PEnGUiN: Partially Equivariant Graph NeUral Networks for Sample Efficient MARL

Anonymous authors
Paper under double-blind review

Keywords: sample efficiency, reinforcement learning, symmetry, equivariance, geometric guarantees, inductive bias

Summary

Equivariant Graph Neural Networks (EGNNs) excel at Multi-Agent Reinforcement Learning (MARL) problems by harnessing symmetries in observations, but struggle in real-world environments where symmetries may be broken to varying degrees. We introduce *Partially Equivariant Graph Neural Networks (PEnGUiN)*, a novel architecture that learns to exploit partial symmetries. PEnGUiN blends equivariant and non-equivariant updates via a learnable parameter, adapting to the degree and type of symmetry present and bridging the gap between fully equivariant and non-equivariant models. In addition, we formalize types of partial equivariance common to real-world environments (subgroup, feature-wise, subspace, and approximate). Experiments on MARL benchmarks demonstrate PEnGUiN's superior performance and robustness compared to EGNNs and GNNs in asymmetric settings. PEnGUiN learns where equivariance holds, improving applicability to real-world MARL problems.

Contribution(s)

1. We present the first generalization of Equivariant Graph Neural Networks (EGNN) to Partial Equivariance with our novel neural network architecture Partially Equivariant Graph Neural Networks (PEnGUiN). We show theoretically that PEnGUiN unifies fully equivariant (EGNN) and non-equivariant (GNN) representations within a single architecture, controlled by a learnable parameter called the symmetry score.
Context: PEnGUiN builds on EGNN (Satorras et al., 2021) and E2GN2 (McClellan et al., 2024), and is designed to handle environments with asymmetries, unlike prior work that primarily focuses on full equivariance.
2. We show the first Partially Equivariant Neural Network applied to Multi-Agent Reinforcement Learning, leading to improved performance over GNNs and EGNNs in MARL.
Context: Prior work has applied equivariance to MARL (Pol et al., 2021; McClellan et al., 2024), these approaches typically assume full equivariance.
3. We formally define and categorize several types of partial equivariance relevant to Multi-Agent Reinforcement Learning (MARL), including subgroup equivariance, feature-wise equivariance, subspace equivariance, and approximate equivariance.
Context: While specific instances of broken symmetries have been discussed (Chen et al., 2023; Park et al., 2024), our work provides a unified and comprehensive categorization tailored to MARL.
4. Through experiments on Multi-Particle Environments (MPE) and the highway-env benchmark, we empirically validate that PEnGUiN outperforms both EGNNs and standard GNNs in MARL tasks with various types of asymmetries.
Context: None

PEnGUiN: Partially Equivariant Graph NeUral Networks for Sample Efficient MARL

Anonymous authors
Paper under double-blind review

Abstract

Equivariant Graph Neural Networks (EGNNs) have emerged as a promising approach in Multi-Agent Reinforcement Learning (MARL), leveraging symmetry guarantees to greatly improve sample efficiency and generalization. However, real-world environments often exhibit inherent asymmetries arising from factors such as external forces, measurement inaccuracies, or intrinsic system biases. This paper introduces *Partially Equivariant Graph NeUral Networks (PEnGUiN)*, a novel architecture specifically designed to address these challenges. We formally identify and categorize various types of partial equivariance relevant to MARL, including subgroup equivariance, feature-wise equivariance, regional equivariance, and approximate equivariance. We theoretically demonstrate that PEnGUiN is capable of learning both fully equivariant (EGNN) and non-equivariant (GNN) representations within a unified framework. Through extensive experiments on a range of MARL problems incorporating various asymmetries, we empirically validate the efficacy of PEnGUiN. Our results consistently demonstrate that PEnGUiN outperforms both EGNNs and standard GNNs in asymmetric environments, highlighting their potential to improve the robustness and applicability of graph-based MARL algorithms in real-world scenarios.

1 Introduction

Multi-Agent Reinforcement Learning (MARL) presents significant challenges due to the complexities of agent interactions, non-stationary environments, and the need for efficient exploration and generalization. Recently, Equivariant Graph Neural Networks (EGNNs) (Satorras et al., 2021) have emerged as a promising approach in MARL, leveraging inherent symmetries in multi-agent systems to improve sample efficiency and generalization performance (Pol et al., 2021; McClellan et al., 2024). By encoding equivariance to transformations like rotations and translations, EGNNs can achieve superior sample efficiency and generalization, particularly in environments where geometric relationships are crucial.

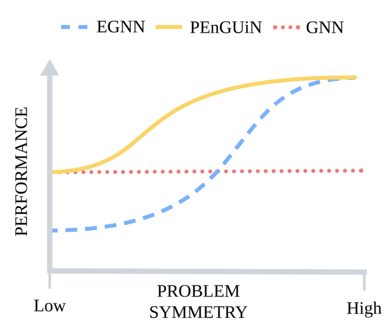


Figure 1: An example of how EGNNs can be advantageous in equivariant environments, and a liability when an environment has increased asymmetries.

However, many real-world MARL scenarios do not exhibit perfect symmetry, and there are concerns that this architecture may be too restrictive in its assumptions. The real world is messy, and it is rare for something to be exactly rotationally equivariant. Factors such as external forces (e.g., wind, gravity), sensor biases, environmental constraints (e.g., obstacles, landmarks, safety zones), or heterogeneous agent capabilities introduce asymmetries that break the assumptions underlying fully equivariant models. Applying standard EGNNs in these partially symmetric environments can lead to suboptimal performance, as the imposed equivariance constraints may not accurately reflect the underlying dynamics. Conversely, standard Graph Neural Networks (GNNs), which lack any inherent equivariance guarantees, may fail to exploit the symmetries that do exist, leading to reduced sample efficiency and weaker generalization.

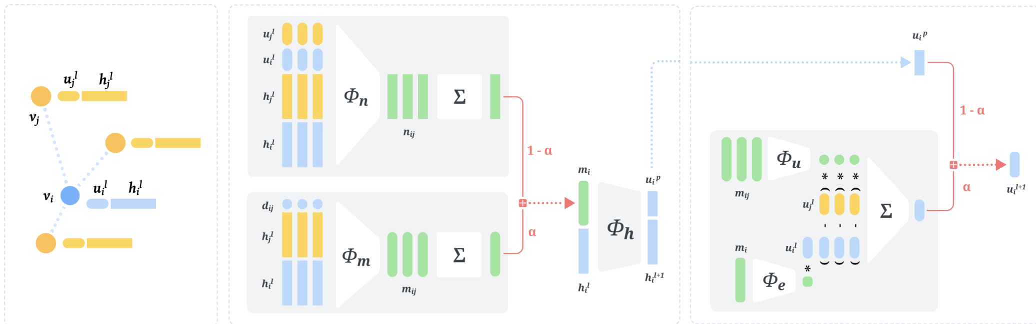
This paper introduces Partially Equivariant Graph NeUral Networks (PEnGUiN), a novel architecture designed to address the challenges of learning in partially symmetric MARL environments. PEnGUiN provides a flexible and unified framework that seamlessly integrates both equivariant and non-equivariant representations within a single model. Unlike traditional approaches that either enforce full equivariance or disregard symmetries entirely, PEnGUiN learns to adaptively adjust its level of equivariance based on the input. This is achieved through a blending mechanism controlled by a learnable parameter that modulates the contribution of equivariant and non-equivariant updates within the network.

Prior works have explored symmetry-breaking cases broadly under the label of "approximately equivariant" Wang et al. (2022c). This work introduces several more precise categories of partial symmetry that commonly emerge in MARL environments. This includes *subgroup equivariance*, *regional equivariance*, *feature-wise equivariance*, and *general approximate equivariance*. These categories are used to design partially equivariant experiments in the Multi-Particle Environments (MPE) (Lowe et al., 2017) and Highway-env Leurent (2018). Experiments show PEnGUiN is able to modulate a 'symmetry score' to adapt to these partially equivariant scenarios. On these benchmarks, PEnGUiN consistently outperforms both EGNNs and standard GNNs.

2 Related Works

Research in equivariant neural networks has explored various architectures and applications, aiming to improve learning and generalization by leveraging symmetries. Equivariant Graph Neural Networks (EGNNs) (Satorras et al., 2021), SEGNNs (Brandstetter et al., 2022), and E3NNs (Geiger & Smidt, 2022) are prominent examples, designed to be equivariant to rotations, translations, and reflections. PEnGUiN builds on the EGNN architecture (Satorras et al., 2021), but extends its capabilities to handle partial equivariance. (Finzi et al., 2021b) introduced Equivariant MLPs, which are versatile but computationally expensive. Within reinforcement learning, van der Pol et al. (2020) and Pol et al. (2021) established theoretical frameworks for equivariant Markov Decision Processes (MDPs) and Multi-Agent MDPs (MMDPs), respectively, focusing on fully equivariant settings with simple dynamics. McClellan et al. (2024) introduced E2GN2 to address exploration challenges in EGNN-based MARL. Chen & Zhang (2024) employed SEGNNs for cooperative MARL, though SEGNNs often have slower training times. Yu et al. (2024) explored adding a symmetry-based loss term, showing limited performance gains. Wang et al. (2022b) investigated rotation equivariance specifically for robotic manipulation with image-based observations. These works primarily address *full* equivariance, or focus on specific tasks or symmetry types, contrasting with PEnGUiN's general and learnable approach to *partial* equivariance.

Research on partial or approximate equivariance includes group CNNs for image processing (Wang et al., 2022c; 2024; McNeela, 2024; Samudre et al., 2024; Ouderaa et al., 2022; Park et al., 2024) and combining MLPs with equivariant components (Finzi et al., 2021a), which are distinct from our graph-based approach. Studies (Wang et al., 2022a; 2023; Petrache & Trivedi) have analyzed the effectiveness of equivariant models in asymmetric scenarios, motivating models like PEnGUiN that can learn equivariance quantities. In the realm of GNNs, Hofgard et al. (2024) concurrently introduce a relaxed equivariant GNN; however, their model is built upon spherical harmonic representations (which increases implementation and computation complexity), unlike PEnGUiN, which is based on EGNNs and allows a smooth transition between fully equivariant and standard GNN behavior. Huang et al. (2023) studies GNN permutation equivariance, not O(n) equivariance. Chen et al. (2023) developed subgroup equivariant GNNs tailored for robotics, specifically to ignore gravity, limiting their applicability compared to PEnGUiN's general framework.



(a) Graph at layer l (b) Node Feature Update $\boldsymbol{h}_i^{l+1}$ (c) Node Coordinate Update $\boldsymbol{u}_i^{l+1}$

Figure 2: Diagram of an individual PEnGUiN layer. The colored boxes represent vectors, where rounded corners indicate the preservation of equivariance and square corners indicate non-equivariance. An example graph is provided in (a), showing coordinates $\boldsymbol{u}_i$ and features $\boldsymbol{h}_i$ corresponding to each node v_i. The update for node v_i (blue) is split into feature and coordinate updates, shown in (b) and (c) respectively. Within each subfigure is a non-equivariant branch (top) and an equivariant branch (bottom), whose outputs are blended via convex combination (red) governed by the symmetry score α.

3 Background

3.1 Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) extends the principles of Reinforcement Learning (RL) to scenarios involving multiple interacting agents within a shared environment. In MARL, each agent aims to learn an optimal policy π_i that maximizes its own expected cumulative reward R_i, which is influenced by the actions of other agents and the environment dynamics. Formally, at each timestep t, each agent i observes a local state s_i^t, takes an action a_i^t according to its policy $\pi_i(a_i^t|s_i^t)$, and receives a reward $r_i^t = R_i(s^t, a^t)$, where $s^t = (s_1^t, ..., s_N^t)$ and $a^t = (a_1^t, ..., a_N^t)$ represent the joint state and action spaces of all N agents (Littman, 1994). The goal of each agent i is to learn a policy $\pi_i(a_i|s)$ that maximizes its expected return: $J(\pi_i) = \mathbb{E}\pi_1, ..., \pi_N\left[\sum_{t=0}^{T}\gamma^t R_i(s_t, a_t^1, ..., a_t^N)\right]$ where T is the time horizon, $\gamma \in (0, 1]$ is a discount factor, and $a_t^j \sim \pi_j(\cdot|s_t)$.

3.2 Equivariance

Equivariance describes how functions behave under transformations. In the context of machine learning, especially for tasks involving geometric data or physical systems, leveraging equivariance can significantly enhance sample efficiency, generalization, and robustness (van der Pol et al., 2020). A function f is said to be equivariant to a group of transformations G if transforming the input x by a group element $g \in G$ results in a predictable transformation of the output $f(x)$. Formally, if T_g represents a transformation of the input space and L_g represents a transformation of the output space, equivariance is defined as:$f(T_g x) = L_g f(x), \quad \forall g \in G, \forall x$. Related to equivaraince is invariance, where the output remains unchanged under the input transformation, i.e., $f(T_g x) = f(x)$.

4 Partially Equivariant Graph Neural Networks

To address the challenges of learning in partially symmetric environments, we introduce Partially Equivariant Graph Neural Networks (PEnGUiN). PEnGUiN is a novel graph neural network architecture designed to seamlessly incorporate varying degrees of equivariance, ranging from full $O(n)$ equivariance, as in E2GN2s, to non-equivariant behavior, akin to standard GNNs. This flexibility is achieved through a blending mechanism controlled by a parameter α, allowing the network to adapt to and learn the specific symmetries present in the data. PEnGUiN follows a similar message-passing paradigm as a standard GNN with message computation, message aggregation, and node feature updates. The forward pass of a single layer l in PEnGUiN, shown in Figure 2, is defined by the following equations:

Table 1: PEnGUiN Update Equations for layer l

Message Computation:	Equivariant: $\boldsymbol{m}_{ij}^l = \phi_m\left(\boldsymbol{h}_i^l, \boldsymbol{h}_j^l, \|\boldsymbol{u}_i^l - \boldsymbol{u}_j^l\|^2\right)$ Non-equivariant: $\boldsymbol{n}_{ij}^l = \phi_n\left(\boldsymbol{h}_i^l, \boldsymbol{h}_j^l, \boldsymbol{u}_i^l, \boldsymbol{u}_j^l\right)$
Message Aggregation:	$\boldsymbol{m}_i^l = \alpha \sum_{j \neq i} \boldsymbol{m}_{ij}^l + (1-\alpha) \sum_{j \neq i} \boldsymbol{n}_{ij}^l$
Equivariant Coordinate Update:	$\boldsymbol{u}_{i,eq}^l = \boldsymbol{u}_i^l \phi_e(\boldsymbol{m}_i^l) + \sum_{j \neq i}\left(\boldsymbol{u}_i^l - \boldsymbol{u}_j^l\right)\phi_u\left(\boldsymbol{m}_{ij}^l\right)$
Feature Update:	$\boldsymbol{h}_i^{l+1}, \boldsymbol{u}_i^p = \phi_h\left(\boldsymbol{h}_i^l, \boldsymbol{m}_i^l\right)$
Partially Equivariant Coordinate Update:	$\boldsymbol{u}_i^{l+1} = \alpha \boldsymbol{u}_{i,eq}^l + (1-\alpha)\boldsymbol{u}_i^p$

Each node i contains two vectors of information: the node embeddings $\boldsymbol{h}_i \in \mathbb{R}^h$ and the coordinate embeddings $\boldsymbol{u}_i \in \mathbb{R}^n$. The node embeddings are *invariant* to $O(n)$. Inputs for layer 0 for $\boldsymbol{h}_i$ may be information about the node itself, such as node type, ID, or status. The coordinate embeddings for node i are *equivariant* to $O(n)$, and inputs will typically consist of positional values (see the appendix for a discussion on how to incorporate velocity and angles).

A layer is updated by first computing the non-equivariant $\boldsymbol{n}_{ij} \in \mathbb{R}^m$ and equivariant $\boldsymbol{m}_{ij} \in \mathbb{R}^m$ messages between each pair of nodes i and j. Each node then aggregates these messages across all neighboring nodes. At this stage, the aggregated non-equivariant and equivariant messages are mixed together. Finally, the updated feature node vector $\boldsymbol{h}_i^{l+1}$ for layer $l+1$ is computed by passing the aggregated message through an MLP $\phi_h : \mathbb{R}^{h+m} \mapsto \mathbb{R}^{h+n}$. This update includes a skip connection to the previous feature node vector. Note that the output of ϕ_h is split into $\boldsymbol{h}_i \in \mathbb{R}^m$ and $\boldsymbol{u}_i^p \in \mathbb{R}^n$ (the latter is used in the Partially Equivariant Coordinate update).

The equivariant coordinate vector is updated using the learnable functions (typically MLPs) $\phi_e : \mathbb{R}^m \mapsto \mathbb{R}$ and $\phi_u : \mathbb{R}^m \mapsto \mathbb{R}$. This update in table 1 is guaranteed to be equivariant to $O(n)$ Satorras et al. (2021). Finally, in the Partially Equivariant update, the equivariant term $\boldsymbol{u}_{i,eq}$ is mixed with a non-equivariant component $\boldsymbol{u}_i^p \in \mathbb{R}^n$.

A key element of PEnGUiN is the addition of the term $\alpha \in (0,1) \subset \mathbb{R}$ to quantify the amount of equivariance in the system. For convenience, we will refer to α as the "symmetry score". The value of the symmetry score has the following important implications:

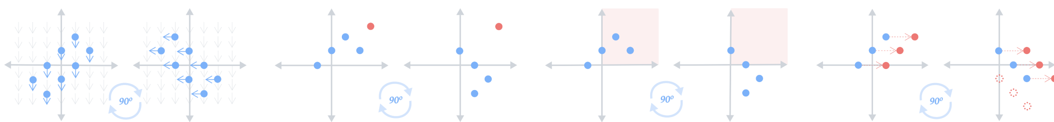
Theorem 1 *Given a Partially Equivariant Graph Neural Network Layer as defined in table 1, when $\alpha = 1$ the Partial Equivariant Layer is exactly equivalent to an E2GN2 layer. (see Appendix A for proof)*

An important implication of this theorem is when $\alpha = 1$ PEnGUiN is exactly equivariant to rotations (the group $O(n)$). This theorem establishes that PEnGUiN embeds EGNN as a special case. When $\alpha = 1$, PEnGUiN fully exploits the benefits of the equivariant inductive bias, such as improved sample efficiency and generalization in environments with symmetric observations.

Theorem 2 *Given a Partially Equivariant Graph Neural Network as defined in table 1, when $\alpha = 0$ the Partial Equivariant Graph Neural Network is equivalent to a GNN (see Appendix A for proof)*

This theorem highlights PEnGUiN's ability to operate in asymmetric settings. As α approaches 0, the network's reliance on equivariant updates diminishes, allowing it to learn arbitrary, non-equivariant relationships.

In practice, the amount of equivariance will rarely be a simple constant. Equivariance may be restricted to a certain region, or a subset of features. Thus, we estimate α using an MLP as a



(a) External Forces (b) Fixed Obstacle (c) Safety Region (d) Sensor Bias

Figure 3: Examples of the types of partial equivariance described in Table 2, with respect to a 90-degree clockwise rotation about the origin.

function of the input features for each node: $\phi_\alpha(\boldsymbol{h}_i^0, \boldsymbol{x}_i^0) = \alpha$. We will refer to this network as the Equivariance Estimator (EE). This allows α to be learned as a spatially and entity-dependent function, enabling the network to adaptively modulate equivariance within the network.

5 Categories of Partial Equivariance

Previous works have noted that functions may have some error in equivariance (Wang et al., 2022c). Others have noted that functions may be equivariant to subgroups instead of an entire group (Chen et al., 2023). In this work, we present a new formalism to unify these asymmetries. We refer to partial equivariance as any situation with asymmetries.

We divide partial equivariance into four categories: subgroup equivariance, feature-wise equivariance, regional equivariance, and approximate equivariance. Approximate equivariance and subgroup equivariance were previously defined in (Wang et al., 2022c) and (Chen et al., 2023) respectively. Recall that an equivariant function f will result in the following equality: $\|f(T_g x) - L_g f(x)\| = 0$ where G is a group with a representation tranformation T_g acting on the input space and a representation L_g acting on the output space.

Table 2: Types of Partial Equivariance

Type Name	Equation	Examples
Relaxed/Approximate Equivariance	$\|f(T_g x) - L_g f(x))\| \leq \epsilon$	External forces, nonlinear dynamics, sensor errors.
Subgroup Equivariance	$f(T_h x) = L_h f(x), \quad \forall h \in H \subseteq G$	Ignoring the gravity vector.
Feature-Wise Equivariance	$f(T_g x_1, x_2) = L_g f(x_1, x_2)$	Fixed Obstacles.
Regional Equivariance	$\|f(T_g x) - L_g f(x)\| = \epsilon(x)$	Safety regions.

Definition 5.1 (Approximate Equivariance) *Let $f : \mathcal{X} \to \mathcal{Y}$ be a function The function f is approximately equivariant if there exists a small constant $\epsilon > 0$ such that:* $\|f(T_g x) - L_g f(x)\| \leq \epsilon, \quad \forall x \in \mathcal{X}, \quad \forall g \in G$

Approximate equivariance is the most general category of Partial Equivariance. Approximate equivariance means that the function is almost equivariant, but there might be small deviations from perfect equivariance. This is a relaxation of the strict equality required for perfect equivariance. Multi-agent systems with unpredictable wind, nonlinear dynamics, or sensor errors may result in approximate equivariance.

Definition 5.2 (Subgroup Equivariance) *A function $f : \mathcal{X} \to \mathcal{Y}$ is subgroup equivariant with respect to a subgroup $H \subseteq G$ if, for all $h \in H$ and all $x \in \mathcal{X}$ the following is true $f(T_h x) = L_h f(x)$*

As an example of subgroup equivariance, consider a quadcopter operating in 3d space. Previous works have shown this will not be equivariant in $E(3)$, specifically due to the effects of the gravity vector (i.e. rotating in the x-z plane affects the dynamics). Instead, (Chen et al., 2023) only enforced equivariance to the group orthogonal to the gravity vector, that is the subgroup of $E(3)$ that only includes rotations orthogonal to gravity.

Definition 5.3 (Feature-wise equivariance) *Let $x = (x_1, x_2, ..., x_n)$ be an input vector where each x_i represents a different feature or subset of features. A function f is feature-wise equivariant if: $f(T_g x_1, x_2, ..., x_n) = L_g f_1(x), f_2(x), ..., f_m(x)$ Where $f(x) = (f_1(x), f_2(x), ..., f_m(x))$.*

Feature-wise equivariance applies when only part of the input is subject to a symmetry transformation. The function is equivariant with respect to that part of the input, while other parts might be invariant or behave in a non-equivariant way. This allows us to handle situations where some entities of the environment are symmetric, and others are not.

Definition 5.4 (Regional Equivariance) *Let $f : \mathcal{X} \to \mathcal{Y}$ be a function, The function f is regional equivariant if there exists a subspace $\mathcal{S} \subset \mathcal{X}$ such that for all $x \in \mathcal{S}$:*

$$\epsilon(x) = \|f(T_g x) - L_g f(x)\|, \qquad \forall g \in G$$

where $\epsilon(x) > 0 \quad if \quad x \in S, \quad and \quad \epsilon(x) = 0 \quad if \quad x \notin \mathcal{S}$

Regional equivariance means that the function exhibits perfect equivariance *only within a specific region or regional of the input space*. Outside this region, the equivariance property might not hold, or it might be violated to varying degrees.

6 Experiments

This section presents an empirical evaluation of Partially Equivariant Graph Neural Networks (PEnGUiN) to address the following key questions: (1) *Does PEnGUiN offer performance improvements over standard Equivariant Graph Neural Network structures (i.e. EGNN, E2GN2)?* (2) *Is PEnGUiN capable of effectively identifying and leveraging symmetries where they exist while accommodating asymmetries where necessary?* (3) *Does the Equivariance Estimator component of PEnGUiN correctly estimate Partial Equivariance?* To investigate these questions, we conducted experiments on the Multi-Particle Environments (MPE) benchmark suite (Lowe et al., 2017) and the more complex highway-env benchmark Leurent (2018). We compared the performance of PEnGUiN against several baselines using the Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm implementation from RLlib (Liang et al., 2018).

6.1 Multi-Particle Environment (MPE)

We utilized two representative scenarios from the MPE benchmark (Lowe et al., 2017). **Simple Tag:** a classic predator-prey environment where multiple pursuer agents, controlled by the RL policy, aim to collide with a more nimble evader agent controlled by a heuristic policy to evade capture. The environment also includes static landmark entities. **Simple Spread:** a cooperative environment in which three agents are tasked with positioning themselves over three landmarks. Agents receive a dense reward for being close to landmarks and are penalized for collisions with each other. These MPE scenarios provide a simplified setting to initially assess the capabilities of PEnGUiN in environments with varying degrees of symmetry. To systematically evaluate PEnGUiN's ability to handle partial equivariance, we introduced three distinct types of asymmetries into the MPE scenarios, corresponding to the categories previously defined:

1. **Sensor Bias (Approximate Equivariance):** We introduced a constant positional bias to the observations of a subset of entities. In *simple tag*, this bias was applied to the observed positions of landmarks and the evader agent. In *simple spread*, the bias was applied to the landmark observations. Critically, this bias was consistently applied to entities not belonging to the agent's team, mimicking biased sensor measurements of external entities.

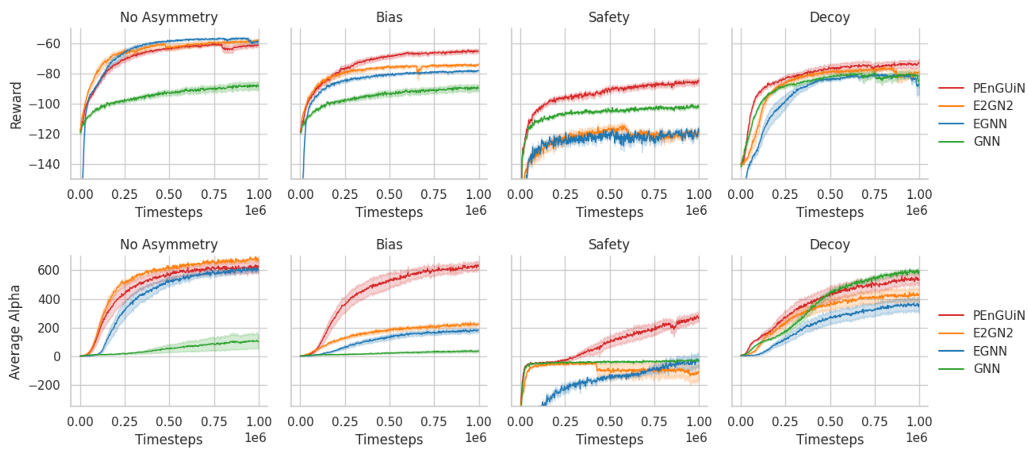


Figure 4: Learning curves on MPE *simple spread* (Top) and *simple tag* (Bottom) environments under 'None', 'Bias', and 'Safety' asymmetry conditions. Results are averaged over 10 seeds with shaded regions indicating standard error. PEnGUiN shows consistent performance, especially in environments with feature-wise and regional equivariance.

2. **Safety Region (Regional Equivariance):** We implemented a safety region by imposing a negative reward penalty whenever an agent entered the upper-right quadrant of the environment. This creates a spatially defined asymmetry.
3. **Decoy (Feature-wise Equivariance):** To test feature-wise equivariance, we added a "decoy" entity to the environment. This decoy visually resembled the agents' objective (evader in *simple tag*, landmarks in *simple spread*) but provided no reward upon interaction. The true objective remained static, while the decoy moved randomly, introducing an asymmetry based on object identity and reward relevance.

We employed the RLLib PPO agent for training all neural network architectures. We compared PEnGUiN against the following baselines. **EGNN:** Equivariant Graph Neural Network Satorras et al. (2021), representing a fully equivariant baseline. **E2GN2:** An unbiased version of EGNNs, with improved MARL performances (McClellan et al., 2024). **GNN:** A standard Graph Neural Network, serving as a non-equivariant baseline.

For PEnGUiN, the α parameter was implemented as a Multi-Layer Perceptron (MLP) that takes as input the node's position $\boldsymbol{u}_i^l$ and node type. This allows α to be learned as a spatially and entity-dependent function, enabling the network to adaptively modulate equivariance.

6.1.1 Results and Discussion (MPE)

Figure 4 presents the learning curves for PEnGUiN, EGNN, E2GN2, and GNN across the standard MPE scenarios and their partially equivariant modifications. In the fully symmetric "None" condition, EGNN and E2GN2 achieve strong performance, validating the benefits of equivariance in symmetric environments. However, their performance significantly degrades in the "Bias" and "Safety" scenarios, demonstrating their sensitivity to symmetry breaking. In contrast, PEnGUiN consistently maintains high performance across all asymmetry conditions, showcasing its robustness and adaptability to partial equivariance. While the standard GNN is less affected by the introduced biases, it consistently underperforms PEnGUiN and equivariant models in symmetric settings, and does not reach the peak performance of PEnGUiN in asymmetric ones. In the "Decoy" environment, PEnGUiN also exhibits superior performance, indicating its effectiveness in handling feature-wise asymmetries.

6.2 PEnGUiN Quantifying Partial Equivariance

Next, we want to explore how well PEnGUiN identifies Partial Equivariance. In theory, we expect the symmetry score (α) to increase as certain regions or features remain equivariant. As asymmetries are introduced into the scenario, the symmetry score should decrease in value where those asymmetries are present.

During training we tracked the average, minimum, and maximum values of the symmetry score. We show these results for the simple tag environment in figure 5. For the scenario with no asymmetries, the symmetry score increases quickly. PEnGUiN is able to learn that the equivariance applies across the scenario. However, it does not reach the exact optimal symmetry score, which would be 1 for this scenario. For the safety and decoy scenarios, we note that the minimum value of α decreases rapidly. It is important to note that the average value seems to stabilize rather quickly, so it appears that learning for the symmetry score occurs primarily in the early stages of training.

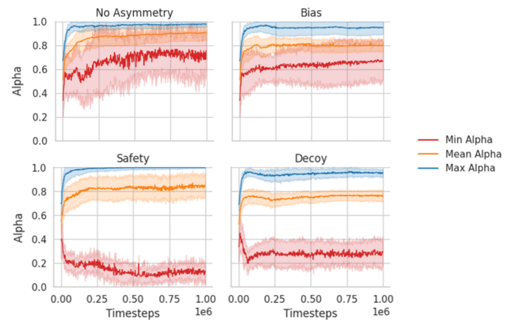


Figure 5: Descriptive statistics of α over training for simple tag. Each statistic is averaged over all 10 seeds for training.

To further investigate PEnGUiN's learned behavior, we visualized the equivariance estimator output in the "Safety Region" scenario (Figure 6).

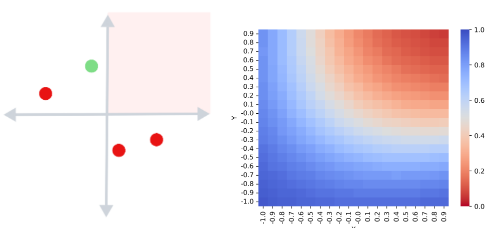


Figure 6: Left: the 'Safety Region' of the simple tag scenario. Right: visualization of learned α values for PEnGUiN in the "Safety Region" scenario.

The heatmap shows the output of the equivariance estimator as a function of agent position (X and Y coordinates). Lower α values (red) indicate reduced equivariance, while higher α values (blue) represent stronger equivariance. PEnGUiN learns to reduce equivariance in the designated safety region (upper-right quadrant), effectively adapting to the regional asymmetry.

Figure 6 reveals that PEnGUiN indeed learns to modulate equivariance spatially. The heatmap shows lower α values concentrated in the upper-right quadrant, corresponding to the safety region. This suggests that PEnGUiN successfully identifies the region where equivariance is broken and reduces its reliance on equivariant updates in that area, while maintaining higher equivariance in the symmetric regions of the environment.

Finally, we experiment with using a hand-designed value for α. If an engineer can identify the symmetries and asymmetries in a scenario, they may encode that into the neural network, improving the inductive bias of the model. For this experiment, we use the simple tag safety environment. We set $\alpha = 0$ when $\boldsymbol{x} > \mathbf{0}$ (i.e. where the safety region violates equivariance and then set $\alpha = 1$ for the remaining locations. In figure 7, we see the results of this simple experiment. Hand designing α, in this case, does indeed seem to improve sample efficiency. This may not always be the case, there are many environments where hand designing α may be nontrivial, especially when one considers that α is used for all layers, and the optimal α may depend on the layer.

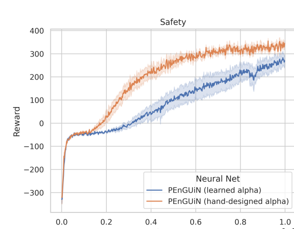


Figure 7: An example of using domain knowledge to hand-design α leading to improved learing

6.3 Experiments on Highway Environment

6.3.1 Environment Setup

To assess PEnGUiN's performance in more complex and realistic scenarios, we evaluated it on the highway-env benchmark Leurent (2018), a suite of environments for autonomous driving. We focused on two challenging environments: Racetrack and Roundabout. **Racetrack:** In this environment, the agent must navigate a closed racetrack, following the track's curvature while maintaining speed and avoiding collisions with other vehicles. **Roundabout:** This scenario requires agents to navigate a roundabout intersection, performing lane changes and speed adjustments to efficiently pass through the roundabout while avoiding collisions.

These environments utilize a more sophisticated bicycle dynamics model for vehicle motion, introducing non-linear dynamics and requiring precise control over steering and throttle actions. Furthermore, the constraint of staying within the road boundaries and lanes naturally introduces a form of regional equivariance, as symmetry is broken at the road edges.

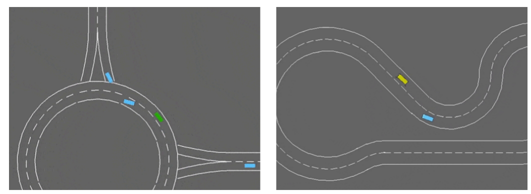


Figure 8: Vizualization of roundabout and racetrack scenario

We maintained consistent implementation details with the MPE experiments, using RLLib PPO and comparing PEnGUiN against the same set of baselines (EGNN, E2GN2, and GNN).

6.3.2 Results and Discussion (highway-env)

Figure 9 presents the learning curves for the highway-env racetrack and roundabout scenarios.

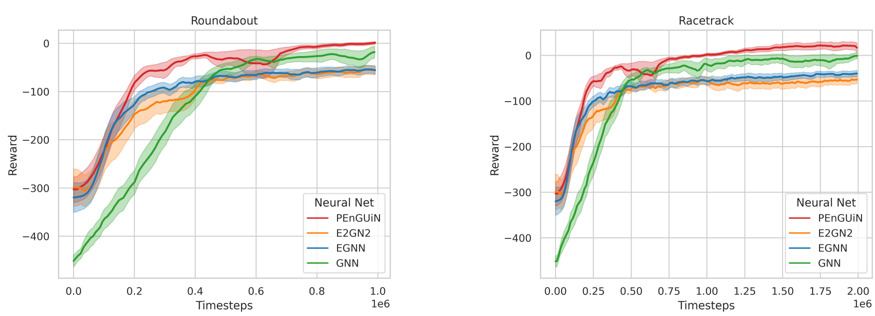


Figure 9: Learning curves on highway-env *racetrack* and *roundabout* environments. Results are averaged over multiple seeds with shaded regions indicating standard error. PEnGUiN consistently outperforms EGNN, E2GN2, and GNN, demonstrating its effectiveness in more complex environments with non-linear dynamics and regional constraints.

The results in Figure 9 demonstrate that PEnGUiN consistently outperforms all baselines in both highway-env scenarios. PEnGUiN achieves higher rewards and exhibits faster convergence compared to EGNN, E2GN2, and GNN. This indicates that PEnGUiN's ability to adapt to partial equivariance is beneficial even in environments with more complex, non-linear dynamics and regional constraints, where full equivariance might be a suboptimal inductive bias.

7 Conclusion

This paper introduced Partially Equivariant Graph Neural Networks (PEnGUiN), a novel architecture for Multi-Agent Reinforcement Learning (MARL) that addresses the limitations of existing fully equivariant models in real-world, partially symmetric environments. Unlike traditional Equivariant Graph Neural Networks (EGNNs) that assume full symmetry, PEnGUiN learns to blend equivariant and non-equivariant updates, controlled by a learnable parameter. This allows it to adapt to various types of partial equivariance, including subgroup, feature-wise, subspace, and approximate equivariance, which we formally defined and categorized.

We theoretically demonstrated that PEnGUiN encompasses both fully equivariant (EGNN) and non-equivariant (GNN) representations as special cases, providing a unified and flexible framework. Extensive experiments on modified Multi-Particle Environments (MPE) and the more complex highway-env benchmark showed that PEnGUiN consistently outperforms both EGNNs and standard GNNs in scenarios with various asymmetries, demonstrating improved sample efficiency and robustness. Furthermore, visualizations of the Equivariance Estimator explored PEnGUiN's ability to identify and exploit regions and features where equivariance holds and where it is violated. PEnGUiN expands the applicability of equivariant graph neural networks to real-world MARL by handling partial symmetries, common in scenarios like robotics, autonomous driving, and multi-agent systems with sensor biases or external forces. By learning to navigate the complexities of partial symmetries, PEnGUiN represents a step towards realizing safe and dependable multi-agent robotic systems in the real world.

A Appendix A: Proofs

A.1 Proof PEnGUiN embeds an E2GN2

Setting $\alpha = 1$ in the PEnGUiN equations directly yields the EGNN equations. For clarity we will rewrite the partially equivariant node and coordinate updates and how it changes when $\alpha = 1$:

$$\boldsymbol{m}_i^l = \alpha \sum_{j \neq i} \boldsymbol{m}_{ij}^l + (1-\alpha) \sum_{j \neq i} \boldsymbol{n}_{ij}^l = \alpha \sum_{j \neq i} \boldsymbol{m}_{ij}^l$$

$$\boldsymbol{u}_i^{l+1} = \alpha \boldsymbol{u}_{i,eq}^l + (1-\alpha) \boldsymbol{u}_i^p = \boldsymbol{u}_{i,eq}^l$$

Thus the update equations become:

$$\boldsymbol{u}_{i,eq}^l = \boldsymbol{u}_i^l \phi_e(\boldsymbol{m}_i^l) + \sum_{j \neq i} \left(\boldsymbol{u}_i^l - \boldsymbol{u}_j^l\right) \phi_u \left(\boldsymbol{m}_{ij}^l\right))$$

$$\boldsymbol{m}_i^l = \sum_{j \neq i} \boldsymbol{m}_{ij}^l \quad \boldsymbol{h}_i^{l+1} = \phi_h \left(\boldsymbol{h}_i^l, \boldsymbol{m}_i^l\right)$$

These are precisely the update equations for an EGNN layer.

A.2 Proof of GNN equivalence

Proof PEnGUiN is equivalent to a GNN when $\alpha = 0$. Recall the node embeddings $\boldsymbol{h}_i \in \mathbb{R}^h$ and the coordinate embeddings $\boldsymbol{u}_i \in \mathbb{R}^n$ For clarity we will rewrite the partially equivariant node and coordinate updates (the equations with α), and how it changes when $\alpha = 0$:

$$\boldsymbol{m}_i^l = \alpha \sum_{j \neq i} \boldsymbol{m}_{ij}^l + (1-\alpha) \sum_{j \neq i} \boldsymbol{n}_{ij}^l = \sum_{j \neq i} \boldsymbol{n}_{ij}^l$$

$$\boldsymbol{u}_i^{l+1} = \alpha \boldsymbol{u}_{i,eq}^l + (1-\alpha) \boldsymbol{u}_i^p = \boldsymbol{u}_i^p$$

Thus far this means that our output $\boldsymbol{h}_i$ will be purely using the GNN update message n_{ij}. Next we will note that we can rewrite $\boldsymbol{h}_i^{l+1}, \boldsymbol{u}_i^p$ as $\boldsymbol{h}_{i,0:h}^l, \boldsymbol{h}_{i,h:h+n}^l$ (essentially this is simply renaming notation. In the main text, we used $\boldsymbol{u}_i^p$ to aid in clarity). We use this renaming to represent that

$\boldsymbol{h}^l_{i,0:h}$ contains the first h elements of the output from ϕ_h, and $\boldsymbol{h}^l_{i,h:h+n}$ is the remaining n elements. Thus the final node update for this layer becomes: $\boldsymbol{h}^l_{i,0:h}, \boldsymbol{h}^l_{i,h:h+n} = \phi_h\left(\boldsymbol{h}^l_i, \boldsymbol{m}^l_i\right)$

To ensure this is equivalent to a GNN, we now look at the next layer in the network. We now see that the next layer becomes:

$$\boldsymbol{n}_{ij} = \phi_n\left(\boldsymbol{h}^{l+1}_i, \boldsymbol{h}^{l+1}_j, \boldsymbol{u}^{l+1}_i, \boldsymbol{u}^{l+1}_j\right) = \phi_n\left(\boldsymbol{h}^{l+1}_{i,0:L-2}, \boldsymbol{h}^{l+1}_{j,0:L-2}, \boldsymbol{h}^{l+1}_{j,L-2:L}, \boldsymbol{h}^{l+1}_{j,L-2:L}\right)$$

This is equivalent to a standard GNN messgae update which is: $\phi_n\left(\boldsymbol{h}_i, vh_j\right)$ The only difference is that we explicitely separate (then later concatenate) the last n elements of $\boldsymbol{h}$ The remainder of the equations of PEnGUiN for layer $l+1$ (with $\alpha = 0$) will be: $\boldsymbol{m}^{l+2}_i = \sum_{j \neq i} \boldsymbol{n}^{l+1}_{ij}$, with the final node update: $\boldsymbol{h}^{l+2}_{i,0:h}, \boldsymbol{h}^l_{i,h:h+n} = \phi_h\left(\boldsymbol{h}^{l+1}_i, \boldsymbol{m}^{l+1}_i\right)$ which is equivalent to a GNN

References

Johannes Brandstetter, Rob Hesselink, Elise van der Pol, Erik J. Bekkers, and Max Welling. Geometric and physical quantities improve E(3) equivariant message passing. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=_xwr8gOBeV1.

Dingyang Chen and Qi Zhang. e(3)-equivariant actor-critic methods for cooperative multi-agent reinforcement learning, 2024.

Runfa Chen, Jiaqi Han, Fuchun Sun, and Wenbing Huang. Subequivariant Graph Reinforcement Learning in 3D Environments. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 4545–4565. PMLR, July 2023. URL https://proceedings.mlr.press/v202/chen23i.html. ISSN: 2640-3498.

Marc Finzi, Gregory Benton, and Andrew Gordon Wilson. Residual Pathway Priors for Soft Equivariance Constraints, December 2021a. URL http://arxiv.org/abs/2112.01388. arXiv:2112.01388 [cs].

Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups, 2021b.

Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks, 7 2022. URL https://arxiv.org/abs/2207.09453v1.

Elyssa Hofgard, Rui Wang, Robin Walters, and Tess Smidt. Relaxed Equivariant Graph Neural Networks, December 2024. URL http://arxiv.org/abs/2407.20471. arXiv:2407.20471 [cs].

Ningyuan Teresa Huang, Ron Levie, and Soledad Villar. Approximately Equivariant Graph Networks. November 2023. URL https://openreview.net/forum?id=5aeyKAZr0L.

Edouard Leurent. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

Michael L. Littman. *Markov games as a framework for multi-agent reinforcement learning*. Morgan Kaufmann, San Francisco (CA), 1994. ISBN 978-1-55860-335-6. DOI: https://doi.org/10.1016/B978-1-55860-335-6.50027-1. URL https://www.sciencedirect.com/science/article/pii/B9781558603356500271.

Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/68a9750337a418a86fe06c1991a1d64c-Paper.pdf.

Josh McClellan, Naveed Haghani, John Winder, Furong Huang, and Pratap Tokekar. Boosting sample efficiency and generalization in multi-agent reinforcement learning via equivariance. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 41132–41156. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/4830a9b95a2f63fc4b3fe09abc18f045-Paper-Conference.pdf.

Daniel McNeela. Almost Equivariance via Lie Algebra Convolutions, June 2024. URL http://arxiv.org/abs/2310.13164. arXiv:2310.13164 [cs].

Tycho F. A. van der Ouderaa, David W. Romero, and Mark van der Wilk. Relaxing Equivariance Constraints with Non-stationary Continuous Filters, November 2022. URL http://arxiv.org/abs/2204.07178. arXiv:2204.07178 [cs].

Jung Yeon Park, Sujay Bhatt, Sihan Zeng, Lawson L. S. Wong, Alec Koppel, Sumitra Ganesh, and Robin Walters. Approximate Equivariance in Reinforcement Learning, November 2024. URL http://arxiv.org/abs/2411.04225. arXiv:2411.04225 [cs].

Mircea Petrache and Shubhendu Trivedi. Approximation-Generalization Trade-offs under (Approximate) Group Equivariance.

Elise Van Der Pol, Herke Van Hoof, Uva-Bosch Deltalab, Frans A Oliehoek, and Max Welling. Multi-agent mdp homomorphic networks, 10 2021. URL https://arxiv.org/abs/2110.04495v2.

Ashwin Samudre, Mircea Petrache, Brian D. Nord, and Shubhendu Trivedi. Symmetry-Based Structured Matrices for Efficient Approximately Equivariant Networks, September 2024. URL http://arxiv.org/abs/2409.11772. arXiv:2409.11772 [stat].

Víctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *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.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Elise van der Pol, Daniel E. Worrall, Herke van Hoof, Frans A. Oliehoek, and Max Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 2020-December, 6 2020. ISSN 10495258. URL https://arxiv.org/abs/2006.16908v2.

Dian Wang, Jung Yeon Park, Neel Sortur, Lawson L. S. Wong, Robin Walters, and Robert Platt. The Surprising Effectiveness of Equivariant Models in Domains with Latent Symmetry. September 2022a. URL https://openreview.net/forum?id=P4MUGRM4Acu.

Dian Wang, Robin Walters, and Robert Platt. $\{SO\}(2)$-equivariant reinforcement learning, 3 2022b. URL https://arxiv.org/abs/2203.04439v1.

Dian Wang, Xupeng Zhu, Jung Yeon Park, Mingxi Jia, Guanang Su, Robert Platt, and Robin Walters. A General Theory of Correct, Incorrect, and Extrinsic Equivariance. November 2023. URL https://openreview.net/forum?id=2FMJtNDLeE&noteId=LtiReb7xPp.

Rui Wang, Robin Walters, and Rose Yu. Approximately Equivariant Networks for Imperfectly Symmetric Dynamics, June 2022c. URL http://arxiv.org/abs/2201.11969. arXiv:2201.11969 [cs].

Rui Wang, Elyssa Hofgard, Han Gao, Robin Walters, and Tess E. Smidt. Discovering Symmetry Breaking in Physical Systems with Relaxed Group Convolution, June 2024. URL http://arxiv.org/abs/2310.02299. arXiv:2310.02299 [cs].

Xin Yu, Rongye Shi, Pu Feng, Yongkai Tian, Simin Li, Shuhao Liao, and Wenjun Wu. Leveraging partial symmetry for multi-agent reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17583–17590, Mar. 2024. DOI: 10.1609/aaai.v38i16.29709. URL https://ojs.aaai.org/index.php/AAAI/article/view/29709.

Supplementary Materials

The following content was not necessarily subject to peer review.

B Additional Training Details

Hyperparameters	value
train batch size	2000
mini-batch size	1000
PPO clip	0.2
learning rate	30e-5
num SGD iterations	10
gamma	0.99
lambda	0.95

Table 3: hyperparameters for MPE

Hyperparameters	value
train batch size	EpisodeLength*16
mini-batch size	EpisodeLength*4
PPO clip	0.2
learning rate	45e-5
num SGD iterations	10
gamma	0.99
lambda	0.95

Table 4: PPO Common Hyperparameters for Highway-env

All MLPs in the GNNs use 2 layers with a width of 32. For all GNN structures we use separate networks for the policy and value functions.

Graph Structure and Inputs The graph structure for MPE environments is set as a complete graph. For MPE environments the input invariant feature for each node $\boldsymbol{h}_i^0$ is the id (pursuer, evader, or landmark). For MPE there is also a velocity feature, which we incoporate following the procedure described in (Satorras et al., 2021). For the Highway-env, we incorporated the

Graph Outputs for Value function and Policy We followed the design choices in (McClellan et al., 2024) for the action space and value function design: the value function output comes from the invariant component of the agent's node of final layer of the EGNN/E2GN2. For MPE the actions are (partially) equivariant, so we use the outputs of the coordinate embeddings. For highway env, the actions are (partially) invariant, so we use the outputs of $\boldsymbol{h}_i$ for the policy output