Strict Cooperative Sokoban: Intent-Grounded Recurrent Communication for Partially Observable MARL

Jiaju Wu [1] **Aolin Zhang** [1] **Jingxing Gao** [1]

Abstract

Cooperative Sokoban is a compact testbed for decentralized coordination, but a common 9-action interface contains an action aliasing issue: a push action into empty space can also move the agent. This aliasing makes behavior-cloning labels and action masks ambiguous. We introduce StrictCoop-Sokoban, a fixed two-agent benchmark with strict push semantics, planner-verified train and evaluation pools, local observations, and a disjoint hard split. Under this protocol, MAPPO is the strongest standard baseline, reaching 0.763 ± 0.012 pass@8 on hard v2 levels, but it remains limited by partial observability and long-horizon role commitment. We propose IGRC-MAPPO, which broadcasts a low-dimensional intent message, grounds the message with future box-assignment labels, and preserves intent through DRC-style ConvLSTM memory. The final encoder-128 model reaches 0.949 ± 0.015 pass@8 and 0.973 ± 0.014 pass@16 across three seeds. Ablations show that communication improves stateless MAPPO, auxiliary grounding improves average recurrent performance, and cross-step memory is the largest contributor to robust cooperation.

1. Introduction

Cooperative Sokoban requires more than local obstacle avoidance. Two agents must choose compatible roles, remember which box they are trying to move, infer the partner's likely role, and avoid irreversible pushes. These requirements make it a useful testbed for partially observable cooperative multi-agent reinforcement learning (MARL), but only if the environment interface does not introduce ambiguity that obscures the coordination problem.

We identify such ambiguity in a common 9-action cooperative Sokoban interface. Although the action space distinguishes four *push* actions from four *move* actions, a push into empty space can also move the agent. As a result, two different action labels can describe the same physical transition, and an action mask cannot cleanly mark boxless pushes as invalid. This ambiguity contaminates both behavior-cloning warmup and policy evaluation.

We address this by defining StrictCoop-Sokoban, where push actions move the agent only when they actually displace a box. Navigation must use move actions. We combine this strict interface with planner-verified levels, local observations, and a disjoint hard split. The resulting benchmark exposes a clear failure mode: standard MAPPO is strong, but it struggles to maintain long-horizon role commitments under partial observability.

Given this benchmark, our central hypothesis is:

> *Under strict local observations, communication helps only if the message is grounded in task-level intent and the policy can remember that intent across time.*

We instantiate this hypothesis as IGRC-MAPPO (Intent-Grounded Recurrent Communication MAPPO). The actor broadcasts a low-dimensional intent vector; an auxiliary classifier grounds that vector by predicting which box the agent will push in the near future; and a DRC-style ConvLSTM recurrent actor stores the agent's plan across environment steps. The improvement is not explained by capacity alone. A reset-each-step DRC ablation and a capacity-scaling study show that cross-step memory and moderate visual capacity matter more than raw size.

Contributions.

1. We introduce StrictCoop-Sokoban, a fixed two-agent Sokoban benchmark with strict push semantics, planner-verified levels, local observations, and a disjoint hard evaluation split.

2. We show that, under this protocol, parameter-shared MAPPO is a strong baseline and outperforms HAPPO

Equal contribution. [1]School of Mathematical Sciences, Peking University, Beijing, China.

Course project report.

and our current HASAC implementation on hard v2 levels.

3. We propose IGRC-MAPPO, which combines broadcast intent messages, future box-assignment grounding, and DRC-style recurrent memory.

4. We separate the main sources of improvement through ablations over communication, auxiliary grounding, recurrence, and capacity, together with a mask sanity check.

2. Related Work

Cooperative MARL policy optimization. MAPPO showed that PPO with careful implementation details can be surprisingly strong in cooperative multi-agent games (1). HAPPO and HATRPO extend trust-region policy optimization to heterogeneous-agent settings, and HARL studies this broader family of heterogeneous-agent algorithms (2; 3). HASAC brings maximum-entropy actor-critic training into the same setting (4). Our results suggest that, in the symmetric two-agent Sokoban regime, parameter-shared MAPPO is the most reliable starting point; the main bottleneck is not algorithmic heterogeneity but partial observability and long-horizon coordination.

Communication under partial observability. Differentiable communication has long been used to let agents exchange latent messages under decentralized execution (5; 6). In our setting, the question is not only whether agents can communicate, but what the message represents. We therefore ground messages with a future box-assignment auxiliary target, making communication closer to a compact declaration of cooperative intent than an unconstrained hidden feature.

Recurrent planning in Sokoban-like domains. Sokoban has been used as a compact testbed for learning implicit planning procedures. Deep repeated ConvLSTM models showed that recurrent computation can act as model-free planning in Sokoban-like tasks (7). We adapt this idea to partial-observation MARL: the recurrent state must preserve an agent's own plan and the inferred role of its partner, not merely solve a single-agent visual puzzle.

3. The StrictCoop-Sokoban Benchmark

3.1. Environment family

Each episode contains two agents, boxes, target cells, walls, and floor cells. The reward is shared, and a level is solved when all boxes are on targets. We fix three benchmark versions (Table 1).

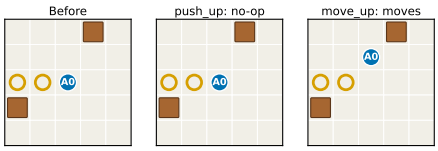

Figure 1. Strict action semantics on a real hard-v2 local state. In the legacy interface, `push_up` into an empty cell would move the agent; in StrictCoop-Sokoban, the same action is a no-op, and navigation must use `move_up`.

Version	Grid	Agents	Boxes	Max steps	Role
v0	7×7	2	2	80	easy cooperative setting
v1	7×7	2	3	100	small map with more boxes
v2	10×10	2	3	140	partial-observation stress test

Table 1. Fixed cooperative Sokoban benchmark versions.

The main experiments use v2. The training pool contains 3000 deduplicated solvable levels. The hard evaluation pool is disjoint and contains levels with planner length at least 10. Unless otherwise stated, we evaluate on 500 hard levels with evaluation seed 300000.

3.2. Strict action semantics

The per-agent action space has nine actions: noop, four push actions, and four move actions. In the default interface, push into empty space behaves like movement. In StrictCoop-Sokoban, push is strict: it is a no-op unless a box is directly ahead and can be displaced. Move actions handle pure navigation.

This change removes an ambiguity rather than adding a heuristic. It makes behavior-cloning labels identifiable and makes valid-action masks well-defined. The mask can now mark a push-without-box as invalid without contradicting the environment dynamics.

3.3. Planner-verified data and local observations

Walls are randomly constructed. Candidate levels are accepted only if a direct search or A* planner finds a valid solution. Stored solution plans provide two signals: they verify solvability, and they provide action and intent labels for behavior-cloning warmup. The planner uses a compact 5-action search in which push actions can also move through empty cells; when replaying these traces under strict semantics, we relabel a boxless push into the corresponding move action and keep push labels only when a box is actually displaced.

Each agent receives a $K \times K$ local crop centered on itself. We use $K = 5$ for all main v2 experiments. This hides much of the global state: an agent may not observe all boxes, all

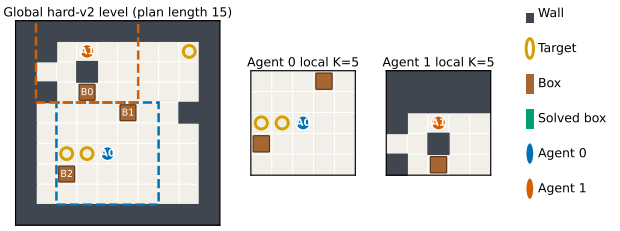

Figure 2. A real fixed-v2 hard benchmark level and the two local observations seen by the decentralized actors. The dashed windows show each agent's $K = 5$ field of view. Neither local crop contains the whole puzzle, so role assignment and memory matter even on a small 10×10 grid.

Figure 3. Planner-verified solution trace for the level in Figure 2, replayed with the same strict-action relabeling used for behavior-cloning warmup. The sequence shows that the hard split contains genuinely sequential multi-box coordination rather than a single local push.

targets, or the other agent. The benchmark therefore asks whether a policy can build and maintain a cooperative plan under partial information.

3.4. Metric

We report pass@k. For each level, the policy receives k stochastic attempts. The level is solved if any attempt succeeds. pass@1 measures single-attempt reliability; pass@4, pass@8, and pass@16 measure whether the policy distribution contains successful cooperative plans under moderate retry budgets. Sokoban is particularly sensitive to this distinction: a policy may sample several plausible role assignments, only some of which solve a level. Each attempt starts from the same level initial state, samples a fresh stochastic rollout, and resets the recurrent state.

4. IGRC-MAPPO

IGRC-MAPPO is a MAPPO-based policy (1) with three ingredients: intent communication, auxiliary grounding, and recurrent memory.

4.1. Principle and formalization

We view strict cooperative Sokoban as a decentralized partially observable Markov decision process. At time t, agent

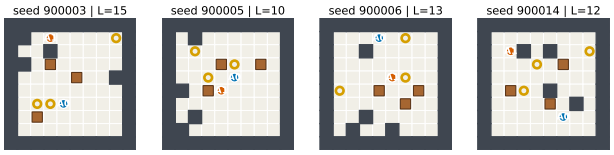

Figure 4. Additional fixed-v2 hard levels sampled from the evaluation split. All panels show real benchmark instances, with seed and planner length shown above each board. The visual diversity comes from random wall construction followed by planner verification, not from hand-designed layouts.

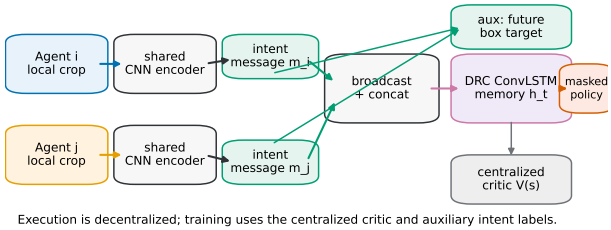

Figure 5. IGRC-MAPPO architecture. Each decentralized actor encodes a local crop, broadcasts an intent vector, receives the partner's intent, and updates a DRC-style recurrent state before producing a masked action distribution. Training adds a centralized critic and an auxiliary future-box target loss.

i receives a local crop o_i^t, while the centralized critic may access the compact global state s^t. A stateless actor must choose $\pi_i(a_i^t \mid o_i^t)$, but the same local crop can correspond to different global roles. IGRC-MAPPO therefore augments the policy input with two variables: a communicated intent m_j^t from the partner and a recurrent state h_i^t that persists across steps.

The actor factorization is:

$$z_i^t = f_\theta(o_i^t, e_i), \tag{1}$$

$$m_i^t = g_\theta(z_i^t), \tag{2}$$

$$h_i^t = F_\theta^{(R)}(h_i^{t-1}, z_i^t, m_{-i}^t), \tag{3}$$

$$\pi_i(a_i^t \mid \tau_i^t, m_{-i}^t) = \mathrm{MaskedSoftmax}\big(q_\theta(h_i^t), M_i^t\big), \tag{4}$$

where e_i is an agent-id embedding, R is the number of repeated ConvLSTM updates per environment step, M_i^t is the valid-action/deadlock mask, and τ_i^t denotes agent i's action-observation history. The centralized critic estimates $V_\phi(s^t)$, while execution remains decentralized because each actor uses only its local crop, its recurrent state, and received messages.

The optimization objective combines MAPPO with super-

vised warmup and intent grounding:

$$\mathcal{L} = \mathcal{L}_{\text{MAPPO}} + \lambda_{\text{bc}}\mathcal{L}_{\text{BC}} + \lambda_{\text{aux}}\mathcal{L}_{\text{intent}}. \quad (5)$$

For the auxiliary term, a planner trajectory defines a label $y_i^t \in \{0, \ldots, B\}$: the first box agent i will push within a 10-step horizon, or a `no_target` sentinel. We train $c_\theta(m_i^t)$ to predict this label. In the balanced version, class weights are inversely proportional to label frequency, preventing the common `no_target` class from dominating the communication channel. Thus the message is pressured to represent task-level intent, and the recurrent state is pressured to maintain that intent across ambiguous local views.

4.2. MAPPO backbone

The backbone is MAPPO with a centralized critic and a decentralized parameter-shared actor. The actor receives each agent's local observation and an agent-id embedding. Parameter sharing is important because the agents are symmetric; it lets both agents contribute to the same policy rather than splitting experience across heterogeneous actors.

All MAPPO-family results in the main tables use the same strict action semantics, valid-action mask, and deadlock mask. Thus the gap between MAPPO and IGRC-MAPPO is not caused by invalid-action avoidance alone.

The valid-action mask removes actions that are impossible under strict dynamics, such as boxless pushes. The deadlock mask removes only locally provable dead pushes, such as pushing a box into an unrecoverable wall or corner configuration. It does not encode a global solution plan or assign boxes to agents.

4.3. Broadcast intent communication

Each agent encodes its local observation into a latent vector z_i. A small head emits an intent message $m_i \in \mathbb{R}^8$. Agent i's policy head receives both its own latent z_i and the other agent's message m_j. Execution remains decentralized: each actor uses its local observation and received messages, while training uses the centralized critic.

4.4. Intent grounding by future box assignment

Unconstrained messages can become arbitrary features. We ground them with an auxiliary task derived from planner trajectories. For each agent and time step, we look ahead 10 planner steps and label the first box the agent pushes. If it pushes no box in that window, the label is `no_target`. A classifier predicts this label from the intent vector.

We test both unbalanced cross-entropy and inverse-frequency class-balanced cross-entropy. The balanced loss prevents the common `no_target` label from dominating the intent representation. As the ablation shows, this auxiliary objective is not a monotonic performance knob; instead, it reshapes the policy distribution and provides a task-level meaning for communication.

4.5. DRC-style recurrent memory

Local observations are ambiguous. The same 5×5 crop can require different actions depending on which box the agent committed to several steps ago or which role the partner implicitly chose. We therefore use a DRC-style ConvLSTM actor (7). The recurrent state is reset at episode start and at each pass@k retry, then preserved across environment steps.

The recurrent update is repeated several times per step. To distinguish memory from extra per-frame computation, we evaluate a reset-each-step variant that keeps repeated computation but discards cross-step state.

5. Experiments

5.1. Protocol

All fixed-v2 main results use:

- environment: `TwoPlayer-Sokoban-v2`;
- train pool: fixed v2 raw pool with 3000 levels;
- evaluation pool: fixed v2 hard disjoint split;
- evaluation size: 500 hard levels;
- local observation: $K = 5$;
- strict push, valid-action mask, and deadlock mask enabled;
- evaluation seed: 300000.

We denote a DRC actor with R recurrent repeats and hidden width H as rR hH. For example, r3 h32 uses three ConvLSTM updates per environment step and hidden width 32. The suffix enc128 denotes a CNN encoder width of 128.

5.2. RQ1: How strong are standard MARL baselines?

Method	pass@1	pass@4	pass@8	pass@16	Seeds
HASAC	0.000	0.000	0.000	–	0
HAPPO	0.337 ± 0.035	0.571 ± 0.017	0.673 ± 0.024	0.771 ± 0.027	0,1,2
MAPPO	0.398 ± 0.009	0.657 ± 0.025	0.763 ± 0.012	0.832 ± 0.006	0,1,2
IGRC r3 h32	0.566	0.842	0.936	–	0
IGRC r3 h32 enc128	0.581 ± 0.062	0.869 ± 0.046	0.949 ± 0.015	0.973 ± 0.014	0,1,2

Table 2. Fixed v2 hard comparison. Entries with multiple seeds are shown as mean $\pm$ sample standard deviation. MAPPO and HAPPO seed 0 were not evaluated at pass@16, so their pass@16 statistics use seeds 1 and 2.

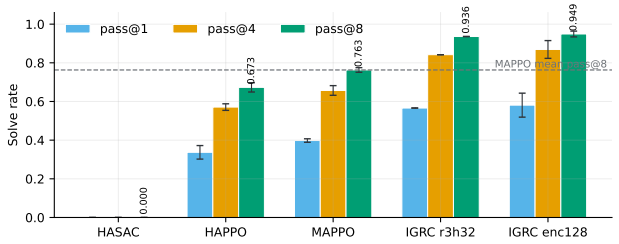

Figure 6. Main fixed-v2 hard comparison. Bars visualize Table 2; error bars show seed standard deviation where available, and the dashed line marks MAPPO's mean pass@8.

Table 2 compares HASAC, HAPPO, MAPPO, and IGRC-MAPPO on the fixed v2 hard split. MAPPO is the strongest standard baseline, reaching 0.763 ± 0.012 pass@8. HAPPO appears less well matched to this symmetric two-agent setting, where parameter sharing is beneficial. In our current strict/masked implementation, HASAC obtains zero hard-level solve rate after online training. The final encoder-128 IGRC-MAPPO model reaches 0.949 ± 0.015 pass@8 and 0.973 ± 0.014 pass@16 across three seeds, a $+0.186$ absolute improvement over MAPPO at pass@8.

5.3. RQ2: Which components explain the improvement?

Method	pass@1	pass@4	pass@8	pass@16
MAPPO	0.394	0.686	0.774	–
MAPPO + Comm	0.476	0.760	0.838	0.890
MAPPO + Comm + Aux	0.464	0.724	0.836	0.910
MAPPO + Comm + Balanced Aux	0.448	0.706	0.854	0.892
+ DRC r3 h32	0.566	0.842	0.936	–
+ encoder 128, seed 0	**0.636**	0.892	0.956	0.976
+ encoder 128, seed 1	0.592	**0.898**	**0.960**	**0.986**

Table 3. Single-seed incremental design path on fixed v2 hard levels unless a seed is explicitly shown.

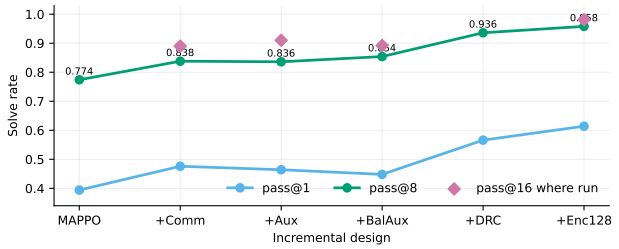

Figure 7. Incremental design path from MAPPO to IGRC-MAPPO. Communication gives the first stateless gain, auxiliary supervision changes the retry profile, and the largest jump appears when DRC memory is introduced.

Communication alone improves stateless MAPPO: pass@8 rises from 0.774 to 0.838. Auxiliary supervision changes the retry distribution rather than improving every metric. The unbalanced auxiliary loss has the best stateless pass@16, while the balanced loss has the best stateless pass@8. The largest improvement comes when DRC memory is added, raising pass@8 to 0.936 before encoder scaling.

5.4. RQ3: Is memory useful beyond extra computation?

Variant	pass@1	pass@4	pass@8	Steps@8
r3 h32 memory	**0.566**	0.842	**0.936**	36.54
r1 h32 memory	0.548	**0.846**	0.924	50.66
r3 h32 reset each step	0.516	0.794	0.910	37.56
r2 h32 memory	0.536	0.844	0.902	36.04
r1 h16 memory	0.508	0.806	0.900	39.67
r1 h64 memory	0.484	0.804	0.896	39.80

Table 4. Single-seed DRC structure ablation. Resetting the state each step keeps repeated computation but removes cross-step memory.

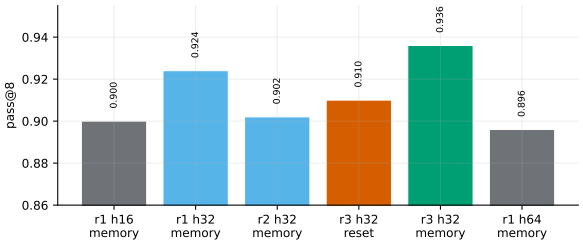

Figure 8. DRC structure ablation visualized by pass@8. The reset-each-step variant keeps repeated computation but removes cross-step state; its drop supports the interpretation that memory, not only extra computation, drives the improvement.

The reset-each-step model keeps repeated ConvLSTM computation within each frame but discards cross-step state. Its lower pass@8 indicates that the gain is not explained only by additional per-step computation; persistent memory carries useful plan information.

5.5. RQ4: Does larger capacity help?

Variant	pass@1	pass@4	pass@8	pass@16	Steps@8
Default enc.	0.566	0.842	0.936	–	36.54
h64 recurrent	0.582	0.870	0.952	–	–
enc128, seed 0	**0.636**	0.892	0.956	0.976	34.60
enc128, seed 1	0.592	**0.898**	**0.960**	**0.986**	41.14
enc512	0.590	0.876	0.948	–	36.19

Table 5. Single-seed capacity scaling after choosing the DRC structure, except for the encoder-128 rows shown separately for seeds 0 and 1. Encoder 128 is better than encoder 512.

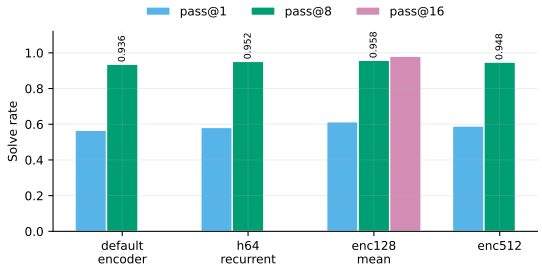

Figure 9. Capacity scaling after selecting the DRC structure. Moderate encoder scaling to 128 gives the best current result, while encoder 512 underperforms despite higher raw capacity.

Moderate scaling is useful, but larger capacity does not monotonically improve performance. Encoder 512 underperforms encoder 128, suggesting that optimization and recurrent inductive bias matter more than raw capacity.

5.6. Seed stability

The final encoder-128 model has three completed seeds:

Seed	pass@1	pass@4	pass@8	pass@16
0	0.636	0.892	0.956	0.976
1	0.592	0.898	0.960	0.986
2	0.514	0.816	0.932	0.958
mean ± std	0.581 ± 0.062	0.869 ± 0.046	0.949 ± 0.015	0.973 ± 0.014

Table 6. Three-seed check for IGRC-MAPPO r3 h32 encoder 128.

The seed check shows two regimes. One-shot execution has visible seed variation: pass@1 ranges from 0.522 to 0.636. Retry-based solve rate is more stable: pass@8 remains between 0.932 and 0.960, and pass@16 remains between 0.958 and 0.986. Thus the learned policy distribution reliably contains successful plans, even when the probability of sampling one on the first attempt varies.

5.7. RQ5: Are masks the whole explanation?

Because action and deadlock masks are enabled in the main protocol, we run a single-seed sanity check that removes the deadlock mask from the final encoder-128 model while keeping strict actions and the valid-action mask.

Variant	pass@1	pass@4	pass@8	pass@16	Seed
IGRC enc128, no deadlock mask	0.528	0.824	0.920	0.954	0

Table 7. Mask sanity check on fixed v2 hard levels. Removing the deadlock mask does not collapse the final recurrent communicative policy.

The no-deadlock result reaches 0.920 pass@8 and 0.954 pass@16. This does not prove that masks are unimportant, but it supports the narrower claim that the main gain is not solely explained by a hand-coded deadlock rule.

5.8. RQ6: Does auxiliary grounding matter?

To isolate the effect of the intent-grounding loss in the final recurrent communicative model, we rerun the encoder-128 DRC MAPPO recipe with broadcast communication but set both auxiliary coefficients to zero. All strict-action, valid-action-mask, deadlock-mask, DRC, and evaluation settings are unchanged.

Variant	pass@1	pass@4	pass@8	pass@16
IGRC-MAPPO (no auxiliary loss)	0.455 ± 0.155	0.769 ± 0.129	0.863 ± 0.107	0.926 ± 0.062
IGRC-MAPPO (full)	0.581 ± 0.062	0.869 ± 0.046	0.949 ± 0.015	0.973 ± 0.014

Table 8. Auxiliary-grounding ablation on fixed v2 hard levels over three seeds. Removing the auxiliary intent loss leaves broadcast communication and recurrent memory intact, but lowers mean solve rate under the same evaluation protocol.

Without auxiliary grounding, mean pass@8 drops from 0.949 to 0.863 in our three-seed no-auxiliary rerun. The no-auxiliary seeds can still exploit broadcast communication, so we interpret the auxiliary loss as a useful grounding signal that improves average performance rather than as a component that is strictly necessary for successful communication.

As a preliminary diagnostic, we also shuffle the partner message at recurrent states near boxes. On the one available auxiliary checkpoint, this changes movement decisions in pre-push coordination states, especially when agents are positioning near different boxes. The same probe is not yet available for all auxiliary seeds, so we treat it as a mechanism hint rather than primary evidence. The main auxiliary-grounding claim in this paper is the multi-seed performance gap in Table 8.

5.9. Diagnostic intent representation

We also inspect the learned intent vectors of the recurrent encoder-128 model. For 300 hard levels, we replay planner trajectories, preserve the DRC hidden state, extract each 8-dimensional broadcast message, and color the resulting points by the future box-assignment label used by the auxiliary objective. We plot the full model and the no-auxiliary ablation side by side.

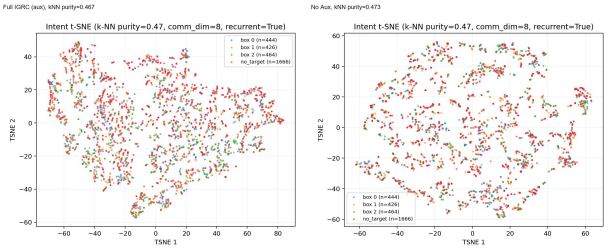

Figure 10. t-SNE comparison of recurrent broadcast messages. Left: full IGRC-MAPPO with auxiliary intent grounding. Right: the same recurrent communicative recipe with the auxiliary loss removed. Colors show future box-assignment labels. The visualization should be read cautiously: kNN purity is similar for the two runs (0.467 vs. 0.473), so the stronger evidence for the auxiliary loss is the multi-seed performance gap in Table 8.

The intent space is not a perfectly separated classifier of box identity. This is expected because `no_target` includes navigation, waiting, and role-maintenance states, and because the message also supports action selection rather than only the auxiliary label. The no-auxiliary run can form a comparably structured embedding under this metric while still showing lower average task performance. We therefore treat the visualization as diagnostic, not as a standalone proof that the auxiliary objective causes clean semantic clustering.

6. Discussion

Benchmark design matters. The strict action protocol removes push-move action aliasing. Without this correction, push and move labels are partially interchangeable, making it harder to interpret behavior-cloning accuracy and action-mask gains.

Communication is useful but insufficient. Broadcast communication improves stateless MAPPO, but it does not close the gap to the recurrent model. Under partial observation, agents need to maintain a role commitment across time; a one-step message cannot by itself solve that problem.

Auxiliary intent grounding acts as a shaping signal. The auxiliary task does not monotonically improve all stateless metrics. Instead, in the final recurrent model it improves mean solve rate. In this three-seed comparison, auxiliary grounding also appears to reduce seed-to-seed variation: removing the auxiliary loss from the recurrent communicative encoder-128 recipe drops mean pass@8 from 0.949 to 0.863, while the pass@8 standard deviation increases from 0.015 to 0.107. The representation visualization supports this interpretation only cautiously: the message space contains measurable future-target structure, but it is not a clean box-id classifier and the no-auxiliary run has similar kNN purity. The more informative diagnostic is behavioral: shuffling the partner message changes movement decisions near boxes, especially in states where agents appear to be positioning or dividing labor before a push. No-auxiliary seeds can sometimes learn a similar sensitivity, so the auxiliary mechanism claim should be read as an average-performance and grounding effect rather than an aux-only capability.

Masks are not the whole story. The main protocol uses valid-action and deadlock masks to keep the strict action interface well defined and avoid locally provable dead pushes. However, the final recurrent communicative model remains strong without the deadlock mask, reaching 0.920 pass@8. This suggests that the performance gap is not only a consequence of hard-coded Sokoban rules.

Larger capacity does not monotonically improve performance. Encoder 512 is worse than encoder 128. The results point to a specific inductive bias—moderate visual abstraction plus recurrent plan memory—not to unconstrained scaling.

7. Conclusion

We introduced StrictCoop-Sokoban, a strict cooperative Sokoban benchmark designed to remove push-move action aliasing and expose partial-observation coordination. On this benchmark, plain MAPPO is already a strong baseline, but it struggles to maintain role commitments across time. IGRC-MAPPO improves MAPPO by making messages predict future box-level intent and by carrying that intent through a DRC-style ConvLSTM actor. The empirical picture is consistent across the main ablations: communication helps, auxiliary grounding shapes the policy distribution, and recurrent memory is the most important observed contributor to robust cooperation on hard v2 levels.

8. Limitations

The main MAPPO, HAPPO, final encoder-128, and recurrent no-auxiliary comparisons now use three seeds for pass@1/pass@4/pass@8, but several ablations remain single-seed: the stateless communication/auxiliary path, DRC structure, capacity scaling, and no-deadlock sanity check. The main paper focuses on v2; v0 and v1 are fixed benchmarks but are not rerun with the final method. The mask ablation is not fully factorial because masks are treated as part of the corrected benchmark interface. The intent-vector visualization shows measurable structure, but it should not be overinterpreted as a clean semantic clustering or as the main evidence for auxiliary grounding. The message-intervention analysis is also preliminary because only the final auxiliary seed-2 checkpoint was locally avail-

able; the older auxiliary seed-0 and seed-1 checkpoints should be recovered or rerun before making a strong multi-seed mechanistic claim. HASAC is represented by the current local implementation with fair strict/mask support; a heavily stabilized HASAC variant may perform better.

9. Reproducibility

The released tree contains the benchmark configuration, training and evaluation code, result snapshots, and the final policy. For the course artifact, we collect the code, fixed v2 benchmark, checkpoint, evaluation summaries, and generated rollout GIFs in `assignment_part2_3_submission/`. The included checkpoint `model/ours_enc128_s2_model.pt` was verified as a `TwoPlayer-Sokoban-v2` two-agent, three-box `marl_mappo` policy. The GIF generator scans fixed benchmark levels, keeps only solved stochastic rollouts, and records level indices, attempts, seeds, and solve lengths in `demo_gifs/manifest.json`.

The main training recipe uses 30 behavior-cloning warmup epochs with learning rate 3×10^{-4}, followed by 600 RL episodes with learning rate 5×10^{-5}. PPO uses four update epochs, minibatches of 256, clip ratio 0.08, discount 0.99, GAE $\lambda = 0.95$, value coefficient 0.5, and an entropy coefficient annealed from 0.015 to 0.002. The auxiliary intent loss has weight 0.5 during warmup and 0.1 during RL with a 10-step planner-label horizon. The final recurrent actor uses communication dimension 8, three DRC repeats, hidden width 32, encoder width 128, and curriculum buckets 0–3, 0–5, 0–7, and 0–10.

References

[1] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre M. Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in Neural Information Processing Systems*, 2022.

[2] Jakub Grudzien Kuba et al. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2022.

[3] Yongcan Zhong et al. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 25:1–67, 2024.

[4] Yuan Li et al. Maximum entropy heterogeneous-agent reinforcement learning. In *International Conference on Learning Representations*, 2024.

[5] Jakob Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2016.

[6] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, 2016.

[7] Arthur Guez et al. An investigation of model-free planning. In *International Conference on Machine Learning*, 2019.