available: http://arxiv.org/abs/2506.21872

[32] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, *Experience replay for continual learning*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[33] S. Powers, E. Xing, E. Kolve, R. Mottaghi, and A. Gupta, "Cora: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents," in *Proceedings of The 1st Conference on Lifelong Learning Agents*, ser. Proceedings of Machine Learning Research, vol. 199. PMLR, 22–24 Aug 2022, pp. 705–743. [Online]. Available: https://proceedings.mlr.press/v199/powers22b.html

[34] C. Wang and K. Ross, "Boosting Soft Actor-Critic: Emphasizing Recent Experience without Forgetting the Past," Jun. 2019, arXiv:1906.04009 [cs]. [Online]. Available: http://arxiv.org/abs/1906.04009

[35] Z. Abbas, R. Zhao, J. Modayil, A. White, and M. C. Machado, "Loss of Plasticity in Continual Deep Reinforcement Learning," Mar. 2023, arXiv:2303.07507 [cs]. [Online]. Available: http://arxiv.org/abs/2303.07507

[36] G. Sokar, R. Agarwal, P. S. Castro, and U. Evci, "The Dormant Neuron Phenomenon in Deep Reinforcement Learning," Jun. 2023, arXiv:2302.12902 [cs]. [Online]. Available: http://arxiv.org/abs/2302.12902

[37] Y. Song, H. Lin, E. Kaufmann, P. Dürr, and D. Scaramuzza, "Autonomous overtaking in gran turismo sport using curriculum reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 9403–9409.

[38] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," in *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, ser. Proceedings of Machine Learning Research, H. J. Escalante and R. Hadsell, Eds., vol. 123. PMLR, 08–14 Dec 2020, pp. 77–89. [Online]. Available: https://proceedings.mlr.press/v123/o-kelly20a.html

[39] R. Coulter, "Implementation of the pure pursuit path tracking algorithm," Pittsburgh, Pennsylvania, Jan 1990.

[40] A. Liniger, A. Domahidi, and M. Morari, "Optimization-Based Autonomous Racing of 1:43 Scale RC Cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, Sep. 2015, arXiv:1711.07300 [math]. [Online]. Available: http://arxiv.org/abs/1711.07300

[41] J. Becker, N. Imholz, L. Schwarzenbach, E. Ghignone, N. Baumann, and M. Magno, "Model- and Acceleration-based Pursuit Controller for High-Performance Autonomous Racing," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 5276–5283, arXiv:2209.04346 [cs]. [Online]. Available: http://arxiv.org/abs/2209.04346

[42] B. D. Evans *et al.*, "Unifying F1TENTH Autonomous Racing: Survey, Methods and Benchmarks," Apr. 2024, arXiv:2402.18558 [cs]. [Online]. Available: http://arxiv.org/abs/2402.18558

[43] J. Betz *et al.*, "Tum autonomous motorsport: An autonomous racing software for the indy autonomous challenge," *Journal of Field Robotics*, vol. 40, no. 4, pp. 783–809, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22153

[44] A. Raji *et al.*, "er.autopilot 1.1: A software stack for autonomous racing on oval and road course tracks," *IEEE Transactions on Field Robotics*, vol. 1, pp. 332–359, 2024.

[45] P. R. Wurman *et al.*, "Outracing champion Gran Turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, Feb. 2022. [Online]. Available: https://www.nature.com/articles/s41586-021-04357-7

[46] M. Bosello, R. Tse, and G. Pau, "Train in Austria, Race in Montecarlo: Generalized RL for Cross-Track F1^{tenth} LIDAR-Based Races," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. Las Vegas, NV, USA: IEEE, Jan. 2022, pp. 290–298. [Online]. Available: https://ieeexplore.ieee.org/document/9700730/

[47] N. Siegel *et al.*, "Keep doing what worked: Behavior modelling priors for offline reinforcement learning," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rke7geHtwH

[48] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin, "Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble," in *5th Annual Conference on Robot Learning*, 2021. [Online]. Available: https://openreview.net/forum?id=AlJXhEI6J5W

[49] N. Baumann *et al.*, "ForzaETH Race Stack – Scaled Autonomous Head-to-Head Racing on Fully Commercial off-the-Shelf Hardware," *Journal of Field Robotics*, p. rob.22429, Sep. 2024, arXiv:2403.11784 [cs]. [Online]. Available: http://arxiv.org/abs/2403.11784

[50] O. Dikici *et al.*, "Learning-based on-track system identification for scaled autonomous racing in under a minute," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 1984–1991, 2025.