THE LAST MILE OF SKILL TRANSFER: IN-SAMPLE TEAM-VALUE ALIGNMENT FOR OFFLINE MULTI-AGENT GENERALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline multi-agent reinforcement learning (MARL) from multi-task datasets aims to learn cooperative policies that generalize zero-shot to unseen tasks with varying numbers of agents, and hierarchical skill learning is the dominant approach. In these methods, however, value supervision stops at the skill: the low-level executor is trained by conditional behavior cloning, so successful and unsuccessful realizations of one skill are blended, and a correctly selected skill is still executed by low-value actions. We formalize this as the **hierarchical value gap**, which closes only under a signal that varies with realization quality within a skill, and that no skill-level objective supplies. We propose **Skill-conditioned Team-value Execution Projection (STEP)**, a training-time framework that completes the value pathway of hierarchical skill policies by extending team-value supervision from skill selection to skill execution. Our approach introduces an in-sample expectile team value that distinguishes the quality of logged executions under the same skill and supplies the weighting signal for execution projection, a team-residual execution projection that uses this signal to make decentralized executors imitate high-value realizations of each skill, and execution-consistent consequence conditioning that aligns the value with the skill semantics actually being executed. On nine unseen SMAC maps, STEP improves over the strongest hierarchical skill baseline by 5.4 and 4.8 win-rate points on medium-expert and medium-replay data, respectively, while matching its performance on expert data, whose returns are saturated. These results show that STEP delivers value supervision to skill execution, strengthening zero-shot cooperation on mixed-quality data without sacrificing performance on high-quality data.

1 INTRODUCTION

Cooperative multi-agent reinforcement learning (MARL) studies how agents coordinate to achieve a shared goal (Oliehoek & Amato, 2016; Samvelyan et al., 2019). Across tasks with different team sizes and opponent configurations, collecting new interaction data for every configuration can be costly or infeasible. Offline multi-task MARL therefore seeks to learn reusable cooperative behavior from fixed source-task trajectories and transfer it zero-shot to unseen configurations, where decentralized agents must turn that behavior into effective actions without target-task data or updates.

To learn reusable cooperation from fixed data, hierarchical methods encode coordination as latent skills that can transfer across tasks. ODIS discovers discrete coordination skills and selects them through high-level value learning (Zhang et al., 2023); HiSSD jointly learns common and task-specific skills, grounding them in team dynamics and value (Liu et al., 2025a). Related designs reuse skill combinations through a tactic codebook or learn transferable subgroup skills (Zhang et al., 2025; Chen et al., 2025). These methods advance how cooperative behavior is represented, selected, and reused across tasks.

Yet offline multi-task logs mix trajectories from different tasks, policy versions, and proficiency levels, so the same skill can have realizations with different team outcomes. Value and dynamics signals mainly shape skill learning or selection, while low-level decoders fit logged actions through

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.

equal-weight conditional behavior cloning. For example, in SMAC, an executor for "focus fire while holding formation" may imitate both precise targeting and positioning from expert trajectories and delayed attacks, wrong targets, or wasted movement from lower-quality trajectories. Uniform imitation fits the frequency-weighted mixture of these realizations, so selecting the right skill does not ensure effective joint action: the executor lacks a team-value signal that distinguishes better realizations within the skill.

We formalize this last-mile failure as the *hierarchical value gap*: skill-level value supervision cannot distinguish better and worse action realizations within the same skill. To close it, we propose **Skill-conditioned Team-value Execution Projection (STEP)**, a training-time framework that extends team-value supervision from skill selection to skill execution. An in-sample expectile team value supplies one-step TD residuals for logged joint transitions; STEP uses these residuals as shared weights on the decentralized executors' skill-conditioned imitation losses, favoring higher-value realizations of each skill. Execution-consistent consequence conditioning then makes the decoder-driving skill code an explicit input to the existing consequence predictor. Thus, STEP bridges the gap between selecting transferable cooperative skills and executing them effectively, thereby improving the zero-shot transfer of learned coordination to unseen tasks.

Our contributions are threefold. (i) We formalize the hierarchical value gap as an execution residual and show that narrowing it requires a signal that distinguishes realization quality within each skill. (ii) We propose STEP, which uses in-sample team-value residuals to weight the decentralized executors' imitation losses and explicitly conditions consequence prediction on the executed skill code. (iii) On nine unseen SMAC maps, STEP improves zero-shot transfer primarily on mixed-quality data. Ablations attribute the gains to residual-weighted execution, while the tested retrieval extension yields no further benefit.

2 Related Work

Offline multi-task cooperative skill learning. ODIS discovers discrete coordination skills through conservative high-level value learning (Zhang et al., 2023); HiSSD learns common and task-specific continuous skills grounded in team dynamics and value (Liu et al., 2025a). Both improve transferable skill representation and selection, yet their skill-conditioned executors typically imitate logged actions equally. STEP extends team-value supervision to the executor, distinguishing quality differences among realizations of a selected skill.

In-sample policy extraction. Advantage-weighted regression, AWAC, and ICQ reweight imitation by advantages (Peng et al., 2019; Nair et al., 2020; Yang et al., 2021); IQL learns in-sample value with expectiles (Kostrikov et al., 2022), while OMAC combines expectile local values with advantage-weighted policy extraction in offline MARL (Wang & Zhan, 2023). COMAD reuses skills for advantage updates across sequential tasks, rather than fixed offline execution (Xiao et al., 2026). Other work in offline RL and MARL addresses distribution shift with conservative or constrained values, joint-behavior regularization, or in-sample updates (Kumar et al., 2020; Pan et al., 2022; Wang et al., 2023; Bui et al., 2025; Qiao et al., 2026; Kim & Sycara, 2026; Liu et al., 2025b). STEP instead weights skill-conditioned imitation with a shared in-sample team TD residual, favoring higher-value logged executions without added constraints.

Other routes to skill transfer. BiKT reuses tactic codebooks (Zhang et al., 2025), VO-MASD transfers subgroup skills (Chen et al., 2025), SD-CQL couples skill discovery with conservative action selection (Wang et al., 2025), and HyGen adds online interaction (Zhang et al., 2024). Retrieval requires an executor that uses its content, while online interaction is unavailable here. STEP acts directly on the executor's training objective, weighting logged executions by team value without changing deployment inputs; we test retrieval as an additional pathway.

3 Background and Problem Setup

3.1 Offline multi-task cooperative MARL

We consider cooperative MARL under centralized training and decentralized execution (CTDE) (Oliehoek & Amato, 2016). In each task k, n_k agents act from local histories and share a team

reward; training may additionally use the global state. Learning is restricted to fixed datasets from source tasks:

$$\mathcal{D} = \bigcup_{k \in \mathcal{T}_{\mathrm{src}}} \mathcal{D}_k, \qquad \mathcal{D}_k = \big\{(s_t, \boldsymbol{\tau}_t, \mathbf{a}_t, r_t, s_{t+1}, \boldsymbol{\tau}_{t+1}, d_t)\big\}. \tag{1}$$

where $\mathcal{T}_{\mathrm{src}}$ is the source-task set; s_t is the global state, $\boldsymbol{\tau}_t$ the agents' local histories, $\mathbf{a}_t$ their joint action, r_t the team reward, and d_t the termination indicator. The objective is zero-shot transfer to unseen tasks $\mathcal{T}_{\mathrm{tgt}}$ with different team sizes or opponent configurations, without target-task data or further interaction.

3.2 Hierarchical skill policies

A hierarchical policy selects a latent skill and executes it through decentralized controllers:

$$z_t \sim p_\theta(\cdot \mid x_t), \qquad \pi_\phi(\mathbf{a}_t \mid \boldsymbol{\tau}_t, z_t) = \prod_{i=1}^{n_k} \pi_i(a_{i,t} \mid \tau_{i,t}, z_t). \tag{2}$$

where x_t is the skill selector's context, z_t is the selected skill, and π_i maps agent i's local history and the skill to an action. Value and dynamics objectives can shape skill learning and selection, but the executor is commonly trained by conditional behavior cloning:

$$\mathcal{L}_{\mathrm{BC}}(\phi) = -\mathbb{E}_{\mathcal{D}} \left[\sum_{i=1}^{n_k} \log \pi_i(a_{i,t} \mid \tau_{i,t}, z_t) \right]. \tag{3}$$

where the expectation is over logged transitions and their associated skill codes; each valid transition has equal weight. This objective therefore treats better and worse logged realizations of the same skill alike.

3.3 The execution residual

Definition 1 (Execution residual) *For a fixed context x, consider a target favoring higher team-value realizations among logged skills and actions. If skill selection matches its marginal while the executor retains the logged action conditional, define the target and remaining discrepancy as*

$$q_w(z, \mathbf{a} \mid x) \propto \mu_h(z \mid x)\, \mu_\ell(\mathbf{a} \mid x, z)\, \exp(\delta/\alpha). \tag{4}$$

$$\mathcal{R}_{\mathrm{exec}} := \mathbb{E}_{q_w(z|x)} D_{\mathrm{KL}}(q_w(\mathbf{a} \mid x, z) \,\|\, \mu_\ell(\mathbf{a} \mid x, z)). \tag{5}$$

where μ_h and μ_ℓ are the dataset's skill and skill-conditioned joint-action distributions, δ scores a logged realization by team value, $\alpha > 0$ controls the tilt, and $q_w(z \mid x)$ and $q_w(\mathbf{a} \mid x, z)$ are the target's marginal and conditional distributions. Thus, q_w favors higher-value realizations without leaving logged support.

The KL residual is positive when effective action weights differ within a supported skill; matching skill selection alone cannot remove this gap. For stochastic outcomes, effective weights average over recorded outcomes conditional on the action as derived in Appendix A.1. STEP supplies this within-skill quality signal during executor training.

4 Method

4.1 Overview

To bridge the hierarchical value gap, we introduce Skill-conditioned Team-value Execution Projection (STEP). As Figure 1 illustrates, STEP extends team-value supervision from skill selection to skill-conditioned action execution. It comprises an in-sample expectile team value that distinguishes the quality of logged executions under the same skill, a team-residual execution projection that favors higher-value logged realizations within each skill, and execution-consistent consequence conditioning that makes the skill code used for action execution an explicit input to consequence prediction. Sections 4.2–4.4 describe each in turn.

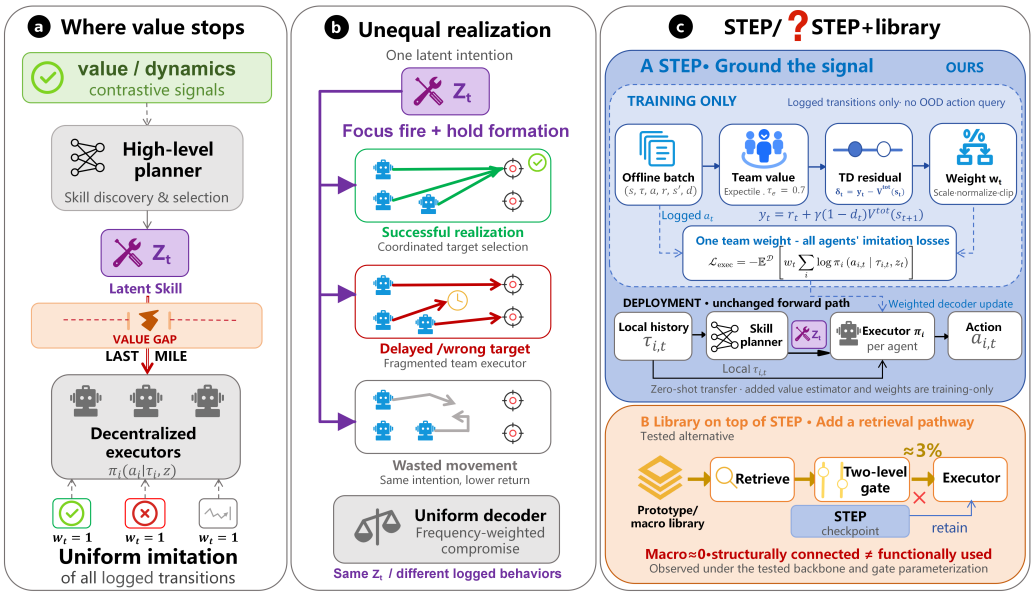


Figure 1: The hierarchical value gap and STEP. (a) Skill-level value does not distinguish logged executions. (b) Realizations of the same skill can have different outcomes. (c) STEP's team-value-guided execution pathway (top) and its separately evaluated extension with skill retrieval (bottom).

4.2 In-sample expectile team value

To distinguish different realizations of a skill, STEP learns a team-value estimator from fixed offline data. Per-agent utilities derived from the shared planner features are combined by an attention mixer:

$$v_{i,t} = v_\xi(h_{i,t}), \qquad V_\xi^{\mathrm{tot}}(s_t) = M_\xi(v_{1,t}, \ldots, v_{n_k,t}; s_t). \tag{6}$$

where $h_{i,t}$ is the own-agent feature produced by the shared planner, v_ξ is the added utility head, and M_ξ is its dedicated attention mixer conditioned on the global state. The notation $V_\xi^{\mathrm{tot}}(s_t)$ suppresses the history-dependent features. Global information is used only for training-time evaluation.

The estimator is added alongside the backbone's value module. It provides a single team-level score for each transition while retaining the backbone representations used across task configurations; STEP later uses this score as the baseline for execution weighting. Following expectile-based offline value learning (Kostrikov et al., 2022), it is trained on recorded rewards and successor states:

$$y_t = r_t + \gamma(1 - d_t)\ \mathrm{sg}\big[\bar{V}_\xi^{\mathrm{tot}}(s_{t+1})\big], \qquad \delta_t = y_t - V_\xi^{\mathrm{tot}}(s_t). \tag{7}$$

$$\mathcal{L}_V = \mathbb{E}_{\mathcal{D}_k}\big[|\tau_e - \mathbb{1}[\delta_t < 0]|\ \delta_t^2\big]. \tag{8}$$

where $\mathcal{D}_k$ contains valid transitions from source task k, r_t is the team reward, d_t the termination indicator, and γ the discount factor. V_ξ^{tot} denotes team value, with its planner-feature inputs suppressed; $\bar{V}_\xi^{\mathrm{tot}}$ uses a Polyak-updated value head and mixer. sg stops gradients, $\mathbb{1}[\cdot]$ is an indicator, and $\tau_e \in (1/2, 1)$ is the expectile level.

The asymmetric loss emphasizes positive TD errors, fitting an upper expectile of logged bootstrap targets. Hence δ_t scores a logged transition against the learned team baseline—an estimated signal, not an exact action advantage. STEP uses it to weight skill-conditioned executor training, without evaluating out-of-dataset actions or adding deployment inputs. We fix $\tau_e = 0.7$ across tasks; its interpretation, target updates, and numerical settings are detailed in Appendix A.2.

4.3 Team-residual execution projection

Team-value information must enter the executor's objective to distinguish realizations within a skill. STEP standardizes the residuals over valid batch entries and converts them into normalized, clipped

imitation weights:

$$\widetilde{\delta}_t = \frac{\delta_t - \mu_{\mathcal{B}}}{\alpha \max(\sigma_{\mathcal{B}}, \epsilon)}, \qquad w_t = \mathrm{sg}\left[\min\left(w_{\max}, \frac{\exp(\widetilde{\delta}_t)}{\mathbb{E}_{\mathcal{B}}[\exp(\widetilde{\delta})]}\right)\right]. \tag{9}$$

where $\mathcal{B}$ contains valid transitions, $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$ are their residual mean and standard deviation, $\alpha > 0$ controls weighting strength, $\epsilon > 0$ stabilizes division, and $w_{\max}$ caps the weight. Settings are reported in Appendix A.2; weights are detached during executor optimization.

The same team weight multiplies every agent's imitation loss for a recorded joint transition:

$$\mathcal{L}_{\mathrm{exec}}(\phi) = -\mathbb{E}_{\mathcal{D}_k}\left[\frac{w_t}{n_k}\sum_{i=1}^{n_k} \log \pi_{\phi,i}(a_{i,t} \mid \tau_{i,t}, z_t)\right]. \tag{10}$$

where n_k is the team size, $a_{i,t}$ is agent i's logged action, $\tau_{i,t}$ its local history, and z_t the execution skill condition. ϕ parameterizes the skill-conditioned executors. The expectation is within a source task; task sampling combines these objectives.

Unlike uniform behavior cloning, the shared w_t favors higher-residual realizations within each skill and ties agents' imitation updates to the logged team outcome without assigning individual credit; other controller objectives remain unchanged. Batch standardization uses relative residuals, mean normalization controls loss scale before clipping, and positive weights softly reweight rather than discard logged transitions. The factor $1/n_k$ prevents loss magnitude from growing solely with team size.

To connect this objective to the hierarchical value gap, let $\mu(e)$ be the empirical distribution of logged transitions with their skill conditions. Reweighting it induces an execution target and an equivalent projection objective:

$$\begin{aligned} q_w(e) &= \frac{\mu(e)w(e)}{C}, \quad C = \mathbb{E}_\mu[w], \quad q_{\mathrm{exec}}(\mathbf{a} \mid \boldsymbol{\tau}, z) = q_w(\mathbf{a} \mid \boldsymbol{\tau}, z), \\ \mathcal{L}_{\mathrm{exec}}(\phi) &= \frac{C}{n_k}\,\mathbb{E}_{q_w(\boldsymbol{\tau},z)}[D_{\mathrm{KL}}(q_{\mathrm{exec}}(\cdot \mid \boldsymbol{\tau}, z) \,\|\, \pi_\phi(\cdot \mid \boldsymbol{\tau}, z))] + \mathrm{const.} \end{aligned} \tag{11}$$

where e includes the recorded transition and its skill code, C normalizes the tilted distribution, and q_{exec} is its conditional joint-action marginal. $\boldsymbol{\tau}$ collects local histories, $\pi_\phi = \prod_i \pi_{\phi,i}$ is the decentralized product policy, and the constant is independent of ϕ.

Weighted imitation therefore projects a team-value-tilted execution target onto the decentralized policy class. It supplies the within-skill signal missing from the conditional execution component in Definition 1. More explicitly, for two supported joint actions under the same execution condition,

$$\frac{q_{\mathrm{exec}}(\mathbf{a} \mid h)}{q_{\mathrm{exec}}(\mathbf{a}' \mid h)} = \frac{\mu_\ell(\mathbf{a} \mid h)}{\mu_\ell(\mathbf{a}' \mid h)}\frac{\bar{w}(h, \mathbf{a})}{\bar{w}(h, \mathbf{a}')}, \qquad \bar{w}(h, \mathbf{a}) = \mathbb{E}_\mu[w(e) \mid h, \mathbf{a}]. \tag{12}$$

where $h = (\boldsymbol{\tau}, z)$ fixes the joint histories and skill, μ_ℓ is the logged action conditional, and $\bar{w}$ averages transition weights over recorded outcomes for a given action.

Equation (12) shows that within-skill action odds change only when outcome-averaged weights differ across actions; a condition-only multiplier cancels. STEP thus passes execution-quality differences to the executor even when value guides skill selection. The outcome average prevents treating a logged TD residual as a deterministic action value. Reweighting also shifts the context distribution; Appendix A.3 derives the projection under decentralized information constraints.

For fixed normalization statistics, w is positive and nondecreasing in δ. Within an execution condition,

$$\mathbb{E}_{q_w(\cdot \mid \boldsymbol{\tau}, z)}[\delta] - \mathbb{E}_{\mu(\cdot \mid \boldsymbol{\tau}, z)}[\delta] = \frac{\mathrm{Cov}_\mu(w, \delta \mid \boldsymbol{\tau}, z)}{\mathbb{E}_\mu[w \mid \boldsymbol{\tau}, z]} \geq 0. \tag{13}$$

where $\boldsymbol{\tau}$ collects the agents' local histories and z is the execution skill. $\mu(\cdot \mid \boldsymbol{\tau}, z)$ is the conditional distribution of recorded transitions and their realized outcomes, and $q_w(\cdot \mid \boldsymbol{\tau}, z)$ is its weighted counterpart from Equation (11). δ is the team TD residual in Equation (7), w its weight in Equation (9), and Cov_μ denotes covariance under the logged conditional distribution.

Positive conditional covariance strictly raises the reweighted target's mean residual; constant weights recover the logged conditional distribution. Thus STEP favors higher-scored executions within a skill only when residual weights distinguish them. This target-level result does not guarantee improved environment return after policy approximation. Appendix A.4 states the precise conditions; Mechanism and ablation studies examine the signal and its transfer effect.

4.4 Execution-consistent consequence conditioning

In a typical offline hierarchical multi-task cooperative MARL method (Liu et al., 2025a), consequence prediction learns skill representations from planner features, whereas action decoders use further-transformed codes. To condition prediction on the codes actually used for execution, STEP concatenates each active predictor feature with its corresponding decoder-driving code and projects the result back to the original dimension:

$$\widetilde{u}_t = W_u u_t + W_c c_t^{\text{exec}} + b, \qquad [W_u, W_c]_0 = [I, 0], \qquad b_0 = 0. \tag{14}$$

where u_t is the original predictor feature, c_t^{exec} the corresponding decoder-driving code, and W_u, W_c, b are trainable parameters. The subscript 0 denotes initialization and I is the identity. Separate projections operate on the active own-agent and enemy feature streams.

Initially, $\widetilde{u}_t = u_t$, preserving the original predictor mapping. The existing reconstruction objective is then evaluated on the fused features:

$$\widehat{Y}_t = f_\eta(\widetilde{u}_t), \qquad \ell_{\text{pred},t} = \ell_{\text{base}}(\widehat{Y}_t, Y_t). \tag{15}$$

where f_η is the existing consequence predictor, Y_t collects recorded decomposed entity states and action-related features in their task-dependent organization, and ℓ_{base} is the backbone's entity reconstruction loss with its original planner weighting and aggregation. STEP conditions the predictor on the decoder-driving code without changing these targets or the decoder's local inputs.

As the projection learns to use the execution code, prediction gradients also reach the transformations that produce it. STEP thereby establishes a direct training path from consequence prediction to the skill code actually used by the executor, complementing residual-weighted imitation. This method reuses logged entity targets and the backbone's reconstruction loss. Appendix A.5 details the gradient path; Appendix B describes joint training and zero-shot evaluation.

5 Experiments

5.1 Experimental Settings

Tasks and data. We evaluate on the StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019). Marine-Hard is the primary task set, with policies trained on three source maps and transferred zero-shot to nine unseen maps. Each ource map provides expert (HE), medium (HM), medium-expert (HME), and medium-replay (HMR) offline data. Marine-Easy and the heterogeneous Stalker–Zealot set serve as secondary transfer checks. Appendix C details the task sets and data; Appendix D the training protocol. STEP and its backbone share five seeds, so all comparisons are paired. **Baselines.** Following Liu et al. (2025a), we compare with transformer behavior cloning (BC-t), its return-conditioned counterpart (BC-r), the offline multi-task UPDeT-m (Hu et al., 2021), and the skill-discovery method ODIS (Zhang et al., 2023). BC-best denotes the better of BC-t and BC-r on each task. The reproduced HiSSD backbone is the direct control for STEP, isolating the effect of extending team-value supervision from skill selection to skill-conditioned execution. **Evaluation.** Following ODIS and HiSSD (Zhang et al., 2023; Liu et al., 2025a), we report best-over-training win rates on the standard evaluation grid.

5.2 Main Results

We evaluate STEP and the baselines on Marine-Hard, with the results reported in Table 1. The key results are as follows: (i) STEP achieves the best or tied-best performance in 37 of the 48 task–data-regime comparisons, demonstrating strong zero-shot transfer across team configurations. (ii) Compared with its HiSSD backbone, STEP attains a higher average unseen-task win rate in three

Table 1: Test win rates (%) of the best policies, averaged over five random seeds, on Marine-Hard across four offline-data qualities. Asymmetric task names are abbreviated (e.g., `5m6m` denotes `5m_vs_6m`), and BC-best reports the better result of BC-t and BC-r. STEP results are bolded when their mean exceeds the HiSSD backbone.

Tasks	Expert					Medium				
	BC-best	UPDeT-m	ODIS	Backbone (HiSSD)	STEP (Ours)	BC-best	UPDeT-m	ODIS	Backbone (HiSSD)	STEP (Ours)
Source Tasks										
3m	97.7±2.6	82.8±16.0	98.4±2.7	100.0±0.0	100.0±0.0	65.4±14.7	51.2±3.4	85.9±10.5	72.4±1.2	**77.7±1.3**
5m6m	50.4±2.3	17.2±28.0	53.9±5.1	77.0±5.0	76.1±2.1	21.9±3.4	6.3±4.9	22.7±7.1	39.5±1.9	**59.0±2.2**
9m10m	95.3±1.6	3.1±5.4	80.4±8.7	99.6±0.3	**100.0±0.0**	63.8±10.9	28.5±10.2	78.1±3.8	82.6±2.3	**85.5±3.8**
Unseen Tasks										
4m	92.1±3.5	33.0±27.1	95.3±3.5	100.0±0.0	100.0±0.0	48.8±21.1	14.1±5.2	61.7±17.7	82.7±11.1	**89.8±3.8**
5m	87.1±10.5	33.6±40.2	89.1±10.0	100.0±0.0	100.0±0.0	76.6±14.1	67.2±21.3	85.9±11.8	94.8±9.1	**98.4±3.3**
10m	90.5±3.8	54.7±44.4	93.8±2.2	99.4±1.4	**100.0±0.0**	56.2±20.6	32.9±11.3	61.3±11.3	99.1±1.0	98.0±1.4
12m	70.8±15.2	17.2±28.0	58.6±11.8	94.0±8.2	**95.4±4.5**	24.0±10.5	3.2±3.8	35.9±8.1	96.9±1.8	91.0±4.5
7m8m	18.8±3.1	0.0±0.0	25.0±15.1	52.4±4.8	43.1±10.7	1.6±1.6	0.0±0.0	28.1±22.0	16.8±7.1	**22.6±4.1**
8m9m	15.8±3.3	0.0±0.0	19.6±6.0	72.0±14.5	70.2±10.3	3.1±3.8	2.3±2.6	4.7±2.7	21.0±9.1	**21.6±5.5**
10m11m	45.3±11.1	0.0±0.0	42.4±7.2	91.1±9.5	**92.5±5.5**	19.7±8.9	4.0±3.4	29.7±15.4	50.6±5.4	**56.5±7.4**
10m12m	1.0±1.5	0.0±0.0	1.6±1.6	17.3±10.0	17.1±6.2	0.0±0.0	0.0±0.0	1.6±1.6	2.4±0.8	**2.7±0.6**
13m15m	0.0±0.0	0.0±0.0	2.3±2.6	6.2±4.4	**9.0±4.6**	0.6±1.3	0.0±0.0	1.6±1.6	2.6±1.5	**3.9±2.1**
Avg. (unseen)	–	–	–	70.26±2.03	69.71±2.77	–	–	–	51.87±3.52	**53.83±1.53**

Tasks	Medium-Expert					Medium-Replay				
	BC-best	UPDeT-m	ODIS	Backbone (HiSSD)	STEP (Ours)	BC-best	UPDeT-m	ODIS	Backbone (HiSSD)	STEP (Ours)
Source Tasks										
3m	67.7±23.7	85.2±17.9	73.6±22.0	98.6±1.7	**100.0±0.0**	81.1±8.8	41.4±20.1	83.6±14.0	92.1±2.9	92.0±1.0
5m6m	31.3±6.3	1.6±1.6	9.4±2.2	50.1±3.7	**73.3±3.8**	25.0±3.1	0.8±1.4	16.6±4.7	35.8±4.6	**44.1±6.8**
9m10m	26.0±13.9	24.3±18.7	31.3±14.5	91.0±7.3	**98.8±1.0**	33.4±13.1	0.8±1.4	34.4±8.0	54.5±4.6	**62.9±4.7**
Unseen Tasks										
4m	81.3±18.9	43.9±39.0	82.8±13.5	99.5±0.8	99.4±0.4	61.5±9.0	35.9±12.6	55.6±14.5	83.2±4.1	**92.8±4.6**
5m	74.0±2.9	33.6±40.2	82.8±17.7	99.9±0.3	**100.0±0.0**	75.0±24.2	61.7±20.3	96.1±4.1	99.9±0.3	99.6±0.3
10m	78.1±6.7	32.8±38.1	82.8±16.8	98.9±1.2	**99.6±0.6**	82.4±8.2	11.0±7.8	84.4±15.1	98.2±1.0	**99.4±0.6**
12m	64.8±24.3	9.4±8.6	81.3±20.6	91.4±4.3	**96.8±2.3**	83.4±4.5	2.3±2.6	84.4±6.6	92.9±5.9	**97.5±1.9**
7m8m	13.3±4.5	2.3±4.1	15.6±4.4	42.5±6.6	**49.6±18.4**	7.3±6.4	1.6±2.7	9.4±2.2	20.9±3.6	**38.0±6.5**
8m9m	10.2±4.6	9.5±8.6	10.9±4.7	44.8±10.1	**60.3±11.2**	11.5±3.9	0.8±1.4	11.7±8.7	18.9±4.5	**19.2±4.2**
10m11m	26.6±4.7	11.8±8.1	33.6±8.9	73.5±15.6	**88.8±5.9**	46.8±6.6	0.8±1.4	35.9±5.2	48.8±3.8	**56.9±3.5**
10m12m	0.0±0.0	0.0±0.0	1.6±1.6	8.2±2.9	**13.0±1.7**	1.6±2.7	0.0±0.0	2.3±1.4	2.6±0.7	**3.1±0.8**
13m15m	0.8±1.4	0.0±0.0	2.3±2.6	6.4±4.9	6.4±2.5	1.6±1.6	0.0±0.0	2.4±1.4	6.5±3.6	**8.5±3.2**
Avg. (unseen)	–	–	–	62.78±1.51	**68.19±2.69**	–	–	–	52.43±1.17	**57.22±1.69**

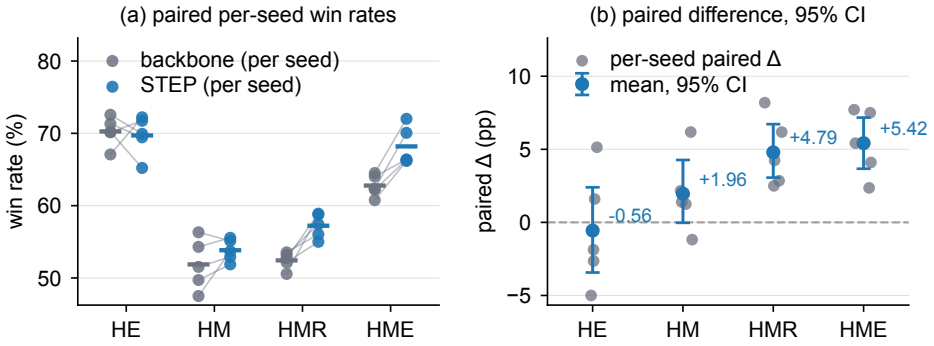


Figure 2: STEP versus the backbone on unseen Marine-Hard maps. (a) Win rates of five paired seeds (grey: backbone; blue: STEP); bars mark means. (b) Paired differences (grey), mean and bootstrap 95% CI (blue); dashed: zero. HE, HM, HMR, and HME denote the four data regimes.

of the four data regimes, with the largest gains on HME (+5.42 points) and HMR (+4.79 points). These results show that extending team-value supervision to skill-conditioned execution improves the transfer of learned cooperation, particularly when the offline data contain behaviors of varying quality. Results on Marine-Easy and Stalker–Zealot are provided in Appendix E. Appendix F confirms the pattern at the final checkpoint, with learning curves in Appendix H.

5.3 Mechanism Analysis

To examine how STEP improves zero-shot transfer, we conduct complementary diagnostics on Marine-Hard. We first compare paired transfer gains across offline data qualities, then inspect which logged episodes receive high execution weights and where gains arise among unseen maps. These analyses test whether team-value guidance can distinguish useful execution experience and whether its benefits extend across target configurations.

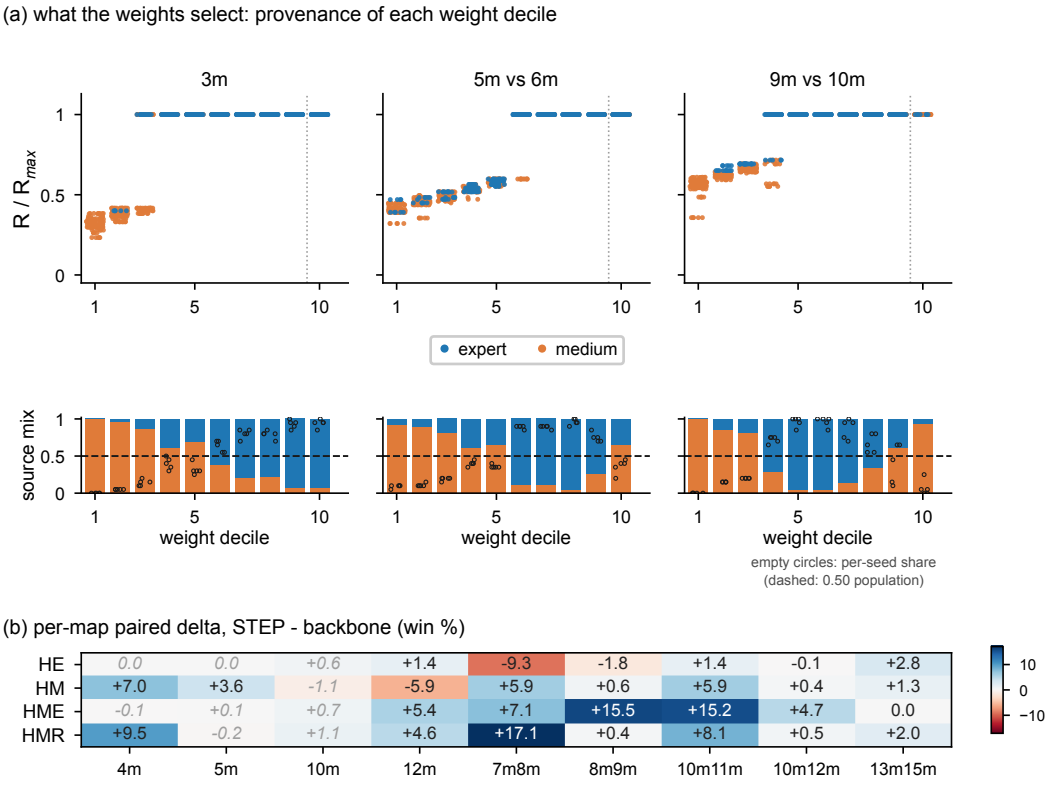


Figure 3: Weight selection and transfer by map. (a) HME episodes grouped by execution-weight decile, with expert-origin share shown below. (b) Mean paired win-rate change on unseen maps (STEP − backbone); gray italic labels denote backbone win rates at or above 98%.

Figure 2 shows that STEP gains more on medium-expert and medium-replay data than on expert data: the paired results favor STEP in the mixed-quality regimes but show little separation on expert data. The logged-return analysis in Appendix G explains this contrast. Mixed-quality logs contain cooperative executions with distinguishable outcomes, whereas some expert source tasks are nearly saturated. STEP therefore has the clearest opportunity to improve skill-conditioned execution when its offline data contain quality differences that the value residual can rank.

To examine what the weights select, Figure 3a groups source episodes by execution weight. Higher-weight groups contain better realized team outcomes from both expert and medium sources; the weights do not simply follow the source label. Figure 3b shows positive transfer changes on several unseen maps where the backbone still has room to improve, while the near-ceiling maps marked in gray italics change little. Thus, STEP's gains occur where informative source experience can guide execution and the target task leaves room for better coordination. Further measurements and the limits of episode-level interpretation appear in Appendix G.

5.4 Ablation Study

To locate STEP's transfer gains, we add to the HiSSD backbone (A) an in-sample team-value estimator (B), residual-weighted execution (C), and consequence conditioning (D, full STEP) across

four Marine-Hard data regimes. Figure 4 plots the paired gains along this chain, and Appendix Table 12 lists the full numbers. The main improvement appears with residual-weighted execution, especially on medium-expert and medium-replay data, indicating that selecting higher-value realizations within each skill drives the observed transfer gains.

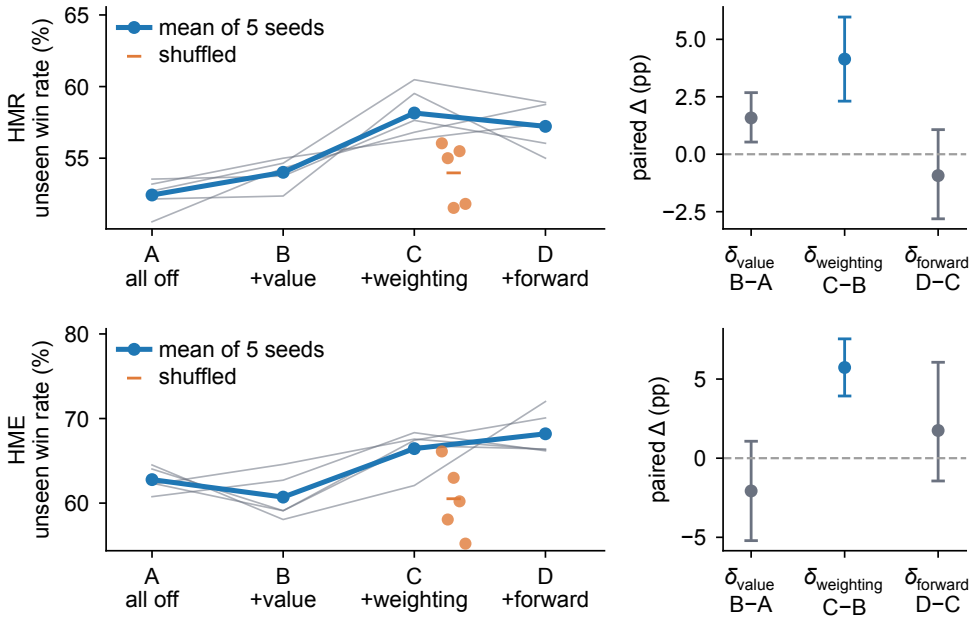


Figure 4: Accumulation chain on HMR (top) and HME (bottom). Left: average unseen-map win rate of arms A–D per seed (grey) and mean (blue); orange marks C-shuf on HMR. Right: paired differences between consecutive arms with 95% CIs.

On medium-replay data, shuffling C's weights across valid batch samples (C-shuf) removes the gain while leaving the weighting procedure intact. Appendix Table 11 shows that keeping only high-return episodes is weaker than graded weighting on both medium-replay and medium-expert data. These controls indicate that the correspondence between a logged execution and its value weight matters, especially when behavior quality varies continuously. Supporting comparisons are in Appendices F and G.

To test whether explicit skill reuse adds to these execution gains, we attach a codebook and segment library to paired STEP checkpoints and continue training (Zhang et al., 2025). The retrieval pathway yields no resolved improvement across the four data regimes; Appendix Table 9 reports the full comparison. Its gate admits only about 3% retrieved content, and the segment library retains few effective entries, as detailed in Appendix E.2. In this tested design, STEP's transfer gains therefore arise from value-guided executor training, with no established added benefit from skill retrieval.

6 Conclusion

We study zero-shot skill transfer in offline multi-task cooperative MARL and identify a hierarchical value gap: recognizing a valuable cooperative skill does not distinguish its execution quality. During training, STEP uses in-sample team-value residuals to guide skill-conditioned decentralized execution and conditions consequence prediction on the executed skill code, addressing this gap. On unseen SMAC tasks, STEP achieves its clearest transfer gains when offline data contain cooperative behaviors of varying quality. Mechanism analysis shows that its weights favor executions with better realized team outcomes across data sources, with gains on target maps that allow further coordination. Ablations attribute the main gains to residual-weighted execution; the tested retrieval pathway adds no established benefit. These findings show that transferable coordination depends on carrying team-value information through to action execution. Future work will extend and test STEP in more environments, including MAMuJoCo.

AI USE STATEMENT

In this work, we used LLMs as assistive tools in three ways, all under author supervision. (1) Writing: drafting and polishing text and restructuring sections. The authors reviewed and revised all AI-assisted text and checked every claim against the original results. (2) Analysis code: writing scripts that parse experiment logs, compute the reported statistics, and generate figures and LaTeX tables. (3) Implementation: routine coding assistance for the training and evaluation pipelines. All suggested code was reviewed and tested by the authors. The research ideas, problem formulation, method design, experimental protocol, and all empirical claims come from the authors. The authors take full responsibility for all content.

ETHICS STATEMENT

This work complies with the ICLR Code of Ethics. All experiments are conducted in simulation (the SMAC StarCraft II benchmark) using publicly released offline datasets; the work involves no human subjects or personal data, and we foresee no direct negative societal impact beyond that common to reinforcement learning research.

REPRODUCIBILITY STATEMENT

Source code, training configurations, and the scripts that derive the dataset splits and build the skill libraries are included in the supplementary material; the offline datasets are the publicly released multi-task SMAC datasets of Zhang et al. (2023). The task sets and offline data qualities are described in Appendix C; the implementation and training details, including all hyperparameters and every protocol difference from prior work, are in Appendix D; and the baselines follow Liu et al. (2025a), as listed in Section 5.1. Final-checkpoint results are reported in Appendix F, full pseudocode is given in Appendix B, and the derivations and proofs of both propositions are in Appendix A.

REFERENCES

Akshat Agarwal, Sumit Kumar, Katia Sycara, and Mike Lewis. Learning transferable cooperative behavior in multi-agent teams. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 2020.

The Viet Bui, Thanh Nguyen, and Tien Mai. Comadice: Offline cooperative multi-agent reinforcement learning with stationary distribution shift regularization. In *International Conference on Learning Representations*, 2025.

Jiayu Chen, Tian Lan, and Vaneet Aggarwal. Variational offline multi-agent skill discovery. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2025.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. Updet: Universal multi-agent reinforcement learning via policy decoupling with transformers. In *International Conference on Learning Representations*, 2021.

Shariq Iqbal, Christian A Schroeder De Witt, Bei Peng, Wendelin Böhmer, Shimon Whiteson, and Fei Sha. Randomized entity-wise factorization for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 4596–4606. PMLR, 2021.

Woojun Kim and Katia Sycara. B3c: A minimalist approach to offline multi-agent reinforcement learning. In *Proceedings of the 25th International Conference on Autonomous Agents and Multi-agent Systems*, 2026.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Sicong Liu, Yang Shu, Chenjuan Guo, and Bin Yang. Learning generalizable skills from offline multi-task data for multi-agent cooperation. In *International Conference on Learning Representations*, 2025a.

Yuntao Liu, Yuan Li, Xinhai Xu, Yong Dou, and Donghong Liu. Heterogeneous skill learning for multi-agent tasks. *Advances in neural information processing systems*, 35:37011–37023, 2022.

Zongkai Liu, Qian Lin, Chao Yu, Xiawei Wu, Yile Liang, Donghui Li, and Xuetao Ding. Offline multi-agent reinforcement learning via in-sample sequential policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19068–19076, 2025b.

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

Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

Ling Pan, Longbo Huang, Tengyu Ma, and Huazhe Xu. Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification. In *International conference on machine learning*, pp. 17221–17237. PMLR, 2022.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Dan Qiao, Wenhao Li, Shanchao Yang, Hongyuan Zha, and Baoxiang Wang. Offline multi-agent reinforcement learning via sequential score decomposition. In *International Conference on Machine Learning*, 2026.

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

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

Xiangsen Wang and Xianyuan Zhan. Offline multi-agent reinforcement learning with coupled value factorization. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems*, 2023.

Xiangsen Wang, Haoran Xu, Yinan Zheng, and Xianyuan Zhan. Offline multi-agent reinforcement learning with implicit global-to-local value regularization. *Advances in Neural Information Processing Systems*, 36:52413–52429, 2023.

Xun Wang, Zhuoran Li, Hai Zhong, and Longbo Huang. Beyond shallow behavior: Task-efficient value-based multi-task offline marl via skill discovery. *arXiv preprint arXiv:2502.08985*, 2025.

Yuchen Xiao, Lei Yuan, Ruiqi Xue, Tieyue Yin, and Yang Yu. Offline multi-agent continual cooperation via skill partition and reuse. *arXiv preprint arXiv:2606.25389*, 2026.

Jiachen Yang, Igor Borovikov, and Hongyuan Zha. Hierarchical cooperative multi-agent reinforcement learning with skill discovery. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 2020.

Mingyu Yang, Yaodong Yang, Zhenbo Lu, Wengang Zhou, and Houqiang Li. Hierarchical multi-agent skill discovery. *Advances in Neural Information Processing Systems*, 36:61759–61776, 2023.

Yiqin Yang, Xiaoteng Ma, Chenghao Li, Zewu Zheng, Qiyuan Zhang, Gao Huang, Jun Yang, and Qianchuan Zhao. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:10299–10312, 2021.

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

Fuxiang Zhang, Chengxing Jia, Yi-Chen Li, Lei Yuan, Yang Yu, and Zongzhang Zhang. Discovering generalizable multi-agent coordination skills from multi-task offline data. In *The Eleventh International Conference on Learning Representations*, 2023.

Junkai Zhang, Jinmin He, Yifan Zhang, Yifan Zang, Ning Xu, and Jian Cheng. Bi-level knowledge transfer for multi-task multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 126266–126295, 2025.

Mingliang Zhang, Sichang Su, Chengyang He, and Guillaume Sartoretti. Hybrid training for enhanced multi-task generalization in multi-agent reinforcement learning. *arXiv preprint arXiv:2408.13567*, 2024.

A Derivations and methodological details

A.1 Hierarchical decomposition and realized outcomes

For a context x, the target in Equation (4) factorizes into a skill marginal and a conditional execution distribution. Its KL divergence from a hierarchical policy decomposes as

$$D_{\mathrm{KL}}(q_w(z, \mathbf{a} \mid x) \,\|\, p(z \mid x)\pi(\mathbf{a} \mid x, z)) = D_{\mathrm{KL}}(q_w(z \mid x) \,\|\, p(z \mid x)) + \mathbb{E}_{q_w(z|x)} D_{\mathrm{KL}}(q_w(\mathbf{a} \mid x, z) \,\|\, \pi(\mathbf{a} \mid x, z)) \,. \tag{16}$$

where p is the skill selector, π is the conditional executor, and q_w is the value-tilted target. The identity follows by splitting the joint log-density ratio into its marginal and conditional terms.

If $p = q_w(z \mid x)$ while $\pi = \mu_\ell$, the first term vanishes and the second is Definition 1. With the execution conditional held fixed, improving the skill marginal alone cannot remove this discrepancy. When rewards or successor states are stochastic, a recorded TD residual is a score of a realized transition, not a deterministic function of its action. In that case, the action target must marginalize over outcomes:

$$q_w(\mathbf{a} \mid x, z) = \frac{\mu_\ell(\mathbf{a} \mid x, z)\, \bar{w}(x, z, \mathbf{a})}{Z(x, z)}, \qquad \bar{w}(x, z, \mathbf{a}) = \mathbb{E}_\mu[w(e) \mid x, z, \mathbf{a}], \qquad Z(x, z) = \mathbb{E}_\mu[w(e) \mid x, z]. \tag{17}$$

where e is a full recorded transition with its skill condition, $\bar{w}$ is its action-conditional mean weight, and Z normalizes the conditional distribution. For the idealized exponential target in Equation (4), $w(e) = \exp(\delta(e)/\alpha)$; the implementation uses Equation (9). There the same α divides the batch-standardized residual, so the implemented effective temperature is $\alpha \max(\sigma_{\mathcal{B}}, \epsilon)$.

Consequently, the execution residual is zero exactly when the effective action weights $\bar{w}(x, z, \mathbf{a})$ are constant over supported actions for each skill of positive target probability. Variation between realized outcomes alone is insufficient if it averages to the same weight for every action. Section 4.3 conditions on the complete execution context $(\boldsymbol{\tau}, z)$; the same decomposition applies, while the deployed policy remains restricted to each agent's local history.

A.2 Expectile baseline and numerical settings

Lemma 1 (Expectile tilt) *Let Y have a finite second moment. The minimizer v_τ of $\mathbb{E}[|\tau - \mathbb{1}[Y - v < 0]|(Y - v)^2]$, for $\tau \in (0, 1)$, is nondecreasing in τ. If Y is nonconstant and $\tau > 1/2$, then $v_\tau > \mathbb{E}[Y]$.*

Proof. Differentiating the loss yields

$$\ell'_\tau(v) = -2\left\{\tau \mathbb{E}[(Y - v)_+] - (1 - \tau)\mathbb{E}[(v - Y)_+]\right\}, \qquad \tau \mathbb{E}[(Y - v_\tau)_+] = (1 - \tau)\mathbb{E}[(v_\tau - Y)_+]. \tag{18}$$

where $(r)_+ = \max(r, 0)$ and Y is a fixed bootstrap-target random variable. The second line is the first-order condition. At $v = \mathbb{E}[Y]$, the positive and negative deviations have the same positive mean for nonconstant Y, so $\ell'_\tau(v) < 0$ when $\tau > 1/2$. Convexity places the minimizer above the mean. In the first-order condition, the left deviation decreases and the right deviation increases with v, giving monotonicity in τ.

This result characterizes an exact expectile fit for a fixed target distribution. A neural value estimate learned with shared features and moving targets only approximates that fit; neither Lemma 1 nor Equation (8) guarantees an exact advantage estimate or an upper bound on return.

Fixed settings. We use $\tau_e = 0.7$, $\alpha = 1$, $w_{\max} = 100$, and $\epsilon = 10^{-6}$ across tasks. The expectile level emphasizes positive residuals while retaining a penalty on negative residuals; it is a fixed experimental setting; we make no optimality or sensitivity claim for 0.7. The weight standard deviation is the population standard deviation over valid batch positions, floored at 10^{-6}. The exponential-weight mean is also floored at 10^{-6} in the implementation. Padding and invalid post-termination entries are excluded from these statistics and from the losses.

Target updates. The estimator reads live shared planner features. Its target uses detached features and a separate target value head and mixer, updated by Polyak averaging:

$$\bar{\xi} \leftarrow (1 - \tau_p)\bar{\xi} + \tau_p \xi, \qquad \tau_p = 0.005. \tag{19}$$

where ξ and $\bar{\xi}$ here denote the online and target value-head/mixer parameters, respectively, and τ_p is the averaging rate. The shared planner trunk is not a separately Polyak-averaged target trunk. Residuals are computed and detached in the value forward pass before its optimizer step. The executor consumes those cached residuals from the same batch; it does not recompute them after that step. The update sequence is given in Appendix B.

A.3 Projection under decentralized information constraints

Proposition 1 (In-sample execution projection) *Fix a source task and a nonnegative detached weight with $0 < C = \mathbb{E}_\mu[w] < \infty$. Weighted maximum likelihood over its logged actions minimizes the conditional forward KL in Equation (11), with contexts distributed according to $q_w(\boldsymbol{\tau}, z)$.*

Proof. Write $h = (\boldsymbol{\tau}, z)$. Changing measure from $\mu(e)$ to $q_w(e) = \mu(e)w(e)/C$ gives

$$\begin{aligned} -\mathbb{E}_\mu[w \log \pi_\phi(\mathbf{a} \mid h)] &= -C\, \mathbb{E}_{q_w(h)} \mathbb{E}_{q_{\text{exec}}(\cdot \mid h)}[\log \pi_\phi(\mathbf{a} \mid h)] \\ &= C\, \mathbb{E}_{q_w(h)} \left[D_{\text{KL}}(q_{\text{exec}} \| \pi_\phi) + \mathcal{H}(q_{\text{exec}}) \right]. \end{aligned} \tag{20}$$

where h is the analytical execution context, $\mathcal{H}$ is conditional action entropy at that context, and both distributions in the final line are conditioned on h. The entropy is independent of ϕ; division by n_k yields Equation (11).

The logarithm of a product policy decomposes into a sum of local log-likelihoods, explaining why the same weight multiplies all agents' losses. An unrestricted local decoder fits $q_w(a_i \mid \tau_i, z)$, the target marginal conditional on its available information; it need not fit the full-context marginal $q_w(a_i \mid \boldsymbol{\tau}, z)$ separately for every joint history. Partial observability and parameter sharing can therefore leave approximation error and prevent exact recovery of correlated joint actions. The identity is an objective equivalence, not a guarantee of exact representability.

The normalizer in Equation (9) rescales weights by a positive batch constant before clipping. Clipping changes the target distribution, so the projection uses the actual clipped weights in both its target and its objective. Across source tasks, the sampler aggregates task-specific objectives, including their $1/n_k$ agent averaging. For an expected minibatch objective with random batch statistics, the analogous target uses the batch-averaged effective weight; the fixed-weight identity applies to each realized batch.

A.4 Residual preference and its limits

Proposition 2 (Within-skill target improvement) *Fix $h = (\boldsymbol{\tau}, z)$ and the batch normalization statistics. If $w = g(\delta) > 0$ with g nondecreasing and the relevant moments finite, then Equation (13) holds. The inequality is strict if and only if $\text{Cov}_\mu(w, \delta \mid h) > 0$.*

Proof. Conditional reweighting gives

$$\begin{aligned} \mathbb{E}_{q_w(\cdot \mid h)}[\delta] - \mathbb{E}_\mu[\delta \mid h] &= \frac{\mathbb{E}_\mu[w\delta \mid h] - \mathbb{E}_\mu[w \mid h]\mathbb{E}_\mu[\delta \mid h]}{\mathbb{E}_\mu[w \mid h]}, \\ 2\,\text{Cov}_\mu(g(\delta), \delta \mid h) &= \mathbb{E}[(g(\delta) - g(\delta'))(\delta - \delta') \mid h] \geq 0. \end{aligned} \tag{21}$$

where δ' is an independent copy of δ under the same conditional logged distribution. Monotonicity makes the product in the second line nonnegative.

If g is strictly increasing, a nonconstant residual suffices for strict inequality. With clipped weights, distinct residuals can receive identical weights, so nondegeneracy alone is insufficient; the positive-covariance condition remains exact. Constant weights recover the logged conditional distribution.

The result concerns the weighted distribution of recorded outcomes. It does not establish that policy execution will reproduce those outcomes, that the KL residual relative to every idealized exponential target strictly decreases, or that environment return necessarily improves. Stochastic outcomes, value-estimation error, and restrictions of decentralized policies distinguish these statements. Sections 5.3 and 5.4 evaluate the empirical relationship between residual weighting and transfer performance.

A.5 Consequence conditioning and the information bound

Equation (14) implements a learned projection of concatenated original features and action-stream skill codes. Separate maps are used for the own-agent and enemy streams; the unused ally input of the existing reconstruction module is not given an additional map. The target collection Y_t in Equation (15) is unchanged, including the entity-state and recorded action-related features used by the backbone. Its reconstruction errors retain the backbone planner's original weighting.

For one stream, the direct derivative through the injected code is

$$\left.\frac{\partial \ell_{\mathrm{pred},t}}{\partial c_t^{\mathrm{exec}}}\right|_{u_t \text{ fixed}} = W_c^{\mathsf{T}} \frac{\partial \ell_{\mathrm{pred},t}}{\partial \widetilde{u}_t}. \tag{22}$$

where W_c is the code block of the projection and u_t is held fixed to isolate the explicit code path. Other gradients through shared features may also exist. At initialization $W_c = 0$, this direct code gradient is zero, although the projection can receive a gradient. As its code block learns, the path can train the action-code transformations. This supplies an explicit conditioning path; it neither forces equality of the original and execution codes nor guarantees that the predictor uses all information in the added code.

The usual variational information bound applies to the fused predictor input:

$$I(\widetilde{U}; Y) = H(Y) - H(Y \mid \widetilde{U}) \geq H(Y) + \mathbb{E}[\log p_\eta(Y \mid \widetilde{U})]. \tag{23}$$

where $\widetilde{U}$ is the fused feature random variable, Y is the recorded target collection, p_η is a conditional predictive density, and I and H denote mutual information and entropy. The statement assumes the quantities are well defined. The inequality follows from the nonnegativity of conditional KL divergence. For a fixed-variance Gaussian observation model, the unweighted squared reconstruction error corresponds to negative log-likelihood up to positive scale and additive constants. This explains the reconstruction component; the backbone's additional planner weighting should not be silently identified with an unweighted information bound. In particular, the bound on $I(\widetilde{U}; Y)$ does not certify an increase in $I(C^{\mathrm{exec}}; Y)$ alone. We therefore describe the component as explicit execution-code conditioning, not guaranteed representational equality or an independent source of return improvement.

B Training and evaluation procedure

After backbone pretraining, STEP samples source-task batches and sequentially updates the expectile team-value estimator, the residual-weighted executor, and the execution-conditioned predictor. On unseen tasks, agents use the original decentralized policy without target data or adaptation; the added training modules do not change policy inputs or communication. Algorithm 1 specifies the update order, masking, and retained backbone objectives. Numerical settings are given in Appendix A.2; the residual used by the executor is cached before the value optimizer step.

Masks and control arm. Target averaging affects only the dedicated value head and mixer. The recorded residual is padded for tensor compatibility, but padded entries do not enter the normalization or executor loss. Action cross-entropies are computed per timestep and agent, multiplied by the shared team-time weight, and averaged over valid agent-time entries. STEP replaces only uniform action imitation; the backbone's task-specific representation objectives, controller regularizers, and planner weighting are retained. The shuffled-weight ablation permutes weights only across valid batch positions, preserving their multiset and changing their association with logged actions. It is a diagnostic control, not an additional STEP training stage.

C Benchmarks and datasets

SMAC. The StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) is a cooperative multi-agent benchmark built on StarCraft II micromanagement. It is the standard testbed for centralized-training decentralized-execution methods, from counterfactual policy gradients and additive or monotonic value decomposition to duplex dueling critics and multi-agent PPO (Foerster et al., 2018; Sunehag et al., 2018; Rashid et al., 2020; Wang et al., 2021; Yu et al., 2022). Skill

Algorithm 1 STEP training and zero-shot evaluation

Require: source-task datasets $\{\mathcal{D}_k\}$; the pretrained backbone, comprising the shared planner, the skill-conditioned executors $\pi_{\phi,i}$ and the consequence predictor f_η; the added value head and mixer with parameters ξ and their target copy $\bar{\xi}$; the injection parameters W_u, W_c, b of Equation (14), initialized at $[W_u, W_c]_0 = [I, 0]$ and $b_0 = 0$. All hyper-parameters, including the batch size and the training-step budget, are listed in Table 5.

1: **repeat**
2: Sample a task k and a batch $\mathcal{B} \sim \mathcal{D}_k$ of recorded joint transitions with validity masks.
3: $\delta_t \leftarrow y_t - V_\xi^{\mathrm{tot}}(s_t)$ and $\mathcal{L}_V$ using Equations (7) and (8); cache $\mathrm{sg}(\delta_t)$ on the batch.
4: Update ξ with $\mathcal{L}_V$ and the backbone value objective; $\bar{\xi} \leftarrow (1-\tau_p)\bar{\xi}+\tau_p\xi$ using Equation (19).
5: $w_t \leftarrow$ Equation (9), computed from the cached residuals and excluding invalid positions.
6: Update ϕ with $\mathcal{L}_{\mathrm{exec}}$ of Equation (10) and the original auxiliary objectives of the controller.
7: $\widetilde{u}_t \leftarrow W_u u_t + W_c c_t^{\mathrm{exec}} + b$; compute the original planner objective, whose reconstruction component uses Equation (15).
8: The outer training loop applies the optimizer step for the resulting planner-side gradients.
9: **until** the training-step budget is reached
10: **Evaluation:** on each unseen task, run the original decentralized skill-and-action pathway from local information, without target-task data or parameter updates. The added value module, weights, and predictor-side projections are not used to generate actions.

discovery on this benchmark has a longer history in the online setting, where hierarchical and heterogeneous skills are learned through environment interaction (Yang et al., 2020; Liu et al., 2022; Yang et al., 2023); the offline multi-task setting used here removes that interaction. Each allied unit is controlled by a decentralized agent that observes only its local field of view, enemy units are driven by the built-in game AI, and the team receives a shared shaped reward for damage dealt, enemy deaths, and victory; an episode is won when every enemy unit is eliminated. For zero-shot skill transfer, maps that share unit types but differ in team composition are grouped into task sets (Agarwal et al., 2020): policies are trained offline on a few source tasks of a set and evaluated on its remaining tasks without any further data or gradient updates.

Task sets. Table 2 lists the three task sets. Marine-Hard, the main benchmark, trains on three source tasks of which two are asymmetric and transfers to nine unseen tasks up to `13m_vs_15m`; its source data already demand outnumbered coordination. Marine-Easy uses symmetric compositions only, and Stalker–Zealot replaces the homogeneous marines with two unit types of different range, speed, and durability. Results on both are reported in Appendix E.

Table 2: Task sets used in this paper. A task $n\mathtt{m}$ pits n allied against n enemy Marines (symmetric); $n\mathtt{m_vs_}k\mathtt{m}$ pits n allied against $k > n$ enemy Marines (asymmetric); $n\mathtt{s}k\mathtt{z}$ pits n Stalkers and k Zealots against a mirrored enemy team.

Task set	Type	Tasks	Properties
Marine-Hard	Source	3m, 5m_vs_6m, 9m_vs_10m	homogeneous, sym. & asym.
	Unseen	4m, 5m, 10m, 12m, 7m_vs_8m, 8m_vs_9m, 10m_vs_11m, 10m_vs_12m, 13m_vs_15m	homogeneous, sym. & asym.
Marine-Easy	Source	3m, 5m, 10m	homogeneous & symmetric
	Unseen	4m, 6m, 7m, 8m, 9m, 11m, 12m	homogeneous & symmetric
Stalker–Zealot	Source	2s3z, 2s4z, 3s5z	heterogeneous & symmetric
	Unseen	1s3z, 1s4z, 1s5z, 2s5z, 3s3z, 3s4z, 4s3z, 4s4z, 4s5z, 4s6z	heterogeneous & symmetric

Data qualities. We follow the offline multi-task setting of Zhang et al. (2023) and use their released datasets unchanged; we collect no data ourselves. For each source task, they collected offline data of four qualities from a QMIX (Rashid et al., 2020) training run, and a data regime trains on one quality across all source tasks of the set. The *expert* data are collected by the fully trained policy; the *medium* data by a partially trained policy whose test win rate is about half the expert's; the *medium-expert* data mix the two in equal parts; and the *medium-replay* data are the replay buffer accumulated while training the medium policy, containing behavior from random to medium quality. Each data

regime draws up to 2,000 logged episodes per source task, taking the most recent episodes where more are available. On Marine-Hard we abbreviate the four regimes as HE, HM, HME, and HMR; return dispersion statistics per task and regime are given in Appendix G.

D IMPLEMENTATION DETAILS

Architecture and observation handling. Our implementation extends the backbone's official codebase and leaves its architecture unchanged. Observations are decomposed into own, ally, and enemy entities; a shared attention encoder embeds each entity into a 64-dimensional representation (Vaswani et al., 2017; Iqbal et al., 2021), from which the planner produces the skill code (dimension 64) and the per-agent decoders produce actions, so the same networks apply across team sizes. Agent identities are encoded up to 15 agents, which is why `25m` and `27m_vs_30m` cannot be evaluated, as listed in Table 3. STEP adds the expectile team-value estimator and its mixing head at training time only ($\approx$47.8k parameters); the deployed forward path is the backbone's. Single-agent offline RL constrains the action set or adds a behavior-cloning term to the policy update to limit out-of-distribution actions (Fujimoto et al., 2019; Fujimoto & Gu, 2021). STEP stays within the logged action set by construction and instead reweights the imitation loss within it.

Controlled protocol. All arms of a data regime share seeds, data, training budget, optimizer, and evaluation grid, so the paired differences we report are internally controlled. Table 3 lists the holdout and non-evaluable maps; every difference from prior work is deliberate and listed in Table 4. Table 5 lists the hyper-parameters: the architecture and optimizer values are the official repository defaults, shared by all arms, and no per-regime tuning is performed.

Evaluation and uncertainty. Transfer is evaluated on unseen maps without target-task data or adaptation. We select each task's best-over-training win rate on the standard 1,000-step reporting grid, as summarized in Table 4. STEP–backbone comparisons pair the five seeds and use task-stratified bootstrap 95% confidence intervals.

Table 3: Additional map groups of the SMAC Marine-Hard task set. The source and unseen maps are listed in Table 2.

Set	Maps	Use
Holdout	4m_vs_5m, 6m_vs_7m, 11m_vs_12m	never inspected during development; one selection-free evaluation (Appendix I)
Not evaluable	25m, 27m_vs_30m	more than 15 agents (limit of the backbone agent-ID encoding)

Table 4: Protocol differences from the backbone paper and its official repository. We inherit the official repository's optimizer so that the backbone reproduction matches the released code; "—" marks settings the source does not specify.

	Backbone paper	Official repository	Ours
Optimizer	Adam, lr 10^{-4}	AdamW, lr 3×10^{-4}	AdamW, lr 3×10^{-4}
Training steps	30,000	251,000	35,000
Evaluation interval	—	1,000 steps	250 steps, reported on the 1,000-step grid
Checkpoint selection	best over training	—	per-task maximum over training

E ADDITIONAL RESULTS

E.1 RESULTS ON OTHER TASK SETS IN SMAC

The main comparison is confined to the Marine-Hard task set. Two further task sets, described in Appendix C, test whether its conclusions hold. Marine-Easy keeps homogeneous marines but uses symmetric source tasks; Stalker–Zealot replaces them with two heterogeneous unit types, and

Table 5: Hyper-parameters of our experiments. The upper block is the backbone configuration, shared by all arms; the lower block exists only in arms that train the STEP components.

Hyper-parameter	Value
Entity embedding dimension	64
Skill code dimension	64
Attention embedding dimension	8
Mixing embedding dimension	32
Discount factor γ	0.99
Batch size (trajectories)	32
Offline episodes per source task	up to 2,000 (most recent)
Training steps	35,000
Optimizer	AdamW, lr 3×10^{-4}, weight decay 10^{-4}
Gradient norm clip	10
Target update interval (hard copy)	80 steps
Expectile τ_e	0.7
Weighting strength α	1.0
Weight clip	100
Weight-std floor ϵ	10^{-6}
Value-target Polyak rate τ_p	0.005

thereby tests whether the signature of STEP (gains where logged behavioral quality is heterogeneous, parity with the backbone on expert data) depends on the homogeneous units of the Marine family. The mechanism account of Section 5.3 fixes what to expect: on medium-replay data, whose returns disperse, STEP should gain; on expert data, whose returns are saturated, it should stay on par with the backbone.

Table 6: Per-map win rates (%) on Marine-Easy, rerun in this work (mean ± sample standard deviation over paired seeds, best-over-training rule). The last row repeats the unseen-map average. Seeds per regime: three (expert), three (medium), three (medium-expert), three (medium-replay).

Tasks	Expert		Medium		Medium-Expert		Medium-Replay	
	Backbone (HiSSD)	STEP (ours)	Backbone (HiSSD)	STEP (ours)	Backbone (HiSSD)	STEP (ours)	Backbone (HiSSD)	STEP (ours)
Source Tasks								
3m	100.0±0.0	100.0±0.0	70.2±3.1	80.2±3.4	99.6±0.7	100.0±0.0	93.3±1.3	91.2±0.6
5m	100.0±0.0	100.0±0.0	85.2±1.3	88.5±1.0	95.6±1.9	100.0±0.0	93.5±1.3	95.6±2.9
10m	100.0±0.0	100.0±0.0	80.6±2.9	90.0±2.9	91.5±2.2	100.0±0.0	92.5±2.3	92.3±1.4
Unseen Tasks								
4m	99.4±0.6	99.8±0.4	77.7±3.7	86.9±6.0	93.5±1.3	94.0±6.1	79.0±2.2	83.5±4.6
6m	100.0±0.0	100.0±0.0	92.1±4.6	91.9±2.5	81.0±12.6	99.4±0.6	100.0±0.0	100.0±0.0
7m	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	87.5±5.0	94.2±5.9	100.0±0.0	100.0±0.0
8m	100.0±0.0	100.0±0.0	97.7±1.3	99.0±0.4	84.6±10.9	98.1±2.7	98.1±0.6	96.9±0.6
9m	100.0±0.0	100.0±0.0	89.6±1.0	91.2±2.3	87.7±5.2	100.0±0.0	92.5±1.1	90.8±1.4
11m	100.0±0.0	100.0±0.0	55.6±2.2	68.1±2.3	77.9±8.9	100.0±0.0	56.2±5.0	66.7±10.0
12m	99.4±0.6	99.4±0.6	44.1±5.2	54.4±7.0	75.2±1.6	99.2±1.0	49.0±1.6	50.6±6.5
Unseen mean	99.82±0.09	99.88±0.14	79.54±2.19	84.49±1.01	83.93±5.37	97.83±1.68	82.11±1.00	84.08±1.38

The pattern established on Marine-Hard in Section 5.2 cannot by itself separate two readings: that the gain follows heterogeneity in logged behavioral quality, or that it follows something particular to the Marine environment. Stalker–Zealot changes the composition of the teams rather than the quality of the logs: source and target tasks are no longer built from a single unit type, so the executor must resolve roles between units that differ in range, speed, and durability. Repeating the controlled contrast there asks whether the reweighting signal still separates realizations of a skill once the units it acts on are heterogeneous. Marine-Easy varies the source tasks instead: its compositions are symmetric throughout, so the source data no longer demand the outnumbered coordination that Marine-Hard requires. Between them the two task sets close the alternative reading from both sides, in the units the executor controls and in what the source tasks demand of it.

Tables 6 and 7 reproduce the fingerprint that Proposition 2 predicts. In each of three task sets that differ from one another in the units a team is made of and in what their source tasks demand, the two arms are closest on expert data, whose logs are collected by a fully trained policy, and further apart

in every mixed-quality regime. What predicts when STEP helps is therefore not a property of the task set but whether logged behavior varies in quality, the quantity Proposition 2 identifies. Proposition 2 ties the improvement to $\mathrm{Cov}_{\mu_\ell}(w, \delta)/\mathbb{E}_{\mu_\ell}[w]$, which is positive when logged realizations of a skill differ in measured quality and contracts as that variation disappears; changing the unit composition while leaving the data qualities in place leaves that quantity, and with it the division between expert and mixed-quality data, intact. The conclusion of Section 5.2 and the mechanism account of Section 5.3 are corroborated outside the task set on which they were formed, and neither requires revision.

Two results limit the claim rather than support it.

Mixed-quality regimes cannot be ranked. Section 5.3 orders the Marine-Hard regimes by return dispersion, yet the three task sets disagree with one another in more than one place: which mixed-quality regime separates the two arms most differs between them, and so does whether the medium regime separates them more than the medium-replay regime. The paper therefore claims the division between expert and mixed-quality data and not an ordering within mixed quality; the observation in Section 5.3 that HME and HMR do not lie on a common mixture scale, so that their relative order carries no further information, continues to hold here. The presence of heterogeneity predicts whether STEP improves execution; its degree does not predict by how much, and this now holds across all three task sets.

One entry is a ceiling effect. On Marine-Easy the medium-expert regime shows by far the widest gap between the two arms' unseen-map means, but there the STEP arm already sits at the ceiling the task allows, with almost no spread across seeds, as Table 6 shows. Once one arm saturates, the distance between the arms records how far the other arm stands from that ceiling and not how high the method can push performance. That entry is therefore not evidence about the size of the gain STEP delivers; read as one, it would overstate STEP.

E.2 The retrieval pathway: configuration and diagnostics

This section specifies the retrieval pathway of the library arm of Section 5.4 and the mechanism by which it remains unused.

Library construction. Two frozen libraries are built once from the 35k-step STEP checkpoint of the same data regime and seed. The micro codebook holds k-means prototypes of per-agent encoder states (100,000 states from 3m and 5m_vs_6m). The macro library holds up to 64 segment embeddings from 200 episodes of 9m_vs_10m: episodes are segmented on the value curve by change-point detection, segments are mean-pooled over agents and time, kept if their mean one-step residual is in the top 30%, and merged into an existing entry above a fixed cosine-similarity threshold.

Retrieval and continued training. At every step each agent retrieves from both libraries by cosine attention; an inner gate initialized toward the codebook mixes the two branches, a FiLM head modulates the result, and a scalar outer gate initialized at $\sigma(-4) = 0.018$ interpolates the executor's skill input between the backbone's code and the retrieved content. Training then continues for 35k further steps from the STEP checkpoint with the encoder frozen. Table 8 lists the settings, which are identical in all 20 runs.

Fixed thresholds. The merge threshold $\tau_{\mathrm{sim}} = 0.9$ is the criterion under which each macro library keeps 1–3 effective entries; the sensitivity of the results to this threshold is not examined here. The same holds for the initial value of the outer gate.

Training and inference cost. On the same GPU, per-step training time of the retrieval arm is $1.23\times$ and $1.26\times$ that of STEP in two consecutive measurements; the arm also requires a second 35k-step training stage. At inference, policy time per decision step is $1.27\times$ STEP's ($1.16\times$ including environment simulation).

The null result of Table 9 has a mechanistic explanation, summarized in Figure 5.

Gradient starvation. The outer gate moves from its initial 0.018 only to 0.023–0.036 over 35k steps (consistent across 20/20 runs), so retrieval contributes 2–3% of the executor input; a small gate passes a small gradient, so the gate never learns to open. Within that share, the inner gate stays at a per-sample median of 0.998 or higher, closing the macro channel almost completely.

Table 7: Test win rates (%) of the best policies, averaged over five random seeds (expert, medium-replay) and three random seeds (medium, medium-expert), on Stalker–Zealot across four offline-data qualities. BC-best reports the better result of BC-t and BC-r. STEP results are bolded when their mean exceeds the HiSSD backbone.

Tasks	Expert					Medium				
	BC-best	UPDeT-m	ODIS	Backbone (HiSSD)	STEP (Ours)	BC-best	UPDeT-m	ODIS	Backbone (HiSSD)	STEP (Ours)
Source Tasks										
2s3z	93.1±4.6	50.0±33.4	97.7±2.6	98.1±0.8	**99.4±0.4**	48.8±9.8	35.0±23.0	49.2±8.4	55.6±2.9	**65.8±7.0**
2s4z	78.1±8.1	23.4±26.6	60.9±6.8	86.4±5.1	**87.5±5.4**	12.5±8.1	18.8±10.3	32.8±12.2	23.1±2.3	**36.2±3.9**
3s5z	92.5±4.2	17.2±19.8	87.5±9.6	95.6±1.5	**96.5±2.3**	24.4±12.4	25.6±24.2	28.9±6.8	43.1±6.0	**58.5±4.4**
Unseen Tasks										
1s3z	45.6±23.8	1.6±1.6	76.6±3.5	99.1±0.6	97.5±2.5	21.9±37.6	3.8±5.0	41.4±18.8	80.2±15.3	79.8±23.3
1s4z	60.0±32.3	26.6±19.3	17.2±10.5	94.9±4.7	**97.6±1.4**	6.2±7.7	2.5±3.6	50.7±7.5	33.3±12.5	**54.4±7.0**
1s5z	45.6±26.9	29.7±26.4	2.5±2.3	63.9±16.0	**77.7±9.3**	3.1±2.6	5.0±4.2	14.1±8.4	21.5±11.0	**32.9±11.2**
2s5z	75.6±11.9	23.4±22.2	27.3±6.0	85.5±12.3	**87.9±4.5**	14.4±9.0	16.9±14.1	32.0±4.6	23.1±1.1	**27.5±8.0**
3s3z	80.6±9.1	20.3±10.9	89.1±5.2	96.5±2.3	93.8±3.0	45.6±14.6	24.4±28.6	23.4±9.2	52.7±13.6	48.7±7.0
3s4z	92.5±5.1	12.5±19.9	96.9±2.2	98.3±1.6	98.1±0.9	40.0±19.0	28.8±31.6	50.8±15.5	53.5±12.6	**57.1±1.6**
4s3z	67.5±19.8	6.2±4.4	64.1±13.0	96.5±2.4	93.4±5.2	28.8±26.4	11.2±18.0	13.3±7.5	44.6±11.2	**45.4±6.3**
4s4z	53.1±18.4	7.8±13.5	79.7±10.9	84.6±9.2	**87.3±7.5**	20.0±12.0	1.2±1.5	12.5±7.0	22.7±8.8	22.1±2.6
4s5z	40.6±19.1	5.5±7.8	86.7±12.6	72.6±17.4	**84.6±8.4**	14.4±8.5	5.6±8.5	7.0±4.1	14.4±7.2	**17.3±6.2**
4s6z	48.1±23.8	4.7±6.4	88.3±8.4	72.6±10.8	71.0±8.2	3.8±3.6	1.9±2.5	1.6±1.6	12.9±5.5	12.9±3.1
Avg. (unseen)	–	–	–	86.45±3.45	**88.89±3.70**	–	–	–	35.90±3.36	**39.81±2.62**

Tasks	Medium-Expert					Medium-Replay				
	BC-best	UPDeT-m	ODIS	Backbone (HiSSD)	STEP (Ours)	BC-best	UPDeT-m	ODIS	Backbone (HiSSD)	STEP (Ours)
Source Tasks										
2s3z	57.5±25.1	57.5±27.1	58.6±15.5	85.8±5.7	**95.6±2.5**	3.1±2.6	14.4±13.2	15.6±18.2	25.0±5.8	**42.8±1.9**
2s4z	37.5±15.3	53.1±24.6	41.4±7.8	60.8±13.3	**74.4±2.7**	5.2±7.4	12.5±9.7	7.8±5.2	19.4±1.9	**41.6±9.8**
3s5z	63.1±13.3	35.0±23.5	41.4±18.5	77.3±10.3	**79.8±3.2**	31.3±6.3	20.0±16.6	18.8±3.1	44.1±10.2	**51.5±3.0**
Unseen Tasks										
1s3z	55.6±37.7	4.4±8.8	72.7±12.2	95.8±6.7	**98.1±2.2**	24.0±15.4	0.0±0.0	21.1±20.4	58.6±6.8	**63.8±8.3**
1s4z	25.0±30.7	11.9±9.8	44.5±20.3	81.9±11.1	**96.2±2.2**	2.1±2.9	7.5±10.0	6.2±7.7	52.9±20.8	**81.7±10.6**
1s5z	14.4±19.4	3.8±4.6	42.2±31.4	45.6±16.3	**46.2±4.7**	7.3±6.4	11.9±9.6	7.8±6.4	27.1±7.4	**35.8±5.6**
2s5z	26.9±20.2	37.5±22.5	43.0±10.7	48.5±5.7	**64.2±9.6**	12.5±15.5	20.0±16.8	14.1±8.1	33.1±8.6	**44.5±13.5**
3s3z	35.6±18.0	33.8±15.0	50.0±13.3	88.5±0.7	87.5±7.7	35.4±12.1	17.5±12.3	25.0±20.1	41.4±9.5	**50.6±11.7**
3s4z	74.4±16.3	43.1±20.7	52.3±9.5	90.8±4.9	**95.4±4.2**	20.8±9.0	15.6±11.2	19.5±16.6	52.2±7.3	**64.1±5.6**
4s3z	69.8±7.8	23.8±21.0	17.2±7.2	85.4±1.0	**90.8±3.4**	17.7±5.3	11.2±15.0	8.6±14.9	35.9±20.1	**54.9±17.2**
4s4z	41.9±14.9	10.6±13.8	20.3±6.8	63.5±6.4	**68.1±9.0**	15.6±6.8	5.6±9.8	4.7±8.1	21.8±8.9	**27.6±12.2**
4s5z	17.3±5.3	11.9±16.1	21.9±2.2	37.7±4.7	37.3±7.3	1.0±1.5	10.6±19.7	0.8±1.4	12.0±8.5	**12.8±5.1**
4s6z	13.8±3.2	5.0±8.5	18.0±5.1	32.1±16.7	**34.2±8.3**	0.0±0.0	6.9±13.8	2.3±4.1	9.4±4.1	**16.5±8.6**
Avg. (unseen)	–	–	–	67.00±0.84	**71.81±1.34**	–	–	–	34.44±2.24	**45.22±6.01**

Collapsed content. The mean-pooled segment embeddings have effective rank 2.8–7.7 of a possible 64, so the merge criterion folds 95–99% of the surviving segments into existing slots and every macro library keeps only 1–3 effective entries of its 64 slots; the micro prototypes are hit more broadly (30–63 of 64 entries), but each cluster is occupied almost entirely by a single task (median dominant-task share about 0.995), so what the prototypes separate is tasks, not tactics.

The two failures are independent: where the gate is open the library is still collapsed, and where the library is populated the gate is still closed.

These diagnostics support the conclusion of Section 5.4: the retrieval branch is structurally connected but functionally unused, and the improvement of the full system over the backbone is the improvement of STEP.

F FINAL-CHECKPOINT RESULTS

The paper reports best-over-training values, following the best-policy convention of prior work. A natural concern about any maximum over a training run is that it could manufacture gains out of evaluation noise. This appendix removes the selection entirely: every main-table cell is re-evaluated at the final checkpoint (the last value of the five-evaluation moving average with $t_{\text{env}} \leq 35$k). The conclusions survive unchanged. The paired differences are -2.64 (HE), $+2.15$ (HM), $+4.40$ (HME), and $+6.31$ (HMR): every sign agrees with the main table, and HME and HMR retain CIs

Table 8: Library-arm settings (identical in all 20 runs).

Setting	Value
Embedding width d	64
Codebook size K; samples; source tasks	64; 100,000; 3m, 5m_vs_6m (50,000 each)
k-means	mini-batch 4096, 10 initializations, $\leq$300 iterations
Macro capacity $J_{\max}$; source task; episodes	64; 9m_vs_10m; 200
Segmentation	PELT (RBF), $\beta_{\text{seg}} = 10$, $L_{\min} = 5$, $L_{\max} = 80$
Segment filter	top 30% by mean one-step residual (0.7 quantile)
Merge threshold; merge rate	$\tau_{\text{sim}} = 0.9$; $\eta = 0.2$
Retrieval temperature T	0.1
Inner gate at initialization	$\sigma(4) = 0.982$ (codebook weight)
Outer gate at initialization	$\sigma(-4) = 0.018$
Further training steps	35,000 from the same-seed STEP checkpoint
Task probabilities (3m, 5m_vs_6m, 9m_vs_10m)	0.2, 0.2, 0.6
Target hard copy; Polyak rate	every 80 steps; $\tau = 0.005$

Table 9: Retrieval pathway added to STEP: paired differences in average unseen-map win rate (%, five paired seeds) and library diagnostics.

	HE	**HM**	**HME**	**HMR**
Library $-$ STEP (paired Δ)	-1.22	$+0.01$	-0.37	$+0.06$
95% CI	$[-3.22, 0.61]$	$[-0.99, 1.15]$	$[-1.19, 0.44]$	$[-1.22, 1.17]$
Macro entries of 64 (range, 5 seeds)	1–3	1–3	1–3	1–3

that exclude zero. Absolute win rates drop by 5.6–8.2 points without the selection, while the paired differences move by only 0.2–2.1 points: the paired design absorbs the choice of reporting rule, so the gains of STEP are not an artifact of it. On HE the difference becomes a measurable cost (-2.64, 95% CI $[-5.10, -0.40]$), consistent with the mechanism account of Section 5.3: on return-saturated data the weights carry no signal, so reweighting can only add variance. The shuffled-weight control also keeps its verdict here: C exceeds C-shuf by $+6.29$ $[4.40, 8.00]$, while C-shuf is indistinguishable from B (-0.15 $[-2.01, 2.39]$).

G Mechanism-diagnostic details

This appendix details the measurements behind Section 5.3. The episode-level rankings describe associations in the logged data; they do not isolate a fixed-skill conditional distribution or establish which individual actions caused a return. The transfer effects are measured separately on unseen maps.

Return variation and ranking. Table 10 reports logged return dispersion by source task and data regime. Two HE tasks are nearly return-saturated, whereas the mixed-quality regimes leave varied outcomes. Among tasks with return variation, the mean return in the highest residual decile exceeds that in the lowest by 2.14–3.38 return standard deviations. This supports the use of residuals to rank recorded realizations, while not proving action-level causality.

Weight provenance and map effects. Figure 3a uses the HME diagnostic recomputation (200 episodes per source task and seed, equally split between expert and medium origin); deciles are formed within each seed from episode residuals. The expert-origin share is 0.46 in the highest decile versus 0.50 overall. Figure 3b uses archived per-map results: eight of nine HMR maps and seven of nine HME maps improve. The residual and weight rankings coincide because the weight transform is monotone; clipping is rare. Per-step correlations with return-to-go are weak and mixed in sign, so the diagnostic supports episode-level selection rather than identification of advantageous individual actions. Weight curves appear in Figure 7.

Hard filtering (%BC). Table 11 gives the full hard-filtering control summarized in Section 5.4.

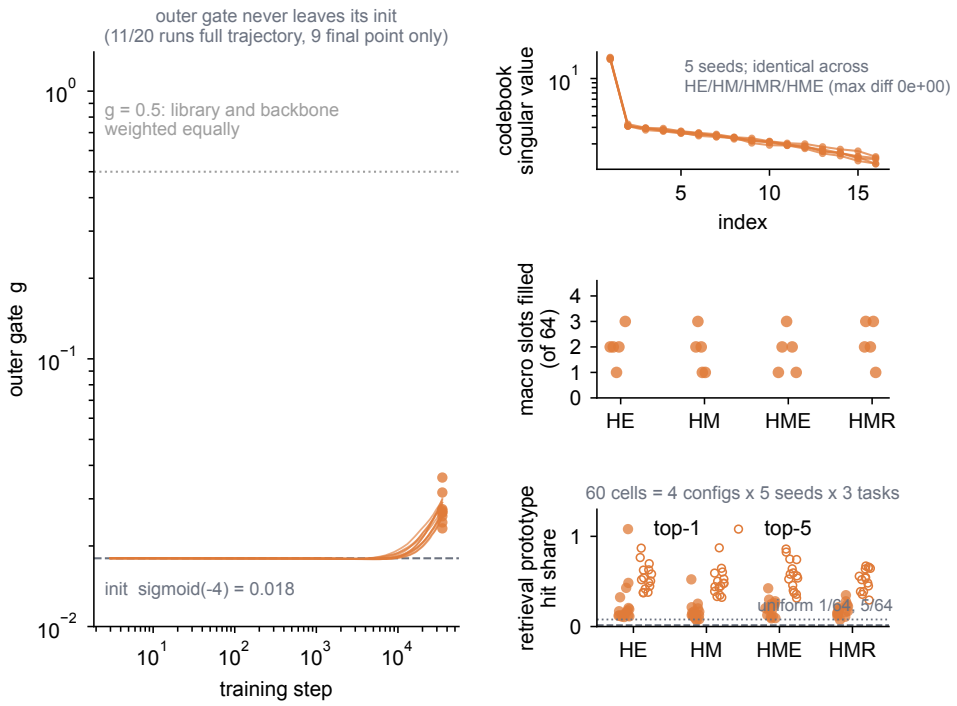


Figure 5: Diagnostics of the retrieval pathway. Left: outer gate g over training for the 20 library-arm runs (log scales on both axes), 11 with the full trajectory and 9 with the final value only; the dashed line is the initial value $\mathrm{sigmoid}(-4) = 0.018$ and the dotted line $g = 0.5$, at which library and backbone would be weighted equally. Right, top to bottom: codebook singular values (five seeds; identical across the four regimes); number of filled macro-library slots out of 64 per run; top-1 and top-5 hit shares of the retrieved prototypes in each of the 60 (regime, seed, task) cells, with the uniform shares $1/64$ and $5/64$ marked.

Table 10: Dispersion of dataset episode returns R on the three source tasks ($n = 200$ episodes per cell; the same fixed sample as the weight diagnostics). Mean, σ_R and $\mathrm{CV}_R = \sigma_R/|\bar{R}|$ are computed on the raw returns; the number of distinct values, the mode and the mode share are computed after rounding R to two decimal places.

Regime	Task	n	Distinct	Mode	Mode share	Mean	σ_R	CV_R
HE (expert)	3m	200	2	20.00	0.99	19.88	1.16	0.058
	5m_vs_6m	200	24	20.00	0.71	17.18	4.48	0.261
	9m_vs_10m	200	13	20.00	0.94	19.54	1.73	0.089
HM (medium)	3m	200	19	20.00	0.53	13.85	6.49	0.468
	5m_vs_6m	200	31	20.00	0.21	11.97	4.28	0.358
	9m_vs_10m	200	47	20.00	0.42	15.49	3.95	0.255
HME (medium-expert)	3m	200	17	20.00	0.77	16.99	5.46	0.321
	5m_vs_6m	200	25	20.00	0.49	14.84	5.14	0.346
	9m_vs_10m	200	34	20.00	0.67	17.51	3.61	0.206
HMR (medium-replay)	3m	200	34	20.00	0.18	7.65	6.07	0.793
	5m_vs_6m	200	55	7.81	0.07	7.90	2.32	0.293
	9m_vs_10m	200	98	10.56	0.04	9.05	2.64	0.291

Table 11: Hard return filtering (%BC) versus residual weighting (C) on HMR and HME (average unseen win rate, %, five paired seeds (1–5) in both regimes). %BC is the B configuration (value estimator on, weighting off) trained only on the episodes whose undiscounted return is at or above the median of their source task (ties kept; 1000–1059 of the 2000 episodes of each task retained), paired with the C arm of the same seed. Paired bootstrap 95% CIs (10,000 resamples). For reference, the same backbone trained on expert data (HE) reaches 70.26 on the same unseen maps.

	HMR	HME
%BC, mean ± std	54.56 ± 1.84	63.47 ± 4.33
C − %BC	$+3.60$ $[1.06, 6.43]$	$+2.97$ $[0.33, 5.61]$
%BC − B	$+0.54$ $[-1.17, 1.92]$	$+2.76$ $[-0.35, 5.99]$

Table 12: Accumulation chain on the four Marine-Hard regimes (average unseen win rate, %, five paired seeds).

Arm	HE	HM	HME	HMR
A backbone (all off)	70.26 ± 2.03	51.87 ± 3.52	62.78 ± 1.51	52.43 ± 1.17
B + value estimator	68.64 ± 0.88	52.15 ± 2.82	60.71 ± 2.80	54.01 ± 1.03
C + residual weighting	67.25 ± 3.81	53.21 ± 3.70	66.44 ± 2.50	58.15 ± 1.78
C-shuf batch-permuted weights	—	—	60.51 ± 4.23	53.97 ± 2.14
D (STEP) + consequence conditioning	69.71 ± 2.77	53.83 ± 1.53	68.19 ± 2.69	57.22 ± 1.69
δ_{value} (B−A)	-1.62 ± 1.61	$+0.28 \pm 3.23$	-2.07 ± 4.02	$+1.58 \pm 1.47$
$\delta_{\text{weighting}}$ (C−B)	-1.39 ± 4.36	$+1.06 \pm 5.08$	$+5.73 \pm 2.30$	$+4.14 \pm 2.38$
δ_{forward} (D−C)	$+2.46 \pm 4.34$	$+0.63 \pm 3.40$	$+1.75 \pm 4.91$	-0.93 ± 2.56
C − C-shuf	—	—	$+5.93 \pm 5.34$	$+4.18 \pm 1.07$
C-shuf − B	—	—	-0.19 ± 5.61	-0.04 ± 2.92

H Full learning curves and weight diagnostics

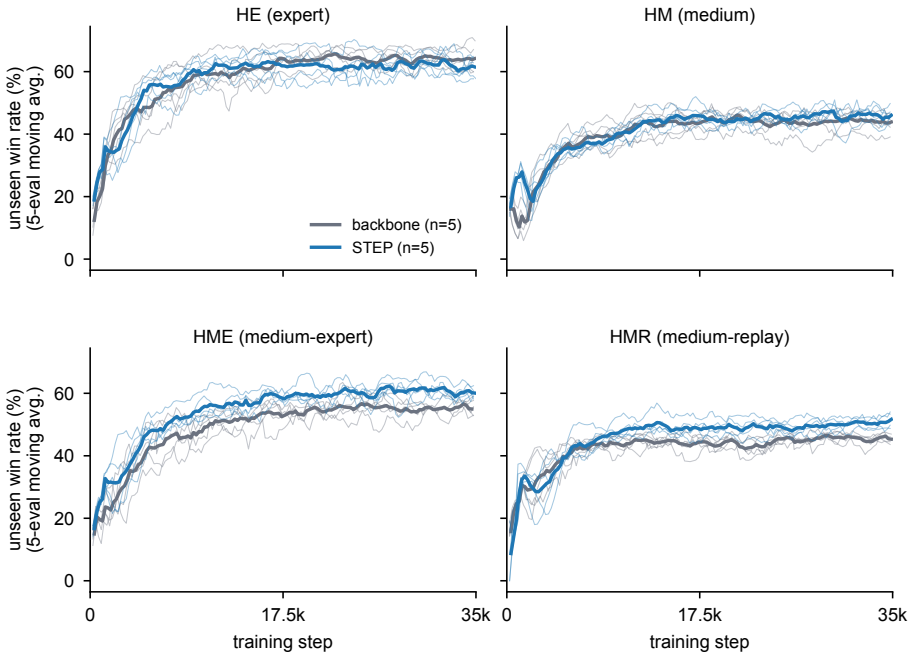
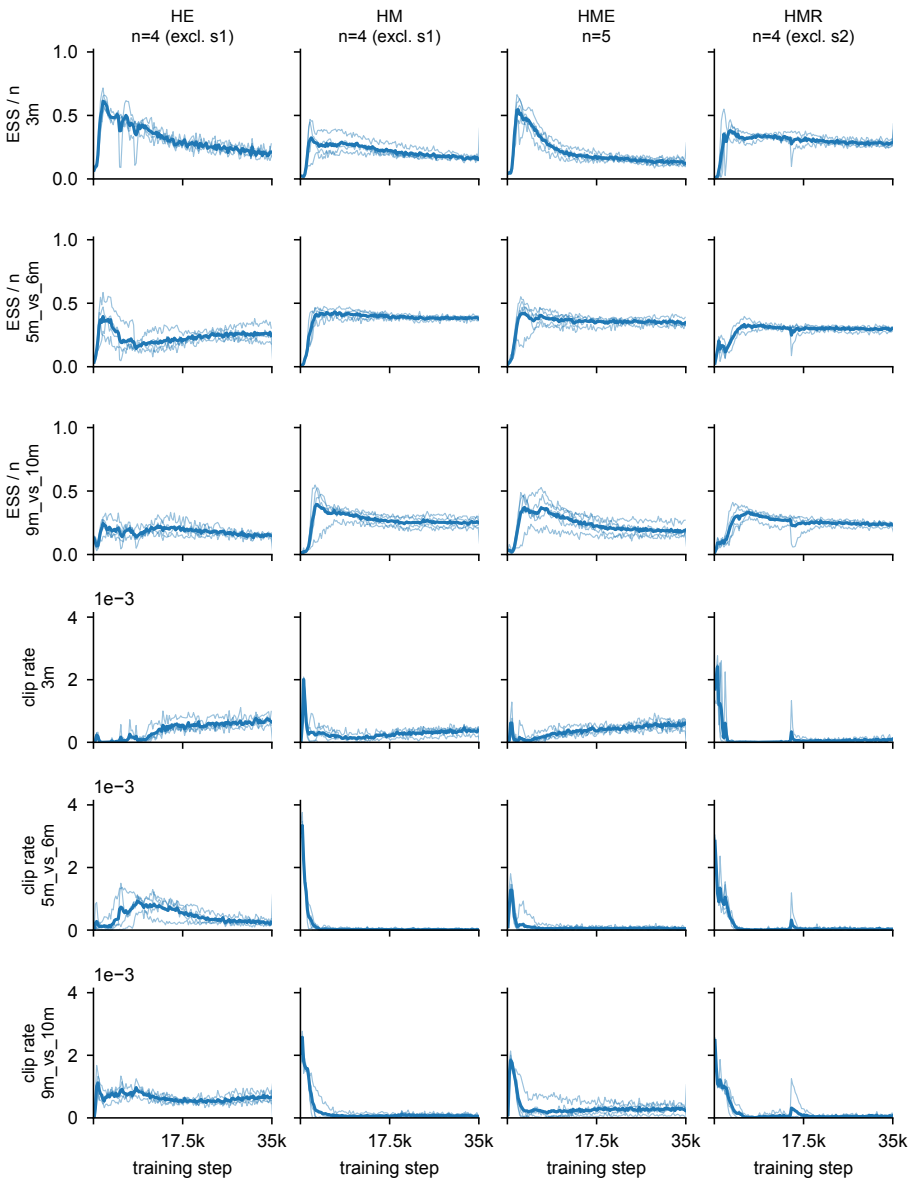


Figure 6: Learning curves by data regime, one panel per regime. The vertical axis is the mean win rate over the nine unseen maps recorded by the training framework, a moving average over five evaluations; thin lines are individual seeds and thick lines the mean over five seeds, for the backbone (grey) and STEP (blue). The curves use the same evaluation records as Table 1.



excl. = seed whose training-time weight log is truncated (weight-diagnostic fields missing); not filled from other sources.

Figure 7: Weight diagnostics of STEP over training, by data regime (columns) and source task (rows): effective sample size relative to batch size (ESS/n, top three rows) and the fraction of clipped weights (bottom three rows). Both are computed from the batch weight statistics recorded during training, aggregated in 250-step bins; thin lines are runs and thick lines their mean. HE, HM, and HMR show four runs each: for HE seed 1, HM seed 1, and HMR seed 2 the main-table results are complete, but the training-time weight log is truncated, so these runs are not in this figure; no other source is used to fill them.

I HOLDOUT-MAP EVALUATION

The nine unseen maps of the main comparison were viewed repeatedly during development, so while no target-task data ever reaches training, they cannot rule out selection pressure from our own development loop. To check that the reported behavior is not an artifact of that exposure, three maps were set aside and never inspected during development, and each of the 40 final checkpoints (four data regimes $\times$ {backbone, STEP} $\times$ five seeds) is evaluated on them exactly once, 32 episodes per map, with no checkpoint selection. The purpose is a collapse check rather than a gain claim, and no collapse appears: Table 13 shows that the per-map paired differences scatter around zero or favor STEP, and the two of ten CIs that exclude zero are both positive. On `4m_vs_5m` both arms win no episode in any seed under HE and HMR, so these two cells carry no information; `25m` and `27m_vs_30m` cannot be evaluated because the backbone's agent-ID encoding supports at most 15 agents. Seed-to-seed variance is large at this sample size, so no single cell is interpreted and no cross-map average is taken.

Table 13: Paired difference per holdout map (STEP $-$ backbone, same seed, win-rate points) with paired bootstrap 95% CI; 35k-step checkpoints, 32 episodes per map, no selection, no cross-map average.

Regime	4m_vs_5m	6m_vs_7m	11m_vs_12m
HE	0.0 (both 0%)	$+8.1$ $[-2.5, +18.8]$	$+3.7$ $[-30.6, +25.6]$
HM	-1.2 $[-2.5, +0.0]$	$+15.0$ $[+1.9, +27.5]$	$+2.5$ $[-9.4, +14.4]$
HME	$+1.3$ $[+0.0, +3.8]$	-2.5 $[-26.2, +21.2]$	-1.9 $[-36.9, +33.1]$
HMR	0.0 (both 0%)	$+8.8$ $[-6.9, +24.4]$	$+14.4$ $[+8.1, +19.4]$