CAPACITY SCALING FRAGILITY IN MULTI-AGENT POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Increasing network capacity is generally expected to improve the performance of reinforcement-learning policies, or at least not to harm it. However, we find that this expectation can fail in cooperative multi-agent reinforcement learning (MARL), because network width interacts with how each policy network is updated. On SMACv2 protoss_5_vs_5, under standard tuned hyperparameters, increasing HAPPO's hidden size from 64 to 256 reduces the win rate from 0.474 to 0.368 over five seeds, whereas MAPPO does not show the same degradation. A four-way identification experiment crossing sequential versus synchronous updates with independent versus shared parameters shows that the capacity penalty appears in every standard configuration except shared, synchronous MAPPO. Further interventions transfer this protection to independent actors, showing that parameter sharing itself cannot explain the difference. Across these configurations and interventions, the key quantity that consistently separates protected and fragile regimes is the realized policy-update scale. At the same nominal learning rate, increasing network width amplifies the net per-iteration policy KL by 2.2–$3.6\times$, while fragile topologies exhibit 2.4–$12\times$ larger absolute policy displacement than protected ones. When the larger network's realized KL is brought back to the small network's default level, its performance is restored. Reducing the learning rate shrinks the capacity gap in every fragile configuration we test, while directly bounding per-update KL provides a partial recovery. Together, these results suggest that network capacity, update topology, and learning rate jointly determine the realized optimization regime: the same nominal learning rate can produce substantially different policy updates across network widths and training configurations. At hidden size 256, the default-rate comparison gives MAPPO a 0.094 performance advantage over HAPPO; applying the same lower learning rate to both algorithms reduces this margin roughly fourfold to 0.022. Thus, fixed-architecture, fixed-learning-rate MARL comparisons can confound algorithmic differences with update-scale calibration.

1 INTRODUCTION

A common assumption in deep reinforcement learning is that increasing network capacity (wider hidden layers, deeper architectures, or both) either improves or does not harm policy performance. However, recent single-agent work shows that realizing the benefits of scale requires care (Ota et al., 2021; Schwarzer et al., 2023; Nauman et al., 2024). The same assumption also shapes cooperative multi-agent reinforcement learning (MARL), where benchmark studies typically fix one architecture and one learning rate and then compare algorithms (Yu et al., 2022; Zhong et al., 2024; Papoudakis et al., 2021). This protocol implicitly treats a nominal learning rate as comparable across architectures and algorithms.

We find that this assumption does not always hold. On SMACv2 protoss_5_vs_5 (Ellis et al., 2023), enlarging HAPPO (Kuba et al., 2022) from hidden size 64 to 256 under tuned defaults reduces the win rate from 0.474 to 0.368 over five seeds, a 22% relative loss. Under the same scaling, MAPPO (Yu et al., 2022) is unharmed ($0.445 \rightarrow 0.462$; Figure 1). The central puzzle is therefore not simply why a larger network can perform worse, but why the same increase in capacity is fragile under one multi-agent update regime and apparently safe under another.

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.

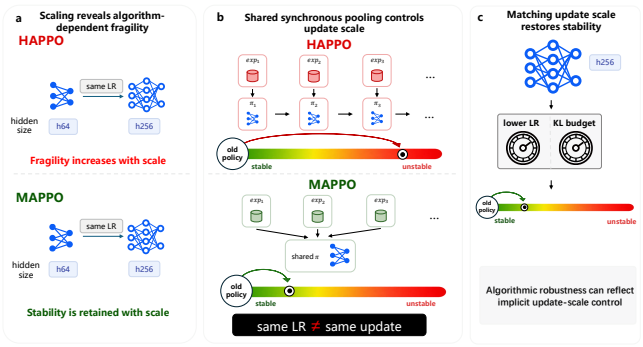


Figure 1: Mechanistic overview. (a) Increasing hidden size under the same tuned learning rate degrades HAPPO but not MAPPO. (b) Update topology changes the per-network update regime: independent networks update from one agent's data, whereas shared synchronous MAPPO pools all agents' experience into one update. Width then amplifies the realized policy step, so *the same learning rate does not imply the same realized update*. (c) Lowering the learning rate to match the realized update scale reduces the capacity gap to a near-zero point estimate; a direct KL budget partially closes the gap.

We resolve this puzzle in two stages. First, we disentangle the apparent algorithmic differences between HAPPO and MAPPO. Crossing sequential versus synchronous updates with independent versus shared parameters, and separately removing HAPPO's cumulative importance ratio, shows that the penalty survives sequential-shared ($+0.125$), synchronous-independent ($+0.134$), and no-ratio ($+0.114$) configurations. It disappears only with shared parameters and synchronous updates, i.e., default MAPPO (-0.017). This pattern suggests that neither sequential updating nor parameter sharing alone explains the penalty, and motivates us to examine the per-network update regime induced by synchronous pooling. To test this interpretation, we transfer the pooled regime to independent actors and find that the capacity penalty falls from $+0.134$ to $+0.010$.

We next consider two possible explanations for the capacity penalty. One is that the larger network simply learns a worse representation. The probes point in the opposite direction. Dimensionality-controlled probes show that, across all algorithm variants, h256 latent representations retain more information about teammates' randomized unit types than h64, even when h256 performs worse. The capacity penalty is therefore difficult to explain by a lack of representational content alone. A second explanation is *realized policy displacement*: how far the policy moves after optimization, measured by policy KL. Shared synchronous pooling gives one network the pooled experience of all agents, whereas the other standard topology cells update each policy network from a smaller per-network experience stream. Width then interacts with this baseline regime: at the same nominal learning rate, h256 makes substantially larger policy-space moves than h64. Across the identification matrix, protected cells begin from much smaller realized displacement, whereas fragile cells occupy a larger-displacement regime. We find that matching the realized update scale across widths restores h256 performance, while interventions that reduce the update scale attenuate the fragility. The resulting account is summarized in Figure 1: *update topology sets the baseline per-network update regime, width amplifies the realized policy step, and their interaction governs scaling fragility in our experiments.*

Contributions. (1) We identify a capacity-scaling failure in multi-agent policy optimization that is selective across update topologies. (2) Through a four-way identification experiment and pre-registered controls, we show that the protected regime can be transferred to independent actors and trace the effect to the per-network update process rather than parameter sharing itself. (3) We identify realized policy displacement as the proximal quantity separating protected and fragile regimes, measure it directly using exact policy KL, and validate this account through learning-rate matching.

Table 1: Win rates on SMACv2 protoss_5_vs_5. HAPPO degrades with capacity while MAPPO remains stable. Values are last-10 evaluation means ± standard deviation over completed seeds (5 per HAPPO cell, 4 per MAPPO cell; per-contrast counts in Appendix G).

Algorithm	**h64**	**h128**	**h256**
HAPPO	$\mathbf{0.474 \pm 0.043}$	0.430 ± 0.017	0.368 ± 0.016
MAPPO	0.445 ± 0.045	$\mathbf{0.512 \pm 0.039}$	0.462 ± 0.028
Gap (h64 − h256)	HAPPO: $+0.105$	MAPPO: -0.017	

2 Background

We consider a cooperative task modeled as a decentralized partially observable Markov decision process (Dec-POMDP) (Oliehoek & Amato, 2016): N agents share a team reward, and each agent i observes o_i and acts according to its policy $\pi_{\theta_i}(a_i|o_i)$, trained under Centralized Training with Decentralized Execution (Lowe et al., 2017; Foerster et al., 2018).

MAPPO and HAPPO. MAPPO (Yu et al., 2022) applies Proximal Policy Optimization (Schulman et al., 2017) synchronously to all agents with a centralized value function and generalized advantage estimation (Schulman et al., 2016); in its default configuration, all agents share one actor network, which is updated once per epoch using the pooled experience of all agents. HAPPO (Kuba et al., 2022) instead trains N independent actors with a sequential update scheme and monotonic improvement guarantees: agents are updated in order $1, \ldots, N$, and the clipped surrogate for agent i incorporates a cumulative importance ratio that accounts for the policy changes of all previously updated agents,

$$m_i = \prod_{j=1}^{i-1} \frac{\pi_{\theta_j^{\text{new}}}(a_j|o_j)}{\pi_{\theta_j^{\text{old}}}(a_j|o_j)}. \tag{1}$$

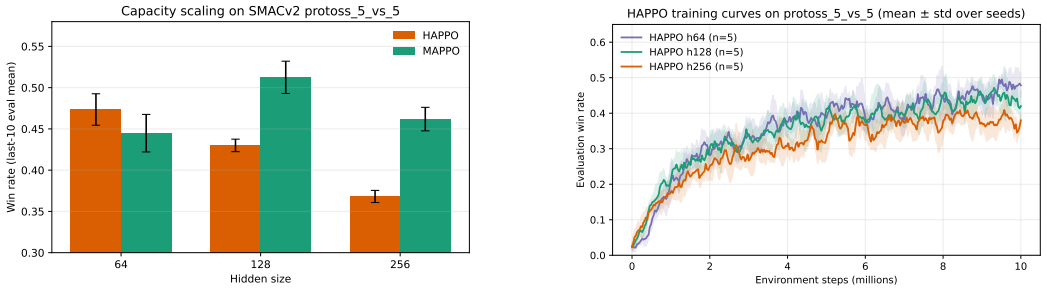
Two design axes therefore separate HAPPO from MAPPO: the update scheme (sequential with the cascading ratio m_i vs. synchronous) and parameter sharing (independent actors vs. one shared actor). Section 4 tests how each axis contributes to the capacity penalty.

We use SMACv2 (Ellis et al., 2023) as our primary benchmark because, unlike SMACv1, it introduces two sources of per-episode randomness: unit-type randomization (each agent's type, e.g. Zealot, Stalker, or Sentry, is resampled every episode) and starting-position randomization. These variations create heterogeneous coordination settings and require policies to generalize across a combinatorial space of agent roles.

3 The Phenomenon: Selective Scaling Fragility

3.1 Experimental Setup

We use the HARL framework (Zhong et al., 2024) with its officially tuned optimizer and algorithm hyperparameters. Unless otherwise noted, we train for 10^7 environment steps on protoss_5_vs_5. Actor and critic networks follow the HARL architecture (a single-hidden-layer MLP encoder with LayerNorm followed by one GRU layer), and the width d of both the MLP hidden layer and the GRU is the primary experimental variable: h64 ($d = 64$, the tuned default), h128, and h256. The learning rate is 5×10^{-4} (the tuned default) unless stated otherwise; rollout, evaluation, and all other settings are held identical across every configuration (Appendix A). The unit of statistical analysis is the random seed: per-seed scores average the last 10 evaluation win rates. All qualitative conclusions are unchanged under last-50 and full-run AUC aggregation (Appendix F); Appendix G reports Welch t-tests with 95% confidence intervals for every headline contrast.



(a) Capacity scaling by algorithm.

(b) HAPPO training curves (mean ± std, 5 seeds).

Figure 2: Algorithm-dependent scaling fragility on SMACv2 protoss_5_vs_5. (a) Larger networks hurt HAPPO but not synchronous MAPPO; error bars show standard error over seeds. (b) The h256 gap develops during training and persists to convergence.

3.2 A Selective Capacity Penalty

Table 1 establishes the phenomenon that motivates the rest of the paper. For HAPPO, increasing hidden size produces a monotone degradation from h64 (0.474) to h128 (0.430) to h256 (0.368). The h64–h256 gap is $+0.105$ (22% relative; 95% CI $[+0.053, +0.157]$, Welch $p = 0.003$) and is consistent across five completed seeds.

The same scaling does not produce the same effect in MAPPO. Its h256 policy (0.462) remains slightly above h64 (0.445), and h128 performs best (0.512). This contrast is the key identification opportunity: if network size, environment difficulty, task–capacity mismatch, or generic overfitting were sufficient explanations, both algorithms should be affected in the same direction. Instead, the contrast points to the optimization process as a possible source of the difference. Section 4 therefore asks which difference between HAPPO and MAPPO creates the protected regime.

The training curves further reveal the nature of the failure. The h256 HAPPO policy does not diverge or fail immediately. It follows the smaller networks for roughly the first 3M environment steps and then plateaus at a lower performance level (Figure 2b). This pattern suggests a persistent optimization limitation rather than an initialization failure or a late-stage collapse.

The penalty is mainly associated with actor width. An actor–critic decomposition provides an initial localization. Enlarging only the critic leaves performance near the h64 baseline (0.492 over 3 seeds), whereas enlarging only the actor approaches the h256 result (0.382). This suggests that the relevant capacity is the width of the network whose gradient updates directly change the policy. This actor–critic contrast therefore motivates the optimization hypothesis that we test directly in Section 4 (Appendix G).

3.3 Cross-Map Consistency

The selective penalty is not confined to the main Protoss map. On terran_5_vs_5, HAPPO performance drops by 0.117 (0.482 $\rightarrow$ 0.365; 3 seeds per cell), while MAPPO again improves (0.356 $\rightarrow$ 0.433; 3 and 2 seeds). On zerg_5_vs_5, HAPPO performance drops by 0.065 (0.377 $\rightarrow$ 0.312; 2 seeds). The same trend therefore appears across all three races.

4 Identifying the Mechanism

4.1 Disentangling Update Topology

The first question is whether the penalty is specific to HAPPO's sequential update. HAPPO and MAPPO differ along two coupled design axes: sequential versus synchronous updating and independent versus shared actor parameters. We therefore cross these axes in a four-way identification experiment. We also add a control that keeps HAPPO's training pipeline intact but fixes the cumulative importance ratio m_i (Eq. 1) to 1 for all agents ("no-prefix"), removing the cascading correction.

Table 2: Four-way identification on protoss_5_vs_5 (last-10 evaluation means over seeds; the standard topology rows share one training pipeline and hyperparameters, and the final row is the dose-matching intervention of Section 4.2). The capacity penalty survives every design change except one: shared parameters combined with synchronous, pooled updates. The last row is the dose-matching control of Section 4.2: the same fragile synchronous-independent configuration, but with each actor given pooled MAPPO's per-update sample count (40 rollout threads instead of 8). Final column: the measured mediator, net per-iteration exact policy KL at h256 from pre-registered instrumented replicates of each cell (Section 4.2; 2 seeds per cell), whose own win rates reproduce the corresponding protected-versus-fragile pattern.

Update	**Params**	**h64**	**h256**	**Gap**	**Net KL**
sequential	indep. (HAPPO)	0.474	0.368	+0.105	0.0023
sequential	shared	0.477	0.352	+0.125	0.0058
synchronous	indep.	0.479	0.345	+0.134	0.0022
no-prefix	indep.	0.483	0.369	+0.114	—
synchronous	shared (MAPPO)	0.445	0.462	−0.017	0.0005
synchronous	indep., dose-matched	0.553	0.543	+0.010	0.0009

This design asks which design feature, or combination of features, separates the protected and fragile regimes.

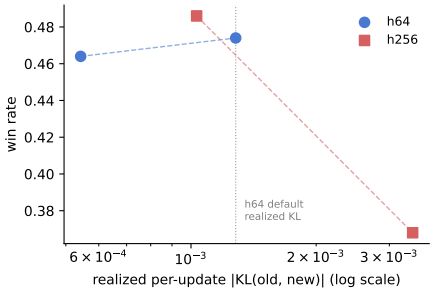
Removing the cumulative ratio leaves a similar gap ($+0.114$). Making the updates synchronous while keeping independent actors also leaves the gap ($+0.134$). Sharing parameters while retaining sequential updates does the same ($+0.125$). Among the standard configurations, the protected regime appears only when shared parameters are combined with a synchronous update that pools all agents' experience into one gradient step (-0.017). Protection therefore appears at the combination rather than either design axis alone. This pattern makes the *pooled per-network update regime* the leading candidate for the protection. The dose-matched row previews a causal test of this interpretation: the same protection can be transferred to fully independent actors without introducing parameter sharing.

4.2 From Update Regime to Realized Policy Displacement

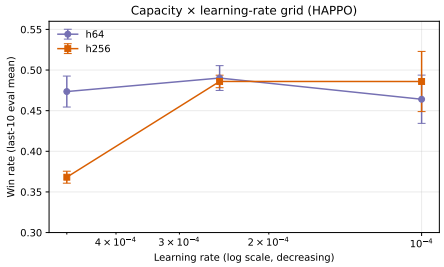
We next separate two levels of the mechanism. The *update regime* describes how much experience enters each network update and how updates are scheduled. The *realized update scale* describes the proximal consequence: how far the policy actually moves in policy space. In shared synchronous MAPPO, one network pools all N agents' experience into an averaged update, so each optimizer phase incorporates $B_{\text{pool}} = 6{,}400$ transitions (N=5, E=8, length L=160). In the other standard topology cells, each independent actor update uses one agent's data ($B = 1{,}280$, $N\times$ less), while the shared-sequential variant applies N separate updates per epoch. These differences establish distinct baseline update regimes before network width is changed.

Width then acts on top of that baseline. In the HARL parameterization we study, a wider network realizes a larger change in policy outputs per gradient step at a fixed learning rate, the finite-width analogue of the width–learning-rate coupling formalized by maximal-update parameterization (Yang & Hu, 2021; Yang et al., 2021) (Appendix D). The resulting hypothesis is therefore interactional: *topology sets the baseline update regime, width amplifies the realized policy step, and fragility appears when their combination produces excessively large policy displacement.*

Measuring the proximal quantity. We first measure how far the policy actually moves. Instrumented HAPPO runs mirror the stock pipeline and record every agent update's realized policy change using the batch-mean log-ratio estimator (Appendix D; 2 seeds per cell, ∼78k update records each). At the tuned default learning rate, h256 realizes a mean per-update $|\mathrm{KL}|$ of 0.0034, compared with h64's 0.0013: the same nominal learning rate produces a $2.7\times$ larger policy-space step in the wider network. Pre-registered exact categorical-KL runs show the same effect without estimator ambiguity: $3.5\times$ seed-level amplification in the 2M-step validation window and $3.6\times$ over the full 10^7-step budget (Appendix D). This measurement also gives a direct matching prediction. At learning rate 1×10^{-4}, h256's realized KL falls to 0.0010, at or slightly below h64's default level, and



(a) Performance across realized update scales.



(b) Capacity $\times$ learning-rate grid.

Figure 3: Realized update scale and capacity-dependent performance. (a) Win rate versus realized per-update policy KL (instrumented runs, 2 seeds per cell; win rates from the independent performance grid); for each width, the left point is learning rate 10^{-4} and the right point 5×10^{-4}. When h256's realized KL is brought down to h64's default level (dotted line), no capacity penalty is detected. (b) h64 is nearly flat across learning rates while h256 swings from worst to best, showing that h256 is substantially more sensitive to update scale than h64.

its win rate (0.486) meets and slightly exceeds h64's default performance (0.474). Figure 3a shows this relation: h64 changes little across the measured update scales, whereas h256 moves from best to worst as its realized update grows.

Transferring the protected update regime. The preceding measurements identify policy displacement as the proximal quantity, but they do not by themselves explain why the pooled cell starts from a smaller update regime. PPO lacks batch-size invariance: the amount of data entering an update co-determines how far the proximal update moves the policy (Hilton et al., 2022). We therefore test whether the protection can be transferred without parameter sharing. In a pre-registered *dose-matching* intervention, the fragile synchronous-independent configuration uses $5\times$ more rollout threads (40 vs. 8), giving each independent actor the same 40 trajectories per update that pooled MAPPO receives, while keeping the same 10^7-step budget. The capacity penalty shrinks from $+0.134$ to $+0.010$ (h64 0.553, h256 0.543), with h256 recovering from 0.345 to 0.543 while h64 moves far less. Thus, parameter sharing is neither sufficient for protection (the shared-sequential cell remains fragile) nor necessary for it; the protection can be transferred through the per-network update regime that pooling provides implicitly.

Isolating what dose matching changes. Increasing rollout threads by $5\times$ changes two quantities at once: each update sees $5\times$ more data, and the fixed environment-step budget contains $5\times$ fewer optimizer phases. Two pre-registered controls separate these effects. A *phase-count control* matches the dose-matched run's $\sim$1,560 optimizer phases while retaining the 8-thread per-update sample count; performance falls to 0.275 (h64) and 0.230 (h256), so fewer phases alone do not provide the rescue. An *accumulated-batch control* instead keeps 8 rollout threads but collects five consecutive batches under a frozen policy and consumes all $6{,}400$ transitions in one optimizer phase. This matches both the dose-matched phase count and its per-phase experience without 40-thread parallel collection. All pre-registered predictions hold: realized KL reproduces the dose-matched cell (0.00034/0.00091 at h64/h256), the capacity gap falls to $+0.004$ (3 seeds per width), and performance reaches 0.527/0.523. Together, these controls show that increasing the experience entering each optimizer phase is sufficient to reproduce the protected regime in this setting. The learning-rate intervention below provides an independent route to reducing realized update scale without altering either batch size or phase count.

The mediator across all topology cells. If realized policy displacement is the proximal variable linking topology and width to fragility, it should separate the protected and fragile regimes in every cell, not only in HAPPO. We therefore instrument every topology and record the exact categorical KL of each update (pre-registered; 2 seeds). Because sequential updating can apply multiple optimizer phases within one iteration, our pre-registered cross-topology unit is the net per-iteration

displacement

$$\mathrm{KL}_k^{\mathrm{net}} \;=\; \frac{1}{|\mathcal{B}_k|} \sum_{o \in \mathcal{B}_k} \sum_{a} \pi_{\theta^{(k,0)}}(a \mid o)\, \log \frac{\pi_{\theta^{(k,0)}}(a \mid o)}{\pi_{\theta^{(k,P)}}(a \mid o)}, \tag{2}$$

with $\theta^{(k,0)}$, $\theta^{(k,P)}$ the parameters before the first and after the last of iteration k's P optimizer phases and $\mathcal{B}_k$ its data; the per-update unit evaluates the same quantity across one phase. The separation is clear (Table 2, last column). At h256, the protected cells realize net KLs of 0.0005 (MAPPO) and 0.0009 (dose-matched), whereas the fragile cells lie at 0.0022–0.0058. Width amplification occurs in every cell, but the absolute displacement differs: fragile cells amplify by 3.5–$3.6\times$, protected cells by 2.2–$2.7\times$ from a much lower baseline. The shared-sequential cell is especially informative: its per-update KL is similar to HAPPO's (0.0022), but five sequential phases accumulate to the largest net displacement in the matrix (0.0058), consistent with its large capacity gap ($+0.125$). Cumulatively, the dose-matched cell's whole-budget summed displacement is $12\times$ smaller than the fragile 8-thread cell's. Within the fragile block the exact gap is not strictly monotone in net KL. Realized displacement therefore serves as a regime-level risk marker, while the exact loss magnitude within the fragile regime depends on additional factors.

Predictions beyond the fitted cells. The same account makes testable predictions outside the h64/h256 comparison. Before running h128 beyond its default-rate baseline (0.430), we registered three predictions: realized-KL amplification over h64 in $[1.4, 2.2]$, win rate at 2.5×10^{-4} in $[0.46, 0.51]$, and realized KL at that rate near h64's default ≈ 0.0013. All three hold ($1.55\times$; 0.507, 3 seeds; 0.0012), and the realized-KL ladder is monotone in width ($0.0013 \to 0.0020 \to 0.0034$ for h64/h128/h256).

Two further pre-registered checks probe the account outside the constructed matrix (Appendix E). De-pooling protected MAPPO raises h256 realized displacement by $1.7\times$ and shifts the capacity gap to $+0.037$, but the resulting displacement (0.0008) remains within the protected regime. This reverse intervention is therefore consistent with absolute realized displacement as the empirical risk variable. HATRPO (Kuba et al., 2022) provides an external algorithmic check. Although it uses sequential independent parameters, its trust region explicitly bounds each update. Its realized KL amplification is only $1.08\times$ across the two tested widths (absolute level 0.0005, in the protected band), and its capacity gap is $+0.019$, compared with HAPPO's $+0.105$. An algorithm not designed for this study therefore lands in the regime predicted by the account.

Three additional predictions are tested next: lowering the learning rate should shrink fragile gaps, directly bounding policy change should help at a fixed rate, and the penalty should be strongest where aggressive updates must generalize across heterogeneous agent roles.

4.3 Boundary Conditions: Randomness and Role Heterogeneity

The capacity penalty does not appear equally strongly in every environment. We therefore vary two sources of randomness in protoss_5_vs_5: unit type and starting position (Table 5, Appendix C). The resulting pattern is an interaction. A $+0.083$ gap remains with no randomization; position randomization alone nearly removes it ($+0.007$); unit-type randomization restores a large gap ($+0.088$ alone, $+0.105$ with both). These results identify stochastic agent-role heterogeneity as an amplifier of capacity fragility: a gap remains without randomization, but unit-type randomization consistently produces a large penalty, consistent with aggressive updates fitted to one team composition transferring poorly to another. On deterministic SMACv1, the effect reverses and larger HAPPO networks win more (3s5z: $0.902 \to 0.944$, h64→h128; Appendix B), further showing that the phenomenon is environment dependent.

5 Intervening on Update Scale

The mechanism above gives a direct prediction: if the larger network fails because its realized policy updates are too large for the regime, then reducing the update scale should selectively repair the fragile configurations.

We first intervene through the learning rate. In the capacity $\times$ learning-rate grid (Table 4, Appendix C; Figure 3b), reducing the rate to 2.5×10^{-4} changes the h64–h256 gap from $+0.105$ to $+0.004$: h256 rises from 0.368 to 0.486, while h64 moves from 0.474 to 0.490. The asymmetry

is central to the prediction. h64 varies little across the tested grid (0.464–0.490), whereas h256 moves from the worst to among the best settings (0.368–0.486). The same reduction also mitigates the other fragile topology variants: the no-prefix gap shrinks from $+0.114$ to $+0.041$, and the synchronous-independent gap from $+0.134$ to $+0.051$. Appendix C provides the MAPPO control; the already-protected h256 policy receives no comparable benefit from the same reduction.

The evaluation confound. This intervention also shows why the phenomenon matters beyond HAPPO. A conventional fixed-architecture, fixed-rate comparison at h256 ranks MAPPO above HAPPO by 0.094 (0.462 versus 0.368). Under the update-scale account, however, the two algorithms are not operating at equally calibrated realized update scales. Giving *both* algorithms the same reduced rate of 2.5×10^{-4} raises HAPPO h256 to 0.486 and MAPPO h256 to 0.508 (3 seeds each), reducing the margin roughly fourfold to 0.022 (Welch 95% CI $[-0.080, +0.125]$, $p = 0.489$). The margin therefore shrinks from 0.094 to 0.022, showing that a large fraction of the fixed-rate advantage at this width is sensitive to update-scale calibration. The fact that MAPPO also improves at the moderate reduction is consistent with the mechanism: pooled MAPPO h256 still exhibits a $2.2\times$ width amplification, while a further reduction to 1×10^{-4} hurts (0.411, two seeds), as expected near its stable region.

Bounding realized policy change directly. Learning rate is an indirect control, so we also intervene on policy displacement itself. We add a target-KL early-stopping rule to HAPPO's PPO epochs: optimization stops when an agent's realized approximate KL exceeds 0.0015, roughly h64's realized KL at the default rate. With this single change, h256 recovers from 0.368 to 0.397, while h64 remains essentially unchanged at 0.466, reducing the gap from $+0.105$ to $+0.068$. The direct KL constraint recovers part of the original gap, providing an independent intervention on realized policy movement. Because it caps displacement without changing the upstream gradient statistics, its effect is smaller than learning-rate matching.

6 Representation Content and Optimization Stability Dissociate

The update-scale account explains why the larger network can optimize poorly, but it leaves a natural competing explanation: perhaps h256 simply learns less useful representations. We test this directly using dimensionality-controlled linear probes. Logistic-regression probes predict a teammate's randomized unit type from each agent's recurrent latent state, after projecting all latents to their top 64 principal components so that h64 and h256 are evaluated at identical probe dimensionality (protocol and caveats in Appendix I).

The larger network does not contain less of this task-relevant information. In every algorithm variant, including MAPPO, h256 latents encode more teammate-identity information than h64. Under default-rate HAPPO, h256 encodes more ($+0.088$ vs. $+0.023$) while winning less (0.368 vs. 0.474). After reducing the learning rate, the representational advantage persists ($+0.130$) while no capacity penalty is detected. Representation content therefore dissociates from optimization performance. Larger latent representations retain more linearly decodable teammate information even when task performance deteriorates, separating representational content from the observed capacity penalty.

7 Related Work

Cooperative MARL. Cooperative MARL under the Dec-POMDP formalism (Oliehoek & Amato, 2016) is dominated by CTDE, spanning centralized critics (Lowe et al., 2017; Foerster et al., 2018), value decomposition (Sunehag et al., 2018; Rashid et al., 2018), and policy gradients (Yu et al., 2022; de Witt et al., 2020) (survey: Hernandez-Leal et al., 2019). We study the policy-optimization branch, where update scheme has not been treated as a scaling distinction.

Trust-region policy optimization and its implementation sensitivity. Trust-region methods constrain per-update policy change (Schulman et al., 2015; 2017), but are sensitive to implementation and hyperparameter choices (Henderson et al., 2018; Engstrom et al., 2020; Ilyas et al., 2020; Andrychowicz et al., 2021); HAPPO's monotonic-improvement construction does not prevent the capacity penalty in our setting. Most directly related, Hilton et al. (2022) show PPO lacks batch-size invariance; our results locate the same coupling inside multi-agent training: topology sets each net-

work's update regime, width amplifies the realized step, and their product separates protected from fragile scaling.

Network scaling in deep RL. Enlarging RL networks often destabilizes training (Ota et al., 2021; Bjorck et al., 2021); a growing literature attributes such failures to optimization pathologies rather than representational limits: capacity loss (Lyle et al., 2022), plasticity loss (Lyle et al., 2023), dormant neurons (Sokar et al., 2023), implicit under-parameterization (Kumar et al., 2021), primacy bias (Nikishin et al., 2022), and policy churn, whose reduction aids PPO scaling (Tang & Berseth, 2024). Successful recipes require targeted interventions (Schwarzer et al., 2023; Obando-Ceron et al., 2024; Nauman et al., 2024). We extend this question to MARL, where the same network scales poorly under HAPPO but acceptably under MAPPO, a topology dependence beyond single-agent effects.

Sequential multi-agent optimization. HAPPO and HATRPO (Kuba et al., 2022), generalized by HARL (Zhong et al., 2024), obtain monotonic improvement via sequential per-agent updates; follow-ups study ordering (Wang et al., 2023) and sequence models (Wen et al., 2022). Closest to us, Le & Ta (2025) show sequential updates' importance-sampled advantage variance can grow exponentially with agent count and propose clipped objectives (cf. Munos et al., 2016; Espeholt et al., 2018). Our experiments identify a complementary mechanism: removing the cumulative ratio leaves the capacity gap intact, whereas realized policy displacement consistently separates protected and fragile regimes. A complementary line engineers the update rule itself (Wu et al., 2021; Zhang et al., 2024; Chen et al., 2024; Wu et al., 2026; Feng et al., 2023); they design update rules, whereas we diagnose how existing regimes implicitly rescale width's effect.

Parameter sharing and independent learning. Independent PPO (de Witt et al., 2020) can be strong on SMAC; selective sharing (Christianos et al., 2021) interpolates between the regimes; sharing is usually treated as a sample-efficiency/expressiveness trade-off, with failures on diverse behaviors attributed to cross-agent gradient interference (Tessera et al., 2025). Our results add a third role: synchronous sharing co-occurs with a pooled update regime that stabilizes large networks. Dose matching transfers this protection to independent actors while the shared-sequential cell remains fragile, separating the pooled update regime from parameter sharing itself.

Generalization, stochasticity, and MARL evaluation. Generalization under procedural randomization is a recognized single-agent challenge (Zhang et al., 2018; Farebrother et al., 2018; Cobbe et al., 2019; Kirk et al., 2023); the shift from deterministic SMACv1 (Samvelyan et al., 2019) to randomized SMACv2 (Ellis et al., 2023) brings it to MARL (Papoudakis et al., 2021; Gorsane et al., 2022); our randomness decomposition exploits this design.

8 Discussion and Conclusion

The central contribution of this work is to identify capacity-scaling fragility as an interaction between network capacity and the update regime experienced by each policy network. The four-way identification experiment shows that sequential updating, the cumulative importance ratio, and parameter sharing each fail to account for the pattern on their own. The protected standard configuration is shared, synchronous MAPPO, where the experience of all agents is pooled before each network update. Transferring a matched update regime to independent actors reduces the capacity gap from $+0.134$ to $+0.010$, while phase-count and accumulated-batch controls link this transfer to the experience entering each optimizer phase rather than to parallel data collection itself. Together, these experiments localize the upstream protection to the per-network update regime.

Across these configurations, the common proximal quantity is realized policy displacement. At the same nominal learning rate, increasing width amplifies the policy KL, while protected configurations operate from a much smaller displacement baseline than fragile ones. Matching the realized update scale across widths restores h256 performance, and reducing or directly constraining policy movement attenuates the capacity gap. More broadly, these findings call for a re-examination of fixed-architecture, fixed-learning-rate comparisons in MARL, because algorithms evaluated under the same nominal hyperparameters may operate under substantially different realized optimization regimes. *Limitations:* evidence is strongest on SMACv2; SMACv1/MPE contrasts (Appendices B, H) are directional; the account is measured, not derived (Appendix D).

AI USE STATEMENT

Generative AI tools were used to assist with literature search, provide feedback on experimental design, support result interpretation, assist with code implementation and debugging, and support manuscript drafting, translation, revision, and grammar editing. The research hypotheses and core conceptual framing were developed and refined by the authors. All AI-assisted outputs incorporated into the work were reviewed and verified by the authors, who take responsibility for the final content.

REPRODUCIBILITY STATEMENT

The supplementary material contains all raw training logs and configurations for every run behind every reported number, the per-update diagnostics streams (realized and exact KL), the preregistration documents with timestamped amendments, and the complete pipeline that regenerates each table, data figure, and statistic from those logs. A machine-readable provenance file maps every headline statistic to its source runs, and an automated audit re-derives the manuscript's numbers from the pipeline outputs inside the package. Appendix A lists all hyperparameters and environment versions; the package README documents the regeneration commands.

REFERENCES

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? A large-scale study. In *International Conference on Learning Representations*, 2021.

Nils Bjorck, Carla P. Gomes, and Kilian Q. Weinberger. Towards deeper deep reinforcement learning with spectral normalization. In *Advances in Neural Information Processing Systems*, 2021.

Ruiqing Chen, Xiaoyuan Zhang, Yali Du, Yifan Zhong, Zheng Tian, Fanglei Sun, and Yaodong Yang. Off-agent trust region policy optimization. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3798–3806, 2024.

Filippos Christianos, Georgios Papoudakis, Muhammad A Rahman, and Stefano V Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1989–1998. PMLR, 2021.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, 2019.

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

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N. Foerster, and Shimon Whiteson. SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems Track on Datasets and Benchmarks*, 2023.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep RL: A case study on PPO and TRPO. In *International Conference on Learning Representations*, 2020.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1407–1416. PMLR, 2018.

Jesse Farebrother, Marlos C. Machado, and Michael Bowling. Generalization and regularization in DQN. *arXiv preprint arXiv:1810.00123*, 2018.

Lang Feng, Dong Xing, Junru Zhang, and Gang Pan. FP3O: Enabling proximal policy optimization in multi-agent cooperation with parameter-sharing versatility. *arXiv preprint arXiv:2310.05053*, 2023.

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*, 2018.

Rihab Gorsane, Omayma Mahjoub, Ruan de Kock, Roland Dubb, Siddarth Singh, and Arnu Pretorius. Towards a standardised performance evaluation protocol for cooperative MARL. In *Advances in Neural Information Processing Systems*, 2022.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33, 2019.

Jacob Hilton, Karl Cobbe, and John Schulman. Batch size-invariance for policy optimization. In *Advances in Neural Information Processing Systems*, 2022.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. A closer look at deep policy gradients. In *International Conference on Learning Representations*, 2020.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76, 2023.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2022.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *International Conference on Learning Representations*, 2021.

Bang Giang Le and Viet Cuong Ta. Low variance trust region optimization with independent actors and sequential updates in cooperative multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 39(1):12, 2025.

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

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations*, 2022.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning*, 2023.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, 2016.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: Scaling for compute and sample-efficient continuous control. In *Advances in Neural Information Processing Systems*, 2024.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*, 2022.

Johan Obando-Ceron, Ghada Sokar, Timon Willi, Clare Lyle, Jesse Farebrother, Jakob Foerster, Gintare Karolina Dziugaite, Doina Precup, and Pablo Samuel Castro. Mixtures of experts unlock parameter scaling for deep RL. In *International Conference on Machine Learning*, 2024.

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

Kei Ota, Devesh K. Jha, and Asako Kanezaki. Training larger networks for deep reinforcement learning. *arXiv preprint arXiv:2102.07920*, 2021.

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

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2018.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

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

Max Schwarzer, Johan Obando-Ceron, Aaron Courville, Marc Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level Atari with human-level efficiency. In *International Conference on Machine Learning*, 2023.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*, 2023.

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

Hongyao Tang and Glen Berseth. Improving deep reinforcement learning by reducing the chain effect of value and policy churn. In *Advances in Neural Information Processing Systems*, 2024.

Kale-ab Tessera, Muhammad Arrasy Rahman, Amos J. Storkey, and Stefano Albrecht. Hyper-MARL: Adaptive hypernetworks for multi-agent RL. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Xihuai Wang, Zheng Tian, Ziyu Wan, Ying Wen, Jun Wang, and Weinan Zhang. Order matters: Agent-by-agent policy optimization. In *International Conference on Learning Representations*, 2023.

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

Cuiling Wu, Yaozhong Gan, Junliang Xing, and Ying Fu. MARPO: A reflective policy optimization for multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 29740–29748, 2026.

Zifan Wu, Chao Yu, Deheng Ye, Junge Zhang, Haiyin Piao, and Hankz Hankui Zhuo. Coordinated proximal policy optimization. In *Advances in Neural Information Processing Systems*, volume 34, pp. 26437–26448, 2021.

Ge Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17084–17097, 2021.

Greg Yang and Edward J. Hu. Tensor programs IV: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, 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. In *Advances in Neural Information Processing Systems Track on Datasets and Benchmarks*, 2022.

Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

Wenjing Zhang, Wei Zhang, Wenqing Hu, and Yifan Wang. B2MAPO: A batch-by-batch multi-agent policy optimization to balance performance and efficiency. In *ECAI 2024 - 27th European Conference on Artificial Intelligence*, pp. 3063–3070, 2024.

Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 2024.

A Experimental Details

All experiments use the HARL framework's officially tuned hyperparameters for SMACv2. Unless otherwise noted, every run uses 8 rollout threads, 4 evaluation threads, and 32 evaluation episodes per checkpoint, with evaluation every $\sim$32k environment steps (312 checkpoints over 10^7 steps). The actor networks are recurrent (one GRU layer, data chunk length 10); the critic receives the global state. PPO settings follow HARL's tuned SMACv2 defaults: clip parameter 0.05, 5 actor and 5 critic epochs, GAE $\lambda = 0.95$, $\gamma = 0.99$, entropy coefficient 0.01, huber value loss, max gradient norm 10, and Adam with learning rate 5×10^{-4} for both actor and critic. The identification variants of Section 4.1 change only the manipulated factor (update scheme, parameter sharing, or the cumulative ratio m_i); we verified programmatically, by diffing the full saved configuration of every completed run, that all remaining configuration entries are identical across compared conditions (cross-map runs use each map's own HARL-tuned configuration, identical within every compared pair). Run inclusion follows a rule fixed before analysis: one valid completed run per seed, kept only if training ran its full budget (an explicit completion marker or at least 300 of the 312 scheduled evaluation checkpoints; completed protoss runs record 311, and conditions with shorter schedules use their own full-budget thresholds). A machine-readable provenance table mapping every reported statistic to its source log, seed, and aggregation function is included in the supplementary material.

Dose-matching control. The dose-matched configuration (Table 2, last row) is the synchronous-independent cell with `n_rollout_threads` raised from 8 to 40, so each independent actor's update is computed from 40 trajectories, the number pooled MAPPO's single shared network receives from 5 agents $\times$ 8 rollouts. All other hyperparameters, the 10^7-step budget, and the evaluation protocol are unchanged; because $5\times$ more experience is collected per iteration, the run performs $5\times$ fewer optimizer phases and produces 62 evaluation checkpoints (every $\sim$160k steps) instead of 312. Predictions for this experiment (the gap shrinks well below $+0.134$; h256 recovers substantially while h64 moves little) were pre-registered in a timestamped document before launch and are included in the supplementary material.

Compute. StarCraft II simulation runs on CPU; one 10^7-step run takes roughly 20–40 hours of wall-clock time depending on machine load. The full study comprises over 200 training runs.

Table 3: Win rates on deterministic SMACv1 maps (HAPPO, last-10 evaluation means over completed seeds: 3 per cell, except 3s5z h128 where one run was interrupted at 116 of 165 checkpoints and is excluded under the standard inclusion rule, leaving 2). The capacity effect *reverses*: larger networks achieve higher performance.

Map	h32	h64	h128	Gap (h32 − h128)
3s5z	0.621	0.902	**0.944**	−0.323
corridor	0.007	0.075	**0.172**	−0.165

B SMACv1 Contrast: Environment-Dependent Reversal

Table 3 shows that on SMACv1 maps, where unit types and positions are fixed across episodes, HAPPO follows the expected "bigger is better" pattern. On 3s5z, performance increases from 0.621 (h32) through 0.902 (h64) to 0.944 (h128). The 3s5z runs use a 5M-step budget and the corridor runs a 2M-step budget, each appropriate to its map. The reversal establishes that the capacity effect is environment dependent. Because SMACv1 and SMACv2 differ along several dimensions, this comparison does not by itself isolate randomization as the cause.

C Identification and Randomness-Decomposition Details

Regularization controls (exploratory, one seed each). If the h256 penalty were classical overfitting, regularization should help. It does not: dropout (0.1) collapses HAPPO h256 to 0.159, weight decay (0.01) to 0.031, and tightening the PPO clip from 0.05 to 0.01 leaves 0.378, close to the unmitigated 0.368. The clip bounds per-sample ratios, not realized policy change.

Table 4: Capacity × learning-rate grid on protoss_5_vs_5 (HAPPO, last-10 evaluation means over seeds; Section 5). The capacity gap is a function of the update scale: large at the default rate, near zero in point estimate at half the rate, slightly reversed at one fifth.

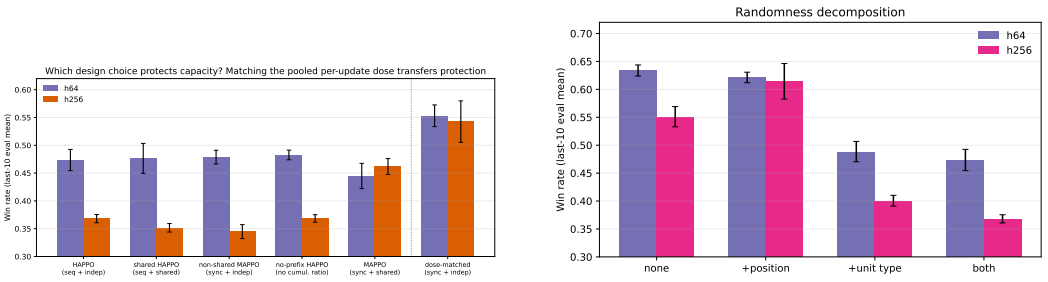
Learning rate	h64	h256	Gap (h64 − h256)
5×10^{-4} (default)	0.474	0.368	$+0.105$
2.5×10^{-4}	0.490	0.486	$+0.004$
1×10^{-4}	0.464	0.486	-0.022

The same reduced rate also mitigates the two fragile variants of Table 2: at 2.5×10^{-4} the no-prefix configuration reaches 0.519 (h64) vs. 0.478 (h256) and the synchronous-independent configuration 0.509 vs. 0.458, the gap figures quoted in Section 5.

Table 5: Randomness decomposition: win rates for HAPPO h64 vs. h256 under different randomness levels (Section 4.3). Unit-type randomization induces a clear h64–h256 gap, while position randomization alone does not.

Setting	Randomness	h64	h256	Gap
fixed_all	None	0.634	0.551	$+0.083$
fixed_team	+ Position	0.621	0.614	$+0.007$
fixed_pos	+ Unit-type	0.489	0.401	$+0.088$
full_random	Both	0.474	0.368	$+0.105$

Figure 4 visualizes Tables 2 and 5; Figure 5 shows the learning-rate intervention alongside the MAPPO control. The effect is selective: h256 HAPPO improves strongly after the reduction, whereas already-protected MAPPO changes much less, as the update-scale account predicts.



(a) Four-way identification.

(b) Randomness decomposition.

Figure 4: Locating the mechanism (visual companion to Tables 2 and 5). (a) Among the standard topology cells, every configuration that gives each policy network its own update stream shows the h64–h256 penalty; sharing + synchronous pooling (default MAPPO) is protected, and handing the same per-update dose to independent actors (rightmost, dotted separator) transfers the protection without sharing. (b) Position randomization alone does not induce a capacity penalty; unit-type randomization restores a clear gap.

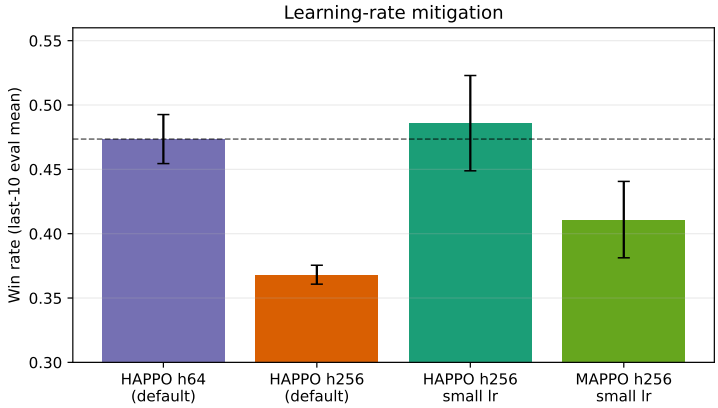


Figure 5: Learning-rate mitigation with controls. Reducing the learning rate improves HAPPO h256, while already-protected MAPPO h256 receives no comparable benefit from the same reduction.

D A Second-Order Sketch of the Update-Scale Account

The account of Section 4.2 can be organized by a standard second-order expansion. For a policy π_θ and parameter update $\Delta\theta$,

$$\mathrm{KL}(\pi_\theta \,\|\, \pi_{\theta+\Delta\theta}) \;\approx\; \tfrac{1}{2}\,\Delta\theta^\top F(\theta)\,\Delta\theta, \tag{3}$$

where F is the Fisher information matrix. For a stochastic gradient step $\Delta\theta = -\eta\,\hat{g}$ with $\hat{g} = g + \varepsilon$, $\mathbb{E}[\varepsilon] = 0$, $\mathrm{Cov}(\varepsilon) = \Sigma/B$ for effective sample count B,

$$\mathbb{E}[\mathrm{KL}] \;\approx\; \tfrac{\eta^2}{2}\left(g^\top F g \;+\; \frac{\mathrm{tr}(F\Sigma)}{B}\right). \tag{4}$$

Equation 4 separates the two levers our experiments manipulate. *Pooling*: the shared-synchronous configuration (default MAPPO) computes one update from all N agents' experience, multiplying B by N and shrinking the noise term (the pooled samples come from the same joint trajectories and are correlated, so the effective gain in independent samples is at most N); the other standard topology cells compute each network's update from one agent's data, and the shared-sequential variant additionally applies N separate updates per epoch whose realized KLs accumulate. *Width*: under standard parameterization, a fixed-η step changes network outputs by an amount that grows

with layer width, the phenomenon underlying zero-shot hyperparameter-transfer results, in which learning rates must shrink with width to keep update sizes stable (Yang & Hu, 2021; Yang et al., 2021). Equation 4 captures the sampling lever directly: a smaller effective B raises the gradient-noise contribution, so reduced pooling pushes $\mathbb{E}[\mathrm{KL}]$ upward at fixed η. Width does not appear explicitly in Equation 4; its effect enters through the parameterization-dependent geometry and is established empirically by our measurements (Section 4.2: $2.7\times$ amplification from h64 to h256 at the default learning rate).

Equation 4 is used as an organizing approximation. PPO performs multiple clipped epochs rather than one gradient step, making the realized mapping $\eta \to \mathrm{KL}$ sublinear (empirically, exponents of roughly 0.5–0.75 in our measurements), while Adam preconditioning alters the quadratic form. Quantitative conclusions therefore come from the direct KL measurements.

Measurement details. The per-update record is the standard batch-mean log-ratio estimator computed on the update's own data,

$$\widehat{\mathrm{KL}} \;=\; \frac{1}{|B|}\sum_{b\in B}\left[\log \pi_{\text{old}}(a_b \mid o_b) - \log \pi_{\text{new}}(a_b \mid o_b)\right], \tag{5}$$

which, as a finite-sample estimate, can be negative for individual updates (16–25% of updates in our runs). The main text reports the mean of $|\widehat{\mathrm{KL}}|$ across updates, which counts policy movement in either direction. Aggregating the signed values instead gives 0.0009 (h64) vs. 0.0029 (h256) at the default rate, a $3.1\times$ amplification; both aggregations therefore show the same width-dependent increase.

Exact-KL validation. Because the action space is discrete, the exact per-state categorical KL $\sum_a \pi_{\text{old}}(a \mid o)\log[\pi_{\text{old}}(a \mid o)/\pi_{\text{new}}(a \mid o)]$ is computable, nonnegative, and low-noise. We pre-registered a validation experiment (predictions and window fixed before launch) recording both estimators in the same updates: 2-million-step runs at the default rate (2 seeds per width) and at 1×10^{-4} (1 seed per width). Aggregating per seed first, exact KL gives 0.00188 (h256) vs. 0.00054 (h64) at the default rate: a $3.5\times$ amplification, inside the pre-registered $[2.0, 3.6]$ window and larger than the $|\widehat{\mathrm{KL}}|$ figure, with per-seed means agreeing to three decimal places, and the two estimators order all four (width, rate) cells identically. At 1×10^{-4}, h256's exact KL (0.00040) falls below h64's default-rate level (0.00054), preserving the matching point. The width amplification is therefore a property of the realized policy change itself, not of the estimator. Full-budget confirmation: HAPPO runs logging exact KL over the entire 10^7-step budget (2 seeds per width) give 0.00231 (h256) vs. 0.00065 (h64), a $3.6\times$ amplification, and their own last-10 win rates reproduce the capacity gap (0.445 vs. 0.382).

The mediator across the identification matrix. A pre-registered follow-up (design and predictions fixed before launch, with a timestamped amendment adding the last two cells and fixing the comparison unit before those cells were launched) instruments every topology cell and records the exact categorical KL in the same runs that produce win rates (2 seeds per cell and width over the full 10^7-step budget). Because sequential updating applies one optimizer phase per agent per iteration, the pre-registered primary unit for cross-topology comparison is the net per-iteration exact KL: the KL between the policy before any update in an iteration and after all of them, evaluated on that iteration's data. Table 6 reports both units. All pre-registered predictions held: fragile cells realize 3.5–$3.6\times$ width amplification (protected cells 2.2–$2.7\times$); the shared-sequential cell, whose per-update KL is indistinguishable from HAPPO's, accumulates the largest net displacement (0.0058, $11.6\times$ the protected MAPPO level); the dose-matched cell realizes both a smaller per-update KL than its 8-thread counterpart (0.0009 vs. 0.0022) and a $12\times$ smaller cumulative displacement over training; and net KL cleanly separates the protected cells from the fragile ones (within the fragile block the ordering is not strictly monotone in the gap), with the same runs' win rates confirming each cell's protected or fragile status. Figure 6 plots the same data at the level of individual seeds; the inference unit throughout is the training seed, not the optimizer update. Absolute realized displacement, rather than width amplification alone, separates the regimes: protected cells also amplify by 2.2–$2.7\times$, but from a much lower baseline. The five topology cells therefore support a regime-level separation in realized displacement, while the numerical bands remain descriptive of these runs rather than transferable KL thresholds.

Phase-count control. The dose-matching intervention bundles a $5\times$ larger per-update sample count with $5\times$ fewer optimizer phases. The pre-registered control isolates the latter: the fragile

Table 6: The mediator measured in every cell (seed-level means over the full 10^7-step budget; 2 seeds per cell and width). Net = per-iteration net exact KL at h256; Amp = h256/h64 net-KL ratio; Win gap = h64−h256 last-10 win-rate gap *in these instrumented runs*; Table 2 gap shown for comparison.

Cell	Net KL (h256)	Amp	Win gap (same runs)	Table 2 gap
MAPPO (shared, sync)	0.00050	2.2×	−0.144	−0.017
dose-matched	0.00091	2.7×	+0.032	+0.010
synchronous-indep.	0.00217	3.5×	+0.093	+0.134
HAPPO	0.00231	3.6×	+0.063	+0.105
shared-sequential	0.00580	3.5×	+0.080	+0.125

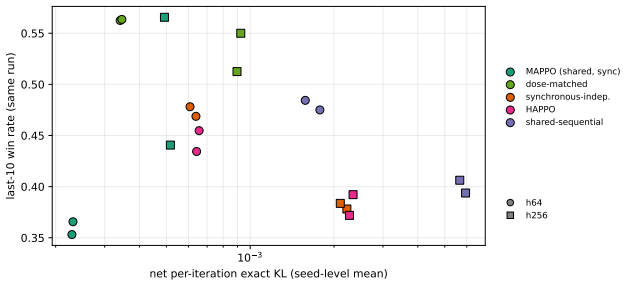


Figure 6: Seed-level view of the mediator matrix: each point is one training seed's net per-iteration exact KL (log scale) against its last-10 win rate, across all instrumented topology cells and both widths. The protected and fragile regimes separate in realized displacement at the level of individual seeds and cell means.

synchronous-independent configuration at 8 threads is trained on only every fifth iteration's data (the remaining iterations collect experience that is discarded), matching the dose-matched run's ∼1,560 optimizer phases while leaving the per-update sample count at the fragile level. Win rates collapse to 0.275 (h64) and 0.230 (h256) over 3 seeds each, far below the 8-thread baseline (0.479/0.345), while the dose-matched run with the same phase count reaches 0.553/0.543. Reducing update frequency while discarding the untrained-on experience therefore does not reproduce the dose-matching rescue. This control isolates the frequency effect, while its fivefold reduction in total optimized experience prevents it from separately identifying the per-update sample count. The nominal capacity gap under the control is $+0.045$ (Welch 95% CI $[-0.024, +0.114]$, $p = 0.144$).

Accumulated-batch control. The complementary pre-registered control matches the dose-matched run's per-phase experience without discarding data and without parallel collection: the fragile synchronous-independent configuration keeps its 8 rollout threads, freezes the policy and critic for five consecutive iterations while storing each rollout batch, and every fifth iteration concatenates the five stored segments ($5 \times 1{,}280 = 6{,}400$ transitions, all collected under the current policy and hence fully on-policy) into a single optimizer phase. This matches the dose-matched run's optimizer-phase count (1,562), per-phase transitions, on-policyness, learning rate, and total budget; the only residual difference is that the transitions come from 8 environment chains contributing five temporally contiguous segments each instead of 40 independent chains. Design, predictions, and analysis rules were pre-registered before launch; timestamped amendments document implementation-equivalence checks and the discard and relaunch of one seed whose diagnostics stream was contaminated by a duplicated process. All three predictions held: per-update exact KL reproduces the dose-matched cell (0.00034 h64, 0.00091 h256; amplification 2.7×, inside the registered $[2.0, 3.6]$ window), the capacity gap falls to $+0.004$ (0.527 vs. 0.523, Welch 95% CI $[-0.044, +0.052]$, $p = 0.830$), and both widths sit at dose-match-level performance, far above the phase-count control. The pre-registered falsification condition (gap $\geq +0.09$ or KL at the 8-thread level) was not triggered. Table 7 summarizes what each condition holds fixed and what it changes.

Agent-count intervention. We ran HAPPO and MAPPO at both widths on zerg_5_vs_5 and zerg_10_vs_10 (tuned defaults, 3 seeds per cell, except zerg_5_vs_5 HAPPO, which reuses the two

Table 7: Design matrix of the dose-decomposition conditions (synchronous-independent topology unless noted). The accumulated-batch control matches the dose-matched run in every training-relevant column except the number of environment chains.

	8-thread	**Dose-matched**	**Phase-count**	**Accumulated**
Rollout chains	8	40	8	8
Transitions per phase	1,280	6,400	1,280	6,400
Optimizer phases	∼7,810	∼1,562	∼1,562	∼1,562
Collected data used	all	all	one fifth	all
Policy frozen during collection	no	no	no	yes
Win rate h64 / h256	0.479 / 0.345	0.553 / 0.543	0.275 / 0.230	0.527 / 0.523
Capacity gap	$+0.134$	$+0.010$	$+0.045$	$+0.004$

completed cross-map runs of Appendix C; crashed partial relaunches are excluded by the standard completeness filter). Moving from $N{=}5$ to $N{=}10$ widens HAPPO's gap from $+0.065$ to $+0.113$, while MAPPO's turns from -0.051 (h256 better on this map) to $+0.044$. The direction matches the account: the sequential-independent topology becomes more sensitive than the pooled one as agent count increases. The MAPPO gap slightly exceeds the pre-registered $+0.03$ target, and absolute win rates on the harder 10v10 map are low (0.13–0.26). This experiment therefore establishes the direction of the agent-count effect rather than a quantitative scaling law.

E Necessity Probe and External Algorithm Check

Both experiments followed pre-registered designs and frozen analysis rules.

De-pooled MAPPO (loss-of-function). The reverse intervention removes most of protected MAPPO's per-phase experience while keeping parameter sharing, synchronous updates, phase count, learning rate, architecture, critic training (full 8-thread buffer), and rollout configuration unchanged. At each iteration the actor update is computed from a random subset of 2 of the 8 rollout threads (the same subset for all agents, resampled each iteration), cutting the actor's per-phase dose from $6{,}400$ to $1{,}600$ transitions. Over 3 seeds per width, realized per-update exact KL rises from the intact 0.00050 to 0.00084 at h256 ($1.7\times$; width amplification $2.27\times$), win rates fall to 0.285 (h64) and 0.249 (h256), an asymmetric loss (0.074 at h64 vs. 0.254 at h256 relative to the intact instrumented cells), and the capacity gap reaches $+0.037$ (Welch 95% CI $[-0.015, +0.088]$, $p = 0.092$). The amplification prediction holds, while the pre-registered fragility criterion ($\geq +0.05$) is not reached. The realized displacement remains below the fragile band (0.0022), favoring absolute realized displacement over per-phase dose alone as the empirical risk variable.

HATRPO external check. HATRPO (Kuba et al., 2022) shares HAPPO's sequential independent-parameter topology but replaces the PPO clip with an explicit per-agent KL constraint and backtracking line search. Under the update-scale account its topology is fragile but its machinery should hold realized displacement approximately width-invariant, so the fixed-learning-rate capacity penalty should be absent or much reduced. We run the HARL implementation under its official tuned protoss_5_vs_5 configuration (kl_threshold 0.005, 20 rollout threads, lr 5×10^{-4}), h64/h256 $\times$ 3 seeds, 10^7 steps, with the same exact-KL instrumentation as the cell matrix; comparisons are within-HATRPO across widths. All three pre-registered predictions held (Table 8): realized per-update exact KL is 0.00047 (h64) vs. 0.00051 (h256), a $1.08\times$ amplification, far below the fragile 3.5–$3.6\times$ band and below the registered $2.0\times$ criterion, with both absolute levels in the protected band and an order of magnitude below the configured constraint; the capacity gap is $+0.019$ (Welch 95% CI $[-0.066, +0.103]$, $p = 0.573$), below the registered $+0.05$ and far below HAPPO's $+0.105$; and the win-rate outcome tracks the realized displacement rather than the width. Neither falsification condition (width-invariant KL with a $\geq +0.09$ gap; amplified KL with no penalty) was triggered.

F Robustness to the Aggregation Metric

Our headline metric is the mean of the last 10 evaluation win rates per seed. Table 9 recomputes the central h64$-$h256 gaps under two alternatives: the mean of the last 50 evaluations and the full-

Table 8: Necessity probe and external check (3 seeds per cell for the new checks; per-update exact KL, seed-level means; win rate = last-10 evaluation mean). Reference rows from the cell-KL matrix (Table 6; 2 seeds per cell).

Configuration	KL h64	KL h256	Amp	Gap (h64−h256)
MAPPO intact (ref.)	0.00023	0.00050	2.2×	−0.017
MAPPO de-pooled	0.00037	0.00084	2.3×	+0.037
sync-indep. (ref., fragile)	0.00062	0.00217	3.5×	+0.134
HATRPO (tuned config)	0.00047	0.00051	1.1×	+0.019
HAPPO (ref., fragile)	0.00065	0.00231	3.6×	+0.105

training AUC (mean over all recorded evaluations). Every qualitative conclusion is unchanged: all four fragile configurations show a positive gap under every metric, and the two protected conditions (default MAPPO; HAPPO at the reduced learning rate) show none under any metric.

Table 9: h64 − h256 win-rate gaps under three aggregation metrics (same completed seeds as the main tables).

Configuration	last-10	last-50	AUC
HAPPO	$+0.105$	$+0.081$	$+0.048$
shared HAPPO	$+0.125$	$+0.141$	$+0.112$
non-shared MAPPO	$+0.134$	$+0.108$	$+0.074$
no-prefix	$+0.114$	$+0.107$	$+0.072$
MAPPO (default)	-0.017	-0.015	-0.007
HAPPO, lr 2.5×10^{-4}	$+0.004$	-0.002	-0.003

G Statistical Significance of the Headline Contrasts

Table 10 reports Welch's unequal-variance t-test and the associated 95% confidence interval for every headline h64−h256 contrast, computed over per-seed last-10 means (the two capacities are independent run sets, so seeds are not treated as paired). Every fragile configuration shows a significant positive gap despite the modest seed counts, on both maps tested at full strength. The protected and mitigated configurations show no detectable gap; with 3–4 seeds their intervals are wide, so we read them as "no measurable penalty" rather than as evidence of exact equality; the dose-matched interval's width comes mostly from one lower h256 seed (0.469 vs. $0.572/0.588$). The actor–critic decomposition's contrast (larger-critic vs. larger-actor cells) falls marginally short of the 0.05 threshold at 3 seeds; its point estimate ($+0.109$) matches the main HAPPO gap almost exactly.

Table 10: Welch t-tests for h64 − h256 win-rate gaps (per-seed last-10 means; the ac-split row contrasts actor64/critic256 against actor256/critic64).

Configuration	Gap	95% CI	p	Seeds
HAPPO	$+0.105$	$[+0.053, +0.157]$	0.003	5, 5
shared HAPPO	$+0.125$	$[+0.019, +0.231]$	0.035	3, 3
non-shared MAPPO	$+0.134$	$[+0.093, +0.175]$	< 0.001	5, 5
no-prefix	$+0.114$	$[+0.088, +0.140]$	< 0.001	5, 5
terran HAPPO	$+0.117$	$[+0.051, +0.182]$	0.008	3, 3
ac-split (a64c256 − a256c64)	$+0.109$	$[-0.013, +0.231]$	0.063	3, 3
MAPPO (default)	-0.017	$[-0.086, +0.052]$	0.553	4, 4
HAPPO, lr 2.5×10^{-4}	$+0.004$	$[-0.051, +0.059]$	0.822	3, 3
dose-matched	$+0.010$	$[-0.123, +0.144]$	0.820	3, 3
accumulated-batch	$+0.004$	$[-0.044, +0.052]$	0.830	3, 3

H Non-StarCraft Check: MPE Simple Spread

Table 11: HAPPO on PettingZoo MPE simple_spread (discrete actions; 5 seeds per cell; last-10 episode-return means ± std, higher is better) with instrumented realized per-update |KL|. The KL measurement replicates across environments; the performance consequence does not follow the SMACv2 pattern on this task (see text).

	lr 5×10^{-4} (default)		**lr 2.5×10^{-4}**	
	h64	**h256**	**h64**	**h256**
Return	-30.7 ± 6.9	-33.6 ± 8.8	-30.3 ± 6.5	-33.6 ± 7.9
Realized $\lvert KL \rvert$	0.0025	0.0054	0.0009	0.0030

We reran the core grid on MPE simple_spread with the same read-only instrumentation (Table 11). The *measurement* half of the account replicates cleanly: at the same default learning rate, h256 realizes $2.1\times$ larger per-update KL than h64 (vs. $2.7\times$ on SMACv2), and halving the learning rate brings h256's realized KL down to near h64's default level. The *performance* half does not follow the SMACv2 pattern on this task: the h64−h256 return gap ($+2.9$ at the default rate) is unchanged at the reduced rate ($+3.4$). The per-seed return distribution suggests a different failure mode: simple_spread outcomes are bimodal (seeds converge near -27 or fall into a ≈ -42 local optimum), h256 enters the poor mode more often (2/5 vs. 1/5 seeds), and reducing the learning rate does not change these mode frequencies. Thus, MPE reproduces the width-dependent KL amplification but not the performance rescue, defining an environmental boundary for the mechanism.

I Probe Methodology

Table 12: Teammate unit-type probe accuracy above the majority baseline (PCA-64 controlled; mean over 3 model seeds; Section 6). Larger latents encode more teammate information in every variant, whether their task performance is worse (default lr) or equal (reduced lr).

Variant	**h64**	**h256**	**Win rate (h256)**
HAPPO (default lr)	$+0.023$	$+0.088$	0.368
HAPPO (lr 2.5×10^{-4})	$+0.035$	$+0.130$	0.486
MAPPO	$+0.073$	$+0.151$	0.462
no-prefix	$+0.048$	$+0.087$	0.369

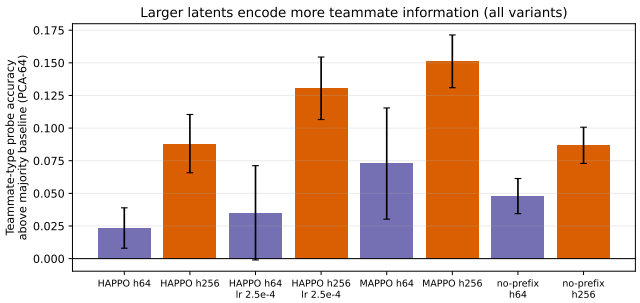


Figure 7: Teammate-type probe accuracy above the majority baseline (PCA-64 controlled, mean ± standard error over 3 model seeds). Larger latents encode more teammate information in every algorithm variant.

Table 12 lists the per-variant accuracies and Figure 7 plots them. For each trained policy we roll out 100 evaluation episodes and record every agent's GRU hidden state at each timestep, together with each agent's per-episode unit type. For an ordered agent pair (i, j), a logistic-regression probe is trained to predict agent j's unit type (3 classes) from agent i's hidden state at each step, using an

episode-level train/test split (70/30) so that no episode contributes to both sets, and skipping steps where either agent is dead. We report test accuracy in excess of the majority-class baseline, averaged over all 20 ordered pairs and over the final policies of 3 training seeds per condition. To rule out the trivial advantage of a higher-dimensional feature vector, all hidden states are projected onto their top 64 principal components (fit on the training split only) before probing, so h64 and h256 are compared at identical probe input dimensionality.

Teammate unit types are partially visible in the current observation (SMACv2 observations include the types of allies within sight range), and probes on the raw observation vector accordingly reach high accuracy; the latent alone decodes types *less* accurately than the raw observation itself (e.g., 0.86 vs. 0.93 for HAPPO h256 at the reduced rate). The probe therefore measures how much teammate information is retained in the recurrent representation under matched dimensionality. Larger recurrent latents retain more of this information in every algorithm variant. The probe measures linearly decodable information and does not test whether the policy uses this information causally.