HyperMARL: Adaptive Hypernetworks for Multi-Agent RL

Kale-ab Abebe Tessera[1], **Arrasy Rahman**[2], **Amos Storkey**[1], **Stefano V. Albrecht**[1]
{k.tessera,a.storkey}@ed.ac.uk,arrasy@utexas.edu

[1]**University of Edinburgh, Edinburgh, UK**
[2]**University of Texas at Austin, Austin, TX, USA**

Abstract

Adaptability to specialised or homogeneous behaviours is critical in cooperative multi-agent reinforcement learning (MARL). Parameter sharing (PS) techniques, common for efficient adaptation, often limit behavioural diversity due to cross-agent gradient interference, which we show can be exacerbated by the coupling of observations and agent IDs. Current remedies typically add complexity through altered objectives, manual preset diversity levels, or sequential updates. We ask: can shared policies adapt without these complexities? We propose *HyperMARL*, a PS approach using hypernetworks for dynamic agent-specific parameters, without altering the RL objective or requiring preset diversity levels. HyperMARL's explicit *decoupling* of observation- and agent-conditioned gradients empirically reduces policy gradient variance, facilitates shared-policy adaptation (including specialisation), and helps mitigate cross-agent interference. Across diverse MARL benchmarks (up to 20 agents), requiring homogeneous, heterogeneous, or mixed behaviours, HyperMARL achieves competitive performance against key baselines – fully shared, non-parameter sharing, and three diversity-promoting methods – while preserving behavioural diversity comparable to non-parameter sharing. These findings establish HyperMARL as a versatile approach for adaptive MARL.

1 Introduction

Specialist and generalist behaviours are critical to collective intelligence, enhancing performance and adaptability in both natural and artificial systems (Woolley et al., 2015; Smith et al., 2008; Surowiecki, 2004; Kassen, 2002; Williams & O'Reilly III, 1998). In Multi-Agent Reinforcement Learning (MARL), this translates to a critical need for policies that can flexibly adapt to meet diverse task demands (Li et al., 2021; Bettini et al., 2024; Albrecht et al., 2024).

Optimal MARL performance thus hinges on being able to represent the required behaviours. While No Parameter Sharing (NoPS) (Lowe et al., 2017) enables specialisation by using distinct per-agent networks, it suffers from significant computational overhead and sample inefficiency (Christianos et al., 2021). Conversely, Full Parameter Sharing (FuPS) (Tan, 1993; Gupta et al., 2017; Foerster et al., 2016), which trains a single shared network, improves efficiency but typically struggles to foster the behavioural diversity necessary for many complex tasks (Kim & Sung, 2023; Fu et al., 2022; Li et al., 2021).

Balancing FuPS efficiency with the capacity for diverse behaviours therefore remains a central open problem in MARL. Prior works have explored intrinsic-rewards (Li et al., 2021; Jiang & Lu, 2021), role-based allocations (Wang et al., 2020a;b), specialised architectures (Kim & Sung, 2023; Li et al., 2024; Bettini et al., 2024), or sequential updates (Zhong et al., 2024). However, these methods often

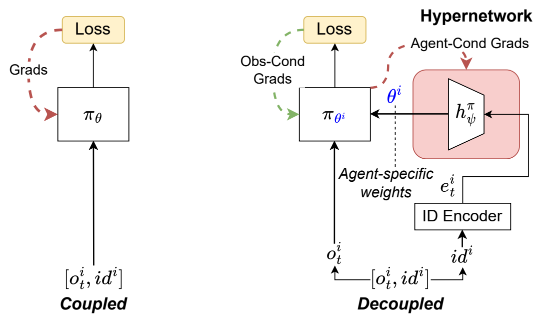


Figure 1: *HyperMARL Policy Architecture.* Common Agent-ID conditioned shared MARL policy (left) vs HyperMARL (right), which uses a hypernetwork to generate agent-specific policies and *decouples* observation- and agent-conditioned gradients.

alter the learning objective, require prior knowledge of optimal diversity levels, involve delicate hyperparameter tuning, or necessitate maintaining agent-specific parameters and sequential updates.

We ask: *Can we design a shared MARL architecture that flexibly supports both specialised and homogeneous behaviours—without altered learning objectives, manual preset diversity levels, or sequential updates?* A key difficulty with FuPS, particularly for diverse behaviours, was hypothesised to be gradient interference among agents, whereby their updates negatively impact each other's learning (Christianos et al., 2021; Zhong et al., 2024). We not only empirically validate this hypothesis but also demonstrate a critical insight: this conflict is significantly exacerbated by the common practice of *coupling observations with agent IDs* within a shared network (Fig. 1, Sec. 3).

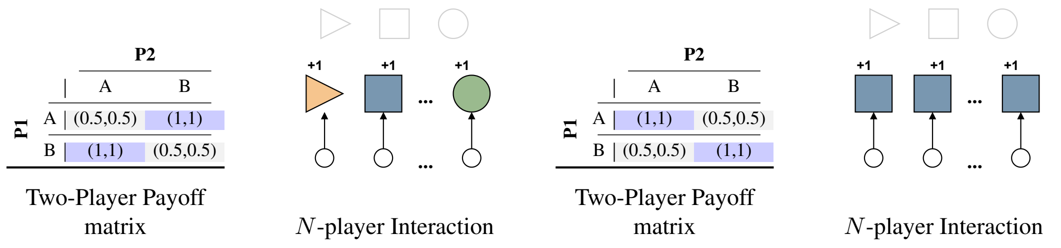
To counteract this coupling, we propose *HyperMARL*, a method using agent-conditioned hypernetworks to generate agent's parameters on the fly and explicitly *decouples* observation- and agent-conditioned gradients (Fig. 1, Section 4.2). While hypernetworks are effective for resolving gradient conflicts in multi-task RL (Navon et al., 2020) and continual learning (von Oswald et al., 2020), we establish their effectiveness in MARL. We also show that HyperMARL empirically attains lower policy gradient variance than FuPS and that this decoupling is critical for specialisation (Sec. 5.1, H), suggesting it helps mitigate cross-agent gradient interference in shared architectures.

We validate HyperMARL on diverse MARL benchmarks—including Dispersion and Navigation (VMAS) (Bettini et al., 2022), Multi-Agent MuJoCo (MAMuJoCo) (Peng et al., 2021), and SMAX (Rutherford et al., 2024)—across environments with two to twenty agents that require homogeneous, heterogeneous, or mixed behaviours. HyperMARL consistently matches or outperforms NoPS, FuPS, and diversity-promoting methods such as DiCo (Bettini et al., 2024),HAPPO (Zhong et al., 2024) and Kaleidoscope (Li et al., 2024), while achieving NoPS-level behavioural diversity while using a shared architecture.

Our contributions are as follows: **I**) We propose *HyperMARL* (Sec. 4), a novel method that uses agent-conditioned hypernetworks to *decouple* observation- and agent-conditioned gradients, enabling the adaptive learning of diverse or homogeneous behaviours without altering the RL objective or requiring preset diversity levels. **II**) We demonstrate through extensive evaluation (Sec. 5) across diverse MARL environments (up to 20 agents) that *HyperMARL* consistently achieves competitive performance against strong baselines (including NoPS, FuPS, DiCo, HAPPO, and Kaleidoscope) while matching NoPS-level behavioural diversity. **III**) We further demonstrate that HyperMARL empirically reduces policy gradient variance compared to FuPS and facilitates specialisation, suggesting the importance of gradient decoupling for mitigating cross-agent interference (Sec. 5.1, H).

2 Background

We model the cooperative task as a Dec-POMDP (Oliehoek & Amato, 2016), using tuple $\langle \mathbb{I}, \mathbb{S}, \{\mathbb{A}^i\}_{i\in\mathbb{I}}, T, R, \{\mathbb{O}^i\}_{i\in\mathbb{I}}, O, \rho_0, \gamma\rangle$, where $\mathbb{S}$ is the set of states with an initial distribution ρ_0,



(a) Specialisation Game **(b)** Synchronisation Game

Figure 2: *Specialisation and Synchronisation Games.* The Specialisation game (left), which encourages *distinct* actions, and the Synchronisation game (right), where rewards encourage *identical* actions. Depicted are their two-player payoff matrices (pure Nash equilibria in blue) and N-player interaction schematics. While simple in form, these games are challenging MARL benchmarks due to non-stationarity and exponentially scaling observation spaces (temporal version).

$\mathbb{A}^i$ and $\mathbb{O}^i$ are the action and observation spaces for each agent $i \in \mathbb{I}$, T and O are the state transition and observation functions, R is the shared reward function and γ is the discount factor. At each timestep t, each agent selects an action $a_t^i \sim \pi^i(\cdot|h_t^i)$ conditioned on its local action-observation history. The goal is to find an optimal joint policy $\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi}} \mathbb{E}_{s_0 \sim \rho_0, \boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \mathbf{a}_t)\right]$.

Specialised Policies and Environments. We say an environment is *specialised* if its optimal joint policy contains at least two distinct, non-interchangeable agent policies (Def. 1 in App. C). Under this mild condition, tasks such as Dispersion (5.1) or our Specialisation Game (F.1) require agents to learn complementary roles rather than identical behaviours.

3 Are Independent or Fully Shared Policies Enough?

Standard independent (**NoPS**) and fully parameter-shared (**FuPS**) policies face inherent trade-offs in MARL. NoPS allows for uninhibited agent specialisation but can be sample inefficient and computationally expensive. FuPS, often conditioned with an agent ID (FuPS+ID), is more efficient but can struggle when agents must learn diverse behaviours. This section investigates the limitations of these common policy architectures. To probe these limitations, we introduce two illustrative environments: the **Specialisation Game**, rewarding *distinct* actions, and the **Synchronisation Game**, rewarding *identical* actions. Both are inspired by prior work (Fu et al., 2022; Bettini et al., 2022; Osborne & Rubinstein, 1994) and extended here to N-agent and temporal settings where agents observe prior joint actions (see Appendix F for full definitions).

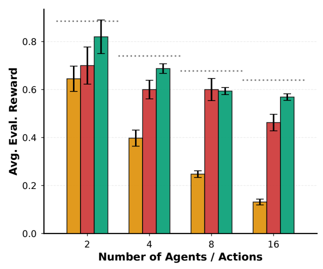
3.1 Limitations of Fully Shared and Independent Policies FuPS *without* agent IDs provably cannot recover optimal pure Nash equilibria in the non-temporal 2-player Specialisation Game (Proof F.3, App. F). In practice, however, FuPS is often conditioned *with* agent IDs, and MARL policies must handle complexities beyond static, two-player interactions. We therefore evaluate standard architectures in the temporal n-player versions of these games[1] We compare three standard architectures trained with REINFORCE (Williams, 1992): 1) **NoPS**: independent policies ($\pi_{\theta^i}(a^i|o^i)$); 2) **FuPS**: a single shared policy ($\pi_\theta(a^i|o^i)$); and 3) **FuPS+ID**: a shared policy incorporating a one-hot agent ID ($\pi_\theta(a^i|o^i, id^i)$). All use single-layer networks, 10-step episodes, and $10,000$ training steps (further details in App. 6).

Empirical Performance. Table 3 shows that neither NoPS nor FuPS consistently achieves the highest mean evaluation rewards. NoPS excels in the Specialisation Game but is outperformed by FuPS (optimal) and FuPS+ID in the Synchronisation Game. Furthermore, the performance gaps widen as the number of agents increases (notably at $n = 8$ and $n = 16$), highlighting the scalability challenges of both fully independent and fully shared policies.

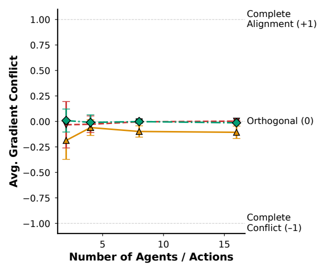
[1]Results for non-temporal (normal-form) variants are in App. F.5.

Figure 3: Average evaluation reward (mean ± 95% CI) for *temporal* Specialisation vs. Synchronisation using REINFORCE (10 seeds). **Bold**: highest mean, no CI overlap. Neither fully shared nor independent policies consistently achieve the highest mean reward.

	Specialisation			Synchronisation		
#Ag	NoPS	FuPS	FuPS+ID	NoPS	FuPS	FuPS+ID
2	**0.88**±0.09	0.50±0.00	0.64±0.10	0.83±0.12	**1.00**±0.00	0.91±0.09
4	**0.74**±0.08	0.25±0.00	0.40±0.07	0.32±0.03	**1.00**±0.00	0.67±0.15
8	**0.68**±0.02	0.12±0.00	0.25±0.03	0.14±0.00	**1.00**±0.00	0.54±0.10
16	**0.64**±0.01	0.06±0.00	0.13±0.02	0.07±0.00	**1.00**±0.00	0.55±0.14



(a) Avg. evaluation reward



(b) Avg. gradient conflict

···· NoPS 🟧 FuPS+ID 🟥 FuPS+ID (No State) 🟩 HyperMARL

Figure 4: *Multi-agent policy gradient methods in the Specialisation environment.* The FuPS+ID (No State) ablation outperforms FuPS+ID, showing near-orthogonal gradients (b), indicating that observation–ID decoupling is important. HyperMARL (MLP) enables this decoupling while leveraging state information, and achieves better performance and reduced gradient conflict than FuPS+ID.

3.2 Why FuPS+ID Fails to Specialise: The Problem of Gradient Conflict Despite being a universal approximator (Hornik et al., 1989), FuPS+ID often struggles to learn diverse policies in practice (Table 3, (Christianos et al., 2021; Zhong et al., 2024)). A key reason is *gradient conflict*: when a single network processes both observation o and agent ID id^i, updates intended to specialise agent i (based on id^i) can conflict with updates for agent j (based on id^j), particularly if they share similar observations but require different actions. This obstructs the emergence of specialised behaviours (conflict measured via inter-agent gradient cosine similarity, App. F.4).

Importance of Observation and ID Decoupling To investigate the effect of entangled observation and ID inputs, we introduce an ablation: *FuPS+ID (No State)*, where the policy $\pi_\theta(a^i \mid id^i)$ conditions *only* on the agent ID, ignoring observations. Surprisingly, *FuPS+ID (No State) outperforms standard FuPS+ID* in the Specialisation Game for all tested N (Figure 4a), even when $N \leq 4$ (where observation spaces are small, suggesting the issue is not merely observation size). Figure 4b reveals why: FuPS+ID (No State) shows near-zero gradient conflict (nearly orthogonal gradients), whereas standard FuPS+ID exhibits negative cosine similarities (conflicting gradients).

These results show that naively coupling observation and ID inputs in shared networks can lead to destructive interference, hindering specialisation. While discarding observations is not a general solution (most tasks require state information), this finding motivates designing architectures that can leverage both state and agent IDs, while minimising interference. Section 4 introduces HyperMARL (Figure 1), which explicitly *decouples observation- and agent-conditioned* gradients through agent-conditioned hypernetworks, leading to improved performance over FuPS variants and reduced gradient conflict compared to standard FuPS+ID (Figure 4).

4 HyperMARL

We introduce *HyperMARL*, an approach that uses agent-conditioned hypernetworks to learn diverse or homogeneous policies *end-to-end*, without modifying the RL objective or requiring preset diversities. By operating within a fully shared paradigm, HyperMARL leverages shared gradient information while enabling specialisation through the decoupling of observation- and agent-conditioned gradients. Pseudocode, scaling, and runtime details are available in App. G.1, G.3, and G.4.

Hypernetworks for MARL As illustrated in Figure 1, for any agent i with context e^i (i.e., either a one-hot encoded ID or a learned embedding), the hypernetworks generate the agent-specific parameters:

$$\theta^i = h^{\pi}_{\psi}(e^i), \quad \phi^i = h^{V}_{\varphi}(e^i), \tag{1}$$

where h^{π}_{ψ} and h^{V}_{φ} are the hypernetworks for the policy and critic, respectively. The parameters θ^i and ϕ^i define each agent's policy π_{θ^i} and critic V_{ϕ^i}, dynamically enabling either specialised or homogeneous behaviours as required by the task.

Linear Hypernetworks Given a one-hot agent ID, $\mathbb{1}^i \in \mathbb{R}^{1\times n}$, a linear hypernetwork h^{π}_{ψ} generates agent-specific parameters θ^i as :

$$\theta^i = h^{\pi}_{\psi}(\mathbb{1}^i) = \mathbb{1}^i \cdot W + b \tag{2}$$

where $W \in \mathbb{R}^{n\times m}$ is the weight matrix (with m the per-agent parameter dimensionality and n is the number of agents), and $b \in \mathbb{R}^{1\times m}$ is the bias vector. Since $\mathbb{1}^i$ is one-hot encoded, each θ^i corresponds to a specific row of W plus the shared bias b. If there is no shared bias term, this effectively replicates training of separate policies for each task (in our case, for each agent) (Beck et al., 2023), since there are no shared parameters and gradient updates are independent.

MLP Hypernetworks for Expressiveness To enhance expressiveness, MLP Hypernetworks incorporate hidden layers and non-linearities:

$$\theta^i = h^{\pi}_{\psi}(e^i) = f^{\pi}_{\psi_1}\left(g^{\pi}_{\psi_2}(e^i)\right) \tag{3}$$

where $g^{\pi}_{\psi_2}$ is an MLP processing the agent context e^i, and $f^{\pi}_{\psi_1}$ is a final linear output layer. Unlike linear hypernetworks, MLP hypernetworks increase parameter count and do not guarantee distinct per-agent weights, creating a trade-off between expressiveness and computational overhead.

4.1 Agent Embeddings and Initialisation

The agent embedding e^i is a one-hot encoded ID for Linear Hypernetworks. For MLP Hypernetworks, we use learned agent embeddings, orthogonally initialised and optimised end-to-end with the hypernetwork. HyperMARL's hypernetworks are themselves initialised such that the generated agent-specific parameters (θ^i, ϕ^i) initially match the distribution of standard direct initialisation schemes (e.g., orthogonal for PPO, preserving fan in/out), promoting stable learning.

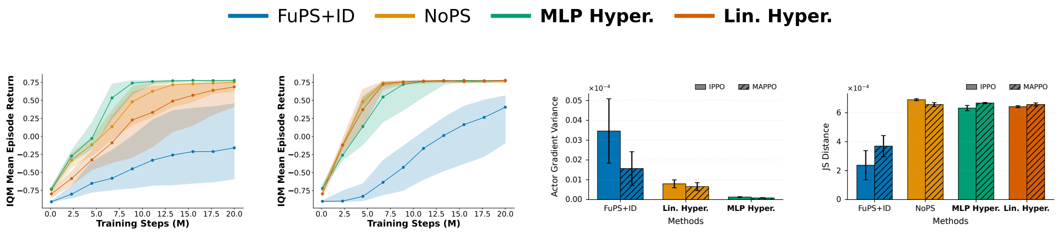
4.2 Gradient Decoupling in HyperMARL

A core difficulty in FuPS is cross-agent gradient interference (Christianos et al., 2021; Zhong et al., 2024). HyperMARL mitigates this by generating each agent's parameters through a shared hypernetwork, thereby *decoupling agent-conditioned* and *observation-conditioned* gradients.

Hypernetwork gradients. Consider a fully cooperative MARL setting with a centralised critic, we can formulate the policy gradient for agent i as follows (Albrecht et al., 2024; Kuba et al., 2021):

$$\nabla_{\theta^i} J(\theta^i) \;=\; \mathbb{E}_{\mathbf{h_t},\mathbf{a_t}\sim\boldsymbol{\pi}}\Big[A(\mathbf{h_t},\mathbf{a_t})\,\nabla_{\theta^i}\log\pi_{\theta^i}(a^i_t \mid h^i_t)\Big],$$

where $\mathbf{h_t}$ and $\mathbf{a_t}$ are the joint histories and joint actions for all agents, θ^i denotes the parameters of agent i, and $A(\mathbf{h_t},\mathbf{a_t}) = Q(\mathbf{h_t},\mathbf{a_t}) - V(\mathbf{h_t})$ is the advantage function.



(a) IPPO (b) MAPPO (c) Actor Gradient Var. (d) Policy Diversity

Figure 5: *Performance and gradient analysis.* **(a,b)** IPPO and MAPPO on Dispersion (20M timesteps) - IQM of Mean Episode Return with 95% bootstrap CIs: Hypernetworks match NoPS performance while FuPS struggle with specialisation. Interval estimates in App. J.2.1. **(c)** Actor gradient variance: Hypernetworks achieve lower gradient variance than FuPS+ID. **(d)** Policy diversity (SND with Jensen–Shannon distance): Hypernetworks achieve NoPS-level diversity while sharing parameters.

Decoupling. In HyperMARL each agent's policy weights are produced by the hypernetwork h^{π}_{ψ}: $\theta^i = h^{\pi}_{\psi}(e^i)$, so we optimise a *single* parameter vector ψ. Applying the chain rule:

$$\nabla_\psi J(\psi) = \sum_{i=1}^{I} \underbrace{\nabla_\psi h^{\pi}_{\psi}(e^i)}_{\mathbf{J}_i \text{ (agent-conditioned)}} \underbrace{\mathbb{E}_{\mathbf{h}_t,\mathbf{a}_t\sim\boldsymbol{\pi}}\left[A(\mathbf{h}_t,\mathbf{a}_t)\,\nabla_{\theta^i}\log\pi_{\theta^i}(a^i_t \mid h^i_t)\right]}_{Z_i \text{ (observation-conditioned)}}. \tag{4}$$

- **Agent-conditioned factor $\mathbf{J}_i$.** This Jacobian depends only on the fixed embedding e^i and the hypernetwork weights ψ, therefore, it is *deterministic* with respect to mini-batch samples (as e^i and ψ are fixed per gradient step), separating agent identity from trajectory noise.
- **Observation-conditioned factor Z_i.** The expectation averages trajectory noise *per agent* i for its policy component π_{θ^i}, prior to transformation by $\mathbf{J}_i$ and aggregation.

The crucial structural decoupling in Equation (4) ensures HyperMARL first averages noise per agent (via factor Z_i) before applying the deterministic agent-conditioned transformation $\mathbf{J}_i$. This mitigates gradient interference common in FuPS+ID (Christianos et al., 2021; Zhong et al., 2024), where observation noise and agent identity become entangled (see Equation (12) in App. G.2). This is the MARL analogue of the task/state decomposition studied by (Sarafian et al., 2021, Eq. 18) in Meta-RL. Section 5.1 empirically verifies the predicted variance drop, and ablations in Section H demonstrate that disabling decoupling degrades performance, underscoring its critical role.

5 Experiments

We structure our experiments to directly answer two key research questions: **Q1: Specialised Policy Learning:** Can *HyperMARL* effectively learn *specialised policies* via a shared hypernetwork? **Q2: Effectiveness in Homogeneous Tasks**: Is *HyperMARL* competitive in environments that necessitate homogeneous behaviours?

Experimental Setup. We evaluate HyperMARL in 18 scenarios across four environments (Dispersion (Bettini et al., 2022), Navigation (Bettini et al., 2022), MAMuJoCo (Peng et al., 2021), and SMAX (Rutherford et al., 2024)) and test varying agent counts (2–20) and behaviours. We compare against standard (**FuPS+ID**, **NoPS**) and diversity-promoting baselines (**DiCo** (Bettini et al., 2024), **HAPPO** (Zhong et al., 2024), **Kaleidoscope** (Li et al., 2024)) using an IPPO/MAPPO (De Witt et al., 2020; Yu et al., 2022) backbone. Following best practices (Patterson et al., 2024), we use original baseline codebases and settings for all comparisons. Full experimental details are in Appendices I.1, I.2.1, and K.

Measuring Policy Diversity. To measure the diversity of the policies we System Neural Diversity (SND) (Bettini et al., 2023) (Equation 5) with Jensen-Shannon distance (details in App. I.2.2).

Table 1: *Mean episode return in MAMuJoCo for MAPPO variants(IQM, 95% CI).* HyperMARL achieves the highest IQM in 3/4 scenarios (bold), and is the only method with shared actors to demonstrate stable learning in the notoriously difficult 17-agent Humanoid environment (see Figure 6 for learning dynamics). * indicates CI overlap with the top score.

Scenario	HAPPO	FuPS+ID	Ind. Actors	HyperMARL (Ours)
Humanoid-v2 17x1	6501.15* (3015.88, 7229.79)	566.12 (536.36, 603.01)	6188.46* (5006.13, 6851.74)	**6544.10** (3868.00, 6664.89)
Walker2d-v2 2x3	4748.06* (4366.94, 6230.81)	4574.39* (4254.21, 5068.32)	4747.05* (3974.76, 6249.58)	**5064.86** (4635.10, 5423.42)
HalfCheetah-v2 2x3	6752.40* (6130.42, 7172.98)	6771.21* (6424.94, 7228.65)	6650.31* (5714.68, 7229.61)	**7063.72** (6696.30, 7325.36)
Ant-v2 4x2	6031.92* (5924.32, 6149.22)	**6148.58** (5988.63, 6223.88)	6046.23* (5924.62, 6216.57)	5940.16* (5485.77, 6280.59)

5.1 Q1: Specialised Policy Learning

Learning Diverse Behaviour (Dispersion) Figures 5a and 5b show that FuPS variants (IPPO-FuPS, MAPPO-FuPS – (●)) can struggle to learn the diverse policies required by Dispersion (even when running for longer - Fig. 19), while their NoPS counterparts (IPPO-NoPS, MAPPO-NoPS–(●)) converge to the optimal policy, corroborating standard FuPS limitations to learn diverse behaviour. In contrast, HyperMARL (both linear and MLP variants) (●, ●) match NoPS performance, suggesting that a shared hypernetwork can effectively enable agent specialisation. *SND policy diversity measurements* (Fig. 5d) confirm FuPS variants achieve lower behavioural diversity than NoPS, while HyperMARL notably matches NoPS-level diversity.

Gradient Variance. HyperMARL (IPPO and MAPPO variants) also exhibits lower mean policy gradient variance than FuPS+ID across actor parameters (Fig. 5c). This aligns with their ability to learn diverse behaviours and supports the hypothesis that its gradient decoupling mechanism (Sec. 4.2) enhances training stability.

Diversity at Complexity and Scale (MAMuJoCo). In the challenging MAMuJoCo heterogeneous control tasks (Table 1), HyperMARL (MLP variant) is broadly competitive. Notably, unlike HAPPO and MAPPO (independent actors), HyperMARL uses a shared actor and parallel updates, and yet manages strong performance, even in the 17-agent Humanoid-v2 notoriously difficult heterogeneous task(Zhong et al., 2024) (Fig. 6), matching methods that employ independent actors and sequential updates.

Adaptability (Navigation). Navigation tasks (Bettini et al., 2022) assess adaptability to homogeneous, heterogeneous, and *mixed* goals (some agents have the same goals, others different). We compare HyperMARL with baselines including DiCo Bettini et al. (2024). While using DiCo's optimal preset diversity for n=2 agents, we note that identifying appropriate diversity levels for DiCo with larger teams ($n > 2$) via hyperparameter sweeps proved challenging (see Tables 12 and 13).

Across all tested goal configurations (shared, unique, and mixed), HyperMARL consistently achieves strong performance (Figure 7). It generally matches or outperforms NoPS and FuPS, and outperforming DiCo. Interestingly, unlike in sparse-reward tasks like Dispersion, FuPS remains competitive with NoPS and HyperMARL in Navigation scenarios requiring diverse behaviours for smaller teams ($n \in \{2, 4\}$), likely due to Navigation's dense rewards. However, HyperMARL distinguishes itself as the strongest method for n=8 agents, highlighting its effectiveness in handling more complex coordination challenges.

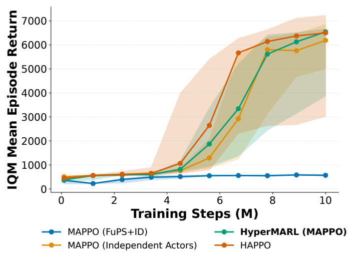
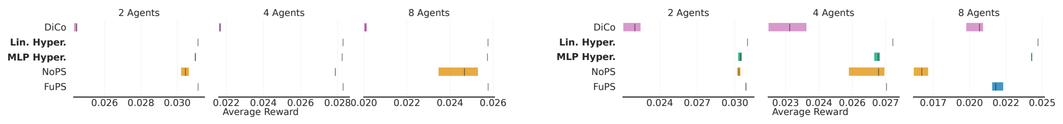
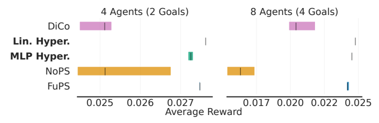


Figure 6: *17-agent Humanoid learning dynamics (IQM, 95% CI).* HyperMARL, using shared actors, outperforms MAPPO-FuPS (non-overlapping CIs) and matches the performance of methods employing non-shared or sequential actors. This challenging environment is known for its high variance (Zhong et al., 2024).

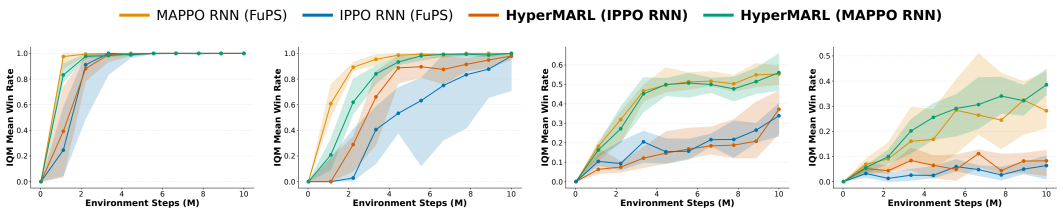


(a) Shared Goals (b) Unique Goals



(c) Mixed Goals (Half Agents Share Goals)

Figure 7: *Average Reward (IQM, 95% CI) in Navigation for IPPO Variants.* HyperMARL adapts robustly across goal configurations—(a) shared, (b) unique, and (c) mixed. Both linear and MLP versions consistently match or outperform IPPO baselines and DiCo, with the margin widening as the number of agents grows. Sample-efficiency curves appear in App. J.4.



(a) 2s3z (b) 3s5z (c) SMACv2 10 Units (d) SMACv2 20 Units

Figure 8: *IQM and 95% CI of mean win rate in SMAX.* Performance of FuPS Recurrent IPPO and MAPPO and HyperMARL (MLP) on SMAX. HyperMARL performs comparably to FuPS baselines across all environments, demonstrating its effectiveness in tasks requiring homogeneous behaviours and using recurrency. Inverval estimates in App. 22.

5.2 Q2: Effectiveness in Homogeneous Tasks

SMAX. Finally, we evaluate HyperMARL (MLP) on SMAX, where recurrent FuPS is the established baseline (Rutherford et al., 2024; Yu et al., 2022; Fu et al., 2022). Figure 8 shows while some FuPS variants might exhibit marginally faster initial convergence on simpler maps, HyperMARL achieves comparable final performance on all maps, using the same GRU backbone for partial observability. These results highlights two points: (i) HyperMARL is fully compatible with recurrent architectures essential under partial observability, and (ii) it has no intrinsic bias toward specialisation and can converge homogenous behaviour when it is optimal (also shown with strong same-goal Navigation performance (Fig. 7a)), even with large observation spaces and many agents.

Summary. Our empirical results confirm HyperMARL effectively addresses both research questions. For **Q1 (Specialisation)**, across Dispersion, MAMuJoCo, and Navigation, HyperMARL learned specialised policies, matched NoPS-level diversity and performance where FuPS+ID struggled, and scaled to complex, high-agent-count heterogeneous tasks. For **Q2 (Homogeneity)**, HyperMARL demonstrated competitive performance against strong FuPS baselines in SMAX and shared-goal Navigation, confirming its versatility. Additionally in App. H, we show the importance of HyperMARL's *gradient decoupling* (Sec. 4.2) and initialisation scaling (Sec. 4.1).

6 Related Work

Learning diverse policies with FuPS is a known challenge in MARL. Prior approaches address this by altering the learning objective with intrinsic rewards (Li et al., 2021), enforcing sequential updates (Zhong et al., 2024), or using architectural constraints (Kim & Sung, 2023; Bettini et al., 2024; Li et al., 2024). In contrast, HyperMARL uses agent-conditioned hypernetworks to generate agent-specific parameters, enabling specialisation without modifying the RL objective or requiring preset diversity levels. While hypernetworks have been used in MARL for value-function mixing (Rashid et al., 2020) or parallel work in zero-shot generalization (Fu et al., 2025), our work is the first to leverage them for adaptive specialisation via gradient decoupling, a mechanism we find critical for learning specialised behaviours. A more detailed literature review is available in Appendix E.

7 Conclusion

We introduced *HyperMARL*, an approach that uses agent-conditioned hypernetworks to generate per-agent parameters without modifying the standard RL objective or requiring preset diversity levels. Our results show HyperMARL can adaptively learn specialised, homogeneous, or mixed behaviours in settings with up to 20 agents. We also observe a link between HyperMARL's performance and reduced policy gradient variance, underscoring the importance of decoupling observation- and agent-conditioned gradients. Overall, these findings establish HyperMARL as a promising architecture for diverse MARL tasks. We discuss limitations in App. A, most notably parameter count, which can be remedied by parameter-efficient hypernetworks (e.g., chunked variants (von Oswald et al., 2020; Chauhan et al., 2024)).

8 Acknowledgements

We would like to thank Samuel Garcin, Max Tamborski, Dave Abel, Timothy Hospedales, Trevor Mcinroe, Elliot Fosong and Aris Filos-Ratsikas for fruitful discussions on early versions of this work. An author on this project has received funding towards this work from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101120726.

Appendix

A Limitations

Hypernetworks generate weights for target networks, which can lead to high-dimensional outputs and many parameters for deep target networks, particularly when using MLP-based hypernetworks. While HyperMARL uses more parameters than NoPS and FuPS for few agents, it scales almost constantly with agent count, an attractive property for large-scale MARL. Parameter efficiency could be improved through chunking techniques (von Oswald et al., 2020; Chauhan et al., 2024), or low-rank weight approximations. This parameter overhead is often acceptable in RL/MARL given typically smaller actor-critic networks, and HyperMARL's favorable agent scaling (see App. G.3).

B Broader Impact

This paper presents work whose goal is to advance the field of Multi-Agent Reinforcement Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

C Specialised Policies and Environments

Specialisation plays a key role in MARL, yet remains under-defined, so we define *specialised environments* and *specialised policies*.

Definition 1. *An environment is specialised if the following both hold:*

1. ***Distinct Agent Policies.*** *The optimal joint policy* $\boldsymbol{\pi}^*$ *consists of at least two distinct agent policies, i.e.,* $\exists i, j \in \mathbb{I}$ *such that* $\pi^i \neq \pi^j$.
2. ***Non-Interchangeability.*** *Any permutation* σ *of the policies in* $\boldsymbol{\pi}^*$*, denoted as* $\boldsymbol{\pi}^\sigma$*, results in a weakly lower expected return:*

$$\mathbb{E}_{\mathbf{h}\sim\boldsymbol{\pi}^\sigma}[G(\mathbf{h})] \leq \mathbb{E}_{\mathbf{h}\sim\boldsymbol{\pi}^*}[G(\mathbf{h})],$$

with strict inequality if the joint policies are non-symmetric *(i.e., swapping any individual policy degrades performance).*

For example, consider a *specialised environment* such as a football game, optimal team performance typically requires players in distinct roles (e.g., "attackers," "defenders"). Permuting these roles (i.e., exchanging their policies) would typically lead to suboptimal results. Here, agents develop *specialised policies* by learning distinct, complementary behaviours essential for an optimal joint policy. While agents with heterogeneous capabilities (e.g., different action spaces) are inherently specialised, homogeneous agents can also learn distinct policies. Such environments are analysed in Sections F.1 and 5.1.

D Measuring Behavioural Diversity

D.1 Quantifying Team Diversity

We quantify policy diversity using System Neural Diversity (SND) (Bettini et al., 2023), which measures behavioural diversity based on differences in policy outputs:

$$\text{SND}\left(\left\{\pi^i\right\}_{i\in\mathbb{I}}\right) = \frac{2}{n(n-1)|\mathcal{O}|}\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{o\in\mathcal{O}} D\left(\pi^i(o), \pi^j(o)\right). \tag{5}$$

where n is the number of agents, $\mathcal{O}$ is a set of observations typically collected via policy rollouts, $\pi^i(o_t)$ and $\pi^j(o_t)$ are the outputs of policies i and j for observation o_t, and D is our distance function between two probability distributions.

In contrast to Bettini et al. (2023), we use Jensen-Shannon Distance (JSD) (Endres & Schindelin, 2003; Lin, 1991) as D, rather than the Wasserstein metric (Vaserstein, 1969). As shown in Appendix D.2, JSD is a robust metric for both continuous and discrete cases, and provides a more reliable measure of policy distance.

D.2 Finding a Suitable Distance Function for Policy Diversity

The choice of distance function D in Equation 5 is crucial for accurately measuring policy diversity. In MARL, policies are often represented as probability distributions over actions, making the choice of distance function non-trivial.

Bettini et al. (2024) use the Wasserstein metric for continuous policies (Vaserstein, 1969) as distance function D, while McKee et al. (2022) use the total variation distance for discrete policies. For discrete policies, Wasserstein distance would require a cost function representing the cost of changing from one action to another, which might not be trivial to come up with. On the other hand, although well-suited for discrete policies, TVD might miss changes in action probabilities because it measures the largest difference assigned to an event (i.e. action) between two probability distributions.

We consider a simple example to illustrate this point. Suppose we have two policies π^1 and π^2 with action probabilities as shown in Figure 9. π^1 stays constant, while π^2 changes gradually over timesteps. We see that even as π^2 changes over time, the $TVD(\pi^1, \pi^2)$ between π^1 and π^2 remains constant. This is because TVD only measures the largest difference between the two distributions, and does not consider the overall difference between them. On the other hand, the Jensen-Shannon distance (JSD) (Endres & Schindelin, 2003), which is the square root of the Jensen-Shannon divergence, does not have this problem as it is a smooth distance function. Furthermore, it satisfies the conditions for being a metric – it is non-negative, symmetry, and it satisfies the triangle inequality.

For continuous policies, as shown in Figure 10, JSD exhibits similar trends to the Wasserstein distance. Since JSD is a reasonable metric for both continuous and discrete probability distributions, we will use it as the distance metric for all experiments and propose it as a suitable distance function for measuring policy diversity in MARL.

We also summarise the properties of the various distance metrics in Table 2.

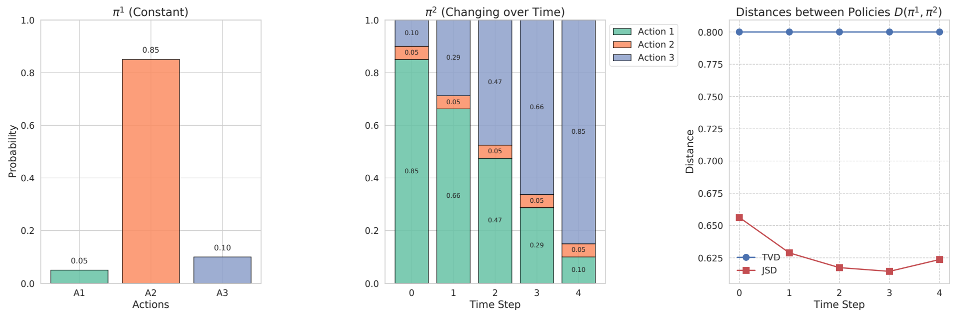


Figure 9: Gradual changes in π^2, result in gradual changes in the Jensen-Shannon distance (JSD), while the Total Variation Distance (TVD) can miss changes in action probabilities.

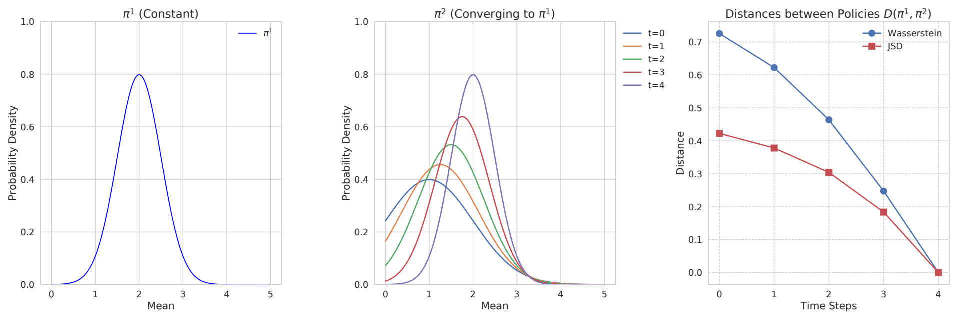


Figure 10: Jensen-Shannon distance (JSD) trends similarly to Wasserstein distance when we have continuous policies.

Method	**Kinds of Actions**	**Metric**	**Smooth**	**Formula**
Wasserstein Distance (Vaserstein, 1969)	Continuous*	Metric	Yes	$W(p,q) = \left(\inf_{\gamma \in \Gamma(p,q)} \int_{\mathbb{R}\times\mathbb{R}} \lvert x-y \rvert \, d\gamma(x,y) \right)^{1/p}$
Total Variation Distance	Discrete	Metric	No	$TV(p,q) = \frac{1}{2}\sum_x \lvert p(x) - q(x) \rvert$
Jensen-Shannon Divergence (Lin, 1991)	Both	Divergence	Yes	$JSD(p \parallel q) = \frac{1}{2} D_{KL}(p \parallel m) + \frac{1}{2} D_{KL}(q \parallel m),\ m = \frac{1}{2}(p+q)$
Jensen-Shannon Distance (Endres & Schindelin, 2003)	Both	Metric	Yes	$\sqrt{JSD(p \parallel q)}$

Table 2: Measure Policy Diversity

E More detailed Related Work

Hypernetworks in RL and MARL. Hypernetworks are effective in single-agent RL for meta-learning, multi-task learning, and continual learning (Beck et al., 2023; 2024; Sarafian et al., 2021; Huang et al., 2021). In MARL, QMIX (Rashid et al., 2020) used hypernetworks (conditioned on a global state) to mix per-agent Q-values; however, each agent's own network remained a standard GRU. Parallel work, CASH (Fu et al., 2025), conditions hypernetworks on local observations and team capabilities for zero-shot generalization with heterogeneous action spaces. In contrast, we focus on agent-conditioned hypernetworks for adaptive specialisation within a fixed state-action setting, leveraging gradient decoupling (absent in CASH) that we find critical for specialised behaviours.

Variants of Parameter Sharing. Selective Parameter Sharing (SePS) (Christianos et al., 2021) shares weights between similar groups of agents, identified via trajectory clustering. Pruning methods (Kim & Sung, 2023; Li et al., 2024) split a single network into agent-specific subnetworks.

Learning Diverse Policies. Shared parameters often limit policy diversity (Christianos et al., 2021; Kim & Sung, 2023; Fu et al., 2022; Li et al., 2021). Proposed solutions include: (**1**) maximizing mutual information between agent IDs and trajectories (Li et al., 2021), (**2**) role-based methods (Wang et al., 2020a;b), or (**3**) methods that use structural modifications or constraints to induce diversity in agent policies (Kim & Sung, 2023; Bettini et al., 2024; Li et al., 2024). Outside FuPS/NoPS, HAPPO (Zhong et al., 2024) uses a non-shared centralised critic with individual actors updated sequentially to learn heterogeneous behaviours.

F Specialisation and Synchronisation Games

To study the challenges of specialisation and coordination in an isolated setting, we introduce the Specialisation and Synchronisation Games, drawing inspiration from a version of the XOR game Fu et al. (2022), VMAS's Dispersion Bettini et al. (2022) and coordination and anti-coordination games in game theory (Osborne & Rubinstein, 1994). These environments encourage agents to take distinct actions (Specialisation) or take identical actions (Synchronisation). Despite their deceptively simple payoff structure, these games present substantial learning challenges – non-stationary reward distributions driven by others' adapting behaviours and in their temporal extension, the joint observation spaces grows exponentially with the number of agents.

F.1 Specialisation and Synchronisation Games Description

Specialisation Game. Agents are encouraged to choose *distinct* actions. In the simplest setting, it is a two-player matrix game where each agent selects between two actions (A or B). As shown in Figure 2a, agents receive a payoff of 1.0 when their actions differ (creating two pure Nash equilibria on the anti-diagonal) and 0.5 when they match. This structure satisfies Definition 1, since optimal joint policies require complementary, non-identical strategies. There is also a symmetric mixed-strategy equilibrium in which each agent plays A and B with probability 0.5.

Synchronisation Game. Conversely, agents are encouraged to coordinate and choose *identical* actions. The payoff matrix inverts the Specialisation game's structure, agents receive 1 for matching actions and 0.5 for differing ones. This creates two pure Nash equilibria along the diagonal of the payoff matrix (Figure 2b), and incentivises uniform behaviour across agents.

N-Agent Extension. Both games naturally scale to n agents and n possible actions. In Specialisation, unique actions receive a payoff of 1.0, while selecting the same action receives payoffs of $\frac{1}{k}$, where k is the number of agents choosing that action. In contrast, in Synchronisation, agents receive maximum payoffs (1.0) only when all actions match. For partial coordination, rewards follow a hyperbolic scale, $\frac{1}{n-k+1}$, encouraging agents to align their choices. Visualisations in Figure 2 and detailed reward profiles appear in Figure 11.

F.2 General-n Payoff Definitions

Both games generalise naturally to n agents and n possible actions. We show the reward profiles for $n = 5$ agents in Figure 11.

Let $\mathbf{a} = (a^1, \dots, a^n) \in \{1, \dots, n\}^n$ and $k_a = \big|\{ j : a^j = a\}\big|$ be the joint action profile and the count of agents choosing action a, respectively.

Temporal Extension. To model sequential decision-making, we extend each normal-form game into a repeated Markov game, where the state at time t is the joint action at time $t-1$. At each step t all agents observe a^{t-1}, choose a_i^t, and receive the original Specialisation or Synchronisation payoff. This repeated setup isolates how agents adapt based on past joint actions, exposing temporal patterns of specialisation and coordination.

F.2.1 Specialisation Game

The reward is formulated as follows:

$$r^i_{\text{spec}}(\mathbf{a}) = \begin{cases} 1.0, & \text{if } k_{a^i} = 1 \quad \big(\text{unique action}\big); \\ \dfrac{1}{k_{a^i}}, & \text{if } k_{a^i} > 1 \quad \big(\text{shared action}\big). \end{cases}$$

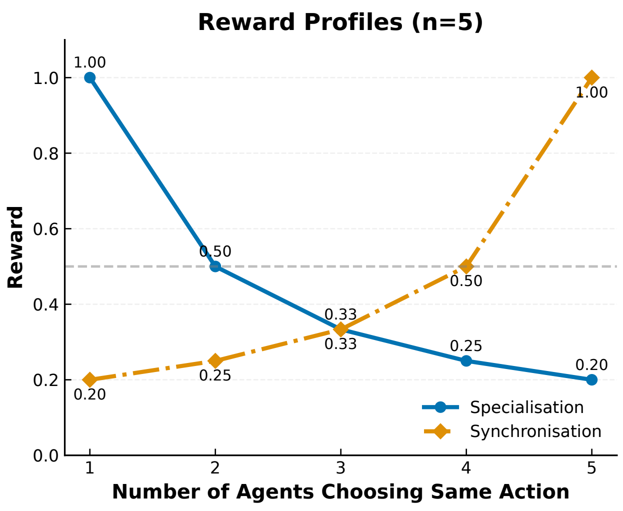


Figure 11: Reward profiles for the Specialisation (blue) and Synchronisation (orange) games with $n = 5$ agents. In the Specialisation game, an agent's payoff peaks when it selects a unique action, and then decays as when actions are shared. In the Synchronisation game, payoffs follow a hyperbolic scale $1/(n - k + 1)$, reaching maximum only under full consensus, thereby incentivising alignment.

F.2.2 Synchronisation Game

The reward is formulated as follows:

$$r^i_{\text{sync}}(\mathbf{a}) = \frac{1}{n - k_{a^i} + 1},$$

so that $r^i_{\text{sync}} = 1.0$ when $k_{a^i} = n$ (all agents select the same action), and otherwise follows a hyperbolic scale encouraging consensus.

F.3 Proof that FuPS cannot represent the optimal policy in the two-player Specialisation Game

Theorem 1. *A stochastic, shared policy without agent IDs cannot learn the optimal behaviour for the two-player Specialisation Game.*

Proof. Let π be a shared policy for both agents, and let $\alpha = \mathbb{P}(a_i = 0)$ represent the probability of any agent choosing action 0. The expected return of π for each agent is:

$$\begin{aligned} E[R(\pi)] &= \mathbb{P}(a_0 = 0, a_1 = 0) \cdot 0.5 + \mathbb{P}(a_0 = 0, a_1 = 1) \cdot 1 && (6)\\ &\quad + \mathbb{P}(a_0 = 1, a_1 = 0) \cdot 1 + \mathbb{P}(a_0 = 1, a_1 = 1) \cdot 0.5 && (7)\\ &= 0.5\alpha^2 + 2\alpha(1-\alpha) + 0.5(1-\alpha)^2 && (8)\\ &= -\alpha^2 + \alpha + 0.5 && (9)\\ &= -(\alpha - 0.5)^2 + 0.75 && (10) \end{aligned}$$

Thus, $E[R(\pi)] \leq 0.75 < 1$ for all $\alpha \in [0, 1]$, with the maximum at $\alpha = 0.5$. Therefore, a shared policy cannot achieve the optimal return of 1, confirming the need for specialised behaviour to optimise rewards. □

F.4 Measuring Agent Gradient Conflict

We measure *gradient conflicts*, via the cosine similarity between agents' gradients $\cos\big(g_t^{(i)}, g_t^{(j)}\big) = \frac{\langle g_t^{(i)}, g_t^{(j)}\rangle}{\|g_t^{(i)}\|\|g_t^{(j)}\|}$, where $g_t^{(i)} = \nabla_\theta \mathcal{L}^{(i)}(\theta_t)$.

F.5 Empirical Results in N-player Specialisation and Synchronisation Normal-Form Game

To assess this limitations of FuPS and NoPS in practice, we compare three REINFORCE Williams (1992) variants in both games with $n = 2, 4, 8, 16, 32$ agents: NoPS (No Parameter Sharing), FuPS (Fully Parameter Sharing), and FuPS+ID (FuPS with one-hot agent IDs). All policies use single-layer neural networks with controlled parameter counts (see Appendix K for details).

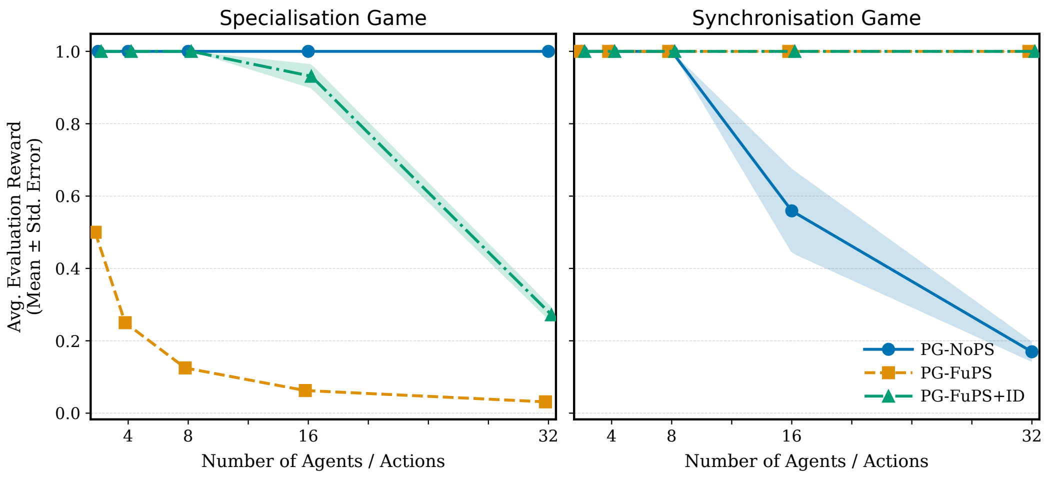


Figure 12: Mean evaluation reward (**mean ± standard error**) as a function of the number of agents/actions in the **Specialisation** (left) and **Synchronisation** (right) games. In the Specialisation game, vanilla policy gradients (PG, i.e. REINFORCE) with FuPS collapse as the team grows, whereas our identity-aware variant (PG-FuPS+ID) retains near-optimal performance. In the Synchronisation game, PG-NoPS performs well at small scales but degrades with more agents, while both PG-FuPS and PG-FuPS+ID remain at optimal reward across all scales.

Algorithm 1 HyperMARL

```
1:  Input: Number of agents $n$, number of training iterations $K$, MARL algorithm parameters (e.g., MAPPO-specific hyperparameters)
2:  Initialise:
3:      Hypernetwork parameters $\psi, \varphi$ {Ensure $\theta^i$ and $\phi^i$ follow standard initialization schemes, e.g., orthogonal}
4:      Agent embeddings $\{e^i\}_{i=1}^n$ {One-hot or orthogonally initialised learnable parameters}
5:  Output: Optimized joint policy $\boldsymbol{\pi}$
6:  for each training iteration $k = 0, 1, \ldots, K-1$ do
7:      for each agent $i = 1, \ldots, n$ do
8:          $\theta^i \leftarrow h_\psi^\pi(e^i)$ {Policy parameters}
9:          $\phi^i \leftarrow h_\varphi^V(e^i)$ {Critic parameters}
10:     end for
11:     Interact with environment using $\{\pi_{\theta^i}\}_{i=1}^n$ to collect trajectories $\mathcal{D}$
12:     Compute shared loss $\mathcal{L}$ from $\mathcal{D}$, using $\{V_{\phi^i}\}_{i=1}^n$ {Standard RL loss function}
13:     Update $\psi$, $\varphi$, and $\boldsymbol{e}$ by minimizing $\mathcal{L}$ {Optimise parameters.}
14: end for
15: Return $\boldsymbol{\pi} = (\pi^1, \ldots, \pi^n)$
```

G HyperMARL Details

G.1 HyperMARL Pseudocode

In Algorithm 1, we present the pseudocode for HyperMARL, with HyperMARL-specific steps highlighted in blue. HyperMARL leverages hypernetworks to dynamically generate the parameters of both actor and critic networks. The weights of the hypernetworks and the agent embeddings are automatically updated through automatic differentiation (autograd) based on the computed loss. Additionally, Figure 1 provides a visual representation of the HyperMARL architecture.

G.2 Variance of the HyperMARL Gradient Estimator

Unbiased estimator. Following from Equation 4, we can write the unbiased estimator for HyperMARLs gradient as follows:

$$\widehat{g}_{\mathrm{HM}} = \sum_{i=1}^{I} \underbrace{\nabla_\psi h_\psi^\pi(e^i)}_{\mathbf{J}_i} \cdot \underbrace{\left(\frac{1}{B} \sum_{b=1}^{B} \sum_{t=0}^{T-1} A(\mathbf{h}_t^{(b)}, \mathbf{a}_t^{(b)})\, \nabla_{\theta^i} \log \pi_{\theta^i}(a_t^{i,(b)} \mid h_t^{i,(b)}) \right)}_{\widehat{Z}_i} = \sum_{i=1}^{I} \mathbf{J}_i\, \widehat{Z}_i, \tag{HM'}$$

where B trajectories $\{\tau^{(b)}\}_{b=1}^B$ are sampled i.i.d. and $\widehat{Z}_i$ is the empirical analogue of the observation-conditioned factor.

Assumptions. (A1) trajectories are i.i.d.; (A2) all second moments are finite; (A3) ψ, θ, e^i are fixed during the backward pass.

Variance expansion. Since each $\mathbf{J}_i$ is deterministic under (A3), we may factor them outside the covariance:

$$\begin{aligned}
\mathrm{Var}\big(\widehat{g}_{\mathrm{HM}}\big) &= \mathrm{Cov}\Big(\sum_i \mathbf{J}_i \widehat{Z}_i,\ \sum_j \mathbf{J}_j \widehat{Z}_j\Big) && \text{(by def. } \mathrm{Var}(X) = \mathrm{Cov}(X, X)\text{)} \\
&= \sum_{i,j} \mathrm{Cov}\big(\mathbf{J}_i \widehat{Z}_i,\ \mathbf{J}_j \widehat{Z}_j\big) && \text{(bilinearity of Cov)} \\
&= \sum_{i,j} \mathbf{J}_i\ \mathrm{Cov}(\widehat{Z}_i, \widehat{Z}_j)\ \mathbf{J}_j^\top && \text{(pull deterministic matrices out of Cov)}
\end{aligned} \tag{11}$$

Equation (11) makes explicit that all trajectory-induced covariance is captured $\mathrm{Cov}(\widehat{Z}_i, \widehat{Z}_j)$, while the agent-conditioned Jacobians $\mathbf{J}_i$ remain trajectory noise-free.

Mini-batch update and covariance. Let $\widehat{Z}_i$ be the unbiased mini-batch estimate of Z_i and $\widehat{g}_{\mathrm{HM}} = \sum_i \mathbf{J}_i \widehat{Z}_i$ the stochastic update. Because every $\mathbf{J}_i$ is deterministic (wrt. to mini-batch),

$$\mathrm{Var}\big(\widehat{g}_{\mathrm{HM}}\big) = \sum_{i,j} \mathbf{J}_i\ \mathrm{Cov}(\widehat{Z}_i, \widehat{Z}_j)\ \mathbf{J}_j^\top, \tag{12}$$

(derivation in Appendix G.2). Equation (12) shows that HyperMARL first averages noise *within each agent* ($\widehat{Z}_i$) and only then applies $\mathbf{J}_i$. FuPS+ID, by contrast, updates the shared weights θ with every raw sample $A\,\nabla_\theta \log \pi_\theta[h, \mathrm{id}]$, leaving observation noise and agent ID entangled and making it susceptible to gradient interference (Christianos et al., 2021; Zhong et al., 2024).

G.3 Scalability and Parameter Efficiency

Hypernetworks generate weights for the target network, which can lead to high-dimensional outputs and many parameters for deep target networks. This challenge is amplified in MLP-based hypernetworks, which include additional hidden layers. Figure 13 shows scaling trends:

- **NoPS** and **linear hypernetworks**: Parameter count grows linearly with the number of agents.
- **FuPS**: More efficient, as growth depends on one-hot vector size.
- **MLP hypernetworks**: Scale better with larger populations, since they only require embeddings of fixed size for each new agent.

To reduce parameter count, techniques like shared hypernetworks, chunked hypernetworks (von Oswald et al., 2020; Chauhan et al., 2024), or producing low-rank weight approximations, can be used. While naive implementations are parameter-intensive, this might be less critical in RL and MARL which commonly have smaller actor-critic networks. Moreover, HyperMARL's near-constant scaling with agents suggests strong potential for large-scale MARL applications.

To isolate the effects of parameter count, we scaled the FuPS networks (Figure 14) to match the number of trainable parameters in HyperMARL. Despite generating 10x smaller networks, HyperMARL consistently outperforms FuPS variants, showing its advantages extend beyond parameter count.

G.4 Speed and Memory Usage

We examine the computational efficiency of HyperMARL compared to NoPS and FuPS by measuring inference speed (Figure 15a) and GPU memory usage (Figure 15b) as we scale the number of agents. The benchmarks were conducted using JAX on a single NVIDIA GPU (T4) with a recurrent (GRU-based) policy architecture. All experiments used fixed network sizes (64-dimensional embeddings and hidden layers) with a batch size of 128 and 64 parallel environments, allowing us to isolate the effects of varying agent count. Each measurement represents the average of 100 forward passes per configuration, with operations repeated across 10 independent trials.

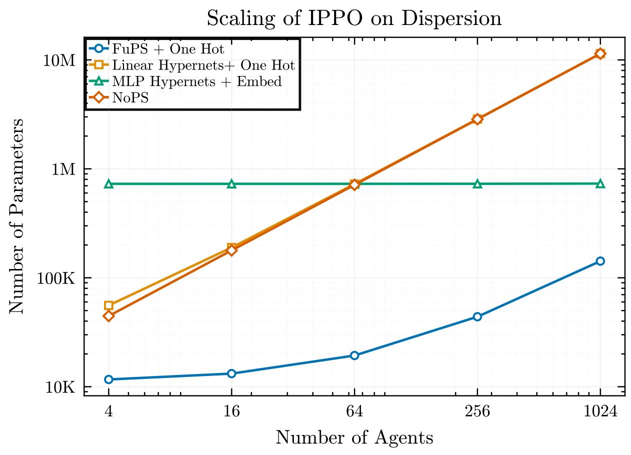


Figure 13: Parameter scaling for IPPO variants with increasing agents (4 to 1024). MLP Hypernetworks scale nearly constantly, while NoPS, Linear Hypernetworks, and FuPS+One-Hot grow linearly. Log scale on both axes.

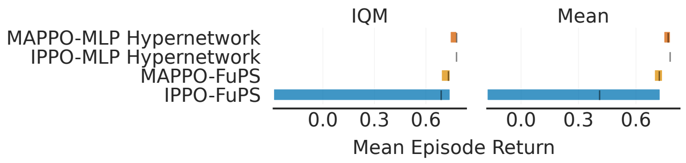
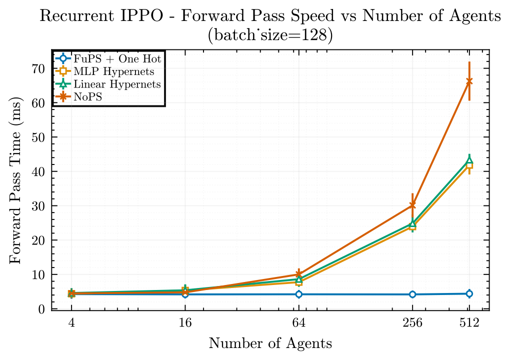
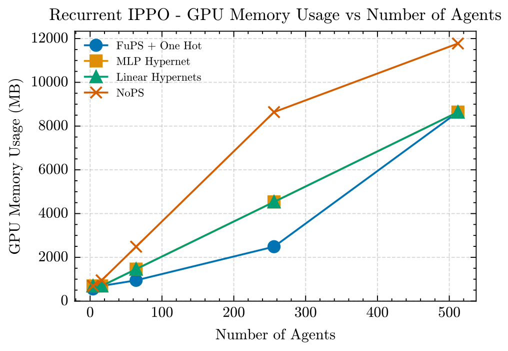


Figure 14: Dispersion performance with four agents. FuPS variants match HyperMARL in parameter count but still underperform.

The results demonstrate that HyperMARL offers a balance between the extremes represented by NoPS and FuPS. In practice, NoPS incurs additional data transfer and update costs, widening the efficiency gap.

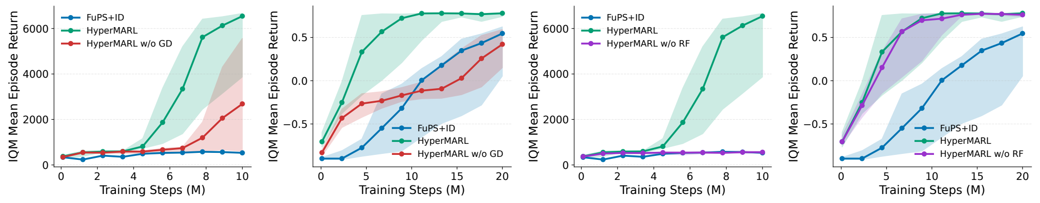


(a) Forward pass inference time



(b) GPU memory usage

Figure 15: Computational efficiency scaling with number of agents. HyperMARL offers a balance between NoPS and FuPS. Notably, in real-world deployments, NoPS incurs additional data transfer and synchronisation costs not reflected here, further widening the efficiency gap.



(a) Humanoid w/o GD (b) Dispersion w/o GD (c) Humanoid w/o RF (d) Dispersion w/o RF

Figure 16: *Ablation results comparing HyperMARL to variants without gradient decoupling (w/o GD) and without reset fan in/out initialisation (w/o RF) across environments.* Gradient decoupling (a,b) is consistently critical across both environments, while initialisation scaling (c,d) shows greater importance in the complex Humanoid task but less impact in the simpler Dispersion environment.

H Ablations

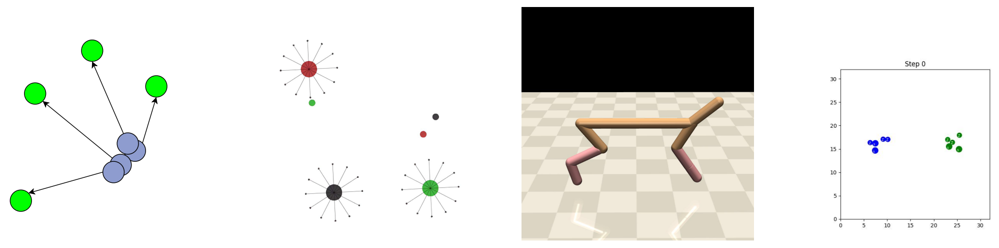
We ablate two critical components of HyperMARL, *gradient decoupling*(Sec. 4.2) and initialisation scaling (Sec. 4.1) on the complex Humanoid-v2 and simpler Dispersion tasks. Removing gradient decoupling (*HyperMARL w/o GD*), forcing joint observation/embedding processing, universally degrades performance (Figure 16). Omitting initialisation scaling (*HyperMARL w/o RF* (reset fan in/out), which aligns initial generated weights with conventional networks, reveals task-dependent effects: it is crucial for the 17-agent Humanoid, echoing prior findings on hypernetwork initialisation (Chang et al., 2020), but has minor impact in Dispersion. Thus, while principled initialisation becomes vital with increasing complexity, gradient decoupling is fundamentally essential across tested scenarios.

Table 3: MARL environments for evaluating *HyperMARL*.

Env.	**Agents**	**Action**	**Behaviour**
Dispersion	4	Discrete	Hetero.
Navigation	2, 4, 8	Continuous	Homo., Hetero., Mixed
MAMuJoCo	2–17	Continuous	Hetero.
SMAX	2–20	Discrete	Homo.

I Experiment Details

I.1 Environments



(a) *Dispersion*. (b) *Navigation*. (c) *MAMuJoCo*. (d) *SMAX*.

Figure 17: Multi-Agent environments used in our experiments.

Dispersion (VMAS) (Bettini et al., 2022): A 2D environment where four agents collect unique food particles. This task requires specialised *heterogeneous* behaviours and resembles the Specialisation Game from Section F.1.

Navigation (VMAS) (Bettini et al., 2022): Agents navigate in a 2D space to assigned goals, receiving dense rewards based on proximity. Configurations include shared goals (*homogeneous*), unique goals (*heterogeneous*), and *mixed* goals, where some agents share goals while others have unique ones.

Multi-Agent MuJoCo (MAMuJoCo) (Peng et al., 2021): A multi-agent extension of MuJoCo, where robot body parts (e.g., a cheetah's legs) are modelled as different agents. Agents coordinate to perform efficient motion, requiring *heterogeneous* policies (Zhong et al., 2024).

SMAX (JaxMARL) (Rutherford et al., 2024): Discrete action tasks with 2 to 20 agents on SMACv1- and SMACv2-style maps. FuPS baselines have been shown optimal for these settings Yu et al. (2022); Fu et al. (2022) indicating *homogeneous* behaviour is preferred here.

I.2 HyperMARL Architecture Details

For the Dispersion and Navigations results (Sec. 5.1) we use feedforward architectures, where we use HyperMARL to generate both the actor and critic networks. For the MAPPO experiments in Section 5.1, for fairness in comparisons with HAPPO and MAPPO, we maintain the centralised critic conditioned on the global state and only use HyperMARL to generate the weights of the actors. For the recurrent IPPO experiments in Section 5.2, HyperMARL only generates the actor and critic feedforward weights, not the GRU weights.

I.2.1 Training and Evaluation

- **Training:**
  - For Dispersion (5.1), we run 10 seeds and train for 20 million timesteps.

  - For Navigation (5.1), SMAX (5.2), and MaMuJoCo (5.1), we run 5 seeds and train for 10 million timesteps, consistent with the baselines.

- **Evaluation:**
  - For Dispersion (5.1), evaluation is performed every 100k timesteps across 32 episodes.
  - For Navigation (5.1), following the baselines, evaluation is performed every 120k timesteps across 200 episodes.
  - For SMAX (5.2), evaluation is performed every 500k timesteps across 32 episodes.
  - For MaMuJoCo (5.1), following the baselines, evaluation is performed every 25 training episodes over 40 episodes.

I.2.2 Measuring Policy Diversity Details

We measure team diversity using the System Neural Diversity (SND) metric (Equation 5 Bettini et al. (2023), details Section D) with Jensen-Shannon distance. SND ranges from 0 (identical policies across all agents) to 1 (maximum diversity). We collect a dataset of observations from IPPO-NoPS and IPPO-FuPS policies checkpointed at 5 and 20 million training steps. Each policy is rolled out for 10,000 episodes, generating 16 million observations. We then sample 1 million observations from this dataset to calculate the SND for each method tested.

Table 4: Baseline Methods Selection and Justification. Selected methods (✓) were chosen based on their relevance to parameter sharing and specialisation/generalisation in MARL, while excluded methods (✗) did not align with our research objectives or had implementation constraints. **Our experimental design systematically addresses key questions on agent specialisation and homogeneity, therefore we selected baselines with demonstrated strong performance in their respective settings, ensuring fair and rigorous comparison.**

Method	Category	Selected	Justification & Experimental Settings
IPPO (De Witt et al., 2020) (NoPS, FuPS+ID)	NoPS/FuPS	✓	Established MARL baseline implementing both independent (NoPS) and fully shared (FuPS+ID) policy configurations. *Tested in:* Dispersion, Navigation, SMAX (two SMACv1 maps and two SMACv2 maps, with 10 and 20 agents).
MAPPO (Yu et al., 2022) (NoPS, FuPS+ID)	NoPS/FuPS	✓	Strong baseline with centralized critics for both NoPS and FuPS+ID architectures. *Tested in:* Dispersion, MAMuJoCo, SMAX (two SMACv1 maps and two SMACv2 maps, with 10 and 20 agents).
DiCo (Bettini et al., 2024)	Architectural Diversity	✓	Provides comparison with a method employing preset diversity levels that balances shared and non-shared parameters. *Tested in:* Dispersion and Navigation (as per original paper). Original hyperparameters used for $n = 2$ agents; parameter sweeps conducted for $n > 2$ to identify optimal diversity levels.
HAPPO (Zhong et al., 2024)	Sequential Updates	✓	Enables comparison with a method designed for heterogeneous behaviours using sequential policy updates with agent-specific parameters. *Tested in:* MAMuJoCo, selecting 4/6 scenarios from the original paper, including the challenging 17-agent humanoid task. Walker and Hopper variants were excluded as MAPPO and HAPPO performed similarly in these environments.
Kaleidoscope (Li et al., 2024)	Architectural Pruning	✓	Implemented for off-policy evaluation using its MATD3 implementation with tuned MaMuJoCo hyperparameters. *Tested in:* MAMuJoCo environments Ant-v2, HalfCheetah-v2, Walker2d-v2 (overlapping with our IPPO experiments), and Swimmer-v2-10x2 (highest agent count variant). Included to evaluate HyperMARL's competitiveness against a method with ensemble critics and diversity loss, in an off-policy setting.
SEAC (Christianos et al., 2020)	Shared Experience	✗	Focuses primarily on experience sharing rather than parameter sharing architecture, falling outside our research scope.
SePS (Christianos et al., 2021)	Selective Parameter Sharing	✗	Requires pretraining phase, which extends beyond the scope of our current study focused on end-to-end learning approaches.
CDAS (Li et al., 2021)	Intrinsic Reward	✗	Only implemented for off-policy methods and has been demonstrated to underperform FuPS/NoPS architectures (Fu et al., 2022), making it less suitable for our primary on-policy comparisons.
ROMA/RODE (Wang et al., 2020a;b)	Role-based	✗	Shows limited practical performance advantages in comparative studies (Christianos et al., 2020), suggesting other baselines provide more rigorous comparison points.
SNP-PS (Kim & Sung, 2023)	Architectural Pruning	✗	No publicly available implementation, preventing direct, reproducible comparison.

Table 5: *Mean episode return in MAMuJoCo for off-policy MATD3 variants.* IQM of the mean episode returns with 95% stratified bootstrap CI. **Bold** indicates the highest IQM score; * indicates scores whose confidence intervals overlap with the highest. Although Kaleidoscope employs an ensemble of five critics and an explicit diversity loss, HyperMARL (using a standard MATD3 setup with two critics) achieved competitive results without these additional mechanisms.

Environment	Ind. Actors (MATD3)	HyperMARL (MATD3)	Kaleidoscope (MATD3)
Ant-v2	5270.38 (4329.73, 5719.78)	*5886.58 (5840.00, 5920.66)	**6160.70** (5798.02, 6463.83)
HalfCheetah-v2	*6777.04 (3169.11, 8233.94)	**7057.44** (3508.70, 8818.11)	*6901.00 (3609.73, 8192.38)
Walker2d-v2	*5771.87 (5144.84, 8103.34)	**7057.68** (5976.50, 8166.09)	*6664.32 (5408.95, 8828.11)
Swimmer-v2-10x2	*453.74 (427.24, 487.86)	**465.91** (410.82, 475.77)	*462.48 (444.22, 475.64)

J Detailed Results

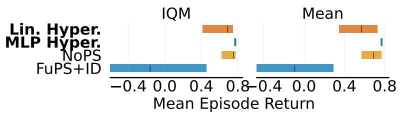
J.1 Kaleidoscope Off-Policy Comparison Details

Our comparison with Kaleidoscope (Li et al., 2024), mentioned in Section 5, is conducted using off-policy methods due to its original design. Kaleidoscope incorporates intricate mechanisms (e.g., learnable masks, an ensemble of five critics, a specific diversity loss) and numerous specialised hyperparameters (e.g., for critic ensembling: ‘critic_deque_len‘, ‘critic_div_coef‘, ‘reset_interval‘; for mask and threshold parameters: ‘n_masks‘, ‘threshold_init_scale‘, ‘threshold_init_bias‘, ‘weighted_masks‘, ‘sparsity_layer_weights‘, etc.). Porting this full complexity to an on-policy PPO backbone would constitute a significant research deviation rather than a direct benchmark of the established method.

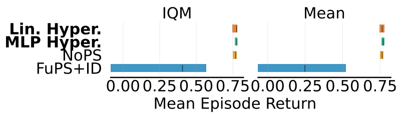
Therefore, we utilised Kaleidoscope's original off-policy implementation to ensure a meaningful comparison. We adopted MATD3 as the algorithmic backbone for this evaluation, as it was the only publicly available Kaleidoscope variant with tuned hyperparameters for Multi-Agent MuJoCo (MaMuJoCo). The MaMuJoCo tasks were chosen for alignment with our primary on-policy (IPPO) results and Kaleidoscope's original evaluation, specifically: Ant-v2, HalfCheetah-v2, Walker2d-v2 (overlapping with our IPPO experiments), and Swimmer-v2-10x2 (which represents the MaMuJoCo variant with the highest number of agents). Comparative results in Table 5 show that HyperMARL achieves competitive results with Kaleidoscope, while only using two critics (standard MATD3) and without additional diversity objectives.

J.2 Dispersion Detailed Results

J.2.1 Interval Esimates Dispersion



(a) IPPO



(b) MAPPO

Figure 18: *Performance of IPPO and MAPPO on Dispersion after 20 million timesteps.* We show the Interquartile Mean (IQM) of the Mean Episode Return and the 95% Stratified Bootstrap Confidence Intervals (CI) using Agarwal et al. (2021). Hypernetworks achieve comparable performance to NoPS, while FuPS struggle with specialisation.

J.3 Detailed MAMujoco Plots

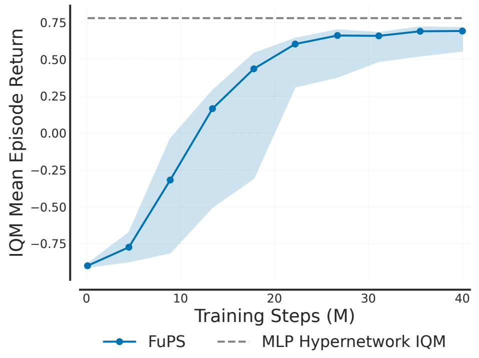
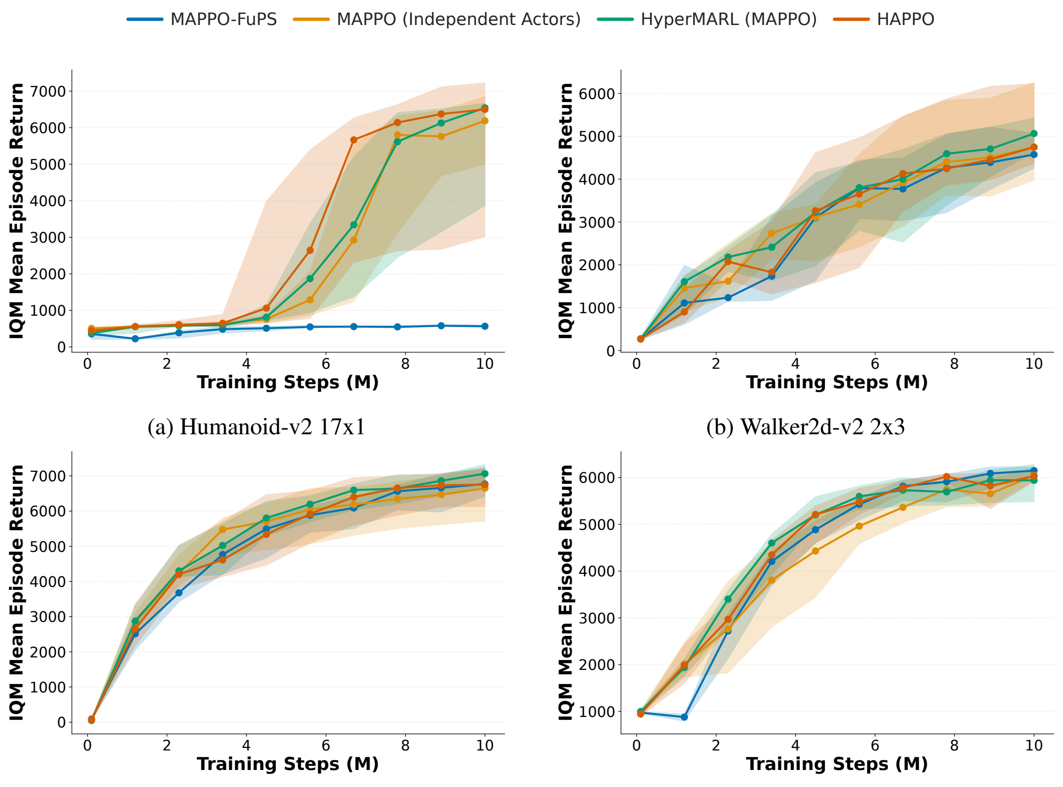


Figure 19: We see that even if we run MAPPO-FuPS on Dispersion for 40 million timesteps (double the timesteps of MLP Hypernetwork), it converges to suboptimal performance.



(a) Humanoid-v2 17x1

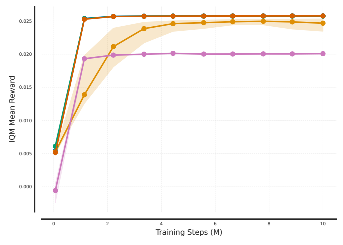
(b) Walker2d-v2 2x3

(c) HalfCheetah-v2 2x3

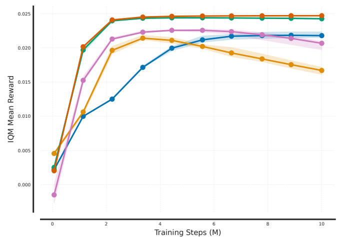
(d) Ant-v2 4x2

Figure 20: Performance of Recurrent IPPO and MAPPO on MaMoJoCo. HyperMARL performs comparably to these baselines, and is the only method with shared actors to demonstrate stable learning in the notoriously difficult 17-agent Humanoid environment.

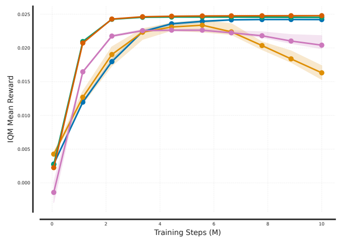
J.4 Detailed Navigation Plots



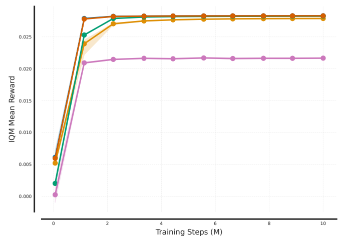
(a) 8 Agents, Same Goal



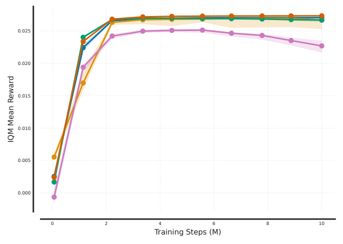
(b) 8 Agents, Different Goals



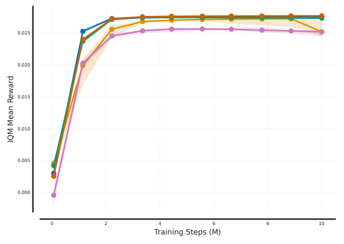
(c) 8 Agents, Four Goals



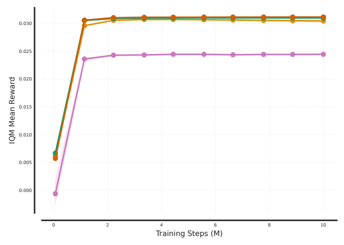
(d) 4 Agents, Same Goal



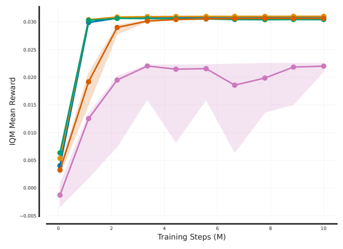
(e) 4 Agents, Different Goals



(f) 4 Agents, Two Goals



(g) 2 Agents, Same Goal



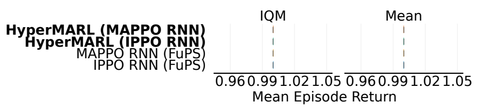
(h) 2 Agents, Different Goals

— FuPS — NoPS — MLP Hyper. — Lin. Hyper. — DiCo

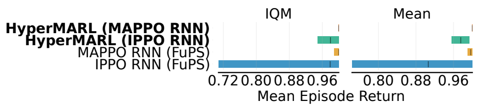
Figure 21: Sample efficiency of IPPO variants in the VMAS Navigation environment. Plots show IQM and 95% CI (shaded regions) of mean episode return against training steps for different agent counts (rows: 8, 4, 2 agents) and goal configurations (columns, where applicable: Same, Different, Specific Goal Counts). Legend shown at bottom applies to all subplots.

J.5 Interval Esimates - SMAX

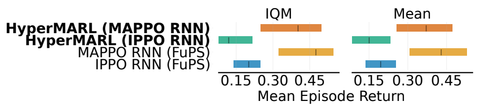
J.6 Additional Ablations



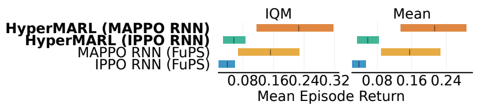
(a) 2s3z



(b) 3s5z



(c) SMACv2 10 Units



(d) SMACv2 20 Units

Figure 22: *Performance comparison in SMAX environments after 10 million timesteps.* We show the Interquartile Mean (IQM) of the Mean Win Rate and the 95% Stratified Bootstrap Confidence Intervals (CI). HyperMARL performs comparably to FuPS baselines across all environments, demonstrating its effectiveness in tasks requiring homogeneous behaviours and using recurrency.

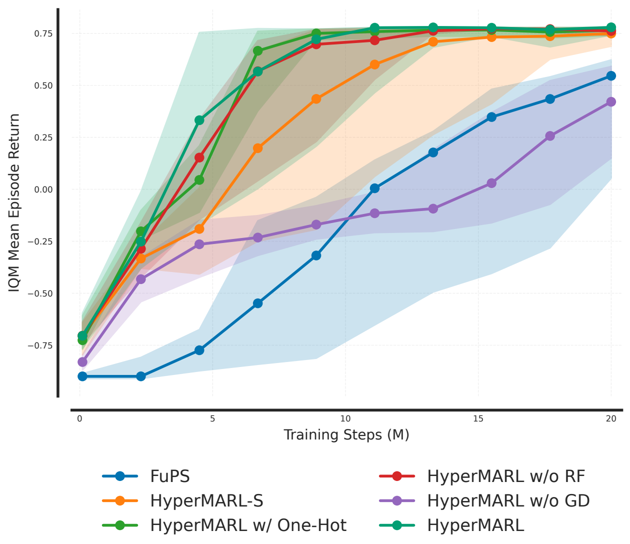


Figure 23: *Ablation results comparing HyperMARL with its variants in Dispersion.* The results highlight that gradient decoupling is essential for maintaining HyperMARL's performance.

K Hyperparameters

Table 6: Hyperparameters, Training and Evaluation for Specialisation and Synchronisation Game

Hyperparameter	**Value**
Environment Configuration	
Number of agents	2, 4, 8, 16
Maximum steps per episode	10
Training Protocol	
Number of seeds	10
Training steps	10,000
Evaluation episodes	100
Evaluation interval	1,000 steps
Batch size	32
Model Architecture	
Hidden layer size	8, 16, 32, 64
Activation function	ReLU
Output activation	Softmax
Optimization	
Learning rate	0.01
Optimizer	SGD
Model Parameter Count	
2 Agents	NoPS: 60
	FuPS: 58
	FuPS+ID: 74
	FuPS+ID (No State): 42
4 Agents	NoPS: 352
	FuPS: 240
	FuPS+ID: 404
	FuPS+ID (No State): 148
8 Agents	NoPS: 2400
	FuPS: 2344
	FuPS+ID: 2600
	FuPS+ID (No State): 552
16 Agents	NoPS: 17728
	FuPS: 17488
	FuPS+ID: 18512
	FuPS+ID (No State): 2128

Table 7: IPPO and MAPPO Hyperparameters in Dispersion

Hyperparameter	**Value**
LR	0.0005
GAMMA	0.99
VF_COEF	0.5
CLIP_EPS	0.2
ENT_COEF	0.01
NUM_ENVS	16
NUM_STEPS	128
GAE_LAMBDA	0.95
NUM_UPDATES	9765
EVAL_EPISODES	32
EVAL_INTERVAL	100000
MAX_GRAD_NORM	0.5
UPDATE_EPOCHS	4
NUM_MINIBATCHES	2
TOTAL_TIMESTEPS	20000000
ANNEAL_LR	false
ACTOR_LAYERS	[64, 64]
CRITIC_LAYERS	[64, 64]
ACTIVATION	relu
SEEDS	30,1,42,72858,2300658
ACTION_SPACE_TYPE	discrete

Table 8: MLP Hypernet Hyperparameters in Dispersion

Parameter	**IPPO**	**MAPPO**
HYPERNET_EMBEDDING_DIM	4	8
EMBEDDING_DIM Sweep	[4, 16, 64]	[4, 8, 16, 64]
HYPERNET_HIDDEN_DIMS	64	64

Table 9: Dispersion Settings

Setting	**Value**
n_food	4
n_agents	4
max_steps	100
food_radius	0.08
share_reward	false
penalise_by_time	true
continuous_actions	false

Table 10: IPPO Hyperparameters for Navigation

Hyperparameters	**Value**
LR	0.00005
NUM_ENVS	600
NUM_STEPS	100
TOTAL_TIMESTEPS	10^6
UPDATE_EPOCHS	45
NUM_MINIBATCHES	30
GAMMA	0.9
GAE_LAMBDA	0.9
CLIP_EPS	0.2
ENT_COEF	0.0
VF_COEF	1.0
MAX_GRAD_NORM	5
ACTIVATION	tanh
ANNEAL_LR	False
ACTOR_LAYERS	[256, 256]
CRITIC_LAYERS	[256, 256]
ACTION_SPACE_TYPE	continuous

Table 11: MLP Hypernet Hyperparameters in Navigation

Parameter	**IPPO**	**MAPPO**
HYPERNET_EMBEDDING_DIM	4	8
EMBEDDING_DIM Sweep	[4, 16, 64]	[4, 8, 16, 64]
HYPERNET_HIDDEN_DIMS	64	64

Table 12: DiCo Algorithm SND_des Hyperparameter

Goal Configuration	**Number of Agents**	**SND_des**
All agents same goal	2	0
	4	0
	8	0
All agents different goals	2	1.2 (From DiCo paper)
	4	[-1,1.2,2.4] ⇒ -1 (Best)
	8	[-1,1.2,4.8] ⇒ -1 (Best)
Some agents share goals	4	[-1,1.2] ⇒ -1 (Best)
	8	[-1,2.4,1.2] ⇒ -1 (Best)

Table 13: Parameter Sweeps for IPPO Variants in Navigation Tasks with Four and Eight Agents

Parameter Sweeps	
CLIP_EPS	0.2, 0.1
LR	5e-5, 5e-4, 2.5e-4

Algorithm	**Setting**	**Selected Values**
IPPO-FuPS	8 Agents (Same Goals)	0.2, 5e-5
	8 Agents (Different Goals)	0.1, 5e-5
	8 Agents (Four Goals)	0.1, 5e-5
	4 Agents (Same Goals)	0.2, 5e-5
	4 Agents (Different Goals)	0.2, 5e-5
	4 Agents (Two Goals)	0.2, 5e-5
IPPO-Linear Hypernetwork	8 Agents (Same Goals)	0.2, 5e-5
	8 Agents (Different Goals)	0.1, 5e-5
	8 Agents (Four Goals)	0.1, 5e-5
	4 Agents (Same Goals)	0.2, 5e-5
	4 Agents (Different Goals)	0.1, 5e-5
	4 Agents (Two Goals)	0.1, 5e-5
IPPO-MLP Hypernetwork	8 Agents (Same Goals)	0.2, 5e-5
	8 Agents (Different Goals)	0.1, 5e-5
	8 Agents (Four Goals)	0.1, 5e-5
	4 Agents (Same Goals)	0.1, 5e-5
	4 Agents (Different Goals)	0.1, 5e-5
	4 Agents (Two Goals)	0.1, 5e-5
IPPO-NoPS	8 Agents (Same Goals)	0.1, 5e-5
	8 Agents (Different Goals)	0.2, 5e-5
	8 Agents (Four Goals)	0.1, 5e-5
	4 Agents (Same Goals)	0.1, 5e-5
	4 Agents (Different Goals)	0.2, 5e-5
	4 Agents (Two Goals)	0.1, 5e-5
IPPO-Dico	8 Agents (Same Goals)	0.2, 5e-5
	8 Agents (Different Goals)	0.1, 2.5e-4
	8 Agents (Four Goals)	0.1, 2.5e-4
	4 Agents (Same Goals)	0.2, 5e-5
	4 Agents (Different Goals)	0.1, 2.5e-4
	4 Agents (Two Goals)	0.1, 5e-4

Table 14: Environment Settings for Navigation Task

Parameter	**Value**
n_agents	2,4,8
agents_with_same_goal	1, n_agents/2, n_agents
max_steps	100
collisions	False
split_goals	False
observe_all_goals	True
shared_rew	False
lidar_range	0.35
agent_radius	0.1
continuous_actions	True

Table 15: Default algorithm and model hyperparameters for the Ant-v2-4x2 environment (from (Zhong et al., 2024)).

Parameter	**Value**
— Algorithm Parameters —	
action_aggregation	prod
actor_num_mini_batch	1
clip_param	0.1
critic_epoch	5
critic_num_mini_batch	1
entropy_coef	0
fixed_order	true
gae_lambda	0.95
gamma	0.99
huber_delta	10.0
max_grad_norm	10.0
ppo_epoch	5
share_param	false
use_clipped_value_loss	true
use_gae	true
use_huber_loss	true
use_max_grad_norm	true
use_policy_active_masks	true
value_loss_coef	1
— Model Parameters —	
activation_func	relu
critic_lr	0.0005
data_chunk_length	10
gain	0.01
hidden_sizes	[128, 128, 128]
initialization_method	orthogonal_
lr	0.0005
opti_eps	1e-05
recurrent_n	1
std_x_coef	1
std_y_coef	0.5
use_feature_normalization	true
use_naive_recurrent_policy	false
use_recurrent_policy	false
weight_decay	0

Table 16: Default algorithm and model hyperparameters for the Humanoid-v2-17x1 environment (from (Zhong et al., 2024)).

Parameter	**Value**
— Algorithm Parameters —	
action_aggregation	prod
actor_num_mini_batch	1
clip_param	0.1
critic_epoch	5
critic_num_mini_batch	1
entropy_coef	0
fixed_order	false
gae_lambda	0.95
gamma	0.99
huber_delta	10.0
max_grad_norm	10.0
ppo_epoch	5
share_param	false
use_clipped_value_loss	true
use_gae	true
use_huber_loss	true
use_max_grad_norm	true
use_policy_active_masks	true
value_loss_coef	1
— Model Parameters —	
activation_func	relu
critic_lr	0.0005
data_chunk_length	10
gain	0.01
hidden_sizes	[128, 128, 128]
initialization_method	orthogonal_
lr	0.0005
opti_eps	1e-05
recurrent_n	1
std_x_coef	1
std_y_coef	0.5
use_feature_normalization	true
use_naive_recurrent_policy	false
use_recurrent_policy	false
weight_decay	0

Table 17: Default algorithm and model hyperparameters for the Walker2d-v2-2x3 environment (from (Zhong et al., 2024)).

Parameter	**Value**
— Algorithm Parameters —	
action_aggregation	prod
actor_num_mini_batch	2
clip_param	0.05
critic_epoch	5
critic_num_mini_batch	2
entropy_coef	0
fixed_order	false
gae_lambda	0.95
gamma	0.99
huber_delta	10.0
max_grad_norm	10.0
ppo_epoch	5
share_param	false
use_clipped_value_loss	true
use_gae	true
use_huber_loss	true
use_max_grad_norm	true
use_policy_active_masks	true
value_loss_coef	1
— Model Parameters —	
activation_func	relu
critic_lr	0.001
data_chunk_length	10
gain	0.01
hidden_sizes	128, 128, 128
initialization_method	orthogonal_
lr	0.001
opti_eps	1e-05
recurrent_n	1
std_x_coef	1
std_y_coef	0.5
use_feature_normalization	true
use_naive_recurrent_policy	false
use_recurrent_policy	false
weight_decay	0

Table 18: Default algorithm and model hyperparameters for the HalfCheetah-v2-2x3 environment (from (Zhong et al., 2024)).

Parameter	**Value**
— Algorithm Parameters —	
action_aggregation	prod
actor_num_mini_batch	1
clip_param	0.05
critic_epoch	15
critic_num_mini_batch	1
entropy_coef	0.01
fixed_order	false
gae_lambda	0.95
gamma	0.99
huber_delta	10.0
max_grad_norm	10.0
ppo_epoch	15
share_param	false
use_clipped_value_loss	true
use_gae	true
use_huber_loss	true
use_max_grad_norm	true
use_policy_active_masks	true
value_loss_coef	1
— Model Parameters —	
activation_func	relu
critic_lr	0.0005
data_chunk_length	10
gain	0.01
hidden_sizes	128, 128, 128
initialization_method	orthogonal_
lr	0.0005
opti_eps	1e-05
recurrent_n	1
std_x_coef	1
std_y_coef	0.5
use_feature_normalization	true
use_naive_recurrent_policy	false
use_recurrent_policy	false
weight_decay	0

Table 19: HyperMARL Hyperparameters Across MaMuJoCo Environments

Parameter	**Humanoid v2-17x1**	**Walker2d v2-2x3**	**HalfCheetah v2-2x3**	**Ant v2-4x2**	**Sweeps**
AGENT_ID_DIM	64	64	64	64	None
HNET_HIDDEN_DIMS	64	64	64	64	None
clip_param	0.075	0.0375	0.0375	0.075	[0.1,0.075,0.05,0.0375]

Table 20: Recurrent IPPO and MAPPO Hyperparameters in SMAX (from JaxMARL paper)

Hyperparameter	**IPPO Value**	**MAPPO Value**
LR	0.004	0.002
NUM_ENVS	128	128
NUM_STEPS	128	128
GRU_HIDDEN_DIM	128	128
FC_DIM_SIZE	128	128
TOTAL_TIMESTEPS	1e7	1e7
UPDATE_EPOCHS	4	4
NUM_MINIBATCHES	4	4
GAMMA	0.99	0.99
GAE_LAMBDA	0.95	0.95
CLIP_EPS	0.05	0.2
SCALE_CLIP_EPS	False	False
ENT_COEF	0.01	0.0
VF_COEF	0.5	0.5
MAX_GRAD_NORM	0.25	0.25
ACTIVATION	relu	relu
SEED	30,1,42,72858,2300658	30,1,42,72858,2300658
ANNEAL_LR	True	True
OBS_WITH_AGENT_ID	-	True

Table 21: Hyperparameter Sweeps and Final Values for Different Maps in SMAX. H- refers to HyperMARL MLP Hypernetworks.

Map	Algorithm	LR Range	Chosen LR	HNET Embedding Dim	HNET Hidden Dims
2s3z	IPPO	0.004	0.004	–	
	MAPPO	0.002	0.002	–	
	H-IPPO	0.004	0.004	4	32
	H-MAPPO	0.002	0.002	64	16
3s5z	IPPO	0.004	0.004	–	
	MAPPO	0.002, 0.005, 0.0003	0.002	–	
	H-IPPO	0.004	0.004	64	16
	H-MAPPO	0.002, 0.005, 0.0003	0.0003	64	16
smacv2_10_units	IPPO	0.005, 0.001, 0.0003, 0.004	0.001	–	
	MAPPO	0.002, 0.005, 0.0003	0.0003	–	
	H-IPPO	0.005, 0.001, 0.0003, 0.004	0.005	4	64
	H-MAPPO	0.002, 0.005, 0.0003	0.0003	64	16
smacv2_20_units	IPPO	0.002, 0.005, 0.0003	0.005	–	
	MAPPO	0.002, 0.005, 0.0003	0.0003	–	
	H-IPPO	0.002, 0.005, 0.0003	0.005	64	64
	H-MAPPO	0.002, 0.005, 0.0003	0.0003	4	64

Note: HNET Embedding Dim refers to the hypernetwork embedding dimension, chosen from the range {4, 16, 64}. HNET Hidden Dims refers to the hidden layer dimensions of the hypernetwork, chosen from the range {16, 32, 64}.

L Computational Resources

Table 22: Computational Resources by Experiment Type

Experiment Category	Hardware Configuration	Execution Time	Total Hours
Specialisation, Synchronisation & Dispersion	8 cores on AMD EPYC 7H12 64-Core Processor	2-6 hours per run (agent-count dependent)	250
Navigation Experiments	8 cores on AMD EPYC 7H12 64-Core Processor + NVIDIA RTX A4500	4-10 hours per run	320
MaMuJoCo Experiments	8 cores on AMD EPYC 7H12 64-Core Processor + NVIDIA RTX A4500	8-24 hours per run (scenario & algorithm dependent)	1,680
SMAX Experiments	8 cores on AMD EPYC 7H12 64-Core Processor + NVIDIA RTX A4500	2-5 hours per run	160

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Stefano V Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-agent reinforcement learning: Foundations and modern approaches*. MIT Press, 2024.

Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, and Shimon Whiteson. Hypernetworks in meta-reinforcement learning. In *Conference on Robot Learning*, pp. 1478–1487. PMLR, 2023.

Jacob Beck, Risto Vuorio, Zheng Xiong, and Shimon Whiteson. Recurrent hypernetworks are surprisingly strong in meta-rl. *Advances in Neural Information Processing Systems*, 36, 2024.

Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. Vmas: A vectorized multi-agent simulator for collective robot learning. *The 16th International Symposium on Distributed Autonomous Robotic Systems*, 2022.

Matteo Bettini, Ajay Shankar, and Amanda Prorok. System neural diversity: Measuring behavioral heterogeneity in multi-agent learning. *arXiv preprint arXiv:2305.02128*, 2023.

Matteo Bettini, Ryan Kortvelesy, and Amanda Prorok. Controlling behavioral diversity in multi-agent reinforcement learning. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=qQjUgItPq4.

Oscar Chang, Lampros Flokas, and Hod Lipson. Principled weight initialization for hypernetworks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1lma24tPB.

Vinod Kumar Chauhan, Jiandong Zhou, Ghadeer Ghosheh, Soheila Molaei, and David A Clifton. Dynamic inter-treatment information sharing for individualized treatment effects estimation. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 3529–3537. PMLR, 02–04 May 2024. URL https://proceedings.mlr.press/v238/chauhan24a.html.

Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In *34th Conference on Neural Information Processing Systems*, 2020.

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

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

Dominik Maria Endres and Johannes E Schindelin. A new metric for probability distributions. *IEEE Transactions on Information theory*, 49(7):1858–1860, 2003.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

Kevin Fu, Pierce Howell, Shalin Jain, and Harish Ravichandar. Learning flexible heterogeneous coordination with capability-aware shared hypernetworks. *arXiv preprint arXiv:2501.06058*, 2025.

Wei Fu, Chao Yu, Zelai Xu, Jiaqi Yang, and Yi Wu. Revisiting some common practices in cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 6863–6877. PMLR, 2022.

Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pp. 66–83. Springer, 2017.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Yizhou Huang, Kevin Xie, Homanga Bharadhwaj, and Florian Shkurti. Continual model-based reinforcement learning with hypernetworks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 799–805. IEEE, 2021.

Jiechuan Jiang and Zongqing Lu. The emergence of individuality. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4992–5001. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/jiang21g.html.

R Kassen. The experimental evolution of specialists, generalists, and the maintenance of diversity. *Journal of evolutionary biology*, 15(2):173–190, 2002.

Woojun Kim and Youngchul Sung. Parameter sharing with network pruning for scalable multi-agent deep reinforcement learning. *arXiv preprint arXiv:2303.00912*, 2023.

Jakub Grudzien Kuba, Muning Wen, Linghui Meng, Haifeng Zhang, David Mguni, Jun Wang, Yaodong Yang, et al. Settling the variance of multi-agent policy gradients. *Advances in Neural Information Processing Systems*, 34:13458–13470, 2021.

Chenghao Li, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:3991–4002, 2021.

Xinran Li, Ling Pan, and Jun Zhang. Kaleidoscope: Learnable masks for heterogeneous multi-agent reinforcement learning. *arXiv preprint arXiv:2410.08540*, 2024.

Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

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

Kevin R McKee, Joel Z Leibo, Charlie Beattie, and Richard Everett. Quantifying the effects of environment and population diversity in multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 36(1):21, 2022.

Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. Learning the pareto front with hypernetworks. *arXiv preprint arXiv:2010.04104*, 2020.

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

Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

Andrew Patterson, Samuel Neumann, Martha White, and Adam White. Empirical design in reinforcement learning. *Journal of Machine Learning Research*, 25(318):1–63, 2024.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34:12208–12221, 2021.

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.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson, Timon Willi, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, et al. Jaxmarl: Multi-agent rl environments and algorithms in jax. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 2444–2446, 2024.

Elad Sarafian, Shai Keynan, and Sarit Kraus. Recomposing the reinforcement learning building blocks with hypernetworks. In *International Conference on Machine Learning*, pp. 9301–9312. PMLR, 2021.

Chris R Smith, Amy L Toth, Andrew V Suarez, and Gene E Robinson. Genetic and genomic analyses of the division of labour in insect societies. *Nature Reviews Genetics*, 9(10):735–748, 2008.

James Surowiecki. *The Wisdom of Crowds*. Doubleday, New York, 2004. ISBN 9780385503860.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

Leonid Nisonovich Vaserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72, 1969.

Johannes von Oswald, Christian Henning, Benjamin F. Grewe, and João Sacramento. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgwNerKvB.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. Roma: multi-agent reinforcement learning with emergent roles. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9876–9886, 2020a.

Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. Rode: Learning roles to decompose multi-agent tasks. *arXiv preprint arXiv:2010.01523*, 2020b.

Katherine Y Williams and Charles A O'Reilly III. Demography and. *Research in organizational behavior*, 20:77–140, 1998.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Anita Williams Woolley, Ishani Aggarwal, and Thomas W Malone. Collective intelligence and group performance. *Current Directions in Psychological Science*, 24(6):420–424, 2015.

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.

Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 25(32): 1–67, 2024. URL http://jmlr.org/papers/v25/23-0488.html.