Evaluating Reinforcement Learning Algorithms For Evolving Military Games

James Chao*, Jonathan Sato*, Crisrael Lucero, Doug S. Lange
Naval Information Warfare Center Pacific
*Equal Contribution
{first.last}@navy.mil

Abstract

In this paper, we evaluate reinforcement learning algorithms for military board games. Currently, machine learning approaches to most games assume certain aspects of the game remain static. This methodology results in a lack of algorithm robustness and a drastic drop in performance upon changing in-game mechanics. To this end, we will evaluate general game playing (Diego Perez-Liebana 2018) AI algorithms on evolving military games.

Introduction

AlphaZero (Silver et al. 2017a) described an approach that trained an AI agent through self-play to achieve superhuman performance. While the results are impressive, we want to test if the same algorithms used in games are robust enough to translate into more complex environments that closer resemble the real world. To our knowledge, papers such as (Hsueh et al. 2018) examine AlphaZero on non-deterministic games, but not much research has been performed on progressively complicating and evolving the game environment, mechanics, and goals. Therefore, we tested these different aspects of robustness on AlphaZero models. We intend to continue future work evaluating different algorithms.

Background and Related Work

Recent breakthroughs in game AI has generated a large amount of excitement in the AI community. Game AI not only can provide advancement in the gaming industry, but also can be applied to help solve many real world problems. After Deep-Q Networks (DQNs) were used to beat Atari games in 2013 (Mnih et al. 2013), Google DeepMind developed AlphaGo (Silver et al. 2016) that defeated world champion Lee Sedol in the game of Go using supervised learning and reinforcement learning. One year later, AlphaGo Zero (Silver et al. 2017b) was able to defeat AlphaGo with no human knowledge and pure reinforcement learning. Soon after, AlphaZero (Silver et al. 2017a) generalized AlphaGo Zero to be able to play more games including Chess, Shogi, and Go, creating a more generalized AI to apply to different problems. In 2018, OpenAI Five used five Long Short-term Memory (Hochreiter and Schmidhuber 1997) neural networks and a Proximal Policy Optimization (Schulman et al. 2017) method to defeat a professional DotA team, each LSTM acting as a player in a team to collaborate and achieve a common goal. AlphaStar used a transformer (Vaswani et al. 2017), LSTM (Hochreiter and Schmidhuber 1997), autoregressive policy head (Vinyals et al. 2017) with a pointer (Vinyals, Fortunato, and Jaitly 2015), and a centralized value baseline (Foerster et al. 2017) to beat top professional Starcraft2 players. Pluribus (Brown and Sandholm 2019) used Monte Carlo counterfactual regret minimization to beat professional poker players.

AlphaZero is chosen due to its proven ability to be play at super-human levels without doubt of merely winning due to fast machine reaction and domain knowledge; however, we are not limited to AlphaZero as an algorithm. Since the original AlphaZero is generally applied to well known games with well defined rules, we built our base case game and applied a general AlphaZero algorithm (Nair, Thakoor, and Jhunjhunwala 2017) in order to have the ability to modify the game code as well as the algorithm code in order to experiment with evolving game environments such as Surprise-based learning (Ranasinghe and Shen 2008).

Game Description: Checkers Modern Warfare

The basic game that has been developed to test our approach consists of two players with a fixed size, symmetrical square board. Each player has the same number of pieces placed symmetrically on the board. Players take turns according to the following rules: the turn player chooses a single piece and either moves the piece one space or attacks an adjacent piece in the up, down, right, or left directions. The turn is

then passed to the next player. This continues until pieces of only one team remain or the stalemate turn count is reached. A simple two turns are shown in Figure 1. The game state is fully observable, symmetrical, zero sum, turn-based, discrete, deterministic, static, and sequential.

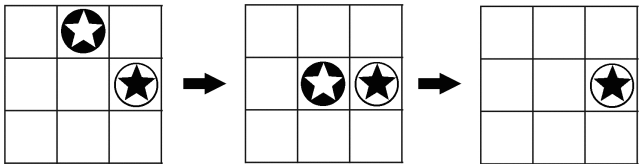

Figure 1: Sample Two Turns: The first of the three boards shows the state of the board before the turn starts. The player of the dark star piece chooses to move one space down resulting in the second board. The third board is a result of the player of the light star piece attacking the dark star piece.

Methodology

The methodology we propose starts from a base case and incrementally builds to more complicated versions of the game. This involves training on less complicated variations of the base case and testing on never-before-seen aspects from the list below. These never-before-seen mechanics can come into play at the beginning of a new game or part-way through the new game. The way we measure the successful adaptation of the agent is based off of comparing the win/loss/draw ratios before the increase in difficulty and after. The different variations to increase game complexity are described in the sections below.

Disrupting Board Symmetry

We propose two methods for disrupting board symmetry. Introducing off-limits spaces that pieces cannot move to, causing the board to not rotate along a symmetrical axis and stay the same board. The second is by disrupting piece symmetry by having non-symmetrical starting positions.

Changing the Game Objective

Changing the game winning mechanisms for the players, suddenly shifting the way the agent would need to play the game. For example, instead of capturing enemy pieces, now the objective is capture the flag. Another example of changing objectives is by having the different players try to achieve a different goal, such as one player having a focus on survival whereas the other would focus on wiping the opponent's pieces out as fast as possible.

Mid-Game Changes

Many of the above changes can be made part-way through game to incorporate timing of changes as part of the difficulty. In addition to the existing changes, other mid-game changes can include a sudden "catastrophe" where the the enemy gains a number of units or you lose a number of units and introducing a new player either as an ally, enemy ally or neutral third party.

Case Study and Results

The base case game consists of a five by five size board and six game pieces with three for each side. The three pieces of each team are set in opposing corners of the board as seen in Figure 2. The top right box of the board is a dedicated piece of land that pieces are not allowed to move to. During each player's turn, the player has the option of moving a single piece or attacking another game piece with one of their game pieces. This continues until only no pieces from one team is left or until 50 turns have elapsed signaling a stalemate. This base case can be incrementally changed according to one or multiple aspects described in the methodology section.

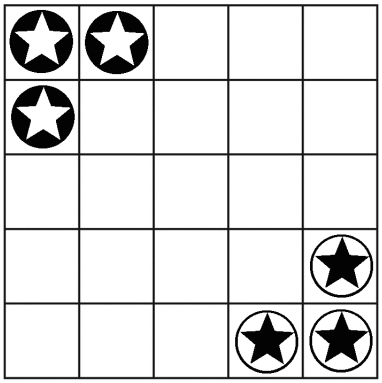

Figure 2: Base case board setup used for initial training and testing.

We trained an AlphaZero-based agent using a Nvidia DGX with 4 Tesla GPUs for 200 iterations, 100 episodes per iteration, 20 Monte Carlo Tree Search (MCTS) simulations per episode, 40 games to determine model improvement, 1 for Cpuct, 0.001 learning rate, 0.3 dropout rate, 10 epochs, 16 batch size, 128 number of channels. The below table and graphs show our results after pitting the model at certain iterations against a random player.

Iteration	Wins	Losses	Draws
0	18	22	60
10	41	8	51
20	45	1	54
30	40	3	57
70	23	4	73
140	41	3	56
200	44	1	55

Convergence occurs around 10 iterations, this is earlier then initially expected possibly due to the lack of game complexity in the base case game. More studies will be conducted once game complexity is increased. We dialed up the Cpuct hyper-parameter to 4 to encourage exploration, the model simply converges at a slower rate to the same winning model as the Cpuct equals 1.

Observations on AlphaZero

Game design is important, since AlphaZero is a Monte Carlo method, we need to make sure the game ends in a timely

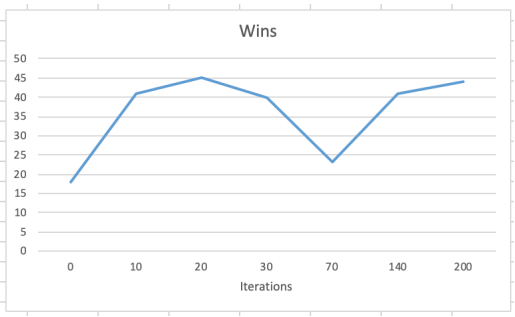

Figure 3: The trained agent starts winning more consistently after 10 iterations.

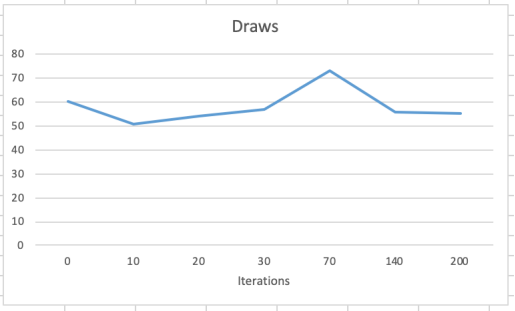

Figure 4: Draws are constant throughout the training process

matter in order to complete an episode before the training becomes unstable due to an agent running from the enemy forever.

Furthermore, we cannot punish draws but instead give a near 0 reward since AlphaZero generally uses the same agent to play both players and simply flips the board to play against itself. This could potentially cause issues down the road if we were to change the two player's goal to be different from one another, for example, player one wants to destroy a building, while player two wants to defend the building at all cost.

The board symmetry does not affect agent learning in AlphaZero.

Non Symmetrical Board

The trained agent will be used to play different Checkers Modern Warfare variants. Starting with one degree variant such as making the board non-symmetrical with random land fills where players cannot move their pieces to. To do this, we disabled locations [0,0],[1,0],[2,0],[3,0],[4,0],[0,1],[1,1],[0,3],[2,3], put player 1 pieces at [2,1],[3,1], and player 2 pieces at [2,4] and [3,4] as shown in figure 5, at the beginning of the game.

The agent trained with 200 iterations from the above section was pitted against a random player. Winning 70 games, losing 1 games, and drawing 29 games. This proves the trained agent can deal with disrupted board symmetry and a game board with different terrain setup.

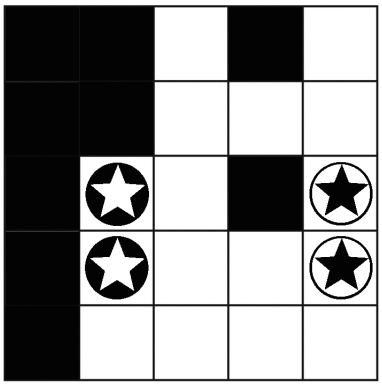

Figure 5: Non-symmetrical board setup used for incremental case testing.

Non Symmetrical Board Change Mid Game

The board starts as the non symmetrical board shown in figure 5, then turns into the board shown in figure 2 without obstacles after 25 turns. 50 turns without a winner results in a draw. The trained agent won 68 games, lost 4 games, and drew 28 games out of 100 games. Proving the agent can perform relatively well with board symmetry change half way through the game.

Non Symmetrical Board with Random Added New Pieces Mid Game

Starting with the non symmetrical board shown in figure 5, at turn 25 in a 100 turn game, add 3 reinforcement pieces for each team at a random location if the space is empty during the turn. The trained agent won 80 games, lost 6, and drew 14, performing relatively well with the new randomness introduced.

Non Symmetrical Board with Random Deleted Pieces Mid Game

Starting with the non symmetrical board shown in figure 5, at turn 25 in a 100 turn game, blow up 3 random spots with bombs, where any pieces at those locations are now destroyed. The trained agent won 84 games, lost 11, and drew 5, performing relatively well with the new randomness introduced.

Non Symmetrical Board with Statically Added New Pieces Mid Game

Starting with the non symmetrical board shown in figure 5, at turn 25 in a 100 turn game, add 3 reinforcement pieces for each team at specific locations if the space is empty during the turn. Team 1 is reinforced at locations [2,1] [3,1] [4,1], team 2 is reinforced at locations [2,4][3,4][4,4]. The trained agent won 77 games, lost 2, and drew 21, performing relatively well with mid game state space changes.

Non Symmetrical Board with Statically Deleted Pieces Mid Game

Starting with the non symmetrical board shown in figure 5, and at turn 25 in a 100 turn game, blow up every piece at locations [2,1] [3,1] [4,1] [2,4] [3,4] [4,4]. The trained agent won 83 games, lost 8, and drew 9, performing relatively well with mid game state space changes.

Non Symmetrical Board with Non Deterministic Moves

Movements and attacks are now non-deterministic, where 20% of the moves or attacks are nullified and resulting in a no-op. Testing on a 50 turn game. The trained agent won 55 games, lost 10, and draw 35. We then tested the same rules with 50% of the movements and attacks nullified, The trained agent won 34 games, lost 10, and drew 56. Finally we changed it to 80% of the movements and attacks nullified, the trained agent won 8 games, lost 3, and drew on 89 games. The results indicate the agent preformed relatively well, with the observation that more randomly assigned no-ops will result in more draw games.

Changing Game Objective

Changed the game objective to capture the flag, and used the agent trained on eliminating the enemy team. the agent won 10 games, lost 4 games, and drew 6 games over 20 games. We then changed the game objective after 25 turns in a 50 turn game, the agent won 9 games, lost 5 games, and drew 6 games over 20 games. The agent performed relatively well with changing game objectives even though it was not trained on this objective. We suspect this is due the trained agent having learning generic game playing techniques such as movement patterns on a square type board.

Non Symmetrical Game Objective

Finally, we changed the game objective to be non symmetrical, meaning the 2 players have different game winning conditions. player 1 has the goal to protect a flag, while player 2 has the goal to destroy the flag. AlphaZero could not train this agent with good results since it uses one neural network to train both player. Therefore, future work will be to change the AlphaZero algorithm to a multi-agent learning system where there are 2 agents trained on 2 different objectives.

Conclusion

As we incrementally increase the complexity of the game, we discover the robustness of the algorithms to more complex environments and then apply different strategies to improve the AI flexibility to accommodate to more complex and stochastic environments. We learned that AlphaZero is robust to board change, but less flexible dealing with other aspects of game change.

References

Brown, N., and Sandholm, T. 2019. Superhuman ai for multiplayer poker. *Science* 11 July 2019.

Diego Perez-Liebana, Jialin Liu, A. K. R. D. G. J. T. S. M. L. 2018. General video game ai: a multi-track framework for evaluating agents, games and content generation algorithms. *arXiv:1802.10363*.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2017. Counterfactual multi-agent policy gradients. *arXiv:1705.08926*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9:1735–80.

Hsueh, C.-H.; Wu, I.-C.; Chen, J.-C.; and Hsu, T.-S. 2018. Alphazero for a non-deterministic game. *2018 Conference on Technologies and Applications of Artificial Intelligence*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv:1312.5602*.

Nair, S.; Thakoor, S.; and Jhunjhunwala, M. 2017. Learning to play othello without human knowledge.

Ranasinghe, N., and Shen, W.-M. 2008. Surprise-based learning for developmental robotics. *2008 ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems (LAB-RS)*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv:1707.06347*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; and Timothy Lillicrap, Madeleine Leach, K. K. T. G. D. H. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T.; Simonyan, K.; and Hassabis, D. 2017a. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv:1712.01815*.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; Chen, Y.; Lillicrap, T.; Hui, F.; Sifre, L.; van den Driessche, G.; Graepel, T.; and Hassabis, D. 2017b. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

Vaswani; Ashish; Shazeer; Noam; Parmar; Niki; Uszkoreit; Jakob; Jones; Llion; Gomez; N, A.; Kaiser; Lukasz; Polosukhin; and Illia. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 5998–6008.

Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A. S.; Yeo, M.; Makhzani, A.; Kttler, H.; Agapiou, J.; Schrittwieser, J.; Quan, J.; Gaffney, S.; Petersen, S.; Simonyan, K.; Schaul, T.; van Hasselt, H.; Silver, D.; Lillicrap, T.; Calderone, K.; Keet, P.; Brunasso, A.; Lawrence, D.; Ekermo, A.; Repp, J.; and Tsing, R. 2017. Starcraft ii: A new challenge for reinforcement learning.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. *arXiv:1506.03134*.