Fall 2023

Group-Invariant Reinforcement Learning

Fnu Ankur
San Jose State University

Recommended Citation
Ankur, Fnu, "Group-Invariant Reinforcement Learning" (2023). *Master's Theses*. 5426.
DOI: https://doi.org/10.31979/etd.mw24-hmpk
https://scholarworks.sjsu.edu/etd_theses/5426

GROUP-INVARIANT REINFORCEMENT LEARNING

A Thesis

Presented to

The Faculty of the Department of Computer Engineering

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Fnu Ankur

December 2023

The Designated Thesis Committee Approves the Thesis Titled

GROUP-INVARIANT REINFORCEMENT LEARNING

by

Fnu Ankur

APPROVED FOR THE DEPARTMENT OF COMPUTER ENGINEERING

SAN JOSÉ STATE UNIVERSITY

December 2023

Stas Tiomkin, Ph.D.	Department of Computer Engineering
Carlos Rojas, Ph.D.	Department of Computer Engineering
Mahima Agumbe, Ph.D.	Department of Computer Engineering

ABSTRACT

GROUP-INVARIANT REINFORCEMENT LEARNING

by Fnu Ankur

Our work introduces a way to learn an optimal reinforcement learning agent accompanied by intrinsic properties of the environment. The extracted properties helps the agent to extrapolate the learning to unseen states efficiently. Out of all the various types of properties, we are intrigued towards equivariant and invariant properties, which essentially translates to symmetry.

Contrary to many approaches, we do not assume the symmetry, rather learn them, making the approach agnostic to the environment and the property.

The learned properties offers multiple perspective of the environment to exploit it to benefit decision making while interacting with the environment. By building this prior group information into our learning agent's architecture, we essentially reduce the sample or State-action space making the interaction much more efficient and intelligent. The results are agnostic of group structures and similar results can be achieved for any environment and its related group structures.

We have shown that inducing such architecture, we can learn much faster and efficiently in multiple environments using a traditional Deep-Q Network. The future scope would be to test our method across all reinforcement learning approaches, with finite and infinite State-action pairs coupled with enabling multiple properties assisting the agent.

Index Terms—Group Theory, Reinforcement Learning Learning, Equivariance, Invariance.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Stas Tiomkin, for his invaluable guidance and support throughout my research. His expertise in the field of artificial intelligence has been instrumental in shaping my ideas and providing me with the necessary resources to complete this thesis.

I would also like to thank my colleagues at San Jose State University for their insightful discussions and feedback, which have helped me to refine my research questions and methodology.

Finally, I would like to thank my family and friends for their unwavering support and encouragement throughout my academic journey. Their love and encouragement have been a constant source of inspiration and motivation.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

We can only repeat the words of a Nobel-winning physicist, Philip Anderson, "it is only slightly overstating the case to say that physics is the study of symmetry." Symmetry finds its applications across multiple domains and sciences including artificial intelligence. Equivariant reinforcement learning agents learn symmetries of a system and enforce them to learn value functions, resulting in a substantial improvement in sample efficiency. We should emphasize that symmetry has historically been a central concept in many fields of science. In the machine learning community, the importance of symmetry has long been recognized in the applications to pattern recognition and computer vision, with early works on equivariant feature detection dating back to Shun'ichi Amari [1] and Reiner Lenz [2]. The intersection of two fields have engendered many breakthroughs like deep neural networks, approximate homomorphisms, homomorphic networks, Geometric Neural Networks. Learning is a multilayer, multi dynamical process, which makes it efficient but extremely difficult to mimic. There are many components at spatial, conscious, temporal, and emotional level working in conjunction with the environment. Even with advance systems like a human brain or a supercomputer, Learning demands repetition, memory, and time. These repetitions make learning sample inefficient, and unauthentic. An intelligent agent should not need to visit all the states to realize the values of each state. Instead, it should learn to extrapolate the learning to similar states. We aim to make advancement at this intersection, our work focuses on learning symmetries from the data. The benefits of exploiting global properties working at all levels like symmetricity have been proven [3]–[5] to increase the sample efficiency, and making the algorithm resilient to changes. Many works have proved the exploitation of global properties to be successful. Most of the work [6]–[9] has assumed the global property rather than learning them. We propose to learn the equivariant and invariant properties of a dynamical system from the samples and utilize them to assist learning. We can reduce the amount of environment interactions to learn. We propose experiments in the reinforcement learning environments like Open.ai Gym and physics-based simulations in the MuJoCo environment to demonstrate efficient learning. The crux of the experimentations has been dedicated to the learning of groups (mathematical object defining symmetries) in a system through search, constraint search, goal constrained learning, multi-goals agent learning. Once we can deduce properties like symmetry, we could reduce the space complexity efficiently and utilize it for adaptive estimations. Successful implementations would make deep reinforcement learning techniques more efficient at solving Markov decision processes, by making use of prior knowledge about symmetries, asymmetries, equivariant and invariant properties.

1.1 Reinforcement Learning

Reinforcement Learning(RL) is a subset of Artificial Intelligence enabled by repetitions and estimations. The repetitions accompanied with randomness enables experiencing the different states of a system followed by quantification of all the visited and even not visited states optimised by quality of the visited states and allowable actions. The objective intuitively follows to maximise the cumulative reward which is equivalent to the collection of discounted rewards received throughout the path to reach the end goal.

Reinforcement Learning in particular works best for sequential decision making problems primarily taking advantage by defining the history, learning and estimations from sequential state transitions and decision making. The sequential decision making algorithms can cover a host of applications spanning but not limited to healthcare, robotics, self-driving cars and many others. In recent times, Deep learning techniques have been combined with RL to create deep RL, which is most useful in problems with high dimensional state-space. Deep RL has been successful in complicated tasks with lower prior knowledge thanks to its ability to learn different levels of abstractions from data. For instance, a deep RL agent can successfully learn from visual perceptual inputs made up of thousands of pixels. This opens up the possibility to mimic some human problem solving capabilities, even in high-dimensional space.

Deep reinforcement learning (RL) has achieved impressive results in games, such as Atari games, Go, and Poker . It also has the potential to be used in real-world applications such as robotics, self-driving cars, finance, and smart grids.However, there are several challenges in applying deep RL algorithms. One challenge is exploring the environment efficiently. Another challenge is generalizing a good behavior to a slightly different context.Researchers have proposed a wide range of deep RL algorithms to address these challenges, depending on the specific sequential decision-making task.

1.2 Background

Here we summarize the fundamentals of the theory behind groups and equivariance. We begin with a brief outline of the concepts of equivalence, invariance, and equivariance, followed by a review.

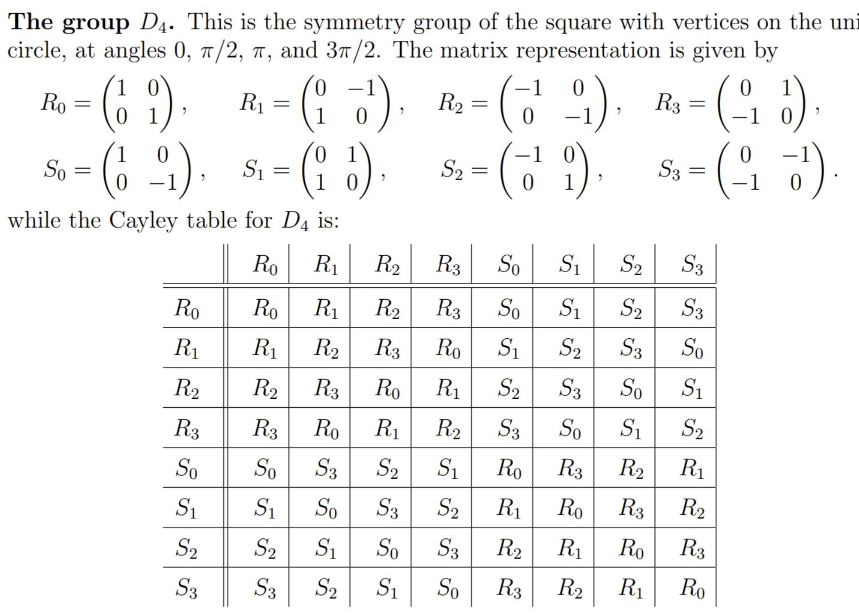
Group A group is a general object in mathematics. A group is a set of elements that can be combined in a binary operation whose output is another member of the group. We're interested in groups of transformations that move points in a space. Operations like rotation, scaling, mirroring, or translating of single points.

A group G is a set of elements a,b,c,e,i,... equipped with a binary operation (a.b = c) whose output is another group element and the following conditions are satisfied.

- **Closure**: The output of the binary operation is always a member of the group.
- **Associativity**: (a.b).c = a.(b.c)
- **Identity**: There is a single identity element e such that ex = x $\forall x \in G$
- **Inverse**: There exists exactly one inverse element i for each x such that xi=e.

Generally, Groups are not commutative, if they are commutative, then they are called **abelian groups**. Groups can be finite or infinite. The number of elements in a group is called the order of the group.

The dihedral group D_4 shown in 1 and 2 is one of the two non-Abelian groups of the five groups total of group order 8. It is sometimes called the octic group. An example of D_4 is the symmetry group of the square. Group Elements: $r_0, r_1, r_2, r_3, s_0, s_1, s_2, s_3$



The group D_4. This is the symmetry group of the square with vertices on the unit circle, at angles 0, $\pi/2$, π, and $3\pi/2$. The matrix representation is given by

$$R_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad R_1 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad R_2 = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \quad R_3 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix},$$

$$S_0 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad S_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad S_2 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \quad S_3 = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}.$$

while the Cayley table for D_4 is:

	R_0	R_1	R_2	R_3	S_0	S_1	S_2	S_3
R_0	R_0	R_1	R_2	R_3	S_0	S_1	S_2	S_3
R_1	R_1	R_2	R_3	R_0	S_1	S_2	S_3	S_0
R_2	R_2	R_3	R_0	R_1	S_2	S_3	S_0	S_1
R_3	R_3	R_0	R_1	R_2	S_3	S_0	S_1	S_2
S_0	S_0	S_3	S_2	S_1	R_0	R_3	R_2	R_1
S_1	S_1	S_0	S_3	S_2	R_1	R_0	R_3	R_2
S_2	S_2	S_1	S_0	S_3	R_2	R_1	R_0	R_3
S_3	S_3	S_2	S_1	S_0	R_3	R_2	R_1	R_0

Fig. 1. D_4: Group elements as matrix, and cayley table

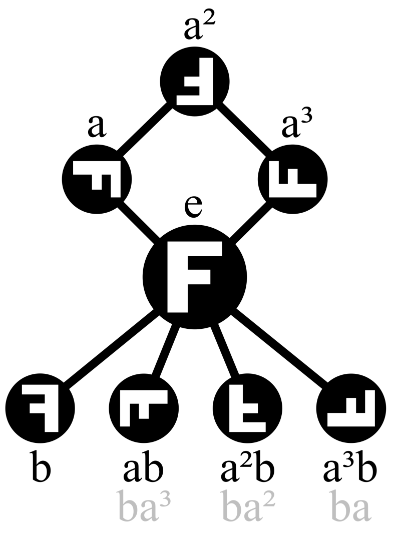


Fig. 2. D_4 visualization

Group Action: A group action $\pi(g, v)$ is a mapping from a group G and a space X to the space X.

$$\pi : G \times X \rightarrow X \tag{1}$$

Group Theory Group theory is the study of symmetry. When an object appears symmetric, group theory can help us study it. We apply the label "symmetric" to anything that is invariant under some transformations. This can apply to geometric figures (a circle is highly symmetric, being invariant under all rotations), but also to algebraic objects like functions: $x^2 + y^2 + z^2$ is invariant under all rearrangements of x, y, and z and the trigonometric functions $\sin(t) and \cos(t)$ are invariant when t is replaced by $t + 2\pi$.

Cayley table describes the structure of a finite group by arranging all the possible products of all the group's elements in a square table reminiscent of an addition or multiplication table. Many properties of a group – such as whether or not it is abelian, which elements are inverses of which elements, and the size and contents of the group's center – can be discovered from its Cayley table.

Equivalence If a function $f : X \rightarrow Y$ maps two inputs $x, x^{'} \in X$ to the same value, that is $f(x) = f(x^{'})$, then we say that x and $x^{'}$ are f-equivalent. Two states $s, s^{'}$ converging to the same Value $V(s) = V(s^{'})$ could be named as V-equivalent.

Invariance Intuitive relationships between points in an equivalence class. This is formalized with the transformation operator $L_g : X \rightarrow X, where\ g \in G$ and G is a mathematical group. if L_g satisfies

$f(x) = f(L_g[x]), for\ all\ g \in G, x \in X$ then we sat that f is invariant or symmetric to L_g and that $(L_g)_{g \in G}$ is a set of symmetries of f.

Equivariance Given a transformation operator $L_g : X \rightarrow X$ and a mapping $f : X \rightarrow Y$, f is equivariant to the transformation if there exists a second transformation operator $K_g : Y \rightarrow Y$ in the output space of s such that $K_g[f(x)] = f(L_g[x]) for\ all\ g \in G, x \in X$ The operators L_g and K_g can be seen to describe the same transformation, but in different spaces. Infact, an equivariant map can be seen to map orbits to orbits. We also see that invariance is a specialcase of equivariance, if we set K_g to the identity operator for all g.

Group-structured MDP Homomorphisms MDP homomorphic networks are neural networks that are equivariant under symmetries in the joint state-action space of an MDP. Current approaches to deep reinforcement learning do not usually exploit knowledge about such structure. By building this prior knowledge into policy and value networks using an equivariance constraint, we can reduce the size of the solution space as shown below in fig 3

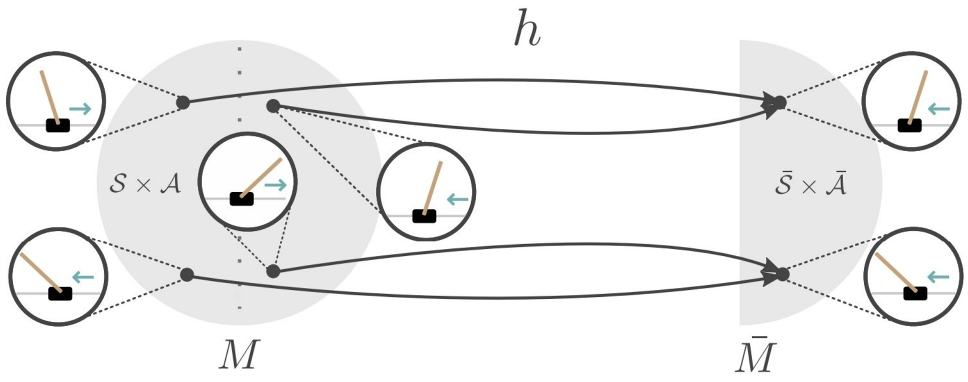

h
$\mathcal{S} \times \mathcal{A}$
$\bar{\mathcal{S}} \times \bar{\mathcal{A}}$
M
$\bar{M}$

Fig. 3. Reduction in an MDP's state-action space under an MDP homomorphism h.

2 LITERATURE REVIEW

While many work has been published over the years regarding the study, utilization of geometric and symmetry in learning independently and in conjunction with multiple domains. The idea to unify and standardize this process remains a big ask. The initial work [4] [3] [5] of Dr. Balaraman Ravindran and Dr. Andrew G. Barto provided an entry point to utilize dynamical properties like symmetries of a system with an exact and approximated Markov decision process. While they set the definitions and proofs and concludes with a constructive proof for the isomorphism completeness of the problem. The work proposes analogy of the Markov Decision Processes to the graphs and derived completeness proof useful in both settings., they did not discuss results and showed no general method to learn symmetries.

With the advancement of Deep reinforcement learning [6] [7] and applicability to huge state-action spaces, we constantly see many approached to address that [8] [9] but none of them fundamentally address the issues. The solutions are reinforcement learning environment-dependent and could not be replicated across the simulations. On the other hand, every environment simulation would have geometric symmetricity to leverage and the method is environment-agnostic.

Some recent works [10] [11] [12] speaks volume of incorporating invariance and equivariance properties in the deep reinforcement learning. All these works have been recognised at respected conferences and Summits but all of these approaches either assume some symmetry from the environment or requires manual inputs to incorporate these properties in the learning process. We want to advance by eradicating these assumptions and learn from our interaction or data samples rather than expecting symmetry property as a known knowledge. Though the type of symmetry in proposed solutions are in-built and not learned, it basically provides end-end platform to experiment and prove the benefits of such structures.

Another popular approach to improve the data sampling efficiency is to either include more transition per state with sub-goals/fake-goals [13] or prioritize the newer transitions [14] [15] [16] sampled from the buffer play. These methods resolve the sparse rewards problem to an extent but none of them utilizes any equivariant properties of the environment to increase the replay buffer transitions. The real incentive of utilizing groups would be that it would help to include unseen symmetric states to replay buffer without ever visiting those states.

There have been developments [2] [1] in deep learning architectures as well. The utility of symmetries is not restricted to any one domain and can be beneficial to many different learning systems. The research work highlights the use of symmetries in convolutional layers. The authors proposed a new convolutional layer equivariant to transformations from the Lie groups. The adaptation of symmetries across multiple domains is powerful to highlight and needs continuous development. All the work aims to exploit symmetries (equivariance and invariance) properties of a system to advantage to make the learning efficient with decreasing the sample space. The difference comes with assumptions of type of symmetries, environments, computations, and objectives of the tasks. Our work with symmetries in general is to learn the symmetries from interactions and leverage it for efficient computations.

3 BACKGROUND

This chapter introduces the problem statement in context with the initial discrete state and action space. We will be establishing the building blocks of a group and reinforcement learning environment and agent.

3.1 Problem Setup

A group essentially is a set of elements, in our case, transformation matrices. The dimension of a group element is dependent on the space it acting on. Our reinforcement learning environment consists of two different space, namely state space and action space.

An example of two group element acting in a state space(L) of four dimension and action space(K) of two dimension respectively can be generalized as:

$$L = \begin{pmatrix} l_{11} & l_{12} & l_{13} & l_{14} \\ l_{21} & l_{22} & l_{23} & l_{24} \\ l_{31} & l_{32} & l_{33} & l_{34} \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix} K = \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$$

A Dihedral group is the group of symmetries of a regular polygon, which includes rotations and reflections. Dihedral groups are among the simplest examples of finite groups. The figure below shows the transformation matrices for the group elements of a D4 group. A Cayley table describes the structure of a finite group by arranging all the possible products of all the group's elements in a square table reminiscent of an addition or multiplication table. The Cayley table is a structured way to understand the interaction among the group elements. Group Elements for a D_4 group: $r_0, r_1, r_2, r_3, s_0, s_1, s_2, s_3$

A regular polygon with n sides has 2n different symmetries: n rotational symmetries and n reflection symmetries. In our discrete set-up, Our state space consists of eight possible rotational and reflection symmetric state of a regular four-sided polygon. All the possible states are shown in the figure 4 below. Consider a square with vertices at $0 : (0.0, -1.0), 1 : (-1.0, 0.0), 2 : (0.0, 1.0), 3 : (1.0, 0.0)$ Starting from r_0 position, you can reach to all the states taking the available action or applying one of the group elements. All the possible states are shown below.

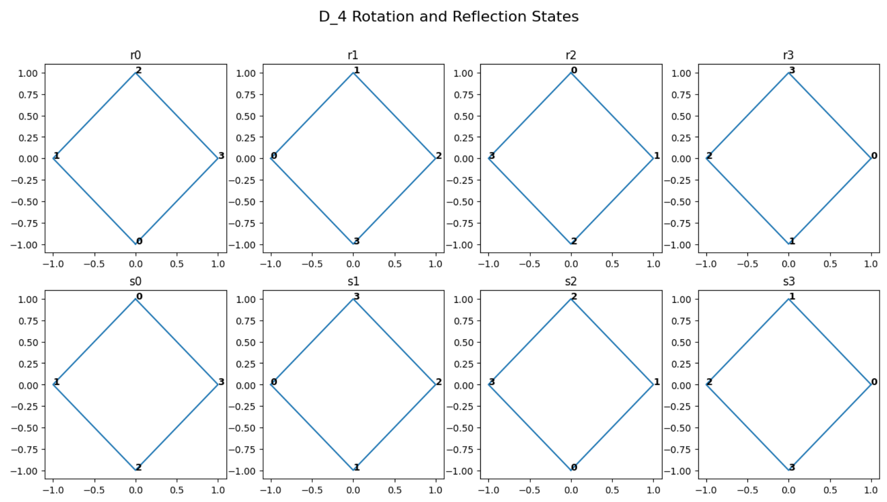


Fig. 4. D_4: group

At this point of time, we have established that a regular polygon with sides n has 2n rotational and reflection symmetries, we consider D_4 group, equivalent to 8 symmetries. We start with a square, apply the symmetries to the square resulting in the complete state space, essentially action space comprises of all the group elements. Our objective is to traverse from one state to the another at a minimum cost with respect to the number of steps and learn a group. We present this problem as a sequential decision making, at each state, we make a decision about the optimal action to reach our goal state. The transition dynamics is deterministic and a positive reward is received upon reaching the goal state.

3.2 Value Iteration

Reinforcement Learning algorithm, Value Iteration works perfectly with finite number of states and action in a sequential decision making process. Value iteration is a method of computing an optimal Markov Decision Process policy and its value. Value iteration starts at the ”end” and then works backward, refining an estimate of either Optimal Q value or Value function $Q^* or V^*$. The algorithm works by iteratively updating the value function for each state until it converges.

In mathematics, a Markov decision process (MDP) is a discrete-time stochastic control process which can be utilized to solve a sequential decision making process.

A **Markov Decision Process (MDP)** is a 5-tuple $< S, A, P, R, s_0 >$

- S: a set of states.
- A: a set of actions.
- $P(S_{t+1}|S_t, A_t)$: the dynamics.
- $R(S_t, A_t, S_{t+1})$: the reward. $R(s, a, s_0)$ is the reward received in state s, with action a and ends up in state s_0.
- s_0: initial state.

A stationary **policy** is a function: $\pi : S \to A$

- Given a state $s, \pi(s)$ specifies what action the agent will do.
- An optimal policy is one with maximum expected value.
- For an MDP with stationary dynamics and rewards with infinite or indefinite horizon, there is always an optimal stationary policy in this case.

The value function is a function that maps from states to the expected reward that can be obtained by starting in that state and following the optimal policy. The optimal policy is the policy that maximizes the expected reward over time. Value iteration is a guaranteed to converge to the optimal value function, but it can be computationally expensive for large MDPs.

```
Algorithm 1 Value Iteration
Input: A Markov decision process (MDP) $(S, A, T, R)$, a discount factor $\gamma$
Input: An initial value function $V(s)$
1 for $s \in S$ do
2     $V(s) \leftarrow \max_a \left[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V(s')\right]$
3 end
4 $V$ converges.
```

The method above describes the use of Value Iteration to get all the possible paths while traversing from one state to the another using optimal actions. There are two caveats in this process though.

- There are no group axioms constraints applied while traversing and collecting actions/group elements. This leads to compute-extensive post processing.
- The ways to incorporate group axioms while extracting the group elements is too complex, time and compute hungry.

Though, this method of extracting groups mainly depends on post-processing with optimal search. One of the advantage of using this method is that we can extract not only groups but sub-groups as well. We have been able to retrieve the group elements successfully but the method does not inspires learning but depends on optimal search with cumbersome post-processing. The value iteration for each state is not ideal to perform for large amount of data. We require better vectorized version of the algorithm which is considered and explained in next method.

It is worth while to note that we have also considered and experimented with graph structures and graph search. The states were expressed as nodes with actions represents the edges. The results were similar to normal structure but with better execution time.

3.3 Goal Dependent- Value Iteration

We tried to revisit and address some of the problems with our first method. Though it shows the validity of the approach but it lacks learning rather depends on optimal search. The goal is to find the optimal group element at every state to reach any state. The problem scales exponentially with states, we need to compute the value functions efficiently and our current method learns goal dependent Value functions which scale our method to bigger state space.

Algorithm 2 Value Iteration

```
Input: A Markov decision process (MDP) $(S, A, T, R)$, a discount factor $\gamma$
Input: An initial value function $V(s)$
5 for $s \in S$ do
6     $V(s|g) \leftarrow \max_a \left[R(s|g,a) + \gamma \sum_{s' \in S} T(s,a,s')V(s'|g)\right]$
7 end
8 $V$ converges.
```

After learning the goal conditioned Value functions. We have experimented to find the effective way of traversing to all the states as well to guarantee a group. Few of the insights from our work.

- We always can guarantee to find the original group if we start with one state and traverse all the states without repetition.
- We cannot guarantee to extract a group if we traverse all the states randomly. Although that the group elements accumulated in such traversing reveals intrinsic properties of the agent. We can even extract some sub-groups from it which can be helpful to explains a lot of variation in the data if not all.

We have experimented with multiple groups possessing variants transformation properties and we have always been able to extract the original group. Our results have been consistent which validates our approach and should be useful to apply the learnt group to deep learning and reinforcement learning architectures.

4 LEARNING GROUPS

The aspiration to learn intrinsic properties of the environment simultaneously with the reinforcement learning agent is of utmost priority to the core idea of alleviating learning cost by acknowledging the environment structure and improving the data sample efficiency.

Though our initial approaches successfully extract a group from an environment but with a vital caveat of assuming either the group structure or limiting the data samples.

Our current structure addresses the above mentioned bottlenecks. We re-imagine a basic reinforcement learning algorithm agent experiencing and acting in an environment in addition with some transformation matrices with respect to state and action, adding different perspectives to help the agent to explore, act and estimate more with the same count of data points.

We have chosen Deep-Q-network to test out methodology and present our results. This pseudocode describes the basic Deep Q-learning algorithm. It works by maintaining a Q-network, which estimates the expected future reward for taking each action in each state. The agent then selects actions based on the Q-network's estimates, and updates the Q-network as it learns. The greedy policy ensures that the agent explores new actions even if they are not predicted to be the best. This is important for learning new behaviors and avoiding getting stuck in local optima. The learning rate controls how quickly the agent updates its Q-network. A higher learning rate means that the agent will learn more quickly, but it may also make the agent more unstable. The discount factor controls how much weight the agent places on future rewards. A higher discount factor means that the agent will value future rewards more highly. The vanila algorithm along with results are shown below 5

Algorithm 3 Deep Q-Learning with Experience Replay

```
Input: Initialize replay memory $D$ to capacity $N$
Input: Initialize action-value function $Q$ with random weights $\theta$
Input: Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
9  for episode = 1, M do
10     Initialize state $s_1$  for $t$ = 1, $T$ do
           • With probability $\epsilon$ select a random action $a_t$, otherwise select $a_t = argmax_a Q(s_t, a; \theta)$
           • Execute action $a_t$ in emulator and observe reward $r_t$ and next state $s_{t+1}$
           • Store transition $(s_t, a_t, r_t, s_{t+1})$ in $D$
           • Sample random minibatch of transitions $(s_j, a_j, r_j, s_{j+1})$ from $D$
           • Set $y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma max_{a'} \hat{Q}(s_{j+1}, a'; \theta^-) & \text{for non-terminal } s_{j+1} \end{cases}$
           Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$ with respect to the network parameters $\theta$
           Every C steps reset $\hat{Q} = Q$
11     end
12 end
```

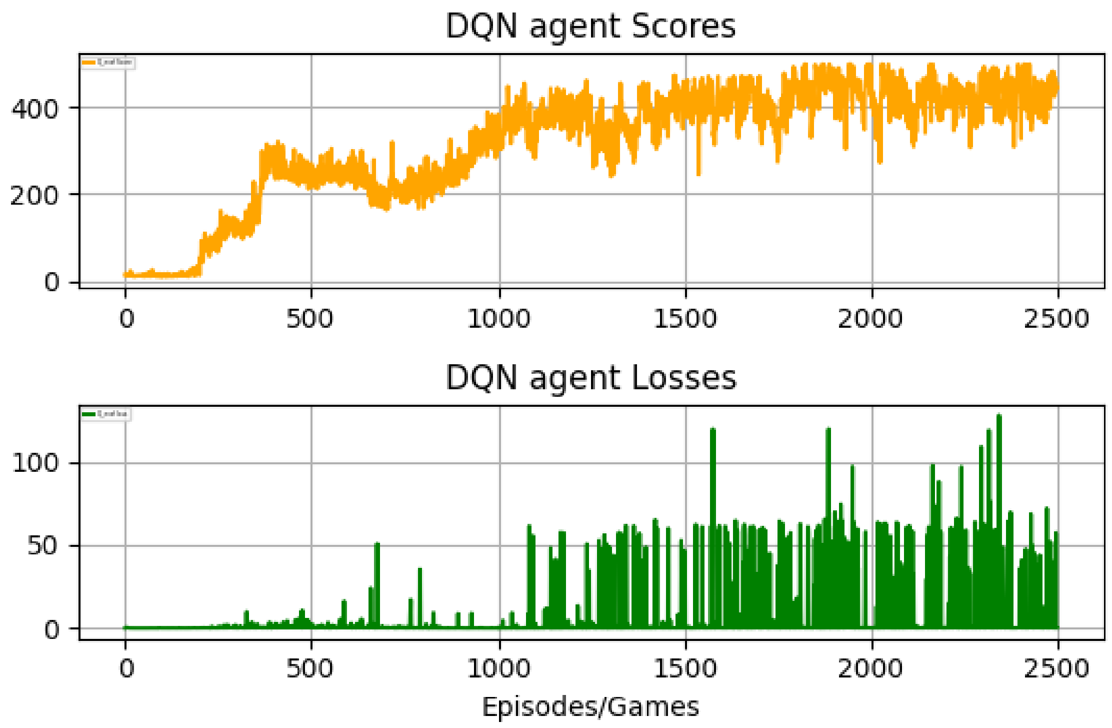


Fig. 5. Deep Q learning results

4.1 Experiment: 1

In our first experiment, we introduce a **symmetric network**, which is a copy of DQN network, with added parameters conditioned to learn a state(L), and an action(K) transformation matrix. The state transformation matrix linearly transform the seen states to give the symmetric states keeping the dimensions same, and action transformation matrix linearly transform the action space to adjust to the new symmetric states while keeping the estimated Q values optimal.

The symmetric network has the same neural network weights as the DQN agent which gets duplicated after every gradient learning step. The objective of the symmetric network is to learn the L and K matrix following the Q value estimated by the original DQN network.

The intention for the addition of the symmetric network was to learn individual transformation matrices equivalent to a group element of an unknown group in state and action space tied together by bellman optimality equation.

$Q^*(s,a) = \max_{a'} [R(s,a) + \gamma \sum_{s'} P(s'|s,a) Q^*(s',a')]$

- $Q^*(s,a)$ optimal Q-value for state-action pair (s,a)
- $R(s,a)$ reward received for taking action a in state s.
- γ discount factor
- $P(s'|s,a)$ probability of transitioning to state s' after taking action a in state s

First Experiment Results Observation:

- As expected, the state and action matrix each converge to a group element although, identity matrix. The convergence to trivial solution validates the architecture and methodology.
- Unexpectedly, the Q network learns faster in presence of a symmetric network as can be seen in 6

Algorithm 4 Deep Q-Learning with Experience Replay and Symmetric Network

```
Input: Initialize replay memory $D$ to capacity $N$
Input: Initialize action-value function $Q$ with random weights $\theta$
Input: Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
Input: Initialize symmetric action-value function $\hat{S}$ with weights $\theta^* = \theta$, along with state
       symmetric matrix $L$ and action symmetric matrix $K$
13 for episode = 1, M do
14     Initialize state $s_1$  for t = 1, T do
           • With probability $\epsilon$ select a random action $a_t$, otherwise select $a_t = argmax_a Q(s_t, a; \theta)$
           • Execute action $a_t$ in emulator and observe reward $r_t$ and next state $s_{t+1}$
           • Store transition $(s_t, a_t, r_t, s_{t+1})$ in $D$
           • Sample random minibatch of transitions $(s_j, a_j, r_j, s_{j+1})$ from $D$
           • Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma max_{a'} \hat{Q}(s_{j+1}, a'; \theta^-) & \text{for non-terminal } \phi_{j+1} \end{cases}$
           Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$ with respect to the network
           parameters $\theta$
           Perform a gradient descent step on $(\hat{S}(s_j, a_j; \theta) - Q(s_j, a_j; \theta))^2$ with respect to the
           network parameters $\theta^*$

           Every C steps reset $\hat{Q} = Q$ and $\hat{S}$=$Q$, partial update
15     end
16 end
```

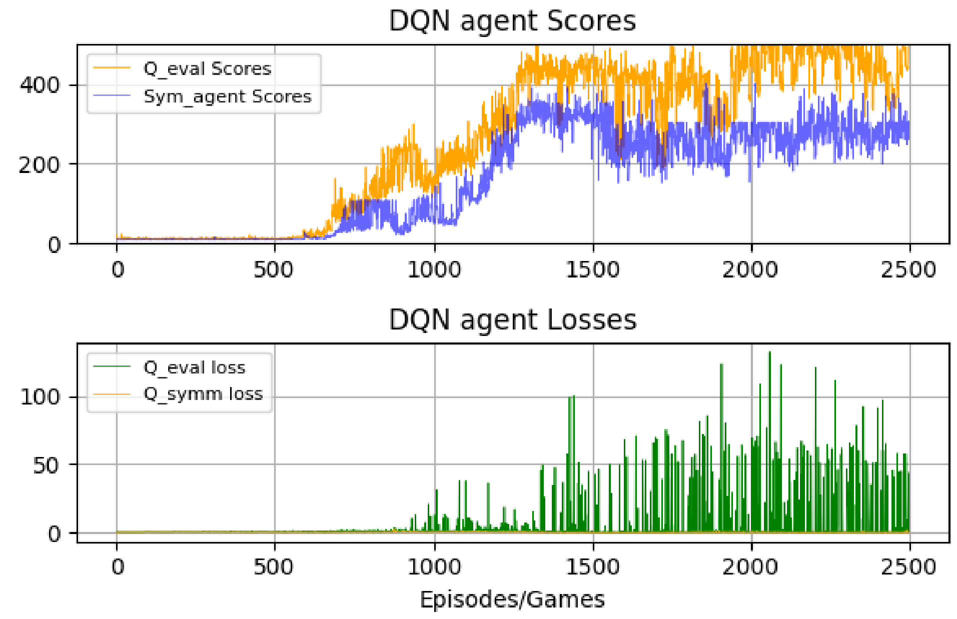


Fig. 6. Deep-Q learning with symmetric network results

- This experiment concludes our proof and validity of the concept and demands further investigation.

In our subsequent experiments, we aim to extract non-trivial group elements solutions from our current architecture.

4.2 Experiment: 2

After a few experiments with loss functions aiming to converge symmetric state and action matrices to group elements apart from identity matrix lead to a much simpler idea to utilize a stable state and action symmetric matrices to generate symmetric states and actions, and aid the DQN agent learning by providing experience from the symmetric states in addition to the visited states resulting in efficient algorithm as shown in 7

Algorithm 5 Deep Q-Learning with Experience Replay and Symmetric Network

```
Input: Initialize replay memory $D$ to capacity $N$
Input: Initialize action-value function $Q$ with random weights $\theta$
Input: Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
Input: Initialize symmetric action-value function $\hat{S}$ with weights $\theta^* = \theta$, along with state
       symmetric matrix $L$ and action symmetric matrix $K$
17 for episode = 1, M do
18     Initialize state $s_1$  for t = 1, T do
           • With probability $\epsilon$ select a random action $a_t$, otherwise select $a_t$ =
             $argmax_a Q(s_t, a; \theta)$
           • Execute action $a_t$ in emulator and observe reward $r_t$ and next state $s_{t+1}$
           • Store transition $(s_t, a_t, r_t, s_{t+1})$ in $D$
           • Sample random minibatch of transitions $(s_j, a_j, r_j, s_{j+1})$ from $D$
           • Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma max_{a'} \hat{Q}(s_{j+1}, a'; \theta^-) & \text{for non-terminal } \phi_{j+1} \end{cases}$
           if $\hat{S}$ is stable, maximum rewards then
               $symmetric\_loss = \text{MSE}((\hat{S}(s_j, a_j; \theta), Q(s_j * l_1, (a_j * l_2))$     (2)
               Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2 +$ symmetric loss
               with respect to the network parameters $\theta$
           else
               Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$ with respect to the
               network parameters $\theta$
           end
           Perform a gradient descent step on $(\hat{S}(s_j, a_j; \theta) - Q(s_j, a_j; \theta))^2$ with respect to the
           network parameters $\theta^*$

           Every C steps reset $\hat{Q} = Q$ and $\hat{S}^= Q$, partial update
19     end
20 end
```

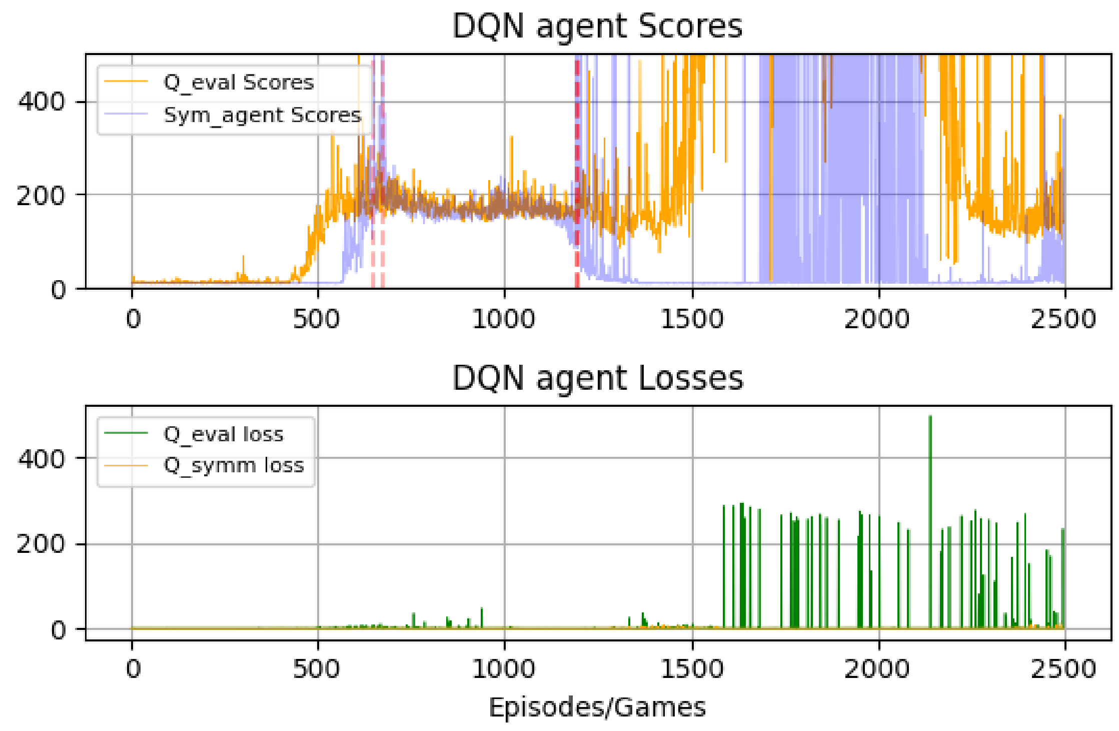


Fig. 7. Deep-Q learning with symmetric network and custom loss function results

5 RESULTS

The addition of symmetric network opened up a lot of possibilities to change the architecture and aid the DQN agent in learning. The main contribution comes from utilizing the symmetric states and actions acquired from transformation matrices in state and action space which helps in providing more data from different perspective given a sample transitional data. Symmetric network helped the DQN network to converge faster, combining with greater stability than original DQn agent, by learning the symmetry rather than assuming them as shown in 8. Finally, more experimenting with the symmetries might reveal much more about the environment and that would be one of the few future improvements.

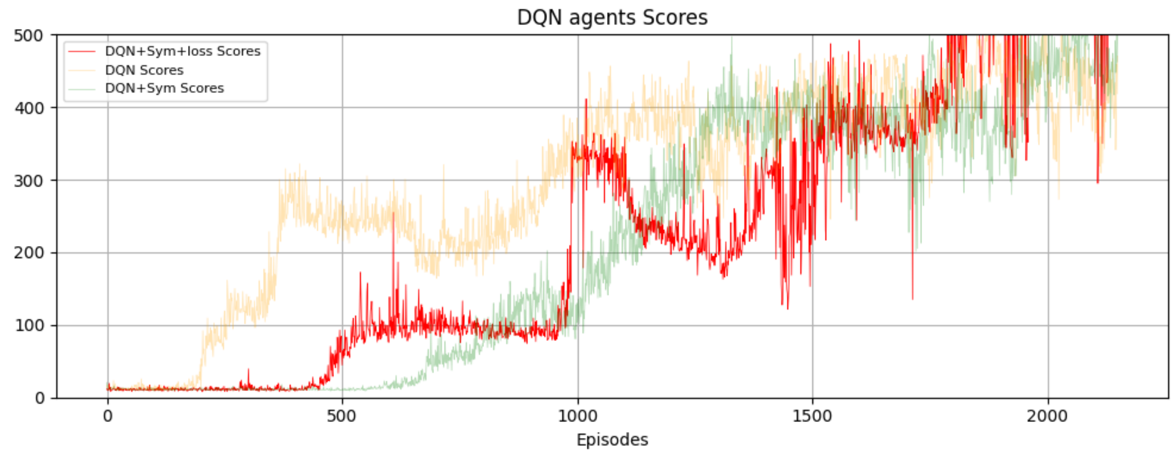


Fig. 8. Results comparison

6 FUTURE WORK

We will continue to research in multiple direction. We shall be testing our current implementation in variety of environments with multiple learning algorithm. We intend to revisit custom loss function approach, which relies on adequate loss functions to converge to non-trivial solutions.

Literature Cited

[1] S.-l. Amari, "Feature spaces which admit and detect invariant signal transformations," in *Proc. 4th Int. Joint Conf. Pattern Recognition*, pp. 452–456, 1978.

[2] R. Lenz, *Group theoretical methods in image processing*, vol. 413. Springer, 1990.

[3] B. Ravindran, "Approximate homomorphisms : A framework for non-exact minimization in markov decision processes," 2004.

[4] B. Ravindran and A. G. Barto, "Symmetries and model minimization in markov decision processes," tech. rep., University of Massachusetts, USA, 2001.

[5] S. M. Narayanamurthy and B. Ravindran, "On the hardness of finding symmetries in markov decision processes," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 688–695, 2008.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[8] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 10674–10681, 2021.

[9] V. Mai, K. Mani, and L. Paull, "Sample efficient deep reinforcement learning via uncertainty estimation," *arXiv preprint arXiv:2201.01666*, 2022.

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

[11] M. Finzi, S. Stanton, P. Izmailov, and A. G. Wilson, "Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data," in *International Conference on Machine Learning*, pp. 3165–3176, PMLR, 2020.

[12] M. Finzi, M. Welling, and A. G. Wilson, “A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups,” in *International Conference on Machine Learning*, pp. 3318–3328, PMLR, 2021.

[13] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, “Hindsight experience replay,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[14] S. Fujimoto, D. Meger, and D. Precup, “Off-policy deep reinforcement learning without exploration,” in *International Conference on Machine Learning*, pp. 2052–2062, PMLR, 2019.

[15] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” *arXiv preprint arXiv:1511.05952*, 2015.

[16] C. Wang and K. Ross, “Boosting soft actor-critic: Emphasizing recent experience without forgetting the past,” *arXiv preprint arXiv:1906.04009*, 2019.

Appendix A

LOSS FUNCTION

The trivial solution motivates us to experiment with the current set-up with custom loss function. We introduced two symmetric networks along with the DQN network with added loss functions. The objective of introducing identity loss is to force away the space transformation matrix$l$ and action transformation matrix$K$ from the identity matrix.

$$identity\_loss = \log|\frac{\det(l_1)}{\det(l_2)}|$$

$$\begin{aligned} S_{next} &= State * l_1 \\ State_2 &= S_{next} * l_2 \\ inverse\_loss &= \mathrm{MSE}(State, State_2) \end{aligned} \tag{2}$$