Fast and Knowledge-Free Deep Learning for General Game Playing (Student Abstract)

Michał Maras, Michał Kępa, Jakub Kowalski, Marek Szykuła,

University of Wrocław, Faculty of Mathematics and Computer Science
mmaras1999@gmail.com, kempus1999@gmail.com, jko@cs.uni.wroc.pl, msz@cs.uni.wroc.pl

Abstract

We develop a method of adapting the AlphaZero model to General Game Playing (GGP) that focuses on faster model generation and requires less knowledge to be extracted from the game rules. The dataset generation uses MCTS playing instead of self-play; only the value network is used, and attention layers replace the convolutional ones. This allows us to abandon any assumptions about the action space and board topology. We implement the method within the Regular Boardgames GGP system and show that we can build models outperforming the UCT baseline for most games efficiently.

Introduction

General Game Playing (GGP) is an Artificial Intelligence challenge focused on developing an autonomous game-playing agent that can play, without human intervention, any game given its rules (**?**). Such a task requires a proper formalism to encode a possibly large class of games in a machine-processable and simultaneously human-readable way. Several such formalisms were developed, Stanford's Game Description Language (GDL) (**?**) being arguably the most famous and deep-researched, providing a great domain for testing AI algorithms, especially Monte Carlo Tree Search (MCTS) with UCT (**?**). A newer approach, focusing on more efficient game processing, is Regular Boardgames (RBG) (**?**), which encodes game rules as regular expressions. A natural path of GGP research is to apply Neural Networks (NN) and Deep Reinforcement Learning (DRL).

The most famous NN-based approach that was advertised as general was AlphaZero (**?**), applying the same learning algorithm for Go, Chess, and Shogi. However, although the proposed method really generalizes (at least among two-player, zero-sum board games), the network architecture had to be manually prepared for each game, which is inconsistent with the pure GGP principles. A clone of AlphaZero, based on GDL and aimed to resolve some of these restrictions, was presented in (**?**). MCTS-based DRL approach for Ludii GGP system, created via a bridge to Polygames can be found in (**?**). Deep Reinforcement Learning using only value networks combined with a variant of Unbounded Minimax is described in (**?**).

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

We address some issues that appear in the GGP context and focus on training within a short time limit and further restricting assumptions about the given game rules.

Our Method

We adapted AlphaZero to the RBG framework and we focus on two-player zero-sum games, but the proposed method is limited only by the requirement of perfect information. Our modifications focus on the following areas:

Action space. The original AlphaZero approach requires the knowledge of action space, which must be either defined manually or inferred from the game rules (in GGP). There are two issues. First, the action space depends on a particular game encoding, and one game can have many implementations. Second, since tensors generally must have a fixed shape, games with a huge action space are particularly problematic, even if they have a small branching factor. The second problem could be alternatively addressed by splitting actions into elementary fragments (**?**), yet this may lose heuristic information of actions.

To address these problems, we omit the policy network and attempt to use only the value network, abandoning any dependence on the action space. Thus, we use the standard UCT formula instead of the PUCT (**?**), a variant of the UCT algorithm that uses a predictor to provide recommendations about the order of actions during exploration.

Board topology. Another piece of information that must be extracted from game rules is the neighborhood of board tiles, which is necessary for the standard convolutional approach. In GGP, extracting such information from the game description requires additional effort and is not always reliable. There is no guarantee that the games will have natural board topology, especially for non-rectangular topologies, or even the descriptions can be intentionally obfuscated. Also, the natural board topology does not guarantee that adjacent cells are, by definition, the most correlated ones.

To avoid assumptions, we propose using an attention-based NN (**?**). The model creates a (sinusoidal or learned positional) embedding for each tile and passes them through the encoder layers. This allows the network to learn the relations between tiles without any game-specific knowledge, regardless of the order of the tiles in the input. The presented method is the first application of attention networks to GGP. To evaluate the potential of self-attention, we propose com-

Game	Board size		Ordered board				Permuted board			
			Attention NN		Convolutional NN		Attention NN		Convolutional NN	
			vs. 1×	vs. 10×	vs. 1×	vs. 10×	vs. 1×	vs. 10×	vs. 1×	vs. 10×
Breakthrough	36	(6×6)	99%±1.1	94%±2.4	99%±0.8	**98**%±1.4	**99**%±0.7	93%±2.5	97%±1.7	93%±2.5
Breakthrough	64	(8×8)	83%±3.7	45%±4.9	**88**%±3.2	**65**%±4.7	**85**%±3.5	**53**%±4.9	77%±4.1	42%±4.8
Breakthrough	100	(10×10)	51%±4.9	9%±2.8	47%±4.9	**15**%±3.5	**17**%±3.7	**4**%±1.9	4%±2.0	1%±0.8
Connect Four	42	(7×6)	**72**%±4.3	57%±4.8	58%±4.8	53%±4.8	44%±4.8	37%±4.6	**49**%±4.9	**42**%±4.8
English Draughts	32	(8×8)	86%±2.3	43%±3.3	85%±2.5	46%±3.4	77%±2.8	36%±3.1	**80**%±2.7	**43**%±3.3
Fox and Hounds	32	(8×8)	91%±2.8	**77**%±4.1	93%±2.5	72%±4.4	87%±3.3	58%±4.8	**94**%±2.4	**72**%±4.4
Gomoku	225	(15×15)	**88**%±3.2	38%±4.8	44%±4.9	39%±4.8	**74**%±4.3	**14**%±3.4	10%±2.9	7%±2.5
Hex	25	(5×5)	83%±3.7	56%±4.9	85%±3.5	60%±4.8	79%±4.0	62%±4.8	**86**%±3.4	65%±4.7
Hex	49	(7×7)	**76**%±4.2	**49**%±4.9	44%±4.9	22%±4.1	**57**%±4.9	**34**%±4.7	19%±3.9	8%±2.7
Hex	81	(9×9)	**10**%±2.9	0%±0.5	2%±1.3	0%	**4**%±1.8	0%±0.5	0%±0.5	0%
Pentago	36	(6×6)	82%±3.8	41%±4.8	**94**%±2.4	**72**%±4.4	70%±4.5	28%±4.4	**80**%±4.0	**46**%±4.9
Reversi	64	(8×8)	88%±3.2	68%±4.5	88%±3.1	71%±4.4	**84**%±3.5	**64**%±4.6	63%±4.7	35%±4.6
The Mill Game	24	$(8 + 8 + 8)$	67%±4.3	36%±4.1	70%±4.2	40%±4.3	61%±3.9	34%±3.6	63%±4.4	34%±4.0
Total average			75%	47%	69%	50%	64%	39%	56%	38%

Table 1: The results of the NN agents with 600 iterations. The baseline opponent is UCT with 600 (1×) and 6,000 (10×) iterations. The dataset for each game was gathered for at most 2h using 20 CPU cores. The 95%-confidence intervals are given.

Game	Size of dataset (MCTS plays)		
	200	400	1,000
	Generation time + Training time		
Breakthrough (6×6)	6s + 20s	11s + 20s	25s + 20s
English Draughts	1.1m + 20s	1.3m + 20s	3.5m + 40s
Reversi	1.2m + 20s	1.2m + 20s	3.5m + 40s
	Attention NN		
Breakthrough (6×6)	4%±1.8	60%±4.8	74%±4.3
English Draughts	31%±2.9	42%±3.2	54%±3.3
Reversi	25%±4.2	22%±4.0	28%±4.3
	Convolutional NN		
Breakthrough (6×6)	41%±4.8	48%±4.9	83%±3.7
English Draughts	42%±3.4	55%±3.5	60%±3.4
Reversi	6%±2.3	7%±2.4	21%±4.0

Table 2: Short training. The results of the NN agents with 600 iterations against UCT baseline with also 600 iterations.

paring it with CNNs based on random permutations of the input board tiles. Reordering the tiles should eliminate the spatially local correlation, which is assumed by default by CNNs. **Fast model generation** To take advantage of the very fast RBG reasoner and to vastly decrease training time, the training dataset is generated by playing games using standard UCT MCTS agents (without NN). This approach provides better quality samples than in the early stages of self-play and allows gathering large amounts of data with limited resources. It can be easily parallelized since MCTS plays can be performed independently on CPU cores and do not require a GPU.

Experiments with Conclusions

We evaluated the trained NN agents by playing 400 games (200 per side) against a standard MCTS agent with game tree reuse. The results of the constant simulations limit are presented in Table 1. Convolutional NN assumes that the board fits within a quad, and the tiles are given in order. To examine the robustness, we tested the behavior after obfuscating by permuting the tiles randomly (the same permutation used for both NNs). Overall, the attention NN has a similar performance to the CNN, even if the order of board tiles is random. The results vary a lot depending on the game, especially on its size. Yet, they suggest the trend that attention is a better choice for larger games, which is particularly visible in size variations of Breakthrough and Hex. It also suffers a smaller win ratio drop on average on our game set when the board is permuted (the mean drop is respectively 11% and 8% for attention NN vs. 13% and 12% for CNN).

Table 2 shows results after very short training. We used smaller NNs here, trained on much less data (less than 3% of the 2h dataset from Table 1). This experiment was inspired by (**?**), where in Breakthrough, after just 400 selfplay games, the winrate close to 80% was achieved. To achieve a similar playing strength, we needed about 1,000 MCTS vs. MCTS training plays. Thus, our plays are of worse quality, but we can generate them in less than a minute. Because of the RBG language efficiency (**?**), in our experiments, optimized Monte-Carlo playouts are much faster than NN evaluation. Such a situation, causing time-based comparison to favor pure MCTS, was not reported in GGP-related literature.

We conclude that knowledge of the action space and board topology is not required for obtaining good models. The data generated from MCTS plays in place of self-plays is still useful; in some cases, it allows beating the baseline after a few minutes of computing. Note that our research was focused on budget training, so the landscape could be different in longer settings.

Acknowledgments

This work was supported in part by the National Science Centre, Poland under project number 2021/41/B/ST6/03691. Maiores dolores itaque rerum consectetur, ea cumque fugiat.