Cluster-Based Generalization in Reinforcement Learning: An Analysis on Atari Games

Bachelor thesis in the department of Human Sciences by Lucas Gaston Dysli
Date of submission: May 14, 2025

1. Review: Prof. Dr. Kristian Kersting
2. Review: Cedric Derstroff
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Human Sciences
Department

Artificial Intelligence and
Machine Learning Lab

Abstract

Reinforcement learning (RL) agents often struggle to generalize their learned skills to unseen environments, significantly hindering real-world deployment. This work explores cluster-based generalization in RL by analyzing zero-shot generalization (ZSG) and transfer learning (TL) across three structurally similar Atari 2600 games: *Phoenix*, *Space Invaders*, and *Galaxian*. It further investigates how object-centric (OC) masked representations and multi-task learning (MTL) strategies influence the ability of the Proximal Policy Optimization algorithm to generalize across these games. Specifically, two OC masked representations, binary masks and multi-channel planes encoding object categories, are compared against raw pixel inputs. Training strategies include single-task baselines, random multi-task (simultaneous training on two games), and block multi-task (sequential training in alternating blocks). Experimental results demonstrate that OC representations substantially improved both generalization and final performance compared to raw pixel inputs, particularly so the planes representation. MTL strategies further amplified these effects, block multi-task training facilitated strong TL between *Galaxian* and *Phoenix* but suffered from catastrophic forgetting, while random multi-task strategies produced more general but less specialized agents. However, ZSG improvements were primarily observed for *Phoenix*, which approached average human performance when trained with *Galaxian* as a source task. Notably, inter-game transfer dynamics varied significantly: *Phoenix* benefited most from OC representations and MTL, *Galaxian* consistently emerged as the strongest source task within the cluster, providing broadly transferable skills, whereas *Space Invaders*, due to its structural simplicity and mechanical differences compared to the other two, proved least effective as a source task. This work underscores the critical role of structured input representations and learning design in enhancing generalization in RL. Additionally, it highlights the ongoing limitations of ZSG and emphasizes the need for further research in this area.

Contents

List of Figures

List of Tables

1. Introduction

Reinforcement learning (RL) [38] has emerged as a powerful tool for training agents to solve complex sequential decision-making problems, achieving remarkable, often superhuman, performance in controlled settings such as board games [34, 35, 37] and video games [21, 26, 40]. A major milestone was the breakthrough success of Deep Q-Networks (DQN) in mastering Atari 2600 games directly from raw pixel inputs, achieving human-level performance across dozens of games [24]. Another remarkable advancement was the development of AlphaGo, which was able to defeat a world champion in Go [37]. Given these breakthroughs, it is not surprising that RL generated considerable excitement within the artificial intelligence community [12] and is increasingly seen as a potential key component in the development of general AI systems [17, 19, 42].

While achieving superhuman performance in games is impressive, the ultimate ambitions of RL extend far beyond simulated worlds. There is immense potential for applying RL to a wide array of real-world challenges, such as robotics [27], business management and healthcare [19], which are only a few examples. Unlocking this potential requires the development of agents capable of acting effectively and reliably in real-world environments that are dynamic, open-ended, and unpredictable by nature [42]. But the current observation is, that agents trained in one specific environment often fail dramatically when deployed in slightly different situations [15, 16, 42], which represents a major bottleneck for practical deployment.

At the heart of this challenge lies the concept of generalization: the ability of an agent to perform well in novel situations that were not explicitly encountered during its training phase [16]. Generalization is not merely a desirable property, it is fundamental and a pivotal step forward, for creating AI systems that are truly intelligent, adaptable, and trustworthy enough for real-world deployment. Within this broader concept, zero-shot generalization (ZSG) is particularly relevant, which demands that an agent can perform in a new environment immediately upon first encountering it, without the opportunity for further adaptation or fine-tuning within that new context [16].

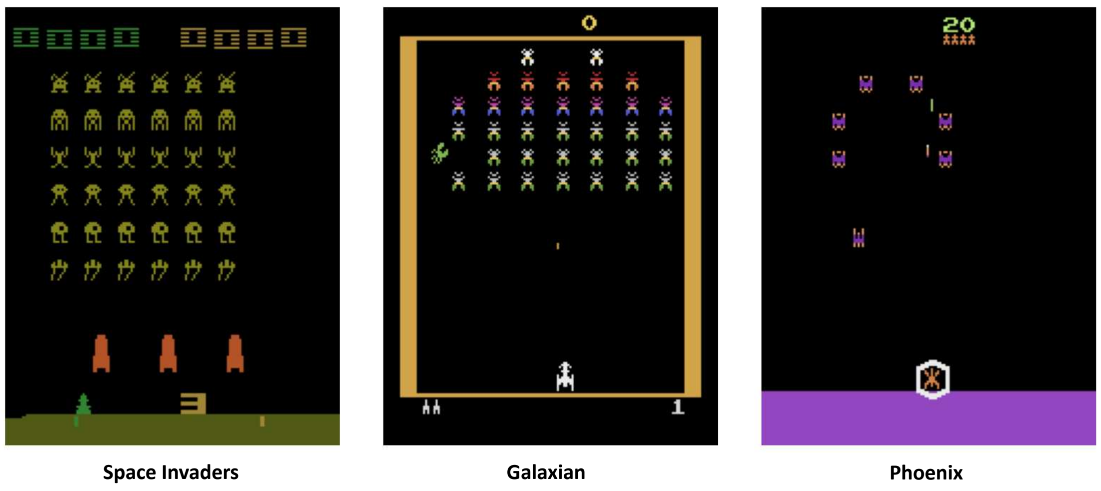

Figure 1.1.: **The three Atari 2600 games analyzed in this work.** Gameplay screenshots from the Atari 2600 games analyzed in this work: *Space Invaders*, *Galaxian*, and *Phoenix*. These games share fundamental shooting mechanics but vary distinctly in visual style and enemy dynamics, providing an ideal scenario to investigate generalization in reinforcement learning.

Another closely related concept to generalization is transfer learning (TL). In the context of RL, TL refers to the leverage of knowledge acquired from previously learned tasks or domains to improve or accelerate the learning process on a new, related task [30, 46]. Humans exhibit a remarkable capacity for TL [31], effortlessly applying skills and knowledge gained in one context to novel situations. For example, someone being proficient at skateboarding may learn snowboarding faster than somebody with no prior experience, by transferring knowledge about balance or coordination.

A significant push forward for progress in RL was the development and adoption of standardized benchmark environments [16]. These benchmarks provide a common ground to evaluate new algorithms, compare different approaches, and track cumulative progress over time [16, 21]. Among the most influential benchmarks for RL is the Arcade Learning Environment (ALE) [3], which is based on classic Atari 2600 video games. it provides agents with high-dimensional visual input (raw pixels) similar to real-world sensory data, it contains a wide range of different game tasks within a unified framework, and even human players continue to find these situations both challenging and engaging [2, 3, 21]. The ALE has been part of numerous RL breakthroughs, including the original

DQN [24], and remains the most used environment for training and testing in RL [11].

This work explores the critical challenge of generalization in reinforcement learning, focusing specifically on zero-shot generalization and transfer learning capabilities. We investigate this through cluster-based training across three similar Atari 2600 games from the ALE: *Phoenix*, *Space Invaders*, and *Galaxian*. Two primary methodologies are explored: (1) multi-task learning (MTL), which aims to enhance generalization and cross-learning by training agents on multiple tasks simultaneously or sequentially, leveraging shared representations across tasks [6, 41], and (2) object-centric (OC) state representations, which move beyond raw pixels to offer structured, symbolic information about environmental objects, drawing inspiration from cognitive science [11]. The OC representations utilized in this work are implemented via attention masking techniques, introduced by Blüml et al. [4]. Using the widely adopted Proximal Policy Optimization (PPO) [36] algorithm with convolutional policies, this study examines how MTL and OC approaches influence generalization within this controlled Atari cluster. This research contributes to the understanding of generalization in RL, which marks a significant step forward for RL and AI applications.

1.1. Problem Statement

Despite significant progress in RL, achieving robust generalization remains a primary obstacle [7, 8]. Agents trained using RL techniques often exhibit strong performance on the specific environments encountered during training, but fail to transfer this competence effectively to unseen, even closely related, environments [12]. This discrepancy between performance on the training distribution and performance on novel environments is referred to as the “generalization gap” [16]. This gap highlights a fundamental problem of generalization in RL: Agents frequently fail to learn the underlying principle governing the task. Consequently, adapting agents to slightly different conditions often requires significant retraining, hindering practical deployment and contradicting the goal of creating truly adaptable AI [16].

Key factor of poor generalization performance is overfitting to the training environment [8, 12, 16, 45]. Similar to supervised learning, overfitting in RL occurs when an agent learns patterns and correlations specific to its training experience that do not hold true in unseen situations [8]. This phenomenon has been systematically studied, demonstrating that RL agents possess the capacity to even memorize large collections of training environments, including those with randomized elements, leading to high training performance but poor test performance [8, 15, 45]. This is why overcoming overfitting remains a central challenge in generalization in RL.

In addition to overfitting, another obstacle for generalization in RL is catastrophic forgetting, which is the tendency of models to forget previously learned behaviors when trained on new tasks [14, 41]. This issue occurs when updating network parameters for new data interferes destructively with prior knowledge learned from old experiences [22]. Catastrophic forgetting is particularly relevant in RL settings where agents are exposed to multiple tasks over time, as would be expected in real-world interactions. Overcoming catastrophic forgetting is one of the biggest challenges to create artificial general intelligence (AGI) [41].

The Arcade Learning Environment (ALE) has been pivotal in advancing RL research by providing a standardized platform for evaluating domain independent AI agents [3]. Despite ALE's vital contributions, it still faces certain limitations when used to examine generalization. Firstly, the lack of structured representations in ALE's raw pixel inputs may hinder the transfer of learned policies between similar tasks [4, 11]. Secondly, the vast diversity among games can introduce confusions, potentially making it hard to isolate factors that contribute to an agent's generalization capabilities. Other studies conducted experiments using different variations of the same game within the ALE [12], which may be less relevant for real-world scenarios. To address these challenges, this work focuses on a specific cluster of games within ALE: *Space Invaders*, *Galaxian* and *Phoenix* (Figure 1.1). These games share similar core mechanics and visual structures. However, they offer variations in enemy movement patterns and specific game mechanics, making this cluster a promising controlled environment to investigate generalization.

While the general problem of generalization in RL is widely recognized, and techniques like multi-task learning (MTL) and Object-Centric (OC) representations are actively researched areas, there still remains a gap in understanding their combined effects, especially for enhancing generalization within a cluster of closely related tasks, such as the Atari shooter games selected for this work. MTL aims to improve learning by training agents on multiple tasks simultaneously or sequentially [6, 41], leveraging shared representations across tasks. On the other hand, OC representations seek to provide more abstract, structured state information compared to raw pixels, potentially facilitating better abstraction and reasoning [11]. Existing research often investigates these techniques in isolation or in different contexts and there has been limited systematic analysis of the combined application of different MTL strategies and different OC state representations on generalization. The interaction between how an agent learns over time and what information it perceives in this specific inter-game generalization setting, remains relatively unexplored.

Combining the mentioned challenges of generalization in RL, the key factor of overfitting, the role and potential of the ALE, and the specific research gap concerning the combined use of MTL and OC representations, leading to the core research problem addressed in this work:

- To what extend can enhanced training methodologies, leveraging MTL strategies and OC state representations, improve zero-shot generalization and transfer learning in RL within a structured cluster of three similar Atari games?

1.2. Objectives and Contributions

The primary objective of this work is to investigate and compare the impact of object-centric state representations with masked attention and multi-task learning strategies on the zero-shot generalization (ZSG) and transfer learning (TL) capabilities of the Proximal Policy Optimization (PPO) algorithm within the defined Atari "shoot-em-up" game cluster (*Phoenix*, *Space Invaders* and *Galaxian*).

To achieve the primary objective, this research seeks to answer the following specific research questions:

1. How effectively does a standard PPO agent, trained on pixel inputs from one game in the cluster, generalize its learned policy to other unseen games within the same cluster?
2. Does the utilization of object-centric representations with masked attention significantly improve the ZSG and TL performance of the PPO agent compared to training on raw pixel inputs?
3. How do multi-task learning strategies influence the ZSG and TL performance, particularly in combination with the different input representations?
4. Does the selected cluster of three Atari games provide a robust benchmark for evaluating generalization, and which games inside the cluster serve as optimal source/target tasks?

Based on the investigation guided by the objectives and research questions, the key contributions of this work are:

- Establishing a quantitative baseline for ZSG and TL using a standard PPO agent trained on pixel inputs within the “shoot-em-up” Atari game cluster. This serves as a concrete benchmark for evaluating inter-game generalization.
- Demonstarating the superiority of object-centric representations with masked attention over raw pixel inputs, particularly in improving final performances and ZSG performance.
- Providing a comprehensive analysis of how different multi-task learning strategies (random, and block curriculum) impact generalization, revealing that block training with specific game sequences enhances transfer capabilities between specific games the most and that random learning facilitates the development of more general agents.
- Validating the selected cluster of three Atari games (*Phoenix*, *Space Invaders*, and *Galaxian*) as a benchmark for studying generalization. Identifying *Galaxian* as the most effective source task for transfer learning and *Phoenix* as the most receptive target task, while acknowledging that *Space Invaders*’ structural differences limit its effectiveness as a source task within the cluster.

Ultimately, this work provides valuable insights regarding different input representations and varying learning approaches for addressing the critical challenge of generalization in RL, particularly within the context of closely related tasks. The findings help investigate the limitations of standard approaches and highlight promising directions for developing more robust and adaptable agents for deployment in the real-world.

1.3. Outline

The structure of this work is as follows: Chapter 2 begins with establishing the necessary theoretical foundations. Chapter 3 describes related work. Chapter 4 details the methodology, encompassing the research design, the complete experimental setup, and the methods used for data analysis and performance evaluation. Chapter 5 presents the results obtained from the experiments, comparing the performance of different approaches based on the defined metrics. Chapter 6 provides an in-depth discussion and interpretation of these results, analyzing their implications, comparing them against the related work surveyed earlier, and acknowledging the study’s limitations. Section 6.3 concludes this work by summarizing the key findings. Section 6.4 highlights future directions for research in this domain.

2. Theoretical Background

This chapter establishes the fundamental and necessary concepts essential for understanding the research presented in this work, from a basic level on.

2.1. Reinforcement Learning

Reinforcement learning (RL) represents a paradigm within artificial intelligence and machine learning focused on how agents ought to take actions in an environment to maximize a cumulative reward signal over time [38]. Unlike supervised or unsupervised learning, RL agents learn through trial and error interactions. The agent performs actions, observes the environment's response, and adjusts its behavior to achieve long-term goals.

The interaction between the agent and the environment itself is often formalized using the mathematical framework of Markov Decision Processes (MDPs). An MDP is defined by a tuple:

$$M = (S, A, p, r, \gamma) \tag{2.1}$$

Where:

- S is the set of all states the environment can possible be in. A state $s \in S$ entails all relevant information about the current situation (e.g., all positions of things).
- A is the set of all possible actions the agent can possibly take. An action $a \in A$ is what the agent can perform (e.g., moving in a direction, or shooting).
- $p : S \times A \times S \to [0, 1]$ is the state transition probability function. $p(s' \mid s, a)$ defines the probability of transitioning to state s' after taking action a in state s.
- $r : S \times A \times S \to \mathbb{R}$ is the reward function. $r(s, a, s')$ gives the immediate reward received after transitioning from state s to state s' after taking action a.

- $\gamma \in [0, 1]$ is the discount factor. It determines the present value of future rewards.

The agent's behaves according to a policy π, which dictates the action the agent will take in a given situation (state). A policy is either a mapping from states to actions ($\pi : S \rightarrow A$) or a probability distributions over actions given a state ($\pi(a \mid s)$). The objective in RL is to find an optimal policy π^* that maximizes the expected cumulative discounted reward over time, known as the return, starting from any state s.

The evaluation of policies in RL is often done via value functions. The state-value function, $V_\pi(s)$, represents the expected return starting from state s and following policy π thereafter. On the other hand, the action-value function, $Q_\pi(s, a)$ is the expected return starting from state s, taking action a, and following policy π. Finding the optimal policy often involves estimating these value functions accurately.

2.2. Deep Reinforcement Learning

Traditional RL methods, often relying on tabular representations of value functions or policies, become computationally infeasible when dealing with problems of high-dimensionality, or continuous state and action spaces. Video games like those in the Atari 2600 suite, with state spaces defined by screen pixels [24] fall under this category. Deep reinforcement learning (DRL) emerged to address this limitation by integrating deep neural networks with RL principles [1, 19].

In DRL, the underlying deep neural networks serve as way to approximate functions, which makes them capable of learning complex mappings from high-dimensional inputs (like the raw pixels seen in Atari games) to outputs (which are represented by the actions, e.g. firing). These networks can estimate the expected return from states ($V_\theta(s)$) or state-action pairs ($Q_\theta(s, a)$), or they can directly learn the mapping from states to actions or action probabilities ($\pi_\theta(a \mid s)$). Where, θ represents the parameters (weights and biases) of the neural network used, which are adjusted during the learning process, typically using variants of gradient descent.

A landmark achievement showcasing the potential of DRL was the development of the Deep Q-Network (DQN) algorithm by Mnih et al. [24], which learned to play a wide range of Atari 2600 games and achieving better than human-level performance on many of them. DQN utilized a Convolutional Neural Network (CNN) to process the visual input frames and approximate the optimal action-value function $Q_*(s, a)$ [24]. CNNs are particularly well-suited for visual data as they employ convolutional layers that can learn hierarchical spatial features (edges, textures, shapes) directly from pixel data with backpropagation [43], mirroring aspects of biological vision [20]. This ability to automatically extract relevant features from high-dimensional sensory input is a key strength of DRL and is utilized by the CNN-based policy used in this work.

2.3. Proximal Policy Optimization

Proximal Policy Optimization (PPO), which is the specific DRL algorithm employed in this work, was first introduced by Schulman et al. [36]. PPO belongs to the family of policy gradient methods, which directly optimize the parameter θ of the agent's policy π_θ to maximize the expected return, via gradient ascent.

A key innovation of PPO lies within its use of a clipped surrogate objective function. This is a mechanism constraining the policy update between a predefined range, typically $[1-\epsilon, 1+\epsilon]$, where ϵ is a small hyperparameter. This results in a new policy that does not deviate to far from the old one, preventing large and destabilizing updates. This approach strikes a balance between allowing sufficient policy improvement and maintaining training stability.

Moreover, the second major practical enhancement of PPO is the use of multiple parallel "workers" to collect data simultaneously, which is inspired by methods like Advantage Actor-Critic (A2C) [23]. The deployment of multiple parallel workers improves sample efficiency and further stabilizes learning by reducing the variance in gradient estimates through more diverse collected experiences.

In summary, PPO's design effectively balances the need for policy improvement with the necessity of maintaining training stability. Its simplicity, efficiency, and robustness made it a widely adopted algorithm for a wide range of tasks and also including better generalization competence than specialized counterparts [28].

2.4. Generalization in Reinforcement Learning

While achieving high performance within specific training environments is a prerequisite for successful RL, the true measure of an agent's intelligence and utility lies in its ability to generalize its learned knowledge to new, unseen situations. This section defines key concepts related to generalization in RL and discusses the associated challenges.

In the context of RL, generalization refers to the ability of an agent, which is trained on a specific set of environments, to perform successfully in novel environments, that were not encountered during training [16]. It marks the agents ability to real abstract learning skills and inherent knowledge gains, other than mere learning hard coded movement patters of the instances it was trained on. This ability is fundamental for deploying RL agents in the real world [13, 42], which is inherently dynamic, open-ended, and unpredictable. Agents only being able to perform well the exact conditions of their training are of limited practical use.

Within the realm of generalization in RL, two more specific concepts are especially relevant:

1. **Zero-Shot Generalization (ZSG):** This refers to the agent's ability to achieve competent performance in a new environment immediately upon deployment, without any opportunity for additional learning, adaptation, or fine-tuning within that new environment [16]. ZSG demands that the policy learned during training is directly applicable and effective in the novel context. This is a challenging but highly desirable property, especially for use in the real-world, where a theoretical infinite amount of situations can occur.

2. **Transfer Learning (TL):** In RL, TL involves leveraging knowledge acquired from training on one or more source tasks (or environments) to improve the learning process on a new, related target task [30, 46]. This improvement can manifest as faster learning, a higher final performance, or requiring less data to learn the target task effectively [46]. Unlike ZSG, TL may involve a phase of learning or fine-tuning on the target task, but the prior knowledge provides a significant advantage compared to learning the target task from scratch. Humans exhibit remarkable transfer learning capabilities, readily applying skills learned in one domain (like skateboarding) to accelerate learning in a related one (like snowboarding) [31].

The distinction between ZSG and TL is crucial for both designing experiments and interpreting results. An approach might enhance an agent's ability to adapt quickly to a new game (good TL) without necessarily improving its initial performance upon first encountering that game (poor ZSG). On the other hand, an agent might have moderate initial performance across several games (some ZSG) but struggle to significantly improve through fine-tuning (poor TL). They both are very desirable properties and the research after them will inherently make agents more capable of handling situations in the real-world.

2.5. Atari 2600 Games

Standardized benchmark environments play a crucial role in driving progress in RL by providing common ground for evaluating algorithms, comparing approaches, and measuring progress over time [16, 21]. Among the most influential and widely used benchmarks in DRL research is the Arcade Learning Environment (ALE) [3].

ALE is based on a suite of classic Atari 2600 video games. Its significance stems from several key features. Firstly, it provides agents with visual input in the form of raw screen pixels, mimicking the high-dimensional sensory data encountered in many real-world tasks. It also encompasses dozens of different games, covering a wide range of genres, objectives, and dynamics, all accessible through a unified interface, making it very promising for testing generalization. Furthermore, many Atari games remain challenging and engaging even for human players, providing complex sequential decision-making problems for AI agents [2, 3, 21]. Lastly, the games were developed independently and without the knowledge of them being used to test generalization or RL algorithms and are therefore free of any experimenters bias [21]. The ALE was instrumental in landmark DRL achievements, most notably the original DQN paper [24], and it continues to be the most commonly used platform for DRL research and evaluation [11].

The standard ALE setup still presents notable limitations for studying generalization. Learning from raw pixel inputs, although powerful, can encourage agents to pick up on superficial visual cues rather than abstract, transferable concepts, limiting skill transfer across visually different yet conceptually similar tasks [4, 11]. Moreover, the high diversity of games within the full ALE suite can make it challenging to isolate factors influencing generalization capacity between closely related tasks compared to vastly different ones [21]. Standard evaluation protocols that primarily involve training and testing on the same game, sometimes with minor variations, may test robustness more than the kind of generalization required for adaptation to fundamentally new, unseen tasks [8, 12, 21].

To mitigate some of these limitations and provide a more controlled environment for investigating generalization, this work focuses on a specific cluster of similar Atari games: *Phoenix*, *Space Invaders*, and *Galaxian*. These games belong to the “shoot-em-up” genre and share core mechanics (e.g. controlling a ship, shooting enemies, avoiding projectiles) and visual structures. However, they differ in specific enemy types, movement patterns, firing behaviors, and scoring rules. With this setup a great in-depth analysis of generalization across distinct but closely related tasks is possible.

2.6. Object-Centric Reinforcement Learning and Masked Attention

Traditional DRL approaches learn policies directly from raw sensory inputs like pixels. This can lead to representations entangled with irrelevant visual details, potentially hindering generalization [4]. Object-centric reinforcement learning (OCRL) explores an alternative approach, motivated by the observation that humans naturally perceive and reason about the world in terms of distinct objects and their interactions [17].

The core idea behind OCRL is that to provide the agent with state representations that explicitly encode information about objects. It provides information about their types, properties and relationships [11]. By structuring the input around objects, OCRL aims to enhance interpretability [10], facilitate abstraction and reasoning [11] and most notably promotes generalization [10]. These benefits make OCRL a very promising technique for overcoming some limitations of purely pixel-based DRL, particularly concerning generalization.

A significant challenge for OCRL research in complex environments like Atari was the lack of an available object-centric benchmark for evaluation, this is the reason why Delfosse et al. [11] introduced the OCAtari benchmark, by extracting symbolic object information directly from the Atari 2600's RAM state. This provides researchers with access to structured, object-based state representations alongside the standard pixel-based ones.

While OCAtari provides raw symbolic lists of object properties, subsequent work from Blüml et al. [4] was done to develop a framework that transform this information into structured spatial representations, which are suitable for input to standard neural network architectures like CNNs, by masking out pixels of non-objects. Our work utilizes two such representations:

1. **Binary Representation:** This representation renders white bounding boxes for all detected objects against a black background. This drastically simplifies the visual input by removing color and texture information, highlighting only the presence and location of objects. [4]
2. **Planes Representation:** Creates a multi-channel binary image, where each channel, or "plane" corresponds to a specific object category (e.g. enemy or player). This representation explicitly encodes both the location and the type of each object. [4]

By incorporating the structured object binary and planes representations, this work investigates, if focusing the agent on key object information, rather than raw pixels has the potential to improve generalization capabilities.

2.7. Multi-Task Learning

Multi-task learning (MTL) represents a learning paradigm where an agent learns multiple related tasks simultaneously or sequentially, leveraging shared knowledge across tasks to improve overall learning efficiency and especially generalization [6]. While single-task RL focuses on optimizing performance in isolated environments, MTL explicitly aims to discover common structures and transferable patterns between tasks and particularly lead to general knowledge [41].

At the core of MTL lies the idea that training on multiple related tasks enables the agent to develop more robust and generalizable representations [6]. By encountering a variety of scenarios, the agent is encouraged to learn abstract features that are not bound to any individual task. Another central advantages of MTL is the potential for positive transfer, which is a phenomenon where the knowledge acquired from learning one task helps the learning process in other related tasks [41].

In this work, we explore different strategies for applying MTL to a cluster of similar Atari games, including training on multiple games simultaneously and training games sequentially in a block structure. This work aims to determine whether such approaches not only enhance the agent's ability to perform well on unseen games but also facilitate the transfer of learned skills and knowledge across these similar environments.

3. Related Work

This chapter reviews existing literature related to the core challenges and method deployed in this work. It specifically focuses on the broad problem of generalization in reinforcement learning (RL), with specific attention to zero-shot generalization and transfer learning. Furthermore, we discuss the role of benchmarks, particularly the Atari 2600 Arcade Learning Environment (ALE) in assessing generalization, and examine techniques aimed at improving generalization, namely multi-task learning (MTL) and object-centric (OC) representations in this section.

3.1. Generalization in Reinforcement Learning

Early studies have shown that deep RL agents often overfit to their training environments and struggle to generalize when even minor changes are introduced [12]. For instance, Zhang et al. [44] demonstrated that agents trained in a fixed simulator have the tendency to memorize action sequences, failing to adapt when the underlying dynamics change. However, they also found that injecting randomness during training significantly improved robustness in altered test conditions [44]. Similarly, Farebrother et al. [12] examined DQN agents on Atari games and revealed that they often overspecialize to the exact training conditions. They also demonstrated that adding regularization techniques from supervised learning, such as dropout and L^2 weight decay, helped DQN learn more generic features, resulting in better scores on unseen levels [12].

To quantify the issue of generalization, Cobbe et al. [8] introduced the CoinRun benchmark with procedurally generated levels, explicitly separating training and test environments. They reported that standard RL algorithms achieved high scores on training levels but generalized poorly to new ones. Interestingly, they also observed that increasing the size of neural networks only improved generalization up to a point, after which additional capacity showed diminishing returns. [8]

Building on top of this, Laskin et al. [18] proposed RAD (Reinforcement Learning with Augmented Data), where small changes to the environment (e.g. jittering) were applied during training. They found that even simple environment augmentation consistently improved generalization more effectively than some specialized algorithms [18].

Despite these major improvements in generalization in RL, fundamental challenges remain. Many techniques provide only incremental gains and often depend on extensive hyperparameter tuning or environment stochasticity, which may not be practical in all settings. Notably, Cobbe et al. [8] observed that simply scaling up network size does not guarantee better generalization. Furthermore, Packer et al. [28] showed that specialized algorithms do not inherently outperform simpler models in terms of generalization. These findings underscore a persistent issue: deep RL agents frequently exploit arbitrary features of their training environments rather than acquiring robust, transferable knowledge and skills. This work addresses this gap by investigating training on similar environments with OC representations and differing MTL strategies, we aim to uncover how shared structure between related games can support the development of more transferable knowledge of the agent.

3.1.1. Zero-Shot Generalization

Zero-shot generalization (ZSG) in RL refers to an agent's ability to perform effectively in novel, unseen environments without additional training [16]. This capability is crucial for deploying RL agents in real-world scenarios, where encountering unfamiliar situations is inevitable.

The ProcGen benchmark, introduced by Cobbe et al. [7], provides a suite of procedurally generated environments designed to evaluate ZSG. It provides 16 procedurally generated games where each episode is a randomly generated level, and ZSG is accessed on held-out levels. Experiments on ProcGen were also conducted with PPO and have shown that increasing diversity during training (more levels) improves test performance, yet there remains a noticeable gap between training and test scores, underscoring the persistent challenge of ZSG. [7]

Achieving ZSG is inherently challenging due to the tendency of RL agents to overfit to their training environments. As highlighted by Kirk et al. [16], procedural content generation (PCG) alone may not suffice for true ZSG. Agents trained solely on procedurally generated levels can still exploit regularities specific to the training distribution, leading to poor performance on unseen levels. This phenomenon underscores the need for strategies beyond PCG to enhance generalization capabilities. [16]

Building upon this, we propose to access ZSG with a different strategy: clustering tasks that share core mechanics and visual structure and are not just a procedually generated version of different levels of the same game. We hypothesize that training across such semantically aligned games will encourage the agent to extract higher-level knowledge, rather than superficial cues.

3.1.2. Transfer Learning

Transfer learning (TL) in RL enables agents to leverage knowledge from one or more source tasks to enhance learning efficiency and performance on a different target task [30, 46]. Zhu et al. [46] provide a comprehensive survey categorizing TL approaches in deep RL, highlighting methods such as inter-task mapping, which involves creating mappings between different task domains to facilitate knowledge transfer. Furthermore, Pan et al. [29] also did a comprehensive study and especially link TL to MTL, both techniques investigated in this work.

Parisotto et al. [30] introduced the Actor-Mimic method, which employs DRL and model compression techniques to train a single policy network capable of handling multiple tasks. They also used the ALE to assess TL capabilities. The network was trained on 20 distinct games, and they observed that pretraining increased learning speed in the target domain. This effect was particularly evident when the training incorporated games with similar mechanics. However, they also noted that excessively long pretraining could lead to overfitting, potentially hindering TL performance. [30]
These findings support the notion that games with similar mechanics can facilitate positive knowledge transfer, a concept further explored in our work.

Nichol et al. [25] introduced a benchmark to evaluate TL in RL. Their benchmark leverages a diverse set of levels from three games of the same franchise. These levels share core mechanics but vary in layout, obstacles, and visual themes, making it bear great potential to test TL capabilities. Baseline experiments with standard RL algorithms revealed that training on these diverse levels, and a fine-tuning phase of 1M steps in the test environment yields moderate improvements over learning from scratch. The authors emphasize that, while their benchmark marks an important step toward measuring generalization in RL, significant challenges remain particular in TL and encourage further research. [25]

3.2. Atari 2600 as a Benchmark

The Arcade Learning Environment (ALE), introduced by Bellemare et al. [3], was pivotal in the development of deep RL. However, early uses of the ALE, notably Mnih et al. [24] treated each game as a separate training task, where an agent was trained and tested on the same game. Another example is highlighted by Badia et al. [2], they developed Agent57, an algorithm which achieved human level performance on 57 Atari games, but was not tested beyond the training distribution.

Over time, researchers realized that high scores on the training game did not equate to a generally capable agent, which prompted efforts to make evaluation in ALE more rigorous with respect to generalization. Machado et al. [21] conducted notable work and revisited the ALE and highlighted the need for consistent evaluation protocols. They added alternate difficulty levels and game variations to the ALE to better analyze generalization and TL. Furthermore, they proposed using sticky actions to introduce randomness, to create more robust agents. [21]

Most recently, Delfosse et al. [9] introduced HackAtari, a modified ALE framework that enables systematically injecting controlled novelties into Atari games. They found that their framework enables more robust RL algorithms to emerge [9]. In this work, ALE is utilized in a new manner to test generalization and TL capabilities. By incorporating MTL, OC inputs, and clustering similar games, this study extends ALE's utility even further.

3.3. Multi-Task Learning and Object-Centric Representations

Object-centric (OC) representations have emerged as a promising approach to improve generalization in RL, motivated by the intuition that agents should reason about high-level entities rather than raw pixels. Delfosse et al. [11] introduced OCAtari, a framework to enable object-centric RL, by extracting OC states from Atari 2600 games by analyzing RAM states or visual features, enabling precise object detection. They demonstrated that agents trained with OC representations achieve performance comparable to or better than pixel-based agents on a set of 8 Atari games [11].

Building upon this, Blüml et al. [4] introduced OCCAM (Object-Centric Attention via Masking), which applies attention mechanisms to filter out irrelevant background features, and thereby highlighting task relevant objects in Atari games. Their empirical evaluations demonstrated that agents trained with OCCAM exhibited improved robustness to unseen visual perturbations and required fewer training samples to achieve competitive performance compared to agents trained on raw pixel inputs. [4]

In parallel, multi-task learning (MTL) explores training agents on multiple tasks simultaneously or alternately to learn shared representations that generalize across tasks. Taïga et al. [39] systematically investigated MTL pretraining using procedurally generated Atari 2600 game variants. They found that, agents pretrained on more game variations generalize better [39]. Another very interesting work was done by Rusu et al. [33], they proposed progressive neural networks, which also deployed sequential learning, but for each new task, a new neural network column is added to the architecture. Also layers of old experiences are just frozen and not updated, making them immune to catastrophic forgetting. Their results showed superior performance compared to standard pretraining and fine-tuning approaches, highlighting the efficacy of this approach in TL scenarios. [33]

While both OC representations and MTL have shown promise in enhancing generalization, further research is needed to explore the combination of these approaches. This work aims to investigate the integration of OC attention mechanisms, more precisely OCCAM [4], within MTL approaches to develop RL agents capable of robust generalization across multiple environments.

4. Methodology

This chapter entails the experimental methodology employed in this work, to investigate the impact of object-centric representations with attention masking and multi-task learning strategies on the generalization capabilities of reinforcement learning agents within a cluster of similar Atari games. Firstly, we will outline the research design, then the specific environments and agent configurations, the different training protocols, and finally the evaluation metrics used to assess zero-shot generalization (ZSG) and transfer learning (TL) performance,

4.1. Research Design

A comparative experimental design is used to evaluate the effects of three different input representations in combination with three different training strategies on generalization capabilities of the Proximal Policy Optimization (PPO) [36] algorithm, on a cluster of similar Atari 2600 games, namely: *Phoenix*, *Space Invaders*, and *Galaxian*. The input representations, which define the format of the observation provided to the agent (see Figure 4.1), are as follows:

- *Pixel-based:* Using the raw $84{\times}84$ grayscale pixel frames.
- *Binary* [4]: A black and white $84{\times}84$ visual representation, displaying white bounding boxes for all detected objects against a black background.
- *Planes* [4]: A structured $84{\times}84$ multi-channel binary representation, encoding both the location and the semantic category of detected objects by mapping predefined, conceptually similar object types (e.g., all enemy projectiles) onto distinct, consistent binary image planes across the different games in the cluster (a detailed list of the mapping can be found in Table 4.2).

The training strategies to compare how the agent interacts with and learns from the environments during the training phase are as follows:

- *Single-Task Baseline:* Involves training a single agent individually on each game. This establishes the baseline performance for each input representation type within a single task.
- *Random Multi-Task:* Covers the training strategy of simultaneously training on a combined set of parallel environments drawn from two different games within the cluster.
- *Block Multi-Task:* Introduces sequential training in alternating blocks.

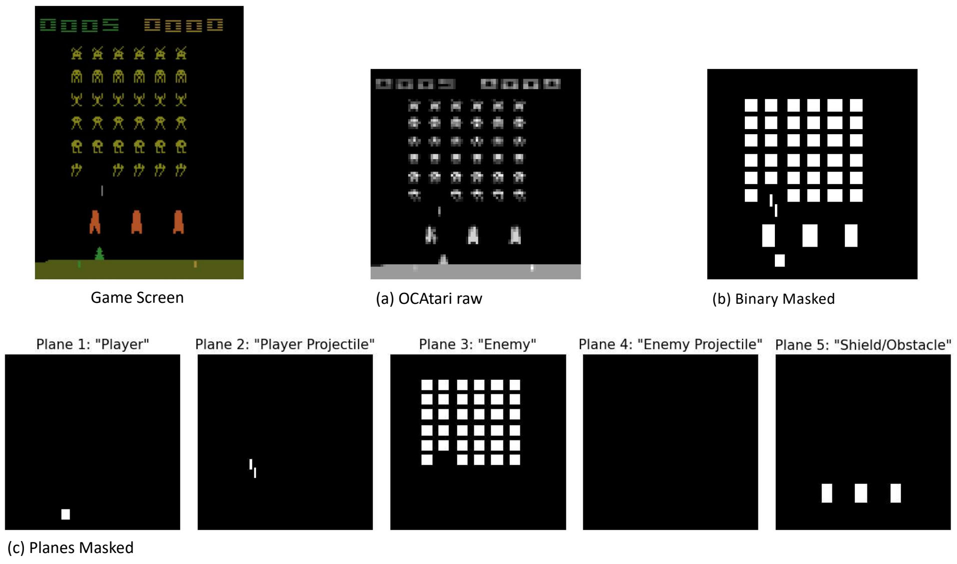

Figure 4.1.: **Comparison of input representations for Space Invaders.** (a) Raw visual output from OCAtari (b) Binary masked representation showing all detected objects as white bounding boxes (c) Planes masked representation decomposed into distinct channels, each corresponding to a semantic object category (Player, Player Projectile, Enemy, Enemy Projectile, Shield/Obstacle)

4.2. Experimental Setup

This section provides the specific technical details concerning the RL agent, the configuration of the game environments, the implementation of the different input representations, and the execution of the training procedures.

4.2.1. Environment Configuration

The experiments utilize the three Atari 2600 games accessed through the Arcade Learning Environment (ALE) [3], using the `gymnasium` library [5]. Each training environment is built using key hyperparameters found in Table 4.1. It is important to note, that evaluation environments are built with the same setup, except for not utilizing *EpisodicLifeEnv* and *ClipRewardEnv*, providing evaluation based on the true game score and structure.

Table 4.1.: Key hyperparameters for environment setup. Top part of table entails the versions of core libraries used. The lower one outlines observation handling, and common preprocessing steps.

Hyperparameter	**Value**	**Hyperparameter**	**Value**
ALE-py [3]	0.8.1	Gymnasium [5]	0.29.1
Environment version	v5	Stable-Baselines3 [32]	2.3.0
OCAtari [11]	2.1.0	OCCAM [4]	0.0.1
Observation Type	DQN/RAM	No-op Reset Max	30
Repeat Action Prob.	0.25	Fire Reset Applied	Yes (if applicable)
Buffer Window Size	4	Clip Reward (Train)	True
Frame Stack	4	Episodic Life (Train)	True
Full Action Space	True		

Object-Category Mapping (Planes Representation)

For the planes representation, a custom wrapper was implemented to ensure consistent semantic mapping of objects across the three games into a fixed set of 5 channels (planes). This mapping is crucial for the agent to potentially learn generalizable features related to object roles rather than game specific appearances. The mapping used can be found in Table 4.2.

Table 4.2.: Conceptual planes and associated object types across Atari games

Conceptual Plane (Type)	**Space Invaders**	**Galaxian**	**Phoenix**
Plane 0: Player	Player	Player	Player
Plane 1: Player Projectile	Bullet	PlayerMissile	Player_Projectile
Plane 2: Enemy / Alien / Target	Alien, Satellite	Alien, EnemyShip, DivingEnemy	Phoenix, Bat, Boss
Plane 3: Enemy Projectile	—	EnemyMissile	Enemy_Projectile
Plane 4: Shield / Obstacle	Shield	Shield	Shield, Boss_Block_Green, Boss_Block_Blue, Boss_Block_Red

4.2.2. Agent Configuration

The learning algorithm analyzed is Proximal Policy Optimization (PPO) [36], with a standard Convolutional Neural Network (CNN) policy architecture for all experiments. The implementation of the whole algorithm and network policy is from Stable-Baselines3 (SB3) [32]. Moreover, the hyperparameters were selected based on common practices for Atari environments. These hyperparameters were kept consistent across all experiments, except for the *multi-task random* structure. The number of parallel environments was set to 8 per game (resulting in 16 total parallel environments) and the batch size was increased to 512, to ensure same conditions per game. In Table 4.3 a detailed overview of the hyperparameters is provided with the corresponding values used.

4.2.3. Training Procedure

To ensure statistical significance and account for the inherent stochasticity in DRL training, each experimental condition is executed 5 times using random seeds. Each full training run consists of environment steps as follows, depending on the training structure:

- *Single-Task Baseline:* A single continuous run of 10M steps on one game.

Table 4.3.: Core PPO hyperparameters. *BL:* Single-Task Baseline, *BC:* Block curriculum and *RC:* Random curriculum

Hyperparameter	**Value**	**Hyperparameter**	**Value**
LR	0.00025	LR Schedule	Linear decay
Rollout Buffer Size	1024(BL, BC) 2048(RC)	Batch Size	256(BL, BC) 512(RC)
Timesteps/Game	10M	Discount Factor	0.9
Epochs	4	Entropy Coefficient	0.01
Value Function Coefficient	0.5	Steps per Update	128
Parallel Environments	8(BL, BC), 16(RC)	Clipping Coefficient	0.2
Max Gradient Norm	0.5	GAE Lambda	0.95

- *Random Curriculum:* A single continuous run of 20M total steps, sampling randomly from two games (i.e. 10M training steps per game).
- *Block Curriculum:* A total of 20M steps split into two blocks of training (each 10M), alternating between the two training games in the block (each game 5M per block, total of 10M at the end of training).

4.3. Evaluation Metrics & Protocols

The evaluation protocol is designed primarily to assess generalization in RL, more specifically ZSG and TL performances. Evaluation is performed every 30000 steps for 5 full episodes. The agent's actions are chosen deterministically during evaluation to measure the policy's direct performance without exploration noise. The evaluation environments cover always all three games in the cluster, regardless of which game(s) are used for training in that run. For each game in the evaluation set, the following metrics are calculated over the 5 episodes: *mean episodic return*, *standard deviation of episodic return* and *mean episode length*.

The performance achieved in the single-task baseline runs serves as the critical reference point. The effectiveness of OC representations and MTL strategies are judged by their ability to improve performance. **Measuring ZSG:** ZSG was directly assessed by tracking the mean episodic return on the game(s) that are held-out during a specific run. Higher and more stable returns on these held-out games indicate better ZSG. The primary analysis involves comparing these ZSG curves across the different input representations and training strategies. **Measuring TL:** In block MTL, observing if performance on the second game improves faster or reaches a higher peak in later blocks compared to the baseline performance for that game suggests positive transfer. In random MTL, comparing the learning speed and final performance on the training games against their respective baseline runs can indicate benefits derived from shared representations learned through simultaneous training.

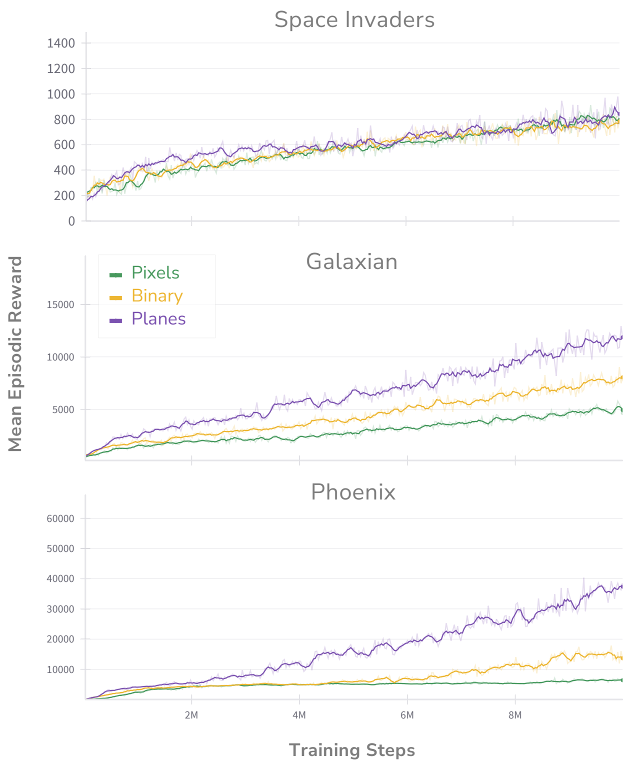

Figure 4.2.: **Baseline performances.** Comparison of the three Atari games: *Space Invaders*, *Galaxian*, and *Phoenix*, where each game was trained separately from scratch using three input modalities: *pixel (green)*, *binary (yellow)*, and *planes (purple)*. The figure compares the training steps with the mean episode reward averaged over 5 evaluation episodes. While the choice of input representation made little difference in *Space Invaders*, huge performance gaps emerged in the other two games. In both cases, the planes input led to significantly higher reward outcome, particularly in *Phoenix*.

5. Results

We set out to investigate cluster-based generalization in reinforcement learning, with a focus on similar Atari games: *Phoenix*, *Space Invaders*, and *Galaxian*. The experiments were conducted using the Proximal Policy Optimization [36] algorithm to assess zero-shot generalization (ZSG) and transfer learning (TL) performance with varying input representations (pixel based, binary representations and planes) and multi-task learning (MTL) strategies (random order, block structure). Training process was always 10 million steps per game, resulting in total of 20 million for MTL setups. It is also important to note that all three games have different scoring rules, so the absolute scores vary significantly. As a reference point in the ZSG tables, we include a random agent that selects actions uniformly at random, serving as a lower bound for expected performance, alongside the average human score where available.

5.1. Baseline Performance

Object-centric input, particularly planes, significantly improved baseline performance on Galaxian and Phoenix, while it had minimal effect on Space Invaders. Figure 4.2 illustrates the baseline performances when each game was trained individually from scratch using the three different input representations. These baseline results serve as a crucial reference point for following TL and ZSG evaluations. The outcome of the different input representations did not differ remarkably in *Space Invaders*. For *Galaxian* and *Phoenix*, differences between the input representations were observed. In both games, the planes input achieved the highest mean rewards, followed by the binary representation, and finally the pixel-based input. The planes representation reached substantially higher reward values, particularly in *Phoenix*, where it achieved more than three times the final mean reward compared to the other inputs. In *Galaxian*, the planes input also showed a clear advantage in performance over the other representations.

Table 5.1.: Final performance on **Space Invaders** after 10M/20M training steps (mean episode reward ± standard deviation).

	Training	Pixels	Binary	Planes
BL	SI	**794.3** (±167.7)	784.0 (±284.5)	841.7 (±314.8)
BC	G → SI	667.8 (±137.1)	707.6 (±202.3)	**974.8** (±415.4)
	SI → G	445.2 (±129.0)	456.0 (±142.1)	566.5 (±128.9)
	P → SI	615.4 (± 88.6)	665.4 (±143.0)	955.3 (±308.2)
	SI → P	145.8 (± 65.7)	378.4 (±157.5)	228.0 (±124.0)
RC	SI + G	618.2 (± 90.3)	687.2 (±204.2)	716.3 (±204.2)
	SI + P	591.6 (±120.0)	**818.6** (±241.2)	824.5 (±241.2)

Bold = best within each representation; Blue shading = Best within the training curriculum. *BL* = Baseline; *BC* = Block curriculum (sequential); *RC* = Random curriculum. $SI \to P$ = Space Invaders trained first, followed by Phoenix under BC; $SI + G$ = Space Invaders and Galaxian trained together under RC.

5.2. Transfer Learning Outcomes

Space Invaders final performance shows only minor differences between training methods. Table 5.1 presents the final scores on *Space Invaders* for each training approach and input representation. When using pixel inputs, the baseline condition achieved the highest score. For binary inputs, the random learning structure slightly outperformed the baseline. The planes representation generally yielded the best results across curricula, with the highest overall score observed in the block learning condition where *Galaxian* was trained first and *Space Invaders* last. Overall, the score differences were relatively small and broadly comparable. A notable exception was the significant drop in performance when *Phoenix* was the second game trained, compared to when *Galaxian* was used.

Baseline performance marks the best outcome across all input types for Galaxian, and in MTL Galaxian benefits most from pairing with Phoenix. Table 5.2 shows that the baseline conditions achieved the highest performance for all three input representations. Binary inputs consistently outperformed pixel inputs in nearly all conditions, and planes representation substantially outperformed both in most cases. MTL generally resulted in lower performance compared to the baseline, with the best performing cross-game condition was block learning when *Phoenix* was trained first and *Galaxian* was trained last, but still being approximately 10% lower compared to the baseline. Block learning

Table 5.2.: Final performance on **Galaxian** after 10M/20M training steps (mean episode reward $\pm$ standard deviation).

	Training	Pixels	Binary	Planes
BL	G	**4923.0** (±1347.6)	**8058.4** (±2718.4)	**11877.8** (±3551.9)
BC	P → G	4497.2 (± 962.6)	6781.8 (±2357.6)	10712.6 (±3914.4)
	G → P	473.6 (± 171.0)	4378.8 (±1404.3)	5322.5 (±2008.2)
	SI → G	3772.0 (±1333.8)	5425.2 (±1250.9)	9327.4 (±2921.4)
	G → SI	1956.0 (± 800.9)	4121.0 (±1217.2)	3619.5 (±1196.5)
RC	G + P	4363.2 (± 770.0)	7192.4 (±2005.5)	9126.8 (±3147.5)
	G + SI	4117.6 (± 665.8)	6692.8 (±1625.1)	8394.3 (±2402.3)

Bold = best within each representation; Blue shading = Best within the training curriculum. *BL* = Baseline; *BC* = Block curriculum (sequential); *RC* = Random curriculum. $G \rightarrow P$ = Galaxian trained first, followed by Phoenix under BC; $G + SI$ = Galaxian and Space Invaders trained together under RC.

generally outperformed random learning. Notably, training together with *Phoenix* led to better transfer performance on *Galaxian* than using *Space Invaders*, across both MTL setups.

In Phoenix, MTL consistently outperforms the baseline, with training alongside Galaxian providing the highest performance gains. Table 5.3 displays that, similar to the other two games, the planes representation achieved the highest performance across all training curricula. Notably, the single-task baseline using planes yielded better results than all conditions using the other two input representations. Additionally, in direct comparison of pixels and binary conditions, binary was superior in every condition except for block curriculum with *Phoenix* first and *Space Invaders* last. However, unlike in *Galaxian*, at least one MTL configuration outperformed the single-task baseline in *Phoenix*. The best result was observed in the block curriculum condition using planes, when trained sequentially on *Galaxian*, which achieved the highest overall mean score. Furthermore, training *Phoenix* together with *Galaxian* consistently produced higher scores than pairing it with *Space Invaders*.

Consistent patterns across all three games in transfer learning. Across all three games *Space Invaders* (Table 5.1), *Galaxian* (Table 5.2), and *Phoenix* (Table 5.3) performance variability, as indicated by the standard deviation, tended to increase with higher performance scores. In all cases, the planes input achieved the highest performance

Table 5.3.: Final performance on **Phoenix** after 10M/20M training steps (mean episode reward ± standard deviation).

	Training	Pixels	Binary	Planes
BL	P	6394.1 (±1001.6)	13678.1 (±4430.9)	37361.4 (±10503.2)
BC	G → P	5784.0 (±1367.4)	**24567.6** (±10321.2)	**40279.5** (±13977.7)
	P → G	1550.8 (±1197.3)	3993.0 (± 818.2)	10152.2 (± 3899.3)
	SI → P	5054.8 (± 364.7)	13421.8 (± 3264.7)	27075.9 (± 6896.1)
	P → SI	3952.8 (± 918.8)	1444.8 (± 864.1)	6456.0 (± 2392.0)
RC	P + G	**7898.4** (±3108.9)	12372.4 (±2973.1)	28771.2 (± 9664.4)
	P + SI	5324.0 (± 262.4)	16420.0 (±3408.0)	23413.8 (± 8203.3)

Bold = best within each representation; Blue shading = Best within the training curriculum. *BL* = Baseline; *BC* = Block curriculum (sequential); *RC* = Random curriculum. $SI \to P$ = Space Invaders trained first, followed by Phoenix under BC; $P + G$ = Phoenix and Galaxian trained together under RC.

across individual training curricula and marked the best overall result in each game. The structure of the learning order in block curriculum played an important role: when the target game was last in the block, performance was consistently higher compared to when it was first in the block.

Block curriculum training yields the highest final reward on Phoenix. Figure 5.1 presents the training progression for *Phoenix* under different curricula using planes input. The single-task baseline (green) steadily increases over 10 million steps, while the block curriculum (BC) conditions—*Phoenix* followed by *Galaxian* (yellow) and *Galaxian* followed by *Phoenix* (orange)—extend to 20 million steps. In the BC curves, a performance drop is visible when switching from the first to the second task. Despite this, the orange BC condition achieves the highest final episode reward among all strategies. The random curriculum (RC, purple), which interleaves the two tasks, shows smoother but lower performance across the full training horizon, ending below the better BC condition.

Overall, the effectiveness of MTL varied considerably across the three games, and block learning is mostly equal or better than random learning order. As shown in Figure 5.2, which compares the best performing condition within each input representation cluster, the performance patterns differ notably between games. For *Space Invaders*, the final scores did not vary substantially across input types, with relatively minor differences between training strategies. In contrast, both *Galaxian* and *Phoenix* show a clear order of

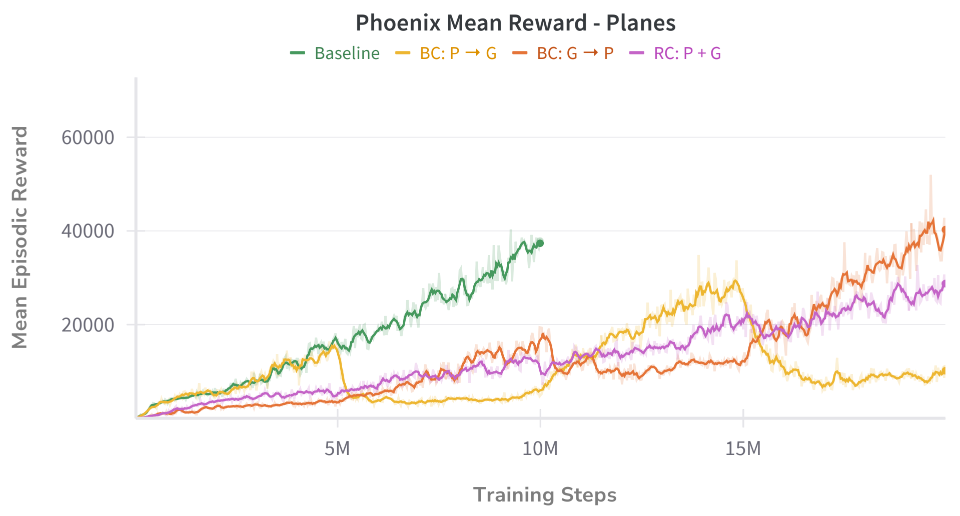

Figure 5.1.: **Comparison of baseline, random curriculum and block curriculum final performances on Phoenix.** Training curves for *Phoenix* showing mean episode reward over training steps, using the object-centric planes input representation, averaged across 5 evaluations. The plot compares single-task baseline training (green), block curriculum (BC) with two task orders—*Phoenix* followed by *Galaxian* (yellow) and *Galaxian* followed by *Phoenix* (orange), and random curriculum (RC) using a mixed schedule of *Phoenix* and *Galaxian* (purple). The figure highlights: (i) stark drops when the game switches in BC (ii) BC leads to higher final performance than BL and RC, when *Phoenix* is trained last.

input performances, where binary inputs consistently outperformed pixels, and planes consistently outperformed both. Among the three games, *Phoenix* benefited most from MTL setups, with all multi-training conditions exceeding the single-task baseline, particularly when paired with *Galaxian*. *Space Invaders* showed smaller gains, with the baseline performing best in the pixel setting, but MTL yielding stronger results for binary and planes inputs. *Galaxian*, on the other hand, showed consistently strong baseline performance across all input types. Finally, block curriculum was generally more effective than random curriculum: block curriculum matched or outperformed the random condition in eight out of nine representation clusters, and produced the best cross-game result in each group.

Galaxian emerged as the most effective source task, while Space Invaders proved the least effective. Figure 5.2 further revealed differences in the effectiveness of the three games as source tasks within the cluster. *Galaxian* consistently facilitated positive transfer to both *Phoenix* and *Space Invaders*, especially in block curriculum setups, where it provided the strongest support in all tested conditions. Even in random learning experiments, it was the most successful source task in half of the cases. In contrast, *Space Invaders* emerged as the least effective source task, contributing to the best MTL outcome in only one out of 12 conditions.

5.3. Zero-Shot Generalization Performance

Overall very low ZSG performance with Space Invaders as the unseen game. When *Space Invaders* was held out during training, as seen in Table 5.4, performance was generally very low across all conditions. The highest mean ZSG return was achieved with pixel-based input trained solely on *Galaxian*. This result was only marginally above random performance. Furthermore, no combination was close to approach an average human score. Binary representation was consistently better within MTL, while planes performed particularly poorly in ZSG for this game.

Table 5.4.: Zero-shot generalization results on **Space Invaders**. Results are final mean episode reward (± standard deviation).

	Training	Pixels	Binary	Planes
BL	G	**241.9** (±102.7)	119.4 (±47.9)	166.6 (±51.0)
	P	196.1 (± 59.9)	161.5 (±93.3)	113.4 (±64.3)
BC	G → P	160.0 (±101.8)	221.2 (±149.8)	68.2 (±22.0)
	P → G	157.4 (± 56.8)	178.6 (± 96.7)	34.0 (± 6.8)
RC	P + G	75.8 (±14.4)	**234.6** (±51.6)	**188.8** (±51.6)
Human Avg.	-	*1668.70*		
Rand. Agent	-	*150.3 (±45.6)*		

Bold = best within each representation; Blue shading = Best within the training curriculum. *BL* = Baseline; *BC* = Block curriculum (sequential); *RC* = Random curriculum. $G \rightarrow P$ = Galaxian trained first, followed by Phoenix under BC; $G + P$ = Galaxian and Phoenix trained together under RC. Gray shaded area = Human average [2] and random agent scores on Space Invaders for comparison.

Moderate ZSG performance when Galaxian was the held-out game. Table 5.5

shows the ZSG performance when *Galaxian* was the unseen game. Every representation outperformed the random baseline at least once. Planes achieved the highest ZSG score under block learning when *Phoenix* was trained first and followed by *Space Invaders*. However, not a single condition approached performances observed in transfer learning contexts. Unfortunately, an average human score is not available for *Galaxian*.

Table 5.5.: Zero-shot generalization results on **Galaxian**. Results are final mean episode reward (± standard deviation).

	Training	Pixels	Binary	Planes
BL	SI	668.6 (±220.4)	**799.6** (±248.3)	578.2 (±163.2)
	P	602.2 (±143.8)	628.1 (±209.8)	814.8 (±190.8)
BC	SI → P	768.0 (±307.1)	688.2 (±245.7)	582.4 (±110.2)
	P → SI	593.6 (±276.6)	482.0 (±147.4)	**927.2** (±206.1)
RC	SI + P	**866.0** (±289.5)	608.0 (± 99.3)	488.3 (±159.3)
Human Avg.	-	—		
Rand. Agent	-	*732.7 (±154.8)*		

Bold = best within each representation; Blue shading = Best within the training curriculum. *BL* = Baseline; *BC* = Block curriculum (sequential); *RC* = Random curriculum. $P \rightarrow SI$ = Phoenix trained first, followed by Space Invaders under BC; $SI + P$ = Space Invaders and Phoenix trained together under RC. Gray shaded area = random agent scores on Galaxian for comparison.

Stand-out case — Phoenix as the unseen game. The standout results for ZSG emerged when *Phoenix* was held out during training, as depicted in Table 5.6. Notably, the planes representation significantly outperformed other representations, demonstrating a nearly 10-fold improvement in ZSG performance compared to the pixel-based representation. Remarkably, training with *Galaxian* resulted in substantially better ZSG performance than training with *Space Invaders*, especially seen in the single-training setup, but still persisted across both MTL strategies as well. The best ZSG performance was achieved by block learning, where training started with *Space Invaders* and subsequently shifted to *Galaxian*. Under this condition, ZSG performance nearly matched the baseline performance of pixels soley trained on *Phoenix*, but still fell short of the planes baseline performance (as seen in Table 5.3).

Training on Galaxian improves ZSG performance on Phoenix. A direct comparison of baseline (BL) and block curriculum (BC) training strategies, with *Phoenix* as the held-out

Table 5.6.: Zero-shot generalization results on **Phoenix**. Results are final mean episode reward (± standard deviation).

	Training	Pixels	Binary	Planes
BL	G	366.2 (±246.2)	2143.4 (±1260.4)	4160.6 (±1875.3)
	SI	234.9 (±104.5)	539.4 (±318.1)	454.8 (±220.5)
BC	SI → G	**427.4** (±317.8)	1312.0 (±1046.8)	**4901.2** (±2219.5)
	G → SI	178.4 (±140.8)	1138.8 (±675.5)	2232.4 (±1277.7)
RC	G + SI	164.0 (±64.8)	**2400.4** (±1164.7)	3429.4 (±1874.6)
Human Avg.	-	*7242.6*		
Rand. Agent	-	*754.5 (±277.2)*		

Bold = best within each representation; Blue shading = Best within the training curriculum. *BL* = Baseline; *BC* = Block curriculum (sequential); *RC* = Random curriculum. $G \rightarrow SI$ = Galaxian trained first, followed by Space Invaders under BC; $G + SI$ = Galaxian and Space Invaders trained together under RC. Gray shaded area = Human average [2] and random agent scores on Phoenix for comparison.

game, is presented in Figure 5.3. This figure clearly highlights the significant impact that training on *Galaxian* has on ZSG performance for *Phoenix*. Especially, the best-performing conditions: single-task training on *Galaxian* (purple line) and BC with *Space Invaders* first and *Galaxian* second (yellow line), are shown alongside a random agent baseline (green line) and the average human score [2] (dotted blue line). Notably, during block training (orange and yellow line), performance markedly increases during training intervals when *Galaxian* is trained, yet stagnates or diminishes when training shifts to *Space Invaders*. Additionally, the stark contrast between the baseline conditions, single-task *Space Invaders* (pink line) versus single-task *Galaxian* (purple line), further highlights *Galaxian*'s superior transferability to *Phoenix*. Finally, while the agent's performance consistently improves over time with *Galaxian* incorporated in the training, it remains below the average human score, although it demonstrates a steady upward trajectory toward that benchmark. This indicates that training with *Galaxian* allows the agent to approach average human level performance, yet it still can not fully bridge the gap.

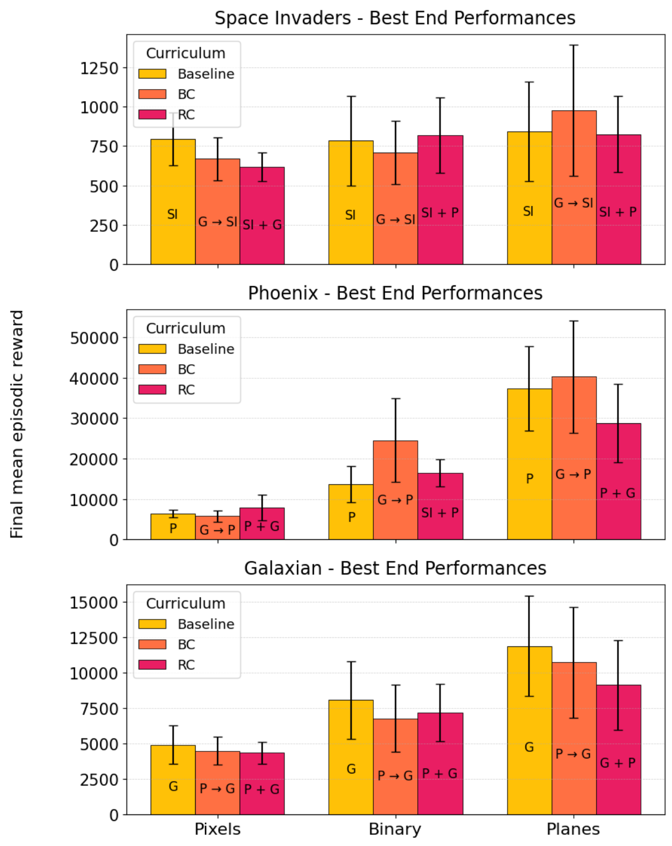

Figure 5.2.: **Best overall end performances in representation clusters.** Final mean episode reward (± standard deviation) for the best performing runs under each training strategy, evaluated after 10M steps for single-task baselines (yellow) and 20M steps for MTL: block curriculum (BC, orange) and random curriculum (RC, red). Each panel corresponds to one of the three Atari games as target tasks: *Space Invaders* (top), *Phoenix* (center), and *Galaxian* (bottom). Within each panel, bars are grouped by input representation (pixels, binary, planes), and color-coded by training strategy. The letter annotations on the bars indicate the combination and order of training games (e.g., $G \rightarrow SI$ = for BC, *Galaxian* trained before *Space Invaders*; $P + G$ = for RC, mixed *Phoenix* and *Galaxian*). The figure highlights several consistent patterns: (i) the object-centric planes representation achieves the highest performance across nearly all conditions; (ii) multi-task learning benefits vary by game, with *Phoenix* showing the most substantial gains over the baseline (iii) BC generally outperforms or matches RC, producing the best result in nearly every representation cluster.

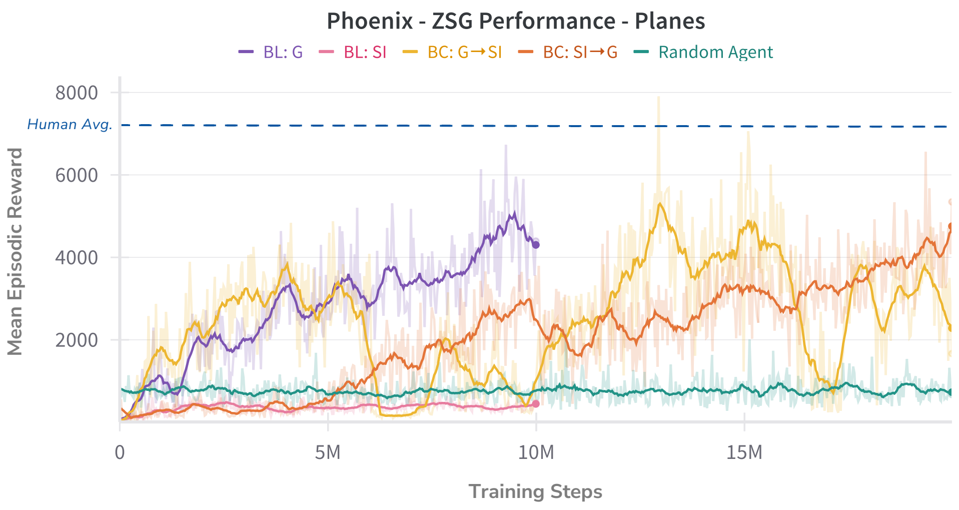

Figure 5.3.: **Zero-shot generalization with Phoenix held-out using planes input.** Training curves showing *Phoenix* test performance (mean episode reward over 20 evaluations) under various training setups, where *Phoenix* is held out during training and the input representation is planes. Shown are: single-task training on *Galaxian* (purple), single-task training on *Space Invaders* (pink), block curriculum training with *Galaxian* followed by *Space Invaders* (orange), the reverse order (yellow), a random agent (green), and the performance of a average human [2](dotted blue). The plot highlights two main trends: (i) agents trained on *Galaxian* generalize significantly better to *Phoenix* than those trained on *Space Invaders* (ii) performance during block training improves markedly when the training switches to *Galaxian*, but often stagnates or declines during *Space Invaders* phases. (iii) although agent performance remains below the human average score, the training on *Galaxian* leads to a noticeable closing of the gap, indicating promising generalization potential.

6. Discussion

The goal of this work was to investigate cluster-based generalization in reinforcement learning (RL), with a focus on three similar Atari games *Phoenix*, *Space Invaders*, and *Galaxian*. Our goal was to analyze the impact of object-centric (OC) representations and multi-task learning (MTL) strategies on the zero-shot generalization (ZSG) and transfer learning (TL) capabilities of the Proximal Policy Optimization (PPO) [36] algorithm. The following chapter will interpret the experimental results in the context of our research questions, compare them with existing literature, acknowledge the limitations of this study, and conclude by summarizing the main contributions and suggesting directions for future research.

6.1. Interpretation of Results

To comprehensively understand the factors influencing generalization in Atari games, this work set out to evaluate the performance of different input representations under various training regimes, where a detailed overview is found in Chapter 4. The following section interprets the experimental results presented in Chapter 5 and compares them to the existing literature, highlighting the strengths and limitations of each approach and gives way for future directions.

6.1.1. Baseline Generalization with Raw Pixel Input

Poor generalization with raw pixels as input. Consistent with prior studies of Cobbe et al. [8] and Farebrother et al. [12], PPO agents trained on raw pixel inputs showed severe overfitting. Even though they achieved strong baseline performance on training games, they failed catastrophically on held-out games (Table 5.4, Table 5.5, and Table 5.6). For instance, agents trained solely on *Galaxian* or *Phoenix* achieved very low ZSG on *Space*

Invaders, performing comparably to random agents. This aligns with the "generalization gap" concept from Kirk et al. [16], where agents exploit superficial visual patterns rather than learning transferable game mechanics. And the poor generalization of raw-pixel agents can also be understood in the light of prior analyses, where Convolutional networks trained on pixel inputs often latch onto spurious correlations and background details that do not hold across environments [4].

6.1.2. Effects of Object-Centric Representations

Object-centric representations significantly benefit baseline performance. Comparing the baseline performances across input types (as done in Figure 4.2) revealed the strong influence of state representation on performance. While *Space Invaders* showed minimal performance differences between pixel, binary, and planes inputs, both *Galaxian* and particularly *Phoenix* benefited substantially from OC approaches. The planes representation yielded the highest scores in these games, which was most prominent in *Phoenix*. This suggests that for games with higher complexity (more enemy types and enemy projectiles in *Phoenix* and *Galaxian*, distinct enemy phases in *Phoenix*), providing structured, semantic information via planes allows the agent to learn more effective policies compared to deciphering raw pixels or simple binary masks. This perfectly aligns with the findings of Blüml et al. [4]. Ultimately, the relative simplicity of *Space Invaders* (fewer object types, no direct enemy projectiles, more predictable movement) might explain why the representation choice mattered less.

Binary abstractions traded detail for generality. Binary representation, which abstracts away texture, color, and object identity into simple bounding boxes, demonstrated mixed effects. By drastically reducing state complexity and masking irrelevant background details, it led to a higher overall performance than pixels-based input representations, which also indicates increased TL capabilities compared to standard pixels input. This supports the findings of Blüml et al. [4], that stripping away details the agent can generalize to lower-level differences. On the other hand, binary representation did not significantly benefit ZSG in this setup as hypothesized, possibly due to the loss of critical object specific features necessary for adapting learned policies directly to entirely new environments without further training.

Planes representation enhance generalization. The multi-channel planes representation consistently outperformed both binary and pixel-based inputs across ZSG and TL, highlighting the value of structured, object specific input. Unlike the binary mask, which only signals object presence, the planes input provides explicit category labels, potentially allowing the agent to learn generalizable rules such as “avoid objects in the enemy projectile plane” or “target objects in the enemy plane.” This labeled understanding appears to support higher-level reasoning, contributing to improved performance and generalization, suggesting that additional knowledge of what an object is (its category via the plane) is more beneficial than simply knowing that an object is present (binary mask). In effect, the planes representation provides the agent with a structured, abstracted view of the environment, which aligns with cognitive science theories that emphasize human learning as relying on causal models of the world [17]. These results were most notably in the ZSG results for *Phoenix* after training on *Galaxian*. The simpler object structure in *Space Invaders* may explain why the difference between binary and planes was less pronounced in that case.

Object-centric representations did not substantially improve ZSG for Space Invaders or Galaxian. Despite the clear benefits in baseline performance and TL, neither binary nor planes representations led to notable ZSG performance, when *Space Invaders* or *Galaxian* were the held-out games. Performance often remained only marginally better than a random agent. This suggests that while OC representations provide benefits, they do not guarantee successful ZSG in this setting. This limitation may stem from the inherent difficulty of the ZSG task, as highlighted by Kirk et al. [16], or from the PPO algorithm’s limited ability to exploit structured input representations for direct policy transfer.

6.1.3. Effects of Multi-Task Learning Design

In block curriculum the order of training influenced outcomes, possibly indicating catastrophic forgetting. Block curriculum (BC) involves training games sequentially in dedicated blocks, allowing the agent to focus on one task at a time. When it was used, agents consistently achieved stronger final performance depending on which game was trained last. For example, when *Phoenix* was evaluated, end performance was substantially higher in conditions where it appeared later in the training sequence. This pattern was consistent across other games as well. However, the sequential nature of BC also introduced a vulnerability to catastrophic forgetting, where knowledge from earlier tasks was partially overwritten as training progressed to new games. This supports the findings of Goodfellow et al. [14], who showed that neural networks tend to overwrite previously learned representations when trained sequentially on distinct tasks, leading to a lower performance in earlier tasks.

Block curricula facilitated positive transfer learning, particularly between Galaxian and Phoenix. When the target game was trained second in the block structure, performance in *Phoenix* consistently exceeded the single-task baseline when *Galaxian* was included in the training, indicating positive transfer between the two games. However, looking at the reverse direction, *Galaxian*, despite benefiting most from *Phoenix*, never surpassed its single-task baseline under any input representation. This asymmetry suggests that while the two games share transferable skills or knowledge, *Galaxian* provides more useful experience for *Phoenix* than vice versa. In contrast, training that involved *Space Invaders* often resulted in neutral or even negative transfer effects, likely because it shares fewer core mechanics with the other two games. This echoes with the findings of Parisotto et al. [30], which found that pretraining on dissimilar tasks (in their case Gopher and Robotank) showed no benefit or slower learning.

Random curriculum yields general but not specialized agents. In contrast, the random curriculum (RC) strategy interleaved training steps from both games throughout the process. This prevented catastrophic forgetting and resulted in more balanced performance across training tasks. While RC agents rarely matched the peak performance of block-trained counterparts on individual games, they developed more general policies that were applicable across both games. Indicating that RC encourages agents to emerge which are more general and capable of handling multiple tasks with relatively uniform competence, approaching the combined performance of two separately trained specialists.

Zero-shot generalization remained limited under both curriculum strategies. While block training occasionally improved ZSG, overall performance remained low when compared to direct training or the performances of an average human player. The most notable ZSG result occurred when *Phoenix* was held out and the agent was trained either on *Galaxian* alone or on *Space Invaders* followed by *Galaxian*. Here, ZSG performance peaked during the *Galaxian* training phase, emphasizing again the potential transferable knowledge or skills across these two games. However, even in the best setup, ZSG results did not reach the levels of full-task training, underscoring the difficulty of ZSG without any direct exposure. *Space Invaders*, in particular, remained especially challenging for the agent, with performance barely above random when it was held out during training. Notably, the average human score was never reached in any ZSG scenario. Only in the case of *Phoenix* as the held-out game did the agent's performance approached the score of an average human, demonstrating promising results, yet still not being able to fully close the gap. This persistent challenge further reinforces the inherent difficulty of ZSG tasks, as highlighted by Kirk et al. [16], where the lack of direct experience prevents agents from learning robust, transferable policies.

6.1.4. Game-Specific Transfer Patterns in the Atari Cluster

Phoenix benefited most from OC representations and MTL strategies. Despite the apparent similarity within the fixed-screen shooter cluster, the degree to which prior experience aided learning differed substantially across *Phoenix*, *Galaxian*, and *Space Invaders*. The first, showed the most substantial gains from both OC inputs and MTL setups. Its complexity, ranging from multi-phase enemy waves to a unique boss level, makes it a challenging task to master from scratch. Prior training on *Galaxian* provided a strong foundation in core mechanics, allowing later fine-tuning on *Phoenix*'s unique challenges. This resulted in a notable performance in ZSG and higher final scores for the agent.

Galaxian and Space Invaders benefited from OC inputs, but showed limited gains from MTL. While both *Galaxian* and *Space Invaders* showed clear improvements under OC input representations, more so for *Galaxian*, their gains from MTL were less pronounced than those observed for *Phoenix*. When looking at *Galaxian* at as the target task, the agent often achieved strong performance in the single-task baseline, reducing the advantage of transfer. In contrast, *Space Invaders*, arguably the most distinct game in the cluster due to having the fewest object types within the conceptual planes and featuring more static enemy patterns without enemy projectiles, showed only modest MTL driven gains under OC conditions. These patterns suggest that MTL benefits were most prominent when combined with OC inputs and when the target game posed either greater learning difficulty or aligned well with the skills provided by the source tasks.

Galaxian frequently emerged as an effective source task, especially in sequential training in the cluster. A notable pattern emerging from the MTL experiments was the frequent effectiveness of *Galaxian* as a source task. Across the 12 multi-task conditions explored (Figure 5.2), whenever *Galaxian* was not the target game, it provided the most benefit to the other games in all block curriculum conditions. In random curriculum it was the more successful helper in half of the scenarios. This pattern suggests that *Galaxian* provides broadly transferable skills that support learning across the cluster. This highlights its strength as a source task and reinforces the idea that task selection within a curriculum should consider not only target difficulty but also source deployment.

Space Invaders consistently emerged as the least effective source task for transferable skills. Experimental results clearly demonstrated that *Space Invaders* contributed the least to performance improvements in MTL setups inside the cluster. As depicted in Figure 5.2, it achieved the best MTL outcome in only one out of 12 conditions, when it was a source task. This suggests that the specific game mechanics and visual simplicity of *Space*

Invaders constrain its ability to foster transferable skills that benefit training on *Galaxian* and *Phoenix*.

6.2. Limitations

While this study provides valuable insights into how OC representations and MTL strategies affect generalization in RL within a specific cluster of Atari games, it is important to acknowledge several limitations to properly assess the findings:

Firstly, the scope of this research was limited to a small cluster of three Atari 2600 games: *Phoenix*, *Space Invaders*, and *Galaxian*. While these games appear similar as fixed-screen "shoot-em up" titles at first glance, the degree of actual overlap "under the hood" be less than initially assumed. Particularly *Space Invaders* lacks direct enemy projectiles and has static enemies, which makes it structurally different from the other two games. This can be a problem, as transfer between tasks with limited similarity may hinder performance rather than to help, as shown in prior work from Parisotto et al. [30]. Furthermore, the limited diversity within the cluster might affect the extent to which the generalization patterns can be extrapolated to more complex tasks or real-world scenarios.

Secondly, our choice of learning algorithm was restricted to the PPO algorithm. While PPO is a widely used and robust algorithm known for its stability and good performance across various tasks, its on-policy nature might limit its sample efficiency compared to off-policy methods. This difference in sample efficiency and incorporation of past experiences could potentially influence the learning dynamics and the final generalization capabilities of the agent.

Moreover, all experiments were conducted using a fixed set of hyperparameters and the same network architecture across all input representations and training regimes. While this approach ensured a fair comparison, it may have introduced suboptimal performance for certain configurations. For instance, the object-centric planes input representations, with more input channels compared to raw pixel inputs, might have benefited from a different network architecture. Similarly, the learning rates, network depth, or other hyperparameters could have been fine-tuned for each input type and training strategy to potentially yield better individual performances.

Furthermore, training steps in this work were limited to 10 million per game. In contrast, other works utilized training periods for far longer. Taïga et al. [39] trained agents for significantly longer durations of 200M environmental frames per game. This discrepancy in training time may have influenced both performance and generalization capabilities. It is possible that with extended training, especially in MTL setups, key learning dynamics or transfer patterns would emerge more clearly.

Lastly, the planes representation utilized in this work relied on a handcrafted mapping of object categories across the three games (Table 4.2). While this mapping was designed to capture semantic similarities, it is inherently limited by the experimenter's understanding of the game objects and their roles. This reliance may have introduced biases or overlooked subtler categorization within the games, potentially limiting the full effectiveness.

6.3. Conclusion

Reinforcement learning (RL) is advancing at a rapid pace and has demonstrated remarkable capabilities in controlled environments, however its widespread application in the real-world remains very limited. Overcoming the challenge of generalization in RL will get humanity closer to truly adaptable, real-world agents. This work addressed the critical issue of generalization in RL, by investigating the impact of input representation and training curricula on the zero-shot generalization (ZSG) and transfer learning (TL) abilities of RL agents within a cluster of similar Atari 2600 games.

Our findings highlight that the agent's perception of the environment and the learning strategy employed, both greatly influence generalization capabilities. Consistent with established research on overfitting in deep RL, training agents simply on raw pixel inputs resulted in poor generalization performance. These agents are capable of achieving high scores within their specific training environments, but fail drastically to transfer their skills to unseen games, indicating that the agent did not learn the underlying core mechanics and rather relied on superficial visual cues.

In stark contrast, the adoption of object-centric (OC) input representations, particularly the multi-channel planes encoding, consistently outperformed raw pixel and binary representations as inputs, in terms of final performances and TL capabilities. The planes representation explicitly provided information about object categories and ultimately led to more abstract and robust policies, aligning with cognitive science theories that emphasize the importance of structured reasoning for generalization.

Furthermore, our exploration of MTL strategies revealed a complex interplay between learning efficiency and generalization. While block curriculum training often led to higher final performance on individual games and facilitated positive transfer between similar games like *Galaxian* and *Phoenix*, it also suffered from catastrophic forgetting. On the other hand, with training using a random curriculum more general agents emerged, which were capable of performing reasonably well across multiple games, suggesting the learning of shared underlying strategies. Notably, *Galaxian* emerged as a particularly effective source task for transfer within our cluster, while *Phoenix* proved to be a more receptive target. *Space Invaders*, however, demonstrated limited transferability, likely due to its more distinct mechanics.

Finally, ZSG remained a significant challenge across all conditions tested. Nonetheless, the planes representation showed promising results. It achieved non-trivial ZSG performance on *Phoenix* when trained on *Galaxian*. This finding suggests that OC input representations hold potential for enabling agents to perform effectively in entirely new environments without further adaptation. Despite this encouraging outcome, ZSG remains a critical and open area for further investigation.

In summary, this research reinforces the critical role of input representation and training design in achieving generalization in RL. By demonstrating the benefits of OC representations and varying effects of MTL strategies within a cluster of similar tasks, this work provides valuable empirical insights that contribute to the ongoing pursuit of more robust and adaptable AI agents for real-world deployment.

6.4. Future Work

This study sets the stage for several promising directions for future research aimed at further enhancing generalization in RL, particularly in the context of MTL scenarios and OC representations. The following areas highlight key opportunities for expanding upon the insights gained in this work:

Given the observed performance drops due to catastrophic forgetting in block curriculum setups and to also overall increase generalization capabilities of the agent in this realm, future work might incorporate continual learning techniques like Progressive Neural Networks, proposed by Rusu et al. [33], which will improve the retention of learned knowledge and make the agent immune to catastrophic forgetting. Additionally, applying regularization strategies, such as weight decay or L2 regularization, as investigated by Farebrother et al. [12], may further help preserve prior knowledge during sequential training, ultimately supporting more stable and generalized agent performance across multiple tasks.

Another promising direction involves refining the task clustering itself. Rather than relying solely on predefined clusters based on perceived similarities, exploring different clustering techniques, such as a sort of hierarchical clustering based on game mechanics or learned representations. This could reveal more nuanced relationships between tasks and potentially lead to more effective TL strategies. For instance, building upon the work of Taïga et al. [39] or Delfosse et al. [9], which both tested generalization with variants of the same Atari 2600 game. One could cluster game variants based on empirically assessed game mechanics, visual features or deviations from the original game. Such clustering could enable a more systematic analysis of generalization capabilities across similar tasks, thereby advancing our understanding of generalization in RL.

Moreover, further research could explore the impact of different RL algorithms on cluster-based generalization. While this study focused on the on-policy PPO algorithm, investigating the performance of off-policy methods like DQN [24] could provide valuable comparative insights into how different learning paradigms interact with OC representations and MTL.

Lastly, another promising avenue is the adaptive tuning of hyperparameters, within the MTL and object-centric RL setting. Employing adaptive tuning strategies could enhance generalization across diverse tasks by tailoring learning dynamics to the specific challenges of each environment. Additionally, extending training durations has been adapted by various works (50M environment steps per game by Mnih et al. [24] or 200M environmental steps by Taïga et. al. [39], whereas this study only deployed 40M environment steps per game). Therefore, combining hyperparameter tuning with extended training periods may offer valuable insights into the realm of generalization in RL with OC representations and MTL strategies.

7. Acknowledgement

We would like to express our sincere gratitude to our advisor, Cedric Derstroff, for his helpful guidance and continuous support throughout this project. We are also very grateful for the support of the AIML team for providing us with the necessary computational resources that made this work possible. We would like to also thank our friends and family for the never-ending support and encouragements.

Bibliography

[1] Kai Arulkumaran et al. “Deep Reinforcement Learning: A Brief Survey”. In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38. DOI: 10.1109/MSP.2017.2743240.

[2] Adrià Puigdomènech Badia et al. “Agent57: Outperforming the Atari Human Benchmark”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 507–517. URL: https://proceedings.mlr.press/v119/badia20a.html.

[3] M. G. Bellemare et al. “The Arcade Learning Environment: An Evaluation Platform for General Agents”. In: *J. Artif. Intell. Res.* 47 (2013). DOI: https://doi.org/10.1613/jair.3912.

[4] Jannis Blüml et al. *Deep Reinforcement Learning via Object-Centric Attention*. 2025. arXiv: 2504.03024 [cs.LG]. URL: https://arxiv.org/abs/2504.03024.

[5] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[6] Rich Caruana. “Multitask Learning”. In: *Machine Learning* 28.1 (July 1997), pp. 41–75. ISSN: 1573-0565. DOI: 10.1023/A:1007379606734. URL: https://doi.org/10.1023/A:1007379606734.

[7] Karl Cobbe et al. “Leveraging Procedural Generation to Benchmark Reinforcement Learning”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 2048–2056. URL: https://proceedings.mlr.press/v119/cobbe20a.html.

[8] Karl Cobbe et al. “Quantifying Generalization in Reinforcement Learning”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 1282–1289. URL: https://proceedings.mlr.press/v97/cobbe19a.html.

[9] Quentin Delfosse et al. *HackAtari: Atari Learning Environments for Robust and Continual Reinforcement Learning*. 2024. arXiv: 2406.03997 [cs.AI]. URL: https://arxiv.org/abs/2406.03997.

[10] Quentin Delfosse et al. "Interpretable and Explainable Logical Policies via Neurally Guided Symbolic Abstraction". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 50838–50858. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/9f42f06a54ce3b709ad78d34c73e4363-Paper-Conference.pdf.

[11] Quentin Delfosse et al. *OCAtari: Object-Centric Atari 2600 Reinforcement Learning Environments*. 2024. arXiv: 2306.08649 [cs.LG]. URL: https://arxiv.org/abs/2306.08649.

[12] Jesse Farebrother, Marlos C. Machado, and Michael Bowling. *Generalization and Regularization in DQN*. 2020. arXiv: 1810.00123 [cs.LG]. URL: https://arxiv.org/abs/1810.00123.

[13] Dibya Ghosh et al. "Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 25502–25515. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/d5ff135377d39f1de7372c95c74dd962-Paper.pdf.

[14] Ian J. Goodfellow et al. *An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks*. 2015. arXiv: 1312.6211 [stat.ML]. URL: https://arxiv.org/abs/1312.6211.

[15] Niels Justesen et al. *Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation*. 2018. arXiv: 1806.10729 [cs.LG]. URL: https://arxiv.org/abs/1806.10729.

[16] Robert Kirk et al. "A Survey of Zero-shot Generalisation in Deep Reinforcement Learning". In: *J. Artif. Intell. Res.* 76 (2023). DOI: https://doi.org/10.1613/jair.1.14174.

[17] Brenden M. Lake et al. "Building machines that learn and think like people". In: *Behavioral and Brain Sciences* 40 (2017), e253. DOI: 10.1017/S0140525X16001837.

[18] Misha Laskin et al. "Reinforcement Learning with Augmented Data". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 19884–19895. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/e615c82aba461681ade82da2da38004a-Paper.pdf.

[19] Yuxi Li. *Deep Reinforcement Learning: An Overview*. 2018. arXiv: 1701.07274 [cs.LG]. URL: https://arxiv.org/abs/1701.07274.

[20] Grace W. Lindsay. “Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future”. In: *Journal of Cognitive Neuroscience* 33.10 (Sept. 2021). _eprint: https://direct.mit.edu/jocn/article-pdf/33/10/2017/1962083/jocn_a_01544.pdf, pp. 2017–2031. ISSN: 0898-929X. DOI: 10.1162/jocn_a_01544. URL: https://doi.org/10.1162/jocn%5C_a%5C_01544.

[21] Marlos C. Machado et al. *Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents*. 2017. arXiv: 1709.06009 [cs.LG]. URL: https://arxiv.org/abs/1709.06009.

[22] Michael McCloskey and Neal J. Cohen. “Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem”. In: ed. by Gordon H. Bower. Vol. 24. Psychology of Learning and Motivation. ISSN: 0079-7421. Academic Press, 1989, pp. 109–165. DOI: https://doi.org/10.1016/S0079-7421(08)60536-8. URL: https://www.sciencedirect.com/science/article/pii/S0079742108605368.

[23] Volodymyr Mnih et al. “Asynchronous Methods for Deep Reinforcement Learning”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1928–1937. URL: https://proceedings.mlr.press/v48/mniha16.html.

[24] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 1476-4687. DOI: 10.1038/nature14236. URL: https://doi.org/10.1038/nature14236.

[25] Alex Nichol et al. *Gotta Learn Fast: A New Benchmark for Generalization in RL*. 2018. arXiv: 1804.03720 [cs.LG]. URL: https://arxiv.org/abs/1804.03720.

[26] OpenAI et al. *Dota 2 with Large Scale Deep Reinforcement Learning*. 2019. arXiv: 1912.06680 [cs.LG]. URL: https://arxiv.org/abs/1912.06680.

[27] OpenAI et al. *Solving Rubik’s Cube with a Robot Hand*. 2019. arXiv: 1910.07113 [cs.LG]. URL: https://arxiv.org/abs/1910.07113.

[28] Charles Packer et al. *Assessing Generalization in Deep Reinforcement Learning*. 2019. arXiv: 1810.12282 [cs.LG]. URL: https://arxiv.org/abs/1810.12282.

[29] Weike Pan. “A survey of transfer learning for collaborative recommendation with auxiliary data”. In: *Neurocomputing* 177 (2016), pp. 447–453. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2015.11.059. URL: https://www.sciencedirect.com/science/article/pii/S0925231215018640.

[30] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. *Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning*. 2016. arXiv: 1511.06342 [cs.LG]. URL: https://arxiv.org/abs/1511.06342.

[31] Judea Pearl and Dana Mackenzie. *The Book of Why*. New York: Basic Books, 2018. ISBN: 978-0-465-09760-9.

[32] Antonin Raffin et al. “Stable-Baselines3: Reliable Reinforcement Learning Implementations”. In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

[33] Andrei A. Rusu et al. *Progressive Neural Networks*. 2022. arXiv: 1606.04671 [cs.LG]. URL: https://arxiv.org/abs/1606.04671.

[34] A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers”. In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229. DOI: 10.1147/rd.33.0210.

[35] Julian Schrittwieser et al. “Mastering Atari, Go, chess and shogi by planning with a learned model”. In: *Nature* 588.7839 (Dec. 2020), pp. 604–609. ISSN: 1476-4687. DOI: 10.1038/s41586-020-03051-4. URL: https://doi.org/10.1038/s41586-020-03051-4.

[36] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[37] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. ISSN: 1476-4687. DOI: 10.1038/nature16961. URL: https://doi.org/10.1038/nature16961.

[38] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[39] Adrien Ali Taiga et al. “Investigating Multi-task Pretraining and Generalization in Reinforcement Learning”. In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=sSt9fROSZRO.

[40] Oriol Vinyals et al. “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. In: *Nature* 575.7782 (Nov. 2019), pp. 350–354. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1724-z. URL: https://doi.org/10.1038/s41586-019-1724-z.

[41] Nelson Vithayathil Varghese and Qusay H. Mahmoud. “A Survey of Multi-Task Deep Reinforcement Learning”. In: *Electronics* 9.9 (2020). ISSN: 2079-9292. DOI: 10.3390/electronics9091363. URL: https://www.mdpi.com/2079-9292/9/9/1363.

[42] Max Weltevrede et al. *Explore-Go: Leveraging Exploration for Generalisation in Deep Reinforcement Learning*. 2024. arXiv: 2406.08069 [cs.LG]. URL: https://arxiv.org/abs/2406.08069.

[43] Rikiya Yamashita et al. “Convolutional neural networks: an overview and application in radiology”. In: *Insights into Imaging* 9.4 (Aug. 2018), pp. 611–629. ISSN: 1869-4101. DOI: 10.1007/s13244-018-0639-9. URL: https://doi.org/10.1007/s13244-018-0639-9.

[44] Amy Zhang, Nicolas Ballas, and Joelle Pineau. *A Dissection of Overfitting and Generalization in Continuous Reinforcement Learning*. 2018. arXiv: 1806.07937 [cs.LG]. URL: https://arxiv.org/abs/1806.07937.

[45] Chiyuan Zhang et al. *A Study on Overfitting in Deep Reinforcement Learning*. 2018. arXiv: 1804.06893 [cs.LG]. URL: https://arxiv.org/abs/1804.06893.

[46] Zhuangdi Zhu et al. *Transfer Learning in Deep Reinforcement Learning: A Survey*. 2023. arXiv: 2009.07888 [cs.LG]. URL: https://arxiv.org/abs/2009.07888.

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 APB TU Darmstadt

Hiermit erkläre ich, Lucas Gaston Dysli, dass ich die vorliegende Arbeit gemäß § 22 Abs. 7 APB der TU Darmstadt selbstständig, ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Ich habe mit Ausnahme der zitierten Literatur und anderer in der Arbeit genannter Quellen keine fremden Hilfsmittel benutzt. Die von mir bei der Anfertigung dieser wissenschaftlichen Arbeit wörtlich oder inhaltlich benutzte Literatur und alle anderen Quellen habe ich im Text deutlich gekennzeichnet und gesondert aufgeführt. Dies gilt auch für Quellen oder Hilfsmittel aus dem Internet.

Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

L. Dysli.

Darmstadt, May 14, 2025

Lucas Gaston Dysli

A. Additional Materials

The following appendix provides supplementary figures illustrating the learning curves for the three Atari 2600 games: *Space Invaders*, *Galaxian*, and *Phoenix*. These plots present a comprehensive view of the agent's performance across different learning strategies and input representations, showcasing the impact of single-task training, block curriculum (BC), and random curriculum (RC) on final outcomes.

For better visualization, BC runs are separated into two categories: those where the target game is trained first in the training order (left side) and those where it is trained last (right side). Pixel-based representations are displayed in light and dark green, binary-based representations in yellow and orange, and planes-based representations in purple and pink.

To maintain clarity, specific curriculum notations are used consistently: $G \rightarrow SI$: *Galaxian* is trained first, followed by *Space Invaders* in BC. $P + G$: *Phoenix* and *Galaxian* are trained simultaneously in RC. All learning curves in the figures are averaged over 25 evaluation episodes.

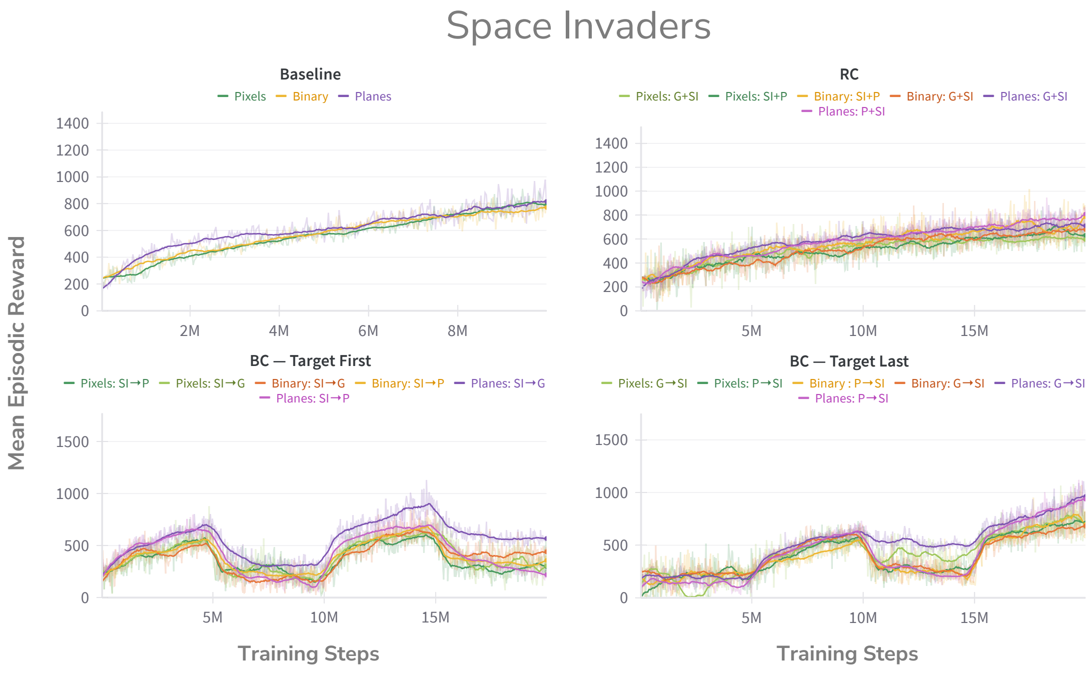

Figure A.1.: Learning curves for Space Invaders across training strategies.

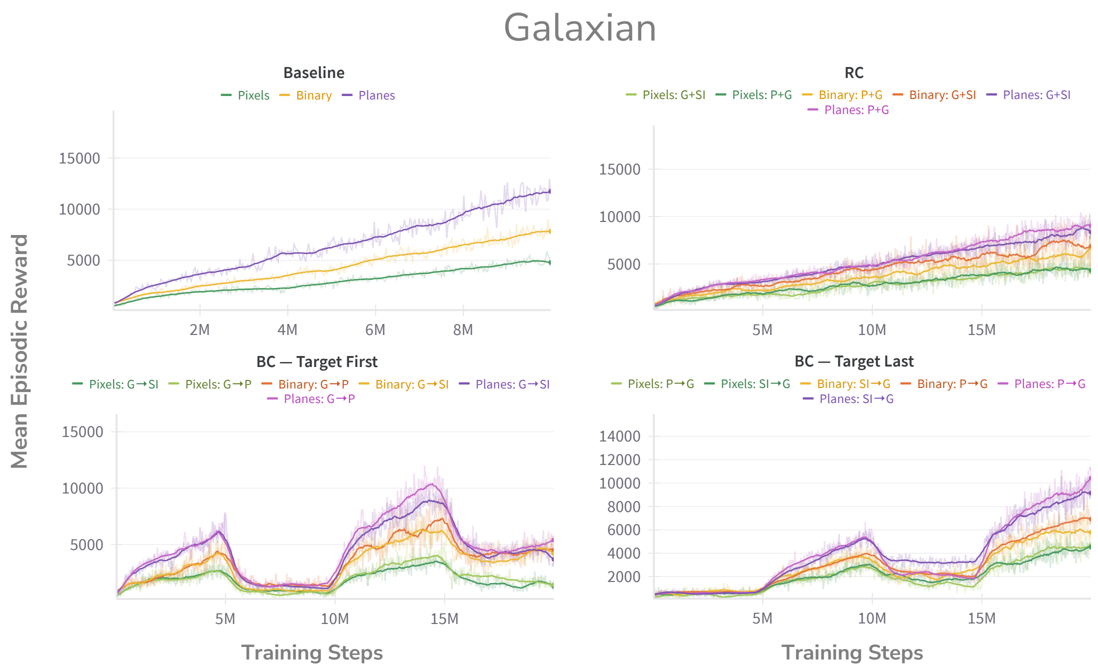

Figure A.2.: Learning curves for Galaxian across training strategies.

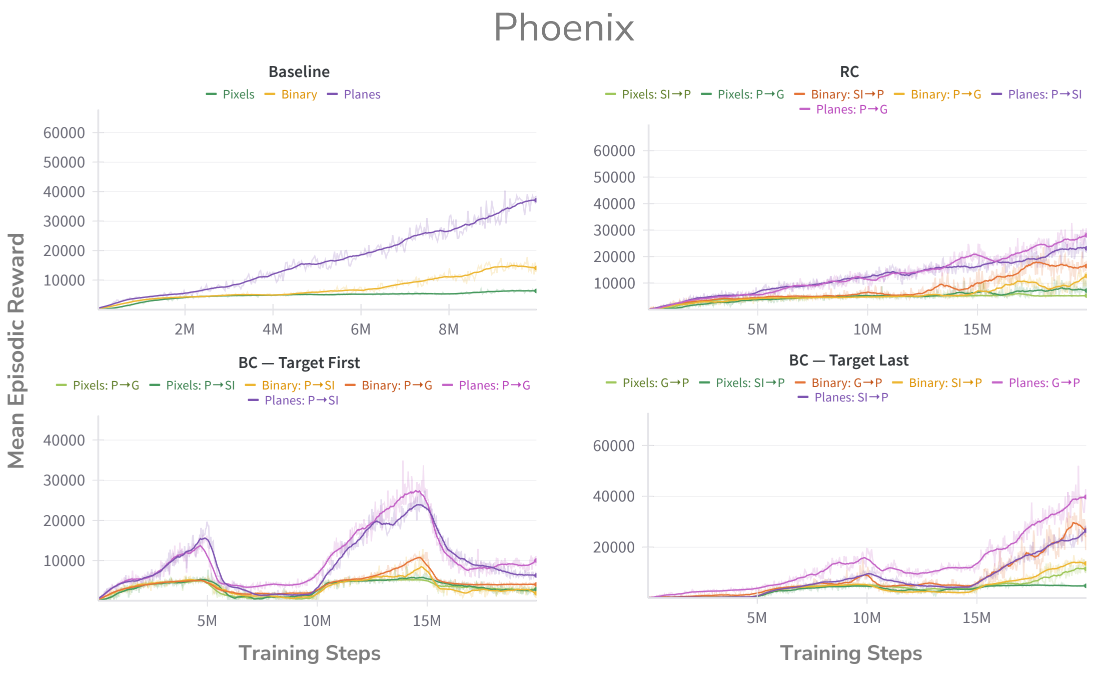

Figure A.3.: Learning curves for Phoenix across training strategies.