Deep Reinforcement Learning using Structure and Symmetries of the World

Arnab Kumar Mondal
Doctor of Philosophy

School of Computer Science
McGill University
Montreal, Quebec, Canada

October 13, 2024

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

Abstract

Learning meaningful abstract representations of data is an open problem in Deep Reinforcement Learning. The majority of the present algorithms rely on reward signals to learn such representations. In this thesis, we develop a novel perspective to address this problem by leveraging the structure and symmetries of observations and actions. First, we consider a setup where known symmetries of the Markov Decision Process can be used to improve sample efficiency and generalization through Equivariant Neural Networks. However, enforcing layer-wise equivariance can be computationally expensive and can limit the choice of network architecture. We propose a more scalable method to build equivariance while using any existing architecture, based on learning canonical orientations of the data samples. An extension of this idea provides a simple and efficient way to make any pre-trained large model robust to transformations of the input data. We demonstrate the efficacy of this method in different data domains and tasks including Deep Reinforcement Learning, where it improves the generalization performance of any pre-trained agent. Next, we move to a more challenging scenario where both the symmetry groups and their actions on the data are unknown, but where we have an agent whose interactions are given by the transformation of the observations of the environment. We provide methods to learn equivariant representations from interactive environments using novel self-supervised objectives in both Euclidean and non-Euclidean representation spaces. This leads to a latent space that mirrors the structure of the data, while providing a strong inductive bias, resulting in improved sample efficiency. We end by connecting this idea to Koopman's Theory for dynamics modeling, demonstrating its use for building an efficient long-range dynamics model for Reinforcement Learning and Model-based Planning.

Abrégé

L'apprentissage de représentations abstraites significatives des données est un problème ouvert en Apprentissage par Renforcement Profond. La majorité des algorithmes actuels s'appuient sur des signaux de récompense pour apprendre de telles représentations. Dans cette thèse, nous développons une perspective novatrice pour aborder ce problème en exploitant la structure et les symétries des observations et des actions. Tout d'abord, nous considérons un cadre où les symétries connues du Processus de Décision Markovien peuvent être utilisées pour améliorer l'efficacité d'échantillonnage et la généralisation grâce aux Réseaux de Neurones Équivariants. Cependant, l'application de l'équivariance couche par couche peut s'avérer coûteuse en termes de calcul et peut restreindre le choix de l'architecture du réseau. Nous proposons une méthode plus évolutive pour construire l'équivariance tout en utilisant n'importe quelle architecture existante, basée sur l'apprentissage d'orientations canoniques des échantillons de données. Une extension de cette idée fournit un moyen simple et efficace de rendre tout grand modèle pré-entraîné robuste aux transformations des données d'entrée. Nous démontrons l'efficacité de cette méthode dans différents domaines de données et tâches, y compris l'Apprentissage par Renforcement Profond où elle améliore la performance de généralisation de tout agent pré-entraîné. Ensuite, nous passons à un scénario plus complexe où les groupes de symétrie et leurs actions sur les données sont inconnus, mais où nous avons un agent dont les interactions sont données par la transformation des observations de l'environnement. Nous fournissons des méthodes pour apprendre des représentations équivariantes à partir d'environnements interactifs en utilisant de nouveaux objectifs auto-supervisés dans des espaces de représentation euclidiens et non euclidiens. Cela conduit à un espace latent qui reflète la structure des données, tout en fournissant un biais inductif fort, résultant en une amélioration de l'efficacité d'échantillonnage. Nous terminons en reliant cette idée à la Théorie de Koopman pour la modélisation dynamique, démontrant son utilisation pour la construction d'un modèle

dynamique à long terme efficace pour l'Apprentissage par Renforcement et la Planification Basée sur un Modèle.

Contributions to Original Knowledge

This thesis makes novel contributions towards understanding the role of equivariance and transformation-aware algorithms in improving the sample complexity and generalization of deep reinforcement learning. Broadly, it provides two ways to enforce equivariance in the learned representations: first, through new efficient methods to build neural networks, and second, through different loss functions that enforce structure in the latent space. Specifically, I make the following contributions:

1. Formalizing the symmetry in the MDP construct and its connections to equivariance in Action-Value (Q) and policy networks in Deep Reinforcement Learning.

2. Providing empirical evidence of sample efficiency and better generalization in DRL with equivariant architectures.

3. Developing simpler and less constrained ways to design equivariant models using canonicalization of the data, and creating an open-source library for this purpose: https://github.com/arnab39/equiadapt.

4. Introducing novel techniques to learn equivariant representations in interactive environments using loss constraints when the symmetry group is unknown.

5. Proposing efficient long-range dynamics modeling from a Koopman theory perspective and exploring its connections to symmetry learning.

Contributions of Authors

- **Chapters 1 and 2**: These chapters provide the introduction and the necessary technical background for this thesis. Both chapters are written by me.

- **Chapter 3**: This chapter is based on (Mondal et al., 2020), which is presented at the ICML BIG workshop 2020. I led the project, developed the idea, conducted all the experiments, and wrote the entire paper.

- **Chapter 4**: This chapter is based on (Kaba et al., 2023), which is presented at ICML 2023. Oumar and I independently arrived at the idea, with Oumar's version being more general. We collaborated on the theoretical aspects, including some proofs, although Oumar wrote most of the theory section in the paper. I contributed to the writing and handled most of the experiments, including those on images and point clouds.

- **Chapter 5**: This chapter is based on (Mondal et al., 2024), which is presented at NeurIPS 2023. I led the project, conceived the idea, designed the experiments and wrote the majority of the paper. I also ran the point cloud experiments and some initial image experiments.

- **Chapter 6**: This chapter is based on (Mondal et al., 2022), which is presented at ICML 2022. Siamak and I jointly came up with the idea of learning equivariance for RL using Lie Groups. I contributed to writing the paper, designing experiments, and scaling them.

- **Chapter 7**: This chapter is based on (Shakerinava et al., 2022), which is presented at NeurIPS 2022. Siamak came up with the idea of group invariant loss to learn equivariant representations. Siamak and Mehran wrote the initial version of the paper with preliminary experiments to

validate the idea. I contributed to rewriting of the paper and added several new experiments, including applying the idea to RL and world modeling.

- **Chapter 8**: This chapter is based on (Mondal et al., 2023), which is presented at ICLR 2024. I conceived the idea, wrote the code, and ran the dynamics modeling experiments. I also provided Siba with the code for model-free RL experiments and wrote the majority of the paper.

PhD Publications. List of all publications where I was the primary contributor (* denotes joint first author).

Publications that are used to write this thesis:

- **A Mondal**, P Nair, K Siddiqi. *Group Equivariant Deep Reinforcement Learning*, Presented at ICML 2020 Workshop on Inductive Bias, Invariance and Generalization in RL (Chapter 3)
- **A Mondal**, V Jain, K Siddiqi, S Ravanbakhsh *Eqr: Equivariant representations for data-efficient reinforcement learning.* In International Conference on Machine Learning, pp. 15908–15926. PMLR, 2022. (Chapter 6)
- M Shakerinava*, **A Mondal***, S Ravanbakhsh *Structuring representations using group invariants.* Advances in Neural Information Processing Systems, 35:34162–34174, 2022 (Chapter 7)
- SO Kaba*, **A Mondal***, Y Zhang, Y Bengio, S Ravanbakhsh *Equivariance with learned canonicalization functions.* In International Conference on Machine Learning, pp. 15546–15566. PMLR, 2023 (Chapter 4)
- **A Mondal***, SS Panigrahi*, SO Kaba, S Rajeswar, S Ravanbakhsh *Equivariant adaptation of large pretrained models.* Advances in Neural Information Processing Systems, 36, 2023 (Chapter 5)
- **A Mondal**, SS Panigrahi, S Rajeswar, K Siddiqi, S Ravanbakhsh *Efficient dynamics modeling in interactive environments with koopman theory.* In The Twelfth International Conference on Learning Representations, 2024. (Chapter 8)

Publications that are not included but contributed to new knowledge:

- **A Mondal**, D Tome, S Alletto *HumMUSS: Human Motion understanding using State Space Models* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024
- KK Agrawal*, **A Mondal***, A Ghosh*, B Richards *α-ReQ: Assessing Representation Quality in Self-Supervised Learning by measuring eigenspectrum decay.*, Advances in Neural Information Processing Systems, 35, 17626-17638
- **A Mondal***, V Jain*, K Siddiqi. *Mini-batch Similarity Graphs for Robust Image Classification*, In Proceedings of the 2021 British Machine Vision Conference (BMVC 2021). BMVC, 2021

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Kaleem and Siamak. Kaleem, thank you for believing in me and providing me the opportunity to pursue my PhD in your group at McGill. Your nurturing approach, treating each student as your own, has been invaluable. I am particularly grateful for the trust you placed in my abilities, allowing me the freedom to explore areas beyond your immediate research interests. Siamak, your guidance in structuring my thoughts and crafting quality Machine Learning research papers has been instrumental. Joining your group at Mila in my second year marked a significant turning point in my PhD journey, exposing me to a vibrant community of students in Mila with a shared interest in Machine Learning. Moreover, your expertise in Geometric Deep Learning has made a significant positive impact on the quality of research I pursued during my PhD.

I would also like to thank Marc for serving on my PhD committee, and Doina and Robin for providing detailed feedback and constructive comments as the internal and external thesis examiners, respectively. Your insights and evaluations have greatly enhanced the quality of this thesis.

This thesis would not have been possible without the contributions of my collaborators and co-authors: Oumar, Siba, Mehran, Vineet, Pratheeksha, Sai, Yan, and Yoshua. Your support has made the often solitary PhD experience far more manageable and enriching. I've learned immensely from each of you, and your assistance has significantly lightened my workload, allowing me to accomplish more than I could have alone. I owe special thanks to Oumar and Siba, whose drive has been crucial in advancing our research agenda of building architecture-agnostic equivariant models over the past two years.

Beyond my primary research, I extend my heartfelt appreciation to Arna and Kumar Krishna. Our collaboration on the power-law project has significantly enhanced my intuition in representation learning. I am also indebted to the students from Siamak and Kaleem's labs - Victor, Hugo, Tara, Daniel,

Kusha, Christopher, Jikael, Mohammad, Yanan, Bridget, Chris, Megan, Marc, Tabish, Jacob, Ben, Chu, Morteza, and Mahsa - for our stimulating discussions and collaborations.

To my Montreal friends who shared this journey - Raul, Nour, Komal, Arna, Zahraa, Tejas, Ishita, Ayush, Lucie, Solenn, Léa, Sarath, Sarthak, Dishank and Sangnie - thank you for making my PhD experience memorable. Raul, your companionship during the challenging times of COVID-19 was a lifeline, and I am deeply appreciative of your support. Thanks to Komal, Arna and Sarath for celebrating my first birthday away from home in Canada. Many thanks to Tejas and Arna for making the last few months of my PhD journey in Montreal amazing.

I owe immense gratitude to my parents, Sanjib and Luton, who have consistently supported my aspirations, even during financial hardships. As their only child, their decision to let me pursue my dreams far from home was a profound act of love and sacrifice. To Danielle, thank you for bringing joy to my workaholic life and for your active support in open-sourcing my PhD projects. Your love and encouragement have been invaluable.

Lastly, I want to acknowledge my family members who have taken such pride in my achievements as the first PhD holder in our family. Your support has emphasized the importance of research and higher education for our future generations. This PhD journey has been transformative, and I am profoundly grateful to everyone who has been part of it.

Contents

List of Figures

List of Tables

Part I

Introduction and Background

1

Introduction

Deep Reinforcement Learning (DRL) stands at the forefront of artificial intelligence research, providing a powerful framework for agents to learn and optimize behaviors directly from complex, high-dimensional inputs (Silver et al., 2016; Mnih et al., 2015b; François-Lavet et al., 2018; Levine et al., 2016a). This thesis advances the field of DRL by leveraging the intrinsic symmetries and structures observed in real-world environments to enhance both the efficiency and generalizability of learning algorithms.

Emergence and Significance of Deep Reinforcement Learning The fusion of deep learning with traditional reinforcement learning frameworks has catalyzed substantial advancements across diverse domains. By enabling direct learning from raw sensory data, Deep Reinforcement Learning (DRL) addresses complex decision-making problems characterized by high-dimensional state or observation spaces, previously infeasible with conventional methods (Schulman et al., 2017c; Haarnoja et al., 2018b).

DRL has achieved superhuman performance in various gaming environments. Notable achievements include mastering Atari games (Mnih et al., 2013c), where DRL algorithms surpassed human performance by learning directly from pixel inputs, and achieving unprecedented success in the game of Go (Silver et al., 2016), which was considered a grand challenge due to

its vast search space and strategic depth. DRL has also demonstrated exceptional prowess in complex multiplayer games such as Dota 2 (Berner et al., 2019), showcasing its ability to handle dynamic and cooperative-competitive scenarios. In robotics, DRL has enabled the development of agents capable of performing intricate manipulation tasks (Levine et al., 2016b), thereby pushing the boundaries of robotic dexterity and adaptability. In the realm of autonomous navigation, DRL has revolutionized self-driving car algorithms (Shalev-Shwartz & Shashua, 2016), optimizing decision-making processes for safe and efficient navigation in real-world environments (Bojarski et al., 2016).

Beyond gaming and robotics, DRL has made significant strides in scientific discovery and healthcare. For instance, in drug discovery, DRL has been employed to predict molecular interactions, expediting the identification of potential therapeutic compounds (Zhavoronkov et al., 2019; Popova et al., 2018). This application underscores the potential of DRL to accelerate research and innovation in critical scientific fields.

Challenges in Deep Reinforcement Learning Despite its promise, DRL faces critical challenges in sample efficiency and generalization that hinder its practical deployment. The issue of sample inefficiency arises because many DRL algorithms use reward as the only signal for representation learning in contexts with high dimensional states and actions which leads to tremendous data inefficiency. Notably, almost all success stories of RL rely on vast amounts of simulation data or the number of interactions with the environment to learn effective policies. This makes them impractical for scenarios where data collection is expensive or slow (Kaiser et al., 2019a; Agarwal et al., 2020). Moreover, these algorithms struggle to generalize to new environments or even slight transformations of the observation space, limiting their applicability in dynamic real-world settings (Henderson et al., 2018; Cobbe et al., 2019). This generalization problem is compounded by overfitting to specific tasks or configurations encountered during training, thereby failing to perform well in unseen scenarios (Justesen & Risi, 2018; Zhang et al., 2020a). Furthermore, the dependency on large amounts of data and computational resources for training not only escalates costs but also restricts the scalability of DRL solutions (Dulac-Arnold et al., 2019; Liang et al., 2018). Addressing these issues requires innovations in algorithm design to enhance learning efficiency and robustness using stronger inductive biases(Bengio et al., 2013a), enabling broader deployment of DRL in diverse applications.

Leveraging Structure and Symmetries This research posits that a deep understanding of the symmetries in the DRL agent's observation data and actions can effectively address these challenges. Symmetry, in a mathematical sense, implies that certain transformations of the environment—such as rotations, reflections, and translations—do not alter the underlying realities of the environment. By encoding these symmetrical properties into DRL algorithms as prior knowledge, one can significantly improve sample efficiency and enable robust generalization across varying environmental configurations.

Building on the concept of equivariance, which dictates that transformations of the input should predictably alter the output, this thesis develops DRL algorithms that inherently recognize and utilize these transformations. This improves sample efficiency because observing a specific trajectory or triplet (s, a, s') effectively provides the agent with the information of all its transformations. Such an approach not only enhances learning efficiency by reducing the need for relearning under new transformations but also mirrors human cognitive abilities to generalize from known situations to new ones. For instance, a driver accustomed to left-hand traffic can adapt to right-hand traffic with relative ease, a conceptually analogous behavior that DRL systems can emulate to adjust to new environments quickly.

Research Objectives The primary aim of this research is to harness the power of equivariance and transformation-aware learning to make DRL models both data-efficient and generalizable across the transformation of the data. The objectives are as follows:

- To identify and formalize the relevant transformations and symmetries in the context of the DRL agent's observations and actions.
- To design novel efficient neural networks that inherently account for these transformations, thereby maintaining consistent outputs across varied observations of the same phenomena.
- To develop novel loss functions and training methodologies that leverage the structure and symmetry in the data, moving beyond equivariant neural networks to enhance learning.
- To empirically validate the effectiveness of these models across diverse application domains including DRL, demonstrating their enhanced generalization capabilities and efficiency.

1.1 Thesis Outline

The thesis is organized into four main parts. Part I starts with an introduction and then provides background that is relevant to the rest of the thesis, while Parts II and III each address specific aspects of utilizing structure and symmetries in DRL. Part IV concludes the thesis.

Part I: Introduction and Background This part delves into the essential background, covering Machine and Deep learning, Groups and their representation theory, Markov Decision Processes (MDPs), DRL and Koopman Theory. It introduces the concept of equivariance in deep learning and symmetry in MDPs, which forms the cornerstone of the subsequent chapters.

Part II: Leveraging Known Symmetry Transformations This part starts with equivariant representation learning in DRL by modifying neural network architectures where the symmetry group is known. Subsequent chapters go beyond Reinforcement Learning and provide a more scalable general technique to build architecture-agnostic equivariance in different data domains and tasks. In this part, we assume that the both the symmetry group and its action on the input space are known.

- **Chapter 3:** Introduces Group Equivariant Deep Reinforcement Learning, discussing related work and foundational concepts like E(2)-equivariant convolution (Weiler & Cesa, 2019c). It presents the design and implementation of equivariant deep Q-networks and vectorized policies.

- **Chapter 4:** Focuses on the design of a novel method to achieve equivariance through canonicalization. It details the design of canonicalization functions for Euclidean groups and demonstrates their application through various experiments, including image classification, N-body dynamics prediction, point cloud classification, and segmentation.

- **Chapter 5:** Extends the idea to adapt large pretrained models to be equivariant with minimal modifications. It explores the problems with canonicalization while adapting it to pre-trained architectures, presenting experimental results on image and point cloud domains, and demonstrating the performance improvements and robustness of the proposed methods. An extension of this idea is also proposed to make a pretrained agent robust to transformations of the observations/ states.

Part III: Learning Structured Representations This part focuses on learning equivariant representations from data without architectural constraints using novel loss functions, fitting the general theme of self-supervised representation learning while maintaining a structured representation space. In this part we do not assume that the symmetry group or its action in the input space are known. We end this part by introducing a technique to learn linear latent structure using Koopman's Theory for long range dynamics modeling in RL and planning.

- **Chapter 6:** Proposes novel methods for learning structured representations in DRL. It introduces the concept of equivariant representations using loss constraints on Lie groups, focusing on parameterizing latent embeddings to maintain equivariance under continuous transformations. The application of these techniques to Atari games showcases improved performance and sample efficiency.

- **Chapter 7:** Provides an alternate approach, proposing symmetry regularization objectives to enhance the learning of equivariant representations using group invariants. This approach leverages invariants of transformation groups to enforce consistent and structured embeddings, critical for effective policy learning.

- **Chapter 8:** Extends the ideas of structured representation learning by modeling the dynamics as an action of a linear operator using Koopman Theory. Unlike previous chapters, it motivates representation learning in interactive environments from a control theory perspective. It discusses the use of Koopman operators for efficient long-range dynamics modeling, demonstrating the integration of these techniques into model-based planning and RL frameworks. Empirical results highlight the advantages of Koopman-based models in terms of stability and computational efficiency, particularly for long-horizon predictions.

Part IV: Conclusion This part concludes the thesis, summarizing the key findings and contributions, discussing their implications, and outlining potential future research directions.

1.2 Preview of Main Results and Insights

This section provides a concise overview of the significant findings and insights gained from the research in this thesis. These insights are based on the comprehensive analysis and experiments detailed in the subsequent chapters.

1. **Equivariant Q-Networks and Policy Networks leads to data-efficiency and generalization (Chapters 3 and 4)**: A significant advancement in this research is the motivation and implementation of equivariant Q-networks and policy networks. These networks leverage the structural and symmetrical properties of the environment to dramatically improve both data efficiency and the ability to generalize across different environments. By incorporating these properties into the network design, the solution space of the Markov Decision Process (MDP) is effectively constrained, leading to a minimized MDP defined by its symmetry. This approach marks a major step forward in reducing the number of environment simulations required to learn a policy, thus making deep reinforcement learning more practical and applicable..

2. **Canonicalization facilitates scaling of Equivariant Deep Neural Network in an efficient way (Chapters 4 and 5)**: The development of the canonicalization technique, as detailed in Chapters 4 and 5, enhances the efficiency of equivariant models and ensures their robustness against various transformations. This improvement is achieved by using a more expressive, non-equivariant architecture that is optimized for canonical data orientations. Such a strategy notably reduces the computational demands typically associated with layer-wise equivariance, yet it preserves robustness and ensures high generalization across diverse tasks in both image and point cloud domains. The effectiveness of this technique has been empirically confirmed through rigorous testing in a range of applications, including classification, segmentation, dynamics modeling, and decision-making. Furthermore, we provide theoretical insights to demonstrate the universality of this technique and its ability to relax the strict architectural constraints traditionally required for equivariance. This technique can be adapted to and integrated with any large, pretrained architecture that is now increasingly common in the field, and this yieelds promising results across various data domains and tasks.

3. **Improved sample efficiency with equivariant representation learning methods (Chapters 6 and 7)**: In Chapters 6 and 7, we introduce novel self-supervised objectives for learning equivariant representations in both Euclidean and non-Euclidean spaces. Our research demonstrates that mere equivariance is not adequate; the symmetry group's actions must be linear in the latent space to enhance performance and generalization. Our results indicate that the inductive bias introduced by these structured equivariant representations through linear actions substantially improves the sample efficiency of deep reinforcement learning (DRL) agents. This innovative approach creates a latent representation space that more accurately mirrors the structure of the data compared to existing contrastive and non-contrastive self-supervised representation learning techniques.

4. **Application of Koopman Theory results in efficient dynamics modeling of complex environments (Chapter 8)**: Koopman theory offers a theoretical foundation for modeling state transitions in interactive environments as an action of a Linear Koopman Operator in the latent representation space. The use of these linear dynamics models facilitates efficient long-range dynamics modeling. Additionally, integrating these models with deep reinforcement learning and planning algorithms that utilize dynamic modeling — discussed in detail in Chapter 8 — enhances their speed, training stability, sample efficiency, and overall performance.

2

Background

This chapter lays the foundation for understanding the concepts explored in subsequent chapters of this thesis. We begin in Section 2.1 with a concise overview of machine learning and deep learning. While not exhaustive, this section serves as a starting point for newcomers to the field. Next, Section 2.2 introduces the fundamental principles of groups and their representation theory. This knowledge is then applied to the realm of deep learning in Section 2.3, providing crucial insights for all chapters except Chapter 8. In Section 2.4, we discuss Markov Decision Processes, their homomorphisms, and how these concepts relate to the notion of symmetry or groups. This section, along with Section 2.5, forms the theoretical backbone for Chapters 3, 5, 6, 7, and 8. Lastly, Section 8.1.1 offers an introduction to Koopman's Theory for dynamical systems. While this section is primarily relevant to Chapter 8, readers may find it enriching for a broader understanding of dynamical systems and their modeling. By progressing through the sections in the present Chapter, readers will gain the necessary background and theoretical grounding to navigate the advanced concepts presented in the remainder of this thesis. We also provide additional background when necessary in subsequent chapters.

2.1 Machine Learning and Deep Learning

Machine Learning Machine learning (ML) is a branch of artificial intelligence focused on building systems that learn from data, identify patterns, and make decisions with minimal human intervention (Mitchell, 1997). In general, machine learning algorithms are divided into three main categories: supervised learning, unsupervised learning, and reinforcement learning (Bishop, 2006). Supervised learning algorithms are trained using labeled data, where the correct output is provided for each input sample, allowing the model to learn a mapping from inputs to outputs (Vapnik, 1998). At their core, supervised learning algorithms aim to approximate a function that maps input features to output labels, utilizing labeled datasets for training (Hastie et al., 2009). This approximation problem is typically posed as an optimization problem where the goal is to find the best parameters that minimize the difference between the predicted and true outputs across the dataset (Friedman et al., 2001). Unsupervised learning explores patterns in unlabeled data. Recently, this field has evolved into self-supervised learning, where the data itself serves as supervision through innovative loss functions (Liu et al., 2021b). This approach eliminates the need for manual labeling and effectively utilizes large, unannotated datasets to enhance learning outcomes. Finally, Reinforcement learning involves agents that learn to make decisions by performing actions in an environment to maximize some notion of cumulative reward (Sutton & Barto, 2018).

Deep Learning Deep learning, a subset of machine learning, utilizes neural networks that are inspired by the structure and function of the human brain. These networks are parametric models composed of layers of interconnected nodes, each representing a non-linear transformation of its inputs. The training of deep neural networks involves adjusting the weights of the connections between neurons or "parameters of the neural network" on the basis of the error in the output compared to the expected result. This process, known as backpropagation, uses gradient descent to minimize some form of the loss function that measures the discrepancy between the actual output of the network and the desired output (Rumelhart et al., 1986). The ability of deep networks to learn effective representations of data without the need for manual feature extraction is one of the key reasons for their success in various domains.

The scope of deep learning applications has expanded dramatically with

the advent of generative AI models, revolutionizing areas such as content creation, personalized medicine, and autonomous systems. Diffusion and transformer-based models are at the forefront, driving innovations in image generation, natural language processing, and beyond (Goodfellow et al., 2014; Brown et al., 2020b; Vaswani et al., 2017a). For instance, deep learning now enables the synthesis of realistic images and videos, complex natural language and code understanding and generation, and the creation of dynamic virtual environments. In the medical field, deep learning models facilitate drug discovery and personalized treatment plans by analyzing vast datasets that were previously intractable (Senior et al., 2020). Furthermore, reinforcement learning, a subset of deep learning, has made significant strides in training autonomous vehicles and sophisticated robotics, where agents learn optimal behaviors in complex, unpredictable environments (Vinyals et al., 2019; Silver et al., 2017b).

In practical applications, deep learning models are designed to exploit the inherent structures in data. For instance, convolutional neural networks (CNNs) leverage the spatial hierarchies in image data (Krizhevsky et al., 2012), while recurrent neural networks (RNNs) exploit the temporal dynamics in sequential data (Sutskever et al., 2014). These specialized architectures not only improve learning efficiency and model performance but also extend the types of functions that can be effectively approximated.

Next, we review groups and their representation theory before we discuss how they are used to improve robustness and sample efficiency in deep learning.

2.2 Groups and their representation theory

Groups A group is a fundamental mathematical concept that captures the notion of symmetry. Groups play a crucial role in many areas of mathematics and physics, and have applications in computer science, cryptography, and machine learning. Formally, a group $\mathcal{G} = \{g\}$ is a set equipped with an associative binary operation, such that the set is closed under this operation, and each element $g \in \mathcal{G}$ has a unique inverse, such that their composition gives the identity $g^{-1}g = e$. A group helps us define a set of invertible transformations that has some structure.

Subgroups Any subset $\mathcal{G}' \leq \mathcal{G}$ that is closed under the binary operation of the groups forms a subgroup. A subgroup $\mathcal{N} \leq \mathcal{G}$ is called a *normal subgroup* if it is invariant under conjugation by elements of $\mathcal{G}$, that is, $g\mathcal{N}g^{-1} = \mathcal{N}$ for

all $g \in \mathcal{G}$. We denote this as $\mathcal{N} \trianglelefteq \mathcal{G}$. A normal subgroup $\mathcal{N}$ of a group $\mathcal{G}$ can be used to decompose $\mathcal{G}$ into a semidirect product $\mathcal{N} \rtimes \mathcal{H}$, where $\mathcal{H}$ acts on $\mathcal{N}$ by conjugation: for $h \in \mathcal{H}$[1] and $n \in \mathcal{N}$, we have $hnh^{-1} \in \mathcal{N}$.

Group Action A group $\mathcal{G}$ can *act* on a set $\mathcal{X}$ by transforming its elements $x \in \mathcal{X}$ through a bijection. We use $\alpha_{\mathcal{X}} : \mathcal{G} \times \mathcal{X} \mapsto \mathcal{X}$ to denote the *group action*, and for brevity we interchangeably use $\alpha_{\mathcal{X}}(g, x)$ and $g \cdot x$ to denote it. The action $\alpha_{\mathcal{X}}$ is *faithful* to $\mathcal{G}$ if transformations of $\mathcal{X}$ using each $g \in \mathcal{G}$ are unique – *i.e.*, $\forall g, g' \; \exists x \in \mathcal{X}$ s.t. $\alpha_{\mathcal{X}}(g, x) \neq \alpha_{\mathcal{X}}(g', x)$. The action captures some of the structure of $\mathcal{G}$ due to two constraints – the identity element acts trivially $e \cdot x = x$; and the composition of actions is equal to the action of the composition, i.e., $(gg') \cdot x = g \cdot (g' \cdot x), \forall g, g' \in \mathcal{G}$. $\mathcal{X}$ is then called a $\mathcal{G}$-set. For example, if $\mathcal{G} = SO(2)$ is the group of 2D rotations, its action α_x on any image $x \in \mathcal{X}$ could rotate it around some center, or it could perform a horizontal translation with a circular boundary. The amount of rotation or translation is identified by choice of $g \in \mathcal{G}$, while the choice of transformation itself (rotation vs. translation) is defined by the action α_x. Here, the underlying abstract group $SO(2)$ dictates the composition rule for the transformations.

Cosets Let $\mathcal{H}$ be a subgroup of $\mathcal{G}$. The left (right) *coset* of $\mathcal{H}$ in $\mathcal{G}$ is the set $g\mathcal{H} = \{gh \mid h \in \mathcal{H}\}$ ($\mathcal{H}g = \{hg \mid h \in \mathcal{H}\}$) for $g \in \mathcal{G}$. We denote the set of the left (right) cosets of $\mathcal{H}$ in $\mathcal{G}$ as $\mathcal{G}/\mathcal{H}$ ($\mathcal{H}\backslash\mathcal{G}$).

Orbits and Stabilizers Any $\mathcal{G}$-action partitions $\mathcal{X}$ into *orbits* $x^{\mathcal{G}} = \{g \cdot x \mid g \in \mathcal{G}\}$, and we denote the set of orbits under $\mathcal{G}$-action as $\mathcal{X}/\mathcal{G}$. A $\mathcal{G}$-action is transitive if and only if its action results in a single orbit. The *stabilizer* of an element $x \in \mathcal{X}$ under the action of $\mathcal{G}$ is the subgroup of $\mathcal{G}$ that leaves x fixed, that is, $\text{Stab}_{\mathcal{G}}(x) = \{g \in \mathcal{G} \mid g \cdot x = x\}$. The stabilizer is also known as the *isotropy subgroup* or the *fixer* of x.

Group Representation Representation theory studies how groups act on vector spaces, and their elements can be represented as linear transformations in these spaces. A representation of a group $\mathcal{G}$ is a homomorphism from $\mathcal{G}$ to the group of invertible linear transformations on a vector space $\mathcal{V}$. We use $\rho(\mathcal{G})$ to denote it, and $\rho(g) : \mathcal{V} \to \mathcal{V}$ is the representation of $g \in \mathcal{G}$. For any two group elements g_1 and g_2, the corresponding linear transformations $\rho(g_1)$ and $\rho(g_2)$ on $\mathcal{V}$ satisfy $\rho(g_1 g_2) = \rho(g_1)\rho(g_2)$, and the identity element e of $\mathcal{G}$

[1]The symbol h which is used to denote a group element, is later repurposed in the text to represent a function. However, the context of the chapter will help differentiate it.

corresponds to the identity transformation on V.

Trivial and Regular Representations Two fundamental types of representations are the trivial representation and the regular representation. The trivial representation maps every element of the group to the identity transformation on a one-dimensional vector space. Formally, for a group $\mathcal{G}$, the trivial representation ρ_{triv} is defined as $\rho_{\text{triv}}(g) = 1$ for all $g \in \mathcal{G}$. On the other hand, the regular representation is a higher-dimensional representation that captures the full structure of the group. For a finite group $\mathcal{G}$, the regular representation acts on a vector space with a dimension equal to the order of the group, where each basis vector corresponds to a group element. The action of the group in this space is defined by left multiplication: for $g, h \in \mathcal{G}$, $\rho_{\text{reg}}(g)e_h = e_{gh}$, where e_h is the basis vector corresponding to h.

Irreducible Representation An irreducible representation of a group $\mathcal{G}$ is a representation that cannot be decomposed into two nontrivial sub-representations, where a sub-representation is a subspace of the original vector space that is invariant under the action of $\mathcal{G}$. In other words, an irreducible representation captures the most fundamental structure of a group's action on a vector space and cannot be further simplified. For example, the irreducible representations of $SO(2)$ can be described using circular harmonics, which are complex-valued functions that are eigenfunctions of the angular momentum operator. They are labeled by an integer m, and associated with functions $Y_m(\theta)$, where θ is the angle of rotation. These functions, called circular harmonics, can be expressed as complex exponentials of the form $e^{im\theta}$.

Induced Representation Induced representations serve as a pivotal method in representation theory to extend a representation from a subgroup to the entire group. Consider a subgroup $\mathcal{H}$, which is a subset of a larger group $\mathcal{G}$. Let ρ_H be a representation of $\mathcal{H}$ on a vector space $\mathcal{V}_H$. The process of inducing this representation to $\mathcal{G}$ results in a new representation, $\rho_G = \text{Ind}_H^G \rho_H$, defined on an expanded vector space $\mathcal{V}_G$.

To construct $\mathcal{V}_G$, we consider functions from $\mathcal{G}$ to $\mathcal{V}_H$. These functions are chosen so that they transform covariantly under the action of $\mathcal{H}$. Specifically, for a function f, the requirement is that $f(gh) = \rho_H(h^{-1})f(g)$ for any h in $\mathcal{H}$ and g in $\mathcal{G}$. Here, $\mathcal{G}$ acts on these functions through left translation, meaning $(\rho_G(g)f)(x) = f(g^{-1}x)$. This action effectively broadens the influence of ρ_H from $\mathcal{H}$ to the whole group $\mathcal{G}$, illustrating how local symmetries of a subgroup can extend to global actions within the larger group.

In the case of the $\mathcal{G} = E(2)$ group, which represents the Euclidean isometries of the 2D plane (including translations, rotations, and reflections), we can explore an example where $\mathcal{H} \leq O(2)$, a subgroup involving rotations. Assume ρ_H is a representation of $\mathcal{H}$ that acts on a vector space $\mathcal{V}_H$ by rotating vectors.

To induce this representation to $E(2)$, represented as $T(2) \rtimes O(2)$ (where $T(2)$ is the translation group and $O(2)$ includes rotations and reflections), we expand to $\mathcal{V}_G$. Here, $\mathcal{V}_G$ consists of functions mapping from the plane $\mathbb{R}^2$ (accounting for all translations in $T(2)$) to $\mathcal{V}_H$. The required covariance under the subgroup's action transforms into the condition: for any translation t and rotation g, the function f satisfies $\left(\rho_G(tg)f\right)(x) = \rho_H(g) \cdot f(g^{-1}(x-t))$.

This construction elegantly expands the vector space to encompass not only the transformations of $\mathcal{H}$ but also the translations, effectively elevating the subgroup's local symmetry to a comprehensive global action across the entire Euclidean group. The power of induced representations becomes evident in their practical applications to machine learning, such as in building E(2)-Steerable Convolutional Neural Networks (Weiler & Cesa, 2019c). These advanced architectures leverage group symmetries to handle transformations under the $E(2)$ group, significantly enhancing robustness and generalization in tasks like image understanding. We discuss this more in the next Chapter.

Parameterizing Lie Groups If we assume $\mathcal{G}$ is any sub-group of a *classical Lie group* over $\mathbb{R}$, then it can be represented using invertible matrices. We use $\rho(\mathcal{G})$ to denote a linear representation of $\mathcal{G}$, and $\rho_g : \mathbb{R}^D \to \mathbb{R}^D$ for the action (a.k.a. the representation) of $g \in \mathcal{G}$. Many such Lie groups are identifiable by their infinitesimal generators, their *Lie algebra* $\mathfrak{g} = Lie(\rho(\mathcal{G}))$.[2] This connection enables a simple parameterization of $\rho(\mathcal{G})$ using a set of linear bases $\{\mathbf{E}^{(i)}\}_i$ for their Lie algebra – that is $\rho_g = \exp\left(\sum_i \beta_{g,i}\, \mathbf{E}^{(i)}\right)$, where $\exp(\mathbf{Y}) = \sum_{j=0}^{\infty} \frac{\mathbf{Y}^j}{j!}$ is the matrix exponential. We refer to this parameterization later in Section 6.1 of Chapter 6. Such linear representations in the form of invertible matrices can be used for both continuous transformations (*e.g.*, 3D rotations) and finite groups (*e.g.*, $\times 90^\circ$ rotations).

2.3 Equivariance in Deep Learning

In the context of machine learning, we are interested in designing models that are invariant or equivariant to the input data's symmetries. This is where

[2]This relation is bijective for "simply connected" Lie groups.

group representations and equivariant functions come into play. Suppose we have a set of input data $\mathcal{X}$ with some symmetry structure that we would like to preserve in our machine learning model. We assume that this symmetry structure can be described by a group $\mathcal{G}$, which acts on $\mathcal{X}$ through a group representation $\rho : \mathcal{G} \to \mathcal{T}$, where $\mathcal{T} \subset \mathrm{GL}(\mathcal{X})$ is the set of invertible linear transformations on $\mathcal{X}$. The group representation ρ assigns to each element g of $\mathcal{G}$ a transformation $\rho(g) \in \mathcal{T}$ that acts on $\mathcal{X}$.

An equivariant function $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$ is a function that commutes with the group action of $\mathcal{G}$ on $\mathcal{X}$ and $\mathcal{Y}$, i.e., for all $g \in \mathcal{G}$ and $\mathbf{x} \in \mathcal{X}$, we have:

$$\boldsymbol{f}(\rho(g) \cdot \mathbf{x}) = \rho'(g) \cdot \boldsymbol{f}(\mathbf{x}) \tag{2.1}$$

where $\rho' : \mathcal{G} \to \mathcal{T}'$ is another group representation that acts on the output space $\mathcal{Y}$. If ρ' is the trivial representation, i.e., $\rho'(g) = I$ or identity transformation for all $g \in \mathcal{G}$, then we say that $\boldsymbol{f}$ is invariant to the group action of $\mathcal{G}$.

In other words, an equivariant function is a function whose output changes in a predictable way under transformations of the input induced by the group $\mathcal{G}$. In context of the data, the orbit $\mathbf{x}^{\mathcal{G}}$ of $\mathbf{x}$ under the action of $\mathcal{G}$ is the set of all possible transformations of $\mathbf{x}$ under the group action of $\mathcal{G}$. The set of all orbits in $\mathcal{X}$ is denoted by $\mathcal{X}/\mathcal{G}$ and forms a partition of $\mathcal{X}$. The goal of equivariant neural networks is to exploit this structure by sharing weights across the orbits to reduce the number of parameters needed to learn the function f. By doing so, one can leverage the symmetry present in the data to improve the generalization performance of the model.

One common example of an equivariant network is the convolutional neural network (CNN), which is designed to be equivariant to translation, meaning that if the input image is translated, the output of the convolution operation will also be translated predictably. The idea of equivariance can be generalized to any symmetry transformation group, but it requires careful design of the different layers of the neural network architecture to ensure that the equivariance property is preserved. In addition to the translation equivariance provided by traditional CNNs, recent advancements have extended equivariance in CNNs to more complex transformations including rotations, scale changes, and other symmetries (Cohen & Welling, 2016b, 2017; Cohen et al., 2018, 2019a; Worrall et al., 2017a; Weiler & Cesa, 2019b; Weiler et al., 2018).

Similarly, permutation equivariance has been incorporated into various architectures, including Deep Sets (Zaheer et al., 2017b; Lee et al., 2019) for handling unordered data collections, Graph Neural Networks (Battaglia et al., 2018; Kipf & Welling, 2017; Gilmer et al., 2017b) for processing node-permutation invariant graph structures, and Transformers (Vaswani et al., 2017b; Devlin et al., 2018; Brown et al., 2020a) through their self-attention mechanism. Recent work has further explored permutation-equivariant architectures in point cloud processing (Qi et al., 2017a; Wang et al., 2019b), set-to-set learning (Vinyals et al., 2015; You et al., 2018), and multi-agent systems (Sukhbaatar et al., 2016; Jiang et al., 2018).

Furthermore, equivariant models have been applied to other domains such as physics-based simulations (Satorras et al., 2021) and molecular property prediction (Anderson et al., 2019; Thomas et al., 2018). These models leverage symmetries inherent in physical systems to improve generalization and sample efficiency, demonstrating the broad applicability of equivariance principles across various scientific and engineering fields.

2.4 Markov Decision Processes

A Markov Decision Process (MDP) is a mathematical framework used for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker. MDPs are useful for studying optimization problems solved via dynamic programming and reinforcement learning. We define an MDP by the 5-tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, R, T, \gamma \rangle$, where:

$\mathcal{S}$ is the set of all possible states in the environment.

$\mathcal{A}$ represents the set of all actions that the decision maker can choose from.

$R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, which assigns a real number to each state-action pair, indicating the immediate payoff received after performing an action in a particular state.

$T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^{\geq 0}$ is the state transition function, which provides the probability distribution over possible next states, given the current state and action. This function satisfies the condition $\sum_{s' \in \mathcal{S}} T(s, a, s') = 1$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, ensuring that the probabilities are properly normalized.

$\gamma \in [0, 1]$ is the discount factor and is included in MDP formulations to account for the preference for immediate rewards over future rewards. The discount factor is a number between 0 and 1 and is used to reduce the value of rewards received in the future, reflecting their decreased utility as compared

to immediate rewards. A lower discount factor results in myopic planning.

2.4.1 Reinforcement Learning

The RL setting can be formalized as a Markov Decision Process (MDP), where an agent learns to make decisions by interacting with an environment. The agent's goal is to maximize a cumulative reward signal. The agent's *policy* $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^{\geq 0}$, denoted by $\pi(a|s)$, defines the probability distribution over actions at a given state. The agent's objective is to learn a policy $\pi(a|s)$ that maximizes the expected cumulative reward, also known as the return:

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \tag{2.2}$$

where s_t and a_t are the state and action at time step t, respectively.

The agent's expected cumulative reward can also be represented by the value function, which is defined as the expected cumulative reward starting from a given state and following a given policy. The value function at state s under policy π is denoted by $V^{\pi}(s)$ and can be computed using the *Bellman Equation* given by:

$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi(a|s)}\left[R(s,a) + \gamma \sum_{s'} T(s'|s,a)V^{\pi}(s')\right] \tag{2.3}$$

Similarly, the action-value function after taking action a at state s and then following policy π is denoted by $Q^{\pi}(s,a)$, and can be computed using the *Bellman Equation* given by:

$$Q^{\pi}(s,a) = R(s,a) + \mathbb{E}_{a'\sim\pi(a|s)}\left[\gamma \sum_{s'} T(s'|s,a)Q^{\pi}(s',a')\right] \tag{2.4}$$

The optimal policy is given by the actions that maximize the optimal action-value function $Q^*(s,a)$ at every state and is denoted as $\pi^*(a|s)$.

2.4.2 MDP Homomorphism

In this section, we ignore the discount factor for brevity by assuming it to be 1. For two MDPs $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, R, T\rangle$ and $\bar{\mathcal{M}} = \langle \bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{R}, \bar{T}\rangle$, an MDP homomorphism can be defined as a tuple $\mathcal{H} = \langle h_{\mathcal{S}}, h_{\mathcal{A}}\rangle$ where $h_{\mathcal{S}} : \mathcal{S} \to \bar{\mathcal{S}}$ is the state mapping and $h_{\mathcal{A}} : \mathcal{S} \times \mathcal{A} \to \bar{\mathcal{A}}$ is the state dependent action mapping. These two mappings satisfy the following invariance and equivariance conditions:

(1) Invariance of the reward:

$$\bar{R}(h_{\mathcal{S}}(s), h_{\mathcal{A}}(s,a)) = R(s,a), \quad \forall s, a \in \mathcal{S} \times \mathcal{A} \tag{2.5}$$

(2) Equivariance of the deterministic transition model under the agent's action:

$$\bar{T}(h_{\mathcal{S}}(s), h_{\mathcal{A}}(s,a)) = h_{\mathcal{S}}(T(s,a)), \quad \forall s, a \in \mathcal{S} \times \mathcal{A} \tag{2.6}$$

A probabilistic variation of the above equation for a stochastic MDP (Bloem-Reddy & Teh, 2020) is:

$$\bar{T}(h_{\mathcal{S}}(s') \mid h_{\mathcal{S}}(s), h_{\mathcal{A}}(s,a)) = \sum_{s'' \in [s']_h} T(s'' \mid s, a) \tag{2.7}$$

for all $s, s' \in \mathcal{S}, a \in \mathcal{A}$, where $[s']_{h_{\mathcal{S}}} = {h_{\mathcal{S}}}^{-1}(h_{\mathcal{S}}(s'))$ is the equivalence class of s' under $h_{\mathcal{S}}$.

In related literature, MDP homomorphism is often used for *minimization* of the MDP, because the optimal policy of $\bar{\mathcal{M}}$ can be lifted to obtain the optimal counterparts for $\mathcal{M}$. Moreover, the optimal action-value function of a state action pair in $\mathcal{M}$ is equal to the optimal action-value function of the mapped state-action pair in $\bar{\mathcal{M}}$, that is:

Theorem 2.4.1. ***(Ravindran & Barto, 2001)*** *Let $\mathcal{H} = \langle h_{\mathcal{S}}, h_{\mathcal{A}} \rangle$ be an* MDP homomorphism *from $\mathcal{M}$ to $\bar{\mathcal{M}}$. Then for any $(s,a) \in \mathcal{S} \times \mathcal{A}$, we have:*

$$Q^*(s,a) = \bar{Q}^*(h_{\mathcal{S}}(s), h_{\mathcal{A}}(s,a)).$$

Lifted Policy Given a policy on the image MDP, we can define a lifted policy on the original MDP that has similar behaviour. Let $\mathcal{H} = \langle h_{\mathcal{S}}, h_{\mathcal{A}} \rangle$ be an *MDP homomorphism* from $\mathcal{M}$ to $\bar{\mathcal{M}}$, and let $\bar{\pi} : \bar{\mathcal{S}} \times \bar{\mathcal{A}} \to \mathbb{R}^{\geq 0}$ be a policy on $\bar{\mathcal{M}}$. Then $\bar{\pi}$ lifted to $\mathcal{M}$ is a policy $\pi^{\uparrow} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^{\geq 0}$ such that for any $(s,a) \in \mathcal{S} \times \mathcal{A}$, we have:

$$\pi^{\uparrow}(a|s) = \frac{\bar{\pi}(h_{\mathcal{A}}(s,a)|h_{\mathcal{S}}(s))}{|\{h_{\mathcal{A}}^{-1}(s)(h_{\mathcal{A}}(s,a))\}|}$$

For these results to hold, it is sufficient for the lifted policy to satisfy:

$$\sum_{a \in \{h_{\mathcal{A}}^{-1}(s)(h_{\mathcal{A}}(s,a))\}} \pi^{\uparrow}(a|s) = \bar{\pi}(h_{\mathcal{A}}(s,a)|h_{\mathcal{S}}(s)) \quad \forall s \in \mathcal{S} \tag{3}$$

and in order to make the lifted policy unique, Ravindran and Barto (2001) choose to uniformly spread the probability of taking $h_{\mathcal{A}}(s,a)$ from $h_{\mathcal{S}}(s)$ across all actions a' satisfying $h_{\mathcal{A}}(s,a) = h_{\mathcal{A}}(s,a')$. Then, we have the lifted policy of the optimal policy of $\mathcal{M}$ is an optimal policy for $\bar{\mathcal{M}}$:

Theorem 2.4.2. ***(Ravindran & Barto, 2001)*** *Let $\mathcal{H} = \langle h_{\mathcal{S}}, h_{\mathcal{A}} \rangle$ be an* MDP homomorphism *from $\mathcal{M}$ to $\bar{\mathcal{M}}$, and let $\bar{\pi}^* : \bar{\mathcal{S}} \times \bar{\mathcal{A}} \to \mathbb{R}^{\geq 0}$ be an optimal policy on $\mathcal{M}$. Then for the original* MDP *$\mathcal{M}$, the lifted policy $\pi^{\uparrow} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^{\geq 0}$ is an optimal policy - π^*.*

2.4.3 Symmetric MDPs

The automorphism group $\mathcal{G}_{\mathcal{M}} = \text{Aut}(\mathcal{M})$ of an MDP identifies the set of symmetry transformations of state-actions that preserve the reward and the transition dynamics:

$$R(s,a) = R(g \cdot \langle s,a \rangle)), \quad \forall g \in \mathcal{G}_{\mathcal{M}}, s \in \mathcal{S}, a \in \mathcal{A} \tag{2.8}$$

$$T(s' \mid s,a) = T(g \cdot s' \mid g \cdot \langle s,a \rangle) \text{ and}$$

$$g \cdot T(s,a) = T(g \cdot \langle s,a \rangle), \quad \forall g \in \mathcal{G}_{\mathcal{M}}, s, s' \in \mathcal{S}, a \in \mathcal{A} \tag{2.9}$$

We refer to a reward function R that satisfies Eq. (2.8) as a $\mathcal{G}_{\mathcal{M}}$-invariant reward function and a deterministic transition function T that satisfies Eq. (2.9) as a $\mathcal{G}_{\mathcal{M}}$-equivariant transition function. Note that this is a distinct notion from invariance and equivariance under the agent's action in the context of MDP homomorphism. Here, the action refers to the action of a symmetry group, while in MDP homomorphism, the equivariance is to the action of the agent. We use group action or $\mathcal{G}$-action to make this distinction clear when necessary.

The connection of symmetric MDPs to MDP homomorphism is due to the fact that symmetries can be used to define a homomorphism $\mathcal{H} : \mathcal{M} \mapsto \bar{\mathcal{M}}$ by collapsing the state-actions that form an orbit under $\mathcal{G}_{\mathcal{M}}$ that is:

$$(h_{\mathcal{S}}(s), h_{\mathcal{A}}(s,a)) = (h_{\mathcal{S}}(g.s), h_{\mathcal{A}}(g \cdot \langle s,a \rangle) \quad \forall g \in \mathcal{G}_{\mathcal{M}}\ s, a \in \mathcal{S} \times \mathcal{A} \tag{2.10}$$

Formally, the collapsed MDP $\bar{\mathcal{M}} = \langle \bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{R}, \bar{T} \rangle$ is defined by $\bar{\mathcal{S}} = \mathcal{S}/\mathcal{G}_{\mathcal{M}}, \bar{\mathcal{A}} = \mathcal{A}/\mathcal{G}_{\mathcal{M}}$, $\bar{R}(\langle s,a \rangle^{\mathcal{G}_{\mathcal{M}}}) = R(s,a)$ and $\bar{T}(s'^{\mathcal{G}_{\mathcal{M}}} \mid \langle s,a \rangle^{\mathcal{G}_{\mathcal{M}}}) = T(s' \mid s,a)$. This results in a symmetry-based model minimization of symmetric MDPs where the optimal action-value function and the lifted policy are invariant along the orbit of the group $\mathcal{G}_{\mathcal{M}}$.

Theorem 2.4.3. ***(Ravindran & Barto, 2001)*** *For a symmetric MDP that satisfies both Eq.* (2.8) *and Eq.* (2.9)*, both the optimal action-value and optimal policy functions become invariant under $\mathcal{G}_{\mathcal{M}}$ action – that is,*

$$\begin{aligned} &Q^*(s,a) = Q^*(g \cdot \langle s,a \rangle) \quad \textit{and} \\ &\pi^*(a,s) = \pi^*(g \cdot \langle a,s \rangle), \quad \forall g \in \mathcal{G}_{\mathcal{M}}\ s,a \in \mathcal{S} \times \mathcal{A}. \end{aligned} \tag{2.11}$$

The above result is easy to derive using the application of Eq. (2.10) in Theorem 2.4.1 and Theorem 2.4.2 and also implies $V^*(s) = V^*(g \cdot s)$. In the following section, we provide a brief overview of Deep Reinforcement Learning. The subsequent chapters will then demonstrate how the concepts introduced in this section can be applied to enhance the sample efficiency and generalization capabilities of Deep RL agents.

2.5 Deep Reinforcement Learning

Deep reinforcement learning (DRL) integrates Deep Neural Networks (DNNs) with reinforcement learning (RL) algorithms to enable learning directly from raw sensory inputs, such as images or audio or other signals. This integration facilitates the solution of complex decision-making problems where traditional RL techniques might struggle. DRL algorithms typically use DNNs to represent either the value function, the policy, or both. Moreover, an alternative approach involves using DNNs to construct a model of the environment. Although this model-based strategy can generate synthetic experiences for policy or value function training, our focus in this thesis will remain on improving model-free DRL approaches from a representation learning perspective.

Value-based RL algorithms function in two main phases: policy evaluation and policy improvement. For instance, *Q-learning*, a prominent method in this category, estimates the optimal action-value function, denoted by $Q^*(s,a)$. This function represents the maximum expected cumulative reward obtainable from taking action a in state s, and the policy is improved by setting it to $arg\max_a Q^*(s,a)$ that is choosing actions that maximize $Q^*(s,a)$ (Watkins & Dayan, 1992; Mnih et al., 2013b, 2015a).

On the other hand, *policy-based* RL algorithms optimize the policy directly to maximize the expected return. *Policy gradient* methods update the policy by adjusting the policy function's parameters, typically by estimating the gradient of the expected return, denoted as J (Sutton et al., 2000; Schulman et al., 2015).

Combining the strengths of both value-based and policy-based approaches, *Actor-critic* methods employ two networks: the Q-network (critic), which learns the action value under the current policy, and the policy network (actor), which seeks to maximize the expected return by following the critic's guidance (Konda & Tsitsiklis, 2000; Haarnoja et al., 2018a; Lillicrap et al., 2016; Schulman et al., 2017a). This approach allows for more stable and efficient learning, particularly in environments with high-dimensional action spaces.

Part II

Leveraging Known Symmetry Transformations

3

Group Equivariant Deep Reinforcement Learning

Reinforcement Learning has always faced challenges when handling high-dimensional sensory input, such as that given by vision or speech. To this end, it was demonstrated that a convolutional neural network could directly learn control policies from raw video data, with success in various Atari game environments (Mnih et al., 2013a). More recently, there has been work to improve both the feature extraction from raw images (Grattarola, 2017) as well as the underlying Deep Q-Learning algorithm (Schaul et al., 2015; Horgan et al., 2018; Van Hasselt et al., 2016; Wang et al., 2015). Despite these advances, the generalization of trained agents to new environments and the improvement of sample efficiency have not been widely explored, especially in the context of transformations of the environment or the agent.

In this chapter, which is based on (Mondal et al., 2020), we show how to exploit the intrinsic properties of an environment, such as its symmetry, to improve the generalization and data efficiency of Deep RL algorithms. In particular, we show that symmetric environments with discrete actions having an equivariant policy can lead to model minimization. We achieve this using an E(2)-Equivariant CNN (Weiler & Cesa, 2019b) architecture

as a function approximator for training RL agents using an Equivariant Q-Learning algorithm. We show that in a game environment, with a high degree of symmetry, such an approach provides a significant performance gain and improves sample efficiency as it learns from fewer experience samples.

We further show that the inherent inductive bias of our proposed approach enables the effective use of knowledge gained across previously unseen transformations of the environment. Our proposed method is complementary to the other generalization ideas in RL and hence can be used in conjunction with them. Using the proposed method improves generalization and facilitates a higher degree of parameter sharing.

3.1 Related Work

Group equivariant CNNs (G-CNN) (Cohen & Welling, 2016b) exploit the group of symmetries of input images to reduce sample complexity, learn faster and improve the capacity of CNNs without increasing the number of parameters. This network architecture uses a new convolution layer whose output feature map changes equivariantly with the group action on the input feature map and promotes higher degrees of weight sharing. The theory of steerable CNNs (Cohen & Welling, 2017; Weiler & Cesa, 2019b; Weiler et al., 2018) generalizes this idea to continuous groups and homogeneous spaces. In this work, we focus on using an E(2)-Equivariant Steerable CNN (Weiler & Cesa, 2019b) architecture for deep RL.

Given an input signal, CNNs extract a hierarchy of feature maps. The weight-sharing of the convolution layers makes them inherently translation-equivariant so that a translated input signal results in a corresponding translation of the feature maps (Cohen & Welling, 2016b). An E(2)-Equivariant Steerable CNN carries out translation, rotation and reflection equivariant convolution on the image plane. The feature spaces of such Equivariant CNNs are defined as spaces of feature fields and are characterized by a group representation that determines their transformation behaviour under transformations of the input, as discussed in Section 3.2.1.

The Deep Q-learning Network (DQN) (Mnih et al., 2013a) has been widely used in RL since its inception. The DQN utilizes “experience replay" (Lin, 1993) where the agent's experiences at each time-step are stored in a memory buffer, and the Q-learning updates are done on samples drawn from this buffer, which breaks the correlation between them. A variant of this strategy is the “prioritized replay buffer" (Schaul et al., 2015), where the

experiences are sampled according to their importance. The importance of an experience is typically defined based on its temporal-difference error, with experiences having higher errors being sampled more frequently as they are considered more informative for learning. A second variant, the Double DQN or DDQN (Van Hasselt et al., 2016), addresses the problem of maximization bias, which occurs due to the usage of the same Q network for the off-policy bootstrapped target. It uses two Q-networks and takes the min of their prediction to calculate the bootstrap target to update both of them. An additional improvement, proposed in a Dueling Network (Wang et al., 2015), is the use of an advantage function and the learning of a value function to estimate a baseline using a common convolutional feature learning module. The advantage function is the difference between the action-value function and the value function. We experiment with the above-mentioned variants [1].

3.2 Background

3.2.1 E(2)-equivariant convolution

In this section, we briefly describe the theory behind $E(2)$-equivariant convolution. First, we define the group $T(2) \rtimes \mathcal{G}$ where $\mathcal{G} \leq O(2)$. $T(2)$ is a translational group on $\mathbb{R}^2$ and $\mathcal{G}$ is a subgroup of the orthogonal group $O(2)$, which are continuous rotations and reflections under which the origin is invariant. Intuitively, we are dealing with the subgroups of the group of isometries of a 2-D plane called $E(2)$. In contrast to regular CNNs, which work with a stack of multiple channels of features $f : \mathbb{R}^2 \to \mathbb{R}$, the steerable CNN defines a steerable feature space of feature fields $f : \mathbb{R}^2 \to \mathbb{R}^c$ which associates a c dimensional feature vector $f(x) \in \mathbb{R}^c$ to every $x \in \mathbb{R}^2$. The feature fields are linked to a transformation law that defines their transformations under the action of a group. The transformation law of a feature field is characterized by the group representation $\rho : \mathcal{G} \mapsto \mathrm{GL}(\mathbb{R}^c)$, where $\mathrm{GL}(\mathbb{R}^c)$ represents the group of all invertible $c \times c$ matrices. This defines how each of these c channels mixes when the vector $f(x)$ is transformed. The operator for a transformation tg, where $t \in T(2)$ and $g \in \mathcal{G}$, is given by:

$$\left(\left[\mathrm{Ind}_{\mathcal{G}}^{T(2) \rtimes \mathcal{G}} \rho \right] (tg) \, . f \right) (x) := \rho(g) . f \left(g^{-1} (x - t) \right) \tag{3.1}$$

[1] This only covers papers before (Mondal et al., 2020) was published in 2019. We mention the follow-up work in the community in the discussion

where $[Ind_{\mathcal{G}}^{T(2)\rtimes\mathcal{G}}\rho]$ is called the induced representation as described in Section 2.2. Analogous to the channels of a regular CNN, we can stack multiple feature fields f_i with their corresponding representation ρ_i and the stack $\bigoplus_i f_i$ then transforms under $\bigoplus_i \rho_i$, which is a block diagonal matrix. Notice that due to ρ being a block diagonal matrix each feature field transforms independently. Having described the feature fields, we will next give the equation for equivariance and the constraint it imposes on the convolution kernel. Consider two feature fields $f_{in} : \mathbb{R}^2 \to \mathbb{R}^{C_{in}}$ with representation ρ_{in}, $f_{out} : \mathbb{R}^2 \to \mathbb{R}^{C_{out}}$ with representation ρ_{out} and a convolution kernel $k : \mathbb{R}^2 \to \mathbb{R}^{c_{out}\times c_{in}}$ then the desired equivariance is given by:

$$\begin{aligned} k * \left(\left[\mathrm{Ind}_{\mathcal{G}}^{T(2)\rtimes\mathcal{G}}\rho_{in}\right](tg) \,.f_{in} \right) = \\ \left[\mathrm{Ind}_{\mathcal{G}}^{T(2)\rtimes\mathcal{G}}\rho_{out}\right](tg) \,.(k * f_{out}) \end{aligned} \tag{3.2}$$

where convolution is defined as usual as:

$$f_{out}(x) := (k * f_{in})(x) = \int_{\mathbb{R}^2} k(y) f_{in}(x+y) dy \tag{3.3}$$

This can only be achieved if we restrict ourselves to G-steerable kernels which satisfy the kernel constraint:

$$k(gx) := \rho_{out}(g) k(x) \rho_{in}(g^{-1}) \quad \forall g \in \mathcal{G} \quad \& \quad x \in \mathbb{R}^2 \tag{3.4}$$

(Weiler & Cesa, 2019b) provide a comprehensive derivation of the basis of the kernel that satisfies Eq. (3.4). Imposing this constraint on the kernels significantly reduces the number of parameters and promotes parameter sharing. Also, by obtaining equivariance in each convolution layer of the network, they can be composed to extract equivariant features from the input 2D image signal. Further details on the kernel basis are provided in (Weiler & Cesa, 2019c).

3.2.2 Choice of group and feature fields

We now discuss how one would choose a group (g) and its representation (ρ) to define a feature field. The group's choice mainly depends on the problem we are tackling and to which kinds of transformation we wish the network to output equivariantly. We have several options for $E(2)$-equivariant convolution, starting from discrete rotations and reflections (D_N) to continuous rotation and reflection ($O(2)$). Once a group is chosen, we need to choose its

representation. The most common ones are trivial, irreducible, and regular representations. The representation chosen determines the dimension c of a feature vector. While a trivial representation implies scalar features with dimension 1 the regular representation uses an N-dimensional feature field, where N denotes the order of the group we are using. Even though a regular representation was shown to perform the best (Weiler & Cesa, 2019b), it is computationally infeasible to use it when using higher-order groups. In such a case, we use an irreducible representation, which takes the smallest dimension while leaving the representation of all the group elements unique.

Let us assume that we are working with a generic D_N group with its regular representation. The next thing we need to choose is the number of feature fields for each intermediate layer. Together the chosen representation and number of feature fields contribute to the dimension of the stack $\bigoplus_i f_i$ of intermediate feature fields, which further determines the depth of the convolution kernel we are using between two feature fields. Although increasing the number of feature fields increases the network's capacity, this comes at the cost of increased computation during a single forward pass.

In an environment where we have a global D_N symmetry, where we want D_N equivariant features and $N > 1$, we can directly choose the group and keep it throughout. But in most environments where there is usually a global D_1 symmetry and occasionally local D_N symmetry, using the same representation throughout would be unnecessary as this is accompanied by an order of N increase in feature field dimension. To alleviate this problem, we start with a higher-order group D_N where $N > 1$ and as we go deeper into our Q network, we restrict it to its subgroups ($\leq D_N$). This makes the network more computationally efficient while still extracting an equivariant feature vector.

3.3 Equivariance in RL

3.3.1 Equivariance in Vectorized Policies

In this section, we demonstrate that for symmetric Markov Decision Processes (MDPs), the policy network or Q network must be equivariant to the symmetry group to achieve the invariances presented in Eq. (2.11), thereby implicitly obtaining model minimization.

For an MDP symmetric with respect to the group $\mathcal{G}_{\mathcal{M}}$, a $\mathcal{G}_{\mathcal{M}}$-equivariant

policy network $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^{\geq 0}$ must satisfy:

$$\pi(a|g \cdot s) = \pi(g^{-1} \cdot a|s) \quad \forall g \in \mathcal{G}_{\mathcal{M}},\ s \in \mathcal{S},\ a \in \mathcal{A}. \tag{3.5}$$

Under this condition, optimal policy functions become invariant under $\mathcal{G}_{\mathcal{M}}$ action, i.e., $\pi^*(a, s) = \pi^*(g \cdot (a, s)) \quad \forall g \in \mathcal{G}_{\mathcal{M}},\ s \in \mathcal{S},\ a \in \mathcal{A}$.

For finite actions and a vectorized policy network

$$\pi(s) = [\pi(a_1|s), \pi(a_2|s), \ldots, \pi(a_n|s)]$$

, equation Eq. (3.5) simplifies to $\pi(g \cdot s) = g \cdot \pi(s)$, where the action of g on $\pi(s)$ is a permutation of the n-dimensional vectorized policy. For other types of policies, such as those with continuous action spaces, the corresponding equivariance constraints can be derived. Existing work on building equivariant networks for both discrete and continuous groups can be utilized. Additionally, different actions of the same group on the state and action spaces can be formalized for each specific problem.

Similarly, for finite actions and Q-learning based methods, assuming the policy is $\texttt{argmax}_a Q(s, a)$ and $Q(s) = [Q(s, a_1), Q(s, a_2), \ldots, Q(s, a_n)]$, an equivariance constraint $Q(g \cdot s) = g \cdot Q(s)$ is required for the optimal action-value function to be invariant under $\mathcal{G}_{\mathcal{M}}$ action, i.e.,

$$Q^*(s, a) = Q^*(g \cdot (s, a)) \quad \forall g \in \mathcal{G}_{\mathcal{M}},\ s \in \mathcal{S},\ a \in \mathcal{A}.$$

This indicates that designing equivariant models for Q and policy networks using the symmetries of the state-action space constrains the solution space. This results in better sample efficiency in deep reinforcement learning algorithms and provides robustness guarantees to the policy under distribution shifts in the state space resulting from symmetry transformations, such as those related to global rotations of the environment with respect to the agent.

Next, we explore how to identify these symmetries in an environment and construct equivariant Q networks to learn an equivariant policy.

3.3.2 Choice of the Environment

In this chapter, we primarily experiment with two environments - the Snake game of the Pygame Learning Environment (Tasfi, 2016) and the Atari Pacman environment (Brockman et al., 2016) [2]. In the Snake game[3], the

[2] https://gym.openai.com/envs/MsPacman-v0/

[3] https://pygame-learning-environment.readthedocs.io/-en/latest/user/games/snake.html

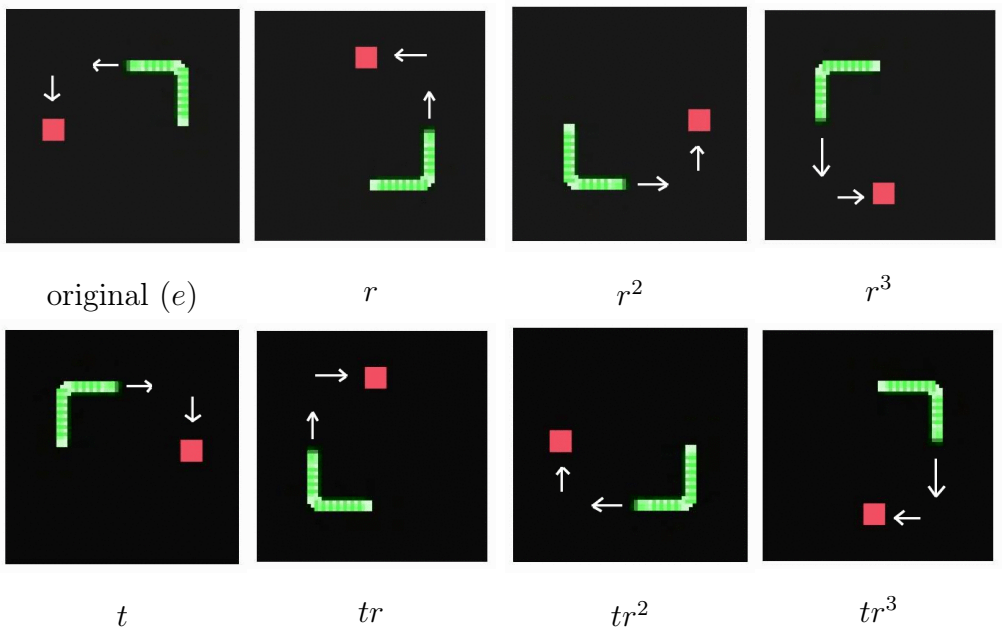


Figure 3.1: If we denote the 90 degree clockwise rotation by r and reflection over the vertical axis as t, then the group elements of D_4 are $\{e, r, r^2, r^3, t, tr, tr^2, tr^3\}$ where e is the identity action. These panels show the action of these group elements (transformations) on a game screen and how they affect the optimal policy (shown by white arrows).

agent is a snake which grows in length each time it feeds on a food particle and gets a reward of $+1$. The food particle is randomly placed somewhere inside the valid area of a screen. The snake can choose four legal actions: move up, move down, move left, and move right. A terminal state is reached when the snake comes in contact with its body or the walls, and the agent then receives a score of -1. From Figure 3.1, we see that under the action of group elements of D_4, the current optimal policy should change equivariantly, which suggests the possible benefits of learning the Q values for each action using equivariant features extracted from the game screen.

The Pacman game consists of a maze, a player agent and a few ghosts. Food particles are placed along the paths of the maze while the ghosts move freely around it. The player agent is also allowed four actions - move up, move down, move right and move left and it gets a positive reward for each particle it consumes without running into any of the ghosts. The game screen has a global D_1 symmetry and a degree of local D_4 symmetry.

3.3.3 Equivariant Deep Q-Network Design

Henceforth in this chapter, "equivariant convolution" refers to $E(2)$-equivariant steerable convolution. Suppose our preprocessed input is of dimension $m \times d \times d$ where m is the number of channels and $d \times d$ is the size of the image. We convert it into a feature field represented by $s = \bigoplus_{i \in I} s_i$ where $I = \{1, .., m\}$ and s_i is an image of dimension $d \times d$. The transformation law of each channel is given by trivial representation (ρ_{triv}) of a chosen discrete group ($\mathcal{G}$) for each channel. We further choose a regular representation (ρ_{reg}) for intermediate feature fields, which are permutation matrices given a group element $g \in \mathcal{G}$, to derive the kernel basis of equivariant convolution. Using regular representation preserves the equivariance with point-wise nonlinear activation functions such as ReLU. We stack equivariant convolutions followed by ReLU to obtain an equivariant feature extractor $F_{eqv} : \mathbb{R}^{m \times d \times d} \to \mathbb{R}^n$ where n denotes the dimension of extracted equivariant features. The detailed architecture of this feature extractor and its relationship to the vanilla feature extractor that is used in DDQN are in Section 3.3.4. Assuming that we do not restrict the group $\mathcal{G}$ along the depth of the network, our transformation rule of the extracted feature vector with respect to the transformation of input is given by:

$$F_{eqv}\left(\left[\mathrm{Ind}_{\mathcal{G}}^{T(2)\rtimes\mathcal{G}}\rho_{triv}\right](tg)\, s; \theta\right) = \rho(g)\left(F_{eqv}\left(s; \theta\right)\right) \tag{3.6}$$

where $\rho(g) = \bigoplus_{j \in J} \rho_{reg}(g)$ and

$$\left(\left[\mathrm{Ind}_{\mathcal{G}}^{T(2)\rtimes\mathcal{G}}\rho_{triv}\right](tg)\, s\right)(x) := \bigoplus_{i \in I}\left[s_i\left(g^{-1}\left(x - t\right)\right)\right]$$

Note that Equation 3.6 gives the desired equivariance and $J = \{1, .., (n/N)\}$ where N is the order of the g. N divides n and n/N is the number of feature fields at the output. Intuitively, Equation 3.6 means that at every feature field, the values are permuted along its dimension when we transform the input by some group element. Also note that if we restrict the group along the depth we will have $g_{res} \in \mathcal{G}_{res} \leq \mathcal{G}$ in the RHS of Equation 3.6 instead of g. Having obtained the feature vector which transforms equivariantly we can add a final linear layer to obtain the Q values:

$$Q_{eqv}(s, a, \phi) = W_a.F_{eqv}(s; \theta) + b_a \tag{3.7}$$

where $a \in \mathcal{A} = \{1, .., |\mathcal{A}|\}$, $W_a = [w_{a1}..w_{an}]$ and $\phi = \{\theta, W_1, .., W_{|\mathcal{A}|}, b_1, .., b_{|\mathcal{A}|}\}$ (the set of all parameters). This choice of model architecture strikes a balance

between equivariance constraints and flexibility. While the intermediate representations maintain equivariance, the inclusion of a standard linear layer in the final stage provides the model with adaptability across diverse datasets. This design offers dual benefits: in symmetric environments, the equivariant structure provides a valuable inductive bias, accelerating learning. Conversely, in non-symmetric scenarios, the model's performance remains uncompromised, ensuring its versatility across a wide range of applications. This approach effectively leverages symmetry when present without imposing restrictions on the model's general applicability. We use DDQN as our baseline model throughout this work whose final loss at iteration l is given by:

$$L(\phi_l) = \mathbb{E}_{s,a,r,s'}\left[\left(y_l^{DDQN} - Q_{eqv}(s,a,\phi_l)\right)^2\right] \tag{3.8}$$

with target:

$$y_l^{DDQN} = r + \gamma Q_{eqv}(s', \arg\max_{a'} Q_{eqv}(s', a', \phi_l), \phi^-) \tag{3.9}$$

where ϕ^- represents the parameters of the frozen network. The gradients computed through both the linear and the equivariant feature extractor networks are backpropagated to update their parameters.

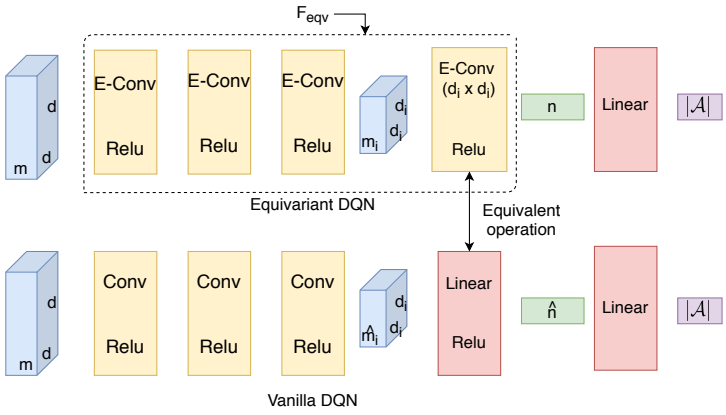


Figure 3.2: Juxtaposed Network architecture

3.3.4 Network Architecture

The baseline Vanilla DDQN used in this work is similar to the one used in (Mnih et al., 2013a), which has an output dimension equal to the number of

actions. As shown in Fig. 3.2, our proposed Equivariant DDQN architecture mainly replaces the Vanilla convolutions($Conv$) and the second last linear layer with equivariant convolutions($E\text{-}Conv$). We call this an equivariant feature extractor. We want to emphasize the last $E\text{-}Conv$ layer and point-out that its operation is similar to the second last linear layer in a Vanilla DDQN. As we use the filter size of the dimension of feature size before that layer, all the information is captured as a weighted sum into a 1-D vector. Although this is the same as the flattening of the feature and then applying a linear layer, using $E - Conv$ renders the output vector equivariant.

The final linear layer is the same for both and maps them to Q-values for each action. Notice, as mentioned in Section 3.2.2, the group representation and the number of feature fields will determine the sizes of intermediate features. We aim to make both networks similar with respect to computation time while not comprising the capacity of the Equivariant model. Below we provide the architecture of the Equivariant and Vanilla DDQN for both Snake and Pacman. We denote a basic convolution by: $Conv(filtersize \times filtersize, inchannels, outchannels, stride, padding)$ and equivariant one by: $E{-}Conv(filtersize{\times}filtersize, infields, outfields, stride, padding)[Group]$ The group restriction operation is denoted by: $GrpRes[Group \to Subgroup]$. In the Vanilla and Equivariant DDQN, we denote the size of the output of the third convolution by $\hat{m}_i \times d_i \times d_i$ and $m_i \times d_i \times d_i$ respectively. Using this, we give the exact architecture of both networks below.

Snake

Vanilla DDQN	**Equivariant DDQN**
$Conv(7 \times 7, m, 32, 2, 2) - ReLU$	$E\text{-}Conv(7 \times 7, m, 8, 2, 2)[D_4] - ReLU$
$Conv(5 \times 5, 32, 64, 2, 1) - ReLU$	$E\text{-}Conv(5 \times 5, 8, 12, 2, 1)[D_4] - ReLU$
$Conv(5 \times 5, 64, 64, 1, 1) - ReLU$	$E\text{-}Conv(5 \times 5, 12, 12, 1, 1)[D_4] - ReLU$
$Linear(\hat{m}_i \times d_i \times d_i, 256) - ReLU$	$E\text{-}Conv(d_i \times d_i, 12, 32, 1, 0)[D_4] - ReLU$
$Linear(256, \lvert\mathcal{A}\rvert)$	$Linear(256, \lvert\mathcal{A}\rvert)$

Although there is a difference in the number of channels and feature fields, the overall runtime of the DDQN algorithms with both networks is similar. The forward pass of the Equivariant network is more computationally expensive as the total dimension of the stack of feature fields in some layers is more than the number of channels in the Vanilla network. But this is

Pacman

Vanilla DDQN

$Conv(7 \times 7, m, 32, 4, 2) - ReLU$
$Conv(5 \times 5, 32, 64, 2, 2) - ReLU$
$Conv(5 \times 5, 64, 64, 2, 1) - ReLU$
$Linear(\hat{m}_i \times d_i \times d_i, 512) - ReLU$
$Linear(512, |\mathcal{A}|)$

Equivariant DDQN

$E\text{-}Conv(7 \times 7, m, 8, 2, 2)[D_4] - ReLU$
$E\text{-}Conv(5 \times 5, 8, 16, 2, 1)[D_4] - ReLU$
$GrpRes[D_4 \to D_1]$
$E\text{-}Conv(5 \times 5, 16, 64, 1, 1)[D_1] - ReLU$
$E\text{-}Conv(d_i \times d_i, 64, 384, 1, 0)[D_1] - ReLU$
$Linear(768, |\mathcal{A}|)$

partially compensated for during the backpropagation where we are updating fewer parameters in an Equivariant network. Note that, in general, adding feature fields increases the capacity at the cost of computation, but we keep the total cost with respect to the Vanilla model in mind while choosing them. Also, as the Pacman environment is globally symmetric to the D_1 group, we restrict the group once D_4 symmetric lower-level features are extracted, which also reduces the dimension of representation and hence the computation cost significantly.

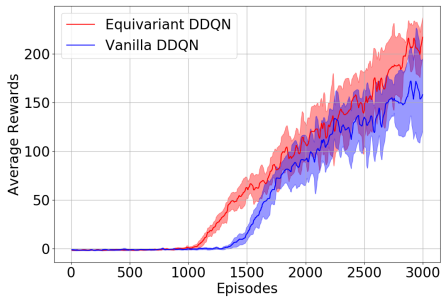
Network Type	Vanilla DQN	Number of Parameters
Vanilla DDQN	Snake	583.46k
Equivariant DDQN	Snake	57.7k
Vanilla DDQN	Pacman	984.36k
Equivariant DDQN	Pacman	649.93k

Table 3.1: This table gives the number of parameters of both the Networks.

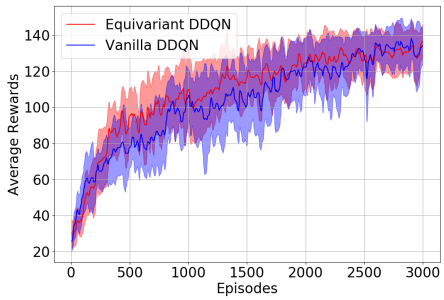
3.4 Experiments

We first consider the performance of a carefully designed equivariant DDQN, keeping in mind the symmetry of the game, compared to a vanilla DDQN. For a fair comparison, we keep the settings of the environment and hyperparameters the same for all the experiments[4]. We report in Figure 3.3 the evolution

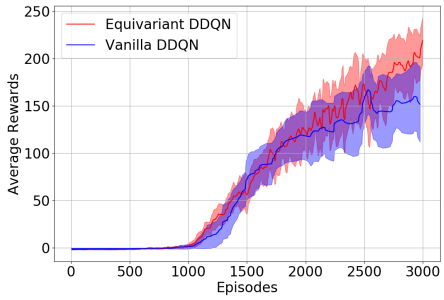
[4]Link to the code: https://github.com/arnab39/EquivariantDQN



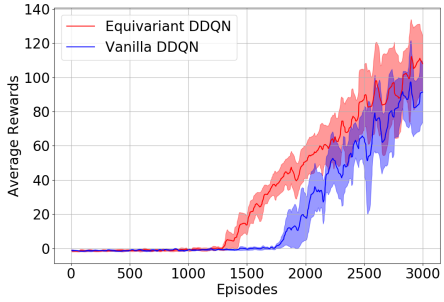
DDQN in Snake



DDQN in Pacman



DDQN with priority replay in Snake



Dueling DQN in Snake

Figure 3.3: Plots of the evolution of average rewards with the number of episodes using different methods in Snake and Pacman. We show the confidence intervals over 10 different seeds. The plots are smoothed with a 1D Gaussian filter with σ=3 for improved visualization.

of rewards collected over the training episodes for both the models in the Snake and the Pacman environments. Our proposed model attains a 30% improvement in average reward collected after training for 3000 episodes in the highly symmetric Snake environment. Interestingly this improvement in data-efficiency comes even with a 90% reduction in the number of parameters. This verifies our hypothesis that the parameters required to learn policies of the identity transformation would be sufficient to generalize to optimal policies in other transformations for the Snake environment. In the case of Pacman, we notice our model performs slightly better in the initial episodes, with a 34% reduction in the number of parameters. But once both the models have

seen enough samples, the margin of difference vanishes. In Fig. 3.3, we also show that the proposed method gives similar results with other subsequent improvements, such as using DDQN with priority replay and the Dueling architecture.

We further investigate the usefulness of the inherent inductive bias in the model to out-of-distribution generalization with respect to the transformation of the environment screen. For this part, we remove the group restriction from the Equivariant DDQN of the Pacman game and make the feature extractor D_4 equivariant. First, we train both the Vanilla and Equivariant models. We then change the environment by rotating the input screen by 90 degrees clockwise (r). Leaving the rest of the network frozen, we retrain the final linear layer for this new environment. We show in Table 3.2 that while a regular CNN-based feature extractor fails, the Equivariant feature extractor can still find a decent policy after learning the linear layers for certain epochs. The results of a simple path planning problem like Snake, indicate that our model can, in principle, be extended to more complex continuous path planning problems such as in UAVs (Zhang et al., 2015; Challita et al., 2018). Such scenarios would benefit both from faster learning due to increased sample efficiency and viewpoint transformation equivariant features for optimal policy learning, which can generalize to new transformations of the environment.

Transformation	Vanilla DDQN	Equivariant DDQN
e	129 ± 2.3	125 ± 4.5
r	48.9 ± 2	99 ± 4.7
r^2	53 ± 3.5	104 ± 3.4
r^3	51 ± 1.9	98 ± 3.9

Table 3.2: Average reward over 200 episodes of Pacman for 5 seeds reported with a confidence level of 95% for different environment transformations. e is the original screen.

3.5 Discussion

We have introduced an Equivariant Deep-Q learning algorithm, and have demonstrated that it provides a considerable boost to performance with parameter and sample efficiency when carefully designed for highly symmetric environments. We have also shown that this approach generalizes policies well to new unseen environments obtained by an affine transformation of the original environment. Although equivariant models in supervised learning

were shown to make the models robust when introduced in (Mondal et al., 2020), this was the first time has been proposed in a Deep RL framework.

Limitations The methods presented in the current chapter are limited to finite discrete action spaces and the discrete group of transformations of state action. In theory, we can leverage more from our equivariance formulation in continuous groups as the reduction in the solution space by constraining the policy is directly proportional to the size of the group of transformation.

Another significant limitation of this method is the need to know the symmetry group and its action in the state and action space a priori, in order to be able to learn the equivariant representation. We tackle this hard problem in the second part of this thesis and design methods to learn equivariant representations directly from the observed data. Moreover, designing a novel equivariant network architecture for different applications is challenging for Deep RL practitioners without the knowledge of relevant techniques from the Equivariant Deep Learning literature. Additionally, adapting this idea to existing techniques for real-world applications is challenging, especially because it requires careful redesign of the architecture and training them from scratch. To address these limitations, in the next two chapters, we motivate and provide novel techniques to build equivariance into the overall model, without constraining every layer in the main architecture of the model.

Follow-up work in the community Many follow-up works on the concepts of symmetry and equivariance have gained significant attention in the fields of deep reinforcement learning (RL) and robotics. As discussed in this chapter, these papers offer promising avenues for improving sample efficiency, generalization, and overall performance of Deep RL algorithms.

Symmetry in RL has been explored in various contexts. (van der Pol et al., 2020b) proposed MDP homomorphic networks for exploiting symmetries in RL algorithms like PPO and Actor-Critic, showing enhanced generalization across different environments. (Wang et al.) introduced SO(2) equivariant actor-critic methods, demonstrating improved sample efficiency in continuous control tasks. (Nguyen et al., 2023) extends these ideas to partially observable environments.

Expanding beyond traditional RL domains, in the realm of AI for Science, Simm et al. (2020) developed SO(3)-equivariant networks for learning generalizable policies using the Actor-Critic method for molecular design. The application of these principles extends to multi-agent reinforcement learning as well. van der Pol et al. (2021) proposed symmetric multi-agent reinforcement

learning, leveraging permutation invariance to improve coordination among agents.

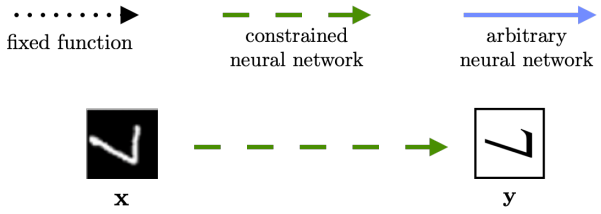
These advancements highlight the potential of symmetry and equivariance in improving the efficiency and generalization capabilities of RL algorithms in both simulated environments and real-world robotic applications. As the field progresses, we anticipate further integration of these principles into more complex and diverse RL scenarios.

4

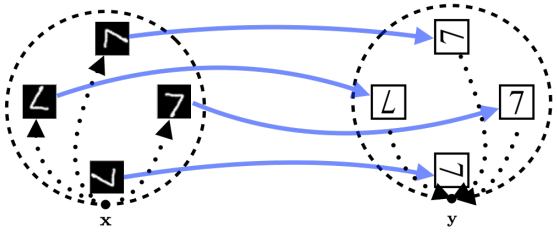
Equivariance Through Canonicalization

In the previous chapter, we have seen that the design of machine learning models that effectively harness the structure and symmetry of data is crucial. In many scenarios, the transformations required for models to maintain invariance or equivariance are already identified, providing a strong inductive bias grounded in similar principles observed in human cognition (*e.g.*, Bronstein et al., 2021b; Bogatskiy et al., 2022; van der Pol et al., 2020b; Mondal et al., 2020; Celledoni et al., 2021). This is evident in visual shape recognition (Shepard & Metzler, 1971; Carpenter & Eisenberg, 1978), where humans can effortlessly differentiate between object orientations and actual structural changes, a capability mirrored in various deep learning approaches.

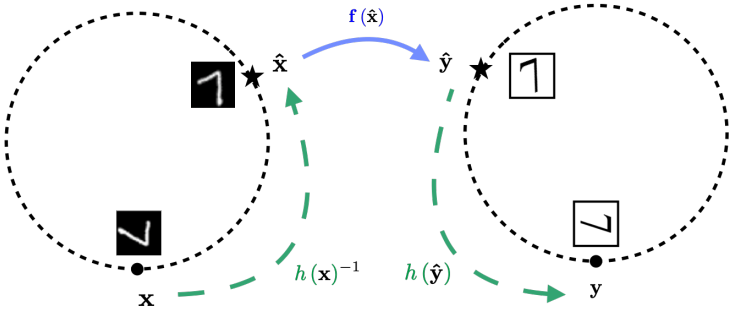
These approaches range from *viewpoint-independent* models that impose equivariance through constraints in the architecture (Shawe-Taylor, 1989; Cohen & Welling, 2016a; Ravanbakhsh et al., 2017) or use invariants as inputs (Villar et al., 2021), to *multiple-view* models that average over different orientations (Manay et al., 2006; Benton et al., 2020; Yarotsky, 2022; Puny et al., 2022), to *single-view-plus-transformation*-based models that transform the objects to a canonical orientation. (see Fig. 4.1) It is particularly noteworthy



(a) Viewpoint-independent



(b) Multiple-view



(c) Single-view-plus-transformation

Figure 4.1: A classification of different frameworks for equivariant predictions. In this example, the task is to restyle an MNIST digit in a rotation equivariant way. We propose a class of models that falls in the single-view-plus-transformation framework.

that *single-view-plus-transformation*-based models have been underexplored in machine learning, despite cognitive science evidence suggesting their use in human cognition (Shepard & Metzler, 1971; Carpenter & Eisenberg, 1978; Hinton & Parsons, 1981). When humans encounter a rotated pattern, the time it takes to recognize the pattern correlates with the rotation angle, indicative of a "mental rotation" process.

This understanding leads us to this chapter, where we introduce a robust method for equivariant machine learning that employs mappings to canonical

samples, seamlessly fitting into existing architectures to ensure adaptability to a wide range of transformations, both discrete and continuous. We propose learning this canonical orientation using canonicalization functions, rather than manually designing them. This yields better performance, merging the expressivity and practicality of frame averaging techniques with the efficiency of end-to-end learning processes (Puny et al., 2022).

Our contributions underscore the versatility and efficacy of this approach: we provide a comprehensive framework for achieving equivariance across diverse groups, proven to be universal approximators of equivariant functions in certain settings. Experimentally, our method has demonstrated significant improvements in processing images, physical dynamics, and point clouds, validating the hypothesis that a learned canonicalization approach surpasses traditional design methods in both theory and practice.

4.1 Canonicalization Functions

Recall from Section 2.3 that we are interested in learning functions $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$ with inputs $\mathbf{x} \in \mathcal{X}$ and outputs $\mathbf{y} \in \mathcal{Y}$ belonging to finite-dimensional normed vector spaces.

Now, a function $\boldsymbol{f}$ is $\mathcal{G}$-*equivariant* if

$$\boldsymbol{f}\left(\rho\left(g\right)\mathbf{x}\right) = \rho'\left(g\right)\boldsymbol{f}\left(\mathbf{x}\right),\ \forall\ g, \mathbf{x} \in \mathcal{G} \times \mathcal{X}, \tag{4.1}$$

where the *group action* ρ on the input and the group action ρ' on the output will be clear from the context.

4.1.1 General Formulation

The invariance requirement on a function $\boldsymbol{f}$ amounts to having all the members of a group orbit mapped to the same image by $\boldsymbol{f}$. It is thus possible to achieve invariance by appropriately mapping all elements to a canonical orbit representative before applying any function. For equivariance, elements can be mapped to a canonical sample and, after a function is applied, transformed back according to their original position in the orbit. This can be formalized by writing the equivariant function $\boldsymbol{f}$ in *canonicalized form* as

$$\boldsymbol{f}\left(\mathbf{x}\right) = h'\left(\mathbf{x}\right)\mathbf{p}\left(h\left(\mathbf{x}\right)^{-1}\mathbf{x}\right), \tag{4.2}$$

where the function $\mathbf{p} : \mathcal{X} \to \mathcal{Y}$ is called the *prediction function* and the function $h : \mathcal{X} \to \rho\left(\mathcal{G}\right)$ is called the *canonicalization function*. Here $h\left(\mathbf{x}\right)^{-1}$

is the inverse of the representation matrix and $h'(\mathbf{x}) = \rho'(\rho^{-1}(h(\mathbf{x})))$ is the counterpart of $h(\mathbf{x})$ on the output.

Equivariance in Eq. (4.2) is obtained for any prediction function if the canonicalization function is itself $\mathcal{G}$-equivariant, $h(\rho(g)\mathbf{x}) = \rho(g)h(\mathbf{x})\ \forall\, g, x \in \mathcal{G} \times \mathcal{X}$. [1]

It may seem like the problem of obtaining an equivariant function has merely been transferred in this formulation. This is, however, not the case: in Eq. (4.2), the equivariance and prediction components are effectively decoupled. The canonicalization function h can therefore be chosen as a simple and inexpressive equivariant function, while the heavy lifting of representation learning is done by the prediction function $\mathbf{p}$.

4.1.2 Partial Canonicalization

A more general condition can be formulated, such that the decoupling is partial. This enables us to impose part of the symmetry constraint on the prediction network and use canonicalization for "additional" symmetries. This could, for example, be used to imbue a translation equivariant architecture, like a CNN, with rotation equivariance.

Theorem 4.1.1. *For some subgroup $\mathcal{K} \leq \mathcal{G}$, if $\forall\ g, \mathbf{x} \in \mathcal{G} \times \mathcal{X}$ there exists a $k \in \mathcal{K}$ such that*

$$h(\rho(g)\mathbf{x}) = \rho(g)h(\mathbf{x})\rho(k), \tag{4.3}$$

and the prediction function $\mathbf{p}$ is $\mathcal{K}$-equivariant, then $\boldsymbol{f}$ defined in Eq. (4.2) is $\mathcal{G}$-equivariant.

Proof. We have

$$\boldsymbol{f}(\rho(g)\mathbf{x}) = h'(\rho(g)\mathbf{x})\,\mathbf{p}\left(h(\rho(g)\mathbf{x})^{-1}\rho(g)\mathbf{x}\right) \tag{4.4}$$

If equation 4.3 is satisfied, then $\forall\ g, \mathbf{x} \in \mathcal{G} \times \mathcal{X}$ there is a $k \in \mathcal{K}$ such that

$$\boldsymbol{f}(\rho(g)\mathbf{x}) = \rho'(g)h'(\mathbf{x})\rho'(k)\,\mathbf{p}\left(\left[\rho(g)h(\mathbf{x})\rho(k)^{-1}\right]^{-1}\rho(g)\mathbf{x}\right) \tag{4.5}$$

$$\boldsymbol{f}(\rho(g)\mathbf{x}) = \rho'(g)h'(\mathbf{x})\rho'(k)\,\mathbf{p}\left(\rho(k)^{-1}h(\mathbf{x})^{-1}\rho(g)^{-1}\rho(g)\mathbf{x}\right) \tag{4.6}$$

[1] Symmetric inputs in $\mathcal{X}$ pose a problem if we use the standard definition of equivariance for the canonicalization function. We explain this in (Kaba et al., 2023) and introduce the concept of relaxed equivariance that solves this issue.

Using the $\mathcal{K}$-equivariance of $\mathbf{p}$, we obtain

$$\boldsymbol{f}\left(\rho\left(g\right)\mathbf{x}\right)=\rho'\left(g\right)h'\left(\mathbf{x}\right)\rho'\left(k\right)\rho'\left(k\right)^{-1}\mathbf{p}\left(h\left(\mathbf{x}\right)^{-1}\mathbf{x}\right) \tag{4.7}$$

$$\boldsymbol{f}\left(\rho\left(g\right)\mathbf{x}\right)=\rho'\left(g\right)h'\left(\mathbf{x}\right)\mathbf{p}\left(h\left(\mathbf{x}\right)^{-1}\mathbf{x}\right) \tag{4.8}$$

□

This is equivalent to saying that the canonicalization function should output a coset in $\mathcal{G}/\mathcal{K}$ in an equivariant way, the applied transformation being chosen arbitrarily within the coset.

This can be simplified when the group factors into a semi-direct product using the following result.

Theorem 4.1.2. *If $\mathcal{K}$ is a normal subgroup such that $\mathcal{G}\simeq\mathcal{J}\ltimes\mathcal{K}$, condition Eq. (4.3) can be realized with a canonicalization function with image $\rho\left(J\right)$, and that is $\mathcal{J}$-equivariant and $\mathcal{K}$-invariant.*

Proof. We consider the special case where $\mathcal{K}$ is a normal subgroup of $\mathcal{G}$ such that the group can be taken to be isomorphic to a semidirect product $\mathcal{G}\simeq\mathcal{K}\rtimes\mathcal{J}$. Then, group elements can be written as $g=\left(k,j\right)$, where $k\in\mathcal{K}$ and $j\in\mathcal{J}$. The product is defined as $g_1g_2=\left(k_1,j_1\right)\left(k_2,j_2\right)=\left(k_1\varphi\left[j_1\right]\left(k_2\right),j_1j_2\right)$, where $\varphi:\mathcal{J}\to\mathrm{Aut}\left(\mathcal{K}\right)$ is a group homomorphism. Setting $k_2=e$ and $j_1=e$, we get any group element as $\left(k_1,e\right)\left(e,j_2\right)=\left(k_1,j_2\right)$.

If the canonicalization function is $\mathcal{J}$-equivariant and $\mathcal{K}$-invariant, we have

$$h\left(\rho\left(k,j\right)\mathbf{x}\right)=h\left(\rho\left(k,e\right)\rho\left(e,j\right)\mathbf{x}\right) \tag{4.9}$$

$$h\left(\rho\left(k,j\right)\mathbf{x}\right)=\rho\left(e,j\right)h\left(\mathbf{x}\right) \tag{4.10}$$

We then show that there is a $k'\in\mathcal{K}$ such that equation 4.3 is satisfied. Multiplying by $\rho\left(e\right)=\rho\left(k,e\right)\rho\left(e,j\right)h\left(\mathbf{x}\right)h\left(\mathbf{x}\right)^{-1}\rho\left(e,j\right)^{-1}\rho\left(k,e\right)^{-1}$ on the left, we have

$$\rho\left(e,j\right)h\left(\mathbf{x}\right)=\rho\left(k,e\right)\rho\left(e,j\right)h\left(\mathbf{x}\right)h\left(\mathbf{x}\right)^{-1}\rho\left(e,j\right)^{-1}\rho\left(k,e\right)^{-1}\rho\left(e,j\right)h\left(\mathbf{x}\right) \tag{4.11}$$

Using the fact that conjugation of an element of $\mathcal{K}$ by an element of $\mathcal{G}$ preserves $\mathcal{K}$ membership, we define $\rho\left(k',e\right)=h\left(\mathbf{x}\right)^{-1}\rho\left(e,j\right)^{-1}\rho\left(k,e\right)^{-1}\rho\left(e,j\right)h\left(\mathbf{x}\right)$

$$\rho\left(e,j\right)h\left(\mathbf{x}\right)=\rho\left(k,e\right)\rho\left(e,j\right)h\left(\mathbf{x}\right)\rho\left(k',e\right) \tag{4.12}$$

which shows that equation 4.3 is satisfied.

Finally, we show that in this case, the image of h can be chosen to be $\rho(\mathcal{J})$. We first remark that in each orbit $\mathcal{X}/\mathcal{G}$ of the group action, the canonical sample $\hat{\mathbf{x}}$ can be obtained from any orbit member $\mathbf{x}$, as $\hat{\mathbf{x}} = h(\mathbf{x})^{-1}\mathbf{x}$. For the canonical sample, we must have a $k \in \mathcal{K}$ such that

$$h\left(h(\mathbf{x})^{-1}\mathbf{x}\right) = h(\mathbf{x})^{-1} h(\mathbf{x}) \rho(k, e) \tag{4.13}$$

If we impose $k = e$ to satisfy this condition, we have $h(\hat{\mathbf{x}}) = \rho(e, e)$.

Since any orbit member can conversely be written as $\mathbf{x} = \rho(k, j)\hat{\mathbf{x}}$ for some $k \in \mathcal{K}$ and $j \in \mathcal{J}$, if the canonicalization function is $\mathcal{J}$-equivariant and $\mathcal{K}$-invariant, we have

$$h(\mathbf{x}) = h(\rho(k, j)\hat{\mathbf{x}}) = \rho(e, j) h(\hat{\mathbf{x}}) = \rho(e, j) \tag{4.14}$$

which completes the proof. □

Going back to the example of using rotation canonicalization with a CNN, Theorem 4.1.1 says that the canonicalization function should output an element of the Euclidean group transforming equivariantly under rotations of the input. Since the translation subgroup is normal, Theorem 4.1.2 can be used to guarantee that the canonicalization network can always simply output a rotation.

In general, when $\mathcal{K} = \{e\}$, only the canonicalization function is constrained, which is the case described at the beginning of the section. In the image domain, this would correspond to canonicalizing with respect to the full Euclidean group and using an MLP as a prediction function. The other extreme, given by $\mathcal{K} = \mathcal{G}$, corresponds to arbitrarily transforming the input and constraining the prediction function as is usually done in equivariant architectures like Group Equivariant Convolutional Neural Networks (G-CNNs) (Cohen & Welling, 2016b). These are, respectively, the *single-view-plus-transformation* and the *viewpoint-independent* implementations described in the introduction. Subgroups $\{e\} < \mathcal{K} < \mathcal{G}$ offer intermediary options; the lattice of subgroups of $\mathcal{G}$, therefore, defines a family of models. Since equivariance to a smaller group is less constraining for the prediction function, set inclusion in the subgroup lattice is equivalent to increased expressivity for the corresponding models.

4.2 Design of Canonicalization Functions

The canonicalization function can be chosen as any existing equivariant neural network architecture with the output being a group element; we call this the *direct approach* (figure 4.2a). For permutation groups and Lie groups, an equivariant multilayer perceptron (Shawe-Taylor, 1989; Finzi et al., 2021) can be used. We provide examples of implementations in the next section.

We also introduce an alternative method, which we call the *optimization approach* (Fig. 4.2b). The canonicalization function can be defined as

$$h(\mathbf{x}) \in \underset{\rho(g)\in\rho(\mathcal{G})}{\arg\min}\, s(\rho(g), \mathbf{x}), \tag{4.15}$$

where $s : \rho(\mathcal{G}) \times \mathcal{X} \to \mathbb{R}$ can be a neural network. When a set of elements minimizes s, one is chosen arbitrarily. s has to satisfy the following equivariance condition

$$s(\rho(g), \rho(g_1)\mathbf{x}) = s\left(\rho(g_1)^{-1}\rho(g), \mathbf{x}\right), \ \forall g_1 \in \mathcal{G}, \tag{4.16}$$

and has to be such that argmin is a subset of a coset of the stabilizer of $\mathbf{x}$. This last condition essentially means that the minimum in each orbit should be unique up to input symmetry.

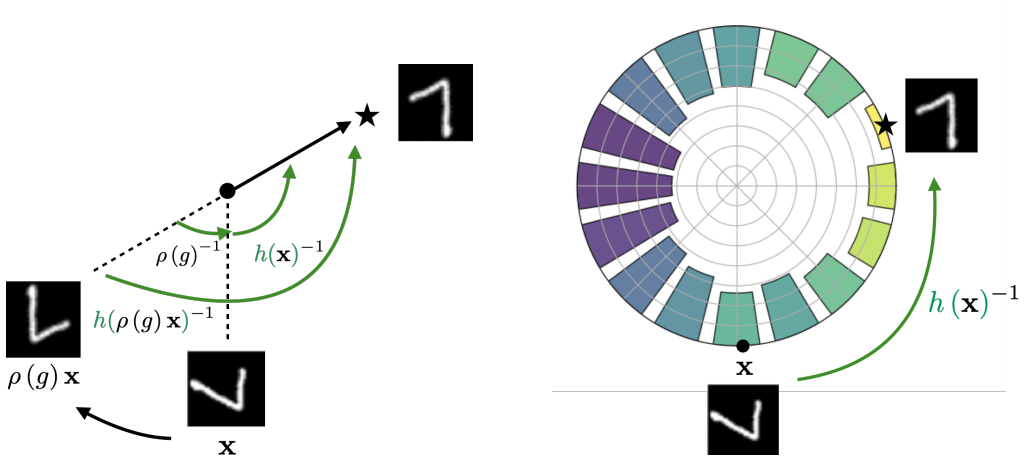
The equivariance condition on s can be satisfied using an equivariant architecture. Remarkably, it can also be satisfied using a non-equivariant function $E : \mathcal{X} \to \mathbb{R}$ and defining

$$s(\rho(g), \mathbf{x}) = E\left(\rho(g)^{-1}\mathbf{x}\right). \tag{4.17}$$

We will call the function E energy. Intuitively, E represents a distance between the input and the canonical sample of the orbit and is therefore minimized when $\rho(g)$ is the transformation that maps to the canonical sample.

This implementation presents a close analogy with the mental rotation phenomenon described in the introduction, as humans try to minimize the distance between their representation of an object and the canonical one. As such, it is expected that the optimization process will take more iterations when the input sample is farther away in orbit from the canonical sample. This is consistent with the experimental evidence for mental rotation (Shepard & Metzler, 1971; Carpenter & Eisenberg, 1978).

Simultaneous minimization and learning of s results in a bi-level optimization problem (Gould et al., 2016; Liu et al., 2021a). This can be performed



(a) Direct approach (b) Optimization approach

Figure 4.2: Two general approaches to canonicalization. In the direct approach, an equivariant neural network outputs the transformation. In the optimization approach, a function of the input is minimized to obtain the canonical sample.

in a variety of ways, including using implicit methods (Blondel et al., 2022). Next, we elaborate on how suitable canonicalization functions can be obtained in different settings.

4.2.1 Euclidean Group

The Euclidean group $E(d)$ describes rotation, translation, and reflection symmetry. Domains in which this type of symmetry is especially relevant include computer vision, point cloud modelling and physics applications. Below we give design principles to obtain equivariant models for image and point cloud inputs.

Image Input. Elements of the Euclidean group can be written as $(\mathbf{O}, \mathbf{t})$, where $\mathbf{O} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix and $\mathbf{t} \in \mathbb{R}^n$ is an arbitrary translation vector. We consider the space of image inputs $x_I \in \mathcal{X}$ as given by a 2 dimensional signal $x_I : \mathbb{R}^2 \to \mathbb{R}^C$, where C is the number of input channels. We adopt a continuous description to facilitate exposition, but in practice, all the operations are discretized using interpolation (Riba et al., 2020). This thus reduces to the *pnm* group, which is the group of n-fold discrete rotations, reflections and discrete translations. However, to maintain consistency with the rest of the chapters we will use Cn and Dn to refer to the group of discrete rotations and reflections.

The action of the representation on image inputs is defined by the following linear operator:

$$[\rho(\mathbf{O}, \mathbf{t}) \cdot x_I](\mathbf{p}) = x_I(\mathbf{O}^{-1}(\mathbf{p} - \mathbf{t})), \quad \forall \mathbf{p} \in \mathbb{R}^2,$$

where $\mathbf{p}$ is pixel position. The canonicalization function should output an element of the $E(2)$. It should also be $E(2)$-equivariant, such that $h(\rho(\mathbf{O}, \mathbf{t}) \cdot x_I) = \rho(\mathbf{O}, \mathbf{t}) \cdot h(x_I)$.

This condition can be satisfied by using a Group Equivariant CNN (G-CNN) (Cohen & Welling, 2016a) and the optimization approach described above. To do this, we define the function to be optimized as $s : O(2) \times \mathbb{R}^2 \times \mathcal{X} \to \mathbb{R}$. This can be reinterpreted as $s : \mathcal{X} \to \mathbb{R}^{O(2) \times \mathbb{R}^2}$, which means where the first dimension, a.k.a. the *fiber*, encodes rotation angles[2] and $\mathbb{R}^2$ is associated with pixel positions. If s is a G-CNN, it correctly satisfies the condition Eq. (4.16), as image rotations act on the fiber and Euclidean transformations on the pixel positions. The canonicalization is then obtained by taking the arg min over pixel positions and fibers

$$h(\mathbf{x}) \in \underset{(\mathbf{O}, \mathbf{t}) \in E(2)}{\arg\min}\ s(\mathbf{x})_{(\mathbf{O}, \mathbf{t})}\,. \tag{4.18}$$

This approach can be further simplified if we use a translation equivariant prediction network, such as a CNN-based architecture. As the translation group $T(2)$ is a normal subgroup of the Euclidean group $E(2)$, using Theorem 4.1.2, we only require the canonicalization function to be equivariant to $O(2)$. This means we can pool over the spatial dimension in the output feature map of the canonicalization function and only need to take an arg min along the rotation fiber dimension to output a rotation that can be applied to the image.

There are two potential problems with this approach. First, extending G-CNNs to a higher number of finer discrete rotations is computationally expensive, and it leads to artifacts. Second, we cannot backpropagate through the canonicalization function as the arg min operation is not differentiable.

We can avoid the first problem by using a shallower network with a larger filter size. We empirically show why this is a sound choice for canonicalization function in Section 8.4. We use the straight-through gradient estimator (Bengio et al., 2013b) to solve the second problem.

[2]In practice, we work with discrete rotations Cn.

Point Cloud Input. The $n+1$ dimensional representation of the Euclidean group (defined by concatenating a constant 1 to the original vectors) is defined in the following way

$$\rho(\mathbf{O}, \mathbf{t}) = \begin{pmatrix} \mathbf{O} & \mathbf{t} \\ \mathbf{t}^T & 1 \end{pmatrix}. \tag{4.19}$$

We seek to define an $E(d)$-equivariant canonicalization function for point clouds. This can be done by defining it as $h(\mathbf{x}) = \rho\big(h^O(\mathbf{x}), h^t(\mathbf{x})\big)$, where the function $h^O : \mathcal{X} \to \mathbb{R}^{n \times n}$ outputs the rotation and reflection and $h^t : \mathcal{X} \to \mathbb{R}^n$ the translation. Since the product of elements of $E(n)$ is given by $(\mathbf{O}_1, \mathbf{t}_1)(\mathbf{O}_2, \mathbf{t}_2) = (\mathbf{O}_1\mathbf{O}_2, \mathbf{O}_2\mathbf{t}_1 + \mathbf{t}_2)$, the equivariance condition requires that we have

$$h^O(\rho(\mathbf{O}, \mathbf{t})\mathbf{x}) = \mathbf{O}h^O(\mathbf{x}), \tag{4.20}$$

$$h^t(\rho(\mathbf{O}, \mathbf{t})\mathbf{x}) = \mathbf{O}h^t(\mathbf{x}) + \mathbf{t}. \tag{4.21}$$

This means that h^O must be $O(d)$-equivariant and translation invariant, and that h^t must be $E(d)$-equivariant. These constraints can be satisfied by using already existing equivariant architectures. Since most of the work will be done by a prediction function that can be very expressive, like Pointnet (Qi et al., 2017a), a simple and efficient architecture can be used for the canonicalization function, for example, Vector Neurons (Deng et al., 2021). The output of h^O can be made an orthogonal matrix by having it output n vectors and orthonormalizing them with the Gram-Schmidt procedure.

4.3 Experiments

To test the versatility of our method we perform experiments in three different data domains and report the results in the following subsections.

Our code can be found in `https://github.com/arnab39/equiadapt`

4.3.1 Image classification

We first perform an empirical analysis of the proposed framework in the image domain. We selected the Rotated MNIST dataset (Larochelle et al., 2007), often used as a benchmark for equivariant architectures. The task is to classify randomly rotated digits. In Table 4.1, we compare our method with different CNN and G-CNN (Cohen & Welling, 2016a) baselines. We denote the network is equivariant to Cn by putting it with the network's name (e.g.

Table 4.1: Comparison with the existing work for Rotated-MNIST.

Method	**Error % ↓**
CNN (base)	4.90 ± 0.20
G-CNN ($C4$)	2.28 ± 0.00
G-CNN ($C4$ & = params)	2.36 ± 0.15
G-CNN ($C64$ & = params)	2.28 ± 0.10
CNN (= params)	4.80 ± 0.37
Ours	
LC(PCA)-CNN	3.35 ± 0.21
LC($C4$ & frozen)-CNN	3.91 ± 0.12
LC(OPT)-CNN	3.35 ± 0.00
LC($C4$)-CNN	2.41 ± 0.10
LC($C64$)-CNN	1.99 ± 0.10

G-CNN($C4$)). For CNN (base), we choose an architecture with 7 layers where layer 1 to 3 has 32, 4 to 6 have 64, and layer 7 has 128 channels, respectively. Instead of pooling, we use convolution filters of size 5×5 with a stride 2 at layers 4 and 7. The remaining convolutions have filters of size 3×3 and stride 1. All the layers are followed by batch-norm (Ioffe & Szegedy, 2015) and ReLU activation with dropout(p=0.4) only at layers 4 and 7. For G-CNN, we took the same CNN architecture as above and replaced the standard convolutions with group convolutions (Cohen & Welling, 2016b).

For the canonicalization function, we choose a shallow G-CNN with three layers. The first layer is a lifting layer which maps the signal in the pixel space to the group with filters that are the same size as the input image. This is followed by ReLU nonlinearity and group equivariant layers with 1×1 filters. We learn the canonicalization function end-to-end with a CNN as the prediction function and denote it as LC(Cn)-CNN where LC stands for learned canonicalization.

We also implement the optimization approach of Eq. (4.17) as LC(OPT)-CNN. Our E converts the input image into a point cloud representation, which is fed into a PointNet that produces the energy. We use gradient descent to optimize this energy with respect to the input rotation for a small number of steps. For the energy function E, the image is transformed to a point cloud and fed into a Deep Sets (Zaheer et al., 2017b) architecture. Then, E is optimized by 5 steps of gradient descent (learning rate 0.1) using

Table 4.2: Ablation study on the effect of augmentation.

Method	**Error %** $\downarrow$
CNN (base)	4.90 ± 0.20
CNN (rotation aug.)	3.30 ± 0.20
LC(pretrained)-CNN	2.05 ± 0.15
LC(p64)-CNN	1.99 ± 0.10

implicit differentiation. This procedure is visualized in Fig. 4.2b.

For the pure G-CNN-based baseline, we provide the value reported by Cohen & Welling (2016a) and design a variant which has a similar architecture to CNN (base) while matching the number of parameters of our LC($C4$)-CNN. We call this G-CNN (p4 & = params).

Lastly, we consider variants where the canonicalization function is not learned. The first one is using a G-CNN similar to LC(Cn) but with weights frozen at initialization. We call them LC($C4$ & frozen)-CNN and LC($C64$ & frozen)-CNN. For the second one, canonicalization is performed by finding the orientation of the digits using Principal Component Analysis (PCA) and we refer to it as LC(PCA)-CNN.

Training details. In all our image experiments, we train the models by minimizing the cross entropy loss for 100 epochs using Adam (Kingma & Ba, 2014) with a learning rate of 0.001. We perform early stopping based on the classification performance of the validation dataset with patience of 20 epochs.

Results As reported in Table 4.1 the direct canonicalization approach outperforms the CNN-based baseline and is comparable to G-CNNs. The optimization version does not perform as well, even if it is still better than the non-equivariant baseline. We have found that this is because gradient descent can get stuck in flat regions. We see that using a fixed canonicalization function technique like PCA or canonicalization function with frozen parameters improves performance over the CNN baseline. Learning the canonicalization function provides a significant performance improvement.

Ablation study We further seek to understand if learning the canonicalization performs better mainly because a meaningful function is learned, or because this implicitly augments the prediction CNN with rotations during training. We perform an ablation study to investigate this. First, we com-

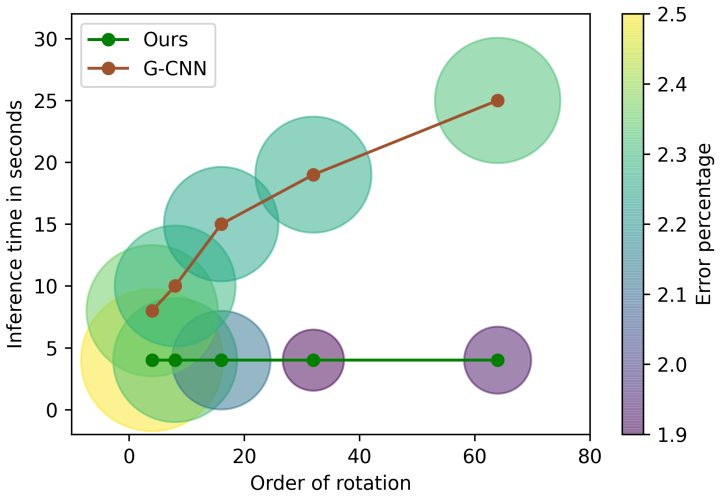


Figure 4.3: Inference time comparison of our method with G-CNN with increasing order of rotations.

pare with a CNN trained with random rotation augmentations. Second, we compare with a setup we call LC(pretrained), in which canonicalization is learned along with a CNN prediction network. Then the prediction function is reinitialized and trained from scratch while the canonicalization function is fixed. If meaningful canonicalization is learned, this setup should perform close to the one where the canonicalization is learned end-to-end. We see from the results of Table 4.2 that this is the case. The pretrained canonicalization is competitive with the end-to-end one and significantly better than data augmentation. Next, we perform experiments to understand the role of different components in our model using the $C8$ group. First, we

Table 4.3: Impact of the number of layers in canonicalization function network and order of the discrete rotations to which it is equivariant on the performance.

#lyrs	**Order of the discrete rotation group**				
	$C4$	$C8$	$C16$	$C32$	$C64$
1	2.52 ± 0.12	2.37 ± 0.09	2.20 ± 0.08	2.05 ± 0.15	2.01 ± 0.09
2	2.44 ± 0.06	2.31 ± 0.05	2.16 ± 0.09	2.00 ± 0.07	2.02 ± 0.12
3	2.41 ± 0.11	2.28 ± 0.09	2.11 ± 0.06	1.98 ± 0.09	1.99 ± 0.10

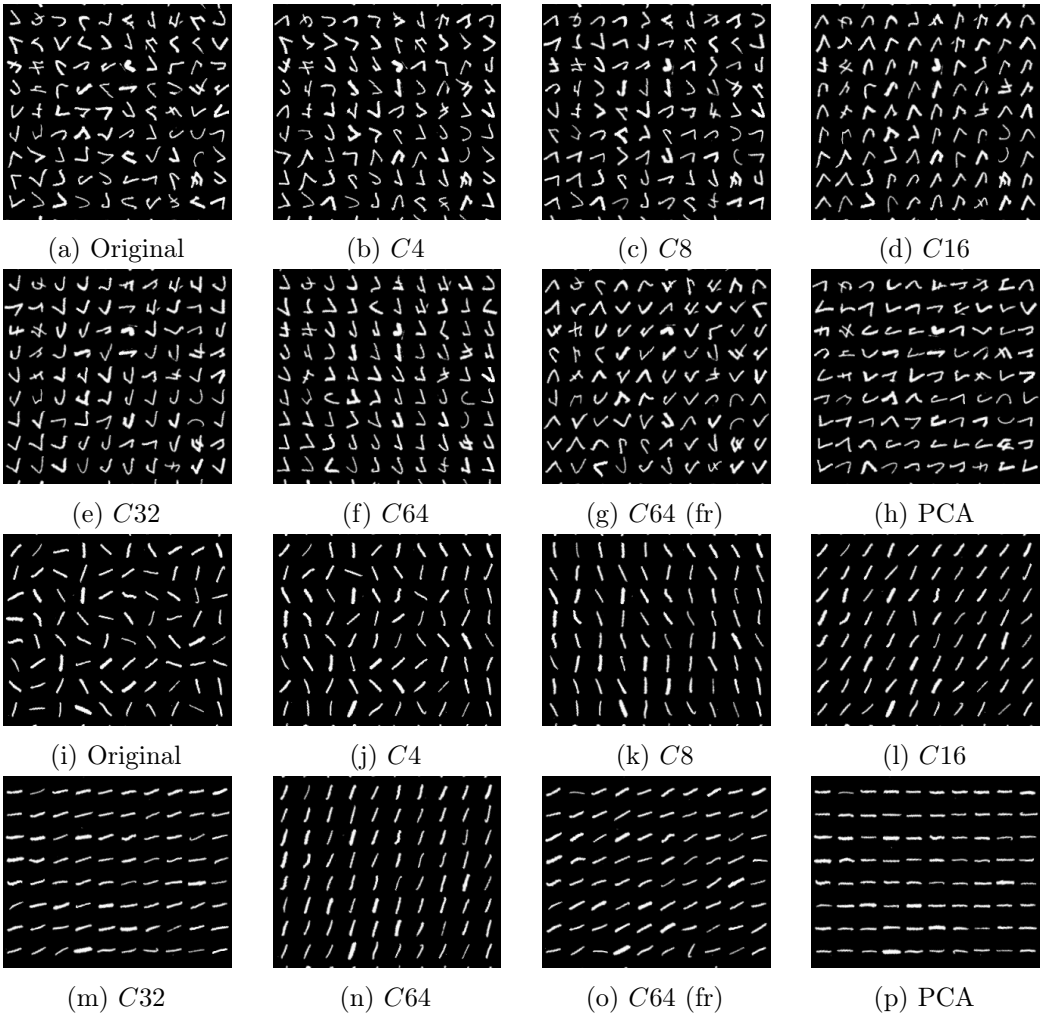
vary the number of layers of the canonicalization network and the number of rotations it is equivariant to. For this, we extend the layers of G-CNN

to any arbitrary rotations. As we noticed that using a larger filter leads to better performance for higher order rotations, we stick to architecture with a lifting layer with image-sized filters followed by 1×1 filters. From Table 4.3, we notice that adding equivariance to higher order rotation in the canonicalization function leads to significant performance improvement compared to adding more layers. Fig. 4.4 shows the canonical orientation resulting from the learnt canonicalization function with a single lifting layer on 90 randomly sampled images of class 7 from the test dataset. This suggests that a shallow network is sufficient to achieve good results with a sufficiently high order of discrete rotations. For $c64$, we see that all the similar-looking samples are aligned in one particular orientation. In contrast, although techniques like PCA or freezing parameters of the canonicalization function find the correct canonicalization function for simple digits like 1, they struggle to find stable mappings for more complicated digits like 7.

Inference time Next, we compare the inference time of our model with pure G-CNN-based architectures. For this experiment, we take the CNN architecture of our predictor network and replace the convolutions with group convolutions. As increasing the rotation order in G-CNN requires more copies of rotated filters in the lifting layer and more parameters in the subsequent group convolution layers, we decreased the number of channels to keep the number of parameters the same as our model. Figure 4.3 shows that although G-CNN's performance is slightly better for the $C4$ group, increasing the order of discrete rotations improves our model's performance significantly compared to G-CNN. In addition to performance gain, our model's inference speed remains more or less constant while encoding invariance to higher-order rotations due to the shallow canonicalization network. This makes our approach more suitable for building equivariance for bigger groups and network architectures.

4.3.2 N-body dynamics prediction

Simulation of physical dynamics is an important class of $E(3)$-equivariant problems due to the symmetry of physical laws under rotations and translations. We evaluate our framework in this setting with the N-body dynamics prediction task proposed by (Kipf et al., 2018) and (Fuchs et al., 2020b). In this task, the model has to predict the future positions of 5 charged particles interacting with Coulomb force given initial positions and velocities. We use the same version of the dataset and setup as (Satorras et al., 2021).



(a) Original (b) $C4$ (c) $C8$ (d) $C16$

(e) $C32$ (f) $C64$ (g) $C64$ (fr) (h) PCA

(i) Original (j) $C4$ (k) $C8$ (l) $C16$

(m) $C32$ (n) $C64$ (o) $C64$ (fr) (p) PCA

Figure 4.4: Canonicalized images from different canonicalization functions for digit 7 and 1.

For this experiment, our architecture uses a simple 2-layer Vector Neurons version of the Deep Sets architecture for the canonicalization function (Deng et al., 2021; Zaheer et al., 2017a). The prediction function is a 4-layer Graph Neural Network (GNN) with the same hyperparameters as the one used in (Satorras et al., 2021), and (Puny et al., 2022) for a fair comparison. The architecture of the prediction network was, therefore, not optimized. The canonicalization network is much smaller than the prediction GNN, with

Table 4.4: Test MSE for the N-body dynamics prediction task.

Method	MSE
Linear	0.0819
SE(3) Transformer (Fuchs et al., 2020b)	0.0244
TFN (Thomas et al., 2018)	0.0155
GNN (Gilmer et al., 2017a)	0.0107
Radial Field (Köhler et al., 2019)	0.0104
EGNN (Satorras et al., 2021)	0.0071
FA-GNN (Puny et al., 2022)	0.0057
LC-GNN	**0.0043 ± 0.0001**
LC-GNN-$O(3)$	0.0045 ± 0.0001
LC-GNN (frozen)	0.0085 ± 0.0002

around 20 times fewer parameters. This allows us to test the hypothesis again that only a simple canonicalization function is necessary to achieve good performance.

Training details. Using the Adam optimizer, we train on mean square error (MSE) loss between predicted and ground truth. We train for 10.000 epochs and use early stopping. We use weight decay 10^{-8} and dropout in the canonicalization function with $p = 0.5$.

Results Table 4.4 shows that we obtain state-of-the-art results. The improvement with respect to Frame Averaging is significant, showing that learning the canonicalization provides an important advantage. Our approach also does better than all the intrinsically equivariant (or *viewpoint-independent*) baselines both in accuracy and efficiency. This shows that canonicalization can be used to obtain equivariant models with high generalization abilities without sophisticated architectural choices.

Ablation study We also test variants of the model. First, we test on a variant of the model where the canonicalization is only learned for the $O(3)$ part of the transformation and where the translation part is given by the centroid. Since, for this system, all the masses are identical, this is the same as the center of mass of the system. The result is reported in Table 4.4 as LC-GNN-$O(3)$. We obtain only marginally worse performance compared to the fully trained canonicalization function. This shows that, in this setting, the centroid provides an already suitable canonicalization function, which is

expected given the physical soundness of choosing the center of mass as the origin of the reference frame. Since the learned translation canonicalization performs on par with this physically motivated canonicalization, this also validates the method.

The comparison with Frame Averaging is also insightful. PCA-based Frame Averaging can also be motivated from a physical point of view since this method is equivalent to identifying the principal axes using the tensor of inertia. It is, therefore, a physical heuristic for $O(3)$ canonicalization. By contrast with the translation canonicalization with the centroid, for orthogonal transformations learning, the canonicalization performs significantly better.

Second, we compare with a version of the model where the weights of the canonicalization function are frozen at initialization. This canonicalization still provides $E(n)$-equivariance and, as expected, provides a significant improvement of more than 20% with respect to the GNN prediction function alone. However, the learned canonicalization function provides a close to 50% improvement in performance compared to this fixed canonicalization.

4.3.3 Point cloud classification and segmentation

We use the ModelNet40 (Wu et al., 2015) and ShapeNet (Chang et al., 2015) datasets for experiments on point clouds. The ModelNet40 dataset consists of 40 classes of 3D models, with a total of 12,311 models. 9,843 models were used for training, and the remaining models were used for testing in the classification task. The ShapeNet dataset was used for part segmentation with the ShapeNet-part subset, which includes 16 categories of objects and more than 30,000 models. In the classification and segmentation task, the train/test rotation setup adhered to the conventions established by (Esteves et al., 2018a) and adopted by (Deng et al., 2021). Three settings were implemented: z/z, $z/SO(3)$, and $SO(3)/SO(3)$. The notation z denotes data augmentation with rotations around the z-axis during training, while $SO(3)$ represents arbitrary rotations. The notation x/y denotes that transformation x is applied during training and transformation y is applied during testing.

We design our Canonicalization Network using layers from Vector Neurons (Deng et al., 2021), where the final output contains three 3D vectors that are obtained by pooling over the entire point cloud. We then orthonormalize the three vectors using the Gram-Schmidt orthonormalization process to define a 3D ortho-normal coordinate frame or a rotation matrix. We canonicalize the point cloud by acting on it using this rotation matrix. We use a two-layered Vector Neuron network followed by global pooling, which we call LC(NL). To

Table 4.5: Test classification accuracy of different point cloud models on the ModelNet40 dataset (Wu et al., 2015) in three train/test scenarios. This table is borrowed from (Deng et al., 2021). z here stands for aligned data augmented by random rotations around the vertical axis, and SO(3) indicates data augmented by random 3D rotations.

Method	z/z	z/SO(3)	SO(3)/SO(3)
Point / mesh inputs			
PointNet (Qi et al., 2017a)	85.9	19.6	74.7
DGCNN (Wang et al., 2019b)	90.3	33.8	88.6
VN-PointNet (Deng et al., 2021)	77.5	77.5	77.2
VN-DGCNN (Deng et al., 2021)	89.5	89.5	90.2
PCNN (Atzmon et al., 2018)	92.3	11.9	85.1
ShellNet (Zhang et al., 2019b)	93.1	19.9	87.8
PointNet++ (Qi et al., 2017b)	91.8	28.4	85.0
PointCNN (Li et al., 2018)	92.5	41.2	84.5
Spherical-CNN (Esteves et al., 2018a)	88.9	76.7	86.9
a^3S-CNN (Liu et al., 2018)	89.6	87.9	88.7
SFCNN (Rao et al., 2019)	91.4	84.8	90.1
TFN (Thomas et al., 2018)	88.5	85.3	87.6
RI-Conv (Zhang et al., 2019a)	86.5	86.4	86.4
SPHNet (Poulenard et al., 2019)	87.7	86.6	87.6
ClusterNet (Chen et al., 2019)	87.1	87.1	87.1
GC-Conv (Zhang et al., 2020b)	89.0	89.1	89.2
RI-Framework (Li et al., 2020)	89.4	89.4	89.3
Ours			
LC(frozen)-PointNet	78.9 ± 2.1	78.7 ± 2.2	78.4 ± 2.5
LC(L)-PointNet	79.8 ± 1.4	79.6 ± 1.3	79.6 ± 1.4
LC(NL)-PointNet	79.9 ± 1.3	79.6 ± 1.3	79.7 ± 1.3
LC(frozen)-DGCNN	88.3 ± 2.1	88.3 ± 2.1	88.3 ± 2.1
LC(L)-DGCNN	88.9 ± 1.8	88.6 ± 1.9	88.6 ± 2.0
LC(NL)-DGCNN	88.7 ± 1.8	88.8 ± 1.9	90.0 ± 1.1

support our hypothesis that the canonicalization function can be inexpressive, we use a single linear layer of Vector neuron followed by pooling and call this model LC(L). Furthermore, to understand the significance of learning canonicalization, we freeze the weights of the Canonicalization Network and call this variant LC(frozen). We use PointNet and DGCNN (Wang et al., 2019b) as the prediction networks in our experiments.

Training Details We use cross entropy loss and Stochastic Gradient Descent (SGD) optimizer to train the network for 200 epochs in all of our pointcloud experiments. We use an initial learning rate of 0.1 and a cosine annealing schedule with an end learning rate of 0.001.

Results Table 4.5 contains the results of the ShapeNet experiment, showing the classification accuracy for different augmentation strategies during training and evaluation: z/z, $z/\mathrm{SO}(3)$, and $\mathrm{SO}(3)/\mathrm{SO}(3)$. Our method, which includes LC(frozen)-PointNet, LC(L)-PointNet, LC(NL)-PointNet, LC(frozen)-DGCNN, LC(L)-DGCNN, and LC(NL)-DGCNN, demonstrates competitive results across all rotation types. We achieve similar results in the ShapeNet part segmentation task as presented in Table 4.6. In particular, we observe three trends in our results. First, learning canonicalization slightly improves the performance, except in the case where the test point clouds are already aligned (z/z column of Table 4.5). Second, using shallow linear canonicalization achieves good results. Third, the performance of the prediction network bottlenecks the model's performance. This verifies our hypothesis that decoupling the equivariance using a simple canonicalization network results in a better and more expressive non-equivariant prediction network to improve the performance of the task while still being equivariant. In Table 4.7, we also show that the inference time of our algorithm is dominated by the prediction network's inference time. The overhead of canonicalization is negligible, which makes our method faster than existing methods that modify the entire architecture like Vector Neurons (Deng et al., 2021).

4.4 Related Works

Methods based on heuristics to standardize inputs have been around for a long time (Yüceer & Oflazer, 1993; Lowe, 2004). However, these approaches require significant hand-engineering and are difficult to generalize. An important early work is the Spatial Transformer Network (Jaderberg et al., 2015) which learns input transformations to facilitate processing in a downstream vision task. PointNet (Qi et al., 2017a) also proposed to learn an alignment network to encourage invariance for point cloud analysis. However, these approaches are closer to regularizers and provide no equivariance guarantees. The works of (Esteves et al., 2018b; Tai et al., 2019) provided equivariant versions of the Spatial Transformer using an approach based on canonical coordinates. One limitation of this approach is that it does not exactly handle equivariance to groups that are larger in dimension than the dimension of the data grid. Some

Table 4.6: ShapeNet part segmentation results. Overall average category mean IoU over 16 categories in two train/test scenarios are reported. z here stands for aligned data augmented by random rotations around the vertical axis, and SO(3) indicates data augmented by random 3D rotations

Methods	z/SO(3)	SO(3)/SO(3)
Point / mesh inputs		
PointNet (Qi et al., 2017a)	38.0	62.3
DGCNN (Wang et al., 2019b)	49.3	78.6
VN-PointNet(Deng et al., 2021)	72.4	72.8
VN-DGCNN(Deng et al., 2021)	**81.4**	**81.4**
PointCNN (Li et al., 2018)	34.7	71.4
PointNet++ (Qi et al., 2017b)	48.3	76.7
ShellNet (Zhang et al., 2019b)	47.2	77.1
RI-Conv (Zhang et al., 2019a)	75.3	75.3
TFN (Thomas et al., 2018)	76.8	76.2
GC-Conv (Zhang et al., 2020b)	77.2	77.3
RI-Framework (Li et al., 2020)	79.2	79.4
Ours		
LC(frozen)-PointNet	72.1 ± 0.8	72.3 ± 1.1
LC(L)-PointNet	73.4 ± 1.2	73.2 ± 0.9
LC(NL)-PointNet	73.5 ± 0.8	73.6 ± 1.1
LC(frozen)-DGCNN	78.1 ± 1.2	78.2 ± 1.2
LC(L)-DGCNN	78.5 ± 1.1	78.3 ± 1.2
LC(NL)-DGCNN	78.4 ± 1.0	78.5 ± 0.9

Table 4.7: Inference time (in seconds) of the networks for ModelNet40 classification test split in 1 A100 and 8 CPUs with a batch size of 32. Vanilla denotes no modification to the base network, while Vector Neuron and Canonicalization denote that the base network is redesigned/enhanced with them to be equivariant.

Base Network	Vanilla	Vector Neuron	Canonicalization
PointNet	18s	30s	20s
DGCNN	23s	39s	25s

recent works have proposed using learned coordinate frames for point clouds (Kofinas et al., 2021; Luo et al., 2022; Du et al., 2022). We provide theoretical and experimental evidence that the neural networks for canonicalization can be made much shallower and simpler without affecting performance. (Bloem-Reddy & Teh, 2020), introduce the concept of *representative equivariant*, which is similar to what we implement in this work. Finally, other recent

works (Winter et al., 2022; Vadgama et al., 2022) have proposed to use canonicalization in an autoencoding setup.

4.5 Discussion

In this chapter, we propose using a learned canonicalization function to obtain equivariant machine-learning models. These canonicalization functions can conveniently be plugged into existing architectures, resulting in highly expressive models. We have described general approaches to obtain canonicalization functions and specific implementation strategies for the Euclidean group (for images and point clouds) and the symmetric group.

We performed experimental studies in the image, dynamical systems and point cloud domains to test our hypotheses. First, we show that our approach achieves comparable or better performance than baselines on invariant tasks. Importantly, learning the canonical network is a better approach than using a fixed mapping, either a frozen neural network or a heuristic approach. This could be due to a combination of two factors. First, the learned canonicalizations have some consistency and help the prediction network perform the task. This is shown explicitly for our results in the image domain. Second, the process of learning the canonicalization induces an implicit augmentation of the data. This should help the prediction function generalize better and be more robust to potential failings of the canonicalization function. The method therefore combines some of the advantages of data augmentation with exact equivariance which is discussed in details in the next chapter. Our results also show that the canonicalization function can be realized with a relatively shallower equivariant network without hindering performance. Finally, we show that this approach reduces inference time and is more suitable for bigger groups than G-CNNs on images.

Limitations and Future work One limitation of our method is that there are no guarantees that the canonicalization function is smooth. This may be detrimental to generalization as small changes in the input could lead to large variations in the input to the prediction function. Another limitation could arise in domains in which semantic content is lacking to identify a meaningful canonicalization, for example, some types of astronomical images or biological images.

Multiple extensions of this framework are possible. Future work could include experimentation on canonicalization for the symmetric group. Other ways to build canonicalization functions could also be investigated, such as

using steerable networks for images. The function would output an orientation fibre that transforms by the irreducible representation of the special orthogonal group. Understanding how design choices for canonicalization functions (for example, the subgroup $\mathcal{K}$) affect downstream performance could also be a fruitful research direction. Finally, making large pretrained architectures equivariant using this framework could be an exciting extension which we explore in the next chapter.

5

Equivariant Adaptation of Large Pretrained Models

In the previous chapter, we introduced a new technique to learn equivariant functions using canonicalization. One main benefit of our method is that it allows us to use any architecture to make task predictions once the input is canonicalized. This brings us to the question: can we plug in any existing powerful pretrained prediction neural network into this equivariance formulation?

Recent research has shown that scaling models, both in terms of the number of parameters and the amount of training data, systematically improve their performance (Kirillov et al., 2023; Tan & Le, 2019; Chen et al., 2020d; Brown et al., 2020a; Radford et al., 2021; Zhai et al., 2022; Xie et al., 2023). Pretraining on massive datasets has emerged as a practical method to harness this advantage. Several large models are now publicly available, and fine-tuning them for specific applications has proven to be successful across various domains. However, such *foundation models* (Bommasani et al., 2021) are typically not equivariant to most transformations of the input data except translations, as current methods to achieve this are non-trivial to adapt to existing architectures and remain computationally expensive.

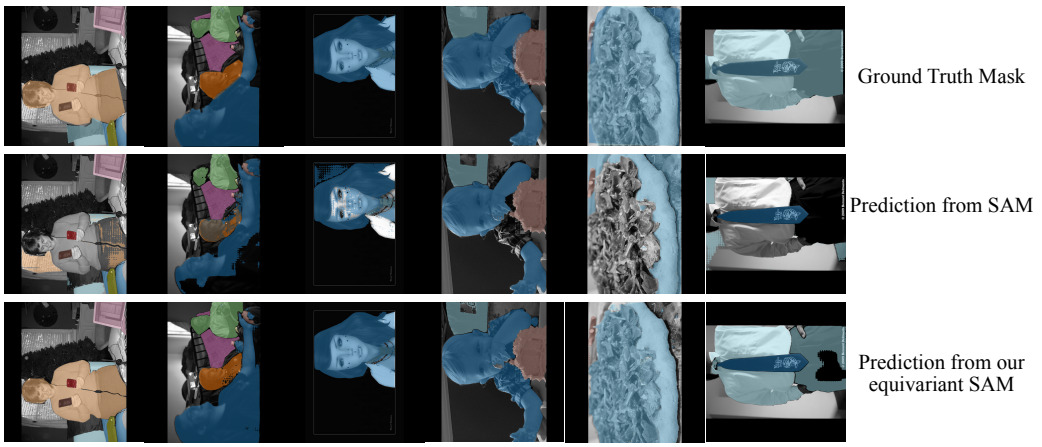


Figure 5.1: Predicted masks from the Segment Anything Model (SAM) (Kirillov et al., 2023), showcasing both the original model and our proposed equivariant adaptation for 90° counter-clockwise rotated input images taken from the COCO 2017 dataset (Lin et al., 2014). Our method makes SAM equivariant to the group of 90° rotations while only requiring 0.3% extra parameters and modestly increasing the inference time by 7.3%.

In this chapter, which is taken from (Mondal et al., 2024), we show that it is possible to bridge the gap between foundation models and the systematic generalization offered by equivariance through the concept of a learned *canonicalization* function from the previous chapter. However, a naive application of the canonicalization idea fails in practice. This is because the canonicalization network's choice of canonical form can result in a change of distribution in the input of the pretrained prediction network – that is, canonicalization can be uninformed about the preference of the prediction network, undermining its performance.

As outlined in Figure 5.4, we resolve this misalignment by matching the distribution of predicted canonical forms with a prior distribution of orientations in the pertaining dataset. We empirically demonstrate that imposing this prior is essential for the equivariant adaptation of pretrained models across different domains and datasets. Our approach offers a practical solution for obtaining large-scale equivariant models by providing an independent module that can be integrated into existing large pretrained foundation models, making them equivariant to a wide range of transformations like rotations.

5.1 Deeper Dive Into Canonicalization

The flexibility of the equivariance with canonicalization approach enables the conversion of any existing large pretrained Deep Neural Network (DNN) into an equivariant model with respect to certain known transformations. To achieve this, the pretrained DNN can be utilized as the prediction network in the formulation provided by Eq. (4.2). Subsequently, a canonicalization function can be designed to produce the elements of the known group of the transformation while maintaining equivariance with respect to this group.

One could learn the canonicalization function while optionally finetuning the prediction function using the same task objective – in our experiments, we consider both the zero-shot and fine-tuning setup.

The performance of the model requires the *alignment* of these two networks. For example, if the canonicalization network produces upside-down natural images, compared to those that the pretrained network is expecting, the overall performance is significantly degraded. In addition to alignment, there is an *augmentation* effect that further muddies the water: during its training, the canonicalization network performs data augmentation. As we see shortly, one needs to consider both alignment and augmentation effects when analyzing the performance of this type of equivariant network.

When both networks are trained together from scratch, the alignment is a non-issue, and (unwanted) augmentation can degrade or improve performance, depending on the extent of symmetry in the dataset. However, when dealing with pretrained prediction networks one needs to also consider the alignment effect. One could then think of freezing the pretrained prediction network, therefore avoiding unwanted augmentation, and backpropagating the task loss through it to align the canonicalization network. However, this can become prohibitively expensive for large pretrained models, such as segment anything (SAM) considered in this work. We propose an alternative, where we directly regularize the canonicalization network to produce canonical forms consistent with the training data, which in turn aligns with the prediction network.

5.1.1 Learning Canonicalization, Augmentation and Alignment

When learning the canonicalization function during training, the process implicitly performs dynamic augmentation of the prediction network. Consider a model designed to be equivariant to a certain group of transformations $\mathcal{G}$ by canonicalization. At the start of training, the randomly initialized weights

of the canonicalization function will output random canonical orientations for each data point. This has a similar effect to data augmentation using the group $\mathcal{G}$ for the prediction network. As training progresses, the canonical orientations for similar-looking images begin to converge, as demonstrated in Figure 5.2, causing the augmentation effect to diminish. Thus, in addition to guaranteeing equivariance, this formulation provides a free augmentation effect to the prediction network during initial training stage.

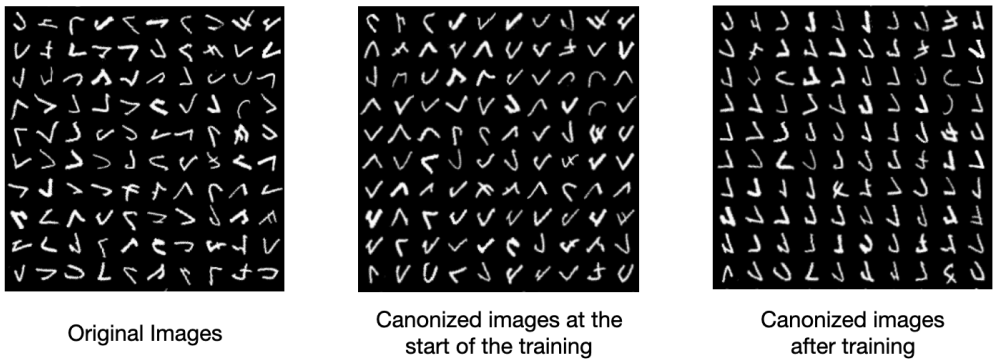


Figure 5.2: Visualization of the diminishing augmentation effect introduced by learning canonicalization (previous chapter) during training for rotated MNIST dataset. In this visualization, the leftmost image represents the original training images. Moving towards the center, we present the canonicalized images at the beginning of the training process. Finally, the rightmost image unveils the transformation of the canonized images after training the model for 100 epochs.

However, there are two scenarios where the augmentation provided by learning the canonicalization function can be detrimental:

First, in cases where the training only requires small transformations as augmentations, providing all the transformations of a group $\mathcal{G}$ can actually hurt the performance of the prediction network during the start of *training* or *fine-tuning*. For example, in natural image datasets like CIFAR10, small rotation augmentations (from -10 to $+10$ degrees) are beneficial, while a canonicalization function would output any rotation from -180 to $+180$ degrees during the beginning phase of training. This can lead to unstable training of the prediction network and hinder the model's performance by training on additional data that is far outside the distribution of the train set. We show this in Table 5.1 by training a prediction network with different

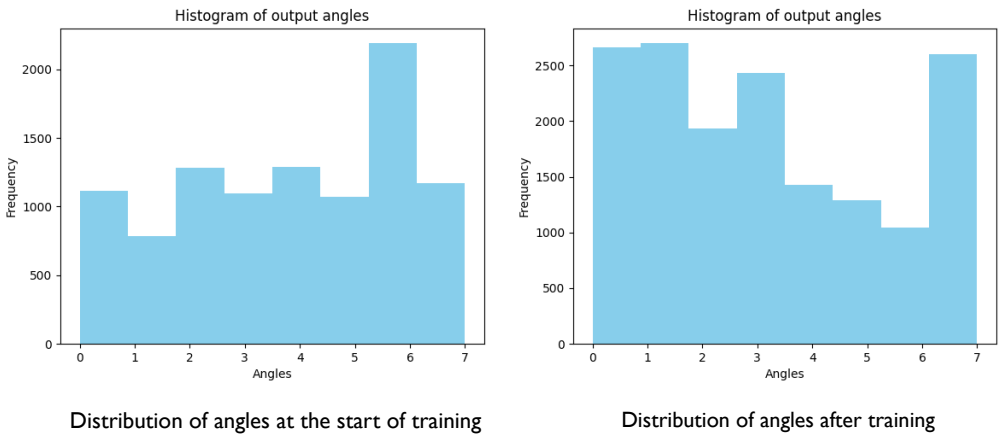


Figure 5.3: Distribution of angles output from canonicalization function in $C8$ for Learned Canonicalization (previous chapter) for CIFAR10 (Krizhevsky & Hinton, 2009) before and after training. We use indices on the x-axis instead of angle values to represent the corresponding multiple of 45°. Frequency denotes the number of images mapped to a particular multiple of 45°.

rotation augmentations, including the one due to learned canonicalization on both CIFAR datasets. Furthermore, we also observe that this effect is more pronounced when the prediction network is trained from scratch, and the dataset is more complicated with a larger number of classes. This effect can be understood as an example of the *variance-invariance tradeoff* introduced by (Chen et al., 2020a). Since, the test distribution is not perfectly symmetric under rotations, training with arbitrary augmentations biases the prediction function.

Second, we notice that relying solely on the task loss objective is not sufficient for the canonicalization function to learn the correct orientation. This could be due to the small size of the finetuning dataset compared to the pretraining dataset. We see experimentally that this leads to the canonicalization function outputting inconsistent canonical orientations during inference, impacting the prediction network's performance. For instance, in natural image datasets like CIFAR10 (non-augmented), we expect the canonical orientation for every datapoint to be the original images after fine-

Table 5.1: Effect of augmentation on the Prediction network. Top-1 classification accuracy and $\mathcal{G}$-Averaged classification accuracy for CIFAR10 and CIFAR100 (Krizhevsky & Hinton, 2009). $C8$-Avg Acc refers to the top-1 accuracy on the augmented test set obtained using the group $\mathcal{G} = C8$, with each element of $\mathcal{G}$ applied on the original test set.

Dataset →		CIFAR10		CIFAR100	
Pred. Network ↓	Rot Aug	Acc	$C8$-Avg Acc	Acc	$C8$-Avg Acc
ResNet50	−10 to +10 deg	**90.96 ± 0.41**	44.87 ± 0.60	**74.83 ± 0.15**	37.14 ± 0.42
	−180 to +180 deg	84.60 ± 1.83	81.04 ± 1.86	61.07 ± 0.27	59.42 ± 0.70
	LC	83.11 ± 0.35	**82.89 ± 0.41**	59.84 ± 0.67	**59.45 ± 0.49**
ResNet50 (pretrained)	−10 to +10 deg	**96.97 ± 0.01**	57.77 ± 0.25	**85.84 ± 0.10**	44.86 ± 0.12
	−180 to +180 deg	94.91 ± 0.07	90.11 ± 0.19	80.21 ± 0.09	74.12 ± 0.05
	LC	93.29 ± 0.01	**92.96 ± 0.09**	78.50 ± 0.15	**77.52 ± 0.07**

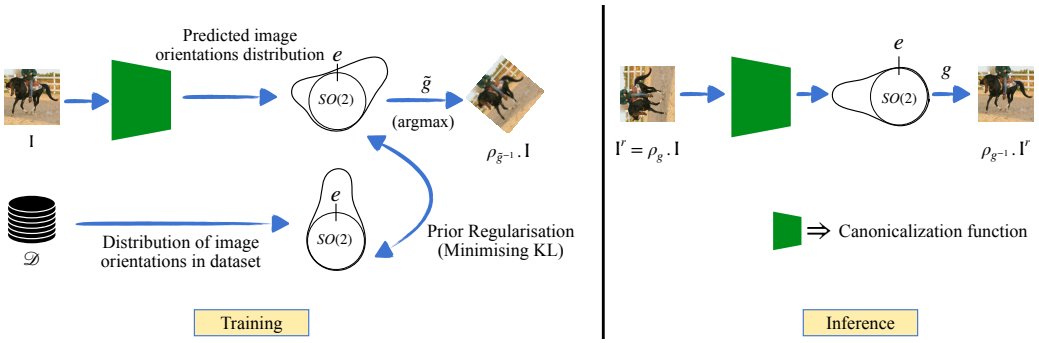


Figure 5.4: Training and inference with our proposed regularized canonicalization method. The canonicalization function outputs a distribution over image transformations and samples from that is used to canonicalize the input image. Additionally, during training, this predicted distribution is regularized to match the transformations seen in the dataset.

tuning. [1] However, from Figure 5.3, we can see that the canonical orientations for the test set are distributed uniformly from −180 to +180 degrees even after training until convergence of the task objective. As a result, during inference, the prediction network will view images with different orientations and underperform. This issue arises from the fact that the prediction networks are not inherently robust to these transformations.

5.1.2 Canonicalization Prior

As we have seen, the canonicalization network may induce a shift in orientations away from those present in the training datasets for the prediction network. To encourage the canonicalization function to align inputs in an orientation that will help the prediction network, we introduce a simple regularizer we call canonicalization prior (CP). It is motivated by noticing that the images or point clouds in the finetuning dataset combined with the pretrained model provide useful information about the existing orientation bias in the pretrained prediction model. In case of natural image datasets, we expect the canonical orientations in the fine-tuning dataset like CIFAR10 to be similar to those of the pretraining dataset like ImageNet-1k. We, therefore, take as prior that the canonicalization should align inputs as closely as possible to their original orientation in the finetuning dataset.

We derive the regularizer by taking a probabilistic point of view. The canonicalization function maps each data point to a distribution over the group of transformations, denoted as $\mathcal{G}$. Let $\mathbb{P}_{c(\mathbf{x})}$ denote the distribution induced by the canonicalization function over $\mathcal{G}$ for a given data point $\mathbf{x}$. We assume the existence of a canonicalization prior associated with the dataset $\mathcal{D}$ that has a distribution $\mathbb{P}_{\mathcal{D}}$ over $\mathcal{G}$. To enforce the canonicalization prior, we seek to minimize the Kullback-Leibler (KL) divergence between $\mathbb{P}_{\mathcal{D}}$ and $\mathbb{P}_{c(\mathbf{x})}$ over the entire dataset $\mathcal{D}$ that is $\mathcal{L}_{\text{prior}} = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[D_{KL}(\mathbb{P}_{\mathcal{D}} \parallel \mathbb{P}_{c(\mathbf{x})})\right]$.

We assume that the canonicalization function c estimates the parameters of a distribution over rotations with probability density function $p\left(\mathbf{R} \mid c\left(\mathbf{x}\right)\right)$. We denote the probability density function of the prior to be $q\left(\mathbf{R}\right)$. Since the prior distribution is kept constant, minimizing the KL divergence is equivalent to minimizing the cross-entropy, and the prior loss simplifies to:

$$\mathcal{L}_{\text{prior}} = -\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\ \mathbb{E}_{\mathbf{R}\sim q}\left[\log p\left(\mathbf{R} \mid c\left(\mathbf{x}\right)\right)\right] \tag{5.1}$$

Hereafter, we derive the regularization for the discrete and continuous cases separately.

Discrete Rotations

We first consider the group of 2D discrete rotations, the cyclic group C_n, which can be seen as a discrete approximation to the full rotation group $SO(2)$. In this case, we consider a categorical distribution over group elements, with

[1] As pretrained networks are trained on ImageNet which consists of upright natural images.

the prior distribution having a probability mass of 1 for the identity element. Then $p_{\mathcal{D}}(\mathbf{R}) = \delta_{\mathbf{R},\mathbf{I}}$, where $\delta_{\mathbf{R},\mathbf{I}}$ is the Kronecker delta function and the cross entropy in Eq. (5.1) becomes $-\log p_{c(x)}(\mathbf{I})$. Hence, the loss becomes $\mathcal{L}_{\text{prior}} = -\mathbb{E}_{x\sim\mathcal{D}} \log p_{c(x)}(\mathbf{I})$. In other words, the regularization loss is simply the negative logarithm of the probability assigned by the canonicalization function to the identity element $\mathbf{I}$ of the group.

Practical Implementation For images, similar to the previous chapter, the canonicalization network needs to output logits corresponding to every group element in the discrete cyclic group C_n. This can be achieved by using a Group Convolutional Network (Cohen & Welling, 2016c) or an $E(2)$-Steerable Network (Weiler & Cesa, 2019c) that produces outputs using *regular* representation. To design the canonicalization function, we take a spatial average and get logits corresponding to every element in the group along the fibre dimension. This is similar to the approach used in the previous chapter. Now, we can get a discrete distribution over the group elements by taking a softmax and minimizing the prior loss along with the task objective. During training, to ensure consistency with the implementation in the previous chapter for fair comparisons across all experiments, we utilize the argmax operation instead of sampling from this distribution using Gumbel Softmax (Jang et al., 2017) and employ the straight-through gradient trick (Bengio et al., 2013b). All our image-based experiments use this discrete rotation model.

Continuous rotations

When considering canonicalization with continuous rotations, it is natural to use the matrix Fisher distribution introduced by (Downs, 1972). It is the analogue of the Gaussian distribution on the $SO(n)$ manifold and is defined as

$$p\left(\mathbf{R} \mid \mathbf{F}\right) = \frac{1}{n\left(\mathbf{F}\right)} \exp\left(\operatorname{Tr}\left[\mathbf{F}^T\mathbf{R}\right]\right) \tag{5.2}$$

where $\mathbf{F} \in \mathbb{R}^{n\times n}$ is the parameter of the distribution and $n\left(\mathbf{F}\right)$ is a normalization constant. Interpretation of the parameter $\mathbf{F}$ and useful properties of the distribution are provided in (Khamsi & Kirk, 2011; Lee, 2018; Mohlin et al., 2020). In particular, considering the proper singular value decomposition $\mathbf{F} = \mathbf{USV}^T$, we find that $\hat{\mathbf{R}} \equiv \mathbf{UV}^T$ is the mode of the distribution and the singular values $\mathbf{S}$ can be interpreted as concentration parameters in the

different axes. We therefore set $\mathbf{S} = s\mathbf{I}$ to obtain the isotropic version of the distribution,

$$p\left(\mathbf{R} \mid \hat{\mathbf{R}}, s\right) = \frac{1}{n\left(s\right)} \exp\left(s \operatorname{Tr}\left[\hat{\mathbf{R}}^T \mathbf{R}\right]\right) \tag{5.3}$$

where the normalization constant only depends on s (Theorem 2.1 of (Lee, 2018)). Note that on $SO\left(2\right)$, this becomes the Von-Mises distribution as expected.

We introduce the following result, which will allow us to derive the regularization.

Proposition 5.1.1. *Let p and q be matrix Fisher distributions of $\mathbf{R}$*

$$\begin{aligned} p\left(\mathbf{R} \mid \hat{\mathbf{R}}_p, s_p\right) &= \frac{1}{n\left(s_p\right)} \exp\left(s_p \operatorname{Tr}\left[\hat{\mathbf{R}}_p^T \mathbf{R}\right]\right), \\ q\left(\mathbf{R} \mid \hat{\mathbf{R}}_q, s_q\right) &= \frac{1}{n\left(s_q\right)} \exp\left(s_q \operatorname{Tr}\left[\hat{\mathbf{R}}_q^T \mathbf{R}\right]\right). \end{aligned}$$

The cross-entropy is given by

$$\mathbb{E}_{\mathbf{R}\sim q}\left[\log p\left(\mathbf{R} \mid \hat{\mathbf{R}}_p, s_p\right)\right] = N\left(s_q\right) s_p \operatorname{Tr}\left(\hat{\mathbf{R}}_p^T \hat{\mathbf{R}}_q\right) + \log c\left(s_p\right) \tag{5.4}$$

where $N\left(s_q\right)$ only depends on s_q.

Proof. The cross-entropy is given by

$$\begin{aligned} &\mathbb{E}_{\mathbf{R}\sim q}\left[\log p\left(\mathbf{R} \mid \hat{\mathbf{R}}_p, s_p\right)\right] \\ &\qquad = \int_{SO(n)} q\left(\mathbf{R} \mid \hat{\mathbf{R}}_q, s_q\right) \log p\left(\mathbf{R} \mid \hat{\mathbf{R}}_p, s_p\right) \mathrm{d}\mathbf{R}, \end{aligned} \tag{5.5}$$

where $\mathrm{d}\mathbf{R}$ is the invariant Haar measure on $SO(n)$. Here, we assume that it is scaled such that $\int_{SO(n)} \mathrm{d}\mathbf{R} = 1$.

We obtain

$$\begin{aligned} \mathbb{E}_{\mathbf{R}\sim q}\left[\log p\left(\mathbf{R} \mid \hat{\mathbf{R}}_p, s_p\right)\right] &= \int_{SO(n)} q\left(\mathbf{R} \mid \hat{\mathbf{R}}_q, s_q\right) \left(s_p \operatorname{Tr}\left[\hat{\mathbf{R}}_p^T \mathbf{R}\right] - \log c\left(s_p\right)\right) \mathrm{d}\mathbf{R}, \\ \mathbb{E}_{\mathbf{R}\sim q}\left[\log p\left(\mathbf{R} \mid \hat{\mathbf{R}}_p, s_p\right)\right] &= s_p \operatorname{Tr}\left(\hat{\mathbf{R}}_p^T \, \mathbb{E}_{\mathbf{R}\sim q}\left[\mathbf{R}\right]\right) - \log c\left(s_p\right). \end{aligned} \tag{5.6}$$

From Theorem 2.2 and Lemma 2.2 of (Lee, 2018), we have

$$\mathbb{E}_{\mathbf{R}\sim q}\left[\mathbf{R}\right] = \frac{d\log c\left(s_q\right)}{ds_q}\hat{\mathbf{R}}_q. \tag{5.7}$$

Therefore, we find

$$\mathbb{E}_{\mathbf{R}\sim q}\left[\log p\left(\mathbf{R} \mid \hat{\mathbf{R}}_p, s_p\right)\right] = \frac{d\log c\left(s_q\right)}{ds_q} s_p \hat{\mathbf{R}}_q - \log c\left(s_p\right), \tag{5.8}$$

which completes the proof. □

Setting the location parameters of the estimated and prior distributions as $\mathbf{R}_{c(x)}$ and $\hat{\mathbf{R}}_q = \mathbf{I}$ respectively, we find that the canonicalization prior Eq. (5.1) is given by

$$\mathcal{L}_{\text{prior}} = -\lambda \operatorname{Tr}\left(\mathbf{R}_{c(x)}\right) = \frac{\lambda}{2}\left\|\mathbf{R}_{c(x)} - \mathbf{I}\right\|_F \tag{5.9}$$

where we have eliminated terms that do not depend on $\mathbf{R}_{c(x)}$ and $\lambda = N\left(s_q\right) s_p$. Following intuition, the strength of the regularization is determined by the concentrations of the distributions around their mode.

Practical Implementation For the image domain, the canonicalization network needs to output rotation matrices $\mathbf{R}_{c(x)} \in SO(2)$ that equivariantly transforms with the input image. This can be achieved by using a $E(2)$-Steerable Network (Weiler & Cesa, 2019c) that outputs two vector fields. To design the canonicalization function we can take a spatial average over both vector fields and Gram-Schmidt orthonormalize the vectors to get a 2D rotation matrix. While this sounds promising in theory, in practice we found it empirically difficult to optimize using the loss to enforce canonicalization prior. We believe this warrants further investigation and presents a potential novel research direction to explore. However, in the domain of point clouds, we discovered that combining the implementation from the previous chapter, which outputs 3D rotation matrices or elements of $SO(3)$, with the regularization loss to enforce the canonicalization prior leads to remarkably effective results. This approach is demonstrated in our model for rotation-robust point cloud classification and part segmentation, leveraging pretrained PointNet (Qi et al., 2017a) and DGCNN (Wang et al., 2019a) architectures (see Section 5.3).

Our code is available at `https://github.com/arnab39/equiadapt`

Table 5.2: Performance comparison of large pretrained models finetuned on different vision datasets. Both classification accuracy and $\mathcal{G}$-averaged classification accuracies are reported. Acc refers to the accuracy on the original test set, and $C8$-Avg Acc refers to the accuracy on the augmented test set obtained using the group $\mathcal{G} = C8$. $C8$-Aug. refers to fine-tuning the pre-trained model with rotation augmentations restricted to $C8$.

Pretrained Prediction Network →		ResNet50		ViT	
Datasets ↓	Model	Acc	$C8$-Avg Acc	Acc	$C8$-Avg Acc
CIFAR10	Vanilla	**96.97 ± 0.01**	57.77 ± 0.25	**98.13 ± 0.04**	63.59 ± 0.48
	Rot Aug	94.91 ± 0.07	90.11 ± 0.19	96.26 ± 0.15	93.67 ± 0.39
	LC	93.29 ± 0.01	92.96 ± 0.09	95.00 ± 0.01	94.80 ± 0.02
	$C8$-Aug.	95.76 ± 0.07	94.36 ± 0.09	96.36 ± 0.02	94.17 ± 0.08
	PRLC (ours)	96.19 ± 0.01	**95.31 ± 0.17**	96.14 ± 0.14	**95.08 ± 0.10**
CIFAR100	Vanilla	**85.84 ± 0.10**	44.86 ± 0.12	**87.91 ± 0.28**	55.87 ± 0.14
	Rot Aug	80.21 ± 0.09	74.12 ± 0.05	82.59 ± 0.44	78.39 ± 0.89
	LC	78.50 ± 0.15	77.52 ± 0.07	80.86 ± 0.17	80.48 ± 0.20
	$C8$-Aug.	83.00 ± 0.09	79.72 ± 0.10	83.45 ± 0.09	80.08 ± 0.38
	PRLC (ours)	83.44 ± 0.02	**82.09 ± 0.09**	84.27 ± 0.10	**83.61 ± 0.01**
STL10	Vanilla	**98.30 ± 0.01**	73.87 ± 1.43	**98.31 ± 0.09**	76.66 ± 0.93
	Rot Aug	98.08 ± 0.06	94.97 ± 0.08	97.85 ± 0.17	94.07 ± 0.11
	LC	95.30 ± 0.19	93.92 ± 0.10	95.11 ± 0.01	94.67 ± 0.02
	$C8$-Aug.	**98.31 ± 0.01**	96.31 ± 0.13	97.83 ± 0.08	94.45 ± 0.35
	PRLC (ours)	97.01 ± 0.01	**96.37 ± 0.12**	96.15 ± 0.05	**95.73 ± 0.16**

5.2 Image Experiments

In this section, we present experimental results on images to evaluate our method of achieving equivariance with minimal modifications to pretrained networks. The key benefit of our approach is showcased by demonstrating its robustness when evaluating out-of-distribution data that arise from known transformations applied to the test set. In addition to supervised learning, we present preliminary experiments in Deep Reinforcement Learning (DRL) to establish a connection with the concepts discussed in Chapter 3.

5.2.1 Classification

Experiment Setup. We use ResNet50 (He et al., 2016) and ViT-Base (Dosovitskiy et al., 2021), which are pretrained on the ImageNet-1K dataset (Deng et al., 2009) for our image experiments. These are widely used for image classification tasks, and their pretrained checkpoints are publicly available [2]

[2]https://pytorch.org/vision/stable/models/generated/torchvision.models.resnet50.html

[3]. We finetune these models on several benchmark natural image classification datasets, including CIFAR10 (Krizhevsky & Hinton, 2009), CIFAR100 (Krizhevsky & Hinton, 2009), and STL10 (Coates et al., 2011). Moreover, we incorporate four different strategies to finetune the pretrained models, namely: 1) Vanilla, 2) Rotation Augmentation, 3) Learn Canonicalization, and 4) Prior-Regularized Learned Canonicalization. For the canonicalization function, we use a $C8$-equivariant convolutional network. We jointly train the canonicalization function and fine-tune the pretrained image classification networks. Our total loss is given by $\mathcal{L}_{total} = \mathcal{L}_{fine-tune} + \beta \cdot \mathcal{L}_{prior}$, where $\mathcal{L}_{prior}$ is defined in Eq. 5.1, $\mathcal{L}_{fine-tune}$ refers to the cross-entropy loss for classification, and β is a hyperparameter which is set to 100.

The *Vanilla* model refers to fine-tuning the pretrained checkpoints using data augmentations such as horizontal flips and small angle rotations, while the *Rotation Augmentation* model covers angles ranging from -180 to $+180$ degrees. Although *Rotation Augmentation* model is not equivariant, our goal was to establish a useful baseline to measure the gain in generalization of our proposed model to the out-of-distribution test dataset resulting from rotating the images. Further, we also report results on a discretized version of *Rotation Augmentation*, mentioned as *C8-Aug.* in Table 5.2, where the fine-tuning dataset is augmented with the application of group elements in $C8$. We employ the *Learned Canonicalization* method from the previous chapter, where the canonical orientation is learned with the task signal only, which in our experiment is the classification. Finally, our proposed *Prior-Regularized Learned Canonicalization* approach adds a prior regularization loss that tries to map all images in the original training dataset to the identity group element e. We refer to the final two equivariant techniques as LC and Prior-Regularized LC.

Evaluation Protocol. In order to test the robustness of the models on rotations, we introduce $\mathcal{G}$-averaged test set that refers to an expanded dataset obtained by applying all group elements to each image, resulting in a test dataset of size $|\mathcal{G}|$ times the original test set. In this section, we consider $\mathcal{G}$ as $C8$ (cyclic group with 8 rotations), thus $|\mathcal{G}| = 8$. We report the top-1 classification accuracy achieved on the original as well as this augmented test set (referred to as $C8$-Average Accuracy) to evaluate both in distribution and out-of-distribution performance of the models.

[3]https://pytorch.org/vision/stable/models/generated/torchvision.models.vit_b_16.html

Results. We report the results of finetuning ResNet50 and ViT on CIFAR10, CIFAR100, and STL10 with various strategies in Table 5.2. As anticipated, we found that large pretrained networks for images are not robust to rotation transformations, as indicated by the large drop in performance from the accuracy to its $C8$-averaged counterpart for both ResNet50 and ViT. Nevertheless, we observe that ViT is more robust to rotations compared to ResNet50, which has also been observed by (Gruver et al., 2022). We notice that augmenting with a full range of rotation angles during training improves the $C8$-Average Accuracy as demonstrated by our *Rotation Augmentation* baseline. However, it hurts the accuracy of the prediction network in the original test set and does not guarantee equivariance. Augmenting with necessary rotations in *C8-Augmentation* does not ensure equivariance to $C8$ but retains performance on the original test set and reduces the gap between original and C8-averaged accuracies.

LC guarantees equivariance, which can be seen from the minor difference between the accuracies of the original and augmented test sets. Nevertheless, in every dataset, we can observe a significant drop in accuracy for the original test set. We extensively discussed this issue in Section 5.1.1. However, with *our Prior-Regularized LC* method, we can reduce the gap between the accuracy on the original test set while still being equivariant to rotations. This demonstrates that this prior regularization on LC is a promising direction to improve the performance of large-pretrained models while guaranteeing robustness to out-of-distribution samples resulting from transformations like rotation.

Ideally, the accuracy of the original test set should be nearly identical for both the Vanilla setup and our Prior-Regularized LC method. However, we observed a slight difference between their corresponding accuracies. This disparity arises from the fact that the canonicalization model is unable to map all data points (images) perfectly to the identity element e, supported by our observations that the regularization loss for prior matching does not diminish to zero. We hypothesize that this stems from the intentional limitation in the expressivity of the canonicalization function, which is done on purpose in the previous chapter to avoid adding excessive computational overhead to the overall architecture. Finally, we note that due to rotation artifacts, a small difference between $C8$-Average Accuracy and Accuracy on the original test set is unavoidable.

5.2.2 Instance Segmentation

Experiment Setup. We use MaskRCNN (He et al., 2017) and Segment Anything Model (SAM) (Kirillov et al., 2023), which are pretrained on Microsoft COCO (Lin et al., 2014) and SA-1B (Kirillov et al., 2023) datasets respectively. MaskRCNN is widely used for instance segmentation tasks, while SAM is a recently proposed foundational model that can leverage prompts (bounding boxes and key points) for instance segmentation, and their pretrained checkpoints are publicly available [4] [5]. In our experiments, we evaluate these models on COCO 2017 dataset, i.e., report zero-shot performance on validation (val) set and use ground truth bounding boxes as prompts for SAM. For the canonicalization function, we use a $C4$-equivariant WideResNet architecture. The details of the architecture are available in (Mondal et al., 2024). As MaskRCNN and SAM are already trained for the instance segmentation task, we only train the canonicalization function using the prior loss $\mathcal{L}_{prior}$ in Eq. 5.1 to make them equivariant to $C4$ group.

Evaluation Protocol. To test the generalization capabilities of these models, we introduce and utilize the $C4$-averaged val set ($C4$ refers to the cyclic group with 4 rotations for reducing the complexities with transforming ground truth boxes and masks) along with the original val set. We report the mask-mAP score for mask prediction on both these datasets, denoted respectively as mAP and $C4$-Avg mAP in Table 5.3.

Results. Owing to its pre-training on larger datasets and the use of bounding boxes as prompts, SAM outperforms MaskRCNN in both mAP and $C4$-Avg mAP scores. The difference between mAP and $C4$-Avg mAP scores demonstrates that SAM is more robust than MaskRCNN in the case of these transformations.

However, we observe that there is a difference between these two reported numbers for both models. With our prior regularized LC framework, we can achieve equivariance with any large pretrained model to the desired group (here $C4$) while retaining the performance on the original val set. Further, we analyze the relationship between the expressivity of the canonicalization function and the downstream effect on mAP values on the original val set in Table 5.3. We compare a $C4$-equivariant convolutional network (G-CNN) with a $C4$-equivariant WideResNet (G-WRN) architecture. We observe that

[4] https://pytorch.org/vision/main/models/generated/torchvision.models.detection.maskrcnn
[5] https://github.com/facebookresearch/segment-anything

although equivariance is guaranteed, a less expressive canonicalization function leads to decreased performance in the original val set due to its inability to map complex images to identity.

Table 5.3: Zero-shot performance comparison of large pretrained segmentation models with and without trained canonicalization functions on COCO 2017 dataset (Lin et al., 2014). Along with the number of parameters in *canonicalization* and *prediction network*, we report mAP and $C4$-averaged mAP values. † indicates G-CNN and ‡ indicates a more expressive G-WRN for canonicalization.

Pretrained Large Segmentation Network →		MaskRCNN (46.4 M)		SAM (641 M)	
Datasets ↓	Model	mAP	$C4$-Avg mAP	mAP	$C4$-Avg mAP
COCO	Zero-shot (0 M)	45.57	27.67	62.34	58.78
	Prior-Regularized LC† (0.2 M)	35.77	35.77	59.28	59.28
	Prior-Regularized LC‡ (1.9 M)	44.51	**44.50**	62.13	**62.13**

5.2.3 Reinforcement Learning

Experiment Setup. We use a similar setup to that described in Section 3.4. Along with the Vanilla Double Deep Q-Network (DDQN) and its equivariant version, we implement a Prior Regularized Learned Canonicalization version based on the pretrained Vanilla DDQN model. To train the canonicalizer using the prior regularization loss, we generate data from 1000 episodes using the pretrained Vanilla DDQN policy.

Evaluation Protocol. For our evaluation protocol, we employ the same methodology outlined in Section 3.4. The results for different rotations of the screen under the cyclic group $C4$ are reported in Table 5.4.

Transformation	Vanilla DDQN	Equivariant DDQN	PRLC DDQN
e	129 ± 2.3	125 ± 4.5	126 ± 3.5
r	48.9 ± 2	99 ± 4.7	109 ± 6.3
r^2	53 ± 3.5	104 ± 3.4	115 ± 5.1
r^3	51 ± 1.9	98 ± 3.9	110 ± 4.6

Table 5.4: Average reward over 200 episodes of Pacman for 5 seeds reported with a confidence level of 95% for different environment transformations. e is the original screen.

Results. Our findings demonstrate that the pretrained Vanilla Double Deep Q-Network (DDQN) agent, when equipped with our prior regularized learned canonicalization technique, successfully generalizes to various transformations of the input screen. While these experiments are preliminary, they reveal a promising avenue for enhancing the generalization capabilities of pretrained Deep Reinforcement Learning (DRL) agents. Specifically, this approach opens up the possibility of enabling any pretrained DRL agent to generalize across transformations in environments with known group actions.

5.3 Point Cloud Experiments

Table 5.5: Classification accuracy of different pointcloud models on the ModelNet40 dataset (Wu et al., 2015) in different train/test scenarios and ShapeNet (Chang et al., 2015) Part segmentation mean IoUs over 16 categories in different train/test scenarios. x/y here stands for training with x augmentation and testing with y augmentation. z here stands for aligned data augmented by random rotations around the vertical/z axis and SO(3) indicates data augmented by random 3D rotations.

Task →	Classification			Part Segmentation	
Dataset →	ModelNet40			ShapeNet	
Method ↓	z/z	z/SO(3)	SO(3)/SO(3)	z/SO(3)	SO(3)/SO(3)
PointNet	85.9	19.6	74.7	38.0	62.3
DGCNN	**90.3**	33.8	88.6	49.3	78.6
VN-PointNet	77.5	77.5	77.2	72.4	72.8
VN-DGCNN	89.5	89.5	90.2	81.4	81.4
LC-PointNet	79.9 ± 1.3	79.6 ± 1.3	79.7 ± 1.3	73.5 ± 0.8	73.6 ± 1.1
LC-DGCNN	88.7 ± 1.8	88.8 ± 1.9	90.0 ± 1.1	78.4 ± 1.0	78.5 ± 0.9
Ours (with pretrained PointNet and DGCNN for each task)					
	no-aug/z	no-aug/SO(3)		no-aug/SO(3)	
PRLC-PointNet	84.1 ± 1.1	84.3 ± 1.2		82.6 ± 1.3	
PRLC-DGCNN	**90.2 ± 1.4**	**90.2 ± 1.3**		**84.3 ± 0.8**	

Datasets. For our experiments involving point clouds, we utilized the ModelNet40 (Wu et al., 2015) and ShapeNet (Chang et al., 2015) datasets. The ModelNet40 dataset comprises 40 classes of 3D models, with a total of 12,311 models. Among these, 9,843 models were allocated for training, while the remaining models were reserved for testing in the classification task. In the case of part segmentation, we employed the ShapeNet-part subset, which

encompasses 16 object categories and over 30,000 models. We only train the canonicalization function using the prior loss $\mathcal{L}_{prior}$ in Eq. 5.9.

Evaluation protocol. To ensure consistency and facilitate comparisons, we followed the established conventions set by (Esteves et al., 2018a) and adopted by (Deng et al., 2021) for the train/test rotation setup in the classification and segmentation tasks. The notation x/y indicates that transformation x is applied during training, while transformation y is applied during testing. Typically, three settings are employed: z/z, $z/SO(3)$, and $SO(3)/SO(3)$. Here, z denotes data augmentation with rotations around the z-axis during training, while $SO(3)$ represents arbitrary rotations. However, since we regularize the output of the canonicalization with the identity transformation, we trained our canonicalization function and fine-tuned our pretrained model without any rotation augmentation. During inference, we tested on both z and $SO(3)$ augmented test datasets.

Results. We present our results on Table 5.5. Notably, our method showcased superior performance in terms of robustness, outperforming existing methods for point cloud tasks. Specifically, the inclusion of the prior loss has led to a significant improvement in PointNet's (Qi et al., 2017a) performance compared to DGCNN (Wang et al., 2019a). This observation aligns with our analysis in Section 5.1.1, where we highlight that training the prediction network with large rotations can hinder its performance and serve as a bottleneck for equivariance within the learnt canonicalization framework. The empirical evidence, particularly in the $SO(3)/SO(3)$ results of vanilla PointNet and DGCNN, where we notice a more pronounced drop in PointNet's performance, supports this and strengthens our findings.

5.4 Discussion

Out-of-distribution generalization is desirable in machine learning, but achieving it has been challenging for state-of-the-art deep models, where an important source of shift in distribution is due to application-specific transformations of the data – from the change of colour palette, magnification and rotation in images to change of volume or pitch in sound. A mechanism for making pretrained models robust to such changes can have a significant impact on their adaptation to different domains. This paper takes the initial steps toward this goal by removing barriers to making the model equivariant through a lightweight canonicalization process. The proposed solution ensures that the pretrained model receives in-distribution data, while canonicalization ensures

equivariance and, therefore, adaptability to a family of out-of-distribution samples. Our extensive experimental results using different pretrained models, datasets, and modalities gives insight into this subtle issue and demonstrate the viability of our proposed solution.

Limitations and Possible Extensions An important limitation of our approach is the dependency on an equivariant canonicalization network, which imposes restrictions on the range of transformations to which we can adapt. The optimization approach discussed in the previous chapter offers more flexibility, as it replaces the equivariant network with the outcome of an optimization process. However, this approach can raise efficiency concerns for continuous groups, which merit further investigation. Another limitation of the current exposition is the limited group of transformations explored in experiments. In future, a comprehensive evaluation of this approach to other groups of transformation can be a promising direction. Furthermore, the experiments on pretrained DRL agents are preliminary, and a thorough study of this technique for Reinforcement Learning is an interesting avenue of future research.

The optimization challenges related to prior regularization for $E(2)$-Steerable Networks (Weiler & Cesa, 2019b) also present an interesting direction for future work. Addressing these challenges would allow us to incorporate continuous rotations into our image framework, thereby expanding its capabilities. Moreover, this observation emphasizes the necessity to delve into the effect of expressivity of the canonicalization function, as it plays a crucial role in determining the overall performance of our model.

Another promising future direction can be the exploration of more flexible adaptation through canonicalization using examples from a target domain. Such examples from a target domain can, in principle, replace the prior knowledge of the transformation group assumed in equivariant networks, thereby bridging the gap between equivariant deep learning and domain adaptation.

Part III

Learning Structured Representations

6

Equivariant Representations using Loss Constraints on Lie Groups

In contrast to the chapters in the previous part of the thesis, this chapter and the subsequent chapters develop a novel perspective on learning representations that are structured using the symmetries and transformations in the data. This chapter, which is mostly taken from (Mondal et al., 2022), shows how to parameterize the latent embeddings of states and actions to make the representations equivariant to continuous transformations of the environment resulting from an agent's action using loss constraints.

In particular, we integrate equivariance under the agent's action and equivariance under the symmetries of the environment into a single latent variable model that is equivariant to an *a priori unknown* group of non-linear transformations of state-action pairs; see Figs. 6.1 and 6.2. In contrast to the traditional approach of using symmetric Markov Decision Processes (MDPs), we model the larger group of state-action symmetries (separate from reward symmetries). We evaluate this approach, which we call Equivariant representations for RL (EqR), on the 26 games in the Atari 100K benchmark

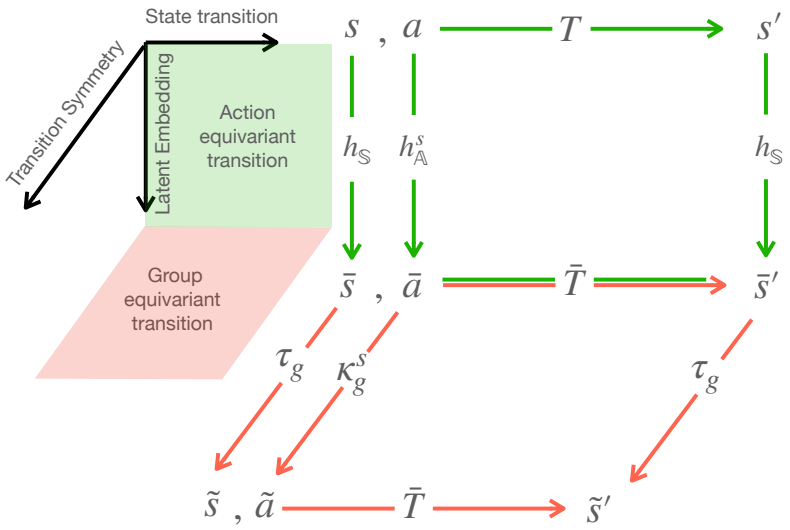


Figure 6.1: This figure demonstrates the relationship between two types of equivariance in latent variable modeling for an MDP with a symmetric transition function. Green arrows (vertical plane) identify a diagram for transition models in an MDP homomorphism. A model $\bar{T}$ and state embedding function $h_{\mathbb{S}}$ that are equivariant under an agent's action makes this diagram commute. Red arrows (horizontal plane) identify the commutativity diagram for a symmetric transition function of an MDP in the latent space. Here the state-action embedding $\langle \tilde{s}, \tilde{a} \rangle$ is produced through the symmetry transformation of another state-action embedding $\langle \bar{s}, \bar{a} \rangle$.

(Kaiser et al., 2019b). Here we outperform other comparable methods using reliable evaluation metrics (Agarwal et al., 2021). Our approach, however, is not restricted to this domain. It is applicable in any setting where the transformations that an agent undergoes can be expressed using matrix Lie groups, including autonomous driving, navigation, and robotics.

6.1 Desiderata for Symmetry-Based Representation in RL

Separating Transition and Reward Symmetries One important choice is between using the symmetry group of the MDP ($\mathcal{G}_{\mathcal{M}}$) versus the symmetry group of state-transitions ($\mathcal{G}_T$), where $\mathcal{G}_T$ is the group of transformations of state-action pairs that lead to equivariant deterministic transitions, as given by Equation 2.9. The former is a subgroup of the latter $\mathcal{G}_{\mathcal{M}} \leq \mathcal{G}_T$,

i.e., the symmetries of a transition model contain the symmetries of the MDP. In fact, it is easy to see that $\mathcal{G}_{\mathcal{M}} = \mathcal{G}_T \cap \mathcal{G}_R$, where $\mathcal{G}_R$ is the group of transformations of state-action pairs that preserve the one step-reward and only satisfy Equation 2.8.

We observe that working with a larger symmetry group $\mathcal{G}_T$ has two benefits. First, it creates a stronger inductive bias for the model because many real-world settings can involve a range of symmetries in transitions that are not present in the reward. For example, an agent's transition function in a 2D space may be equivariant to the Euclidean group, while the reward may not be invariant to the same group (*e.g.*, the reward for arriving at a particular location could break this symmetry). Second, the separate modeling of transition symmetries facilitates transfer to new tasks where the reward is changing.

Equivariance in Model-Free and Model-Based RL If the objective is to carry out model-free RL, Eq. (2.11) motivates the need to learn action-value functions or the policies that are *invariant* to symmetries of the MDP ($\mathcal{G}_{\mathcal{M}}$). For a deterministic policy, the invariance of Eq. (2.11) becomes an equivariance constraint: $g \cdot \pi(s) = \pi(g \cdot s)$. Since this essentially leads to model minimization, van der Pol et al. (2020b); Mondal et al. (2020) use this idea to improve sample efficiency when the group's actions in the agent's action space are known permutations, which is explored in Chapter 3. However, if our objective is to learn only a symmetry-based model of the environment (*i.e.*, transition and reward functions), Eq. (2.9) suggests that we need to learn a $\mathcal{G}_T$-*equivariant* transition function.

Symmetries in a Latent Transition Model While it is possible to learn the state transition model in the observation space that is equivariant to the agent's action, for high-dimensional inputs this could be quite challenging since the model has to learn details of the environment that are irrelevant to the RL agent. Using state and action embeddings enables learning of the transition model in the latent space. Indeed the constraint on the model and the embedding is that of the MDP homomorphism (Section 2.4.2). Working in the latent space has an additional benefit when it comes to symmetries: *we can assume that the $\mathcal{G}$ action on the latent state-action pairs is linear through $\rho(\mathcal{G})$ despite having non-linear transformations in the observation space.*

From the fact that symmetries of states $\mathcal{G}_{\mathcal{S}} \leq \mathcal{G}$ is a subgroup of the state-action or transition symmetry, it follows that $\rho_g \in \rho(\mathcal{G})$ can be divided into two parts: 1) $\tau_g \in \tau(\mathcal{G}_{\mathcal{S}})$ the group representation acting on the state

embedding, and; 2) $\kappa_g^s \in \kappa(\mathcal{G})$, the group representation for state-dependent action embedding.[1]

At this point, we can combine the requirement for an MDP homomorphism in Eq. (2.6), with that of the $\mathcal{G}$-equivariant transition model, Eq. (2.9) of a symmetric MDP. The result is the following two constraints in our symmetric latent variable model (see Fig. 6.1): $\forall s, a \in \mathcal{S} \times \mathcal{A}$ and $g \in \mathcal{G}$

$$\bar{T}(h_{\mathcal{S}}(s), h_{\mathcal{A}}(s,a)) = h_{\mathcal{S}}(T(s,a)) \tag{6.1}$$

$$\tau_g \bar{T}\left(h_{\mathcal{S}}(s), h_{\mathcal{A}}(s,a)\right) = \bar{T}\left(\tau_g h_{\mathcal{S}}(s), \kappa_g^s h_{\mathcal{A}}(s,a)\right) \tag{6.2}$$

Matrix Embedding of States and Actions We now consider a design choice which can significantly simplify the constraints discussed above, though strictly speaking it is not required. We propose to use group representations for our state, and state action embeddings $h_{\mathcal{S}} : \mathcal{S} \to \tau(\mathcal{G}_{\mathcal{S}})$ and $h_{\mathcal{A}} : \mathcal{S} \times \mathcal{A} \to \kappa(\mathcal{G})$ – that is we use matrices to represent both states and actions. This choice assumes that a $\mathcal{G}$ action on state and state-action pairs is *transitive*, so that each state and state-action pair can be mapped to a group member. To emphasize this in our notation, we use $\kappa(s)$ instead of $h_{\mathcal{S}}(s)$ and similarly use $\tau(s,a)$ instead of $h_{\mathcal{A}}(s,a)$ for state, and state-dependent action embedding respectively. This choice of embedding has several benefits: First, the learned embeddings are automatically equivariant to symmetry transformations of the state, and state-actions:

$$\begin{aligned} &\tau(g \cdot s) = \tau_g \tau(s) \quad \text{and} \quad \kappa(g \cdot \langle s, a \rangle) = \kappa_g^s \kappa(s,a), \\ &\forall s, a \in \mathcal{S} \times \mathcal{A}, g \in \mathcal{G}. \end{aligned} \tag{6.3}$$

This means that the symmetries of the state-action pairs are preserved and now take a linear form in the latent space. While the embeddings are automatically equivariant, they may be equivariant to irrelevant non-linear transformations of the input. The world modeling constraint (Eq. (6.1)) ensures the relevance of the non-linear transformations that are captured by the group equivariant embeddings above. Moreover, this embedding enables matrix multiplication for the transition model

$$\bar{T}(\tau(s), \kappa(s,a)) = \kappa(s,a)\tau(s), \tag{6.4}$$

[1]This is because $\rho(\mathcal{G})$ can be seen as a representation that is *induced* by the representation $\tau(\mathcal{G}_{\mathcal{S}})$ of its subgroup: $\rho = \mathrm{Ind}_{\mathcal{G}_{\mathcal{S}}}^{\mathcal{G}} \tau$.

which simply transforms the state-embedding $\tau(s)$ through the linear group action of state-dependent action encoding $\kappa(s, a)$. Using this transition model, the action equivariance constraint of Eq. (6.1), and $\mathcal{G}$-equivariance constraint of Eq. (6.2), simplify to:

$$\tau(s') = \kappa(s, a)\tau(s) \tag{6.5}$$

$$\tau_g \kappa(s, a)\tau(s) = \kappa_g^s \kappa(s, a)\tau_g \tau(s) \tag{6.6}$$

for any state transition triplet $\{s, a, s'\}$. In practice, our model seeks to satisfy these two constraints via the direct minimization of appropriate loss functions, as will be discussed in Section 6.2.

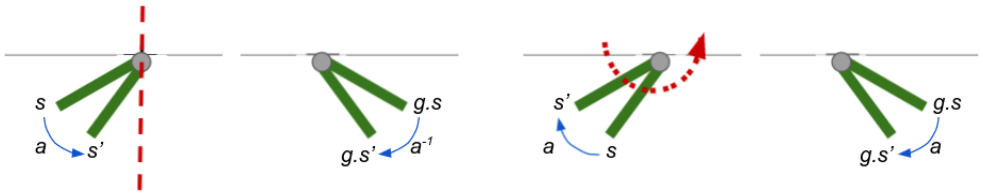


Figure 6.2: An illustration of typical symmetries in a pendulum, and the corresponding transformations of the state and action for a group equivariant transition model: (a) shows how reflection of the agent's state results in a permutation of the action, denoted by a^{-1}. (b) shows how the rotation of the agent's state results in invariance of the action in the absence of gravity. The state transitions can be modeled as group actions (2D rotations in this example), which our symmetry transformation-based transition model can capture. Note that rotational symmetry can hold even when gravity is present. In this case, symmetry transformations include rotations (and reflections) that preserve the Hamiltonian. Such non-linear energy-preserving transformations of state-actions in the pixel space can become linear in the embedding space.

Decomposition of the Latent Space The decomposition $\mathcal{G} = \mathcal{G}_1 \times \ldots \times \mathcal{G}_K$ into a direct product of subgroups can disentangle the factors of variation in the dataset (Higgins et al., 2018).[2] This gives us a way to represent the latent embedding space as a direct product of K factors, where each factor varies independently by actions of a subgroup of $\mathcal{G}$. Intuitively such a symmetry-based disentanglement provides an effective inductive bias particularly when

[2]As noted by Caselles-Dupré et al. (2019), simply having a product structure in the latent space does not guarantee disentanglement, and further constraints are required. In this work, we do not impose any additional constraints for disentanglement.

there is modularity so that temporally coherent changes in the environment are due to the change of a (sparse) subset of factors. In our case, this constraint takes the form of block-diagonal matrices for state and action embeddings. More precisely, we have a direct sum for state representation and the state-dependent action representation:

$$\tau(s) = \bigoplus_k \tau_k(s) \quad \text{and} \quad \kappa(s,a) = \bigoplus_k \kappa_k(s,a)$$

where $k \in \{1, \ldots, K\}$ and $g = (g_1, \ldots, g_K)$. Accordingly, the representation of the symmetry group $\mathcal{G}$ acting on the state embedding and the state-dependent action embedding is decomposed as $\tau_g = \bigoplus_k \tau_{g_k}$ and $\kappa^s_g = \bigoplus_k \kappa^s_{g_k}$. Combining this block structure with the Lie parameterization of Section 2.2 we obtain

$$\begin{aligned} \tau_\theta(s) &= \bigoplus_k \exp\left(\sum_i \beta_{i,k,\theta}(s)\mathbf{E}^{(i)}\right) \quad \text{and} \\ \kappa_\phi(\tau_\theta(s), a) &= \bigoplus_k \exp\left(\sum_i \alpha_{i,k,\phi}(\tau_\theta(s), a)\mathbf{E}^{(i)}\right) \end{aligned} \tag{6.7}$$

where we use any standard neural network to implement the α_ϕ and β_θ functions that represent coefficients for the bases of the Lie algebra[3]. As we can backpropagate through this function, the network parameters θ, ϕ can be learned end to end. We refer the readers to Section 6.1.1 for more detail. The choice of the subgroup depends on the symmetries of the RL environment, and this choice only affects the set of bases $\{\mathbf{E}^{(i)}\}_i$ in Eq. (6.7). For example, in Atari games, the screen often has multiple objects undergoing 2-D translations and rotations, and one can use blocks of the 2-D Special Euclidean ($SE(2)$) group, that comprise translation and rotations of Euclidean space. For more realistic 3D environments, such as those of interest in robotics, self-driving cars and third-person games, one can use $SE(3)$, which is the group of 3-D translations and rotations. Also, in theory, we only need to specify a group that "contains" the group of interest as a subgroup. For example, if our state-actions only have 90° rotational symmetry, we may use a more general group for the representation (*e.g.*, $SE(2)$). The embedding function can define a homomorphism into the relevant subgroup. For more realistic

[3]We denote both the neural networks which map to group representations and the network parameters by lowercase Greek letters.

3D environments, such as those of interest in robotics, self-driving cars and third-person games, one can use $SE(3)$, which is the group of 3-D translations and rotations. This should capture both the transformations of the objects in the environment and changes in viewpoint due to the agent's actions.

6.1.1 Implementing parameterization

We consider three types of subgroups: $GL(n)$ - the set of all invertible linear transformations, $SE(n)$ - the set of all rotations and translations and $T(n)$ - the set of all translations. We provide a general method to parameterize each of these, based on the type of group.

$GL(n)$ As the matrix representation of $GL(n)$ is the set of invertible matrices which has a measure of 1 it is easy to parameterize it. We just generate n^2 parameters using a network corresponding to each element of the matrix. This gives an element from $GL(n)$.

$T(n)$ As $T(n)$ just denotes translation in a n-dimensional space with group action being addition, implementing it is straightforward. We generate n parameters using a neural network and instead of using matrix multiplication use addition for the group action. Note that we can also use a matrix representation for $T(n)$ but it is unnecessary and inefficient.

$SE(n)$ Unlike $GL(n)$ and $T(n)$, parameterizing $SE(n)$ is a bit tricky because it involves parameterizing $SO(n)$. We use a homogeneous co-ordinate based representation of $SE(n) = \left\{ \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}, R \in SO(n) \text{ and } t \in T(n) \right\}$. So we need n parameters for the t and another $D = \frac{n(n-1)}{2}$ parameters for $SO(n)$ from the neural network. As explained in Section 6.1, we can use a Lie parameterization to get the elements of $SO(n)$ by $R = \exp\left(\sum_{d=1}^{D} \beta_d \mathbf{E}^{(d)}\right)$ where $\mathbf{E}^{(d)}$ denote D bases of the space of skew symmetric matrices and the β_is are the parameters of the neural network. For example, in the case of $SO(2)$ we can use the basis $\mathbf{E}^{(1)} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$. Similarly, we can extend this to $SO(n)$ by using a basis given by D $n \times n$ matrices $\mathbf{E}^{(ij)}$ $\forall \{1 \leq i < j \leq n\}$ whose only non-zero elements are $\mathbf{E}^{(ij)}_{i,j} = -1$ and $\mathbf{E}^{(ij)}_{j,i} = 1$.

Although Lie parameterization gives us a general recipe to output a representation of simple connected Lie groups like $SO(n)$, in our implementation we use Euler parameterization because it runs faster in Pytorch. We provide the code for both. Following (Quessard et al., 2020), we parameterize each

rotation matrix in $SO(n)$ using the product of rotations on D orthogonal planes in $\mathbb{R}^n$: $R = \prod_{i=1}^{n} \prod_{1 \le i < j \le n} R^{ij}$. Here $R^{ij} \in \mathbb{R}^{n \times n}$ is the rotation matrix in the $i-j$ plane, and its non-zero elements besides the diagonal are the four values on the i, j rows and columns, which comprise the 2D rotation matrix that is $R^{ij}_{i,j} = \begin{bmatrix} \cos(\theta_{i,j}) & \sin(\theta_{i,j}) \\ -\sin(\theta_{i,j}) & \cos(\theta_{i,j}) \end{bmatrix}$. We have D parameters $\theta_{i,j}$ which we can obtain from a neural network.

The parameters in all the parameterization techniques mentioned here can be back-propagated. We summarize the number of parameters required from a neural network output, representation type and the associated group actions of different subgroups in Table 6.1.

Table 6.1: Subgroup Properties

Subgroup	#Parameters	Representation type	Group action
$GL(n)$	n^2	$\text{Matrix}_{n \times n}$	Matrix multiplication
$SE(n)$	$\frac{n(n-1)}{2} + n$	$\text{Matrix}_{(n+1) \times (n+1)}$	Matrix multiplication
$SO(n)$	$\frac{n(n-1)}{2}$	$\text{Matrix}_{n \times n}$	Matrix multiplication
$T(n)$	n	Vector_n	Addition

6.2 Symmetry Enforcing Loss Functions

We consider a standard RL setup where the agent interacts with its environments in episodes and we have access to $(\{s_t, a_t, r_t, s_{t+1}\})_{t=1,..,T}$ where s_t is the state, a_t is the action taken by the agent, r_t is the reward received and s_{t+1} is the observed next state at timestep t. Below we describe three loss functions that encode the equivariance/invariance constraints of Eqs. (2.5), (6.5) and (6.6).

Action Equivariant Transition Loss - Eq. (6.5) Given triplets $\langle s_t, a_t, s_{t+1} \rangle$ from our dataset we simply apply a loss function ℓ such as a square loss[4] that penalizes the difference between two arguments:

$$L_{AET}(\theta, \phi) = \ell \left(\tau_\theta(s_{t+1}), \kappa_\phi(s_t, a_t) \tau_\theta(s_t) \right). \tag{6.8}$$

[4]In practice we use the normalized square loss $\ell(\mathbf{Y}, \mathbf{Y}') = \left\| \frac{\mathbf{Y}}{\|\mathbf{Y}\|_2} - \frac{\mathbf{Y}'}{\|\mathbf{Y}'\|_2} \right\|_2^2$

The choice of the embedding space and the latent transition function ensures that state embeddings are transformed by linear group action of the action embeddings. Minimization of L_{AET} encourages these symmetry transformations to capture state transitions resulting from the agent's action.

Group Equivariant Transition Loss - Eq. (6.6) For this, we need a $s_{t'}$ in addition to $\langle s_t, a_t \rangle$, where t' can be any state (at a different time step in the same or in a different episode.) We find the group transformation that maps s to s' in the latent space using $\tau_g = \tau_\theta(s_{t'})\tau_\theta(s_t)^{-1}$. Using this we can rewrite Eq. (6.6) as

$$\underbrace{\tau_\theta(s_{t'})\tau_\theta(s_t)^{-1}}_{\tau_g} \kappa_\phi(s_t, a_t)\tau_\theta(s_t) = \kappa_g^s \kappa_\phi(s_t, a_t) \underbrace{\tau_\theta(s_{t'})}_{\tau_g \tau_\theta(s_t)}. \tag{6.9}$$

Since the state-dependent action encoding $\kappa_\phi(s_t, a_t)$ for the pair $\langle s_t, a_t \rangle$ is also produced by a neural network, the only missing part in the equation above is κ_g^s, the state-dependent action transformation. We use a neural network $\rho_\omega : \tau_g \mapsto \kappa_g^s$ to infer it from state transformation τ_g.

Example 1. To get an intuition for what this network is doing consider the example of a pendulum without gravity, with rotation and reflection symmetry $O(2)$ as shown in Fig. 6.2, where inputs to the networks (s) are image sequences and the (ideal) embeddings $\tau_\theta(s), \kappa_\phi(s, a)$ are the angle plus angular velocity and torque respectively. If we rotate the pendulum using a rotation matrix τ_g, we expect the state-dependent action embedding to remain the same since the effect of torque remains similar after rotation. However, if we transform the pendulum by reflection around the vertical axis, we expect that the effect of torque will be negated. ρ_ω parameterizes this dependence.

A loss function ℓ could then measure the difference between the left and right-hand sides of the equation above

$$\begin{aligned} L_{GET}(\theta, \phi, \omega) = &\ell\big(\tau_\theta(s_{t'})\tau_\theta(s_t)^{-1}\kappa_\phi(s_t, a_t)\tau_\theta(s_t), \\ &\rho_\omega(\tau_\theta(s_{t'})\tau_\theta(s_t)^{-1})\kappa_\phi(s_t, a_t)\tau_\theta(s_{t'})\big). \end{aligned} \tag{6.10}$$

Action Invariant Reward Loss - Eq. (2.5) While L_{AET} and L_{GET} enforce the equivariance of the latent transition model to an agent's action and the symmetry group, they do not encode information about the reward in the state representations. In order for the latent model to be homomorphic to the underlying MDP of the environment we match the reward at every state

embedding using a reward predictor network $r_\psi : \tau_\theta(s) \mapsto \mathbb{R}$. We measure the difference between the predicted reward and the actual reward at time step $t+1$:

$$L_R(\psi, \theta, \phi) = \left(r_\psi(\kappa_\phi(\tau_\theta(s_t), a_t)\tau_\theta(s_t)) - r_{t+1}\right)^2. \tag{6.11}$$

Example 2 (Sliding Ball). Here, we visualize the matrix embedding produced using our loss functions on a simple toy example. We design a sliding ball environment where a ball moves on the screen vertically in a closed loop with two actions: up and down. We parameterize the latent representation to be a single block of $SO(2)$, that is a rotation matrix $\tau_\theta(s) = \begin{bmatrix} \cos(\beta) & \sin(\beta) \\ -\sin(\beta) & \cos(\beta) \end{bmatrix}$, where β is the output of the image encoder network. Since in this example, the action encoding is not state-dependent, we obtain it by feeding the action to the action encoder network $\kappa(a) = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix}$ where β is the output of the network. As the action is also invariant to the group transformation $SO(2)$, Eq. (6.10) becomes unnecessary, and we can only use Eq. (6.8). To this end, we randomly sample actions (up and down) and generate trajectories to train the encoder using Eq. (6.8). Fig. 6.3 visualizes the learned embedding. For visualization, we transform a 2D unit vector ($[1, 0]$ using the matrix embedding of the state. As the ball moves in a 1D loop in the environment, the encoder learns to map the transformation to the correct rotation matrix in $SO(2)$.

6.3 Application to Model-free RL

While the framework discussed so far is ideal for model-based RL, here we confine our experiments to a model-free setting. Following the success of transition models for representation learning in model-free RL (Gelada et al., 2019; Schwarzer et al., 2021), we add the losses discussed above to the Temporal Difference (TD) error in Deep Q-learning. In practice, we need to make three modifications to our model/loss. These modifications are from the self-supervised representation learning literature (Grill et al., 2020) and were introduced in the RL setup in (Schwarzer et al., 2021). For ablation studies on these additional components, we refer the reader to (Schwarzer et al., 2021).

Target Network A trivial solution to both equivariance enforcing losses of Section 6.2 is to encode all states and actions using an identity matrix.

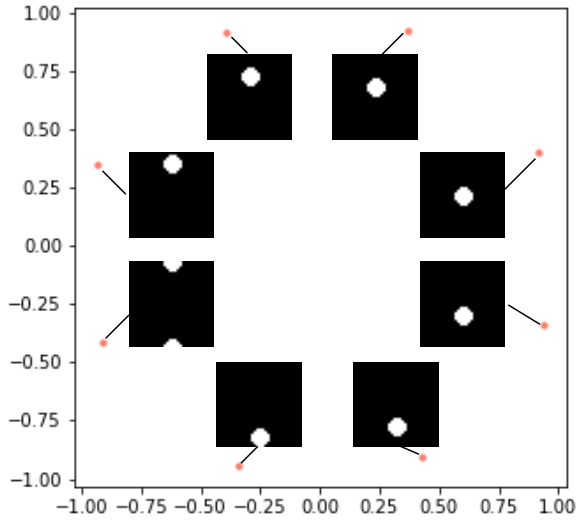


Figure 6.3: (a) Latent visualization of a sliding ball environment. The ball moves up or down in one dimension as dictated by the action. We show that our latent parameterization combined with L_{AET} learns the $SO(2)$ manifold of the ball's transition. To obtain the visualization, we start with a 2D unit vector ($[1, 0]$ here) and transform it using the representation matrix obtained from the trained encoder by feeding the image observations. Images of eight uniformly separated positions of the ball are mapped to the red points, which denote the transformed unit vector.

This problem in different contexts is known as the problem of collapse in representation learning. While using the reward signal helps in avoiding the collapse, it is often not sufficient in sparse reward settings. Following Schwarzer et al. (2021), we use a target network to encode state s_{t+1} and $s_{t'}$ in Eq. (6.10) in which the network parameters do not receive a gradient and are copied from the online network. We explicitly drop the subscripts to differentiate the target from the online network in this section (*e.g.*, $\tau_\theta \to \tau$).

Projection Head for Transition Losses The strict enforcement of symmetry constraints by our model can be overly restrictive when the environment has non-symmetric components, or when our transition model is too simplistic. For this reason, following previous work, we enforce the losses on a learnable *projection* of the state embedding. That is, before application of the loss ℓ in Eqs. (6.8) and (6.10) we pass the embedding through a projection head.

M-step prediction Following the success of (Schwarzer et al., 2021) in long-term state embedding predictions, we predict state embeddings and rewards for M-steps.

6.3.1 Putting it All Together

Considering M consecutive state-actions $\{s_{t:t+M}, a_{t:t+M}\}$ and $\hat{x}_t = x_t = \tau_\theta(s_t)$, we predict the state embeddings and the rewards of the next M steps:

$$\begin{aligned} &\hat{x}_{t+m} = \kappa_\phi(\hat{x}_{t+m-1}, a_{t+m-1})\hat{x}_{t+m-1} \quad \text{and} \\ &\hat{r}_{t+k} = r_\psi(\hat{x}_{t+k}) \quad \forall m \in \{1, \dots, M\} \end{aligned}$$

Here we are using $\hat{x}$ for M-step model prediction of the embedding to distinguish this from the latent embedding x, and the embedding produced by the target network $\bar{x} = \tau(s_{t+m})$. This also applies to the M-step predicted reward $\hat{r}$ and observed reward r. We then project these embeddings using a projection head p_ζ to produce $\hat{z}_{t+m} = p_\zeta(\hat{x}_{t+m})$ and $\bar{z}_{t+m} = p(\bar{x}_{t+m})$. Using this notation, our final expressions for L_{AET} and L_R are:

$$L_{AET} = \sum_{m=1}^{M} \left\| \frac{\hat{z}_{t+m}}{\|\hat{z}_{t+m}\|_2} - \frac{\bar{z}_{t+m}}{\|\bar{z}_{t+m}\|_2} \right\|_2^2 \quad \text{and} \quad L_R = \sum_{m=1}^{M} (\hat{r}_{t+m} - r_{t+m})^2 \tag{6.12}$$

For L_{GET} we need $\langle s_t, a_t, s_{t+1} \rangle$ and another state s'. From their embedding using the notation above we obtain $\tau_g = \bar{x}_{t'} {x_t}^{-1}$, the linear transformation between them, and $\kappa_g^s = \rho_\omega(\tau_g)$, the state-dependent action transformation. Now for $\bar{x}_{t'}$, we obtain the predicted next state from $\bar{x}_{t'}$ as $\bar{x}_{t'+1} = \kappa_g^s \kappa(x_t, a_t)\bar{x}_{t'} = \rho_\omega(x_{t'} x_t^{-1})\kappa(x_t, a_t)\bar{x}_{t'}$ and from $\hat{x}_{t+1}$ as $\hat{x}_{t'+1} = \tau_g \hat{x}_{t+1}$. Before penalizing the difference between these embeddings, we project them to $\hat{y}_{t'+1} = b_\eta(\hat{x}_{t'+1})$ and $\bar{y}_{t'+1} = b(\bar{x}_{t'+1})$ using projection head b_η, to get the final expression for L_{GET}:

$$L_{GET} = \left\| \frac{\hat{y}_{t'+1}}{\|\hat{y}_{t'+1}\|_2} - \frac{\bar{y}_{t'+1}}{\|\bar{y}_{t'+1}\|_2} \right\|_2^2 \tag{6.13}$$

Q-learning We pass the representation x_t to a Q-learning head q_ξ to learn policies based on the output of the Q-value estimator. The Q-value estimator is learnt by minimizing:

$$L_{DQN}(\xi, \theta) = \Big(q_\xi(\tau_\theta(s_t), a_t) - (r_t + \gamma \max_a q_\xi(\tau(s_{t+1}), a)) \Big)^2$$

We use the data efficient adaptation of Rainbow (van Hasselt et al., 2019; Hessel et al., 2018) which combines many improvements over the original DQN(Mnih et al., 2013a) such as Distributional RL(Dabney et al., 2018), Dueling DQN (Wang et al., 2016), and Double DQN (Van Hasselt et al., 2016). The total loss optimized by our model is:

$$L = L_{DQN} + \lambda_1 L_R + \lambda_2 L_{GET} + \lambda_3 L_{AET} \tag{6.14}$$

where λ_1, λ_2 and λ_3 are hyper-parameters. We use $\lambda_1 = \lambda_2 = \lambda_3 = 1$ in all experiments.

Motivated by the performance improvements due to augmentation reported in recent literature (Yarats et al., 2021b; Schwarzer et al., 2021), we also augment our states by shifting and changing the pixel intensity before encoding them. Below, we provide the algorithm in Algorithm 1, a detailed schematic in Fig. 6.4 and the network architectures of our model.

Network Architecture. We follow the baseline RL implementation of DrQ (Yarats et al., 2021b) and SPR (Schwarzer et al., 2021) by using the 3-layer convolutional encoder from (Mnih et al., 2015a) and then use a linear layer to get the parameters for the Group Parameterization. The output size of this layer varies depending on the group type, the number of blocks used and the size of the group. This defines our τ_θ. Note that the output of our encoder is a matrix for $GL(n)$ and $SE(n)$. We flatten it before we feed it to other neural networks like the Q-head $q_\xi(\cdot)$.

For the action encoder $\kappa(\cdot)$ we use a simple 1 layer MLP with batchnorm, ReLU and a hidden size of 256. We concatenate the one-hot encodings of the actions with the state representations coming from τ_θ and pass it through the action encoder to get the matrix representation of the group after parameterization.

For the reward predictor network, r_ψ we use a 2-layered MLP with batch norm, ReLU and a hidden size of 256.

For the Q-head $q_\xi(\cdot)$ we use 2-layered MLP as well.

For the projection head $p_\zeta(\cdot)$ we share the first layer of Q-head whereas, for the projection head $b_\eta(\cdot)$, we use a single layer MLP.

6.4 Experiments

We test our method on a suite of 2D Atari games, which is a popular benchmark used in RL. The full Atari suite consists of 57 games with typically 50 million environment steps. We use the sample-efficient Atari suite introduced

Algorithm 1 Equivariant Representations for RL

Denote the parameters of online networks τ_θ, κ_ϕ, p_ζ, b_η as Θ_o
Denote the parameters of target networks τ, κ, p, b as Θ_c
Denote the parameters of networks ρ_ω, q_ξ as Φ
Denote the depth of the prediction as M and batch size as N
Initialize the replay buffer $\mathcal{B}$
while *Training* **do**
  Collect $\{s, a, r, s'\}$ using policy with (Θ_o, Φ) and add to the buffer $\mathcal{B}$
  Sample a minibatch of M length sequences $\{s_{0:M}, a_{0:M}, r_{0:M}\} \sim \mathcal{B}$
  for i *in range*$(0, N)$ **do**
    if *augmentation* **then**
      $s^i_{0:M} \leftarrow \text{augment}(s^i_{0:M})$
    end
    $x^i_0 \leftarrow \tau_\theta(s^i_0)$; // state representation
    $\hat{x}^i_0 \leftarrow x^i_0$
    $l^i \leftarrow 0$
    for k *in range*$(1, M+1)$ **do**
      $\hat{x}^i_k \leftarrow \kappa_\phi(\hat{x}^i_{k-1}, a^i_{k-1})\hat{x}^i_{k-1}$; // state transition
      $\bar{x}^i_k \leftarrow \tau(s^i_k)$; // target state representation
      $\hat{z}^i_k \leftarrow p_\zeta(\hat{x}^i_k)$, $\bar{z}^i_k \leftarrow p_\zeta(\bar{x}^i_k)$; // projections
      $l^i \leftarrow l^i + \lambda_2 \| \frac{\hat{z}_{t+k}}{\|\hat{z}_{t+k}\|_2} - \frac{\bar{z}_{t+k}}{\|\bar{z}_{t+k}\|_2} \|^2_2$; // compute L_{AET} at step k
      $\hat{r}^i_k \leftarrow r_\psi(\hat{x}^i_k)$; // predict rewards
      $l^i \leftarrow l^i + \lambda_1 \|\hat{r}^i_k - r^i_k\|^2_2$; // compute L_R at step k
    end
    $j \sim \{0, \ldots, N-1\}$; // uniformly sample an index
    $\bar{x}^j_0 \leftarrow \tau(s^j_0)$; // encode the state for that index
    $\tau^i_g = \bar{x}^j_0 {x^i_0}^{-1}$; // find the group representation
    $\hat{x}^j_1 \leftarrow \tau^i_g \hat{x}^i_1$; // next state by group action
    $\bar{x}^j_1 \leftarrow \rho_\omega(\tau^i_g)\kappa(x^i_0, a^i_0)\bar{x}^j_0$; // next state by action-embedding
    $\hat{y}^i_1 \leftarrow b_\eta(\hat{x}^j_1)$, $\bar{y}^i_1 \leftarrow b(\bar{x}^j_1)$; // projections
    $l^i \leftarrow l^i + \lambda_3 \| \frac{\hat{y}^i_1}{\|\hat{y}^i_1\|_2} - \frac{\bar{y}^i_1}{\|\bar{y}^i_1\|_2} \|^2_2$; // compute L_{GET}
    $l^i \leftarrow l^i + \text{RLloss}(\hat{x}^i_0, a^i_0, r^i_0, \bar{x}^i_1; q_\xi)$
  end
  $l \leftarrow \frac{1}{N} \sum_{i=0}^{N} l_i$; // average over minibatch
  $\Theta_o, \Phi \leftarrow \text{optimize}((\Theta_o, \Phi), l)$; // update online networks
  $\Theta_c \leftarrow \Theta_o$; // copy weights to target networks
end

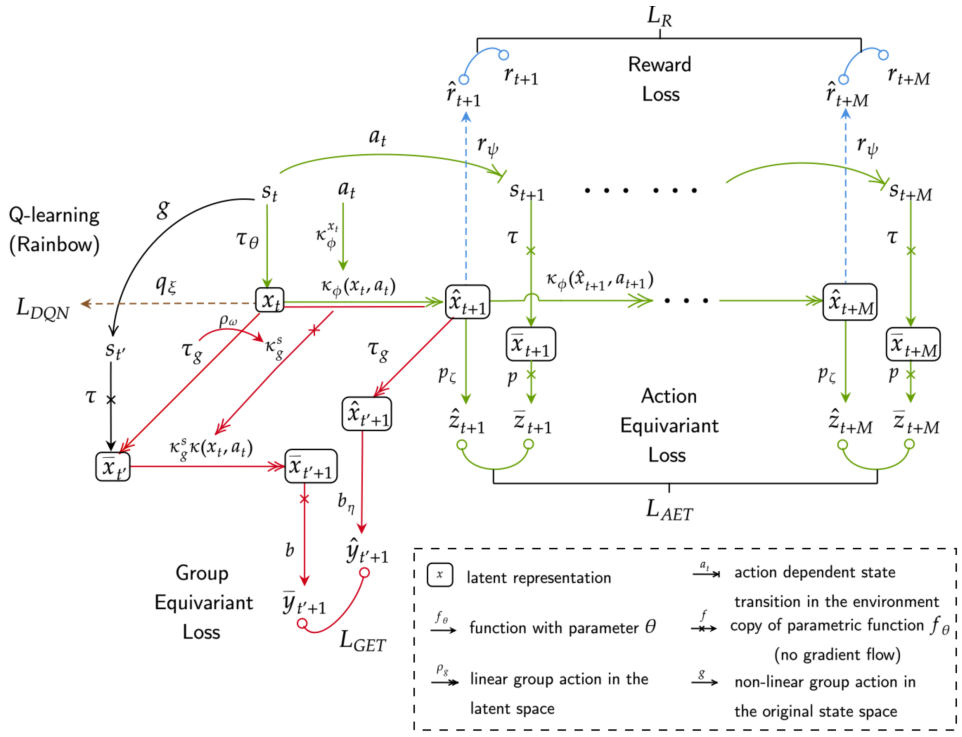


Figure 6.4: A schematic of the EqR model, applied to model-free RL. Green in the framework corresponds to learning equivariance under the agent's action and red corresponds to learning equivariance of the transition model with respect to symmetry transformation of the state-action. This color scheme is consistent with Figure 2. The part of the framework that corresponds to reward matching and Q-learning is shown in blue and brown respectively. The arrows in the schematic are differentiated by their heads and are described in the legend.

by Kaiser et al. (2019b), which consists of 26 games with only 100,000 environment steps of training data available. In our experiments, we use three types of simple connected Lie subgroup blocks including General Linear $GL(2)$, Special Euclidean $SE(2)$, and Translation $T(2)$. We provide details about the groups and a general recipe to implement them efficiently in (Mondal et al., 2022). Unless stated otherwise, our EqR model uses $SE(2)$ subgroup blocks, with $K = 12$ blocks and $M = 5$ steps during training. L_{AET} is always used to train EqR. L_{GET}, which makes the transition model equivariant with respect to the symmetry transformation of state-actions, and L_R are optional. We build our implementation on top of SPR's (Schwarzer et al.,

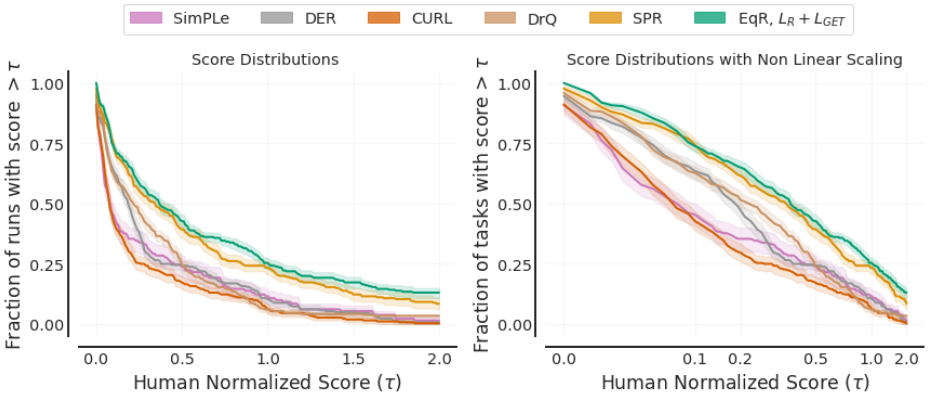


Figure 6.5: Performance profiles for different methods based on score distributions (a), and average score distributions (b). Shaded regions show pointwise 95% confidence bands. The higher the curve, the better the method is.

2021), which is based on `rlpyt` (Stooke & Abbeel, 2019) and PyTorch (Paszke et al., 2019). We use the same underlying RL algorithm and hyperparameters used by SPR for a fair comparison. Our implementation is available at https://github.com/arnab39/Symmetry-RL.

Evaluation Metrics We compute the average episodic return (the 'game score') at the end of training and normalize it with respect to human scores, as is standard practice. The human-normalized score (HNS) is given by $\frac{\text{agent score - random score}}{\text{human score - random score}}$. Since there is considerable variance across different runs, the mean and the median are not very reliable metrics. Instead, Agarwal et al. (2021) propose using bootstrapped confidence intervals (CI) with stratified sampling which is more suitable for small sample sizes (10 runs per game in our case). We report the Interquartile Mean (IQM), which is the mean across the middle 50% of the runs, as well as the Optimality Gap, which is the amount by which the algorithm fails to meet a minimum HNS of 1.0. We also provide performance profiles showing the fraction of runs above a certain normalized score, which gives a more complete picture of the performance.

Results We use 10 seeds for every game, for every variation of our model. Figure 6.5 shows performance profiles for our model, EqR with $L_R + L_{GET}$, along with other comparable methods. If one curve is strictly above another, the better method is said to "stochastically dominate" the other (Agarwal

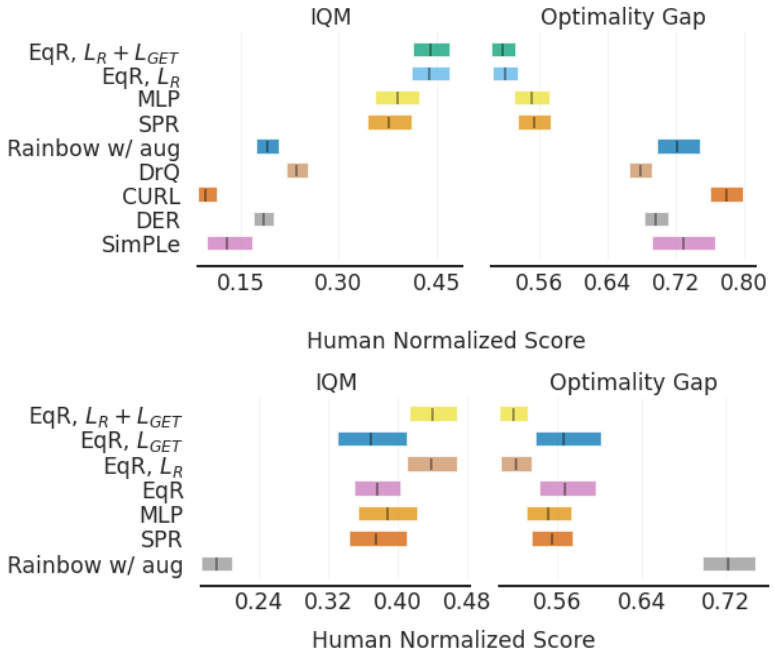


Figure 6.6: Plots of Interquartile Mean (IQM) and Optimality Gap (Agarwal et al., 2021) computed from human-normalized scores, showing the point estimates along with 95% confidence intervals (over 10 runs for all methods, 5 runs for SimPLe). A higher IQM and a lower optimality gap reflect better performance. (a) shows different methods for all 26 games. (b) shows our proposed method with different loss components for all 26 games.

et al., 2021). The curves for both variations of the proposed method are almost always above the next best method, SPR (Schwarzer et al., 2021), indicating it has a better tail distribution of scores. Figure 6.6 provides results for different methods on all 26 games. The two best variations of the proposed method outperform previous methods, and the difference is statistically significant considering the CI.

In order to better understand the effect of various modeling choices, loss functions and implementation details on the performance, we now consider different variations of EqR, with the same augmentation as the baseline for ablation studies.

Choice of Group To understand the role of the choice of a group in the embedding space, we use our EqR model with L_R. This variation of EqR is similar to DeepMDP (Gelada et al., 2019), except for the group structured

latent embedding space and group action-based state transition. In order to investigate the effect of the above two group-related constructs, we remove them and use an action encoder to predict the next states directly, referring to this as MLP which makes it like DeepMDP but with normalized mean square error loss for the model learning part.

Loss functions Figure 6.6 compares the performance of EqR with different loss components. Using EqR with the default L_{AET} results in a considerable improvement over Rainbow with augmentation (note that this is still using symmetry-based representation and transition with $SE(2)$ subgroup blocks.) Adding L_{GET} improves the performance slightly while adding only L_R improves the performance even further. We hypothesize that the reward loss plays a role in both preventing representation collapse and preserving more information about the reward distribution in the latent state embeddings. Adding both L_{GET} and L_R improves the performance only slightly. The reason why the contribution of L_{GET} is not significant is likely that this prior of an equivariant transition model with respect to symmetry transformations of state-actions is too restrictive for some games while being beneficial for others. Notably, in 17 out of a total of 26 games, including this loss term leads to a statistically significant boost in performance. These 17 games are: 'Alien', 'BankHeist', 'BattleZone', 'Boxing', 'ChopperCommand', 'CrazyClimber', 'DemonAttack', 'Freeway', 'Hero', 'Jamesbond', 'MsPacman', 'Pong', 'PrivateEye', 'Qbert', 'RoadRunner', 'Seaquest', 'UpNDown'.

6.5 Related work

The use of transformations, be it in data augmentation or self-supervision, has become a common ingredient in recent representation learning methods for deep RL. However, theoretical work on symmetry in RL goes back to Zinkevich & Balch (2001) and Ravindran & Barto (2001), both of which use symmetric MDPs. More recent use of this formalism is in van der Pol et al. (2020b); Mondal et al. (2020), where policy networks, with built-in equivariance, are shown to improve data efficiency. Closely related notions, that motivated the early work on symmetric MDPs, are model minimization (Ravindran & Barto, 2002), state abstraction (Ravindran & Barto, 2003; Li et al., 2006), MDP homomorphism (Ravindran & Barto, 2004) and lax bisimulations (Taylor, 2008). In particular, MDP homomorphism, which requires equivariance under an agent's action, encompasses the general idea of model-based reinforcement learning. For example, a latent MDP that

matches the state dynamics and the reward distribution of the environment is learned in (van der Pol et al., 2020a; Gelada et al., 2019).

Other work in RL that is relevant to our objective includes attempts to increase data efficiency using a learned model of the environment. While some methods such as SimPLe (Kaiser et al., 2019b), learn this transition model at the pixel level, the majority of approaches use a latent space model. The latent space is either learned using reconstruction (Hafner et al., 2019a,b), or through self-supervision and contrastive methods (Laskin et al., 2020b) (CURL). However, there is evidence that the improvement in sample efficiency is largely due to image augmentation, as seen in Laskin et al. (2020a) and DrQ (Yarats et al., 2021b). Using a reconstruction-based method is also inefficient because similar to pixel-level models, one needs to learn potentially irrelevant details. The fact that variations of model-free algorithms such as Data-Efficient Rainbow (DER) (van Hasselt et al., 2019) and OTRainbow (Kielak, 2019) are competitive with reconstruction-based methods without explicit representation learning components confirms this intuition. More recently SPR (Schwarzer et al., 2021) shows that data augmentation and improvements in Rainbow combined with particular forms of self-supervision, can significantly improve the sample efficiency, leading to state-of-the-art results in sample-efficient representation learning in RL.

6.6 Discussion

In this chapter, we introduced a latent variable model for representation learning in RL, considering both equivariance to an agent's action and symmetry transformations in the environment. The proposed model has the capacity to become equivariant to non-linear symmetry transformations of state-actions.

We have considered three major symmetry-related constructs within a single coherent framework. First, we use the group equivariant state and action embedding, which we achieve through Lie parameterization. We believe our Lie parameterization will have applications beyond RL, for learning symmetric representations. The world modeling constraints further ensure that the transformations captured by the equivariant embedding are relevant. Second, the equivariance to the agent's action, which when combined with group equivariant embeddings, ensures that the state transitions are captured by symmetry transformations in the latent space. Our empirical results suggest the importance of these two components in improving performance in Atari games with limited data. However, symmetry in RL can also appear in

the form of a symmetric MDP. By this we mean that not only are the state and action embeddings equivariant and that the transition model uses group transformation, but also that the latent transition model itself is equivariant under symmetry transformations of state-action pairs. This is a stricter constraint, and we found it marginally helpful in some settings of Atari games.

Key Limitations One limitation of the current approach is its reliance on a pre-defined group of transformations, which may not capture all relevant symmetries in complex environments or might impose overly restrictive constraints that hinder performance in less structured scenarios. The latter becomes more of a problem especially when we impose the equivariance constraint of the transition dynamics. Additionally, the practical application of the model among RL practitioners may be limited by its complexity and the high level of technical expertise required for effective implementation and tuning.

Future Research Directions: Future work could explore adaptive methods that learn the symmetry group directly from interactions with the environment, potentially leading to more flexible and broadly applicable models. Investigating the scalability of this approach to more complex and higher-dimensional tasks, such as those involving 3D environments or more intricate agent interactions, could also prove fruitful. Furthermore, adding model-based RL techniques within this framework might lead to more robust and efficient learning algorithms. These directions not only promise to extend the applicability of the model but also contribute to a deeper understanding of learning symmetry in reinforcement learning. Moreover, the insights from this work can also lead to the development of new theoretically grounded methods for combining both symmetric and asymmetric aspects of the environment in one model.

7

Equivariant Representations using Group Invariants

This chapter presents an alternate approach to learning equivariant representations from the data and is taken from (Shakerinava et al., 2022). While recent years have witnessed a range of exciting equivariant deep models, there are several limitations. First, most equivariant networks constrain the network architecture, often requiring specialized implementations. Moreover, transformations considered by the existing methods are often assumed to be linear in both input and representation space. This is the case for architectures designed for finite permutation groups and continuous Lie groups. Approaches that go beyond linear transformations in the input space often assume access to group information – *i.e.*, the group member that transforms one input to another is known. This paper introduces a simple approach that addresses all of these limitations.

Our approach uses the invariants of a given linear representation of a transformation group. Previously invariants were used to connect different geometries, and group theory in Klein's Erlangen program (Klein, 1893). According to this view, geometries are concerned with invariant quantities under certain transformations. For example, Euclidean geometry is concerned

with the length, angle, and parallelism of lines, among others, because Euclidean transformations preserve these. However, moving to the more general and less structured Affine geometry, notions of distance and angle are no longer relevant, while parallelism remains an invariant of the geometry. The corresponding symmetry groups are examples of Lie groups that have a subgroup relation, $E(n) < Aff(n)$, thereby enabling the groups to characterize a hierarchy (or lattice) of different geometries.

From this geometric perspective, our proposal is to induce a geometry on the embedding and make it equivariant to a given group by enforcing the invariants of their defining action. For example, distance is the invariant for Euclidean geometry, which means all distance-preserving transformations are Euclidean. Therefore, to enforce equivariance to the Euclidean group, it is sufficient to ensure that the embedding of any two data points has the same distance before and after the same transformation of the inputs; see Fig. 7.1. While this approach uses the *defining action* of different groups in the embedding space, the same group can have a non-linear and unknown action on the input space. In the pendulum example of Fig. 7.1, the group $E(3)$ acts on the value of each input image pixel using an unknown and non-linear action. Moreover, this approach does not require the pairing of group members with transformations, a piece of information that is often unavailable.

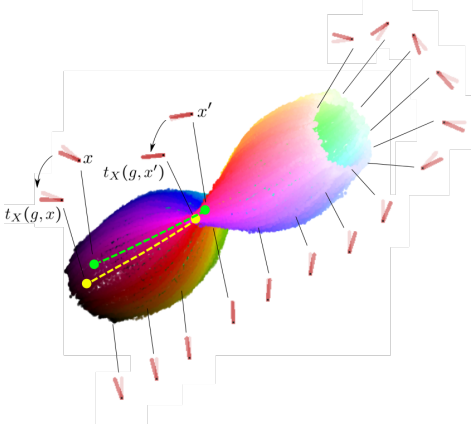


Figure 7.1: **E(3)-equivariant embedding for the pendulum.** The input x consists of a pair of images that identify both the angle and the angular velocity of a pendulum. The equivariant embedding learns to encode both: the true angle is shown by a change of color and angular velocity using a change of brightness. The two circular ends (black and white) correspond to states of maximum angular velocity in opposite directions. The SymReg objective for the *Euclidean group* learns this embedding by preserving the pairwise distance between the codes before $(f(x), f(x'))$ and after $(f(t_{\mathcal{X}}(g, x)), f(t_{\mathcal{X}}(g, x)))$ transformations of the input by $t_{\mathcal{X}}$. Therefore dashed lines have equal lengths. For the pendulum, the transformations are in the form of applying positive or negative torque in some range.

In the rest of the chapter, we arrive at the idea above from a different path by first observing that equivariance, in its general form, can be a weak inductive bias. This is because having an injective code is sufficient for equivariance to "any" transformation group. However, in this manifestation of equivariance, the group action on the embedding can be highly non-linear. Since the simplicity of the action on the embedding seems essential for equivariance to become a useful learning bias, Section 7.2 proposes to *regularize* the group action on the code to make it "simple". This *symmetry regularization* (SymReg) objective is group-dependent and the essence of our approach. Enforcing geometric invariants in the latent space is proposed as a symmetry regularization. While we focus on equivariant representation learning through self-supervision, in principle, supervised tasks can also benefit from the proposed SymReg.

7.1 Actions in the latent space matter more in equivariant models

A symmetry-based representation or embedding is a function $f : \mathcal{X} \to \mathcal{Z}$ such that both $\mathcal{X}$ and $\mathcal{Z}$ are G-sets, and furthermore, f "knows about" G-actions, in the sense that transformations of the input using $t_{\mathcal{X}}$ have the same effect as transformations of the output using some action $t_{\mathcal{Z}}$:

$$f(t_{\mathcal{X}}(g, x)) = t_{\mathcal{Z}}(g, f(x)) \quad \forall g, x \in G \times \mathcal{X} \tag{7.1}$$

The following claim shows that despite many efforts in designing equivariant networks, simply asking for the representation to be equivariant is not a strong inductive bias, and we argue that the action matters. Put another way, the strong performance of existing equivariant networks should be attributed to the fact that the group action on the embedding space is simple (linear).

Proposition 7.1.1. *Given a transformation group $t_{\mathcal{X}} : G \times \mathcal{X} \to \mathcal{X}$, the function $f : \mathcal{X} \to \mathcal{Z}$ is an equivariant representation if $\forall g \in G, x, x' \in \mathcal{X}$*

$$f(x) = f(x') \Leftrightarrow f(t_{\mathcal{X}}(g, x)) = f(t_{\mathcal{X}}(g, x')). \tag{7.2}$$

That is, two embeddings are identical iff they are identical for all transformations.

Proof. Let $\sim_f$ be an equivalence relation on $\mathcal{X}$, such that two points are "equivalent" if they have the same embedding $x \sim x' \Leftrightarrow f(x) = f(x')$. We use

$[x]_\sim$ to denote the equivalence class of x. To get an intuition for this result, first consider an injective f, where the equivalence classes are trivial $[x]_\sim = x$. In this case for any G-set $\mathcal{X}$, f is G-equivariant with G-action on $\mathcal{Z}$ defined by

$$t_{\mathcal{Z}}(g, y) \doteq f(t_{\mathcal{X}}(g, f^{-1}(y))) \quad \forall g, y \in G \times \mathcal{Z} \tag{7.3}$$

Now to see why f is equivariant to any action $t_{\mathcal{X}}$ and the corresponding $t_{\mathcal{Z}}$ defined above, simply replace the definition of $t_{\mathcal{Z}}$ into definition of equivariance Eq. (7.1)

$$t_{\mathcal{Z}}(g, f(x)) = f(t_{\mathcal{X}}(g, f^{-1}(f(x)))) = f(t_{\mathcal{X}}(g, x)) \tag{7.4}$$

For general functions, note that $f^{-1}(f(x)) = [x]_\sim$. The equation above makes sense iff $t_{\mathcal{X}}(g, x') = t_{\mathcal{X}}(g, x'') \forall x', x'' \in [x]_\sim$, which is basically the assumption of Eq. (7.2). This means $[t_{\mathcal{X}}(g, x)]_\sim \doteq [t_{\mathcal{X}}(g, x)]_\sim$, and using the $t_{\mathcal{Z}}$ of Eq. (7.3) in the definition of equivariance, we see that its condition is satisfied

$$\begin{aligned} t_{\mathcal{Z}}(g, f(x)) &= f(t_{\mathcal{X}}(g, f^{-1}(f(x)))) = f(t_{\mathcal{X}}(g, [x]_\sim)) \\ &= f([t_{\mathcal{X}}(g, x)]_\sim) = f(t_{\mathcal{X}}(g, x)) \end{aligned} \tag{7.5}$$

□

The condition above is satisfied by all injective functions, indicating that many functions are equivariant to any group.

Corollary 7.1.2. *Any* injective *function $f : \mathcal{X} \to \mathcal{Z}$ is equivariant to any transformation group $t_{\mathcal{X}} : G \times \mathcal{X} \to \mathcal{X}$, if we define G action on the embedding space as*

$$t_{\mathcal{Z}}(g, z) \doteq f(t_{\mathcal{X}}(g, f^{-1}(z))) \quad \forall g, z \in G \times \mathcal{Z} \tag{7.6}$$

The ramification of the results above in what follows is two-fold:
1. While injectivity ensures equivariance, the group action on the embedding, as shown in Eq. (7.6), can become highly non-linear. Intuitively, this action recovers $x = f^{-1}(z)$, applies the group action $x' = t_{\mathcal{X}}(x)$ in the input domains and maps back to the embedding space $f(x')$ to ensure equivariance. In the following, we push $t_{\mathcal{Z}}$ towards a simple linear G-action through optimization of f. This objective can be interpreted as a *symmetry regularization or a symmetry prior* (SymReg).

2. Although Theorem 7.1.2 uses injectivity of f for the entire $\mathcal{X}$, we only need this for the data manifold. In practice, one could enforce injectivity on the training dataset $\mathcal{D}$ using a decoder, architectural choices such as momentum encoder (He et al., 2020), or loss functions defined on the training data, such as a hinge loss (Hadsell et al., 2006) $L_{\text{hinge}}(f, \mathcal{D}) = \sum_{x,x' \neq x \in \mathcal{D}} \max(\epsilon - \|f(x) - f(x')\|, 0)$ or other losses that monotonically decrease with distance, such as $\frac{1}{\|f(x)-f(x')\|}$, or its logarithm $-\log(\|f(x) - f(x')\|)$. In experiments, we use the logarithmic barrier function.

7.2 Symmetry Regularization Objectives

In learning equivariant representations, we often do not know the abstract group G and how it transforms our data, $t_{\mathcal{X}}$. We assume that one can pick a reasonable abstract group G that "contains" the ground truth abstract group acting on the data – *i.e.*, G action on the input may not be faithful. Our goal is to learn an $f : \mathcal{X} \to \mathcal{Z}$ that is equivariant w.r.t. the actions $t_{\mathcal{X}}, t_{\mathcal{Z}}$, where $t_{\mathcal{X}} : G \times \mathcal{X} \to \mathcal{X}$ is unknown and $t_{\mathcal{Z}}$ is some (simple) G-action on $\mathcal{Z}$ of our choosing.

More Informed but Less Practical Setting. In the most informed case, the dataset also contains information about which group member $g \in G$ can be used to transform x to x' – that is, the dataset consists of triples $\langle x, g, x_t = t_{\mathcal{X}}(g, x) \rangle$. By having access to this information, we can regularize the embedding using the following loss function: $L_G^{\text{informed}}(f, D) = \sum_{\langle x,g,x_t \rangle \in \mathcal{D}} \ell\big(f(x_t) - t_{\mathcal{Z}}(g, f(x))\big)$, where ℓ is an appropriate loss function, such as the square loss. At its minimum, we have $f(x_t) = t_{\mathcal{Z}}(g, f(x))$ or $f(t_{\mathcal{X}}(g, x)) = t_{\mathcal{Z}}(g, f(x))$, enforcing equivariance condition of Eq. (7.1). However, even if the optimal value is not reached, due to its injectivity, f is still G-equivariant, and the objective above is to regularize the G action on the code. This informed setup is used in equivariant contrastive learning of (Dangovski et al., 2021). The assumption of having access to g is realistic when we know the action $t_{\mathcal{X}}$, so that we can generate $\langle x, g, x_t \rangle$ triplets. Fortunately, using group invariants, we may still learn an equivariant embedding, even if we do not have the group information tied to the dataset.

Here, we introduce our method for several well-known groups first and then elaborate on the more general treatment.

Example 3 (**Euclidean Group**). The defining action of the Euclidean group $E(n)$ is the set of transformations that preserve the Euclidean distance between

any two points in $\mathbb{R}^n$, a.k.a. isometries. These transformations are compositions of translations, rotations, and reflections. Since, for the real domain, all Euclidean isometries are linear and belong to $E(n)$, we can enforce the group structure on the embedding by ensuring that distances between the embeddings before and after any transformation match. For this, we need the dataset $\mathcal{D}$ to be a set of pairs of pairs $\big(\langle x, x_t = t_{\mathcal{X}}(g, x)\rangle, \langle x', x'_t = t_{\mathcal{X}}(g, x')\rangle\big)$, where x, x' are transformed using the same *unknown* group member g. Distance-preservation loss below combined with injection loss is sufficient to produce an $E(n)$-regularized embedding:

$$L_{E(n)}(f, \mathcal{D}) = \sum_{\big(\langle x,x_t\rangle,\langle x',x'_t\rangle\big)\in\mathcal{D}} \ell\big(\overbrace{\|f(x) - f(x')\|}^{\substack{\text{distance before the}\\ \text{transformation}}} - \overbrace{\|f(x_t) - f(x'_t)\|}^{\substack{\text{distance after the}\\ \text{transformation}}}\big) \tag{7.7}$$

For example, in the standard RL setup, where we have access to triplets (s, a, s'), we can easily form D by unrolling an episode and collecting two different state transitions corresponding to a particular action. In practice, with a finite number of actions, we can efficiently generate this dataset by keeping a separate buffer for each action where we store state transitions for that action and sample from that buffer to train the embedding function f.

Example 4 (**Orthogonal and Unitary Groups**). The defining action of the orthogonal group $O(n)$ preserves the inner product between two vectors. The analogous group in the complex domain is the unitary group, which preserves the complex inner product. Our symmetry-regularization objective enforces this invariant: $L_{O(n)}(f, \mathcal{D}) = \sum_{\big(\langle x,x_t\rangle,\langle x',x'_t\rangle\big)\in\mathcal{D}} \ell\big(f(x)^\top f(x') - f(x_t)^\top f(x'_t)\big)$.

For the unitary group, one additionally needs to embed to complex domain $\mathcal{Z} = \mathbb{C}^n$, where the only difference is in the definition of the inner product.

7.2.1 Practical Implementation

Choice of Lie group Deciding on a Lie group for each application and in particular working with the corresponding invariants can be cumbersome. A simple alternative is to use an $E(n)$-equivariant embedding for sufficiently large n. This is because Lie groups have isometric Euclidean embedding for sufficiently large n. We demonstrate this in the experiments with $SO(3)$ group in Section 7.3.1.

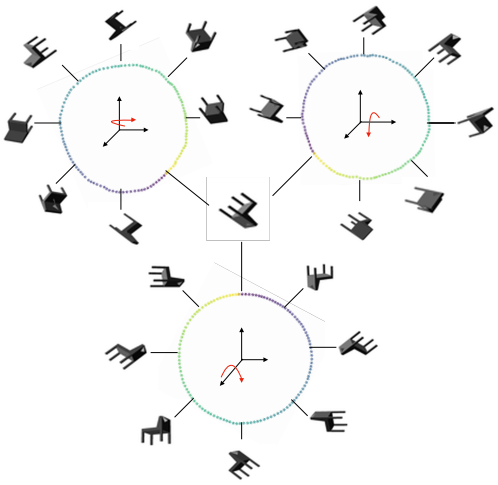


Figure 7.2: **Visualization of SymReg's latent projection for the rotating Chair dataset.** The chair is rotated in three orthogonal axes from 0 to 2π. The latent embedding for each chair pose is projected from a 16D embedding space to a 2D space for visualization. The colors of the representations are mapped to the chair's angle of rotation. We notice that the mapping function f learned is continuous with respect to the transformations of the object, and it maps the rotations along an axis to a circular manifold. This is true for each orthogonal axis of rotation. We observe a similar result for any other initial pose for the chair.

Algorithm Although SymReg is using simple loss functions for equivariant representations, for the benefit of clarity, here we give the algorithm for $E(n)$ equivariance. Algorithm 1 gives a generic training step of SymReg for $E(n)$, and Algorithm 2 provides its integration in RL environments for encoder pre-training. The policy used to collect data in the RL setting can be a behavior policy for offline RL and a random policy for online RL. Note that there are many ways SymReg can be exploited in RL but in this work, we limit ourselves to preliminary representation learning experiments.

7.3 Experiments

We conducted many experiments to qualitatively study the representation learned by SymReg and its ability to produce a disentangled representation, and quantitatively compare it against simple baselines in representation learning and downstream RL tasks. Details of architecture and training can be found in (Shakerinava et al., 2022).

7.3.1 Qualitative Analysis

In this section, we visualize the representation learned for the pendulum example from the Gym environment (Brockman et al., 2016), followed by an experiment involving a rotating object where we know the ideal embedding is the $SO(3)$ manifold. Our objective here is to visually demonstrate the behavior of SymReg and its remarkable ability to learn an embedding informed

Algorithm 2 SymReg for $E(n)$ - Training Steps

Denote the encoder as e_θ with parameters θ
Given a batch of samples $\{(x_1^i, x_2^i), (\hat{x}_1^i, \hat{x}_2^i)\}_{i=1}^B$ where B is the batch size
$l_{eqv},\ l_{barrier}, z_{all} = 0,\ 0, \{\}$
for *i in range*$(1, B)$ **do**
  $z_1^i,\ z_2^i,\ \hat{z}_1^i,\ \hat{z}_2^i = e_\theta(x_1^i),\ e_\theta(x_2^i),\ e_\theta(\hat{x}_1^i),\ e_\theta(\hat{x}_2^i)$; `// encode the data`
  $l_{eqv} = l_{eqv} + (\|z_1^i - z_2^i\| - \|\hat{z}_1^i - \hat{z}_2^i\|)^2$; `// compute SymReg loss`
  $z_{all} = z_{all} \cup \{z_1^i,\ z_2^i,\ \hat{z}_1^i,\ \hat{z}_2^i\}$
end
for z_i, z_j *in* z_{all} **do**
  if $z_i \neq z_j$ **then**
    $l_{barrier} = l_{barrier} - \log(\|z_i - z_j\|)$; `// compute barrier loss`
  end
end
$l_{eqv},\ l_{barrier} = l_{eqv}/B,\ l_{barrier}/(4B-1)^2$; `// normalize the losses`
$l_{total} = l_{eqv} + l_{barrier}$; `// compute the total loss`
$\theta \leftarrow \text{optimize}(\theta, l_{total})$; `// update the encoder params`

Algorithm 3 SymReg for Pretraining the Encoder in RL Environments

Denote the encoder as e_θ with parameters θ and action space by $\mathcal{A}$
if $\mathcal{A}$ *is continuous* **then**
  Discretize $\mathcal{A}$ into k actions $\{a_1, \ldots, a_k\}$
end
Initialize k replay buffers $\{\mathcal{B}_1, \ldots, \mathcal{B}_k\}$ for each action
for *step in range*$(1, max\_pretraining\_steps)$ **do**
  Collect $\{s, a, s'\}$ using a given policy and add to the buffer $\mathcal{B}_i$ of a
  if *step* $>$ *warm_up_steps* **then**
    Sample a batch B of buffers with repetition
    Sample two state transitions from each sampled buffer
    Create a batch of samples $\{(s_1^i, s_2^i), (\hat{s}_1^i, \hat{s}_2^i)\}_{i=1}^B$
    Perform SymReg training step with e_θ and $\{(s_1^i, s_2^i), (\hat{s}_1^i, \hat{s}_2^i)\}_{i=1}^B$
  end
end

by the non-linear transformation of the input.

The Pendulum. For this experiment, the input x is two consecutive frames of the pendulum that have been grayscaled and downsampled to 32×32 pixels. The action space is a range of torques that can be applied to the base of the pendulum. We use the action to transform the data. We use the objective of Eq. (7.7) to learn an $E(3)$-equivariant representation. To efficiently estimate $L_{E(n)}$, we use a mini-batch that consists of 64 randomly sampled observations from the environment and their transformations via three randomly sampled actions (4×64 samples in total). The model learns to parameterize the embedding using the angle and the angular momentum of the pendulum from the input data; see Fig. 7.1.

Rotating Chair. We consider a 3D chair from ModelNet40 (Wu et al., 2015) and transform it through the action of the group $SO(3)$. The group action on the input is the 2D projection into a 48×48 image after the 3D rotation of the chair. While the group of interest is $SO(3)$, we use SymReg loss of Eq. (7.7) following Section 7.2.1. We embed the chair in $\mathbb{R}^{16}$ using SymReg[1]and visualize the latent by rotating the chair along three orthogonal axes and projecting the latent codes into a 2D space. Fig. 7.2 shows three circular latent traversals of SymReg embedding corresponding to rotation around each axis, which is consistent with the structure of the $SO(3)$ manifold. The process of learning the $SO(3)$ manifold is a challenging task, and previous works assumed that the group member corresponding to each transformation is given (Quessard et al., 2020; Anonymous, 2022). In contrast, we only use the observations corresponding to similar actions during training and not the group members themselves. As we see later, this is critical in settings such as RL, where group information is unavailable. We could not produce a similar latent traversal for VAE due to collapse when rotating around some axes.

7.3.2 Quantitative Evaluation in Downstream Tasks

World Modelling

We select the Atari games Pong and Space Invaders as our environments for the world modeling experiments. These environments were previously used by Kipf et al. (2020) to evaluate the Contrastive Structured World Model (C-

[1]Note that while $SO(3)$ manifold is 3-dimensional, its isometric embedding requires a higher number of dimensions. Using a larger embedding dimension also often helps with the optimization.

Environment	Method	H@1	MRR
Atari Pong	World Model(AE)	23.8± 3.3	44.7± 2.4
	World Model(VAE)	1.0± 0.0	5.1± 0.1
	C-SWM	36.5± 5.6	56.2± 6.2
	Ours	**45.2**± 3.4	**60.2**± 3.9
Space Invaders	World Model(AE)	40.2± 3.3	59.6± 3.5
	World Model(VAE)	1.0±5.3	5.3± 0.1
	C-SWM	48.5± 7.0	66.1± 6.6
	Ours	**54.2**± 6.3	**68.7**± 5.1

Table 7.1: Hits at Rank 1 (H@1) and Mean Reciprocal Rank (MRR) of different methods.

SWM). We train the encoder using Euclidean SymReg of Eq. (7.7), freeze it, and then learn a Multi-Layer Perceptron (MLP) based transition function in the latent space. Following Kipf et al. (2020), we report Hits at Rank 1 (H@1) and Mean Reciprocal Rank (MRR), which are invariant to the embedding scale. These evaluation metrics measure the relative closeness of the following state's representation predicted by the transition model and the representation of the observed next state. We use a set of reference state representations to measure the relative closeness (embedding random observations from the experience buffer). Section 7.3.2 reports these measures and shows that a simple transition model learned on top of our embedding outperforms C-SWM in both games. Other reported baselines use an AutoEncodcer (AE) and a Variational AutoEncoder (VAE) to learn embeddings.

Reinforcement Learning

Next, we consider three Mujoco environments: InvertedPendulum, Reacher, and Swimmer from OpenAI Gym (Brockman et al., 2016) and learn directly from the image observations. We compare our model with Auto-Encoder (AE) and Self-supervised Learning (SSL) based baselines. While AE learns to reconstruct the image observations of the states, SSL learns to inject invariance (IN-SSL) or equivariance (EQ-SSL) to agent actions. Given a triplet (s, a, s), IN-SSL maximizes the likelihood of $f(s)$ and $f(s')$ being similar (SimCLR (Chen et al., 2020c)). EQ-SSL of Dangovski et al. (2022), in this context, additionally predicts the action that leads to the state transition. We introduce two variations of each model. In the first variation, the low-dimensional embedding is used as a substitute for the high-dimensional input data without further adjustment (-decoupled). The second variation

Methods	Inverted Pendulum	Reacher	Swimmer
Vanilla	500± 150	-11± 2.5	25.6± 3.4
AE-decoupled	30± 15	-13± 3.0	16± 3.9
AE-finetuned	580± 130	-11.5± 3.2	26± 4.3
In-SSL-decoupled	100± 17	-15± 2.6	12 ± 2.5
In-SSL-finetuned	550± 21	-12± 4.1	25.9 ± 4.8
Eq-SSL-decoupled	456± 190	-14.8± 3.1	18 ± 4.5
Eq-SSL-finetuned	710 ± 120	**-10± 2.6**	27 ± 3.5
SymReg-decoupled	800± 180	-14.5± 3.1	21± 4.1
Dec-SymReg-decoupled	600± 200	-12.8± 2.7	19± 5.6
SymReg-finetuned	**950± 50**	-10± 3.4	**31.5± 3.9**

Table 7.2: Average reward collected over 10 episodes for various models in Inverted Pendulum, Reacher and Swimmer. We provide the standard errors using 5 random seeds.

allows for fine-tuning during the reinforcement learning stage (-fine-tuned). We use random policy to collect trajectories for the pre-training and use Proximal Policy Optimization (PPO) (Schulman et al., 2017b) algorithm for the downstream RL task. To evaluate the data efficiency of these models, we report the average reward collected over 10 episodes in the first 100,000 steps for Reacher and Swimmer and 30,000 steps for Inverted Pendulum in Section 7.3.2 (since Inverted Pendulum generally learns faster, we took a fewer number of steps.)

We see that out of all the representation learning methods, learned representations of SymReg most adequately capture the structure of the environment in Inverted Pendulum since the RL agent just trained on the fixed representation (SymReg-decoupled) outperforms all of them, including vanilla PPO. In Reacher, SymReg, along with other non-generative models, performs poorly compared to the AE. We believe that this is because the representation is focused on transformations caused by the agent's actions while details that can be valuable from the reward's perspective — in this case, the small object that the Reacher should reach - are ignored. This observation points to a limitation of all non-generative approaches that fine-tuning can resolve. To further verify this, we combined SymReg with a Decoder (Dec-SymReg) and noticed a significant improvement in the performance of the decoupled variation. In Swimmer, again, we see that learning the agent's transformations is not enough to get all the reward information as the background movement decides how far the agent has swum. Indeed, allowing the encoder to fine-tune allows the representations to reflect the reward information and improve

performance.

7.4 Related Works

Finding effective priors and objectives for deep representation learning is an integral part of the quest for AI (Bengio et al., 2013a). Among these priors, learning equivariant deep representations has been the subject of many works over the past decade. Many recent efforts in this direction have focused on the design of equivariant maps (Wood & Shawe-Taylor, 1996; Cohen & Welling, 2016b; Ravanbakhsh et al., 2017; Kondor & Trivedi, 2018; Cohen et al., 2019b; Finzi et al., 2021; Villar et al., 2021; Dehmamy et al., 2021; Bronstein et al., 2021a) where the "linear" action of the group on the data is known. A particularly relevant example here is Villar et al. (2021), which uses group invariants to construct equivariant maps where the group acts using its linear defining action in the input space. Due to this constraint, the application of these models has been focused on fixed geometric data such as images (LeCun et al., 1995), sets (Zaheer et al., 2017b; Qi et al., 2017a), graphs (Maron et al., 2018; Kondor et al., 2018), spherical data and the (special) orthogonal group (Cohen et al., 2018; Anderson et al., 2019; Shakerinava & Ravanbakhsh, 2021; Finkelshtein et al., 2022), the Euclidean group (Thomas et al., 2018; Weiler & Cesa, 2019a; Fuchs et al., 2020a) or other physically motivated groups such as the Lorentz (Bogatskiy et al., 2020) or Poincare group (Villar et al., 2021), among others.

In the present work, the group action is unknown and possibly non-linear. Our setup is closer to the body of work on generative representation learning (Burgess et al., 2018; Chen et al., 2016; Mita et al., 2021), in which the (linear) transformation is applied to the latent space (Quessard et al., 2020; Worrall et al., 2017b; Kulkarni et al., 2015; Lenc & Vedaldi, 2016; Cohen & Welling, 2014; Falorsi et al., 2018). Among these generative coding methods, transforming autoencoder (Hinton et al., 2011) is a closely related early work, which in addition to equivariance, seeks to represent the part-whole hierarchy in the data. What additionally contrasts our work with the follow-up works on capsule networks (Sabour et al., 2017; Lenssen et al., 2018) is that SymReg is agnostic to the choice of architecture and training. We only rely on our objective function to enforce equivariance.

Since we consider learning equivariant representations by self-supervision, exciting recent progress in this area is also quite relevant (Hadsell et al., 2006; Oord et al., 2018; Chen et al., 2020b; Tian et al., 2019; He et al., 2020;

Zbontar et al., 2021; Ermolov et al., 2021). While the use of transformations is prominent in these works, in many settings, the objective encourages *invariance* to certain transformations, making such models useful for invariant downstream tasks such as classification. Similar to many of these methods, we also use transformed pairs to learn a representation, with the distinction of learning an *equivariant* representation. An exception is the recent work of Dangovski et al. (2021), which learns an equivariant representation by separating the invariant embedding from the pose, where the relative pose is learned through supervision. Therefore, in that work, in contrast to ours, one needs to know the transformation that maps one input to another. When considering the Euclidean group, SymReg preserves distances in the embedding space under non-linear transformations of the input. This embedding should not be confused with isometric embedding (Tenenbaum et al., 2000), where the objective is to maintain the pairwise distances between points in the input and the embedding space.

7.5 Discussion

In this chapter, we have developed a novel approach for learning equivariant representations by leveraging the properties of group invariants to construct embeddings that are regularized towards a simple linear action. Our method, SymReg, significantly simplifies the learning process by obviating the need for explicit group member information, which is often not readily available in many practical applications. We show how this idea can be integrated with deep reinforcement learning algorithms to improve their sample efficiency.

Key Challenges and Limitations: Our method assumes the availability of data corresponding to the same action of a symmetry group across two different starting orientations of an object, as demonstrated in the rotating chair example. This data can typically be generated by applying the same action to two different states of an object in interactive environments. However, the scalability of this method becomes problematic when these actions are continuous. This challenge complicates the generation and management of the required transformations, raising concerns about the practical implementation and scalability of the method in broader applications.

Moreover, the current framework predominantly assumes the Euclidean group to implement SymReg. This assumption may not always be optimal or appropriate depending on the underlying physical or geometrical properties of the data, potentially limiting the applicability of our method.

Future Research Directions: Future research should focus on developing automated methods for detecting the most suitable Lie group based on the data's inherent characteristics. This would enable the method to adaptively select the appropriate group, enhancing both the accuracy and the applicability of the learned representations. Addressing the challenges posed by continuous actions or symmetry transformations is crucial for extending the utility of SymReg to more dynamic and less controlled environments. Research into more efficient algorithms for generating and handling continuous transformation data will be essential.

Extending the applicability of SymReg to include a wider variety of Lie groups, such as symmetric groups or combinations thereof with Euclidean groups, could significantly broaden its utility. This extension would allow for a more nuanced representation of complex interactions and transformations across different domains. Further investigation is needed to develop and refine the objectives or mechanisms integrated within SymReg to prevent the collapse of representations. Exploring regularization techniques or loss functions that maintain the diversity and richness of the embeddings will be crucial for ensuring robust and meaningful equivariant representations.

By addressing these challenges and exploring these avenues, we can significantly enhance SymReg making it capable of handling the complexities of real-world data in a principally efficient manner.

8

Learning representations using Koopman Theory

Although Chapter 6 and Chapter 7The ability to predict the outcome of an agent's action over long horizons is a crucial unresolved challenge in Reinforcement Learning (RL) (Sutton & Barto, 2018; Mnih et al., 2015a; Silver et al., 2017a; Mnih et al., 2016). This is especially important in model-based RL and planning, where deriving a policy from the learned dynamics models allows one to efficiently accomplish a wide variety of tasks in an environment (Du & Narasimhan, 2019; Hafner et al., 2020; Sikchi et al., 2021; Hansen et al., 2022; Schrittwieser et al., 2020; Jain et al., 2022). In fact, state-of-the-art model-free techniques also rely on dynamics models to learn a better representation for downstream value prediction tasks (Schwarzer et al., 2020). Thus, obtaining accurate long-range dynamics models in the presence of input and control is crucial.

In this chapter, we leverage techniques and perspectives from *Koopman theory* (Koopman, 1931; Mauroy et al., 2020; Koopman & Neumann, 1932; Brunton et al., 2021) to address this key problem in long-range dynamics modeling of interactive environments. This chapter provides a unifying perspective over several existing directions on stabilizing gradients through

time, including long-range sequence modeling using state-space models (Gu et al., 2020, 2022b,a; Gupta et al., 2022), the use of unitary matrices (Arjovsky et al., 2016) and certain lie group representations (Mondal et al., 2022) in latent dynamic models. We arrive at this formulation of structuring representations from dynamical systems while exploring methods to speed up dynamics modeling technique introduced in Chapter 4.

The application of Koopman theory allows us to linearise a nonlinear dynamical system by creating a bijective mapping to linear dynamics in a possibly infinite dimensional space of *observables*.[1]

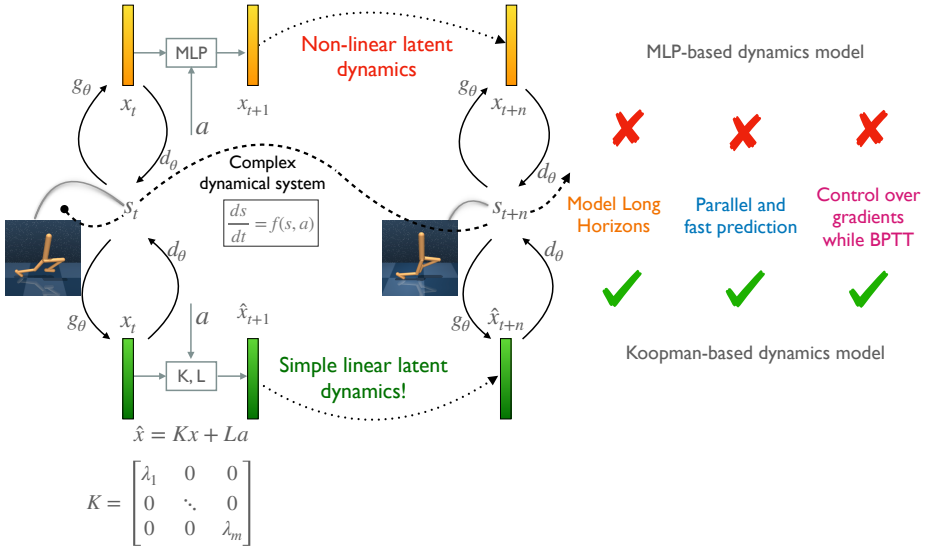


Figure 8.1: A comparison of our Koopman-based linear dynamics model with a non-linear MLP-based dynamics model. The Diagonal Koopman formulation allows for modeling longer horizons efficiently with control over gradients. Here BPTT stands for Backpropagation Through Time.

Conveniently, a deep neural network can learn to produce such a mapping, enabling a reliable approximation of the non-linear dynamics in the finite dimension of a linear latent space so that only a finite subset of the most relevant (complex-valued) Koopman observables need to be tracked.

[1]The term observables can be misleading as it refers to the latent space in the machine learning jargon.

We show that this linearization has two major benefits: 1) The eigenvalues of the linear latent-space operator are directly related to the stability and expressive power of the controlled dynamics model. 2) This can help to avoid a computational bottleneck due to the sequential nature of dynamics. This is achieved by a reformulation using convolution and diagonalization of the linear operator. In other words, one can perform efficient parallel training of the model over time steps, despite dealing with a controlled dynamical system.

Our experimental results in offline-RL datasets demonstrate the effectiveness of our approach for reward and state prediction over a long horizon. In particular, we report competitive results against dynamics modeling baselines that use Multi-Layer Perceptrons (MLPs), Gated Recurrent Units (GRUs), Transformers and Diagonal State Space Models (Gu et al., 2022a) while being significantly faster. Finally, we also present encouraging results for model-based planning and model-free RL with our Koopman-based dynamics modeling.

8.1 Background

8.1.1 Koopman Theory for Dynamical Systems

In the context of non-linear dynamical systems, the *Koopman operator*, a.k.a. the Koopman-von Neumann operator, is a linear operator for studying the system's dynamics. The Koopman operator is defined as a linear transformation that acts on the space of functions or observables of the system, known as the observables space $\mathcal{F}$. For a non-linear discrete or continuous time dynamical system

$$x_{t+1} = F(x_t) \quad \text{or} \quad \frac{dx}{dt} = f(x) \tag{8.1}$$

the Koopman operator $\mathcal{K} : \mathcal{F} \to \mathcal{F}$, is defined as $\mathcal{K}h \cong h \circ F$ where $\mathcal{F}$ is the set of all functions or observables that form an infinite-dimensional Hilbert space. In other words, for every function $h : X \to \mathbb{R}$ belonging to $\mathcal{F}$, where $x_t \in X \subset \mathbb{R}^n$, we have

$$(\mathcal{K}h)(x_t) = h(F(x_t)) = h(x_{t+1}).$$

The infinite dimensionality of the Koopman operator presents practical limitations. Addressing this, we seek to identify a subspace $\mathcal{G} \subset \mathcal{F}$, spanned by a

finite set of observables $h_1, \ldots, h_m$ where typically $m >> n$, that approximates invariance under the Koopman operator.

Constraining the Koopman operator on this invariant subspace results in a finite-dimensional linear operator $K \in \mathbb{C}^{m \times m}$, called the *Koopman matrix*, which satisfies $h(x_{t+1}) = Kh(x_t)$. Usually, the base observation functions $\{h_i\}_i$ are hand-crafted using knowledge of the system's underlying physics. However, data-driven methods have recently been proposed to learn the Koopman operator by representing the base observations or their eigenfunction basis using deep neural networks, where a decoder reconstructs the input from linear latent dynamics (*e.g.*, Lusch et al., 2018; Champion et al., 2019). Linearizing the dynamics allows for closed-form solutions for the predictions of the dynamical system, as well as stability analysis based on the eigendecomposition of the Koopman matrix (Fan et al., 2022; Yi & Manchester, 2023).

The observation functions $\{h_i\}_i$ spanning an invariant subspace can always be chosen to be *eigenfunctions* $\{\phi_i\}_i$ of the Koopman operator, *i.e.*, $\mathcal{K}\phi_i(x) = \lambda_i \phi_i(x), \forall x$. With this choice of observation functions, the Koopman matrix becomes diagonal,

$$K_D = \text{diag}(\lambda_1, \ldots, \lambda_m), \quad \lambda_i \in \mathbb{C} \quad \forall i. \tag{8.2}$$

By using a diagonal Koopman matrix, we are effectively offloading the task of learning the suitable eigenfunctions of the Koopman operator that form an invariant subspace to the neural network encoder h. Additionally, for a continuous time input, to be able to model higher-order frequencies in the eigenspectrum and provide better approximations, one could adaptively choose the most relevant eigenvalues and eigenfunctions for the current state (Lusch et al., 2018). This means $h(x_{t+1}) = K_D(\lambda(h(x_t)))h(x_t)$, where $\lambda : \mathcal{G} \to \mathbb{C}^m$ can be a neural network.

8.1.2 Approximate Koopman with Control Input

The Koopman operator can be extended to non-linear *control* systems, where the state of the system is influenced by an external control input u_t such that $x_{t+1} = F(x_t, u_t)$ or $\frac{dx}{dt} = f(x, u)$. Simply treating the pair (x, u) as the input x, the Koopman operator becomes $(\mathcal{K}h)(x_t, u_t) = h(F(x_t, u_t), u_{t+1}) = h(x_{t+1}, u_{t+1})$. If the effect of the control input on the system's dynamics is

linear, i.e., the *control affine* setting, we have

$$f(x, u) = f_0(x) + \sum_{i=1}^{m} f_i(x)u_i.$$

assuming a finite number of eigenfunctions. For such control-affine system, one could show that the Koopman operator is *bilinearized* (Brunton et al., 2021; Bruder et al., 2021):

$$h(x_{t+1}) = \mathcal{K}(u)h(x_t) = (\mathcal{K}_0 + \sum_{i=1}^{m} u_i\mathcal{K}_i)h(x_t).$$

More generally, one could make the Koopman operator a function of u_t, so that the resulting Koopman matrix satisfies $h(x_{t+1}) = K(u_t)h(x_t)$.

Connections to Symmetry-based methods. This is the approach taken by (Weissenbacher et al., 2022) to model symmetries of dynamics in offline RL. When combined with the diagonal Koopman matrix of Eq. (8.2), and fixed modulus of the complexed valued eigenspectrum $|\lambda_i| = 1\,\forall i$, the Koopman matrix becomes a product of matrix representation of $SO(2)$. This resulting latent dynamics model, resembles a special setting of the symmetry-based approach of Chapter 6, where group representations are used to encode states and state-dependent actions. The latent representation can be both represented as complex numbers or their equivalent matrix representation which combines rotation matrices with a scaling factor given by the magnitude of the complex number.

An alternate approach to approximating the Koopman operator with a control signal assumes a decoupling of state and control observables $h(x, u) = [h(x), f(u)] = [h_1(x), ..., h_m(x), f_1(u), ..., f_n(u)]$ (Brunton et al., 2021; Bruder et al., 2019). This gives rise to a simple linear evolution:

$$h(x_{t+1}) = Kh(x_t) + Lf(u_t) \tag{8.3}$$

where $K \in \mathbb{C}^{m\times m}$ and $L \in \mathbb{C}^{m\times l}$ are matrices representing the linear dynamics of the state observables and control input, respectively. We build on this approach in our proposed methodology. Although, as shown in (Brunton et al., 2016, 2021; Bruder et al., 2019), even assuming a linear control term Lu_t can perform well in practice, we use neural networks for both f and h. Our rationale for choosing this formulation is that the additive form of

Eq. (8.3) combined with the *diagonalized Koopman matrix* of Eq. (8.2), enable fast parallel training across time-steps using a convolution operation where the convolution kernel can be computed efficiently. Moreover, this setting is amenable to the analysis of gradient behaviour, as discussed in Section 8.2.2. For a more comprehensive overview of Koopman theory in the context of controlled and uncontrolled dynamics, we direct readers to (Brunton et al., 2021; Mauroy et al., 2020).

8.2 Dynamics Model

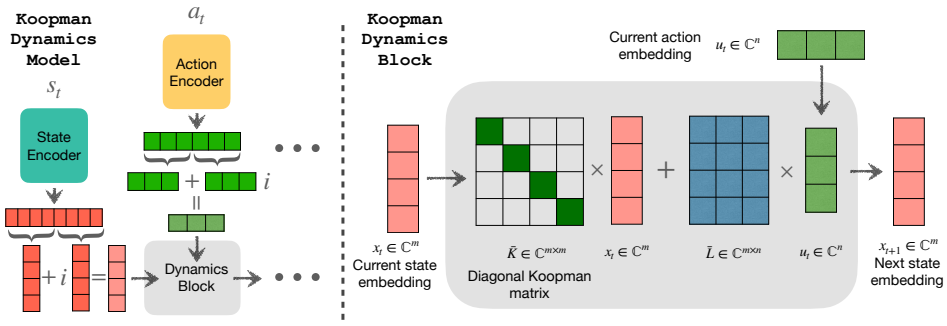


Figure 8.2: A schematic of the latent Koopman dynamics model. Both actions and initial state embedding are encoded into a latent space in complex ($\mathbb{C}$) domain before passing through the Koopman dynamics block.

8.2.1 Linear Latent Dynamics Model

Our task in dynamics modeling is to predict the sequence of future states $s_{t+1}, ..., s_{t+\tau}$ given a starting state s_t and some action sequence $a_t, a_{t+1}, ..., a_{t+\tau-1}$, assuming a Markovian system. Following the second approach described in Section 8.1.2, we assume that state and control observables are decoupled, and use neural network encoders[2] to encode both states and actions

$$x_t = h_\theta(s_t) \quad \text{and} \quad u_t = f_\phi(a_t). \tag{8.4}$$

[2]We use a CNN for pixel input and an MLP for state-based input and actions.

Due to this decoupling, the continuous counterpart of latent space dynamics Eq. (8.3) is

$$\frac{\mathrm{d}x}{\mathrm{d}t} = Kx(t) + Lu(t), \tag{8.5}$$

and the solution is given by $x(s) = e^{Ks}x(0) + \int_0^s e^{K(s-t)}Lu(t)\mathrm{d}t$ where e^{Ks} is a matrix exponential. One could discretize this to get $\hat{x}_{t+1} = \bar{K}x_t + \bar{L}u_t$, where $\bar{K}$ and $\bar{L}$ are obtained by *Zero-Order Hold (ZOH)* discretization of the continuous-time equation (Iserles, 2009):

$$\bar{K} = \exp(\Delta t K) \qquad \bar{L} = (K)^{-1}(\exp(\Delta t K) - 1)L$$

In practice, we assume that observations are sampled uniformly in time and use a learnable time step parameter Δt. As $\hat{x}_{t+1} = \bar{K}x_t + \bar{L}u_t$, we can unroll it to get future predictions up to τ time steps that is:

$$\hat{x}_{t+1} = \bar{K}x_t + \bar{L}u_t$$

$$\hat{x}_{t+2} = \bar{K}\hat{x}_{t+1} + \bar{L}u_{t+1} = \bar{K}^2x_t + \bar{K}\bar{L}u_t + \bar{L}u_{t+1}$$

$$\hat{x}_{t+3} = \bar{K}\hat{x}_{t+2} + \bar{L}u_{t+2} = \bar{K}^3x_t + \bar{K}^2\bar{L}u_t + \bar{K}\bar{L}u_{t+1} + \bar{L}u_{t+2}$$

$$\cdots$$

$$\hat{x}_{t+\tau} = \bar{K}^\tau x_t + \bar{K}^{\tau-1}\bar{L}u_t + \bar{K}^{\tau-2}\bar{L}u_{t+1} + \cdots + \bar{L}u_{t+\tau-1}$$

Writing the above equations in a matrix form we get:

$$[\hat{x}_{t+1}, \cdots, \hat{x}_{t+\tau}] = [\bar{K}, \cdots, \bar{K}^\tau]x_t + [c_t, \cdots, c_{t+\tau-1}] \underbrace{\begin{bmatrix} I & \bar{K} & \dots & \bar{K}^{\tau-1} \\ 0 & I & \cdots & \bar{K}^{\tau-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I \end{bmatrix}}_{\Gamma}, \tag{8.6}$$

where $c_{t+k} = \bar{L}u_{t+k}$, Γ encodes the effect of each input on all future observables, and the $\hat{}$ in $\hat{x}$ distinguishes the prediction from the ground truth observable x.

We can then recover the predicted state sequence by inverting the function h_θ, using a decoder network $\hat{s}_{t+k} = d_\xi(\hat{x}_{t+k})$. Loss functions in both input and latent space can be used to learn the encoder, decoder, and latent dynamics

model. We refer to these loss functions as the *Koopman consistency loss* and the *state-prediction loss*, respectively:

$$\mathcal{L}_{\text{consistency}} = \sum_{k=1}^{\tau} \|\hat{x}_{t+k} - x_{t+k}\|_2^2 \qquad \mathcal{L}_{\text{state-pred}} = \sum_{k=1}^{\tau} \|d_\xi(\hat{x}_{t+k}) - s_{t+k}\|_2^2 \tag{8.7}$$

8.2.2 Diagonalization, Efficiency and Stability of the Koopman Operator

Since the set of diagonalizable matrices is dense in $\mathbb{C}^{m\times m}$ and has a full measure, we can always diagonalize the Koopman matrix as shown in Eq. (8.2). Next, we show that this diagonalization can be leveraged for efficient and parallel forward evolution and training.

To unroll the latent space using Eq. (8.6) we need to perform matrix exponentiation and dense multiplication. Using a diagonal Koopman matrix $\bar{K} = \text{diag}(\bar{\lambda}_1, \ldots, \bar{\lambda}_m)$, this calculation is reduced to computing an $m \times (\tau+1)$ complex-valued *Vandermonde matrix* Λ, where $\Lambda_{i,j} = \bar{\lambda}_i^j$, and doing a row-wise circular convolution of this matrix with a sequence of vectors. We use the property of matrix exponential of diagonal matrices. In particular, we have $\bar{K} = \text{diag}(\bar{\lambda}_1, \ldots, \bar{\lambda}_m)$ implies that $\bar{K}^\tau = \text{diag}(\bar{\lambda}_1^\tau, \ldots, \bar{\lambda}_m^\tau)$. Recall that $x, c \in \mathbb{C}^m$ are complex vectors. We use superscript to index their elements – *i.e.*, $x_t = [x_t^1, \ldots, x_t^m]$. With this notation, we get that for i-th index of predicted vectors $\hat{x}_{t+k}$ s the following equations hold:

$$\begin{aligned}
\hat{x}_{t+1}^i &= \bar{\lambda}_i x_t^i + c_t^i \\
\hat{x}_{t+2}^i &= \bar{\lambda}_i^2 x_t^i + \bar{\lambda}_i c_t^i + c_{t+1}^i \\
&\cdots \\
\hat{x}_{t+\tau}^i &= \bar{\lambda}_i^\tau x_t^i + \bar{\lambda}_i^{\tau-1} c_t^i + \bar{\lambda}_i^{\tau-2} c_{t+1}^i + \cdots + c_{t+\tau-1}^i
\end{aligned}$$

The above equations can be denoted by an expression using the circular convolution operator given by:

$$[\hat{x}_{t+1}^i, \ldots, \hat{x}_{t+\tau}^i] = [\bar{\lambda}_i, \ldots, \bar{\lambda}_i^\tau] x_t^i + [1, \bar{\lambda}_i, \ldots, \bar{\lambda}_i^{\tau-1}] \circledast [c_t^i, ..., c_{t+\tau-1}^i] \tag{8.8}$$

where $\circledast$ is circular convolution with zero padding. The convolution can be efficiently computed for longer time steps using the Fast Fourier Transform. (Brigham, 1988)

Gradients Through Time

We show how we can control the behavior of gradients through time by constraining the real part of the eigenvalues of the Diagonal Koopman matrix. Let μ and ω refer to the real and imaginary part of the Koopman eigenvalues – that is $\lambda_j = \mu_j + i\omega_j$.

Theorem 8.2.1. *For every time step $k \in \{1, .., \tau\}$ in the discrete dynamics, the norm of the gradient of any loss at k-step given by $\mathcal{L}_k$ with respect to latent representation at time step t given by x_t is a scaled version of the norm of the gradient of the same loss by x_{t+k}, where the scaling factor depends on the exponential of the real part of the Koopman eigenvalues, that is:*

$$|\frac{\partial \mathcal{L}_k}{\partial x_t^j}| = e^{k\Delta t \mu_j} |\frac{\partial \mathcal{L}_k}{\partial x_{t+k}^j}| \quad \forall j \in \{1, .., m\}.$$

and similarly, for all $l \leq k$, the norm of the gradient of $\mathcal{L}_k$ with respect to the control input at time step $t + l - 1$ given by is c_{t+l-1}^j is a scaled version of the norm of the gradient of $\mathcal{L}_k$ by x_{t+k}, where the scaling factor depends on the exponential of the real part of the Koopman eigenvalues, that is:

$$|\frac{\partial \mathcal{L}_k}{\partial c_{t+l-1}^j}| = e^{(k-l)\Delta t \mu_j} |\frac{\partial \mathcal{L}_k}{\partial \hat{x}_{t+k}^j}| \quad \forall j \in \{1, .., m\}$$

Proof. We now provide a proof sketch of Theorem 3.1. As $\lambda_j = \mu_j + i\omega_j$, discretizing the diagonal matrix (using ZOH) and taking its k-th power gives us $\bar{K}_j^k = e^{k\Delta t \mu_j} e^{ik\Delta t \omega_j}$. Using this we can write

$$\hat{x}_{t+k}^j = \bar{K}_j^k x_t^j + \sum_{l=1}^{k} \bar{K}_j^{k-l} c_{t+l-1}^j$$

Now applying the chain rule, we get derivatives of the loss with respect to x_t^j and c_{t+l-1}^j:

$$\implies \frac{\partial \mathcal{L}_k}{\partial x_t^j} = \frac{\partial \hat{x}_{t+k}^j}{\partial x_t^j} \frac{\partial \mathcal{L}_k}{\partial \hat{x}_{t+k}^j} = \bar{K}_j^k \frac{\partial \mathcal{L}_k}{\partial \hat{x}_{t+k}^j} = e^{k\Delta t \mu_j} e^{ik\Delta t \omega_j} \frac{\partial \mathcal{L}_k}{\partial \hat{x}_{t+k}^j}$$

$$\implies \frac{\partial \mathcal{L}_k}{\partial c_{t+l-1}^j} = \frac{\partial \hat{x}_{t+k}^j}{\partial c_{t+l-1}^j} \frac{\partial \mathcal{L}_k}{\partial \hat{x}_{t+k}^j} = \bar{K}_j^{k-l} \frac{\partial \mathcal{L}_k}{\partial \hat{x}_{t+k}^j} = e^{(k-l)\Delta t \mu_j} e^{i(k-l)\Delta t \omega_j} \frac{\partial \mathcal{L}_k}{\partial \hat{x}_{t+k}^j}$$

As $|e^{i\theta}| = 1$, we get the result of partial derivatives given in Theorem 3.1. □

The theorem implies that the amplitude of the gradients from a future time step scales exponentially with the real part of each diagonal Koopman eigenvalue. We use this theorem for better initialization of the diagonal Koopman matrix as explained in the next section.

Initialization of the Eigenspectrum

The *imaginary* part of the eigenvalues of the diagonal Koopman matrix captures different frequency modes of the dynamics. Therefore, it is helpful to initialize them using increasing order of frequency, that is, $\omega_j := \alpha j\pi$, for some constant α, to cover a wide frequency range.

From Theorem 8.2.1, we know that the real part of the eigenvalues impacts the gradient's behavior through time. To avoid vanishing gradients and account for prediction errors over longer horizons, one could eliminate the real part $\mu_j := 0$. This choice turns the latent transformations into blocks of 2D rotations (Mondal et al., 2022). This is also related to using Unitary Matrices to avoid vanishing or exploding gradients in Recurrent Neural Networks (Arjovsky et al., 2016). However, intuitively, we might prefer to prioritize closer time steps. This can be done using small negative values *e.g.*, $\mu_j \in \{-0.1, -0.2, -0.3\}$. An alternative to having a fixed real part is to turn it into a bounded learnable parameter $\mu_j \in [-0.3, -0.1]$. We empirically found $\mu_j := -0.2\ \forall j$, and $\omega_j := j\pi$ to be good choices and use this as our default initialization.

8.3 Dynamics modeling in RL and Planning

8.3.1 Forward dynamics modeling in RL

Dynamics models have been instrumental in attaining impressive results on various tasks such as Atari (Schrittwieser et al., 2020; Hafner et al., 2020) and continuous control (Hafner et al., 2020; Jiang et al., 2020; Sikchi et al., 2021; Lowrey et al., 2019). By building a model of environment dynamics, *model-based RL* can generate trajectories used for training the RL algorithm. This can significantly reduce the sample complexity in comparison to model-free techniques. However, in model-based RL, inaccurate long-term predictions can generate poor trajectory rollouts and lead to incorrect expected return calculations, resulting in misleading policy updates. Forward dynamics modeling has also been successfully applied to *model-free RL* to improve the sample efficiency of the existing model-free algorithms. These methods use it to design self-supervised auxiliary losses for representation learning

using consistency in forward dynamics in the latent and the observation space (Jaderberg et al., 2017; Schwarzer et al., 2020; Srinivas et al., 2020).

8.3.2 Koopman Self-Predictive Representations

To avoid overconstraining the latent space using the dynamics model, SPR uses *projection heads*. Similarly, we encode the Koopman observables using a projection layer above the representations used for Q-learning. Using s_t for the input pixel space, the representation space for the Q-learning is given by $z_t = e_\theta(s_t)$ where e_θ is a CNN. The Koopman observables x_t is produced by encoding z_t using a projector p_θ such that $x_t = p_\theta(z_t)$. Moreover, it can be decoded into z_t using the decoder $d_\theta(\hat{x}_t)$. Loss functions are the prediction loss Eq. (8.7) and TD-error

$$\mathcal{L}_{\text{SSL}} = \sum_{k=1}^{\tau} \|d_\theta(\hat{x}_{t+k}) - e_{\theta^-}(s_{t+k})\|_2^2 + \|Q_\theta(z_t, a_t) - (r_t + Q_{\theta^-}(e_{\theta^-}(s_{t+1}), a_{t+1}))\|_2^2. \tag{8.9}$$

Here, a_{t+1} is sampled from the policy π. The policy is learned from the representations using Soft Actor-Critic (SAC) (Haarnoja et al., 2018a). As opposed to SPR, we do not use moving averages for the target encoder parameters and simply stop gradients through the target encoders and denote it as e_{θ^-}. Moreover, we drop the consistency term in the Koopman space in Eq. (8.9) as we empirically observed adding consistency both in Koopman observable (x), and Q-learning space (z) promotes collapse and makes training unstable, resulting in a higher variance in the model's performance.

8.3.3 Model-based Planning

Model Predictive Control (MPC) is a control strategy that uses a dynamics model $s_{t+1} = f(s_t, a_t)$ to plan for a sequence of actions $a_t, a_{t+1}, \ldots, a_{t+\tau-1}$ that maximizes the expected return over a finite horizon.

The optimization problem is:

$$\arg\max_{a_{t:t+\tau-1}} \mathbb{E}\left[\sum_{i=t}^{t+\tau-1} \gamma^i r(s_i, a_i)\right] \tag{8.10}$$

where γ is typically set to 1, that is, there is no discounting. Heuristic population-based methods, such as a cross entropy method (Rubinstein & Kroese, 2004) are often used for dynamic planning. These methods perform

a local trajectory optimization problem, corresponding to the optimization of a cost function up to a certain number of time steps in the future (Williams et al., 2015, 2018; Okada & Taniguchi, 2020). In contrast to Q-learning, this is myopic and can only plan up to a certain horizon, which is predetermined by the algorithm.

One can also combine MPC with RL to approximate long-term returns of trajectories that can be calculated by bootstrapping the value function of the terminal state. In particular, methods like *TD-MPC* (Hansen et al., 2022) and *LOOP* (Sikchi et al., 2021) combine value learning with planning using MPC. The learned value function is used to bootstrap the trajectories that are used for planning using MPC. Additionally, they learn a policy using the *Deep Deterministic Policy Gradient* (Lillicrap et al., 2016) or *Soft Actor-Critic (SAC)* (Haarnoja et al., 2018a) to augment the planning algorithm with good proposal trajectories. Alternative search methods such as Monte Carlo Tree search (Coulom, 2006) are used for planning in discrete action spaces. The performance of the model-based planning method heavily relies on the accuracy of the learned model.

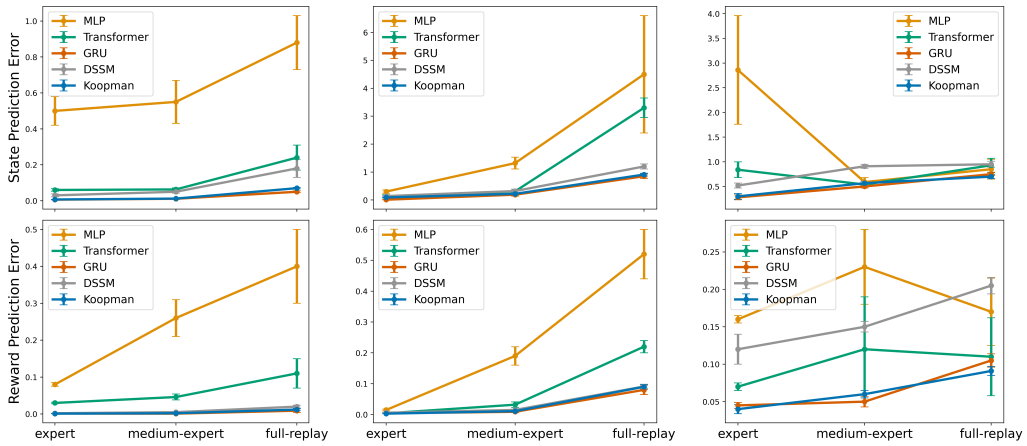
8.3.4 Koopman TDMPC

TDMPC uses an MLP to design a Task-oriented Latent Dynamics (TOLD) model, which learns to predict the latent representations of the future time steps. The learnt model can then be used for planning. To stabilize training TOLD, the weights of a *target encoder network* h_{θ^-}, are updated with the exponential moving average of the *online network* h_θ.

Since dynamics modeling is no longer the only objective, in addition to the latent consistency of Eq. (8.7), the latent x^t is also used to predict the reward and Q-function, which takes the latent representations as input. Hence, the dynamics model learns to jointly minimize the following:

$$\mathcal{L}_{\text{TOLD}} = c_1 \sum_{k=1}^{\tau} \lambda^k \|\hat{x}_{t+k} - h_{\theta^-}(s_{t+k})\|_2^2 + c_2 \sum_{k=0}^{\tau} \lambda^k \|R_\theta(\hat{x}_{t+k}, a_{t+k}) - r_{t+k}\|_2^2 + c_3 \sum_{k=0}^{\tau} \lambda^k \|Q_\theta(\hat{x}_{t+k}, a_{t+k}) - y_{t+k}\|_2^2 \tag{8.11}$$

where R_θ, π_θ, Q_θ are respectively reward, policy and Q prediction networks, where $y_{t+k} = r_{t+k} + Q_{\theta^-}(\hat{x}_{t+k+1}, \pi_\theta(\hat{x}_{t+k}))$ is the 1-step bootstrapped TD target. Moreover, the policy network is learned from the latent representation



(a) `hopper-v2` (b) `walker2d-v2` (c) `halfcheetah-v2`

Figure 8.3: Forward state and reward prediction error in Offline Reinforcement Learning environments. We consider five dynamics modeling techniques and perform this prediction task over a **horizon of 100 environment steps**. The results are over 3 runs. Our Koopman-based method is competitive with the best performing GRU baseline while being 2× faster. See (Mondal et al., 2023) for exact numerical values.

by maximizing the Q value at each time step. This additional policy network helps to provide the next action for the TD target and a heuristic for planning. For details on the planning algorithm and its implementation, see (Hansen et al., 2022).

8.4 Experiments

8.4.1 Long-Range Dynamics Modeling with Control

We model the non-linear controlled dynamics of MuJoCo environments (Todorov et al., 2012) using our Koopman operator. Our goal is to understand how well a simple linear dynamics model in the latent space, introduced in Section 8.2.1, can capture the complex dynamics of these interactive environments. We choose a standard MLP-based latent non-linear dynamics model with two linear layers followed by a ReLU non-linearity as one of our baselines. To make it easier to compare with Koopman dynamics model, we use state embeddings of the same dimensionality using g_θ, which is an MLP, as described in Section 8.2.1. This embedding is then fed to the MLP-based latent dynamics model or the diagonal Koopman operator. This approach is widely used in RL

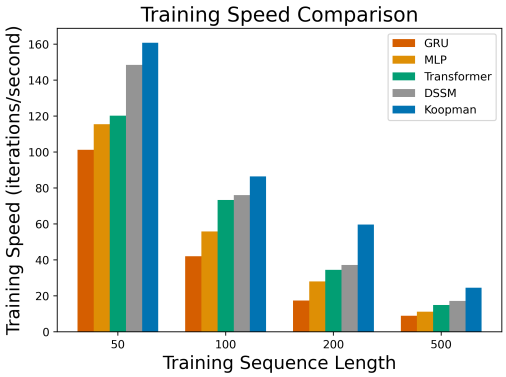


Figure 8.4: Training speed in iterations/second (↑) for the state prediction task using different dynamics model on `halfcheetah-expert-v2`. Each iteration consists of one gradient update of the entire model using a mini-batch of 256 in A100 GPU.

for dynamics modeling (Sikchi et al., 2021; Hansen et al., 2022; Schrittwieser et al., 2020; Schwarzer et al., 2020). To further compare our model with more expressive alternatives that can be trained in parallel, we design a causal Transformer (Vaswani et al., 2017b; Chen et al., 2021) based dynamics model, which takes a masked sequence of state-actions and outputs representations that are used to predict next states and rewards.[3] Following the success of (Hafner et al., 2020), we also design a GRU-based dynamics model that also takes a masked sequence of state-actions to output representations for state and reward prediction. Finally, we use the same strategy to also design a DSSM-based (Gu et al., 2022a) dynamics model. To ensure a fair comparison in terms of accuracy and runtime, we maintain an equal number of trainable parameters for all the aforementioned models. (see (Mondal et al., 2023) for more details)

For the forward dynamics modeling experiments, we use the D4RL (Fu et al., 2020) dataset, which is a popular offline-RL environment. We select offline datasets of trajectories collected from three different popular Gym-MuJoCo control tasks: `hopper`, `walker`, and `halfcheetah`. These trajectories are obtained from three distinct quality levels of policies, offering a range of data representing various levels of expertise in the task: expert, medium-expert, and full replay. We divide the dataset of 1M samples into 80:20

[3]All the states after the starting state are masked, but actions are fed to the model for all future time-steps.

splits for training and testing, respectively. To train the dynamics model, we randomly sample trajectories of length τ from the training data, where τ is the horizon specified during training. We test our learned dynamics model for a horizon length of 100 by randomly sampling 50,000 trajectories of length 100 from the test set. We use our learned dynamics model to predict the ground truth state sequence given the starting state and a sequence of actions.

Training stability We can only train the MLP-based dynamics model for 10-time steps across all environments. Using longer sequences often results in exploding gradients and instability in training due to backpropagation through time (Sutskever, 2013). The alternative of using tanh in MLP-based models can lead to vanishing gradients. In contrast, our diagonal Koopman operator handles long trajectories without encountering vanishing or exploding gradients, in agreement with our discussion in Section 8.2.2.

State and Reward Prediction

In Fig. 8.3, we evaluate our model's accuracy in state and reward prediction over long horizons involving 100 environment steps. For this experiment, we add a reward predictor and jointly minimize the reward prediction loss and the state prediction loss. We set the weight of the consistency loss in the latent space to 0.001. We see that for longer horizon prediction, our model is considerably better in predicting rewards and states accurately for `hopper`, `walker`, and `halfcheetah` in comparison to MLP, Transformer and DSSM-based model while being competitive with GRU-based model. Furthermore, Fig. 8.4 empirically verifies that our proposed Koopman dynamics model is significantly faster than an MLP, GRU, DSSM and transformer-based dynamics model. The results also reveal the trend that our model's relative speed-up over baselines grows with the length of the horizon.

8.4.2 Koopman Dynamics Model for RL and Planning

We now present promising results from integrating the diagonal Koopman model into two areas: (I) model-based planning, and (II) model-free reinforcement learning (RL). In the latter, the dynamics model enhances representation learning. Both approaches aim to solve continuous control tasks in a data-efficient manner. We provide a brief summary of these methods and their modifications with the Koopman dynamics model in Section 8.3. While the same dynamics model can also be used for (III) model-based RL, we leave that direction for future work.

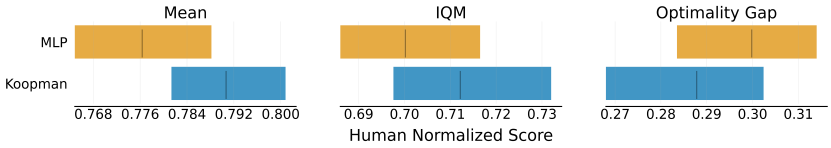


Figure 8.5: Comparison of our Koopman-based dynamics model (with a horizon of 20) and an MLP-based dynamics model of vanilla TD-MPC (Hansen et al., 2022). The results are over 5 random seeds for each environment. Higher Mean & IQM and lower Optimality Gap is better.

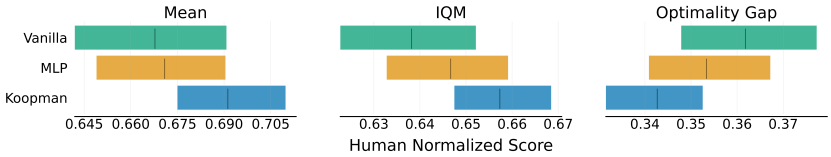


Figure 8.6: Comparison of vanilla SAC (Haarnoja et al., 2018a) and its integration with an MLP-based (SPR) and a Koopman-based dynamics model for incorporating self-predictive representations in the DeepMind Control Suite. The results are over 5 random seeds for each environment. Higher Mean & IQM and lower Optimality Gap is better.

Evaluation Metric To assess the performance of our algorithm, we calculate the average episodic return after training and normalize it with the maximum score, i.e., 1000. However, due to significant variance over different runs, relying solely on mean values may not provide reliable metrics. To address this, (Agarwal et al., 2021) suggests using bootstrapped confidence intervals (CI) with stratified sampling, which is particularly suitable for small sample sizes, such as the five runs per environment in our case. By utilizing bootstrapped CI, we obtain interval estimates indicating the range within which an algorithm's aggregate performance is believed to fall and report the Interquartile Mean (IQM). The IQM represents the mean across the middle 50% of the runs, providing a robust measure of central tendency. Furthermore, we calculate the Optimality Gap (OG), which quantifies the extent to which the algorithm falls short of achieving a normalized score of 1.0. [4]

[4]A smaller Optimality Gap indicates superior performance.

Model-Based Planning

We utilize TD-MPC (Hansen et al., 2022) as the baseline for our model-based planning approach. TD-MPC combines the benefit of a model-free approach in long horizons by learning a value function while using an MLP-based "Task-Oriented Latent Dynamics" (TOLD) model for planning over shorter horizons. To assess the effectiveness of our proposed dynamics model, we replace TOLD with our diagonal Koopman operator. Subsequently, we trained the TD-MPC agent from scratch using this modified Koopman TD-MPC approach. See Section 8.3.4 for details on this adaptation. To evaluate the performance of our variation, we conducted experiments in four distinct environments: `Quadruped Run, Quadruped Walk, Cheetah Run,` and `Acrobot Swingup`. We run experiments in the state space and compare them against vanilla TD-MPC.

Our Koopman dynamics model was trained with a horizon of 20 time steps. We observe that TD-MPC suffers from training instability, performance degradation and high variance when using 20-time steps. Consequently, we opted to use a horizon of 5-time steps for the vanilla approach. Fig. 8.5 suggests that the Koopman TD-MPC outperforms the MLP-based baseline while being more stable.

Model-Free RL

Current data-efficient model-free RL approaches like SPR;(Schwarzer et al., 2020) use dynamics modeling as an auxiliary task to improve representation learning. We apply our dynamics model to this model-free RL framework, where we use an adaptation of SPR as our baseline. This adaptation uses an MLP-based dynamics model rather than the original CNN used in SPR in order to eliminate the advantage of the original SPR for image inputs. SPR uses an auxiliary consistency loss similar to Eq. (8.7), as well as augmentations and other techniques from (Grill et al., 2020) to prevent representation collapse. It also improves the encoder's representation quality by making it invariant to the transformations that are not relevant to the dynamics using augmentation; see also (Srinivas et al., 2020; Yarats et al., 2021a). To avoid overconstraining the latent space using the dynamics model, SPR uses *projection heads*. Similarly, we encode the Koopman observables using a projection layer above the representations used for Q-learning. We provide more details on the integration of the Koopman dynamics model into the SPR framework in Section 8.3.2.

We perform experiments on five different environments from the DeepMind Control (DMC) Suite (Tunyasuvunakool et al., 2020; Tassa et al., 2018) (`Ball in Cup Catch, Cartpole Swingup`,`Cheetah Run, Finger Spin,` and `Walker Walk`) in pixel space, and compare it with an MLP-based dynamics model (SPR) and no dynamics model, i.e., SAC (Haarnoja et al., 2018a) as baselines. Figure 8.6 demonstrates that for a horizon of 5 time steps Koopman-based model outperforms the SAC plus MLP-based dynamics model baseline while being more efficient.

8.5 Related Work

Diagonal state space models (DSSMs). DSSMs (Gu et al., 2022a; Gupta et al., 2022; Mehta et al., 2022) have been shown to be an efficient alternative to Transformers for long-range sequence modeling. DSSMs with certain initializations have been shown (Gu et al., 2022a) to approximate convolution kernels derived for long-range memory using the HiPPO (Gu et al., 2020) theory. This explains why DSSMs can perform as well as structured state space models (S4) (Gu et al., 2022b), as pointed out by (Gupta et al., 2022). Inspired by the success of gating in transformers (Hua et al., 2022), (Mehta et al., 2022) introduced a gating mechanism in DSS to increase its efficiency further.

While our proposed Koopman model may look similar to a DSSM (Gu et al., 2022a; Gupta et al., 2022; Mehta et al., 2022), there are four major differences. First, our model is specifically derived for dynamics modeling with control input using Koopman theory and not for sequence or time series modeling. Our motivation is to make the dynamics and gradients through time stable, whereas, for DSSMs, it is to approximate the 1D convolution kernels derived from HiPPO (Gu et al., 2020) theory. Second, a DSSM gives a way to design a cell that is combined with non-linearity and layered to get a non-linear sequence model. In contrast, our Koopman-based model shows that simple linear latent dynamics can be sufficient to model complex non-linear dynamical systems with control. Third, DSSMs never explicitly calculate the latent states and even ignore the starting state. Our model works in the state space, where the starting state is crucial to backpropagate gradients through the state encoder. Fourth, a DSSM learns structured convolution kernels for 1D to 1D signal mapping so that for higher dimensional input, it has multiple latent dynamics models running under the hood, which are implemented as convolutions. In contrast to this, our model runs a single linear dynamics

model for any dimensional input.

Probabilistic latent spaces. Several early papers use approximate inference with linear and non-linear latent dynamics so as to provide probabilistic state representations (Haarnoja et al., 2016; Becker et al., 2019; Shaj et al., 2021; Hua et al., 2022; Fraccaro et al., 2017). A common theme in several of these articles is the use of Kalman filtering, using variational inference or MCMC, for estimating the posterior over the latent and mechanisms for disentangled latent representations. While these methods vary in their complexity, scalability, and representation power, similar to an LSTM or a GRU, they employ recurrence and, therefore, cannot be parallelized.

Koopman Theory for Control. The past work in this area has solely focused on learning the Koopman operator or discovering representations for Koopman spectrum for control (Shi & Meng, 2022; Watter et al., 2015). (Korda & Mezić, 2018) extends the Koopman operator to controlled dynamical systems by computing a finite-dimensional approximation of the operator. Furthermore, they demonstrate the use of these predictors for model predictive control (MPC). (Han et al., 2020) tasks a data-driven approach to use neural networks to represent the Koopman operator in a controlled dynamical system setting. (Kaiser et al., 2021) introduces a method using Koopman eigenfunctions for effective linear reformulation and control of nonlinear dynamical systems. (Song et al., 2021) proposes deep Koopman reinforcement learning that learns control tasks with a small amount of data. However, none of these works focus on training speed and stability for predicting longer horizons.

8.6 Discussion

Dynamics modeling has a key role to play in sequential decision-making: at training time, having access to a model can lead to sample-efficient training, and at deployment, the model can be used for planning. Koopman theory is an attractive foundation for building such a dynamics model in Reinforcement Learning. The present work shows that models derived from this theoretical foundation can be made computationally efficient. We also demonstrate how this view gives insight into the stability of learning through gradient descent in these models. Empirically, we show the remarkable effectiveness of such an approach in long-range dynamics modeling and provide some preliminary yet promising results for model-based planning and model-free RL.

Limitations and Future work Although our proposed Koopman-based dynamics model has produced promising results, our current treatment has certain limitations that motivate further investigation. Our current model is tailored for deterministic environments, overlooking stochastic dynamics modeling. We intend to expand our research in this direction, based on developments around *stochastic* Koopman theory (Mezić, 2005; Wanner & Mezic, 2022), where Koopman observables become random variables, enabling uncertainty estimation in dynamics. Secondly, although our state prediction task yielded impressive results, we have identified areas for further growth, particularly in its applications to RL and planning.

We hypothesize that the distribution shift during training and the objective change in training and evaluation hurt the performance of our Koopman dynamics model. We also plan to conduct a more comprehensive study involving a wider range of tasks and environments. Additionally, we aim to explore the compatibility of our model with different reinforcement learning algorithms to showcase its adaptability. Finally, expanding the model's application to model-based RL could unlock significant benefits. By leveraging the model's predictive capabilities to simulate future states and rewards, it's possible to enhance policy learning with fewer real-world interactions, leading to more sample-efficient algorithms.

Addressing these points would not only improve the model's performance and robustness but also broaden its applicability across different domains and settings in reinforcement learning and control tasks.

Part IV

Concluding Remarks

9

Conclusion and Future Work

This thesis leverages symmetrical and structural properties inherent in interactive environments to enhance Deep Reinforcement Learning (DRL). It provides a novel perspective by integrating the mathematical concept of symmetry groups with practical DRL applications, thus addressing fundamental challenges such as sample inefficiency and generalization across different environmental transformations.

A critical aspect discussed extensively across the chapters is the principle of *equivariance*—the ability of a model to handle changes in input without losing the essence of the information processed. This property has been effectively applied to the design of neural network architectures, enabling them to maintain consistent outputs despite transformations in input data. This approach mirrors cognitive abilities in humans, such as adaptation to solving a problem in different orientations, suggesting a cognitively inspired angle to AI agent development that could lead to more intuitive and efficient AI systems.

The first half of the thesis demonstrates how to build these equivariant models when the symmetry group and its action on the input are known. It not only underscores the role of equivariant networks in improving DRL but also explores the area of equivariant model design. The thesis provides a novel,

efficient, and scalable technique to incorporate equivariance into any neural network. Furthermore, it extends this idea to scenarios where equivariance can be built post-training, making equivariant model design more practical in the era of large foundation models. These contributions offer significant insights into enhancing the sample efficiency and generalization capabilities of existing DRL algorithms when the symmetry of the problem is known.

The second half of the thesis tackles the more challenging scenario where the symmetry group or its action on the input data is unknown. It introduces novel loss function-based techniques to learn representations that are equivariant and structured using the linear action of the symmetry group. The experiments demonstrate that this can be a useful inductive bias while training these DRL agents, extending existing self-supervised learning or world modeling techniques that augment RL objectives. Additionally, the integration of Koopman Theory for dynamic modeling represents another significant stride made by this thesis. By employing this theoretical framework, the thesis showcases how linearizing the dynamics in the representation space can predict the environment's evolutions over extended periods with improved accuracy while offering substantial benefits like stable and faster training.

In conclusion, this thesis not only deepens the understanding and practical applications of symmetry and structured representation learning within the field of Deep Reinforcement Learning but also establishes a foundational framework for Geometric Deep Learning. The implications of this research extend beyond DRL, potentially influencing the design of more efficient and generalizable AI systems across various domains. While realizing the full potential of symmetry and equivariance in machine learning remains an ongoing endeavor, this thesis provides a foundation for future researchers to build upon, potentially catalyzing significant advancements in related areas.

The work presented here opens up several avenues for future research, each promising to further the capabilities and applications of Deep Learning through the lens of symmetry and equivariance.

Enhanced Symmetry Discovery One promising direction is the development of algorithms that autonomously identify and exploit symmetries in unstructured data. In many real-world scenarios, the underlying symmetries are complex or unknown, making it challenging to design models that can effectively utilize them. Creating methods that can learn these symmetries

directly from data would greatly broaden the applicability of DRL to more complex environments where symmetries are not readily apparent. This could involve leveraging unsupervised learning techniques to discover patterns and invariances in data or employing meta-learning strategies where the model learns to recognize and exploit symmetries across different tasks. Such advancements would enhance the adaptability of DRL agents, allowing them to perform efficiently in a wider range of situations without explicit prior knowledge of the environment's structure.

Model-Agnostic Equivariance Investigating the potential of different architecture-agnostic equivariant model design methods represents another significant area for future research. Current approaches often require specialized neural network architectures to enforce equivariance, which can limit their general applicability. Developing techniques that can imbue equivariance into any model, regardless of its architecture, would make this powerful property more accessible across the field of machine learning. This could involve creating general-purpose layers or modules that can be inserted into existing networks or designing training procedures that encourage the emergence of equivariant representations. By making equivariant design more flexible and widely applicable, we could enhance the performance of models on a variety of tasks, including those in computer vision, reinforcement learning, and beyond.

Post-Training Robustness Exploring equivariant adaptation post-training at larger scales with various foundation models is another fruitful avenue. As large-scale pre-trained models become increasingly prevalent, finding ways to adapt these models to generalize to new tasks or environments without retraining from scratch is crucial. Incorporating equivariance post-training could improve generalization and robustness, allowing models to perform well even when faced with transformations not seen during initial training. This could involve developing methods to fine-tune models to respect certain symmetries or creating algorithms that adjust the representations learned by the model to be equivariant with respect to specific groups of transformations. Such research would not only enhance the utility of existing large models but also contribute to more sustainable AI practices by reducing the need for extensive retraining.

Interpretability and Explainability Further developing the connection between cognitively inspired AI and equivariant models could significantly enhance model transparency and understanding. By aligning AI systems

more closely with human cognitive processes, we can create models that are not only more efficient but also more interpretable. Equivariant models, with their structured approach to handling transformations, offer a natural avenue for such alignment. Future research could focus on visualizing how these models process and represent data, providing insights into their decision-making processes. Additionally, exploring the theoretical underpinnings of why certain symmetries lead to better performance could contribute to a deeper understanding of both machine learning models and the tasks they are applied to. This could ultimately lead to AI systems that are easier to trust and integrate into human-centric applications.

Latent Canonicalization Designing representation learning methods that can canonicalize the representations instead of the data itself presents another intriguing research direction. Canonicalization involves transforming data into a standard form, which can simplify processing and improve performance. By focusing on the latent representations within models, we can achieve canonicalization in a way that is more efficient and potentially more powerful. This approach would bridge the symmetry learning aspects of the second half of the thesis with the equivariant model design of the first half, unifying these concepts into a cohesive framework. Future work could involve developing algorithms that learn to map inputs to a canonical latent space where symmetry transformations are explicitly represented or modeled. This allows for canonicalizing data where the transformations are not explicitly defined, for example modeling 3D rotations in 2D images. This could have far-reaching implications for DRL and other areas of machine learning, enabling models to better handle variability in input data and improving generalization across different tasks and environments.

Integration with Other Learning Paradigms Finally, integrating the concepts of symmetry and equivariance with other learning paradigms such as meta-learning, transfer learning, and continual learning could open new horizons. For instance, meta-learning algorithms that can quickly adapt to new tasks might benefit from incorporating symmetry principles to generalize across transformations more effectively. Similarly, transfer learning could be enhanced by transferring not just learned features but also the symmetries and equivariances recognized by a model. Continual learning models could use symmetry and structure-aware strategies to retain and build upon knowledge without catastrophic forgetting. Exploring these integrations could lead to more robust and versatile AI systems capable of learning and adapting in

complex, dynamic environments.

By pursuing these avenues, future research can build upon the foundation laid by this thesis, pushing the boundaries of what is possible in Deep Reinforcement Learning and beyond. The continued exploration of symmetry, equivariance, and structured representation learning holds the promise of creating AI systems that are not only more efficient and generalizable but also more aligned with human cognition and reasoning.

Bibliography

Agarwal, A., Jiang, N., Kakade, S., and Sun, W. Optimistic posterior sampling and adaptive regret in stochastic bandits. *arXiv preprint arXiv:2002.12478*, 2020.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Anderson, B., Hy, T.-S., and Kondor, R. Cormorant: Covariant molecular neural networks. *arXiv preprint arXiv:1906.04015*, 2019.

Anonymous. Learning symmetric representations for equivariant world models. In *Submitted to The Tenth International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=D637S6zBRLD`. under review.

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In *International conference on machine learning*, pp. 1120–1128. PMLR, 2016.

Atzmon, M., Maron, H., and Lipman, Y. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Becker, P., Pandya, H., Gebhardt, G., Zhao, C., Taylor, C. J., and Neumann, G. Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces. In *International conference on machine learning*, pp. 544–552. PMLR, 2019.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013a.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013b.

Benton, G., Finzi, M., Izmailov, P., and Wilson, A. G. Learning invariances in neural networks from training data. *Advances in neural information processing systems*, 33:17605–17616, 2020.

Berner, C. et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Bishop, C. M. *Pattern recognition and machine learning*. Springer, 2006.

Bloem-Reddy, B. and Teh, Y. W. Probabilistic symmetries and invariant neural networks. *The Journal of Machine Learning Research*, 21(1):3535–3595, 2020.

Blondel, M., Berthet, Q., marco cuturi, Frostig, R., Hoyer, S., Llinares-López, F., Pedregosa, F., and Vert, J.-P. Efficient and modular implicit differentiation, 2022. URL `https://openreview.net/forum?id=TQ75Md-FqQp`.

Bogatskiy, A., Anderson, B., Offermann, J., Roussi, M., Miller, D., and Kondor, R. Lorentz group equivariant neural network for particle physics. In *International Conference on Machine Learning*, pp. 992–1002. PMLR, 2020.

Bogatskiy, A., Ganguly, S., Kipf, T., Kondor, R., Miller, D. W., Murnane, D., Offermann, J. T., Pettee, M., Shanahan, P., Shimmin, C., et al. Symmetry group equivariant architectures for physics. *arXiv preprint arXiv:2203.06153*, 2022.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al.

On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Brigham, E. O. *The fast Fourier transform and its applications*. Prentice-Hall, Inc., 1988.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Bronstein, M. M., Bruna, J., Cohen, T., and Veli**c**ković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021a.

Bronstein, M. M., Bruna, J., Cohen, T., and Veli**c**ković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021b.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020a.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *Advances in neural information processing systems*. NeurIPS, 2020b.

Bruder, D., Gillespie, B., Remy, C. D., and Vasudevan, R. Modeling and control of soft robots using the koopman operator and model predictive control. *arXiv preprint arXiv:1902.02827*, 2019.

Bruder, D., Fu, X., and Vasudevan, R. Advantages of bilinear koopman realizations for the modeling and control of systems with unknown dynamics. *IEEE Robotics and Automation Letters*, 6(3):4369–4376, 2021.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

Brunton, S. L., Budi**s**ić, M., Kaiser, E., and Kutz, J. N. Modern koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Carpenter, P. A. and Eisenberg, P. Mental rotation and the frame of reference in blind and sighted individuals. *Perception & Psychophysics*, 23(2):117–124, 1978. doi: 10.3758/BF03208291. URL `https://doi.org/10.3758/BF03208291`.

Caselles-Dupré, H., Garcia Ortiz, M., and Filliat, D. Symmetry-based disentangled representation learning requires interaction with environments. *Advances in Neural Information Processing Systems*, 32:4606–4615, 2019.

Celledoni, E., Ehrhardt, M. J., Etmann, C., Owren, B., Schönlieb, C.-B., and Sherry, F. Equivariant neural networks for inverse problems. *Inverse Problems*, 37(8):085006, 2021.

Challita, U., Saad, W., and Bettstetter, C. Deep reinforcement learning for interference-aware path planning of cellular-connected uavs. In *2018 IEEE International Conference on Communications (ICC)*, pp. 1–7. IEEE, 2018.

Champion, K., Lusch, B., Kutz, J. N., and Brunton, S. L. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. Shapenet: An information-rich 3d model repository. Technical report, arXiv:1512.03012 [cs.GR], 2015.

Chen, C., Li, G., Xu, R., Chen, T., Wang, M., and Lin, L. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. pp. 4994–5002, 2019.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Chen, S., Dobriban, E., and Lee, J. A group-theoretic framework for data augmentation. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F.,

and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21321–21333. Curran Associates, Inc., 2020a.

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, 2020b.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020c. URL `https://arxiv.org/abs/2002.05709`.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E. Big self-supervised models are strong semi-supervised learners. In *Advances in neural information processing systems*, volume 33, pp. 22243–22255, 2020d.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *arXiv preprint arXiv:1606.03657*, 2016.

Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In Gordon, G., Dunson, D., and Dudík, M. (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pp. 215–223. PMLR, 2011.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2019.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999, 2016a.

Cohen, T. and Welling, M. Group equivariant convolutional networks. pp. 2990–2999, 2016b.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pp. 2990–2999. PMLR, 2016c.

Cohen, T., Weiler, M., Kicanaoglu, B., and Welling, M. Gauge equivariant convolutional networks and the icosahedral CNN. In Chaudhuri, K. and

Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1321–1330, Long Beach, California, USA, 09–15 Jun 2019a. PMLR. URL http://proceedings.mlr.press/v97/cohen19d.html.

Cohen, T. S. and Welling, M. Transformation properties of learned visual representations. *arXiv preprint arXiv:1412.7659*, 2014.

Cohen, T. S. and Welling, M. Steerable CNNs. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rJQKYt5ll.

Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. Spherical CNNs. *arXiv preprint arXiv:1801.10130*, 2018.

Cohen, T. S., Geiger, M., and Weiler, M. A general theory of equivariant CNNs on homogeneous spaces. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 9145–9156. Curran Associates, Inc., 2019b. URL https://proceedings.neurips.cc/paper/2019/file/b9cfe8b6042cf759dc4c0cccb27a6737-Paper.pdf.

Coulom, R. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. Distributional reinforcement learning with quantile regression. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Dangovski, R., Jing, L., Loh, C., Han, S., Srivastava, A., Cheung, B., Agrawal, P., and Soljačić, M. Equivariant contrastive learning. *arXiv preprint arXiv:2111.00899*, 2021.

Dangovski, R., Jing, L., Loh, C., Han, S., Srivastava, A., Cheung, B., Agrawal, P., and Soljacic, M. Equivariant self-supervised learning: Encouraging equivariance in representations. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=gKLAAfiytI.

Dehmamy, N., Walters, R., Liu, Y., Wang, D., and Yu, R. Automatic symmetry discovery with lie algebra convolutional network. *Advances in Neural Information Processing Systems*, 34, 2021.

Deng, C., Litany, O., Duan, Y., Poulenard, A., Tagliasacchi, A., and Guibas, L. J. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12200–12209, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. IEEE, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Downs, T. D. Orientation statistics. *Biometrika*, 59(3):665–676, 1972.

Du, W., Zhang, H., Du, Y., Meng, Q., Chen, W., Zheng, N., Shao, B., and Liu, T.-Y. SE(3) equivariant graph neural networks with complete local frames. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, pp. 5583–5608, 2022. URL `https://proceedings.mlr.press/v162/du22e.html`.

Du, Y. and Narasimhan, K. Task-agnostic dynamics priors for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1696–1705. PMLR, 2019.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Ermolov, A., Siarohin, A., Sangineto, E., and Sebe, N. Whitening for self-supervised representation learning. In *International Conference on Machine Learning*, pp. 3015–3024. PMLR, 2021.

Esteves, C., Allen-Blanchette, C., Makadia, A., and Daniilidis, K. Learning $so(3)$ equivariant representations with spherical cnns. pp. 52–68, 2018a.

Esteves, C., Allen-Blanchette, C., Zhou, X., and Daniilidis, K. Polar transformer networks. In *International Conference on Learning Representations*, 2018b. URL `https://openreview.net/forum?id=HktRlUlAZ`.

Falorsi, L., de Haan, P., Davidson, T. R., De Cao, N., Weiler, M., Forré, P., and Cohen, T. S. Explorations in homeomorphic variational auto-encoding. *arXiv preprint arXiv:1807.04689*, 2018.

Fan, F., Yi, B., Rye, D., Shi, G., and Manchester, I. R. Learning stable koopman embeddings. In *2022 American Control Conference (ACC)*, pp. 2742–2747. IEEE, 2022.

Finkelshtein, B., Baskin, C., Maron, H., and Dym, N. A simple and universal rotation equivariant point-cloud network. *arXiv preprint arXiv:2203.01216*, 2022.

Finzi, M., Welling, M., and Wilson, A. G. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *arXiv preprint arXiv:2104.09459*, 2021.

Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. A disentangled recognition and nonlinear dynamics model for unsupervised learning. *Advances in neural information processing systems*, 30, 2017.

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., and Pineau, J. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*, 2018.

Friedman, J., Hastie, T., and Tibshirani, R. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Fuchs, F. B., Worrall, D. E., Fischer, V., and Welling, M. Se(3)-transformers: 3d roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503*, 2020a.

Fuchs, F. B., Worrall, D. E., Fischer, V., and Welling, M. Se(3)-transformers: 3d roto-translation equivariant attention networks. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, 2020b.

Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pp. 2170–2179. PMLR, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272, International Convention Centre, Sydney, Australia, 06–11 Aug 2017a. PMLR. URL `http://proceedings.mlr.press/v70/gilmer17a.html`.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272. PMLR, 2017b.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016.

Grattarola, D. Deep feature extraction for sample-efficient reinforcement learning. 2017.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Gruver, N., Finzi, M., Goldblum, M., and Wilson, A. G. The lie derivative for measuring learned equivariance. *arXiv preprint arXiv:2210.02984*, 2022.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Ré, C. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33:1474–1487, 2020.

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. 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=yJE7iQSAep.

Gu, A., Goel, K., and Re, C. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Gupta, A., Gu, A., and Berant, J. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems*, 35:22982–22994, 2022.

Haarnoja, T., Ajay, A., Levine, S., and Abbeel, P. Backprop kf: Learning discriminative deterministic state estimators. *Advances in neural information processing systems*, 29, 2016.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018a. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Haarnoja, T. et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565. PMLR, 2019b.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=S1lOTC4tDS`.

Han, Y., Hao, W., and Vaidya, U. Deep learning of koopman representation for control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 1890–1895. IEEE Press, 2020.

Hansen, N. A., Su, H., and Wang, X. Temporal difference learning for model predictive control. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/hansen22a.html`.

Hastie, T., Tibshirani, R., and Friedman, J. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, K., Gkioxari, G., Dollar, P., and Girshick, R. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Higgins, I., Amos, D., Pfau, D., Racanière, S., Matthey, L., Rezende, D. J., and Lerchner, A. Towards a definition of disentangled representations. *ArXiv*, abs/1812.02230, 2018.

Hinton, G. E. and Parsons, L. M. Frames of reference and mental imagery. *Attention and performance IX*, pp. 261–277, 1981.

Hinton, G. E., Krizhevsky, A., and Wang, S. D. Transforming auto-encoders. In *International conference on artificial neural networks*. Springer, 2011.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

Hua, W., Dai, Z., Liu, H., and Le, Q. Transformer quality in linear time. In *International Conference on Machine Learning*, pp. 9099–9117. PMLR, 2022.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Iserles, A. *A first course in the numerical analysis of differential equations*. Number 44. Cambridge university press, 2009.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJ6yPD5xg`.

Jain, A. K., Sujit, S., Joshi, S., Michalski, V., Hafner, D., and Ebrahimi Kahou, S. Learning robust dynamics through variational sparse gating. *Advances in Neural Information Processing Systems*, 35:1612–1626, 2022.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Jiang, J., Dun, C., Huang, T., and Lu, Z. Graph convolutional reinforcement learning. In *International Conference on Learning Representations*, 2018.

Jiang, X., Chen, Q., Han, S., Li, M., Dong, J., and Zhang, R. When to trust your model: Model-based policy optimization, 2020. URL `https://openreview.net/forum?id=SkgPIpcGar`. Submitted to NeurIPS 2019 Reproducibility Challenge.

Justesen, N. and Risi, S. Illuminating generalization in deep reinforcement learning through procedural level generation. *arXiv preprint arXiv:1806.10729*, 2018.

Kaba, S.-O., Mondal, A. K., Zhang, Y., Bengio, Y., and Ravanbakhsh, S. Equivariance with learned canonicalization functions. In *International Conference on Machine Learning*, pp. 15546–15566. PMLR, 2023.

Kaiser, E., Kutz, J. N., and Brunton, S. L. Data-driven discovery of koopman eigenfunctions for control. *Machine Learning: Science and Technology*, 2 (3):035023, 2021.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., and Levine, S. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019a.

Kaiser, Ł., Babaeizadeh, M., Miłos, P., Osiński, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2019b.

Khamsi, M. A. and Kirk, W. A. *An Introduction to Metric Spaces and Fixed Point Theory*. John Wiley & Sons, 2011.

Kielak, K. P. Do recent advancements in model-based deep reinforcement learning really improve data efficiency? 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pp. 2688–2697. PMLR, 2018.

Kipf, T., van der Pol, E., and Welling, M. Contrastive learning of structured world models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gax6VtDB.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., and Girshick, R. Segment anything. 2023.

Klein, F. A comparative review of recent researches in geometry. *Bulletin of the American Mathematical Society*, 2(10):215–249, 1893.

Kofinas, M., Nagaraja, N. S., and Gavves, E. Roto-translated local coordinate frames for interacting dynamical systems. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=c3RKZas9am.

Köhler, J., Klein, L., and Noé, F. Equivariant flows: sampling configurations for multi-body systems with symmetric energies. *arXiv preprint arXiv:1910.00753*, 2019.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2000.

Kondor, R. and Trivedi, S. On the generalization of equivariance and convolution in neural networks to the action of compact groups. *arXiv preprint arXiv:1802.03690*, 2018.

Kondor, R., Son, H. T., Pan, H., Anderson, B., and Trivedi, S. Covariant compositional networks for learning graphs. *arXiv preprint arXiv:1801.02144*, 2018.

Koopman, B. O. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

Koopman, B. O. and Neumann, J. v. Dynamical systems of continuous spectra. *Proceedings of the National Academy of Sciences*, 18(3):255–263, 1932.

Korda, M. and Mezić, I. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2018.03.046. URL `https://www.sciencedirect.com/science/article/pii/S000510981830133X`.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Kulkarni, T. D., Whitney, W., Kohli, P., and Tenenbaum, J. B. Deep convolutional inverse graphics network. *arXiv preprint arXiv:1503.03167*, 2015.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 473–480, 2007.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020a.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020b.

LeCun, Y., Bengio, Y., et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10): 1995, 1995.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. pp. 3744–3753, 2019.

Lee, T. Bayesian attitude estimation with the matrix fisher distribution on $so(3)$. *IEEE Transactions on Automatic Control*, 63(10):3377–3392, 2018.

Lenc, K. and Vedaldi, A. Learning covariant feature detectors. In *European conference on computer vision*, pp. 100–117. Springer, 2016.

Lenssen, J. E., Fey, M., and Libuschewski, P. Group equivariant capsule networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31, pp. 8844–8853. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/c7d0e7e2922845f3e1185d246d01365d-Paper.pdf`.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1): 1334–1373, 2016a.

Levine, S. et al. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016b.

Li, L., Walsh, T. J., and Littman, M. L. Towards a unified theory of state abstraction for mdps. *ISAIM*, 4:5, 2006.

Li, X., Li, R., Chen, G., Fu, C.-W., Cohen-Or, D., and Heng, P.-A. A rotation-invariant framework for deep point cloud analysis. *arXiv preprint arXiv:2003.07238*, 2020.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. PointCNN: Convolution on x-transformed points. pp. 820–830, 2018.

Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., Gonzalez, J., Jordan, M., and Stoica, I. Rllib: Abstractions for distributed reinforcement learning. *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In Bengio, Y. and LeCun, Y. (eds.), *ICLR*, 2016.

Lin, L.-J. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Liu, M., Yao, F., Choi, C., Sinha, A., and Ramani, K. Deep learning 3d shapes using alt-az anisotropic 2-sphere convolution. 2018.

Liu, R., Gao, J., Zhang, J., Meng, D., and Lin, Z. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10045–10067, 2021a.

Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., and Tang, J. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):677–694, 2021b.

Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Lowrey, K., Rajeswaran, A., Kakade, S., Todorov, E., and Mordatch, I. Plan online, learn offline: Efficient learning and exploration via model-based control. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Byey7n05FQ`.

Luo, S., Li, J., Guan, J., Su, Y., Cheng, C., Peng, J., and Ma, J. Equivariant point cloud analysis via learning orientations for message passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18932–18941, June 2022.

Lusch, B., Kutz, J. N., and Brunton, S. L. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

Manay, S., Cremers, D., Hong, B.-W., Yezzi, A. J., and Soatto, S. Integral invariants for shape matching. *IEEE Transactions on pattern analysis and machine intelligence*, 28(10):1602–1618, 2006.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Mauroy, A., Susuki, Y., and Mezić, I. Introduction to the koopman operator in dynamical systems and control theory. *The koopman operator in systems and control: concepts, methodologies, and applications*, pp. 3–33, 2020.

Mehta, H., Gupta, A., Cutkosky, A., and Neyshabur, B. Long range language modeling via gated state spaces. *arXiv preprint arXiv:2206.13947*, 2022.

Mezić, I. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41:309–325, 2005.

Mita, G., Filippone, M., and Michiardi, P. An identifiable double vae for disentangled representations. In *International Conference on Machine Learning*, pp. 7769–7779. PMLR, 2021.

Mitchell, T. M. *Machine learning*. McGraw-Hill, 1997.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013a.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. Playing atari with deep reinforcement learning. *CoRR*, 2013b.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015a.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. *ICML*, 2016.

Mnih, V. et al. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013c.

Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015b.

Mohlin, D., Sullivan, J., and Bianchi, G. Probabilistic orientation estimation with matrix fisher distributions. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 4884–4893. Curran Associates, Inc., 2020.

Mondal, A. K., Nair, P., and Siddiqi, K. Group equivariant deep reinforcement learning. *arXiv preprint arXiv:2007.03437*, 2020.

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.

Mondal, A. K., Panigrahi, S. S., Rajeswar, S., Siddiqi, K., and Ravanbakhsh, S. Efficient dynamics modeling in interactive environments with koopman theory. In *The Twelfth International Conference on Learning Representations*, 2023.

Mondal, A. K., Panigrahi, S. S., Kaba, O., Mudumba, S. R., and Ravanbakhsh, S. Equivariant adaptation of large pretrained models. *Advances in Neural Information Processing Systems*, 36, 2024.

Nguyen, H. H., Baisero, A., Klee, D., Wang, D., Platt, R., and Amato, C. Equivariant reinforcement learning under partial observability. In *Conference on Robot Learning*, pp. 3309–3320. PMLR, 2023.

Okada, M. and Taniguchi, T. Variational inference mpc for bayesian model-based reinforcement learning. In Kaelbling, L. P., Kragic, D., and Sugiura, K. (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 258–272. PMLR, 30 Oct–01 Nov 2020. URL `https://proceedings.mlr.press/v100/okada20a.html`.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

Popova, M., Isayev, O., and Tropsha, A. Deep reinforcement learning for de novo drug design. *Science Advances*, 4(7):eaap7885, 2018.

Poulenard, A., Rakotosaona, M.-J., Ponty, Y., and Ovsjanikov, M. Effective rotation-invariant point cnn with spherical harmonics kernels. In *IEEE International Conference on 3D Vision*, pp. 47–56, 2019.

Puny, O., Atzmon, M., Smith, E. J., Misra, I., Grover, A., Ben-Hamu, H., and Lipman, Y. Frame averaging for invariant and equivariant network design. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=zIUyj55nXR`.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017a.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. pp. 5099–5108, 2017b.

Quessard, R., Barrett, T. D., and Clements, W. R. Learning group structure and disentangled representations of dynamical environments. *arXiv preprint arXiv:2002.06991*, 2020.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Rao, Y., Lu, J., and Zhou, J. Spherical fractal convolutional neural networks for point cloud recognition. pp. 452–460, 2019.

Ravanbakhsh, S., Schneider, J., and Poczos, B. Equivariance through parameter-sharing. In *International Conference on Machine Learning*, pp. 2892–2901. PMLR, 2017.

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

Ravindran, B. and Barto, A. G. Model minimization in hierarchical reinforcement learning. In *International Symposium on Abstraction, Reformulation, and Approximation*, pp. 196–211. Springer, 2002.

Ravindran, B. and Barto, A. G. Smdp homomorphisms: an algebraic approach to abstraction in semi-markov decision processes. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pp. 1011–1016, 2003.

Ravindran, B. and Barto, A. G. An algebraic approach to abstraction in reinforcement learning. 2004.

Riba, E., Mishkin, D., Ponsa, D., Rublee, E., and Bradski, G. Kornia: an open source differentiable computer vision library for pytorch. In *Winter Conference on Applications of Computer Vision*, 2020. URL `https://arxiv.org/pdf/1910.02190.pdf`.

Rubinstein, R. Y. and Kroese, D. P. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 3856–3866. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/2cad8fa47bbef282badbb8de5374b894-Paper.pdf`.

Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*, 2021.

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

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, 2020.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. Trust region policy optimization. *CoRR, abs/1502.05477*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017a.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017b.

Schulman, J. et al. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017c.

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A. C., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2020.

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=uCQfPZwRaUu`.

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Zidek, A., Nelson, A. W. R., Bridgland, A., et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792): 706–710, 2020.

Shaj, V., Becker, P., Büchler, D., Pandya, H., van Duijkeren, N., Taylor, C. J., Hanheide, M., and Neumann, G. Action-conditional recurrent kalman networks for forward and inverse dynamics learning. In *Conference on Robot Learning*, pp. 765–781. PMLR, 2021.

Shakerinava, M. and Ravanbakhsh, S. Equivariant networks for pixelized spheres. In *International Conference on Machine Learning*, pp. 9477–9488. PMLR, 2021.

Shakerinava, M., Mondal, A. K., and Ravanbakhsh, S. Structuring representations using group invariants. *Advances in Neural Information Processing Systems*, 35:34162–34174, 2022.

Shalev-Shwartz, S. and Shashua, A. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

Shawe-Taylor, J. Building symmetries into feedforward networks. In *1989 First IEE International Conference on Artificial Neural Networks, (Conf. Publ. No. 313)*, pp. 158–162, 1989.

Shepard, N. and Metzler, J. Mental rotation of three-dimensional objects. *Science*, pp. 701–703, 1971.

Shi, H. and Meng, M. Q.-H. Deep koopman operator with control for nonlinear systems. *IEEE Robotics and Automation Letters*, 2022.

Sikchi, H., Zhou, W., and Held, D. Learning off-policy with online planning. In *5th Annual Conference on Robot Learning*, 2021. URL `https://openreview.net/forum?id=1GNV9SW95eJ`.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Driessche, G. V. D., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., and Lanctot, M. Mastering the game of go with deep neural networks and tree search. *Nature*, 550:354–359, 2017a.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017b.

Silver, D. et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Simm, G. N., Pinsler, R., Csányi, G., and Hernández-Lobato, J. M. Symmetry-aware actor-critic for 3d molecular design. *arXiv preprint arXiv:2011.12747*, 2020.

Song, L., Wang, J., and Xu, J. A data-efficient reinforcement learning method based on local koopman operators. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2021.

Srinivas, A., Laskin, M., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, 2020.

Stooke, A. and Abbeel, P. rlpyt: A research code base for deep reinforcement learning in pytorch. *arXiv preprint arXiv:1909.01500*, 2019.

Sukhbaatar, S., Fergus, R., et al. Learning multiagent communication with backpropagation. In *Advances in neural information processing systems*, pp. 2244–2252, 2016.

Sutskever, I. *Training recurrent neural networks*. University of Toronto Toronto, ON, Canada, 2013.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 2000.

Tai, K. S., Bailis, P., and Valiant, G. Equivariant transformer networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6086–6095. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/tai19a.html`.

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Tasfi, N. Pygame learning environment. *GitHub repository*, 2016.

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.

Taylor, J. Lax probabilistic bisimulation. 2008.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500): 2319–2323, 2000.

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.

Tian, Y., Krishnan, D., and Isola, P. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., and Tassa, Y. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL `https://www.sciencedirect.com/science/article/pii/S2665963820300099`.

Vadgama, S., Tomczak, J. M., and Bekkers, E. J. Kendall shape-VAE : Learning shapes in a generative framework. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*, 2022. URL `https://openreview.net/forum?id=nzh4N6kdl2G`.

van der Pol, E., Kipf, T., Oliehoek, F. A., and Welling, M. Plannable approximations to mdp homomorphisms: Equivariance under actions. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1431–1439, 2020a.

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, 2020b.

van der Pol, E., van Hoof, H., Oliehoek, F. A., and Welling, M. Multi-agent mdp homomorphic networks. *arXiv preprint arXiv:2110.04495*, 2021.

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

van Hasselt, H. P., Hessel, M., and Aslanides, J. When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32:14322–14333, 2019.

Vapnik, V. *Statistical learning theory*. Wiley-Interscience, 1998.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017a.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017b.

Villar, S., Hogg, D. W., Storey-Fisher, K., Yao, W., and Blum-Smith, B. Scalars are universal: Equivariant machine learning, structured like classical physics. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=ba27-RzNaIv`.

Vinyals, O., Bengio, S., and Kudlur, M. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations*, 2015.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

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

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019a.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38(5):1–12, 2019b.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

Wanner, M. and Mezic, I. Robust approximation of the stochastic koopman operator. *SIAM Journal on Applied Dynamical Systems*, 21(3):1930–1951, 2022.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Watter, M., Springenberg, J., Boedecker, J., and Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, 2015.

Weiler, M. and Cesa, G. General e(2)-equivariant steerable cnns. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a. URL `https://proceedings.neurips.cc/paper/2019/file/45d6637b718d0f24a237069fe41b0db4-Paper.pdf`.

Weiler, M. and Cesa, G. General e (2)-equivariant steerable cnns. *Advances in Neural Information Processing Systems*, 32, 2019b.

Weiler, M. and Cesa, G. General $e(2)$-equivariant steerable cnns. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019c.

Weiler, M., Hamprecht, F. A., and Storath, M. Learning steerable filters for rotation equivariant cnns. pp. 849–858, 2018.

Weissenbacher, M., Sinha, S., Garg, A., and Yoshinobu, K. Koopman q-learning: Offline reinforcement learning via symmetries of dynamics. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp.

23645–23667. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/weissenbacher22a.html.

Williams, G., Aldrich, A., and Theodorou, E. Model predictive path integral control using covariance variable importance sampling, 2015. URL https://arxiv.org/abs/1509.01149.

Williams, G., Drews, P., Goldfain, B., Rehg, J. M., and Theodorou, E. A. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, 2018.

Winter, R., Bertolini, M., Le, T., Noe, F., and Clevert, D.-A. Unsupervised learning of group invariant and equivariant representations. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=47lpv23LDPr.

Wood, J. and Shawe-Taylor, J. Representation theory and invariant neural networks. *Discrete applied mathematics*, 69(1-2):33–60, 1996.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5028–5037, 2017a.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Interpretable transformations with encoder-decoder networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5726–5735, 2017b.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. pp. 1912–1920, 2015.

Xie, Z., Zhang, Z., Cao, Y., Lin, Y., Wei, Y., Dai, Q., and Hu, H. On data scaling in masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10365–10374, 2023.

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*, 2021a. URL https://openreview.net/forum?id=GY6-6sTvGaf.

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*, 2021b. URL https://openreview.net/forum?id=GY6-6sTvGaf.

Yarotsky, D. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, 55(1):407–474, 2022.

Yi, B. and Manchester, I. R. On the equivalence of contraction and koopman approaches for nonlinear stability and control. *IEEE Transactions on Automatic Control*, 2023.

You, J., Liu, B., Ying, Z., Pande, V., and Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in neural information processing systems*, pp. 6410–6421, 2018.

Yüceer, C. and Oflazer, K. A rotation, scaling, and translation invariant pattern classification system. *Pattern recognition*, 26(5):687–710, 1993.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 3391–3401. Curran Associates, Inc., 2017a. URL http://papers.nips.cc/paper/6931-deep-sets.pdf.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. pp. 3391–3401, 2017b.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12104–12113, 2022.

Zhang, B., Mao, Z., Liu, W., and Liu, J. Geometric reinforcement learning for path planning of uavs. *Journal of Intelligent & Robotic Systems*, 77(2): 391–409, 2015.

Zhang, Y., Hare, J., and Prügel-Bennett, A. Overcoming the disentanglement vs reconstruction trade-off via jacobian supervision. *arXiv preprint arXiv:2002.02886*, 2020a.

Zhang, Z., Hua, B.-S., Rosen, D. W., and Yeung, S.-K. Rotation invariant convolutions for 3d point clouds deep learning. In *IEEE International Conference on 3D Vision*, pp. 204–213, 2019a.

Zhang, Z., Hua, B.-S., and Yeung, S.-K. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. pp. 1607–1616, 2019b.

Zhang, Z., Hua, B.-S., Chen, W., Tian, Y., and Yeung, S.-K. Global context aware convolutions for 3d point cloud understanding. *arXiv preprint arXiv:2008.02986*, 2020b.

Zhavoronkov, A. et al. Deep learning enables rapid identification of potent ddr1 kinase inhibitors. *Nature Biotechnology*, 37(9):1038–1040, 2019.

Zinkevich, M. and Balch, T. Symmetry in markov decision processes and its implications for single agent and multi agent learning. In *In Proceedings of the 18th International Conference on Machine Learning*. Citeseer, 2001.