Contextualized Hybrid Ensemble Q-learning: Learning Fast with Control Priors

Emma Cramer,[*] Bernd Frauenknecht,[*] Ramil Sabirov,[*] Sebastian Trimpe
Institute for Data Science in Mechanical Engineering
RWTH Aachen University
52068 Aachen, Germany
emma.cramer@dsme.rwth-aachen.de

Abstract

Combining Reinforcement Learning (RL) with a prior controller can yield the best out of two worlds: RL can solve complex nonlinear problems, while the control prior ensures safer exploration and speeds up training. Prior work largely blends both components with a fixed weight, neglecting that the RL agent's performance varies with the training progress and across regions in the state space. Therefore, we advocate for an adaptive strategy that dynamically adjusts the weighting based on the RL agent's current capabilities. We propose a new adaptive hybrid RL algorithm, *Contextualized Hybrid Ensemble Q-learning* (CHEQ). CHEQ has three key ingredients: (i) a time-invariant formulation of the adaptive hybrid RL problem treating the adaptive weight as a context variable, (ii) a weight adaption mechanism based on the parametric uncertainty of a critic ensemble, and (iii) ensemble-based acceleration for data-efficient RL. Evaluating CHEQ on a car racing task reveals substantially stronger data efficiency, exploration safety, and transferability to unknown scenarios than state-of-the-art adaptive hybrid RL methods.

This paper has been reviewed and accepted at the *Reinforcement Learning Conference (RLC)*, to be held in August 2024.

1 Introduction

Deep reinforcement learning (RL) methods have shown great success in challenging control problems such as gameplay (Mnih et al., 2015; Silver et al., 2018a; OpenAI et al., 2019) and robotic manipulation (Gupta et al., 2021; Büchler et al., 2022). Despite the great potential of RL methods, their data inefficiency, unstructured exploration behavior, and inability to generalize to unknown scenarios represent a significant hurdle to their application to real-world problems.

A prime reason for limited real-world applications is the task-agnostic architecture of state-of-the-art RL approaches (Schulman et al., 2017; Haarnoja et al., 2018) that does not incorporate prior knowledge on how to solve the task at hand. In contrast, control theory provides a rich set of methods for deriving near-optimal controllers in many applications. This motivates the drive for hybrid RL methods (Silver et al., 2018b; Johannink et al., 2019) that blend control priors with deep RL policies. Hybrid algorithms thus combine the prior controller's generalization capabilities and informed behavior with the power of deep RL for solving general nonlinear problems.

Notwithstanding the conceptual benefit of hybrid RL formulations, how to systematically combine the control prior with the RL agent largely remains an open problem. The majority of prior work (Silver et al., 2018b; Johannink et al., 2019; Schoettler et al., 2020; Ceola et al., 2024) proposes a fixed

[*]These authors contributed equally.

Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2024).

weighting between the control prior and the RL agent. A fixed blending, however, disregards the fact that the capability of the RL agent depends on training time and state. In general, as more data is observed, the RL agent improves its behavior, ultimately outperforming the control prior in large portions of the domain. The core idea of our approach is to adapt the weighting between RL agent and control prior based on the agent's confidence. As the RL agent improves over time, this induces a time-variant weighting mechanism. This time dependency leads to structural problems of prior formulations in uncertainty-adapted hybrid RL (Cheng et al., 2019; Rana et al., 2023).

We provide a unified view on hybrid RL that allows us to classify prior work within a general framework. Analyzing this framework highlights the necessity for a novel adaptive hybrid RL formulation with descriptive, time-invariant dynamics. We define the contextualized hybrid Markov decision process (MDP), introducing the adaptive weight as a context variable. Building upon this formulation, we propose the *Contextualized Hybrid Ensemble Q-learning* (CHEQ) algorithm that systematically adapts the weighting between RL agent and control prior based on an uncertainty estimate of a critic ensemble. CHEQ combines the contextualized hybrid RL formulation with uncertainty-based weight adaption and existing ensemble-based acceleration techniques for data-efficient RL.

We evaluate our algorithm on a racing task (Schier et al., 2023), which requires operating a car close to its stability limits in order to achieve maximum return. We find that compared to prior work in adaptive hybrid RL, the CHEQ algorithm shows (i) reduced failures during training, (ii) increased sample efficiency, and (iii) improved transfer behavior on unseen race tracks.

In summary, our main contributions are:

- A unified framework that allows us to classify existing approaches and reveal key limitations.
- A hybrid MDP formulation, introducing the adaptive weight as a context variable and thus addressing structural problems of prior work in hybrid RL with adaptive weighting.
- A novel hybrid RL algorithm, CHEQ, that systematically adapts the weighting between RL agent and control prior based on Q-ensemble disagreement.

2 Related Work

This section discusses relevant prior work combining RL and a control prior. We distinguish two types; hybrid RL with fixed and adaptive weighting between the RL agent and controller.

Hybrid Reinforcement Learning with Fixed Weighting: Two concurrent works (Silver et al., 2018b; Johannink et al., 2019) first combined RL and a control prior and introduced the term residual RL. In residual RL, the control prior is assumed to be fixed, and the RL agent learns a residual on top of this. In this work, we use the general term hybrid RL to include approaches that adapt the controller's weight. Silver et al. (2018a) and Johannink et al. (2019) show advantages of hybrid RL, such as sample efficiency, improved sim-to-real transfer, and robustness towards uncertainties. Hybrid RL with fixed weights has then successfully been applied to real robot insertion tasks (Schoettler et al., 2020), peg insertion under uncertainty (Ranjbar et al., 2021), driving (Kerbel et al., 2022) and to learn a residual RL policy on top of a pre-trained RL agent (Ceola et al., 2024). A fixed mixing, however, does not allow one to consider the improving capabilities of the RL agent.

Hybrid Reinforcement Learning with Adaptive Weighting: Daoudi et al. (2023) assume a given controller confidence function, employing a controller in instances of high confidence and an RL agent in other scenarios. Our work focuses on the RL agent's confidence and proposes to estimate the confidence based on a critic ensemble. Similar to our approach, Hoel et al. (2020a,b) train an ensemble of bootstrapped Q-networks for a driving task with discrete actions. They evaluate the uncertainty as the coefficient of variation of Q-estimates and resort to safe fallback actions in case of high uncertainty. However, they do not combine controller and RL agent but switch between both. In this work, we investigate a seamless blending approach for continuous control. Rana et al. (2020b) estimate the policy uncertainty using Monte-Carlo dropout and based on this uncertainty either sample from a residual policy or the controller alone. Rana et al. (2020a) directly fuse a prior control distribution with an RL policy in a multiplicative fashion and anneal the influence of the control prior over training time. Rana et al. (2023) use a policy ensemble to estimate how certain the RL agent is in the current action. The combined action is then computed as the Bayesian posterior of control prior and policy distribution. Cheng et al. (2019) use the TD-error as an uncertainty estimate

and combine controller and RL agent based on this. Both Rana et al. (2023) and Cheng et al. (2019) base their adaption mechanism on a form of policy uncertainty. Both approaches train based on the combined action, which becomes brittle when facing large distributional shifts. We further discuss this limitation in Section 5.

3 Background

The following introduces the key components and the general concept of hybrid RL.

Reinforcement Learning RL is a method for solving sequential decision problems based on the interaction between an agent and an environment (Sutton & Barto, 2018). The environment is modeled as a discounted Markov decision process defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \rho_0, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, and start state distribution ρ_0. The commonly unknown transition function $p(\mathbf{s}_{t+1}, r_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}})$ describes transitions between states $\mathbf{s}_t \in \mathcal{S}$ and actions $\mathbf{a}_t^{\mathrm{RL}} \in \mathcal{A}$. During transitions, rewards $r_t \in \mathbb{R}$ are emitted according to a reward function $r_{t+1} \sim r(\mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}})$. The objective of the RL agent is to learn a policy $\pi^{\mathrm{RL}}(\mathbf{a}_t^{\mathrm{RL}} \mid \mathbf{s}_t)$ that maximizes the expected cumulative sum of rewards discounted by $\gamma \in (0, 1)$. This results in the RL objective

$$J(\pi^{\mathrm{RL}}) = \max_{\pi^{\mathrm{RL}}} \mathbb{E}_{\pi^{\mathrm{RL}}, \mathcal{M}} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right]. \tag{1}$$

The discounted sum of rewards is referred to as return and is accumulated along trajectories under the policy π^{RL} and the environment MDP $\mathcal{M}$. State value functions condition expected return on a particular state $V^{\pi^{\mathrm{RL}}}(\mathbf{s}_t) = \mathbb{E}_{\pi^{\mathrm{RL}}, \mathcal{M}} \left[\sum_{k=t}^{\infty} \gamma^{k-t} r_{k+1} \mid \mathbf{s}_t \right]$ while, action value or Q-functions condition expected return on specific state action pairs $Q^{\pi^{\mathrm{RL}}}(\mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}) = \mathbb{E}_{\pi^{\mathrm{RL}}, \mathcal{M}} \left[\sum_{k=t}^{\infty} \gamma^{k-t} r_{k+1} \mid \mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}} \right]$.

Control Prior The prior policy $\pi^{\mathrm{prior}}(\mathbf{a}_t^{\mathrm{prior}} \mid \mathbf{s}_t)$ represents prior knowledge for solving the RL objective (1), while typically not providing the optimal policy over the whole domain $\mathcal{S} \times \mathcal{A}$. This work focuses on control priors based on classic control theory. These can be derived with limited effort in many applications and often provide a good baseline for interaction with $\mathcal{M}$. We assume a control prior that is time-invariant and without an internal state.

Hybrid Reinforcement Learning: Hybrid RL combines the control prior and the RL agent by blending their actions via some mixing function $\mathbf{a}_t^{\mathrm{mix}} = f(\mathbf{a}_t^{\mathrm{prior}}, \mathbf{a}_t^{\mathrm{RL}}, \boldsymbol{\lambda}_t)$ depending on a weight $\boldsymbol{\lambda}_t$.

4 A Unified View on Hybrid Reinforcement Learning

Next, we develop a unified view on hybrid RL that allows classifying prior methods (cf. Section 2). In the standard RL setup depicted in Figure 1a, the RL agent π^{RL} interacts with the time-invariant MDP $\mathcal{M}$ with dynamics $p(\mathbf{s}_{t+1}, r_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}})$ that represents the controlled system. Hybrid RL (see Figure 1b) incorporates a control prior π^{prior} which requires reformulating the standard framework. Here, π^{RL} and π^{prior} apply a combined action $\mathbf{a}_t^{\mathrm{mix}}$ to $\mathcal{M}$. The mixing function f generates a combined action by blending the individual actions based on a weighting vector $\boldsymbol{\lambda}_t$ provided by a weight adaption function Λ. Within this generalized framework, prior work in hybrid RL can be categorized based on the choice of f and Λ.

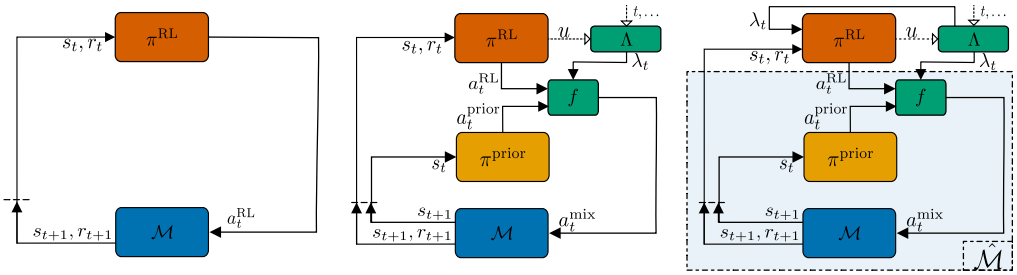
4.1 Mixing Function f

We consider mixing functions based on a weighted sum with a weighting vector $\boldsymbol{\lambda}_t = [\lambda_t^{\mathrm{prior}}, \lambda_t^{\mathrm{RL}}]^\top$

$$\mathbf{a}_t^{\mathrm{mix}} = f(\mathbf{a}_t^{\mathrm{prior}}, \mathbf{a}_t^{\mathrm{RL}}, \boldsymbol{\lambda}_t) = \lambda_t^{\mathrm{prior}} \cdot \mathbf{a}_t^{\mathrm{prior}} + \lambda_t^{\mathrm{RL}} \cdot \mathbf{a}_t^{\mathrm{RL}}. \tag{2}$$

This formulation allows to distinguish a *residual* and a *regularized* setting.

In the residual setting, $\lambda_t^{\mathrm{prior}}$ is typically constant while λ_t^{RL} can be variable $\boldsymbol{\lambda}_t = [1, \lambda_t^{\mathrm{RL}}]^\top$ (Silver et al., 2018b; Johannink et al., 2019; Schoettler et al., 2020). Thus, the RL agent interacts with the closed control loop between π^{prior} and $\mathcal{M}$ and learns a residual action on top of $\mathbf{a}_t^{\mathrm{prior}}$. Consequently, $\boldsymbol{\lambda}_t$ modulates the RL agent's impact on the closed loop dynamics. As the control prior is not scaled



(a) (b) (c)

Figure 1: A standard RL setting (a), hybrid RL settings from prior work (b) and our contextualized hybrid RL setting based on RL action $\mathbf{a}_t^{\mathrm{RL}}$ and weighting factor $\boldsymbol{\lambda}_t^{\mathrm{RL}}$ (c).

down, it might interpret the RL agent as a disturbance and counteract it (Ranjbar et al., 2021), which can limit the overall performance of residual formulations.

In the regularized setting, both weights are adaptable such that $\lambda_t^{\mathrm{prior}} + \lambda_t^{\mathrm{RL}} = 1$ (Cheng et al., 2019; Rana et al., 2023). This results in a mixing function of the form

$$\mathbf{a}_t^{\mathrm{mix}} = f(\mathbf{a}_t^{\mathrm{prior}}, \mathbf{a}_t^{\mathrm{RL}}, \lambda_t^{\mathrm{RL}}) = (1 - \lambda_t^{\mathrm{RL}}) \cdot \mathbf{a}_t^{\mathrm{prior}} + \lambda_t^{\mathrm{RL}} \cdot \mathbf{a}_t^{\mathrm{RL}}. \tag{3}$$

with $\lambda_t^{\mathrm{RL}} \in [0, 1]$. In the limit $\lambda_t^{\mathrm{RL}} = 0$, the control prior interacts with $\mathcal{M}$ without the interference of the RL agent, while the regularized setting reduces to the standard RL problem for $\lambda_t^{\mathrm{RL}} = 1$. Thus, λ_t^{RL} indicates not only the impact of $\mathbf{a}_t^{\mathrm{RL}}$ but also whether the RL agent interacts with the open loop dynamics of $\mathcal{M}$ or the closed loop dynamics as in the residual setting. Consequently, the control prior can be interpreted as a regularization of the RL agent. Our proposed algorithm operates in the regularized setting, allowing the agent to take over complete control when $\lambda_t^{\mathrm{RL}} = 1$.

4.2 Weight Adaption Function Λ

Hybrid RL approaches can further be classified, based on the choice of the weight adaption function Λ modulating the weighting vector $\boldsymbol{\lambda}_t$ of the mixing function f.

A large body of work, which we refer to as *fixed-weight hybrid RL* (Silver et al., 2018b; Johannink et al., 2019; Schoettler et al., 2020; Ranjbar et al., 2021; Ceola et al., 2024) chooses $\boldsymbol{\lambda}_t$ fixed throughout training. Neglecting the time- and state-dependent capabilities of the RL agent.

Approaches that adapt $\boldsymbol{\lambda}_t$, which we refer to as *adaptive hybrid RL* methods, rely on different mechanisms. Scheduling approaches (Rana et al., 2020a) change the weight explicitly with time, i.e. $\boldsymbol{\lambda}_t = \Lambda(t)$, typically increasing the weight of the RL agent as training progresses. Domain-based approaches (Kulkarni et al., 2022; Daoudi et al., 2023) adapt the weights based on the point of operation within the domain $\mathcal{S} \times \mathcal{A}$, i.e. $\boldsymbol{\lambda}_t = \Lambda(\mathbf{s}_t, \mathbf{a}_t)$. Uncertainty-based approaches (Cheng et al., 2019; Rana et al., 2023) adapt the weight based on the confidence of the RL agent, indicated by an uncertainty estimate $u(\mathbf{s}_t, \mathbf{a}_t, t)$, giving more weight to the RL agent when it has high confidence. Thus, they aim to leverage the benefits of the RL agent whenever possible, while resorting to a safe controller in situations where the RL agent has not seen enough data. The time dependency of the uncertainty estimate, however, increases the complexity of the hybrid RL setting, requiring a reformulation of the learning problem. In Section 5, we discuss the shortcomings of prior formulations and propose our own.

5 Contextualized Hybrid Reinforcement Learning

In Section 5.1, we propose a novel contextualized formulation of the adaptive hybrid RL problem and illustrate its benefits over prior approaches in Section 5.2. Based on that framework, we propose the Contextualized Hybrid Ensemble Q-learning (CHEQ) algorithm in Section 5.3.

5.1 General Concept of Contextualized Hybrid Reinforcement Learning

Based on the unified view provided in Section 4, we propose a general formulation for the hybrid RL problem we call *contextualized hybrid RL*.

The environment in the hybrid setting not only consists of the controlled system $\mathcal{M}$ but also comprises the control prior, the mixing function, and the weight adaption function. We consider both the control prior and the mixing function to be time-invariant. In contrast, the weight adaption function Λ can have time-varying behavior, i.e. $\boldsymbol{\lambda}_t = \Lambda(t, \dots)$, as discussed in Section 4.2. This leads to time-varying dynamics of the hybrid environment, violating the assumption of time-invariance in the MDP formulation (Bellemare et al., 2023). Instead, we exclude Λ from the definition of the hybrid environment and introduce the adaptive weight vector $\boldsymbol{\lambda}_t$ as a context variable to the agent and the environment (see Figure 1c). We model the hybrid environment introducing the contextualized hybrid MDP $\hat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{W}, \hat{p}, r, \rho_0, \gamma)$ with $\mathcal{W}$ the set of weighting vectors $\boldsymbol{\lambda}_t$ and the contextualized dynamics function $\hat{p}(\mathbf{s}_{t+1}, r_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}, \boldsymbol{\lambda}_t)$.

The MDP formulation $\hat{\mathcal{M}}$ induces the contextualized hybrid RL objective

$$\hat{J}(\pi^{\mathrm{RL}}) = \max_{\pi^{\mathrm{RL}}} \mathbb{E}_{\pi^{\mathrm{RL}}, \hat{\mathcal{M}}} \left[\sum_{t=0}^{T} \gamma^t r_{t+1} \right], \tag{4}$$

which enforces to learn a policy $\hat{\pi}^{\mathrm{RL}}(\mathbf{a}_t^{\mathrm{RL}} \mid \mathbf{s}_t^{\mathrm{RL}}, \boldsymbol{\lambda}_t)$ that maximizes expected return in $\hat{\mathcal{M}}$. Introducing $\boldsymbol{\lambda}_t$ as a context variable further yields the contextualized hybrid value functions $\hat{V}^{\pi^{\mathrm{RL}}}(\mathbf{s}_t, \boldsymbol{\lambda}_t) = \mathbb{E}_{\pi^{\mathrm{RL}}, \hat{\mathcal{M}}} \left[\sum_{k=t}^{\infty} \gamma^{k-t} r_{k+1} \mid \mathbf{s}_t, \boldsymbol{\lambda}_t \right]$ and $\hat{Q}^{\pi^{\mathrm{RL}}}(\mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}, \boldsymbol{\lambda}_t) = \mathbb{E}_{\pi^{\mathrm{RL}}, \hat{\mathcal{M}}} \left[\sum_{k=t}^{\infty} \gamma^{k-t} r_{k+1} \mid \mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}, \boldsymbol{\lambda}_t \right]$. Thus, we can optimize (4) using standard RL methods by additionally conditioning on $\boldsymbol{\lambda}_t$. The general mechanism of contextualized hybrid RL is illustrated in Algorithm 1.

Algorithm 1 Contextualized Hybrid Reinforcement Learning

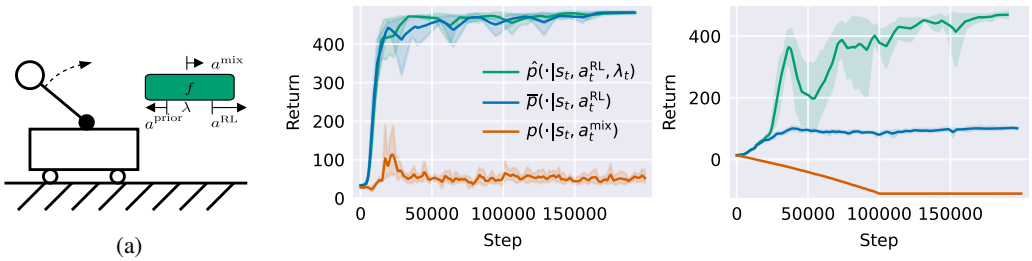
Require: RL policy $\hat{\pi}_\phi^{\mathrm{RL}}(\mathbf{a}_t^{\mathrm{RL}} \mid \mathbf{s}_t^{\mathrm{RL}}, \boldsymbol{\lambda}_t)$, control prior $\pi^{\mathrm{prior}}(\mathbf{a}_t^{\mathrm{prior}} \mid s)$, mixing function $f(\mathbf{a}_t^{\mathrm{RL}}, \mathbf{a}_t^{\mathrm{prior}}, \boldsymbol{\lambda}_t)$, weight adaption function Λ, replay buffer $\mathcal{D} \leftarrow \emptyset$.

```
for each episode do
    Sample initial state s_0 ~ ρ_0, initialize λ_0.
    for each step do
        Sample RL action a_t^RL ~ π_φ^RL (a_t^RL | s_t, λ_t).
        Sample control prior action a_t^prior ~ π_φ^prior (a_t^RL | s_t).
        Get combined action a_t^mix = f(a_t^RL, a_t^prior, λ_t).
        Observe state transition s_{t+1}, r_{t+1} ~ p(·, · | s_t, a_t^mix).
        Store (s_t, a_t^RL, λ_t, s_{t+1}, r_{t+1}) into replay buffer D.
        Get next adaptive weight λ_{t+1} = Λ.
        Sample set of transitions (s, a, λ, s', r) ~ D.
        Optimize φ with respect to (4) using RL with sampled transitions.
    end for
end for
```

Prior work takes different approaches to formulating the hybrid learning problem. Approaches with a time-invariant weight adaption function, such as fixed-weight hybrid methods, include Λ in the definition of the environment MDP $\bar{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \bar{p}, r, \rho_0, \gamma)$ with dynamics $\bar{p}(\mathbf{s}_{t+1}, r_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}})$. This formulation is directly applicable to the standard RL objective (1), however, does not generalize to time-varying adaption mechanisms such as uncertainty-adapted methods. Approaches with a time-varying adaption mechanism (Cheng et al., 2019; Rana et al., 2023) typically formulate the hybrid RL problem concerning the controlled system MDP and the combined action with dynamics $p(\mathbf{s}_{t+1}, r_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t^{\mathrm{mix}})$. This likewise yields a formulation that is directly applicable to (1), however, this can lead to problems as the agent is unaware of the downstream mixing process. Furthermore, this introduces a distributional shift between trained policy and data-collecting behavior policy. The distributional shift can lead to training instability and divergence (**?**Fujimoto et al., 2018).

5.2 Illustrative Example

We exemplify the strength of the contextualized hybrid RL formulation based on $\hat{\mathcal{M}}$ introduced in Section 5.1 by comparing it to prior approaches on the cart pole system depicted in Figure 2a. The



(a) (b) (c)

Figure 2: We illustrate different hybrid RL formulations on a cart pole system (a) with a biased control prior, pushing to the left. Return of hybrid agents with fixed λ_t^{RL} (b) and variable λ_t^{RL} (c). Only the contextualized hybrid RL agent observing dynamics $\hat{p}$ can cope with both scenarios.

goal is to balance the pole upright while keeping the cart close to its initial position. The system is controlled via continuous forces on the cart. We choose π^{prior} to apply a constant force to the left, which destabilizes formulations unconscious of the mixing process. We investigate a time-invariant fixed weight setting as well as a time-varying schedule-based weight adaption setting to highlight the capability of the respective formulations to deal with both scenarios. This simple example illustrates that the contextualized hybrid MDP formulation can deal with destabilizing control priors and time-varying weight adaption functions while prior formulations fail.

Fixed Weighting: First, we consider a residual setting with fixed weights $\lambda_t^{\text{RL}} = \lambda_t^{\text{prior}} = 0.5$. Figure 2b depicts the performance of RL agents trained under $\hat{\mathcal{M}}$, $\bar{\mathcal{M}}$, and $\mathcal{M}$ with respective dynamics $\hat{p}(\mathbf{s}_{t+1}, r_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t^{\text{RL}}, \boldsymbol{\lambda}_t)$, $\bar{p}(\mathbf{s}_{t+1}, r_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t^{\text{RL}})$, and $p(\mathbf{s}_{t+1}, r_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t^{\text{mix}})$. While agents trained under $\hat{\mathcal{M}}$ and $\bar{\mathcal{M}}$ learn to stabilize the cart pole, the hybrid formulation concerning $\mathcal{M}$ fails. When formulating the RL problem concerning $\mathbf{a}_t^{\text{mix}}$ the RL agent observes the combined action in its data and therefore learns the combined action in its policy. This, however, neglects the fact that the policy action is mixed with the controller action before being applied to $\mathcal{M}$. Assuming the cart pole is not moving, and the pole is upright, an agent trained under $\mathcal{M}$ provides the optimal combined action, namely applying no force, while $\mathbf{a}_t^{\text{prior}}$ pushes the pole to the left. This results in $\mathbf{a}_t^{\text{mix}}$ pointing to the left, causing the pole to fall while giving the agent no mechanism to observe and counteract this phenomenon. Instead, formulating the hybrid RL problem concerning $\mathbf{a}_t^{\text{RL}}$ allows the agent to observe the mixing mechanism and compensate for the destabilizing control prior.

Adaptive Weighting: Second, we consider an adaptive hybrid RL problem with time-varying Λ. We choose a schedule-based approach with a regularizing mixing function (3) and $\lambda_t^{\text{RL}} \in [0, 1]$ linearly increasing over time. Figure 2c shows the performance of agents trained under formulations based on $\hat{\mathcal{M}}$, $\bar{\mathcal{M}}$, and $\mathcal{M}$. While agents trained under $\bar{\mathcal{M}}$ and $\mathcal{M}$ fail, the formulation based on $\hat{\mathcal{M}}$ succeeds. In the beginning, when the RL agent is given only low weight, the formulation under $\mathcal{M}$ suffers from a high distributional shift between the action of the behavior policy $\mathbf{a}_t^{\text{mix}}$ and the action of the target policy $\mathbf{a}_t^{\text{RL}}$. The large distributional shift causes the agent to diverge (**?**Fujimoto et al., 2018). Although the distributional shift decreases with increasing weight lambda, the agent does not manage to recover. The formulation under $\bar{\mathcal{M}}$ fails due to the missing information about the time-variant behavior of the mixing process. The proposed contextualized hybrid RL formulation solves these issues by formulating the task concerning $\mathbf{a}_t^{\text{RL}}$ and introducing the context variable $\boldsymbol{\lambda}_t$.

5.3 Contextualized Hybrid Ensemble Q-learning (CHEQ)

Based on the contextualized hybrid RL formulation introduced in Section 5.1, we propose the Contextualized Hybrid Ensemble Q-learning (CHEQ) algorithm. At the heart of CHEQ is a critic ensemble that (i) provides an uncertainty estimate enabling an uncertainty-adapted hybrid RL mechanism, and (ii) allows to incorporate ensemble-based acceleration techniques for data-efficient RL.

We base CHEQ on the Soft Actor-Critic (SAC) (Haarnoja et al., 2018) algorithm and a regularizing mixing function (3) with $\boldsymbol{\lambda}_t = [(1 - \lambda_t^{\text{RL}}), \lambda_t^{\text{RL}}]^\top$. The weight adaption mechanism relies on a critic ensemble comprising of E contextualized Q-functions with parameters θ_e, $e \in \{1, \ldots, E\}$ and corresponding target Q-functions with parameters $\bar{\theta}_e$, $e \in \{1, \ldots, E\}$. We update the critics with the mechanism of Randomized Ensemble Double Q-learning (REDQ) (Chen et al., 2021)

and enforce sufficient independence between Q-estimates using Bernoulli masking of the training data (Osband et al., 2016; Lee et al., 2021; Mai et al., 2022). Model ensembles estimate parametric uncertainty, referred to as epistemic uncertainty, from disagreement between individual models within the ensemble. If different critics disagree about the outcome of taking action $\mathbf{a}_t^{\mathrm{RL}}$ in $\mathbf{s}_t$ while weighting with $\boldsymbol{\lambda}_t$, this indicates a weak understanding of the task in the particular area of $\mathcal{S} \times \mathcal{A} \times \mathcal{W}$. Thus, the control prior should be prioritized over the RL agent in such situations. Therefore, epistemic uncertainty represents a suitable quantity for adapting the weighting of control prior and RL agent. We define epistemic uncertainty as the standard deviation of critic predictions

$$u(\mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}, \lambda_t^{\mathrm{RL}}) = \sqrt{\frac{1}{E} \sum_{e=1}^{E} \left(\hat{Q}_{\theta_e}(\mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}, \lambda_t^{\mathrm{RL}}) - \mu(\mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}, \lambda_t^{\mathrm{RL}}) \right)^2} \tag{5}$$

with $\mu(\mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}, \lambda_t^{\mathrm{RL}}) = \frac{1}{E} \sum_{e=1}^{E} \hat{Q}_{\theta_e}(\mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}, \lambda_t^{\mathrm{RL}})$. We aim to give low weight to the RL agent in areas of high uncertainty and vice versa. Thus, the weight adaption function $\Lambda(u(\mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}, \lambda_t^{\mathrm{RL}}))$ maps the critics epistemic uncertainty to the weighting factor $\lambda_t^{\mathrm{RL}} \in [\lambda_{\min}, \lambda_{\max}] \subseteq [0, 1]$ via the piece-wise linear function

$$\lambda_{t+1}^{\mathrm{RL}} = \begin{cases} \lambda_{\max} & \text{if } u(\mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}, \lambda_t^{\mathrm{RL}}) < u_{\min} \\ \frac{u(\mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}, \lambda_t^{\mathrm{RL}}) - u_{\max}}{u_{\min} - u_{\max}} (\lambda_{\max} - \lambda_{\min}) + \lambda_{\min} & \text{if } u(\mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}, \lambda_t^{\mathrm{RL}}) \in [u_{\min}, u_{\max}] \\ \lambda_{\min} & \text{if } u(\mathbf{s}_t, \mathbf{a}_t^{\mathrm{RL}}, \lambda_t^{\mathrm{RL}}) > u_{\max}. \end{cases} \tag{6}$$

Besides providing an uncertainty estimate of the RL agent, the critic ensemble used in CHEQ has proven effective in mitigating overestimation bias (Thrun & Schwartz, 1993) in Q-learning-based approaches (Lan et al., 2021; Wang et al., 2021; Chen et al., 2021). The Update-To-Data (UTD) ratio describes the number of gradient steps per environment interaction. Due to the reduction of the overestimation bias, the critic ensemble allows for increasing the UTD ratio while maintaining stable learning. This substantially improves the data efficiency of value-based actor-critic methods (Chen et al., 2021). A detailed pseudocode algorithm of CHEQ is provided in Algorithm 2 of Appendix A.

6 Experiments

We evaluate CHEQ on a racing task and compare it to standard RL, fixed-weighting hybrid RL, and state-of-the-art adaptive hybrid RL. In our experiments, CHEQ yields substantial improvements in (i) data efficiency compared to other hybrid methods, as well as (ii) exploration safety, and (iii) zero-shot transferability to unknown scenarios as compared to all competitor approaches.

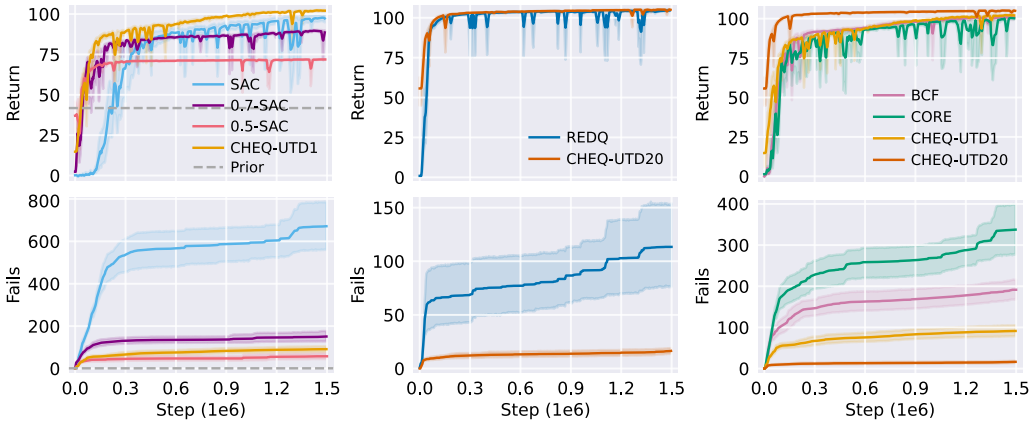
6.1 Experimental Setup

We base our evaluation on a car racing setting adapted from (Schier et al., 2023). Achieving high returns requires advanced trajectory planning and control while operating the vehicle close to stability limits, including tire slip. The control prior is a trajectory-following task along the center line of the track using a Stanley controller (Thrun et al., 2006) for lateral and a proportional controller for longitudinal control. Further details are provided in Appendix B.

We compare CHEQ against the standard RL approaches SAC (Haarnoja et al., 2018) and REDQ (Chen et al., 2021), fixed-weighting hybrid RL based on SAC, and the state-of-the-art adaptive hybrid RL methods Controller Regularized RL (CORE) (Cheng et al., 2019) and Bayesian Controller Fusion (BCF) (Rana et al., 2023). In all experiments, we provide results for CHEQ with a high UTD ratio (CHEQ-UTD20) to demonstrate the capabilities of the approach and a low UTD ratio (CHEQ-UTD1) for a fair comparison to SAC-based methods. All implementations[2] are based on either the Clean RL library (Huang et al., 2022) or the original paper implementation (Rana et al., 2023). We provide a detailed description of the hyperparameter settings in Appendix A.1.

We train all our approaches on ten random seeds and one fixed race track. We report return and cumulative training failures. Runs are considered a failure when the car leaves the track. For zero-shot transfer, we evaluate ten trained models per algorithm on ten unseen racetracks. Return and failure plots show the respective mean (solid lines) and 95 % confidence interval (shaded areas).

[2]The code is available at github.com/Data-Science-in-Mechanical-Engineering/cheq .



(a) (b) (c)

Figure 3: Performance of all trained RL approaches in terms of evaluation return and training failures on the training track. Comparison of (a) CHEQ with fixed-weight hybrid RL and standard RL, (b) increased UTD ratios and (c) prior work in adaptive hybrid RL.

6.2 Evaluation on the Car Racing Environment

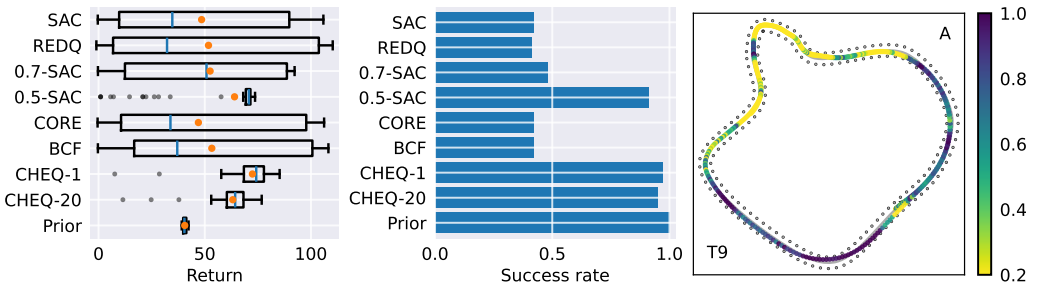
We compare CHEQ to standard RL, fixed-weight hybrid RL, and adaptive hybrid RL concerning learning performance (see Figure 3) and zero-shot transfer to unknown tracks (see Figure 4).

Comparison against Fixed-Weight Hybrid RL and Standard RL: Comparing the CHEQ algorithm based on SAC (CHEQ-UTD1) to a standalone SAC agent and the control prior in Figure 3a illustrates the general benefit of hybrid RL. While the control prior operates safely without failing, it shows limited performance due to the conservative driving policy. SAC shows strong asymptotic performance at the cost of frequent failures throughout training. CHEQ-UTD1 considerably outperforms SAC concerning data efficiency and exploration safety, learning faster with fewer failures while yielding comparable asymptotic performance. The comparison of CHEQ to fixed-weight hybrid RL methods further illustrates the advantage of an adaptive weighting scheme. The fixed-weight hybrid RL approaches (0.5-SAC, 0.7-SAC) combine the control prior with a SAC agent using the mixing function in (3) with $\lambda_t^{\mathrm{RL}} = 0.5$ and $\lambda_t^{\mathrm{RL}} = 0.7$, respectively. Here, the choice of λ_t^{RL} represents a trade-off between exploration safety and asymptotic performance, where a higher λ_t^{RL} enables better performance while reducing safety. A fixed weight of $\lambda_t^{\mathrm{RL}} = 0.5$ arguably reduces failures compared to CHEQ-UTD1, however, this comes at the cost of substantially lower performance.

Update-To-Data Ratio: As discussed in Section 5.3, the critic ensemble of CHEQ allows the use of acceleration techniques originally proposed in the REDQ algorithm. Increasing the UTD ratio to 20 notably improves data efficiency as compared to SAC, both as a standalone RL algorithm (REDQ) and as an adaptive hybrid RL algorithm (CHEQ-UTD20). The speed-up in training helps to reduce training failures as REDQ reports a drastically reduced number of failures compared to SAC. The benefit is further amplified in the adaptive hybrid formulation of CHEQ-UTD20 as indicated by its strong initial performance and the ability to reduce the mean cumulative fails to less than 20.

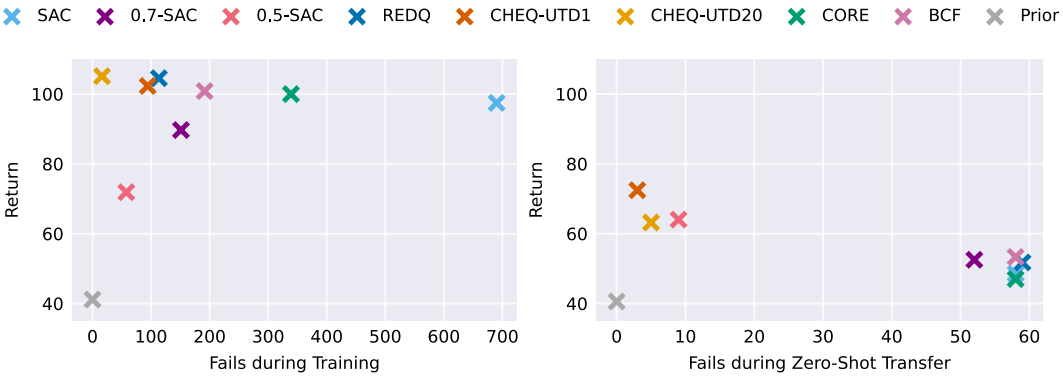
Comparison against state-of-the-art Adaptive Hybrid RL: Finally, Figure 3c compares CHEQ to the most relevant adaptive hybrid RL methods. CHEQ-UTD1 shows similar data efficiency and performance compared to CORE and BCF while considerably reducing accumulated fails. CHEQ-UTD20 substantially outperforms all competitor approaches in all performance metrics. A more detailed hyperparameter analysis of the prior approaches, as well as results for reformulations of CORE and BCF as contextualized hybrid RL methods are provided in in Appendix C.

Zero-shot Transfer: Next, we perform a zero-shot transfer of the trained agents. Returns are depicted in Figure 4a while Figure 4b shows the success rate of the respective methods. CHEQ-UTD1, CHEQ-UTD20 and the control prior achieve a success rate of 97 %, 95 %, and 100 %, respectively. The other standard and hybrid RL methods frequently fail in unseen scenarios. While the CHEQ variants fail slightly more often than the controller, they drive notably faster, i.e., they achieve higher returns. Figure 4c illustrates the adaption mechanism of CHEQ on one example track. In challenging and unseen curves, the agent gradually hands over to the control prior as can be seen



(a) (b) (c)

Figure 4: Comparison of return (a) and failures (b) of ten trained models per algorithm on ten transfer tracks. Development of λ^{RL} of the UTD-20 agent for an exemplary transfer track (c).



(a) (b)

Figure 5: Final return over number of fails during training (a) and zero-shot transfer (b).

in Figure 4c. We find that in the few failure cases (3 out of 100 for CHEQ-UTD1 and 5 out of 100 for CHEQ-UT20), the agent correctly identifies its uncertainty, and hands over to the controller, but the controller is unable to navigate the situation safely. We provide an illustration of all transfer tracks, as well as the weight adaption of CHEQ-UTD20 on these tracks in Appendix C.2. In summary, CHEQ shows strong zero-shot transfer behavior, driving faster than the controller with only a few failures.

Summary: We summarize the trade-off between failures and asymptotic performance in Figure 5. Figure 5a illustrates the training results of the respective approaches while Figure 5b depicts the transfer results. Fixed weight hybrid RL effectively reduces failures as compared to standard SAC. This, however, comes at the cost of asymptotic performance. Our adaptive CHEQ algorithm avoids this trade-off, achieving high return with only a few failures. In zero-shot transfer, the CHEQ agent again performs best due to its ability to detect unforeseen situations reliably and then fall back to the safe control prior.

7 Conclusion

This work addresses how to systematically combine an RL agent with a control prior. We propose a novel formulation of the adaptive hybrid RL problem which introduces the adaptive weighting parameter as a context variable of the MDP, and based on this, propose the Contextualized Hybrid Ensemble Q-learning (CHEQ) algorithm. CHEQ combines a reliable critic uncertainty-based weight adaption mechanism with the data efficiency of critic ensemble methods, yielding substantially stronger results than state-of-the-art adaptive hybrid RL methods on a racing task concerning data efficiency, exploration safety, and transferability.

Acknowledgments

We thank Paul Brunzema, Johanna Menn, and David Stenger for their helpful comments. We also thank Devdutt Subhasish and Lukas Kesper for their help with the cartpole example. This work was funded in part by the German Federal Ministry of Education and Research ("Demonstrations- und Transfernetzwerk KI in der Produktion (ProKI-Netz)" initiative, grant number 02P22A010) and the German Federal Ministry for Economic Affairs and Climate Action (project EEMotion). Computations were performed with computing resources granted by RWTH Aachen University under the projects <thes1594>, <rwth1490>, and <rwth1501>.

References

Marc G. Bellemare, Will Dabney, and Mark Rowland. *Distributional Reinforcement Learning*. MIT Press, 2023. `http://www.distributional-rl.org`.

Dieter Büchler, Simon Guist, Roberto Calandra, Vincent Berenz, Bernhard Schölkopf, and Jan Peters. Learning to Play Table Tennis From Scratch Using Muscular Robots. *IEEE Transactions on Robotics*, 2022.

Federico Ceola, Lorenzo Rosasco, and Lorenzo Natale. RESPRECT: Speeding-up Multi-fingered Grasping with Residual Reinforcement Learning, 2024. arXiv:2401.14858 [cs].

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model, 2021.

Richard Cheng, Abhinav Verma, Gabor Orosz, Swarat Chaudhuri, Yisong Yue, and Joel Burdick. Control Regularization for Reduced Variance Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019.

Paul Daoudi, Bogdan Robu, Christophe Prieur, Ludovic Dos Santos, and Merwan Barlier. Enhancing Reinforcement Learning Agents with Local Guides. In *International Conference on Autonomous Agents and Multiagent Systems*, 2023.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018.

Abhishek Gupta, Justin Yu, Tony Z. Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. Reset-Free Reinforcement Learning via Multi-Task Learning: Learning Dexterous Manipulation Behaviors without Human Intervention. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 2018.

Carl-Johan Hoel, Tommy Tram, and Jonas Sjöberg. Reinforcement Learning with Uncertainty Estimation for Tactical Decision-Making in Intersections. In *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020a.

Carl-Johan Hoel, Krister Wolff, and Leo Laine. Tactical Decision-Making in Autonomous Driving by Reinforcement Learning with Uncertainty Estimation. In *IEEE Intelligent Vehicles Symposium (IV)*, 2020b.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G. M. Araújo. CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. *Journal of Machine Learning Research*, 2022.

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)*, 2019.

Lindsey Kerbel, Beshah Ayalew, Andrej Ivanco, and Keith Loiselle. Residual Policy Learning for Powertrain Control. *IFAC-PapersOnLine*, 2022.

Padmaja Kulkarni, Jens Kober, Robert Babuška, and Cosimo Della Santina. Learning assembly tasks in a few minutes by combining impedance control and residual recurrent reinforcement learning. *Advanced Intelligent Systems*, 2022.

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin Q-learning: Controlling the Estimation Bias of Q-learning, 2021.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2021.

Vincent Mai, Kaustubh Mani, and Liam Paull. Sample efficient deep reinforcement learning via uncertainty estimation. *arXiv preprint arXiv:2201.01666*, 2022.

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

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with Large Scale Deep Reinforcement Learning, 2019.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep Exploration via Bootstrapped DQN. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Krishan Rana, Vibhavari Dasagi, Ben Talbot, Michael Milford, and Niko Sünderhauf. Multiplicative Controller Fusion: Leveraging Algorithmic Priors for Sample-efficient Reinforcement Learning and Safe Sim-To-Real Transfer. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020a.

Krishan Rana, Ben Talbot, Vibhavari Dasagi, Michael Milford, and Niko Sünderhauf. Residual Reactive Navigation: Combining Classical and Learned Navigation Strategies For Deployment in Unknown Environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020b.

Krishan Rana, Vibhavari Dasagi, Jesse Haviland, Ben Talbot, Michael Milford, and Niko Sünderhauf. Bayesian controller fusion: Leveraging control priors in deep reinforcement learning for robotics. *The International Journal of Robotics Research*, 2023.

Alireza Ranjbar, Ngo Anh Vien, Hanna Ziesche, Joschka Boedecker, and Gerhard Neumann. Residual Feedback Learning for Contact-Rich Manipulation Tasks with Uncertainty. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

Maximilian Schier, Christoph Reinders, and Bodo Rosenhahn. Learned Fourier Bases for Deep Set Feature Extractors in Automotive Reinforcement Learning. In *IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, Bilbao, Spain, 2023.

Gerrit Schoettler, Ashvin Nair, Jianlan Luo, Shikhar Bahl, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

J. Schulman, F. Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *ArXiv*, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 2018a.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018b.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Sebastian Thrun and Anton Schwartz. Issues in Using Function Approximation for Reinforcement Learning. In *Proceedings of the Fourth Connectionist Models Summer School*, 1993.

Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 2006.

Hang Wang, Sen Lin, and Junshan Zhang. Adaptive Ensemble Q-learning: Minimizing Estimation Bias via Error Feedback. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

A Algorithmic Details

Algorithm 2 shows the pseudocode of the Contextualized Hybrid Ensemble Q-learning algorithm.

Algorithm 2 CHEQ

Initialize control prior $\pi^{\text{prior}}(\mathbf{a}_t^{\text{prior}} \mid \mathbf{s}_t)$, contextualized RL policy $\hat{\pi}_\phi^{\text{RL}}(\mathbf{a}_t^{\text{RL}} \mid \mathbf{s}_t, \lambda_t^{\text{RL}})$, contextualized critic ensemble $\hat{Q}_{\theta_e}(\mathbf{s}_t, \mathbf{a}_t^{\text{RL}}, \lambda_t^{\text{RL}})$, $e \in \{1, \ldots, E\}$, contextualized target critic ensemble $\hat{Q}_{\bar{\theta}_e}(\mathbf{s}_t, \mathbf{a}_t^{\text{RL}}, \lambda_t^{\text{RL}})$, $e \in \{1, \ldots, E\}$, replay buffer $\mathcal{D} \leftarrow \emptyset$, weighting interval $[\lambda_{\min}, \lambda_{\max}]$, uncertainty limits $[u_{\min}, u_{\max}]$, UTD ratio G, Bernoulli masking rate κ, minimization targets F, Polyak averaging factor τ

for each epoch **do**
  $\mathbf{s}_0 \sim \rho_0$, $\lambda_0^{\text{RL}} = \lambda_{\min}$
  for each epoch step **do**
    $\mathbf{a}_t^{\text{RL}} \sim \hat{\pi}_\phi^{\text{RL}}(\cdot \mid \mathbf{s}_t, \lambda_t^{\text{RL}})$
    $\mathbf{a}_t^{\text{prior}} \sim \pi^{\text{prior}}(\cdot \mid \mathbf{s}_t)$
    $\mathbf{a}_t^{\text{mix}} = (1 - \lambda_t^{\text{RL}})\mathbf{a}_t^{\text{prior}} + \lambda_t^{\text{RL}}\mathbf{a}_t^{\text{RL}}$
    $u(\mathbf{s}_t, \mathbf{a}_t^{\text{RL}}, \lambda_t^{\text{RL}})$ according to (5)
    $\lambda_{t+1} = \Lambda(u(\mathbf{s}_t, \mathbf{a}_t^{\text{RL}}, \lambda_t^{\text{RL}}))$ according to (6)
    $\mathbf{s}_{t+1}, r_{t+1} \sim \hat{p}(\cdot, \cdot \mid \mathbf{s}_t, \mathbf{a}_t^{\text{RL}}, \lambda_t^{\text{RL}})$
    for $e = 1, \ldots, E$ **do**
      Sample Bernoulli Mask $m_t^e \sim Ber(\kappa)$
    end for
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t^{\text{RL}}, \lambda_t^{\text{RL}}, \mathbf{s}_{t+1}, r_{t+1}), m_t^1, \ldots, m_t^E\}$
    for G updates **do**
      Sample mini-batch $\mathcal{B} = \{(\mathbf{s}, \mathbf{a}^{\text{RL}}, \lambda^{\text{RL}}, \mathbf{s}', r\}$ from $\mathcal{D}$
      Sample a set $\mathcal{F}$ with $|\mathcal{F}| = F$ uniform at random from $\{1, \ldots, E\}$
      $\tilde{\mathbf{a}}'^{\text{RL}} \sim \hat{\pi}_\phi^{\text{RL}}(\cdot \mid \mathbf{s}', \lambda^{\text{RL}})$
      $y = r + \gamma \left(\min_{e \in \mathcal{F}} \hat{Q}_{\bar{\theta}_e}(\mathbf{s}', \tilde{\mathbf{a}}'^{\text{RL}}, \lambda^{\text{RL}}) - \alpha \log \hat{\pi}_\phi^{\text{RL}}(\tilde{\mathbf{a}}'^{\text{RL}} \mid \mathbf{s}', \lambda^{\text{RL}}) \right)$
      for $e = 1, \ldots, E$ **do**
        Update θ_e with gradient descent using
        $1_{m^e} \nabla_{\theta_e} \frac{1}{|\mathcal{B}|} \sum_{(\mathbf{s}, \mathbf{a}^{\text{RL}}, \lambda^{\text{RL}}, r, \mathbf{s}') \in \mathcal{B}} \left(\hat{Q}_{\theta_e}(\mathbf{s}, \mathbf{a}, \lambda^{\text{RL}}) - y \right)^2$
        $\bar{\theta}_e \leftarrow \tau \bar{\theta}_e + (1 - \tau)\theta_e$
      end for
    end for
    update ϕ with gradient ascent using
    $\tilde{\mathbf{a}}_{\text{RL}} \sim \hat{\pi}_\phi^{\text{RL}}(\cdot \mid \mathbf{s}, \lambda^{\text{RL}}))$
    $\nabla_\phi \frac{1}{|\mathcal{B}|} \sum_{\mathbf{s} \in \mathcal{B}} \left(\frac{1}{E} \sum_{e=1}^{E} \hat{Q}_{\theta_e}(\mathbf{s}, \tilde{\mathbf{a}}_{\text{RL}}, \lambda^{\text{RL}}) - \alpha \log \hat{\pi}_\phi^{\text{RL}}(\tilde{\mathbf{a}}_{\text{RL}} \mid \mathbf{s}, \lambda^{\text{RL}})) \right)$
  end for
end for

A.1 Hyperparameters Settings

We build our SAC implementation based on CleanRL (Huang et al., 2022). We keep the SAC-specific hyperparameters consistent between all approaches and report them in Table 1.

CHEQ (UTD1 and UTD20) uses an ensemble of $E = 5$ critics. For CHEQ (UTD1 and UTD20) we set the upper bound of the uncertainty as $u_{\max} = 0.15$ and the lower bound as $u_{\min} = 0.03$. Further we set $\lambda_{\max} = 1.0$ and $\lambda_{min} = 0.2$. We use a Bernoulli masking rate of $\kappa = 0.8$ and $F = 2$ minimization targets.

BCF trains an ensemble of policy networks. We maintain the original ensemble size from (Rana et al., 2023) which uses ten policy networks. We set the standard deviation of the control prior in BCF as $\sigma^{\text{prior}} = 6.0$.

Table 1: Shared Hyperparameters.

Hyperparameter	Value
number of steps	1.5×10^6
batch size	256
learning rate actor	3×10^{-4}
learning rate critic	3×10^{-4}
target entropy H_t	-3
replay buffer size	1×10^6
discount factor γ	0.99
gradient update start	1×10^3 steps
Polyak averaging factor τ	0.005

For the uncertainty estimate in CORE we set $A = 7$, $C = 0.02$. Note that in the original paper A is denoted as $\lambda_{\max}$, which we change to avoid ambiguous notation. SAC uses a UTD ratio of 1. REDQ implementation both use an ensemble size of 5 and a UTD ratio of 20.

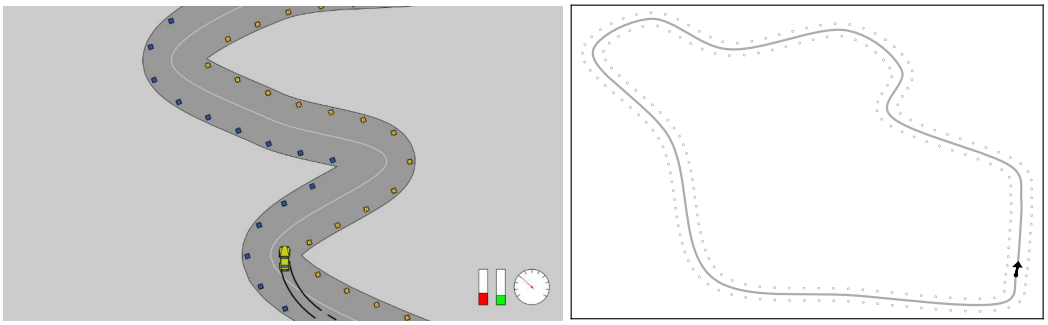
For all algorithms, we include a random sampling phase for the first 1×10^3 steps where we sample the RL action uniformly random and do not update our agent. In this setting, we keep λ_t^{RL} small for the hybrid agents. For CHEQ we vary λ^{RL} between [0.2, 0.3]. As CORE and BCF are unable to observe changes in λ^{RL} we use a fixed $\lambda^{\mathrm{RL}} = 0.2$ which has shown to favourable in our experiments. After the random sampling phase, agent training starts, but λ^{RL} is kept small for another 4×10^3 steps and afterward λ adaption starts.

For performance evaluation, we conduct a greedy evaluation run every 20 episodes. Evaluation happens in an adapted setting, together with the controller where the weight is calculated as in the training procedure.

B Environment Details

B.1 Racing Environment

We test our agent on the simulated racing task adapted from Schier et al. (2023). Figure 6 shows an example of the environment.



(a) (b)

Figure 6: Racing task and the training track.

The vehicle uses a dynamic single-track model with a coupled Dugoff tire model. The throttle, brake, and steering are continuous actions. The vehicle is a front-wheel drive. The RL agent may learn to control brake balance by applying throttle and brake individually. We define the state of the RL agent as $\mathbf{s}_t = (v_x, v_y, \omega, \beta, o_{\mathrm{track}})$, with the ego vehicle's velocity vector $\mathbf{v}_{\mathrm{ego}} = (v_x, v_y)$ in vehicle reference frame, steering angle β, and yaw rate ω. The observation of the track $o_{\mathrm{track}} = (\mathbf{x}, \mathbf{y})^T$ is given as a vector of 20 cartesian distances (x_i, y_i) to the centerline of the track. The points are sampled equidistantly from the 60m track segment ahead.

We use the original reward formulation from Schier et al. (2023) where the RL agent receives a penalty $r_{\text{collision}}$ whenever it collides with the track boundary and a penalty r_{fail} for leaving the track with the center of mass. The latter case also terminates the episode. The RL agent receives a positive reward for driving fast: the scalar projection of its velocity vector v_{ego} onto the forward track direction n_{track}. The complete reward is then given by

$$r(s, a) = -r_{\text{fail}} - 0.2 \cdot r_{\text{collision}} + 0.01 \cdot \mathbf{n}_{\text{track}} \cdot \mathbf{v}_{\text{ego}}. \tag{7}$$

B.2 Control Prior

For the racing task, we design a simple path-following controller with adaptive speeds. For the lateral control, we use a Stanley Controller (Thrun et al., 2006) following the steering control law

$$\delta(t) = \psi(t) + \frac{k_{\text{cross}} \cdot e(t)}{v(t) + k_{\text{soft}}},$$

where $\psi(t)$ denotes the heading error, $e(t)$ denotes the crosstrack error of the front axle and $v(t)$ describes the velocity of the vehicle.

For the longitudinal control, we design two symmetric P-controllers; one for the brake and one for the throttle. First, we compute the target velocity dependent on the curve radius $R(t)$ of the track directly in front of the vehicle as

$$v_{\text{target}}(t) = \min\{k_r \cdot R(t), v_{\max}\},$$

where $v_{\max}$ is the maximum desired velocity. Then, we design the throttle control as

$$\text{throt}(t) = \begin{cases} k_v(t) * (v_{target}(t) - v(t)), & v_{target}(t) - v(t) \geq 0 \\ 0, & \text{else} \end{cases}$$

and the brake control as

$$\text{br}(t) = \begin{cases} k_v(t) * (v(t) - v_{target}(t)), & v_{target}(t) - v(t) \leq 0 \\ 0, & \text{else,} \end{cases}$$

with shared gain k_v. Following this control law, the control prior accelerates if it is going too slow and brakes if it is going too fast. It never uses the brake and the throttle at the same time.

To avoid aggressive braking behavior when the RL agent hands over to the controller in risky situations (high velocity around curves), we additionally introduce a simple clipped linear gain schedule on k_v attenuating the braking control for higher velocities as

$$k_v(t) = \text{clip}\left(\frac{k_v^{\max} - k_v^{\min}}{v_{\text{low}} - v_{\text{high}}}(v(t) - v_{\text{low}}) + k_v^{\max}; k_v^{\max}, k_v^{\min}\right).$$

We tuned the controller gains and coefficients to $k_{\text{cross}} = 0.5[1/\text{s}]$, $k_{\text{soft}} = 1[\text{m}\,\text{s}^{-1}]$, $k_r = 0.4[1/\text{s}]$, $v_{\max} = 8[\text{m}\,\text{s}^{-1}]$, $k_v^{\max} = 0.25[\text{s}\,\text{m}^{-1}]$, $k_v^{\min} = 0.05[\text{s}\,\text{m}^{-1}]$, $v_{\text{low}} = 8[\text{m}\,\text{s}^{-1}]$ and $v_{\text{high}} = 28[\text{m}\,\text{s}^{-1}]$.

C Additional Results and Ablation Study

In this section, we formulate contextualized hybrid variants of CORE and BCF, C-CORE and C-BCF. Further, we present additional results on hyperparameter sensitivity and the distribution of λ^{RL} for CHEQ-UTD1, CORE, and BCF. For a reasonable comparison, we focus only on CHEQ-UTD1 using a UTD ratio of 1.

C.1 Contextualized Hybrid Variants of Prior Work

To further substantiate our claim that the contextualized hybrid RL formulation aids the training progress, we developed contextualized variants of the CORE and BCF algorithm, which we call C-CORE and C-BCF. In order to use the adaptive weight as a context variable, a weighting parameter λ_t^{RL} needs to be determined.

The CORE algorithm comes with a direct weight estimate(Cheng et al., 2019), which can be written as

$$\lambda_t^{\mathrm{RL}} = \frac{1}{1 + \lambda_t^{\mathrm{CORE}}}$$

where $\lambda_t^{\mathrm{CORE}} = A(1 - e^{-C\delta_{t-1}})$ with the TD-error δ_{t-1} and C, A being tuning parameters[3].

For C-BCF, we derive a pseudo-weight, as the BCF algorithm (Rana et al., 2023) does not have a straightforward weighting factor λ^{RL}. In BCF, at timestep $t-1$, the fusion of the prior $\mathcal{N}_{\psi,t-1}(\mu_{\psi,t-1}, \sigma_{\psi,t-1})$ and the SAC-ensemble $\mathcal{N}_{\pi,t-1}(\mu_{\pi,t-1}, \sigma_{\pi,t-1})$ results in a Gaussian distribution with mean

$$\mu_{\mathrm{fuse},t-1} = \frac{\sigma^2_{\psi,t-1}}{\sigma^2_{\psi,t-1} + \sigma^2_{\pi,t-1}} \cdot \mu_{\pi,t-1} + \frac{\sigma^2_{\pi,t-1}}{\sigma^2_{\psi,t-1} + \sigma^2_{\pi,t-1}} \cdot \mu_{\psi,t-1}.$$

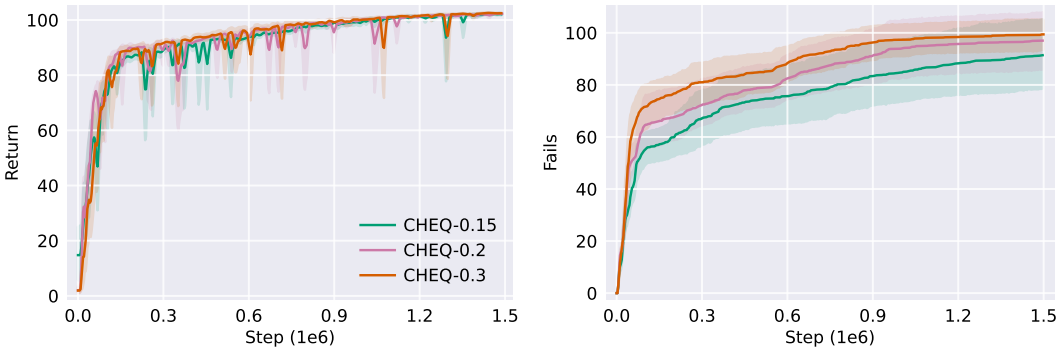
Thus, in BCF the weight has the dimension of the action space, whereas the contextualized mechanism requires a scalar weight λ_t^{RL}. For C-BCF, we compute a scalar weight

$$\lambda_t^{\mathrm{RL}} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\sigma^2_{\psi,t-1}}{\sigma^2_{\psi,t-1} + \sigma^2_{\pi,t-1}} \right)_i.$$

The next action $\mathbf{a}_t^{\mathrm{mix}}$ is computed according to Equation 3 where $\mathbf{a}_t^{\mathrm{RL}} \sim \mathcal{N}_{\pi,t}$ and $\mathbf{a}_t^{\mathrm{prior}} = \mu_{\psi,t}$.

For C-BCF and C-CORE we use the same warm-up phase as for our CHEQ agent.

C.2 Additional Results on CHEQ



(a) (b)

Figure 7: Comparison of CHEQ-UTD1 with different $u_{\max}$ thresholds. Plotting return (a) and number of fails (b) for the racing environment.

Our algorithm has only two important hyperparameters, upper and lower bounds of the uncertainty $u_{\max}$ and $u_{\min}$. We chose these hyperparameters by conducting one training run and investigating the uncertainty range within this run. CHEQ is generally robust against changes in these thresholds.

[3]CORE (Cheng et al., 2019) uses the term $\lambda_{\max}$ instead of A. As we use $\lambda_{\max}$ in a different context, we stick to A here.

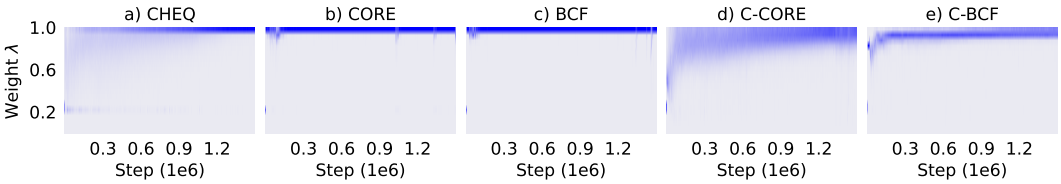


Figure 8: Distributions of λ^{RL} over training steps shown for all hybrid agents.

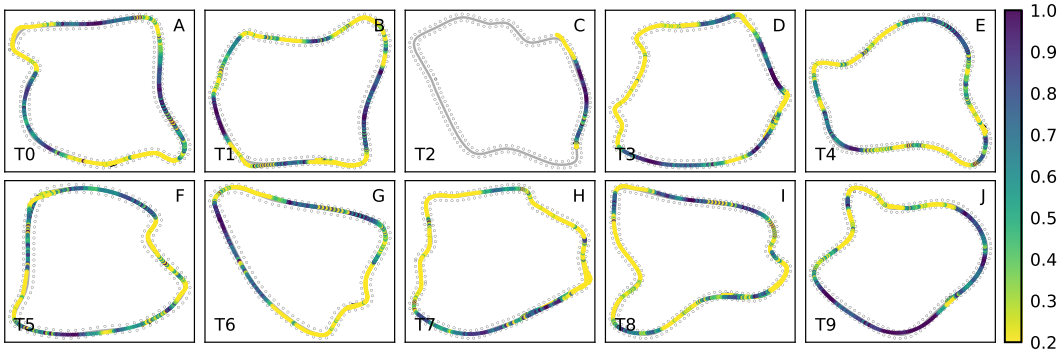


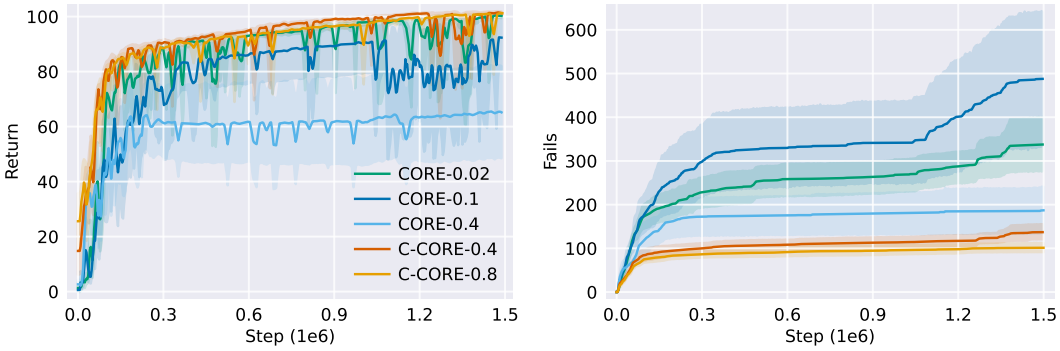
Figure 9: The 10 transfer tracks and the corresponding zero-shot transfer of one seed of CHEQ-UTD20.

We observe slightly lower final return and fewer fails for lower upper bounds $u_{\max}$. This is to be expected as frequent hand-over to the control prior results in lower velocities and thus lower return. Figure 7 shows the return and the number of fails during training for our CHEQ-UTD1 variants. For CHEQ-UTD20 we were able to use the same upper and lower uncertainty bounds as for CHEQ-UTD1.

Figure 8 shows the distribution of the weight λ^{RL} over the training progress. We find that for CHEQ-UTD1 the agent starts with an almost uniform distribution of the weight and slowly moves towards a $\lambda^{\mathrm{RL}} = 1$ regime. Even in later training stages the agent hands over to the controller from time to time.

Lastly, we investigated the transfer behavior of the CHEQ-UTD20 agent further. Figure 9 shows the ten transfer tracks. We plot λ^{RL} over the track for one evaluated model. Here, we find that the agent frequently becomes uncertain and hands over to the controller, especially in unknown curves. In plot C we see one of the two failure cases (out of 100 runs) that we experience during transfer. We find that the agent becomes uncertain and hands over to the controller. In this specific scenario however, the controller is not able to safely navigate the situation and leaves the track.

C.3 Additional Results on CORE and C-CORE



(a) (b)

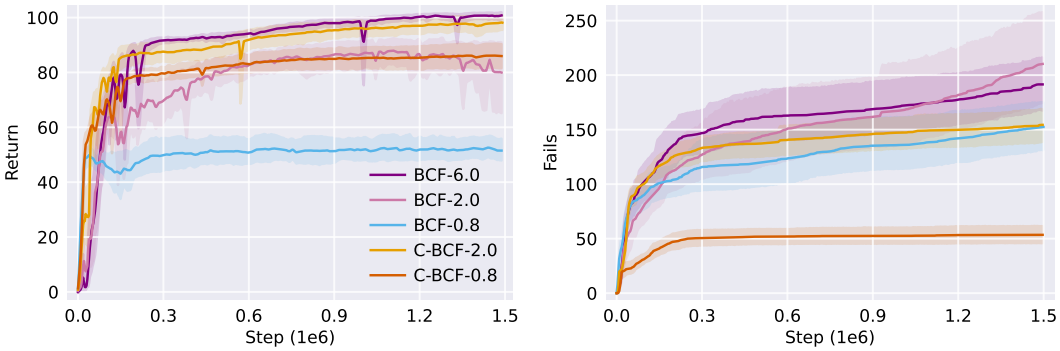
Figure 10: Comparison of return (a) and number of fails (b) CORE and C-CORE runs with different C parameters.

In Figure 10 we compared different parameters C. We find stable training progress and high return for $C = 0.02$ but since this uses high λ^{RL} values, this setting results in many failures. Figure 8 illustrates the high λ regime of the $C = 0.02$ agent. C-CORE using our contextualized hybrid framework, notably outperforms CORE in terms of asymptotic return, training stability and the number of training failures. Using the contextualized formulation, C-CORE is able to use a much wider λ^{RL} distribution (see Figure 8).

C.4 Additional Results on BCF and C-BCF

The BCF algorithm is sensitive to the parameters of the uncertainty threshold, which in this case is the variance of the control prior σ^{prior}. Higher variances, lead to less weight on the control prior and thus high λ^{RL} regimes. In Figure 11 we compared different parameters σ^{prior}. We find stable training progress and high return for $\sigma^{\text{prior}} = 6$. However, this setting, as expected, uses a λ^{RL} regime close to one and thus results in a high number of failures. Figure 8 illustrates this regime.

Our C-BCF variant is able to resolve this problem only partially. Due to its construction, the BCF algorithm has a separate weighting factor for each action of which we take the mean. In addition, our pseudo λ^{RL} factor is only a rough estimate of the actual mixing as BCF samples from the posterior distribution. Both factors result in information loss and make the weight λ^{RL} only a rough estimate. We find that C-BCF-2.0 and BCF-6.0 achieve similar asymptotic performance, while C-BCF-2.0 leads to fewer failures.



(a) (b)

Figure 11: Comparison of return (a) and number of fails (b) of BCF and C-BCF runs with different σ_{prior} parameters.