SKARL: KERNEL MEAN-FIELD REINFORCEMENT LEARNING FOR ZERO-SHOT POPULATION-SIZE TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Large-scale multi-agent reinforcement learning (MARL) requires policies that scale to many agents and tolerate population mismatch between training and deployment. Mean-field reinforcement learning (MFRL) provides a size-independent interface, but standard moment summaries discard distributional structure. We introduce Scalable Kernel MeAn-Field Multi-Agent Reinforcement Learning (SKARL), which embeds empirical populations by kernel mean embeddings and represents individual values by cylindrical kernel functionals. Under a characteristic kernel, we prove density of this function class. Under same-law sampling, we further bound the distance between the laws of empirical embeddings at sizes N and M by a term proportional to $\sqrt{|N-M|/(NM)}$, yielding the same rate for a fixed Lipschitz cylindrical functional. This result isolates representation sensitivity rather than end-to-end return preservation. For smooth kernels, we derive an RKHS Riesz gradient and its projection onto a fixed-size Nyström dictionary. Across variable-size MPE, MAgent2, and PettingZoo tasks, SKARL scales to large populations and transfers zero-shot. In MAgent2 pursuit, training with $N = 11$ and testing with up to $M = 500$ improves over first-moment MFRL by 15–17% return per agent; in combined arms, training with $N = 120$ retains positive mean return from $M = 18$ to $M = 450$, while all evaluated baselines are negative.

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) has achieved remarkable progress in domains such as multi-robot coordination (Vinyals et al., 2019; Berner et al., 2019). However, scaling MARL to large populations remains a fundamental challenge (Du et al., 2023). As the number of agents increases, the joint state–action space grows exponentially, and interaction dynamics become increasingly complex. This induces a curse of dimensionality that makes conventional learning unstable and inefficient (Tan, 1993; Tampuu et al., 2015). Equally important, many MARL policies are tied to the population size used during training: a policy trained with N agents can degrade sharply when deployed with a different number M of agents. This raises a central question: can we design a MARL algorithm whose zero-shot population-size transfer from N to M is not only empirically robust, but also theoretically quantifiable?

A promising direction is the use of mean-field approximations (Huang et al., 2006; Lasry & Lions, 2007). By summarizing agent interactions through a population distribution, mean-field MARL (MFRL) (Yang et al., 2018) avoids exponential complexity growth and exploits the permutation invariance of homogeneous swarms. Prior work has demonstrated the feasibility of mean-field methods in large-scale settings (Angiuli et al., 2021; Gu et al., 2025). However, designing a universally effective representation of the population distribution remains a bottleneck. Traditional distribution representations in mean-field settings mainly fall into two categories. The first employs spatial discretization techniques such as histograms (Carmona et al., 2019) and ϵ-nets (Gu et al., 2021), which preserve distributional information with theoretical guarantees but suffer from the curse of dimensionality in high-dimensional spaces, as the number of discrete units grows exponentially with the state dimension. The second relies on statistical moments, ranging from first-order means (Yang et al., 2018) to higher-order statistics (Pham & Warin, 2023). This paradigm adapts well to high-dimensional state spaces because moment calculations avoid explicit space partitioning, but

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.

its representational capacity is limited: distinct distributions can be conflated, and critical structural details such as multi-modality can be lost. As a result, current mean-field approaches still struggle to scale to sufficiently large populations.

In recent years, kernel-based methods have emerged as a third direction for distribution representation (Wang et al., 2020b; Liu et al., 2020; Cui et al., 2024; Fiedler et al., 2023; 2025), aiming to combine the scalability of moment-based methods with the expressiveness of discretization-based approaches. In mean-field systems, these methods use kernels to embed population distributions into high-dimensional feature spaces and transform distribution-level interactions into tractable feature operations (Wang et al., 2020b; Liu et al., 2020; Cui et al., 2024). However, most existing kernelized mean-field methods focus on representation or prediction at a fixed population scale. They do not directly answer the key deployment question in scalable MARL: how does the learned value functional change when the empirical population at training size N is replaced by another empirical population at deployment size M? Without such a transfer analysis, population flexibility remains primarily an empirical observation rather than a property that can be controlled.

To address this problem, we introduce Scalable Kernel MeAn-Field Multi-Agent Reinforcement Learning (SKARL), a mean-field MARL framework for population-size-robust representation. SKARL represents the local population around each agent as an empirical measure and embeds it with kernel mean embeddings (KME), which retain richer distributional information than low-order moments while remaining independent of the number of neighbors. We then model each individual Q-function as a cylindrical kernel functional of this embedded population, inspired by Guo et al. (2023). This function class is expressive over probability measures and admits explicit sensitivity bounds under empirical population sampling. The bound concerns a fixed value representation under a same-law sampling model. Population-dependent dynamics and optimization error remain outside its scope. For computational tractability, we derive the RKHS Riesz gradient under a smooth-kernel assumption and project that RKHS element onto a finite Nyström dictionary (Williams & Seeger, 2000). Empirically, SKARL achieves strong performance on large-scale cooperative tasks and maintains zero-shot performance when deployed with population sizes up to $45\times$ larger than those used during training.

In summary, our contributions are as follows:

- We propose SKARL, a kernelized mean-field MARL framework for variable-size multi-agent systems, using KME to obtain a size-agnostic population representation beyond low-order moments and fixed-size critics.
- We model individual Q-functions as cylindrical kernel functionals and prove density of the resulting function class over probability measures when the kernel is characteristic.
- We establish a coupling-aware population-size sensitivity bound: the distance between the laws of empirical KMEs at sizes N and M, and hence the expected gap of a fixed cylindrical value functional, is $O(\sqrt{|N-M|/(NM)})$.
- For smooth kernels, we derive an RKHS Riesz-gradient update that remains well defined for empirical Dirac measures and give its regularized projection onto a finite Nyström dictionary. With matched encoders and training loops, SKARL improves over first-moment MFRL by 15–17% in MAgent2 pursuit and transfers across combined-arms deployment sizes where the evaluated baselines have negative mean return.

2 Preliminaries

2.1 Multi-Agent Stochastic Game

We consider an episodic mean-field reinforcement learning game with a fixed number of agents $N \in \mathbb{N}$. Such a game is defined by the tuple $\langle \mathcal{S}^N, \mathcal{A}^N, P, (r^i)_{i=1}^N, \gamma \rangle$, where $\mathcal{S}^N = \mathcal{S}_1 \times \cdots \times \mathcal{S}_N$ denotes the joint state space: a vector $\boldsymbol{s} = (s^1, \ldots, s^N)$ collects the local state $s^i \in \mathcal{S}_i$ of each agent. Similarly, the joint action space is $\mathcal{A}^N = \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$, where a joint action $\boldsymbol{a} = (a^1, \ldots, a^N)$ consists of local actions $a^i \in \mathcal{A}_i$. In the homogeneous setting, agents share the same state and action spaces, i.e., $\mathcal{S} = \mathcal{S}_1 = \cdots = \mathcal{S}_N$ and $\mathcal{A} = \mathcal{A}_1 = \cdots = \mathcal{A}_N$. We assume $\mathcal{A} \subseteq \mathbb{R}^{d_a}$ is equipped with the standard Euclidean inner product, which is used in the distributional derivatives below. System dynamics are governed by a stochastic kernel $P : \mathcal{S}^N \times \mathcal{A}^N \to \mathcal{P}(\mathcal{S}^N)$, where

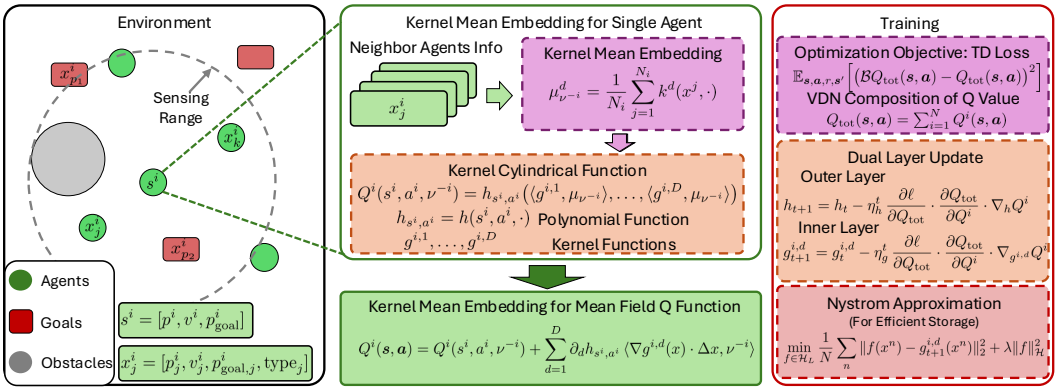


Figure 1: Overview of SKARL. Agent interactions are embedded by kernel mean embeddings and evaluated by cylindrical functions. Learning uses a fixed-size kernel dictionary; the RKHS-gradient variant projects its Riesz gradient onto this dictionary.

$\mathcal{P}(\mathcal{S}^N)$ denotes the set of probability measures over $\mathcal{S}^N$. Each agent receives an instantaneous reward $r^i(s,a) = r(s^i, a^i)$, which couples individual behavior with the global population. Finally, $0 < \gamma < 1$ is the discount factor. The objective is to learn a joint policy $\pi = (\pi^1, \ldots, \pi^N)$, where each $\pi^i : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, that maximizes for every agent i the expected discounted return $J^i(\pi) := \mathbb{E}_{s_0\sim d_0,\, P,\, \pi}[\sum_{t=0}^{T-1} \gamma^t r^i(s_t, a_t)]$. The expectation is taken over the initial state distribution d_0, the transition kernel P, and the stochastic choices of the joint policy π. We use the stochastic-game formulation because mean-field limits are derived from the N-player game as N grows; partial observability is handled through each agent's local neighborhood distribution rather than by maintaining an explicit Dec-POMDP belief state. Although the interaction is episodic, the discounted objective and our baselines use stationary policies, which is standard in deep MARL practice.

2.2 Mean Field Reinforcement Learning

In multi-agent reinforcement learning with N agents, the Q-function of agent i depends on the joint action $\boldsymbol{a} = (a^1, \ldots, a^N)$, where each a^j is represented by a one-hot vector. This leads to an exponential blow-up of the action space, a manifestation of the curse of dimensionality. Mean-field reinforcement learning (MFRL) (Yang et al., 2018) addresses this by approximating pairwise interactions through a mean-field term. Specifically, $Q^i(s, \boldsymbol{a}) = N_i^{-1} \sum_{j\in\mathcal{N}^i} Q^i(s, a^i, a^j) \approx Q^i(s, a^i, \bar{a}^{-i})$, where $\bar{a}^{-i} := \frac{1}{N_i}\sum_{j\in\mathcal{N}^i} a^j$ denotes the empirical mean action of agent i's neighbors $\mathcal{N}^i$ with size N_i. This induces a dynamical system in which each agent responds to the mean-field action via a softmax policy as $\pi^i_t(\cdot \mid s) = \text{softmax}\big(-\beta Q^i_t(s, \cdot, \bar{a}^{-i}_t)\big)$, where the softmax is taken over all $a \in \mathcal{A}$. For continuous action spaces, the mean-field action is modeled as a distribution on the 2-Wasserstein space $\mathcal{P}_2(\mathcal{A})$ (Guo & Xu, 2019) as $\nu^{-i} = \frac{1}{N_i}\sum_{j=1}^{N_i} \delta_{a^j}$, where δ_{a^j} is the Dirac measure at action a^j.

Lions derivative. Let $U : \mathcal{P}_2(\mathbb{R}^{d_a}) \to \mathbb{R}$. On an atomless probability space $(\Omega, \mathcal{F}, \mathbb{P})$, lift it to $L^2(\Omega; \mathbb{R}^{d_a})$ by $\widetilde{U}(X) := U(\mathcal{L}(X))$. We call U Lions differentiable at $\nu = \mathcal{L}(X)$ if $\widetilde{U}$ is Fréchet differentiable at X. Its Lions derivative is the measurable vector field $\partial_\nu U(\nu) : \mathbb{R}^{d_a} \to \mathbb{R}^{d_a}$, unique ν-a.e., satisfying $D\widetilde{U}(X)[H] = \mathbb{E}[\langle \partial_\nu U(\nu)(X), H\rangle]$ for every $H \in L^2(\Omega; \mathbb{R}^{d_a})$ (Guo et al., 2023; Carmona & Delarue, 2018). Thus, ∂_ν denotes the vector-valued lifted derivative, not the scalar linear functional derivative. When $\bar{Q}^i(s, a^i, \cdot)$ is twice Lions differentiable (Tang et al., 2024), the Lions–Taylor approximation is

$$Q^i(s, \boldsymbol{a}) \;\approx\; \bar{Q}^i(s, a^i, \nu^{-i}) + \frac{1}{N_i}\sum_{j=1}^{N_i} \big\langle \partial_\nu \bar{Q}^i(s, a^i, \nu^{-i})(a^j), \bar{a}^{-i} - a^j \big\rangle_{\mathcal{A}}, \tag{1}$$

where $\bar{Q}^i(s, a^i, \nu^{-i})$ is the Q-function lifted to the Wasserstein space, $\bar{a}^{-i} = \frac{1}{N_i}\sum_j a^j$ is the mean neighbor action, and $\partial_\nu \bar{Q}^i(s, a^i, \nu^{-i})(\cdot)$ is the vector-valued Lions derivative defined above.

2.3 RKHS AND KERNEL MEAN EMBEDDING IN MFRL

In recent years, kernel-based methods have been successfully integrated into MFRL to enhance the expressiveness and scalability of value functions and policies. A key tool for this integration is the RKHS, which provides a nonparametric framework for representing distributions over agent states (Muandet et al., 2017). An RKHS $\mathcal{H}_k$ over a domain $\mathcal{X}$ is a Hilbert space of functions $g : \mathcal{X} \to \mathbb{R}$ associated with a symmetric positive-definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Its defining reproducing identity is $g(x) = \langle g, k(x, \cdot)\rangle_{\mathcal{H}_k}$ for every $g \in \mathcal{H}_k$ and $x \in \mathcal{X}$ (Muandet et al., 2017). In the context of MFRL, the kernel mean embedding (KME) method is used to represent the distribution of agents in a high-dimensional feature space. For any distribution $\mathbb{P}$ over $\mathcal{X}$, its embedding is defined as $\mu_{\mathbb{P}}(\cdot) := \mathbb{E}_{X\sim\mathbb{P}}[k(\cdot, X)] = \int_{\mathcal{X}} k(\cdot, x), \mathrm{d}\mathbb{P}(x)$, which is a mapping from the distribution to an element of the RKHS. This embedding preserves expectations, such that for any function $g \in \mathcal{H}_k$, the expected value of $g(X)$ under the distribution $\mathbb{P}$ is given by the inner product in $\mathcal{H}_k$ as $\mathbb{E}_{X\sim\mathbb{P}}[g(X)] = \langle g, \mu_{\mathbb{P}}\rangle_{\mathcal{H}_k}$. This allows the representation of the entire mean-field distribution of agents as a single element in the RKHS, facilitating efficient computation and flexible modeling of the agent population's behavior.

3 THE SKARL FRAMEWORK

This section presents SKARL within the Reproducing Kernel Hilbert Space (RKHS), as shown in Figure 1. Our design goal is not only to obtain an expressive population representation, but also to analyze its sensitivity to empirical population size. Classical spatial discretization methods provide theoretical guarantees but suffer from the curse of dimensionality, as the number of cells explodes with the state dimension, making them unsuitable for large-scale RL (Carmona et al., 2019; Gu et al., 2021). Moment-based approaches alleviate this by summarizing populations via first-order means (Yang et al., 2018) or a few higher-order statistics (Pham & Warin, 2023), yet they discard fine-grained distributional structure and struggle to capture complex agent interactions. Kernel-based methods offer a middle ground, but a fixed kernel representation alone does not quantify how a value functional changes when its empirical population input changes size (Cui et al., 2023). Motivated by this gap, SKARL combines three components: (i) KME for size-agnostic population representation, (ii) cylindrical kernel functionals for value functions over distributions, and (iii) finite-dictionary learning, with an RKHS Riesz-gradient projection available for smooth kernels. The first two components yield the representation-level sensitivity bound in Theorem 4.1.

3.1 KERNEL MEAN EMBEDDING REPRESENTATION OF MEAN-FIELD Q-FUNCTIONS

Kernel Cylindrical Representation of Pairwise Interactions. Let $e_\psi : \mathcal{S} \times \mathcal{A} \to \mathcal{M} \subset \mathbb{R}^q$ be the learned encoder and set $x^j = e_\psi(s^j, a^j)$. The induced latent empirical measure is $\rho^{-i} := N_i^{-1} \sum_{j=1}^{N_i} \delta_{x^j}$, with KME $\mu_{\rho^{-i}} = N_i^{-1} \sum_{j=1}^{N_i} k(x^j, \cdot)$. This pushforward measure is distinct from the action measure ν^{-i} in Eq. (1). We model local Q-functions directly as functionals of ρ^{-i}. To obtain an expressive distributional model, we use *kernel cylindrical functions*, inspired by Guo et al. (2023). Formally, for a measure $\rho \in \mathcal{P}(\mathcal{M})$, a kernel cylindrical functional takes the form

$$\varphi(\rho) := \phi\big(\langle g^1, \mu_\rho\rangle_{\mathcal{H}_k}, \ldots, \langle g^D, \mu_\rho\rangle_{\mathcal{H}_k}\big), \tag{2}$$

where D is the number of kernel features, $d \in \{1, \ldots, D\}$ indexes a feature, $g^d \in \mathcal{H}_k$, μ_ρ is the KME of ρ, and $\phi : \mathbb{R}^D \to \mathbb{R}$ is an outer function with parameters θ_ϕ. The inner products $\langle g^d, \mu_\rho\rangle_{\mathcal{H}_k} = \int_{\mathcal{M}} g^d(x)\, \mathrm{d}\rho(x)$ are kernel-based summaries of the population. Theorem 3.1 states that the union of these finite-dimensional classes can uniformly approximate continuous functionals on probability measures.

Theorem 3.1 (Density of Kernel Cylindrical Functions)**.** *Let $\mathcal{M}$ be a compact metric space, $\mathcal{P}(\mathcal{M})$ carry the weak topology, and let k be a continuous characteristic kernel with RKHS $\mathcal{H}_k$. Define*

$$\mathcal{G}_D(\mathcal{M}) := \big\{\mu \mapsto \phi(c_1(\mu), \ldots, c_D(\mu)) : \phi \textit{ polynomial}, c_d(\mu) = \textstyle\int_{\mathcal{M}} g^d \,\mathrm{d}\mu,\ \ g^d \in \mathcal{H}_k\big\}. \tag{3}$$

Then $\mathcal{G}(\mathcal{M}) := \bigcup_{D\geq 1} \mathcal{G}_D(\mathcal{M})$ is dense in $C(\mathcal{P}(\mathcal{M}))$ under the uniform norm, namely, for every $f \in C(\mathcal{P}(\mathcal{M}))$ and $\epsilon > 0$, there exist a finite D and $h \in \mathcal{G}_D(\mathcal{M})$ such that $\sup_{\mu\in\mathcal{P}(\mathcal{M})} |f(\mu) - h(\mu)| < \epsilon$.

The characteristic-kernel condition is essential to ensure that RKHS test functions separate distinct probability measures. The result concerns the union over finite D and unrestricted RKHS test functions and it does not provide a rate or guarantee for a fixed pair (D, L), a particular neural outer function, or the outcome of optimization. Differentiability required by the RKHS-gradient update is imposed separately below and is not a consequence of uniform density. This directly yields a representation of the pairwise interaction in agent i's Q-function: $Q^i(s^i, a^i, \rho^{-i}) = h_{s^i,a^i}\big(\langle g^{i,1}, \mu_{\rho^{-i}}\rangle, \ldots, \langle g^{i,D}, \mu_{\rho^{-i}}\rangle\big)$, where $h_{s^i,a^i}(\cdot) = h(s^i, a^i, \cdot) : \mathbb{R}^D \to \mathbb{R}$ is differentiable with its parameters, and $g^{i,d} = \sum_{l=1}^{L} \alpha_l^d k(z^l, \cdot)$, with dictionary points $\{z^l\}_{l=1}^{L}$ and learnable weights $\{\alpha_l^d\}$. We use a shared dictionary in the notation; the implementation may use a separate size-L dictionary for each feature, with all formulas applied componentwise. The spatial gradient of $g^{i,d}$ is $\nabla g^{i,d}(x) = \sum_{l=1}^{L} \alpha_l^d \nabla_1 k(x, z^l)$. To control sensitivity and make derivative evaluation an RKHS operation, we impose the following conditions.

Assumption 3.2 (Bounded Kernel and Controlled Test Functions)**.** The latent domain $\mathcal{M} \subset \mathbb{R}^q$ is compact. The base kernel is twice continuously differentiable and satisfies $\sup_{x\in\mathcal{M}} k(x, x) \le \kappa^2$. Its first-derivative evaluation functionals are bounded: for every coordinate r, $\partial_{1,r}k(x, \cdot) \in \mathcal{H}_k$ and $\sup_{x,r} \|\partial_{1,r}k(x, \cdot)\|_{\mathcal{H}_k} \le \kappa_1$. Each learned test function satisfies $\|g^{i,d}\|_{\mathcal{H}_k} \le G_d$ and is L_d-Lipschitz: $|g^d(x) - g^d(y)| \le L_d\|x - y\|_2, \quad \forall x, y \in \mathcal{M}$. For the population summaries reached by the model, the outer function is L_ϕ-Lipschitz in its vector input. For Proposition 3.3, it is also twice continuously differentiable, with bounded first and second partial derivatives on the reached summary set. These bounds may be enforced in practice by norm regularization or projection; here they are explicit analytical assumptions.

For $H(\rho) := h(c_1(\rho), \ldots, c_D(\rho))$, the Lions chain rule on the latent space $\mathbb{R}^q$ (using the lifted definition from Section 2.2 with d_a replaced by q) gives (Guo et al., 2023) $\partial_\rho H(\rho)(x) = \sum_{d=1}^{D} \partial_d h(c(\rho))\, \nabla g^d(x)$, where $\partial_d h$ denotes the derivative with respect to the d-th Euclidean argument. This identity is used only for the latent functional H; it is not an identification with the action-space derivative in Eq. (1).

Local Value Function Approximation. Combining state-action embeddings, cylindrical functionals, and latent empirical measures yields a computational representation of the local Q-function. Define $c_d(\rho^{-i}) := \langle g^{i,d}, \mu_{\rho^{-i}}\rangle_{\mathcal{H}_k}$ and $A_d(\rho^{-i}) := \langle \nabla g^{i,d}(x) \cdot (\bar{x}^i - x), \rho^{-i}\rangle$, where $\bar{x}^i := \int_{\mathcal{M}} x \,\mathrm{d}\rho^{-i}(x) = N_i^{-1} \sum_j x^j$ for the empirical measure. Motivated by the first-order structure of Eq. (1), but defined directly on the latent measure rather than derived from the action-space Lions expansion, SKARL uses the model

$$Q^i_\Theta(\boldsymbol{s}, \boldsymbol{a}) = h_{s^i,a^i}\big(c_1(\rho^{-i}), \ldots, c_D(\rho^{-i})\big) + \sum_{d=1}^{D} \partial_d h_{s^i,a^i}\, A_d(\rho^{-i}), \tag{4}$$

where all derivatives of h are evaluated at $(c_1(\rho^{-i}), \ldots, c_D(\rho^{-i}))$ and Θ collects the encoder, outer-network, dictionary, and coefficient parameters. The first term captures distributional interactions, while the second is an explicitly defined latent first-order correction. No Taylor-remainder claim is made for the learned encoder map. This representation integrates seamlessly with standard multi-agent value-decomposition methods such as VDN (Sunehag et al., 2017), QMIX (Rashid et al., 2018), and QPLEX (Wang et al., 2020a). Analogous constructions apply to the state-value function $V^i(\boldsymbol{s})$ and advantage function $A^i(\boldsymbol{s}, \boldsymbol{a})$.

3.2 Value Function Update with Storage Efficiency

Finite-Parameter TD Learning and RKHS-Preconditioned Updates. The total value function Q_{tot} is decomposed into agent-wise functions Q^i under the Individual Global Max (IGM) principle (Rashid et al., 2018) (see Appendix F). To update Q^i, we optimize the temporal-difference (TD) loss (Sutton, 1988) $\ell(\mathcal{B}Q_{\text{tot}}, Q_{\text{tot}}) = \mathbb{E}_{\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}'}[\big(\mathcal{B}Q_{\text{tot}}(\boldsymbol{s}, \boldsymbol{a}) - Q_{\text{tot}}(\boldsymbol{s}, \boldsymbol{a})\big)^2]$, where $\mathcal{B}$ denotes the Bellman operator (Puterman, 2014), i.e., $(\mathcal{B}Q_{\text{tot}})(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\boldsymbol{s}'}\Big[r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \max_{\boldsymbol{a}'} Q_{\text{tot}}(\boldsymbol{s}', \boldsymbol{a}')\Big]$. The finite model can be trained end-to-end by differentiating through its coefficients and dictionary points. We additionally consider an RKHS-preconditioned coefficient update. Let $\xi_t^i := (\partial\ell/\partial Q_{\text{tot}})(\partial Q_{\text{tot}}/\partial Q^i)$, and let $\Gamma_t^{i,d}$ denote the RKHS Riesz gradient of Q^i with respect to $g^{i,d}$.

With θ collecting the encoder, outer-network, and dictionary parameters, the two updates are

$$\theta_{t+1} = \theta_t - \eta_\theta^t \nabla_\theta \ell, g_{t+1}^{i,d} = g_t^{i,d} - \eta_g^t \, \xi_t^i \Gamma_t^{i,d}. \tag{5}$$

The second line is realized by the finite-dictionary projection below; it never differentiates an empirical Dirac mass.

Proposition 3.3 (RKHS Riesz Gradient)**.** *Under Assumption 3.2, define* $R_{\rho^{-i}} := \int_{\mathcal{M}} \sum_{r=1}^{q} (\bar{x}_r^i - x_r) \, \partial_{1,r} k(x, \cdot) \, \mathrm{d}\rho^{-i}(x) \in \mathcal{H}_k$*. For the model in Eq.* (4)*, the Fréchet derivative with respect to* $g^{i,d} \in \mathcal{H}_k$ *is represented by the RKHS element*

$$\Gamma^{i,d} = \left(\partial_d h + \sum_{d'=1}^{D} \partial_{dd'}^2 h \, A_{d'}(\rho^{-i}) \right) \mu_{\rho^{-i}} + \partial_d h \, R_{\rho^{-i}}. \tag{6}$$

That is, $D_{g^{i,d}} Q^i[u] = \langle u, \Gamma^{i,d} \rangle_{\mathcal{H}_k}$ *for every* $u \in \mathcal{H}_k$*. In particular, if* ρ^{-i} *is empirical, both terms in Eq.* (6) *are finite sums of kernel sections or their first derivatives and therefore belong to* $\mathcal{H}_k$*.*

Finite-Dictionary Projection. A naive functional update adds $O(N_i)$ kernel or derivative sections at every iteration. To retain fixed storage, let $\mathcal{H}_L := \mathrm{span}\{k(z^l, \cdot)\}_{l=1}^{L}$ and approximate the Riesz gradient itself in this subspace. For $\lambda > 0$, define

$$\boldsymbol{\beta}_t^{i,d} := \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^L} \left\| \sum_{l=1}^{L} \beta_l k(z^l, \cdot) - \Gamma_t^{i,d} \right\|_{\mathcal{H}_k}^2 + \lambda \|\boldsymbol{\beta}\|_2^2. \tag{7}$$

Let $\boldsymbol{K}_{LL} := [k(z^l, z^{l'})]_{l,l'=1}^{L}$ and $\boldsymbol{b}_t^{i,d} := [\Gamma_t^{i,d}(z^l)]_{l=1}^{L}$. The normal equations give the unique ridge-regularized solution

$$\boldsymbol{\beta}_t^{i,d} = (\boldsymbol{K}_{LL} + \lambda \boldsymbol{I})^{-1} \boldsymbol{b}_t^{i,d}, \qquad \boldsymbol{\alpha}_{t+1}^{i,d} = \boldsymbol{\alpha}_t^{i,d} - \eta_g^t \xi_t^i \boldsymbol{\beta}_t^{i,d}. \tag{8}$$

For the empirical measure, let $\Delta x^j := \bar{x}^i - x^j$ and $C_d := \partial_d h + \sum_{d'=1}^{D} \partial_{dd'}^2 h \, A_{d'}(\rho^{-i})$. Each projection target then has the directly computable form

$$b_{t,l}^{i,d} = \Gamma_t^{i,d}(z^l) = \frac{C_d}{N_i} \sum_{j=1}^{N_i} k(x^j, z^l) + \frac{\partial_d h}{N_i} \sum_{j=1}^{N_i} (\Delta x^j)^\top \nabla_1 k(x^j, z^l). \tag{9}$$

Thus, the update uses only finite kernel and kernel-derivative evaluations; neither pointwise evaluation nor an RKHS inner product is applied to a generalized function. The dictionary cardinality remains L: coefficient storage is $O(DL)$, or $O(DL(q+1))$ including the feature-specific dictionary locations used in our implementation. Complete pseudocode is provided in Appendix A.

4 Analysis, Experiments and Results

4.1 Analysis of Computational Complexity, Scalability, and Population-Size Transfer

Let S and A denote local state and action feature dimensions and let q be the latent dimension. Table 1 reports only the extra representation-level cost, excluding the shared encoder, replay buffer, optimizer states, and environment simulation. For the feature-specific dictionaries used in our implementation, materializing all batchwise kernel tensors additionally requires $O(BDNL + DL^2)$ working memory; streaming over neighbors can reduce this workspace. The table is therefore an asymptotic accounting of the population representation, not a claim about total wall-clock or accelerator memory. QMIX and

Table 1: Representation-level asymptotic costs w/o shared-system costs.

Method	Input	Extra update	Persistent
SKARL	$O(S + A + D)$	$O(BDNL + DL^3)$	$O(DL(q+1))$
MFRL	$O(S + A)$	$O(BNq)$	$O(q)$
QMIX/QPLEX	local fixed	arch.-dependent	arch.-dependent

QPLEX depend on mixer width and centralized-state construction and are not assigned a misleading architecture-independent order.

Population-size sensitivity. We next compare the probability laws of empirical population embeddings at sizes N and M. A shared-sample coupling retains the common part of the two empirical populations and isolates the perturbation caused by adding or removing agents. This produces a cross term that depends explicitly on the distance between N and M.

Theorem 4.1 (Population-Size Coupling Bound for Empirical KMEs)**.** *Let $X_1, X_2, \ldots \overset{\text{i.i.d.}}{\sim} \nu$. For $n \geq 1$, let $\nu_n := n^{-1}\sum_{i=1}^n \delta_{X_i}$ and let $\widehat{\mu}_n := \mu_{\nu_n} = n^{-1}\sum_{i=1}^n k(X_i, \cdot) \in \mathcal{H}_k$ be its empirical KME; write $P_n := \mathcal{L}(\widehat{\mu}_n)$. Define the kernel variance by $\sigma_k^2(\nu) := \mathbb{E}\|k(X,\cdot) - \mu_\nu\|_{\mathcal{H}_k}^2$, equivalently $\sigma_k^2(\nu) = \mathbb{E}k(X,X) - \|\mu_\nu\|_{\mathcal{H}_k}^2$. Then the 2-Wasserstein distance between the laws of the empirical KMEs, computed using the Hilbert norm on $\mathcal{H}_k$, satisfies*

$$W_{2,\mathcal{H}_k}(P_N, P_M) \leq \sigma_k(\nu)\sqrt{\left|\frac{1}{N} - \frac{1}{M}\right|} = \sigma_k(\nu)\sqrt{\frac{|N-M|}{NM}}.$$

Corollary 4.2 (Expected Cylindrical-Value Gap)**.** *Let $G := \left(\sum_{d=1}^D \|g^d\|_{\mathcal{H}_k}^2\right)^{1/2}$, $h(\rho) = \phi\big(\langle g^1, \mu_\rho\rangle_{\mathcal{H}_k}, \ldots, \langle g^D, \mu_\rho\rangle_{\mathcal{H}_k}\big)$, and suppose that ϕ is L_ϕ-Lipschitz. Under the assumptions of Theorem 4.1,*

$$|\mathbb{E}h(\nu_N) - \mathbb{E}h(\nu_M)| \leq L_\phi G \sigma_k(\nu)\sqrt{\frac{|N-M|}{NM}}.$$

Under the shared-sample coupling used in the proof, the same upper bound holds for $\mathbb{E}|h(\nu_N) - h(\nu_M)|$.

Remark 4.3 (Effect of Population Overlap). If empirical populations of sizes N and M share exactly $r \leq \min(N, M)$ samples, the same second-moment calculation gives

$$\mathbb{E}\|\widehat{\mu}_N - \widehat{\mu}_M\|_{\mathcal{H}_k}^2 = \sigma_k^2(\nu)\left(\frac{1}{N} + \frac{1}{M} - \frac{2r}{NM}\right).$$

Independent populations correspond to $r = 0$, whereas maximal overlap $r = \min(N, M)$ recovers Theorem 4.1.

The theorem and corollary control the size-dependent perturbation of a fixed representation under a same-law sampling model. They do not control changes in transition dynamics, state visitation, population density, exploration, optimization error, or policy return. Agents in a finite interacting MARL system are generally dependent; extending the result to that setting requires an additional conditional-independence, mixing, or propagation-of-chaos assumption. The experiments below therefore test full-system transfer empirically rather than treating it as a corollary of the bound.

4.2 Experimental Setup

We evaluate our method following Nayak et al. (2023) on four particle environments: (i) **Move**: Each agent tries to move as fast as possible and avoid collisions. (ii) **Target**: Each agent tries to reach its assigned goal and avoid collisions. (iii) **Coverage**: Each agent tries to reach a goal and avoid collisions while ensuring that no more than one agent reaches the same goal. (iv) **Line**: There are two landmarks, and the agents try to position themselves equally spread out in a line between the two. To isolate the representation question at larger scale, we additionally evaluate on three fixed-local-observation benchmarks: **MAgent2 Adversarial-Pursuit**, where a homogeneous predator swarm must corner moving prey; **SISL Pursuit**, where pursuers must coordinate a two-agent surround to remove an evader; and **MAgent2 Combined-Arms** (Zheng et al., 2018), where a homogeneous melee swarm must coordinate in a chaotic mixed-arms battlefield. The main text follows a deliberate progression from the basic Move task, through fixed-observation Adversarial-Pursuit, to the more complex Combined-Arms battlefield. The complementary $N = 11$ Adversarial-Pursuit result and complete results for Target, Coverage, Line, and SISL Pursuit are reported in Appendix G. For detailed observation, reward, and action designs, see the Appendix Environments. We compare SKARL against: (i) **QMIX** (Rashid et al., 2018), (ii) **QPLEX** (Wang et al., 2020a), (iii) **MAPPO** (Yu et al., 2022), (iv) **MFRL** (Yang et al., 2018), and (v) **Fixed-Kernel MFRL** (Cui et al., 2023). The

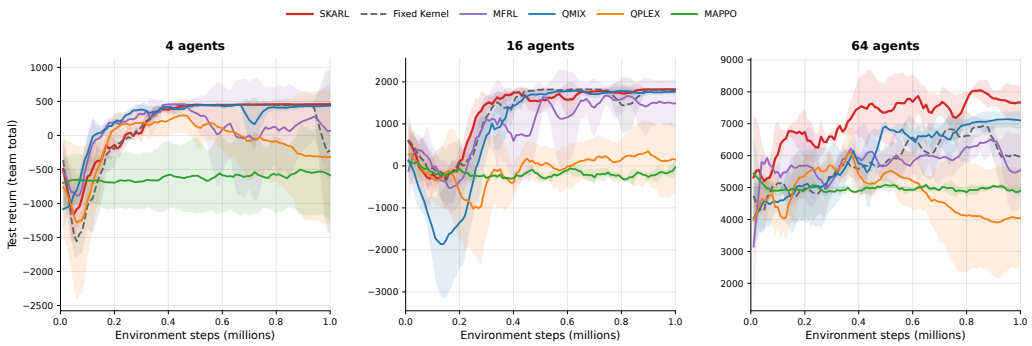


Figure 2: Move training curves at $N \in \{4, 16, 64\}$ agents (mean and uncertainty band across three random seeds).

Table 2: MAgent2 adversarial-pursuit zero-shot transfer from direct training at $N = 101$. Entries are return per deployed predator, mean $\pm$ 95% CI across five training seeds; each seed is evaluated for eight episodes without fine-tuning.

Algorithm	$M = 4$	$M = 11$	$M = 25$	$M = 45$	$M = 70$	$M = 101$	$M = 180$	$M = 281$	$M = 500$
MAPPO	7.11$\pm$ 19.75	7.71$\pm$ 21.41	7.42$\pm$ 20.59	7.01$\pm$ 19.45	6.91$\pm$ 19.17	6.96$\pm$ 19.33	6.84$\pm$ 18.98	7.00$\pm$ 19.42	6.85$\pm$ 19.00
QMIX	1.69$\pm$ 24.47	1.90$\pm$ 24.80	1.19$\pm$ 23.83	1.57$\pm$ 24.51	1.22$\pm$ 23.83	1.05$\pm$ 23.80	1.13$\pm$ 23.93	1.06$\pm$ 23.78	1.01$\pm$ 23.85
QPLEX	-7.83$\pm$ 16.08	-7.55$\pm$ 16.18	-7.71$\pm$ 16.19	-8.01$\pm$ 15.94	-7.94$\pm$ 15.97	-7.91$\pm$ 16.15	-8.06$\pm$ 15.82	-7.90$\pm$ 16.07	-8.06$\pm$ 15.93
MFRL	29.53$\pm$ 21.46	29.03$\pm$ 20.96	28.31$\pm$ 20.27	27.47$\pm$ 19.48	27.18$\pm$ 19.52	27.14$\pm$ 19.45	26.76$\pm$ 19.12	26.98$\pm$ 19.29	26.78$\pm$ 19.14
Fixed-Kernel	5.76$\pm$ 12.77	6.88$\pm$ 15.85	6.47$\pm$ 14.03	6.43$\pm$ 14.02	6.06$\pm$ 13.14	6.06$\pm$ 13.37	5.96$\pm$ 13.09	6.11$\pm$ 13.27	6.11$\pm$ 13.41
SKARL	**46.90**$\pm$ 7.83	**47.44**$\pm$ 3.61	**46.34**$\pm$ 1.44	**44.82**$\pm$ 2.28	**44.89**$\pm$ 2.31	**45.11**$\pm$ 2.73	**44.40**$\pm$ 2.48	**44.71**$\pm$ 2.46	**44.58**$\pm$ 2.46

value-based methods share the same local-view encoder, replay data, optimizer, and training budget. QMIX and QPLEX use a parameter-shared local Q-network and their respective centralized training mixers, while Fixed-Kernel freezes SKARL's dictionary points and expansion weights. At zero-shot deployment, QMIX and QPLEX discard the training-only mixer and select actions from the shared local network; consequently, changing M requires neither a size-specific mixer input nor test-time padding or masking. MAPPO uses a parameter-shared decentralized actor and a permutation-invariant centralized critic. The main tables report completed six-method comparisons; no missing method is encoded as a zero-valued observation. For detailed implementations of SKARL and the baselines, see Appendix F. We report test results with a maximum of 100 steps.[1].

4.3 Main Results

The main experiments progress from the simple Move task to fixed-observation MAgent2 Adversarial-Pursuit and finally to the more complex MAgent2 Combined-Arms battlefield. The Move learning curves in Figure 2 use three random seeds, while the tabular Move results use the original five-seed protocol. Adversarial-Pursuit uses five training seeds and eight evaluation episodes per seed. The Combined-Arms transfer study uses five recovered seeds for SKARL/MFRL/QMIX and eight seeds for Fixed-Kernel/QPLEX/MAPPO, as noted in Table 3. The complementary $N = 11$ Adversarial-Pursuit result, results for Target, Coverage, Line, and SISL Pursuit, and the ablations are provided in Appendix G.

Basic scalability in Move. Figure 2 and Table 8 summarize performance across swarm sizes $N \in \{4, 16, 64\}$. In the small swarm, SKARL remains competitive with the value-decomposition baselines. As the swarm grows, its advantage becomes clearer: at $N = 64$, SKARL obtains the highest reward while retaining low collision counts and competitive speed. The learning curves also show that this advantage emerges during training rather than only at the final checkpoint.

Zero-shot population-size transfer in Move. Table 9 evaluates full-system transfer: a policy is trained with $N \in \{16, 64, 256\}$ agents and tested with a different deployment size M, without retraining or architectural changes. Training at $N = 64$ or 256 improves transfer to large deployment populations relative to $N = 16$; in particular, SKARL trained with $N = 64$ maintains strong

[1]Code at https://anonymous.4open.science/r/SKARL-050D, based on JaxMARL (Rutherford et al., 2023)

Table 3: MAgent2 combined-arms zero-shot transfer from $N = 120$. Entries are return per deployed agent, mean ± 95% CI. SKARL/MFRL/QMIX use five recovered checkpoint seeds; Fixed-Kernel/QPLEX/MAPPO use eight seeds.

Algorithm	$M = 18$	$M = 45$	$M = 72$	$M = 120$	$M = 242$	$M = 450$
MAPPO	-0.54± 0.00	-0.54± 0.00	-0.54± 0.00	-0.54± 0.00	-0.54± 0.00	-0.54± 0.00
QMIX	-0.54± 0.00	-0.54± 0.00	-0.54± 0.00	-0.54± 0.00	-0.54± 0.00	-0.54± 0.00
QPLEX	-7.51± 3.62	-7.68± 3.70	-7.67± 3.70	-7.75± 3.73	-7.72± 3.72	-7.74± 3.73
MFRL	-0.55± 0.01	-0.54± 0.01	-0.54± 0.01	-0.54± 0.01	-0.54± 0.00	-0.54± 0.00
Fixed-Kernel	-0.54± 0.01	-0.54± 0.01	-0.54± 0.01	-0.54± 0.00	-0.54± 0.00	-0.54± 0.00
SKARL	**2.66**± 2.26	**2.91**± 2.49	**3.60**± 3.01	**3.35**± 2.87	**2.99**± 2.73	**2.16**± 2.11

per-agent reward at $M = 256$. Theorem 4.1 provides a representation-level notion of population-size separation through $\sqrt{|N - M|/(NM)}$, but does not predict policy return in an interacting environment. Table 9 should therefore be read as empirical evidence for the complete learning system rather than verification of a return guarantee.

Distributional representation in Adversarial-Pursuit. The MPE tasks demonstrate population flexibility, but their per-agent observations can still grow with the number of nearby agents, making them a weaker test of the population representation itself. We therefore use MAgent2 adversarial-pursuit to isolate the KME-vs-mean comparison with fixed-size local observations. Table 2 shows that direct training at $N = 101$ is substantially harder: SKARL is the only method with a consistently high mean across deployment sizes, retaining 44.58 ± 2.46 return per agent at $M = 500$. MFRL is bimodal, while Fixed-Kernel, QMIX, QPLEX, and MAPPO remain close to failure. As a complementary small-training-population test, Appendix Table 10 shows that SKARL trained with only $N = 11$ predators transfers zero-shot to $M = 500$, a $45\times$ extrapolation, and outperforms MFRL by 15–17% return per agent at every tested deployment size.

Complex large-scale coordination in Combined-Arms. Table 3 extends the study to MAgent2 Combined-Arms. When checkpoints trained at $N = 120$ are deployed zero-shot, SKARL retains positive mean return at every tested $M \in \{18, 45, 72, 120, 242, 450\}$, whereas all five baselines have negative mean return. This result sharpens the empirical scope of SKARL: the KME representation is most useful in homogeneous, cooperative, large-population tasks where the population distribution carries coordination signal; it is not expected to dominate reactive tasks or small swarms where the first moment is already sufficient.

4.4 Ablation Study

Full ablation results are provided in Appendix G. Table 20 shows that value-decomposition variants provide only marginal gains over base SKARL, while Figure 3 compares the RKHS-preconditioned and ordinary coefficient-space updates and finds improved stability and final performance for the former. Figure 4 shows that even $L = 1$ can attain full performance, with additional anchors mainly improving stability; Figure 5 finds similar final performance for Gaussian and polynomial kernels.

5 Conclusion

We introduced **SKARL**, a kernelized mean-field MARL framework for variable-size populations. SKARL combines kernel mean embeddings with cylindrical value functionals, for which we establish a density result and an explicit representation-level population-size sensitivity bound. For smooth kernels, its Riesz gradient is a valid RKHS element even for empirical measures and admits a fixed-dictionary projection. Across variable-size benchmarks, SKARL shows robust zero-shot transfer and improves over first-moment MFRL in large homogeneous coordination tasks. Its current scope is limited to homogeneous agents, and performance remains sensitive to exploration in some environments.

AI USE STATEMENT

In preparing this work, the authors used generative AI tools to assist with language and structural editing; drafting of the exposition; LaTeX formatting; and routine coding, debugging, and plotting tasks. AI outputs were treated as suggestions rather than as evidence or authoritative sources.

All reported experimental values were produced by the authors' experimental pipeline; generative AI was not used to fabricate or synthesize measurements or citations. The authors made the final decisions regarding the method, theoretical claims, experimental design, result inclusion, analysis, and presentation.

The authors reviewed and edited every AI-assisted contribution. Mathematical statements and derivations were independently checked by the authors, cited works were verified against their original sources, code changes were reviewed and tested, and figures and tables were checked against the underlying experiment logs. The authors take full responsibility for the final text, claims, code, and experimental artifacts.

REPRODUCIBILITY STATEMENT

Appendix A gives the complete SKARL training procedure. Appendix B states the assumptions used in the conditional optimization analysis, and Appendix C contains the proofs of the theoretical results. Environment definitions, model and baseline configurations, training procedures, evaluation protocols, and hyperparameters are documented in Appendix F; additional transfer results and ablations are reported in Appendix G. The anonymized repository at `https://anonymous.4open.science/r/SKARL-050D` provides the core JAX implementation, configuration files, and training and evaluation scripts.

REFERENCES

Andrea Angiuli, Jean-Pierre Fouque, and Mathieu Lauriere. Reinforcement learning for mean field games, with applications to economics. *arXiv preprint arXiv:2106.13755*, 2021.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019. URL `http://arxiv.org/abs/1912.06680`.

Vivek S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Springer, 2008.

René Carmona and François Delarue. *Probabilistic theory of mean field games with applications I-II*. Springer, 2018.

René Carmona, Mathieu Laurière, and Zongjun Tan. Linear-quadratic mean-field reinforcement learning: convergence of policy gradient methods. *arXiv preprint arXiv:1910.04295*, 2019.

Kai Cui, Sascha Hauck, Christian Fabian, and Heinz Koeppl. Learning decentralized partially observable mean field control for artificial collective behavior. *arXiv preprint arXiv:2307.06175*, 2023.

Kai Cui, Sascha Hauck, Christian Fabian, and Heinz Koeppl. Learning decentralized partially observable mean field control for artificial collective behavior. In *International Conference on Learning Representations*, volume 2024, pp. 40755–40794, 2024.

Ruan de Kock, Omayma Mahjoub, Sasha Abramowitz, Wiem Khlifi, Callum Rhys Tilbury, Claude Formanek, Andries Smit, and Arnu Pretorius. Mava: a research library for distributed multi-agent reinforcement learning in jax. *arXiv preprint arXiv:2107.01460*, 2021.

Yali Du, Joel Z Leibo, Usman Islam, Richard Willis, and Peter Sunehag. A review of cooperation in multi-agent learning. *arXiv preprint arXiv:2312.05162*, 2023.

Christian Fiedler, Michael Herty, and Sebastian Trimpe. On kernel-based statistical learning theory in the mean field limit. *Advances in Neural Information Processing Systems*, 36:20441–20468, 2023.

Christian Fiedler, Michael Herty, Chiara Segala, and Sebastian Trimpe. Recent kernel methods for interacting particle systems: first numerical results. *European Journal of Applied Mathematics*, 36 (2):464–489, 2025.

Haotian Gu, Xin Guo, Xiaoli Wei, and Renyuan Xu. Mean-field controls with q-learning for cooperative marl: convergence and complexity analysis. *SIAM Journal on Mathematics of Data Science*, 3(4):1168–1196, 2021.

Haotian Gu, Xin Guo, Xiaoli Wei, and Renyuan Xu. Mean-field multiagent reinforcement learning: A decentralized network approach. *Mathematics of Operations Research*, 50(1):506–536, 2025.

Xin Guo and Renyuan Xu. Stochastic games for fuel follower problem: N versus mean field game. *SIAM Journal on Control and Optimization*, 57(1):659–692, 2019.

Xin Guo, Huyên Pham, and Xiaoli Wei. Itô's formula for flows of measures on semimartingales. *Stochastic Processes and their applications*, 159:350–390, 2023.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Minyi Huang, Roland P Malhamé, and Peter E Caines. Large population stochastic dynamic games: closed-loop mckean-vlasov systems and the nash certainty equivalence principle. 2006.

Jean-Michel Lasry and Pierre-Louis Lions. Mean field games. *Japanese journal of mathematics*, 2 (1):229–260, 2007.

Siting Liu, Matthew Jacobs, Wuchen Li, Levon Nurbekyan, and Stanley J Osher. Computational methods for nonlocal mean field games with applications. *arXiv preprint arXiv:2004.12210*, 2020.

Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, Bernhard Schölkopf, et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017.

Siddharth Nayak, Kenneth Choi, Wenqi Ding, Sydney Dolan, Karthik Gopalakrishnan, and Hamsa Balakrishnan. Scalable multi-agent reinforcement learning through intelligent information aggregation. In *International Conference on Machine Learning*, pp. 25817–25833. PMLR, 2023.

Huyên Pham and Xavier Warin. Mean-field neural networks: learning mappings on wasserstein space. *Neural Networks*, 168:380–393, 2023.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *ICML*, 2018.

Alessandro Rudi, Raffaello Camoriano, and Lorenzo Rosasco. Less is more: Nyström computational regularization. *Advances in neural information processing systems*, 28, 2015.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Gardar Ingvarsson, Timon Willi, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert Tjarko Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktaschel, Chris Lu, and Jakob Nicolaus Foerster. Jaxmarl: Multi-agent rl environments in jax. *arXiv preprint arXiv:2311.10090*, 2023.

Friedhelm Schwenker, Hans A Kestler, and Günther Palm. Three learning phases for radial-basis-function networks. *Neural networks*, 14(4-5):439–458, 2001.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning. *CoRR*, abs/1706.05296, 2017. URL `http://arxiv.org/abs/1706.05296`.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3 (1):9–44, 1988.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *CoRR*, abs/1511.08779, 2015. URL http://arxiv.org/abs/1511.08779.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *In Proceedings of the Tenth International Conference on Machine Learning*, pp. 330–337. Morgan Kaufmann, 1993.

Huaze Tang, Yuanquan Hu, Fanfan Zhao, Junji Yan, Ting Dong, and Wenbo Ding. M^3arl: Moment-embedded mean-field multi-agent reinforcement learning for continuous action space. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7250–7254. IEEE, 2024.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyun Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nature*, pp. 1–5, 2019.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020a.

Lingxiao Wang, Zhuoran Yang, and Zhaoran Wang. Breaking the curse of many agents: Provable mean embedding q-iteration for mean-field reinforcement learning. In *International conference on machine learning*, pp. 10092–10103. PMLR, 2020b.

Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. *ICML*, 2018.

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.

Lianmin Zheng, Jiacheng Yang, Han Cai, Ming Zhou, Weinan Zhang, Jun Wang, and Yong Yu. MAgent: A many-agent reinforcement learning platform for artificial collective intelligence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

A FULL SKARL ALGORITHM

Algorithm 1 summarizes the complete training loop described in Section 3.

Algorithm 1 Mean-Field Cylindrical Kernel Method

Input: Agent count N, iterations K, batch size B, dictionary $\{z^l\}_{l=1}^{L}$, learning rates (η_θ, η_g), coefficient-update rule

Initialize the shared encoder and outer network, coefficients $\{\boldsymbol{\alpha}^{i,d}\}$, and replay buffer $\mathcal{T}$.
for $t = 1, \ldots, K$ **do**
  while Sampling phase **do**
    Sample trajectories in the environment using the current policy $\{\pi^i\}_{i=1}^{N}$ and store them in $\mathcal{T}$.
  end while
  Sample B trajectories from $\mathcal{T}$ with length T for each trajectory.
  Form ρ^{-i}, evaluate Eq. (4), and compute the TD loss.
  Update the encoder, outer network, and any learnable dictionary points by ordinary autodiff.
  if using the RKHS-preconditioned rule **then**
    Compute $\Gamma_t^{i,d}$ by Eq. (6), then $\boldsymbol{b}_t^{i,d}$ by Eq. (9).
    Solve Eq. (8) and update each coefficient vector $\boldsymbol{\alpha}_t^{i,d}$.
  else
    Update $\boldsymbol{\alpha}_t^{i,d}$ by ordinary coefficient-space autodiff.
  end if
end for
Return the shared local value model Q_Θ^i.

B ADDITIONAL OPTIMIZATION AND APPROXIMATION ANALYSIS

B.1 CONDITIONAL TWO-TIME-SCALE INTERPRETATION

Convergence of Updates. The RKHS-preconditioned rule in Eq. (5) can be interpreted as two-time-scale stochastic approximation. Because the practical value functions are nonlinearly parameterized, Bellman contraction in value space does not by itself imply global convergence of the parameter iterates. We therefore state the standard conditions under which this interpretation is valid (Borkar, 2008).

Assumption B.1 (Robbins–Monro)**.** Step sizes η_θ and η_g each satisfy $\sum_t \eta = \infty$ and $\sum_t \eta^2 < \infty$, with $\lim_{t\to\infty} \eta_g/\eta_\theta = 0$.

Proposition B.2 (Conditional Two-Time-Scale Convergence)**.** *Suppose Assumption B.1 holds, the iterates are almost surely bounded, the mean update fields are Lipschitz, and the stochastic errors are martingale differences with bounded conditional second moments. Further suppose that, for every fixed g, the limiting fast ODE for θ has a unique globally asymptotically stable equilibrium $\theta^*(g)$, and that the reduced slow ODE for g has a globally asymptotically stable equilibrium set $\mathcal{Z}$. Then the SKARL iterates converge almost surely to the corresponding equilibrium set $\{(\theta^*(g), g) : g \in \mathcal{Z}\}$.*

This proposition is conditional on stability of the limiting ODEs. It does not claim convergence to a global minimizer of the nonlinear TD objective or to the optimal joint action-value function.

B.2 ERROR OF FINITE-DICTIONARY GRADIENT APPROXIMATION

Let $\mathcal{P}_L$ be the orthogonal projection onto the RKHS subspace spanned by the L dictionary points, and let $\widetilde{\Gamma}_t^{i,d} := \sum_l \beta_{t,l}^{i,d} k(z^l, \cdot)$. The triangle inequality gives the deterministic decomposition

$$\|\Gamma_t^{i,d} - \widetilde{\Gamma}_t^{i,d}\|_{\mathcal{H}_k} \le \underbrace{\|(I - \mathcal{P}_L)\Gamma_t^{i,d}\|_{\mathcal{H}_k}}_{\text{subspace approximation}} + \underbrace{\|\mathcal{P}_L\Gamma_t^{i,d} - \widetilde{\Gamma}_t^{i,d}\|_{\mathcal{H}_k}}_{\text{ridge bias}}.$$

The first term depends on dictionary coverage and the kernel spectrum; the second is induced by λ. Standard Nyström rates may be invoked only when their sampling, effective-dimension, and source

assumptions are verified (Rudi et al., 2015). We do not identify the number of population samples with the number of dictionary points, and we do not claim that a supervised Nyström excess-risk rate automatically transfers to the nonlinear TD update.

C Proofs of Theorems, Lemmas and Propositions

C.1 Proof of Theorem 3.1

Proof. Because $\mathcal{M}$ is compact, $\mathcal{P}(\mathcal{M})$ is compact under the weak topology. The class $\mathcal{G}(\mathcal{M})$ contains the constants and is closed under addition and multiplication: sums and products of two finite-dimensional polynomial cylindrical functionals can be written as a polynomial cylindrical functional using the union of their test functions.

It remains to show that $\mathcal{G}(\mathcal{M})$ separates points. Let $\mu, \nu \in \mathcal{P}(\mathcal{M})$ with $\mu \neq \nu$. Since k is characteristic, their mean embeddings differ: $m_k(\mu) \neq m_k(\nu)$ in $\mathcal{H}_k$. Hence there exists $g \in \mathcal{H}_k$ such that

$$\langle g, m_k(\mu)\rangle_{\mathcal{H}_k} \neq \langle g, m_k(\nu)\rangle_{\mathcal{H}_k}.$$

Choosing $D = 1$ and the identity polynomial therefore gives a member of $\mathcal{G}(\mathcal{M})$ that separates μ and ν. The real Stone–Weierstrass theorem now implies that $\mathcal{G}(\mathcal{M})$ is dense in $C(\mathcal{P}(\mathcal{M}))$ under the uniform norm. □

C.2 Statement and Proof of Wasserstein Lipschitz Continuity

Lemma C.1 (Wasserstein Lipschitz Continuity)**.** *Let $h(\nu) = \phi(\int g^1 \,\mathrm{d}\nu, \ldots, \int g^D \,\mathrm{d}\nu)$, where ϕ is L_ϕ-Lipschitz and each g^d is L_d-Lipschitz. Then, for probability measures with finite second moments,*

$$|h(\nu_0) - h(\nu_1)| \leq L_\phi \left(\sum_{d=1}^{D} L_d^2\right)^{1/2} W_1(\nu_0, \nu_1) \leq L_\phi \left(\sum_{d=1}^{D} L_d^2\right)^{1/2} W_2(\nu_0, \nu_1).$$

Proof. For any coupling $\pi \in \Pi(\nu_0, \nu_1)$,

$$\left|\int g^d \,\mathrm{d}\nu_0 - \int g^d \,\mathrm{d}\nu_1\right| = \left|\int (g^d(x) - g^d(y)) \,\mathrm{d}\pi(x, y)\right| \leq L_d \int \|x - y\| \,\mathrm{d}\pi(x, y).$$

Applying the Lipschitz property of ϕ, summing the squared coordinate bounds, and taking the infimum over couplings gives the W_1 inequality. The W_2 inequality follows from $W_1 \leq W_2$ on probability measures with finite second moments. □

C.3 Proof of Proposition 3.3

Proof. Fix ρ^{-i} and abbreviate $\mu = \mu_{\rho^{-i}}$, $A_{d'} = A_{d'}(\rho^{-i})$, and $\Delta x = \bar{x}^i - x$. For any direction $u \in \mathcal{H}_k$, the reproducing property gives

$$Dc_d[g^{i,d}][u] = \int_{\mathcal{M}} u(x) \,\mathrm{d}\rho^{-i}(x) = \langle u, \mu\rangle_{\mathcal{H}_k}.$$

Because first-derivative evaluation is bounded by Assumption 3.2, differentiating under the integral and using the derivative reproducing identity yields

$$\begin{aligned} DA_d[g^{i,d}][u] &= \int_{\mathcal{M}} \nabla u(x) \cdot \Delta x \,\mathrm{d}\rho^{-i}(x) \\ &= \left\langle u, \int_{\mathcal{M}} \sum_{r=1}^{q} \Delta x_r \, \partial_{1,r} k(x, \cdot) \,\mathrm{d}\rho^{-i}(x) \right\rangle_{\mathcal{H}_k} \\ &= \langle u, R_{\rho^{-i}}\rangle_{\mathcal{H}_k}. \end{aligned}$$

Applying the ordinary chain and product rules to Eq. (4) therefore gives

$$D_{g^{i,d}} Q^i[u] = \left(\partial_d h + \sum_{d'=1}^{D} \partial^2_{dd'} h \, A_{d'} \right) \langle u, \mu \rangle_{\mathcal{H}_k} + \partial_d h \, \langle u, R_{\rho^{-i}} \rangle_{\mathcal{H}_k}$$
$$= \langle u, \Gamma^{i,d} \rangle_{\mathcal{H}_k}.$$

The integrand defining $R_{\rho^{-i}}$ is an $\mathcal{H}_k$-valued bounded continuous function on compact $\mathcal{M}$, so its Bochner integral belongs to $\mathcal{H}_k$. For an empirical measure the integral is simply a finite sum. The Riesz representation theorem now gives Eq. (6), without integration by parts or derivatives of Dirac masses. □

C.4 Proof of Theorem 4.1

Proof. Without loss of generality, let $M \geq N$ and construct the two empirical KMEs from the same infinite i.i.d. sequence. Write

$$Z_i := k(X_i, \cdot) - \mu_\nu, \qquad \mathbb{E} Z_i = 0, \qquad \mathbb{E}\|Z_i\|^2_{\mathcal{H}_k} = \sigma^2_k(\nu).$$

The difference between the two centered empirical KMEs is

$$\widehat{\mu}_N - \widehat{\mu}_M = \left(\frac{1}{N} - \frac{1}{M} \right) \sum_{i=1}^{N} Z_i - \frac{1}{M} \sum_{i=N+1}^{M} Z_i.$$

Independence and centering make all cross inner products vanish in expectation. Consequently,

$$\mathbb{E}\|\widehat{\mu}_N - \widehat{\mu}_M\|^2_{\mathcal{H}_k} = \sigma^2_k(\nu) \left[N \left(\frac{1}{N} - \frac{1}{M} \right)^2 + \frac{M-N}{M^2} \right]$$
$$= \sigma^2_k(\nu) \frac{M-N}{MN} = \sigma^2_k(\nu) \left| \frac{1}{N} - \frac{1}{M} \right|.$$

This shared-sample construction is a valid coupling of P_N and P_M. Since the squared 2-Wasserstein distance is the infimum of the expected squared Hilbert-space distance over all couplings,

$$W^2_{2,\mathcal{H}_k}(P_N, P_M) \leq \sigma^2_k(\nu) \left| \frac{1}{N} - \frac{1}{M} \right|.$$

Taking square roots proves the theorem. The case $N > M$ follows by symmetry. □

C.5 Proof of Corollary 4.2

Proof. Define $A : \mathcal{H}_k \to \mathbb{R}^D$ by

$$Au = (\langle g^1, u \rangle_{\mathcal{H}_k}, \ldots, \langle g^D, u \rangle_{\mathcal{H}_k}).$$

By Cauchy–Schwarz, $\|Au\|_2 \leq G\|u\|_{\mathcal{H}_k}$. Hence $F := \phi \circ A$ is $L_\phi G$-Lipschitz on $\mathcal{H}_k$. For any coupling π of P_N and P_M,

$$|\mathbb{E} F(\widehat{\mu}_N) - \mathbb{E} F(\widehat{\mu}_M)| \leq L_\phi G \, \mathbb{E}_\pi \|\widehat{\mu}_N - \widehat{\mu}_M\|_{\mathcal{H}_k}.$$

Taking the infimum over couplings, using $W_1 \leq W_2$, and applying Theorem 4.1 gives the expected-value bound. If π is instead fixed to the shared-sample coupling constructed in the theorem proof, Jensen's inequality applied to its exact second moment gives the stated bound on $\mathbb{E}_\pi |h(\nu_N) - h(\nu_M)|$. □

C.6 Proof of Proposition B.2

Proof. Write the two parameter updates as coupled stochastic-approximation recursions whose conditional mean fields are F_θ and F_g. Under the assumed martingale-noise, boundedness, and Robbins–Monro conditions, their continuous-time interpolation tracks the limiting fast and slow ODEs. The time-scale separation $\eta^t_g / \eta^t_\theta \to 0$ makes the fast θ recursion track $\theta^*(g)$ while g is quasi-static. The assumed global asymptotic stability of the fast ODE and of the reduced slow ODE then gives almost-sure convergence to the stated equilibrium set by the standard two-time-scale stochastic-approximation theorem (Borkar, 2008).

The ODE-stability assumptions are not consequences of Bellman contraction for a nonlinear parameterization. Accordingly, the proposition establishes conditional convergence to the stable equilibrium set of the expected update dynamics, not global minimization of the TD loss. □

Table 4: Kernel Functions and Corresponding Partial Derivative

Kernel Type	Kernel $k(y, x)$	Gradient of kernel $\partial_x k(y, x)$
Linear	$x \cdot y + c$	y
Polynomial	$(\alpha x \cdot y + c)^d$	$\alpha d(\alpha x \cdot y + c)^{d-1} y$
Gaussian	$\exp(-\gamma\|x - y\|^2)$	$-2\gamma(x - y)\exp(-\gamma\|x - y\|^2)$
Laplacian	$\exp(-\gamma\|x - y\|_1)$	$-\gamma \text{sign}(x - y)\exp(-\gamma\|x - y\|_1)$
Sigmoid	$\tanh(\alpha x \cdot y + c)$	$\alpha y(1 - \tanh^2(\alpha x \cdot y + c))$

D Appended Remarks

D.1 Remarks on Kernel Cylindrical Functions and Mean Field Embedding

Remark D.1 (Requirements on kernel by Lipschitz continuity). The Lipschitz continuity requirement limits the choice of kernel functions. Such as

- **Polynomial kernels**: $k(y, x) = (\alpha x \cdot y + c)^d$ violates the condition when input space $\mathcal{X}$ is unbounded, as the gradients grow polynomially with $\|x\|_2$.
- **Sigmoid kernels**: $k(y, x) = \tanh(\alpha x \cdot y + c)$ could fail to satisfy global Lipschitz continuity due to saturation effects in nonlinear regions.
- **Gaussian kernels**: $k(y, x) = \exp(-\gamma\|x - y\|_2^2)$ generally meet the requirement with $L_g = \gamma \sup_x \|x\|_2/2$.

Remark D.2 (Inner Product between mean-field measure and component functions). The inner product between mean field measure and component function $g^{i,d}$ evaluates to:

$$\langle g^{i,d}, \mu_{\rho^{-i}} \rangle = \frac{1}{N_i} \sum_{l=1}^{L} \sum_{j=1}^{N_i} \alpha_l^d k(x^j, z^l) = \frac{\mathbf{1}^\top \boldsymbol{K} \boldsymbol{\alpha}^d}{N_i}, \tag{10}$$

where $\boldsymbol{K} \in \mathbb{R}^{N_i \times L}$ has entries $\boldsymbol{K}_{jl} = k(x^j, z^l)$ and $\mathbf{1} \in \mathbb{R}^{N_i}$ is an all-ones vector.

E Remarks on Kernel Functions

We list several kernels frequently appearing in the literature.

In our experiments, we use polynomial and Gaussian kernels on bounded latent features. The density theorem applies only when the selected kernel is characteristic (for example, the Gaussian kernel); we do not claim that an arbitrary finite-degree polynomial kernel satisfies that hypothesis. The RKHS-gradient result additionally requires the smoothness conditions in Assumption 3.2, which exclude the nonsmooth Laplacian kernel as written.

E.1 Remarks on Mean-Field Representation of Value Functions

Remark E.1 (Expansion of Eq. (4)). Eq. 4 is expanded as:

$$\begin{aligned} Q^i(\boldsymbol{s}, \boldsymbol{a}) = h_{s^i, a^i} \left(\frac{\mathbf{1}^\top \boldsymbol{K} \boldsymbol{\alpha}^1}{N_i}, \ldots, \frac{\mathbf{1}^\top \boldsymbol{K} \boldsymbol{\alpha}^D}{N_i} \right) \\ + \frac{1}{N_i} \sum_{d=1}^{D} \partial_d h_{s^i, a^i} \sum_{l=1}^{L} \sum_{j=1}^{N_i} \alpha_l^d \nabla_1 k(x^j, z^l) \cdot (\bar{x}^i - x^j). \end{aligned}$$

Remark E.2 (Mean-field representation of state-value and advantage functions). The same latent-measure construction can parameterize $V^i(\boldsymbol{s})$ or an advantage $A^i(\boldsymbol{s}, \boldsymbol{a})$ by replacing the outer network and test functions in Eq. (4). Their population inputs remain ρ^{-i}, rather than the action-space measure ν^{-i}. We focus on the Q-function in this work.

Remark E.3 (Explicit form of Fréchet derivative). For $\rho^{-i} = N_i^{-1} \sum_{j=1}^{N_i} \delta_{x^j}$ and $\Delta x^j = \bar{x}^i - x^j$, set

$$C_d := \partial_d h + \sum_{d'=1}^{D} \partial^2_{dd'} h \, A_{d'}(\rho^{-i}).$$

Then Eq. (6) becomes the RKHS element

$$\Gamma^{i,d} = \frac{C_d}{N_i} \sum_{j=1}^{N_i} k(x^j, \cdot) + \frac{\partial_d h}{N_i} \sum_{j=1}^{N_i} \sum_{r=1}^{q} \Delta x_r^j \, \partial_{1,r} k(x^j, \cdot).$$

This finite-particle formula contains derivative kernel sections, not Dirac masses or their distributional derivatives.

E.2 Remarks on Finite-Dictionary Projection

Remark E.4. For each dictionary point z^l, the right-hand side of Eq. (8) is computed explicitly as

$$b_l = \Gamma^{i,d}(z^l) = \frac{C_d}{N_i} \sum_{j=1}^{N_i} k(x^j, z^l) + \frac{\partial_d h}{N_i} \sum_{j=1}^{N_i} (\Delta x^j)^\top \nabla_1 k(x^j, z^l). \tag{11}$$

This is ordinary kernel and kernel-derivative evaluation at finite points.

Remark E.5 (Dictionary Construction). Several principled strategies can initialize or refresh a fixed-cardinality dictionary $\{z^l\}_{l=1}^L$:

- **Random Subsampling**: Select L points uniformly from a finite replay-buffer candidate set $\mathcal{C}$.
  $$z^l \sim \text{Uniform}(\mathcal{C}), \quad l = 1, \ldots, L.$$
  Pros: $O(1)$ computational cost. *Cons*: May miss important regions.
- **k-means Centers**: Solve
  $$\{z^l\} = \arg\min_{\{c_l\}} \sum_{x \in \{x_m\}} \min_{1 \le l \le L} \|x - c_l\|^2.$$
  Pros: Captures data geometry. *Cons*: $O(N_i L T)$ computation complexity for T iterations.
- **Kernel Herding**: Select points maximizing the minimum kernel similarity:
  $$z_{l+1} = \arg\max_{x \in \{x_m\}} \sum_{l'=1}^{l} k(x, z_{l'}) - \frac{2}{N_i} \sum_{j=1}^{N_i} k(x, x^j).$$
  Pros: Constructs maximally representative points. *Cons*: $O(N_i L T)$ computation complexity for T iterations.
- **Leverage Score Sampling**: Sample with probability proportional to diagonal entries of the kernel matrix:
  $$p_j = \frac{(K_{MM})_{jj}}{\text{tr}(K_{MM})}, \quad z^l \sim p_j.$$
  Pros: Preserves spectral structure of the RKHS.

Our implementations keep the dictionary cardinality fixed; depending on the experiment, points are initialized by subsampling and then either learned by backpropagation or frozen for the Fixed-Kernel baseline.

F Implementation Details of SKARL and Baselines

F.1 Implementation Details of SKARL

Base Algorithm of Credit Assignment for SKARL We apply VDN (Sunehag et al., 2017) as the basic credit assignment algorithm for SKARL. Namely, the total Q_{tot} value is calculated by

$$Q_{\text{tot}}(\boldsymbol{s}, \boldsymbol{a}) = \sum_{i=1}^{N} Q^i(\boldsymbol{s}, \boldsymbol{a}).$$

Kernel Cylindrical Function Implementation We adopt a hypernetwork (Ha et al., 2016) for the outer cylindrical network. The ego state and action (s^i, a^i) condition this network, while the population branch processes the latent KME $\mu_{\rho^{-i}}$.

Coefficient Optimizers The finite-dictionary architecture admits two distinct optimizers, both shown in Algorithm 1. The Move implementation uses the RKHS-preconditioned coefficient update in Eqs. (6)–(8); the fixed-local-observation MAgent2 and SISL implementations differentiate the same finite expansion end-to-end in coefficient space. Proposition 3.3 justifies the former update and is not used to claim that ordinary coefficient-space autodiff is an identical optimizer. Figure 3 compares these choices directly.

Training Stabilization We apply several techniques to stabilize and accelerate training.

- **Dual Network Update**: To avoid over-estimation of Q value, we apply double Q learning framework (Van Hasselt et al., 2016).
- **Entropy Regularization**: To avoid the performance drops in the last epochs during training, we apply entropy regularization on the actor policy.

Codebase The MPE implementation is in JAX and organized around JaxMARL (Rutherford et al., 2023); the fixed-local-observation benchmarks use a PyTorch implementation. We plan to release the complete code after review. The current anonymous repository is available at `https://anonymous.4open.science/r/SKARL-050D`.

Hyperparameters In this paragraph, we list the hyperparameters in Table 5 and Table 6.

Table 5: Environment & Training Configuration

Environment		Training		Optimizer	
Hyperparameter	Value	Hyperparameter	Value	Hyperparameter	Value
Agent Number	4 / 16 / 64	Total Time Steps	2M	Learning Rate	7e-4
Environments Number	128	Update Steps Number	50	Max Grad Norm	10
Test Environment Number	8	Target Update Interval	8	Optimizer	ADAM
Max Train Env Timesteps	50	Test Interval	50k	EPS	1e-5
Max Test Env Timesteps	100			Weight Decay	0
Buffer		**Exploration**		**Learning rate Decay**	
Hyperparameter	Value	Hyperparameter	Value	Hyperparameter	Value
Buffer Size	8192	Epsilon	$1.0 \rightarrow 0.05$	η_θ	$1/t^{0.6}$
Buffer Batch Size	32	Epsilon Anneal Time	50k	η_g	$1/t^{0.8}$
Buffer Sample	Uniform	Anneal Method	Linear	Basic LR	7e-5

Table 6: Network & Algorithm Architecture

Network		Algorithm	
Hyperparameter	Value	Hyperparameter	Value
Embedding Net Layer	3	TD Lambda	0.95
Agent Hidden Dim	16	Gamma	0.99
Mixer Embedding Dim	256	Entropy Rate	0.5
Mixer Hypernet Hidden Dim	256	Anchor Points Number	$L = 64$
Attention Dim	64	Tikhonov Coefficient	0.5
Activation	ReLU	Polynomial Kernel	(α,d,c)=(1,2,1),(1,3,1)
FC Init Scale	2.0	Gaussian Kernel	γ = 0.5, 1.0

F.2 Common Settings for Environment

For learning stability and environment consistency, we use the following settings:

Environment Rescaling To make the environment scalable, we rescale the world size according to the number of agents as follows:

$$\text{world size} = \max(\sqrt{N} - 1, 1),$$

where world size is the half-width of the square boundary $[-\text{world size}, \text{world size}]^2$, and N denotes the number of agents. Thus the side length is 2 world size and the area grows as $O(N)$ for large N, approximately preserving initial spatial density. This environment rescaling is separate from the same-law sampling model in Theorem 4.1; the MPE transfer results therefore test changes in the full environment and policy system, not only empirical-KME sampling.

Resetting Agents and Landmarks We generate new agents and landmarks uniformly in the environment, namely, $p^i \sim \text{Uniform}([-\text{world size}, \text{world size}] \times [-\text{world size}, \text{world size}])$ for $i \in \{1, \ldots, N\}$. Some implementations adopt rejection sampling to avoid collisions between generated agents and landmarks, including the codebases of InforMARL (Nayak et al., 2023), JaxMARL (Rutherford et al., 2023), and Mava (de Kock et al., 2021). However, we do not use rejection sampling because of its computational cost. Instead, we divide the environment into grids and sample from the grids to avoid collisions.

During both training and evaluation phases in the **Target** and **Coverage** environments, the episode terminates and resets automatically once all agents successfully reach their assigned goals (or all landmarks are uniquely covered for the Coverage task). This design ensures episodic training and prevents infinite loops. However, since agents may receive one-time rewards multiple times, the total episodic reward may temporarily exceed the theoretical maximum (e.g., $N \times 10$ for N agents) during resets due to reward accumulation in the final timestep.

Size and Velocity Settings of Agents and Landmarks The settings for agents and landmarks are listed in Table 7.

Table 7: Environment Setup

Hyperparameter	Value
Agent Size	0.15
Landmark Size	0.225
Agent Maximum Speed	0.65 (Move) N/A (Target/Coverage)
Agent Acceleration	5 (Move) 2 (Target/Coverage)

G Additional Experiments

G.1 Detailed Move Results

Figure 2 in the main text reports the Move learning curves. Tables 8 and 9 provide the corresponding final performance and population-size transfer measurements.

Table 8: Performance Comparison between SKARL and Baselines in Move Environment

Algorithm	$N = 4$			$N = 16$			$N = 64$		
	R(↑)	# col(↓)	S(↑)	R(↑)	# col(↓)	S(↑)	R(↑)	# col(↓)	S(↑)
MAPPO	**947.6**	0.56± 0.174	**0.16**± 0.00779	3360.2	2.6± 1.12	0.14± 0.0562	14284.8	9.6± 6.98	**0.15**± 0.0459
QMIX	835.4	4.94± 4.94	0.15± 0.0431	2845.4	21.9± 12.7	0.13± 0.08517	10446.2	2.8± 1.00	0.11± 0.0732
QPLEX	911.4	0.56± 0.194	0.14± 0.0213	3625.8	20.5± 10.2	**0.17**± 0.0622	14073.8	22.5± 9.55	**0.15**± 0.0404
MFRL	734.6	**0**± 0	0.12± 0.0247	3083.69	38.4± 6.99	0.12± 0.0440	11411.1	204.2± 22.8	0.14± 0.0451
Fixed-Kernel	838.3	**0**± 0	0.14± 0.0143	3578.13	26.5± 3.52	0.15± 0.0172	13021.5	23.1± 10.8	0.15± 0.0722
SKARL	902.8	**0**± 0	0.15± 0.0192	**3755.9**	**12.32**± 5.847	**0.17**± 0.0500	**14423.8**	7.9± 5.37	**0.15**± 0.0334

Table 9: Zero-Shot Flexibility Performance of SKARL in Move Environment

Training	Metric	$M = 4$	$M = 8$	$M = 16$	$M = 32$	$M = 64$	$M = 128$	$M = 256$
$N = 16$	R/N	236.9	235.2	234.7	235.2	225.4	205.6	202.6
	(# col)/N	$0_{\pm 0}$	$0.98_{\pm 0.437}$	$0.77_{\pm 0.365}$	$0.57_{\pm 0.207}$	$0.17_{\pm 0.115}$	$0.12_{\pm 0.0652}$	$0.04_{\pm 0.0221}$
	S	$0.16_{\pm 0.00434}$	$0.17_{\pm 0.0612}$	$0.17_{\pm 0.0500}$	$0.17_{\pm 0.0469}$	$0.15_{\pm 0.0323}$	$0.14_{\pm 0.0521}$	$0.14_{\pm 0.0542}$
$N = 64$	R/N	231.5	221.3	227.3	224	223.2	221.6	218.7
	(# col)/N	$0_{\pm 0}$	$0.45_{\pm 0.408}$	$0.28_{\pm 0.257}$	$0.44_{\pm 0.275}$	$0.15_{\pm 0.109}$	$0.11_{\pm 0.0866}$	$0.09_{\pm 0.0476}$
	S	$0.15_{\pm 0.0126}$	$0.15_{\pm 0.0406}$	$0.15_{\pm 0.0237}$	$0.16_{\pm 0.0591}$	$0.15_{\pm 0.0459}$	$0.15_{\pm 0.0401}$	$0.15_{\pm 0.0436}$
$N = 256$	R/N	279.1	278.2	263.2	261.5	252.4	237.4	220.8
	(# col)/N	$0_{\pm 0}$	$0_{\pm 0}$	$0.14_{\pm 0.235}$	$0.28_{\pm 0.254}$	$0.17_{\pm 0.315}$	$0.13_{\pm 0.312}$	$0.10_{\pm 0.451}$
	S	$0.17_{\pm 0.0723}$	$0.17_{\pm 0.109}$	$0.16_{\pm 0.124}$	$0.16_{\pm 0.273}$	$0.16_{\pm 0.301}$	$0.16_{\pm 0.334}$	$0.15_{\pm 0.356}$

G.2 Additional Adversarial-Pursuit Transfer Result

Table 10 complements the main-text experiment trained at $N = 101$ by testing whether a policy learned from only $N = 11$ predators can extrapolate to much larger deployment populations.

Table 10: MAgent2 adversarial-pursuit zero-shot transfer from $N = 11$. Entries are return per deployed predator, mean $\pm$ 95% CI across five training seeds; each seed is evaluated for eight episodes without fine-tuning.

Algorithm	$M = 4$	$M = 11$	$M = 25$	$M = 45$	$M = 70$	$M = 101$	$M = 180$	$M = 281$	$M = 500$
SKARL	**52.48**$_{\pm 1.69}$	**51.20**$_{\pm 1.37}$	$49.78_{\pm 1.10}$	$48.43_{\pm 0.51}$	$48.30_{\pm 0.66}$	**48.43**$_{\pm 0.63}$	**48.04**$_{\pm 0.60}$	**48.25**$_{\pm 0.45}$	$47.87_{\pm 0.51}$
Fixed-Kernel	$43.83_{\pm 3.80}$	$40.85_{\pm 3.80}$	$38.22_{\pm 4.57}$	$37.33_{\pm 3.70}$	$36.48_{\pm 4.05}$	$36.60_{\pm 3.73}$	$36.10_{\pm 4.13}$	$36.37_{\pm 3.50}$	$35.85_{\pm 3.46}$
MFRL	$46.12_{\pm 4.20}$	$44.39_{\pm 2.46}$	$42.91_{\pm 1.98}$	$42.14_{\pm 2.07}$	$41.78_{\pm 2.99}$	$41.80_{\pm 2.62}$	$41.48_{\pm 3.06}$	$41.72_{\pm 3.11}$	$41.30_{\pm 3.29}$
QMIX	$52.09_{\pm 3.22}$	$49.00_{\pm 3.22}$	**50.30**$_{\pm 1.00}$	**48.64**$_{\pm 2.07}$	**48.62**$_{\pm 1.55}$	$48.26_{\pm 2.12}$	$47.98_{\pm 1.51}$	**48.25**$_{\pm 2.06}$	**47.88**$_{\pm 1.97}$
QPLEX	$8.91_{\pm 49.17}$	$8.32_{\pm 48.14}$	$7.36_{\pm 46.49}$	$6.93_{\pm 45.77}$	$6.77_{\pm 45.50}$	$6.79_{\pm 45.53}$	$6.74_{\pm 45.45}$	$6.73_{\pm 45.44}$	$6.56_{\pm 45.14}$
MAPPO	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$

G.3 Experiments in Target Environment

In this section, we provide the analysis of results for task **Target**. The experimental results in the Target environment demonstrate SKARL's ability to maintain task performance while balancing safety and scalability across different swarm sizes.

Table 11: Performance Comparison between SKARL and Baselines in Target Environment

Algorithm	$N = 4$				$N = 16$				$N = 64$			
	R(↑)	T(↓)	# col(↓)	S%(↑)	R(↑)	T(↓)	# col(↓)	S%(↑)	R(↑)	T(↓)	# col(↓)	S%(↑)
MAPPO	327.3	**0.14**	$2.6_{\pm 1.45}$	**100**	**12**	**0.56**	$8.93_{\pm 5.87}$	40.6	0	1.00	**0**	0
QPLEX	330.3	0.18	$1.3_{\pm 0.982}$	**100**	-3.1e4	1.00	**4.92**$_{\pm 2.55}$	0	-1.9e5	1.00	$19.8_{\pm 12.42}$	0
QMIX	337.0	**0.14**	**0.67**$_{\pm 0.35}$	**100**	-5.1e4	1.00	$6.6_{\pm 4.28}$	0	-6.4e5	1.00	$41.6_{\pm 18.1}$	0
MFRL	330.8	**0.14**	$6.4_{\pm 3.38}$	**100**	1.1	0.81	$14.375_{\pm 10.28}$	31.2	-5.7e5	1.00	$35.8_{\pm 15.1}$	0
Fixed-Kernel	**350.8**	**0.14**	$5.6_{\pm 2.54}$	**100**	4.2	0.18	$11.23_{\pm 8.12}$	80.2	10.2	**0.97**	$26.7_{\pm 8.2}$	1.2
SKARL	329.3	0.18	$7.2_{\pm 3.15}$	**100**	5.6	0.96	$23.2_{\pm 20.5}$	**100**	**44.75**	0.98	$44.3_{\pm 10.6}$	**3.1**

Table 12: Flexibility Performance of SKARL in Target Environment

Training	Metric	$M = 4$	$M = 8$	$M = 16$	$M = 32$	$M = 64$	$M = 128$	$M = 256$
$N = 4$	R/N	82.3	-36.25	-444.0	-2.8e3	-9.0e3	-1.8e4	-3.7e4
	T (step)	18	95	96.5	100	100	100	100
	(# col)/N	0.5$\pm$ 0.42	23$\pm$ 14.0	37.6$\pm$ 26.9	34.875$\pm$ 11.34	46$\pm$ 15.1	138$\pm$ 18.9	342$\pm$ 32.8
	S%	100	37.5	6.25	0	0	0	0
$N = 16$	R/N	85.3	7.5	0.35	-2.4e3	-8.3e3	-1.7e3	-2.6e4
	T (step)	17.4	13.8	96.3	98.5	99.4	100	100
	(# col)/N	0.4$\pm$ 0.13	19.25$\pm$ 13.0	23.2$\pm$ 20.5	34.875$\pm$ 11.34	46$\pm$ 15.1	75.5$\pm$ 14.6	116$\pm$ 21.1
	S%	100	100	100	75	6.25	0	0
$N = 64$	R/N	84.0	77.3	69.8	10.8	0.70	-0.25	-10.5
	T (step)	18.7	27.8	30.6	67.2	98.1	100	100
	(# col)/N	0.5$\pm$ 0.342	3$\pm$ 2.35	6.7$\pm$ 6.45	16.1$\pm$ 5.83	44.3$\pm$ 10.6	66.3$\pm$ 15.2	96.8$\pm$ 17
	S%	100	100	93.75	75	12.5	0	0
$N = 256$	R/N	87.2	80.5	78.6	22.9	20.8	15.4	12.1
	T (step)	18.6	22.2	28.5	30.4	32.8	45.1	60.2
	(# col)/N	0$\pm$ 0	0.5$\pm$ 0.412	1.4$\pm$ 0.24	5.7$\pm$ 1.32	11.2$\pm$ 5.2	12.1$\pm$ 9.4	14.2$\pm$ 10.5
	S%	100	100	100	100	100	98.71	95.21

For small swarms ($N = 4$), SKARL achieves near-optimal performance with a global reward of 329.3, comparable to QMIX (337.0) and QPLEX (330.3), while ensuring a 100% success rate. However, it exhibits a higher collision count (7.2 $\pm$ 3.15) compared to QMIX (0.67 $\pm$ 0.35) and QPLEX (1.3 $\pm$ 0.982), suggesting a trade-off between task completion and collision avoidance in simpler settings.

As the swarm scales to $N = 16$, SKARL significantly outperforms value-based methods (QMIX, QPLEX, MFRL), which suffer from catastrophic reward degradation (e.g., QPLEX: -3.1×10^4). Although MAPPO achieves a higher reward (12.0), its success rate drops to 40.6%, whereas SKARL maintains a 100% success rate despite increased collisions (23.2 $\pm$ 20.5). Additionally, SKARL reduces collisions by 32% compared to MFRL, indicating its robustness in mid-scale coordination, which aligns with findings from the Move environment in 8.

In large-scale swarms ($N = 64$), SKARL demonstrates superior scalability, achieving a positive reward (44.75) while all baselines fail (rewards ≤ 0). Notably, while the collision count remains high (44.3 $\pm$ 10.6), the drastic improvement in reward over MFRL (-5.7×10^5) and QMIX (-6.4×10^5) suggests that SKARL effectively prevents catastrophic failures in complex scenarios. The low success rate (3.1%) implies that further optimization is needed for very large swarms, but the results highlight SKARL's ability to maintain functional performance where other methods collapse.

Overall, SKARL exhibits strong scalability in the Target environment, particularly in maintaining task success and reward stability as the swarm size increases, with a trade-off in collision avoidance at larger scales. This aligns with its performance in the Move environment, where it achieves a 96% collision reduction at $N = 64$, reinforcing its effectiveness in large-scale multi-agent coordination. However, scaling up in the Target environment remains challenging and requires further work.

G.4 Experiments in SISL Pursuit

SISL Pursuit probes a coordination failure mode in which the reward requires a two-pursuer surround. With 200k training steps at $N = 32$, SKARL and Fixed-Kernel reach positive-return basins on three and four of five seeds, respectively, whereas MFRL succeeds on one seed and the remaining baselines do not solve the task. The wide confidence intervals in Table 13 reveal strong seed-level bimodality. In zero-shot transfer, SKARL has the highest sample mean at every tested deployment size, although its confidence intervals overlap substantially with those of Fixed-Kernel. Thus, these results show that kernel representations can discover transferable surround policies, but do not establish a statistically significant advantage for learning the kernel in this environment.

Table 13: SISL pursuit direct training at $N = 32$ for 200k steps from scratch. We report the mean of each seed's final five logged team returns, aggregated as mean ± 95% CI across five seeds, and the fraction of seeds with positive final return.

Algorithm	Final team return	Positive seeds
SKARL	**190.6**± 317.3	**4/5**
Fixed-Kernel	181.0± 438.0	3/5
MFRL	-77.5± 348.6	1/5
QMIX	-208.1± 99.2	0/5
QPLEX	-270.8± 39.0	0/5
MAPPO	-275.3± 25.5	0/5

Table 14: SISL pursuit zero-shot population-size transfer from the 200k checkpoints trained from scratch at $N = 32$. Entries are return per deployed pursuer, mean ± 95% CI across five training seeds and eight evaluation episodes per seed, with no fine-tuning. Fixed-Kernel has the highest sample mean at every deployment size, although its confidence intervals overlap substantially with those of SKARL because both methods exhibit seed-level bimodality.

Algorithm	$M = 16$	$M = 32$	$M = 64$	$M = 96$	$M = 128$	$M = 160$	$M = 192$	$M = 256$
SKARL	**6.41**± 9.00	**5.74**± 7.96	**4.31**± 5.98	**3.90**± 5.99	**2.46**± 5.25	**1.58**± 4.71	**1.24**± 4.04	**0.12**± 2.91
Fixed-Kernel	2.35± 12.52	1.76± 12.23	0.35± 10.95	-0.45± 9.15	-1.14± 9.26	-1.72± 8.06	-2.50± 7.95	-3.45± 6.94
MFRL	-5.43± 10.37	-5.72± 9.65	-6.12± 8.18	-6.29± 7.21	-6.64± 6.50	-6.96± 5.80	-6.89± 6.06	-7.14± 5.07
QMIX	-6.47± 4.36	-6.27± 4.96	-5.65± 4.93	-6.00± 3.93	-5.89± 3.59	-5.90± 4.04	-5.93± 3.68	-6.11± 3.29
QPLEX	-8.45± 1.94	-8.39± 1.63	-8.48± 1.20	-8.34± 1.77	-8.55± 1.28	-8.49± 1.56	-8.45± 1.60	-8.41± 1.79
MAPPO	-9.02± 0.50	-8.90± 0.59	-8.77± 0.68	-8.86± 0.61	-8.90± 0.59	-8.95± 0.56	-8.87± 0.61	-8.98± 0.51

Table 15: SISL pursuit zero-shot transfer under constant-density scaling. The completed 100k checkpoints trained at $N \in \{64, 128\}$ are deployed without fine-tuning. Entries are return per pursuer, mean ± 95% CI across five training seeds and eight evaluation episodes per seed.

Train N	Algorithm	$M = 32$	$M = 64$	$M = 128$	$M = 256$
64	SKARL	**2.65**± 7.10	**3.10**± 8.75	**1.71**± 7.79	**-0.17**± 6.54
	Fixed-Kernel	0.30± 9.82	-1.27± 7.87	-1.36± 8.05	-2.82± 6.46
	MFRL	-8.65± 0.66	-8.30± 0.77	-7.71± 1.42	-7.72± 1.24
	QMIX	-8.24± 1.56	-7.56± 0.74	-7.26± 0.78	-7.33± 0.51
	QPLEX	-8.80± 0.75	-8.48± 1.04	-8.40± 1.12	-8.26± 1.34
	MAPPO	-8.77± 0.69	-8.93± 0.39	-8.92± 0.45	-9.04± 0.35
128	SKARL	**-1.93**± 10.97	**-2.36**± 10.16	**-2.22**± 9.31	**-3.60**± 7.52
	Fixed-Kernel	-3.06± 10.05	-3.04± 10.36	-3.95± 9.05	-4.58± 7.38
	MFRL	-8.70± 0.80	-7.72± 1.49	-7.06± 2.09	-7.23± 1.83
	QMIX	-8.23± 1.44	-7.62± 1.65	-7.26± 2.02	-7.35± 1.67
	QPLEX	-8.87± 0.45	-8.10± 0.91	-7.82± 1.10	-7.77± 1.14
	MAPPO	-9.30± 0.11	-9.31± 0.06	-9.31± 0.07	-9.35± 0.03

G.5 Experiments in Coverage Environment

Table 16: Performance Comparison between SKARL and Baselines in Coverage Environment

Algorithm	$N = 4$				$N = 16$				$N = 64$			
	R(↑)	T(↓)	# col(↓)	S(↑)	R(↑)	T(↓)	# col(↓)	S(↑)	R(↑)	T(↓)	# col(↓)	S(↑)
MAPPO	339.6	0.40	0.26± 0.561	**1.00**± 0.0	167.6	0.57	5.3± 2.72	0.13± 0.562	97.3	0.87	18.4± 8.35	0.05± 0.009
MFRL	**396.6**	0.52	**0.03**± 0.0	**1.00**± 0.0	187.0	0.62	2.5± 1.65	0.12± 0.456	216.2	0.86	15.1± 2.32	0.04± 0.871
QMIX	275.4	0.39	4.94± 2.46	**1.00**± 0.0	259.5	**0.52**	19.5± 5.3	0.19± 0.76	324.2	0.92	**11.8**± 4.13	0.10± 0.526
QPLEX	318.5	**0.38**	0.56± 0.194	**1.00**± 0.0	298.7	0.61	7.3± 6.22	0.21± 0.512	834.5	0.85	21.5± 3.65	0.14± 0.290
SKARL	387.2	0.51	0.15± 0.870	**1.00**± 0.0	**320.8**	0.61	**2.42**± 1.67	**0.22**± 0.342	**907.3**	**0.76**	15.3± 5.37	**0.17**± 0832

Table 17: Flexibility Performance of SKARL in Coverage Environment

Training	Metric	$M = 4$	$M = 16$	$M = 64$	$M = 128$	$M = 256$
$N = 4$	R/N	96.8	23.7	0.3	-1.2	-9.2
	T (step)	51	74	92	100	100
	(# col)/N	0.0375	0.76	6.932	32.4	78.9
	S%	100	72	4	0	0
$N = 16$	R/N	97.5	24.05	22.3	4.3	0.82
	T (step)	43	61	67	94	100
	(# col)/N	0.0457	0.19	0.203	2.54	5.21
	S%	100	79	6	6.25	0
$N = 64$	R/N	96.2	25.8	14.2	9.3	3.52
	T (step)	41	56	76	89	92
	(# col)/N	0.0557	0.285	0.239	0	9.68
	S%	100	84	13	75	5

G.6 Experiments in Line Environment

Table 18: Performance Comparison between SKARL and Baselines in Line Environment

Algorithm	$N = 4$				$N = 16$				$N = 64$			
	R(↑)	T(↓)	# col(↓)	S(↑)	R(↑)	T(↓)	# col(↓)	S(↑)	R(↑)	T(↓)	# col(↓)	S(↑)
MAPPO	422.3	0.31	0.10$\pm$ 0.20	**1.00**$\pm$ 0.00	563.4	0.43	1.50$\pm$ 0.90	0.30$\pm$ 0.20	1462.7	0.72	8.00$\pm$ 3.00	0.22$\pm$ 0.08
MFRL	444.8	0.25	0.05$\pm$ 0.10	**1.00**$\pm$ 0.00	591.2	0.43	0.90$\pm$ 0.60	0.36$\pm$ 0.18	1604.3	0.68	6.00$\pm$ 2.50	0.27$\pm$ 0.09
QMIX	421.6	0.25	0.12$\pm$ 0.25	**1.00**$\pm$ 0.00	572.1	0.49	1.80$\pm$ 1.10	0.32$\pm$ 0.19	1510.4	0.64	7.20$\pm$ 2.80	0.24$\pm$ 0.09
QPLEX	**449.7**	**0.27**	0.07$\pm$ 0.15	**1.00**$\pm$ 0.00	608.0	**0.42**	1.20$\pm$ 0.70	0.38$\pm$ 0.17	1624.9	0.67	6.50$\pm$ 2.60	0.26$\pm$ 0.09
SKARL	418.9	0.23	**0.03**$\pm$ 0.08	**1.00**$\pm$ 0.00	**615.6**	0.41	**0.70**$\pm$ 0.50	**0.40**$\pm$ 0.16	**1765.8**	**0.66**	**5.50**$\pm$ 2.20	**0.30**$\pm$ 0.10

Table 19: Flexibility Performance of SKARL in Line Environment

Training	Metric	$M = 4$	$M = 16$	$M = 64$	$M = 128$	$M = 256$
$N = 4$	R/N	104.7	32.2	-4.3	-10.5	-36.4
	T (step)	23	54	87	100	100
	(# col)/N	0.0075	0.076	0.950	4.1	12.1
	S%	100	72	8	0	0
$N = 16$	R/N	117.5	38.5	20.4	6.3	0.72
	T (step)	32	41	84	91	100
	(# col)/N	0.0005	0.044	0.103	0.874	1.54
	S%	100	40	24	3.25	0
$N = 64$	R/N	123.2	53.4	27.58	18.9	2.31
	T (step)	21	31	66	77	82
	(# col)/N	0.0002	0.029	0.085	0.376	0.985
	S%	100	84	30	27	18

G.7 Ablation Study

Is it necessary to apply gradients in RKHS? An alternative way to update a cylindrical function is to apply gradients directly in Euclidean space (Schwenker et al., 2001). Figure 3 compares this method with RKHS-gradient updates in the Move environment with $N = 4$ and 64 kernel features. The results indicate that RKHS-gradient updates improve both training stability and final performance.

Table 20: Performance comparison between different value-decomposition methods.

Algorithm	$N = 4$			$N = 16$			$N = 64$		
	R(↑)	# col(↓)	S(↑)	R(↑)	# col(↓)	S(↑)	R(↑)	# col(↓)	S(↑)
SKARL	902.8	**0**± 0	0.15± 0.0192	3755.9	12.32± 5.847	**0.17**± 0.0500	14423.8	7.9± 5.37	0.15± 0.0334
SKARL-QMIX	921.2	**0**± 0	0.15± 0.0102	3857.2	10.23± 8.421	**0.18**± 0.0431	14512.3	6.2± 4.32	0.15± 0.781
SKARL-QPLEX	**922.7**	**0**± 0	0.15± 0.0021	**3920.1**	**9.42**± 3.412	**0.18**± 0.0622	**14589.1**	7.9± 2.98	**0.16**± 0.676

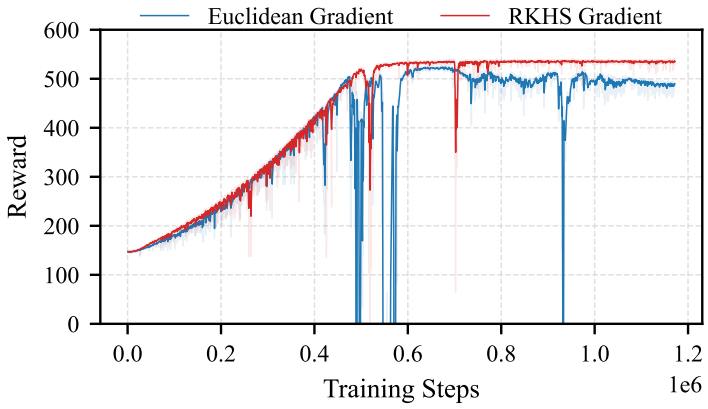


Figure 3: Comparison between gradient in RKHS space and Euclidean space.

How does the number of anchors affect the result? We compare different numbers of anchor points, $L = 1, 2, 8, 32$, on the Move task with $N = 4$ agents. As shown in Figure 4, more anchor points mainly stabilize training (with $L = 32$ producing the most stable training curve), while convergence speed and final performance are only slightly affected. Furthermore, since full performance can be achieved with a single anchor point, SKARL can use as few as one kernel anchor with $L \leq \sqrt{N}$ to achieve lower computational complexity than value-decomposition algorithms such as QMIX (as discussed in Section 4).

How do kernel types affect the result? We compare different kernel choices on the Move task with $N = 4$ agents. Specifically, we compare Gaussian and polynomial kernels. For the Gaussian kernel, we set γ to $(0.5, 1.0, 2.0)$; for the polynomial kernel, we set $(\alpha, d, c) = (1, 2, 1), (1, 3, 1), (1, 4, 1)$. The results are shown in Figure 5. We conclude that the kernel choice may not affect final performance as long as the kernel has sufficient representational capacity.

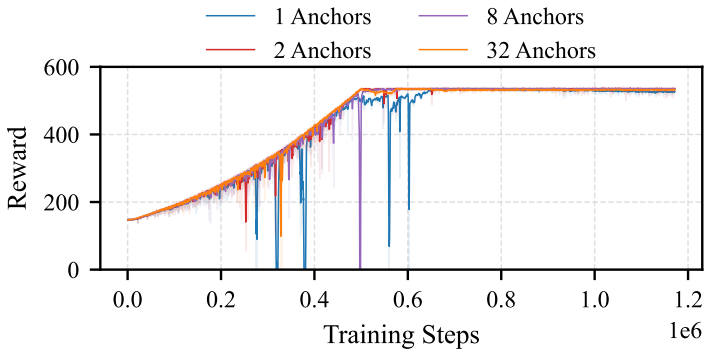


Figure 4: Comparison between different numbers of anchor points.

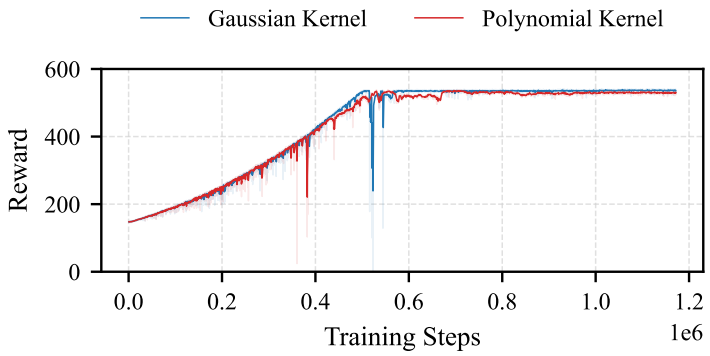
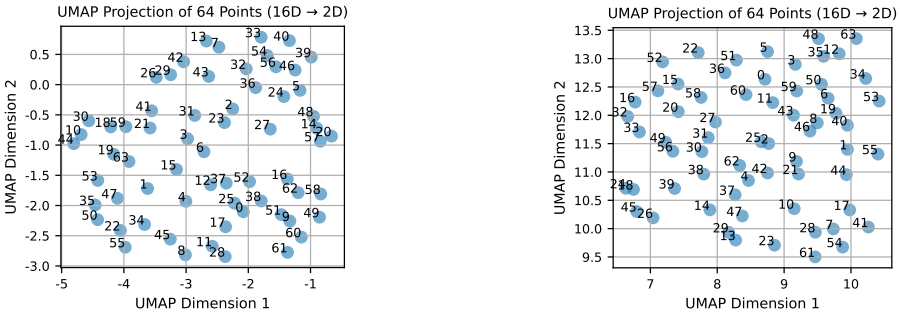


Figure 5: Comparison between different kernel types.

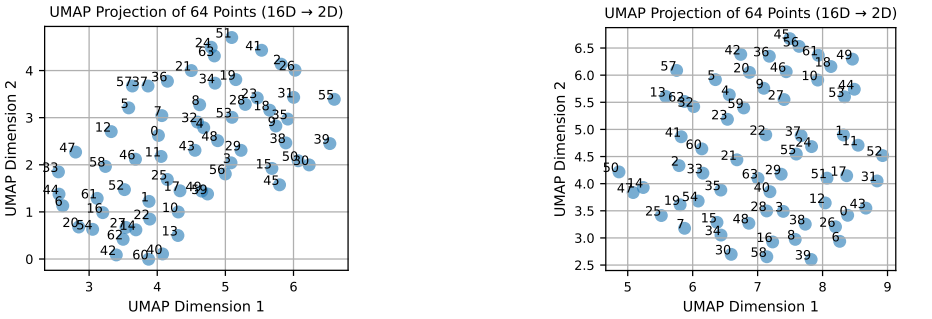
How are anchor points distributed? We plot the distribution of anchor points with UMAP in Figure 6 for $N = 4$. The anchor points of the Gaussian kernel follow an approximately uniform distribution, while those of the polynomial kernel exhibit a clearer structure.



(a) Gaussian Kernel $\gamma = 0.5$

(b) Gaussian Kernel $\gamma = 1.0$

Figure 6: Gaussian kernel anchor points distribution.



(a) Polynomial Kernel $(\alpha, d, c) = (1, 2, 1)$

(b) Polynomial Kernel $(\alpha, d, c) = (1, 3, 1)$

Figure 7: Polynomial kernel anchor points distribution.