Self-Supervised On-Policy Reinforcement Learning via Contrastive Proximal Policy Optimisation

Asim Osman[1,2,†], **Sasha Abramowitz**[1,†], **Mark Bergh**[1,2], **Ulrich Armel Mbou Sob**[1], **Ruan John de Kock**[1], **Omayma Mahjoub**[1], **Oussama Hidaoui**[1,2], **Noah De Nicola**[1], **Arnol Manuel Fokam**[1], **Felix Chalumeau**[1], **Daniel Rajaonarivonivelomanantsoa**[1,3], **Siddarth Singh**[1], **Refiloe Shabe**[1], **Juan Claude Formanek**[1], **Simon Verster Du Toit**[1], **Arnu Pretorius**[1]

[1]**InstaDeep**
[2]**African Institute for Mathematical Sciences (AIMS)**
[3]**University of Stellenbosch**

Abstract

Contrastive reinforcement learning (CRL) learns goal-conditioned Q-values through a contrastive objective over state-action and goal representations, removing the need for hand-crafted reward functions. Despite impressive success in achieving viable self-supervised learning in RL, all existing CRL algorithms rely on off-policy optimisation and are mostly constrained to continuous action spaces, with little research invested in discrete environments. This leaves CRL disconnected from widely used and effective, modern on-policy training pipelines adopted across both single-agent and multi-agent RL in continuous and discrete environments. To establish a first connection, we introduce Contrastive Proximal Policy Optimisation (CPPO). CPPO is an on-policy contrastive RL algorithm that derives policy advantages directly from contrastive Q-values and optimises them via the standard PPO objective, without requiring a reward function or a replay buffer. We evaluate CPPO across continuous and discrete, single-agent and cooperative multi-agent tasks. Whilst the existence of an on-policy approach is inherently useful, we observe that **CPPO not only significantly outperforms the previous CRL baselines in 14 out of 18 tasks, but also matches or exceeds PPO's performance, which uses hand-crafted dense rewards, in 12 out of the 18 tasks tested.**

1 Introduction

Designing reward functions is one of the most fragile parts of applying reinforcement learning in practice. Sparse rewards make exploration intractable, dense rewards introduce specification bias and reward hacking, and small misspecifications can produce degenerate behaviour (Ng et al., 1999; Pan et al., 2022; Skalse et al., 2022). Contrastive reinforcement learning (CRL) (Eysenbach et al., 2022; Liu et al., 2024), sidesteps this entirely: rather than maximising a scalar reward, the agent learns to reach goal states, using a Q-function trained via a contrastive objective. Since its introduction, CRL has matured rapidly, with progress in emergent exploration, scalable training, deeper architectures, and multi-agent extensions (Bortkiewicz et al., 2024; Nimonkar et al., 2025; Wang et al., 2025)

Yet every existing CRL method to date shares an underlying design choice: they are all off-policy, built on SAC (Haarnoja et al., 2018a) or DQN (Mnih et al., 2013). This reflects a design choice inherited from CRL's SAC-based origins that has propagated through the literature, leaving the

[†]Equal contribution. Correspondence: asim@aims.ac.za

Table 1: ***Design comparison of contrastive RL algorithms***. We highlight with * the method naming convention used in our experiments. It aims to remove ambiguity by connecting the underlying base RL algorithm to the contrastive learning approach. In our work, CPPO, is a self-supervised RL method with a PPO base that is on-policy and naturally suited for both discrete and continuous action spaces. We categorise prior work similarly.

Source	Method*	On-Policy	Discrete	Continuous	Multi-Agent
CRL (Eysenbach et al., 2022)	CSAC	×	×	✓	×
CRL (Bastankhah et al., 2025)	CDQN	×	✓	×	×
ICRL (Nimonkar et al., 2025)	ICSAC	×	✓†	✓	✓
Ours	I/CPPO	✓	✓	✓	✓

†Via Gumbel-Softmax approximation.

contrastive paradigm without an on-policy counterpart. In reward-based RL, on-policy and off-policy methods have developed as parallel families with complementary strengths: DQN (Mnih et al., 2013) alongside A3C (Mnih et al., 2016), SAC (Haarnoja et al., 2018a) alongside PPO (Schulman et al., 2017), MADDPG (Lowe et al., 2017) alongside IPPO (De Witt et al., 2020), in the multi-agent setting. Off-policy methods offer sample efficiency through replay; on-policy methods offer stability, simpler implementation (Sutton & Barto, 2018), and strong compatibility with massively parallel simulation (Lu et al., 2022; Bonnet et al., 2023; de Kock et al., 2023), where PPO has become the standard choice (Makoviychuk et al., 2021; Rudin et al., 2022). In cooperative MARL on-policy methods have become the prevalent paradigm (Wen et al., 2022; Yu et al., 2022; Mahjoub et al., 2024). The contrastive RL literature has no counterpart for on-policy learning, meaning practitioners who want a goal-conditioned method must accept the instabilities and design constraints of off-policy learning, even in settings where on-policy methods would otherwise be the natural choice.

In this work, we introduce Contrastive Proximal Policy Optimisation (CPPO)[1], the first on-policy contrastive RL algorithm. CPPO derives policy advantages directly from a contrastive Q-function and optimises its policy via PPO's clipped surrogate objective. CPPO is a self-supervised method that handles discrete action spaces natively, extends cleanly to the MARL setting, and integrates into existing on-policy training pipelines with minimal modification. Of the 18 tasks benchmarked, CPPO significantly outperforms contrastive baselines on 14 and, despite using no reward signal, it matches or exceeds the performance of reward-based PPO on 12.

Our contributions are as follows:

1. **CPPO**, an on-policy contrastive RL algorithm that computes advantages directly from contrastive Q-values and optimises them via PPO's clipped objective, requiring no reward function, replay buffer or target network.
2. **A general method** that operates across single-agent and multi-agent, discrete and continuous settings without significant algorithmic modifications, while outperforming CRL baselines in discrete settings.
3. **Evaluation of contrastive RL against dense-reward baselines.** Prior CRL work compares primarily against sparse-reward or goal-conditioned methods; we benchmark against PPO with dense-rewards, and contrast goal engineering with reward engineering.

2 Background

Problem Formulation We model the single-agent reinforcement learning (RL) problem as a partially observable Markov decision process (POMDP), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, P, R, \gamma \rangle$, where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ denote the state, action, and observation spaces respectively. At each timestep, the agent receives an observation $o \in \mathcal{O}$, selects an action $a \in \mathcal{A}$, and the environment transitions

[1]See our website for the implementation and hyperparameters

to a new state according to the transition function $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, producing a scalar reward $r = R(s, a)$ and a new observation. The agent's objective is to learn a policy $\pi(a|o)$ that maximises the expected discounted return $J(\pi) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$ where $T \in \mathbb{N}$ is the episode horizon.

We model the cooperative multi-agent reinforcement learning (MARL) problem as a decentralised-POMDP (Dec-POMDP) (Oliehoek & Amato, 2016) where n agents act simultaneously. At each step, agent i selects an action a_i based on its local observation o_i, forming a joint action $\mathbf{a} = (a_1, \ldots, a_n)$ that transitions the environment according to P. In cooperative settings, all agents share a common reward $R : \mathcal{S} \times \mathcal{A}^n \to \mathbb{R}$. A popular approach is independent learning, where each agent optimises its own policy $\pi_i(a_i|o_i)$ using the shared reward signal (De Witt et al., 2020).

Goal-Conditioned RL Goal-conditioned reinforcement learning (GCRL) replaces the scalar reward with a goal: the agent receives a target state $g \in \mathcal{S}$ and must learn a policy $\pi(a \mid s, g)$ that reaches the goal (Kaelbling, 1993; Schaul et al., 2015). Rather than maximising cumulative reward, the objective is to maximise the γ-discounted state occupancy measure at the goal (Eysenbach et al., 2022; Liu et al., 2024):

$$\max_{\pi} \; \rho_\gamma^\pi(g), \qquad \rho_\gamma^\pi(g) \triangleq (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \, p_t^\pi(s_t = g), \tag{1}$$

where $p_t^\pi(s_t = g)$ is the probability of visiting state g at time t under policy π. The corresponding goal-conditioned Q-function is the conditional occupancy measure:

$$Q^\pi(s, a, g) \triangleq \rho_\gamma^\pi(g \mid s, a). \tag{2}$$

In practice, we follow Liu et al. (2024) and condition on a single fixed target goal g^* rather than sampling from a goal distribution, which has been shown to be sufficient to learn goal-conditioned policies via contrastive RL (Liu et al., 2024; Bastankhah et al., 2025; Nimonkar et al., 2025).

Contrastive RL The goal-conditioned Q-function (Equation 2) forms the basis of CRL. It is parameterised by two encoders: a state-action encoder $\phi(s, a)$ and a goal encoder $\psi(g)$, both mapping to a d-dimensional representation space. A critic is defined as a similarity function over these representations and is trained to approximate the goal conditioned Q-function.

The encoders are trained using the InfoNCE objective (van den Oord et al., 2018) on batches of trajectories. For each state-action pair (s_i, a_i), a positive goal g_i^+ is sampled from a future state the policy reached in the same trajectory, a procedure known as hindsight relabeling (HER) (Andrychowicz et al., 2017), while negative goals g_j^- are drawn from other trajectories:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(f(s_i, a_i, g_i^+))}{\sum_{j=1}^{N} \exp(f(s_i, a_i, g_j^-))} \tag{3}$$

Thus the critic is trained to map state-action pairs and achieved goals to a shared latent space where their representations are aligned. A similarity measure over these representations serves as a proxy for the probability of reaching the goal. Our method uses negative L2 distance (see Equation 6) but other distance measures are common in the literature such as taking the inner product (Bortkiewicz et al., 2024). Thus, this is equivalent to the Q-function described by Equation 2.

Prior work takes inspiration from SAC (Haarnoja et al., 2018a) when training the actor. Simply training the actor to pick the action which maximises it's future value.

Multi-Agent CRL Independent Contrastive RL (ICRL) (Nimonkar et al., 2025) extends CRL to cooperative multi-agent settings by reframing the Dec-POMDP as a joint goal-reaching problem. A mapping $m_g \colon \mathcal{O}^{1:n} \to \mathcal{G}$ produces a goal representation from joint observations, with a single fixed target g^* (Liu et al., 2024; Nimonkar et al., 2025). The team's objective is then to maximise the joint state-occupancy of g^*.

ICRL follows the independent learning paradigm with parameter sharing: each agent acts on local observations o_i but all agents share the policy π_θ and the contrastive encoders ϕ_ξ, ψ_ω. Like single-agent CRL, it builds on an off-policy SAC backbone, but uses Straight-Through Gumbel-Softmax (Jang et al., 2016) to handle discrete actions through a continuous relaxation.

Proximal Policy Optimisation PPO (Schulman et al., 2017) is a policy gradient algorithm that stabilises training by constraining each policy update to a trust region. Given a batch of transitions collected under a behaviour policy $\pi_{\theta_{\text{old}}}$, PPO optimises the clipped surrogate objective

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\big[\min\big(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta),\ 1-\epsilon,\ 1+\epsilon)\,\hat{A}_t\big)\big], \tag{4}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)}$ is the probability ratio and ϵ is a clipping threshold. The advantages $\hat{A}_t$ are estimated via Generalised Advantage Estimation (GAE) (Schulman et al., 2015b), which requires a learned value function $V_\psi(o_t)$ and per-step rewards:

$$\hat{A}_t^{\text{GAE}} = \sum_{l=0}^{T-t} (\gamma\lambda)^l\,\delta_{t+l}, \qquad \delta_t = r_t + \gamma\,V_\psi(o_{t+1}) - V_\psi(o_t). \tag{5}$$

In the multi-agent setting, IPPO (De Witt et al., 2020) applies PPO independently to each agent with shared parameters, while MAPPO (Yu et al., 2022) additionally conditions the value function on the global state. Both have become two of the dominant on-policy optimisation approaches in cooperative MARL (Yu et al., 2022; de Kock et al., 2023; Rutherford et al., 2024b).

3 Method

We introduce Contrastive Proximal Policy Optimisation (CPPO), our approach to on-policy contrastive reinforcement learning. We describe: (1) how we use contrastive learning to estimate the advantage, (2) how we tie this into PPO's existing optimisation objective, and (3) how we can extend CPPO to the multi-agent setting.

3.1 Advantage Estimation Without Rewards

CPPO is a policy gradient algorithm that uses contrastive Q-values to compute advantages, replacing the standard GAE (Schulman et al., 2015b) that requires a learned value function and reward signal. Prior CRL methods train the actor by directly maximising the critic (Eysenbach et al., 2022; Nimonkar et al., 2025; Wang et al., 2025): $\max_\pi\ \mathbb{E}_{a\sim\pi}\left[Q(s,a)\right]$ by following either SAC-style policies learned through critic hill-climbing (Haarnoja et al., 2018a) or DQN style ϵ-greedy policies (Mnih et al., 2013). Instead, we use the critic to compute advantages for PPO's clipped objective, with advantage defined as $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$, where $V^\pi(s) = \mathbb{E}_{a\sim\pi}[Q^\pi(s,a)]$.

The advantage formulation above only requires access to a Q function. To instantiate it in the reward-free regime, we obtain Q through contrastive learning, which replaces rewards with goal-reaching as the supervision signal. Concretely, CPPO uses three networks:

- Policy network: $\pi_\theta(a \mid o, g)$ that maps observations and goals to action distributions.
- State-action encoder: $\phi_\xi(o, a)$ that maps observation-action pairs to a d-dimensional representation space.
- Goal encoder: $\psi_\omega(g)$ that maps goals to the same d-dimensional space.

The combination of the state-action encoder and the goal encoder form our Q function as described in Section 2. Therefore, in the goal conditioned regime, we can compute advantages as follows, always using the single fixed target goal g^*:

1. Compute Q-values for the chosen action using the encoders:

$$Q(o, a, g^*) = -\|\phi(o,a) - \psi(g^*)\|_2 \tag{6}$$

Algorithm 1 CPPO: On-Policy Contrastive RL

Initialise policy π_θ, encoders ϕ_ξ, ψ_ω
for iteration $= 1, \ldots, N$ **do**
  Collect rollouts with π_θ across M parallel environments
  Relabel (with HER): sample future observation o_{t+k} from the same trajectory as goal g
  $Q(o, a, g^*) = f\big(\phi_\xi(o, a),\ \psi_\omega(g^*)\big), \quad V(o, g^*) = \sum_a \pi_\theta(a \mid o, g^*)\, Q(o, a, g^*)$ ▷ Eqs. 6,7
  $A = Q - V$ ▷ Eq. 9
  for epoch $= 1, \ldots, K$ **do**
    Update ϕ_ξ, ψ_ω via $\mathcal{L}_{\text{InfoNCE}}$ ▷ Eq. 3
    Update π_θ via $\mathcal{L}_{\text{PPO}}$ with advantages A ▷ Eq. 4
  end for
end for

2. Compute the value of the current state. For discrete action spaces, we compute the expectation over all actions:

$$V(o, g^*) = \mathbb{E}_{a \sim \pi(\cdot \mid o, g^*)}[Q(o, a, g^*)] = \sum_{a \in \mathcal{A}} \pi(a \mid o, g^*) \cdot Q(o, a, g^*). \tag{7}$$

For continuous action spaces, the value is the integral over all actions $V(o, g^*) = \int_{\mathcal{A}} \pi(a \mid o, g^*) Q(o, a, g^*) da$. Since this is intractable in practice, we estimate it via Monte Carlo sampling with K actions drawn from the policy:

$$V(o, g^*) \approx \frac{1}{K} \sum_{k=1}^{K} Q(o, a_k, g^*), \quad a_k \sim \pi(\cdot \mid o, g^*). \tag{8}$$

3. Compute advantages:

$$A(o, a, g^*) = Q(o, a, g^*) - V(o, g^*) \tag{9}$$

3.2 Contrastive Proximal Policy Optimisation

Now that we have defined a method to estimate advantages using contrastive learning, we incorporate this directly into the PPO algorithm in a straightforward way. Each training iteration consists of two phases:

1. **Encoder update**: update ϕ and ψ using the InfoNCE loss (Equation 3) on hindsight-relabeled data. Goals are relabeled using future observations from the same episode, with negative samples drawn from other episodes in the batch.
2. **Policy update**: compute contrastive advantages A using the method defined in Section 3.1, standardize them to zero mean and unit variance level then update π using the PPO clipped objective (Equation 4).

The complete CPPO algorithm is summarised in Algorithm 1 and additional implementation details can be found in Appendix 11.

3.3 Multi-Agent CPPO

Cooperative MARL is a setting where reward shaping is particularly fragile: dense team rewards must implicitly solve the credit assignment problem across agents, and small misspecifications can produce degenerate cooperative behaviour. This makes it a natural target for reward-free methods. ICRL (Nimonkar et al., 2025) was the first to demonstrate that contrastive RL extends to cooperative settings. It inherits the off-policy SAC backbone from single-agent CRL and relies on Gumbel-Softmax to handle discrete actions. We instead extend CPPO to the multi-agent case, yielding an on-policy contrastive MARL algorithm compatible with the IPPO/MAPPO training paradigm that is

widely adopted in the field (De Witt et al., 2020; Kuba et al., 2021; Wen et al., 2022; Yu et al., 2022; Mahjoub et al., 2024).

We adopt an IPPO-style independent learning formulation with full parameter sharing: all agents share the policy π_θ and the encoders ϕ_ξ, ψ_ω, with agent identity supplied via the observation. Following ICRL, all agents condition on a single shared goal g^*, and the InfoNCE loss draws negatives from other trajectories in the batch. We refer to the independent version of multi-agent CPPO as ICPPO. A centralised critic analogue in the spirit of MAPPO is a natural extension to consider for future work.

4 Experiments

4.1 Experimental design

Environments Our experiments include discrete and continuous, single- and multi-agent environments. Specifically, we evaluate CPPO/ICPPO in the following JAX-based environments: Navix (Pignatelli et al., 2024) {single-agent, discrete}, JaxGCRL suite (Bortkiewicz et al., 2024) {single-agent, continuous}, SMAX (Rutherford et al., 2024b) and Connector (Bonnet et al., 2023) {multi-agent, discrete} and JaxNav (Rutherford et al., 2024a) {multi-agent, continuous}. From these environment suites we test on a total of 18 tasks. The complete list of environments used and their descriptions are given in Appendix 7.

Baselines We compare against baselines from the literature including several existing off-policy contrastive RL methods, here referred to as: CSAC (Eysenbach et al., 2022) for single-agent continuous tasks, CDQN (Bastankhah et al., 2025) for single-agent discrete tasks and ICSAC (Nimonkar et al., 2025) for multi-agent tasks. We note an important change. As mentioned in Table 1, our naming differs from the original names given to these approaches. For example, Eysenbach et al. (2022) refer to their approach as CRL, while follow-up work by Bastankhah et al. (2025) propose a discrete version of the same approach but instead of following a SAC-style optimisation, they closely follow a DQN-style algorithm without providing an explicit name for it. Our aim with the above naming convention is to remove ambiguity in the naming by connecting the underlying base RL algorithm to the contrastive learning approach.

Evaluation protocol and hyperparameters Unless otherwise specified, each algorithm is trained for 10 independent trials per task with a fixed budget of 80 million environment steps. We evaluated at 80 evenly spaced intervals with 2048 episodes per evaluation, recording the mean win rate in line with the recommendations from Gorsane et al. (2022). For per-task results, we report the mean with 95% confidence intervals. For environment level aggregations, we report the min–max normalised inter-quartile mean (IQM) following Agarwal et al. (2021). All per-task results can be found in Appendix 9. Our evaluation aggregations, metric calculations, and plotting leverage the MARL-eval library (Gorsane et al., 2022).

We obtained our hyperparameters from prior work when available (Bortkiewicz et al., 2024; Mahjoub et al., 2024). When these were not available or the parameters provided performed poorly, we obtained them through a hyperparameter sweep using the Tree-structured Parzen Estimator (TPE) Bayesian optimization algorithm from the Optuna library (Akiba et al., 2019). For details on hyperparameters, we refer the reader to Appendix 10.

4.2 Empirical Results and Discussion

We organise our empirical investigation and discussion around four key questions:

1. How does our method compare to existing off-policy CRL?
2. Can CPPO compete with hand-crafted dense rewards?
3. How sensitive is reward design compared to goal specification?

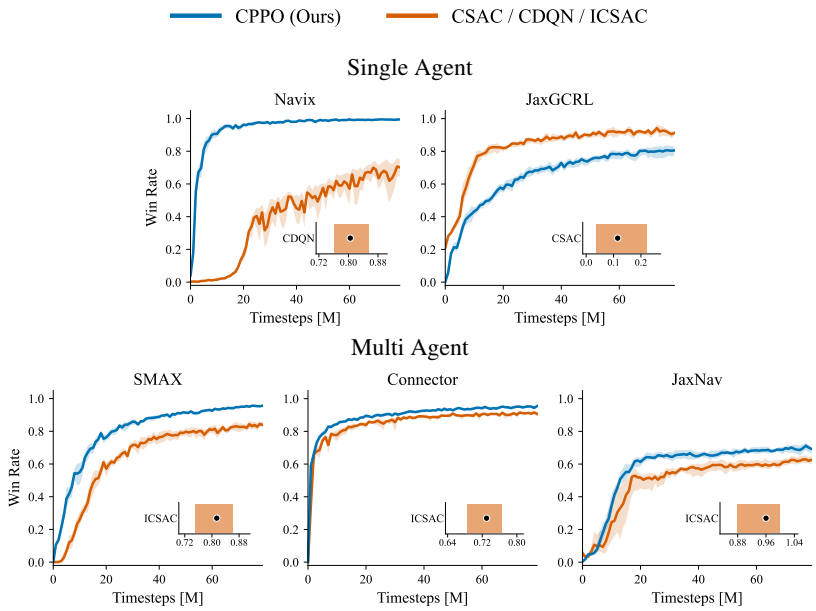

Figure 1: ***Per-environment IQM sample-efficiency curves*** (shaded 95% CI) with inset probability-of-improvement bars, i.e. $P(\text{CPPO} > \text{baseline})$. Results are aggregated over multiple tasks from each environment suite. CPPO achieves higher mean performance than CRL baselines in 4/5 environments.

4. Does CPPO's performance scale with environment complexity?

How does CPPO compare to existing off-policy CRL? Figure 1 and Table 2 show per-environment IQM sample-efficiency curves and win rates comparing CPPO/ICPPO against existing off-policy contrastive methods. Figure 1 insets also show the probability of improvement of CPPO/ICPPO compared to the baseline. Per-task breakdowns are provided in Appendix 9.

When comparing the off-policy and on-policy approaches, we find that CPPO significantly outperforms the existing CRL baselines in all settings except the single-agent continuous control benchmark. CPPO natively supports discrete actions, without requiring a Gumbel-Softmax approximation as in Nimonkar et al. (2025) or reverting to DQN as in Bastankhah et al. (2025). Gumbel-Softmax introduces a biased continuous relaxation of the discrete action distribution, while reverting to DQN forgoes a parameterised policy and the actor-critic structure that most of CRL has been built around. We hypothesise both workarounds weaken the resulting CRL baseline relative to CPPO's more natural fit for the discrete setting.

A natural explanation for the worse performance in single-agent continuous environments is variance in the Monte-Carlo value estimate required for continuous actions (Eq. 8); however, sweeping the number of value samples shows no effect on performance (Appendix 11.4), so value-estimate variance does not appear to be the cause. Our hypothesis is the baseline itself: CSAC builds on SAC, which consistently outperforms PPO on continuous control (Haarnoja et al., 2018a;b; Huang et al., 2024) and has been refined by a long line of work in this setting (Eysenbach et al., 2022; Zheng et al., 2023a; Bortkiewicz et al., 2024; Wang et al., 2025).

A question arises from this, if CPPO struggles in continuous control, why does it outperform ICSAC in JaxNav, a continuous action space, multi-agent environment? We hypothesise that using the PPO backbone, which is considered state-of-the-art in MARL (Mahjoub et al., 2024), helps CPPO perform well in these tasks. In summary, CPPO is the stronger contrastive RL method in the discrete and multi-agent settings, while off-policy contrastive RL retains an advantage in single-agent continuous control.

Table 2: ***Per-environment aggregate IQM scores*** (normalised win rate / success rate) with 95% stratified bootstrap confidence intervals in brackets.

				CPPO (Ours)	CSAC/CDQN/ICSAC
Navix	{4 tasks}	discrete	single-agent	**0.995** [0.993, 0.996]	0.702 [0.646, 0.749]
JaxGCRL	{3 tasks}	continuous	single-agent	0.806 [0.774, 0.832]	**0.916** [0.902, 0.930]
SMAX	{6 tasks}	discrete	multi-agent	**0.956** [0.951, 0.960]	0.840 [0.824, 0.859]
Connector	{4 tasks}	discrete	multi-agent	**0.955** [0.950, 0.961]	0.909 [0.899, 0.918]
JaxNav	{1 task}	continuous	multi-agent	**0.692** [0.672, 0.705]	0.626 [0.600, 0.653]

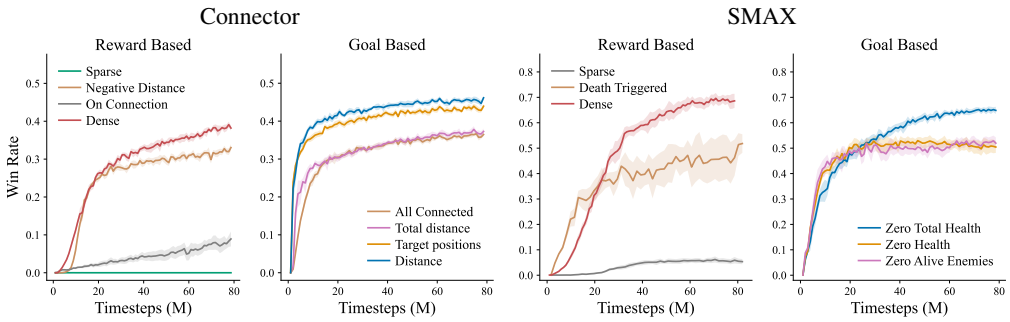

Figure 3: Reward- vs Goal-design sensitivity on Connector 10×10 and SMAX (smacv2_10_units). For each environment, the reward panel shows IPPO trained under several hand-crafted reward variants, and the goal panel shows ICPPO trained under different goal representations. The reward range far exceeds the goal spread on both environments.

Can CPPO compete with hand-crafted dense rewards? Prior contrastive RL methods inherit the goal-conditioned formulation in which the reward is sparse by construction (Equation 1). Consequently, most of the work in CRL benchmarks performance almost exclusively against sparse-reward baselines (Eysenbach et al., 2022; 2021; Zheng et al., 2023a; Bortkiewicz et al., 2024; Liu et al., 2024). This is fair given the problem formulation, but in many real-world settings, a sensible reward function might be available, or could be designed with effort. In such cases, a practitioner would want the method that performs the best, regardless of whether it is self-supervised or uses a dense reward signal. We therefore evaluate CPPO against the hand-crafted dense rewards specifically designed for each environment in our study as originally proposed. Figure 2 shows the aggregate comparison across discrete and continuous environments, respectively. In discrete environments, CPPO matches I/PPO on SMAX and Navix and exceeds it on Connector, all without any reward signal. The gap reverses in continuous environments (JaxNav and JaxGCRL) The gap reverses in the continuous environments (JaxNav and JaxGCRL), consistent with CPPO's weaker performance in continuous settings discussed above. This result is practically significant. It suggests that for discrete domains, practitioners could potentially bypass reward engineering entirely, a process that is brittle and environment-specific, with some confidence that they will not be sacrificing performance.

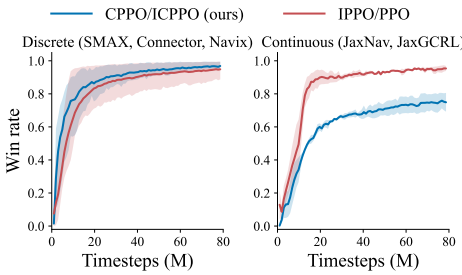

Figure 2: ***CPPO vs PPO with hand-crafted dense rewards***, aggregated across discrete and continuous domains. In discrete settings CPPO matches or exceeds IPPO/PPO; in continuous settings a gap remains.

How sensitive is reward design compared to goal design? In practice, both reward-based and contrastive methods require design. Useful reward functions often require careful shaping, and contrastive methods require a sensible goal state definition. To quantify the sensitivity of these methods to misspecification, we select the Connector 10×10 and Smax V2 10 agent tasks and train IPPO under seven different,

but all arguably sensible, reward functions, ranging from dense per step rewards to sparse rewards only provided on success. We run the same analysis for goal design and train ICPPO with seven goals of varying granularity from target positions to full connectivity. Full descriptions of each reward and goal are given in Appendix 8.

Figure 3 shows that the range from the best to worst reward is far greater than the range from the best to worst goal. It is important to note that we specifically included goals far more coarse than the default, which still perform relatively well. This means that goal specification is far more forgiving than reward design: even a coarse goal representation works reasonably well, whereas small changes to the reward function can cause large drops in performance.

Does CPPO's performance scale with environment complexity? We compare CPPO and IPPO on four Connector variants of increasing size and agent count: 5×5 (3 agents), 7×7 (5 agents), 10×10 (10 agents), and 15×15 (23 agents). Figure 4 shows the per-task learning curves. At the smallest scale (5×5) both methods converge to similar win rates ($\sim91\%$). As the grid grows, reward-based methods degrade faster: at 10×10, ICPPO reaches 46.2% vs. IPPO's 39.3%. At 15×15 the gap is most pronounced as ICPPO attains 10.7% while IPPO collapses to 0.6%.

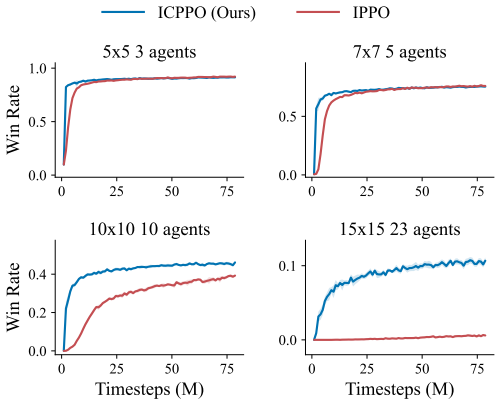

Figure 4: Connector scaling: per-task learning curves across four grid sizes. All methods are similar at 5×5; ICPPO's advantage widens as coordination complexity grows.

Two properties of the environment plausibly drive this divergence. First, optimal trajectories lengthen with grid size, so bootstrapped value targets must propagate across more steps thus accumulating more error; a contrastive objective supervises returns directly rather than chaining through bootstrapped estimates, sidestepping this compounding. Second, successful connections become rarer as the task grows harder, so the signal a critic regresses against when following a mean squared error objective becomes increasingly sparse and dominated by near-zero returns; a contrastive critic only needs to distinguish better trajectories from worse ones and remains informative even when absolute reward magnitudes carry little signal. These properties are not unique to Connector and suggest that the contrastive critic provides a more useful learning signal in complex environments, even when MSE-based value regression suffices at small scale.

5 Related Work

Contrastive reinforcement learning. Goal-conditioned RL has historically relied on hindsight relabeling and incentivising goal-reaching behaviour via sparse rewards (Kaelbling, 1993; Andrychowicz et al., 2017; Sun et al., 2019; Chane-Sane et al., 2021; Abramowitz & Nitschke, 2022). Eysenbach et al. (2022) showed that contrastive learning on action-labled trajectories yields goal-conditioned Q-functions, unifying CRL and GCRL. Recent work has addressed practical challenges by introducing JAX-based implementations of GCRL benchmarking environments and algorithms (Bortkiewicz et al., 2024), scaled to large networks (Wang et al., 2025), introduced offline variants (Zheng et al., 2023a; Park et al., 2024), added TD bootstrapping (Zheng et al., 2023b), demonstrated emergent exploration (Bastankhah et al., 2025) and extended algorithms to the coooperative multi-agent case (Nimonkar et al., 2025). All contrastive methods rely on off-policy optimisation.

On-policy policy gradient methods. PPO (Schulman et al., 2017), a practical trust region method and clipped-surrogate successor to TRPO (Schulman et al., 2015a), is a widely used on-policy

algorithm in modern RL. It improves over actor-critic methods like A2C (Mnih et al., 2016) and uses generalised advantage estimation (Schulman et al., 2015b) to balance the bias variance trade-off. It has become even more popular due to massively parallel simulators (Makoviychuk et al., 2021; Lange, 2022; Bonnet et al., 2023) where high throughput outweighs the sample efficiency of off-policy methods. Furthermore trust region methods remain a target for theoretical (Grudzien et al., 2022) and algorithmic (Lu et al., 2022) refinement. Yet existing contrastive RL methods are off-policy (Eysenbach et al., 2022; Bastankhah et al., 2025; Wang et al., 2025). CPPO closes this gap.

Cooperative multi-agent RL. Cooperative MARL is broadly split into two paradigms: independent learning, where each agent independently optimises its own policy from local observations (De Witt et al., 2020), and centralised training with decentralised execution (Oliehoek et al., 2008), which exploits global information at training time (Lowe et al., 2017; Sunehag et al., 2017; Rashid et al., 2020; Yu et al., 2022). Heterogeneous-agent variants (Kuba et al., 2021) introduce monotonic improvement guarantees and are extended by sequence-modeling approaches (Wen et al., 2022; Mahjoub et al., 2024). All rely on hand-designed dense team rewards. ICRL (Nimonkar et al., 2025), the sole multi-agent contrastive extension builds on SAC, which is at odds with cooperative MARL, where on-policy methods are typically preferred (Yu et al., 2022). ICPPO is its on-policy counterpart.

6 Conclusion

We introduced CPPO, an on-policy contrastive reinforcement learning algorithm that derives advantages directly from contrastive Q-values and optimises them with PPO's clipped surrogate objective. The result is a single self-supervised algorithmic template that operates across discrete and continuous action spaces and across single- and multi-agent settings, integrating naturally into the on-policy pipelines widely used in modern industrial RL applications.

Empirically, CPPO matches or exceeds reward-engineered PPO and IPPO on every discrete benchmark we evaluate, despite using no reward signal, and surpasses prior contrastive baselines in every setting except single-agent continuous control. Our analysis further illustrates how goal design can be markedly more forgiving than reward design and that CPPO's advantage over reward-based baselines widens as environment complexity grows.

Limitations and future work A clear limitation remains that in single-agent continuous control, CPPO trails off-policy SAC-based contrastive methods. We initially hypothesised that this gap stems from variance in the Monte-Carlo state-value estimate required for continuous actions. However, sweeping the number of value samples shows no effect on performance, indicating variance is not the primary cause. Identifying and developing methods to fix this gap is a natural direction for future work. While the multi-agent ICPPO performs remarkably well, a centralised training with decentralised execution version is another promising avenue for further investigation.

References

Sasha Abramowitz and Geoff Nitschke. Scalable evolutionary hierarchical reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '22, pp. 272–275, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392686. DOI: 10.1145/3520304.3528937. URL https://doi.org/10.1145/3520304.3528937.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

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

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Mahsa Bastankhah, Grace Liu, Dilip Arumugam, Thomas L Griffiths, and Benjamin Eysenbach. Demystifying the mechanisms behind emergent exploration in goal-conditioned rl. *arXiv preprint arXiv:2510.14129*, 2025.

Clément Bonnet, Daniel Luo, Donal Byrne, Shikha Surana, Sasha Abramowitz, Paul Duckworth, Vincent Coyette, Laurence I Midgley, Elshadai Tegegn, Tristan Kalloniatis, et al. Jumanji: a diverse suite of scalable reinforcement learning environments in jax. *arXiv preprint arXiv:2306.09884*, 2023.

Michał Bortkiewicz, Władysław Pałucki, Vivek Myers, Tadeusz Dziarmaga, Tomasz Arczewski, Łukasz Kuciński, and Benjamin Eysenbach. Accelerating goal-conditioned rl algorithms and research. *arXiv preprint arXiv:2408.11052*, 2024.

Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In *International conference on machine learning*, pp. 1430–1440. PMLR, 2021.

Ruan de Kock, Omayma Mahjoub, Sasha Abramowitz, Wiem Khlifi, Callum Rhys Tilbury, Claude Formanek, Andries P. Smit, and Arnu Pretorius. Mava: a research library for distributed multi-agent reinforcement learning in jax. *arXiv preprint arXiv:2107.01460*, 2023. URL https://arxiv.org/pdf/2107.01460.pdf.

Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36:37567–37593, 2023.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tc5qisoB-C.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

Rihab Gorsane, Omayma Mahjoub, Ruan de Kock, Roland Dubb, Siddarth Singh, and Arnu Pretorius. Towards a standardised performance evaluation protocol for cooperative marl, 2022. URL https://arxiv.org/abs/2209.10485.

Jakub Grudzien, Christian A Schroeder De Witt, and Jakob Foerster. Mirror learning: A unifying framework of policy optimisation. In *International Conference on Machine Learning*, pp. 7825–7844. PMLR, 2022.

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*, 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, et al. Soft actor-critic algorithms and applications. arxiv 2018. *arXiv preprint arXiv:1812.05905*, 2018b.

Shengyi Huang, Quentin Gallouédec, Florian Felten, Antonin Raffin, Rousslan Fernand Julien Dossa, Yanxiao Zhao, Ryan Sullivan, Viktor Makoviychuk, Denys Makoviichuk, Mohamad H. Danesh, Cyril Roumégous, Jiayi Weng, Chufan Chen, Md Masudur Rahman, João G. M. Araújo, Guorui Quan, Daniel Tan, Timo Klein, Rujikorn Charakorn, Mark Towers, Yann Berthelot, Kinal Mehta, Dipam Chakraborty, Arjun KG, Valentin Charraut, Chang Ye, Zichen Liu, Lucas N. Alegre, Alexander Nikulin, Xiao Hu, Tianlin Liu, Jongwook Choi, and Brent Yi. Open RL Benchmark: Comprehensive Tracked Experiments for Reinforcement Learning. *arXiv preprint arXiv:2402.03046*, 2024. URL https://arxiv.org/abs/2402.03046.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence*, volume 2, pp. 1094–8, 1993. URL https://api.semanticscholar.org/CorpusID:5538688.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. *arXiv preprint arXiv:2109.11251*, 2021.

Robert Tjarko Lange. gymnax: A JAX-based reinforcement learning environment library, 2022. URL http://github.com/RobertTLange/gymnax.

Grace Liu, Michael Tang, and Benjamin Eysenbach. A single goal is all you need: Skills and exploration emerge from contrastive rl without rewards, demonstrations, or subgoals. *arXiv preprint arXiv:2408.05804*, 2024.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468, 2022.

Omayma Mahjoub, Sasha Abramowitz, Ruan de Kock, Wiem Khlifi, Simon du Toit, Jemma Daniel, Louay Ben Nessir, Louise Beyers, Claude Formanek, Liam Clark, et al. Sable: a performant, efficient and scalable sequence model for marl. *arXiv preprint arXiv:2410.01706*, 2024.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Chirayu Nimonkar, Shlok Shah, Catherine Ji, and Benjamin Eysenbach. Self-supervised goal-reaching results in multi-agent cooperation and exploration. *arXiv preprint arXiv:2509.10656*, 2025.

Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. Springer, 2016.

Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. *arXiv preprint arXiv:2201.03544*, 2022.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. *arXiv preprint arXiv:2410.20092*, 2024.

Eduardo Pignatelli, Jarek Liesen, Robert Tjarko Lange, Chris Lu, Pablo Samuel Castro, and Laura Toni. Navix: Scaling minigrid environments with jax. *arXiv preprint arXiv:2407.19396*, 2024.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on robot learning*, pp. 91–100. PMLR, 2022.

Alex Rutherford, Michael Beukman, Timon Willi, Bruno Lacerda, Nick Hawes, and Jakob Foerster. No regrets: Investigating and improving regret approximations for curriculum discovery. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 16071–16101. Curran Associates, Inc., 2024a. DOI: 10.52202/079017-0512. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/1d0ed12c3fda52f2c241a0cebcf739a6-Paper-Conference.pdf.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian S de Witt, et al. Jaxmarl: Multi-agent rl environments and algorithms in jax. *Advances in Neural Information Processing Systems*, 37: 50925–50951, 2024b.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schaul15.html.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 9460–9471. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/3d719fee332caa23d5038b8a90e81796-Paper-Conference.pdf.

Hao Sun, Zhizhong Li, Xiaotong Liu, Bolei Zhou, and Dahua Lin. Policy continuation with hindsight inverse dynamics. *Advances in Neural Information Processing Systems*, 32, 2019.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

Kevin Wang, Ishaan Javali, MichaĹ Bortkiewicz, Benjamin Eysenbach, et al. 1000 layer networks for self-supervised rl: Scaling depth can enable new goal-reaching capabilities. *arXiv preprint arXiv:2503.14858*, 2025.

Muning Wen, Jakub Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. *Advances in Neural Information Processing Systems*, 35:16509–16521, 2022.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

Chongyi Zheng, Benjamin Eysenbach, Homer Walke, Patrick Yin, Kuan Fang, Ruslan Salakhutdinov, and Sergey Levine. Stabilizing contrastive rl: Techniques for robotic goal reaching from offline data. *arXiv preprint arXiv:2306.03346*, 2023a.

Chongyi Zheng, Ruslan Salakhutdinov, and Benjamin Eysenbach. Contrastive difference predictive coding. *arXiv preprint arXiv:2310.20141*, 2023b.

Supplementary Materials

The following content was not necessarily subject to peer review.

7 Environment Details

We provide a short conceptual description of each environment. Implementation details (exact observation layouts, hyperparameters, code-level flags) are available in the original references. Reward and goal variants used in our ablations or not mentioned here are described separately in Appendix 8.

7.1 SMAX

SMAX (Rutherford et al., 2024b) is a JAX-reimplementation of the StarCraft Multi-Agent Challenge, with SMACv2 scenarios that randomise unit composition and starting positions (Ellis et al., 2023). Teams of allied units must coordinate to defeat an enemy team controlled by a hand-coded heuristic that attacks the nearest visible opponent.

Observation space. Each agent receives a partial observation limited by a unit-type-dependent sight range, encoding its own state and the relative state (position, health, unit type) of allied and enemy units within range. A binary mask indicates which actions are currently legal.

Action space. Discrete: four cardinal movement directions, stop, no-op, and one "attack enemy i" action per enemy unit.

Scenarios. We evaluate on six scenarios: **3m** and **8m** (symmetric homogeneous matchups), **5m_vs_6m** (asymmetric, requires focus-fire), **6h_vs_8z** (heterogeneous, requires kiting), and the SMACv2 scenarios **SMACv2 5 units** and **SMACv2 10 units**, which randomise unit types and starting positions each episode.

Reward. The default dense reward equally incentivises tactical engagement and overall victory: agents earn 50% of their return from per-step damage events (damage dealt minus damage received) and 50% from winning the episode.

Goal definition. The goal is the elimination of the enemy team. Following Nimonkar et al. (2025), ICPPO conditions on a goal corresponding to this end-state, a scalar representing the (normalised) total enemy health, with target value zero (all enemies eliminated).

7.2 Connector

The Vector Connector environment (Bonnet et al., 2023) is a cooperative multi-agent grid world in which each agent must trace a connected path from its designated start cell to its target cell without overlapping the paths of other agents.

Observation space. Each agent observes its own position and target, an egocentric local view of nearby agents and their trail cells, and an egocentric view of all agents' targets.

Action space. Five discrete actions: move in one of the four cardinal directions, or stay in place.

Tasks. We evaluate on four configurations: 5×5 grid with 3 agents, 7×7 with 5 agents, 10×10 with 10 agents, and 15×15 with 23 agents. The episode horizon scales with grid size (set to the number of grid cells, capped at $T{=}225$ for the largest).

Reward. The default reward grants $+1$ when an agent connects its endpoints and applies a -0.03 per-step penalty otherwise. An episode is "won" when every agent has connected.

Goal definition. The goal is for each agent to reach its assigned target cell, completing its individual connection. ICPPO/ICRL uses a default of per-agent normalised Manhattan distance from current to target position, with target value zero (agent at its endpoint).

7.3 JaxNav

JaxNav (Rutherford et al., 2024a;b) is a JAX-native 2D continuous-space navigation environment for differential-drive robots. Each robot must reach an individual goal position on a randomly generated cluttered map while avoiding walls and the other robots.

Observation space. Continuous. Each robot observes a vector of LiDAR range readings sampled over a 360° arc, the polar coordinates of its goal relative to its current pose, and its own linear and angular velocity.

Action space. Continuous and two-dimensional: a target linear velocity and target angular velocity, integrated through differential-drive kinematics.

Reward. The default reward combines a goal-arrival bonus, a distance-shaping term that rewards moving closer to the goal, a collision penalty (walls or other robots), a proximity penalty for near-collisions, and a small per-step time penalty. See Rutherford et al. (2024a) for the full equations.

Goal definition. The goal is for the robot to arrive at its assigned target location without colliding with obstacles or other robots. Our contrastive experiments use the Euclidean distance from the agent's current position to its goal, with a target value of zero (agent at its goal).

7.4 JaxGCRL

The JaxGCRL suite (Bortkiewicz et al., 2024) is a JAX-native benchmark of single-agent continuous-control goal-reaching tasks built on the Brax physics engine. We evaluate on three tasks: *Reacher*, *Ant*, and *Ant U-Maze*. We refer the reader to Bortkiewicz et al. (2024) for full implementation details and hyperparameter defaults.

Reacher. A planar two-link arm whose end-effector must reach a randomly sampled target position. Observations are continuous and include the joint angles, end-effector position, and end-effector linear velocity; actions are continuous joint torques. A run is successful when the end-effector is within a small radius of the target. The default reward is the negative Euclidean distance from end-effector to target.

Ant. A quadruped (eight hinge actuators) that must reach a randomly sampled goal position on a flat plane. Observations include the torso pose, joint angles, and their velocities; actions are continuous joint torques. Goals are sampled at a fixed distance from the spawn with a uniformly random angle, and a run succeeds when the torso is within 0.5 units of the goal. The default reward combines a velocity-toward-goal term, a healthy-pose bonus, and a control penalty.

Ant U-Maze. The same Ant morphology placed inside a U-shaped corridor on a 5×5 cell grid. The agent spawns at a fixed reset cell and must reach a goal cell sampled uniformly from the six valid goal cells. The shortest path requires navigating *around* the central wall, making straight-line heuristics misleading. Observation, action, success criterion, and reward formulation are identical to the flat-ground Ant task.

Goal definition. The goal in each task is the target end-state: in Reacher, the end-effector positioned at the sampled target; in Ant, the torso positioned at the sampled goal location; in Ant U-Maze, the torso at one of possible goal cells inside the U-shaped maze, sampled uniformly each episode. CPPO/CSAC conditions on these end-states, a slice of the observation corresponding to the relevant body part position (the end-effector for Reacher, the root (x, y) for Ant and U-Maze), with target value equal to the sampled goal.

7.5 Navix

Navix (Pignatelli et al., 2024) is a JAX-native reimplementation of the MiniGrid grid-world suite. An agent occupies a single cell on a discrete grid and must navigate to a designated goal cell, optionally completing intermediate sub-goals such as picking up a key and toggling a door.

Observation space. Under partial observability the agent receives a $7\times7\times3$ egocentric crop aligned with its facing direction, where the three channels follow MiniGrid's symbolic encoding (object tag, colour, and state) per cell.

Action space. Seven discrete actions following the MiniGrid default set: rotate counter-clockwise, rotate clockwise, move forward, pick up, drop, toggle, and `done`.

Tasks. We evaluate on four scenarios: **Empty-16×16** (empty room, fixed start/goal), **Empty-Random-16×16** (empty room, randomised start and goal each episode), **DoorKey-Random-16×16** (locked door with randomised key/door/start/goal positions; the agent must pick up the key, toggle the door open, and reach the goal), and **FourRooms** (four-room layout with narrow doorways; the agent must traverse multiple rooms to reach a randomised goal).

Reward. Navix's default reward is sparse: $+1$ on reaching the goal, 0 otherwise. Our PPO baseline uses a lightly-shaped composite reward built from Navix primitives: $+1$ on reaching the goal, -0.01 per environment step, and -0.01 for each wall collision. The shaping terms are two orders of magnitude smaller than the goal bonus, so the signal remains close to sparse; in our experiments, PPO achieves the same final performance under the purely sparse variant.

Goal definition. The goal is for the agent to reach its assigned goal cell. On Empty and FourRooms, we encode it as the goal cell's (y, x) grid coordinates (2D); the contrastive Q-function learns to encode integer grid positions directly. On DoorKey the agent must first pick up a key and toggle a door open before reaching the goal; we encode it as a 1D scalar progress signal in $[0, 1]$ that combines the key-to-door and player-to-goal Manhattan distances, with target value one (full progress).

8 Goal and Reward Designs

This appendix lists the goal and reward variants used in our goal- and reward-design ablations (Section 4.2). We tested variants on Connector and SMAX; the default goals and rewards for the other environments are described in Appendix 7.

8.1 Connector

Reward variants. We compare four reward functions on Connector spanning a range of informativeness. *Dense* and *Negative Distance* provide a per-step gradient that pushes each agent toward its target. *On Connection* fires a one-shot $+1$ on the step an agent connects. *Sparse* is the strictest: it fires only when every agent is simultaneously connected.

Table 3: Connector reward variants used in the reward sensitivity ablation. Per-agent rewards are summed across agents and broadcast as the team total, as is the default in Mava.

Name	Per-agent reward
Dense	$+1$ on the step the agent connects; -0.03 on every step it remains unconnected.
Negative Distance	Negative Manhattan distance from the agent's position to its target.
On Connection	$+1$ on the single step the agent connects; zero otherwise.
Sparse	$+1$ to every agent only when all agents are simultaneously connected; zero at all other steps.

Goal variants. We compare four goal representations on Connector spanning a range of granularity. *Distance* (default) gives each agent its own progress signal toward its individual target. *Total distance* collapses this to a single team-mean signal broadcast to every agent. *All Connected* reports only how many agents have reached their endpoints. *Target positions* uses the target cell's (x, y) coordinates as the goal vector, with the agent's own position as the achieved state.

Table 4: Connector goal variants used in the goal sensitivity ablation.

Name	Description
Distance (default)	Per-agent normalised Manhattan distance to target; target value zero (agent at its endpoint).
Total distance	Team-mean of the per-agent distances, broadcast to every agent; target value zero.
All Connected	Fraction of agents currently connected, broadcast to every agent; target value one.
Target positions	Per-agent: (x, y) position as achieved state, target (x, y) as goal vector.

8.2 SMAX

Reward variants. We compare three reward functions on SMAX spanning a range of informativeness. All three share the same terminal win/loss bonus; they differ only in the per-step shaping signal. SMAX's default *Dense* reward (Rutherford et al., 2024b) provides per-step shaping proportional to the normalised damage dealt to enemies. Our *Death Triggered* variant replaces this continuous damage signal with a discrete bonus that fires only when an enemy is killed. This is a sub-component

analogous to the `reward_death_value` term in the original SMAC (Ellis et al., 2023). *Sparse* drops both shaping terms entirely, leaving only the terminal win/loss bonus.

Table 5: SMAX reward variants used in the reward sensitivity ablation. All variants additionally include a terminal win/loss bonus.

Name	**Per-step shaping**
Dense (default)	Sum of normalised damage dealt to enemies at each step, $+1$ if episode won.
Death Triggered	Bonus of $R_{\text{win}}/num_enemies$ on each step an enemy is killed, $+1$ if episode won; zero per-step otherwise.
Sparse	No per-step shaping; reward only at the terminal win/loss event.

Goal variants. We compare three goal representations on SMAX, all formalising the same end-state (defeat the enemy team) at different granularities. *Zero Total Health* (default) follows Nimonkar et al. (2025) and uses a single scalar that aggregates the normalised health of all enemies. *Zero Health* replaces this with a per-enemy vector, one entry per enemy unit. *Zero Alive Enemies* collapses the signal to a single team-level fraction over currently-observable enemies. All three rely on per-agent visibility: enemies that are not currently observed are assumed to be at full health (i.e., still alive).

Table 6: SMAX goal variants used in the goal sensitivity ablation.

Name	**Description**
Zero Total Health (default)	Scalar sum of normalised enemy healths; target value zero (all enemies eliminated).
Zero Health	Per-enemy vector of normalised damage values (one entry per enemy); target corresponds to all enemies eliminated.
Zero Alive Enemies	Scalar fraction of currently-visible enemies still alive; target value zero (no alive enemies in view).

9 Benchmark

The body of the paper reports per-environment IQM aggregates (Figure 1 and Table 2), here we present per-task learning curves below. Each panel shows the per-task IQM (mean of the middle 50% of seeds at every evaluation step) over 10 seeds for the three method families: CPPO/ICPPO (ours, blue), the on-policy dense-reward baseline (PPO/IPPO, red), and the off-policy contrastive baseline of Eysenbach et al. (2022); Bastankhah et al. (2025); Nimonkar et al. (2025) (CSAC/CDQN/ICSAC, orange). Shaded bands are 95% bootstrap confidence intervals over seeds. Figure 5 provides a single-page grid of all 18 tasks for at-a-glance comparison.

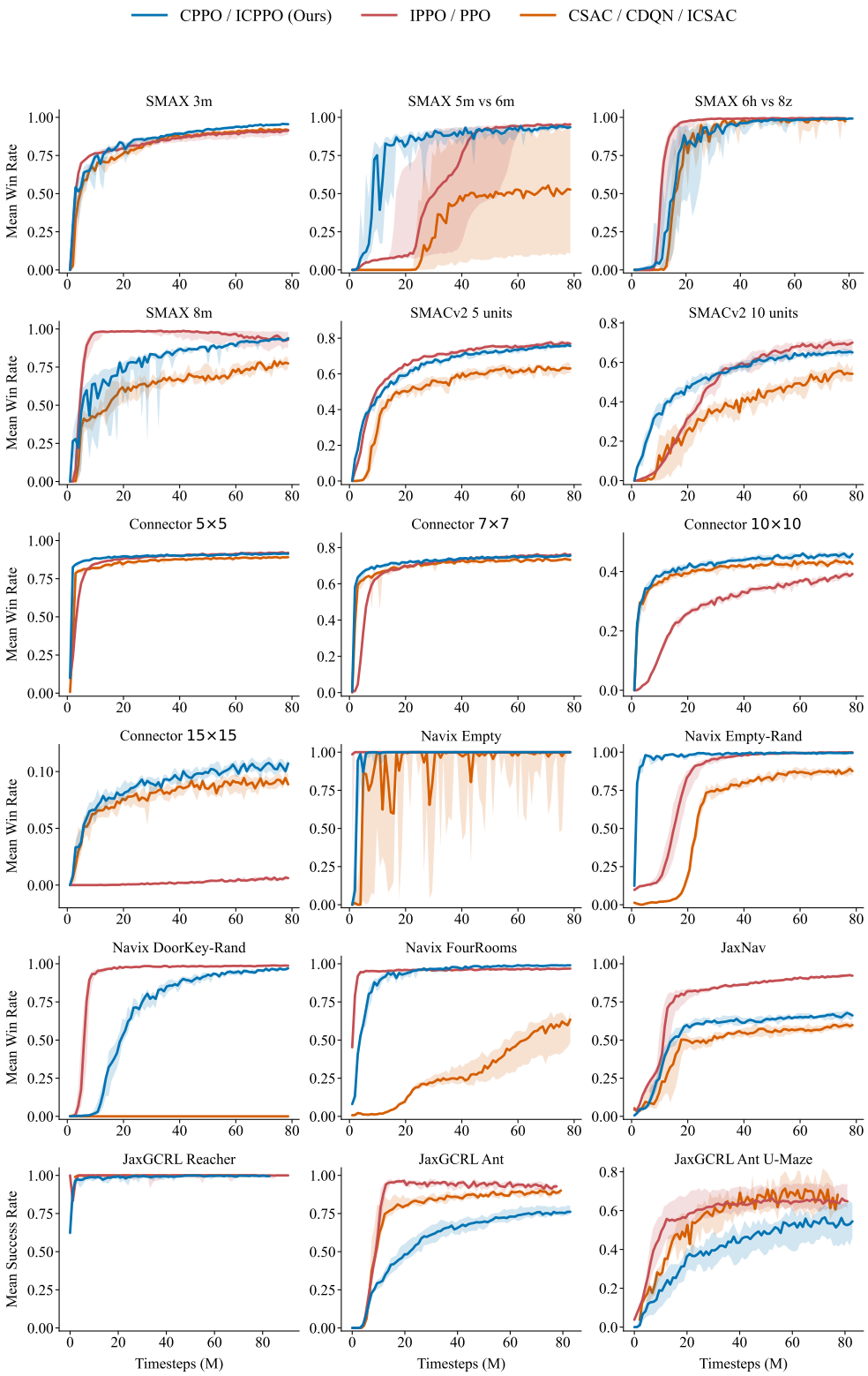

Figure 5: Mean Win Rate / Success Rate with 95% bootstrap confidence intervals on all tasks.

10 Hyperparameters

This appendix documents the hyperparameters used in our experiments. For our method (CPPO/ICPPO), we report the values that we held fixed across all tasks. Table 7 lists the architectural, contrastive, and training-scale defaults that apply to every one of the 18 tasks; Table 8 lists the optimisation knobs whose recommended value depends on the action space (discrete vs. continuous). For the baselines, we adopt prior-work configurations when available and otherwise re-tune via a per-task sweep, as detailed in Section 10.1. To ensure a fair comparison, the training-scale settings of Table 7 (rollout length, batch size, parallel environments, total number of updates, and evaluation budget) are matched across all algorithms, including baselines. All experiments were run on NVIDIA H100 GPUs with 8 CPU cores allocated per job. A single training seed of a single task took between roughly 10 minutes (smallest tasks, e.g. SMAX 3m or Connector 5×5) and 2 hours (largest tasks, e.g. Connector 15×15 or SMACv2 10 units) to complete the full 80M-step training budget. The remaining per-task tuned values for our method, the full baseline configurations, and all sweep specifications are released on the project website[2].

Table 7: **Default hyperparameters for CPPO/ICPPO**

Hyperparameter	**Value**
Network	
Hidden layer sizes	[512, 512, 512, 512]
Activation	swish
Layer normalisation	true
Contrastive	
Contrastive objective	forward InfoNCE
Energy function	L2
Representation dimension	64
Training scale	
Rollout length	128
Batch size	256
Number of updates	1 250
Number of evaluations	80
Parallel environments	512
Evaluation episodes per evaluation	2 048

Table 8: **Action-space-dependent default hyperparameters for CPPO/ICPPO.** The discrete column covers SMAX (6), Connector (4) and Navix (4); the continuous column covers JaxNav (1) and JaxGCRL (3). Bracketed entries in the continuous column give the $[\min, \max]$ range of selected values across the four tasks; the exact per-task value is released on the project website.

Hyperparameter	**Discrete (14 tasks)**	**Continuous (4 tasks)**
Gamma (γ)	0.99	$[0.99,\ 0.9999]$
PPO clip (ϵ)	0.2	$[0.12,\ 0.20]$
PPO epochs	1	$\{1,\ 8\}$
Actor LR	2.5×10^{-4}	$[1.7\times10^{-5},\ 3\times10^{-4}]$
Q LR	2.5×10^{-4}	$[3.9\times10^{-5},\ 3\times10^{-4}]$
Max gradient norm	0.5	$[0.5,\ 5]$
LR decay schedule	cosine	none,linear
LR end (final value)	1×10^{-7}	$[1\times10^{-7},\ 1\times10^{-6}]$

[2] https://sites.google.com/view/contrastive-ppo/home

10.1 Baseline hyperparameters

We adopt baseline hyperparameters from prior work whenever a reference configuration is published for the corresponding task, and re-tune via a per-task sweep otherwise, either because no reference exists or because the published values failed to reproduce the performance reported in the original work on our benchmark. All sweeps use the Tree-structured Parzen Estimator (TPE) Bayesian optimisation algorithm from Optuna (Akiba et al., 2019).

We adopt published configurations directly for the following baseline–environment pairs: IPPO on SMAX and Connector uses hyperparameters published in Mahjoub et al. (2024); IPPO on JaxNav uses the reference configuration released with the JaxNav codebase (Rutherford et al., 2024a). PPO and CSAC on JaxGCRL inherit the canonical Brax recipes from Bortkiewicz et al. (2024); and ICSAC on SMAX uses the hyperparameters recommended by Nimonkar et al. (2025) and reproduced results consistent with their released codebase

The remaining baseline-environment pairs are tuned via TPE sweep. PPO and CDQN on Navix are tuned from scratch, and ICSAC on Connector and JaxNav is tuned on top of the Nimonkar et al. (2025) codebase using their published values as the starting point.

11 CPPO Implementation Details

This appendix expands on CPPO's advantage estimation, its normalisation, and the goal representation. It complements the method (Section 3) and the goal and reward descriptions (Appendix 8).

11.1 Advantage Estimation and Why GAE Is Not Required

Recall from Section 3.1 that CPPO forms advantages directly from the contrastive critic as

$$A(o, a, g^*) = Q(o, a, g^*) - V(o, g^*), \qquad V(o, g^*) = \mathbb{E}_{a\sim\pi(\cdot|o,g^*)}[Q(o, a, g^*)]\,,$$

with $Q(o, a, g^*) = -\|\phi(o, a) - \psi(g^*)\|_2$. For discrete actions, V is computed exactly as a policy-weighted sum over actions (Eq. 7); for continuous actions it is estimated by Monte-Carlo sampling of K actions from the policy (Eq. 8).

Standard PPO instead relies on Generalised Advantage Estimation (GAE) (Schulman et al., 2015b), which builds the advantage from reward temporal-difference (TD) residuals $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ and a learned value function. Two properties of our setting make GAE unnecessary. First, CPPO is reward-free, and with no reward signal the TD residuals δ_t are undefined, so GAE does not directly apply. Second, the contrastive critic provides the action-value Q directly, so the advantage follows in closed form from its definition, evaluated entirely at the current state o. The long-horizon information that GAE reconstructs from rewards is instead learned by the critic during training: hindsight relabelling draws positive goals from future observations along the trajectory, so the contrastive Q approximates the discounted probability of reaching a goal in the future (Eysenbach et al., 2022) and encodes long-horizon reachability directly, rather than requiring it to be reconstructed in the advantage estimator.

11.2 Advantage Normalisation

Before the clipped objective (Eq. 4), advantages are standardised to zero mean and unit variance at the mini-batch level, $\hat{A} = (A - \mathrm{mean}(A))/\mathrm{std}(A)$. This is standard practice in PPO and matters in our setting as the contrastive advantage $A = Q - V$ has a different natural scale from a reward-based advantage, and standardisation places it on the scale for which PPO's clipping coefficient is calibrated.

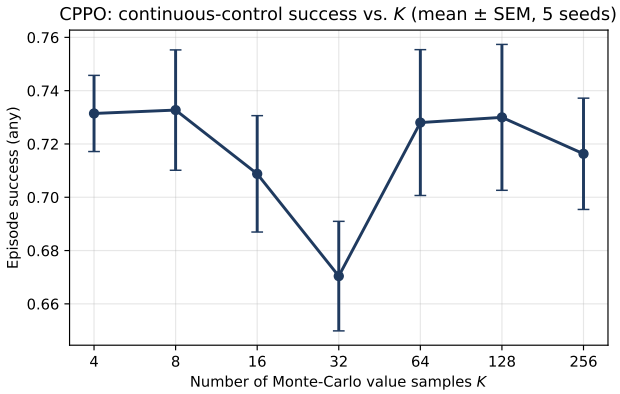

Figure 6: Continuous-control success (episode success, any) versus the number of Monte-Carlo value samples K (mean $\pm$ SEM over 5 seeds). Performance across different K values indicates that value-estimate variance is not the main driver of the continuous-control gap.

11.3 Goal Representation and Goal Dimensionality

CPPO conditions on low-dimensional goals derived from the state, for example an agent's position, its distance to a target, or a scalar such as remaining enemy health rather than on full observations, the exact per-environment goal definitions are given in Appendix 8. We find that goal dimensionality matters as overly specific or high-dimensional goals like the full state lead to high contrastive accuracy but a policy that fails to learn the task, whereas low-dimensional goals train successfully.

Learning from high-dimensional or full-state goals is an open challenge for contrastive RL in general, and is not specific to CPPO. To our knowledge, existing CRL methods likewise rely on low dimensional goal representations or state slices (Eysenbach et al., 2022; Bortkiewicz et al., 2024).

11.4 Value Estimation and the Continuous-Control Gap

A natural hypothesis for CPPO's weaker continuous-control performance is variance in the Monte-Carlo estimate of V (Eq. 8). To test this, we swept the number of value samples $K \in \{4, 8, 16, 32, 64, 128, 256\}$ (5 seeds each). As shown in Figure 6, success does not improve with larger K, remaining within ≈ 0.67–0.73 across two orders of magnitude, with the smallest K competitive with the largest. This indicates that value-estimate variance is not the primary cause of the gap; identifying the underlying cause is left to future work.

The full learning curves tell the same story (Figure 7): across K they overlap throughout training, so no value of K learns faster or reaches a higher final success.

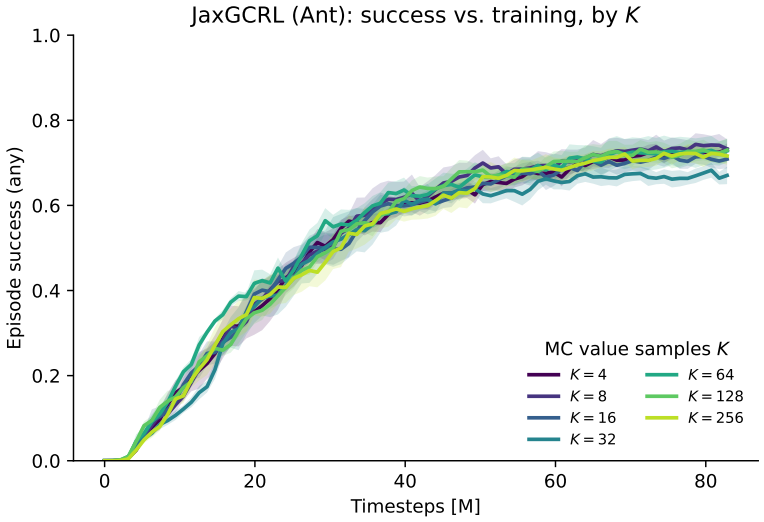

Figure 7: JaxGCRL (Ant) success over training for each number of Monte-Carlo value samples K (mean $\pm$ SEM over 5 seeds). The curves overlap throughout training, further indicating that value-estimate variance does not drive the continuous-control gap.