$\mathrm{E}(3)$-Equivariant Actor-Critic Methods for Cooperative Multi-Agent Reinforcement Learning

Dingyang Chen [1] **Qi Zhang** [1]

Abstract

Identification and analysis of symmetrical patterns in the natural world have led to significant discoveries across various scientific fields, such as the formulation of gravitational laws in physics and advancements in the study of chemical structures. In this paper, we focus on exploiting Euclidean symmetries inherent in certain cooperative multi-agent reinforcement learning (MARL) problems and prevalent in many applications. We begin by formally characterizing a subclass of Markov games with a general notion of symmetries that admits the existence of symmetric optimal values and policies. Motivated by these properties, we design neural network architectures with symmetric constraints embedded as an inductive bias for multi-agent actor-critic methods. This inductive bias results in superior performance in various cooperative MARL benchmarks and impressive generalization capabilities such as zero-shot learning and transfer learning in unseen scenarios with repeated symmetric patterns.

1. Introduction

It is widely believed by scientists that the organization and operation of our universe follow certain symmetry patterns and principles. These symmetry structures in physics lead to profound implications, such as the existence of conservation laws. When done properly, artificial intelligence (AI) can and has already benefited tremendously from exploiting these symmetries, with perhaps the most well-known example of convolutional neural networks (CNNs) being translation invariant to the input images (Goodfellow et al., 2016). Symmetries have also been identified and exploited for single-agent reinforcement learning (RL), where symmetric state-action pairs essentially define a homomorphism from the original Markov decision process (MDP) to a smaller abstract MDP (Ravindran & Barto, 2001), which has been recently shown to be effective for deep single-agent RL (Van der Pol et al., 2020; Wang et al., 2022a; Zhao et al., 2022b; Mondal et al., 2022; Nguyen et al., 2023).

In this paper, we are interested in cooperative multi-agent reinforcement learning (MARL) problems, where symmetry structures are also prevalent and often exist in two forms. The first form is permutation invariance which exists if agents are homogenous in terms of their effects on state dynamics and reward function, which has been well-studied in prior works, including both formalisms (Nguyen et al., 2017; Yang et al., 2018; Chen et al., 2022) and algorithmic techniques such as actor parameter sharing (Lowe et al., 2017; Rashid et al., 2020b; Chen et al., 2022), permutation-invariant centralized critics (Liu et al., 2020), and mean-filed approximation (Yang et al., 2018). We focus on the second form of Euclidean symmetries, including transformations of translation, rotation, and reflection, which exist if the MARL problem is situated in an Euclidean space. Intuitively, MARL problems exhibit Euclidean symmetries whenever coordinate frames are used to provide references for the agents and their environment, because Euclidean transformations can be simply viewed as being applied to the reference frames, without changing the essence of the agents and environment. We are particularly dealing with 3D Euclidean, i.e., $\mathrm{E}(3)$, multi-agent symmetries, which are prevalent in applications grounded in the physical world.

Although prevalent, multi-agent Euclidean symmetries are relatively underexplored, including only van der Pol et al. (2022) and Yu et al. (2023) as the prior work we know of. Although sharing the same motivation, prior work falls short of providing "genuine" $\mathrm{E}(3)$ multi-agent symmetries in the sense that their Euclidean equivariance is only preserved for rotations of discrete angles in $\{k * 360^\circ/n\}_{k=0}^{n-1}$, which degenerates to the cyclic symmetries of C_n, a subgroup of $\mathrm{E}(2)$ that is only 2D. We identify challenges that have prevented prior work from exploiting continuous $\mathrm{E}(3)$ multi-agent symmetries and highlight our solutions as the three-fold contribution of this work:

(i) The first challenge is to find problem representations that

[1]Artificial Intelligence Institute, University of South Carolina, Columbia, SC, USA. Correspondence to: Dingyang Chen <dingyang@email.sc.edu>, Qi Zhang <qz5@cse.sc.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

are suitable to describe $\mathrm{E}(3)$-symmetries that are distributed among multiple agents. We first rigorously formulate a subclass of Markov games (MGs) (Shapley, 1953), group-symmetric MGs, that describes multi-agent symmetries by mathematical group transformations performed on states, actions, and agents' observations. With $\mathrm{E}(3)$ being the special case, we use 3D point clouds to represent states, actions, and observations in a way that conveniently accommodates all transformations in $\mathrm{E}(3)$.

(ii) The second challenge is to develop architectures that are capable of exploiting continuous $\mathrm{E}(3)$-symmetries. We prove several main properties of group-symmetric MGs, including symmetries in the value functions of symmetric policies and the existence of an optimal policy/value that is group-symmetric. As a key difference from prior work, we exploit those properties by leveraging steerable message passing neural networks (Thomas et al., 2018; Geiger & Smidt, 2022; Brandstetter et al., 2022) as the actor-critic architecture for MARL under the centralized training and decentralized execution (CTDE) paradigm, which are capable of preserving equivariance under all $\mathrm{E}(3)$ transformations.

(iii) As a result, our method achieves superior sample efficiency and generalization performance in a range of benchmark MARL tasks that exhibit continuous $\mathrm{E}(3)$-symmetries but were not accommodated by prior work.

2. Related Work

In single-agent RL, the exploitation of symmetries has been originally formulated by Ravindran & Barto (2001) to reduce the redundancies in state and action spaces. Recent work mostly focuses on the combination of symmetries and deep RL. Some works (Laskin et al., 2020; Yarats et al., 2021) utilize data augmentation, where an image-based observation undergoes transformation such as rotation and translation, to improve data efficiency. This approach mostly focuses on the invariant value functions with respect to the transformed input. Instead of generating more data through symmetric transformations which increases computation time, Van der Pol et al. (2020) build equivariant neural policies that directly support C_n symmetries of discrete rotations. Wang et al. (2022b) extend C_n symmetries to more general continuous $SO(2)$ symmetries that support equivariancy in translation and rotation in 2D. A recent work (Chen et al., 2023) considers a more general continuous $\mathrm{E}(3)$ symmetries in 3D space that additional supports equivariancy in reflection.

Multi-agent symmetries are currently underexplored. Some work (Liu et al., 2020; Chen et al., 2022) focus on the permutation invariance of homogeneous agents, and build permutation invariant value functions for better sample and computation efficiency. The work by Li et al. (2021) considers symmetries specific to multi-agent pathfinding problems to reduce the search space. A recent work by van der Pol et al. (2022) considers C_n symmetries in MARL by extending the framework in the single-agent counterpart. However, it does not provide motivation as to why such symmetries result in equivariant actors and critics, and implementation-wise, their homomorphism network has an input size that scales with $|C_n|$. Instead, our work focuses on continuous $\mathrm{E}(3)$ multi-agent symmetries, and the implementation has the input size of the original observation dimension and still preserves equivariance for any continuous angle.

Yu et al. (2023) share the same motivation of exploiting Euclidean symmetries for cooperative MARL. Our key differences lie in methodology and theoretical results. Our work uses point cloud representations to characterize continuous $\mathrm{E}(3)$ symmetries, which facilitates the incorporation of $\mathrm{E}(3)$-equivariant neural networks, while Yu et al. (2023) use traditional flat vector representations, which are not compatible with equivariant neural networks and they turn to data augmentation which is a less principled method. Our theoretic results state the main properties of symmetric MGs. Besides the optimal value equivalence property included in Yu et al. (2023), we include theoretical results pertaining to observation-based policies, which are unique to multi-agent settings and provide theoretical justification for equivariant multi-agent actor-critic methods.

This work focuses on Euclidean symmetries commonly seen in practice, e.g., position and velocity in 3D space. Special neural architectures have been proposed to directly incorporate geometric inductive bias. Some works (Satorras et al., 2021; Schütt et al., 2021; Jing et al., 2020; Le et al., 2022) preserve equivariancy by equivariant operations in the original 3D Euclidean space. In contrast, other works lift the physical quantities from 3D space to higher-dimensional spaces for more expressive power, either through Lie algebra (Finzi et al., 2020) or spherical harmonics (Thomas et al., 2018) and message passing (Brandstetter et al., 2022), which we adopt for actor-critic architectures.

3. Preliminaries

3.1. Cooperative Markov Games

We use the framework of cooperative Markov game (MG) (Shapley, 1953) to formulate our cooperative multi-agent setting, which consists of N agents indexed by $i \in \mathcal{N} := \{1, ..., N\}$, state space $\mathcal{S}$, joint action space $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$ factored into local action spaces, transition function $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, (team) reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and initial state distribution $\mu \in \Delta(\mathcal{S})$, where we use $\Delta(\mathcal{X})$ to denote the set of probability distributions over $\mathcal{X}$. The MG evolves in discrete time steps: at each time step t, all agents are in some state $s_t \in \mathcal{S}$ and each agent $i \in \mathcal{N}$

chooses its local action $a_t^i \in \mathcal{A}^i$, forming a joint action $a_t = (a_t^1, ..., a_t^N) \in \mathcal{A}$ that induces a transition to a new state at the next time step according to the transition function, i.e., $s_{t+1} \sim P(\cdot|s_t, a_t)$, and the team reward signal $r_t := r(s_t, a_t)$. The state $s_0 \sim \mu$ at time step 0 is drawn from the initial state distribution.

Full and partial observability. For ease of exposition, we will assume the agents can fully observe the states until Section 4, and our methods (Section 5) and experiments (Section 6) accommodate the partial observability setting. We shall consider a more general notion of full observability than directly observing the raw state: each agent $i \in \mathcal{N}$ has access to a function $o^i : \mathcal{S} \to \mathcal{O}^i$ that maps the state space to its local observation space $\mathcal{O}^i$. The agents effectively fully observe the state if and only if o^i is bijective. We will use the notion of $o = \{o^i\}_{i\in\mathcal{N}}$ and write $o(s) := \left(o^1(s), ..., o^N(s)\right) \in \mathcal{O} := \times_{i\in\mathcal{N}}\Delta(\mathcal{O}^i)$. The MG is therefore defined by tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, r, o\rangle$.

Policies and values. The general, state-based joint policy, $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, maps from the state space to distributions over the joint action space. As the size of action space $\mathcal{A}$ grows exponentially with N, the commonly used joint policy subclass is the *product policy*, $\pi = (\pi^1, \cdots, \pi^N) : \mathcal{S} \to \times_{i\in\mathcal{N}}\Delta(\mathcal{A}^i)$, which is factored as the product of local policies $\pi^i : \mathcal{S} \to \Delta(\mathcal{A}^i)$, $\pi(a|s) = \prod_{i\in\mathcal{N}} \pi^i(a^i|s)$, each mapping the state space only to the action space of an individual agent. Define the discounted return from time step t as $R_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$. For agent i, product policy $\pi = (\pi^1, ..., \pi^N)$ induces a value function defined as $V_\pi(s_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t:\infty}\sim\pi}[R_t|s_t]$, and action-value function $Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}\sim\pi}[R_t|s_t, a_t]$. Following policy π, agent i's cumulative reward starting from $s_0 \sim \mu$ is denoted as $V_\pi(\mu) := \mathbb{E}_{s_0\sim\mu}[V_\pi(s_0)]$. Under full observability, a state-based local policy can translate from and to an observation-based policy with $\nu^i(o^i(s)) = \pi^i(s)$ and the corresponding observation-based product policy and its values are denoted as $\nu(o(s)) = (\nu^1(o^1(s)), ..., \nu^N(o^N(s)))$, Q_ν, V_ν, respectively.

3.2. Groups and Transformations

Mathematical symmetry means a type of invariance: a property of an object remaining unchanged after some transformation. This notion of symmetry can be formally described by invariant functions. A function $f : \mathcal{X} \to \mathcal{Y}$ is *invariant* under transformation operator $T : \mathcal{X} \to \mathcal{X}$ if $f(T[x]) = f(x)$ for any $x \in \mathcal{X}$. More generally, function $f : \mathcal{X} \to \mathcal{Y}$ is *equivariant* under transformation operators $T : \mathcal{X} \to \mathcal{X}$ and $T' : \mathcal{Y} \to \mathcal{Y}$ if $f(T[x]) = T'[f(x)]$ for any $x \in \mathcal{X}$. Most often, a symmetric object is invariant to not only one transformation but a set of transformations. Such a set of symmetry transformations often forms a (mathematical) group G, which is a set of elements equipped with a binary operator satisfying the group axioms of closure, associativity, identity, and inverse (Dummit & Foote, 1991). For function f to be equivariant to group G, there exists transformation operators T_g and T'_g, called *group actions*, associated with each group element $g \in G$, such that $f(T_g[x]) = T'_g[f(x)]$ for any $x \in \mathcal{X}$.

The Euclidean group $\mathrm{E}(3)$**.** We are mostly interested in group $\mathrm{E}(3)$ that comprises the group actions of translations, rotations, reflections, and finite combinations of them in 3D. Therefore, it has as its subgroups the 3D translation group $\mathrm{T}(3)$, and the orthogonal group $\mathrm{O}(3)$ for 3D rotations and reflections. The group actions of $\mathrm{O}(3)$ can implemented by multiplying square matrices, called *representation matrices*. For example, for the group element of $g = (\alpha, \beta, \gamma)$ that rotates a 3D vector $x = \mathbf{x} \in \mathbb{R}^3$ by α, β, and γ, about x-, y-, and z- axes respectively, the rotation can be represented by matrix multiplication as $T_g[x] = \mathbf{R}_{\alpha,\beta,\gamma}\mathbf{x}$, where $\mathbf{R}_{\alpha,\beta,\gamma} \in \mathbb{R}^{3\times3}$ is the 3D rotation matrix.

4. Markov Games with Euclidean Symmetries

With the notions introduced, we are ready to define group-symmetric MGs by asking the transition, reward, and observations functions to be equivariant under group actions, as stated in Definition 4.1.

Definition 4.1 (G-symmetric MG)**.** Consider MG $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, r, o\rangle$ and a (mathematical) group G equipped with group actions $\{L_g : \mathcal{S} \to \mathcal{S}, K_g^s : \mathcal{A} \to \mathcal{A}, I_g^{s,a} : \mathbb{R} \to \mathbb{R}, H_g^s : \mathcal{O} \to \mathcal{O}\}_{g\in G}$. The MG is *$G$-symmetric* if, for any $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$, and $g \in G$, we have

$$P(s'|s, a) = P\left(L_g[s'] \mid L_g[s], K_g^s[a]\right),$$
$$r\left(L_g[s], K_g^s[a]\right) = I_g^{s,a}\left[r(s, a)\right],\ o\left(L_g[s]\right) = H_g^s\left[o(s)\right]$$

i.e., P, r, and o are equivariant to the group actions.

Definition 4.1 is general enough to accommodate arbitrary symmetries in MGs. For example, choosing the group actions to be permutations of $\mathcal{N}$ (i.e., $G = \mathrm{S}(N)$) can describe permutation invariance between homogenous agents (Liu et al., 2020; Chen et al., 2022).

Moreover, these group actions in general are allowed to depend on the state and/or the action. In this work, we focus on the special case of 3D Euclidean symmetries in MGs, i.e., $G = \mathrm{E}(3)$, where the group actions are Euclidean transformations and do not depend on the state or the action.

For ease of exposition, we will ground our discussion with a running example of Cooperative Navigation, a popular cooperative multi-agent benchmark task (Lowe et al., 2017).

MPE's Cooperative Navigation and its symmetries. In Cooperative Navigation of Multi-Agent Particle Environment (MPE) (Lowe et al., 2017), a popular MARL benchmark, N agents move as a team to cover N landmarks in a

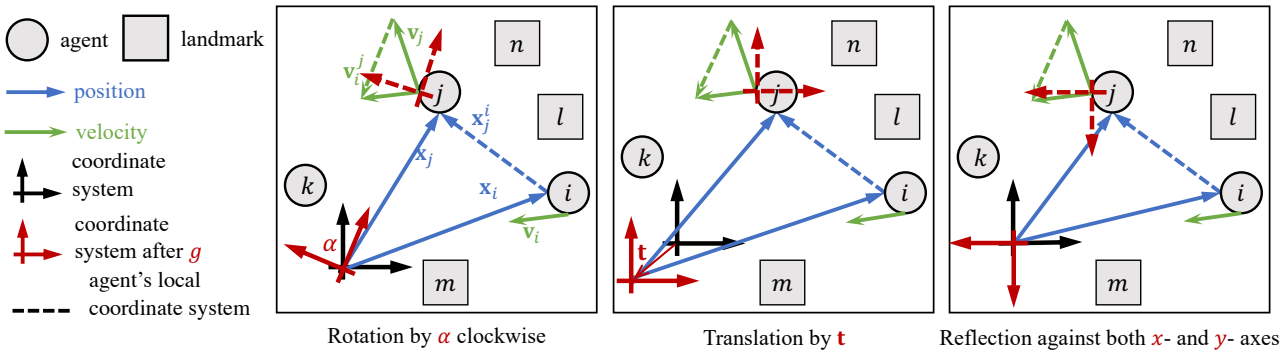

Figure 1: Illustration of Cooperative Navigation ($N = 3$) and its Euclidean symmetries.

2D space. The landmarks are randomly initialized at the beginning of an episode and fixed throughout the episode. The reward functions determine the reward for all agents according to the distances between all agents and the landmarks to encourage coverage, as well as penalties to discourage collisions if any. Under full observability, the observation of an agent contains its own absolute location and velocity, the relative locations and velocities of all other agents, and the relative locations of all the nearest landmarks. Figure 1 (left) shows the case where $N = 3$ and illustrates the rotational symmetry therein. The states and actions can be represented as 2D vectors under a global reference frame (e.g., agent i's position, $\mathbf{x}_i$), whereas each agent's observation is 2D vectors referenced to its local frame (e.g., agent i's velocity relative to agent j, $\mathbf{v}_i^j = \mathbf{v}_i - \mathbf{v}_j$). Upon a rotation of all entities, which is shown in Figure 1 (left) as a rotation of the reference frames, the states, actions, and observations are rotated accordingly, yet the essence of the entities remain unchanged, and therefore the transition, reward, and observation functions are equivariant/invariant. Similarly, it is easy to see that Cooperative Navigation also exhibits translational and reflectional symmetries, as shown in Figure 1 (middle and right, respectively). Further, it is straightforward to extend these symmetries from 2D to 3D, thus making Cooperative Navigation $\mathrm{E}(3)$-symmetric.

With the intuition from Cooperative Navigation, we now formally define $\mathrm{E}(3)$-symmetric MGs in Definition 4.2, where we will represent states and observations as 3D point clouds and specify the corresponding group actions applied to them.

Definition 4.2 ($\mathrm{E}(3)$-symmetric MGs)**.** MG $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, r, o \rangle$ is $\mathrm{E}(3)$-*symmetric* if the following conditions hold.

(i) A state is a 3D point cloud consisting of agents $\mathcal{N}$ and other uncontrollable entities $\mathcal{M}$, i.e., $s = \{(\mathbf{x}_v, \mathbf{f}_v)\}_{v \in \mathcal{V}}$, where $\mathcal{V} = \mathcal{N} \cup \mathcal{M}$ is the set of all entities, $\mathbf{x}_v \in \mathbb{R}^3$ is the 3D position vector of entity v with feature vector $\mathbf{f}_v \in \mathbb{R}^{d_v}$.

(ii) The local action spaces are Euclidean spaces, i.e., $\mathbf{a}^i \in \mathcal{A}^i = \mathbb{R}^{d_{\mathbf{a}^i}}, \forall i \in \mathcal{N}$.

(iii) Agent i's observation is represented by a 3D point cloud consisting of the entities in its own view, $o^i(s) = \{(\mathbf{x}_v^i, \mathbf{f}_v^i)\}_{v \in \mathcal{V}}$ where $\mathbf{x}_v^i \in \mathbb{R}^3$ is the 3D position vector of entity v with feature vector $\mathbf{f}_v^i \in \mathbb{R}^{d_{\mathbf{f}_v^i}}$, both relative to i.

The MG's P and o are equivariant and r is invariant to the $\mathrm{E}(3)$ group actions in Definition 4.1 that are specified as follows:

(iv) Under translation, the state is transformed with only the entities' positions being translated, i.e., $s = \{(\mathbf{x}_v, \mathbf{f}_v)\}_{v \in \mathcal{V}} \rightarrow_g s = \{(\mathbf{x}_v + \mathbf{t}_g, \mathbf{f}_v)\}_{v \in \mathcal{V}}$ where $\mathbf{t}_g \in \mathbb{R}^3$ is the 3D translation for $g \in \mathrm{T}(3)$, while actions and observations remain unchanged.

(v) Under rotations or reflections, all vectors in states, local actions, and observations are transformed according to their respective group representations. For example, the state transforms as $s = \{(\mathbf{x}_v, \mathbf{f}_v)\}_{v \in \mathcal{V}} \rightarrow_g s = \{(\mathbf{D}_g^{\mathbf{x}} \mathbf{x}_v, \mathbf{D}_g^{\mathbf{f}} \mathbf{f}_v)\}_{v \in \mathcal{V}}$ where $\mathbf{D}_g^{\mathbf{x}}$ and $\mathbf{D}_g^{\mathbf{f}}$ are the representation matrices of $g \in \mathrm{O}(3)$ for the vector spaces of $\mathbf{x}_v$ and $\mathbf{f}_v$, respectively.

In essence, Definition 4.2 requires the states and observations to be represented as 3D point clouds with the entities being the points with corresponding features, which are transformed according to Euclidean symmetries that are intuitively just changes of the reference frames. The actions can be understood as features associated with the subset of agents that also are transformed accordingly. In the example of Cooperative Navigation, the point feature vector for an entity $v \in \mathcal{V}$ includes its entity type to differentiate from agent vs landmark, which in our experiments is represented by a one-hot vector of two classes, $\mathbf{f}_v \in \mathbb{R}^2$; the action of agent $i \in \mathcal{N}$ includes its velocity, $\mathbf{a}^i = \mathbf{v}_i \in \mathcal{A}^i = \mathbb{R}^3$ with dummy z values.

Examples of $\mathrm{E}(3)$-symmetric MGs. Besides Cooperative Navigation, all other scenarios in MPE are $\mathrm{E}(3)$-symmetric.

Further, by Definition 4.2, any MG involving multiple entities interacting in a 2D/3D space is $\mathrm{E}(3)$-symmetric, which includes many real-world applications that are grounded in physical words, such as multi-robot systems, video games, materials design, etc. Two other domains in our experiments that exhibit $\mathrm{E}(3)$-symmetric are the continuous control tasks in DeepMind Control Suite and game scenarios in StarCraft Multi-Agent Challenge, both being popular MARL benchmarks, with more details presented with our experiments. We give more examples of $\mathrm{E}(3)$-symmetric MGs in the appendix.

We would like to remark that, in Definition 4.2 that applies to the MARL benchmarks, an agent's observation uses a local reference frame with the agent's position as the origin and with its orientation aligned with the global reference frame. So the agent's own reference frame is the same as the global one up to a position shift, and that is the fundamental reason why observations, just like states, are transformed under rotations/reflections. It is also straightforward to extend this definition (and the method) to the case where local reference frames have orientations different from the global reference frame, with several key points in doing so: 1) An agent's local orientation should be part of its features and therefore part of the global state. Precisely, in Definition 4.2, the orientation should be included in feature vector $\mathbf{f}_v$, where v is the vertex in the state Euclidean graph corresponding to the agent; 2) Upon a rotation, this orientation in feature vector $\mathbf{f}_v$ should be rotated accordingly; and 3) If the observation is represented using the local reference frame, then it is often invariant to rotations, since the local reference frame is also rotated.

We are now ready to derive the main properties of group symmetric MGs: since the transition, reward, and observation functions are group-symmetric, if the policy is also group-symmetric, then we can expect its value functions are invariant under the group actions. For example, under a rotation angle α if all agents in Cooperative Navigation always choose their velocities with the same rotation angle α, then the policy value remains unchanged. Definition 4.3 formally states this requirement in general G-symmetric MGs for both state-based and observation-based policies.

Definition 4.3 (G-invariant MG policies)**.** Let $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ be a state-based policy in a G-symmetric MG. We say π is *G-invariant* if it is invariant to the group actions of G, i.e., for any $s \in \mathcal{S}$, $a \in \mathcal{A}$, and $g \in G$, $\pi(a|s) = \pi(K_g^s[a] \mid L_g[s])$. Similarly, an observation-based policy $\nu : \mathcal{O} \to \times_{i\in\mathcal{N}}\Delta(\mathcal{A}^i)$ is *G-invariant* if $\nu(a|o(s)) = \nu(K_g^s[a] \mid H_g^s[o(s)])$ for any $s \in \mathcal{S}$, $a \in \mathcal{A}$, and $g \in G$.

Under the CTDE paradigm and bijective observation functions, the value functions of an observation-based product policy ν can directly condition on states, i.e., $V_\nu(s)$ and $Q_\nu(s,a)$. We list the properties for observation-based G-invariant product policies and their state(-action) value functions in Theorem 4.4 below, which justifies how our method will exploit these properties under the CTDE paradigm.

Theorem 4.4 (Main properties of G-symmetric MGs, proof in the appendix)**.** *For a G-symmetric MG,*

(i) The optimal values are G-invariant, $V_*(s) = V_*(L_g[s])$, $Q_*(s,a) = Q_*(L_g[s], K_g^s[a])$.

(ii) There exists an observation-based policy ν that are G-invariant and optimal, $V_\nu(s) = V_*(s), Q_\nu(s,a) = Q_*(s,a)$.

Further, for a G-invariant observation-based policy ν,

(iii) Its value function is G-invariant: for any $s \in \mathcal{S}$, $a \in \mathcal{A}$, and $g \in G$, $V_\nu(s) = V_\nu(L_g[s]), Q_\nu(s,a) = Q_\nu(L_g[s], K_g^s[a])$.

(iv) Similarly, if ν is parameterized by θ as ν_θ and differentialble, then $\nabla_\theta \nu_\theta(a|o(s)) = \nabla_\theta \nu_\theta(K_g^s[a] \mid o(L_g[s]))$.

Theorem 4.4 extends prior works on properties in single-agent symmetric MDPs (Ravindran & Barto, 2001; Rezaei-Shoshtari et al., 2022), and we are the first to take care of the distributed nature of the symmetries in MGs to make a rigorous proof. Theorem 4.4 establishes the properties our method will exploit next.

5. $\mathrm{E}(3)$-Equivariant Multi-Agent Actor-Critic

The properties stated in Theorem 4.4 naturally prompt us with the idea of adopting group-invariant architectures for cooperative MARL. In this work, we consider multi-agent actor-critic methods, such as MADDPG (Lowe et al., 2017) and MAPPO (Yu et al., 2022). Specifically, properties (i) and (ii) in Theorem 4.4 suggests that we can reduce the search of optimality within group-invariant functions for actors and critics. Moreover, properties (iii) and (iv) imply that group-invariant actors enjoy symmetric policy gradients:

$$\begin{aligned}&\nabla_\theta \pi_\theta(a|s) \cdot Q_{\pi_\theta}(s,a)\\ =&\nabla_\theta \pi_\theta(K_g^s[a] \mid L_g[s]) \cdot Q_{\pi_\theta}(L_g[s], K_g^s[a])\end{aligned}$$

which suggests that the optimization landscape is symmetric and therefore can be more efficiently optimized via gradient-based search (Zhao et al., 2022a; 2023).

This idea is further motivated by our empirical analysis that finds emergence of group-invariancy in traditional actor-critic architectures that have no guaranteed invariancy. Specifically, we train the agents in Cooperative Navigation ($N = 3$) via MADDPG with MLP-based actors and critics. Figure 2 plots the degree of the (observation-based) actors,

$\{\nu^i\}_{i\in\mathcal{N}}$, being invariant to rotations and translations, respectively, during training. For rotations, we select a finite set of angles, $A = \{30°, 60°, \cdots, 330°\}$, and quantify the corresponding invariancy measure in state s as

$$\frac{1}{|A|N}\sum_{i,\alpha\in A}\cos\left(\mathrm{rot}_\alpha[\nu^i(o^i(s)],\ \nu^i(o^i(\mathrm{rot}_\alpha[s]))\right) \quad (1)$$

where $\mathrm{rot}_\alpha[\cdot]$ performs the rotation by α and $\cos(\cdot,\cdot)$ measures the cosine similarity. We measure the translation invariancy similarly.

5.1. E(3)-Equivariant Message Passing

In this work, we implement E(3)-equivariant/invariant actor-critic architectures with E(3)-equivariant message passing neural networks (E3-MPNNs) (Thomas et al., 2018; Geiger & Smidt, 2022; Brandstetter et al., 2022), a type of graph neural networks that process 3D *Euclidean graphs*, graphs where vertices are located at 3D positions, as an E(3)-equivariant function. Formally, an input Euclidean graph is represented as a tuple $G^{\text{in}} = (\mathcal{V}, \mathcal{E}, \mathbf{x}, \mathbf{f}^{\text{in}})$ where $\mathcal{V}$ is a set of vertices with 3D positions $\mathbf{x}_v \in \mathbb{R}^3$, $\mathcal{E} \subseteq \mathcal{V}\times\mathcal{V}$ is a set of edges, and $\mathbf{f}^{\text{in}}$ is a set of feature vectors, each associated with a vertex or an edge. The backbone of E3-MPNNs is E(3)-equivariant message passing layers, denoted as $\texttt{E3-MP}(\cdot)$, maps the input Euclidean graph to an output Euclidean graph by updating only the feature set, $G^{\text{out}} = (\mathcal{V}, \mathcal{E}, \mathbf{x}, \mathbf{f}^{\text{out}})$, in an E(3)-equivariant manner, $\texttt{E3-MP}(T_g^{\text{in}}[G^{\text{in}}]) = T_g^{\text{out}}[\texttt{E3-MP}(G^{\text{in}})]$ for $g \in \mathrm{E}(3)$, where T_g^{in} and T_g^{out} include Euclidean transformations (i.e., translation, rotation, etc.) applied to the vertices and features in the input and output graph, respectively. After the message passing layers, E3-MPNNs produce the final output $y \in \mathcal{Y}$ using a graph readout layer, $\texttt{E3-Readout}(\cdot)$, which is also E(3)-equivariant, $\texttt{E3-Readout}(T_g^{\text{out}}[G^{\text{out}}]) = T_g^{\mathcal{Y}}[\texttt{E3-Readout}(G^{\text{out}})]$ for $g \in \mathrm{E}(3)$. This ensures overall equivariancy from the input graph to the final output.

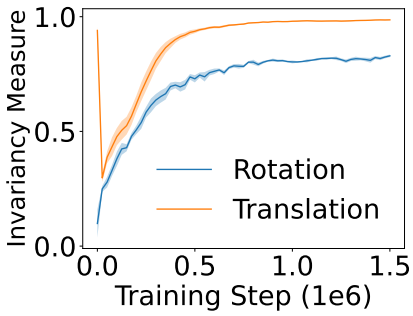

Figure 2: Emergence of rotation- and translation-invariancy in MLP actors trained on 3-agent Cooperative Navigation.

5.2. Integration Into Multi-Agent Actor-Critic Methods

In order to use E3-MPNNs for multi-agent actor-critic architectures, we need to first represent state s, state-action pair (s, a), and observation $o^i(s)$ as Euclidean graphs for centralized state-value critic $V(s)$, action-value critic $Q(s, a)$, and observation-based actor $\nu^i(o^i(s))$, respectively. This is straightforward for E(3)-symmetric MGs, because by Definition 4.2 the states and observations are represented as 3D point clouds which can be directly cast into 3D Euclidean graphs if edges are added. In our experiments, we use some heuristics to add edges, e.g., each vertex is connected with all others as a complete graph or with a fixed number of its nearest neighbors. For a state-action pair, we simply treat the actions as additional feature vectors of individual agent vertices. Figure 3 illustrates the E(3)-equivariant action-value critic and observation-based actor for MADDPG.

We below highlight several implementation considerations. More details are provided in the appendix.

SEGNN. For our actor-critic architecture, we employ steerable E(3) equivariant GNNs (SEGNNs), a recently developed E3-MPNN architecture that has superior performance on supervised learning tasks in computational physics and chemistry (Brandstetter et al., 2022). SEGNNs split the input set of feature vectors into vertex feature vectors and edge feature vectors, $\mathbf{f}^{\text{in}} = (\mathbf{f}^{\text{node}}, \mathbf{f}^{\text{edge}})$, which comprises so-called steerable feature vectors concatenated by $(2l+1)$-dimensional vectors with $l = 0, 1, ...$, often representing properties of the nodes (i.e., vertices) and edges. For example, in Cooperative Navigation, the node feature vector of agent i is $\mathbf{f}_i^{\text{node}} = [\mathbf{v}_i, \|\mathbf{v}_i\|, \mathbf{a}_i, \|\mathbf{a}_i\|, \text{node\_type}]$, which is the concatenation of $(2l+1)$-dimensional vectors with $l \in \{0, 1\}$.

Observability. We adopt the paradigm of centralized training and decentralized execution, where the centralized critic can fully observe the state, while the local actors are observation-based with partial observability in general. For instance, in Cooperative Navigation, the velocities of other agents are not observable.

Our method and experiments are restricted to memory-less,

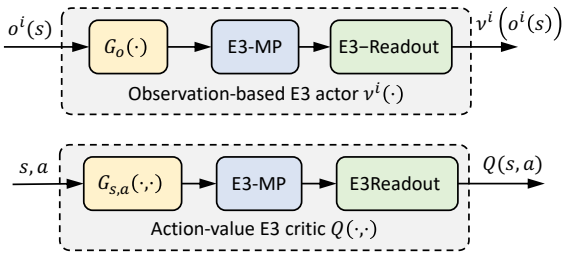

Figure 3: Architectures for E(3)-equivariant MADDPG.

non-recurrent actor-critic architectures even under partial observability. This is because realizing group-equivariancy in recurrent neural networks needs further technical treatments, which is left for future work.

To enforce uniform dimensionality required by SEGNN, we handle missing quantities for certain nodes due to partial observability by padding dummy vectors of zeros.

6. Experiments

Environments. We choose the popular cooperative MARL benchmarks of MPE, MuJoCo continuous control tasks (MuJoCo tasks), including the 2D ones from Tassa et al. (2018) and 3D ones from Chen et al. (2023) with single- and multi-agent variations, and StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) to evaluate the effectiveness of our $\mathrm{E}(3)$-equivariant multi-agent actor-critic methods described in Section 5. Specifically, there are three task scenarios chosen in MPE, Cooperative Navigation, Cooperative Push, and Predator and Prey, where the collective goals of the controllable agents are to navigate the landmarks, push a ball to a target location, and catch all the preys, respectively. In MuJoCo tasks, we consider the representative tasks of cartpole, single- and multi-agent reacher, single- and multi-agent swimmer, multi-agent 3D hopper, and multi-agent 3D walker. The tasks vary in the degree of $\mathrm{E}(3)$ symmetries. In SMAC, the selected scenarios are 8m_vs_9m with *hard* difficulty and 6h_vs_8z with *super hard* difficulty. These tasks exhibit different levels of $\mathrm{E}(3)$-symmetries. All MPE and MuJoCo tasks exhibit perfect $\mathrm{E}(3)$-symmetries. SMAC, however, only exhibits imperfect $\mathrm{E}(3)$-symmetries due to two reasons: 1) the uncontrollable enemies in SMAC tasks might be $\mathrm{E}(3)$-symmetric, which might break the symmetries for the overall MG, and 2) another reason is that the navigation actions are categorical in four Cartesian directions, which limit the actions' expressiveness for $\mathrm{E}(3)$-symmetries.

Baselines. In MPE, we choose MADDPG as the framework of the algorithms. We consider the classic implementation of both actor and critic by MLPs as the baseline. Another baseline (Liu et al., 2020) achieving state-of-the-art performance in MPE implements the critic by a graph convolutional neural network (GCN) that has no guarantee of being $\mathrm{E}(3)$-invariant. Our algorithms incorporate $\mathrm{E}(3)$-invariancy in critic and/or actor by SEGNN-based implementations. We denote the algorithms in the format of [critic_type, actor_type], with the baselines being [MLP, MLP] and [GCN, MLP], and our algorithms [SEGNN, SEGNN] and [SEGNN, MLP]. In MuJoCo tasks, we also apply MADDPG as the underlying framework of our algorithms. Similar to the case in MPE, the baselines are denoted as [MLP, MLP] and [GCN, MLP], and ours is denoted as [SEGNN, SEGNN]. For fair comparison, the input graph for the GCN-based critic is the same as that of the SEGNN-based one. In SMAC, the framework of the algorithms is MAPPO, where both actor and critic are commonly implemented by recurrent neural networks (RNNs) to incorporate trajectory-level information. However, group-equivariancy in RNNs needs further technical treatments so here we focus on non-recurrent MLP-based baselines, denoted as MAPPO-[MLP, MLP]. Another popular baseline we consider is QMIX (Rashid et al., 2020a) with recurrency, i.e., the value function is implemented by an RNN. Due to the categorical nature of SMAC's action space, our algorithm only incorporates $\mathrm{E}(3)$-invariancy in the critic, denoted as MAPPO-[SEGNN, MLP].

For fair comparison, all the algorithms have comparable amounts of parameters than those of the baselines, with the details in Appendix G. Our code is publicly available at `https://github.com/dchen48/E3AC`.

Results overview. The MPE results show that the SEGNN-based critic and/or actor outperforms the baselines by a significant margin. Further, since the SEGNN-based architecture can deal with point clouds with an arbitrary number of points, the capability of zero-shot learning and transfer learning is empirically verified in MPE. Moreover, the emergence of invariancy of the baselines is found in all scenarios of MPE. In the MuJoCo tasks, the SEGNN-based architecture performs similarly to the MLP-based baseline in cartpole, consistent with the fact that the symmetries therein are relatively sparse. In all the other selected tasks with inherently heavier $\mathrm{E}(3)$-symmetries, SEGNN-based architectures achieve noticeable improvements over the baseline. In SMAC, the experiments show that even the environment only partially satisfies the requirements of $\mathrm{E}(3)$-symmetric MGs, our SEGNN-based architecture can still be better performed in 6h_vs_8z. However, in SMAC, there is no obvious emergence of invariancy in MLP-based architectures, consistent with its limited $\mathrm{E}(3)$-symmetries.

6.1. Results on MPE

Performance. In MPE, Figure 4 shows the learning curves comparing our SEGNN-based architecture described in Section 5 against the baselines of MLP- and GCN-based architecture. The result clearly illustrates the effectiveness of the SEGNN-based algorithm. Specifically, by only using the SEGNN-based critic, i.e., [SEGNN, MLP] shown in the green curve, the learned architecture has already outperformed the baselines [GCN, MLP] and [MLP, MLP], shown in the blue and orange curves, respectively, in all scenarios in MPE. There is further performance boosting in Push_N3, Prey_N3, Navigation_N6, and Prey_N6 if we also use SEGNN-based actor, i.e., [SEGNN, SEGNN] shown with the red curves.

Zero-shot/transfer learning. Since SEGNN-based architectures can deal with point clouds of variable numbers of

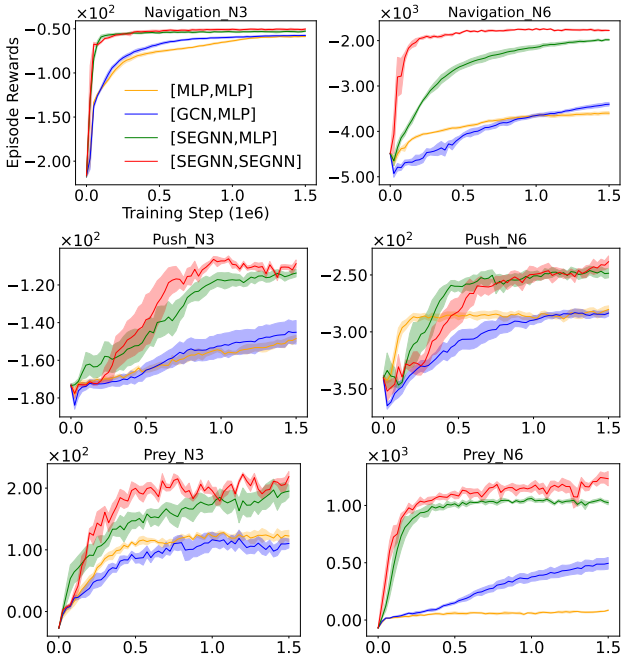

Figure 4: Performance comparison on MPE.

points, it is a natural question if they can perform well in different scenarios with similar setups, i.e., zero-shot learning. To answer this question, we test the performance of the SEGNN-based actors learned in 3-agent tasks for the corresponding 6-agent tasks. The performance is normalized by linear mapping according to the performance of [SEGNN, SEGNN], whose initial performance is set to 0, and the final performance is set to 1. The result is averaged across all seeds and scenarios, plotted in Figure 5. At step 0, the performance of SEGNN-based architecture learned in the 3 agents' scenarios (black curve) is close to the final performance of [MLP, MLP] and that of [GCN, MLP] after training 0.75e6 steps. This empirically proves its good capability of zero-shot learning. Further, we can continue training the SEGNN-based architecture learned in 3 agents' scenarios in the corresponding 6 agents' scenarios to show its capability

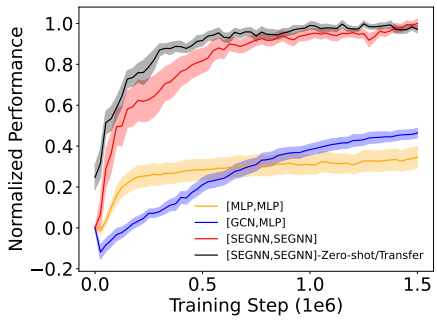

Figure 5: Performance of zero-shot and transfer learning.

of transfer learning. Its effectiveness of transfer learning is empirically verified by the transferred architecture's faster convergence rate than the non-transferred counterpart.

Emergence of Euclidean invariancy. The GCN- and MLP-based architectures are by construction neither invariant to rotations nor translations with arbitrary weights. However, they may gradually adapt to have such abilities after training with a large number of data with invariant transformation, as shown in some previous works on data augmentation (Perez & Wang, 2017) and contrastive learning (Chen et al., 2020). The measurements of Euclidean invariancy are defined similarly to Equation (1), with details in Appendix E.1 for MPE and E.3 for SMAC.

As illustrated in Figure 9 in the appendix, the actor trained with a SEGNN-based critic demonstrates the highest increase of rotation- and translation-invariancy, followed by those trained with GCN- and MLP-based critics. In contrast, there is no improvement in rotation- and translation-invariancy for both GCN- and MLP-based critics. This underscores the necessity of a SEGNN-based critic with inherent invariancy.

6.2. Results on (Multi-Agent) MuJoCo Tasks

Point cloud representations. MuJoCo tasks have Euclidean symmetries since they involve 3D robotic control. Here we choose several representative ones: cartpole (balance, sparse), single- and multi-agent reacher (hard), single- and multi-agent swimmer with two links, and 3D variants of multi-agent hopper and walker modified from the single-agent counterparts from Chen et al. (2023). Note our method subsumes the single-agent setting as the special case of $N = 1$: we treat the single-agent tasks just like the multi-agent tasks, except that only one vertex in the point cloud has an action space to be controllable (because it's single-agent), while all other vertices are non-controllable entities. This is contrastively different from standard methods for these single-agent tasks (i.e., [MLP, MLP] baseline) where all state variables are cluttered into a single vector.

The goals and the levels of $\mathrm{E}(3)$-symmetries of the selected scenarios are described in detail in Appendix C.2. The state contains physical attributes in $\mathbb{R}^3$, e.g., Cartesian coordinates, velocities, and angles of the robots' sub-components. The corresponding state-based point cloud in each task is described in detail in Appendix D.2.

Performance. The performance of the algorithms in MuJoCo tasks are shown in Figure 6. The selected tasks have different levels of symmetry, ranging from 2D (cartpole, reacher, and swimmer) to 3D (hopper and walker). The task cartpole (balance, sparse) only has rotation equivariancy of rot_{180° with respect to the z axis. Due to the lack of symmetries, we do not see much difference for SEGNN-based

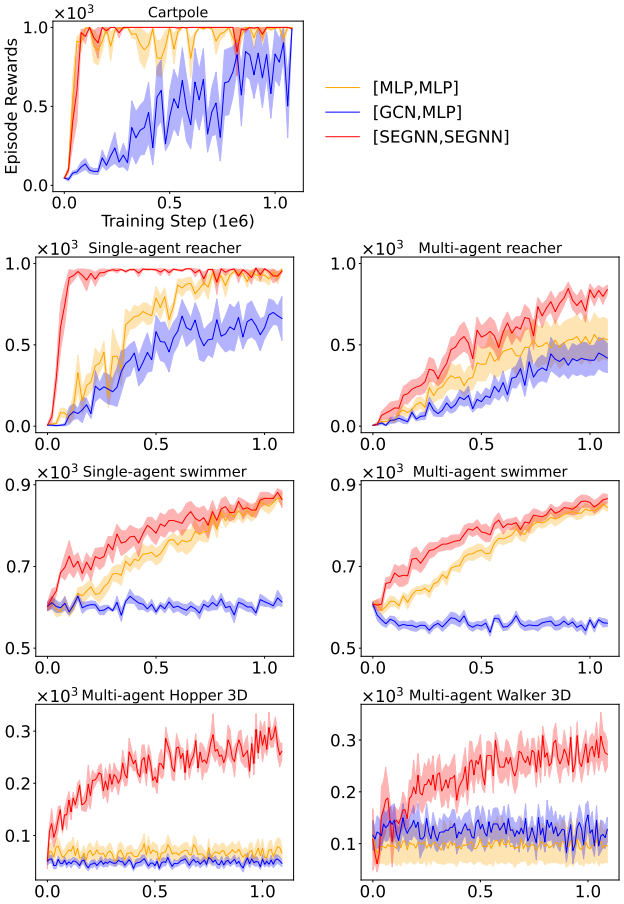

Figure 6: Performance comparison on MuJoCo tasks.

architectures against the baseline of [MLP, MLP]. The other 2D tasks, single- and multi-agent reacher, and single- and multi-agent swimmer inherently have stronger 2D rotation and translation equivariancy. Compared to the baseline of [MLP, MLP], our method [SEGNN, SEGNN] shows both faster convergence rate and better performance in single- and multi-agent reacher, and a faster convergence rate in single- and multi-agent swimmer. Due to gravity, which always points in the negative direction of the z axis, the 3D tasks, multi-agent hopper and walker, also have 2D rotation and translation equivariancy in the xy plane which is utilized by our method [SEGNN, SEGNN] to have superior performance compared to the baseline of [MLP, MLP]. In all the selected tasks, the baseline of [GCN, MLP] performs the worst, even if it uses the same input graph as the [SEGNN, SEGNN]. This illustrates the benefit of exploiting the inductive bias of Euclidean symmetries to reduce the search space of the neural networks' parameters.

6.3. Results on SMAC

Performance. The scenarios in SMAC (Samvelyan et al., 2019) are competitive in nature, so the invariancy of the

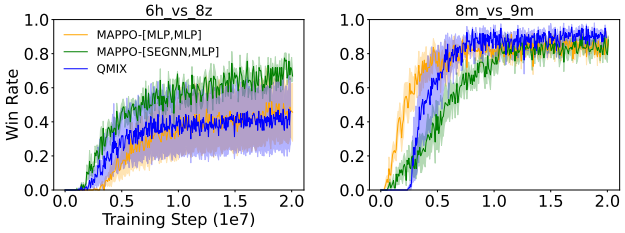

Figure 7: Performance comparison on SMAC.

game breaks due to the non-invariant behaviors of the uncontrollable enemies. Another possible reason for breaking invariancy is that the categorical actions in SMAC are not Euclidean spaces, as required by $\mathrm{E}(3)$-symmetric Markov game. Therefore, we may not observe the big increase in performance as the ones shown in MPE, especially in the non-competing ones navigation and push. We indeed see a mixed of performance in Figure 7 that the SEGNN-based critic helps learn a better-performing actor in 6h_vs_8z but not in 8m_vs_9m.

Emergence of Euclidean invariancy. As illustrated in Figure 10 in the appendix, there is no obvious emergence of the rotation-invariancy for both actor and critic. This can be caused by the competitive nature of the scenarios, where we also see a decrease in actors' translation-invariancy in competitive scenarios Prey_N3 and Prey_N6. Another possible reason is the categorical nature of the actions, e.g., the move directions $\{$north, south, east, west$\}$ are only rotation-equivariant to multiples of 90° instead of arbitrary angles.

7. Conclusion

We have developed formalisms and methods for exploiting symmetric structures in cooperative multi-agent reinforcement learning, with a focus on 3D Euclidean symmetries. Specifically, we have formulated the novel notion of group-symmetric Markov games and derived its key properties that admit group-symmetric optimal values and policies. Consistent with these properties, we have discovered the emergence of Euclidean symmetries in vanilla MLP-based multi-agent actor-critic architectures. Then, we have developed $\mathrm{E}(3)$-equivariant message passing actor-critic architectures that specifically suit group-symmetric Markov games, which results in superior sample efficiency and generalization capabilities in most benchmark MARL tasks.

We observe two limitations of this work. First, the proposed method requires knowledge and annotation of strict symmetries inherent in the multi-agent task, which might not be easily available. This prompts future work of automatic discovery of symmetries, which can even be approximate. Second, we are restricted to memory-less, non-recurrent architectures in this work even under partial observability.

Acknowledgements

Dingyang Chen acknowledges funding support from NSF award IIS-2154904. Qi Zhang acknowledges funding support from NSF award IIS-2154904 and NSF CAREER award 2237963. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the views of the sponsors. The authors thank the anonymous reviewers for their insightful and constructive reviews.

Impact Statement

This paper presents work whose goal is to make the learning process of multi-agent reinforcement learning more efficient. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E. J., and Welling, M. Geometric and physical quantities improve e(3) equivariant message passing. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=_xwr8gOBeV1.

Chen, D., Li, Y., and Zhang, Q. Communication-efficient actor-critic methods for homogeneous markov games. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xy_2w3J3kH.

Chen, R., Han, J., Sun, F., and Huang, W. Subequivariant graph reinforcement learning in 3D environments. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 4545–4565. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/chen23i.html.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Dummit, D. S. and Foote, R. M. *Abstract algebra*, volume 1999. Prentice Hall Englewood Cliffs, NJ, 1991.

Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. G. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3165–3176. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/finzi20a.html.

Geiger, M. and Smidt, T. e3nn: Euclidean neural networks, 2022. URL https://arxiv.org/abs/2207.09453.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Jing, B., Eismann, S., Suriana, P., Townshend, R. J., and Dror, R. Learning from protein structure with geometric vector perceptrons. *arXiv preprint arXiv:2009.01411*, 2020.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020.

Le, T., Noe, F., and Clevert, D.-A. Representation learning on biomolecular structures using equivariant graph attention. In *The First Learning on Graphs Conference*, 2022. URL https://openreview.net/forum?id=kv4xUo5Pu6.

Li, J., Harabor, D., Stuckey, P. J., Ma, H., Gange, G., and Koenig, S. Pairwise symmetry reasoning for multi-agent path finding search. *Artificial Intelligence*, 301:103574, 2021.

Liu, I.-J., Yeh, R. A., and Schwing, A. G. Pic: permutation invariant critic for multi-agent deep reinforcement learning. In *Conference on Robot Learning*, pp. 590–602. PMLR, 2020.

Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Abbeel, O. P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pp. 6379–6390, 2017.

Mondal, A. K., Jain, V., Siddiqi, K., and Ravanbakhsh, S. Eqr: Equivariant representations for data-efficient reinforcement learning. In *International Conference on Machine Learning*, pp. 15908–15926. PMLR, 2022.

Nguyen, D. T., Kumar, A., and Lau, H. C. Policy gradient with value function approximation for collective multiagent planning. *Advances in neural information processing systems*, 30, 2017.

Nguyen, H. H., Baisero, A., Klee, D., Wang, D., Platt, R., and Amato, C. Equivariant reinforcement learning under

partial observability. In *7th Annual Conference on Robot Learning*, 2023.

Perez, L. and Wang, J. The effectiveness of data augmentation in image classification using deep learning, 2017.

Rashid, T., Farquhar, G., Peng, B., and Whiteson, S. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 33: 10199–10210, 2020a.

Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1):7234–7284, 2020b.

Ravindran, B. and Barto, A. G. Symmetries and model minimization in markov decision processes, 2001.

Rezaei-Shoshtari, S., Zhao, R., Panangaden, P., Meger, D., and Precup, D. Continuous mdp homomorphisms and homomorphic policy gradient. *Advances in Neural Information Processing Systems*, 35:20189–20204, 2022.

Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G. J., Hung, C.-M., Torr, P. H. S., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge, 2019.

Satorras, V. G., Hoogeboom, E., and Welling, M. E(n) equivariant graph neural networks. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9323–9332. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/satorras21a.html.

Schütt, K., Unke, O., and Gastegger, M. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning*, pp. 9377–9388. PMLR, 2021.

Shapley, L. S. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Van der Pol, E., Worrall, D., van Hoof, H., Oliehoek, F., and Welling, M. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4199–4210, 2020.

van der Pol, E., van Hoof, H., Oliehoek, F. A., and Welling, M. Multi-agent MDP homomorphic networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=H7HDG--DJF0.

Wang, D., Walters, R., and Platt, R. So (2)-equivariant reinforcement learning. In *International Conference on Learning Representations*, 2022a.

Wang, D., Walters, R., and Platt, R. SO(2)-equivariant reinforcement learning. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=7F9cOhdvfk_.

Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., and Wang, J. Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 5571–5580. PMLR, 2018.

Yarats, D., Kostrikov, I., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=GY6-6sTvGaf.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., and Wu, Y. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

Yu, X., Shi, R., Feng, P., Tian, Y., Luo, J., and Wu, W. Esp: Exploiting symmetry prior for multi-agent reinforcement learning. In *ECAI 2023*, pp. 2946–2953. IOS Press, 2023.

Zhao, B., Dehmamy, N., Walters, R., and Yu, R. Symmetry teleportation for accelerated optimization. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022a. URL https://openreview.net/forum?id=MHjxpvMzf2x.

Zhao, B., Ganev, I., Walters, R., Yu, R., and Dehmamy, N. Symmetries, flat minima, and the conserved quantities of gradient flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9ZpciCOunFb.

Zhao, L., Zhu, X., Kong, L., Walters, R., and Wong, L. L. Integrating symmetry into differentiable planning with steerable convolutions. In *The Eleventh International Conference on Learning Representations*, 2022b.

A. Proof of Theorem 4.4

We first prove property (iv) in Section A.1 and then properties (i), (ii), and (iii) in Section A.2.

A.1. Proof of Property (iv)

This property can be directly derived by the definition of G-invariant MG policies. For state-based G-invariant policy π:

$$\nabla_\theta \pi_\theta(a|s) = \lim_{\Delta\theta\to 0} \frac{\pi_{\theta+\Delta\theta}(a|s) - \pi_\theta(a|s)}{\Delta} = \lim_{\Delta\theta\to 0} \frac{\pi_{\theta+\Delta\theta}(K_g^s[a] \mid L_g[s]) - \pi_\theta(K_g^s[a] \mid L_g[s])}{\Delta} = \nabla_\theta \pi_\theta(K_g^s[a] \mid L_g[s])$$

where the second equality holds because we assume π_θ is G-invariant for any θ. The similar statement holds for observation-based G-invariant policy ν.

A.2. Proof of Properties (i), (ii), and (iii)

These properties can be established by extending the notion of single-agent MDP homomorphism and its properties to MGs.

Definition A.1 (MG homomorphism)**.** An MG homomorphism (l, k_s, h_s) is a surjective map from an MG $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, r, o\rangle$ onto an abstract MG $\langle \mathcal{N}, \overline{\mathcal{S}}, \overline{\mathcal{A}}, \overline{P}, \overline{r}, \overline{o}\rangle$ by surjective maps $l : \mathcal{S} \to \overline{\mathcal{S}}$, $k_s : \mathcal{A} \to \overline{\mathcal{A}}$, and $h_s : \mathcal{O} \to \overline{\mathcal{O}} = \overline{\mathcal{O}}^1 \times \cdots \times \overline{\mathcal{O}}^N$, such that

$$r(s,a) = \overline{r}(l(s), k_s(a)) \qquad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{2}$$

$$P([s']_l|s,a) = \overline{P}(l(s')|l(s), k_s(a)) \qquad \forall s, s' \in \mathcal{S}, a \in \mathcal{A} \tag{3}$$

$$o(l(s)) = h_s(o(s)) \qquad \forall s \in \mathcal{S}. \tag{4}$$

Definition A.2 (Policy lifting in MG homomorphisms)**.** We define policy lifting for state-based and observation-based policies separately, assuming full observability:

- State-based MG policy lifting. Let $\pi_\uparrow : \mathcal{S} \to \Delta(\mathcal{A})$ and $\overline{\pi} : \overline{\mathcal{S}} \to \Delta(\overline{\mathcal{A}})$ be two state-based policies in the actual and abstract MGs, respectively. We say $\pi_\uparrow$ is a lift of $\overline{\pi}$, denoted as $\pi_\uparrow = \texttt{lift}(\overline{\pi})$, if $\sum_{a\in k_s^{-1}(\overline{a})} \pi_\uparrow(a|s) = \overline{\pi}(\overline{a}|l(s))$ for any $s \in \mathcal{S}$ and $\overline{a} \in \overline{\mathcal{A}}$.
- Observation-based MG policy lifting. Let $\nu_\uparrow = \{\nu_\uparrow^i : \mathcal{O}^i \to \Delta(\mathcal{A}^i)\}_{i\in\mathcal{N}}$ and $\overline{\nu} = \{\overline{\nu}^i : \overline{\mathcal{O}}^i \to \Delta(\overline{\mathcal{A}}^i)\}_{i\in\mathcal{N}}$ be two observation-based policies in the actual and abstract MGs, respectively. We say $\nu_\uparrow$ is a lift of $\overline{\nu}$, denoted as $\nu_\uparrow = \texttt{lift}(\overline{\nu})$, if $\sum_{a\in k_s^{-1}(\overline{a})} \nu_\uparrow(a|o(s)) = \overline{\nu}(\overline{a}|h_s(o(s)))$ for any $s \in \mathcal{S}$ and $\overline{a} \in \overline{\mathcal{A}}$.

Theorem A.3 (Value equivalence in MG homomorphisms)**.** *Consider an MG homomorphism (l, k_s, h_s) from MG $M = \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, r, o\rangle$ with bijective observation functions to abstract MG $\overline{M} = \langle \mathcal{N}, \overline{\mathcal{S}}, \overline{\mathcal{A}}, \overline{P}, \overline{r}, \overline{o}\rangle$. We have*

$$V_*(s) = \overline{V}_*(l(s)), \quad Q_*(s,a) = \overline{Q}_*(l(s), k_s(a)) \quad \textit{for any } s \in \mathcal{S}, a \in \mathcal{A}. \tag{5}$$

For a state-based abstract policy $\overline{\pi}$ with its lift $\pi_\uparrow$, we have

$$V_{\pi_\uparrow}(s) = \overline{V}_{\overline{\pi}}(l(s)), \quad Q_{\pi_\uparrow}(s,a) = \overline{Q}_{\overline{\pi}}(l(s), k_s(a)) \quad \textit{for any } s \in \mathcal{S}, a \in \mathcal{A}. \tag{6}$$

For an observation-based abstract policy $\overline{\nu}$ with its lift $\nu_\uparrow$, we have

$$V_{\nu_\uparrow}(o(s)) = \overline{V}_{\overline{\nu}}(h_s(o(s))), \quad Q_{\nu_\uparrow}(s,a) = \overline{Q}_{\overline{\nu}}(h_s(o(s)), k_s(a)) \quad \textit{for any } s \in \mathcal{S}, a \in \mathcal{A}. \tag{7}$$

Proof of Theorem A.3. For state-based policies, we use similar proof techniques for single-agent MDP homomorphisms (Ravindran & Barto, 2001; Rezaei-Shoshtari et al., 2022). Under bijective observation functions, we can prove (7) by first translating them into state-based policies and then applying (6). □

Proof of properties (i) and (iii). We first establish that a G-symmetric MG induces a MG homomorphism, where G-invariant MG policy can be viewed as lifted from the induced abstract MG. Then, properties (i) and (iii) directly follow from Theorem A.3.

Specifically, consider a G-symmetric MG as defined in Definition 4.1. Define maps (l, k_s, h_s) as follows, such that the state-action-observation tuples in the same orbit under G map to the same abstraction:

$$l(s) = l(L_g[s]), \quad k_s(a) = k_{L_g[s]}(K^s_g[a]), \quad h_s(o(s)) = h_{L_g[s]}(o(L_g[s])) = h_{L_g[s]}(H^s_g[o(s)])$$

for any $s \in \mathcal{S}, a \in \mathcal{A}, g \in G$. It is easy to verify that maps (l, k_s, h_s) are surjective and satisfy conditions (2), (3), and (4) in Definition A.1, and therefore maps (l, k_s, h_s) induce an MG homomorphism. Property (i) then directly follows from (5) and the assumption that observation functions are bijective.

Under this an MG homomorphism, a state-based G-invariant policy induces an abstract policy, $\pi(a|s) = \pi(K^s_g[a] \mid L_g[s]) = \overline{\pi}(\overline{a}|\overline{s})$, where $(\overline{s}, \overline{a}) = (l(s), k_s(a)) = (l(L_g[s]), k_{L_g[s]}(K^s_g[a]))$. Because $K^s_g[\cdot]$ is reversible via $K^s_{g^{-1}}[\cdot]$, k_s is actually bijective. To see this, we can pick an arbitrary state-action pair in every state-action orbit as the canonical state-action pair for that orbit, then the mapping of any state-action pair can be done via two steps, from the state-action pair to the canonical state-action pair and then to the abstraction, with both steps being bijective. Thus, we have $\sum_{a' \in k_s^{-1}(\overline{a})} \pi(a'|s) = \pi(a|s) = \overline{\pi}(\overline{a}|\overline{s})$, and therefore π is a lift policy from $\overline{\pi}$. Property (iii) for π then directly follows from (6). Under bijective observation functions, we can similarly derive property (iii) for state-based G-invariant policy ν from (7).

Proof of property (ii). Such a state-based policy π can be induced from the optimal G-invariant value function. Letting Q_* be the optimal action-value function that is G-invariant, the existence of which has been proved as property (i), which induces a deterministic optimal state-based policy π_* as the greedy policy with respect to Q_*, i.e., $\pi_*(s) = \arg\max_a Q_*(s, a)$ for any $s \in \mathcal{S}$. Since Q_* is G-invariant, it is easy to see that π_* is also G-invariant, $K^s_g[\pi_*(s)] = \pi_*(L_g[s])$. Such an observation-based policy ν can be similarly derived.

B. Examples of Group-Symmetric MGs

B.1. $\mathrm{E}(3)$-Symmetric MGs

It is quite common that in the 3D physical world, entities' positions and some other physical quantities are described with respect to an arbitrary reference whose change has no impact on the behaviors of the entities. Such scenarios can be described as an $\mathrm{E}(3)$-symmetric MGs. Examples include traffic with vehicles, team sports, surveillance with drones, and games such as SMAC and MPE.

Below we highlight three MARL benchmark games to show the generality of $\mathrm{E}(3)$-symmetric MGs.

Multi-Agent MuJoCo Tasks. In a Multi-Agent MuJoCo task, a given single robotic agent is split into several sub-parts (agents). Each (sub-part) agent can be viewed as a point in the 3D point cloud, with the its centroid as the 3D position. The $\mathrm{E}(3)$-equivariant features can be velocities, forces, etc, and the $\mathrm{E}(3)$-intvariant features can be id, unit_type, etc. The actions are the forces applied to each sub-parts.

Google Research Football (GRF) In GRF, each unit in the field, such as a player and a ball can be viewed as a point in the 3D point cloud, with $\mathrm{E}(3)$-equivariant features such as the velocity and $\mathrm{E}(3)$-invariant features such as the unit_type. The actions are categorical, with movement actions and some other non-movement actions such as shooting and passing. The non-movement actions are $\mathrm{E}(3)$-invariant. If the player can move to any direction, i.e., the movement actions are continuous, then the game satisfies the requirement of the action space in (ii). The scenarios in GRF are competitive where the (uncontrollable) enemies are viewed as part of the environment. The game is an $\mathrm{E}(3)$-symmetric MGs if the enemies' policies are $\mathrm{E}(3)$-invariant.

SMAC In SMAC, each unit in the combat can be view as a point in the 3D point cloud, with dummy z coordinate. There are some competitive scenarios in SMAC, where the controllable agents are fighting against some (uncontrollable) enemies which are viewed as part of the environment. Therefore, similar to GRF, whether the game is an $\mathrm{E}(3)$-symmetric MGs or not depends on the $\mathrm{E}(3)$-invariancy of the uncontrollable enemies' policies. The actions are also categorical including movement actions and some other $\mathrm{E}(3)$-invariant non-movement actions such as attacking.

B.2. S_N-Symmetric MGs

Besides the Euclidean transformation, the permutation is another common transformation in the physical world. Intuitively, for homogeneous entities with the same exact behaviors, permute them will have no impact. Such multi-agent symmetries can be captured by the symmetric group S_N with permutations as the group actions. The Homogeneous MGs, which is S_N-symmetric MGs under our definition, is formallly defined by the work (Chen et al., 2022) with the permutation

transformation defined on state, action, observation, and the transition and the reward function. Enforcing permutation-invariancy has been found beneficial by (Chen et al., 2022; Liu et al., 2020). Our definition of the G-symmetric MGs is general enough to cover it.

The examples include MPE tasks with homogeneous agents, SMAC scenarios with homogeneous ally units, team sports with homogeneous players (assume players have the same capabilities), traffic with homogeneous vehicles, and surveillance with homogeneous drones.

C. Experiment Details

Below we describe in general notation of states and observations in scenarios in MPE and SMAC.

C.1. MPE

Multi-Agent Particle Environment (MPE) with the efficient implementation by (Liu et al., 2020) is a classical benchmark with homogeneous agents for multi-agent reinforcement learning, each of which has versions with $N = 3, 6$ agents, respectively. These MPE environments can be cast as $\mathrm{E}(3)$-Symmetric MGs, where we have dummy z coordinate. Below we describe the general form of state and observation in all scenarios of MPE, and the corresponding state-action based point cloud and observation-based point cloud which are processed by SEGNN-based critic and actor, respectively, and also the details of each scenarios in MPE.

Notation We denote the set of agents, landmarks, and preys, which are basic units in scenarios of MPE, as $\mathcal{N}^a = \{1, ..., N^a\}$, $\mathcal{N}^l = \{N^a + 1, ..., N^l\}$, and $\mathcal{N}^p = \{N^a + N^l + 1, ..., N + N^l + N^p\}$, where N^a, N^l, and N^p are the number of agents, landmarks and preys, respectively. Here only the agents are controllable so $N^a = N$. We have the set of nodes $\mathcal{V} = \mathcal{N}^a \cup \mathcal{N}^l \cup \mathcal{N}^p$. The absolute positions associated with $\mathcal{N}^a, \mathcal{N}^l$ and $\mathcal{N}^p$ are $\{\mathbf{x}_i\}_{i\in\mathcal{N}^a}, \{\mathbf{x}_v\}_{v\in\mathcal{N}^l}$ and $\{\mathbf{x}_v\}_{v\in\mathcal{N}^p}$, respectively. The absolute velocities associated with $\mathcal{N}^a$ and $\mathcal{N}^p$ are $V^a = \{\mathbf{v}_i\}_{i\in\mathcal{N}^a}$ and $V^p = \{\mathbf{v}_v\}_{v\in\mathcal{N}^p}$, respectively. The landmarks are fixed, so the associate velocities is $V^l = \{\mathbf{0}\}_{i\in\mathcal{N}^l}$. Let $V = V^a \cup V^l \cup V^p$, which is the set of velocities associated with the set of nodes $\mathcal{V}$.

State The state consists of the positions $\mathbf{x}$ of the set of entities $\mathcal{N}$, with the associated features $\mathbf{f} = \{\mathbf{f}_v = \mathbf{v}_v\}_{v\in\mathcal{V}}$. The state is therefore a point cloud $\{(\mathbf{x}_v, \mathbf{f}_v)\}_{v\in\mathcal{V}}$, which satisfies the requirement (i) of $\mathrm{E}(3)$-symmetric MGs.

Observation The agent i's local observation consists of the relative positions $\mathbf{x}^i = \{\mathbf{x}^i_i = \mathbf{x}_i\} \cup \{\mathbf{x}^i_v = \mathbf{x}_v - \mathbf{x}_i\}_{v\in\mathcal{N}\setminus\{i\}}$ of the set of entities $\mathcal{N}$, with the associated (relative) features $\mathbf{f}^i = \{\mathbf{f}^i_i = \mathbf{v}_i\} \cup \{\mathbf{f}^i_v = \mathbf{0}\}_{v\in\mathcal{V}\setminus\{i\}}$. (Other agents' velocities are not observable). The agent i's local observation is therefore a point cloud $\{(\mathbf{x}^i_v, \mathbf{f}^i_v)\}_{v\in\mathcal{V}}$ from agent i's perspective, which satisfies the requirement (iii) of $\mathrm{E}(3)$-symmetric MGs.

Action The local action spaces are absolute velocities, and are therefore Euclidean spaces which satisfies the requirement of (ii). Denote the set of actions for the controllable agents as $A^a = \{\mathbf{a}_i\}_{i\in\mathcal{N}^a}$.

The details of each scenarios in MPE is as the following.

Cooperative Navigation: The collective goal of agents is to cover all the landmarks. There are two versions with $N^l = 3, 6$ landmarks for $N = 3, 6$ agents, respectively. There are no preys so $N^p = 0$.

Cooperative Push: The collective goal of agents is to push a large ball to a target position. There are two versions with $N^l = 2, 2$ landmarks for $N = 3, 6$ agents, respectively. There are no preys so $N^p = 0$.

Predator-and-Prey: The collective goal of slowly moving agents (predators) is to capture some fast moving preys. The preys are pre-trained and controlled by the environment. There are $N^p = 1, 2$ preys and $N^l = 2, 3$ landmarks (blocks) for $N = 3, 6$ agents (predators), respectively.

C.2. MuJoCo Continuous Control Tasks

We choose several representative tasks with different levels of symmetries from the MuJoCo continuous control tasks, including the 2D ones from (Tassa et al., 2018) and 3D ones from (Chen et al., 2023) with single- and multi-agent variations: cartpole (balance, sparse), single- and multi-agent reacher (hard), single- and multi-agent swimmer with two links, and multi-agent variants of 3D hopper and walker. The details of the state, observation, and action in the selected scenarios are as the following.

Cartpole (balance, sparse): the goal of cartpole is to balance a pole connecting to cart moving in the x axis. It has rotation-equivariancy of rot_{180° with respect to the z axis and no translation-invariancy along the x axis because the agent is tasked to balance the cart and the pole around the origin ($x = 0$), not around some arbitrary position. The state contains the position of the cart, pole, and the origin, and the velocity of the cart and the pole. This task is fully observable, so the observation is the same as the state. The action is the 1D force acted on the cart on the x (horizontal) axis.

Single-agent reacher (hard): the goal of single-agent reacher is to control a robot with two links to reach a target. The second link (root, arm) can move around a hinge fixed at the root (origin), and the first link (hand, finger) can move around a hinge at the end of the second link. This task has rotation-equivariancy in the x, y plane, and no translation-invariancy due to the hinge fixed at the origin. The state contains the positions of the target, finger, hand, arm, and root, and the velocities of the finger, hand, arm, and root. This task is fully observable, so the observation is the same as the state. The action is the torques applied to the two links. The target is generated randomly between circles whose centers are located at the origin with radius 0.05 and 0.2, respectively.

Multi-agent reacher: the multi-agent reacher has the same goal as the single-agent reacher. The state is the same as the single-agent reacher. This task is partially observable. Agent 1 controlling the the first link can observe its own id, and the positions of the target, finger, and hand, and the velocities of the finger and hand. Agent 2 controlling the second link can observe its own id, and the positions of the target, arm, and root, and the velocities of the arm and root. The actions are the torques applied on the two links for agent 1 and agent 2, respectively.

Single-agent swimmer: the goal of single-agent swimmer is to control a robot with two controllable links to move to a target. It is similar to the single-agent reacher, except that the second link is not fixed at the origin. This task has both rotation- and translation-invariancy in the xy plane. The state is the positions of the target, nose (in the front of the swimmer's head), and the first joint, the velocities and the angular velocities of the swimmer's bodies, and the joint angles. This task is fully observable, so the observation is the same as the state. The actions are the torques applied on the two controllable links. The target is generated randomly inside a square box whose center is located at the origin with width=height=0.3.

Multi-agent swimmer: the multi-agent swimmer has the same goal as the single-agent swimmer. The state is the same as the single-agent swimmer. We split the robot into two agents, with agent 1 controlling the first link, and agent 2 controlling the second link. This task is fully observable, so the observation is the same as the state, except that each agent can also observe their own ids. The actions are the torques applied on the two links for agent 1 and agent 2, respectively.

Multi-agent hopper 3D (3_shin): the goal of multi-agent hopper is to control a robot with 3 body parts (torso, thigh, foot) to reach a target position. This task has both rotation- and translation-invariancy in the xy plane. The state contains the positions of the target, torso, thigh, and foot, the velocities of the torso, thigh, foot, the rotation axes of the thigh and foot, and the gravity which is a constant (0,0,-9.81). Agent 1 control the thigh, and agent 2 control the foot. This task is fully observable, so the local observation is the same as the state, except that each agent can also observe their own ids. The actions are the torques applied on thigh for agent 1 and the torques applied on foot for agent 2, respectively.

Multi-agent walker 3D (3_left_leg_right_foot): the goal of multi-agent walker is to control a robot with 5 body parts (torso, right thigh, right shin, left thigh, left shin) to reach a target position. This task has both rotation- and translation-invariancy in the xy plane. The state contains the positions of the target, torso, right thigh, right shin, left thigh, and left shin, the velocities of the torso, right thigh, right shin, left thigh, and left shin, the rotation axes of the right thigh, right shin, left thigh, and left shin, and the gravity which is a constant (0,0,-9.81). Agent 1 control the right thigh, and agent 2 control the right shin. This task is fully observable, so the local observation is the same as the state, except that each agent can also observe their own ids. The actions are the torques applied on the right thigh for agent 1 and the torques applied on right shin for agent 2, respectively.

C.3. SMAC

The StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) has become one of the most popular MARL benchmarks. All scenarios in SMAC can be described by $\mathrm{E}(3)$-symmetric Markov games in a similar way to MPE. We choose the *Hard* scenario 8m_vs_9m and *Super Hard* scenario 6h_vs_8z to evaluate our proposed algorithm, which has 8 agents and 6 agents, respectively.

Notation We denote the set of controllable ally agents and uncontrollable enemies, which are basic units in scenarios of SMAC, as $\mathcal{N}^a = \{N+1, ..., N^a\}$ and $\mathcal{N}^e = \{N^a+1, ..., N^a+N^e\}$, where N^a and N^e are the number of agents and enemies, respectively. Here only agents are controllable and so $N^a = N$. We have the set of nodes $\mathcal{V} = \mathcal{N}^a \cup \mathcal{N}^e$. The

absolute positions associated with $\mathcal{N}^a$ and $\mathcal{N}^e$ are $\{\mathbf{x}_i\}_{i\in\mathcal{N}^a}$ and $\{\mathbf{x}_v\}_{v\in\mathcal{N}^e}$, respectively.

The local action spaces of the controllable ally agents are categorical, which is the union of the set of move directions, attack enemies, stop and no operation, i.e., $\mathcal{A}^i = \{\text{north, south, east, west}\} \cup \{\text{attack[enemy\_id]}\}_{\text{enemy\_id}\in\mathcal{N}^e} \cup \{\text{stop, no-op}\}$. The health associated with $\mathcal{N}^a$ and $\mathcal{N}^e$ are $H^a = \{\text{h}_i\}_{i\in\mathcal{N}}$ and $H^e = \{\text{h}_v\}_{v\in\mathcal{N}^e}$, respectively. The cooldown of the weapons associated with $\mathcal{N}^a$ and $\mathcal{N}^e$ which is not applicable are $CD^a = \{\text{cd}_i\}_{i\in\mathcal{N}}$ and $CD^e = \{\mathbf{0}\}_{v\in\mathcal{N}^e}$, respectively. The shield associated with $\mathcal{N}^a$ and $\mathcal{N}^e$ are $SH^a = \{\text{sh}_i\}_{i\in\mathcal{N}}$ and $SH^e = \{\text{sh}_v\}_{v\in\mathcal{N}^e}$, respectively. Let $H = H^a \cup H^e$, $CD = CD^a \cup CD^e$, $SH = SH^a \cup SH^e$, which are the sets of health, cooldown, and shield of the allies' and enemies' agents associated with the set of nodes $\mathcal{V}$.

State The state consists of the positions $\mathbf{x}$ of the set of entities $\mathcal{N}$, with the associated features $\mathbf{f} = \{\mathbf{f}_v = (\text{h}_v, \text{cd}_v, \text{sh}_v)\}_{v\in\mathcal{V}}$. The state is therefore a point cloud $\{(\mathbf{x}_v, \mathbf{f}_v)\}_{v\in\mathcal{V}}$, which satisfies the requirement (i) of $\mathrm{E}(3)$-symmetric MGs.

Observation Agent i's observation $o^i(s)$ only contains information of itself and some other observable entities, denoted as $\mathcal{V}^i \subseteq \mathcal{V}$. The information of the non-observable entities are padded as $\mathbf{0}$. From agent i's perspective, denote the visibility of the entities as $VI^i = \{\text{vi}_v = 1\}_{v\in\mathcal{V}^i} \cup \{\text{vi}_v = 0\}_{v\in\mathcal{V}\setminus\mathcal{V}^i}$, and denote the set of (one-hot) move directions $\in \{\text{north, south, east, west}\}$ as $MD^i = \{\text{md}_i\} \cup \{\mathbf{0}\}_{v\in\mathcal{V}\setminus\{i\}}$. (only its own move direction is observable). Then, $o^i(s)$ consists of the relative positions $\mathbf{x}^i = \{\mathbf{x}^i_i = \mathbf{0}\} \cup \{\mathbf{x}^i_v = \mathbf{x}_v - \mathbf{x}_i\}_{v\in\mathcal{N}\setminus\{i\}}$ of the set of entities $\mathcal{N}$, with the associated (relative) features $\mathbf{f}^i = \{f^i_i = (\text{agent\_id} = i, \text{md}_i, \text{vi}_i = 1, \|\mathbf{x}^i_i\| = 0, \mathbf{x}^i_i = \mathbf{0}, \text{h}_i, \text{sh}_i\} \cup \{f^i_v = (\text{agent\_id} = 0, \text{md}_v = \mathbf{0}, \text{vi}_v, \|\mathbf{x}^i_v\|, \mathbf{x}^i_v, \text{h}_v, \text{sh}_v)\}_{v\in\mathcal{V}\setminus\{i\}}$. (Other entities' ids are not observable, so padded as 0). The agent i's local observation is therefore a point cloud $\{(\mathbf{x}^i_v, \mathbf{f}^i_v)\}_{v\in\mathcal{V}}$ from agent i's perspective, which satisfies the requirement (iii) of $\mathrm{E}(3)$-symmetric MGs.

Action Each agent i's action space is categorical, which is the union of the sets of move directions, attack enemies, stop and no operation, i.e., $\mathcal{A}^i = \{\text{north, south, east, west}\} \cup \{\text{attack[enemy\_id]}\}_{\text{enemy\_id}\in\mathcal{N}^e} \cup \{\text{stop, no-op}\}$. Obviously, $\mathcal{A}^i$ is not a Euclidean space and therefore does not satisfy the requirement of (ii).

The details of the two scenarios in SMAC we test our algorithm is as the following.

8m_vs_9m: 8 controllable ally agents of type Marines are fighting against 9 uncontrollable enemies of type Marines. Shield is not applicable for Marines so both SH^a and SH^e are $\emptyset$.

6h_vs_8z: 6 controllable ally agents of type Hydralisks are fighting against 9 uncontrollable enemies of type Zealots. Shield is not applicable for Hydralisks so $SH^a = \emptyset$.

D. Implementation Details

As described in 5.1, the input to SEGNN is an input Euclidean graph is represented as a tuple $G^{\text{in}} = (\mathcal{V}, \mathcal{E}, \mathbf{x}, \mathbf{f}^{\text{in}})$ where $\mathcal{V}$ is a set of vertices with 3D positions $\mathbf{x}_v \in \mathbb{R}^3$, $\mathcal{E} \subseteq \mathcal{V}\times\mathcal{V}$ is a set of edges, and $\mathbf{f}^{\text{in}}$ is a set of feature vectors, each associated with a vertex or an edge. Precisely, the input feature $\mathbf{f}^{\text{in}}$ can be further decomposed into node feature, node attribute, and edge attribute, i.e., $\mathbf{f}^{\text{in}} = (\mathbf{f}^{\text{node}}_{\text{feature}}, \mathbf{f}^{\text{node}}_{\text{attribute}}, \mathbf{f}^{\text{edge}}_{\text{attribute}})$. Note that in MPE and SMAC, entities are in a 2D world, so we pad the dummy z coordinate of 0 for them to be in a 3D space whenever applicable, e.g. velocities, positions, etc.

Below we describe in details, whichever applicable, how to represent state s, state-action pair (s, a), and observation $o^i(s)$ as Euclidean graphs for centralized state-value critic $V(s)$, action-value critic $Q(s, a)$, and observation-based actor $\nu^i(o^i(s))$, respectively, in MPE, MuJoCo tasks, and SMAC.

D.1. MPE

In MPE, we build our architecture based on MADDPG, where the critic is based on state and action pair and therefore we construct state-action based Euclidean graph processed by $\mathrm{E}(3)$-invariant critic. We also construct an observation-based Euclidean graph processed by an $\mathrm{E}(3)$-equivariant actor.

State-action based Euclidean graph Based on the information in the state $s = \{(\mathbf{x}_v, \mathbf{f}_v)\}_{v\in\mathcal{V}}$, $\mathcal{V}$ and $\mathbf{x}$ have already been well-defined. The graph $\mathcal{E}$ is densely connected, i.e., $\mathcal{E} = \{(v_1, v_2) : v_1 \in \mathcal{V}, v_2 \in \mathcal{V}, v_1 \neq v_2\}$. We can pad the dummy $\mathbf{0}$ action for non-controllable entities, i.e., landmarks and preys, to have actions for all entities: $A = \{\mathbf{a}_i\}_{i\in\mathcal{N}^a} \cup \{\mathbf{0}\}_{v\in\mathcal{N}^l} \cup \{\mathbf{0}\}_{v\in\mathcal{N}^p}$. We can then design the node feature $\mathbf{f}^{\text{node}}_{\text{feature}}$, node attribute $\mathbf{f}^{\text{node}}_{\text{attribute}}$, and edge attribute $\mathbf{f}^{\text{edge}}_{\text{attribute}}$ based on the information in the input features $\mathbf{f}^{\text{in}} = \mathbf{f} = \{\mathbf{v}_v\}_{v\in\mathcal{V}}$ and actions A. The node feature of each node consists of its own absolute velocity, the l_2 norm of its own absolute velocity, its own action, and l_2 norm of its own action.

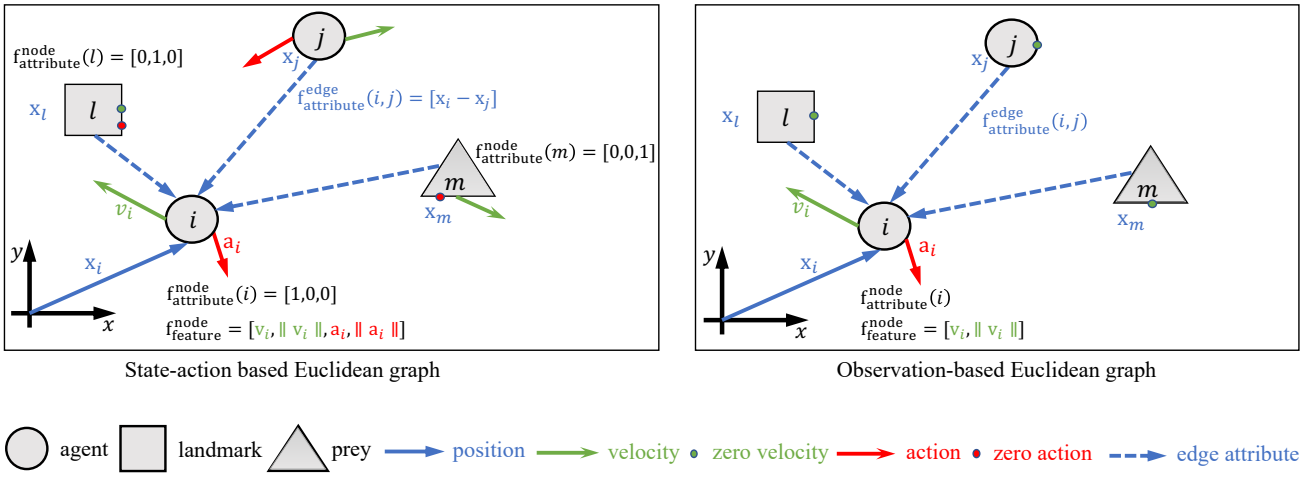

Figure 8: Illustration of state-action based point cloud and observation-based point cloud for a scenario in MPE with agent, landmark and prey.

Specifically, $\mathbf{f}^{\text{node}}_{\text{feature}} = \{[\mathbf{v}_v, \|\mathbf{v}_v\|, \mathbf{a}_v, \|\mathbf{a}_v\|]\}_{v\in\mathcal{V}}$. We can specify the $\mathrm{E}(3)$-equivariancy of both absolute velocities and actions, and $\mathrm{E}(3)$-invariancy of the associated l_2 norms in SEGNN to preserve physical constraints. The node attribute is the set of node_types $\in$ {agent, landmark, prey}, i.e., $\mathbf{f}^{\text{node}}_{\text{attribute}} = \{[\text{node\_type}(v)]\}_{v\in\mathcal{V}}$. The node types are $\mathrm{E}(3)$-invariant. The edge attribute is the set of relative positions, i.e., $\mathbf{f}^{\text{edge}}_{\text{attribute}} = \{[\mathbf{x}_{v_1} - \mathbf{x}_{v_2}]\}_{(v_1,v_2)\in\mathcal{E}}$, which is $\mathrm{E}(3)$-equivariant.

Observation-based Euclidean graph The relative positions in the observation can be restored to the absolute location, i.e., $\mathbf{x}^i = \{\mathbf{x}^i_i = \mathbf{x}_i\} \cup \{\mathbf{x}_v - \mathbf{x}_i\}_{v\in\mathcal{N}\setminus\{i\}} \longrightarrow \{\mathbf{x}^i_i = \mathbf{x}_i\} \cup \{\mathbf{x}_v - \mathbf{x}_i + \mathbf{x}_i\}_{v\in\mathcal{N}\setminus\{i\}}$. Then we can generate the observation-based point cloud in almost the same way, with two differences. The first difference is that due to partial observability, the velocities of other agents and preys are not available, so we have $V^a = \{\mathbf{v}_i\} \cup \{\mathbf{0}\}_{j\in\mathcal{N}\setminus\{i\}}$ and $V^p = \{\mathbf{0}\}_{\mathcal{N}^p}$. Another difference is that the observation-based point cloud does not condition on actions, so correspondingly the node features do not contain action related elements and is only velocity related, i.e., $\mathbf{f}^{\text{node}}_{\text{feature}} = \{[\mathbf{v}_v, \|\mathbf{v}_v\|]\}_{v\in\mathcal{V}}$.

D.2. MuJoCo Continuous Control Tasks

In MuJoCo continuous control tasks, we build our architecture based on MADDPG, where the critic is based on state and action pair and therefore we construct state-action based Euclidean graph processed by $\mathrm{E}(3)$-invariant critic. We also construct an observation-based Euclidean graph processed by an $\mathrm{E}(3)$-equivariant actor. Note that the input to both our architecture and the MLP-based baselines contains the same information. Below we specify the state-action based Euclidean graph and the observation-based Euclidean graph for all the selected tasks. For all the point sets defined below, the graph $\mathcal{E}$ is densely connected, i.e., $\mathcal{E} = \{(v_1, v_2) : v_1 \in \mathcal{V}, v_2 \in \mathcal{V}, v_1 \neq v_2\}$. The node attribute is the set of node_types, where each point in the point set has its own node type in all the selected MuJoCo tasks. The node types are $\mathrm{E}(3)$-invariant. The edge attribute is the set of relative positions, i.e., $\mathbf{f}^{\text{edge}}_{\text{attribute}} = \{[\mathbf{x}_{v_1} - \mathbf{x}_{v_2}]\}_{(v_1,v_2)\in\mathcal{E}}$, which is $\mathrm{E}(3)$-equivariant.

cartpole (Balance Sparse) State-action based Euclidean graph: the point set $\mathcal{V} = \{\text{origin}, \text{cart}\}$. For $\text{point}_{\text{cart}}$, the node feature $\mathbf{f}^{\text{node,cart}}_{\text{feature}} = \{[\mathbf{v}_j, \|\mathbf{v}_j\|]\}_{j\in\{\text{cart,pole}\}} \cup \{[\mathbf{x}_{\text{pole}}, \|\mathbf{x}_{\text{pole}}\|]\} \cup \{\mathbf{a}, \|\mathbf{a}\|\}$. For $\text{point}_{\text{origin}}$, the node feature $\mathbf{f}^{\text{node,origin}}_{\text{feature}} = \{\mathbf{0}\}$. The node attribute is the node_type: $\forall i \in \mathcal{V}, \mathbf{f}^{\text{node}}_{\text{attribute}} = [\text{node\_type}(i)]$. Observation based Euclidean graph: it is the same as the state-action based Euclidean graph, except that action is not included in the node feature.

Single-agent reacher (hard) State-action based Euclidean graph: the point set $\mathcal{V} = \{\text{target}, \text{finger}\}$. For $\text{point}_{\text{finger}}$, the node feature $\mathbf{f}^{\text{node,finger}}_{\text{feature}} = \{[\mathbf{v}_j, \|\mathbf{v}_j\|]\}_{j\in\{\text{finger,hand,arm,root}\}} \cup \{[\mathbf{x}_j, \|\mathbf{x}_j\|]\}_{j\in\{\text{hand,arm,root}\}} \cup \{\mathbf{a}, \|\mathbf{a}\|\}$. For $\text{point}_{\text{target}}$, the node feature $\mathbf{f}^{\text{node,target}}_{\text{feature}} = \{\mathbf{0}\}$. The node attribute is the node_type: $\forall i \in \mathcal{V}, \mathbf{f}^{\text{node}}_{\text{attribute}} = [\text{node\_type}(i)]$. Observation based Euclidean graph: it is the same as the state-action based Euclidean graph, except that action is not included in the node feature.

Multi-agent reacher (hard) The state-action based Euclidean graph is the same as the single-agent reacher. For agent 1 controlling the green part in reacher in Figure 6, the point set is $\mathcal{V} = \{\text{target}, \text{finger}\}$. The node feature $\mathbf{f}^{\text{node,finger}}_{\text{feature}} =$

$\{\mathrm{id}_{\mathrm{agent\_1}}\} \cup \{[\mathbf{v}_{\mathrm{j}}, \|\mathbf{v}_{\mathrm{j}}\|]\}_{\mathrm{j}\in\{\mathrm{finger,hand}\}} \cup \{[\mathbf{x}_{\mathrm{j}}, \|\mathbf{x}_{\mathrm{hand}}\|]\}\} \cup \{\mathbf{a}_1, \|\mathbf{a}_1\|\}$. For $\mathrm{point}_{\mathrm{target}}$, the node feature $\mathbf{f}^{\mathrm{node,target}}_{\mathrm{feature}} = \{\mathbf{0}\}$. For agent 2 controlling the yellow part in reacher in Figure 6, the point set is $v = \{\mathrm{target}, \mathrm{arm}\}$. The node feature $\mathbf{f}^{\mathrm{node,arm}}_{\mathrm{feature}} = \{\mathrm{id}_{\mathrm{agent\_2}}\} \cup \{[\mathbf{v}_{\mathrm{j}}, \|\mathbf{v}_{\mathrm{j}}\|]\}_{\mathrm{j}\in\{\mathrm{arm,root}\}} \cup \{[\mathbf{x}_{\mathrm{j}}, \|\mathbf{x}_{\mathrm{root}}\|]\}\} \cup \{\mathbf{a}_2, \|\mathbf{a}_2\|\}$. For $\mathrm{point}_{\mathrm{target}}$, the node feature $\mathbf{f}^{\mathrm{node,target}}_{\mathrm{feature}} = \{\mathbf{0}\}$.

Single-agent swimmer State-action based Euclidean graph: the point set $\mathcal{V} = \{\mathrm{target}, \mathrm{nose}, \mathrm{first\_joint}\}$. For $\mathrm{point}_{\mathrm{nose}}$, the node feature $\mathbf{f}^{\mathrm{node,nose}}_{\mathrm{feature}} = \{[\mathbf{v}_j, \|\mathbf{v}_j\|, \omega_j, \|\omega_j\|]\}_{j\in\{\mathrm{bodies}\}} \cup \{[\theta_j]\}_{j\in\{\mathrm{joints}\}} \cup \{\mathbf{a}, \|\mathbf{a}\|\}$. For $\mathrm{point}_{\mathrm{target}}$ and $\mathrm{point}_{\mathrm{first\_link}}$, the node features are $\{\mathbf{0}\}$. The node attribute is the node_type: $\forall i \in \mathcal{V}, \mathbf{f}^{\mathrm{node}}_{\mathrm{attribute}} = [\mathrm{node\_type}(i)]$. Observation based Euclidean graph: it is the same as the state-action based Euclidean graph, except that action is not included in the node feature.

Multi-agent swimmer The state-action based Euclidean graph is the same as the single-agent swimmer. The observation-based Euclidean graphs for both agents are the same as the state-action based Euclidean graph, except that action is not included in the node feature and their own ids are included in the node feature which are $\mathrm{E}(3)$-invariant.

Multi-agent hopper State-action based Euclidean graph: the point set $\mathcal{V} = \{\mathrm{target}, \mathrm{torso}, \mathrm{thigh}, \mathrm{foot}\}$. For the for noncontrollable components target and torso, we set the dummy action of $\{\mathbf{0}\}$. For $i \in \mathcal{V}$, the node feature $\mathbf{f}^{\mathrm{node,i}}_{\mathrm{feature}} = [\mathbf{v}_i, \|\mathbf{v}_i\|, \mathbf{x}_i, \|\mathbf{x}_i\|, , \mathrm{rotation\_axis\_i}, \|\mathrm{rotation\_axis\_i}\|, \mathbf{a}_{\mathrm{i}}, \|\mathbf{a}_{\mathrm{i}}\|]$. The node attribute is the node_type: $\forall i \in \mathcal{V}, \mathbf{f}^{\mathrm{node}}_{\mathrm{attribute}} = [\mathrm{node\_type}(i)]$. Observation based Euclidean graph: it is the same as the state-action based Euclidean graph, except that action is not included in the node feature and their own ids are included in the node feature which are $\mathrm{E}(3)$-invariant.

Multi-agent walker State-action based Euclidean graph: the point set $\mathcal{V} = \{$target, torso, right thigh, right shin, left thigh, left shin$\}$. For the for noncontrollable components $\{$target, torso, left thigh, left shin$\}$, we set the dummy action of $\{\mathbf{0}\}$. For $i \in \mathcal{V}$, the node feature $\mathbf{f}^{\mathrm{node,i}}_{\mathrm{feature}} = [\mathbf{v}_i, \|\mathbf{v}_i\|, \mathbf{x}_i, \|\mathbf{x}_i\|, , \mathrm{rotation\_axis\_i}, \|\mathrm{rotation\_axis\_i}\|, \mathbf{a}_{\mathrm{i}}, \|\mathbf{a}_{\mathrm{i}}\|]$. The node attribute is the node_type: $\forall i \in \mathcal{V}, \mathbf{f}^{\mathrm{node}}_{\mathrm{attribute}} = [\mathrm{node\_type}(i)]$. Observation based Euclidean graph: it is the same as the state-action based Euclidean graph, except that action is not included in the node feature and their own ids are included in the node feature which are $\mathrm{E}(3)$-invariant.

D.3. SMAC

In SMAC, we build our architecture based on MAPPO, where the critic is based on state only and therefore we construct state based Euclidean graph processed by $\mathrm{E}(3)$-invariant critic. The action space in SMAC is categorical and therefore we use a traditional MLP-based actor to process the observation.

State based Euclidean graph Based on the information in the state $s = \{(\mathbf{x}_v, \mathbf{f}_v)\}_{v\in\mathcal{V}}$, $\mathcal{V}$ and $\mathbf{x}$ have already been well-defined. The graph $\mathcal{E}$ is a nearest neighbor graph degree k, i.e., $\mathcal{E} = \{(v_1, v_2) : v_1 \in \mathcal{V}, v_2 \in \mathcal{V}, v_1 \in \mathrm{neighbor}(v_2, k)\}$. We can then design the node feature $\mathbf{f}^{\mathrm{node}}_{\mathrm{feature}}$, node attribute $\mathbf{f}^{\mathrm{node}}_{\mathrm{attribute}}$, and edge attribute $\mathbf{f}^{\mathrm{edge}}_{\mathrm{attribute}}$ based on the information in the input features $\mathbf{f}^{\mathrm{in}} = \mathbf{f} = \{\mathbf{v}_v\}_{v\in\mathcal{V}}$. The node feature of each node consists of its health, cooldown, and shield. Specifically, $\mathbf{f}^{\mathrm{node}}_{\mathrm{feature}} = \{[\mathrm{h}_v, \mathrm{cd}_v, \mathrm{sh}_v]\}_{v\in\mathcal{V}}$. All quantities in node features are $\mathrm{E}(3)$-invariant. The node attribute is the set of node_types $\in \{$[Team,Type]$\}_{\mathrm{Team}\in\{\mathrm{ally,\ enemy}\},\mathrm{Type}\in\{\mathrm{Marines,\ Hydralisks,\ Zealots}\}}$, i.e., $\mathbf{f}^{\mathrm{node}}_{\mathrm{attribute}} = \{[\mathrm{node\_type}(v)]\}_{v\in\mathcal{V}}$. The node types are $\mathrm{E}(3)$-invariant. The edge attribute is the set of relative positions, i.e., $\mathbf{f}^{\mathrm{edge}}_{\mathrm{attribute}} = \{[\mathbf{x}_{v_1} - \mathbf{x}_{v_2}]\}_{(v_1,v_2)\in\mathcal{E}}$, which is $\mathrm{E}(3)$-equivariant.

E. Supplementary Results

E.1. Invariancy Measure in MPE

In MPE, the metrics for measuring invariancy for actor and critic in terms of rotation and translation are defined in the following:

$$\text{Invariancy}_{\text{Q}}^{\text{rot}} = -\frac{1}{|A|}\sum_{\alpha\in A}|Q(s,a) - Q(\text{rot}_\alpha[s], \text{rot}_\alpha[a])| \tag{8}$$

$$\text{Invariancy}_{\text{Q}}^{\text{transl}} = -\frac{1}{|L|}\sum_{l\in L}|Q(s,a) - Q(\text{transl}_l[s], \text{transl}_l[a])| \tag{9}$$

$$\text{Invariancy}_{\nu}^{\text{rot}} = \frac{1}{|A|N}\sum_{i\in\mathcal{N}}\sum_{\alpha\in A}\cos\left(\text{rot}_\alpha[\nu^i(o^i(s)],\ \nu^i(o^i(\text{rot}_\alpha[s]))\right) \tag{10}$$

$$\text{Invariancy}_{\nu}^{\text{transl}} = \frac{1}{|L|N}\sum_{i\in\mathcal{N}}\sum_{l\in L}\cos\left(\text{transl}_l[\nu^i(o^i(s)],\ \nu^i(o^i(\text{transl}_l[s]))\right) \tag{11}$$

,where $\text{rot}_\alpha[\cdot]$ and $\text{transl}_l[\cdot]$ performs the rotation by α and translation by l, respectively, and $\cos(\cdot,\cdot)$ measures the cosine similarity. A is the list of angles which we use $[30°, 60° \cdots, 330°]$, L is the list of translations which we use $[(-l_x, 0), (-0.5l_x, 0), (0.5l_x, 0), (l_x, 0)]$ for translations along x axis and $[(0, -l_y), (0, -0.5l_y), (0, 0.5l_y), (0, l_y)]$ for translations along y axis (the size of the map is l_x by l_y, where $l_x = l_y = 1, 1.5, 1$ for the scenarios of push, navigation, and prey, respectively). We calculate these four metrics by averaging over state and observation collected in different time steps in 200 episodes. The ranges of those metrics are not important. By construction, the larger the values of $\text{Invariancy}_{\text{Q}}^{\text{rot}}$, $\text{Invariancy}_{\text{Q}}^{\text{transl}}$, $\text{Invariancy}_{\nu}^{\text{rot}}$, and $\text{Invariancy}_{\nu}^{\text{transl}}$ are, the closer the behaviors the non-SEGNN-based architecture are compared to those of the SEGNN-based architecture, which has the largest possible values for those metrics, in terms of their invariancy to the corresponding transformations.

E.2. The Emergence of Invariancy in MPE.

As illustrated in Figure 9, the GCN-based critic with permutation invariancy has better rotation- and translation-invariancy than the MLP-based one for the whole training process. One possible reason is that for certain configurations of the state, permutation invariancy is a special type of Euclidean invariancy, e.g., 6 agents whose positions are in a circle with 60° between adjacent ones, which displays a rotation-invariancy of rot_{60°. However, there is no increase in rotation- and translation-invariancy for both GCN- and MLP-based critics, which may explain why actors learned with SEGNN-based critics have better performance. On the other hand, the actor learned with a SEGNN-based critic emerges better rotation- and translation-invariancy than the MLP and GCN baselines. The actor learned with a GCN-based critic achieves similar rotation-invariancy but worse translation-invariancy than the one learned with an MLP-based critic. Note that the translation-invariancy for actors is initially high, because in the observation only the agent's own position is absolute and can be modified by translation, whereas all other values are relative and translation-invariant. The randomly initialized actors will therefore output similar actions.

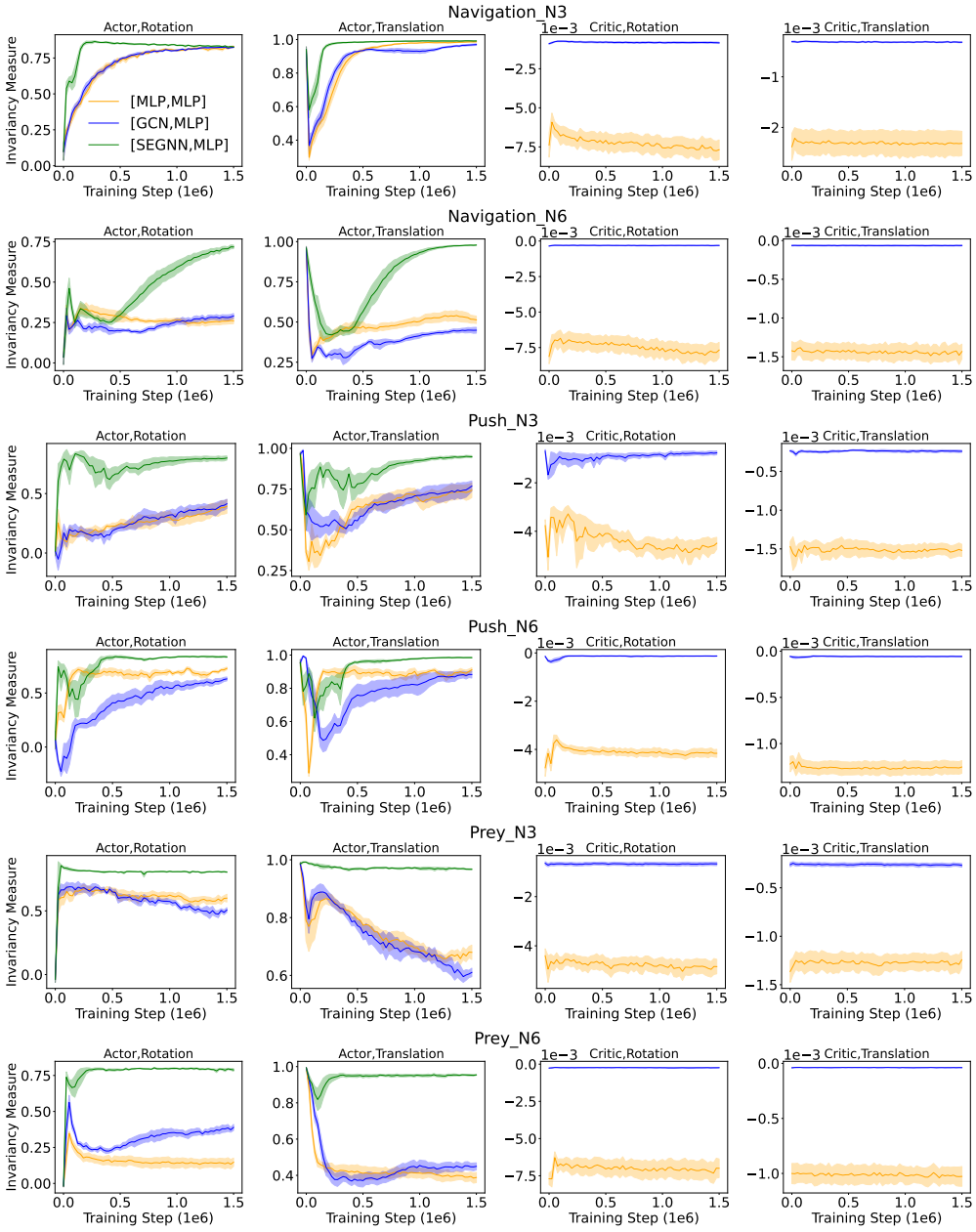

Figure 9: The emergence of invariancy and equivariancy in non-SEGNN-based critic and actor, respectively, in MPE.

E.3. Invariancy Measure in SMAC

The critic used by MAPPO is a value function, so the invariancy for critic is defined in terms of V instead of Q. The action of controllable ally agents in SMAC is categorical, so we use KL divergence instead of cosine similarity to measure the difference between two probability distributions. The action is the union of the set of move directions, attack enemies, stop and no operation, i.e., $A^a = \{\text{north, south, east, west}\} \cup \{\text{attack[enemy\_id]}\}_{\text{enemy\_id}\in\mathcal{N}^e} \cup \{\text{stop, no-op}\}$. Under rotation, only move directions will be changed. Its hard to measure how the move direction will change for an arbitrary rotation angle, so we only consider (counter clockwise) rotation angles of $[90°, 180°, 270°]$, which will roll the move direction accordingly. For example, for a (counter clockwise) rotation angle of $90°$, move to the north will change to move to the west. The translation for actor is already achieved in scenarios in SMAC due to the missing of absolute positions in the observation, so it is not included here.

$$\text{Invariancy}_{\text{V}}^{\text{rot}} = -\tfrac{1}{|A|}\textstyle\sum_{\alpha\in A} |V(s) - V(\text{rot}_\alpha[s])| \tag{12}$$

$$\text{Invariancy}_{\text{V}}^{\text{transl}} = -\tfrac{1}{|L|}\textstyle\sum_{l\in L} |V(s) - V(\text{transl}_l[s])| \tag{13}$$

$$\text{Invariancy}_{\nu}^{\text{rot}} = \tfrac{1}{|A|N}\textstyle\sum_{i\in\mathcal{N}}\sum_{\alpha\in A} \text{KL}\left(\text{rot}_\alpha[\nu^i(\cdot|o^i(s)],\ \nu^i(\cdot|o^i(\text{rot}_\alpha[s]))\right) \tag{14}$$

, where $\text{rot}_\alpha[\cdot]$ and $\text{transl}_l[\cdot]$ performs the rotation by α and translation by l, respectively, and $\text{KL}(\cdot,\cdot)$ measures the KL divergence. A is the list of angles which we use $[90°, 180°, 270°]$, L is the list of translations which we use $[(-l_x, 0), (-0.5l_x, 0), (0.5l_x, 0), (l_x, 0)]$ for translations along x axis and $[(0, -l_y), (0, -0.5l_y), (0, 0.5l_y), (0, l_y)]$ for translations along y axis (the normalized size of the map is l_x by l_y, where $l_x = l_y = 1$ for the scenarios in SMAC. We calculate these four metrics by averaging over state and observation collected in different time steps in 32 episodes. The ranges of those metrics are not important. By construction, the larger the values of $\text{Invariancy}_{\text{V}}^{\text{rot}}$, $\text{Invariancy}_{\text{V}}^{\text{transl}}$, and $\text{Invariancy}_{\nu}^{\text{rot}}$ are, the closer the behaviors the non-SEGNN-based architecture are compared to those of the SEGNN-based architecture, which has the largest possible values for those metrics, in terms of their invariancy to the corresponding transformations.

E.4. The Emergence of Invariancy and Equivariancy in SMAC.

As illustrated in Figure 10, there is no obvious emergence of the rotation-invariancy for both actor and critic. This can be caused by the competitive nature of the scenarios in SMAC, where we also see a decrease in actors' translation-invariancy in competitive scenarios Prey_N3 and Prey_N6, as shown in the appendix. Another possible reason is that some of the categorical actions, the move directions {north, south, east, west}, are only rotation-equivariant to multiples of $90°$ instead of arbitrary angles and therefore break the overall invariancy of the game.

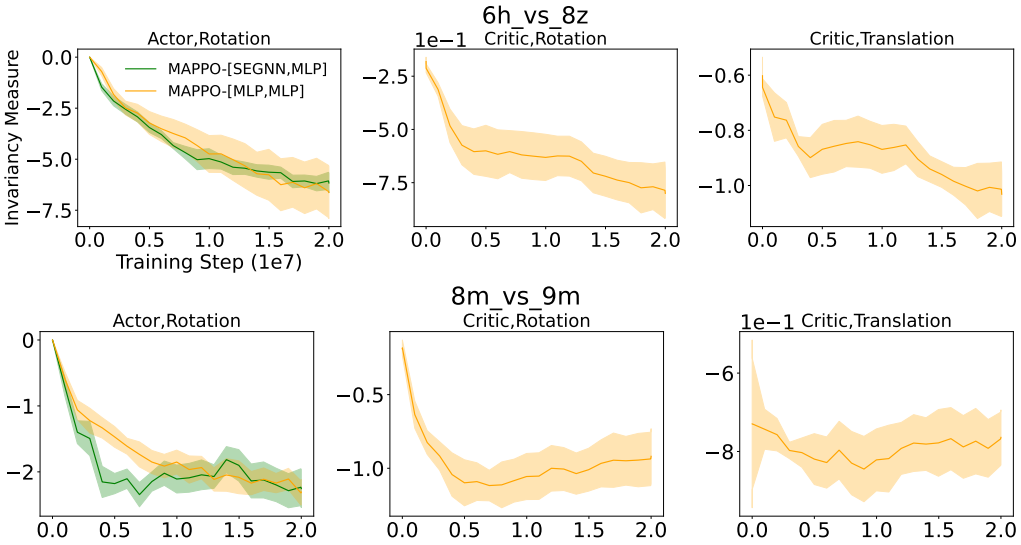

Figure 10: The emergence of invariancy and equivariancy in non-SEGNN-based critic and actor, respectively, in SMAC.

F. Hyperparameters

F.1. MPE

Table 1: Hyperparameters for MPE

Hyperparameter	Value
Episode length	25
Number of training episodes	60000
Discount factor	0.95
Batch size from replay buffer for [MLP, MLP] and [GCN, MLP]	1024
Batch size from replay buffer for [SEGNN, MLP] and [SEGNN, SEGNN]	128
Actor's learning rate for [MLP, MLP] and [GCN, MLP]	1e-4
Acror's learning rate for [SEGNN, MLP] in ordered scenarios	3e-4,3e-4,1e-4,3e-4,3e-4,3e-4
Acror's learning rate for [SEGNN, SEGNN] in ordered scenarios	1e-4,3e-4,3e-5,3e-5,3e-4,3e-4
Critic's learning rate for [MLP, MLP] and [GCN, MLP]	1e-3
Critic's learning rate for [SEGNN, MLP] in ordered scenarios	1e-3,3e-4,1e-3,1e-3,1e-3,1e-3
Critic's learning rate for [SEGNN, SEGNN] in ordered scenarios	1e-3,1e-3,1e-3,1e-3,1e-3,1e-3
Actor's and critic's learning rates for transfer learning in Navigation	2e-4,1e-3
Actor's and critic's learning rates for transfer learning in Push	3e-5,1e-3
Actor's and critic's learning rates for transfer learning in Prey	3e-4,1e-3
Graph used by SEGNN for 6h_vs_8z	complete graph
#episodes per evaluation	200
#seeds	5

The ordered scenarios are: Navigation_N3, Navigation_N6, Push_N3, Push_N6, Tag_N3, Tag_N6.
Unless specified, the mentioned hyperparameter is applied to all scenarios in MPE.
Actors' learning rates are searched in [1e-4, 3e-5, 1e-5].
Critics' learning rates are searched in [1e-3, 3e-4, 1e-4].

F.2. The MuJoCo Continuous Control Tasks

Table 2: Hyperparameters for 2D tasks (cartpole, reacher, swimmer) in the MuJoCo tasks

Hyperparameter	Value
Episode length	1000
Number of training steps	1.1e6
Discount factor	0.99
Batch size	256
Acror's learning rate for [MLP, MLP] and [GCN, MLP]	1e-4
Acror's learning rate for [SEGNN, SEGNN]	5e-5
Critic's learning rate for [MLP, MLP] and [GCN, MLP]	1e-4
Critic's learning rate for [SEGNN, SEGNN]	5e-5
Graph used by SEGNN in all 2D tasks	complete graph
#episodes per evaluation for cartpole (balance, sparse)	20
#episodes per evaluation for single- and multi-agent reacher	10
#episodes per evaluation for single- and multi-agent swimmer	100
#seeds for cartpole (balance, sparse), single-agent reacher, and single-agent swimmer	5
#seeds for multi-agent reacher, and multi-agent swimmer	10

Unless specified, the mentioned hyperparameter is applied to all the tasks and the algorithms.
The hyperparameters except SEGNN's learning rates are the default ones used by (Rezaei-Shoshtari et al., 2022)

Table 3: Hyperparameters for 3D tasks (hopper, walker) in the MuJoCo tasks

Hyperparameter	Value
Episode length	1000
Number of training steps	1.1e6
Discount factor	0.99
Batch size	256
Acror's learning rate	3e-4
Critic's learning rate	3e-4
Graph used by SEGNN in all 3D tasks	complete graph
#episodes per evaluation	10
#seeds	5

Unless specified, the mentioned hyperparameter is applied to all the tasks and the algorithms. The hyperparameters are the default ones used by (Chen et al., 2023)

F.3. SMAC

Table 4: Hyperparameters for SMAC

Hyperparameter	Value
Episode length	400
Number of training steps	2e7
Discount factor	0.99
#Rollout threads	8
#Training threads	1
PPO epoch for 8m_vs_9m, 6h_vs_8z	5,10
# mini-batch for 8m_vs_9m, 6h_vs_8z	1,4
The degree of nearest neighbor graph used by SEGNN for 8m_vs_9m	4
The degree of nearest neighbor graph used by SEGNN for 6h_vs_8z	13 (complete graph)
Acror's learning rate for all algorithms	5e-4
Critic's learning rate for all algorithms	5e-4
#episodes per evaluation	32
#seeds	5

Unless specified, the mentioned hyperparameter is applied to both 6h_vs_8z and 8m_vs_9m. The hyperparameters are the default ones used by the original MAPPO (Yu et al., 2022)

G. Number of Parameters in Neural Networks

G.1. MPE

MPE						
Alg/Env	Navigation_N3	Navigation_N6	Push_N3	Push_N6	Prey_N3	Prey_N6
MLP (critic)	22913	38273	22145	32129	23681	39809
GCN (critic)	37505	40577	36993	38529	38017	41089
SEGNN (critic)	33791	33791	42344	42344	42344	42344
MLP (actor)	18690	20226	18434	19202	18946	20482
SEGNN (actor)	40553	40553	48532	48532	48532	48532

G.2. The MuJoCo 2D Continuous Control Tasks

The MuJoCo 2D continuous control tasks					
Alg/Env	Cartpole	Reacher	Multi-agent Reacher	Swimmer	Multi-agent Swimmer
MLP (critic)	70401	74497	74497	71937	71937
GCN (critic)	37505	40065	40065	38529	38529
SEGNN (critic)	33911	34169	34169	43193	43193
MLP (actor)	70145	74242	71425	71682	71937
GCN (actor)	36993	39426	38273	37890	38273
SEGNN (actor)	32694	39234	34003	48971	43124

G.3. The MuJoCo 3D Continuous Control Tasks

The MuJoCo 3D continuous control tasks		
Alg/Env	Hopper	Walker
MLP (critic)	81665	90881
GCN (critic)	72705	73217
SEGNN (critic)	51781	69329
MLP (actor)	80387	88838
GCN (actor)	72963	74246
SEGNN (actor)	67511	134508

G.4. SMAC

SMAC		
Alg/Env	6h_vs_8z	8m_vs_9m
MLP (critic)	12529	12727
SEGNN (critic)	33811	33765

H. Computing Resources

The code is implemented by PyTorch, and runs on NVIDIA Tesla V100 GPUs with 32 CPU cores. For MPE, a single run with [MLP, MLP], [GCN, MLP], [SEGNN, MLP], [SEGNN, SEGNN] takes approximately 2 hours, 3 hours, 7 hours, 10 hours to run, respectively. For the MuJoCo continuous control tasks, [MLP, MLP] takes approximately 3 hours to run, [GCN, MLP] takes approximately 5 hours to run, and [SEGNN, SEGNN] takes approximately 4 days to run. For SMAC, [MLP, MLP] and [SEGNN, MLP] takes approximately 8 hours and 4 days to run, respectively.