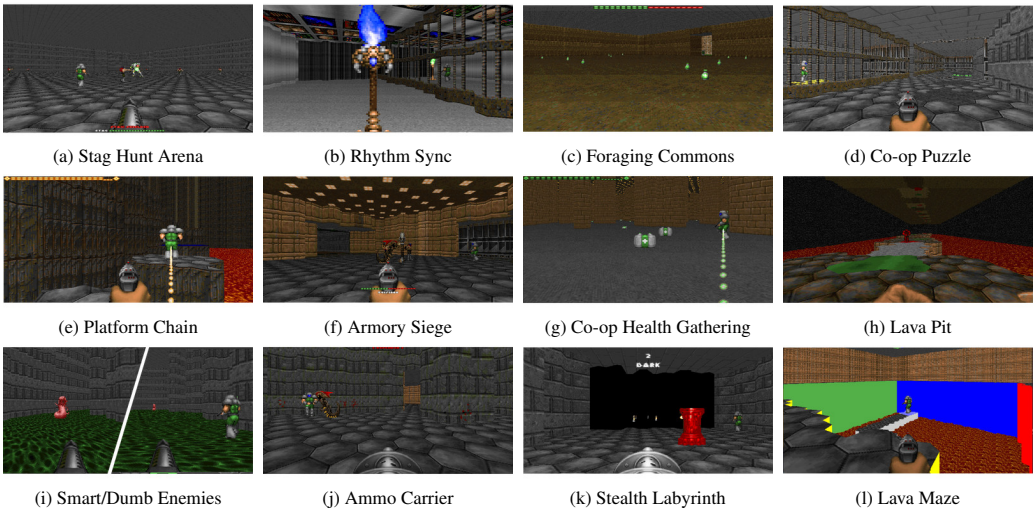
COMRAD: A BENCHMARK FOR EMBODIED MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Benchmarks are central to the development and evaluation of multi-agent reinforcement learning (MARL) algorithms. As the cooperative MARL community has grown, two categories of evaluation environments have proven indispensable: low-dimensional feature-vector benchmarks that isolate algorithmic behavior in compact state spaces, and two-dimensional pixel-based benchmarks that rely on overhead visual observations. Embodied cooperation, however, requires high-dimensional 3D egocentric perception, partial observability arising from line-of-sight occlusion, and the ability to close the gap between recognizing a coordination pattern and executing it from a first-person view. While various 3D environments have been explored for vision-based RL, no existing platform simultaneously provides a standardized, purely cooperative multi-agent benchmark with 3D first-person observations and high-throughput simulation. To address this gap, we introduce **COMRAD**, a **CO**operative **M**ulti-Agent **R**einforcement Le**A**rning benchmark suite in **D**oom, featuring a diverse set of challenging scenarios spanning role asymmetry, temporal synchronization, and spatial navigation. To introduce within-scenario variability, we develop **DoomGen**, a procedural map generator that produces diverse layout configurations for every scenario. We integrate COMRAD with Sample Factory, a high-throughput asynchronous RL framework, and implement seven MARL baselines on top of it, reaching $\sim$25K frames per second during training. Our experiments show that COMRAD poses significant challenges for current CTDE methods, establishing visual cooperative MARL as an important open frontier.



(a) Stag Hunt Arena (b) Rhythm Sync (c) Foraging Commons (d) Co-op Puzzle

(e) Platform Chain (f) Armory Siege (g) Co-op Health Gathering (h) Lava Pit

(i) Smart/Dumb Enemies (j) Ammo Carrier (k) Stealth Labyrinth (l) Lava Maze

Figure 1: First-person viewpoints across all 13 COMRAD scenarios. Agents observe the environment through private 3D egocentric cameras, requiring coordination strategies to emerge from local visual observations.

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.

1 INTRODUCTION

Standardized benchmarks have driven much of the empirical progress in reinforcement learning (RL). The Arcade Learning Environment (Bellemare et al., 2013) and OpenAI Gym (Brockman et al., 2016) gave deep RL a standardized evaluation platform, enabling landmark results such as DQN (Mnih et al., 2015), while MuJoCo (Todorov et al., 2012) played the analogous role in continuous control. The same pattern holds in multi-agent RL (MARL): early studies explored matrix games and small particle environments (Mordatch & Abbeel, 2018; Bard et al., 2020), SMAC and SMACv2 (Samvelyan et al., 2019; Ellis et al., 2023) then established the common evaluation standard for Centralized Training with Decentralized Execution (CTDE), and JaxMARL (Rutherford et al., 2024) reduced wall-clock training costs across these domains. Well-designed benchmarks do more than measure progress; they create the conditions for reproducible, comparable science, and it is precisely this framing that enabled the rapid development of grounding algorithms such as QMIX (Rashid et al., 2018), MAPPO (Yu et al., 2022), and HAPPO (Kuba et al., 2022).

Yet this ecosystem lacks a standardized CTDE benchmark for the 3D egocentric visual setting. Current cooperative MARL benchmarks evaluate algorithms almost exclusively on compact state vectors or top-down 2D images (Papadopoulos et al., 2025; Jin et al., 2025); in contrast, embodied cooperation requires acting from 3D first-person perception, where viewing angles and physical obstacles create natural line-of-sight occlusions. While 3D platforms exist, they are not built as CTDE testbeds: XLand (Team et al., 2021) is closed source, and Minecraft, AI2-THOR, or household simulators (Kolve et al., 2022; Long et al., 2024; Nayak et al., 2024; Chang et al., 2024; Puig et al., 2021) target language-conditioned semantic planning instead of end-to-end visual control, so none combine purely cooperative tasks, first-person visual observations, and CTDE baselines.

We address these challenges with **COMRAD**, a cooperative MARL benchmark suite built on ViZDoom (Kempka et al., 2016). COMRAD evaluates multi-agent coordination in 3D egocentric visual setting while maintaining the throughput needed for multi-seed algorithm comparisons. To achieve this, COMRAD provides 13 cooperative scenarios in which coordination must emerge solely from local, egocentric, visual observations (Figure 1). In particular, agents act via factorized discrete action spaces covering movement, orientation, interaction, and combat primitives; as in SMAC (Samvelyan et al., 2019), this design isolates cooperative decision-making from low-level continuous joint control. Moreover, supported by asynchronous experience collection via Sample Factory (Petrenko et al., 2020), COMRAD is highly practical for training at scale, achieving roughly 25,000 frames per second on a single GPU. Finally, to prevent policies from overfitting to static map geometry, COMRAD includes procedural map generation and curriculum learning tools.

Our contributions are as follows. (1) We introduce a suite of 13 purely cooperative visual MARL scenarios spanning social dilemmas, role asymmetry, spatial and temporal synchronization, and combat coordination. (2) We develop **DoomGen**, a procedural generation library based on relaxed Voronoi tessellation that generates seedable map batches, so algorithms face a distribution of spatial configurations rather than one memorizable map. (3) We design the benchmark around two independently configurable difficulty axes, separating task-mechanical parameters from spatial layout geometry, supported by Automated Curriculum Learning (ACL) tools. (4) We implement seven CTDE baselines (IPPO, MAPPO, HAPPO, IDQN, VDN, QMIX, QPLEX) on a unified, high-throughput asynchronous training stack. (5) We conduct an extensive empirical evaluation analyzing the specific coordination and credit-assignment mechanisms where current algorithms succeed and fail.

2 RELATED WORK

Low-Dimensional Feature-Vector Cooperative Benchmarks. Most cooperative MARL research relies on benchmarks with compact state vectors, a design that abstracts perception away to isolate algorithmic coordination. SMAC and SMACv2 (Samvelyan et al., 2019; Ellis et al., 2023) are the standard CTDE testbeds for unit micromanagement, Overcooked AI (Carroll et al., 2019) provides a common environment for workspace collaboration, Multi-Agent MuJoCo (Peng et al., 2021) plays the same role for factored continuous control, and JaxMARL (Rutherford et al., 2024) reimplements several of these suites with accelerated baselines. The remainder of the category covers complementary axes at low observation cost: grid-world foraging and warehouse tasks for fast prototyping (Christianos et al., 2020; Papoudakis et al., 2021), particle domains for communication

Table 1: Comparison of representative cooperative MARL benchmarks. COMRAD provides 3D egocentric observations, purely cooperative tasks, procedural layout generation, and integrated CTDE baselines in a single open-source suite (OSS). A standard baseline trained for 100M environment steps completes in approximately one hour on a single GPU.

Benchmark	3D/Ego.	Vision	Coop.	ProcGen	CTDE Baselines	OSS	N agents	Scenarios	Year
SMAC / SMACv2	✗	✗	✓	✗/✓	✓	✓	2–27	14 / ∞	2019/23
Google Research Football	✗	Opt.	✓	✗	✗	✓	2–11	11+	2019
Overcooked AI	✗	✗	✓	✗	✗	✓	2	5	2019
Melting Pot 2.0	✗	✓	Mixed	✗	✗	✓	2–16	256	2023
JaxMARL	✗	✗	✓	✗	✓	✓	2–10	8+	2024
MEAL	✗	✓	✓	✗	✓	✓	2–4	4	2025
XLand	✓	✓	Mixed	✓	✗	✗	2–16	∞	2021
Watch&Help	✓	✓	✓	✗	✗	✓	2	5	2020
Habitat 3.0	✓	✓	✓	✗	✗	✓	2	2	2024
TeamCraft	✓	✓	✓	✓	✗	✓	2	10	2024
VIKI-Bench	✓	✓	✓	✗	✗	✓	≥ 2	3 tiers	2025
COMRAD (Ours)	✓	✓	✓	✓	✓	✓	2–8	**13** / ∞	2026

(Mordatch & Abbeel, 2018), Hanabi for theory-of-mind reasoning (Bard et al., 2020), and Google Research Football (Kurach et al., 2020), whose optional pixel observations are rarely used in practice. Comprehensive surveys (Papoudakis et al., 2021; Papadopoulos et al., 2025) confirm that CTDE algorithms are evaluated almost exclusively on low-dimensional state vectors, leaving performance on high-dimensional visual observations largely open question, and it is this gap that motivates COMRAD.

2D Pixel-Based Cooperative Benchmarks. A second class of benchmarks provides visual observations within two-dimensional environments. Melting Pot 2.0 (Agapiou et al., 2023) includes 256 substrate scenarios rendered as top-down grid images, studying social dilemmas and shared resources. PettingZoo (Terry et al., 2021) unifies multi-agent API across diverse visual environments, while Multi-Agent Craftax (Omari et al., 2025) and MEAL (Tomilin et al., 2025b) extend JAX-based survival and Overcooked-style tasks to cooperative and continual settings. Although these environments require image processing, their top-down or isometric viewpoints give agents global arena awareness, avoiding the line-of-sight occlusions and perspective shifts inherent in 3D first-person navigation.

3D Embodied Multi-Agent Environments. Three-dimensional, first-person environments are the closest to embodied cooperative control. Yet, this category remains underdeveloped for standard CTDE evaluation; existing 3D platforms are designed to address broader embodied AI questions that are distinct from ours. TeamCraft (Long et al., 2024), multimodal models predict high-level skills while a scripted controller handles execution. In contrast, COMRAD requires agents to select primitive actions (movement, aiming, interaction) at every simulation step, so coordination is learned directly from experience. XLand (Team et al., 2021) explores open-ended multi-agent learning in 3D worlds, but is closed-source, which disqualifies it as a shared community benchmark: the scientific value of a benchmark rests on the community being able to run, modify, and extend it. Other platforms, including Watch&Help (Puig et al., 2021), PARTNR (Chang et al., 2024), CH-MARL (Sharma et al., 2022), and AI2-THOR variants (Kolve et al., 2022; Nayak et al., 2024; 2025), focus on language-conditioned household tasks where the primary challenge is instruction grounding. Habitat 3.0 (Puig et al., 2024) offers 3D first-person simulation of collaborative home tasks, but its target is human-in-the-loop social embodiment. VIKI-Bench (Kang et al., 2025) is a hierarchical embodied multi-agent benchmark, but it focuses on vision-language models. While these environments serve as important reference points, they differ fundamentally from COMRAD's goal of studying *decentralized cooperative learning from scratch* under CTDE, not RL post-training or zero-shot evaluation of vision-language-action agents. COMRAD is therefore not the first 3D multi-agent environment; it is, to our knowledge, the first to combine purely cooperative tasks, first-person pixel control at every simulation tic, procedural layout generation, and an integrated CTDE baseline suite at a throughput that makes multi-seed comparison affordable. Table 1 situates COMRAD relative to representative existing benchmarks.

3 BACKGROUND

3.1 DECENTRALIZED PARTIALLY OBSERVABLE MDP

We formalize each cooperative task as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek & Amato, 2016), defined by the tuple $\langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}_i\}, \{\mathcal{O}_i\}, T, R, \Omega, \gamma \rangle$. Here $\mathcal{N} = \{1, \ldots, N\}$ denotes the set of agents, and $\mathcal{S}$ denotes the global simulator state, including map geometry, actor poses, inventories, and game scripts. At each timestep t, each agent $i \in \mathcal{N}$ receives a private observation $o_t^i \in \mathcal{O}_i$ from the observation function $\Omega(o_t^i \mid s_t, i)$, which renders the first-person RGB view from agent i's camera pose. Agents choose actions through local policies $\pi_i(a_t^i \mid \tau_i^t)$ conditioned on their private action-observation histories $\tau_i^t = (o_i^0, a_i^0, \ldots, o_i^t)$. The joint action $\mathbf{a}_t = (a_t^1, \ldots, a_t^N) \in \prod_i \mathcal{A}_i$ transitions the environment according to $s_{t+1} \sim T(\cdot \mid s_t, \mathbf{a}_t)$ and yields a shared team reward $r_t = R(s_t, \mathbf{a}_t)$. The team objective is to maximize the expected discounted return $\mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r_t\big]$ with discount factor $\gamma = 0.99$ across all experiments (Appendix B.4). In COMRAD, partial observability arises directly from physical occlusion and viewing angles.

3.2 CENTRALIZED TRAINING WITH DECENTRALIZED EXECUTION

Under the CTDE paradigm (Lowe et al., 2017), training can utilize privileged global states or all agents' observations to stabilize learning and coordinate credit assignment. During execution, however, each agent acts independently using only its local history τ_i^t. COMRAD adheres strictly to this paradigm: centralized value functions and mixing networks have access to team-level states during training, whereas decentralized actor policies receive only their private egocentric camera frames during deployment. Section 5.2 and Appendix B.6 outline the specific CTDE baselines evaluated in this work.

4 COMRAD

COMRAD is built on ViZDoom (Kempka et al., 2016), a lightweight 3D engine based on the classic first-person shooter game, *Doom*. Each scenario is self-contained in a standard WAD file containing geometry, textures, item placements, and an Action Code Script (ACS) program that controls runtime logic, such as enemy behavior, door locks, and objective tracking. During training and evaluation, agents connect to a synchronized multiplayer instance via local UDP. Each agent receives a private first-person frame and sends actions through ViZDoom interface, while a thin Python layer manages observation preprocessing, reward computation, and map loading. Because scenarios live entirely in compiled WAD files, COMRAD remains compatible with any standard RL library.

4.1 ENVIRONMENTS AND SCENARIOS

COMRAD provides 13 scenarios that collectively stress a broad range of cooperative challenges under 3D, first-person partial observability. Agents observe the environment through their local first-person cameras (Section 4.2), so coordination strategies must emerge from local perception alone. Figure 5 maps these scenarios across four dimensions, verifying that the suite covers a diverse space of coordination capability, and Table 2 summarizes the key design parameters. Detailed mechanics, reward functions, and environment parameters are provided in Appendix A.3. Appendix A.2 evaluates these design choices against established environment characteristics.

4.2 OBSERVATION AND ACTION SPACES

Observations. Agents observe egocentric RGB frames resized to 128×72 pixels, which preserves sufficient visual detail for agents to distinguish teammates, hazards, and items while keeping image processing lightweight. In selected scenarios, visual inputs are augmented with a local measurement vector (*Vision+Vec*) tracking variables such as health, ammo, or progress counters, to accelerate learning. These measurements remain strictly local to the individual agent, and contain no privileged information about teammates' states or positions.

Actions. ViZDoom's button controls are exposed as factorized discrete action spaces. For each scenario, the action space forms a Cartesian product over primitive commands: movement, turning,

Table 2: Overview of the 13 COMRAD cooperative scenarios. *Diff.* groups scenarios into difficulty tiers: Easy (1), Medium (3), Difficult (5), and Very Difficult (7). N denotes supported team sizes (each scenario supports at least 2 agents). *Vis+Vec* indicates that egocentric RGB frames are supplemented with a local measurement vector. $|\mathcal{A}|$ is the per-agent factorized discrete action space size. *Objective* is the task-level benchmark score.

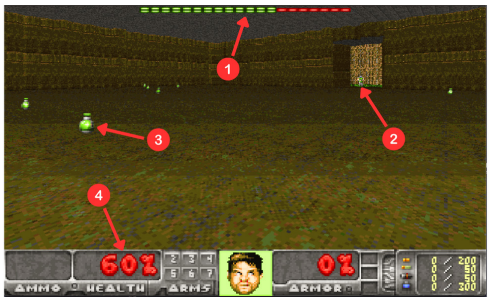
Diff.	Scenario	N	Obs.	$\lvert\mathcal{A}\rvert$	Objective	Coordination Challenge
1	Stag Hunt Arena	2–4	Vision	54	Maximize stag kills	Agents must resolve the hare-versus-stag coordination dilemma under partial observability.
3	Rhythm Sync	2	Vis+Vec	54	Synchronize timing	Separated agents must activate switches within synchronized timing windows without communication.
	Foraging Commons	2–4	Vis+Vec	27	Maximize survival time	Balance individual harvesting against replenishing a shared, limited resource pool.
	Co-op Puzzle	2	Vision	18	Completed puzzle pairs	Agents sequentially unlock gates for one another in turn-based progression.
4	Platform Chain	2	Vision	54	Joint checkpoint progress	A physical tether constrains agents to maintain synchronized spatial proximity.
	Armory Siege	2–8	Vis+Vec	162	Maximize survival & kills	A team defends a core while rotating members to collect distant resources.
	Coop Health Gathering	2	Vision	27	Maximize survival time	Agents navigate a decaying health field together to maximize overlapping collection.
5	Lava Pit	2	Vision	36	Joint traversal	Agents alternate holding pressure plates to support each other's crossing over hazards.
	Smart Enemies	2–4	Vision	54	Maximize enemy kills	Enemies accelerate under concentrated fire, forcing agents to maintain dispersed positions.
	Dumb Enemies	2–4	Vision	54	Maximize enemy kills	Enemies flee solo agents but slow near groups, requiring cooperative pursuit patterns.
6	Ammo Carrier	2	Vis+Vec	36	Maximize survival time	A mobile carrier resupplies a stationary defender to maintain continuous defensive fire.
7	Stealth Labyrinth	2	Vision	54	Activate isolated relays	A torch-gunner pair survives shooting enemies under extreme asymmetric observability.
	Lava Maze	2	Vis+Vec	90	Maximize mazes completed	Form emergent communication between an active navigator and a remote information-providing signaler.

interaction, and weapon use. This yields a single compound action per decision step, with action space size $|\mathcal{A}|$ ranging from 18 to 162 across scenarios. Discrete action factorization isolates cooperative decision-making from continuous motor control, and it allows both policy-gradient and value-decomposition methods to operate on an identical interface: value-decomposition methods such as QMIX, QPLEX, and VDN factorize a joint action-value over per-agent discrete Q-values and have no natural continuous equivalent, so a continuous interface would remove half of the baseline suite from the comparison. Appendix I reports an ablation on *Stag Hunt Arena* showing that continuous motor control adds learning noise without changing any coordination outcomes.

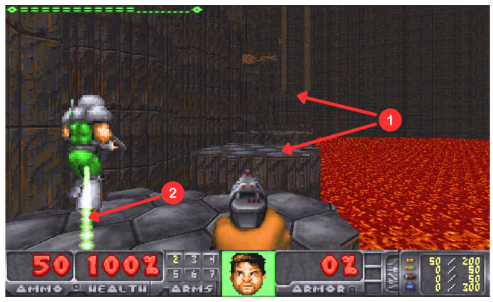
Rewards. Each scenario defines both a shaped reward used during optimization and an unshaped task-level benchmark score that measures cooperative success, such as survival time, completed stages, or targets defeated. We report the unshaped score in evaluation tables because shaped returns differ substantially across reward formulations and do not allow fair comparisons between algorithm families. Details of wrapper implementations and reward formulas are provided in Appendix A.3. Figure 2 illustrates how dense shaping signals provide dense local guidance while task-level scores track the cooperative outcome that the scenario is meant to test.

4.3 DoomGen: Procedural Map Generation

Overfitting to fixed map layouts is a well-documented bottleneck in visual RL (Cobbe et al., 2019; 2020): policies trained on a single environment memorize route sequences, approach angles, and visual landmarks instead of the underlying task logic, and in the multi-agent case they memorize exactly *where* to meet or shoot instead of *how* to coordinate. COMRAD therefore introduces **DoomGen**, a Python library that procedurally generates complete Doom map files (WADs), altering the spatial realization of a task while strictly preserving its semantics; as Appendix Figure 19 shows, geometry can change completely while objective, role structure, and reward logic stay fixed.



(a) *Foraging Commons*. Arrows indicate: (1) shared spawn-rate bar, (2) agent in the Cleanup Station restoring spawn rate, (3) a harvestable health pot, and (4) the agent's decaying health. The shaped reward incentivizes both cleanup and harvesting, but the benchmark score is survival time, requiring agents to balance restoring the commons against immediate collection.



(b) *Platform Chain*. Arrows indicate: (1) target platforms, and (2) the chain connecting agents. The shaped reward signals joint platform arrival (+) and death (-), while the benchmark score is the maximum platform reached, requiring coordinated movement rather than independent speed.

Figure 2: Representative cases illustrating the distinction between shaped training reward and the task-level benchmark score in COMRAD.

To make spatial memorization as difficult as possible, DoomGen avoids orthogonal grid-based rooms. It starts from a relaxed Voronoi tessellation (Aurenhammer, 1991), groups the irregular cells into semantically defined areas such as spawn rooms and corridors, and translates the resulting graph into valid Doom geometry. This irregularity broadens the distribution of boundary shapes, sightlines, and local visual cues, in line with evidence that visual RL agents are highly sensitive to layout diversity (Wu et al., 2018; Li et al., 2023). DoomGen also exposes a reusable task-pool interface: a scenario can be a finite collection of deterministic WADs and sampled episode-by-episode, alternative to being reused as one static map, turning spatial and difficulty variation into a controlled experimental knob while preserving exact reproducibility (Appendix C).

4.4 Difficulty and Variability

Because DoomGen separates the *rules* of a task from its *geometry*, COMRAD can vary difficulty along two independent axes without changing the goal of the scenario (Figure 3). The purpose of this decomposition is to clarify exactly what makes a cooperative scenario hard. In visual MARL, failure generally stems from two distinct challenges: agents either fail to discover the cooperative strategy itself, or fail to execute that strategy under partial observability in unfamiliar layouts. COMRAD isolates these two sources of difficulty.

The first axis is **task-mechanical difficulty**: most scenarios expose explicit parameters such as enemy count, spawn rate, time limits, or hazard intensity, and many contain an internal progression from easier to harder coordination regimes (the early bridge handoffs in *Lava Pit* are intentionally simpler than later ones). These controls define calibrated difficulty tiers that test strategic and mechanical reasoning. The second axis is **spatial difficulty**, provided by DoomGen: holding task rules constant, map generation can increase arena scale, lengthen connecting corridors, add obstacles, and introduce occluding walls. Many cooperative failures in first-person settings are spatial, since agents may know *what* the required pattern is but fail to execute it when sightlines are blocked. Thus, this tests whether agents can maintain visual contact and navigate effectively when direct sightlines are interrupted.

Crucially, the two axes are **independently configurable controls**. In *Armory Siege*, for example, `enemy_counts` $\in \{5, 10, 15, 20, 30\}$ sets the mechanical tier, while `corridor_length` $\in \{5, 10, 15, 30\}$ sets spatial tier. Setting parameters to $(5, 5)$ creates a compact and easy baseline, whereas $(5, 30)$ increases spatial navigation difficulty while keeping combat pressure fixed. A stratified 3×3 evaluation across these tiers (Appendix D.5) confirms that the two axes operate independently, with spatial axis posing the steeper challenge for decentralized execution. This factorization also makes the conclusions sharper: a method that succeeds along the mechanical axis but fails along the spatial axis is bottlenecked by visual-spatial generalization rather than by

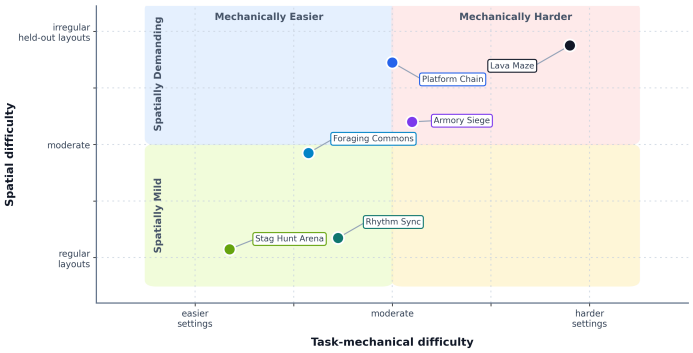


Figure 3: COMRAD's two orthogonal difficulty axes. The horizontal axis reflects task-mechanical difficulty, controlled by scenario parameters such as enemy count or hazard intensity. The vertical axis reflects spatial difficulty, controlled by DoomGen through layout variation on held-out maps. This separation supports curriculum progression and generalization evaluation.

coordination reasoning. Both axes further offer structured stages for Automated Curriculum Learning (ACL), discussed in Appendix D.

5 Experiments

5.1 Training Infrastructure

A usable visual MARL benchmark must be trainable at enough throughput to support multi-seed evaluation, broad hyperparameter sweeps, and repeated ablations. We therefore build our training stack on Sample Factory (Petrenko et al., 2020), retaining its asynchronous design, which separates experience collection, inference, and learning across dedicated workers, and adapting the core framework to natively support synchronized multi-agent environments and a broader set of MARL algorithms. We prefer it to JAX-based frameworks such as JaxMARL (Rutherford et al., 2024) because ViZDoom is a CPU-bound C++ rendering engine that cannot execute natively on an accelerator, so the appropriate pattern is an asynchronous CPU simulation pipeline paired with GPU policy learning; Appendix B.7 develops this argument further.

To bridge this pipeline with our scenarios, we wrap synchronized ViZDoom instances in a multi-agent interface that routes data uniformly, while still supporting diverse model topologies across baselines. Detailed network architectures and hyperparameters are documented in Appendices B.6 and B.4, the evaluation protocol in Appendix B.3, and training throughput metrics in Appendix B.7.

5.2 Baseline Algorithms

We evaluate seven representative CTDE algorithms from two main paradigms, porting standard reference implementations into our asynchronous architecture. For on-policy policy gradients, we implement independent PPO (IPPO), Multi-Agent PPO (MAPPO) (Yu et al., 2022), and Heterogeneous-Agent PPO (HAPPO) (Kuba et al., 2022), following the HARL library (Kuba et al., 2022). For off-policy value decomposition, we implement independent DQN (IDQN), Value-Decomposition Networks (VDN) (Sunehag et al., 2018), QMIX (Rashid et al., 2018), and QPLEX (Wang et al., 2021), following PyMARL codebase (Samvelyan et al., 2019) for VDN and QMIX, and the official release for QPLEX (Wang et al., 2021). For QPLEX, we evaluate both the duplex-dueling mixer with transformation heads (QPLEX-D) and the attention-based mixer (QPLEX-Q).

5.3 Baseline Results

We evaluate eight baseline configurations across the seven algorithm families under a shared 100M-step budget. Table 3 reports final benchmark scores, and Figure 4 compares the normalized gap between the best on-policy and off-policy baseline for each scenario. Appendix J provides learning curves and a normalized score matrix, and Appendix G a behavioral analysis of when centralized training hurts.

A clear pattern emerges: no single algorithmic family dominates across all scenarios, and COMRAD remains far from solved. Instead, relative performance reflects the specific cooperative requirements

Table 3: Baseline evaluation across 100M training steps, evaluated zero-shot on held-out layouts. All runs use teams of $N = 2$ agents, so the comparison is not confounded by team size effects (documented in Appendix E). For each scenario, policies trained across five random seeds $\{42, 68, 81, 97, 154\}$ are evaluated for 100 episodes on a held-out map batch generated with separate seeds. Entries report mean $\pm$ 95% CI half-width across the five training seeds. **Ceiling** indicates the theoretical maximum score achievable under optimal play; *Lava Maze* scores the number of mazes completed per episode and has no finite ceiling (–). An analysis of near-zero scores on the most challenging scenarios is provided in Appendix H.

Scenario	IPPO	MAPPO	HAPPO	IDQN	VDN	QMIX	QPLEX-D	QPLEX-Q	Ceiling
Stag Hunt Arena	17.41 ± 0.24	$\mathbf{25.48 \pm 0.30}$	2.63 ± 0.11	14.19 ± 0.17	18.11 ± 0.36	0.26 ± 0.06	1.53 ± 0.09	3.35 ± 0.21	36
Rhythm Sync	0.57 ± 0.05	$\mathbf{0.67 \pm 0.04}$	0.21 ± 0.02	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	1.00
Foraging Commons	2341.18 ± 29.74	2497.55 ± 32.06	2212.40 ± 38.15	2178.36 ± 33.41	2339.82 ± 41.27	$\mathbf{2531.07 \pm 45.32}$	1918.44 ± 24.10	2358.61 ± 38.05	21000
Co-op Puzzle	2.98 ± 0.08	$\mathbf{3.82 \pm 0.11}$	2.15 ± 0.16	0.02 ± 0.01	0.06 ± 0.02	0.00 ± 0.01	0.00 ± 0.01	0.05 ± 0.03	5
Platform Chain	1.96 ± 0.02	$\mathbf{4.85 \pm 0.18}$	4.59 ± 0.19	0.71 ± 0.06	1.09 ± 0.07	0.94 ± 0.03	3.33 ± 0.21	1.01 ± 0.03	47
Armory Siege	15.95 ± 3.10	19.20 ± 2.45	-0.85 ± 1.15	10.50 ± 1.05	17.80 ± 1.55	$\mathbf{20.45 \pm 4.05}$	1.05 ± 0.95	13.60 ± 2.65	100
Co-op Health Gathering	$\mathbf{1092.44 \pm 121.65}$	561.20 ± 28.44	201.36 ± 8.24	161.87 ± 2.20	566.41 ± 26.83	168.95 ± 2.13	190.42 ± 5.31	273.05 ± 17.82	2100
Lava Pit	0.90 ± 0.01	$\mathbf{1.00 \pm 0.01}$	$\mathbf{1.00 \pm 0.01}$	0.93 ± 0.01	0.78 ± 0.05	0.96 ± 0.03	0.00 ± 0.01	$\mathbf{1.00 \pm 0.01}$	11
Smart Enemies	11.39 ± 0.84	$\mathbf{14.47 \pm 0.62}$	6.71 ± 0.40	3.38 ± 0.22	4.77 ± 0.36	0.61 ± 0.08	0.00 ± 0.01	0.68 ± 0.07	84
Dumb Enemies	$\mathbf{12.21 \pm 1.87}$	2.35 ± 0.21	0.99 ± 0.12	1.11 ± 0.12	2.68 ± 0.23	1.14 ± 0.14	0.00 ± 0.01	1.86 ± 0.17	90
Ammo Carrier	$\mathbf{7031.42 \pm 495.12}$	1351.10 ± 74.30	1310.45 ± 62.15	3795.80 ± 390.11	2511.20 ± 225.40	2341.15 ± 145.20	1515.60 ± 61.22	2045.30 ± 201.10	8400
Stealth Labyrinth	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	$\mathbf{0.21 \pm 0.01}$	0.00 ± 0.01	0.00 ± 0.01	1.00
Lava Maze	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	–

of each scenario. On-policy actor-critic methods perform best on tasks requiring precise temporal alignment and spatial synchronization. MAPPO is the most reliable general-purpose baseline, leading on *Stag Hunt Arena*, *Rhythm Sync*, and *Co-op Puzzle*, which test synchronized movements and precise timing, while HAPPO performs well on traversal tasks such as *Platform Chain* and *Lava Pit*, but does not consistently outperform MAPPO. Across the suite, the strongest on-policy method leads in 8 of 13 scenarios (Figure 4). In contrast, off-policy value decomposition succeeds on tasks requiring long-horizon credit assignment over shared resource pools: QMIX achieves the highest returns on *Foraging Commons* and *Armory Siege*, and it is the only baseline to achieve non-zero success on *Stealth Labyrinth* at the 100M training steps. The two *Very Difficult* scenarios are nonetheless learnable under this protocol. On the original maps, a single learner acquires each role once its partner is scripted, and *Lava Maze* becomes easy once communication is removed. This places their difficulty in co-adaptation between roles instead of in perception or reward design (Appendix H.1).

Zero-shot generalization across layouts. All results in Table 3 represent zero-shot evaluation on held-out maps. While training samples from a fixed batch of procedurally generated layouts, evaluation tests frozen checkpoints over 100 episodes on an independently seeded test batch. Performance does not degrade noticeably between training and held-out evaluation, which is the expected from our layout variations; as the training pool contains a wide range of layouts and difficulties (Appendix C), agents cannot rely on memorizing a specific map, so they are encouraged to learn generalized navigation and coordination behaviors. Consequently, the gaps analyzed below cannot come from spatial overfitting, and should instead be because of coordination.

5.4 Why Methods Succeed and Fail

A scoreboard only ranks method performance; a benchmark reveals ***why*** those differences exist. Because all baselines share the same visual observations, action spaces, and training budgets, the differences in Table 3 trace back to specific algorithmic mechanisms. We summarize the three findings of Appendix F, each with a direction for future work. **(i) A centralized critic helps synchronization and hurts asymmetric roles.** MAPPO beats IPPO on tightly coupled tasks such as *Rhythm Sync* and *Co-op Puzzle*, but IPPO scores more than five times higher on the role-asymmetric *Ammo Carrier*, where its runner learns a supply loop that MAPPO's runner never acquires (Appendix G). We attribute this to the critic's static inputs; critics whose inputs adapt to local, role-specific, or team context could fix this. **(ii) Replay buffers help long-horizon credit assignment and introduce teammate non-stationarity.** QMIX leads on *Foraging Commons* and *Armory Siege*, yet stale teammate trajectories and monotonic mixing make value-decomposition methods collapse on narrow-window synchronization tasks. Possible fixes are policy-age correction, explicit spatial or teammate memory, and, for larger teams, set-attention or graph critics in place of fixed observation concatenation. **(iii) More expressive architectures did not perform better.** HAPPO's sequential updates and QPLEX's advantage and transformation networks are hard to train stably through convolutional backbones, so both underperform MAPPO and QMIX. Hierarchical credit assignment over meaningful events (handoffs, room clearing, supply loops) or grounded communication channels

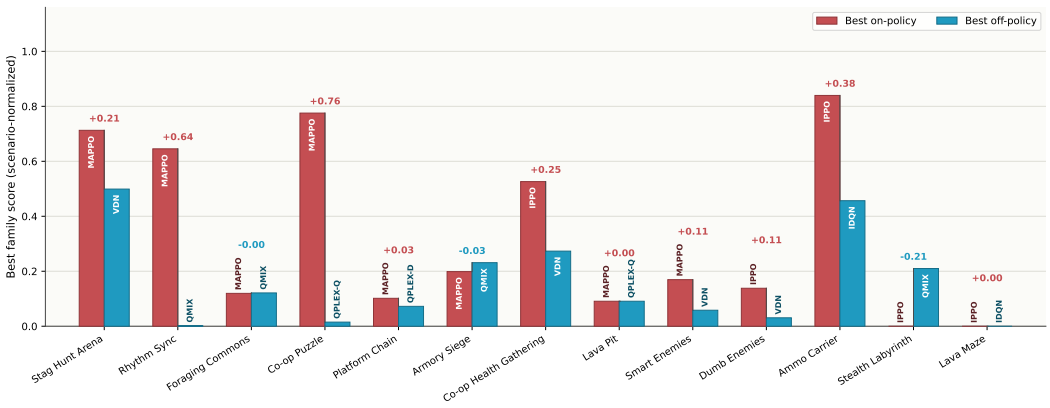


Figure 4: Scenario-normalized gap between the strongest on-policy and off-policy baseline. Each scenario is normalized by its own score range before comparison, using $\tilde{s} = \mathrm{clip}(s/s_{\max}, 0, 1)$ where s is the mean benchmark score from Table 3 and $s_{\max}$ is that scenario's ceiling (*Lava Maze* has no finite ceiling, so we use $s_{\max} = 4$ completed mazes; every scenario has a natural floor of 0. On-policy actor-critic methods lead in 8 of 13 scenarios overall.

is likely more effective than larger mixing networks. Overall, the scope of credit assignment explains more of the results than the on-policy/off-policy split.

6 Discussion and Limitations

COMRAD shifts benchmark pressure from low-dimensional feature tasks toward embodied coordination from first-person observations, and the baseline results suggest the shift is significant: methods that are competitive on established cooperative benchmarks encounter distinct challenges under visual partial observability, and the hardest scenarios remain open even at substantial training budgets.

Four of COMRAD's scopes were defined at the start to remove specific confounds. **(i) Purely cooperative, with no rival learning agents.** Introducing rivals would turn every scenario into a general-sum or competitive problem, which requires different metrics, different baselines, and different notions of success; this would obscure the coordination signal the suite is built to measure. Thus, we regard adversarial teammates as a valuable extension, but it answers a different question. **(ii) Partner cooperation as the main regime.** Most scenarios are designed around shared interactions or complementary roles between two agents, because it is far easier to attribute a failure to a specific coordination mechanism when only one partnership is involved; this also matches the scope of Overcooked (Carroll et al., 2019) and Bi-DexHands (Chen et al., 2022). However, larger teams are supported, and Appendix E reports what happens to both learning and throughput as N grows: coordination difficulty and simulation cost rise together. **(iii) Factorized discrete actions.** This is what lets actor-critic and value-decomposition methods share a single task interface, and Appendix I shows the cost of abandoning it: continuous control leaves the benchmark score unchanged or lower, fluctuates training, and removes four of the eight baseline configurations from the comparison. **(iv) Benchmark scores are scenario-specific.** This is to reflect natural, interpretable task metrics, but it requires score normalization for cross-scenario comparisons (as shown in Figure 4 and Appendix J).

7 Conclusion

We introduced COMRAD, a benchmark suite for cooperative MARL in 3D first-person environments. By combining 13 diverse cooperative tasks, procedural map generation with DoomGen, adjustable difficulty axes, and high-throughput training, COMRAD provides a rigorous platform for embodied coordination research. Our evaluations show that current CTDE algorithms face significant limitations when coordinating under visual partial observability, with on-policy and off-policy methods failing on complementary challenges. We hope COMRAD becomes an accessible, reproducible testbed for studying visual multi-agent credit assignment, representation learning, and decentralized execution.

AI USE STATEMENT

The paper was written and reviewed by human authors. We used Overleaf checker for typographical errors checking, AI paper reviews (e.g. Google's Paper Assistant Tool, Stanford Paper Review, and OpenAI's GPT Paper Review) for getting feedback before submitting, and GPT models for formatting diagrams/plots (we had pre-built Python templates for all diagrams/plots, GPT models were only involved in modifying the template to reflect what we envision).

All training architecture and algorithms were written and adapted manually (without AI) from established RL libraries (e.g. PyMARL, HARL). Benchmark scenarios/tasks and taxonomies were also designed by human. The procedural map generation component was a spark we had in the middle of the project, so it was initially written with the help of Claude Opus 4.6; however, when the idea is verified, all generated code and underlying logic were tested and reviewed line-by-line by human authors.

We confirm we have complete understanding of the entire codebase and the paper, and we have reviewed all AI-assisted components. We take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

COMRAD is released as a benchmark, so reproducibility is a primary deliverable. The complete benchmark suite, DoomGen library, scenario scripts, baseline implementations, and launcher commands are available in an open-source repository[1], alongside pre-generated layout batches. Appendix B outlines the experimental protocol: training seeds and evaluation checkpoints (Appendix B.3), hyperparameter tables (Appendix B.4), neural network architectures (Appendix B.6), hardware specifications and throughput profiles (Appendix B.7), and statistical checks on the game engine's random number generator (Appendix B.2). Appendix A.3 documents within-episode randomization and objective metrics J, and Appendix C explains layout pool sampling. Additional studies, including the difficulty grid (Appendix D.5), action-space ablation (Appendix I), agent scaling (Appendix E), curriculum dynamics (Appendix D), hyperparameter sensitivity (Appendix B.5), and human performance baselines (Appendix H.2), are detailed with dedicated experimental settings.

[1] https://anonymous.4open.science/r/COMRAD-42C6

REFERENCES

Sam Adams, Itamar Arel, Joscha Bach, Robert Coop, Rod Furlan, Ben Goertzel, J. Hall, Alexei Samsonovich, Matthias Scheutz, Matthew Schlesinger, Stuart Shapiro, and John Sowa. Mapping the landscape of human-level artificial general intelligence. *Ai Magazine*, 33:25–42, 03 2012. doi: 10.1609/aimag.v33i1.2322.

John P. Agapiou, Alexander Sasha Vezhnevets, Edgar A. Duéñez-Guzmán, Jayd Matyas, Yiran Mao, Peter Sunehag, Raphael Köster, Udari Madhushani, Kavya Kopparapu, Ramona Comanescu, DJ Strouse, Michael B. Johanson, Sukhdeep Singh, Julia Haas, Igor Mordatch, Dean Mobbs, and Joel Z. Leibo. Melting pot 2.0, 2023. URL https://arxiv.org/abs/2211.13746.

Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, September 1991. ISSN 0360-0300. doi: 10.1145/116873.116880. URL https://doi.org/10.1145/116873.116880.

Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. The Hanabi challenge: A new frontier for AI research. *Artificial Intelligence*, 280:103216, 2020.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL https://arxiv.org/abs/1606.01540.

Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019. URL https://arxiv.org/abs/1910.05789.

Matthew Chang, Gunjan Chhablani, Alexander Clegg, Mikael Dallaire Cote, Ruta Desai, Michal Hlavac, Vladimir Karashchuk, Jacob Krantz, Roozbeh Mottaghi, Priyam Parashar, Siddharth Patki, Ishita Prasad, Xavier Puig, Akshara Rai, Ram Ramrakhya, Daniel Tran, Joanne Truong, John M. Turner, Eric Undersander, and Tsung-Yen Yang. Partnr: A benchmark for planning and reasoning in embodied multi-agent tasks, 2024. URL https://arxiv.org/abs/2411.00081.

Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuang Jiang, Stephen Marcus McAleer, Yiran Geng, Hao Dong, Zongqing Lu, Song-Chun Zhu, and Yaodong Yang. Towards human-level bimanual dexterous manipulation with reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, volume 35, 2022. URL https://arxiv.org/abs/2206.08686.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.

Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Filippos Christianos, Georgios Papoudakis, Arrasy Rahman, and Stefano V. Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *PMLR*, pp. 1989–1998, 2021. URL https://proceedings.mlr.press/v139/christianos21a.html.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *PMLR*, pp. 1282–1289, 2019. URL https://proceedings.mlr.press/v97/cobbe19a.html.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *PMLR*, pp. 2048–2056, 2020.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Nicolaus Foerster, and Shimon Whiteson. SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=5OjLGiJW3u.

Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *PMLR*, pp. 1146–1155, 2017. URL https://proceedings.mlr.press/v70/foerster17b.html.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1311–1320. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/graves17a.html.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *PMLR*, pp. 4940–4950, 2021. URL https://proceedings.mlr.press/v139/jiang21b.html.

Weiqiang Jin, Hongyang Du, Biao Zhao, Xingwu Tian, Bohang Shi, and Guang Yang. A comprehensive survey on multi-agent cooperative decision-making: Scenarios, approaches, challenges and perspectives, 2025. URL https://arxiv.org/abs/2503.13415.

Li Kang, Xiufeng Song, Heng Zhou, Yiran Qin, Jie Yang, Xiaohong Liu, Philip H. S. Torr, Lei Bai, and Zhenfei Yin. VIKI-R: Coordinating embodied multi-agent cooperation via reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2025. URL https://arxiv.org/abs/2506.09049.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A Doom-based AI research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8, 2016. doi: 10.1109/CIG.2016.7860433.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai, 2022. URL https://arxiv.org/abs/1712.05474.

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. In *International Conference on Learning Representations (ICLR)*, 2022.

Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Google Research Football: A novel reinforcement learning environment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 4501–4510, 2020.

John E. Laird and Robert E. Wray. Cognitive architecture requirements for achieving AGI. In *Artificial General Intelligence*, 2010. URL https://api.semanticscholar.org/CorpusID:19032843.

David Lewis. *Convention: A Philosophical Study*. Wiley-Blackwell, Cambridge, MA, USA, 2008.

Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3461–3475, 2023. doi: 10.1109/TPAMI.2022.3190471.

Qian Long, Zhi Li, Ran Gong, Ying Nian Wu, Demetri Terzopoulos, and Xiaofeng Gao. Teamcraft: A benchmark for multi-modal multi-agent systems in minecraft, 2024. URL https://arxiv.org/abs/2412.05255.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Xueguang Lyu, Andrea Baisero, Yuchen Xiao, Brett Daley, and Christopher Amato. On centralized critics in multi-agent reinforcement learning, 2024. URL https://arxiv.org/abs/2408.14597.

Anuj Mahajan, Mikayel Samvelyan, Tarun Gupta, Benjamin Ellis, Mingfei Sun, Tim Rocktäschel, and Shimon Whiteson. Generalization in cooperative multi-agent systems, 2022. URL https://arxiv.org/abs/2202.00104.

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

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Siddharth Nayak, Adelmo Morrison Orozco, Marina Ten Have, Vittal Thirumalai, Jackson Zhang, Darren Chen, Aditya Kapoor, Eric Robinson, Karthik Gopalakrishnan, Brian Ichter, James Harrison, Anuj Mahajan, and Hamsa Balakrishnan. MAP-THOR: Benchmarking long-horizon multi-agent planning frameworks in partially observable environments. In *Multi-modal Foundation Model meets Embodied AI Workshop @ ICML2024*, 2024. URL https://openreview.net/forum?id=ZygZN5egzy.

Siddharth Nayak, Adelmo Morrison Orozco, Marina Ten Have, Vittal Thirumalai, Jackson Zhang, Darren Chen, Aditya Kapoor, Eric Robinson, Karthik Gopalakrishnan, James Harrison, Brian Ichter, Anuj Mahajan, and Hamsa Balakrishnan. Llamar: Long-horizon planning for multi-agent robots in partially observable environments, 2025. URL https://arxiv.org/abs/2407.10031.

Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. Springer, 2016. doi: 10.1007/978-3-319-28929-8.

Bassel Al Omari, Michael Matthews, Alexander Rutherford, and Jakob Nicolaus Foerster. Multi-agent craftax: Benchmarking open-ended multi-agent reinforcement learning at the hyperscale, 2025. URL https://arxiv.org/abs/2511.04904.

George Papadopoulos, Andreas Kontogiannis, Foteini Papadopoulou, Chaido Poulianou, Ioannis Koumentis, and George Vouros. An extended benchmarking of multi-agent reinforcement learning algorithms in complex fully cooperative tasks, 2025. URL https://arxiv.org/abs/2502.04773.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Böhmer, and Shimon Whiteson. FACMAC: Factored multi-agent centralised policy gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021. URL https://arxiv.org/abs/2003.06709.

Aleksei Petrenko, Zhehui Huang, Tushar Kumar, Gaurav S. Sukhatme, and Vladlen Koltun. Sample factory: Egocentric 3D control from pixels at 100000 FPS with asynchronous reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *PMLR*, pp. 7652–7662, 2020.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey, 2020. URL https://arxiv.org/abs/2003.04664.

Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Yuan-Hong Liao, Joshua B. Tenenbaum, Sanja Fidler, and Antonio Torralba. Watch-and-help: A challenge for social perception and human-ai collaboration, 2021. URL https://arxiv.org/abs/2010.09890.

Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, Vladimír Vondruš, Theophile Gervet, Vincent-Pierre Berges, John M. Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars, and robots. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.13724.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4295–4304. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/rashid18a.html.

Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 10199–10210, 2020. URL https://arxiv.org/abs/2006.10800.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert Tjarko Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktäschel, Chris Lu, and Jakob Nicolaus Foerster. Jaxmarl: Multi-agent rl environments and algorithms in jax. In *The Thirty-eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Mutsuo Saito and Makoto Matsumoto. Simd-oriented fast mersenne twister: a 128-bit pseudorandom number generator. In Alexander Keller, Stefan Heinrich, and Harald Niederreiter (eds.), *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pp. 607–622, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-74496-2.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

Vasu Sharma, Prasoon Goyal, Kaixiang Lin, Govind Thattai, Qiaozi Gao, and Gaurav S. Sukhatme. Ch-marl: A multimodal benchmark for cooperative, heterogeneous multi-agent reinforcement learning, 2022. URL https://arxiv.org/abs/2208.13626.

Ryan Sullivan, Ryan Pégoud, Ameen Ur Rahmen, Xinchen Yang, Junyun Huang, Aayush Verma, Nistha Mitra, and John P Dickerson. Syllabus: Portable curricula for reinforcement learning agents. *Reinforcement Learning Journal*, 2025.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 2085–2087, 2018.

Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-ended learning leads to generally capable agents, 2021. URL https://arxiv.org/abs/2107.12808.

J. K. Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, Niall Williams, Yashas Lokesh, and Praveen Ravi. Pettingzoo: Gym for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Tristan Tomilin, Meng Fang, Yudi Zhang, and Mykola Pechenizkiy. COOM: A game benchmark for continual reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Tristan Tomilin, Meng Fang, and Mykola Pechenizkiy. Hasard: A benchmark for vision-based safe reinforcement learning in embodied agents. In *The Thirteenth International Conference on Learning Representations (ICLR 2025)*, 2025a. URL https://arxiv.org/abs/2503.08241.

Tristan Tomilin, Luka van den Boogaard, Samuel Garcin, Bram Grooten, Meng Fang, and Mykola Pechenizkiy. Meal: A benchmark for continual multi-agent reinforcement learning. In *Proceedings of the Datasets & Benchmarks Track at NeurIPS 2025*, 2025b. URL https://arxiv.org/abs/2506.14990.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1801.02209.

Marek Wydmuch, Michal Kempka, and Wojciech Jaskowski. ViZDoom competitions: Playing *Doom* from pixels. *IEEE Transactions on Games*, 11(3):248–259, 2019. ISSN 2475-1510. doi: 10.1109/tg.2018.2877047. URL http://dx.doi.org/10.1109/TG.2018.2877047.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 24611–24624, 2022.

Shenao Zhang, Li Shen, Lei Han, and Li Shen. Learning meta representations for agents in multi-agent reinforcement learning. In Sarath Chandar, Razvan Pascanu, Hanie Sedghi, and Doina Precup (eds.), *Proceedings of The 2nd Conference on Lifelong Learning Agents*, volume 232 of *Proceedings of Machine Learning Research*, pp. 292–317. PMLR, 22–25 Aug 2023. URL https://proceedings.mlr.press/v232/zhang23a.html.

A Benchmark Design and Scenario Specification

A.1 ViZDoom as a Platform for Embodied Cooperative RL

ViZDoom (Kempka et al., 2016) occupies a useful middle ground between visual complexity and practical training throughput. Its lightweight C++ rendering engine delivers first-person visual observations and flexible scenario scripting at speeds required for systematic MARL experimentation. Prior benchmark efforts have successfully leveraged ViZDoom for single-agent continual learning (Tomilin et al., 2023) and safe RL (Tomilin et al., 2025a), demonstrating suitability for complex visual tasks at scale. While competitive deathmatch multiplayer is well-known within the ViZDoom community (Wydmuch et al., 2019), no prior work has established a standardized, purely cooperative multi-agent testbed with diverse tasks, rigorous baselines, and procedural layout generation. COMRAD extends this lineage to the cooperative MARL setting.

A.2 Required Environmental Characteristics

		Stag Hunt Arena	Rhythm Sync	Foraging Commons	Co-op Puzzle	Platform Chain	Armory Siege	Co-op Health Gathering	Lava Pit	Smart Enemies	Dumb Enemies	Ammo Carrier	Stealth Labyrinth	Lava Maze
Game Theoretic Properties	Corresponding>Conflicting Interests	x	x		x	x	x	x	x	x	x	x	x	x
	Conflicting>Corresponding Interests			x										
	Asymmetric roles											x	x	x
	Near-perfect Information	x			x	x				x	x	x		
	Far-from-perfect Information		x	x			x	x					x	x	x
	Dyadic (1:1) interactions cause immediate rewards	x	x		x	x		x	x					x	x
	1:many interactions cause immediate rewards												x		
	many:many interactions cause immediate rewards			x			x				x	x			
	Social Dilemma (Mixed-Motive)			x											
	Pure Coordination Game	x	x												
	Equilibrium selection	x	x												
Game Design Properties	Puzzle-like		x		x	x				x				x	x
	Some physical state changes are irreversible	x	x	x	x	x	x	x		x	x	x	x	x	x
	Strict execution/timing thresholds		x			x				x					
	Procedural Generation / High variance				x		x		x	x					x
Reinforcement Learning Properties	Dynamic intra-episode learning		x								x	x			
	Delayed rewards (long after proximal cause)				x		x	x		x			x	x	
	Extreme Partial Observability (Dec-POMDP)													x	x
	Spatially-separated action/reward				x	x		x		x			x		
Properties of Potential Emergent Behaviors	Task Partitioning							x					x	x	x
	Resource Sharing				x			x					x		
	Implicit Signaling / Comm Protocol			x										x	x
	Convention Following	x	x			x								x	x
	Focal point (there is a clear default convention)	x	x												
	Turn-taking					x				x					
	Altruistic Sacrifice / Public Good				x	x				x					
	Reciprocity				x	x				x					
	Spatial Formation / Team Spacing						x		x		x	x			
	Flexibility						x	x			x	x			

Figure 5: Cooperation properties of COMRAD's 13 scenarios across four analytical dimensions. A mark (✗) indicates that the property is a defining characteristic of the scenario. This taxonomy confirms that the benchmark spans a diverse space of cooperative challenges rather than clustering around a single challenge type.

This section is a design audit rather than a separate empirical claim. The main paper already explains COMRAD's scenarios, observation model, difficulty axes, and training stack. Here we relate those choices to the eight environmental characteristics proposed by Laird & Wray (2010) and later refined by Adams et al. (2012). We keep this discussion in the appendix because it motivates the benchmark design, but it is not required to interpret the experimental results.

COMRAD is not presented as a literal test of general intelligence, and a finite ViZDoom benchmark should not be described as satisfying these criteria in their strongest open-world sense. The comparison is still useful because it clarifies which properties of rich embodied environments COMRAD is designed to capture directly, which ones it only approximates, and where the benchmark is intentionally scoped for controlled MARL evaluation rather than unrestricted world modeling.

- **C1. The environment is complex, with diverse, interacting, and richly structured objects.** COMRAD satisfies this criterion through both its entities and its geometry. Rooms, doors, pressure plates, hazards, enemies, pickups, resource depots, and scripted objectives are embedded in irregular Voronoi-relaxed layouts, creating structured spatial and object relations that agents must infer from local first-person perception.

- **C2. The environment is dynamic and open.** COMRAD is dynamic at both the within-episode and across-episode levels. Enemy waves, pickup spawns, timing windows, and hazard states evolve during an episode, while procedural map generation and pre-generated layout batches expose agents to new layouts across training. This remains a bounded approximation of openness rather than an unbounded world model.
- **C3. Task-relevant regularities exist at multiple time scales.** COMRAD exhibits this property from the game-tic level to long-horizon coordination. Agents must react to immediate combat and movement events, sustain cooperative behavior over long episodes, and adapt across procedurally varied layouts and curriculum stages.
- **C4. Other agents impact performance.** This is central to COMRAD. Most scenarios are explicitly interdependent, and several are impossible or degenerate without coordination, as success depends on synchronization, role complementarity, shared resource management, or the maintenance of joint spatial constraints.
- **C5. Tasks can be complex, diverse, and novel.** COMRAD is task-diverse within a fixed cooperative benchmark setting. The 13 scenarios span social dilemmas, temporal synchronization, combat coordination, asymmetric logistics, tethered traversal, and communication-style tasks, while procedural variation, pre-generated task pools, and difficulty control prevent a single memorized solution template from sufficing.
- **C6. Interactions between agent, environment, and tasks are complex and limited.** Each agent acts from a local first-person view with a constrained action interface and no access to privileged teammate state at evaluation time. As a result, policies must be decided under severe partial observability and cannot rely on globally synchronized symbolic planning.
- **C7. The agent's computational resources are limited.** Limited computation is an explicit part of the benchmark setting. Policies must learn from high-dimensional visual input under fixed GPU and CPU budgets, and the training stack is designed around realistic throughput constraints rather than unlimited batched simulation.
- **C8. Agent existence is long-term and continual.** COMRAD only partially instantiates this criterion. Many tasks are long-horizon, and the curriculum interface supports repeated adaptation across large task pools, but the benchmark still operates through episodic resets and finite scenario families. It therefore supports continual-style experimentation without claiming to be a full lifelong environment.

Taken together, these criteria explain why COMRAD emphasizes first-person partial observability, role asymmetry, procedural variation, and high-throughput training under realistic system limits. They also highlight an important boundary of the benchmark. COMRAD is intended as a rigorous embodied cooperative MARL testbed rather than a complete open-world evaluation of general intelligence. That narrower scope is deliberate because standardized comparison requires controlled tasks, reproducible difficulty, and interpretable failure cases.

A.3 Environment Details and Reward Functions

Throughout this appendix, r_t denotes the per-step training reward after wrapper shaping, and J denotes the scenario-level benchmark score reported in the evaluation tables. We distinguish episode-level randomization, which changes the task within a fixed generated WAD, from WAD-level configuration, which changes the underlying scenario instance used for curriculum learning or WAD swapping. The statistical validation of the ACS random source that drives this within-scenario stochasticity is reported separately in Appendix B.2. The figures in this section are representative in-game captures of the corresponding scenarios.

Notation. The reward expressions below are algebraic abstractions of the implemented wrappers, written so that the optimization objective of each scenario is unambiguous. They share one convention: for any tracked quantity x, the symbol $\Delta x_t = x_t - x_{t-1}$ is the change over one environment step, and T is the terminal step of the episode. A superscript $+$ or $-$ restricts the change to its positive or negative part, so $\Delta h_t^+ = \max(0, h_t - h_{t-1})$ is health *gained* this step and $\Delta h_t^- = \max(0, h_{t-1} - h_t)$ is health *lost*; the same convention applies to ammo (a_t), weapons held (w_t), and core health (c_t). The recurring symbols are: h_t the agent's health; a_t is its ammo; k_t is its cumulative kill count; hit_t is its cumulative count of damaging hits landed on enemies, so that $\Delta\mathrm{hit}_t$ rewards connecting a

shot; $n_t^{\bullet}$ is a cumulative count of events of type $\bullet$; and d_t is a term for distance, always has a sign so that $\Delta d_t > 0$ means the agent moved *closer* to the currently relevant target. Specifically, Δd_t^{stag} measures approach to the stag, Δd_t in *Ammo Carrier* measures approach to the depot or hub selected by the runner's current delivery phase, Δd_t^{path} measures progress along the safe path in *Lava Maze*, and $\Delta d_t^{\text{guided}}$ measures the subset of that progress made while a signal from the observer was active, which is the only term for *using* communication. $\mathbf{1}\{\cdot\}$ is an indicator that equals 1 on the step where the named event occurs and 0 otherwise. Greek letters $(\alpha, \beta, \gamma, \delta, \eta, \kappa, \lambda, \mu, \nu, \rho, \xi)$ are non-negative scenario-specific constants; they are reused across scenarios as generic weights. We do not reproduce their numeric values here because they are scenario-specific.

Stag Hunt Arena This scenario instantiates a two-player coordination dilemma between a reliable small solo payoff and a larger payoff that can be realized only through simultaneous commitment to a shared target. A high-level reward abstraction is

$$r_t = \alpha \Delta n_t^{\text{rabbit}} + \beta \Delta d_t^{\text{stag}} + \gamma \Delta n_t^{\text{stag}} - \lambda \Delta h_t^{-} - \mu \mathbf{1}\{\text{death}\},$$

where n_t^{rabbit} and n_t^{stag} count rabbits and stags killed, Δd_t^{stag} rewards closing distance on the stag while it is jointly vulnerable, and Δh_t^{-} is the agent's health loss on this step. The benchmark score is $J = n_T^{\text{stag}}$, the number of stags killed, rather than the total shaped return. Within a fixed WAD, episode-level variability comes from repeated rabbit spawns over perimeter and interior candidate locations, while the stag follows a fixed center-spawn cycle. At the WAD level, the generator exposes the arena size, alcove layout, stag health and regeneration, the cooperative vulnerability radius, rabbit spawn timing and cap, and the reward magnitudes attached to solo and cooperative outcomes.

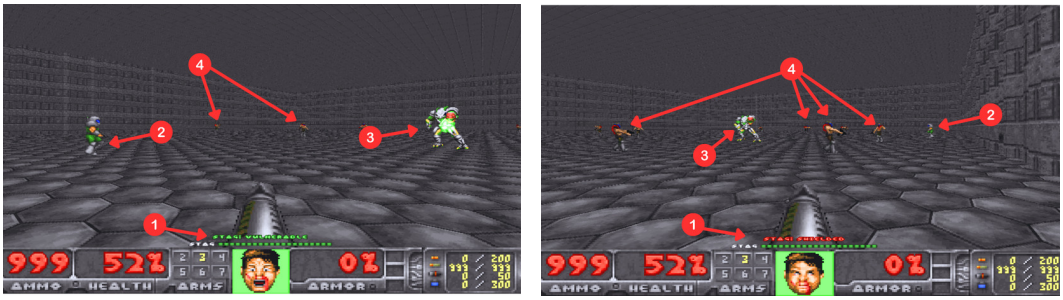


Figure 6: *Stag Hunt Arena*. Two representative views of the coordination dilemma. The arrow annotations indicate: (1) the boss-level stag, which is shielded when the players are too far apart and becomes damageable when they stay sufficiently close to one another; (2) the second player; (3) the boss-level stag; and (4) the small stags.

Rhythm Sync Rhythm Sync is a temporal coordination task in which two separated agents must press paired switches within a shared tolerance window, with unilateral activation causing immediate failure. Timing can be done through a shared periodic visual cue. A concise abstraction is

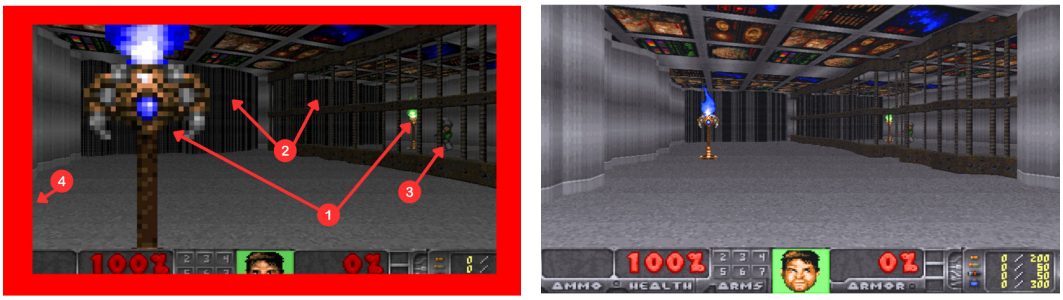
$$r_t = \alpha \Delta c_t - \lambda \mathbf{1}\{\text{mistimed activation}\} + \beta \mathbf{1}\{\text{full completion}\},$$

and the benchmark score is $J = c_T$, the number of stages completed. Within a fixed WAD, each episode resamples switch positions, resamples the cue jitter used in the harder stages, and alternates the cue owner in the single-observer stages according to an episode-specific offset. At the WAD level, the generator controls the number of stages, the split between regular, jittered, and single-observer stages, the beat period and tolerance schedule, and the room and switch geometry.

Foraging Commons Foraging Commons is a shared-resource survival task in which harvesting improves short-term health but depletes the underlying regeneration process, while cleanup restores the commons at an opportunity cost. The shaped reward can be summarized as

$$r_t = \alpha \Delta n_t^{\text{harvest}} + \beta \Delta n_t^{\text{cleanup}} - \lambda \mathbf{1}\{\text{starvation}\},$$

where the benchmark score is survival length $J = T$. Within a fixed WAD, each episode repeatedly samples harvest pickups from random candidate locations based on the current commons state, so the



(a) Arrow annotations indicate: (1) the torch, (2) the door, (3) the second player, and (4) the cue.

(b) The player on the right is waiting for the player on the left to run up to the torch.

Figure 7: *Rhythm Sync*. Two representative views of the temporal synchronization task.

spatial distribution of resources changes over time even when the geometry remains unchanged. At the WAD level, the generator controls the field and cleanup-zone sizes, their separation, the number of agents, the initial regeneration rate, the degradation and restoration dynamics, the health drain, and the density of candidate spawn spots.

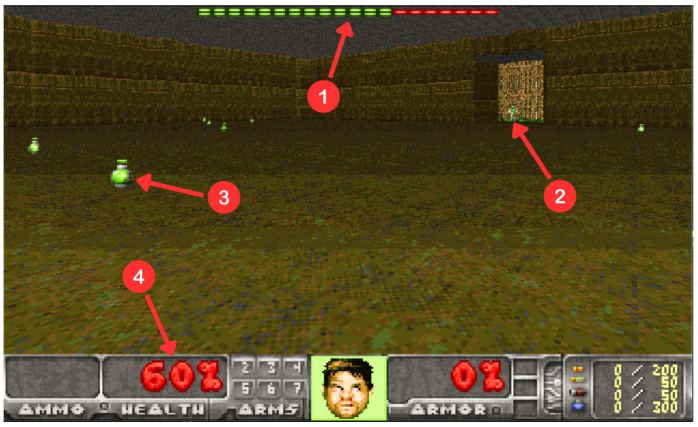


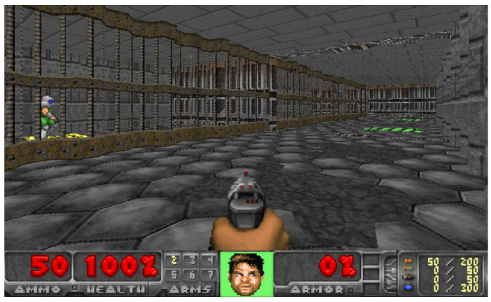
Figure 8: *Foraging Commons*. Arrow markers indicate: (1) the shared spawn-rate bar for harvestable health pots, (2) the second agent standing in the Cleanup Station to restore spawn rate, (3) a harvestable health pot that restores health and yields a small reward, and (4) the agent's own health, which decreases over time. The shaped reward provides a local incentive for both cleanup and harvesting, while the benchmark score is survival time. Policies must therefore balance restoring the commons against immediate collection rather than over-harvest greedily.

Co-op Puzzle Co-op Puzzle is a paired-lane progression task in which agents alternately unlock one another's path through relay plates, with periodic stages that require simultaneous occupancy on both sides. A compact abstraction is

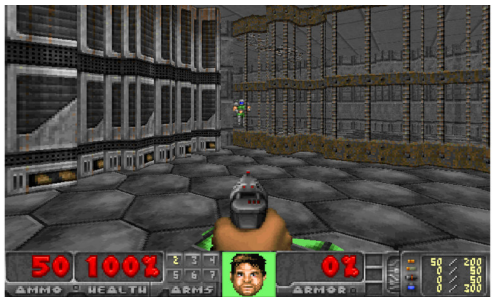
$$r_t = -\eta + \alpha \Delta z_t + \beta \Delta p_t + \gamma \mathbf{1}\{\text{success}\} - \lambda \mathbf{1}\{\text{timeout}\},$$

where z_t is the team-max zone index, p_t is the number of completed zone pairs, and the benchmark score is $J = p_T$. Within a fixed WAD, each episode randomizes which candidate plate is active in each relay zone, while the synchronization zones remain determined by the designed stage schedule. At the WAD level, the generator controls the number of zones, the length and height of each lane, the number and size of candidate plates, the spacing of synchronization stages, and the visual textures used to mark active affordances.

Platform Chain Platform Chain is a tethered vertical traversal task requiring coordinated movement to accumulate monotonic joint progress while avoiding chain overstretch and falls. The reward



(a) The agent on the left is holding the plate for the agent on the right, which opens the door in the right hallway.



(b) The agent on the right is holding the plate to open the door for the agent on the left.

Figure 9: *Co-op Puzzle*. Two representative views of the alternating door-and-plate coordination structure.

structure can be abstracted as

$$r_t = \alpha \Delta q_t - \beta \Delta b_t - \gamma d_t - \lambda \mathbf{1}\{\text{death}\},$$

where q_t is the best joint progress level, b_t counts new chain-break events, d_t counts dragged links, and the benchmark score is $J = \lfloor q_T \rfloor$. Within a fixed WAD, each episode resamples the safe opening length, the per-level motion phase, and the hazard variant assigned to most later checkpoints, so memorizing a single timing pattern is insufficient. At the WAD level, the generator controls the route length, height spacing, map radius, inter-platform spacing, lava damage, chain length, maximum player count, and the number of moving platform hazards.



(a) The two agents are jumping to the next platform together.



(b) The chain is red, meaning one agent is being dragged.

Figure 10: *Platform Chain*. Two representative views of coordinated movement under a tether constraint.

Armory Siege Armory Siege is a defensive combat task in which the team protects a central core while rotating through specialized weapon, ammunition, and medical rooms whose access is periodically gated. A high-level abstraction is

$$r_t = \alpha \Delta k_t + \beta \Delta h_t^+ + \gamma \Delta a_t^+ + \xi \Delta w_t^+ - \lambda \Delta c_t^- - \mu \mathbf{1}\{\text{core destroyed}\} - \nu \mathbf{1}\{\text{agent death}\},$$

where k_t denotes kills, h_t^+ health recovery, a_t^+ ammunition pickup, w_t^+ weapon acquisition, and c_t^- core damage. The benchmark score in this scenario is the cumulative shaped episodic return $J = \sum_t r_t$, so it should be interpreted only within Armory Siege rather than as a task-independent physical quantity. Within a fixed WAD, each episode randomizes enemy wave composition, wave entry side, and the exact spawn positions of pickups within the supply rooms. At the WAD level, the generator controls the distance from the hub to the supply rooms, corridor width, door timing, core health, enemy difficulty, and the maximum number of cooperative players.



(a) Enemies are attacking the core.

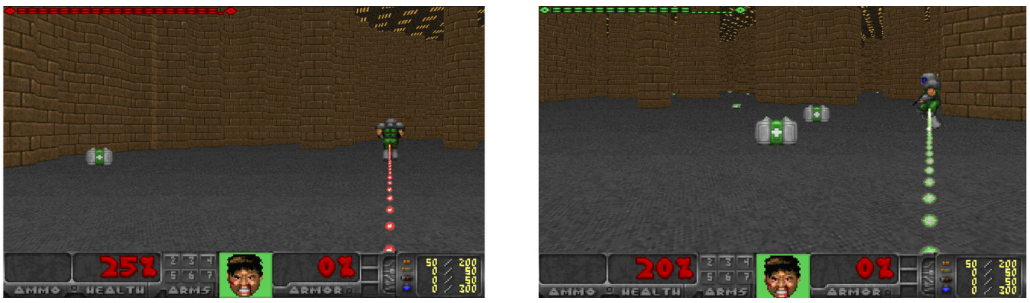
(b) An agent kills an enemy.

Figure 11: *Armory Siege*. Two representative views of the cooperative base-defense scenario.

Co-op Health Gathering Co-op Health Gathering is a cooperative survival task in a toxic maze with a shared tether, where the team must balance exploration, collection, and chain management under persistent environmental damage. A concise abstraction is

$$r_t = \alpha \Delta h_t^+ - \beta \Delta h_t^- + \gamma \Delta m_t - \lambda \Delta s_t - \mu \mathbf{1}\{\text{death}\},$$

where Δh_t^+ and Δh_t^- are health gained and lost to the toxic floor, m_t is the number of newly collected medical kits, s_t counts tether overstretch events, and the benchmark score is survival length $J = T$. Within a fixed WAD, health kits are repeatedly spawned at random candidate positions, so the immediate routing problem changes over time, even when the maze is fixed. At the WAD level, the generator controls the map radius, tether length, toxic-floor damage, the maximum number of available health kits, maze density, and the maximum number of players.



(a) The chain is red, so one agent is being dragged.

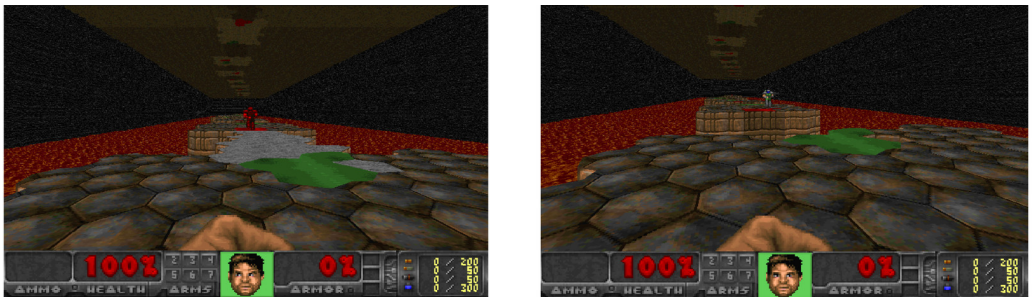
(b) Both agents cooperate to gather health.

Figure 12: *Co-op Health Gathering*. Two representative views of joint survival under tether and health-pressure constraints.

Lava Pit Lava Pit is a reciprocal bridge-handoff task in which one agent must press a plate to keep the bridge on while the other crosses, after which the roles invert before joint progress can continue. A compact abstraction is

$$r_t = -\eta + \alpha \Delta q_t + \sum_{m=1}^{4} \beta_m \Delta b_{m,t} + \gamma \mathbf{1}\{\text{success}\} - \lambda \mathbf{1}\{\text{failure}\} - \mu\, u_t \mathbf{1}\{t = T\},$$

where q_t is the best joint platform index, $b_{m,t}$ are ordered frontier milestones within the current handoff (entry support, outbound bridge, far-side support takeover, and return bridge, each paid at most once per handoff), and u_t is the number of bridges still unfinished. The unfinished-bridge term is a terminal penalty rather than a per-step cost, which the indicator $\mathbf{1}\{t = T\}$ makes explicit: it is charged once, on the final step of the episode. The benchmark score is $J = q_T - 1$, the number of jointly cleared bridges. Within a fixed WAD, later bridges resample the active support plates from safe candidate sets on every episode, while the earliest bridges remain intentionally deterministic to reduce the initial exploration burden. At the WAD level, the generator controls the number of platforms, the platform and gap ranges, the route's lateral curvature, the number of easier-opening bridges, the lava damage, and the optional runtime timing jitter.



(a) One agent is holding the plate so the bridge stays on.

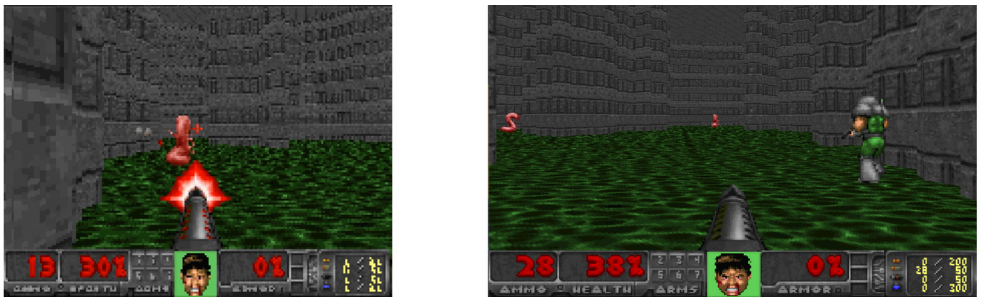
(b) The other agent is not holding the plate, so the bridge collapses and the agent cannot cross.

Figure 13: *Lava Pit*. Two representative views of the reciprocal bridge-handoff mechanic.

Smart Enemies Smart Enemies places the team in a shared toxic arena against fleeing enemies that become harder to finish when several players converge on the same target. A high-level abstraction is

$$r_t = \alpha \Delta k_t + \beta \Delta \mathrm{hit}_t + \gamma \Delta h_t^+ - \lambda \Delta h_t^- - \mu \Delta f_t - \nu f_t - \rho \Delta a_t^-,$$

where Δk_t counts new kills, $\Delta \mathrm{hit}_t$ counts damaging hits landed this step, Δh_t^+ and Δh_t^- are health gained and lost, Δa_t^- is ammunition spent, and f_t measures how many enemies are currently in the strengthened fast state, so the team is penalized both for newly triggering that state (Δf_t) and for allowing it to persist (f_t). The benchmark score is $J = k_T$, the number of kills. Within a fixed WAD, enemy spawn locations are sampled continuously from a pool of candidate spots, and the live enemy state changes online as a function of local player density. At the WAD level, the generator controls the arena radius, obstacle count, and obstacle layout seed, the enemy cap, the spawn interval, the density threshold radius, and the number of players.



(a) An agent shoots an enemy.

(b) Agents stay together to hunt an enemy.

Figure 14: *Smart Enemies*. Two representative views of density-dependent cooperative hunting.

Dumb Enemies Dumb Enemies uses the same arena family as Smart Enemies, but reverses the density-response rule so that coordinated pressure makes fleeing enemies easier to corner rather than harder. The reward abstraction is

$$r_t = \alpha \Delta k_t + \beta \Delta \mathrm{hit}_t + \gamma \Delta h_t^+ - \lambda \Delta h_t^- + \mu \Delta s_t + \nu s_t - \rho \Delta a_t^-,$$

where the combat and health terms carry the same meaning as in Smart Enemies and s_t measures how many enemies are currently in the slowed state, which is now rewarded both on onset (Δs_t) and for persisting (s_t) because slowing enemies is the cooperative behavior this variant is meant to elicit. The benchmark score is again $J = k_T$. Within a fixed WAD, enemy spawns are resampled continuously from random candidate locations, while the slow-state dynamics are induced online by the players' relative positions. At the WAD level, the configurable factors match Smart Enemies: arena size, obstacle layout and count, enemy cap, spawn interval, proximity radius, and player count.

Ammo Carrier Ammo Carrier is a fixed-role asymmetric defense task in which one immobilized defender protects the hub while one or more runners maintain the defender's ammunition supply



Figure 15: *Dumb Enemies*. An agent is about to shoot an enemy without the presence of the other agent.

under enemy pressure. A high-level abstraction is role-dependent:

$$r_t^{\text{def}} = \alpha \Delta k_t + \beta \Delta d_t - \lambda \mathbf{1}\{\text{death}\}, \qquad r_t^{\text{run}} = \gamma \Delta p_t + \delta \omega_t \Delta u_t - \xi \Delta h_t^- - \lambda \mathbf{1}\{\text{death}\},$$

where Δu_t counts completed deliveries, p_t counts useful pickup events, Δd_t is the defender's engagement progress term, Δh_t^- is the runner's health loss, and the weight ω_t increases when defender ammunition is scarce or enemy pressure is high; the benchmark score is survival length $J = T$. This is the one scenario in the suite whose shaped rewards are genuinely role-specific rather than a shared team signal, which is also why it is the scenario where centralized and decentralized credit assignment diverge most sharply (Appendix G). Within a fixed WAD, each episode resamples runner spawn locations, ammunition pack spawns, and enemy wave composition and entry points. At the WAD level, the generator controls the map radius, the number of peripheral depots, the enemy density, and the overall layout seed.



(a) The defender is shooting an enemy while the runner agent collects ammunition.



(b) The shooter agent is about to aim at an enemy.

Figure 16: *Ammo Carrier*. Two representative views of the fixed-role defense-and-resupply interaction.

Stealth Labyrinth Stealth Labyrinth is an asymmetric illumination-and-combat task in which a non-attacking scout reveals rooms for a visually dependent gunner, and combat rooms activate sequentially only after earlier rooms are cleared. A concise abstraction is

$$r_t = \alpha \Delta n_t^{\text{room}} + \beta \Delta k_t - \lambda \Delta h_t^{\text{team},-} - \mu \mathbf{1}\{\text{failure}\},$$

where n_t^{room} counts rooms illuminated and cleared, Δk_t counts kills, and $\Delta h_t^{\text{team},-}$ is health lost by the team as a whole, so that damage taken by either agent is charged to both; an extra first-room discovery bonus applies in the default setting. The benchmark score is

$$J = \frac{\text{enemies destroyed}}{\text{total enemies}}.$$

Within a fixed WAD, the scenario is essentially deterministic once the episode starts: the active room order is fixed by progression, so the primary variability is behavioral rather than procedural. At the WAD level, the generator controls the number of branches, rooms per branch, room and corridor geometry, the enemy count range per room, turret health, and projectile parameters, and the easier first-room settings that reduce initial exploration difficulty.



Figure 17: *Stealth Labyrinth*. The hallway is dark, and the shooter agent is waiting for the runner agent to go first and light up the room.

Lava Maze Lava Maze is an asymmetric communication task in which one player navigates a dynamically carved safe path over lava while another player observes from above and sends color-coded guidance signals. A high-level abstraction is

$$r_t = -\eta + \alpha \Delta \ell_t + \beta \Delta d_t^{\text{path}} - \lambda \Delta h_t^- - \mu \mathbf{1}\{\text{signal use}\} - \nu \mathbf{1}\{\text{death}\} + \kappa \Delta d_t^{\text{guided}},$$

where ℓ_t is the number of completed mazes, Δd_t^{path} is the navigator's progress along the carved safe path, Δh_t^- is health lost to lava, the term $\mu \mathbf{1}\{\text{signal use}\}$ charges the observer a small cost for emitting a signal so that constant broadcasting is not free, and $\kappa \Delta d_t^{\text{guided}}$ pays for the portion of path progress made while a signal was active. Note that κ rewards the *coincidence* of signalling and correct movement without ever fixing which colour should mean which direction, which is exactly what leaves the protocol to be discovered rather than imposed. The benchmark score is $J = \ell_T$, the number of mazes completed in the episode. Within a fixed WAD, every episode samples a fresh start location, and each completion immediately generates a new maze instance, starting at the navigator's position, whose size grows $(8, 10, 14, 18,$ then 20 tiles) until it reaches the configured maximum. The episode ends only when the navigator dies or the time limit is reached, so J has no finite ceiling. J is computed by counting goal relocations, because the in-engine level counter of the released WAD advances twice per completion (the goal is relocated one tic after the completion check). At the WAD level, the generator controls the physical grid size, cell size, initial and maximum maze size, and the severity of lava punishment.



(a) The navigator is at the beginning of the maze, while the observer agent can see everything from above.



(b) The observer agent is shooting to signal going right, matching the yellow color.

Figure 18: *Lava Maze*. Two representative views of the asymmetric navigation-and-signaling task.

B Reproducibility and Experiment Details

B.1 DoomGen Environment Seeding.

Environment layouts generated by DoomGen are fully deterministic given an integer seed. The `ProceduralMapBuilder` accepts an explicit seed argument that controls every stochastic step of Voronoi tessellation, Lloyd relaxation, room grouping, and corridor routing. Offline-generated environments stored in WAD files are fed into `comrad/scenarios` in COMRAD. Therefore, the exact geometric configurations used during training are not stochastic. However, because DoomGen can generate an infinite number of geometric configurations for a scenario with different seeds in batches, and COMRAD can dynamically switch environments during training, a scenario can become nearly spatially generalizable. The exact configurations used in a batch's training process can be tracked in the scenario's `batch_registry.json`.

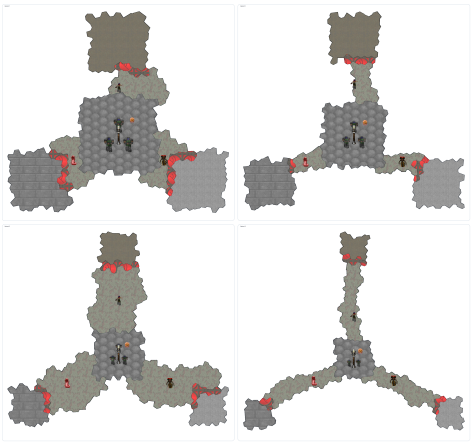


Figure 19: Four *Armory Siege* layouts generated from the same scenario template under different integer seeds and configurations. DoomGen preserves the scenario semantics, namely three outer supply rooms connected to a defended central armory through timed door bottlenecks (indicated by red textures), while varying room geometry, corridor routing, visibility structure, and local traversal distances.

B.2 Within-Scenario Stochasticity.

Episode-level randomness (spawn locations, item timings, cue jitter) is driven by ViZDoom's ACS `Random` function. To ensure the fairness of procedural generation and within-scenario stochasticity, we conducted a statistical validation of the ACS pseudo-random number generator (PRNG) used within the ViZDoom engine.

SFMT Algorithm. ViZDoom uses a SIMD-oriented Fast Mersenne Twister (SFMT) random generator. This algorithm is a successor to the standard Mersenne Twister, optimized for modern SIMD-capable CPUs to provide high-throughput randomness with a period of $2^{19937} - 1$ (Saito & Matsumoto, 2008). SFMT is specifically designed for superior equidistribution properties. This is important for COMRAD because stochasticity minimizes the risk of a policy overfitting to specific environmental variations. Since COMRAD relies on VizDoom's `Random()` for spawning locations, item timing, and *DoomGen* layout parameters, any bias in the PRNG could lead to over-representation of spatial configurations.

Empirical Uniformity. We performed a distribution analysis on $1,000,000$ samples generated within VizDoom. As shown in the left of Figure 20, the observed counts for a discrete uniform range $[0, 9]$ fall within the $\pm 1\sigma$ (9/10 times) and $\pm 2\sigma$ (1/10) confidence bands (where $\sigma = 300.0$ for $n = 10^6$). On the right, a Chi-squared test for uniformity was performed, yielding $p = 0.7965$. We therefore fail to reject the null hypothesis of uniformity. In the right plot, the ECDF shows a maximum deviation of 0.0007 from the theoretical CDF, which is negligible.

B.3 Experimental Setup.

Hardware and configuration. All main-table experiments were launched through the Sample Factory runner with one NVIDIA A100 GPU and 18 CPU cores per job, with 128GB of memory, frame skip $k = 4$, and a training budget of 100M environment steps per run. Every main-table run

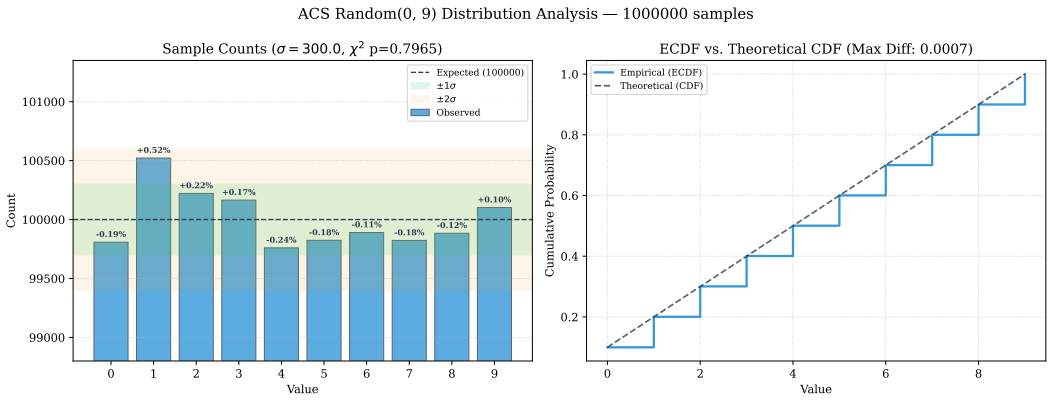


Figure 20: Distribution analysis of ACS `random(0, 9)` using the SFMT PRNG over 10^6 samples. **Left:** Observed sample counts per value relative to the expected frequency $(100,000)$, showing deviations within the confidence bounds ($\sigma = 300.0, p = 0.7965$). **Right:** The Empirical Cumulative Distribution Function (ECDF) compared to the theoretical CDF with a maximum deviation of 0.0007.

uses a team of $N = 2$ agents, including in the scenarios that support larger teams, so that Table 3 is not confounded by the team-size effects of Appendix E. The main text table reports performance at a common 100M-step checkpoint. The agent-count study in Appendix E reuses the same single-seed configuration for $N \in \{2, 3, 4, 6, 8\}$; each job used the same A100 GPU class and 18-core allocation and was evaluated under the same two-hour wall-clock limit, which, as explained there, makes it a fixed-compute comparison.

Shared parameters. Parameter sharing is treated as a comparison control rather than as a universal modeling assumption. IPPO, MAPPO, IDQN, VDN, QMIX, and both QPLEX variants use a single shared agent backbone across teammates, so these methods differ primarily in their optimization rule, critic structure, or joint-value aggregation, rather than in extra per-agent model capacity. This follows standard MARL benchmark practice for exchangeable agents (Yu et al., 2022). However, we deliberately retain the same shared parameterization, even for role-asymmetric COMRAD tasks, so that specialization must emerge from the learning rule and local observations rather than from manually assigning separate networks. HAPPO is the only structural exception: its sequential trust-region update is defined over agent-specific actors, so our implementation keeps per-agent actor pathways separate, along with a centralized critic, within a single policy object (Kuba et al., 2022). For the value-decomposition family, sharing stops at the per-agent utility network, while the mixer alone receives joint context. This keeps the benchmark comparison controlled without violating each algorithm family's intended inductive bias. Full architecture and hyperparameter details are provided in Appendices B.6 and B.4.

Evaluation metric. Algorithm performance is reported using the scenario-level benchmark score J described in Section 4.2, instead of the shaped reward used internally for optimization. This distinction is essential because shaping terms are scenario-specific and are introduced only to densify learning, whereas J measures the cooperative outcome the benchmark is meant to evaluate, such as survival time, completed stages, kills, or monotone joint progress. Appendix A.3 gives the exact definition of J for all 13 scenarios. Main-table comparisons are taken at a common 100M-step checkpoint rather than at each method's best retrospective snapshot, so the benchmark rewards data efficiency and training stability under a fixed experience budget. This budget is expressed in logical environment steps, not in the per-agent FPS metric used by Sample Factory for systems monitoring. All baseline entries in Table 3 and Table 14 are computed from five independent runs of each benchmark configuration, using experiment seeds $\{42, 68, 81, 97, 154\}$, and we report the mean together with the 95% confidence interval half-width across these runs. Because the raw units differ across scenarios, scores are compared directly only within a scenario; whenever we summarize performance across scenarios, we first normalize within task, as in Figure 4 and Appendix J. Any reported uncertainty should therefore be interpreted only within a fixed scenario and protocol, not as a quantity that can be mixed across heterogeneous task units.

B.4 HYPERPARAMETERS AND TRAINING DETAILS

Tables 4 and 5 summarize the effective training settings used by the benchmark launcher. We report only settings that materially affect optimization, credit assignment, or system throughput. Unless noted otherwise, all baselines used 128×72 resized visual observations with input normalization, narrow-aspect rendering, action repeat $k = 4$, Adam optimization, and a training budget of 100M environment steps. Appendix B.5 complements these tables with an explicit learning-rate validation study for the QMIX setting used in our Armory Siege analysis.

Table 4: On-policy training configuration used for IPPO, MAPPO, and HAPPO in the COMRAD baseline suite.

Setting	**IPPO / MAPPO**	**HAPPO**
Training budget	100M env. steps	100M env. steps
Parallel rollout setup	16 actor workers, 8 envs/worker, 2 inference workers	Same
Execution regime	Asynchronous	Asynchronous
Collected batch size	4096	4096
Epochs per collected batch	4	1
Rollout horizon	32 steps	32 steps
Discount factor γ	0.99	0.99
Advantage estimator	GAE with $\lambda = 0.95$	GAE with $\lambda = 0.95$
PPO clipping	0.1	0.1
Optimizer and learning rate	Adam, 10^{-4}	Adam, 10^{-4}
Learning-rate schedule	Constant	Linear decay
Exploration regularization	Symmetric KL, coefficient 10^{-3}	Symmetric KL, coefficient 10^{-3}
Gradient clipping	Max norm 4.0	Max norm 4.0
Recurrent policy	1-layer LSTM, hidden size 512	1-layer GRU, hidden size 512
Maximum policy lag	1000	3000
Reward handling	Scenario-dependent team blending	Scenario-dependent team blending

Table 5: Off-policy training configuration used for IDQN, VDN, QMIX, and QPLEX in the COMRAD baseline suite.

Setting	**IDQN**	**VDN / QMIX**	**QPLEX**
Training budget	100M env. steps	100M env. steps	100M env. steps
Parallel rollout setup	16 actor workers, 8 envs/worker, 2 inference workers	Same	Same
Execution regime	Asynchronous	Asynchronous	Asynchronous
Collected batch size	4096	4096	4096
Replay sample per update	256 transitions	64 recurrent sequences	16 recurrent sequences
Recurrent core	None	1-layer GRU, hidden size 512	1-layer GRU, hidden size 512
Joint value aggregation	Independent per-agent Q-learning	Additive (VDN) or monotonic mixing (QMIX)	Duplex dueling mixing with DMAQ or DMAQ-Qatten
Discount factor γ	0.99	0.99	0.99
Optimizer and learning rate	Adam, 10^{-4}	Adam, 10^{-4}	Adam, 10^{-4}
Replay capacity and warm-up	500K transitions, 50K-step warm-up	Same nominal setting	Same nominal setting
Target-network updates	Every 1000 learner steps, $\tau = 0.005$	Same	Same
Exploration schedule	ϵ annealed from 1.0 to 0.01 over 20M steps	Same	Same
Replay weighting	Prioritized replay	Uniform recurrent sequence replay	Uniform recurrent sequence replay
Replay update cap	At most 4 updates per learner cycle	At most 4 updates per learner cycle	At most 4 updates per learner cycle
QPLEX-specific settings	Not applicable	Not applicable	Gradient-accumulation mini-batch 16; state-value bias disabled
Reward handling	Per-agent shaped rewards	Per-agent shaped rewards scalarized in the learner	Per-agent shaped rewards scalarized in the learner

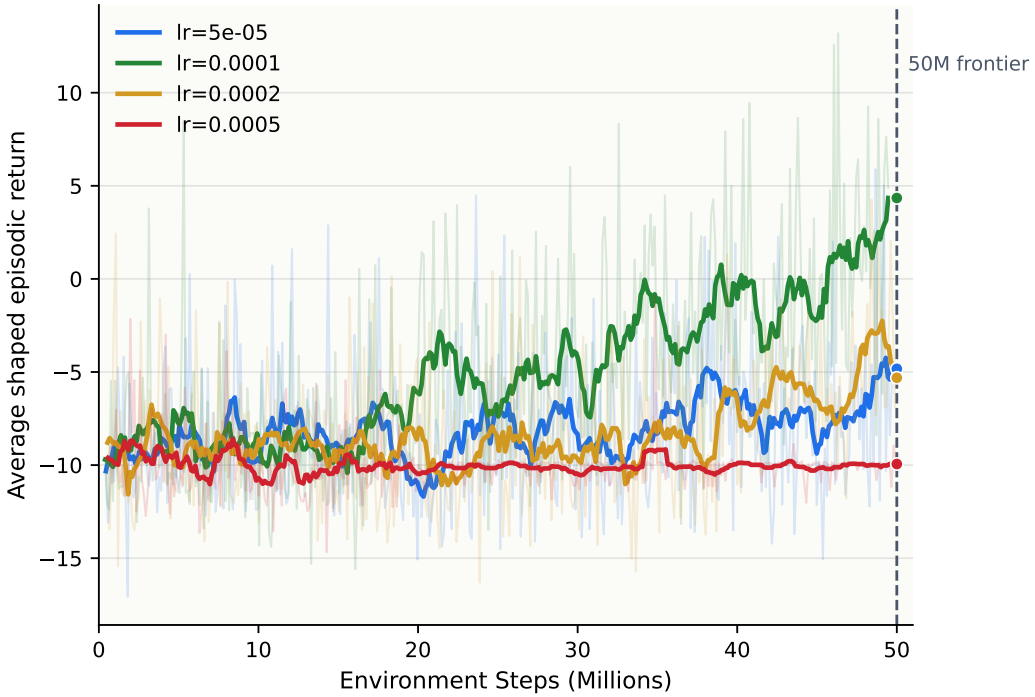


Figure 21: Learning-rate sensitivity on *Armory Siege* with $N = 3$ under a nominal 50M-step pilot budget. The dashed marker indicates the intended 50M frontier. The conservative setting 10^{-4} is the only pilot that reaches a positive near-frontier return.

B.5 Learning-rate sensitivity study.

We selected the QMIX learning rate empirically rather than by adopting a canonical off-policy default. On *Armory Siege* with QMIX and $N = 3$, we ran a single-seed pilot sweeping $\{5 \times 10^{-5}, 10^{-4}, 2 \times 10^{-4}, 5 \times 10^{-4}\}$ while holding the worker layout fixed (16 rollout workers, 8 environments per worker, 2 inference workers), together with the recurrent QMIX settings used in the scaling launcher (`batch_size` $= 2304$, `qmix_sequence_batch_size` $= 32$).

Table 6: Single-seed learning-rate pilot on *Armory Siege* with $N = 3$. All runs shared the same worker layout and recurrent QMIX settings; only the Adam learning rate differed.

Learning Rate	Final Score	Tail FPS
5×10^{-5}	-5.61	12.72K
10^{-4}	**4.88**	12.53K
2×10^{-4}	-6.88	13.27K
5×10^{-4}	-9.90	13.36K

Table 6 and Figure 21 show a clear ranking despite the small residual step mismatch: 10^{-4} is the only setting that reaches a positive final return (4.88), whereas 5×10^{-5} underfits (-5.61) and the larger rates 2×10^{-4} and 5×10^{-4} degrade further (-6.88 and -9.90). Throughput varies only modestly across the sweep (12.53K–13.36K FPS), so the winning setting is not explained by a slower systems path that happened to collect fewer updates. At the same time, these pilots remain single-seed, fixed-wall-clock runs with nontrivial multiplayer synchronization noise, so we use them only to justify retaining the conservative Adam learning rate of 10^{-4} for QMIX rather than to claim a broader hyperparameter optimum.

During training, the code tracks "best" checkpoints using the scenario-level benchmark score after the first 100K environment steps, while the main paper reports the common 100M-step frontier rather

than per-run early stopping. This keeps the comparison tied to a matched training budget instead of to algorithm-specific stopping behavior.

B.6 Network Architecture Details

All visual baselines start from the same observation preprocessing path. In the main benchmark suite, frames are rendered in the standard 4:3 configuration, yielding a 160×120 RGB image, which is then resized to 128×72 and converted to a channel-first format before entering the network. The default visual backbone follows the simple convolutional encoder used in Sample Factory (Petrenko et al., 2020): three convolutional layers with $(32, 64, 128)$ output channels, kernel sizes $(8, 4, 3)$, strides $(4, 2, 2)$, and an ELU nonlinearity after each convolution, followed by a 512-unit fully connected projection. The codebase also supports an auxiliary measurement branch, implemented as a small MLP whose output is concatenated with the visual embedding, but this branch is not enabled by default in the main benchmark.

Let o_i^t denote agent i's resized RGB observation at decision step t, and let $e_i^t = f_{\text{enc}}(o_i^t)$ be the encoder output. Recurrent baselines then maintain a hidden state

$$h_i^t = R_\psi(e_i^t, h_i^{t-1}),$$

where the recurrent cell is algorithm-dependent rather than globally fixed. The benchmark launcher uses a single-layer 512-unit LSTM (Hochreiter & Schmidhuber, 1997) for IPPO and MAPPO, while HAPPO uses the default single-layer 512-unit GRU (Cho et al., 2014). The recurrent off-policy baselines VDN, QMIX, and QPLEX also use a single-layer 512-unit GRU, and configuration validation explicitly forces GRU whenever recurrent value decomposition is selected.

IPPO and MAPPO inherit Sample Factory's shared-weight actor-critic template. A single encoder and recurrent core are shared across agents, then passed through separate policy and value heads. IPPO uses the standard per-agent scalar critic. MAPPO replaces this critic with a centralized value module that concatenates the same-timestep decoder features from all agents,

$$z^t = [d_1^t; \ldots; d_N^t], \qquad [V_1^t, \ldots, V_N^t] = g_{\text{MAPPO}}(z^t),$$

where each d_i^t is the post-recurrent decoder feature for agent i. In the implementation, g_{MAPPO} is a two-hidden-layer MLP with widths 512 and 256 and one output value per agent, so the actor remains decentralized while the critic conditions on joint team context.

HAPPO uses a materially different structure. It still appears as a single policy object to the runner, but that object contains separate encoder, recurrent core, decoder, and action-head modules for each agent, along with a centralized critic. Agent identity is provided as a one-hot routing signal so that each sample is processed by the appropriate subnetwork rather than by a single shared actor. The critic first builds per-agent features c_i^t with dedicated critic encoders, then applies a recurrent critic in the training, groups features by environment step, and predicts a scalar joint value

$$V(s^t) = g_{\text{HAPPO}}([c_1^t; \ldots; c_N^t]),$$

using an MLP with hidden sizes $[512, 256]$. During inference, the code can reuse actor-encoder features for critic bootstrapping to avoid redundant CNN computation; during learning, it executes the full critic pathway.

For the value-decomposition baselines, the per-agent utility network is shared across agents. Because COMRAD uses factorized tuple action spaces, the Q-network emits one linear head per action factor rather than one giant flat head over the Cartesian product. If agent i has M_i action factors and selects sub-actions $a_{i,1}^t, \ldots, a_{i,M_i}^t$, the chosen utility is

$$Q_i(o_i^t, a_i^t) = \sum_{m=1}^{M_i} Q_{i,m}(o_i^t, a_{i,m}^t).$$

The mixing state is not constructed from raw pixels. Instead, the learner concatenates encoder features,

$$s^t = [e_1^t; \ldots; e_N^t],$$

which keeps the joint state dimension tractable for a 128×72 RGB input. This is the only notion of "global state" used anywhere in the benchmark: the centralized modules see only concatenated

egocentric features, never an unoccluded top-down view, a ground-truth object list, or teammate coordinates that the engine withholds from the actors. Partial observability is therefore guaranteed centralized training, it is only pooled across the team. VDN sums per-agent utilities directly (Sunehag et al., 2018). QMIX uses a two-layer hypernetwork mixer with a 32-dimensional mixing embedding and 64-dimensional hypernetwork hidden size, with absolute-valued generated weights enforcing the monotonicity constraint (Rashid et al., 2018). QPLEX is reported in two published mixer variants built on the duplex-dueling factorization of Wang et al. (2021); we treat them as separate evaluated configurations because they instantiate meaningfully different joint-action decomposition mechanisms on top of the same shared utility backbone. Table 7 summarizes these architecture choices compactly.

Table 7: Summary of the neural architectures used in the benchmark.

Family	Agent Backbone	Recurrent Core in Sweep	Joint Module / Output Structure
IPPO	Shared visual encoder and shared actor-critic trunk across agents	1-layer 512-unit LSTM	Decentralized actor and per-agent scalar value head
MAPPO	Shared visual encoder and shared actor-critic trunk across agents	1-layer 512-unit LSTM	Centralized critic MLP on concatenated decoder features, widths 512 and 256, one output per agent
HAPPO	Separate encoder, core, decoder, action modules per agent inside one policy object	1-layer 512-unit GRU actor; critic RNN enabled in launcher	Centralized scalar critic on concatenated per-agent critic features, hidden widths 512 and 256
IDQN	Shared per-agent visual Q-network	None in main sweep (feed-forward)	Independent per-agent Q heads only
VDN, QMIX, QPLEX	Shared per-agent visual utility network; one Q head per action factor	1-layer 512-unit GRU	VDN sum mixer; QMIX hypernetwork mixer; QPLEX DMAQ or Qatten duplex-dueling mixer

B.7 Training Infrastructure

Why an asynchronous CPU pipeline instead of JAX. JAX-based MARL excel when environment dynamics can be written as accelerator-friendly tensor programs and executed natively on a GPU, which is why they work so well for particle environments and gridworld logic. ViZDoom however, does not meet this condition: it is a compiled C/C++ engine relying on CPU-based software rendering, and it can be neither JIT-compiled to XLA nor executed on an accelerator. For CPU-bound environments, the correct architectural pattern is an asynchronous CPU simulation pipeline paired with GPU policy learning, which is precisely what COMRAD uses: rollout workers run many ViZDoom instances in parallel on CPU, while policy workers and the learner batch inference and optimization on GPU. JAX-based frameworks and this therefore address complementary classes of environments.

Launcher and wrapper stack. All benchmark runs are launched through a common experiment driver that registers scenario factories, model factories, and optional curriculum components before delegating execution to Sample Factory's parallel runner (Petrenko et al., 2020). The main benchmark suite fixes one policy per job, 16 rollout workers, 8 environments per worker, 2 inference workers, and a training budget of 100M environment steps. The paper's main results are then reported at the 100M-step checkpoint rather than only at the terminal step. At environment construction time, each scenario passes through the same wrapper stack: resolution selection, resize to 128×72, channel-order conversion, time-limit enforcement, scenario-specific reward shaping, and algorithm-specific coordination aids such as explicit agent-identity signals for HAPPO or team-reward sharing when required by an actor-critic baseline. Reward sharing is applied only to actor-critic baselines and only when enabled by the scenario or launcher; VDN, QMIX, and QPLEX keep per-agent rewards until the learner combines them, matching the assumptions of value decomposition. Table 8 summarizes the training configurations, while Figure 22 illustrates the end-to-end training pipeline.

Each logical multi-agent environment is implemented as a lock-step group of N ViZDoom multiplayer clients. The multiplayer wrapper spawns one worker thread per agent, creates one client connection per thread, and dispatches reset and step tasks to the whole group synchronously. This use of threads is deliberate: the expensive part of environment stepping happens inside ViZDoom's C++ engine

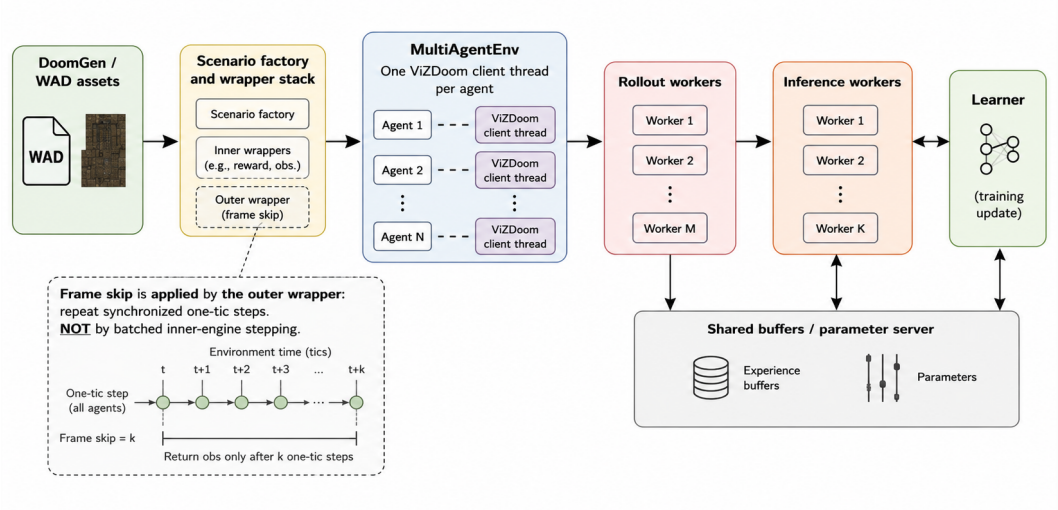


Figure 22: High-level training infrastructure for COMRAD. Each multiplayer scenario is executed as a synchronized group of per-agent ViZDoom clients connected over local UDP. Rollout workers collect trajectories in parallel, inference workers batch and execute the visual policy, and a dedicated learner updates parameters asynchronously. This design makes benchmark throughput a function of environment synchronization latency, batched visual inference speed, and crash-tolerant multiplayer process management.

and local UDP I/O, so these calls can overlap despite Python's GIL, while threads avoid the extra inter-process serialization that would arise if every teammate were placed in a separate Python process. The wrapper also implements robustness logic needed by a multiplayer engine. Because all teammates share a single UDP game, a crashed peer invalidates the entire match; the system therefore tears down the entire group, reserves a fresh port, and reinitializes all peers together. Environment startup is additionally guarded by file locks, both to cap the number of simultaneous ViZDoom initializations and to reserve UDP ports cooperatively across concurrent jobs.

Frame skip is handled at the wrapper level rather than delegated to ViZDoom's built-in multi-tic stepping. In the multi-agent branch, each inner client always advances one game tic at a time, and the outer wrapper repeats the same action for $k = 4$ synchronized tics before returning a new observation. This matters because the engine uses a lockstep UDP multiplayer architecture, so every tic must finish on all peers before the next one begins. The resulting Sample Factory FPS metric, therefore, counts per-agent frames. For a team of size N with frame skip k, the environment-step rate is

$$\text{env_steps/s} = \frac{\text{FPS}}{Nk},$$

which is the throughput normalization used elsewhere in the paper.

Above the environment layer, Sample Factory separates sampling, inference, and learning. Rollout workers own vectors of environments and write observations, rewards, dones, actions, and recurrent states into preallocated shared trajectory buffers. Inference workers batch policy requests across rollout workers, move the resulting observation batch to the policy device, normalize inputs, execute a forward pass, and write actions plus new recurrent states back into shared output buffers. A dedicated learner process then trains concurrently and publishes updated parameters through a versioned parameter server. In asynchronous mode, the buffer manager allocates at least two buffer sets per sampling device, decoupling experience collection from optimization and bounding policy staleness via finite accumulation rather than full synchronous pauses. The same infrastructure also supports curriculum training over multiple map variants. When that option is enabled, the environment layer can hot-swap scenario files and configuration pairs on episode boundaries without restarting the surrounding rollout worker.

Table 8: Launcher configurations for the main benchmark experiment.

Setting	Value in main experiment
Training target	100M steps
Rollout workers	16
Environments per rollout worker	8
Worker buffering	Two buffer partitions per worker group for double buffering
Inference workers per policy	2
Policies per job	1
Frame skip	4, applied by the outer multi-agent wrapper
Hardware	1 NVIDIA A100 GPU, 18 CPU cores, 128GB RAM

Table 9: Deterministic scenario-batch specifications currently bundled with COMRAD, ordered by the difficulty progression from Table 2.

Scenario	Batch Instances	Main variation axes
Stag Hunt Arena	$486\ (=3^5\times 2)$	Arena size, alcove layout, stag threat profile, cooperative vulnerability radius, rabbit pressure, payoff schedule
Rhythm Sync	$243\ (=3^5)$	Stage count, regular/jittered split, beat/tolerance schedule, room geometry, switch-layout jitter
Foraging Commons	$729\ (=3^6)$	Field/cleanup geometry, separation, agent count, commons dynamics, health drain, spawn-spot density
Co-op Puzzle	$162\ (=3^4\times 2)$	Zone count, lane geometry, candidate plate count/size, sync frequency, affordance color theme
Platform Chain	$243\ (=3^5)$	Route length, opening safety margin, mover density, tether slack, lava severity
Armory Siege	$1,215\ (=3^5\times 5)$	Room distance, corridor width, door timing, core health budget, enemy pressure, team size
Coop Health Gathering	$729\ (=3^6)$	Map radius, tether slack, toxic-floor damage, health-kit cap, maze density, team size
Lava Pit	$486\ (=3^5\times 2)$	Platform count, platform/gap profile, route curvature, opening bootstrap preset, lava severity, timing jitter
Smart Enemies	$729\ (=3^6)$	Arena size, obstacle layout/count preset, enemy cap, spawn interval, proximity radius, team size
Dumb Enemies	$729\ (=3^6)$	Arena size, obstacle layout/count preset, enemy cap, spawn interval, proximity radius, team size
Ammo Carrier	$81\ (=3^4)$	Map radius, depot count, enemy density, layout seed
Stealth Labyrinth	$486\ (=3^5\times 2)$	Branch count, rooms per branch, room/corridor geometry, enemy count per room, turret strength, first-room bootstrap
Lava Maze	$81\ (=3^4)$	Physical grid size, cell size, maze growth schedule, lava severity

C Training on Pre-Generated Layout Batches

The motivation for pre-generated batches is to separate two questions that a single static map conflates: whether a policy has discovered the coordination pattern itself, and whether it has merely memorized one convenient spatial realization (Cobbe et al., 2019; 2020). For a given scenario, DoomGen can materialize a finite task pool $\mathcal{B} = \{\omega_1, \ldots, \omega_K\}$ of deterministic WAD instances. At episode e, training samples one instance $\omega_e \sim p_e$ and keeps it fixed for the full episode. Uniform batch training corresponds to a fixed p_e, while curriculum variants adapt p_e online. Episode-boundary swapping is important because each collected trajectory then has a single unambiguous generating instance, which keeps learner-side attribution clean when statistics are routed back to the corresponding task. Table 9 summarizes the deterministic batch specifications currently shipped with the repository.

The current paper instantiates this interface in the *Platform Chain* curriculum study of Section D.1. There, direct training uses only the default hard target, whereas the remaining five curriculum variants train on a separate pool and are then evaluated on the default hard task. This comparison already shows why pre-generated batches matter. Uniform exposure to the pool reaches a comparable common-target score much earlier than direct hard training, achieving 5.62 checkpoints at 75.0M steps versus 5.44 only after 124.3M steps, and adaptive controllers further improve either early discovery or peak retained target performance (Table 11; Figure 23). At the same time, all variants remain far below the 47-checkpoint maximum. The correct conclusion is therefore that batch training materially improves learning dynamics and provides a controlled substrate for curriculum research on hard embodied MARL problems.

D CURRICULUM BENCHMARK SETUP AND IMPLEMENTATION DETAILS

D.1 AUTOMATED CURRICULUM LEARNING

Learning from scratch on high-difficulty visual MARL scenarios is often infeasible due to delayed task signals. To mitigate this, COMRAD provides an Automated Curriculum Learning (ACL) framework, inspired by the curriculum abstractions of *Syllabus* (Sullivan et al., 2025) and adapted for pre-generated layout batches. This ensures agents are consistently presented with tasks at the frontier of their current capabilities.

Before training, DoomGen generates map variants across multiple difficulty tiers, decoupling layout diversity from mechanical difficulty (Section 4.4). A curriculum controller then maintains a dynamic sampling distribution over these instances, adjusting the probability of sampling each task based on real-time learner metrics. COMRAD supports several curriculum strategies that avoid the need for static progressions: uniform sampling, sequential progression, learning-progress heuristics, OMNI-style dependency masking, and Prioritized Level Replay (PLR). Table 10 provides an overview, and algorithmic details are provided in Appendix D.3.

Table 10: Implementation specifics for curriculum variants in COMRAD. All variants operate on the same discrete pool of tasks ($\mathcal{T}$) and differ mathematically in how they update the sampling distribution $p_e(\tau)$.

Variant	Update Signal	Algorithmic Behavior
Direct Training	None	Static sampling of the target (endgame) instance.
Uniform	Exposure counts	$p_e(\tau_i) = 1/K$. Isolates the effect of instance diversity without adaptive selection.
Sequential	Rolling mean success	Advances strictly ordered tasks when $\frac{1}{W}\sum y_i \geq \eta$.
Learning Progress	Fast/slow moving averages	Prioritizes tasks with the highest absolute disagreement between timescale updates.
OMNI-style	LP + dependency mask	Applies a strict frontier-ordering graph mask to suppress mastered or unreached tasks.
PLR	Learner-side statistics	Replays tasks based on estimated learning potential (e.g., value error) blended with staleness.

Difficulty in COMRAD varies along two axes introduced in Section 4.4, which makes the curriculum machinery useful for precisely determining whether a method is limited by mechanical exploration, spatial generalization, or both. A researcher can hold task mechanics fixed and broaden layout diversity, hold layouts fixed and scale the scenario mechanics, or change both jointly.

D.2 CURRICULUM BENCHMARK SETUP

As outlined in Appendix D.1, COMRAD formulates curriculum learning as a benchmark-level mechanism over discrete, pre-generated task instances drawn from the layout-batch training setup of Appendix C. Formally, a curriculum task pool is a finite set $\mathcal{T} = \{\tau_1, \ldots, \tau_K\}$ that preserves core task semantics while varying parameters like spatial layout, route length, and hazard intensity. At episode e, an instance $\tau_e \sim p_e(\tau)$ is sampled from the curriculum-controlled distribution p_e and used for the duration of the episode.

Due to the significant variance in score ranges across task instances, raw performance cannot be compared directly. For instance, direct MAPPO training achieves only 4.78 units of progress on the *Platform Chain* task, whereas the maximum raw score is 47. To address this, COMRAD incorporates task-specific normalization metadata, transforming the terminal benchmark score m_e into a standardized success signal:

$$y_e = \text{clip}\left(\frac{m_e - b(\tau_e)}{s(\tau_e)}, 0, 1\right), \tag{1}$$

where $s(\tau_e) > 0$ is the task-specific scale and $b(\tau_e)$ is an optional offset. This normalization ensures that longer or higher-scoring tasks do not dominate the curriculum update signal simply due to numerical scale.

D.3 CONTROLLER IMPLEMENTATIONS

Sequential Progression. Tasks are strictly ordered, and training concentrates on one instance at a time. Let W denote the window size and η the promotion threshold. The controller advances from

task j to task $j+1$ only when the rolling mean success satisfies:

$$\frac{1}{W}\sum_{i=e-W+1}^{e} y_i \geq \eta. \tag{2}$$

For the *Platform Chain* profile, we set $\eta = 0.6$ and $W = 50$, ensuring promotion depends on consistent competence rather than isolated lucky trajectories.

Learning Progress (LP). COMRAD maintains fast and slow exponential moving averages, f_j and s_j, of the normalized success for each task j (Graves et al., 2017). Using a mastery reweighting transform $u(p) = \frac{p(1-\theta)}{p+\theta(1-2p)}$, it prioritizes tasks by the absolute disagreement between timescales:

$$p_j \propto (1-\lambda)\,|u(f_j) - u(s_j)| + \lambda\frac{1}{K}, \tag{3}$$

where λ is a fixed uniform floor. We use $\lambda = 0.25$ and a mastery reweighting constant $\theta = 0.1$, with the fast and slow averages differing only in their smoothing rate. The distribution remains uniformly flat until every task is sampled at least once, preventing early collapse, and the uniform floor ensures that temporarily flat tasks never become mathematically unsampleable.

OMNI-style Dependency Masking. This variant builds on the LP controller by applying an "interestingness" mask over a predefined task graph. For the *Platform Chain* curriculum task pool, this graph enforces a strict frontier ordering: harder tasks keep their immediate harder successors eligible while dynamically zeroing out the weights of early warm-up tasks once later stages become solvable.

Prioritized Level Replay (PLR). Rather than episodic success, PLR (Jiang et al., 2021) consumes learner-side trajectory statistics (e.g., mean absolute value error, advantage magnitude, or policy entropy). Replay weights are computed by transforming task scores q_j and interpolating them with task staleness σ_j:

$$p_j \propto (1-\beta)\,\phi(q_j) + \beta\,\psi(\sigma_j), \tag{4}$$

where β controls the score-staleness tradeoff. We use $\beta = 0.1$, a rank-based score transform ϕ under a temperature of 0.1, and a power-law staleness transform ψ. PLR is currently integrated with PPO-style actor-critic learners without V-trace, extracting replay scores directly from on-policy statistics rather than off-policy buffers.

D.4 Systems Integration and Evaluation Protocol

At the systems level, task selection strictly occurs at episode boundaries, allowing scenario instances to be hot-swapped without rebuilding the learning stack. The active task identity is logged directly into the trajectories, so learner-side curricula (like PLR) can accurately map statistics back to the generating task. Curriculum state is also natively checkpointed; resuming a preempted training job restores the exact $p_e(\tau)$ distribution to maintain experiment integrity.

Protocol and budget. All six curriculum variants share one configuration, which differs from the main baseline protocol in two respects: the base learner is MAPPO throughout, chosen because PLR consumes on-policy learner statistics and because MAPPO is the strongest general-purpose baseline in Table 3, and the training budget is 125M environment steps.

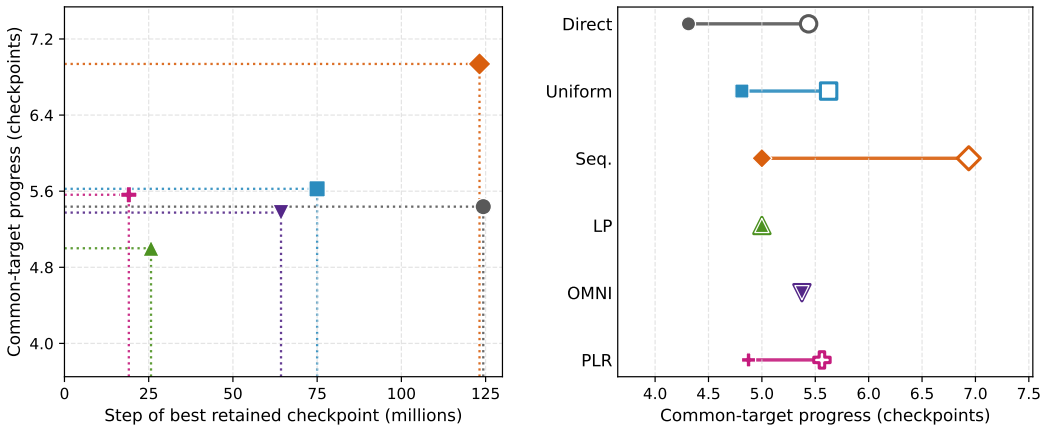
Table 11 summarizes common-target evaluations on the default *Platform Chain* task. The curricula improve learning dynamics significantly more than final task saturation. PLR is the strongest early method, reaching 5.56 checkpoints by only 19.1M training steps. By contrast, direct hard training does not exceed 5.44 until 124.3M steps. Sequential progression achieves the highest observed target score at $6.94/47$, but its final checkpoint falls back to 5.00. OMNI yields the strongest last retained checkpoint at $5.38/47$. Even so, all variants remain far below the 47-checkpoint maximum, indicating that curriculum learning helps agents discover transferable coordination conventions without solving the task outright.

Figure 23 instantiates this ACL protocol as a standardized six-way comparison on *Platform Chain*. The left panel isolates *sample efficiency*: PLR produces the earliest strong common-target score,

Table 11: Common-target evaluation on the default *Platform Chain* task (maximum progress = 47).

Variant	Best step (M)	Best target progress	Last retained progress
Direct hard	124.3	5.44	4.31
Uniform	75.0	5.62	4.81
Sequential	123.2	**6.94**	5.00
Learning progress	25.7	5.00	5.00
OMNI	64.3	5.38	**5.38**
PLR	19.1	5.56	4.88

while the right panel isolates *retention*: sequential progression gives the highest peak, but OMNI preserves the strongest final retained checkpoint. This makes the figure a useful benchmark and an audit of when a curriculum first produces transferable competence and how much of that competence persists at the end of training.



(a) **Best retained checkpoint.** Each marker shows the strongest common-target evaluation score recorded for one curriculum variant. Dotted guide lines project each point onto the step and score axes, making sample-efficiency differences explicit.

(b) **Best versus last retained checkpoint.** Hollow markers denote the best retained checkpoint; filled markers denote the final retained checkpoint. The horizontal segment visualizes how much common-target performance was retained or lost over the remainder of training.

Figure 23: Retained-checkpoint comparison for the default *Platform Chain* curriculum task pool. Because the training protocol retains only the best checkpoint and the last two periodic saves, these panels summarize representative behavior rather than dense equal-horizon learning curves. **(a)** Sample efficiency: PLR produces the earliest strong common-target score, reaching a competitive checkpoint at only 19.1M steps. **(b)** Retention: sequential progression achieves the highest peak score (6.94), while OMNI preserves the strongest final retained checkpoint (5.38), demonstrating a sample-efficiency versus retention tradeoff across curriculum strategies.

D.5 Stratified Ablation Across the Two Difficulty Axes

The curriculum study above moves layout and scenario parameters together, so it cannot tell us whether the two axes of Section 4.4 are genuinely separable controls or merely two names for one underlying notion of difficulty. This subsection isolates them.

Protocol. We run a controlled, stratified 3×3 evaluation on *Armory Siege*, the scenario that exposes both axes most cleanly: `enemy_counts` $\in \{5, 10, 15, 20, 30\}$ controls task mechanics, while `corridor_length` $\in \{5, 10, 15, 30\}$ controls spatial layout. DoomGen creates a map variant for each (`enemy_count`, `corridor_length`) pair: $(5, 5)$ is the joint-easy corner, while $(5, 30)$ isolates spatial difficulty at fixed mechanics. Each cell bins the two axes into Easy/Medium/Hard tiers and reports the mean benchmark score $\pm$ 95% CI half-width across five independent evaluation

runs of 50 episodes each. The *same* QMIX checkpoint is used in every cell, and only the held-out map variant changes between cells, so any difference is attributable to the environment.

Table 12: Stratified difficulty experiment on *Armory Siege* held-out maps. **Top:** mean benchmark score $\pm$ 95% CI half-width for each (spatial, mechanical) tier pair, across five evaluation runs of 50 episodes using one fixed QMIX checkpoint. **Bottom:** the same cells re-binned by *combined* difficulty $d \in \{0, \ldots, 4\}$, the sum of the spatial and mechanical tier indices (Easy $= 0$, Medium $= 1$, Hard $= 2$). Because a single checkpoint is evaluated, the intervals reflect evaluation stochasticity rather than training-seed variance.

Spatial \ Mechanical	**Easy**	**Medium**	**Hard**
Easy	20.45 ± 3.17	17.98 ± 1.85	18.10 ± 1.79
Medium	22.88 ± 2.88	20.04 ± 1.12	21.12 ± 2.14
Hard	17.66 ± 1.14	15.68 ± 0.62	18.98 ± 2.13

Combined difficulty d	**0 (E/E)**	**1**	**2**	**3**	**4 (H/H)**
Mean benchmark score	20.45	20.43	18.60	18.40	18.98

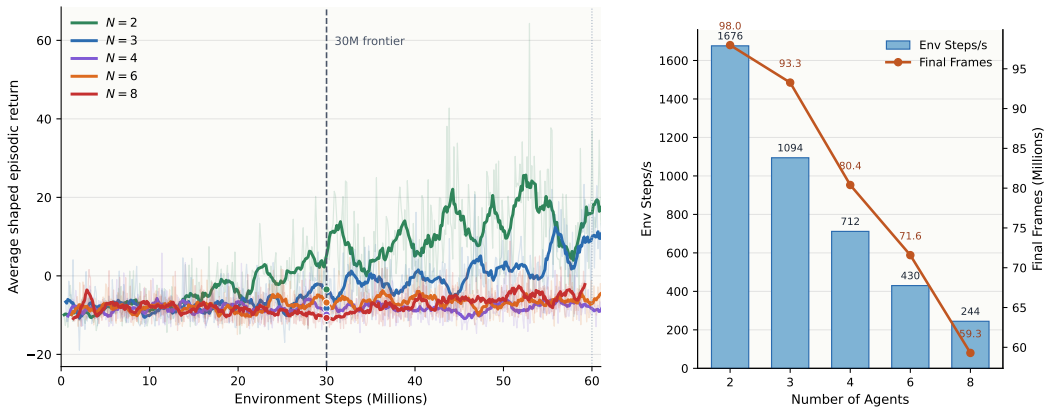
Findings. Table 12 supports three conclusions. First, the axes are separable: moving along one axis changes performance while the other is held fixed, which is precisely the property that makes them usable as independent curriculum knobs. Second, performance degrades with combined difficulty, from ≈ 20.45 in the joint-easy region to ≈ 18.60 once both axes reach their harder tiers. Third, and most usefully for algorithm design, the spatial axis is the dominant driver: averaged over mechanical tiers, moving spatial Easy $\rightarrow$ Hard costs 1.40 points, whereas the corresponding move along the mechanical axis costs only 0.93, and the global minimum (15.68) sits in the hardest spatial tier. This supports the claim in Section 4.4 that spatial layout is a primary bottleneck in the egocentric setting, and it is the reason COMRAD's curriculum learning is designed to include the spatial axis (Portelas et al., 2020).

E SCALING NUMBER OF AGENTS

Scaling team size is a critical axis of benchmark evaluation. Cooperative policies must remain effective as the number of teammates increases (Mahajan et al., 2022; Zhang et al., 2023). In COMRAD, increasing N fundamentally alters the task. It tightens resource contention, creates denser combat dynamics, and exponentially expands the joint action space (reaching $|\mathcal{A}_{\text{joint}}| \approx 1.8 \times 10^{13}$ at $N = 8$).

To quantify these dynamics, we ran a single-seed QMIX pilot on *Armory Siege* across $N \in \{2, 3, 4, 6, 8\}$ under matched hardware and wall-clock controls. To isolate the effect of team sizes, all runs used an identical configuration: the same network architecture, random seed, and worker layout (16 environment workers, 8 environments per worker, 2 inference workers). The shared optimizer settings follow Appendix B.4, and the QMIX learning-rate choice is justified separately in Appendix B.5. Compute was fixed to one NVIDIA A100 GPU and 18 CPU cores per run. Each training session targeted 60M environment steps but was evaluated under a hard, shared two-hour wall-clock limit to measure practical throughput. Because team size also changes rendering cost and multiplayer stability, we treat this appendix as a systems-oriented scaling diagnostic rather than a clean causal scaling law.

Figure 24 highlights how larger teams exacerbate two major challenges. First, larger teams are harder to train efficiently at the shared 30M frames (Figure 24a). Within this protocol, only $N = 2$ exhibits meaningful coordination behavior by that point, whereas $N \geq 4$ policies fail to achieve positive final scores. Second, larger teams incur heavy computational system penalties (Figure 24b). Because rendering overhead scales with N, normalized throughput falls by $6.9\times$ between $N = 2$ and $N = 8$. This is accompanied by a $4\times$ increase in engine synchronization timeouts. Scaling team size in COMRAD, therefore, exposes a dual vulnerability: harder coordination tasks compounded by severe wall-clock friction.



(a) Learning curves for average shaped episodic return, plotted against Sample Factory frames. The common 30M-frame is marked explicitly.

(b) Final collected frames and normalized environment steps per second at the shared two-hour timeout.

Figure 24: Controlled agent-count scaling on *Armory Siege* (QMIX). Larger teams suffer from both degraded sample efficiency and severely reduced wall-clock throughput.

The empirical results (Table 13) reveal two distinct scaling challenges. First, the performance collapse at larger N is not only a wall-clock artifact: at the common 30M-frame budget, $N = 2$ leads with a return of -3.5, while $N = 8$ stagnates at -10.7, and by the end of training the best observed score drops from 64.3 at $N = 2$ to single digits for all $N \geq 4$.

Second, raw FPS (measured in per-agent frames by Sample Factory) masks the true system overhead of larger teams. Using our normalized metric (Env Steps/s), throughput falls from $1{,}676.2$ to 244.5 between $N = 2$ and $N = 8$, a 85.4% reduction, equivalently a $6.9\times$ slowdown. Furthermore, engine stability degrades under heavy multi-agent rendering load, with synchronization timeouts increasing monotonically from 266 ($N = 2$) to 1056 ($N = 8$).

Table 13: Controlled agent-count scaling results under a fixed two-hour wall-clock budget. All runs used identical hardware, hyperparameters, and a target of 60M frames. *Score @ 30M* and *Final Score* are average shaped episodic returns at the common 30M frames and at run termination, respectively. Frames are Sample Factory's per-agent unit that includes frame skips, so the 30M equalizes learner-side transitions across N but corresponds to fewer environment steps per agent as N grows; *Env Steps/s* is the corresponding normalized throughput, FPS$/(N \cdot k)$ with frame skip $k = 4$. Single seed per setting, so these are diagnostics rather than seeded estimates.

N	Mean FPS	Env Steps/s	Score @ 30M	Final Score	Best Score
2	13.4K	1676.2	-3.5	28.5	64.3
3	13.1K	1094.4	-8.3	6.2	41.0
4	11.4K	711.7	-9.9	-2.4	5.8
6	10.3K	430.1	-6.8	-5.4	4.7
8	7.8K	244.5	-10.7	-1.4	8.9

We recommend that future COMRAD scaling studies report both common-step and fixed-wall-clock comparisons, as increasing N simultaneously raises the coordination difficulty and reduces the volume of experience a system can collect within a realistic compute budget.

F Why Methods Succeed and Fail: Extended Analysis

This appendix expands the summary in Section 5.4 and analyzes the three primary performance patterns below; Appendix G then examines the first of them behaviorally on *Ammo Carrier*.

Centralized critics improve synchronization but hurt asymmetric roles. IPPO and MAPPO share identical actor architectures and nearly all PPO settings, differing only in MAPPO's centralized value function. MAPPO leads on tasks requiring tightly coupled progress: *Rhythm Sync* (0.67 vs. 0.57) and *Co-op Puzzle* (3.82 vs. 2.98). In these settings, an individual agent's observation cannot predict team success on its own, so conditioning on joint team states provides a more informative baseline. Conversely, IPPO substantially outperforms MAPPO when agents must specialize in distinct roles: in *Ammo Carrier*, IPPO outscores MAPPO more than fivefold (7031 vs. 1351). Under partial observability and role asymmetry, concatenating joint features into a single global critic can introduce bias or variance (Lyu et al., 2024), and parameter sharing mixes up gradients across behaviorally heterogeneous agents (Christianos et al., 2021). Appendix G shows this directly: IPPO's runner agent quickly discovers an effective depot-to-hub supply route, whereas MAPPO's runner fails to specialize. The culprit is the *staticness* of the critic, and modulating critic inputs according to local, role-specific, or team context could solve this in future work.

Replay buffers help long-horizon credit assignment, but introduce teammate non-stationarity. QMIX achieves the highest scores on tasks where agents must survive long horizon and make infrequent but critical decisions, such as sharing ammo only once a teammate is depleted. By storing past trajectories in a replay buffer, recurrent value-decomposition methods can revisit sparse, high-value coordination events across extended horizons. However, off-policy value-based methods collapse on precise synchronization tasks that MAPPO handles well, like *Rhythm Sync* and *Co-op Puzzle*. Replay buffers retain trajectories from older teammate policies, a known source of non-stationarity in multi-agent Q-learning (Foerster et al., 2017), and stale teammate behavior is most detrimental exactly when success requires reacting to a partner's instantaneous actions within narrow timing windows. Furthermore, monotonic mixing constraints prevent QMIX from representing value functions where an agent's optimal action depends non-monotonically on teammate actions (Rashid et al., 2020), the exact structure of a synchronization task. Addressing this trade-off may require policy-age conditioning/correction, together with explicit spatial or teammate memory for first-person vision. For larger teams, set-attention or graph critics are likely more suitable than fixed observation concatenation.

Architectural expressiveness does not guarantee higher performance. HAPPO and QPLEX provide stronger theoretical guarantees than MAPPO and QMIX, yet both struggle in several visual scenarios. HAPPO yields negative returns on *Armory Siege* (-0.85), and QPLEX-D achieves near-zero performance across five tasks. HAPPO's sequential updates is designed for separate heterogeneous policies, which adds optimization overhead under visual inputs, while QPLEX's duplex-dueling factorization introduces complex advantage and transformation networks that are difficult to train stably through convolutional backbones. Theoretical expressiveness does not translate into practical sample efficiency when learning high-dimensional representations under computational constraints. We therefore caution against assuming a richer coordination architecture will perform better; for complex visual coordination, hierarchical event-level credit assignment over meaningful units (such as handoffs, room clearing, supply loops) or grounded communication channels is likely more effective than simply adding complexity to the mixing networks.

Summary of algorithmic trade-offs. Across all 13 scenarios, performance is primarily decided by the **scope of credit assignment**, not the on-policy/off-policy split on its own. Algorithms that maintain decentralized credit assignment, such as IPPO and IDQN, support sustained role specialization, which explains why IPPO dominates on *Ammo Carrier* and *Dumb Enemies* and why IDQN outscores QMIX on *Ammo Carrier*. Conversely, methods that pool credit across the team, such as MAPPO via its centralized critic and QMIX via its mixing network, excel when success requires tightly coupled progress or shared resource budgeting. The on-policy versus off-policy then determines **which** of these two regimes a method can reach at all (e.g. experience replay helps long-horizon sample reuse on *Foraging Commons* and *Armory Siege*, but introduces non-stationarity that hinders off-policy methods on narrow-window synchronization tasks). Because COMRAD isolates these algorithmic bottlenecks through its taxonomy and difficulty axes, it is a clean tool for testing future directions without breaking changes to the benchmark. This framework also directly explains the unsolved tier, where *Stealth Labyrinth* and *Lava Maze* requires both asymmetric specialization and emergent communication under extreme visual occlusion (Appendix H localizes the failure in these two

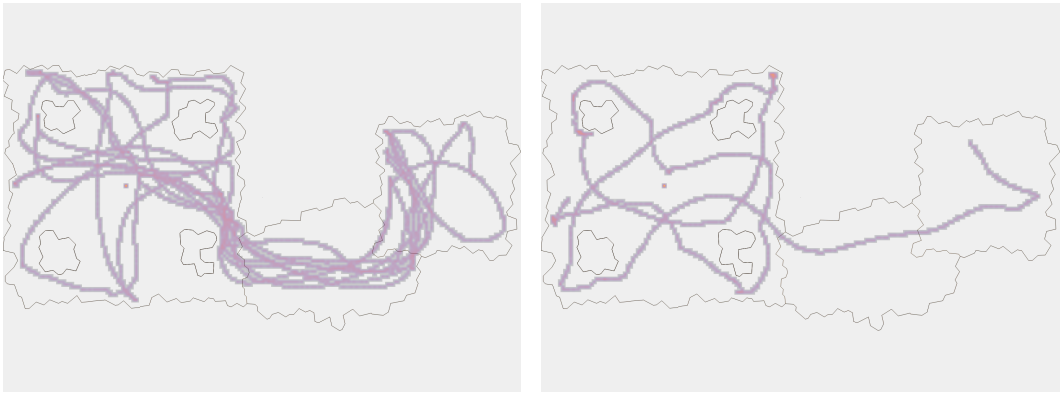
scenarios). The remaining gap is therefore coordination, not perception, control, or task impossibility, and the difficulty axes of Appendix D.5 let future work further isolate its spatial component directly.

G Behavioral Analysis: When Centralized Training Hurts

Aggregate benchmark scores reveal *which* algorithms perform best, while behavioral analysis helps explain *why* coordination succeeds or fails in a given scenario. We therefore study *Ammo Carrier*, the fixed-role asymmetric defense-and-resupply scenario that produces the largest reversal in the benchmark: decentralized IPPO substantially outperforms every CTDE baseline in Table 3.

Task structure. *Ammo Carrier* assigns agents to permanently differentiated roles: a stationary defender must hold the hub under enemy pressure, while a mobile runner must sustain a repeated depot-to-hub resupply loop. The central challenge is whether the runner learns a stable long-horizon logistics routine whose payoff is delayed and role-specific, rather than fine-grained handoff timing.

Behavioral signature of failure. To visualize this qualitatively, Figure 25 compares occupancy density heatmaps from representative rollout trajectories. IPPO concentrates movement along a recurring depot-to-hub corridor, consistent with a learned resupply cycle. HAPPO, by contrast, shows a much less regular pattern: the runner intermittently abandons the logistics routine to move toward the defender or revisits depots in irregular patterns. The result is episodic ammo starvation and substantially lower survival times.



(a) IPPO. The runner repeatedly traverses the corridor between the depot region and the defender hub, consistent with a learned resupply loop.

(b) HAPPO. Movement does not organize into the depot-to-hub delivery pattern, consistent with failed resupply behavior.

Figure 25: Agent trajectory density heatmaps on *Ammo Carrier*, overlaid on the map automap, recorded from each method's best-checkpoint rollout. IPPO (left) shows dense occupancy along the hub-to-depot corridor, consistent with a learned resupply loop. HAPPO (right) lacks this recurring delivery structure, suggesting it fails to learn the role-specialized logistics behavior required by the task. Because carrier spawn positions are randomized each episode, the figure should be interpreted as a representative behavioral diagnostic rather than a proof of causality.

Interpretation. A reasonable explanation is that strict role asymmetry changes what effective coordination looks like under optimization. In tightly coupled joint tasks, centralized training and shared optimization pressure can help agents align on moment-to-moment joint behavior. In *Ammo Carrier*, however, the optimal joint policy also requires one agent to sustain an independent long-horizon logistics loop whose payoff is delayed and role-specific. Centralized updates may therefore introduce counterproductive coordination pressure that disrupts this independence: the runner's gradient updates are influenced by the defender's value signal in a way that interferes with the sustained logistics commitment the scenario demands. A decentralized update, by contrast, allows the runner to optimize its loop policy without cross-agent interference. We treat this as an informed interpretation rather than a causal proof; critic-scope, reward-sharing, and role-factorization ablations would be needed to isolate the effect.

Even so, this failure mode exposes a recurring Cooperative AI challenge: CTDE methods optimized for tightly coupled joint credit assignment may be poorly suited to *role-specialized understanding/trust* settings, where effective cooperation depends less on tighter joint coupling than on learning stable complementary behavioral contracts. Making this failure class behaviorally visible, rather than merely observable as lower return, is one of the contributions of COMRAD.

H ABLATION STUDIES ON *Very Difficult* SCENARIOS

The two *Very Difficult* scenarios, *Stealth Labyrinth* and *Lava Maze*, produce near-zero benchmark scores across all eight evaluated baselines at the 100M-step budget. Before accepting this as evidence of a fundamental algorithmic limitation, it is necessary to rule out three alternative explanations: insufficient reward shaping, inadequate training budget, or a fundamental flaw in scenario design. This appendix addresses each concern through controlled task simplifications that preserve the core coordination mechanism while systematically reducing a single, targeted source of complexity. The goal is to show that if a simplified variant is learnable and the full version is not, the difficulty is structural and algorithmic rather than incidental.

Stealth Labyrinth The full scenario requires a scout to light up sequential dark rooms while a gunner engages enemies; combat rooms activate only after the prior room is fully cleared. This creates a compounding credit assignment problem: the correct joint policy at room k yields a learning signal only if room $k-1$ was already solved cooperatively. This creates a temporal dependency chain in which early coordination failures produce zero gradient for all later stages. A stable local optimum exists in which both agents remain in the first lit room indefinitely.

Stealth Labyrinth Simple simplifies this to a single room with a single enemy, preserving asymmetric scout-then-shoot roles. As Table 14 shows, this variant is clearly learnable: QMIX reaches 0.35 ± 0.03 and all on-policy baselines produce nonzero scores. Reward shaping is, therefore, not the bottleneck. Failure on the full task is structural, from the temporal dependency chain across rooms under first-person partial observability. Figure 26 demonstrates the simplified layout.

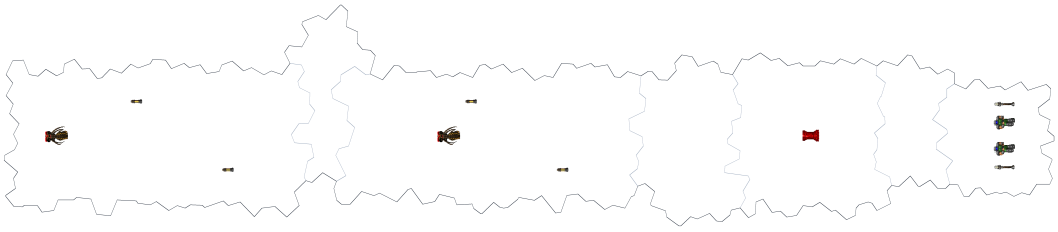


Figure 26: Simplified version of Stealth Labyrinth, with only 1 enemy per room, and agents are in the rightmost room. The enemy in second room only starts attacking after the enemy in the first room is defeated.

Table 14: Ablation results on the simplified variant of the *Stealth Labyrinth* scenario at the 100M-step budget. Entries report mean $\pm$ 95% CI half-width over five runs with seeds $\{42, 68, 81, 97, 154\}$. Full-task rows are reproduced from Table 3. Nonzero performance on the simplified variant confirms that near-zero full-task scores reflect structural algorithmic difficulty rather than degenerate reward design.

Scenario	IPPO	MAPPO	HAPPO	IDQN	VDN	QMIX	QPLEX-D	QPLEX-Q	Ceiling
Stealth Labyrinth (Full)	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.01	$\mathbf{0.21 \pm 0.01}$	0.00 ± 0.01	0.00 ± 0.01	1.00
Stealth Labyrinth Simple	0.02 ± 0.01	0.11 ± 0.03	0.10 ± 0.02	0.01 ± 0.01	0.04 ± 0.01	$\mathbf{0.35 \pm 0.03}$	0.16 ± 0.03	0.21 ± 0.04	1.00

Lava Maze *Lava Maze* is structurally a complex version of the Lewis signaling game (Lewis, 2008). The lava maze grid spans two dimensions, so the navigator can move in four directions total, and the observer can signal via four color channels. However, because there are no fixed reward-shaping mappings between signals and directions, the agents must develop an emergent communication protocol. Since there are no intermediate rewards for the navigator going in a specific direction given

a signal, even for the initial maze size of 8, there are $5^8 = 390{,}625$ distinct possible signal sequences the observer can produce over the 8 maze tiles (4 colors plus silence). Simply reducing the size of the initial maze yields a local reward optimum for the navigator agent, who learns to ignore the signal entirely and arbitrarily pick a direction, thus preventing the emergence of a communication protocol. The initial size of 8 is chosen to make this degenerate strategy insufficient, since the probability of reaching the goal by random direction selection decays exponentially with maze depth.

However, having a larger initial size also makes it extremely difficult for the agents to learn a reliable communication protocol. To test whether current MARL algorithms can still develop an emergent communication protocol with extreme reward sparsity, we introduce *Lava Maze Simple*. This simplified version has a lava maze that spans only one dimension, so the navigator can move in just two directions and the observer can signal via just two color channels. Figure 27 shows a view of the simplified environment. The version still maintains the reward-shaping design with no fixed mappings between signals and directions. Assuming that each signal maps to the navigator moving to the next tile, with a maze of size 8, the observer has $3^8 = 6{,}561$ distinct possible signal sequences over the 8 maze tiles (2 colors plus silence), compared to $5^8 = 390{,}625$ in *Lava Maze*, a reduction of approximately $60\times$. Accounting for the navigator's possible reactions to each signal, the joint sender-receiver sequence space reduces from $(5 \cdot 5)^8 \approx 1.53 \times 10^{11}$ in *Lava Maze* to $(3 \cdot 2)^8 \approx 1.68 \times 10^6$ in *Lava Maze Simple*, a reduction of approximately $90{,}000\times$.

In the simplified scenario, the navigator agent learns to map a signal to a specific action (specifically, green to moving forward), but the observer agent does not learn to reliably send the correct signals to advance through the maze. The benchmark scores for *Lava Maze Simple* are not included in this ablation study, as the results are not fully representative of the agents' learned behavior in this simplified version. Benchmark score is driven by the completion of levels for which both agents need to develop a communication protocol, but currently it is only "one-sided". Thus, when reproducing the results, it is suggested that one observe the generated videos to see patterns of learned behavior for each agent individually.

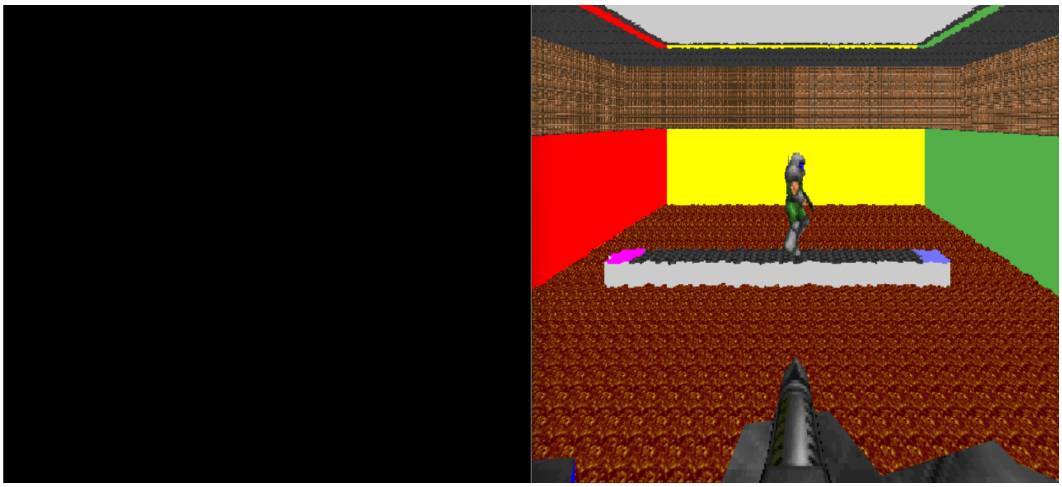


Figure 27: Simplified version of Lava Maze with the navigator agent placed in a one-dimensional maze. The observer agent (on the right) has not sent a signal, so the navigator agent's vision is blacked out (on the left). The blue tile is the start of the maze level, and the purple one is the end.

H.1 Learning references on the original tasks

Simplified variants change the task, so they cannot show that the *original* scenarios are learnable under the benchmark protocol. We therefore ran two families of controlled experiments on the original, unsimplified maps of both scenarios. Each experiment removes exactly one part of the problem and keeps everything else fixed: map geometry, reward shaping, the benchmark score J, the hyperparameters of Appendix B.4, the five training seeds $\{42, 68, 81, 97, 154\}$, and zero-shot evaluation over 100 episodes on the held-out layouts used for Table 3.

Scripted partner. One role is played by a fixed scripted policy, and a single learner is trained on the other role of the full task. In *Stealth Labyrinth*, an engine-side (ACS) scout follows waypoints from the start room through the hub into the currently active room and holds at the room threshold until the room is cleared. It never attacks, and a learned gunner must do all the fighting. In *Lava Maze*, a scripted observer computes the breadth-first-search next step toward the goal from the observer's own telemetry and signals it with a fixed colour convention (one colour per direction) through the ordinary weapon-fire channel. A learned navigator must discover this convention from reward alone. Both learners use their algorithm's single-agent path (IPPO, IDQN).

Privileged (oracle) variants. Both agents are trained with the best on-policy and off-policy baselines (MAPPO, QMIX) for 100M steps, but a hypothesised bottleneck is removed: (i) *oracle message head* (*Lava Maze*): the observer's discrete 5-way signal head is copied losslessly into the navigator's measurement vector on the next step, with no weapon-switch latency, flash timer, or signal cost; the meaning of each symbol must still emerge; (ii) *oracle direction hint* (*Lava Maze*): the navigator directly receives the one-hot BFS next step, which removes communication altogether; (iii) *always-lit rooms* (*Stealth Labyrinth*): all rooms and corridors are permanently lit, while turrets remain invulnerable until the scout enters the active room; (iv) *shared vision* (*Stealth Labyrinth*): each agent receives its own frame stacked with its partner's frame, so the gunner literally sees the scout's camera.

Table 15: Learning references on the original *Very Difficult* maps, evaluated zero-shot on the held-out layouts of Table 3. Entries report the mean benchmark score $J \pm 95\%$ CI half-width (Student-t, $t_{0.975,4}$) over five training seeds, each averaged over 100 evaluation episodes. Full-task rows repeat the best baseline in Table 3. For *Lava Maze*, J is the number of mazes completed per episode (Appendix A.3); it has no finite ceiling, because a new maze is generated after every completion until the time limit.

Setting	Algorithm	Benchmark score J
Stealth Labyrinth (fraction of relays cleared; ceiling 1.00)		
Full task (Table 3)	QMIX	0.21 ± 0.01
Scripted scout → learned gunner	IPPO	0.16 ± 0.04
Scripted scout → learned gunner	IDQN	$\mathbf{0.28 \pm 0.03}$
Always-lit rooms	MAPPO	0.00 ± 0.00
Always-lit rooms	QMIX	0.01 ± 0.03
Shared vision	MAPPO	0.00 ± 0.00
Shared vision	QMIX	0.00 ± 0.00
Lava Maze (mazes completed per episode; no finite ceiling)		
Full task (Table 3)	all	0.00 ± 0.01
Scripted observer → learned navigator	IPPO	0.97 ± 0.12
Scripted observer → learned navigator	IDQN	0.06 ± 0.06
Oracle message head	MAPPO	0.02 ± 0.03
Oracle message head	QMIX	0.00 ± 0.01
Oracle direction hint	MAPPO	$\mathbf{35.83 \pm 16.28}$
Oracle direction hint	QMIX	1.53 ± 1.48

Each role is learnable on the original maps. With a scripted scout, a learned gunner clears 28% of the relays on held-out layouts (IDQN; IPPO 16%). This exceeds the best full-task baseline (0.21, QMIX), even though IDQN scores 0.00 on the full task. With a scripted observer, an IPPO navigator completes about one maze per held-out episode ($J = 0.97$), where every baseline scores zero on the full task. The navigator is never told the colour convention, so it learns to decode a fixed signalling protocol from reward alone. Hence the individual skills that each scenario requires (clearing turrets in the dark behind a torch-bearer, and following visual signals over lava) can be learned under the benchmark protocol and budget.

Where the joint task fails. The oracle rows locate the remaining difficulty. In *Lava Maze*, removing communication altogether makes the task easy for MAPPO ($J = 35.8$ mazes per episode), so perception, control, and navigation are not the bottleneck. A lossless, zero-latency discrete message head, however, does not help: both algorithms stay at zero. The obstacle is therefore not the noisy engine-level signalling channel but the emergence of a shared convention. Sender and receiver must co-adapt a meaning for each symbol under sparse reward, and a learned receiver succeeds only once the sender's convention is held fixed. In *Stealth Labyrinth*, neither permanent lighting nor shared vision produces non-zero scores, while a scripted scout does. Visual information is therefore not what

the learned teams lack. What fails is the scout's leading behaviour: entering and holding the active room in sequence so that the turrets become vulnerable while the gunner fires. This is the temporal dependency chain described above. In both scenarios, the failure of current CTDE baselines is thus a failure of *co-adaptation between roles*, not of learning either role in isolation, and this is precisely the capability the *Very Difficult* tier is meant to test.

H.2 Human Reference Baseline

Simplified variants show that the *core mechanics* of the two hardest scenarios are learnable, but they cannot say whether the *full* scenarios are solvable at all. This human reference baseline therefore supplies that missing upper bound.

Protocol. We recruited two independent pairs of participants, each playing both *Stealth Labyrinth* and *Lava Maze*. Participants were moderately familiar with Doom-style controls. They were told each scenario's goal and available actions but were deliberately **not** told the underlying mechanism, and they played 10 unrecorded warm-up episodes before measurement. During recorded play, participants could neither talk to each other nor see each other's screen, which reproduces the partial observability and absence of a communication channel that the learned policies face. We recorded 10 episodes per pair, which proved sufficient for *Stealth Labyrinth* but not for *Lava Maze*; the second pair therefore continued to 35 episodes on that scenario. Tables 16 and 17 report running means, so column 1–k is the mean benchmark score over the first k recorded episodes.

Table 16: *Stealth Labyrinth* human running-mean benchmark score (ceiling 1.00).

Episodes	1–2	1–4	1–6	1–8	1–10
Pair 1	0.25	0.62	0.75	0.81	0.85
Pair 2	0.50	0.75	0.83	0.88	0.90

Table 17: *Lava Maze* human running-mean benchmark score (mazes completed per episode; no finite ceiling). Data collection for Pair 1 was discontinued after 10 episodes; however, the trend suggests potential improvement.

Episodes	1–2	1–4	1–6	1–8	1–10	1–15	1–20	1–25	1–30	1–35
Pair 1	0.00	0.00	0.00	0.13	0.10	–	–	–	–	–
Pair 2	0.00	0.25	0.50	0.63	0.90	1.20	1.20	1.16	1.40	1.66

Stealth Labyrinth is solvable; agents fail to coordinate. Human performance stabilizes within four to six episodes and then holds, flattening at 0.85 and 0.90 for the two pairs. Two independently recruited pairs converging on a near-identical ceiling, without communication and without being told the mechanism, is strong evidence that the task is learnable from egocentric observation alone. That ceiling is roughly four times the 0.21 achieved by the best learned policy (QMIX), which places the remaining gap on coordination.

Lava Maze is the harder communication problem, for humans too. Pair 1 stays near zero throughout (0.10 mazes per episode after 10 episodes, with only weak signs of improvement), while Pair 2 climbs steadily from 0.00 to 1.66 mazes per episode over 35 episodes, with high-scoring episodes appearing consistently after extended practice. This asymmetry between two pairs, and the fact that neither pair flattens within the recorded window, indicates that *Lava Maze* presents a non-trivial communication and navigation challenge, which is exactly the difficulty the learned agents face. It also sets a meaningful target: an algorithm that solves *Lava Maze* would be overcoming a challenge that remains hard for coordinated human players.

H.3 Conclusion on *Very Difficult* scenarios

Together, these studies show that the *Very Difficult* tier represents a genuine algorithmic limitation. The simplified variants show that the core mechanics are learnable. The scripted-partner and oracle references show that, on the original maps and under the benchmark protocol, each role is learnable and navigation alone is easy, which localizes the failure to co-adaptation between roles. The human reference shows that the full scenarios are solvable.

I Action Space Design and Continuous Control Ablation

Section 4.2 shows that COMRAD uses a factorized multi-discrete action space. Because this is a design choice, this appendix states the argument behind it and reports a measured comparison against continuous alternatives.

Why COMRAD uses factorized actions. The primary reason for this choice is comparability; a benchmark whose purpose is to compare CTDE families must present *one* task interface to all of them. Actor-critic methods (IPPO, MAPPO, HAPPO) can in principle, consume either discrete or continuous actions, but the value-decomposition family cannot: QMIX, QPLEX, and VDN all rely on discrete Q-values decompositions and select actions by an $\arg\max$ (or a softmax over action logits) across a finite set. There is no natural continuous equivalent to this, so adopting continuous control would either remove half of the baseline suite from the benchmark or force each family onto a different action interface, which would confound every cross-family comparison the benchmark makes.

Continuous Control Ablation. To check that this choice does not hide a substantive result, we implemented two continuous variants of *Stag Hunt Arena*, the scenario in which precise movement and aiming matter most. We expose three relevant continuous *delta* buttons for movement and aiming: `TURN_LEFT_RIGHT_DELTA` (yaw), `MOVE_FORWARD_BACKWARD_DELTA` (forward/backward), and `MOVE_LEFT_RIGHT_DELTA` (strafe). The *hybrid* variant makes only the mouse-equivalent aiming axis continuous and leaves locomotion discrete; the *fully continuous* variant replaces all movement with continuous buttons. Each variant was trained with MAPPO for 100M steps, and evaluated zero-shot on held-out maps with a mechanically harder setting. *Reward fluctuation* is the peak-to-trough spread of the training return curve over the final third of training, which therefore measures learning stability.

Table 18: Action-space ablation on *Stag Hunt Arena* under MAPPO at a 100M-step budget, evaluated zero-shot on held-out maps. *Reward fluctuation* is the peak-to-trough spread of the smoothed training return over the final third of training; lower is more stable. *Off-policy support* records whether the variant can still be run with the value-decomposition baselines. These are single-run pilots and should be read as an interface-feasibility check rather than a seeded performance comparison.

Action interface	Benchmark score	Reward fluctuation	FPS	Off-policy support
Multi-discrete (COMRAD)	**26.75**	**95**	$\sim$25K	Native
Hybrid (continuous aiming only)	**26.75**	109	$\sim$25K	Breaks QMIX/QPLEX/VDN
Fully continuous	24.85	361	$\sim$24.5K	Breaks QMIX/QPLEX/VDN

Findings. Table 18 shows that continuous control gives nothing and costs something. The hybrid variant matches the discrete interface exactly (26.75) while being measurably less stable during training; the fully continuous variant is both worse (24.85) and far noisier, with a reward fluctuation of 361 against 95 for the discrete baseline. Throughput is unchanged, so the cost is mainly exploration: continuous motor enlarge the space the policy must search without introducing any new coordination structure. Therefore, continuous control is orthogonal to the question COMRAD is built to answer, and that under a fixed budget, it adds learning noise and breaks half of the baseline suite without changing any coordination conclusion.

J Extended Baseline Results

This appendix complements Table 3 with two audit-friendly views of the baseline study. Figure 28 shows per-scenario learning curves, providing a qualitative view of data efficiency and optimization behavior rather than a variance estimate. Figure 29 adds a scenario-normalized score matrix that makes scenario-wise winners and unresolved tasks visually explicit without averaging incompatible raw units across scenarios.

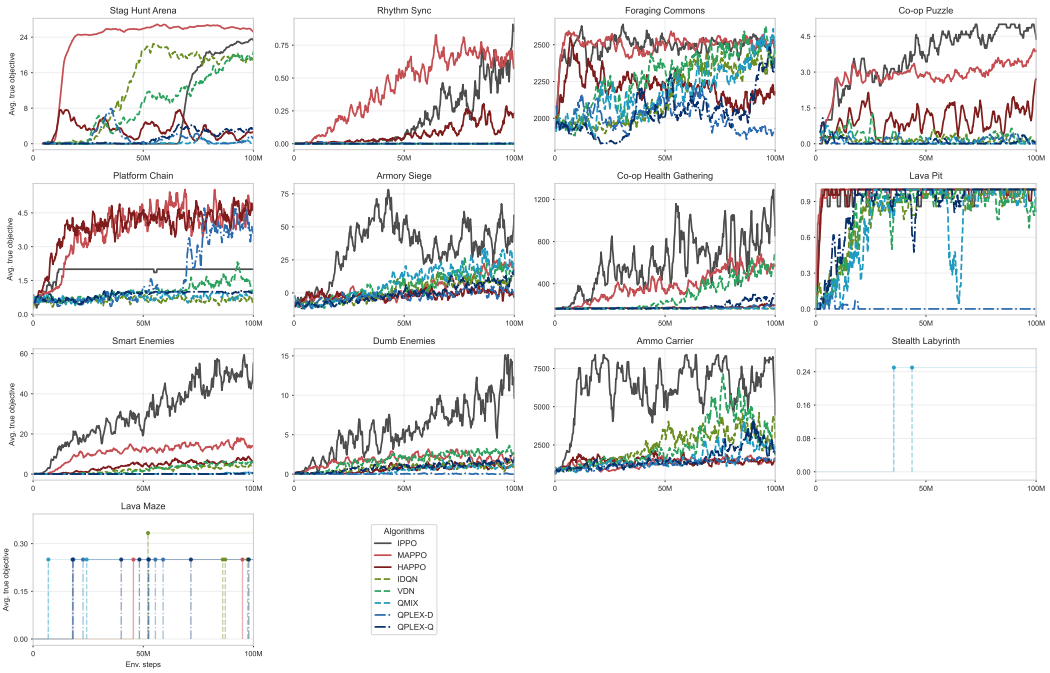


Figure 28: Per-scenario learning curves for all eight baselines, plotted as scenario-level benchmark score against environment steps, aligned to the common 100M-step budget. Each panel shows a *single representative seed* per algorithm because overlaying five curves for eight algorithms across 13 panels is unreadable at this size; Table 3 is the seeded estimate. Raw score scales are kept scenario-specific because the benchmark intentionally uses different task-level units across scenarios, including survival time, completed stages, progress checkpoints, and kills.

	IPPO	MAPPO	HAPPO	IDQN	VDN	QMIX	QPLEX-D	QPLEX-Q	
Overall score	**0.29**	0.25	0.11	0.10	0.12	0.08	0.04	0.07	**Overall score**
Stag Hunt Arena	0.49	**0.71**	0.07	0.40	0.50	0.01	0.04	0.10	Stag Hunt Arena
Rhythm Sync	0.59	**0.65**	0.22	0.00	0.00	0.00	0.00	0.00	Rhythm Sync
Foraging Commons	0.11	0.12	0.10	0.10	0.11	**0.12**	0.09	0.11	Foraging Commons
Co-op Puzzle	0.61	**0.78**	0.45	0.01	0.01	0.00	0.00	0.01	Co-op Puzzle
Platform Chain	0.04	**0.10**	0.10	0.01	0.02	0.02	0.07	0.02	Platform Chain
Armory Siege	0.17	0.20	0.00	0.11	0.18	**0.23**	0.01	0.14	Armory Siege
Co-op Health Gathering	**0.53**	0.26	0.09	0.08	0.27	0.08	0.09	0.13	Co-op Health Gathering
Lava Pit	0.08	**0.09**	**0.09**	0.08	0.07	0.09	0.00	**0.09**	Lava Pit
Smart Enemies	0.14	**0.17**	0.08	0.04	0.06	0.01	0.00	0.01	Smart Enemies
Dumb Enemies	**0.14**	0.03	0.01	0.01	0.03	0.01	0.00	0.02	Dumb Enemies
Ammo Carrier	**0.84**	0.16	0.16	0.46	0.30	0.28	0.18	0.25	Ammo Carrier
Stealth Labyrinth	0.00	0.00	0.00	0.00	0.00	**0.21**	0.00	0.00	Stealth Labyrinth
Lava Maze	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	Lava Maze

Figure 29: Scenario-normalized score matrix derived from Table 3. Each cell is $\tilde{s} = \mathrm{clip}(s/s_{\max}, 0, 1)$, the mean benchmark score divided by that scenario's ceiling and clipped to $[0, 1]$ (for *Lava Maze*, which has no finite ceiling, $s_{\max} = 4$ completed mazes, after which the maze has its maximum size; every baseline scores 0 there); the floor is 0 for every scenario, so the normalization has no offset term, and the clip only binds for the single negative entry in the table. Cells are rounded to two decimals for display. Brighter entries indicate stronger within-scenario performance without mixing incompatible raw units, and the top row reports the mean normalized score across scenarios. This view reinforces the main-text claim that a single baseline family does not dominate COMRAD. Leadership is distributed across algorithms, while multiple scenarios remain low-scoring for nearly all methods.