ARBITRATED HANDOVER FROM BASELINE CONTROL TO REINFORCEMENT LEARNER

Anonymous authors
Paper under double-blind review

ABSTRACT

A running system cannot pause for a new controller to learn. Online controller adaptation avoids the pause by keeping a fixed baseline in control while a reinforcement learning policy, the learner, trains on the live system. Gating blends the two actions, passing control gradually from baseline to learner over training. The gate must judge when the learner is skilled enough to take over; handing over too early costs performance and can cause failures the baseline would have avoided. We identify two core challenges in this setting. The first is that improving the learner and judging the handover need different value estimates: one of the learner acting alone, one of the blend that actually runs. The second is that, because the learner does not act on its own until the end of training, its value estimate carries extrapolation error. We find that neglecting either results in premature handover, causing performance degradation and task failure during training. We propose arbitrator-gated handover: an arbitrator judges the executed blend with value estimates of its own, and we apply techniques to limit the effects of value estimation error in this setting.

1 INTRODUCTION

Controllers are designed under assumptions about the system they act on, and their reliability degrades when operating conditions depart from those assumptions. These departures can be present immediately at deployment, most commonly when the model parameters used for design differ from those of the physical system, a discrepancy commonly referred to as the sim-to-real gap (Jakobi et al., 1995). Departures can also arise while the controller is in operation, whether from equipment degradation, disturbances, or shifting operating requirements. Deviations of this kind are difficult to eliminate or to account for at design time of the controller: system identification carries error, and the system itself changes over its lifetime (Simpkins, 2012). Remedies like re-identifying the system and redesigning or retuning the controller require engineering effort and costly downtime.

This makes online controller adaptation attractive. The preexisting controller, which we call the baseline, keeps the system running while a learned controller is trained in the loop, so control adjustment and improvement can happen without interrupting operation. The baseline also serves as a foundation for learning rather than something to be discarded. This constrains early exploration to a region the system tolerates and supplies informative trajectories from the outset, which improves the sample efficiency of learning and lowers constraint violations, relative to starting from scratch (Silver et al., 2018; Johannink et al., 2019; Cheng et al., 2019; Rana et al., 2023; Cramer et al., 2025; Tian et al., 2024).

One such strategy for online controller adaptation is a handover, where responsibility for control is gradually shifted from the baseline to a learned controller over the course of training. A gating mechanism controls this shift, by, for example, blending the action proposals, either on a preset schedule (Ross et al., 2011) or with a blend adapted by RL (Cheng et al., 2019; Rana et al., 2023; Cramer et al., 2025; Tian et al., 2024; Silver et al., 2018). The gating can also be performed by adjusting how often the learner acts alone, as in jump start reinforcement learning where the learner's window of control extends over training (Uchendu et al., 2023).

When a controller is adapted online, the learner should gain authority only where it is at least as competent as the baseline. Existing gating schemes use quantities produced directly from learner

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

training: the magnitude of the temporal-difference error of its critic (Cheng et al., 2019), the disagreement of an ensemble of its policies (Rana et al., 2023) or of its critics (Cramer et al., 2025), or, in RL-AR (Tian et al., 2024), on which we build, the value estimate of its critic. None decides the handover by the return of the blended controller that is executed.

Two challenges of critic-gated handover, which we identify here, argue that it should. First, the learner and the gate need different value estimates. Improving the learner requires the value of the learner acting alone, while judging the handover requires the value of the blend that will run. We show that gating on the critic of the learner can reverse the sign of the gate update, even with an exact critic. Second, the critic of the learner values actions that were never executed. The learner acts alone only once handover is complete; until then the executed actions are the blend, and the value of the learner acting alone is an extrapolation. This error is well known from offline RL, where the dataset is fixed and nothing corrects it, and from offline-to-online RL, where the switch to online data exposes and then corrects it (Fujimoto et al., 2019; Kumar et al., 2019; Levine et al., 2020; Nair et al., 2020). Online adaptation has no switch. Executed actions move toward those of the learner only as the gate opens, so the estimate is corrected only gradually, and only through the handover it is used to decide.

We then propose an arbitrator-gated handover, in which the blend is governed by a critic of its own, the arbitrator, that evaluates the blended controller. The arbitrator is formulated as an ensemble of arbitrators aggregated by a lower confidence bound, so that control passes to the learner where the value of the blend rises and the arbitrator is confident that it does. We mitigate the estimation error of both critics using layer normalization and a prior on the actor toward the baseline.

Our contributions are:

- Identification of two core challenges of critic-gated handover: defining appropriate value estimates for the learner and the gate, and the estimation error this setting introduces.
- Proposal of arbitrator-gated handover, addressing both failure modes by gating with a critic the learner never trains against, and applying mechanisms to address value estimation errors.
- Experiments on four environments demonstrating substantially fewer training failures and higher final return than the strongest baseline.

2 Problem Statement

We consider the following setting for online control adaptation. A system with state space $\mathcal{S}$ and continuous action space $\mathcal{A} \subset \mathbb{R}^k$ is controlled by a fixed baseline controller $\pi_{\text{base}} : \mathcal{S} \to \mathcal{A}$ with acceptable but suboptimal performance. The system continues to run while a learned controller π_θ is being trained. The learner assumes control as it becomes competent.

Handover by action blending. Both controllers propose an action at every state, and the executed action is their per-dimension convex combination. For a weight $w \in [0,1]^k$, the blended action is

$$a_w(s) = w \odot a_{\text{base}} + (\mathbf{1} - w) \odot a_{\text{rl}}, \qquad a_{\text{base}} = \pi_{\text{base}}(s), \quad a_{\text{rl}} \sim \pi_\theta(\cdot \mid s), \tag{1}$$

where $\odot$ is the element-wise product and $\mathbf{1}$ is the all-ones vector. The weight used at each state is chosen by a function $\beta : \mathcal{S} \to [0,1]^k$, which we call the gate, and π_β denotes the policy that executes $a_{\beta(s)}(s)$ at every state. When the state is clear from context, we write β for $\beta(s)$, e.g. $a_\beta(s)$ for $a_{\beta(s)}(s)$. Control starts with the baseline, $\beta \equiv \mathbf{1}$, and lowering β gives the learner more control authority.

The gating problem. The task is to choose β. Lowering β at a state s is warranted only if acting there with the lower weight, and following π_β afterwards, yields a higher expected return than keeping the current weight. This decision must be made online, from transitions generated by π_β itself, while π_θ and β are both being updated.

3 PRELIMINARIES

3.1 OFF-POLICY ACTOR-CRITIC AND SAC

We consider an infinite-horizon Markov decision process $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ with continuous action space $\mathcal{A}$, transition kernel P, reward function r, and discount factor $\gamma \in (0, 1)$. A policy π maps states to distributions over actions. Its action-value function and its discounted entropy are

$$Q^\pi(s,a) = \mathbb{E}\Big[\sum_{t\geq 0}\gamma^t r(s_t,a_t)\Big], \qquad H^\pi(s) = \mathbb{E}\Big[-\sum_{t\geq 0}\gamma^t \log\pi(a_t \mid s_t)\Big], \tag{2}$$

with $s_0 = s$, with $a_0 = a$ in Q^π, and with all other actions drawn from π. Its value function is $V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a)]$. We call the policy acting after the first step the continuation policy; for Q^π, it is π.

Off-policy actor-critic methods maintain a parametric policy π_θ (the actor) and a critic Q_ϕ, trained from a replay buffer $\mathcal{D}$ of transitions (s, a, r, s', d) collected by a behavior policy μ, where d indicates termination, after which rewards and entropy terms are zero. Soft Actor-Critic (SAC) (Haarnoja et al., 2018b;a), the learner in this work, instantiates this scheme with a tanh-squashed Gaussian policy, entropy regularization with temperature $\alpha \geq 0$, and two critics Q_{ϕ_1}, Q_{ϕ_2} with slowly updated target copies $\bar\phi_1, \bar\phi_2$; taking the minimum of the two critics curbs overestimation (Fujimoto et al., 2018). The actor maximizes

$$J_\pi(\theta) = \mathbb{E}_{s\sim\mathcal{D}}\,\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\Big[\min_{i=1,2} Q_{\phi_i}(s,a) - \alpha\log\pi_\theta(a \mid s)\Big] \tag{3}$$

via the reparameterization trick, and the critics regress onto

$$y = r + \gamma(1-d)\left(\min_{i=1,2} Q_{\bar\phi_i}(s',\tilde a') - \alpha\log\pi_\theta(\tilde a' \mid s')\right), \qquad \tilde a' \sim \pi_\theta(\cdot \mid s'), \tag{4}$$

for $(s, a, r, s', d) \sim \mathcal{D}$. The bootstrap action $\tilde a'$ is sampled from the current actor rather than read from the data, so the continuation policy of the estimate is π_θ rather than μ. Because the distribution of (r, s') given (s, a) depends only on the environment, the critics estimate the same quantity whichever policy collected the data: the soft value of the learner, its return plus α times its entropy from the next state onward,

$$Q_{\phi_i}(s,a) \;\approx\; Q^{\pi_\theta}(s,a) + \alpha\gamma\,\mathbb{E}_{s'\sim P(\cdot|s,a)}\big[H^{\pi_\theta}(s')\big]. \tag{5}$$

3.2 CRITIC ENSEMBLES AND PESSIMISM

The disagreement of independently initialized critics estimates their error (Osband et al., 2016; Lakshminarayanan et al., 2017) and grows with the distance of (s, a) from the data (An et al., 2021). Shared data and targets erode the disagreement, so each member of an ensemble $Q_{\phi_1}, \ldots, Q_{\phi_N}$ can be trained on its own bootstrap sample of the data (Osband et al., 2016) and its own target copy (Ghasemipour et al., 2022), with a fixed random prior network added so that the members stay apart where the data do not constrain them (Osband et al., 2018). Pessimistic offline reinforcement learning lowers the value estimate by its uncertainty (Jin et al., 2021), which ensemble methods measure by the standard deviation of the members (An et al., 2021; Bai et al., 2022; Ghasemipour et al., 2022), giving the lower confidence bound

$$Q^{\mathrm{lcb}}_\phi(s,a) = \mu_\phi(s,a) - \kappa\,\sigma_\phi(s,a), \qquad \mu_\phi = \frac{1}{N}\sum_{j=1}^{N} Q_{\phi_j}, \qquad \sigma^2_\phi = \frac{1}{N}\sum_{j=1}^{N}\big(Q_{\phi_j} - \mu_\phi\big)^2, \tag{6}$$

with $\kappa \geq 0$. The minimum over two critics in equation 4 is equation 6 with $N = 2$ and $\kappa = 1$, and the minimum over N members is approximately equation 6 with κ fixed by N (An et al., 2021); equation 6 sets κ independently of the ensemble size.

3.3 RL-AR: CONTROL HANDOVER BY ACTION BLENDING

RL-AR (Tian et al., 2024) implements the gate β of Section 2 as a network with parameters ψ, pretrained to $\beta \approx \mathbf{1}$ everywhere. Transitions are stored as $(s, a, a_{\mathrm{base}}, r, s', d)$, with a the executed action, and the learner is trained on them with SAC (Section 3.1).

The handover objective. RL-AR makes the criterion of Section 2 precise: at each state, the gate is moved toward the weight that maximizes the value of the blended action, with the current blend π_β as the continuation policy and no entropy term,

$$\beta'(s) = \arg\max_{w \in [0,1]^k} F_s(w), \qquad F_s(w) = \mathbb{E}_{a_{\mathrm{rl}} \sim \pi_\theta(\cdot|s)} \Big[Q^{\pi_\beta}\big(s, a_w(s)\big) \Big]. \tag{7}$$

Since this is greedy improvement on π_β with respect to its own value, Tian et al. (2024, Theorem 2) gives $V^{\pi_{\beta'}} \geq V^{\pi_\beta}$ at every state in the tabular setting, where $\pi_{\beta'}$ executes the new gate. We call equation 7 the handover objective.

Critic reuse. RL-AR trains no estimator of Q^{π_β}. Instead, it plugs the learner's critic, already trained and differentiable in the action, into F_s,

$$\hat{F}_s(w) = \mathbb{E}_{a_{\mathrm{rl}} \sim \pi_\theta(\cdot|s)} \Big[\min_{i=1,2} Q_{\phi_i}\big(s, a_w(s)\big) \Big], \tag{8}$$

and replaces the maximization with a gradient step on the gate's parameters,

$$\psi \;\leftarrow\; \psi + \eta_\beta \, \nabla_\psi \, \mathbb{E}_{(s,\, a_{\mathrm{base}}) \sim \mathcal{D}} \big[\hat{F}_s(\beta) \big], \tag{9}$$

with a_{base} read from the data, a_{rl} resampled from the current actor, and the gradient taken through β only. We call this substitution critic reuse. Section 4 characterizes its effect.

4 Failure Modes of a Shared Critic

4.1 Evaluation Under the Wrong Continuation Policy

Ceding control to the learner is justified by the guarantee behind the handover objective equation 7: every gate update improves on the policy in control. Critic reuse measures the update against the learner instead, and can point the gate the wrong way.

Take the critic to be exact, so that equation 5 holds with equality. The substitute objective equation 8 then has the form of the handover objective equation 7, with the learner's soft value in place of the blend's value. Both share the reward and the kernel, so the two objectives differ only in what they take the next state to be worth,

$$\hat{F}_s(w) - F_s(w) = \gamma \, \mathbb{E}\big[V^{\pi_\theta}(s') - V^{\pi_\beta}(s') + \alpha H^{\pi_\theta}(s') \big], \tag{10}$$

with the expectation over $a_{\mathrm{rl}} \sim \pi_\theta(\cdot \mid s)$ and $s' \sim P(\cdot \mid s, a_w(s))$. The value difference credits an action with what the learner would earn afterward rather than what the blend would. For $\alpha > 0$ the last term also credits it with the entropy of the learner from s' onward. Write $\hat{\beta}(s) = \arg\max_{w \in [0,1]^k} \hat{F}_s(w)$, as in equation 7, and $\hat{\pi}$ for the blend that executes it.

By the policy improvement theorem (Howard, 1960; Puterman, 1994; Sutton & Barto, 1998), a policy π' with $\mathbb{E}_{a \sim \pi'(\cdot|s)}[Q^\pi(s,a)] \geq V^\pi(s)$ at every state satisfies $V^{\pi'} \geq V^\pi$, and the policy improved on is π, the one whose Q scored the actions. The maximum of each objective is at least the value of its own continuation policy, since $\beta(s)$ reproduces the blend under F_s and, at $\alpha = 0$, the weight $\mathbf{0}$ reproduces the learner under $\hat{F}_s$:

$$F_s\big(\beta'(s)\big) \;\geq\; F_s\big(\beta(s)\big) = V^{\pi_\beta}(s), \qquad\qquad \hat{F}_s\big(\hat{\beta}(s)\big) \;\geq\; \hat{F}_s(\mathbf{0}) = V^{\pi_\theta}(s).$$

Critic reuse therefore certifies its update only as an improvement on the learner. Nothing certifies it as an improvement on the policy in control.

A lost guarantee is not yet a failure. The next result shows that the gap equation 10 can flip the sign of the gate update even with an exact critic. It can do so while the learner is still worse than the baseline downstream, which is when a gate is needed.

Proposition 1 (Critic reuse can misdirect the gate)**.** *Let the critic be exact, $\alpha = 0$, and $\gamma \in (0,1)$. There exist an MDP with state space $\mathcal{S} = \{s_{\mathrm{risk}}, s_{\mathrm{ok}}, s_{\mathrm{fail}}\}$ and action space $\mathcal{A} = [0,1]$, and a baseline with $V^{\pi_{\mathrm{base}}}(s_{\mathrm{ok}}) > 1/\gamma$, such that the following holds for every gate with $\beta(s_{\mathrm{risk}}) > 0$ and $\beta(s_{\mathrm{ok}}) = 1$ and every learner with*

$$\bar{a} := \mathbb{E}_{a \sim \pi_\theta(\cdot|s_{\mathrm{risk}})}[a] > 0, \qquad V^{\pi_\theta}(s_{\mathrm{ok}}) < \frac{1}{\gamma}.$$

At s_{ok} the substitute keeps the baseline, $\hat{\beta}(s_{\mathrm{ok}}) = 1$. At s_{risk} the substitute and the true handover objective have opposing slopes for every $w \in [0, 1]$,

$$\partial_w \hat{F}_{s_{\mathrm{risk}}}(w) = -\bar{a}\left(1 - \gamma V^{\pi_\theta}(s_{\mathrm{ok}})\right) < 0,$$
$$\partial_w F_{s_{\mathrm{risk}}}(w) = -\bar{a}\left(1 - \gamma V^{\pi_{\mathrm{base}}}(s_{\mathrm{ok}})\right) > 0,$$

so the substitute is maximized only by full handover, $\hat{\beta}(s_{\mathrm{risk}}) = 0$, although the resulting blend is worse than the one in control, $V^{\hat{\pi}}(s_{\mathrm{risk}}) < V^{\pi_\beta}(s_{\mathrm{risk}})$.

Proof. The result is proven in Appendix B □

4.2 Critic Overestimation in Online Controller Adaptation

The gate steers by the value estimate of the critic. The extent to which we can trust these estimates changes over training. Until handover is complete, no executed action is from the learner alone, so the critic evaluates π_θ on transitions another controller generated. A critic can only be accurate on a policy whose actions the data contains (Fujimoto et al., 2019). The $(1 - \beta)$-weighted share of the actions of π_θ grows over training, so the accuracy of the critic value estimates likely also does.

At $\beta \equiv 1$ the critic is essentially learning offline, where this extrapolation is the central difficulty, known as extrapolation error (Fujimoto et al., 2019), bootstrapping error (Kumar et al., 2019) or distributional shift (Levine et al., 2020). Maximization turns the error into overestimation (Thrun & Schwartz, 1993; Fujimoto et al., 2018), and since the actor is trained by gradient ascent on the critic equation 3, it moves toward the unsupported actions whose values are inflated (Kumar et al., 2019; Fujimoto & Gu, 2021).

As β decreases, the critic passes toward the online regime, where an overestimated value estimate can be revised once the action is executed (Fujimoto et al., 2019). Since the learner acts alone only at the very end, the critic of the learner becomes reliable only as a consequence of handover, not before it, and even this correction is not guaranteed (Yue et al., 2023). Offline-to-online fine-tuning is unstable in exactly this window, where a critic conservative only on its training data carries unregularized overestimates at newly visited states (Nair et al., 2020; Lee et al., 2022; Luo et al., 2023).

Reusing this critic for the gate puts the extrapolation where it does the most damage. The gate update equation 9 is gradient ascent on the critic in β, with gradient $\partial_a Q \cdot (a_{\mathrm{base}} - a_\theta)$: it follows the critic's slope along the segment between the baseline's action and the learner's. Fujimoto et al. (2018) show that ascent on an approximate critic moves toward actions whose estimate exceeds their value. In this setting, this corresponds to the gate moving β toward whichever end of the segment the critic inflates. Because of overestimation of values at unsupported actions, this end is likely the action of the learner. The gate needs a value it can trust before handover, and the learner's critic cannot provide one by construction.

5 Methodology: Arbitrator-Gated Handover

Under critic reuse, the gate is steered by a critic that evaluates the wrong policy and is queried where it extrapolates (Section 4). Arbitrator-gated handover scores the gate by a critic of its own, trained on the same transitions with the blend as continuation policy; we call it the arbitrator. The estimation error of both critics is limited by a prior on the actor toward the baseline and an ensemble aggregated by a lower confidence bound.

5.1 Scoring the Gate by the Policy in Control

Arbitrator critic. The arbitrator is a critic Q_ξ with target copy $\bar{\xi}$, trained on the replay buffer of the learner. Transitions are stored as $(s, a, a_{\mathrm{base}}, r, s', a'_{\mathrm{base}}, d)$, with a'_{base} the action the baseline proposes at s', taken from the following step. The bootstrap action is the one the blend executes at s',

$$a_\beta(s') = \beta(s') \odot a'_{\mathrm{base}} + (\mathbf{1} - \beta(s')) \odot \tilde{a}', \qquad \tilde{a}' \sim \pi_\theta(\cdot \mid s'), \tag{11}$$

and the arbitrator regresses onto

$$y^{\mathrm{arb}} = r + \gamma(1-d)\, Q_{\bar{\xi}}\big(s', a_\beta(s')\big), \qquad (s, a, a_{\mathrm{base}}, r, s', a'_{\mathrm{base}}, d) \sim \mathcal{D}, \tag{12}$$

by minimizing $\mathbb{E}_{\mathcal{D}}\big[(Q_\xi(s,a) - y^{\mathrm{arb}})^2\big]$. The bootstrap action sets the continuation policy of the estimate (Section 3.1), here the blend, whichever policy collected the data:

$$Q_\xi(s,a) \;\approx\; Q^{\pi_\beta}(s,a). \tag{13}$$

At $\beta \equiv \mathbf{1}$ the arbitrator evaluates the baseline; at $\beta \equiv \mathbf{0}$, the learner.

Gate update. The gate is updated as in equation 9 with the arbitrator in place of the critic of the learner,

$$\hat{F}_s^{\mathrm{arb}}(w) = \mathbb{E}_{a_{\mathrm{rl}} \sim \pi_\theta(\cdot|s)}\Big[Q_\xi\big(s, a_w(s)\big)\Big], \qquad \psi \;\leftarrow\; \psi + \eta_\beta\, \nabla_\psi\, \mathbb{E}_{(s,\, a_{\mathrm{base}}) \sim \mathcal{D}}\big[\hat{F}_s^{\mathrm{arb}}(\beta)\big]. \tag{14}$$

The learner is unchanged by this. With an exact arbitrator, $\hat{F}_s^{\mathrm{arb}} = F_s$, and the update improves on the policy whose value scored it, which is now the policy in control (Section 4). The arbitrator bootstraps on the action of the blend at s', the action it would execute there; unlike the critic of the learner, it is supported by the executed data at every stage of the handover.

5.2 Limiting Estimation Error

Over the course of training, critics are trained using transitions from the blended policy. The action stored with each transition is most influenced by the baseline contribution at the beginning of training, and increasingly by the learner as training proceeds. Because of this, the critic of the learner, by bootstrapping from the learner alone, is queried away from its data. As described in Section 4.2, this can lead to estimation error.

Offline reinforcement learningkeeps the actor near the data by a support constraint (Fujimoto et al., 2019; Kumar et al., 2019), a divergence penalty (Wu et al., 2019), or a behavior cloning term (Fujimoto & Gu, 2021); offline-to-online fine-tuning relaxes the constraint as online data accrue (Nair et al., 2020; Zhao et al., 2022; Beeson & Montana, 2022; Luo et al., 2023). Here the data initially come from the baseline, so the constraint becomes a prior toward it. We add to equation 3 a penalty toward a_{base}, weighted by the gate in each action dimension,

$$\begin{aligned} J_\pi(\theta) = \mathbb{E}_{(s,\, a_{\mathrm{base}}) \sim \mathcal{D}}\, \mathbb{E}_{\tilde{a} \sim \pi_\theta(\cdot|s)}\Big[& \min_{i=1,2} Q_{\phi_i}(s, \tilde{a}) - \alpha \log \pi_\theta(\tilde{a} \mid s) \\ & - \lambda\, |\bar{Q}| \sum_{l=1}^{k} \beta_l(s) \Big(\frac{(\tilde{a} - a_{\mathrm{base}})_l}{h_l} \Big)^2 \Big], \end{aligned} \tag{15}$$

with $\lambda \geq 0$, h_l the half-range of action dimension l, and $\bar{Q}$ the batch mean of $\min_i Q_{\phi_i}(s, \tilde{a})$, gradient stopped, which scales the penalty to the critic (Fujimoto & Gu, 2021). The gate weight makes the pull toward the baseline proportional to its share of control, so where the baseline is in control, the actor and every candidate blend $a_w(s)$ between it and a_{base} stay close to the executed action, and both critics are queried close to their data.

Pessimistic ensemble. A separate critic fixes what the arbitrator estimates, not how well it estimates it. The gate ascends its critic in β, so the argument of Section 4.2 applies to the arbitrator as well: each step moves the blend away from the executed actions, a single critic extrapolates there, and the ascent turns its error into overestimation the gate cannot detect. The arbitrator is therefore an ensemble of N critics $Q_{\xi_1}, \ldots, Q_{\xi_N}$, each trained as in equation 12 with the bootstrap samples, target copies and prior networks of Section 3.2, whose disagreement grows where the data thins. The gate objective equation 14 uses Q_ξ^{lcb} of equation 6 in place of Q_ξ: the gate ascends the mean value of the blended action and descends the disagreement about it, so handover is held where the blended action leaves the executed data and proceeds where the data supports it.

Layer normalization. The prior keeps the queries of the critic near the data but does not remove extrapolation, since the critics of the learner still bootstrap from the learner's own actions. We therefore apply layer normalization after every hidden layer of each critic. With normalized features

the output of a critic is bounded by the norm of its output weights, so a query away from the data returns a bounded value rather than an extrapolated one (Ball et al., 2023); Yue et al. (2023) show that divergence under bootstrapping is self-excitation, in which generalization from the data to the policy's actions inflates targets faster than predictions, and that normalization can suppress it. Normalized critics are now standard in offline and offline-to-online methods (Tarasov et al., 2023; Nauman et al., 2024). Because normalization bounds extrapolation without removing it (Kim et al., 2025), we keep the prior and the pessimistic ensemble.

The full procedure is provided in Algorithm 1 (Appendix A).

6 Experiments

We evaluate on four continuous control tasks. The baseline controller for each task is model-predictive controller (MPC), built on a nominal model of the plant. In each case, the true plant deviates from that nominal model in a way the MPC cannot correct on its own. Failure is defined per task as a state-constraint violation, which terminates the episode. Details are in Appendix C

- CartPole. A continuous-action version of the classic CartPole balancing task (Towers et al., 2023). In addition to the mass and length mismatch of Tian et al. (2024), the plant is subject to a constant unmodeled lateral force, equivalent to balancing on an inclined track. The MPC sees only the level-track model, so it settles into a persistent tilt that only a constant feedforward can remove.
- CSTR. An exothermic continuous stirred-tank reactor (CSTR), a standard nonlinear process-control benchmark (Fiedler et al., 2023; Tian et al., 2024). The task is to regulate product concentration by adjusting inlet flow and jacket cooling while keeping reactor and jacket temperatures within safe bounds. The plant's reaction kinetics are faster than those of the nominal model, so the reactor releases more heat than the MPC anticipates and thermal runaway becomes the failure mode.
- BiGlucose. Blood-glucose regulation in a simulated type 1 diabetes patient, using a bi-hormonal extension of the Hovorka model (Kalisvaart et al., 2023; Tian et al., 2024). The agent doses insulin and glucagon around a meal, observing only the glucose measurement, its rate of change, and time. The plant differs from the nominal model across a set of patient parameters, and we tighten the hypoglycemia bound so that overdosing insulin is a hard failure.
- Quadrotor2D. The planar quadrotor and MPC from safe-control-gym (Yuan et al., 2022), tracking a figure-8 reference in the vertical plane using two rotor thrusts. The plant experiences a constant lateral wind absent from the nominal model, so the tracking error of the MPC is a bias that can only be canceled by a persistent tilt-and-thrust correction.

We compare against RL-AR (Tian et al., 2024), which combines the same MPC prior with an off-policy learner through a state-dependent focus module, referred to as the gate in this work. The CartPole, CSTR, and BiGlucose environments appear in their work; however, we added additional difficulty. We also compare with SAC (Haarnoja et al., 2018a;b) trained from scratch with no access to the MPC action proposals. SAC indicates the return that unconstrained exploration can reach and the failures incurred in getting. We also report the MPC acting alone on the perturbed plant, which establishes the performance of the prior.

6.1 Performance Return and Violation Results

For each of the four tasks, we evaluate performance by return and number of failures, both during training and in standalone evaluation. Training return is the per-step return of the executed policy, which for the gated methods is the blend at that point in training; training failures count constraint-violation terminations. Standalone evaluation runs the final learner alone ($\beta \equiv 0$, mean action) for twelve episodes on the training plant. Since β need not reach zero by the end of training, this is the direct test of the learner alone. All methods use the same step budget, ten seeds and shared network sizes and hyperparameters (Appendix D).

Across tasks, our method reduces training failures to zero on three tasks and by more than an order of magnitude on CartPole. Figure 1 shows when the two gated methods hand over. The gate of

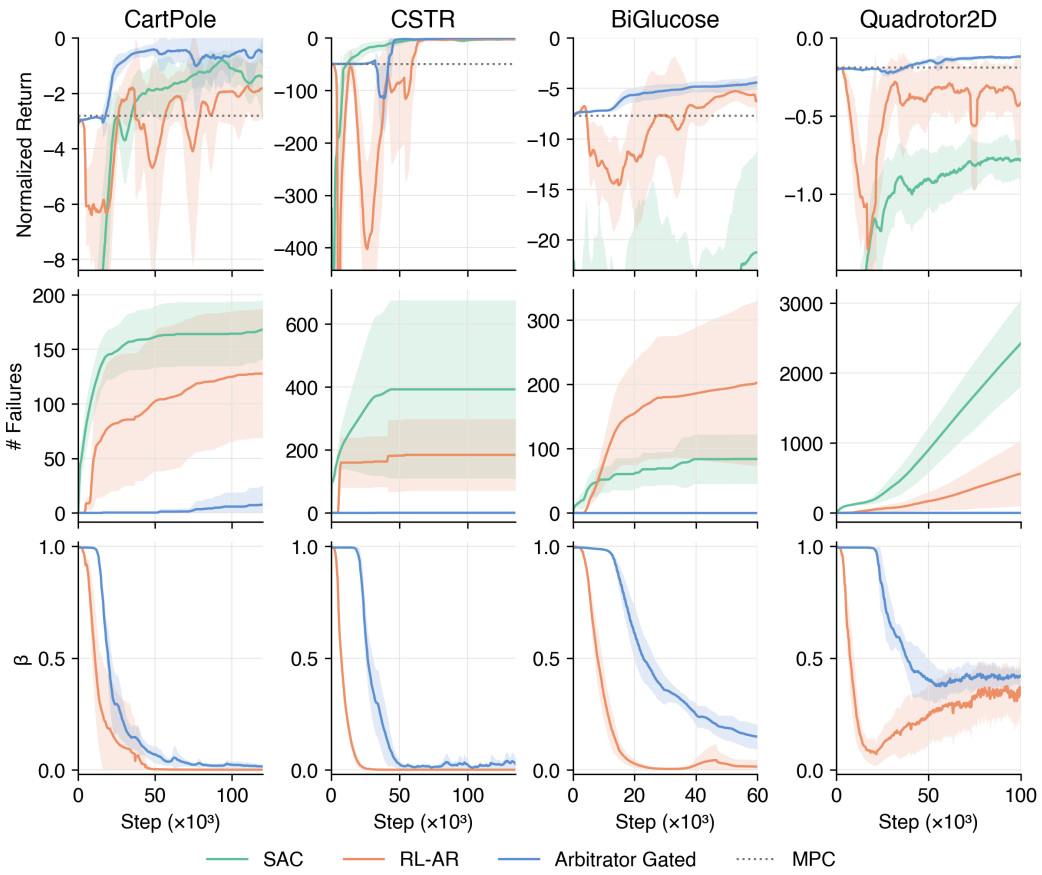


Figure 1: Training curves, mean over ten seeds with 95% confidence bands. Top: per-step return of the executed policy; dotted line, the fixed MPC. Middle: cumulative failures. The return axis is clipped at three times the MPC return, which is why SAC is not visible early on on some tasks. Bottom: gate weight β, mean over action dimensions; SAC has no gate and the MPC is $\beta \equiv 1$.

RL-AR decreases to zero within the first 10–15k steps on CSTR, CartPole and BiGlucose, which is the same window in which its failures accumulate. On Quadrotor2D, our gate settles at a partial blend of about 0.4, where the controller is already close to optimal and the blend outperforms it (Table 1). On CartPole our failure count is 7.6, but is carried entirely by one seed of ten. Per-seed failure counts for every method and task are listed in Appendix E.

The reduction in failures does not come at the cost of return. The executed policy starts at the level of the baseline control and improves past it on every task. RL-AR reaches comparable final returns on CSTR and CartPole, but only after its failures, and remains below ours on BiGlucose and Quadrotor2D. SAC matches the final return only on CSTR, and plateaus below the controller elsewhere; on BiGlucose it converges to a hyperglycemic policy that avoids the constraint violation at the expense of return.

The learner is also competent on its own. Our standalone learner is safe in all forty runs and obtains the best standalone return on every task. The learners of both RL-AR and SAC are safe two of four environments.

6.2 Ablation Study

We ablate the method on CSTR and Quadrotor2D by adding one component at a time to RL-AR: LayerNorm on the learner critic, the independent gate critic as a twin critic, then as the ensemble,

Table 1: Training and standalone results, mean $\pm$ standard deviation over ten seeds. Training failures: constraint-violation terminations over a run. Standalone: the final learner alone ($\beta \equiv \mathbf{0}$, mean action), twelve episodes on the training plant. The MPC row is the fixed controller over twenty episodes.

Task	Method	Training failures	Standalone return	Standalone failures
CartPole	MPC	–	-2.797	0
	SAC	168.3 ± 37.3	-6.828 ± 19.989	1.2 ± 3.8
	RL-AR	127.7 ± 82.4	-1.839 ± 1.476	0.0 ± 0.0
	Arbitrator Gated	7.6 ± 24.0	-0.464 ± 0.722	0.0 ± 0.0
CSTR	MPC	–	-49.761	0
	SAC	392.6 ± 396.0	-2.144 ± 1.455	0.0 ± 0.0
	RL-AR	184.4 ± 159.4	-1.691 ± 1.840	0.0 ± 0.0
	Arbitrator Gated	0.4 ± 1.0	-0.183 ± 0.084	0.0 ± 0.0
BiGlucose	MPC	–	-7.674	0
	SAC	83.8 ± 53.8	-22.314 ± 15.610	0.0 ± 0.0
	RL-AR	202.6 ± 179.1	-64.164 ± 186.243	1.2 ± 3.8
	Arbitrator Gated	0.0 ± 0.0	-2.945 ± 0.353	0.0 ± 0.0
Quadrotor2D	MPC	–	-0.188	0
	SAC	2424.7 ± 862.9	-3.624 ± 0.993	11.8 ± 0.6
	RL-AR	561.1 ± 646.9	-0.909 ± 0.959	6.4 ± 5.7
	Arbitrator Gated	0.1 ± 0.3	-0.071 ± 0.016	0.0 ± 0.0

and finally the actor prior (Appendix F). LayerNorm on the learner critic alone removes most of the training failures, from 184 to 1.8 per seed on CSTR and from 561 to 33 on Quadrotor2D, and the gate settles higher, which is the expected effect of a critic that no longer overrates the learner. The learner it produces is safe standalone on Quadrotor2D but fails in a third of the standalone episodes on CSTR. The independent gate critic brings Quadrotor2D to about one failure per seed, but the gate then rarely opens: its final β averages 0.78 on CSTR and 0.79 on Quadrotor2D. This refusal is the desired behavior, as the learner it withholds fails in most standalone episodes, so the gate declines to hand over to a learner that is not ready. The ensemble holds the gate at the same level but holds it more safely: on CSTR nine of ten seeds have at most five training failures, against four twin-critic seeds at 15 to 119, and one seed diverges, a failure of the gate critic that does not recur in the full method. What resolves the gate hold is the actor prior, which keeps the learner close to the executed actions while the gate is closed. With it, the gate opens in every CSTR seed at 0.4 failures per seed, with a standalone learner two orders of magnitude better than the controller, and Quadrotor2D reaches 0.1 failures per seed with every standalone learner safe.

7 Conclusion

Existing handover gates steer by the learner's own critic. We showed that this critic evaluates the wrong policy, since handover must be judged by the value of the blend in control. We also examine how critic value estimation error may manifest in the online control adaptation setting, relative to knowon issues in the offline and offline-to-online learning domains. Either error can misdirect the gate, the first even when the critic is exact. Arbitrated handover gates on an arbitrator that evaluates the blend, keeps the learner near the executed actions with a prior toward the baseline, and withholds handover where an ensemble of arbitrators disagrees. On four control tasks with model mismatch, this cut training failures by one to three orders of magnitude, raised the return of the executed policy above the baseline controller, and produced final learners that are safe and outperform the controller on their own. In this work, although our gate reduces violations empirically, it does not bound them. As a direction for future work, this method could be extended with formal safety guarantees, for example, using a safety filter.

AI USE STATEMENT

In this work, we used generative AI tools for assistance in code implementation, writing review, and finding related literature. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments or to interpret results.

Tasks including data set generation, cleaning and reformatting, translation assistance, qualitative and thematic data analysis support are not applicable to this work.

We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The entire code base and raw results will be published in a GitHub repository for final publication of the work.

REFERENCES

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified Q-ensemble. In *Advances in Neural Information Processing Systems*, volume 34, pp. 7436–7447. Curran Associates, Inc., 2021.

Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhihong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Y4cs1Z3HnqL.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 1577–1594. PMLR, 2023.

Alex Beeson and Giovanni Montana. Improving TD3-BC: Relaxed policy constraint for offline learning and stable online fine-tuning. In *3rd Offline Reinforcement Learning Workshop at NeurIPS 2022*, 2022. URL https://openreview.net/forum?id=roLMh1mcp6. arXiv:2211.11802.

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*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1141–1150. PMLR, 2019.

Emma Cramer, Lukas Jäschke, and Sebastian Trimpe. CHEQ-ing the box: Safe variable impedance learning for robotic polishing. *IFAC-PapersOnLine*, 59(18):325–330, 2025. doi: 10.1016/j.ifacol.2025.10.241.

Felix Fiedler, Benjamin Karg, Lukas Lüken, Dean Brandner, Moritz Heinlein, Felix Brabender, and Sergio Lucia. do-mpc: Towards FAIR nonlinear and robust model predictive control. *Control Engineering Practice*, 140:105676, 2023. doi: 10.1016/j.conengprac.2023.105676.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20132–20145. Curran Associates, Inc., 2021.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2052–2062. PMLR, 2019.

Seyed Kamyar Seyed Ghasemipour, Shixiang Shane Gu, and Ofir Nachum. Why so pessimistic? estimating uncertainties for offline RL through ensembles, and why their independence matters. In *Advances in Neural Information Processing Systems*, volume 35, pp. 18267–18281. Curran Associates, Inc., 2022. doi: 10.52202/068431-1328.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b. doi: 10.48550/arXiv.1812.05905.

Ronald A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in Artificial Life: Third European Conference on Artificial Life*, volume 929 of *Lecture Notes in Computer Science*, pp. 704–720, Berlin, Heidelberg, 1995. Springer. doi: 10.1007/3-540-59496-5_337.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline RL? In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5084–5096. PMLR, 2021.

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 *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029, 2019. doi: 10.1109/ICRA.2019.8794127.

Dylan Kalisvaart, Jorge Bonekamp, and Sergio Grammatico. Bi-hormonal linear time-varying model predictive control for blood glucose regulation in type 1 diabetes patients. In *2023 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 552–558, 2023. doi: 10.1109/CCTA54093.2023.10252997.

Jeonghye Kim, Yongjae Shin, Whiyoung Jung, Sunghoon Hong, Deunsol Yoon, Youngchul Sung, Kanghoon Lee, and Woohyung Lim. Penalizing infeasible actions and reward scaling in reinforcement learning with offline data. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 30769–30790. PMLR, 2025.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy Q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic Q-ensemble. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1702–1712. PMLR, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. doi: 10.48550/arXiv.2005.01643.

Yicheng Luo, Jackie Kay, Edward Grefenstette, and Marc Peter Deisenroth. Finetuning from offline reinforcement learning: Challenges, trade-offs and practical solutions. *arXiv preprint arXiv:2303.17396*, 2023. doi: 10.48550/arXiv.2303.17396.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020. doi: 10.48550/arXiv.2006.09359.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, Regularized, Optimistic: scaling for compute and sample efficient continuous control. In *Advances in Neural Information Processing Systems*, volume 37, pp. 113038–113071. Curran Associates, Inc., 2024. doi: 10.52202/079017-3592.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. John Wiley & Sons, New York, 1994. doi: 10.1002/9780470316887.

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*, 42(3):123–146, 2023. doi: 10.1177/02783649231167210.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018. doi: 10.48550/arXiv.1812.06298.

Alex Simpkins. System identification: Theory for the user, 2nd edition (Ljung, l.; 1999) [on the shelf]. *IEEE Robotics & Automation Magazine*, 19(2):95–96, 2012. doi: 10.1109/MRA.2012.2192817.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 1998. ISBN 0-262-19398-1.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11592–11620. Curran Associates, Inc., 2023. doi: 10.52202/075280-0511.

Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In Michael C. Mozer, Paul Smolensky, David S. Touretzky, Jeffrey L. Elman, and Andreas S. Weigend (eds.), *Proceedings of the 1993 Connectionist Models Summer School*, pp. 255–263, Hillsdale, NJ, 1993. Lawrence Erlbaum Associates.

Haozhe Tian, Homayoun Hamedmoghadam, Robert Shorten, and Pietro Ferraro. Reinforcement learning with adaptive regularization for safe control of critical systems. In *Advances in Neural Information Processing Systems*, volume 37, pp. 2528–2557. Curran Associates, Inc., 2024. doi: 10.52202/079017-0083.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Jin Shen Tan, and Omar G. Younis. Gymnasium. Zenodo, software release v0.28.1, 2023. URL `https://zenodo.org/records/8127026`.

Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, Sergey Levine, and Karol Hausman. Jump-start reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 34556–34583. PMLR, 2023.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019. doi: 10.48550/arXiv.1911.11361.

Zhaocong Yuan, Adam W. Hall, Siqi Zhou, Lukas Brunke, Melissa Greeff, Jacopo Panerati, and Angela P. Schoellig. Safe-Control-Gym: A unified benchmark suite for safe learning-based control and reinforcement learning in robotics. *IEEE Robotics and Automation Letters*, 7(4):11142–11149, 2022. doi: 10.1109/LRA.2022.3196132.

Yang Yue, Rui Lu, Bingyi Kang, Shiji Song, and Gao Huang. Understanding, predicting and better resolving Q-value divergence in offline-RL. In *Advances in Neural Information Processing Systems*, volume 36, pp. 60247–60277. Curran Associates, Inc., 2023. doi: 10.52202/075280-2632.

Yi Zhao, Rinu Boney, Alexander Ilin, Juho Kannala, and Joni Pajarinen. Adaptive behavior cloning regularization for stable offline-to-online reinforcement learning. In *Proceedings of the 30th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2022)*, pp. 545–550, Bruges, Belgium, 2022. Ciaco – i6doc.com. doi: 10.14428/esann/2022.ES2022-110.

A Algorithm Details

Algorithm 1 Arbitrated Gated Handover

Require: baseline π_{base}; actor π_θ; learner critics Q_{ϕ_1}, Q_{ϕ_2}; arbitrator ensemble $Q_{\xi_1}, \dots, Q_{\xi_N}$ with random priors; target copies of all critics; gate β_ψ pretrained to $\approx \mathbf{1}$; replay buffer $\mathcal{D}$; mask probability p

1: **for** each environment step **do**
2: $a_{\text{base}} \leftarrow \pi_{\text{base}}(s)$, $a_{\text{rl}} \sim \pi_\theta(\cdot \mid s)$; execute $a \leftarrow \beta(s) \odot a_{\text{base}} + (\mathbf{1} - \beta(s)) \odot a_{\text{rl}}$; observe r, s', d
3: bootstrap mask $m_j \sim \text{Bernoulli}(p)$, $j = 1, \dots, N$; store $(s, a, a_{\text{base}}, r, s', a'_{\text{base}}, d, m)$ in $\mathcal{D}$, with a'_{base} from the following step
4: sample a batch from $\mathcal{D}$ and $\tilde{a}' \sim \pi_\theta(\cdot \mid s')$ for each transition
5: learner critics: regress $Q_{\phi_i}(s, a)$ onto y of equation 4, $i = 1, 2$
6: arbitrator: regress $Q_{\xi_j}(s, a)$ onto y^{arb} of equation 12 with its own target copy $\bar{\xi}_j$, on the transitions with $m_j = 1$, $j = 1, \dots, N$
7: actor: ascend equation 15 in θ; update α as in SAC
8: gate: ascend equation 14 with equation 6 in ψ
9: update the target copies of all critics
10: **end for**

B Proof of Proposition 1

Let $\mathcal{S} = \{s_{\text{risk}}, s_{\text{ok}}, s_{\text{fail}}\}$, $\mathcal{A} = [0, 1]$, $\pi_{\text{base}} \equiv 0$, and $c > (1 - \gamma)/\gamma$. For every $a \in \mathcal{A}$, let

$$\begin{aligned} r(s_{\text{risk}}, a) &= a, & P(s_{\text{ok}} \mid s_{\text{risk}}, a) &= 1 - a, \quad P(s_{\text{fail}} \mid s_{\text{risk}}, a) = a, \\ r(s_{\text{ok}}, a) &= (1 - a)c, & P(s_{\text{ok}} \mid s_{\text{ok}}, a) &= 1, \\ r(s_{\text{fail}}, a) &= 0, & P(s_{\text{fail}} \mid s_{\text{fail}}, a) &= 1, \end{aligned}$$

with all other transition probabilities zero. The states s_{ok} and s_{fail} are absorbing, and $V^\pi(s_{\text{fail}}) = 0$ for every policy π.

A policy with mean action m at s_{ok} has value $(1-m)c/(1-\gamma)$ there. The blend in control executes the baseline at s_{ok} because $\beta(s_{\text{ok}}) = 1$, and the learner has mean action $\bar{a}_{\text{ok}} := \mathbb{E}_{a \sim \pi_\theta(\cdot \mid s_{\text{ok}})}[a]$, so

$$v := V^{\pi_\beta}(s_{\text{ok}}) = V^{\pi_{\text{base}}}(s_{\text{ok}}) = \frac{c}{1 - \gamma} > \frac{1}{\gamma}, \qquad u := V^{\pi_\theta}(s_{\text{ok}}) = \frac{(1 - \bar{a}_{\text{ok}})\, c}{1 - \gamma} < \frac{1}{\gamma},$$

by the choice of c and the hypothesis on the learner, so that $\bar{a}_{\mathrm{ok}} > 0$.

For every policy π and every $a \in \mathcal{A}$, since $V^{\pi}(s_{\mathrm{fail}}) = 0$,

$$Q^{\pi}(s_{\mathrm{risk}}, a) = a + \gamma(1-a)V^{\pi}(s_{\mathrm{ok}}) = \gamma V^{\pi}(s_{\mathrm{ok}}) + a\left(1 - \gamma V^{\pi}(s_{\mathrm{ok}})\right), \tag{16}$$

which is affine in a. The baseline action is 0, so a weight w executes $(1-w)a_{\mathrm{rl}}$ at s_{risk}, with mean $(1-w)\bar{a}$. With Q^{π_θ} in the substitute and Q^{π_β} in the handover objective, taking expectations in equation 16 gives

$$\hat{F}_{s_{\mathrm{risk}}}(w) = \gamma u + (1-w)\,\bar{a}\,(1-\gamma u), \qquad F_{s_{\mathrm{risk}}}(w) = \gamma v + (1-w)\,\bar{a}\,(1-\gamma v),$$

with slopes $-\bar{a}\,(1-\gamma u) < 0$ and $-\bar{a}\,(1-\gamma v) > 0$ for every $w \in [0,1]$, since $\bar{a} > 0$ and $u < 1/\gamma < v$. The substitute is strictly decreasing in w, so $\hat{\beta}(s_{\mathrm{risk}}) = 0$ is its only maximizer.

At s_{ok}, $Q^{\pi_\theta}(s_{\mathrm{ok}}, a) = (1-a)c + \gamma u$, so $\hat{F}_{s_{\mathrm{ok}}}(w) = \big(1 - (1-w)\bar{a}_{\mathrm{ok}}\big)c + \gamma u$ has slope $c\,\bar{a}_{\mathrm{ok}} > 0$, and $\hat{\beta}(s_{\mathrm{ok}}) = 1$ is its only maximizer, matching the gate in control.

Both $\hat{\pi}$ and π_β therefore execute the baseline at s_{ok}, so $V^{\hat{\pi}}(s_{\mathrm{ok}}) = V^{\pi_\beta}(s_{\mathrm{ok}}) = v$ and, by equation 16, $Q^{\hat{\pi}}(s_{\mathrm{risk}}, \cdot) = Q^{\pi_\beta}(s_{\mathrm{risk}}, \cdot)$. Since $\hat{\pi}$ executes the weight 0 at s_{risk} and π_β the weight $\beta(s_{\mathrm{risk}})$,

$$V^{\pi_\beta}(s_{\mathrm{risk}}) - V^{\hat{\pi}}(s_{\mathrm{risk}}) = F_{s_{\mathrm{risk}}}\big(\beta(s_{\mathrm{risk}})\big) - F_{s_{\mathrm{risk}}}(0) = \beta(s_{\mathrm{risk}})\,\bar{a}\,(\gamma v - 1) > 0.$$

C Environments

We extend the reference implementation of of RL-AR Tian et al. (2024) for CSTR, CartPole and BiGlucose; dynamics, nominal parameters, rewards and MPC formulations are in its Appendix B. In every environment the MPC plans on the nominal model and only the plant is changed. We list what differs from the reference and specify Quadrotor2D in full as it is not included in the RL-AR work.

C.1 CartPole

Episodes of 250 steps, $\Delta t = 0.02$ s; dynamics, reward and the mass/length mismatch as in Tian et al. (2024), Eqs. (47)–(49) and Table 6. Failure at $|\theta| > 12^\circ$ or $|x| > 4.8$ (the reference implementation's rail limit), penalty 10^4. In addition the plant applies a constant force bias,

$$F = 10\,(a_f + b), \qquad b = 0.2, \tag{17}$$

added before the dynamics and not clipped to the actuator range. MPC as in RL-AR: horizon 100, stage and terminal cost $100\,x^2 + 100\,\theta^2$, input-rate penalty 100, bounds $|a_f| \le 1$, $|\theta| \le 12^\circ$.

C.2 CSTR

Episodes of 300 steps, $\Delta t = 0.005$ h; dynamics, reward and failure bounds as in Tian et al. (2024), Eqs. (44)–(46). The plant's rate constants are

$$K_1 = \beta\, k_{0,ab}\, e^{-E_{A,ab}/(T_R+273.15)}, \qquad K_3 = k_{0,ad}\, e^{-\alpha E_{A,ad}/(T_R+273.15)}, \tag{18}$$

with $\alpha = \beta = 1$ in the nominal model and $\alpha = 1.15$, $\beta = 1.3$ on the plant (RL-AR: 1.05, 1.10). MPC as in RL-AR: horizon 20, stage and terminal cost $(C_B - 0.6)^2$, input-rate penalties 10^{-3} (F) and 2.5×10^{-4} ($\dot{Q}$), state bounds equal to the failure bounds except $T_K \le 140\,^\circ$C, inputs $F \in [0.5, 10]$, $\dot{Q} \in [-8.5, 0]$.

C.3 BiGlucose

Episodes of 200 steps, $\Delta t = 10$ min, one announced meal of 80 g at $t = 0$; dynamics, reward and plant parameters as in Tian et al. (2024), Eqs. (39)–(43) and Table 3. The hypoglycemia failure bound is raised from 10 to 80 mg/dL (failure at $G < 80$ or $G > 1000$, reference penalty). The action space is $a_I \in [0, 1]$, $a_N \in [0, 5]$. MPC as in RL-AR except the glucose bound: horizon 20,

stage and terminal cost the Magni risk of predicted glucose, input penalties $100\,a_I^2 + a_N^2$, planner bounds $a_I \in [0, 4]$, $a_N \in [0, 5]$ (the solution is clipped to the action space before execution) and $G \in [80, 800]$ mg/dL (the reference lower bound is 50), meal announced over the horizon. Only G is measured; the MPC's remaining eleven states come from a nominal-model copy driven by the executed actions.

C.4 QUADROTOR2D

The 2D quadrotor environment from safe-control-gym's Yuan et al. (2022) (commit `6b5391d`, pybullet 3.2.7) with its nominal dynamics, thrust limits and state bounds unchanged: PyBullet physics at 1000 Hz, control at 50 Hz, 6 s episodes (300 steps). Task `traj_tracking` on one `figure8` cycle in the x–z plane, scale 0.75, offset $(0, 1)$; quadratic cost with state weights $(5, 0.1, 5, 0.1, 0.1, 0.1)$ and input weights $(0.1, 0.1)$; initial state uniform in $p_x \in [-0.5, 0.5]$, $p_z \in [0.5, 1.5]$, $\theta \in [-0.2, 0.2]$, velocities in $[-0.1, 0.1]$. Our wrapper adds: failure (penalty 100) on the library's out-of-bounds flag or $p_z < 0.05$ m; y, roll and yaw reset to zero after every step; the learner observes $(x_t, x_t^r - x_t)$ while the MPC receives x_t. Perturbation: a constant lateral force of $0.25\,mg$ on the plant (library `step` disturbance, mask $(1, 0)$), absent from the nominal model. MPC: the library's nonlinear MPC, horizon 20, the task's cost weights, soft constraints.

D TRAINING DETAILS

All three arms share one code base, the RL-AR reference implementation Tian et al. (2024), and one budget per environment (Table 3). SAC is the learner alone: the same actor and twin critic with no regularizer and no gate. RL-AR is the reference method unchanged. Two protocol changes apply to every arm: only constraint violations are terminal, so time-limit truncations bootstrap; and each episode starts from the reference initial state perturbed per dimension by $1 + 0.05\,u$, $u \sim U(-1, 1)$, from the environment's seeded generator (Quadrotor2D draws its initial state from the library ranges, Appendix C.4). The gate network starts from a checkpoint pretrained once per environment to $\beta = 0.999$ on uniformly sampled states and shared by all seeds of RL-AR and ours.

Training failures count constraint-violation terminations over the budget. The standalone actor is the deterministic policy mean with $\beta = 0$ for 12 episodes on the training plant (reset seeds $6000 + i$); MPC alone is the regularizer for 20 episodes. Final β is the episode-mean β averaged over the last 15 training episodes and over action dimensions. Every arm uses 10 seeds per environment.

Table 2: Training hyperparameters. The shared block follows RL-AR (Table 7) and applies to all arms; the remaining blocks list the components each arm adds. The actor-prior coefficient λ is set per environment (Table 3).

All arms	
Actor, critic hidden layers (ReLU)	$[256, 256]$
Critic / policy learning rate (Adam)	10^{-3} / 3×10^{-4}
Batch size, learning starts	256
Replay buffer	10^6
Discount γ	0.99
Target update τ	0.005
Policy update period	2
Entropy temperature α (fixed)	0.2
Policy log-std range	$[-5, 2]$
RL-AR and ours: focus module	
Hidden layers	$[128, 32]$
Learning rate	5×10^{-6}
Pretraining target, learning rate	0.999, 10^{-5}
Ours: gate critic ensemble	
Members K	5
Member trunk	$[256, 256]$, LayerNorm
Randomized prior	fixed $[256, 256]$ net, scale 10
Bootstrap mask probability	0.8
Learning rate, τ	10^{-3}, 0.005
Ours: other components	
Learner critic	LayerNorm
Actor prior target, weight	MPC action, $\beta(s)$
Actor prior applied to, schedule	policy sample, constant λ

Table 3: Per-environment settings. Plant perturbations and MPC formulations are in Appendix C.

	CSTR	CartPole	BiGlucose	Quadrotor2D
Training steps	135k	120k	60k	100k
Episode length (steps)	300	250	200	300
MPC horizon	20	100	20	20
Actor-prior coefficient λ	10	1	0.3	0.1

E Additional Training Results

Table 4: Training failures per seed for each task and method (AG: Arbitrator Gated), with totals. Seeds are shared across methods within a task.

Seed	CartPole			CSTR			BiGlucose			Quadrotor2D		
	SAC	RL-AR	AG	SAC	RL-AR	AG	SAC	RL-AR	AG	SAC	RL-AR	AG
1	211	20	0	1340	130	0	77	140	0	449	333	0
2	193	92	0	302	30	0	78	248	0	2156	1473	0
3	201	181	76	97	584	3	22	169	0	2745	430	0
4	168	252	0	109	306	0	106	194	0	3961	1679	0
5	194	161	0	236	212	0	106	17	0	2275	1231	0
6	116	18	0	175	122	1	42	161	0	2490	26	0
7	119	98	0	235	81	0	207	356	0	2727	34	1
8	151	71	0	632	161	0	96	620	0	2752	36	0
9	203	239	0	703	117	0	84	121	0	2458	37	0
10	127	145	0	97	101	0	20	0	0	2234	332	0
Total	1683	1277	76	3926	1844	4	838	2026	0	24247	5611	1

F Ablation Studies

Table 5: Ablation training results. Each row adds one component to the row above; the last row is the full method. Training failures and the final β, mean $\pm$ sd over ten seeds.

Task	Variant	Training failures	Final β
CSTR	RL-AR	184.4 ± 159.4	0.00 ± 0.00
	+ learner-critic LayerNorm	1.8 ± 3.5	0.30 ± 0.48
	+ twin gate critic	20.9 ± 37.8	0.78 ± 0.37
	+ ensemble gate critic[†]	4210 ± 13310	0.70 ± 0.47
	+ actor prior (full)	0.4 ± 1.0	0.03 ± 0.04
Quadrotor2D	RL-AR	561.1 ± 646.9	0.34 ± 0.17
	+ learner-critic LayerNorm	33.0 ± 26.0	0.55 ± 0.14
	+ twin gate critic	1.1 ± 1.6	0.79 ± 0.17
	+ ensemble gate critic	1.0 ± 1.7	0.71 ± 0.12
	+ actor prior (full)	0.1 ± 0.3	0.42 ± 0.05

[†]One seed of the ensemble-gate variant on CSTR diverged (42,091 failures; see table below). Over the remaining nine seeds the variant has 1.0 ± 2.0 training failures, final β 0.78 ± 0.42.

Table 6: Ablation standalone results. Standalone return and failures of the final learner ($\beta \equiv 0$, mean action, twelve episodes) for the variants in the table above. The return includes the failure penalty.

Task	Variant	Standalone return	Standalone failures
CSTR	RL-AR	-1.691 ± 1.840	0.0 ± 0.0
	+ learner-critic LayerNorm	-3002 ± 4831	3.6 ± 5.8
	+ twin gate critic	-3842 ± 4474	7.2 ± 6.2
	+ ensemble gate critic†	-5037 ± 4561	6.1 ± 6.2
	+ actor prior (full)	-0.183 ± 0.084	0.0 ± 0.0
Quadrotor2D	RL-AR	-0.909 ± 0.959	6.4 ± 5.7
	+ learner-critic LayerNorm	-0.059 ± 0.056	0.0 ± 0.0
	+ twin gate critic	-2.424 ± 3.331	4.8 ± 6.2
	+ ensemble gate critic	-1.272 ± 2.198	3.7 ± 5.7
	+ actor prior (full)	-0.071 ± 0.016	0.0 ± 0.0

Table 7: Training failures per seed for ablation method, with totals. Seeds are shared across methods within a task.

Seed	CSTR					Quadrotor2D				
	RL-AR	LN	Twin	Ensemble	Full	RL-AR	LN	Twin	Ensemble	Full
1	130	0	1	0	0	333	66	2	0	0
2	30	0	48	5	0	1473	18	1	3	0
3	584	11	0	4	3	430	26	5	1	0
4	306	4	23	0	0	1679	26	0	0	0
5	212	0	3	0	0	1231	82	1	0	0
6	122	1	15	42091	1	26	54	2	5	0
7	81	0	0	0	0	34	5	0	0	1
8	161	0	0	0	0	36	7	0	0	0
9	117	0	119	0	0	37	14	0	0	0
10	101	2	0	0	0	332	32	0	1	0
Total	1844	18	209	42100	4	5611	330	11	10	1