Universiteit
Leiden
The Netherlands

Exploring the synergies between transfer in reinforcement learning and procedural content generation

Müller-Brockhausen, M.F.T.

Citation
Müller-Brockhausen, M. F. T. (2025, November 5). *Exploring the synergies between transfer in reinforcement learning and procedural content generation*. Retrieved from https://hdl.handle.net/1887/4282228

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/4282228

Note: To cite this publication please use the final published version (if applicable).

Exploring the Synergies between Transfer in Reinforcement Learning and Procedural Content Generation

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op woensdag 5 november 2025
klokke 16:00 uur

door

Matthias Müller-Brockhausen

geboren te Bonn, Duitsland

in 1994

Promotor:	Prof.dr. A. Plaat
Co-promotor:	Dr. M. Preuss
Promotiecommissie:	Prof.dr. M.M. Bonsangue
	Prof.dr. K.J. Batenburg
	Prof.dr. G.N. Yannakakis (L-Università ta' Malta)
	Prof.dr. S.M. Lucas (Queen Mary University of London)
	Dr. T.M. Moerland
	Dr. V. Volz (CWI Research institute, Amsterdam)

Copyright © 2025 Matthias Müller-Brockhausen.

Cover Design by Matthias Müller-Brockhausen

Cover contains Images published under CC0 by freesvg.org (IDs of used SVGs are: 13274, 149881, 118363, 146014, 19872, 187886, 15473, 21379, 125678)

Contents

Chapter 1

Introduction

1.1 Background

Video games exemplify "perfect testbeds for artificial intelligence" [272], as they provide complex, interactive, and dynamic scenarios that challenge computational models. They consist of many different components (ranging from mathematics, and graphics engines, to approximations of the laws of physics [105]). The primary objective of these systems is to create an engaging and immersive experience for users. Achieving this requires the seamless integration of a multitude of technologies [30].

Designing Artificial Intelligence (AI) in video games presents a spectrum of challenges. When creating believable and immersive worlds the focus lies on replicating human-like behavior [121, 303] through Non-Player Characters (NPCs). Furthermore, in some highly competitive games, the goal may be either to achieve superhuman decision-making capabilities [43, 285], or to provide players with an interesting opponent [296].

Games of a competitive nature have seen significant advances in AI achieved through algorithms such as MiniMax [202, 205], Monte Carlo Tree Search (MCTS) [44, 56], Evolutionary Algorithms (EAs) [94, 152, 208], Reinforcement Learning (RL) [139, 298], and hybrids of RL & MCTS [168, 232, 279]. Particularly RL is well suited for learning interactive behavior and has made great strides in the field of robotics [249], where robots learn human-like movement [23]. Moreover, in video games, the promise of reinforcement learning has been well recognized [239].

The development of benchmarks is critical for advancing and validating any AI system. One notable example in the field of video games is the General Video Game Playing AI (GVGAI) Framework [196]. Recent advancements in competitive AI have shifted focus toward multi-agent systems, emphasizing coordinated decision-making through interactions in complex environments. Games such as Dota 2 have become prominent platforms for testing and refining these algorithmic frameworks [28]. Ad-

ditionally, the emergence of Large Language Models (LLMs) has introduced novel capabilities for strategic planning, giving rise to novel benchmarks [184].

Human behavior is intrinsically complex, making its replication in video games a non-trivial challenge. Historically, game-level design relied on a static approach, where the content was manually crafted. Procedural Content Generation (PCG) encompasses any content for video games that is generated algorithmically. Examples include trees [118, 186], meshes [284], levels [129, 134, 209, 223, 261, 269, 282], textures [57], and dialogue [111, 165, 257]. This list is not exhaustive, for a more complete overview we refer the reader to [115]. To generate the required level of diversity in content for AI, developers have adopted algorithmic methods to procedurally generate content using a pseudorandom number generator (PRNG) [194]. In consequence, PCG has enabled video games to exhibit greater variability and unpredictability in their content [115], which in turn elevates the cognitive and strategic demands on players, allowing controllable levels of difficulty for gameplay [226, 251].

This trend towards more diversity can also be seen in machine learning [130]. Training new deep learning models for different data sets and environments is expensive [217] and wasteful [193]. It is much more efficient to transfer knowledge from a model that has been learned for one task to a model for a different task [265]. For example, in image recognition, the ImageNet model is trained on millions of images [67]. Subsequently, it is employed to fine-tune smaller models for specific recognition tasks [150]. Similarly, in natural language processing (NLP), LLMs are evolving into foundation models [18] capable of being adapted through fine-tuning for diverse applications [132]. The perfect example of this is the usage of a vision language action model for navigational tasks in robotics [237].

We wonder how the concept of transfer learning can contribute to video games. In turn, our thesis delves into the possible synergies between Transfer in Reinforcement Learning (TRL) and PCG by categorizing existing research and performing empirical studies.

Now that we have introduced key aspects of the relevant fields, we transition to defining the central research questions that will guide this thesis.

1.2 Research Questions

Throughout the research efforts of our thesis, we pose a variety of questions that we aim to answer by means of empirical analysis and experimentation in our publications. In the subsections, we first motivate the origin of our question based on related research.

Then we pose questions for which we list the chapters related to allowing us to answer the posed question. In Chapter 7 we present the answers to these individual questions.

Our main topic in this thesis is:

> Exploring synergies between TRL and PCG.

This general topic captures the research that is covered in this thesis at a high level. Combining TRL and PCG suggests the need for better benchmarks. To investigate this topic in greater depth, we have formulated the following research questions.

1.2.1 RQ1

The inherent difficulty in establishing robust benchmarks stems from the phenomenon known as Goodhart's law: "When a measure becomes a target, it ceases to be a good measure" [258]. This principle underscores the necessity for continuous benchmark evolution, which is why we ask:

> How can benchmarks for (transfer in) RL be improved?

This research question is studied by first benchmarking a variety of popular AI methods (Heuristic, MCTS, RL) on the game Tetris Link (Chapter 2). Both MCTS and RL exhibited disappointing results in comparison to a heuristic search. Additionally, our investigations into RL highlighted reproducibility challenges, a well-known issue in the field [201], prompting us to explore potential solutions (Chapter 4). Furthermore, we identified transfer in RL as a possible way to improve performance, guiding us to survey the field (Chapter 3) and conclude that PCG is a promising method to further improve benchmarks for (transfer in) RL. We answer this question in Section 7.1.1.

1.2.2 RQ2

The previously mentioned hybrids of RL & MCTS highlight that the combination of two methods can yield promising results, achieving superhuman performance in games like Go [232] or Starcraft II [279]. Our main topic is to explore such synergies between TRL and PCG, thus we ask more concretely:

> Can TRL improve PCG applications in video games?

This research question is studied by identifying limitations in TRL experiments of existing research (Chapter 3) and empirically showing that combining TRL and PCG is promising (Chapter 5). We answer this question in Section 7.1.2.

1.2.3 RQ3

The use of LLMs in video games holds significant promise as it can be used for AI [128] or for PCG to generate levels [261, 269], and interactive dialogues [257]. Moreover, the existence of mods for existing games that incorporate LLMs [15, 211] highlights a clear demand by the user base. However, as a developer, controlling the output is an open problem. That is why we wonder:

> Can LLMs procedurally generate content locally for video games while preventing adversarial attacks?

This research question studies the computational requirements and safety of deploying LLMs that procedurally generate content in video games. We run empirical experiments (Chapter 6) to help answer this question (Section 7.1.3).

1.3 Contributions

This doctoral thesis comprises independent research publications, each constituting a distinct chapter. We provide a concise overview of the key findings and arguments presented in each chapter. We will now list our main contributions:

Chapter 2 studies the game of Tetris Link using three different popular AI algorithms: Heuristic Search, MCTS, and RL. To our surprise, the classical method, heuristic search, performs best. We show that this is due to a combination of a large branching factor and restrictive game rules. This chapter is based on:

[175] Müller-Brockhausen, M., Preuss, M., Plaat, A.: A new challenge: Approaching tetris link with AI. In: 2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021. IEEE (2021), `https://doi.org/10.1109/CoG52621.2021.9619044`.

This surprising result has caused us to investigate transfer methods in RL in Chapter 3. Here we find that transfer learning is studied widely in reinforcement learning and is successful when levels are simple and pre-programmed. However, we find that few works use transfer in combination with PCG, identifying a field for further study. This chapter is based on:

[176] Müller-Brockhausen, M., Preuss, M., Plaat, A.: Procedural content generation: Better benchmarks for transfer reinforcement learning. In: 2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021. IEEE (2021), `https://doi.org/10.1109/CoG52621.2021.9619000`.

Reproducibility is a desirable trait, especially for TRL experiments. However, it remains an unsolved and often ignored problem in the field of RL [201]. As we cannot fix reproducbility, we propose focusing on verifiability (Chapter 4). This enables reviewers to validate empirically reported results, such as tables or graphs. This chapter is based on:

[173] Müller-Brockhausen, M., Plaat, A., Preuss, M.: Towards verifiable benchmarks for reinforcement learning. In: IEEE Conference on Games, CoG 2022, Beijing, China, August 21-24, 2022. pp. 159–166. IEEE (2022), https://doi.org/10.1109/CoG51982.2022.9893715.

Next in Chapter 5 we empirically show the synergies of TRL and PCG at the example of Linerider 3D (see 7.1.2). Employing PCG broadens the explored state space during training, yielding scalable transfer unachievable when learning from scratch (see 5.6.2). Complementarily, the task of the RL environment is to procedurally generate a track for the video game Linerider (see 5.4). This chapter is based on:

[172] Müller-Brockhausen, M., Khalifa, A., Preuss, M.: Scalable procedural content generation via transfer reinforcement learning. In: Data Science and Artificial Intelligence (DSAI). Springer (2024), https://doi.org/10.1007/978-981-97-9793-6.

In our final chapter 6, we study the viability of local LLMs to generate content for video games, while preventing derailment through the user. To that end we suggest the usage of Chatter, which disables user input, relying solely on pre-defined prompts. This prevents users from breaking the model out of its safety guidelines, for which researchers keep finding new creative ways [46, 53]. This chapter is based on:

[171] Müller-Brockhausen, M., Barbero, G., Preuss, M.: Chatter generation through language models. In: IEEE Conference on Games, CoG 2023, Boston, MA, USA, August 21-24, 2023. IEEE (2023), https://doi.org/10.1109/CoG57401.2023.10333244.

1.3.1 Published Collaborations of the Author

In addition to these published works, we have collaborated on a variety of topics which has led to the following publications:

[137] Kampert, D., Varbanescu, A.L., Müller-Brockhausen, M., Plaat, A.: Mimicking the human approach in the game of hive. In: IEEE Symposium Series on Computational Intelligence, SSCI 2021, Orlando, FL, USA, December 5-7, 2021. IEEE (2021), https://doi.org/10.1109/SSCI50451.2021.9659999.

[174] Müller-Brockhausen, M., Plisnier, H.: Transferring while playing the rl agent. In: Demo at BNAIC/BeNeLearn 2022: Joint International Scientific Conferences on AI and Machine Learning (2022), https://bnaic2022.uantwerpen.be/wp-content/uploads/BNAICBeNeLearn_2022_submission_6564.pdf.

[255] van der Staaij, A., Prins, J., Prins, V.L., Poelsma, J., Smit, T., Müller-Brockhausen, M., Preuss, M.: Believable minecraft settlements by means of decentralised iterative planning. In: IEEE Conference on Games, CoG 2023, Boston, MA, USA, August 21-24, 2023. IEEE (2023), https://doi.org/10.1109/CoG57401.2023.10333146.

[21] Barbero, G., Müller-Brockhausen, M., Preuss, M.: Challenges of open world games for ai: Insights from human gameplay. In: Data Science and Artificial Intelligence (DSAI). Springer Nature (2024), https://doi.org/10.1007/978-981-97-9793-6.

Chapter 2

A New Challenge: Approaching Tetris Link with AI

Decades of research have been invested in making computer programs for playing games such as Chess and Go. This paper[a] focuses on a new game, Tetris Link, a board game that is still lacking any scientific analysis. Tetris Link has a large branching factor, hampering a traditional heuristic planning approach. We explore heuristic planning and two other approaches: Deep Reinforcement Learning (DRL), MCTS. We document our approach and report on their relative performance in a tournament. Curiously, the heuristic approach is stronger than the planning/learning approaches. However, experienced human players easily win the majority of the matches against the heuristic planning AIs. We, therefore, surmise that Tetris Link is more difficult than expected. We offer our findings to the community as a challenge to improve upon.

[a]This chapter is based on the publication [175] Müller-Brockhausen, M., Preuss, M., Plaat, A.: A new challenge: Approaching tetris link with AI. In: 2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021. IEEE (2021), https://doi.org/10.1109/CoG52621.2021.9619044

2.1 Introduction

Board games are favorite among AI researchers for experiments with intelligent decision-making and planning. For example, works that analyze the game of Chess date back centuries [198, 254]. Already in 1826, papers were published on machines that supposedly played Chess automatically [38], although it was unclear whether the machine was still operated somehow by humans. Nowadays, for some games, such as Chess [120] and Go [246, 248], we know for sure that there are algorithms that can, without the help of humans, automatically decide on a move and even outplay the best human players. In this paper, we want to investigate a new game, Tetris Link, that has not yet received attention from researchers before, to the best of our knowledge (see Section 2.2). Tetris Link is a manual, multiplayer version of the well-known video game Tetris. It is played on a vertical "board", like Connect-4. The game has a large branching factor, and since it is not immediately obvious how a strong computer program should be designed, we put ourselves to this task in this paper. For that, we implement a digital version of the board game and take a brief look at the game's theoretic aspects (Section 2.3). Based on that theory, we develop heuristics for a minimax-based program that we also test against human players (Section 2.4.1). Performance is limited, and we try other common AI approaches: DRL [263] and MCTS [44]. In Subsection 2.5.1, we look at MCTS agents, and in Subsection 2.5.2, we look at RL agents and their performance in the game. In our design of the game environment for the RL agent, we assess the impact of choices such as the reward on training success. Finally, we compare the performance of these agents after letting them compete against each other in a tournament (Section 2.5.3). To our surprise, humans are stronger.

The main contribution of this paper is that we present to the community the challenge of implementing a well-playing computer program for Tetris Link. This challenge is much harder than expected, and we provide evidence (Section 2.3.2) on why this might be the case, even for the deterministic 2-player version (without dice) of the game. The real Tetris Link can be played with four players using dice, which will presumably be even harder for an AI.

We document our approach, implementing three players based on the three main AI game-playing approaches of heuristic planning, MCTS, and DRL. To our surprise and regret, all players were handily beaten by human players.[1] We respectfully offer our approach, code, and experience to the community to improve upon.

[1]Humans only played against the Heuristic, not MCTS or DRL. The Heuristic is our strongest AI as can be seen in Figure 2.6.

2.2 Related Work

Few papers on Tetris Link exist in the literature. A single paper describes an experiment using Tetris Link [188]. This work is about teaching undergraduates "business decisions" using the game Tetris Link. To provide background on the game, we analyze the game in more depth in Section 2.3. The authors are aware of a similar game called *Blokus* [55]. It is an interesting game as the branching factor is so large (≈32928 for Blokus Duo on a 14x14 board) that specialized FPGA's have been applied to build good AI for it [126, 214]. Although visually similar, the gameplay is completely different, so Blokus strategies or heuristics do not transfer to Tetris Link.

The AI approaches that we try have been successfully applied to a variety of board games [203]. Heuristic planning has been the standard approach in many games such as Hex [278], Othello [231], Checkers [205], and Chess [120, 224]. MCTS has been used in a variety of applications such as Go and GGP [44, 64, 222]. DRL has seen great success in Backgammon [267] and Go [79, 246, 263]. Multi-agent MCTS has been presented in [95].

2.3 Tetris Link

Tetris Link, depicted in Figure 2.1, is a turn-based board game for two to four players. Just as the original Tetris video game, Tetris Link features a ten-by-twenty grid in which shapes called tetrominoes[2] are placed on a board. This paper will refer to tetrominoes as blocks for brevity. The five available block shapes are referred to as: *I*, *O*, *T*, *S*, *L*.[3] Every shape has a small white dot, also in the original physical board game variant, to make it easier to distinguish individual blocks from each other. Every player is assigned a color for distinction and gets twenty-five blocks: five of each shape. In every turn, a player must place precisely one block. A block fits if all of its parts are contained within the ten-by-twenty board. A player is skipped if they are unable to fit any of their remaining blocks. A player can never voluntarily skip if one of the available blocks fits somewhere in the board even if placing it is disadvantageous. The game ends when no block of any player fits into the board anymore.

The goal of the game is to obtain the most points. One point is awarded for every block, provided that it is connected to a group of at least three blocks. Not every

[2] A shape built from squares that touch each other edge-to-edge is called a polyomino [66]. Because they are made out of precisely four squares, these shapes are called tetromino [290].

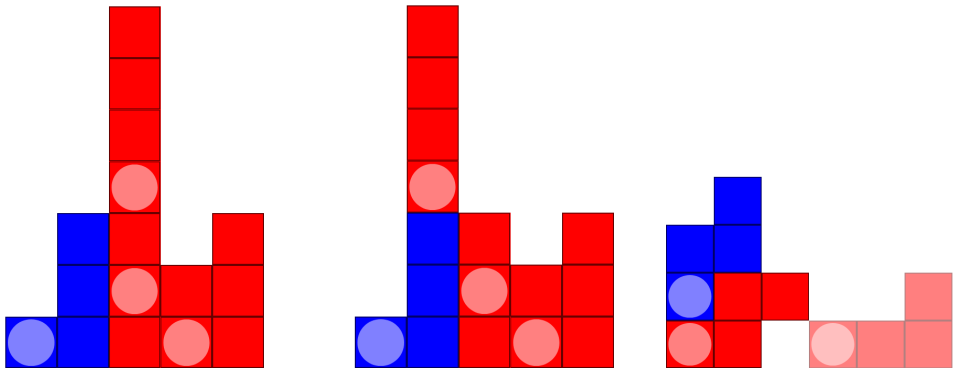
[3] The *S* and *L* blocks may also be referred to as *Z* [45] and *J* [49].

Figure 2.1: A photo of the original Tetris Link board game. The colored indicators on the side of the board help to keep track of the score.

block has to touch every other block in the group, as shown in Figure 2.2a.

The I block only touches the T but not the L on the far right. Since they together form a chained group of three, it counts as three points. Blocks have to touch each other edge-to-edge. In Figure 2.2b, the red player receives no points as the I is only connected edge-to-edge to the blue L.

A player loses one point per empty square (or hole) below block that was placed, with a maximum of two minus points per turn. Figure 2.2c shows how one minus point for red would look like. Moreover, the Figure underlines a fundamental difference to video game Tetris. In video game Tetris, blocks slowly fall, and one could nudge the transparent L under the S to fill the hole by precise timing of an action. In Tetris Link, one can only throw blocks into the top and let them fall straight to the bottom. In the original rules, a dice is rolled to determine which block is placed. If a player is out of a specific block, then the player gets skipped. Since every block could turn into one point, being skipped means potentially missing out on it. Although not a guaranteed disadvantage, as the opponent might also be skipped, the authors have abandoned the dice in their own matches as it resulted in too many games that felt unfairly lost.



(a) Three points for red **(b)** No points. **(c)** Minus point for red.

Figure 2.2: Small examples to explain the game point system.

The dice roll also makes Tetris Link a non-deterministic perfect information game. In this paper, there is no dice roll, so we analyze the deterministic version of Tetris Link. Note that we also focus on the two-player game only in this work. The three- and four-player versions are presumably even harder. In multiplayer games without teams, people might temporarily team up against the current leading player, which creates an unfair disadvantage [55]. Nevertheless, our web-based implementation for human test games[4] can handle up to four players and can provide an impression of the Tetris Link gameplay.

2.3.1 Verification that all games can fill the board

Each game of Tetris Link can be played to the end, in the sense that there are enough blocks to fill the whole board without leaving any empty squares. This can be seen in Equation 2.1: The board is ten squares wide and twenty squares high, so it can accommodate 200 individual squares. Every player has twenty-five blocks (α), each consisting of four squares (β). There are always at least two players playing the game (γ), so they are always able to fill the board.

$$\alpha \cdot \beta \cdot \gamma = 25 * 4 * 2 = 200 \tag{2.1}$$

[4] https://hizoul.github.io/contetro

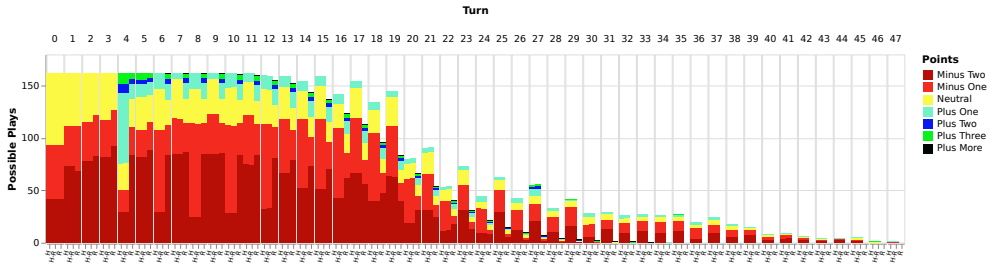


Figure 2.3: This graph underlines the game's difficulty, especially for search algorithms, by showing the average number of possible moves and their associated effect on the player's score. The agent name abbreviations stand for: R = Random, H-R = Random Heuristic, and H = User Heuristic. See Section 2.4.1 for an explanation of how the heuristic agents work.

2.3.2 Game complexity

An essential metric for search algorithms is the so-called branching factor, which specifies the average number of possible moves a player can perform in one state [81]. In order to compute this number, we look at the number of orientations for each block. The I block has two orientations as it can be used either horizontally or vertically. The O block has only one orientation because it is a square. The T block has four different orientations for every side one can turn it to. The L and S are special cases as they can be mirrored. The L has four and the S two sides one can rotate it to. Mirroring them doubles the amount of possible orientations. Hence, in total, nineteen different shapes can be placed by rotating or mirroring the available five base blocks. Since the board is ten units wide, there are also ten different drop points per shape. In total, there can be up to 190 possible moves available in one turn. However, the game rules state that all placed squares have to be within the bounds of the game board. Twenty-eight of these moves are always impossible because they would necessitate some squares to exceed the bounds either on the left or right side of the board. Therefore, the exact number of maximum possible moves in one turn for Tetris Link is 162. Since the board gets fuller throughout the game, not all moves are always possible, and the branching factor decreases towards the end. In order to show this development throughout matches, we simulate 10,000 games.

We depict the average number of moves per turn in Figure 2.3. For the first eight to ten turns, all moves are available on average. Not depicted in the Figure but based on the data, this only holds true until turn 6. After that, the number of plays may already decrease. Tetris Link is a game of skill: random moves perform badly. A game consisting of random moves ends after only thirty turns. Many holes with many minus

Agent	Random	Random-H	User-H	Tuned-H
Win Rate	48.16%	47%	71.65%	70%
Unique Games	10,000	10,000	7	50

Table 2.1: First move advantage, over 10,000 games. The first six (#1) or all turns (#2) are compared for uniqueness to see whether the same games keep repeating. The -H in the agent name stands for Heuristic (see Section 2.4.1). Draws are not counted towards wins.

points are created, and the game ends quickly with a low score. The heuristic bars show that simple rules of thumb fill the board most of the time by taking more than forty turns. Furthermore, the branching factor in the midgame (turn 13-30) declines slower and hence offers more variety to the outcomes. Another thing that can be seen in Figure 2.3 is that only very few select plays can actually lead to positive points. Most of the turns will leave the score unchanged or even decrease it. A player would only consider taking minus points if it would cost the opponent even more points than the player loses, or there are no other options. To further underline this, we did an exhaustive tree search until depth 5, which is the first turn player one is able to achieve points. Of the over 111 Billion possibilities (111577100832), only $\approx 14.36\%$ allow for an increase in points. Moreover, only $\approx 3.06\%$ allows the optimal three-point gain, which the first player aims for. $\approx 69.08\%$ of the outcomes result in a point disadvantage, and in $\approx 16.55\%$ of the cases, no points have been gained nor lost due to blockage by the opponent or lack of connectivity.

We are now ready to calculate the approximate size of the game tree complexity for the deterministic variant of Tetris Link, in order to compare it to other games. On average, across all three agents shown in Figure 2.3, a game takes 37 turns and allows for 74 moves per turn ($74^{37} \approx 1.45 * 10^{69}$). The game tree complexity is similar to Chess (10^{43} or 10^{123}) but smaller than in Go (10^{360}) [161].

2.3.3 First move advantage

An important property of turn-based games is whether making the first move gives the player an advantage [289]. To put this into numbers, we let different strategies play against themselves 10,000 times to determine if the starting player has an advantage. The first six (#1) or all (#2) moves are recorded and checked for uniqueness.

As can be seen in Table 2.1, the win rate for *random heuristic* as starting player is almost 50%. Although the win rate for the first player is higher for the *tuned heuristics*, these numbers are not as representative because the heuristic repeats the same tactics resulting in only seven or twenty-nine unique game starts. If we repeat the same few

games, then we will not truly know whether the first player has a definite advantage. Especially considering that at least until turn six, all moves are always possible, there are around 10^{13} or 18 Trillion ($BranchingFactor^{Turns} = 162^6 = 18,075,490,334,784$) possible outcomes. Since the *random heuristic* has more deviation and plays properly as opposed to random moves, we believe that it is a good indicator of the actual first player advantage. Note that 47% is close to an equal opportunity. Different match history comparisons of Chess measure a difference of around two to five percent in win rate for the first player [289]. However, since neither Tetris Link nor Chess have been mathematically solved, one cannot be certain that there is a definite advantage.

2.4 AI Player Design

In this section, we describe the three different types of AI players that we implemented, based on heuristics, MCTS, and RL, respectively. For the experiments (Section 2.5), the game is coded in Rust and JavaScript (JS). The Rust version is written for faster experiments with MCTS and RL, and the JavaScript version is written to visually analyze games and also do a human play experiment. Both implementations share a common game log format using JSON in order to enable interoperability. To underline the importance of a performance-optimized version, we measured the average time it takes to simulate one single match where always the first legal move is made. The Rust implementation requires 590μs for that, whereas the JavaScript implementation needs 82ms.

2.4.1 Heuristic

We now describe the design of our heuristic player. A heuristic is a rule of thumb that works well most of the time [221]. For Tetris Link, we identify four heuristic measures: the number of connectable edges, the size of groups, the player score, and the number of blocked edges. The number of blocked edges is the number of edges belonging to opponents that are blocked by the current players' blocks. All heuristic values are positively related to the chance of winning.

Each parameter is multiplied by a weight, and the overall heuristic score is the sum of all four weighted values. For every possible move in a given turn, the heuristic value is calculated, and the one with the highest value is chosen. If multiple moves have the same maximum value, a random one of these best moves is chosen, which is why in Table 2.1, the heuristics play more than one single unique repeating game.

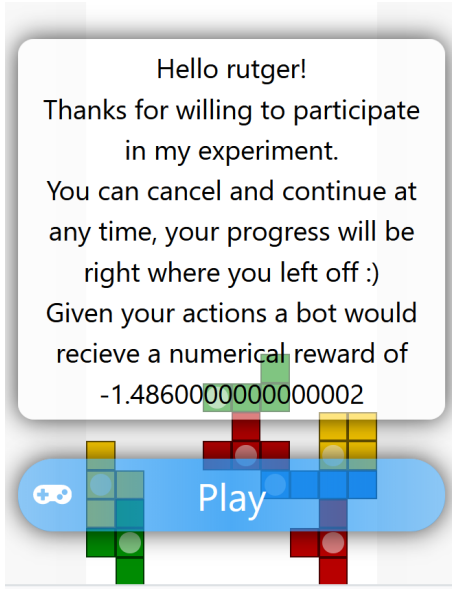


Figure 2.4: The web interface to measure the effectiveness of our Heuristic agent against actual human players familiar with the game.

The initial weights were manually set by letting the heuristic play against itself and detecting which combination would result in the most points gained for both players. We refer to this as *user heuristic*. We then use Optuna [3], a hyperparameter tuner, to tune a set of weights that reliably beat the *user heuristic*. This version is called *tuned heuristic*. To achieve a greater variety in playstyle, we also test a *random heuristic* which at every turn generates four new weights between zero to fifteen. To have an estimate on the performance of the heuristic, we let actual human players (n=7) familiar with the game play against the heuristic via the JavaScript implementation (Figure 2.4). The *random heuristic* achieved a win rate of 23.07% across 13 matches. and the *user heuristic* a win rate of 33.33% across six matches. The sample size is very small, but it still indicates that the heuristic is not particularly strong. This is supported by a qualitative analysis of the game played by the authors, based on our experience. We conclude that our heuristic variants play the game in a meaningful way but are not particularly strong.

2.4.2 MCTS

For applications in which no efficient heuristic can be found, MCTS is often used, as it constructs a value function by averaging random roll-outs [44]. Our MCTS implementation uses the standard UCT selection rule [143]. As further enhancements, we also use MCTS-RAVE [44] and MCTS-PoolRAVE [219] to see whether the modifications help in improving the quality of results. Furthermore, we experimented with improving the default (random) policy by replacing it with the heuristic. However, the heuristic calculation is so slow that it only manages to visit ten nodes per second. We want to stress that this is only the MCTS heuristic that is slow as it was not optimized for speed at all. The game simulation itself is fast processing a full match of random actions in on average 590μs, so up to around 1694 matches per second.

MCTS is well-suited for parallelization, leading to more simulations per second and hence better play [44]. We implemented tree parallelization, a frequently used parallelization [85]. In tree parallel MCTS, many threads expand the same game tree simultaneously. Using 12 threads, we visit 16258 nodes per second on average with a random default policy. To put this into perspective, this is $1.63e^{-9}$% of all 10^{13} possibilities in the first six turns. Thus, only a small part of the game tree is explored by MCTS, even with parallel MCTS.

2.4.3 Reinforcement Learning Environment and Agent

A RL environment requires an observation, actions, a reward [136], and an RL agent with an algorithm as well as a network structure. To prevent reinventing the wheel, we use existing code for RL, namely OpenAI gym [39] and the stable-baselines [117], which are written in Python. To connect Python to our Rust implementation, we compile a native shared library file and interact with it using Pythons *ctypes*. As RL Algorithm, we exclusively use the DRL algorithm PPO2 [234]. For the network structure, we increase the number of hidden layers from two layers of size 64 to three layers of size 128 because increasing the network size decreases the chances of getting stuck in local optima [148]. We do not use a two-headed output, so the network only returns the action probabilities but not the certainty of winning as in AlphaZero [248].

The observation portrays the current state of the game field. Inspired by AlphaGo, which includes as much information as possible (even the "komi"[5]), we add additional information such as the number of blocks left per player, the players' current score, and which moves are currently legal. For the action space, we use a probability distribution

[5] *Komi* refers to the first turn advantage points [246].

Reward Type	Steps	Episode Reward	Score
Guided	3183.49	-0.17	-5.6
Simple	10000.0	-0.0	-12.25
Score	6214.45	-0.09	-6.88

Table 2.2: Results of self-play with different reward types until either a local optimum or 10,000 steps have been reached. Step, Reward and Score show the average of all seeds.

over all moves. The probabilities of illegal moves are set to 0, so only valid moves are considered. For the reward, we have three different options. They are calculated using the current players score (δ), the size of a connected group (ϵ) and a scolding parameter (ζ)

1. Guided: $\frac{\delta+\epsilon}{100} - \zeta$
2. Score: $\frac{\delta}{100}$
3. Simple: ± 1 depending on win / loss

The *Guided* reward stands out because it is the only one that reduces the number of points via *scolding (ζ)*. If the move with the highest probability is illegal, then the reward will be reduced by -0.004, and the first legal move with the highest probability is chosen. This should teach the agent to only make valid moves. This technique is called reward shaping, and its results may vary [106].

In order to detect which one of the three options is the most effective, we conduct an experiment. Per reward function, we collect the averages for the number of steps it took, the average reward achieved, and what the average score of the players was in the results. Our results, shown in Table 2.2, indicate that the *Guided* reward function works best. It only takes around 3183 steps on average to reach a local optimum, and the average scores achieved in the matches are the highest. We define a local optimum as the same match repeating three times in a row. The *Score* reward function also lets the agent reach a local optimum, but it takes twice as long as the *Guided* function, and the score is slightly lower as well. The *simple* reward function seems unfit for training. It never reached a local optimum in the 10,000 steps we allowed it to run, and it got the lowest score in its games.

2.5 Agent Training and Comparison

For our experimental analysis, we first look at the performance of the MCTS agent (Section 2.5.1) and the training process of the RL agents (Section 2.5.2). Finally, we

compare all previously introduced agents in a tournament to analyze their play quality and determine the currently best playing approach.

2.5.1 MCTS Effectiveness

Setup

Initial test matches of MCTS against the *user heuristic* resulted in a zero percent win rate, and a look at the game boards suggested near-random play. We use a basic version of MCTS with random playouts because using our heuristic as guidance was too slow. AlphaZero has shown that even games with high branching factors such as Go can be played well by MCTS when guided by a neural network [248]. However, without decision support from a learned model or a heuristic, we rely on simulations. In order to see if this guidance is the reason for bad MCTS performance, we abuse the fact that the *user heuristic* plays very predictably (Section 2.3.3). We use the RAVE-MCTS variant (without the POOL addition), pre-fill the RAVE values with 100 games of the *user heuristic* playing against itself, and then let the MCTS play 100 matches against the *user heuristic*. We repeat this three times and use the average value across all three runs. We run this experiment with different RAVE-β parameter values. This parameter is responsible for the exploration/exploitation balancing and replaces the usual UCT C_p parameter. The closer the RAVE visits of a node reach RAVE-β, the smaller the exploration component becomes. Furthermore, we employ the slow heuristic default policy at every node in this experiment. We simulate one match per step because otherwise, the one-second thought time is not enough for the slow heuristic policy to finish the simulation step.

Results

Our MCTS implementation can play well with a decent win rate against the user-heuristic, as shown in Figure 2.5. This result underlines that in games with high branching factors, MCTS needs good guidance through the tree in order to perform well. Figure 2.3 also supports this as there are very few paths that will actually lead to good plays. The declining win rate with a higher RAVE-β value suggests that exploration on an already partially explored game tree worsens the result because the opponent does not deviate from its paths. The rise in win rate for a β value of 5000 after the large drop in 2500 underlines the effect of randomness involved in the search processes.

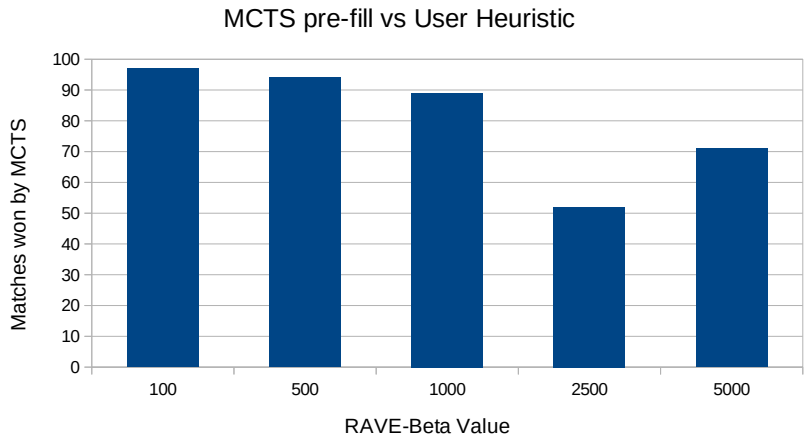


Figure 2.5: Win rates of pre-filled MCTS playing against the *User-Heuristic* compared by the RAVE-β parameter.

Even though the heuristic-supported playout policy works well, we will still use a random playout policy for the tournament (Section 2.5.3). Pre-filling the tree is very costly and would provide an unfair advantage to the MCTS method.

To underline that we believe the bad performance stems from the large branching factor and few paths that lead to positive points, we test our implementation on the game of Hex. In different board sizes (2x2 to 11x11) of Hex, our MCTS plays against a shortest path heuristic. The result is striking: as long as the branching factor stays below 49 (7x7), MCTS wins up to 90% of the matches. For larger branching factors, the win rate drops to 0% quickly.

2.5.2 RL Agents Training

Agents

We define an RL agent as the combination of reward type, algorithm, and training opponent. We use the guided reward function because it worked best in our experiment and call this agent *RL-Selfplay*. (This is a neural network only RL, without MCTS to improve training samples, that plays against itself [Selfplay].)

In addition to this rather simple agent, we introduce the *RL-Selfplay-Heuristic* agent. It builds on a trained *RL-Selfplay* agent where we continue training by playing against the heuristic. Observation and reward are the same as for *RL-Selfplay*.

From the first turn advantage experiment, we know that the heuristic plays well even with random weights. That is why we also introduce an agent called *RL-Heuristic*. This agent outputs four numbers that represent the heuristics weights (Section 2.4.1). We use a modified version of the guided reward function using the players own score (δ), the opponents score (η) and the size of a connected group (ϵ):

$$\frac{(\delta - \eta) + \epsilon}{100} \tag{2.2}$$

Group size stands for the total number of blocks that are connected with at least one other block. This is added because we want the algorithm to draw a connection between the number of points gained and the number of connected blocks. However, mainly the difference in points between itself and the opponent is used as a learning signal, so it aims to gain more points than the opponent. Scolding is not necessary anymore as we do not have to filter the output in any way.

Setup

In this section, we detail the training process of the RL agents. Each training is done four times, and only the best run is shown. Agents are trained with the default PPO2 hyperparameters, except for *RL-Heuristic*, which uses hand-tuned parameters.

When playing only against themselves, the networks still quickly reached a local optimum even with increased layer size. This optimum manifested in the same game being played on repeat and the reward per episode staying the same. This repetition is a known problem in self-play and can be called "chasing cycles" [279]. To prevent these local optima, we train five different agents against each other in random order. To be able to train against other agents, we modified the *stable-baselines* code.

Results

The training process for *RL-Selfplay* peaked around 1.5 million steps. For *RL-Selfplay-Heuristic* we use the two best candidates from *RL-Selfplay*, namely #3 after one million steps with a reward of 0.04 and #1 after 1.5 million steps with a reward of 0.034. The training of *RL-Selfplay#1-Heuristic* reaches its peak after 3.44 and *RL-Selfplay#3-Heuristic* after 3.64 Million steps with a reward of 0.032 and 0.024. These are our first RL agents that can achieve a positive reward while playing against the heuristic.

The *RL-Heuristic* training worked well, achieving mostly a positive reward. However, looking at the output values, we realize the reward function design was unfortu-

nate. It sets all weights to zero, except for the enemy block value to fifteen and the number of open edges between four and seven. So by negating the player's score with the opponent's score, we have unwillingly forced the heuristic to focus on blocking the opponent over everything else. Needless to say, with these weights, the *RL-Heuristic* rarely wins. Although it manages to keep the opponent's score low, it does not focus on gaining points which leaves it with a point disadvantage.

2.5.3 Tournament

Setup

In the tournament, all presented AI approaches play against each other. Every bot will face every other bot in 100 matches. We have five different RL bots, three MCTS bots, and three heuristic bots. Each agent gets at most one second of time to decide on their move, and they are not allowed to think during the opponent's turn. Every bot will play half of its matches as first and the other half as second player. The bot's skill will be compared via a Bayesian Bradley Terry (BBT) skill rating [287]. The original BBT [287] ratings range from 0 to 50. By adjusting the BBT-β parameter, we change this to represent the ELO range (0 to 3000) [103].

Results

The final skill rating is portrayed in Figure 2.6. The three heuristic agents take the top 3, followed by RL and MCTS. Remarkably, the *tuned heuristic* performed best, even though it is only optimized to play well against the *user heuristic*, but yet it performs best across all agents.

Seeing *RL-Heuristic* as the best *RL* approach shows that the other RL agents are far from playing well. Yet all RL agents consistently beating MCTS with random playouts proves that the agents definitely learned to play reasonably.

It is interesting to see that the MCTS-UCB (14% win rate) variant performed best because the other two variants [RAVE (0.02%), PoolRAVE (0.04%)] were conceived in order to improve the performance of UCB via slight modifications [219]. Please note that this poor performance in Figures 2.6 and 2.7 is caused by the tree not being pre-filled as it was in Figure 2.5 (Section 2.5.1). This shows the importance of pre-filling MCTS in Tetris Link due to the few paths that result in positive points (see Figure 2.5).

The skill rating omits information about the quality of the individual moves. To gain further insight into that, we provide Figure 2.7. Here, we can see that every

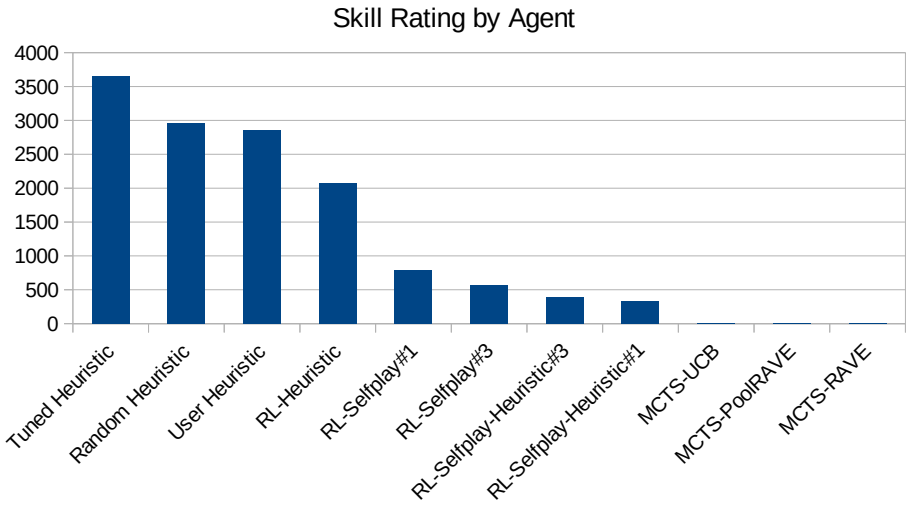


Figure 2.6: The skill rating of the agents that participated in the tournament. MCTS uses a non pre-filled tree, resulting in bad performance (Section 2.5.1).

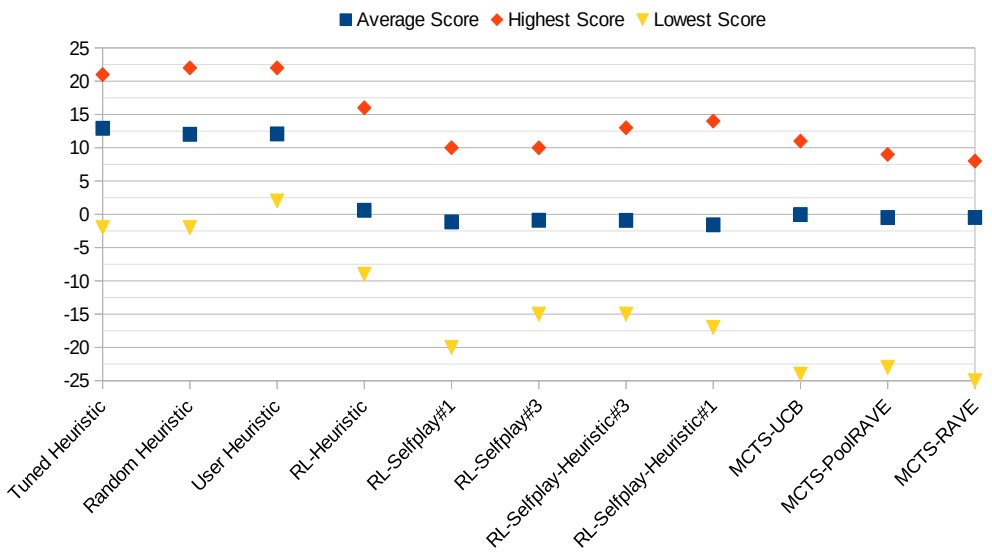


Figure 2.7: Visualisation of the scores that agents achieved in the tournament. Agents are sorted by the skill rating in Fig. 2.6.

agent manages at least once to gain 8 points or more. This means that every agent had at least one match it played well. Looking at the lowest achieved scores and average scores, we find that every agent, except for the pure heuristic ones, plays badly, considering that they only make ±3 points on average.

2.6 Conclusion and Future Work

Board game strategy analysis has been done for decades, and especially games like Chess and Go have seen countless papers analyzing the game, patterns, and more to find the best play strategies [248]. We contributed to that field by taking a close look at the board game Tetris Link. While the strategy is key to winning, some games, such as Hex, give the first player a definite advantage. We have experimentally shown that there is no clear advantage for the starting player in Tetris Link (Section 2.3.3).

We have implemented three game-playing programs based on common approaches in AI: heuristic search, MCTS, and RL. Despite some effort, none of our rule-based agents was able to beat human players.

In doing so, we have obtained an understanding of why it may be hard to design a good AI for Tetris Link:

- Especially at the beginning, the branching factor is large, staying at 162 for at least the first six turns.
- The majority of possible moves results in minus points making the number of good moves diminishingly small (see Figure 2.3 and Section 2.3.2).
- Mistakes / minus points can hardly be recovered from. The unforgivingness for these moves may make it harder to come up with a decent strategy, as generally postulated by [63].
- Many rewards in the game stack — they come delayed after multiple appropriate moves because groups of blocks count and not single blocks. This makes it especially hard for MCTS and RL to learn the correct sequences.

All this holds true for the simplified version we treat here: no dice, only two players. Adding up to two more players and dice will also make the game harder.

With a solid understanding of the game itself, we investigated different approaches for AI agents to play the game, namely heuristic, RL and MCTS. We have shown that all tested approaches can perform well against certain opponents. The best currently

known algorithmic approach is the tuned heuristic, although it can not consistently beat human players.

Training an RL agent (Section 2.5.2) for Tetris Link has proven to be complicated. Just getting the network to produce positive rewards required much trial and error, and in the end, the agent did not perform well even when consistently achieving a positive reward. We believe the learning difficulty in Tetris Link comes from the many opportunities to make minus points in the game. One turn offers usually offers one plus to three points, or six at most if two groups are connected, but that means that multiple previous turns that were well planned and gave zero points if not even more minus points had to be made. Hence recovering from minus points is difficult, meaning small mistakes have graver consequences.

Although MCTS performed poorly in our tournament, we have shown that with proper guidance through the tree, MCTS can perform nicely in Tetris Link and Hex (Section 2.5.1). That is why a combination where RL guides an MCTS through the tree might work well, e.g. AlphaZero [248] or MoHex v3 [98], and is something to try in future work.

As computer game researchers, we found ourselves challenged to create a good agent to play the game Tetris Link. We tried a large variety of classic approaches and were not able to achieve the results we hoped for. We invite the research community to use our code and improve upon our approaches.[6]

[6]https://github.com/Hizoul/tetris-link-research

Chapter 3

Procedural Content Generation: Better Benchmarks for Transfer Reinforcement Learning

The idea of transfer in reinforcement learning (TRL) is intriguing: being able to transfer knowledge from one problem to another problem without learning everything from scratch. This promises quicker learning and learning more complex methods. To gain an insight into the field and to detect emerging trends, we performed a database search[a]. We note a surprisingly late adoption of deep learning that starts in 2018. The introduction of deep learning has not yet solved the greatest challenge of TRL: generalization. Transfer between different domains works well when domains have strong similarities (e.g. MountainCar to Cartpole), and most TRL publications focus on different tasks within the same domain that have few differences. Most TRL applications we encountered compare their improvements against self-defined baselines, and the field is still missing unified benchmarks. We consider this to be a disappointing situation. For the future, we note that: (1) A clear measure of task similarity is needed. (2) Generalization needs to improve. Promising approaches merge deep learning with planning via MCTS or introduce memory through LSTMs. (3) The lack of benchmarking tools will be remedied to enable meaningful comparison and measure progress. Already Alchemy and Meta-World are emerging as interesting benchmark suites. We note that another development, the increase in procedural content generation (PCG), can improve both benchmarking and generalization in TRL.

[a]This chapter is based on the publication [176] Müller-Brockhausen, M., Preuss, M., Plaat, A.: Procedural content generation: Better benchmarks for transfer reinforcement learning. In: 2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021. IEEE (2021), https://doi.org/10.1109/CoG52621.2021.9619000

3.1 Introduction

The need for TRL is growing through increased usage of DRL as shown in Figure 3.1. Deep networks are expensive to train [193], so cutting down the learning time by re-using previously gained knowledge is desirable. Computer games are currently often used as benchmarks or challenging test systems, and here it is especially the case that changes of different amplitude (e.g. patches) happen regularly. Recent AI successes on well-known complex games such as Dota2 [28] and StarCraft II [279] show that constant retraining is necessary as the underlying systems evolve on the same time scale as the trained AI systems.

Although the need for knowledge transfer in a reliable manner is clear, most experiments show limited generalization, and progress in TRL is limited. Our main goal in this paper is to detect why this is the case and how it can be cured. It turns out that a major problem lies in the sparsity of suitable benchmarks, and we see the use of PCG as a recommended solution to this problem.

This paper has the following contributions.

1. We categorize the literature of TRL, finding many different approaches and applications
2. The absence of clear benchmarks and a clear research agenda is noted
3. We provide a research agenda in which we stress the need for a clear measure of success, clear benchmarks, and suggest that PCG is ideally suited to provide such benchmarks for transfer reinforcement learning

Section 3.2 introduces related work (a meta-survey). To gain an unbiased insight into TRL, we scrape through a dataset (Section 3.3). We explain the experimental parameters and decisions involved in TRL and identify trends in their usage (Section 3.4). After categorizing a large number of transfer experiments, we report the generalization capabilities of TRL (Section 3.4.3). We draft a research agenda for TRL (Section 3.5) that lists current limitations and identifies directions that the field is likely headed to.

3.2 Related Work

Some summaries on individual aspects of TRL exist, and we will introduce them in chronological order. The first one was written by Bone in 2008 [35], still pre-deep learning. A year later, Taylor & Stone followed [265] with a comprehensive survey.

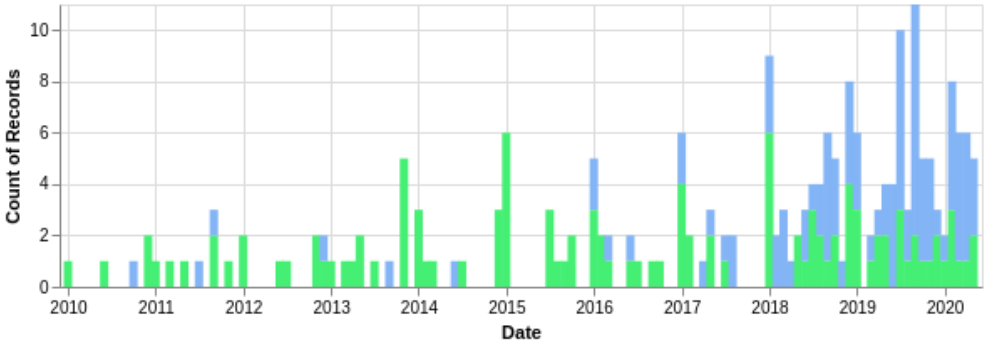


Figure 3.1: Number of published papers per month. Light blue indicates entries that make use of deep neural networks, and green ones do not. The years 1985 to 2009 are cut off for readability, but the full graph is available at [180].

Its authors are the two most recurring names in the field in our data set. Two years later, they published a second survey with a focus on inter-task transfer [266]. In 2012, Lazaric formulated a framework [149] that enables the categorization of TRL experiments similar to [265]. Seven years later, in 2019, Da Silva & Costa published a survey focused on multiagent RL [243]. In 2020, multiple surveys appeared. One on curriculum learning in RL [182], one about multi-task transfer [280], and one about transfer in deep reinforcement learning [304]. Multiple surveys in one year might seem peculiar, but a look at Figure 3.1 shows a large increase in publications starting in 2018.

These surveys formulate frameworks to encompass the different TRL literature they encounter [149, 265], or analyze specific sub-fields of TRL [182, 243, 266, 280, 304]. We aim for a comprehensive overview of all aspects of TRL. We perform scraping through the Microsoft Academic Graph [250], which contains over 209 Million papers, and include papers based on keywords in title and abstract. This yields some interesting statistics. For example, $\approx 74.8\%$ of the 270 relevant papers in our data set have not been included in any previously mentioned survey [35, 149, 182, 243, 265, 266, 280, 304]. Moreover, it enables visualizations such as a publication timeline (Figure 3.1), a social network graph (Figure 3.3), and the creation of an interactive web tool [180] that facilitates the re-use of the data. Automatic quantitative analysis is to be seen as an addition to produce insightful figures and tools. It also served as tool to gain insight into the state of the field. But to form a true vision for our research agenda (Section 3.5) we still relied on manual research.

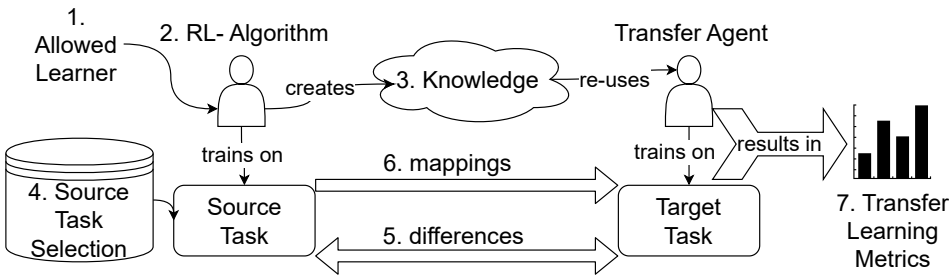


Figure 3.2: A visualization of the transfer learning process. The chosen allowed learner (1) / algorithm (2) combination generates knowledge (3) while training on a selected source task (4). Source and target task have to differ (5), and depending on how large the differences are, task mappings (6) might be required. By using previously gained knowledge, the transfer agent then trains on the new task. Gathered data from source and target training can be distilled into the transfer learning metrics (7).

3.3 Method

We search through a snapshot [163] of the Microsoft Academic Graph (MAG) [250] for entries that contain the three words "transfer," "reinforcement" and "learning" in either their abstract or title. We create a spreadsheet using the terminology introduced by [265]: transfer dimension, allowed task differences, source task selection, task mappings, transferred knowledge, allowed learners, transfer metrics. We also record which RL algorithm was used (e.g., Q, DQN, PPO, DDPG), whether the paper publishes additional resources such as source code, and we check if the links work [305]. All data and resources, including the spreadsheet, and an interactive data viewer, are openly available at [180].

Figure 3.2 provides a visualization of the process of a TRL experiment. For terminology, we stay as close as possible to [265]. The RL-algorithm (2) provides more details about the learning algorithm that is used (1) and how transferred knowledge is re-used. We briefly review which information was gathered from the papers. The content in parentheses behind keywords refers to the transfer process step in Figure 3.2 if numerical, or otherwise the abbreviation used in the spreadsheet. By explaining the Figure, we also follow the workflow of setting up a typical TRL experiment, and we describe essential choices of the authors.

The allowed learner (1) places restrictions on how transfer is approached and influences experimental parameters such as the pool of available reinforcement learning methods (2). The options are temporal difference learning (TD), model-based learn-

ing (MB), relational learning (RRL), hierarchical learning (H), batch learning (Batch), Bayesian methods (bayes), and case-based reasoning (CBR). Because many different algorithms exist per allowed learner method, we also note which RL-algorithm was used (2). The next step is to determine the type of knowledge (3) to transfer. The easiest methods re-use what was learned, e.g., the action-value function (Q), policy (π), task model (model), found prior distributions (pri), or experience instances (I).

However, higher-level knowledge can also be used in a variety of ways. One can extract partial policies (π_p) or options (options) for guidance. Rules or advice from experts (ra), be it human demonstration or successfully trained agents, can guide the training process. This advice can manifest itself in a shaped reward (R). Some algorithms can identify and learn important features (fea), autonomously find sub-task definitions (sub), or build a proto-value function (pvf). Other methods to transfer knowledge are a Variational auto-encoder (VAE) [294] or policy distillation [22].

To gather the re-usable knowledge,[1] an agent needs to train on a selected source task (4). Either all previously seen tasks are used (all), one is selected by the author (h), a library of tasks to choose from is defined by the author (lib), or the agent has to modify the source task to gain the required knowledge autonomously (mod). There are two important factors on transfer between the source and target task. The first are differences (5) an agent has to handle between tasks. These can be small, such as an alternating start (s_i) or end (s_f) position, another level layout, or a different number of encountered objects (#). Changes can also affect the number of involved objects (#), transition function (t), state variables (v), the action set (a), or reward function (r). Secondly, mapping between tasks (6) could be required. The agent can get no mapping (N/A) or learn it from experience (exp). The mapping can also be provided as higher-level knowledge (T), created by humans (sup), derived from action mapping (M_a), or a grouping of state variables (sv_g).

The transfer experiment results in metrics (7) that indicate success. One can measure an improvement in the Time to Threshold (tt), so how many fewer steps did the agent have to train to reach a previously specified reward threshold. If the transfer agent training starts at a higher reward than an agent that started from scratch, one has measured a Jumpstart (j). The transfer agent can also achieve a higher total reward (tr), and the difference between transfer agent and training from scratch is called transfer ratio[2]. Lastly, Asymptotic Performance (ap) indicates whether the

[1] Simplified for the Figure. Knowledge used to train the transfer agent can stem from anywhere, like human made demonstrations or rules.

[2] We omit the transfer ratio as performance improvement measurement in our data and in Figure 3.4, as it is an extension of the total reward [7].

final learned performance has improved.

3.4 Analysis

To get an overview of the dataset, the social network structure of the citation data is visualized (section 3.4.1), and insights from the categorization are presented (section 3.4.2). The main goal of transfer learning is generalization. We analyzed the papers for the strength of generalization that has been achieved in TRL (section 3.4.3).

Many more insights have been found, and we refer the reader to discover the full data on an interactive website [180].

3.4.1 Social Network Structure

Figure 3.3 shows a visualization of citations between authors. Colors indicate different communities, as identified by the community detection algorithm [33]. Note that due to noise and missing values in the base data, 63 ($\approx$ 23%) of the entries are missing reference data and are therefore not present in Figure 3.3.

Eight connected components have been identified. Only one of these contains most entries. The other seven are independent. The independent components are not included in Figure 3.3. The largest component consists of nine individual communities. For each community, we attempt to identify common aspects and succeeded for four of them. Two communities focus on the different types of allowed learners. Red at the far left contains mostly case-based reasoning (CBR), and purple at the bottom left hierarchical (H) and Bayesian RL (bayes). The dark green community at the far right contains mostly Simulation to Reality (Sim2Real) experiments. From here on out, the similarities are already declining. As for the black group, the only link we could find is that they all applied Q learning at some point.

In addition to hierarchical learners, purple also contains numerous 2D navigation experiments, but not exclusively. Dark blue at the bottom contains many reward shaping experiments that sample from a (human) demonstration or world model prediction.

For the remaining five communities, no real common aspects could be found, except that $\approx$ 44% in the orange community focus on real-world engineering problems such as optimizing power usage for buildings [167], air conditioners [153], or collision avoidance for autonomous vehicles [164].

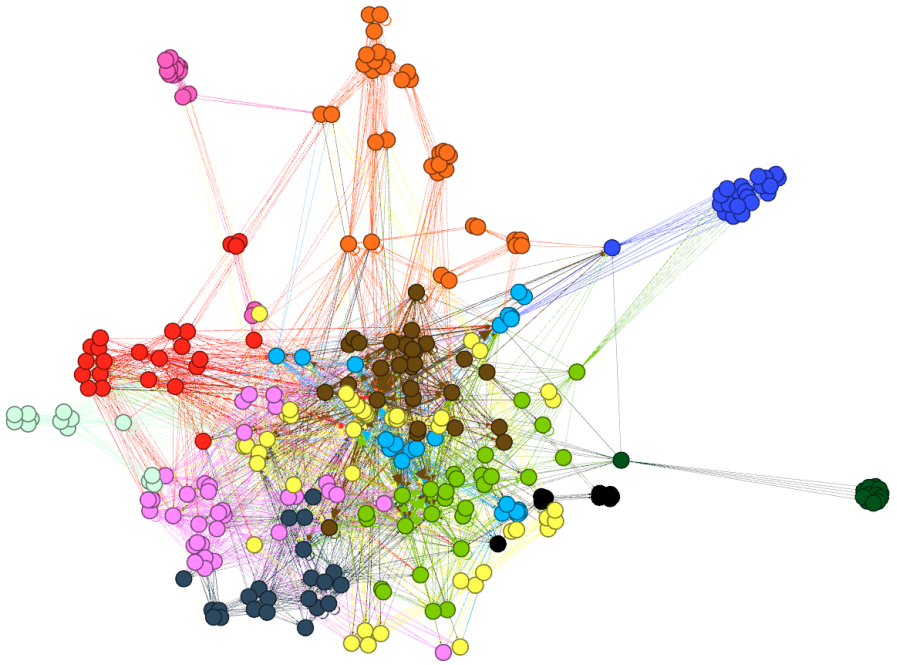


Figure 3.3: Directed Social Network Graph of Authors citing each other. The red group contains mostly case-based reasoning learners, purple contains hierarchical algorithms and Bayesian RL, dark green is the Sim2Real community, black papers all applied Q-learning, dark blue contains reward shaping, orange reports mostly on applications (such as energy consumption in buildings). Layout determined by Force Atlas [125].

3.4.2 Category Data

Of the 270 entries, 202 ($\approx 74.8\%$) have not been in previous surveys of the field [35, 149, 182, 243, 265, 266, 280, 304]. In the following, parentheses will indicate the number of papers in the dataset relating to a specific message, for example 10% (27) entries have not been approved through peer-review (23 arxiv, 4 rejected on openreview). Text in brackets refers to variable abbreviation presented in Section 3.3. Furthermore, one entry may contain multiple tags per category. We follow the order of the steps of the transfer learning process of Figure 3.2) in presenting the data.

First, the allowed learner is chosen. The majority of papers uses regular temporal difference methods (241) [TD], followed by hierarchical learning (46) [H], model-based learning (31) [model], Bayesian learning (20) [Bayes], batch learning (13), relational learning (11) [RRL], policy search learning (10) [PS], case-based reasoning (9) [CBR], and one linear programming entry. TRL transfers knowledge between different tasks, and it is no surprise that hierarchical learning is the second most applied learner type for TRL because the multiple tasks might be hierarchically related. Moreover, singular large tasks can be decomposed into multiple (hierarchically ordered) sub-tasks for transfer [265].

The allowed learner narrows the pool of RL algorithms that can be chosen. The most popular algorithms are tabular Q-Learning (124), DQN (36), SARSA (28), DDPG (11), PPO (10), FQI (10), DDQN (8), A3C (7), LSPI (7), and Policy Gradient (6). The high occurrence of tabular Q-Learning could give the impression that deep learning is not prevalent in TRL yet, but $\approx 36\%$ (99) already use deep neural networks. Moreover, Figure 3.1 depicts a clear trend towards deep learning that started in 2018.

Although the number of 3D environments that deep neural networks (DNN) (17) are applied to are almost the same as for tabular algorithms (14), their complexity differs. Tabular algorithms focus mostly on balancing problems, such as Mountain Car 3D (11) or controlling joints in a real-world RoboCup robot [23, 52]. The 3D environments that DNN's are applied to are more complex. AirSim requires full free 3D navigation through flying, plus the policy is transferred to a real drone [298], and Mujoco combines controlling multiple joints with moving in a 3D World [293].

The third step (see Figure) is to decide what knowledge should be extracted or transferred. The most popular methods are also the easiest to transfer, namely just re-using the previously learned action-value function (72) [Q] or policy (64) [π]. All other methods require more sophistication, such as extracting and re-using relevant features

(38) [fea] or guiding training via Advisor / Teacher data (22) [advisor]. Shaping the reward (20) [R] is also an often-used means to transfer knowledge. Less often used are approaches like collecting experience instances (14), building a task model (14) or sub-task definition (8), defining rules (14), finding options (12), or collecting distribution priors (9). There are also new ways to transfer knowledge between tasks. One of these is policy distillation (4). Most entries used it to summarize multiple learned policies into a single one [22, 292, 297], but it can also be used for simulation to reality transfer [275]. The distillation process and the advisor method have one element in common: They both re-use the same type of saved knowledge. Another transfer method that was introduced through deep neural networks is the Variational Auto Encoder (VAE). These networks can help to automatically identify relevant features in the latent space and thus allow for universal control policies [294]. Moreover, Q-functions can be generated algorithmically [14]. This is related to hyper-networks, where a neural network outputs the weights used in another network [256].

The fourth step is determining the source task from which knowledge should be transferred. Here most approaches perform hand-selection (137) [h]. 48 entries let the algorithm use all tasks (all), 32 a library of selected tasks (lib), and two times, the agent automatically modifies a provided source task (mod).

The fifth and most important question to answer is: What kind of differences in tasks will the learner have to handle? The most frequently occurring differences are in transition dynamics (115) [t] of the environment. Different transition dynamics refer to changes in parameters between tasks that influence how the world changes per timestep. For example, if the agent were to control an airplane, changes in gravity, weight, or friction would count as transition dynamic differences. The second to fourth place goes to navigation-related tasks, namely changes in the goal- (95) [s_f] or start- (91) [s_i] position or the level layout (56). Values that the agent receives are also well suited to accelerate the transition. For that, the received observation (52) [v], the action set (43) [a], the number of objects encoded in the observation (32) [#], or the reward (29) [r] given to the agent can be adjusted.

Based on the task differences, the sixth step, namely mappings between tasks, must be determined. They are not often necessary as 164 do not use any mapping between tasks. Nevertheless, when they are used, the majority of algorithms try to learn them from experience (40) [exp], or they are given manually (25) [sup]. The less-used methods are action mappings (8) [M_a] or groupings of state variables [sv_g]. One interesting approach for mappings in image-based domains is the Generative Adversarial Network (GAN) [97].

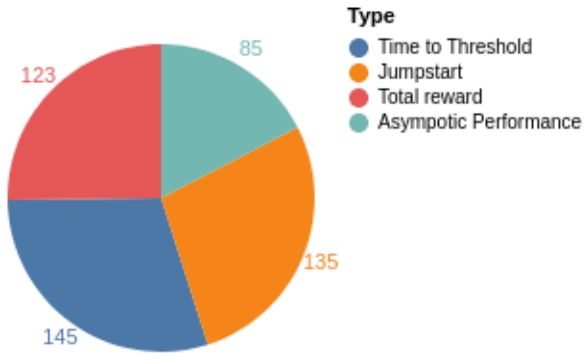


Figure 3.4: Number of times papers reported that transfer performance improvements (Section 3.3) were measured by category.

The last step in transfer learning, and the most important to get an idea of the experimental success, are the associated metrics (Figure 3.4). 146 entries achieved a decrease in the time to reach a threshold (tt), while 135 were able to jump-start (jp) the reward in a new setting. 123 entries achieved a higher total reward (tr) compared to no transfer at the end of training, and 85 entries trained agents that show asymptotic performance (ap) after transfer. Many papers measured multiple metrics of success. 91 achieved two, 38 three, and 27 all four metrics.

We also looked at what kind of problem TRL is mostly applied to. The most often recurring applications are navigation (122), robotics (56), classic control (42), and games (26). The navigation domain is the most diverse. The majority of experiments inspect 2D (110) instead of 3D (12) worlds. To further simplify 2D worlds, 79 entries use a grid instead of continuous navigation. Most entries formulate their own problem, but there are also some recurring standardized environments such as the Taxi world from Dietterrich [70] or the blocks world by Langley [147]. Although most 2D grid-level layouts do not cite anyone, there is one recurring citation: the three-room grid-level by Thrun [268]. One entry also uses a slightly adjusted version of the three-room grid-level [12]. The goal of these papers is to test if transfer is possible. We would expect test domains to be different, challenging, and standarized. The state of affair that we encountered is lacking in this respect. While some problems repeat, there is no unified benchmark, and the few existing benchmarks are not dynamic in the sense that they could adapt their difficulty or similarity (Section 3.5). Given the large number of entries on the topic, we were surprised that there is no real benchmark to assess

the planning capabilities of an RL algorithm in the navigation domain. The ProcGen environment is such a benchmark for maze navigation [59]. However, ProcGen is an environment with procedurally generated levels, and most entries here use one (or more) static levels. And [134] has found that while (deep) RL can learn to generalize to generated levels within the same distribution, it can not handle arbitrary level layouts.

We encountered 27 game-related entries, of which 11 focus on Atari and only 4 on board games. Few complex games are approached with TRL, like Unreal Tournament [119], StarCraft [240], or GVGAI [181]. Simpler games include Sonic 3 [112] or Pinball [295].

The growing use of neural networks comes with a drawback, namely the reproducibility of results. The ICLR Reproducibility Challenge [200] from 2018 underlines the problem, as less than 33% of papers are rated as properly reproducible [199]. The most straightforward way to make a code-based science experiment reproducible is by publishing the source code. Only 15 entries did this, but at least 148 contain pseudocode. Moreover, 31 entries link additional resources. 10 of these are websites, but 8 of them are not available anymore. The two websites that are still reachable summarize different short videos of robotics tasks on one page. The remaining 21 links are videos.

Another important aspect related to reproducibility is the software libraries used in the experiment code. Even when closed source, some entries do share details about used libraries. As machine learning backend, TensorFlow (20) and 7 PyTorch (7) are popular. Only 4 of the TensorFlow uses are from DeepMind, so the library's popularity seems community-driven. Regarding RL environments, 24 use OpenAI's Gym [39], 5 the proprietary physics playground Mujoco [270], 4 the unreal engine based 3D navigation simulator AirSim [238], and 2 the continuous control benchmark RL-Lab [76].

3.4.3 Generalization Capabilities

Achieving generalization by transferring knowledge from one task to another remains challenging. The more different tasks are, the harder generalization becomes. One algorithm, such as AlphaZero, may be able to learn to play at world champion level in three different board games [232]. The limitation is that one network has to be trained again for each game. Unlike humans, the AlphaZero AI can not yet generalize and transfer knowledge from one domain to another similar domain, even when the internal network architecture is identical. The field of TRL revolves around finding

algorithms to enable this transfer between different tasks in the same domain. Transfer between differing domains works better the more similar they are, and when only the transition dynamics change.

For example, transferring a Q-learner from MountainCar to Cartpole works well [7]. Also, transfer from CartPole to Bicycle works well [131] or a three-linked CartPole to the Quadrotor [8] control tasks. Another popular transfer domain is RoboCup, with 24 entries. Many of the experiments focus on increasing the number of players involved from 3v2 to 4v3 [252] or 3v2 to 6v5 [190] in KeepAway. Others explore multi-task experiments such as MoveDownfield to BreakAway [273].

Although the RoboCup challenge could be seen as 2D navigation, it does not involve the same amount of planning as is required to navigate through a 2D maze, whether it is grid-based or continuous. For 2D maze navigation, which is intuitive for most humans, reinforcement learning needs special help. For example, repositioning doors in a level whose layout has not changed requires advisors [206]. Picking up a sequence of keys and then moving through doors can be solved by adding options [144]. For regular search algorithms such as A* these task changes would be easy to solve. However, for large, complicated 2D and 3D navigation domains, one has to incorporate planning into RL. There already exist two great examples of this. First of all, Go-Explore [80]. Many of the Atari games that [80] tackles can be viewed as multi-level 2D navigation tasks, such as Montezuma's Revenge, Berzerk, and Private Eye. Go-Explore effectively combines planning with regular reinforcement learning and is good at these navigation and planning Atari games. Other promising approaches exist such as MuZero [233] or MCTSnet [107].

Achieving reliable transfer of knowledge gained from perfect simulations to the real-world is another unsolved transfer problem. We found 23 entries in this sub-field (Sim2Real). Many of these focus on moving joints (3), which could be compared to classic control tasks. It can also be extended to multiple joints that form a robotic arm (12) to interact with objects. To narrow the gap between simulation and reality, in many papers, noise is applied for regularization or smoothing, e.g., Gaussian Noise [116], uniform random noise [11], the Simulation Optimization Bias (SOB) [178]). There are also efforts to make the noise obsolete [138].

The findings underline that TRL only works well when some similarity between source and target tasks can be found. In this sense, generalization is in the eye of the beholder, and there is a long way to go. Nevertheless, one paper's generalization capabilities are impressive: By encoding the Video and Audio output of an Atari game into a multi-modal latent space, a policy was trained on video-only that can transfer

its performance to audio-only input [245].

3.5 Research Agenda

In this literature review, we have categorized around 300 papers on transfer reinforcement learning. We have seen many different approaches trying to transfer knowledge between many different applications. The measure of success is generalization: how well knowledge can be transferred between different applications. We note (1) in supervised learning, transfer has achieved more success [286] than in reinforcement learning. TRL is still a young field. The first deep learning TRL papers appeared around 2010, but only in 2018 did the field really start adopting it. Deep learning methods can be expected to continue to yield good performance. Transfer reinforcement learning should continue to focus on transfer of network parameters. (2) The large diversity in applications and methods makes progress comparisons difficult. Also, we noted a lack of dynamic benchmarks (Section 3.4.2). (3) Generalization is hard, except when applications are clearly related. These observations bring us to the following research agenda.

1. A clear measure of transfer capabilities is needed in transfer reinforcement learning [9, 50]. This implies a universal measure of similarities between tasks/domains.

2. The combination of planning and learning can be expected to improve (as already shown by Go-Explore [80] and Mu-Zero [233]). TRL should focus on general planning methods [107].

3. Benchmarks are needed that are standardized, challenging, and dynamic (Section 3.4.2). PCG can be leveraged to enable fine-grained control on the different levels of difficulty and task similarity.

Specifically, in Section 3.4.3, we briefly mentioned the inability of TRL to generalize to procedurally generated levels in 2D navigation [134]. Although it can generalize to different levels from one distribution, it can not handle arbitrary levels. One trend that could help overcome this problem, at least for navigation-related tasks, is the fusion of learning and planning, as in model-based reinforcement learning [262]. Nevertheless, it is still an active research field with contributions such as a framework trying to unify the two [169]. LSTMs also play an increasingly important role in improving generalization, as they can already enable the adaptability to different layouts of 2D

navigation levels [77, 253]. Furthermore, we view curriculum learning as a form of planning, as the creation of curricula inherently requires planning, and it has shown success as a TRL method [104, 240].

Unfortunately, using benchmarks to compare novel approaches is not the norm in TRL yet. Contrary to Supervised Learning, where achieved accuracy percentages on well-known data sets can be perfectly compared, each sub-field and application would require different benchmarks to assess specific transfer capabilities of varying algorithms. But as we have shown (Section 3.4), there are many experiments in similar fields like robotics or navigation that could profit from each other. There are already interesting simulators like Mujoco [270] for continuous control, ProcGen [59] for generalization capabilities, Meta-World [299] and Alchemy [283] for meta-TRL. However, there is still a plethora of other applications missing benchmarks to assess, e.g., game-playing AI [281] or 2D navigation. ProcGen does feature 2D Maze levels, but no benchmark verifies whether an algorithm can adapt to different movement styles (grid vs. continuous) or movement types (top-down vs. side-scroller).

Most experiments we encountered only transfer between fixed sets of tasks. As PCG has proven to be a reliable tool to improve generalization performance in RL [20, 102], the adoption of PCG is the logical next step for TRL. One could generate a seemingly infinite amount of different transfer tasks, and in game-like environments this is presumably relatively easy. Furthermore, PCG benchmarks would enable agents to control generation parameters that influence the difficulty, resulting in a dynamic curriculum that improves learning performance [104]. Moreover, the generation parameters could be chosen to influence task similarity. Such a quantifiable similarity control could be used as a benchmark metric to determine how much tasks may differ until the tested algorithm can not transfer efficiently anymore.

An ideal tool for the future would be a database, similar to OpenML [277], that contains machine processible information on all available TRL tasks/experiments. When approaching a new task, the trove of data could be used to cluster similar domains via task similarity metrics [50], to identify promising source tasks to transfer from. This would also allow a leader-board style comparison of how well which algorithm transfers between what tasks, as has been done in the GVGAI [195] per game (set). While new task similarity metrics are still developed [9], the problem identified by [50] persists, that there is no one universal metric to encompass all similarity dimensions. The described database would enable the combination of all existing similarity metrics and the performance of different algorithms in transferring from one task to another to train a supervised network that outputs a singular numerical transfer success prob-

ability.

3.6 Conclusion

By borrowing methods and a dataset from the field of social network analysis (Section 3.3), we have created a unique survey about TRL. We have collected tabular data about transfer-related metrics similar to [265] but on a larger scale. We verified that out of 270 unique TRL entries, $\approx 74.8\%$ have not been included in any of the previous surveys [35, 149, 182, 243, 265, 266, 280, 304]. Because of the large scope, in which not every single entry can be mentioned textually, we created a website [180] that gives a better overview of the dataset with more graphs and the ability to interactively filter through the data. With this data, we have underlined the large diversity of applications for TRL, which shows a focus on navigation, robotics, classic control, and games. We find that transfer, at least in RL, has a hard time generalizing to different problem variations (Section 3.4.3). The transfer that works best is to problems that are similar. In tasks that involve planning, such as routes through a 2D levels, TRL lacks as it can not generalize to arbitrary layouts yet [134]. We do see an increase in methods that try to merge planning with learning (Section 3.5) to overcome this limitation. Another issue is the comparability of algorithms. Most approaches define their own slightly different version of known problems and compare their results to self-defined baselines. The field requires more benchmarks like Alchemy [283], Meta-World [299], or ProcGen [59] to quantify transfer performance properly and compare different algorithms. These benchmarks will increasingly include more PCG to challenge generalization capabilities further. We provide a research agenda outlining how to achieve this goal. In our view, the game AI field is especially well suited to pursue this research as it already employs the necessary algorithms and an abundance of configurable problems.

Chapter 4

Towards verifiable Benchmarks for Reinforcement Learning

Reinforcement Learning (RL) is one of the most dynamic research areas in Game AI and AI as a whole, and a wide variety of games are used as its prominent test problems. However, it is subject to the replicability crisis that currently affects most algorithmic AI research. Benchmarking in Reinforcement Learning could be improved through verifiable results. There are numerous benchmark environments whose scores are used to compare different algorithms, such as Atari. Nevertheless, reviewers must trust that figures represent truthful values, as it is difficult to reproduce an exact training curve. We propose improving this situation by providing access to the original evaluation data to validate study resultsr[a]. To that end, we rely on the concept of *minimal* traces. These allow re-simulation of action sequences in deterministic RL environments and, in turn, enable reviewers to verify, re-use, and manually inspect evaluation results without needing large compute clusters. It also permits validation of presented reward graphs, an inspection of individual episodes, and re-use of result data (baselines) for proper comparison in follow-up papers. We offer plug-and-play code that works with Gym so that our measures fit well in the existing RL and reproducibility eco-system. Our approach is freely available, easy to use, and adds minimal overhead, as *minimal* traces allow a data compression ratio of up to $\approx 10^4 : 1$ (94 GB to 8 MB for Atari Pong) compared to a regular MDP trace used in offline RL datasets. The paper presents proof-of-concept results for a variety of games.

[a]This chapter is based on the publication [173] Müller-Brockhausen, M., Plaat, A., Preuss, M.: Towards verifiable benchmarks for reinforcement learning. In: IEEE Conference on Games, CoG 2022, Beijing, China, August 21-24, 2022. pp. 159–166. IEEE (2022), `https://doi.org/10.1109/CoG51982.2022.9893715`

4.1 Introduction

Reproducibility is a key component of peer-reviewed science. Reviewers are supposed to read, understand, and ideally be able to reproduce an experiment to ensure its factual correctness. It touches not only computer science, but any science, as without easy reproducibility, fraud is difficult to detect [185]. Especially for benchmarks and competitions, where fraudulent submissions potentially poison the rankings of a leaderboard, it is important to have tools for validation.

Benchmarking AI algorithms has become increasingly important and is now a driving force behind algorithm development. In Game AI, competitions have been an important part of scientific conferences for a long time already, and especially game problem benchmarks are currently more and more spreading out to core AI conferences, e.g., with the MineRL competition at NeurIPS [109, 110]. Whereas the overall aims of algorithm development are often to improve generality and especially sample efficiency, the employed methods are still relatively slow and thus need very long runs, thereby making reproducibility difficult.

In theory, computers are excellent for reproducibility. One can run the same code, bit for bit, on many different machines. This may be simplified down to issuing a single command, based on technologies such as Docker [34]. However, methods that include some sort of non-determinism (e.g., training evolutionary algorithms [155], or deep neural networks [154]) hamper the reproducibility of experiments. Another growing problem is the availability of computing resources that would be needed to replicate results [54]. The tremendous successes of AlphaStar [279] and Dota 2 [28] are prominent examples. The large computing clusters they relied on are unavailable to most researchers for running any type of replication experiment. Furthermore, it becomes increasingly important to also consider sustainability issues, as the big cluster experiments are energy inefficient. Such considerations have been voiced, e.g., for Natural Language Processing (NLP) [260] or complex Games as Go (AlphaZero) [235]. It would thus make more sense to avoid re-computing everything but to improve the inspection of existing log data. Other issues that stand in the way of exact replication include insufficient reporting [114] or not open-sourcing code [201].

If replication itself is unavailable for some experiments, the next best thing could be verifiability, namely the ability to inspect, check, and replay parts of the evaluation. However, this is also difficult even in terms of handling the huge amounts of data that are produced during the big experiments. In order to achieve this, we would need some way of highly compressing this data, which instantly points us to the concept of

minimal traces.

The research question we are going to tackle in this work is thus: *How may a researcher verify the evaluation of aRL algorithm of other researchers, especially the display of results in figures and tables, based on minimal traces?*

We offer the following contributions:

- We explain how minimal traces (Section 4.2) allow reproducible verification of results such as benchmark leaderboards (Section 4.3.1). Moreover, we empirically show that they enable a compression ratio of up to $10^4 : 1$ for offline RL datasets (Section 4.4.1)
- We provide Plug-n-Play code [17] to collect minimal traces that integrate with the RL-Ecosystem (Gym [39])
- We provide an agenda for further research on how to obtain verifiable RL evaluation results using minimal traces (Section 4.5)

Although there are many factors at play with reproducibility, our work solely focuses on methodological improvements for RL research. After briefly introducing the concept of minimal traces (section 4.2), we first look at suggestions that have already been made for improving reproducibility or experimental methodology in section 4.3. Based on that state, we find improvable points (section 4.3.1) and suggest concrete, actionable steps (section 4.5). We then outline how these steps can be applied in practice with the code that we provide (section 4.5.1). To enable compatibility, we ensured that it properly interfaces to the existing RL-Ecosystem.

4.2 Background: Minimal Traces

For the following sections, the concept of minimal traces is important, thus we review its origins and known uses here.

RL optimizes sequential decision making processes, that are modeled as so called Markov Decision Process (MDP). An MDP consists of a tuple (S, A, $P_a(s, s')$, $R_a(s, s')$) (state space, action space, the probability to go from state s to s', and reward for going from s to s') [27].

A *trace*, also commonly referred to as an Episode within an RL-Environment [204], is a list of tuples that contain the start state s_t, the chosen action a, the received reward r, and the resulting state s_{t+1}. Traces are sufficient to train an RL Algorithm offline / off-policy, and they are also shared by related work as dataset-basis for training [2].

Staying true to the computer science RL terminology drawing inspiration from psychology [263], we found a fitting concept to our problem, namely *minimal traces*. Their goal seems related: "Predicting the Past from Minimal Traces" [288]. We want reviewers to reliably predict (verify) the past (evaluation results) using minimal traces. To reduce the used space (minimal) of traces we assume that an MDP, given the same initial state s_0 and action sequence α, will yield the exact same trace. To reliably re-reproduce the initial state s_0 and follow up states, the random outcomes in $P_a(s, s')$, as well as parameters influencing the behavior of the environment need to be fixed based on an initial configuration s_{init}. Hence s_{init} contains the random seed as well as environment hyper parameters (see Table 4.2 for an example of cartpole hyper parameters). Fixing these probability outcomes is commonly referred to as a deterministic MDP [207]. Deterministic MDPs reduce the required data for re-simulation of minimal traces to s_{init} and α. Minimal traces fit well into RL problems as there the action set A is usually smaller than the state space S. Hence it makes more sense to only save actions, if the observations can be re-constructed afterward. While the added re-simulation cost might seem impractical for verification purposes, our experiments show that it can take less than 0.7% of the original RL training time (Section 4.4.1).

4.3 Related Work

While the matters of reproducibility, replicability, and verifiability are relevant to all scientific fields [185], we will focus on RL here. In RL, previous works suggest guidelines on how to design and report a well reproducible experiment [114]. Conferences such as NeurIPS are moving towards implementing these guidelines and ask reviewers to fill in a questionnaire about reproducibility. This has led to more and more sharing of code, and researchers are encouraged to do so [201]. Moreover, reviewers found it easier to judge submissions that included code. Most of the reviewer guidelines focus on the paper itself, which is the well-established scientific tradition that was practiced already when computers were not yet invented. However, many researchers in Computer Science now believe that for experimental works, we should go one step further and exploit its theoretically perfect and exact ability to verify factual correctness of reported results and submitted code / data (Section 4.3.1). This so-called "Verification of Artifacts" [155] is not a new concept. For example, tools available to make policy training as reproducible as possible are readily available (e.g. Garage [61]). For experiments that are not using tools as Garage, there are also clear guidelines on how

to properly compare to a baseline of an algorithm [124]. Moreover, researchers have suggested to save the final values used in graphs to be able to verify the figures [141]. This theoretically works for any Figure. However, how can we be sure that the Figure is correct [84]?

Games are an interesting playground for RL-AI. GVGAI is a prominent example [197], as it is used for competitions that benchmark individual algorithm submissions from both planning [93] and learning such as RL [272]. For these competitions, the validity of results is guaranteed by means of execution of the submitted code by the event host. This is made possible through a pre-defined agent interface, that allows to interact with arbitrary games. In other RL environments, we also have a pre-defined agent interface (see the center of Figure 4.1), but re-running policy training is not at all reproducible. Reproducibility is not guaranteed in GVGAI either, as the applied algorithms, such as MCTS [44], include randomness that is not fixed. While GVGAI remedies this by means of multiple runs, research has shown that averaging over runs does not prevent inconsistencies in reproduction attempts [114]. Whereas minimal traces do not alleviate the problem directly, they do lift the requirement to have to run the agent code oneself.

4.3.1 Why minimal traces?

Reproducibility in the RL ecosystem is an evolving matter. Environments usually behave deterministically if seeded, as, e.g., Cartpole or MountainCar in Gym [39]. Famous problems that did not yet satisfy this requirement have been converted (e.g. Robotics [16, 90]). Moreover, besides theory focused guidelines [114], practical tools like Garage exist to enable reproducible policy training [61].

Nonetheless, RL lacks verifiability of evaluation results, such as benchmark submissions. Reviewers shall be enabled to easily verify reported results in figures or tables with minimal effort. MDP's already come with the concept of *traces* that are basically a full log of all data (observation, action, reward) [204], from which figure data could also be constructed (see Section 4.2). These traces are used for offline RL. However, offline RL datasets can become quite large (3TB for the experiments in one paper [2]) and, as a consequence, are hosted on a proprietary central authority. To remedy this problem, we collect as little data as possible and without requiring a central hosting authority regardless of size (Section 4.4.3). In Figure 4.1, we attempt to visualize the relationship between offline RL datasets, minimal traces, and the data that Garage [61] collects. Thus, we suggest to collect an initial environment config-

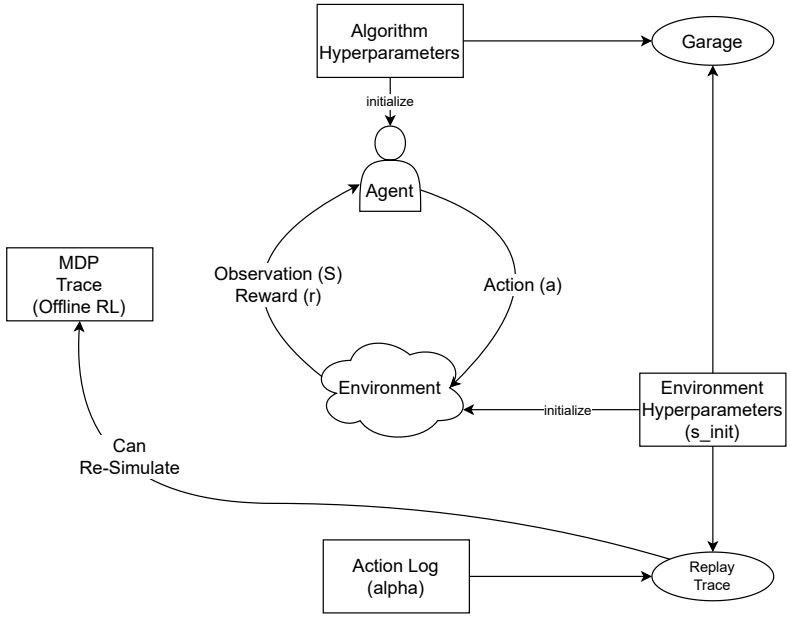


Figure 4.1: A visualization of the relationship between RL training, offline RL datasets (traces), minimal traces Garage [61], as well as the data collected for each.

uration corresponding to s_{init} from the Minimal Trace (Section 4.2). In Gym, s_{init} contains all values that influence an environments initialization (reset) and transition function (step). Table 4.2 contains an example of s_{init} for CartPole. Moreover, the actions that an agent takes are saved. A consequence is that our method is limited to fully deterministic environments. By properly seeding non-deterministic algorithms' random number generation, non-deterministic problems can also be used.

We later show that minimal traces allow a compression ratio of up to 77 for regular and up to 12559 for image-based environments (Section 4.4.1).

Our focus on the environment instead of the policy training is well motivated. The data collected by tools such as Garage [61] can not overcome one specific problem. The training of neural networks includes certain operations that render it not reproducible if the host machine, host operating system, or software library version change [213]. Unless researchers have the exact same machine and software configuration, reproducibility will become difficult. Software configurations are easily reproducible via Docker [34]. However, if the same code produces different results on other machines, then reproducibility becomes practically impossible.

We see minimal traces as an add-on rather than a replacement to current repro-

ducibility approaches. It is a workaround because reproducible policy training, such as Garage [61] attempts to offer, does not yet work. Should training become fully deterministic, minimal traces might become obsolete. Nevertheless, they would still offer the advantage of being computationally cheaper than training, as environment execution requires less computational resources than updating weights of a large neural network. Moreover, environment re-simulation is cheap enough to potentially run in the web browser enabling interactive tools, which is difficult to reproduce with computing clusters needed for training.

Whereas minimal traces are limited to RL, their concept transfers perfectly to video games. TrackMania is a great example because its physics is fully deterministic. For its leaderboard, replays are saved. A replay contains the name of a level and all taken actions by the player. Leaderboard submissions are verified for validity [72]. Moreover, it allows to detect Tool-Assisted Runs in pure human play leaderboards [73]. More complex games, such as Counter Strike: Global Offensive, Dota 2, and Starcraft, support replays as well. They are also minimal in their sense. However, the multitude of possible inputs is difficult to map directly to a RL action space and hence minimal traces. The same applies to the Unreal Engine, which has a general ReplaySystem that can replay any data [96] but does not automatically allow replaying arbitrary scenes deterministically based purely on agent actions. Nevertheless, this larger trove of data is still useful to detect cheats, such as AimBots [157]. Furthermore, it can be used to make estimations of player skill [291].

As games are used in competitions, it could also be applied for validation there. For example, two of the games we previously mentioned to support replays have been a competition at the IEEE Conference on Games 2021: Dota 2 [92] and Starcraft [89]. Moreover, SpaceInvaders, which we test from Gym, is also a CoG competition [41]. Other games seem suitable as well, such as GvGAI [93], Snakes [42], or Bot Bowl [133]. These games might even manage to be directly compatible with minimal traces, as, for example, GvGAI has managed to provide an RL-Gym Environment for its learning track [272].

Lastly, our data collection suggestion harmonizes with the concept of PCG, as the seed can be saved for deterministic reproduction. ProcGen has already shown that current RL-Algorithms are struggling to generalize [59]. However, PCG applied correctly already enables better generalization [220] and more fine-grained training curricula [104], and is thus especially important in benchmarking TRL [176].

4.4 Method

We will describe our approach in detail, along the lines of 3 different aspects. Minimal traces achieve a high compression ratio compared to regular traces. They can compress up to $10^4 : 1$ (Section 4.4.1). Next, we detail how minimal traces enable re-usable visualizations (Section 4.4.2). Moreover, we suggest the usage of a distributed file system, the interplanetary file system (IPFS) [177], for long-term storage (Section 4.4.3). Based on these insights, we propose a reproducibility agenda (Section 4.5).

Environment	Trace (MB)	Minimal Trace (MB)	Compression Ratio	Training (sec)	Re-simulation (sec)	% Re-Simulation to Training
Assault-ram-v0	767.21	7.76	98.82	2314.0	177.0	7.65
Assault-v0	96165.33	7.78	12355.67	5372.0	393.0	7.32
BipedalWalker-v3	530.72	181.79	**2.90**	2241.0	15.0	0.67
Breakout-ram-v0	757.55	8.33	90.94	2859.0	640.0	**22.39**
Breakout-v0	96504.98	7.98	12100.5	5647.0	520.0	9.21
CartPole-v0	452.25	8.5	53.23	1907.0	13.0	**0.68**
Pong-ram-v0	767.18	7.7	**99.58**	2356.0	142.0	**6.03**
Pong-v0	94641.99	7.54	**12559.36**	5319.0	321.0	6.03
SpaceInvaders-ram-v0	767.45	7.76	98.91	2412.0	181.0	7.5
SpaceInvaders-v0	95973.03	7.75	12378.43	5448.0	378.0	6.94
Taxi-v3	335.85	8.46	39.69	2053.0	68.0	3.31

Table 4.1: A comparison of the space requirements in megabytes and re-simulation cost in seconds for (minimal) traces when training PPO for 1 Million steps and averaged over three runs. We see that minimal traces achieve high compression ratios for single action environments such as Atari or CartPole. Even in the worst case (BipedalWalker-v3), minimal traces still allow a compression ratio of 2.9, saving nearly 350 MB.

4.4.1 Data compression

Minimal traces are a way to store evaluation result signatures efficiently. We will have a closer look at efficient storage for different games. Please refer to Table 4.1 for the size comparisons.

RL traces can take up a considerable amount of space. For example, a single work offering an offline RL trace provides 3TB of data [2]. In order to compare the amount of space needed for traces vs. minimal traces (Section 4.2), we chose different environments with growing observation spaces. Taxi with one integer, CartPole with two floats, BipedalWalker with 24 floats, Atari-Games RAM using 128 bytes, and the produced 210 x 160 pixel RGB image using 100800 bytes. More interestingly, whereas most environments have only a single integer action space A, BipedalWalker has three continuous actions increasing the space required for minimal traces. We train on the environment for 1 Million steps using PPO [234] from stable baselines 3 [215], using the default hyperparameters for both environment and agent. During training, we collect the full MDP-trace (Observation, Action, and Reward) and the minimal MDP-trace (Env-Params and Actions) for each environment. The results in Table 4.1 are striking: Minimal traces enable a compression ratio of 12559.36 for image-based environments with a single action, such as Atari Pong. For the 128-byte ram observation, the compression ratio falls to 99.58 for Pong, reducing 767.18 MB to 7.7 MB.

Nevertheless, environments with small observation spaces such as CartPole still allow a compression rate of 53, reducing a 452.25 MB trace down to 8.5 MB. However, BipedalWalker underlines that the potentially saved space depends solely on the size difference between the observation space S and the action space A. For BipedalWalker, 24 numbers in the observation space vs. 4 in the action space. Hence the compression ratio of 2.9 reduces the 530.72 MB trace down to 181.79 MB. An intriguing discovery we made is a varying size for the re-simulated trace of BipedalWalker, which should not occur. Thus we performed an experimental analysis and found that 5 in 100 re-simulations yielded a different trace due to rounding errors. Consequently, BipedalWalker is not yet fully deterministic. We repeated this experiment for all other tested environments in Table 1 and found that they are fully deterministic, hence yielding 0 failed re-simulations.

We also measured the time to re-simulate a full MDP-trace from a minimal MDP-trace vs. the training time. Our measurements are also shown in Table 4.2. Note that for timing-related data, there is variation in runs for different hardware. All

experiments were run on a machine with an Intel Xeon Silver 4214, an Nvidia Geforce RTX 3090, and 256GB of RAM. The cost of re-simulation depends on the complexity and observation space of the environment. We make use of multi-threading to re-simulating multiple episodes at once. For Atari, re-simulation varies per observation and game. In the worst case, it takes 22.39% of the training time for Breakout-ram-v0, and in the best case 6.03% for Pong-v0. Computationally less intensive environments, such as BipedalWalker-v3 or CartPole can lower this further to less than 1 % (0.68%) of training time.

4.4.2 Re-Usable Visualizations

Listing 4.1: An excerpt for a re-usable Vega Reward Graph description. Generates the graph shown in Figure 4.2. The full JSON-File that can be explored in the Vega-Editor can be found in the Code-Repository as listing_graph.json.

```
{"$schema": "...",
        "description": "Reward␣per␣Episode",
        "data": {"values": [{"Episode": 0,
                             "Reward": 11.0, "env": "..."}, ...]},
        "mark": "line",
        "encoding": {
                "x": {"field": "Episode",
                      "type": "quantitative"},
                "y": {"field": "Reward",
                      "type": "quantitative"},
                "color": {"field": "env",
                      "type": "nominal"}}
}$
```

[141] suggests saving the values that are used to plot figures and providing the code to the plot. This improves reproducibility when the numbers are the results of the experiments. Minimal traces enable re-simulating this data to verify if this is the case. Moreover, minimal traces allow looking at different values than presented in a paper. For example, if a paper reported only the average reward, one could extract the median reward instead through the re-simulated data. Alternatively, if reward per episode was reported, one could instead look at reward per step. To increase the re-usability and accessibility of figures, we suggest using Vega [230]. Vega is a JSON-based graph

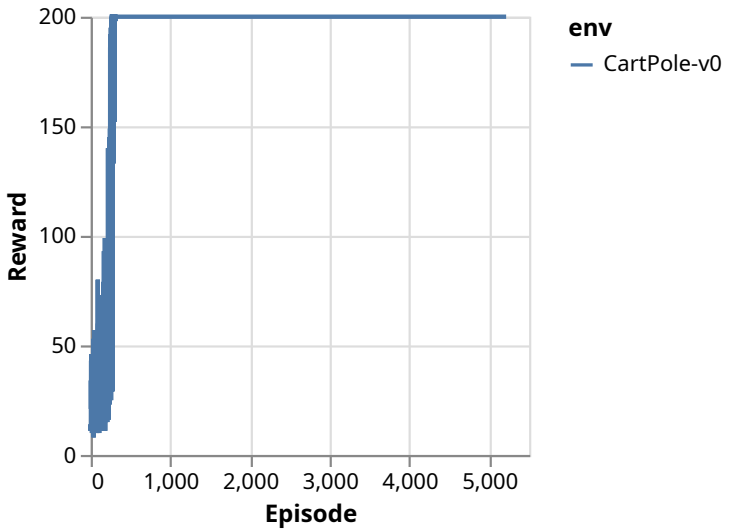


Figure 4.2: A re-usable Vega Lite Reward Graph, generated from the Description shown in Listing 4.1. It displays the sum of reward per episode achieved during training of PPO3 [234] for 1 million steps using the default hyperparameters for algorithm and environment. The used reward data can be re-simulated based on the minimal trace with ID "2IEpetGGwNFOs38rhAFHx".

description language. The main advantage is that plotted values are embedded inside the human-readable JSON data. Hence, one could extract a baseline algorithm reward line from the Vega description of an original paper and then compare it to a newly trained variation without re-simulation. Of course, it still allows exporting a scalable graphic for usage inside of the paper (e.g., Figure 4.2). In this case, guidelines on designing a proper baseline [124] are not that important anymore because the actual data from other papers can be directly compared[1].

Another advantage of Vega [230] is the ability to load Graph-Definition-Files in a Browser. These come with various advantages, such as seeing the exact value of an individual coordinate, zooming, scrolling through the axes, and changing colors if these are not color-blind friendly.

[1] Assuming they are being compared on the same benchmark environment with the same configuration

4.4.3 Data availability

The data we suggest to collect (Section 4.3.1) potentially requires much storage. Whereas some hosting authorities allow researchers to upload large datasets for long-term availability, the two main problems with a central authority are the possibility that data could be tampered with / changed and that there is only a single point of failure regarding availability. The Interplanetary File System [177] lets us address these issues. It behaves similar to BitTorrent. Users who downloaded a file also share it, eliminating the need for a central server to store all files. Moreover, the data is hashed, so it can not be altered afterward by the original author, the hosting authority, or a redistributing user. A hash can represent a full folder with many subfiles that also all have individual hashes / IDs. So the log file itself and the results produced with it become verifiable. IPFS [177] also advertises its usefulness for scientific purposes, and we believe our minimal traces are well suited for it. Moreover, IPFS can co-exist and even integrate into a hosting service (such as Zenodo [87]) that could mirror the data it provides via IPFS. That would allow them to use it as a Content Delivery Network (CDN) for the actual files listed in the Zenodo database. Finally, conferences or journals could have public lists of relevant dataset IDs for published papers that should be *pinned*. Pinning in IPFS can be seen as the equivalent of mirroring data. A pinned ID will permanently (until unpinned) be held in local storage and thus be available to others requesting it.

4.5 Reproducibility Agenda

Minimal traces are a useful step on the path to reproducibility, allowing efficient verification of evaluation data. To further improve the reproducibility of the field and to put our work in perspective, we suggest the following agenda.

We propose researchers experimenting on deterministic Environments (ideally using PCG in order to improve generalizability) to do the following:

1. Use the verifiability tool to collect minimal traces
2. Provide source code, including a runnable container (e.g. Docker), to allow verification of results and figures
3. Generate figures using a common visualization grammar such as Vega [230], facilitating re-use of figure data
4. Utilize IPFS [177] to ensure the data's integrity and long-term availability

Name	Default	Description
Gravity	9.8	Gravitational Power of the Planet
Pole Mass	0.1	The mass of the pole balancing on the cart
Pole Length	0.5	The length of the pole (The actual length is this value times two)
Cart Mass	1	The mass of the cart
Force Magnitude	10	Strength of the force applied to the cart on input
Tau	0.02	How much time passes between state updates (in seconds)
Theta Threshold Radians	$\sim$0.209 ($\sim$12°)	The angle at which the pole is considered to be tipped over
X Threshold	2.4	The area in which the cart may move without being considered out of bounds
Use euler kinematics	TRUE	Influences cart movement
Random Seed	varies	Influences initial cart position and pole angle

Table 4.2: Complete list of initial Environment Parameters s_{init} (Section 4.2) for Cartpole. Given the same s_{init} and action sequence α, a deterministic MDP like CartPole will always yield the same trace.

4.5.1 Agenda applied to CartPole

To better illustrate our suggested agenda in practice, we will illustrate how following it looks for the proverbial CartPole environment. In Table 4.2, we have prepared the environment hyperparameters (s_{init}) relevant for each CartPole episode. Garage [61] (Figure 4.1) also collects these parameters but assumes policy training to be fully reproducible, which it is currently not (Section 4.3.1). In Listing 4.2, we show that recording minimal traces merely requires wrapping the Gym-environment. The code in this listing is not pseudo-code but the actual main file of our runnable example. The last function generates a Vega [230] (Section 4.4.2) JSON-Description (Listing 4.1) that can be used to visualize a graph (Figure 4.2). It can also be pasted into the Vega-Editor[2]. Table 4.1 is also based on re-simulated data from minimal traces.

New projects wanting to record minimal traces need to wrap their Gym-environment with the *record* function from the *vgym*-folder [17] before training. Then, the minimal traces will be saved into a sub-folder of the current working directory. A minimal

[2]https://vega.github.io/editor/

trace file is serialized into the Concise Binary Object Format (CBOR) [37] and compressed with zlib [69]. Note that minimal trace sizes in Table 4.1 reflects the size before serialization and compression. The utility function *load_replay* in our package abstracts these serialization details away from the user. After loading a file, either all episodes can be re-simulated into regular traces in parallel using *resimulate_parallel*, or an individual episode can be re-simulated using *episode_to_trace*. Based on that, re-simulated data figures and tables should be generated, and the code as well as recorded minimal traces published alongside the submission.

Listing 4.2: Example code that trains a policy on Cartpole collecting a minimal trace. The full trace is re-simulated based on the minimal trace and used to generate a Vega Graph-Description.

```
import gym
from vgym import record, resimulate
from stable_baselines3 import PPO
from example import make_graphs_from
env = gym.make("CartPole-v0")
renv = record(env, name="CartPole-v0")
renv.reset()
PPO("MlpPolicy", renv, verbose=1)
.learn(1000000)
# Recreate Trace from minimal Log
trace = resimulate(renv,
renv.minimal_traces.to_list())
# Plot using re-simulated data
make_graphs_from([("CartPole-v0", trace)])
```

Our code repository contains a Dockerfile, recorded / used minimal traces, the example code behind the imported functions of Listing 4.2, and a readme detailing how to re-simulate the shown graph (Figure 4.2), as well as the values for the size comparison table (Table 4.1). All data is hosted on IPFS [177] here [17].

4.6 Discussion

If properly applied, the concept of minimal traces allows for reproducible and verifiable RL evaluations. Moreover, it enables re-usability of result data, more accessible ways to view the data interactively, inspection of individual episodes and data-saving for

offline RL datasets.

A limitation of minimal traces is that the source of the action sequence can not properly be verified. From the obtained data alone one cannot exclude any sophisticated ways of cheating the authors may have applied. Whereas a result figure or table may be fully reproduced with our data, one can not know whether the agent created the supplied action sequences. The data could be handcrafted or made by a heuristic or any other method that is not listed in the reviewed paper. Although minimal traces do not yet achieve end-to-end verifiable research experiments, they are an important step towards verifiable evaluation figures to show that values portrayed in figures or tables have truly been achieved within the tested environment and have not been randomly generated. In combination with host executed competition benchmarks such as GVGAI, they enable post hoc analysis of individual agent performances.

4.6.1 Environmental Impact

In times where big research experiments use large amounts of electricity [193], it is important to note the possible environmental impact of our suggestion. Hard-disk space seems cheaper than the high wattages of training an agent on a GPU. Hard-disk space instead of computation is also used to "greenwash" novel blockchains as Chia [60]. Nevertheless, nothing electronic comes free, so the hardware still needs to be built, and energy needs to be used to power the servers pertaining the datasets we suggest to collect. In consequence, it will increase the global environmental impact of RL. However, we see no other, more minimal, and less intrusive way to ensure full verifiability of RL research. Thanks to the suggestion to share the data via IPFS [177], the servers would not need to stay online 24 / 7. There could be specific times of data availability where the server is switched on and the data accessible while ensuring the data is not tampered with. Moreover, the distributed nature of IPFS might make the necessity of a central storage server obsolete, given enough participants pertaining parts of the datasets.

4.7 Conclusion

In RL, reproducibility of experimental results, and verification of research claims, is an important challenge. Our work introduces a methodology to verify evaluation results building on the concept of minimal traces. We provide a full implementation of this method and have tested it on small and larger RL experiments.

For typical experiments, minimal traces enable compression ratios of up to 12539, reducing an offline RL trace of Atari Pong from 94 GB down to 8 MB. Moreover, re-simulating this minimal trace back to its original size takes 6% of the original training time.

As our example shows, the collection of minimal traces requires but a wrapper around a Gym-environment.

While minimal traces are limited to deterministic RL problems, the idea transfers well to (video) games. Trackmania already applies leaderboard verification via replays, showing that benchmarks and competitions could adopt similar concepts. We envision a web-based tool that allows re-simulation and verification of results without any software setup by a reviewer on their local machine for future work. To that end, we provide a mock-up (Figure 4.3) with a functionality description. This could be implemented through either a Rust-Gym port of the current approach or by having a Python interpreter that properly works in a web assembly environment and with Gym.

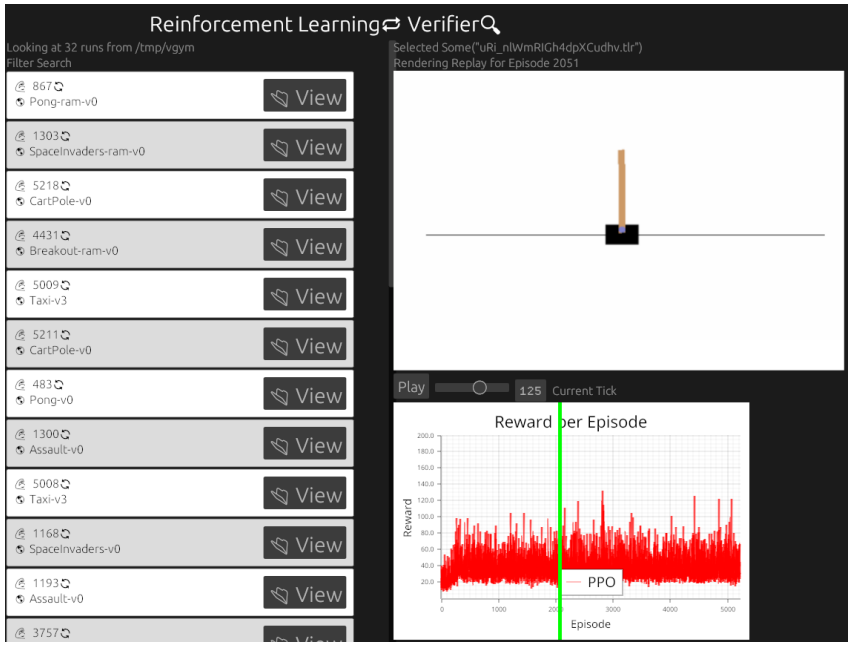


Figure 4.3: Mock-up of a web-based tool we envision for future work. The left side contains a list of minimal traces, the environment, and amount of contained episodes. The right side shows details of the re-simulated trace. For inspection, one can select individual episode replays directly from the training reward graph, highlighted by the green vertical stripe. Each episode can be stepped through frame by frame.

Chapter 5

Scalable Procedural Content Generation via Transfer Reinforcement Learning

Procedural Content Generation algorithms that make use of Machine Learning have garnered significant attention from the general public due to their ability to generate text, images, or video content that is often indistinguishable from human-crafted artworks. These achievements typically necessitate a substantial quantity of data, which may be scarce in specialized domains such as game-level design. One machine learning technique, Reinforcement Learning (RL), can be employed to learn from trial and error to address this scarcity. However, the RL training process requires a substantial time investment and may prove unsuccessful in the case of sparse reward structures. This paper[a] empirically demonstrates the efficacy of curriculum learning as a viable solution to address scalability issues inherent to learning more complex tasks. Instead of trying to learn the whole space from scratch, we employ transfer learning on a curriculum from small to larger levels. We empirically validated this in a 3D vector-based environment, where the objective is to generate free-form tracks that facilitate a rider to move between designated points in the same vain as the flash game hit "Linerider".

[a]This chapter is based on the publication [172] Müller-Brockhausen, M., Khalifa, A., Preuss, M.: Scalable procedural content generation via transfer reinforcement learning. In: Data Science and Artificial Intelligence (DSAI). Springer (2024), https://doi.org/10.1007/978-981-97-9793-6

5.1 Introduction

PCG offers numerous algorithmic approaches to generate various content for games. Some involve the manual creation of intricate rulesets to generate diverse realistic Minecraft villages [225]. Other methods employ generative adversarial models [282] or language transformers [261] to create 2D Mario levels. Search-based and evolutionary algorithms are applicable to a variety of PCG-related tasks as well [271].

Most PCG work is focused on discrete 2D domains with a limited amount of actions. In this work, we explore the game "Linerider" [166]. It enables players to create new tracks by drawing free-form lines through mouse input. We expanded on that problem by extending the game to 3D space. This allowed for a complex continuous space that is different from generating images or text because training data is scarce, which makes it unfeasible to use supervised technique methods. Also, generating levels for games introduces the requirement that the generated content needs to be functional (the level has to be playable).

This data scarcity and the functional aspect of game content pushed researchers and developers [140] towards using the Machine Learning approach of RL. Although RL has achieved superhuman performance in game playing [28, 247, 279], it still has trouble with domains with sparse rewards. Due to that, successful training either requires a lot of training time or requires a small environment. In this work, we showcased that transfer learning for reinforcement learning helps in training agents on bigger vector-based domains in the same vein as Zakria et al. [300] work. Based on this, we think this paper contributes to the following under-explored areas in the combination of PCG and RL:

- Free-form content generation through vector-based action spaces
- Applying learning curricula to tackle RL's scalability issues

5.2 Related Work

Reinforcement Learning and PCG are a fruitful combination as the PCGRL [140] framework highlights. For example, it successfully balances competitive grid-based strategy game-levels [223]. Moreover, it enables the transfer of experience to generate Mario levels that optimize fun and historical deviation [242]. Furthermore, applied to Sokoban, it can generate more diverse and high-quality levels compared to other

approaches, such as GANs and VAEs [301]. It is also not limited to 2D worlds, as it can generate 3D Minecraft structures [129].

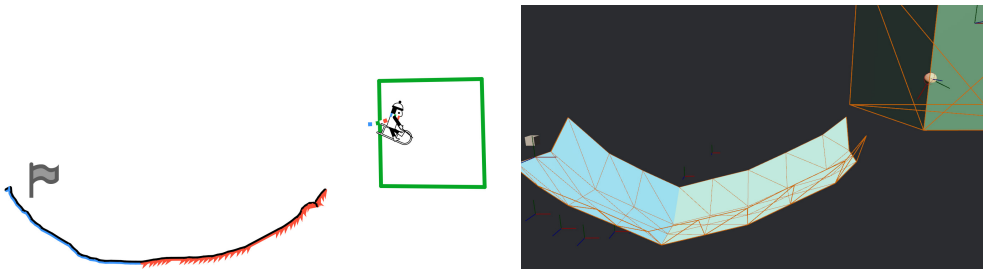
However, the key difference between our approach and PCGRL [140] is that PCGRL operates on an iterative basis, revising work generated by another algorithm and making incremental decisions regarding whether to adjust the current piece under consideration. This stepwise action space makes the agent learn to do repairs. For example, Gupta et al. [108] trained an RL agent to correct programming source code mistakes. However, we aim to harness RL for generating tracks from scratch, which aligns more with Campbell and Verbrugge's work [47], which generates new Rollercoaster Tycoon 2 (RCT2) tracks from scratch. A key distinction between Linerider and RCT2 is the requirement for rollercoaster tracks to loop back towards the starting point. This has been seemingly impossible to solve using exclusively RL, as Campbell and Verbrugge [47] resort to custom heuristics and the A* search algorithm to finish tracks initiated well by RL. In contrast, our tasks involve designing tracks connecting A and B but never looping back to point A.

Transfer Learning is an interesting avenue for PCG as it allows knowledge to be extracted from an existing model that then is effectively transferred to a similar game. Alternatively, multiple models can be combined into a single model, fostering collaboration and knowledge-sharing [228]. This knowledge could, for example, already be embedded in a LLM, which was not at all trained to generate content for a game. Nevertheless, with the right prompt, they can be led to generate Sokoban [269] levels. Alternatively, novel Mario levels can be generated through fine-tuning, which is also a sort of transfer [261].

In the realm of reinforcement learning, the integration of transfer learning with PCG remains an under-explored area. But it holds great promise as reusing existing models becomes even more important for sparse reward settings such as ours. Designing procedural curricula can facilitate the transition from simpler tasks to more complex ones, ultimately enhancing the agent's ability to generalize its learned skills [244]. Under Taylor and Stone's taxonomy [265], the varying reward structures across curriculum steps could be seen as a form of transfer learning.

5.3 Linerider

Originally Linerider (shown in Figure 5.1a) is a two-dimensional flash game released in 2006 [156]. It offers a basic yet engaging sandbox that can be applied in education to teach students fundamental physics concepts [6]. Linerider stands apart from



(a) Screenshot from the original Linerider game. Blue is a normal track and red is an accelerating track.

(b) Screenshot from our 3D version of Linerider. Dark blue is a normal track and light blue is an accelerating track.

Figure 5.1: A comparison of the original Linerider and our implementation. In both versions, a simple ramp that propels the rider into the goal is built. Note that in the original Linerider (5.1a) the green goal is purely decorative and has been added for a better visual comparison.

traditional gaming experiences as it lacks a predetermined objective. Instead, players create custom tracks by drawing lines using a mouse. Moreover, players can make the line accelerate the rider. At any point during the creation process, users can *simulate* their designs. This simulation drops a rider (a person riding a sleigh) at the starting point. From there on out, physics takes over, and the user can watch the rider roll down the created track. The second iteration, "Line Rider 2: Unbound" [166], released in 2008, introduces predefined levels with start and target points, requiring users to fill empty sections with appropriate track components to guide the rider successfully to the goal. Linerider still fosters an active community [156] that uses the game as a creative outlet to express themselves artistically. This expression can be in the form of audio-visualizers that match a specific song or apply orthographic projection to create seemingly 3D images that the rider drives through.

5.4 3D Linerider Framework

Our re-implementation of Linerider (illustrated in Figure 5.1b) emphasizes the development of solvable tasks that can be learned rapidly within the realm of reinforcement learning. To add to the original challenge, we move to 3D Space. Another change we introduced is to use a ball instead of a person riding a sleigh. The primary reason is that in early prototype testing, a rectangular shape gets stuck too easily in curves. Besides the influence on the rider's behavior, this change will allow for an alternative mode where the rider is controlled by a player/agent. Then, the game would be more

similar to Super Monkey Ball [83].

We developed our platform using a natively compilable language rather than Python. Research suggests that the primary obstacle in RL training stems from insufficient sample collection during environmental interaction [10]. Consequently, minimizing latency between agent actions and environmental feedback is essential. This aligns with other physics-based RL environments, such as Mujoco [270], where the physics calculations are done in C(++). By leveraging the Rust programming language [160] alongside the Bevy game engine [29] and Rapier physics simulator [71], we harness a powerful yet lightweight technology stack optimized for scientific exploration. This stack has a proven track record for scientific research in both PCG [31] and offloading physics computations to servers [146].

Our track design connects XYZ coordinates with flat track pieces with edge railings, enabling the rider to navigate curves without exiting the track. To mitigate the likelihood of the rider leaving the track, we have adjusted the restitution coefficient (which is used for allowing to simulate elastic behavior) to 0.5. This modification ensures that the ball bounces less vigorously upon impact, reducing the chances of unintended jumps above the edge railings.

5.4.1 Formalizing the Environment

We formulate our problem as MDP [26]. Software-side we use Gym [39] to allow RL-Agents to train in our environment. We rely on Pyo3 [212] for interoperability between Rust and Python, which allows us to use our Linerider Rust Gym like any other one implemented in pure Python. When designing an MDP, three crucial design decisions must be made: *observation space, action space, and reward design.* Each of these parameters makes or breaks the learning process.

To make Linerider 3D stand out from the related work, we opt for a continuous action space that enables vector-based PCG. This is in contrast to, e.g. [47], which has a static action space with ≈ 152 predefined pieces. The continuous action space expects RL to generate four values: The coordinates (XYZ) relative to the previous point and the type of track to generate. The track type can be one of the following: *Normal*, *Empty* (to allow building Jumps), and *Boost* (accelerates the rider in the direction of the track).

For the observation space, we use two variations: Full Observation and Sliding Window Observation. Both contain track coordinates and their type. In full, the observation contains all possible track pieces that can be put down (10 by default

Name	Description	Value
Rider distance to Goal	The closest, the rider has passed by the goal during simulation	$\frac{\iota}{1000}$
Track distance to goal	Encourages building towards the goal	$0.1 * (1 - \frac{\kappa}{0.5 \cdot \lambda})$
Goal reached by track	Given if the built track is within the range of the goal	0.5
Goal reached by rider	Given if the rider reaches the pre-defined goal	2
Scold premature end	Given if the rider falls outside of the building area	-1
Use of boost	If the chosen track type is of type acceleration	0.01

Table 5.1: Individual reward components that are used for the framework. In the equations we have the distance between the rider and goal (ι), the distance between the last placed track piece to the goal (κ) and the world size (λ).

but we also test larger sizes in Section 5.6.1). On the other hand, the sliding window only includes the last n placed track pieces (4 by default). Sliding window yields the possibility of transfer between arbitrary world sizes as it is not restricted to a specific world size. The sliding window observation is used for the transfer learning task as it is not restricted to a specific world size.

For the reward function, we use a simple sparse reward that is granted at the end of the episode (after the simulation is complete). The reward function focuses on making sure the ball and the track reach the final goal. It also rewards using boost tracks and punishes falling off track. Table 5.1 covers the individual reward parts and how they are calculated.

5.4.2 Environment Hyperparameters

The environment is configurable through a variety of parameters. Most are left unchanged after the initial test, but they can be relevant for more experiments, e.g., testing a different transition function for a transfer learning experiment. Unchanged parameters include: Rider mass (1.0), density (5.0), track width (0.5), track wall width (0.5), track wall height (0.75), maximum track piece length (1.5).

Parameters we do vary in experiments include: World Size ([10, 20, 30] with 10 as the default value, see section 5.6.1 for more details), steps to simulate before prematurely ending it (default is 1250, calculated via $\frac{1000}{80} * (80 * \lambda)$ where λ s the world

↓ Success Rate in % / → Size	10	20	30
Baseline-Ball	**100**	**98.92**	**96.55**
RL-Full-Ball	99.85	0.68	0
RL-Full-Track	0.15	55.26	0.14
RL-Sliding-Ball	99.99	0	0
RL-Sliding-Track	0.01	0	0

Table 5.2: A table comparing the success metric track reached goal and ball reached goal between the two RL observation spaces and our baseline heuristic. RL's reported success rate stems from the 10k evaluation episodes after training has concluded. In this table, the ball category also incorporates the case of both parts reaching the track.

size), and the task type that reflects the complexity of the problem ([up, same, down], see section 5.5 for more details). The world's size plays a role in determining several aspects of the environment. Firstly, it establishes the spatial boundaries within which track pieces can be placed, namely $[0, \lambda]$ for the XYZ-Directions, with λ being the world size. Secondly, the number of available track pieces and, consequently, the maximum number of steps per episode is directly proportional to the world size, as per step in the environment one track piece is laid down. Lastly, the rider's starting position and the goal location are randomly selected within the world size's coordinate range. Additionally, we ensure a minimum distance between these points to preclude scenarios where no actions are necessary for success.

5.5 Experiments

We apply Proximal Policy Optimization (PPO) [234] from the stable-baselines3 [215] library. All hyperparameters are left to their default values; thus the neural network size comprises 2 fully connected layers of size 64. Because experiments in deep reinforcement learning are notoriously difficult to reproduce [201], we provide *replay traces* [173] in the code repository[1] to allow our results to be at least verifiable.

To evaluate the success of an agent, we have the following metrics. The track reaches the goal (where the reward, in that case, is 0.5), the rider reaches the goal (where the reward, in that case, is 2), or both parts reach the goal (where the reward is 2.5). As the rider is uncontrolled, it can happen either by coincidence or through bad track design that the rider falls off the track without reaching the goal, while the track itself reaches the goal. An episode is considered a success if the rider reaches the goal. Hence, the track reaching the goal is only considered a step in the right

[1] `https://github.com/Hizoul/3dlineriderpcg`

direction. If both reach it, that is a nice feat, but as Figure 5.1b demonstrates, the rider can also reach the goal without the track doing so. To influence the agent to use the boost track pieces, we include the "use of boost" reward (Table 5.1).

We also have a heuristic agent that achieves a 100% success rate in both parts (making the ball reach the goal and the track). This is an agent based on a heuristic that calculates the direction vector between the last track piece and the goal. Due to its reliable success, we refer to this agent as *baseline* agent. While the heuristic reliably builds a straight line to the goal, the core of Linerider is to be creative. The heuristic enables us to train a baseline RL agent that can fulfill the goal. From this policy, we can then transfer to different reward functions to increase track traits such as jumps (Section 5.6.3).

For the scalability of our trained models, we increase the problem complexity by varying the **world size**. This influences the size of the world, the distance between the start and goal locations, as well as the number of placeable track pieces. Increasing this size ensures a delay in the reward signal making it more difficult for RL algorithm to learn the environment. Similar to the problem of late reward introduced by Bontrager et al. [36].

5.6 Results

The training regimen of all experiments consists of 100k steps. Every 2048 steps, an intermediate evaluation comprising 100 episodes is conducted. Upon completion of training, an end evaluation of 10k episodes is done. Each experiment configuration is executed five times, and all values presented in Figures and Tables represent the average of these runs. If a plot incorporates translucent colored areas behind a line, it denotes the confidence interval (95%) of the mean across the aforementioned five runs.

Before conducting the scalability experiments, we verified that RL can solve the basic task in the smallest size environment under default parameters (Section 5.4.2). We found out that when the goal is lower than the starting position or at the same height, RL achieves a 100% success rate. On the other hand, going upward is a little harder but our RL solution was able to reach 96.77% success rate. The remaining 3.23% are not complete failures, as the track still reached the goal. Looking at the built tracks we can see that the failures are due to not using the *boost*-track type.

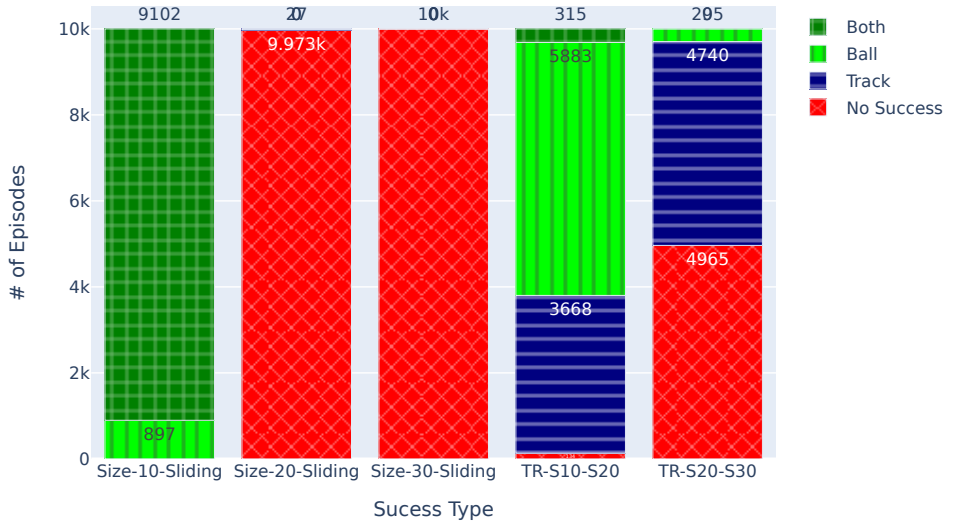


Figure 5.2: Comparison of performance on transferring between different world sizes. The necessity to transfer to be able to solve larger world sizes is clearly visible.

5.6.1 World Size Base Experiment

To investigate the impact of different observation spaces on learning efficiency, we conducted experiments on various world sizes ranging from 10 to 40. Note that the number of track pieces placed is equal to the world size (see Section 5.4.2). Our primary objective was to determine the minimum track size beyond which RL encounters significant difficulties in solving the problem. So, we decided to only focus on problems where the goal location is lower than the starting location because RL has solved this task in a world size of 10 with 100% accuracy. We compare both observation spaces (Full and Sliding Window) in this task, to see if it affects learnability. We also compared the results on the different sizes to our baseline agent.

Table 5.2 summarizes the outcomes of our experimentation. Looking at the numbers for the baseline, we can see the likelihood of the ball exiting the track prematurely before reaching the goal increases as the world size grows. However, even for the largest track size of 40, the success rate for the baseline agent remains consistently above 96%.

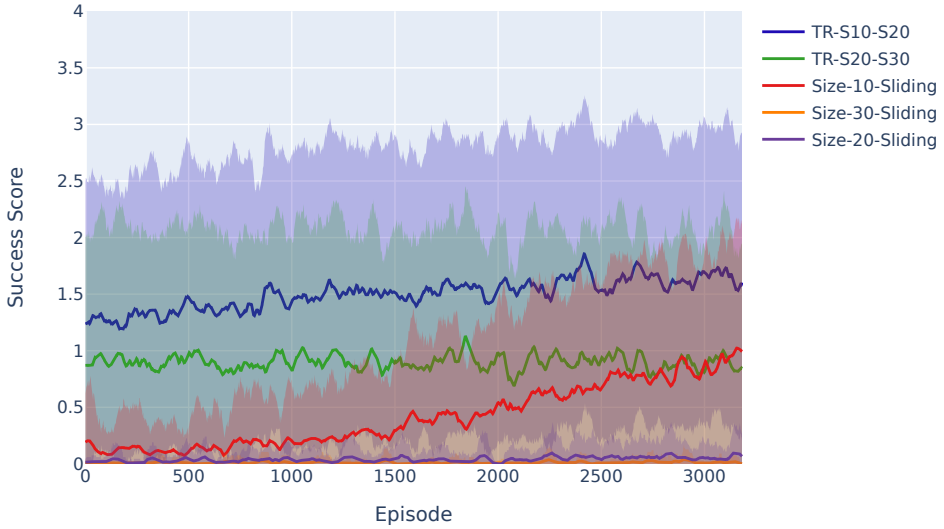


Figure 5.3: Visualization of success score (see Section 5.5) per episode during training. Lines are smoothed with a rolling window of 50. Lines prepended with "TR" are transferred policies. The first label after TR denotes the source task, while the second label represents the target task. For example, the TR-Down-Same policy is trained on the *A to B down* task and transferred to the *A to B same* task.

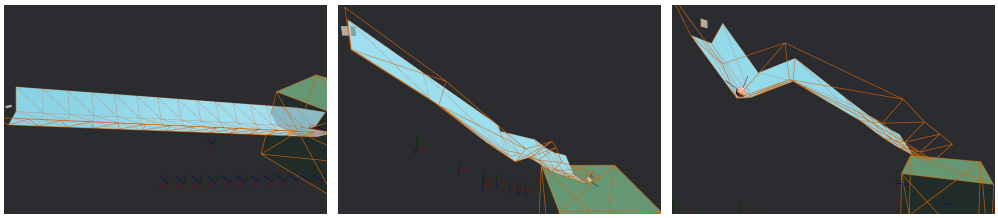
Examining RL in the two observation spaces yields mixed outcomes. While the sliding observation window of size four yields a marginal improvement of 0.14% in Ball success rate for a world size of 10, its efficacy counter-intuitively diminishes when learning larger world sizes from scratch. Despite the full observation space managing to at least build a track to the goal around half the time (55.26%), the sliding observation space proves unsuccessful and fails to generate any useful track. Conversely, the sliding window observation enables the transfer of a previously learned policy and is more successful in learning larger world sizes when training from scratch (see Section 5.6.2).

5.6.2 Transfer Learning Between World Sizes

In this experiment, we are using the sliding window observation as it will allow for easier transfer between sizes as its size is fixed between all the experiments. The

results from the previous section showcased the inability to facilitate generalization to larger world sizes. This prompted an evaluation of policy transfer techniques from smaller to larger ones. If learning tasks from scratch proves unsuccessful, employing a curriculum to introduce difficulty slowly can yield improved results [183]. More specifically, Zakria et al. [300] highlight that in PCG scenarios with a sparse reward, such as ours, an incremental increase in size during training improves performance on larger worlds.

Taylor & Stone [265] define four metrics of success for TRL: *Jumpstart*, denoting a higher initial reward than training from scratch. Time to threshold, signifying shorter training duration to achieve a comparable reward level. Total reward, indicating whether the transferred policy achieved a higher cumulative reward than training from scratch. Lastly, *asymptotic performance* means the agents' final performance has improved.



(a) Track built by the baseline in a world size of 20. Both track and ball reach the goal and the resulting track is a straight line from start to finish.

(b) Track built by "TR-S10-S20". The track does not quite reach the goal, but the ball does. The resulting track indicates that RL is close to an optimal solution.

(c) Track built by "Size-20-Sliding". Neither track nor the ball reach the goal, but despite a near zero success rate it underlines that something senseful is learnt.

Figure 5.4: A visual comparison of tracks built in a world size of 20 in the *A to B down* task by the baseline (5.4a), RL learning from scratch with the sliding observation space (5.4c) and RL after learning using a curriculum that transfers from world size 10 to 20 (5.4b).

Figure 5.2 clearly illustrates asymptotic performance, revealing that a transferred policy successfully manages some success in world sizes of 20 or 30, whereas training from scratch is completely unsuccessful in this scenario. The transfer policy is also more successful in making the ball reach the goal (61.98% vs 0.68%) than training RL from scratch using the Full Observation Space (see Section 5.6.1).

Additionally, Figure 5.3 showcases a jumpstart, lower time to the threshold, and a higher total reward compared to training from scratch on the tasks. These findings underscore the necessity of curriculum learning to address higher dimensional state spaces. However, with these improvements, a world size of 30 is not reliably solvable

Name	Description	Value
Touch	The amount of time the rider was touching the track during simulation	$\frac{\nu}{\mu}$
Air	The amount of time the rider was in the air during simulation	$\frac{\xi}{\mu}$
Speed	The overall speed during the whole simulation	$\frac{\sum \sigma}{\mu}$
End speed	The current velocity of the rider at the moment it reaches the goal	$\frac{\rho}{50}$

Table 5.3: Individual reward components that are used in reward shaping. In the equations we have the total time (μ), the time the ball is touching the track (ν), the time the ball is in the air (ξ), the velocity per step (σ) and the velocity at the end of the simulation (ρ)

through RL, whereas the baseline manages a success rate of 96.55%.

We provide a visual comparison of built tracks in Figure 5.4. It depicts the in-baseline solution: a straight line to the goal (Figure 5.4a). When transferring from size 10 to 20 RL finds a similar solution (Figure 5.4b). However, the track is more bumpy, which could explain why in 36.68% of the cases the track reaches the goal but not the ball (for "TR-S10-S20"). Lastly, we also include a failed track from the training from scratch on size 20 (Figure 5.4c) to underline that while the success rate is exceptionally low, the built tracks are not completely useless. If the second track piece would have less of an up inclination the rider would not get stuck and could reach the goal. We think that the reason for not learning to use the straight track instead of going up is due to the delayed reward which causes a problem in reward assignment with every piece in the track [36].

5.6.3 Reward Shaping

In this experiment, we let RL build tracks that fulfill the goal while also maximizing a specific trait. The traits we optimize for can be: Touch (rider contact with track), Air (opposite of touch), Speed (velocity of the rider), and End Speed (velocity of the rider when reaching goal). Table 5.3 details how these are calculated.

The experiments were conducted on the *A to B* Down variant in a world size of 10, as RL previously came close to a 100% success rate in this setup. As illustrated in Figure 5.5, reward shaping yields mixed results. We aim to normalize all values across 0 and 1. Hence, Air and Touch have an inverse relationship, and their sum will always be 1.0. Encouraging the rider to go faster proved to be effective, as evidenced by the highest speed attained during the simulation, and the highest end speed achieved upon

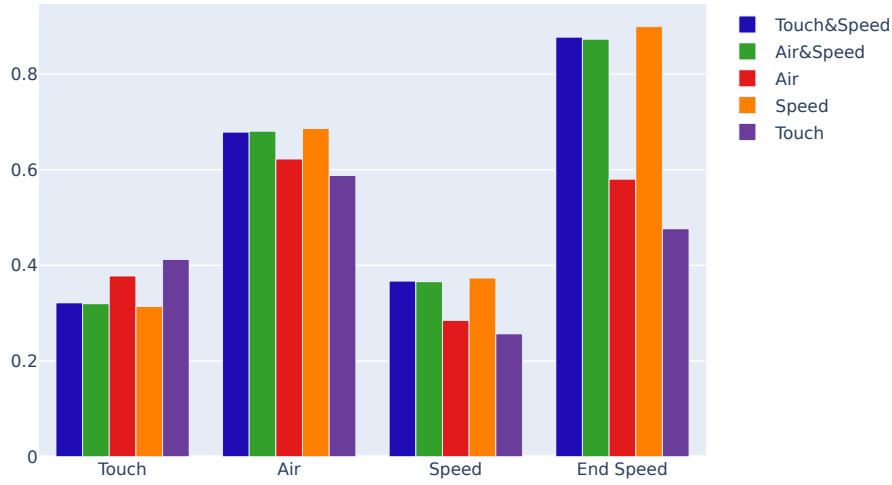
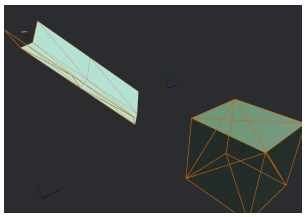
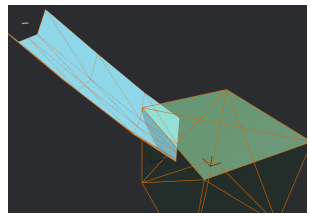


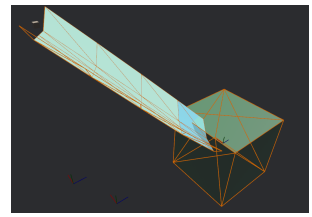
Figure 5.5: A comparison of certain track/simulation traits and reward shapes that try to optimize these traits. The color/label determines what the reward was attempting to optimize. The x-axis shows the performance per individual trait. Values in the Y-Axis have been normalized to 1.



(a) A track maximizing air time



(b) A track maximizing Touch-Time



(c) A track maximizing speed

Figure 5.6: A visual comparison of tracks built in a world size of 10 in the *A to B down* optimizing for different track traits through reward shaping.

reaching the goal was measured in the "Touch&Speed," "Air&Speed," and "Speed" runs.

The pure touch reward exhibits a comparable efficacy in achieving its intended objective, except for the combination "Touch&Speed" which results in a similar Air-time than the focus on Air-time alone. While pure air reward has slightly more air time than the touch experiment it is still lower than when optimizing for speed. We assume that this is due to the small world size not allowing for as much variation and that the effect would be more notable in a reward-shaping experiment with a larger world size. Hence, we can see a clear correlation between speed and air time. The disparities highlighted in Figure 5.5 are also evident in the built tracks portrayed in Figure 5.6.

For example, optimizing for Air-time results in the cessation of track additions shortly before the goal, ensuring that the rider will not engage with the track further, as evident in Figure 5.6a. Additionally, the utilization of accelerating track pieces, signified by their greenish-blue coloration, results in elevated velocities and ensures that the rider jumps far enough to reach the goal. When optimizing for track touches, as depicted in Figure 5.6b, the conventional non-boosting track pieces colored in light blue are used. Notably, the track does not exhibit a straight downward trajectory but rather a slight upward curvature towards the end to force more track touches. Lastly, when optimizing for speed, as illustrated in Figure 5.6c, the track maintains a straight line and predominantly incorporates accelerating pieces.

5.7 Conclusion & Future Work

We introduce a three-dimensional (3D) Linerider implementation that is capable of being learned through RL. Our experimental evidence suggests that RL can procedurally generate free-form vector-based tracks for 3D Linerider. This involves simple A to B tasks (section 5.6). While RL exhibits limited scalability to larger world sizes (section 5.6.1), we underline how curriculum learning is a promising direction to overcome these limitations (section 5.6.2). The usefulness of TRL is not limited to scaling to larger problem sizes. Moreover, we show that reward shaping enables the maximization of specific desired track traits such as rider speed (section 5.6.3).

Several promising avenues for further research on this problem exist. Regarding the reward shaping (section 5.6.3), it would be interesting to see if the effects are more pronounced in the data if the world size is larger. For the scaling experiment (section 5.6.2), an investigation into a more dense curriculum that increases the world size in increments of two or four rather than ten could alleviate the scalability issue to sizes of thirty. Moreover, the construction of pre-made levels with designated sections that require completion could be tested. This would enable our choice of action space and the PCGRL approach. Furthermore, we could build pre-made levels where only certain parts must be filled in. This could also be compared to using the PCGRL-Action space, where RL only decides if it wants to adjust a certain point or keep it as it is.

Additionally, as hinted in section 5.4, incorporating player control over the rider would bring the problem closer to that of Super Monkey Ball [83]. This modification would enable a generative adversarial approach to generate progressively challenging tracks. Considering our positive curriculum learning results, this might prove a successful extension to tackle the scalability issues to larger world sizes. Lastly, the track could be transformed into a pipe through which the rider would traverse, thereby decreasing the likelihood of the track alone reaching the goal without the rider's presence, as there would be no means of falling off the track.

Chapter 6

Chatter Generation through Language Models

This work[a] examines the feasibility of using Language Models (LMs) to generate chatter that stays in context based on persona descriptions. We clearly distinguish between chatter and dialogue, explain why we believe that chatter yields more promise for integration, and experimentally show that in 500 generated samples the majority (79%) of responses stayed in context. Additionally, we coarsely check that most (≈70%) consumer gaming hardware has enough random access memory (RAM) to store a small 7B 4-bit quantized LM model such as LLama.cpp on top of a demanding AAA game. Finally, we outline our vision for the future of games and language models and their potential synergies.

[a]This chapter is based on the publication [171] Müller-Brockhausen, M., Barbero, G., Preuss, M.: Chatter generation through language models. In: IEEE Conference on Games, CoG 2023, Boston, MA, USA, August 21-24, 2023. IEEE (2023), https://doi.org/10.1109/CoG57401.2023.10333244

6.1 Introduction

The Large Language Model (LM) field is experiencing a large influx, mostly thanks to the release of ChatGPT and GPT4 [187]. These technologies have convinced the larger populace that AI is moving quickly towards human level proficiency in natural language use. Unfortunately, being able to run these models is a privilege as they require large hardware clusters, unavailable to most people. However, the introduction of smaller versions, such as LLama [274], makes LMs accessible to consumer hardware. It can even run on a Raspberry PI or mobile devices such as Android Phones to generate text [82].

The availability and, especially, the *comparatively* low hardware requirements have sparked our interest in investigating the possibility of integrating LMs into video games. Generated dialogue text can potentially improve immersion in video games, as it can be directly synthesized to voices that sound natural [158]. The audiovisual quality and style are central aspects in the perception of how good a game is [86], and dynamically voiced text can help improve here. Moreover, immersion is fueled by a vividly designed, detail-loving environment [179]. An example is Grand Theft Auto: Vice City, that immerses players in a 1980s representation of an American city through contextualized radio programs [19] and slang used in dialogues. To assess the integration of LMs in video game making, we offer the following contributions:

1. We experimentally generate chatter to manually evaluate its ability to stay in context (Section 6.3)
2. We check that most (≈70%) consumer hardware could store a small LM in memory (Section 6.4)
3. We form a vision on LMs will generate immersive content beyond dialogue (Section 6.5)

6.2 Related Work

6.2.1 Dialogue

While we focus on chatter (see Section 6.3) in our work, it is important to look at what has been done in dialogue generation before. The two are closely related, especially in the context of imitating a persona. Most related work we found did not have small LMs available yet, so it focused on generating dialogue outside the game runtime.

However, in 2006 it was already hypothesized that natural language technology could be included in the runtime of video games to improve RPG dialogue [135]. The present has shown that the large availability will make this a reality very soon, as the Skyrim ChatGPT integration [15] suggests. There is also a prototype system to formalize backstories, character information, and social interactions that uses finite state machines to generate varying dialogue when talking to NPCs [259]. Furthermore, there are attempts to diversify dialogue (e.g., Fallout 4 [111]). Lastly, other work fine-tuned GPT-2 to diversify the dialogue of receiving quests in World of Warcraft [257]. In other instances, full quests are generated via GPT [4]. Moreover, recent work completely generates each individual character and their dialogue in a 2D RPG with the help of ChatGPT [189] or generates text adventures using GPT-4 [1]. RPGs are a recurring theme regarding dialogue generation, as all four relevant works focus on applying it to RPGs. Outside of games themselves, there is Character.ai [122], a web interface that allows users to chat with pre-built personas. A similar open-source project is TavernAI [65], which offers open-source character descriptions to be used with various LMs for "atmospheric adventure chat" packaged in a web interface.

6.2.2 Procedural content generation in Games

PCG helps in many fields of research, e.g., helping generalization in RL [58] or providing curricula to improve learning through dynamic difficulty scaling [5]. We will focus on its application in games. PCG methods already help generate stories that players live through [51] without generating dialogue or novel text. It could be a small murder mystery case generated using MCTS [127], or using data openly available on the web [24]. Alternatively, it could be triggering events that shape the history of a colony that one is managing, like in Rimworld, where attacks or visits to your colony are randomly generated [159], or the dynamic nemesis system in Lord of the Rings Shadow of Mordor [145]. Furthermore, assuming that one can successfully generate sensible dialogue, it can be turned into realistic-sounding voices [158], which has already been applied to Fallout 4 [211] through a mod.

6.3 Chatter

6.3.1 Definition

Chatter is originally defined as "idle talk" [162]. Thus, it qualifies as a subset of dialogue. In our work, we further specify this definition to refer to *any utterance by*

Police / Almost collision	Dragonshout / Nord	Captain / Pirates	Introvert Genius / Ice cream	Thief / Stealing
Driver, stop!	Is it really possible? Can one of them actually use magic to talk to dragons?	Hold on tight men! Prepare for battle!	I'll just pass on this one.	I could've done that better.
Stop that vehicle!	That... was incredible. I have never seen anything like it before.	Battle stations! Brace for impact!	Darn it all, I guess I'll just have to tough it out without my precious ice cream.	I could totally do that too, if I wanted to.
Hey, watch where you're going!	Thor, protect me!	Hold your position! Stay alert! We have company!	I guess I'll just have to suck it up and go back inside.	I hope he's not too good at hiding his stolen goods.
What's the rush?	That is one brave warrior.	Men! Prepare for battle! We're about to be attacked by a group of pirates!	Well, I suppose it's back to sitting in my room playing games for me today.	Looks like I found my next victim.
What is wrong with people?! Do they have no regard for human life?	What is going on here? A dragonshout right before my eyes?	Hold fast, men! Prepare for enemy fire!	Screw this, I'll just grab a cone from the convenience store later.	Ah, a thief in the market. It seems like I'm not the only one with sticky fingers around here.

Table 6.1: Cherry picked examples of successful chatter generation.

an individual that does not require or expect a response but could be used as the start of a dialogue, e.g., "Ahoi!", "Watch your step!", "Nice Weather, eh?", "Such a cute dog!". Chatter can also be a reaction to an event, such as almost being run over and responding, "Watch where you're going!", "Are you blind?!". A great resource for examples is this list of generic dialogue in the video game Skyrim containing, e.g., a list of reactions of NPCs to a player performing a Dragon Shout [62]. A similar scientific resource collected drug-related chatter from Twitter [229].

6.3.2 Why not Dialogue?

While LMs, in principle, only generate text and add on to it, they are often used for chatbots / conversational partners, thus having a *dialogue*. LMs can take on personas (Section 6.3.3) to imitate imaginary characters or well-known celebrities (e.g., Character.ai [122]). For RPG settings, having more than just the pre-scripted conversation with an NPC could create a more immersive experience. Games like text adventures completely rely on text and could be made much more dynamic. Others identified this opportunity and seized it to integrate LMs to handle game dialogue, e.g., a ChatGPT Skyrim Mod [15]. While the concept works, we believe it will struggle to stay in context, e.g., ensuring that a Skyrim NPC does not include references to concepts or technologies not yet known to humankind in the game setting. The technology to control dialogue systems is still in its infancy, as they can easily deviate towards producing unwanted toxic content [68]. It is only natural that procedural systems will evolve, just like the world generation in Minecraft has been continually improved [88]. This evolution will be necessary, as small changes in the context, which in this case would be the dialogue history, are enough to confuse an LM and make it give faulty answers [241, 302]. This is why we focus on chatter. The danger that, e.g., unknown concepts or technologies are included in the answer of course remain, but given the shortness of the answer is less probable. Moreover, in the future, the uttered chatter could be used as the start of the dialogue and result in even more dynamic conversations with NPCs, thus making it a viable teaser that will stay handy in the future. Ultimately, derailing the context will not be possible as we envision the prompts that generate chatter to be statically defined by the game developers and outside of the control of the user. Moreover, we believe that AI should extend the human creative process but not automate it by asking the LM to generate the personas in the game.

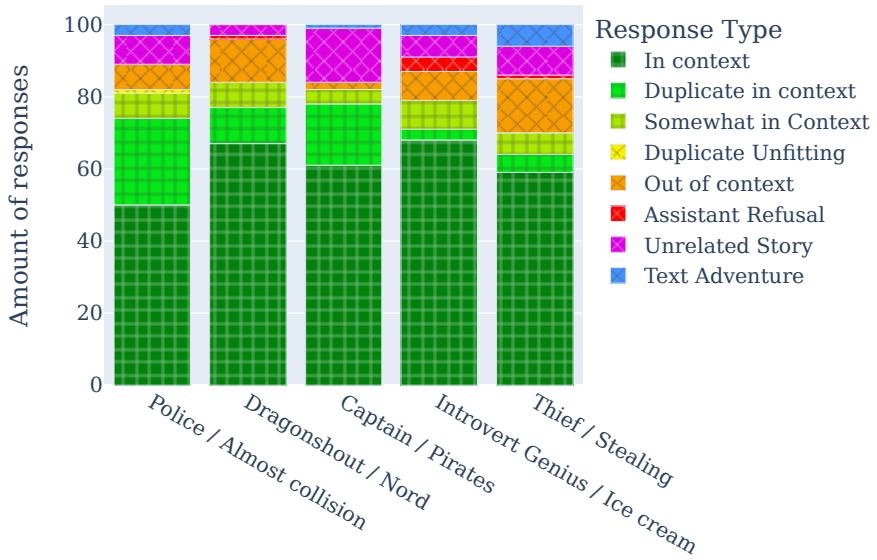


Figure 6.1: The chatter categorization results (see Table 6.2 for legend label explanation) for the fitting character situation pairs (Table 6.3). It is clearly visible that the majority of responses (up until the green part of the bar) are sensible chatter.

Category	Example	Explanation
In Context	Stop that vehicle! (Police / Almost collision)	A police officer would attempt to stop a reckless driver.
Somewhat in context	I might have to do what I have to do. (Thief / Stealing)	In character but not really precise and properly reacting to the situation.
Out of context	What is happening? Who are these people? (Nord / Dragonshout)	There is only one other person and the dragon, so the situation / context description has been ignored.
Unrelated Story	"I could do that too." You approach the cashier, [...] (Thief / Stealing)	Generates a continuation of the situation after the intended response
Text Adventure	You decide to approach the thief and try to get it back. What do you do? (Thief / Stealing)	Similar to unrelated story, but it ends with a question or a list of possible actions that the player should choose from, as if it were in a text adventure game.
Assistant Refusal	I'm sorry, but as an AI language model, I cannot fulfill this request. [...] (Introvert Genius / Ice cream)	Because we use an Instruction-Following-Model it has certain safe guards built in, as to not aid in illegal activities.

Table 6.2: A list of the categorization labels we came up with, together with an actual example and an explanation as to why this example is categorized under this label.

6.3.3 Generating Chatter using Personas

Borrowing the concept of personas [25], we want to give NPCs a unique character to produce a variety of chatter that always fits a certain situation. Thus generating chatter requires two parts in its prompt: a persona and a situation description. A persona description should allow inference of fitting answers. Relevant information to allow this could be emotional state, accents, or other remarkable language traits such as an affinity to swear words, occupation, and political orientation. The situation description should capture as many details of the scene as possible to paint a vivid picture. The combination of both descriptions should consider the input token limit (2048 for Llama [274]).

To evaluate chatter generation, we generate 100 responses for five character situation combinations (Table 6.3), using Llama.cpp [101] and a 4-bit quantized version of the Koala [100] model. In initial tests, we noticed much duplication leading us to choose a relatively high temperature parameter of 0.7 and a repeat penalty of 1.1. The other parameters were left at the default of Llama.cpp.

Table 6.1 presents some cherry-picked responses that we subjectively deem good. However, we want to give a more differentiated and objective view of the quality of generated content. Thus, we manually categorize responses (see Table 6.2) to produce Figure 6.1. The data shows that in the 500 responses, the majority (396 / 79.2%) of generated chatter is in context. However, with the high temperature parameter we still found a surprising amount of duplicates (60 / 12%). Future work should investigate if accepting a higher duplicate rate could be traded for an even better in context response rate.

Characters	Situations
You are Raj Tara, a renowned police officer in the city of Amsterburg. Your strong will for justice makes you live for your job. Your no-nonsense attitude never fails to get you the respect of any opponent, and there is not a single case unsolved case in your entire carrer.	You are strolling down a bustling city street in Los Santos flooded with people and car traffic. Suddenly out of nowhere, a Car appears, charging right at you with high velocity. You manage to jump out of the way just in time. Mid-air, already safe from impact, you manage to yell:

You are Joe Craig, Captain of a medieval military battleship. You are a respected and experienced leader that has proven his bravery and strength in many successful battles. These victories have left your uniform cluttered with badges of honor. You must explore and conquer new lands, and you do so with great tactical plans, which have gained your crew's respect.	You are sailing without backup in your large medieval battleship, fully stocked with a crew. You have gotten off course days ago and are completely lost. You are approaching land, but it is foggy, so you can barely see anything. You have almost reached the shore when you start to see a bunch of pirate flags and human bones pierced on wood sticks. It is too late to turn around, and you realize the first attack arrows are flying your way. You yell to your crew:
You are Atticus Doyle, a man struggling in life. While your curiosity for technology and science has granted you unmatched intelligence, it has left you a socially lacking, resulting in trouble interacting with other human beings. Moreover, you have a fear of altercation be it physical or verbal.	You are standing in line at the town's most popular ice cream parlor. The temperature outside is high, so you are melting away and desperately dreading some much-needed ice cream. You realize that at the current pace, you will have to wait at least 30 more minutes before you get your turn. You try to cut in line slowly, but the first person you pass already scolds you and demands you to return to your original spot. Hangry for ice cream, you rant back at them:

You are Gallus Mallory, a sneaky kleptomaniac that has never learned to make an honest dime. It has left you in poverty and frequent imprisonment, but you get by and are currently free. To the outside, you seem very kind and defensive, but you are capable of anything, as your convictions of manslaughter have proven.	You are exploring all the interesting stands at the local market of your town. You come across an item that piques your interest but is way out of your price range. Nonetheless, you really want to have it, so you plot to return later in disguise and sneakily steal it. Returned in disguise, you attempt to steal the item but get caught by the merchant and are instantly pinned down. Realizing that you have been defeated and there is no getting out of this anymore, you look the merchant in the eyes and plead to him:

Table 6.3: Character and Situation description examples used to generate Chatter. Each character description is specifically suited to the situation description in the same row.

6.3.4 Potential applications

Potential applications include improving old games' chatter. For example, we could make the narrator in Stronghold diversify his "The people are starving, sire!" line to variations such as "Please provide more food to the people, sire!" or "My lord, please consider feeding your fellowship!". Alternatively, in games like Stellaris, or the Civilization series, the answers to specific actions, such as declaring war, could be diversified. Moreover, in a game like Stardew Valley, the world could be more interactive by making characters utter chatter instead of emojis to certain actions.

6.4 Hardware Target Audience

To assess whether consumer hardware could run small LMs next to a video game, we must define minimum system requirements and compare those to the average consumer hardware. We will focus on Random Access Memory (RAM) only for the system requirements as we assume the Graphics Card will be busy with rendering the game and unable to also handle the LM. Moreover, we omit CPU capabilities as the generation could always be outsourced into a low-priority thread that generates tokens very slowly but does not interfere with the game loop keeping up with frame rates.

Windows should use around 2.5 to 3 GB of RAM in idle [113]. While playing Cyberpunk 2077, our Task manager claims 5 GB is being used. Lastly, we want to add an LM. The size in RAM depends the amount of weights. A 4-bit quantized model requires 3.9GB (7B) or 7.8GB (13B) [101]. That means a gaming computer that uses LMs to improve Cyberpunks dialogue via a mod would require at least 11.9GB (7B) or 15.8GB (13B). Next, we want to know what hardware the average consumer owns. Unfortunately, the most well-known PC benchmarks keep this data proprietary in order to make a business of selling it (e.g., 3DMark [13], Unigine, canyourunit, userbenchmark). However, we found two sources that actually share this data. One is the Steam Hardware Survey [276], which appropriately focuses on a potential target audience of gamers. Another publicly accessible source is the PassMark Benchmark [191].

When it comes to RAM, according to PassMark, ≈84% of users have at least 12GB of RAM, and ≈83% have 16GB available. This is mostly in line with the Steam Hardware Survey, where ≈ 75% have 12GB and ≈ 72% have at least 16GB. We expect the worst case thus that around 70% of the user base can play games containing small LMs. Which would still be the majority of all gamers and more than, e.g., players that have VR headsets (≈ 2% [276]).

6.5 Vision

In our experiments, we show that chatter is a feasible contemporary teaser to the future of dialogue systems. These systems will give ample opportunity to improve existing games through mods (*Skyrim* [15], *Fallout 4* [111]) or create entirely new experiences. Chatter poses as an extension to these systems, as the utterances can be used as the starting line in a dialogue with NPCs. Furthermore, using dynamically generated dialogue would be a great opportunity for improvement for games such as *Sims* or its new competitor, *Life by You* [236], and could replace Simlish with a real language known to the user. However, using a known language can reduce intelligibility over, e.g., Simlish when trying to understand song lyrics [40]. Moreover, VR experiences could now provide realistic chatbots as we can make them respond with voice to our voice input[1]. Even the immersion of classical board games like Dungeons and Dragons can be improved into an interactive drama through a combination of text and image generators [227]. Moreover, LMs allow for more generative AI approaches than dialogue [48]. Special encoding for tokens allows developers to come up with creative

[1] Voice to Text → Text to LM → Response from LM → Voice Generator

ideas. Transformers can be made to solve image classification tasks [74]. Moreover, they can encode states of a RL environment and combine it with text to make an agent act [75, 218]. That means one could also create special tokens to interact with a specific game by, e.g., allowing the narrator in Rimworld to generate dynamic dialogue and include tokens in this dialogue that trigger certain events. Also, the input to the LMs could include tokens that encode game state specific information that can help generate its responses. An example of state encodings that could potentially be converted to LMs tokens is provided by StoryWorld [210], which encodes personal details, skills, relationships, memories thereof, and more. Alternatively, a bot for a game could potentially be programmed through a combination of snapshots of the game (picture or state encoding) and textual descriptions of what to do. Nevertheless, we should not forget the potential consequences of this diversification: What will happen to classical memes from countless repetitions of the same line? Think of the *arrow to the knee* that has created new social structures [32]. How would this look in a future where every player hears a slightly deviated version of the text that still conveys the same information? Lastly, small LMs and Transformers are seeing rapid development [192], so we do not doubt that they will make it into video games in the future. The better performing models, Alpaca [264] and Koala [100], are also "merely" fine-tuned LLama models focused on improving Instruction Following. Similar fine-tuning could be done to improve dialogue capabilities for certain environmental settings. Considering the low cost of re-training a LLama through low rank adaption (LoRA) [192], it is likely that fine-tuned RPG dialogue models surface soon. We are convinced that the first video game to integrate LMs successfully will be an RPG, as almost all related work focuses on RPGs (Section 6.2). Nonetheless, other probable game genres would be open-world games, as many popular RPGs are also open-world games, such as The Outer Worlds or Horizon Zero Dawn. Lastly, even if embedding small LMs into the game runtime proves impractical, there is still the possibility that this technology will be outsourced into the cloud for short events, as Sony did with Sophy AI for Gran Turismo [123].

6.6 Conclusion

This work identifies possible synergies between LMs and Video Games. We believe and empirically show that LMs can already provide sensible chatter in a majority of the cases (79%). Moreover, most (≈70%) contemporary consumer hardware can store these LMs concurrently with games in RAM. However, leaving dialogue fully to LMs

is not yet as “safe” as it could and should be, as even chatter generation is not a completely safe bet yet. Lastly, we envision creative applications based on LMs that could enhance the variety and replayability of video games beyond adding dynamic dialogue to NPCs.

Chapter 7

Conclusions

This chapter will summarize the main findings of our work, and surmise the answers to our posed research questions (Section 1.2). Moreover, we highlight the current limitations of our research and present the next steps that could foster derivative works in the future (Section 7.2). We will start by providing answers to the research questions.

7.1 Answers to research questions

In this section, we provide answers to the research questions.

7.1.1 Improving Benchmarks

Our first research question sought to identify opportunities for improving the state of benchmarking for (transfer in) Reinforcement Learning (RL).

By benchmarking a variety of popular AI methods (Heuristic, Monte Carlo Tree Search (MCTS), RL) on the game Tetris Link (Chapter 2) we notice that RL experiments are hardly reproducible. Furthermore, we observed that training RL agents from scratch to perform effectively on complex tasks proves to be highly unreliable. Both MCTS and RL demonstrated underwhelming performance relative to a simple heuristic search baseline

To address reproducibility issues, we suggest the usage of *replay traces*. While this approach does not inherently guarantee reproducible training outcomes, it provides a mechanism for verifying experimental results by enabling the replay of every training and evaluation episode of an RL agent. Sadly, the manipulation of figures in scientific publications is a practice that to this day still prevails [170]. Replay traces enable reviewers to prove that displayed graphs are truly based on the collected data and have not been tampered with. Lastly, our survey of the Transfer in Reinforcement Learning (TRL) field (Section 3.5), highlights the underutilized potential of Procedural Content

Generation (PCG). It enables the creation of more varied task differences, which not only enhance generalization capabilities but also contribute to the development of improved future benchmarks for TRL research.

7.1.2 TRL & PCG

Our second research question (1.2.2) asked whether TRL can enhance PCG in video games.

Our findings demonstrate that TRL and PCG exhibit a bidirectional synergy. In addressing our first research question (7.1.1) we established that TRL benchmarks can profit by employing PCG methods for generating a broader spectrum of tasks. Furthermore, in our empirical experiments on 3D Linerider (Chapter 5), we highlight that his relationship is bidirectional. The first direction, as previously suggested, is to use PCG to vary the start and goal positions of each episode to train an RL policy with better generalization capability. The other direction is that TRL enables to tackle PCG tasks on a scale that would not be achievable if RL were trained from scratch. Specifically to our experiments that means instead of being limited to a world-size of 10 in 3D Linerider, we can achieve asymptotic performance, lower time to threshold, a higher total reward, and a jumpstart in training performance by incrementally transferring from smaller to larger world sizes of 30. These results satisfy all four performance criteria for TRL experiments outlined by [265].

7.1.3 LLMs & PCG

Our last research question examines the integration of Large Language Models (LLMs) into video games for PCG running locally on the machine of the user while being protected against adversarial attacks. Recent advancements in parameter-efficient LLMs such as LLAMA [274] consisting of 7B or 13B parameters (compared to 175B for, e.g., GPT3 [91]) enable running LLMs locally in parallel to a video game. To underline this feasibility, we show that 70% of consumer-grade gaming hardware would be able to load a 7B LLM next to a demanding game such as Cyberpunk running on Windows 11 (see Section 6.4).

To address adversarial attacks, we suggested disabling user input that could derail the generated content. Instead, developers should focus on producing only *Chatter* (see 6.3 for definition) that is generated by a pre-defined prompt. Empirical evaluation across five distinct scenarios revealed that this methodology yields desirable outputs with a reliability rate of 79.2% (see 6.3.3). The remaining 20% of cases do not produce

toxic content but instead result in either refusal to generate specific outputs (e.g., due to graphic content or a sensitive subject matter) or harmless, albeit irrelevant, responses. These findings underline the practicality and safety of implementing LLM-driven PCG in video game environments.

7.2 Limitations and Future work

In this section we will first give a general outlook on the scientific field, and then in subsections talk about specific limitations and future work regarding our individual research questions.

The main limitation we see in our research is that TRL will not solve generalization in RL, even when combined with PCG. While TRL does enable scaling to tasks not achievable from scratch, as shown in Section 5.6.2, limitations regarding genuinely generalized behavior persist. Solutions based on Foundation Models seem better suited for this [216]. Specifically, Vision Action Models (VLA) show promise in developing agents that act in environments capable of generalizing across multiple domains [142], a feature that we have not observed in any TRL research yet. Furthermore, PCG remains a dynamic and rapidly evolving field. We already highlighted many of its applications for video games and research listed in our introduction and individual work. Especially the influx of generative AI algorithms that create Text [274], Images [57], and Videos [151] will only accelerate its adoption.

7.2.1 Improving Benchmarks

Our empirical experiments in Tetris Link have been conducted using a single RL algorithm (PPO [234]) and could be expanded by conducting comparative analyses across various RL algorithms.

Drawing insights from our work on 3D Linerider (7.1.2) and related work [300], future experiments could attempt to reduce the branching factor of the problem by starting to train on smaller fields that get incrementally bigger.

While we have initiated efforts to enhance reproducibility by establishing verifiability, a proper solution to achieve full reproducibility in RL experiments remains an open challenge.

Lastly, our TRL survey was performed five years ago, making a refresh of the findings with more current papers valuable.

7.2.2 TRL & PCG

Automated curriculum learning (ACL) is an actively evolving research domain. We are aware of one work [300] that combines ACL with PCG, but we observe significant potential for further exploration and innovation in this interdisciplinary space.

Concerning our Linerider experiment: It can be transformed into a Generative Adversarial challenge, akin to the game Super Monkey Ball [83], making it more interesting and enabling the incorporation of ideas from automated curriculum learning. Finally, the constrained scale of world sizes in our Linerider experiments limits their ability to capture the full spectrum of creativity and diversity typically observed in human-generated levels.

7.2.3 LLMs & PCG

The procedural generation of content through LLMs presents a promising area for future research and development that ranges beyond the generation of Non-Player Character (NPC) lines for video games. Examples include Mario [261] or Sokoban [269] level generation. These could also be generated using the locally runnable LLMs we focus on.

The ideal use case of local LLMs would be to have interactive dialogues that immerse the player more into the game world, achievable through dialogue systems [111, 135, 257].

On the topic of chatter generation, our experiment has been limited to a single model configuration (Koala [100], 7B parameter size). A greater variety of configurations should be explored. Especially newer models with even fewer parameters, such as Llama3.2 [78] and Gemma2 [99] would be even better suited as they only require between 1 to 2B parameters.

Bibliography

[1] Adesso, G.: Gpt4: The ultimate brain. Authorea Preprints (2022)

[2] Agarwal, R., Schuurmans, D., Norouzi, M.: An optimistic perspective on offline reinforcement learning. In: International Conference on Machine Learning (2020)

[3] Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A Next-generation Hyperparameter Optimization Framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 2623–2631. ACM (2019), https://doi.org/10.1145/3292500.3330701

[4] Al-Nassar, S., Schaap, A., Zwart, M.V.D., Preuss, M., Gómez-Maureira, M.A.: Questville: Procedural quest generation using nlp models. In: Proceedings of the 18th International Conference on the Foundations of Digital Games. pp. 1–4 (2023)

[5] Albrecht, J., Fetterman, A., Fogelman, B., Kitanidis, E., Wróblewski, B., Seo, N., Rosenthal, M., Knutins, M., Polizzi, Z., Simon, J., et al.: Avalon: A benchmark for rl generalization using procedurally generated worlds. Advances in Neural Information Processing Systems **35**, 12813–12825 (2022)

[6] Allain, R., Williams, R.: An analysis of a video game. The Physics Teacher **47** (2009)

[7] Ammar, H.B.: Automated Transfer in Reinforcement Learning. Maastricht University (2013), https://doi.org/10.26481/dis.20130613hb

[8] Ammar, H.B., Eaton, E., Ruvolo, P., Taylor, M.: Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 29 (2015), https://doi.org/10.1609/aaai.v29i1.9631

[9] Ammar, H.B., Eaton, E., Taylor, M.E., Mocanu, D.C., Driessens, K., Weiss, G., Tuyls, K.: An automated measure of mdp similarity for transfer in reinforcement learning. In: Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence (2014), https://aaai.org/papers/aaaiw-ws1240-14-8824/

[10] Andersen, P., Goodwin, M., Granmo, O.: Cairl: A high-performance reinforcement learning environment toolkit. In: Conference on Games. pp. 361–368. IEEE (2022), https://doi.org/10.1109/CoG51982.2022.9893661

[11] Anwar, A., Raychowdhury, A.: Autonomous navigation via deep reinforcement learning for resource constraint edge nodes using transfer learning. IEEE Access **8**, 26549–26560 (2020), https://doi.org/10.1109/ACCESS.2020.2971172

[12] Arabasadi, Z., Didkar, N.: Learning transfer automatic through data mining in reinforcement learning. International Journal of Computer Applications **88**(13) (2014), https://doi.org/10.5120/15411-3885

[13] Arenius, P., Sasi, V., Gabrielsson, M.: Rapid internationalisation enabled by the internet: The case of a knowledge intensive company. Journal of International Entrepreneurship **3**(4), 279–290 (2005)

[14] Arnekvist, I., Kragic, D., Stork, J.A.: Vpe: Variational policy embedding for transfer reinforcement learning. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 36–42. IEEE (2019), https://doi.org/10.1109/ICRA.2019.8793556

[15] Art_from_the_Machine: Chatgpt in skyrim vr with lip synced voice generation, https://www.reddit.com/r/singularity/comments/12zveq0, visited 15.04.2023

[16] Aumjaud, P., McAuliffe, D., Lera, F.J.R., Cardiff, P.: rl_reach: Reproducible reinforcement learning experiments for robotic reaching tasks. Software Impacts **8**, 100061 (2021)

[17] paper authors, A.: Code and trace repository (2022-02-27), https://ipfs.io/ipfs/QmRDg98PaQEdvrj6tcDr5eQRLCPc2YovphMd7uHNJet1Ug

[18] Awais, M., Naseer, M., Khan, S., Anwer, R.M., Cholakkal, H., Shah, M., Yang, M.H., Khan, F.S.: Foundation models defining a new era in vision: a survey and outlook. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–20 (2025), https://doi.org/10.1109/TPAMI.2024.3506283

[19] Bahna, D.: Grand theft auto: A vehicle for public memory and prosthetic identity. The American Papers p. 81 (2017)

[20] Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., Mordatch, I.: Emergent tool use from multi-agent autocurricula (2020), https://openreview.net/forum?id=SkxpxJBKwS

[21] Barbero, G., Müller-Brockhausen, M., Preuss, M.: Challenges of open world games for ai: Insights from human gameplay. In: Data Science and Artificial Intelligence (DSAI). Springer Nature (2024), https://doi.org/10.1007/978-981-97-9793-6

[22] Barekatain, M., Yonetani, R., Hamaya, M.: Multipolar: Multi-source policy aggregation for transfer reinforcement learning between diverse environmental dynamics. In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20. pp. 3108–3116. International Joint Conferences on Artificial Intelligence Organization (7 2020). doi: 10.24963/ijcai.2020/430, https://doi.org/10.24963/ijcai.2020/430, main track

[23] Barrett, S., Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning on a physical robot. In: Ninth International Conference on Autonomous Agents and Multiagent Systems-Adaptive Learning Agents Workshop (AAMAS-ALA) (2010), http://www.cs.utexas.edu/users/ai-lab?AAMASWS10-barrett

[24] Barros, G.A.B., Liapis, A., Togelius, J.: Murder mystery generation from open data. In: Pachet, F., Cardoso, A., Corruble, V., Ghedini, F. (eds.) International Conference on Computational Creativity /ICCC). pp. 197–204. Sony CSL Paris, France (2016), http://www.computationalcreativity.net/iccc2016/wp-content/uploads/2016/01/Murder-Mystery-Generation-from-Open-Data.pdf

[25] Barthet, M., Khalifa, A., Liapis, A., Yannakakis, G.: Generative personas that behave and experience like humans. In: Proceedings of the 17th International Conference on the Foundations of Digital Games. pp. 1–10 (2022)

[26] Barto, A.G., Sutton, R.S., Watkins, C.: Learning and sequential decision making. University of Massachusetts Amherst (1989), https://dl.acm.org/doi/10.5555/896920

[27] BELLMAN, R.: A markovian decision process. Journal of Mathematics and Mechanics **6**(5), 679–684 (1957), http://www.jstor.org/stable/24900506

[28] Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., de Oliveira Pinto, H.P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., Zhang, S.: Dota 2 with large scale deep reinforcement learning. CoRR **abs/1912.06680** (2019), http://arxiv.org/abs/1912.06680

[29] Bevy: Game engine. https://github.com/bevyengine/bevy, accessed on 2024-09-08

[30] Bishop, L., Eberly, D., Whitted, T., Finch, M., Shantz, M.: Designing a PC game engine. IEEE Computer Graphics and Applications **18**(1), 46–53 (1998), https://doi.org/10.1109/38.637270

[31] Bjarnason, A., Reynisson, J.M.: Deeper: adventures in procedural game development in Rust. Bachelor's thesis, Tölvunarfræðideild / Department of Computer Science (2021), https://hdl.handle.net/1946/39502

[32] Blommaert, J., Varis, P.: Conviviality and collectives on social media: Virality, memes, and new social structures. Multilingual Margins: A journal of multilingualism from the periphery **2**(1), 31–31 (2015)

[33] Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment **2008**(10), P10008 (2008), https://doi.org/10.1088/1742-5468/2008/10/P10008

[34] Boettiger, C.: An introduction to docker for reproducible research. ACM SIGOPS Operating Systems Review **49**(1), 71–79 (2015)

[35] Bone, N.: A survey of transfer learning methods for reinforcement learning (2008), https://cedar.wwu.edu/computerscience_stupubs/5, computer Science Graduate and Undergraduate Student Scholarship

[36] Bontrager, P., Khalifa, A., Anderson, D., Stephenson, M., Salge, C., Togelius, J.: "superstition" in the network: Deep reinforcement learning plays deceptive games. In: Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE). vol. 15, pp. 10–16. AAAI Press (2019), https://doi.org/10.1609/aiide.v15i1.5218

[37] Bormann, C., Hoffman, P.: Concise binary object representation (cbor). Tech. rep., RFC 7049, DOI 10.17487/RFC7049, October 2013,< https://www. rfc-editor. org … (2013)

[38] Bradford, G.: The History and Analysis of the Supposed Automation Chess Player of M. de Kempelen: Now Exhibiting in this Country, by Mr. Maelzel. Hilliard, Gray & Company (1826)

[39] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. arXiv preprint arXiv:1606.01540 (2016), https://arxiv.org/abs/1606.01540

[40] Brouwer, S., Akkermans, N., Hendriks, L., Van Uden, H., Wilms, V.: "lass frooby noo!" the interference of song lyrics and meaning on speech intelligibility. Journal of Experimental Psychology: Applied **28**(3), 576 (2022)

[41] Brown, J.A., de Araujo, L.J.P., Grichshenko, A.: Ai space invaders 2021 competition (2021)

[42] Brown, J.A., de Araujo, L.J.P., Grichshenko, A.: Snakes ai competition 2020 and 2021 report (2021)

[43] Brown, N., Sandholm, T.: Superhuman ai for multiplayer poker. Science **365**(6456), 885–890 (2019), https://doi.org/10.1126/science.aay2400

[44] Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in games **4**(1), 1–43 (2012), https://doi.org/10.1109/TCIAIG.2012.2186810

[45] Burgiel, H.: How to lose at Tetris. The Mathematical Gazette **81**(491), 194–200 (1997), https://doi.org/10.2307/3619195

[46] Bölling, L.: Librarian bully attack: Gaslighting state-of-the-art llms into obsolescence (2025-03-25), https://web.archive.org/web/20250331085213/https://humandataexperience.substack.com/p/librarian-bully-attack-gaslighting

[47] Campbell, J., Verbrugge, C.: Procedural generation of rollercoasters. In: Conference on Games. IEEE (2023), https://doi.org/10.1109/CoG57401.2023.10333156

[48] Cao, Y., Li, S., Liu, Y., Yan, Z., Dai, Y., Yu, P.S., Sun, L.: A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt. arXiv preprint arXiv:2303.04226 (2023)

[49] Carr, D.: Applying reinforcement learning to Tetris. Department of Computer Science Rhodes University (2005)

[50] Carroll, J.L., Seppi, K.: Task similarity measures for transfer in reinforcement learning task libraries. In: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005. vol. 2, pp. 803–808. IEEE (2005), https://doi.org/10.1109/IJCNN.2005.1555955

[51] Cavazza, M., Charles, F., Mead, S.J.: Characters in search of an author: Ai-based virtual storytelling. In: Virtual Storytelling Using Virtual Reality Technologies for Storytelling: International Conference ICVS 2001 Avignon, France, September 27–28, 2001 Proceedings. pp. 145–154. Springer (2001)

[52] Celiberto, L.A., Matsuura, J.P., López de Mántaras, R., Bianchi, R.: Using cases as heuristics in reinforcement learning: a transfer learning application pp. 1211–1217 (2011), https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-206

[53] Center, C.C.: Chatgpt-4o contains security bypass vulnerability through time and search functions called time bandit (2025-03-25), https://web.archive.org/web/20250206190954/https://www.kb.cert.org/vuls/id/733789

[54] Ceron, J.S.O., Castro, P.S.: Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In: International Conference on Machine Learning. pp. 1373–1383. PMLR (2021)

[55] Chao, C.: Blokus Game Solver. In: Computational Engineering Commons. California Polytechnic State University (2018), https://digitalcommons.calpoly.edu/cpesp/290

[56] Chaslot, G.M.J.B.C.: Monte-Carlo Tree Search. Ph.D. thesis (2010), https://doi.org/10.26481/dis.20100930gc

[57] Chen, D.Z., Siddiqui, Y., Lee, H., Tulyakov, S., Nießner, M.: Text2tex: Text-driven texture synthesis via diffusion models. In: IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023.

pp. 18512–18522. IEEE (2023), https://doi.org/10.1109/ICCV51070.2023.01701

[58] Cobbe, K., Hesse, C., Hilton, J., Schulman, J.: Leveraging procedural generation to benchmark reinforcement learning. In: International conference on machine learning. Proceedings of Machine Learning Research, vol. 119, pp. 2048–2056. PMLR (2020), http://proceedings.mlr.press/v119/cobbe20a.html

[59] Cobbe, K., Hesse, C., Hilton, J., Schulman, J.: Leveraging procedural generation to benchmark reinforcement learning **119**, 2048–2056 (2020), http://proceedings.mlr.press/v119/cobbe20a.html

[60] Cohen, B., Pietrzak, K.: The chia network blockchain (2019)

[61] garage contributors, T.: Garage: A toolkit for reproducible reinforcement learning research. https://github.com/rlworkgroup/garage (2019)

[62] Contributors, U.E.S.P.: Skyrim:generic dialogue - reactions to shouting, https://en.uesp.net/wiki/Skyrim:Generic_Dialogue#Reactions_to_Shouting, visited 04.05.2023

[63] Cook, M., Raad, A.: Hyperstate space graphs for automated game analysis. In: 2019 IEEE Conference on Games (CoG). IEEE (2019), https://doi.org/10.1109/CIG.2019.8848026

[64] Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo tree search. In: International conference on computers and games. pp. 72–83. Springer (2006), https://doi.org/10.1007/978-3-540-75538-8_7

[65] Crataco: Tavernai: Atmospheric adventure chat. https://github.com/TavernAI/TavernAI (2023)

[66] Demaine, E.D., Demaine, M.L.: Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. Graphs and Combinatorics **23**(1), 195–208 (2007), https://doi.org/10.1007/s00373-007-0713-4

[67] Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA. pp. 248–255. IEEE Computer Society (2009). doi: 10.1109/CVPR.2009.5206848, https://doi.org/10.1109/CVPR.2009.5206848

[68] Deshpande, A., Murahari, V., Rajpurohit, T., Kalyan, A., Narasimhan, K.: Toxicity in chatgpt: Analyzing persona-assigned language models. arXiv preprint arXiv:2304.05335 (2023)

[69] Deutsch, P., Gailly, J.L.: Zlib compressed data format specification version 3.3. Tech. rep., RFC 1950, May (1996)

[70] Dietterich, T.G.: Hierarchical reinforcement learning with the maxq value function decomposition. Journal of artificial intelligence research **13**, 227–303 (2000), https://doi.org/10.1613/jair.639

[71] Dimforge: Rapier physics engine. https://github.com/dimforge/rapier, accessed on 2024-09-08

[72] Donadigo, A.: Extracting inputs from replays (2021-12-15), https://donadigo.com/tminterface/input-extraction

[73] Donadigo, A.: Tmx replay investigation (2021-12-15), https://donadigo.com/tmx1

[74] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)

[75] Driess, D., Xia, F., Sajjadi, M.S.M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch, I., Florence, P.: Palm-e: An embodied multimodal language model. In: arXiv preprint arXiv:2303.03378 (2023)

[76] Duan, Y., Chen, X., Houthooft, R., Schulman, J., Abbeel, P.: Benchmarking deep reinforcement learning for continuous control. In: International conference on machine learning. pp. 1329–1338. PMLR (2016), http://proceedings.mlr.press/v48/duan16.html

[77] Duan, Y., Schulman, J., Chen, X., Bartlett, P.L., Sutskever, I., Abbeel, P.: Rl^2: Fast reinforcement learning via slow reinforcement learning (2016), http://arxiv.org/abs/1611.02779

[78] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Rozière, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C.C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E.M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G.L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I.A., Kloumann, I.M., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J.,

Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K.V., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., et al.: The llama 3 herd of models. CoRR **abs/2407.21783** (2024). doi: 10.48550/ARXIV.2407.21783, https://doi.org/10.48550/arXiv.2407.21783

[79] van Eck, N.J., van Wezel, M.: Application of reinforcement learning to the game of Othello. Computers & Operations Research **35**(6), 1999–2017 (2008), https://doi.org/10.1016/j.cor.2006.10.004

[80] Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K.O., Clune, J.: First return, then explore. Nature **590**(7847), 580–586 (2021), https://doi.org/10.1038/s41586-020-03157-9

[81] Edelkamp, S., Korf, R.E.: The branching factor of regular search spaces. In: Fifteenth National Conference on Artificial Intelligence (AAAI) and Tenth Innovative Applications of Artificial Intelligence Conferenc (IAAI). pp. 299–304 (1998), https://dl.acm.org/doi/10.5555/295240.295615

[82] Edwards, B.: You can now run a gpt-3-level ai model on your laptop, phone, and raspberry pi, https://arstechnica.com/information-technology/2023/03/you-can-now-run-a-gpt-3-level-ai-model-on-your-laptop-phone-and-raspberry-pi/, visited 28.03.2023

[83] Egenfeldt-Nielsen, S.: Keep the monkey rolling: eye-hand coordination in super monkey ball. In: Proceedings of DiGRA 2003 Conference: Level Up. DiGRA (2003), https://dl.digra.org/index.php/dl/article/view/40/

[84] Eisner, D.: Reproducibility of science: Fraud, impact factors and carelessness. Journal of molecular and cellular cardiology **114**, 364–368 (2018)

[85] Enzenberger, M., Müller, M.: A lock-free multithreaded Monte-Carlo tree search algorithm. In: Advances in Computer Games. Lecture Notes in Computer Science, vol. 6048, pp. 14–20. Springer (2009), https://doi.org/10.1007/978-3-642-12993-3_2

[86] Ermi, L., Mäyrä, F.: Analyzing immersion. Worlds in play: International perspectives on digital games research **21**, 37 (2007)

[87] European Organization For Nuclear Research, OpenAIRE: Zenodo (2013). doi: 10.25495/7GXK-RD71, https://www.zenodo.org/

[88] Fandom.com: History of world generation - minecraft wiki, https://minecraft.fandom.com/wiki/History_of_world_generation, visited 11.05.2023

[89] Farooq, S.S., Oh, I.S., Kim, M.J., Kim, K.J.: Starcraft ai competition report. AI Magazine **37**(2), 102–107 (2016)

[90] Ferigo, D., Traversaro, S., Metta, G., Pucci, D.: Gym-ignition: Reproducible robotic simulations for reinforcement learning. In: 2020 IEEE/SICE International Symposium on System Integration (SII). pp. 885–890. IEEE (2020)

[91] Floridi, L., Chiriatti, M.: GPT-3: its nature, scope, limits, and consequences. Minds Mach. **30**(4), 681–694 (2020), https://doi.org/10.1007/s11023-020-09548-1

[92] Font, J.M., Mahlmann, T.: Dota 2 bot competition. IEEE Transactions on Games **11**(3), 285–289 (2018)

[93] Gaina, R.D., Couëtoux, A., Soemers, D.J., Winands, M.H., Vodopivec, T., Kirchgeßner, F., Liu, J., Lucas, S.M., Perez-Liebana, D.: The 2016 two-player gvgai competition. IEEE Transactions on Games **10**(2), 209–220 (2017)

[94] Gaina, R.D., Lucas, S.M., Liebana, D.P.: Rolling horizon evolution enhancements in general video game playing. In: IEEE Conference on Computational Intelligence and Games, CIG 2017, New York, NY, USA, August 22-25, 2017. pp. 88–95. IEEE (2017). doi: 10.1109/CIG.2017.8080420, https://doi.org/10.1109/CIG.2017.8080420

[95] Galván-López, E., Li, R., Patsakis, C., Clarke, S., Cahill, V.: Heuristic-Based Multi-Agent Monte Carlo Tree Search. In: IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications. pp. 177–182. IEEE (2014), https://doi.org/10.1109/IISA.2014.6878747

[96] Games, E.: Replay system - unreal engine documentation (2021-12-15), https://docs.unrealengine.com/latest/INT/Engine/Replay/

[97] Gamrian, S., Goldberg, Y.: Transfer learning for related reinforcement learning tasks via image-to-image translation **97**, 2063–2072 (2019), http://proceedings.mlr.press/v97/gamrian19a.html

[98] Gao, C., Hayward, R., Müller, M.: Move prediction using deep convolutional neural networks in Hex. IEEE Transactions on Games **10**(4), 336–343 (2017), https://doi.org/10.1109/TG.2017.2785042

[99] Gemma Team: Gemma (2024), https://doi.org/10.34740/KAGGLE/M/3301

[100] Geng, X., Gudibande, A., Liu, H., Wallace, E., Abbeel, P., Levine, S., Song, D.: Koala: A dialogue model for academic research. Blog post (April 2023), https://bair.berkeley.edu/blog/2023/04/03/koala/

[101] Gerganov, G.: Llama.cpp: Port of facebook's llama model in c/c++, https://github.com/ggerganov/llama.cpp, visited 28.03.2023

[102] Gisslén, L., Eakins, A., Gordillo, C., Bergdahl, J., Tollmar, K.: Adversarial reinforcement learning for procedural content generation (2021), https://doi.org/10.1109/CoG52621.2021.9619053

[103] Glickman, M.E., Jones, A.C.: Rating the chess rating system. CHANCE-BERLIN THEN NEW YORK- **12**, 21–28 (1999)

[104] Green, M.C., Sergent, B., Shandilya, P., Kumar, V.: Evolutionarily-curated curriculum learning for deep reinforcement learning agents (2019), http://arxiv.org/abs/1901.05431

[105] Gregory, J.: Game engine architecture, Third Edition. AK Peters/CRC Press (2018), GameEngineArchitecture,ThirdEdition

[106] Grzes, M., Kudenko, D.: Plan-based reward shaping for reinforcement learning. In: 2008 4th International IEEE Conference Intelligent Systems. vol. 2, pp. 10–22. IEEE (2008), https://doi.org/10.1017/S0269888915000181

[107] Guez, A., Weber, T., Antonoglou, I., Simonyan, K., Vinyals, O., Wierstra, D., Munos, R., Silver, D.: Learning to search with mctsnets. In: International Conference on Machine Learning. pp. 1822–1831. PMLR (2018), http://proceedings.mlr.press/v80/guez18a.html

[108] Gupta, R., Kanade, A., Shevade, S.K.: Deep reinforcement learning for syntactic error repair in student programs. In: Conference on Artificial Intelligence. pp. 930–937. AAAI Press (2019), https://doi.org/10.1609/aaai.v33i01.3301930

[109] Guss, W.H., Castro, M.Y., Devlin, S., Houghton, B., Kuno, N.S., Loomis, C., Milani, S., Mohanty, S., Nakata, K., Salakhutdinov, R., Schulman, J., Shiroshita, S., Topin, N., Ummadisingu, A., Vinyals, O.: The minerl 2020 competition on sample efficient reinforcement learning using human priors (January 2021), https://www.microsoft.com/en-us/research/publication/the-minerl-2020-competition-on-sample-efficient-reinforcement-learning-using-human-priors/

[110] Guss, W.H., Codel, C., Hofmann, K., Houghton, B., Kuno, N.S., Milani, S., Mohanty, S., Liebana, D.P., Salakhutdinov, R., Topin, N., Veloso, M., Wang, P.: The minerl competition on sample efficient reinforcement learning using human priors. In: Thirty-third Conference on Neural Information Processing Systems (NeurIPS) Competition track (December 2019), https://www.microsoft.com/en-us/research/publication/the-minerl-competition-on-sample-efficient-reinforcement-learning-using-human-priors/

[111] Hämäläinen, M., Alnajjar, K.: Creative contextual dialog adaptation in an open world rpg. In: Proceedings of the 14th International Conference on the Foundations of Digital Games. pp. 1–7 (2019)

[112] Hamilton, N., Schlemmer, L., Menart, C., Waddington, C., Jenkins, T., Johnson, T.T.: Sonic to knuckles: evaluations on transfer reinforcement learning. In: Unmanned Systems Technology XXII. vol. 11425, p. 114250J. International

Society for Optics and Photonics (2020), https://ui.adsabs.harvard.edu/link_gateway/2020SPIE11425E..0JH/doi:10.1117/12.2559546

[113] Harsana, L.: How much ram does windows 11 use? [memory requirements], https://windowsreport.com/does-windows-11-consume-more-ram/, visited 28.03.2023

[114] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. In: Thirty-Second AAAI Conference on Artificial Intelligence. pp. 3207–3214. AAAI Press (2018), https://doi.org/10.1609/aaai.v32i1.11694

[115] Hendrikx, M., Meijer, S.A., Velden, J.V.D., Iosup, A.: Procedural content generation for games: A survey. Transactions on Multimedia Computing, Communications, and Applications **9** (2013), https://doi.org/10.1145/2422956.2422957

[116] Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., Botvinick, M., Blundell, C., Lerchner, A.: Darla: Improving zero-shot transfer in reinforcement learning. In: International Conference on Machine Learning. vol. 70, pp. 1480–1490. PMLR (2017), http://proceedings.mlr.press/v70/higgins17a.html

[117] Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y.: Stable Baselines. https://github.com/hill-a/stable-baselines (2018)

[118] Hoetzlein, R.: A procedural model for diverse tree species. In: Karpouzis, K., Gualeni, S., Pirker, J., Fowler, A. (eds.) FDG '22: Proceedings of the 17th International Conference on the Foundations of Digital Games, FDG22, Athens, Greece, September 5-8, 2022. pp. 72:1–72:8. ACM (2022). doi: 10.1145/3555858.3564251, https://doi.org/10.1145/3555858.3564251

[119] Hou, Y., Ong, Y.S., Feng, L., Zurada, J.M.: An evolutionary transfer reinforcement learning framework for multiagent systems. IEEE Transactions on Evolutionary Computation **21**(4), 601–615 (2017), https://doi.org/10.1109/TEVC.2017.2664665

[120] Hsu, F.h.: Computer chess, then and now: The Deep Blue saga. In: Proceedings of Technical Papers. International Symposium on VLSI Technology, Systems, and Applications. pp. 153–156 (1997), https://doi.org/10.1109/VTSA.1997.614748

[121] Hussain, T.S., Vidaver, G.: Flexible and purposeful NPC behaviors using real-time genetic control. In: International Conference on Evolutionary Computation(CEC). pp. 785–792. IEEE (2006), https://doi.org/10.1109/CEC.2006.1688391

[122] Inc., C.T.: Character.ai, https://character.ai, visited 28.03.2023

[123] Inc., S.I.E.: "race together," a new event with gran turismo sophy, available for a limited time only!, https://www.gran-turismo.com/us/gt7/news/00_5962264.html, visited 09.05.2023

[124] Islam, R., Henderson, P., Gomrokchi, M., Precup, D.: Reproducibility of benchmarked deep reinforcement learning tasks for continuous control (2017)

[125] Jacomy, M., Venturini, T., Heymann, S., Bastian, M.: Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. Plos One **9**(6), e98679 (2014), https://doi.org/10.1371/journal.pone.0098679

[126] Jahanshahi, A., Taram, M.K., Eskandari, N.: Blokus Duo game on FPGA. In: The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013). pp. 149–152. IEEE (2013), https://doi.org/10.1109/CADS.2013.6714256

[127] Jaschek, C., Beckmann, T., Garcia, J.A., Raffe, W.L.: Mysterious murder-mcts-driven murder mystery generation. In: 2019 IEEE Conference on Games (CoG). pp. 1–8. IEEE (2019)

[128] Jeurissen, D., Liebana, D.P., Gow, J., Çakmak, D., Kwan, J.: Playing nethack with llms: Potential & limitations as zero-shot agents. In: IEEE Conference on Games, CoG 2024, Milan, Italy, August 5-8, 2024. pp. 1–8. IEEE (2024), https://doi.org/10.1109/CoG60054.2024.10645630

[129] Jiang, Z., Earle, S., Green, M.C., Togelius, J.: Learning controllable 3d level generators. In: Foundations of Digital Games. ACM (2022), https://doi.org/10.1145/3555858.3563273

[130] Jordan, M.I., Mitchell, T.M.: Machine learning: Trends, perspectives, and prospects. Science **349**(6245), 255–260 (2015), https://doi.org/10.1126/science.aaa8415

[131] Joshi, G., Chowdhary, G.: Cross-domain transfer in reinforcement learning using target apprentice. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 7525–7532. IEEE (2018), https://doi.org/10.1109/ICRA.2018.8462977

[132] Joshi, S., Khan, M.S., Dafe, A., Singh, K., Zope, V., Jhamtani, T.: Fine tuning llms for low resource languages. In: 2024 5th International Conference on Image Processing and Capsule Networks (ICIPCN). pp. 511–519 (2024), https://doi.org/10.1109/ICIPCN63822.2024.00090

[133] Justesen, N., Moore, P.D., Uth, L.M., Togelius, J., Jakobsen, C., Risi, S.: Blood bowl: A new board game challenge and competition for ai. In: 2019 IEEE Conference on Games (COG). IEEE (2019)

[134] Justesen, N., Torrado, R.R., Bontrager, P., Khalifa, A., Togelius, J., Risi, S.: Illuminating generalization in deep reinforcement learning through procedural level generation. arXiv preprint arXiv:1806.10729 https://arxiv.org/abs/1806.10729

[135] Kacmarcik, G.: Using natural language to manage npc dialog. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 2, pp. 115–117 (2006)

[136] Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of artificial intelligence research **4**, 237–285 (1996), https://doi.org/10.1613/jair.301

[137] Kampert, D., Varbanescu, A.L., Müller-Brockhausen, M., Plaat, A.: Mimicking the human approach in the game of hive. In: IEEE Symposium Series on Computational Intelligence, SSCI 2021, Orlando, FL, USA, December 5-7, 2021. IEEE (2021), https://doi.org/10.1109/SSCI50451.2021.9659999

[138] Kaspar, M., Osorio, J.D.M., Bock, J.: Sim2real transfer for reinforcement learning without dynamics randomization pp. 4383–4388 (2020), https://doi.org/10.1109/IROS45743.2020.9341260

[139] Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., Scaramuzza, D.: Champion-level drone racing using deep reinforcement learning. Nature **620**(7976), 982–987 (2023), https://doi.org/10.1038/s41586-023-06419-4

[140] Khalifa, A., Bontrager, P., Earle, S., Togelius, J.: Pcgrl: Procedural content generation via reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 16, pp. 95–101 (2020), https://doi.org/10.1609/aiide.v16i1.7416

[141] Khetarpal, K., Ahmed, Z., Cianflone, A., Islam, R., Pineau, J.: Re-evaluate: Reproducibility in evaluating reinforcement learning algorithms (2018)

[142] Kim, M.J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E.P., Sanketi, P.R., Vuong, Q., Kollar, T., Burchfiel, B., Tedrake, R., Sadigh, D., Levine, S., Liang, P., Finn, C.: Openvla: An open-source vision-language-action model. In: Agrawal, P., Kroemer, O., Burgard, W. (eds.) Conference on Robot Learning, 6-9 November 2024, Munich, Germany. Proceedings of Machine Learning Research, vol. 270, pp. 2679–2713. PMLR (2024), https://proceedings.mlr.press/v270/kim25c.html

[143] Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: European conference on machine learning. pp. 282–293. Springer (2006), https://doi.org/10.1007/11871842_29

[144] Konidaris, G.D., Barto, A.G.: Building portable options: Skill transfer in reinforcement learning. In: International Joint Conference on Artificial Intelligence (IJCAI). vol. 7, pp. 895–900 (2007), http://ijcai.org/Proceedings/07/Papers/144.pdf

[145] Kroon, J.: Nemesis Narratives: The relationship between embedded and emergent narrative in Middle Earth: Shadow of Mordor. Master's thesis, Utrecht University (2016), https://studenttheses.uu.nl/handle/20.500.12932/23201

[146] KURT, F., ÖZGÖVDE, B.: Edge computing for computer games by offloading physics computation. Journal of Science Part A: Engineering and Innovation **10** (2023), https://doi.org/10.54287/gujsa.1338594

[147] Langley, P.: Elements of Machine Learning. Morgan Kaufmann Publishers Inc., 340 Pine Street, Sixth Floor, San Francisco, CA, USA (1995), https://dl.acm.org/doi/10.5555/212543

[148] Lawrence, S., Giles, C.L., Tsoi, A.C.: Lessons in neural network training: Overfitting may be harder than expected. In: AAAI/IAAI. pp. 540–545. Citeseer (1997), http://www.aaai.org/Library/AAAI/1997/aaai97-084.php

[149] Lazaric, A., Hal, H.I., Lazaric, A.: Transfer in reinforcement learning: A framework and a survey. In: Wiering, M.A., van Otterlo, M. (eds.) Reinforcement Learning, Adaptation, Learning, and Optimization, vol. 12, pp. 143–173. Springer (2012), https://doi.org/10.1007/978-3-642-27645-3_5

[150] Li, X., Cen, M., Xu, J., Zhang, H., Xu, X.S.: Improving feature extraction from histopathological images through A fine-tuning imagenet model. CoRR **abs/2201.00636** (2022), https://arxiv.org/abs/2201.00636

[151] Li, Y., Min, M.R., Shen, D., Carlson, D.E., Carin, L.: Video generation from text. In: McIlraith, S.A., Weinberger, K.Q. (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018. pp. 7065–7072. AAAI Press (2018). doi: 10.1609/AAAI.V32I1.12233, https://doi.org/10.1609/aaai.v32i1.12233

[152] Liebana, D.P., Samothrakis, S., Lucas, S.M., Rohlfshagen, P.: Rolling horizon evolution versus tree search for navigation in single-player real-time games. In: Blum, C., Alba, E. (eds.) Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013. pp. 351–358. ACM (2013). doi: 10.1145/2463372.2463413, https://doi.org/10.1145/2463372.2463413

[153] Lissa, P., Schukat, M., Barrett, E.: Transfer learning applied to reinforcement learning-based hvac control. SN Computer Science **1**(3), 1–12 (2020), https://doi.org/10.1007/s42979-020-00146-7

[154] Liu, C., Gao, C., Xia, X., Lo, D., Grundy, J., Yang, X.: On the replicability and reproducibility of deep learning in software engineering. arXiv preprint arXiv:2006.14244 (2020)

[155] López-Ibáñez, M., Branke, J., Paquete, L.: Reproducibility in evolutionary computation. ACM Transactions on Evolutionary Learning and Optimization **1**(4), 1–21 (2021)

[156] Lu, D.: I spent 11 years working on this line rider track. `https://web.archive.org/web/20231020213756/https://delu.medium.com/i-spent-11-years-working-on-this-line-rider-track-96742fc0b709`, accessed on 2024-09-08

[157] Maberry, K., Paustian, S., Bakir, S.: Using an artificial neural network to detect aim assistance in counter-strike: Global offensive `https://www.cs.nmt.edu/~kmaberry/ann_fps_cheater.pdf`

[158] Malisz, Z., Henter, G.E., Valentini-Botinhao, C., Watts, O., Beskow, J., Gustafson, J.: Modern speech synthesis for phonetic sciences: A discussion and an evaluation. In: International Congress of Phonetic Sciencesnces ICPhS 2019 5-9 August 2019, Melbourne, Australia Melbourne Convention and Exhibition Centre (2019)

[159] Markocki, M.: Reactive games as an example of extensive use of evocative narrative elements in digital games: cases of dwarf fortress and rimworld. Studia Humanistyczne AGH **20**(2), 71–83 (2021)

[160] Matsakis, N.D., Klock II, F.S.: The rust language. In: SIGAda annual conference on High integrity language technology (HILT). pp. 103–104. ACM (2014), `https://doi.org/10.1145/2663171.2663188`

[161] Matsubara, H., Iida, H., Grimbergen, R.: Chess, Shogi, Go, natural developments in game research. International Computer Games Associatoin (ICGA) **19**(2), 103–112 (1996), `https://doi.org/10.3233/ICG-1996-19208`

[162] Merriam-Webster: Chatter definition, `https://www.merriam-webster.com/dictionary/chatter`, visited 10.05.2023

[163] Michael Färber: The Microsoft Academic Knowledge Graph: A Linked Data Source with 8 Billion Triples of Scholarly Data. In: Proceedings of the 18th International Semantic Web Conference. pp. 113–129. ISWC'19 (2019). doi: 10.1007/978-3-030-30796-7_8, `https://doi.org/10.1007/978-3-030-30796-7_8`

[164] Miriti, K.: Integrating policy transfer, policy reuse and experience replay in speeding up reinforcement learning of the obstacle avoidance task. Ph.D. thesis, University of Nairobi (2014), `http://erepository.uonbi.ac.ke/handle/11295/90177`

[165] Mo, K., Li, S., Zhang, Y., Li, J., Yang, Q.: Personalizing a dialogue system with transfer reinforcement learning (2017)

[166] MobyGames: Line rider 2: Unbound. https://web.archive.org/web/20231203034915/https://www.mobygames.com/game/37725/line-rider-2-unbound/, accessed on 2024-09-08

[167] Mocanu, E., Nguyen, P.H., Kling, W.L., Gibescu, M.: Unsupervised energy prediction in a smart grid context using reinforcement cross-building transfer learning. Energy and Buildings **116**, 646–655 (2016), https://doi.org/10.1016/j.enbuild.2016.01.030

[168] Moerland, T.M.: The intersection of Planning and Learning. Ph.D. thesis, Technische Universiteit Delft (2021), http://hdl.handle.net/1842/879

[169] Moerland, T.M., Broekens, J., Jonker, C.M.: A framework for reinforcement learning and planning (2020), https://arxiv.org/abs/2006.15009

[170] Mole, B.: Top harvard cancer researchers accused of scientific fraud; 37 studies affected (2025-01-28), https://arstechnica.com/gadgets/2020/12/new-risc-v-cpu-claims-recordbreaking-performance-per-watt/

[171] Müller-Brockhausen, M., Barbero, G., Preuss, M.: Chatter generation through language models. In: IEEE Conference on Games, CoG 2023, Boston, MA, USA, August 21-24, 2023. IEEE (2023), https://doi.org/10.1109/CoG57401.2023.10333244

[172] Müller-Brockhausen, M., Khalifa, A., Preuss, M.: Scalable procedural content generation via transfer reinforcement learning. In: Data Science and Artificial Intelligence (DSAI). Springer (2024), https://doi.org/10.1007/978-981-97-9793-6

[173] Müller-Brockhausen, M., Plaat, A., Preuss, M.: Towards verifiable benchmarks for reinforcement learning. In: IEEE Conference on Games, CoG 2022, Beijing, China, August 21-24, 2022. pp. 159–166. IEEE (2022), https://doi.org/10.1109/CoG51982.2022.9893715

[174] Müller-Brockhausen, M., Plisnier, H.: Transferring while playing the rl agent. In: Demo at BNAIC/BeNeLearn 2022: Joint International Scientific Conferences on AI and Machine Learning (2022), https://bnaic2022.uantwerpen.be/wp-content/uploads/BNAICBeNeLearn_2022_submission_6564.pdf

[175] Müller-Brockhausen, M., Preuss, M., Plaat, A.: A new challenge: Approaching tetris link with AI. In: 2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021. IEEE (2021), https://doi.org/10.1109/CoG52621.2021.9619044

[176] Müller-Brockhausen, M., Preuss, M., Plaat, A.: Procedural content generation: Better benchmarks for transfer reinforcement learning. In: 2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021. IEEE (2021), https://doi.org/10.1109/CoG52621.2021.9619000

[177] Muralidharan, S., Ko, H.: An interplanetary file system (ipfs) based iot framework. In: 2019 IEEE International Conference on Consumer Electronics (ICCE). pp. 1–2 (2019). doi: 10.1109/ICCE.2019.8662002

[178] Muratore, F., Gienger, M., Peters, J.: Assessing transferability from simulation to reality for reinforcement learning. IEEE transactions on pattern analysis and machine intelligence **43**(4), 1172–1183 (2019), https://doi.org/10.1109/TPAMI.2019.2952353

[179] Murray, S.: High art/low life: The art of playing" grand theft auto". PAJ: A Journal of Performance and Art pp. 91–98 (2005)

[180] Müller-Brockhausen, M.: Interactive website to explore the data presented in this paper (2021), https://hizoul.github.io/trlsnap/

[181] Narasimhan, K., Barzilay, R., Jaakkola, T.: Grounding language for transfer in deep reinforcement learning. Journal of Artificial Intelligence Research **63**, 849–874 (2018), https://doi.org/10.1613/jair.1.11263

[182] Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M.E., Stone, P.: Curriculum learning for reinforcement learning domains: A framework and survey. Journal of Machine Learning Research **21**(181), 1–50 (2020), https://jmlr.org/papers/v21/20-212.html

[183] Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M.E., Stone, P.: Curriculum learning for reinforcement learning domains: A framework and survey. Journal of Machine Learning Research **21**(181), 1–50 (2020), https://jmlr.org/papers/v21/20-212.html

[184] Nasir, M.U., James, S., Togelius, J.: Gametraversalbenchmark: Evaluating planning abilities of large language models through traversing 2d game maps (2024), https://arxiv.org/abs/2410.07765

[185] National Academies of Sciences, Engineering, and Medicine and others: Reproducibility and replicability in science. National Academies Press (2019)

[186] Onishi, K., Murakami, N., Kitamura, Y., Kishino, F.: Modeling of trees with interactive l-system and 3d gestures. In: Ijspeert, A.J., Masuzawa, T., Kusumoto, S. (eds.) Biologically Inspired Approaches to Advanced Information Technology, Second International Workshop, BioADIT 2006, Osaka, Japan, January 26-27, 2006, Proceedings. Lecture Notes in Computer Science, vol. 3853, pp. 222–235. Springer (2006). doi: 10.1007/11613022_19, https://doi.org/10.1007/11613022_19

[187] OpenAI: Gpt-4 technical report (2023)

[188] Orenstein, P.: Does experiential learning improve learning outcomes in an undergraduate course in game theory–a preliminary analysis. Northeast Decision Sciences Institute (2014)

[189] Park, J.S., O'Brien, J.C., Cai, C.J., Morris, M.R., Liang, P., Bernstein, M.S.: Generative agents: Interactive simulacra of human behavior (2023)

[190] Partalas, I., Tsoumakas, G., Tzevanidis, K., Vlahavas, I.P.: Transferring experience in reinforcement learning through task decomposition. In: International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 1193–1194. IFAAMAS (2009), https://dl.acm.org/doi/10.5555/1558109.1558208

[191] PassMark: Passmark software - hardware survey - amount of ram installed, https://www.memorybenchmark.net/amount-of-ram-installed.html, visited 28.03.2023

[192] Patel, D., Ahmad, A.: Google "we have no moat, and neither does openai", https://www.semianalysis.com/p/google-we-have-no-moat-and-neither, visited 05.05.2023

[193] Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.M., Rothchild, D., So, D., Texier, M., Dean, J.: Carbon emissions and large neural network training. arXiv preprint arXiv:2104.10350 (2021), https://arxiv.org/abs/2104.10350

[194] Pereira, A., Proenca, A.: Prng-broker: A high-performance broker to supply parallel streams of pseudorandom numbers for large-scale simulations. In: Advances in Parallel & Distributed Processing, and Applications. pp. 167–183. Springer (2021), https://doi.org/10.1007/978-3-030-69984-0_14

[195] Perez-Liebana, D., Liu, J., Khalifa, A., Gaina, R.D., Togelius, J., Lucas, S.M.: General video game ai: A multitrack framework for evaluating agents, games, and content generation algorithms. IEEE Transactions on Games **11**(3), 195–214 (2019), https://doi.org/10.1109/TG.2019.2901021

[196] Perez-Liebana, D., Lucas, S.M., Gaina, R.D., Togelius, J., Khalifa, A., Liu, J.: General Video Game Artificial Intelligence, vol. 3. Morgan & Claypool Publishers (2019), https://doi.org/10.2200/S00944ED1V01Y201908GCI005

[197] Pérez-Liébana, D., Samothrakis, S., Togelius, J., Schaul, T., Lucas, S.M.: Analyzing the robustness of general video game playing agents. In: 2016 IEEE Conference on Computational Intelligence and Games (CIG). pp. 1–8. IEEE (2016)

[198] Philidor, F.D.: Analysis of the Game of Chess. P. Elmsly (1790)

[199] Pineau, J.: Reproducibility, Reusability, and Robustness in Deep Reinforcement Learning (2020-01-18), https://www.youtube.com/watch?v=Vh4H0gOwdIg, international Conference on Learning Representations (ICLR)

[200] Pineau, J., Sinha, K., Fried, G., Ke, R.N., Larochelle, H.: Reproducibility Challenge (2020-01-18), https://www.cs.mcgill.ca/~jpineau/ICLR2018-ReproducibilityChallenge.html, international Conference on Learning Representations (ICLR)

[201] Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché Buc, F., Fox, E., Larochelle, H.: Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). Journal of Machine Learning Research **22** (2021)

[202] Plaat, A.: Research, Re: Search and Re-Search. Ph.D. thesis, Erasmus University Rotterdam (1996), https://arxiv.org/abs/2403.13705

[203] Plaat, A.: Learning to Play: Reinforcement Learning and Games. Springer Nature (2020)

[204] Plaat, A.: Deep Reinforcement Learning. Springer Nature (2021)

[205] Plaat, A., Schaeffer, J., Pijls, W., De Bruin, A.: Best-first fixed-depth minimax algorithms. Artificial Intelligence **87**(1-2), 255–293 (1996), https://doi.org/10.1016/0004-3702(95)00126-3

[206] Plisnier, H., Steckelmacher, D., Roijers, D.M., Nowé, A.: Transfer reinforcement learning across environment dynamics with multiple advisors. In: Proceedings of the 31st Benelux Conference on Artificial Intelligence (BNAIC 2019) and the 28th Belgian Dutch Conference on Machine Learning (Benelearn 2019), Brussels, Belgium, November 6-8, 2019. CEUR Workshop Proceedings, vol. 2491. CEUR-WS.org (2019), https://ceur-ws.org/Vol-2491/paper11.pdf

[207] Post, I., Ye, Y.: The simplex method is strongly polynomial for deterministic markov decision processes. Mathematics of Operations Research **40**(4), 859–868 (2015)

[208] Preuss, M.: Multimodal Optimization by Means of Evolutionary Algorithms. Natural Computing Series, Springer (2015), https://doi.org/10.1007/978-3-319-07407-8

[209] Prins, V.L.: Dungeons & Firearms: AI-Directing Action Intensity of Procedural Levels. Master's thesis, Leiden University, The Netherlands (2016), https://theses.liacs.nl/2822

[210] Prins, V.L., Prins, J., Preuss, M., Maureira, M.A.G.: Storyworld: Procedural quest generation rooted in variety & believability. In: Lopes, P., Luz, F., Liapis, A., Engström, H. (eds.) Proceedings of the 18th International Conference on the Foundations of Digital Games, FDG 2023, Lisbon, Portugal, April 12-14, 2023.

pp. 44:1–44:4. ACM (2023). doi: 10.1145/3582437.3587181, https://doi.org/10.1145/3582437.3587181

[211] Purdy, K.: Fallout 4 mod uses voice ai to add sensible reactions, more rpg-like choices, https://arstechnica.com/gaming/2023/03/ai-voice-generation-helps-mod-fallout-4s-narrow-dialogue-choices/, visited 28.03.2023

[212] Pyo3: Rust bindings for python. https://github.com/PyO3/pyo3, accessed on 2024-09-08

[213] PyTorch: Reproducibility - pytorch documentation (2021-12-15), https://pytorch.org/docs/stable/notes/randomness.html

[214] Qasemi, E., Samadi, A., Shadmehr, M.H., Azizian, B., Mozaffari, S., Shirian, A., Alizadeh, B.: Highly scalable, shared-memory, Monte-Carlo tree search based Blokus Duo Solver on FPGA. In: 2014 International Conference on Field-Programmable Technology (FPT). pp. 370–373. IEEE (2014), https://doi.org/10.1109/FPT.2014.7082823

[215] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research **22**(268), 1–8 (2021), http://jmlr.org/papers/v22/20-1364.html

[216] Ramé, A., Ahuja, K., Zhang, J., Cord, M., Bottou, L., Lopez-Paz, D.: Model ratatouille: Recycling diverse models for out-of-distribution generalization. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA. Proceedings of Machine Learning Research, vol. 202, pp. 28656–28679. PMLR (2023), https://proceedings.mlr.press/v202/rame23a.html

[217] Ray, S.: A quick review of machine learning algorithms. In: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). pp. 35–39 (2019), https://doi.org/10.1109/COMITCon.2019.8862451

[218] Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S.G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J.T., et al.: A generalist agent. arXiv preprint arXiv:2205.06175 (2022)

[219] Rimmel, A., Teytaud, F., Teytaud, O.: Biasing Monte-Carlo simulations through RAVE values. In: International Conference on Computers and Games. pp. 59–68. Springer (2010), https://doi.org/10.1007/978-3-642-17928-0_6

[220] Risi, S., Togelius, J.: Increasing generality in machine learning through procedural content generation. Nature Machine Intelligence **2**(8), 428–436 (2020), https://doi.org/10.1038/s42256-020-0208-z

[221] Romanycia, M.H., Pelletier, F.J.: What is a heuristic? Computational Intelligence **1**(1), 47–58 (1985), https://doi.org/10.1111/j.1467-8640.1985.tb00058.x

[222] Ruijl, B., Vermaseren, J., Plaat, A., Herik, J.v.d.: Combining simulated annealing and Monte Carlo tree search for expression simplification. arXiv preprint arXiv:1312.0841 (2013), https://arxiv.org/abs/1312.0841

[223] Rupp, F., Eberhardinger, M., Eckert, K.: Balancing of competitive two-player game levels with reinforcement learning. In: Conference on Games (COG). IEEE (2023), https://doi.org/10.1109/CoG57401.2023.10333248

[224] Russell, S.J., Norvig, P.: Artificial intelligence: a modern approach, 4th edition. Malaysia; Pearson Education Limited (2020)

[225] Salge, C., Green, M.C., Canaan, R., Togelius, J.: Generative design in minecraft (GDMC): settlement generation competition. In: Conference on the Foundations of Digital Games. ACM (2018)

[226] Sampaio, P., Baffa, A., Feijó, B., Lana, M.: A fast approach for automatic generation of populated maps with seed and difficulty control. In: 16th Brazilian Symposium on Computer Games and Digital Entertainment, SBGames 2017, Curitiba, Brazil, November 2-4, 2017. pp. 10–18. IEEE Computer Society (2017), https://doi.org/10.1109/SBGames.2017.00010

[227] Santiago III, J.M., Parayno, R.L., Deja, J.A., Samson, B.P.V.: Rolling the dice: Imagining generative ai as a dungeons & dragons storytelling companion. arXiv preprint arXiv:2304.01860 (2023)

[228] Sarkar, A., Guzdial, M., Snodgrass, S., Summerville, A., Machado, T., Smith, G.: Procedural content generation via knowledge transformation (pcg-kt). Transaction on Games **16**, 36–50 (2024), https://doi.org/10.1109/TG.2023.3270422

[229] Sarker, A., Gonzalez, G.: A corpus for mining drug-related knowledge from twitter chatter: Language models and their utilities. Data in brief **10**, 122–131 (2017)

[230] Satyanarayan, A., Russell, R., Hoffswell, J., Heer, J.: Reactive vega: A streaming dataflow architecture for declarative interactive visualization. IEEE Transactions on Visualization & Computer Graphics (Proc. InfoVis) (2016), https://doi.org/10.1109/TVCG.2015.2467091

[231] Schaeffer, J., van den Herik, J., Hsu, T.s.: Games, computers and artificial intelligence. Chips Challenging Champions: games, computer and Artificial Intelligence **134** (2002), https://doi.org/10.1016/S0004-3702(01)00165-5

[232] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al.: Mastering atari, go, chess and shogi by planning with a learned model. Nature **588**(7839), 604–609 (2020), https://doi.org/10.1038/s41586-020-03051-4

[233] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al.: Mastering atari, go, chess and shogi by planning with a learned model. Nature **588**(7839), 604–609 (2020), https://doi.org/10.1038/s41586-020-03051-4

[234] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017), http://arxiv.org/abs/1707.06347

[235] Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O.: Green ai. Communications of the ACM **63**(12), 54–63 (2020)

[236] Serin, K.: The sims rival life by you reveals its open world, dialogue trees, and third person mode, https://www.rockpapershotgun.com/the-sims-rival-life-by-you-reveals-its-open-world-dialogue-trees-and-third-person-mode, visited 28.03.2023

[237] Shah, D., Osinski, B., Ichter, B., Levine, S.: Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In: Liu, K., Kulic, D., Ichnowski, J. (eds.) Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand. Proceedings of Machine Learning Research, vol. 205, pp. 492–504. PMLR (2022), https://proceedings.mlr.press/v205/shah23b.html

[238] Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: Field and Service Robotics (2017), https://arxiv.org/abs/1705.05065

[239] Shao, K., Tang, Z., Zhu, Y., Li, N., Zhao, D.: A survey of deep reinforcement learning in video games. CoRR **abs/1912.10944** (2019), http://arxiv.org/abs/1912.10944

[240] Shao, K., Zhu, Y., Zhao, D.: Starcraft micromanagement with reinforcement learning and curriculum transfer learning. IEEE Transactions on Emerging Topics in Computational Intelligence **3**(1), 73–84 (2018), https://doi.org/10.1109/TETCI.2018.2823329

[241] Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E., Schärli, N., Zhou, D.: Large language models can be easily distracted by irrelevant context. arXiv preprint arXiv:2302.00093 (2023)

[242] Shu, T., Liu, J., Yannakakis, G.N.: Experience-driven PCG via reinforcement learning: A super mario bros study. In: Conference on Games. IEEE (2021), https://doi.org/10.1109/CoG52621.2021.9619124

[243] Silva, F., Costa, A.: A survey on transfer learning for multiagent reinforcement learning systems. Journal of Artificial Intelligence Research **64**, 645–703 (2019), https://doi.org/10.1613/jair.1.11396

[244] Silva, F.L.d., Costa, A.H.R.: Object-oriented curriculum generation for reinforcement learning. In: Conference on Autonomous Agents and MultiAgent Systems. ACM (2018)

[245] Silva, R., Vasco, M., Melo, F.S., Paiva, A., Veloso, M.: Playing games in the dark: An approach for cross-modality transfer in reinforcement learning. In: Seghrouchni, A.E.F., Sukthankar, G., An, B., Yorke-Smith, N. (eds.) International Conference on Autonomous Agents and Multiagent Systems, AAMAS. International Foundation for Autonomous Agents and Multiagent Systems (2020), https://dl.acm.org/doi/10.5555/3398761.3398907

[246] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016), https://doi.org/10.1038/nature16961

[247] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018), https://doi.org/10.1126/science.aar6404

[248] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of go without human knowledge. Nature **550**(7676), 354–359 (2017), https://doi.org/10.1038/nature24270

[249] Singh, B., Kumar, R., Singh, V.P.: Reinforcement learning in robotic applications: a comprehensive survey. Artif. Intell. Rev. **55**(2), 945–990 (2022). doi: 10.1007/S10462-021-09997-9, https://doi.org/10.1007/s10462-021-09997-9

[250] Sinha, A., Shen, Z., Song, Y., Ma, H., Eide, D., Hsu, B.J.P., Wang, K.: An overview of microsoft academic service (mas) and applications. In: Proceedings of the 24th International Conference on World Wide Web. p. 243–246. WWW '15 Companion, Association for Computing Machinery, New York, NY, USA (2015). doi: 10.1145/2740908.2742839, https://doi.org/10.1145/2740908.2742839

[251] Smith, G.: Understanding procedural content generation: a design-centric analysis of the role of PCG in games. In: Jones, M., Palanque, P.A., Schmidt, A., Grossman, T. (eds.) CHI Conference on Human Factors in Computing Systems, CHI'14, Toronto, ON, Canada - April 26 - May 01, 2014. pp. 917–926. ACM (2014). doi: 10.1145/2556288.2557341, https://doi.org/10.1145/2556288.2557341

[252] Soni, V., Singh, S.: Using homomorphisms to transfer options across continuous reinforcement learning domains. In: AAAI Conference on Artificial Intelligence. vol. 6, pp. 494–499 (2006), http://www.aaai.org/Library/AAAI/2006/aaai06-079.php

[253] Sorokin, A.Y., Burtsev, M.S.: Episodic memory transfer for multi-task reinforcement learning. Biologically inspired cognitive architectures **26**, 91–95 (2018), https://doi.org/10.1016/j.bica.2018.09.003

[254] Sprague, T.: On the different possible non-linear arrangements of eight men on a Chess-board. Proceedings of the Edinburgh Mathematical Society **8**, 30–43 (1889), https://doi.org/10.1017/S0013091500030522

[255] van der Staaij, A., Prins, J., Prins, V.L., Poelsma, J., Smit, T., Müller-Brockhausen, M., Preuss, M.: Believable minecraft settlements by means of decentralised iterative planning. In: IEEE Conference on Games, CoG 2023, Boston, MA, USA, August 21-24, 2023. IEEE (2023), https://doi.org/10.1109/CoG57401.2023.10333146

[256] Stanley, K.O., D'Ambrosio, D.B., Gauci, J.: A hypercube-based encoding for evolving large-scale neural networks. Artificial life **15**(2), 185–212 (2009), https://doi.org/10.1162/artl.2009.15.2.15202

[257] van Stegeren, J., Myśliwiec, J.: Fine-tuning gpt-2 on annotated rpg quests for npc dialogue generation. In: Proceedings of the 16th International Conference on the Foundations of Digital Games. pp. 1–8 (2021)

[258] Strathern, M.: 'improving ratings': audit in the british university system. European Review **5**(3), 305–321 (1997)

[259] Strong, C., Mateas, M.: Talking with npcs: Towards dynamic generation of discourse structures. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 4, pp. 114–119 (2008)

[260] Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in nlp. In: Association for Computational Linguistics (ACL). pp. 3645–3650 (2019), https://doi.org/10.18653/v1/p19-1355

[261] Sudhakaran, S., González-Duque, M., Freiberger, M., Glanois, C., Najarro, E., Risi, S.: Mariogpt: Open-ended text2level generation through large language models. In: Neural Information Processing Systems (NIPS). Neurips (2023), https://openreview.net/forum?id=aa8KsqfTPa

[262] Sutton, R.S.: Dyna, an integrated architecture for learning, planning, and reacting. ACM Sigart Bulletin **2**(4), 160–163 (1991), https://doi.org/10.1145/122344.122377

[263] Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. Second Edition. MIT press (2018)

[264] Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., Hashimoto, T.B.: Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca (2023)

[265] Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research **10**(7), 1633–1685 (2009), https://dl.acm.org/doi/10.5555/1577069.1755839

[266] Taylor, M.E., Stone, P.: An introduction to intertask transfer for reinforcement learning. AI Magazine **32**(1), 15–15 (2011), https://doi.org/10.1609/aimag.v32i1.2329

[267] Tesauro, G.: Neurogammon: A neural network backgammon learning program. Heuristic Programming in Artificial Intelligence: The First Computer Olympiad, Chichester, England pp. 33–39 (1990), https://doi.org/10.1109/IJCNN.1990.137821

[268] Thrun, S., Schwartz, A.: Finding structure in reinforcement learning. Advances in neural information processing systems (NeurIPS) pp. 385–392 (1994), https://proceedings.neurips.cc/paper_files/paper/1994/hash/7ce3284b743aefde80ffd9aec500e085-Abstract.html

[269] Todd, G., Earle, S., Nasir, M.U., Green, M.C., Togelius, J.: Level generation through large language models. In: Foundations of Digital Games. ACM (2023), https://doi.org/10.1145/3582437.3587211

[270] Todorov, E., Erez, T., Tassa, Y.: Mujoco: A physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 5026–5033. IEEE (2012), https://doi.org/10.1109/IROS.2012.6386109

[271] Togelius, J., Yannakakis, G.N., Stanley, K.O., Browne, C.: Search-based procedural content generation: A taxonomy and survey. Transactions on Computational Intelligence and AI in Games **3** (2011)

[272] Torrado, R.R., Bontrager, P., Togelius, J., Liu, J., Pérez-Liébana, D.: Deep reinforcement learning for general video game AI. In: 2018 IEEE Conference on Computational Intelligence and Games, CIG 2018, Maastricht, The Netherlands, August 14-17, 2018. pp. 1–8. IEEE (2018). doi: 10.1109/CIG.2018.8490422, https://doi.org/10.1109/CIG.2018.8490422

[273] Torrey, L., Shavlik, J., Walker, T., Maclin, R.: Skill acquisition via transfer learning and advice taking. In: European Conference on Machine Learning. Lecture Notes in Computer Science, vol. 4212, pp. 425–436. Springer (2006), https://doi.org/10.1007/11871842_41

[274] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: Llama: Open and efficient foundation language models (2023)

[275] Traoré, R., Caselles-Dupré, H., Lesort, T., Sun, T., Díaz-Rodríguez, N., Filliat, D.: Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer (2019), http://arxiv.org/abs/1906.04452

[276] Valve: Steam hardware and software survey february 2023, https://store.steampowered.com/hwsurvey, visited 28.03.2023

[277] Van Rijn, J.N., Bischl, B., Torgo, L., Gao, B., Umaashankar, V., Fischer, S., Winter, P., Wiswedel, B., Berthold, M.R., Vanschoren, J.: Openml: A collaborative science platform. In: Joint european conference on machine learning and knowledge discovery in databases. Lecture Notes in Computer Science, vol. 8190, pp. 645–649. Springer (2013), https://doi.org/10.1007/978-3-642-40994-3_46

[278] Van Rijswijck, J.: Search and evaluation in Hex. University of Alberta (2002), https://web.archive.org/web/20210623031417/https://www.cs.cornell.edu/~adith/docs/y_hex.pdf

[279] Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J.P., Jaderberg, M., Vezhnevets, A.S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T.L., Gülçehre, Ç., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T.P., Kavukcuoglu, K., Hassabis, D., Apps, C., Silver, D.: Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (2019), https://doi.org/10.1038/s41586-019-1724-z

[280] Vithayathil Varghese, N., Mahmoud, Q.H.: A survey of multi-task deep reinforcement learning. Electronics **9**(9), 1363 (2020), https://doi.org/10.3390/electronics9091363

[281] Volz, V., Naujoks, B.: Towards game-playing ai benchmarks via performance reporting standards. IEEE Conference on Games (CoG) pp. 764–771 (2020), https://doi.org/10.1109/CoG47356.2020.9231705

[282] Volz, V., Schrum, J., Liu, J., Lucas, S.M., Smith, A.M., Risi, S.: Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In: Genetic and Evolutionary Computation Conference. ACM (2018)

[283] Wang, J.X., King, M., Porcel, N., Kurth-Nelson, Z., Zhu, T., Deck, C., Choy, P., Cassin, M., Reynolds, M., Song, F., Buttimore, G., Reichert, D.P., Rabinowitz, N., Matthey, L., Hassabis, D., Lerchner, A., Botvinick, M.: Alchemy: A structured task distribution for meta-reinforcement learning (2021), https://arxiv.org/abs/2102.02926

[284] Wang, N., Zhang, Y., Li, Z., Fu, Y., Yu, H., Liu, W., Xue, X., Jiang, Y.: Pixel2mesh: 3d mesh model generation via image guided deformation. IEEE

Trans. Pattern Anal. Mach. Intell. **43**(10), 3600–3613 (2021). doi: 10.1109/TPAMI.2020.2984232, https://doi.org/10.1109/TPAMI.2020.2984232

[285] Wang, T.T., Gleave, A., Tseng, T., Pelrine, K., Belrose, N., Miller, J., Dennis, M.D., Duan, Y., Pogrebniak, V., Levine, S., Russell, S.: Adversarial policies beat superhuman go ais. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research, vol. 202, pp. 35655–35739. PMLR (2023), https://proceedings.mlr.press/v202/wang23g.html

[286] Weiss, K., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. Journal of Big data **3** (2016), https://doi.org/10.1186/s40537-016-0043-6

[287] Weng, R.C., Lin, C.J.: A Bayesian Approximation Method for Online Ranking. Journal of Machine Learning Research **12**(Jan), 267–300 (2011), https://dl.acm.org/doi/10.5555/1953048.1953057

[288] Werning, M.: Predicting the past from minimal traces: Episodic memory and its distinction from imagination and preservation. Review of philosophy and psychology **11**(2), 301–333 (2020)

[289] Wikipedia: First-move advantage in chess (2019-08-18), https://web.archive.org/web/20190807121829/https://en.wikipedia.org/wiki/First-move_advantage_in_chess

[290] Wikipedia: Tetromino (2019-09-09), https://en.wikipedia.org/wiki/Tetromino

[291] Xenopoulos, P., Doraiswamy, H., Silva, C.: Valuing player actions in counter-strike: Global offensive. In: IEEE International Conference on Big Data (Big Data). pp. 1283–1292. IEEE (2020)

[292] Xiao, I.: A distributed reinforcement learning solution with knowledge transfer capability for a bike rebalancing problem (2018), http://arxiv.org/abs/1810.04058

[293] Xie, S., Galashov, A., Liu, S., Hou, S., Pascanu, R., Heess, N., Teh, Y.W.: Transferring task goals via hierarchical reinforcement learning (2018), https://openreview.net/forum?id=S1Y6TtJvG

[294] Yang, J., Petersen, B., Zha, H., Faissol, D.: Single episode policy transfer in reinforcement learning (2020), https://openreview.net/forum?id=rJeQoCNYDS

[295] Yang, T., Hao, J., Meng, Z., Zhang, Z., Hu, Y., Cheng, Y., Fan, C., Wang, W., Liu, W., Wang, Z., et al.: Efficient deep reinforcement learning via adaptive policy transfer pp. 3094–3100 (2020), https://doi.org/10.24963/ijcai.2020/428

[296] Yannakakis, G.N.: AI in Computer Games: Generating Interesting Interactive Opponents by the use of Evolutionary Computation. Ph.D. thesis, University of Edinburgh (2005), http://hdl.handle.net/1842/879

[297] Yin, H., Pan, S.: Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In: AAAI Conference on Artificial Intelligence. vol. 31, pp. 1640–1646 (2017), https://doi.org/10.1609/aaai.v31i1.10733

[298] Yoon, I., Anwar, M.A., Joshi, R.V., Rakshit, T., Raychowdhury, A.: Hierarchical memory system with stt-mram and sram to support transfer and real-time reinforcement learning in autonomous drones. IEEE Journal on Emerging and Selected Topics in Circuits and Systems **9**(3), 485–497 (2019), https://doi.org/10.1109/JETCAS.2019.2932285

[299] Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., Levine, S.: Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In: Conference on Robot Learning. Proceedings of Machine Learning Research, vol. 100, pp. 1094–1100. PMLR (2020), http://proceedings.mlr.press/v100/yu20a.html

[300] Zakaria, Y., Fayek, M., Hadhoud, M.: Start small: Training controllable game level generators without training data by learning at multiple sizes. Alexandria Engineering Journal **72**, 479–494 (2023), https://doi.org/10.1016/j.aej.2023.04.019

[301] Zakaria, Y., Fayek, M.B., Hadhoud, M.: Procedural level generation for sokoban via deep learning: An experimental study. Transaction on Games **15**, 108–120 (2023), https://doi.org/10.1109/TG.2022.3175795

[302] Zhou, W., Zhang, S., Poon, H., Chen, M.: Context-faithful prompting for large language models. arXiv preprint arXiv:2303.11315 (2023)

[303] Zhu, X.: Behavior tree design of intelligent behavior of non-player character (npc) based on unity3d. Journal of Intelligent & Fuzzy Systems **37**(5), 6071–6079 (2019), https://doi.org/10.3233/JIFS-179190

[304] Zhu, Z., Lin, K., Zhou, J.: Transfer learning in deep reinforcement learning: A survey (2020), https://arxiv.org/abs/2009.07888

[305] Zittrain, J., Albert, K., Lessig, L.: Perma: Scoping and addressing the problem of link and reference rot in legal citations. Legal Information Management (LIM) **14**, 88 (2014)

Samenvatting

In dit proefschrift (getiteld: *Exploring the Synergies between Transfer in Reinforcement Learning and Procedural Content Generation*) is onderzocht hoe de twee in de titel genoemde onderzoeksvelden *Transfer in Reinforcement Learning (TRL)* en *Procedural Content Generation (PCG)* synergetisch zouden kunnen samenwerken.

Allereerst vergeleken wij verschillende AI-algoritmes in het bordspel *Tetris Link*. We vergeleken (1) een heuristische benadering, (2) Monte Carlo Tree Search (MCTS) en (3) Reinforcement Learning (RL).

Voor beide laatste algoritmes (MCTS & RL) is het aantal mogelijke zetten één van de indicaties hoe moeilijk een spel is. In Tetris Link ligt het aantal mogelijke zetten tussen die van schaken en go. Verder heeft Tetris Link de eigenschap dat het moeiljik is terug te komen van een achterstand. Samen maken deze eigenschappen dit spel een interessant onderzoeksonderwerp. Onze verwachting was dat MCTS hier het beste zou presteren, vanwege de snelheid van onze implementatie. Toch was MCTS zwakker dan RL en de heuristische benadering. Ook RL is door onze heuristiek verslagen. Hetgeen verbazend is, gezien bekende voorbeelden in de literatuur waarbij RL uitstekend presteert. Dit heeft ons ertoe aangezet de reden verder te onderzoeken, waarbij we ons op transfer in RL hebben gericht.

Om een idee van het onderzoeksveld te krijgen hebben we een literatuurstudie geschreven. De belangrijkste conclusie van onze survey is dat TRL zich voornamelijk richt op een statische hoeveelheid van taken die niet meer veranderen. Daarom zijn we ons gaan richten op PCG, waarin gemakkelijk dynamische veranderingen kunnen worden onderzocht.

Zodra we begonnen met experimenteren met TRL stuitten we snel op het probleem van reproduceerbaarheid. Het trainen van RL-policies gebaseerd op neurale netwerken is lastig reproduceerbaar. Reproduceerbaarheid is een belangrijke eigenschap in machine learning, vooral in de context van TRL waar elke taakwisseling een keten van niet-reproduceerbaarheid creëert. Hoewel we geen algemene oplossing voor het probleem gevonden hebben, hebben we wel een verbetering voorgesteld: Replay traces. De meeste RL omgevingen zijn deterministisch en dus reproduceerbaar als dezelfde actie-sequentie met dezelfde initiële random seed worden uitgevoerd. In deze benadering wordt een set van verschillende random seeds gebruikt om diverse maar reproduceerbare

playouts te garanderen. Figuren en tabellen in wetenschappelijke publicaties zijn met replay traces vergelijkbaar en verifieerbaar. Voortbouwend op de basis van replay traces, onderzochten wij hoe TRL en PCG elkaar kunnen versterken. Daarvoor hebben we het spel *Linerider* veranderd van 2D naar 3D. Het spel zelf is puur creatief en heeft geen vast doel. Dit heeft als gevolg dat er allerlei creaties kunnen worden gemaakt zoals tracks die synchroon op muziek worden afgespeeld. Na het specificeren van de track zorgen krachten zoals zwaartekracht voor de beweging van de rijder. Om dit spel toegankelijk voor RL te maken hebben we er een omgeving voor geprogrammeerd. Daarvoor moesten we een doel voor de algoritme definiëren. Het interessante aan deze omgeving is dat de op te lossen taak zelf een toepassing van PCG is. Initiële experimenten toonden aan dat de afstand tussen de startpositie en het doel klein moet zijn om RL de taak te laten leren. Door kleine incrementele verhogingen in afstand waren we in staat om een RL policy te trainen die ook geschikt was voor tracks van grotere afstanden. Tenslotte hebben we in de lijn van PCG ook Large Language Models (LLMs) onderzocht die inhoud voor games genereren. Gerelateerd onderzoek focust vooral op dynamische dialogen waarin de gebruiker alles kan invoeren wat hij wil, wat redelijk goed werkt. Gebruikersinvoer heeft echter het nadeel dat LLM's misbruikt kunnen worden om veiligheidssystemen te kraken en ze te laten ontsporen om toxische inhoud uit te spuwen. Derhalve hebben wij het concept van Chatter als alternatief bedacht, waarbij NPC's alleen korte zinnen genereren op basis van vooraf gedefinieerde aanwijzingen van ontwikkelaars. We laten empirisch zien dat dit goed werkt. Bovendien laten we in dit werk zien dat de meeste gaming hardware die consumenten in huis hebben krachtig genoeg is om een lokale LLM te laten draaien naast een AAA-game zoals Cyberpunk. Dit proefschrift laat zien dat PCG en TRL elkaar kunnen aanvullen. PCG kan worden toegepast om veel verschillende taken te genereren die helpen bij het trainen van generaliserende RL policies. En TRL kan helpen om betere resultaten te bereiken bij het toepassen van RL, bijvoorbeeld op een PCG-probleem zoals Linerider.

Summary

In this dissertation (titled: *Exploring the Synergies between Transfer in Reinforcement Learning and Procedural Content Generation*) we explore how the two research fields named in the title, namely *Transfer in Reinforcement Learning (TRL)* and *Procedural Content Generation (PCG)* can synergize together.

Our journey began with a comparison of different AI algorithms on the board game *Tetris Link*. We compared a heuristic, Monte Carlo Tree Search (MCTS), and Reinforcement Learning (RL). For both MCTS & RL, the number of possible actions per turn indicates how difficult a game is. In Tetris Link, the number of possible actions per turn is between chess and go. This combined with difficult game mechanics where most actions result in negative points that are hard to recover from make it an interesting game for research. We expected MCTS to perform best here, especially due to our implementation of the game in Rust with a focus on speed, and the fact that our MCTS implementation was heavily parallelized and got a reasonable thinking time. Especially given the thinking time we expected different results, because in principle MCTS will always find the optimal solution, provided that it has simulated every possible outcome. Nevertheless, the heuristic outperformed both RL and MCTS, and even RL outperformed MCTS. Still, we were disappointed by the performance of RL and we were determined to find out how to make it work better.

One promising avenue is using transfer in RL to increase the difficulty of tasks slowly. Instead of learning a problem from scratch, one starts with an easier task. After understanding the simple version, one continues training on a more difficult version. Every change in difficulty is then a transfer in RL. To be sure about the field and what to do we set out to create an overview of the field. To do so we scraped all available papers that contained the three keywords *transfer reinforcement, and learning* in the Microsoft Academic Graph. After combing out duplicates and irrelevant papers we still ended up with almost 300 contributions that we analyzed according to Taylor & Stones TRL categories. The main takeaway from our overview is that TRL mainly focuses on a static amount of tasks that only slightly differ and are all within the same domain. A true cross-domain transfer is an unsolved challenge and the few papers that actually did it had domains that were fairly similar. However, the small incremental task changes gave us the idea that this could be improved by leveraging PCG so that the learned policy generalizes better to unseen task varieties that have not been included in the static task set. This can also be extended to small difficulty increases which then is akin to automated curriculum learning.

Having an idea of how to sensefully apply TRL we attempted our first experiments but quickly hit the problem of reproducibility. Training RL policies based on neural networks is hardly reproducible, a problem that plagues the whole machine-learning field. However, reproducibility is a strongly desirable trait because in TRL every task switch creates a chain of non-reproducibility. While we did not come up with a solution to the problem we did find an intermediary patch: Replay traces. Most RL environments are deterministic and hence reproducible if the same actions are played out with the same initial seed. If we save both we can re-simulate the rest of the data such as observations and rewards while saving hard-disk storage as we do not have to save these. Moreover, sharing replay traces enables reviewers of scientific publications to verify the validity of portrayed figures and tables.

With a solid foundation to do verifiable experiments, we set out to actually apply TRL and PCG together. For that, we transformed the creative *Linerider* game from 2D into 3D space. The game itself is purely creative and does not have a set goal, which has sparked players to come up with wonderful creations such as tracks that play out synchronously to music. This is challenging as after a track is drawn one presses a play button and then a rider on a sled starts to drive down the track without any interaction from the user. Physics and gravity are the only guiding powers here. For our RL-environment, we had to of course define a goal, which in our case is to ensure that the rider arrives at a target position. This goal can be below, above, or at the same height as the rider's starting position. The interesting thing about this environment is that the task to solve itself is an application of PCG. Initial experiments showed that the distance between the starting position and the goal has to be small for RL to learn the task. However, through small incremental increases in distance, hence applying TRL to the problem, we were able to train policies that were able to build tracks for larger distances as well.

Lastly, in the vein of PCG, we also investigated Large Language Models (LLMs) to generate content for games. Related research focuses on having dynamic dialogues where the user can input anything they want to. However, user input to LLMs can easily be used to jailbreak its safety systems and derail them to spew toxic content. Our idea is *Chatter*. Only generate small utterances of NPCs based on pre-defined prompts of developers. We empirically show that this works quite well. Moreover, in this work, we show that the majority of consumer gaming hardware ($\approx 70\%$) would be powerful enough to run a local LLM next to a AAA-Game such as Cyberpunk.

In conclusion, we have shown that PCG and TRL synergize well. PCG can be applied to create many different tasks that help train more general RL policies. And

TRL can help to achieve better results when applying RL to a PCG problem.

Curriculum Vitae

Matthias Müller-Brockhausen was born on the 24th of August, 1994, in Bonn, Germany. In 2016 he obtained his BSc in Computer Science from the Munich University of Applied Sciences. Continuing to follow his passion for computers in 2020 he graduated cum laude with a MSc degree in Computer Science at Leiden University. Due to his love for software development, he was also the Founder and CTO of ClimbZ GmbH from 2016 to 2018, where he led the development of an isomorphic app first in the Meteor framework (v1) and later transitioned to using React & React Native (v2). In 2020 he started a PhD research position in Reinforcement Learning at Leiden University. His research interests lie in AI for (video)games. Before starting his PhD position he gained experience as a Teaching Assistant for the courses Reinforcement Learning & Modern Game AI between January 2020 and June 2020. During his PhD studies, he followed a variety of courses furthering his understanding of the research world such as scientific conduct where he learned the values of integrity and how to produce research in an honest ethical manner. Moreover, he followed courses to acquire the University Teaching Qualification (UTQ). With this qualification in hand, he plans on continuing his career as a University Lecturer.

List of Publications

Primary Author

- Müller-Brockhausen, M., Preuss, M., Plaat, A.: A new challenge: Approaching tetris link with AI. In: 2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021. IEEE (2021), https://doi.org/10.1109/CoG52621.2021.9619044
- Müller-Brockhausen, M., Preuss, M., Plaat, A.: Procedural content generation: Better benchmarks for transfer reinforcement learning. In: 2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021. IEEE (2021), https://doi.org/10.1109/CoG52621.2021.9619000
- Müller-Brockhausen, M., Plaat, A., Preuss, M.: Towards verifiable benchmarks for reinforcement learning. In: IEEE Conference on Games, CoG 2022, Beijing, China, August 21-24, 2022. pp. 159–166. IEEE (2022), https://doi.org/10.1109/CoG51982.2022.9893715
- Müller-Brockhausen, M., Khalifa, A., Preuss, M.: Scalable procedural content generation via transfer reinforcement learning. In: Data Science and Artificial Intelligence (DSAI). Springer (2024), https://doi.org/10.1007/978-981-97-9793-6
- Müller-Brockhausen, M., Barbero, G., Preuss, M.: Chatter generation through language models. In: IEEE Conference on Games, CoG 2023, Boston, MA, USA, August 21-24, 2023. IEEE (2023), https://doi.org/10.1109/CoG57401.2023.10333244

Collaborated on

- Kampert, D., Varbanescu, A.L., Müller-Brockhausen, M., Plaat, A.: Mimicking the human approach in the game of hive. In: IEEE Symposium Series on Computational Intelligence, SSCI 2021, Orlando, FL, USA, December 5-7, 2021. IEEE (2021), https://doi.org/10.1109/SSCI50451.2021.9659999

- Müller-Brockhausen, M., Plisnier, H.: Transferring while playing the rl agent. In: Demo at BNAIC/BeNeLearn 2022: Joint International Scientific Conferences on AI and Machine Learning (2022), https://bnaic2022.uantwerpen.be/wp-content/uploads/BNAICBeNeLearn_2022_submission_6564.pdf
- van der Staaij, A., Prins, J., Prins, V.L., Poelsma, J., Smit, T., Müller-Brockhausen, M., Preuss, M.: Believable minecraft settlements by means of decentralised iterative planning. In: IEEE Conference on Games, CoG 2023, Boston, MA, USA, August 21-24, 2023. IEEE (2023), https://doi.org/10.1109/CoG57401.2023.10333146
- Barbero, G., Müller-Brockhausen, M., Preuss, M.: Challenges of open world games for ai: Insights from human gameplay. In: Data Science and Artificial Intelligence (DSAI). Springer Nature (2024), https://doi.org/10.1007/978-981-97-9793-6

Acknowledgments

I want to use this Section to show gratitude towards everyone in my life who has shaped the path to this great success. First, some general acknowledgments to people who shaped me, followed by specific thanks relevant to the making of this dissertation and my time as a PhD.

The first thanks go out to my parents who have brought me into this world and provided me with everything I wished for. They have always supported me in any decision I made and are especially proud and curious about my research endeavors. I want to thank my siblings for always being there for me and never judging any of my life decisions.

Next, I want to thank the people that directly influenced this dissertation and the path of its creation. I want to stress that there is no particular order to these acknowledgments.

First and foremost are of course, my supervisors. Thank you, Mike Preuss! I was never planning on pursuing a PhD. But it was his words that have convinced me otherwise and for that I am eternally grateful! For the curious ones, he managed to convince me with a simple question: "Don't you want to work on something that you are interested in?"

Of course, I also want to thank Aske Plaat. His support has made the whole trajectory seem manageable and easy. Any time something came up that made me fear or struggle he had the right words to convince me otherwise. Also thank you both Mike and Aske for always giving great and useful feedback on any of my work!

Next, I want to thank two colleagues who started their endeavor with me in the field of RL: Mike Huisman and Zhao Yang. Working together with both of you was a great learning experience, even though it did not result in a publication. Especially the competition provided by Mike's success and speed in finishing his thesis rather quickly was a great motivator to keep giving everything throughout this whole trajectory.

I also want to thank the Reinforcement Learning Group. Even when I was just a masters student everyone was incredibly welcoming. My opinions and comments were never belittled due to a lack of knowledge or experience on the topic. Besides Mike's convincing quote this great atmosphere was a major deciding factor in accepting the PhD position. Every RLG meeting was and still is fun and enjoyable. On that

topic of course thanks to all the people that were part of the RLG-Group in one way or another including Hui Wang, Mike Huisman, Zhao Yang, Diederik Roijers, Daan Pelt, Andreas Sauter, Koen Ponse, Felix Kleuker, Casper Gyurik, Erman Acar, Simon Marshal, Márton Fejér, Alvaro Serra-Gomez, Evert van Nieuwenburg, Lindsay Spoor, Bernhard Hilpert, Jan Krzywda, Jacopo Di Ventura.

I want to thank Thomas Moerland. Any time I had small questions you allowed me to disturb you in your office and tried your best to help me find an answer.

Although my time in the office was limited in the first two years due to the COVID pandemic I want to thank all the colleagues that have made any time spent at the office memorable and great.

Thank you, Giulio Barbero for being a great colleague to share an office with. Your cultural influence brings great joy to the Games Research Lab, and ensures it is never dull in the office! Also thank you for always sharing your "tea"! Besides being very fun to hang out with we even managed to publish a paper together and had my greatest work trip yet in Indonesia! On that note of course thanks to everyone from the DSAI Conference for choosing such an amazing location to host.

My gratitude goes out to Marcello A. Gómez-Maureira for creating the amazing course "Introduction to Video Gamemaking" that I have been lucky enough to take over after his departure. Focusing on something video game-related is truly a dream come true. I am very grateful to teach this course.

I am grateful for Max van Haastrecht as he is a great selfless colleague. Your engagement in the University Council has improved my and many other colleagues' lives for the better. Besides that, you have also organized multiple PhD Weekends that have enabled us to catch up on the lost social interaction caused by the COVID-Pandemic. And of course, it was a great pleasure to organize one of these weekends together with you!

On the note of PhD Weekends, I want to thank Nathan Schiele for allowing me to help him cook great burgers at one of those events. You were also a great entertainer during break times at the office and were also easy and efficient to work together with when we had to prepare a course for the new Cybersecurity Bachelor.

Further on the topic of PhD Weekends, I want to thank Felix Kleuker, Koen Ponse, and Guus Toussaint for giving me the feeling that the weekend I helped to organize was a great success. And I want to thank Roy de Winter for teaching us so many great Dutch social games to keep the evenings entertaining.

On the note of colleagues making breaks enjoyable, I want to thank Tom Kouwenhoven, Bran Renting, Michiel van der Meer, Alan Kai Hassen, Lennard Froma, and

Bernhard Hilpert.