The Effect of Antagonistic Behavior in Reinforcement Learning

Ted Fujimoto[*], **Timothy Doster, Adam Attarian, Jill Brandenberger, Nathan Hodas**
Pacific Northwest National Laboratory
{ted.fujimoto, timothy.doster, adam.attarian, jill.brandenberger, nathan.hodas}@pnnl.gov

Abstract

The significant achievements of deep reinforcement learning (RL) have motivated researchers to also investigate its shortcomings. Such work has shown that typical methods in deep RL tend to produce brittle policies that overfit to the training environment. In this paper, we introduce the notion of purely antagonistic behavior in value-based agents, where the objective is not to maximize reward but to minimize the victim's value over time. This notion is motivated by the scenario in which an antagonistic human architect, without access to the environment's reward function, wants to build an RL agent that can impede another well-trained RL victim agent. First, we formalize a notion of antagonistic behavior in RL. Then, we provide experiments that show how a purely antagonistic agent performs compared to a well-trained victim that learns directly from the game's rewards. Our results suggest that if one's goal is to find vulnerabilities in well-trained agents, direct access to the environment's rewards is not necessary, and antagonistic behavior can be measured independently from environment wins and losses.

Introduction

The recent accomplishments of RL and self-play in Go (Silver et al. 2016), Starcraft 2 (Vinyals et al. 2019), DOTA 2 (Berner et al. 2019), and poker (Brown and Sandholm 2019) are seen as pivotal benchmarks in AI progress. While these feats were being accomplished, work was also being done showing the vulnerabilities of these methods. (Huang et al. 2017) showed that policies are especially vulnerable against adversarial perturbations of image observations when white-box information is utilized. (Gleave et al. 2020) showed that an adversarial agent can easily learn a policy that can reliably win against a well-trained opponent in high-dimensional environments by learning from the rewards provided by the environment. One thing missing from their findings is investigating if the adversarial agent might produce negative side effects that cannot be observed if the focus is just on environment rewards. Our work builds on these past accomplishments by taking a more realistic view at how a hypothetical architect might actually develop an RL agent that learns how to exploit a well-trained agent. We formalize such an architect by assuming the following scenario:

1. A well-trained RL agent ν.
2. An antagonistic RL agent α that does not have access to the environment's reward function.
3. The (human) architect of α who can observe ν's behavior, and interpret when ν is succeeding or failing at a given state in the environment. For example, the architect can accomplish this by using ν's value function to measure ν's level of frustration at each state.

Contributions We contribute a definition of purely antagonistic behavior that intuitively represents an agent that intentionally sabotages a victim agent. We define such behavior as the actions that minimize the victim's state-value over time, not as the actions that optimize the expected value of the cumulative sum of environment rewards. Here, purely antagonistic behavior forces the victim into sub-optimal situations but is not concerned with long-term wins or loses. This notion of antagonistic behavior is motivated by the assumption that even when two humans in conflict have no knowledge of their respective victim's long-term goals, they can quickly recognize their victim's immediate, short-term pain or frustration. Although placing short-term, antagonistic motivation as the sole focus of an RL agent runs counter to the goal of long-term sequential decision making, the results presented here show that an antagonist can be surprisingly effective against a victim by just exploiting immediate weaknesses. Our results suggest that (1) if one's goal is to find vulnerabilities in well-trained agents, direct access to the environment's rewards is not necessary in all cases, (2) purely antagonistic behavior toward an agent in a particular environment can be encapsulated in a single value function, (3) having access to the victim's state-value function does not guarantee a winning policy for the antagonist, and (4) antagonistic behavior can be measured independently from environment wins and losses.

In Section 2, we review past related work in adversarial RL and differentiate it from our notion of antagonistic behavior. In Section 3, we formally define antagonistic behavior and justify our definition. In Section 4, we provide the results of experiments that show how such agents behave in certain board games. In Section 5, we provide some conclusions and suggestions for future work.

[*]Corresponding author.

Related Work

The work presented here attempts to further understand negative side effects in AI, which are one of the concrete problems in AI safety mentioned in (Amodei et al. 2016). Specifically, our work formally defines a RL agent that learns to frustrate the victim, which can deliberately increase the negative side effects a victim may encounter. This failure mode can also be seen as an example of adversarial optimization (Manheim 2019). Gary Marcus makes this problem clearer by necessitating the need to define harm produced by AI systems so that it can be avoided (Fridman 2019). Although these authors explain the intuition behind these problems, they do not formally define a RL framework that a hypothetical antagonist would use to train an agent that exacerbates the problems they describe.

Some accomplishments have been made in introducing an adversarial element to the process of policy improvement in RL agents. Some examples include Robust Adversarial RL (Pinto et al. 2017), and Risk Adverse Robust Adversarial RL (Pan et al. 2019). Although our work in this paper mainly focuses on observing agent behavior, investigating how our definition of antagonistic behavior may lead to more robust policies can be explored in future research.

There has also been research in RL that assumes an adversary that subverts the training of an agent. As mentioned in the previous section, (Huang et al. 2017) use the victim's image observations to negatively affect its policy. (Huang and Zhu 2019) and (Zhang et al. 2020) use reward poisoning to trick the victim into learning a nefarious policy. Our work does not assume the antagonist has the ability to manipulate the victim's observations, or the ability to poison the environment rewards the victim receives.

We will model our multiagent environment as a Markov game similar to what was defined in (Littman 1994). Like in (Gleave et al. 2020), we will fix the victim's policy so that the antagonist can act as if in a single-agent environment. Unlike (Gleave et al. 2020), the antagonistic agents in this paper do not have black-box access to the victim's actions and do not learn from rewards provided by the environment. In this paper, we use the term "antagonistic" instead of "adversarial". This choice was made since the term "antagonist" is closer to our intention of having an agent learn to "frustrate" a victim. Also, the term "adversarial" has been defined differently in past research on adversarial policies (Gleave et al. 2020) and adversarial value functions (Bellemare et al. 2019), which both learn from the environment's rewards.

Defining Antagonistic Behavior

Preliminaries

We model the agents as a two-player Markov game similar to the framework in (Shapley 1953) and (Gleave et al. 2020). Assume the antagonist (α) and the victim (ν) play a Markov game $M = (\mathcal{S}, (A_\alpha, A_\nu), P, (R_\alpha, R_\nu))$ where $\mathcal{S}$ is the state set, A_α and A_ν are action sets, R_α and R_ν are reward functions, and P is the state-transition probability distribution. There are also the antagonist's policy (π_α) and the victim's policy (π_ν). Like in (Gleave et al. 2020), we hold the victim's policy fixed during the antagonist's training. Next, we will define the antagonist's reward function R_α to represent antagonistic behavior and not rewards provided by the environment.

Value-Based Antagonistic Behavior in Reinforcement Learning

The scenario of focus will be creating an agent that learns purely antagonistic behavior without direct access to a reward function. To accomplish this, we introduce the notion of antagonistic behavior in reinforcement learning. Defining antagonistic behavior provides a foundation for creating an agent that has no access to the environment's reward function, but learns to choose the action that leads to the state with the smallest victim state-value.

In normal-form zero-sum games, antagonistic behavior can be defined as the actions that minimize the victim's utility function. This is appropriate since each player makes only one action. In RL, where one must account for a sequence of states and actions, this is not generally appropriate. To understand this complication, we consider the game of Go. In Figure 1, assume the victim is well-trained and the antagonist wants to pick the move that immediately leads to the victim being worse off. If the antagonist simply picks the point on the board that would have produced the smallest victim state-value, it would have picked a point having little impact on the groups of stones that are the current focus. This follows the intuition that a trained Go player will ascribe a higher value to moves that have a higher impact on winning, and lower values to moves that have little impact on winning. In this case, the best move for both agents is to put a stone on the same point. The antagonist that strictly adheres to the negative of the victim's state-value function will not make that move. Even if the antagonist does not receive rewards from the environment, this presents a dilemma since the best move and the move that would most "frustrate" the victim are the same. Hence, just using the negative of the victim's value function as the antagonist's reward is not enough for the antagonist to learn the purely antagonistic behavior we desire since the antagonist may learn actions that have no impact on the victim's state-value.

To remedy this, we acknowledge the existence of different states over time as a useful contrast between normal-form games and RL. That is, we want a notion of purely antagonistic behavior to "frustrate" the victim given the current state. More clearly, the antagonist at state s_n would want to take the action that leads to a lower victim state-value at state s_{n+1}. Hence, a purely antagonistic agent minimizes the value $V_{\pi_\nu}(s_{n+1}) - V_{\pi_\nu}(s_n)$, or equivalently, maximizes the value $V_{\pi_\nu}(s_n) - V_{\pi_\nu}(s_{n+1})$. This fixes the Go situation presented above. Assume, using the victim's state-value function, the current state s_n has a value of 100, the value of the state s_{n+1} where the antagonist places a stone on the opposite corner is 99, and the value of the state s'_{n+1} where the antagonist places a stone on the best current point is 50. Then $V_{\pi_\nu}(s_{n+1}) - V_{\pi_\nu}(s_n)$ is -1 while $V_{\pi_\nu}(s'_{n+1}) - V_{\pi_\nu}(s_n)$ is -50, which would lead the antagonist to value the action that leads to s'_{n+1} higher than the action leading to s_{n+1}. This follows our intuition of how an antagonist would act in a way that would "frustrate" its victim the most. Such an-

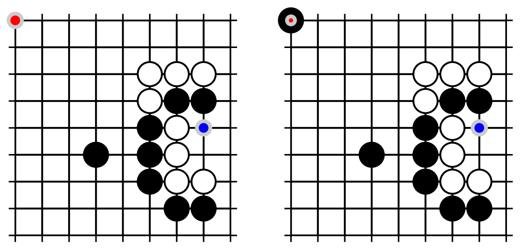


Figure 1: **Left**: The blue dot shows the white player's action that would lead to the state with the highest state-value. The red dot would lead to the state with the lowest state-value. **Right**: If black chooses the move that leads to white's lowest state-value assuming it were white's turn, then it would choose to place its stone on the red dot. This move would likely be to white's benefit since (1) it has little impact on the current state of the game, and (2) black's best move would also be the blue dot.

tagonistic behavior is focused more on the victim and not necessarily on environment reward.

Now that the intuition has been explained, we can formally quantify learned antagonistic behavior in RL agents. Let $V_\nu : \mathcal{S} \to \mathcal{R}$ be the victim's value function and $\Delta V_t = V_{\pi_\nu}(s_{t+1}) - V_{\pi_\nu}(s_t)$. Let a policy $\pi_\alpha : \mathcal{S} \times \mathcal{A} \to [0, 1]$ be the antagonistic policy.

Definition 1. The *Antagonistic Value Function* of a state s under a policy π_α is defined as:

$$V_{\pi_\alpha}(s) = -\mathbb{E}_{\pi_\alpha}[\Sigma_t^\infty \gamma^t \Delta V_t | S_t = s] \tag{1}$$

where γ is the discount rate. Hence, $R_\alpha(t) = -\Delta V_t$.

Hence, maximizing this value is equivalent to minimizing the positive change of the victim's state-value at each time step t. This can be seen as an example of adversarial optimization since its goal is to lower the victim's state-value over time (Manheim 2019). Since the value function only takes states as inputs, this definition is appropriate for both turn-based and simultaneous move Markov game environments. Estimating the value function does not imply knowledge of the reward function, because inverse RL is fundamentally under-constrained.

The motivation here is different from fictitious play (Berger 2007). Our goal is not to minimize the opponent's returns but to observe the behavior of agents with access to their victim's value function. Research into the empirical similarities and differences could be of further interest.

Assumption 1. The antagonistic value function represents the ideal architect with the ability to observe and measure the victim's "frustration" level at each state and use those measurements as rewards for the antagonistic agent.

The following theorem can be seen as an explanation for what happens when an opponent lowers the victim agent's state-value over time.

Theorem 1. Assume a finite, turn-based, zero-sum, deterministic game with no intermediate rewards and let ν have the static, optimal value function V_ν^*. If ν wins, $0 < \gamma < 1$, ν's policy is greedy, and n is the number of time steps (or game moves) left at state s to traverse and win the game, then $n = \log_\gamma V_\nu^*(s)$.

This theorem states that in a board game, like Breakthrough or Connect-4, the number of steps left to win the game monotonically decreases as V_ν^* increases. This will help because if the antagonistic agent is successful at lowering the value function of the victim, and the victim follows the optimal value function, the length of the game will increase. This is relevant to our investigation of antagonistic behavior if we make the following assumption:

Assumption 2. In board games like Breakthrough, Havannah and Connect-4, antagonistic behavior is measured by the average number of moves it takes to complete a game.

This is motivated by the intuition that an expert player wants to win as quickly as possible. For example, in Breakthrough and Havannah, an antagonistic agent needs to know effective blocking strategies that prevent the victim from winning. The experiments in the next section will verify this theorem for some games. However, in games like Go, this assumption might not hold since an artificial agent in this game might not stop playing until all possible moves have been exhausted.

Justification

To justify the antagonistic value function above as a definition that represents antagonistic behavior, we make the following assumptions: (1) antagonistic behavior means minimizing the victim's value at every state, (2) the Markov property holds, and (3) antagonistic behavior and self-interested behavior can be distinct. From these premises, we conclude that the antagonistic value function encapsulates antagonistic behavior in RL agents.

It is not clear how (1) could be false if the victim's value function is optimal. It could be that the victim has a policy that follows a sub-optimal value function, but an agent using the antagonistic value function would still seem antagonistic from the victim's perspective since the antagonist is still trying to minimize the victim's current state-value. Intuitively, (3) is true in human social interaction since we occasionally work with our adversaries to prevent a scenario that would be against the self-interests of both parties. Although the difference becomes less clear in games or RL environments that are zero-sum (Gleave et al. 2020), (3) still holds since agents in these scenarios learn from rewards. It is possible (as will be shown in the Experiments section) that even in zero-sum games, making it harder for the victim to win does not necessarily imply the antagonist will win the game.

Any value function that represents antagonistic behavior would still need access to the victim's value function V_{π_ν}. As illustrated earlier in the Go example, minimizing V_{π_ν} alone will not work in general. Since the Markov property ensures that each state includes information about all aspects of the past agent-environment interaction that make a difference for the future (Sutton and Barto 2018), knowing the difference ΔV_t alone is enough for the antagonist to learn how to exploit the victim. That is, we do not need any other previous states since the current state is all we need. If (3) is true, then we don't need to include the rewards provided by the environment to learn antagonistic behavior. If (1) is true, then we can conclude that our definition encapsulates antagonistic behavior the same way regular value functions encapsulate behavior optimized to find environment rewards.

Experiments

Training Progress

We train the antagonist and victim using the game environment OpenSpiel (Lanctot et al. 2019). Both agents' policies are deep Q-networks that are trained in a manner similar to (Mnih et al. 2015). Specifically, the agents use ϵ-greedy policies with 6-layer or 7-layer linear neural network value functions implemented in PyTorch (Paszke et al. 2019). The victim is trained first over 500,000 games of Breakthrough, Havannah, and Connect-4 against random agents. The victims have their discount rate γ set to 0.95. Against random agents, each victim agent reaches around 95% to 99% win rate. Then the antagonist is trained against the victim. For the antagonist to learn the value function defined above, the victim provides the values of the states before and after the antagonist chooses its action. The antagonist accepts the difference $V_\nu(s_n) - V_\nu(s_{n+1})$ as its reward. To make the experiments easier to interpret, the adversaries have their discount rate γ set to 1.

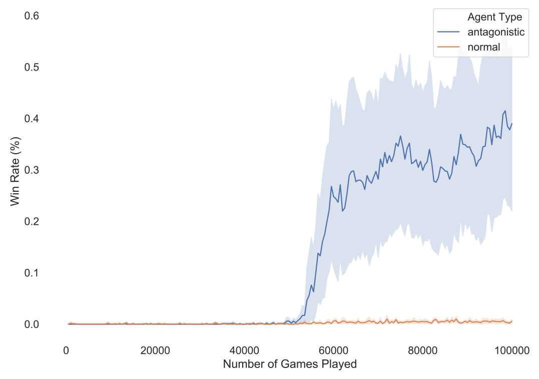
Training over 100,000 games against a pretrained victim, we measured the training performance of 10 antagonistic DQN agents (each with their own random seed), and 10 normal DQN agents (each with the same random seed as their respective antagonist) that learn from environment rewards. We measure both the win rate and the number of moves played. In Figure 2, we see the antagonist has both a higher win rate and a steady increase in the number of moves per game. Although the antagonist performed better on average than the victim in Breakthrough, we also focus on the move count since it measures how well the antagonist learns antagonistic behavior. Interestingly, as seen in Figure 3, two antagonistic agents consistently had a win rate near 0% but were still able to increase the number of moves over time. In Havannah, the antagonistic agent overall increases the number of moves during the training process but ends at around the same number. The results in Figure 4 show that in Connect-4, the metrics for both agents are about the same. This is likely due to the simplicity of the game compared to Breakthrough and Havannah. However, the difference in the number of moves between antagonist and victim began to increase in Connect-4 when we used a longer training method. The results from Connect-4 and Go, are explained in later subsections.

Methodology

Training Evaluation To evaluate the antagonist, we catalog its performance as it trains over many games. However, when evaluating two trained game-playing DQN agents, it is not immediately obvious how to measure aggregate performance within a snapshot in time. During evaluation, ϵ-greedy agents need to set ϵ to 0 to avoid arbitrary randomness that does not contribute to the measurement of performance. On the other hand, if we set ϵ to 0 in games where the first state is the same in every game, a greedy policy is just a function from states to actions. That is, the greedy policy makes the same move for every state. Hence, two greedy agents playing Go or Breakthrough from the very beginning will just make the same moves every game.

Every 500 games during the training of the antagonistic agent, the method we devised is the following:

1. Start a new game between the victim agent and a random agent.
2. Have the agents play a constant number of moves (typically less than half of the average game length between the victim and a random agent).
3. When the constant number of moves is reached, replace the random agent with the antagonist. Have both the victim and antagonist agents continue playing until the end of the game.



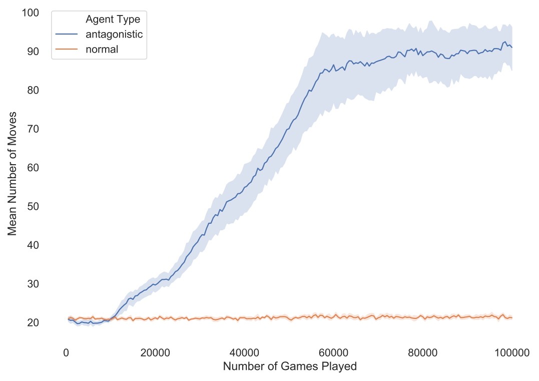
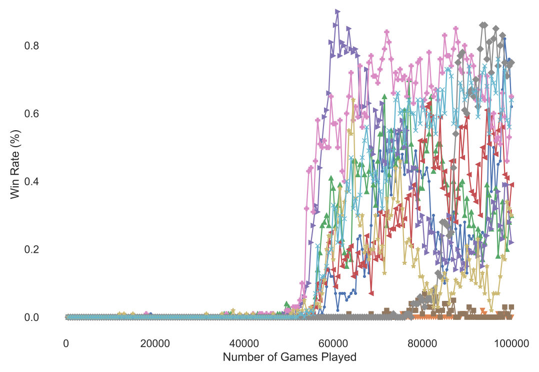


Figure 2: Breakthrough Training Results: **Top**: While the normal DQN has a win rate consistently close to 0%, the antagonistic DQN ends with a mean win rate of around $40 \pm 15\%$. **Bottom**: The normal DQN mean number of moves is consistently around 21. The antagonistic DQN ends with a mean number of moves of around 90 ± 5.



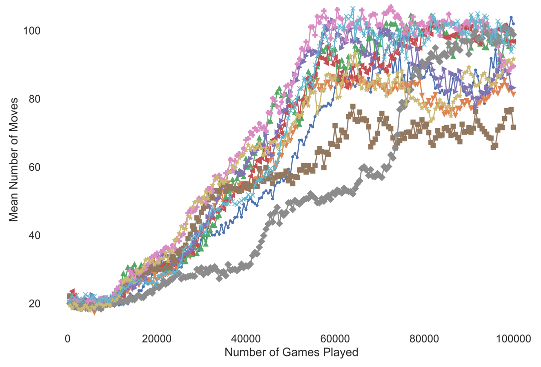


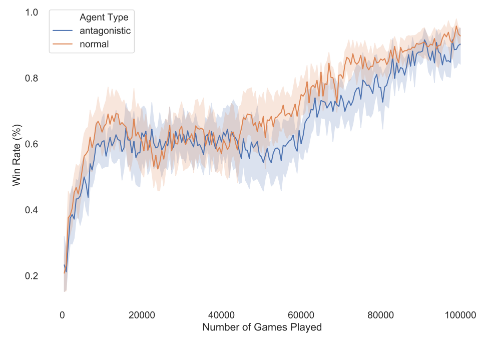
Figure 3: Breakthrough Training win rate (**top**) and number of moves (**bottom**) of each antagonistic agent. We see that all agents learn to increase the amount of moves per game, but some agents fail to learn how to win.

4. Record the win.
5. Repeat process until you reach the amount needed for your sample size (100 games).

This tests the ability of the agent's policy to pick the actions that lead to the highest state-value over many different scenarios without forcing ϵ to be nonzero. Hence, the agents will always pick what they consider to be the best action at the current state.

The justification for measuring move count is the following: if your objective is to make the victim's path to winning as arduous as possible, how could an observer measure such behavior? To win in Breakthrough, the player must either (1) have one pawn reach the opposite side of the board, or (2) capture all of the opponent's pawns. The trained victim agent tends to choose the first strategy. Hence, to make things difficult for the victim, the antagonist would want to completely focus on blocking the victim's pawns from reaching the other side. As stated previously, one agent was able to increase the number of moves without improving win rate, providing evidence that the agents are learning antagonistic behavior and not self-interested behavior.

Connect-4 Experiments and Multilevel Training In Figure 4, our initial results for Connect-4 are consistent with Theorem 2 in that a weaker victim and its corresponding antagonist will exhibit similar performance. One reason for this is because the victim agent is still a relatively weak agent when trained using the same methodology as the other games. To address this complication, we used a form of multilevel training in this game. To accomplish this, the victim is trained 500,000 games per stage for 3 stages. After each stage, a new antagonistic agent trains against a static version of the victim. As seen in Figure 5, more thoroughly training the agent using our multilevel approach seems to have led to an increase in the number of moves for the antagonistic agent. These results provide evidence for our theorems in that the antagonists perform similarly to weaker victims, while training against stronger victims lead to the antagonist increasing the amount of moves played during the game.



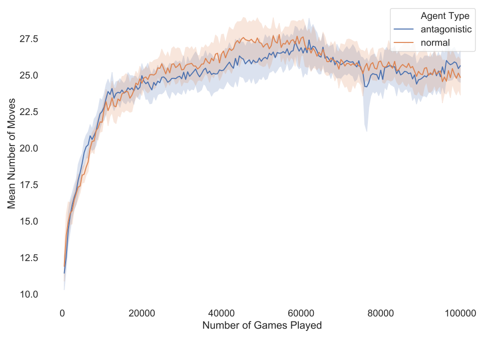


Figure 4: Connect-4 Training Results: **Top**: The win rate for both agents are about the same. **Bottom**: The mean number of moves for both agents are about the same.

Go Experiments Initially, we tried measuring antagonistic behavior of the board game Go by counting the number of moves, but realized such measurements seemed highly dependent on the current state. As mentioned previously in Section 3.2, it's possible that the antagonist's most valued position and the victim's most valued position on the board are the same. Hence, the context of the state matters in this game when evaluating antagonistic behavior. It is unclear

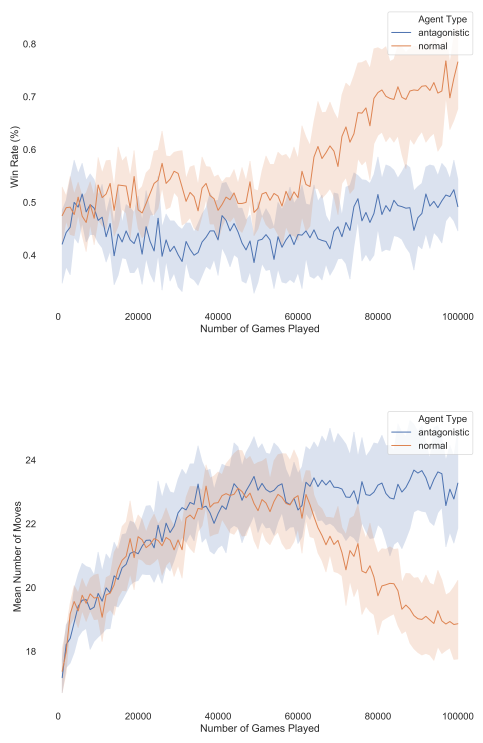


Figure 5: Multi-level Connect-4 Training Results: **Top**: The win rate for the normal agents is higher than the antagonistic agents. **Bottom**: The mean number of moves for the antagonistic agents is higher than the normal agents.

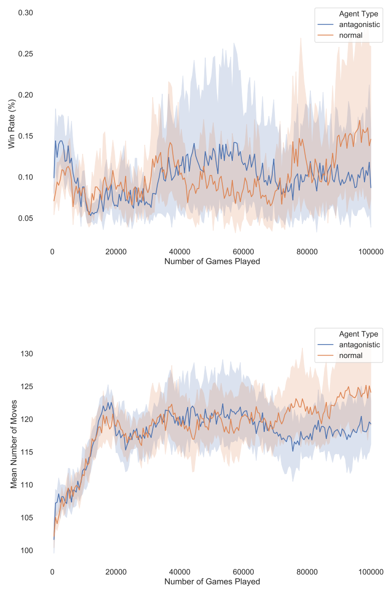


Figure 6: Go Training Results:
Left: The win rate for the normal agents is higher than the antagonistic agents.
Right: The mean number of moves for the normal agents is higher than the antagonistic agents.

if a simple metric can represent such behavior without the aid of a reasonably skilled human Go player evaluating each state. Such a metric would need to account for the territory the victim gained if the antagonist had not made the previous move, which is a counterfactual. Merely counting the number of moves is insufficient since (1) the artificial Go agents keep playing until there are no legal moves left, which tends to be somewhat constant in Go, and (2) intuitively, antagonistic behavior in Go would be seen as preventing the victim from gaining territory. This underscores the need for a methodology where the experimenter first chooses a metric for antagonistic behavior and uses evidence, formal or experimental, to show that the chosen metric adequately represents such behavior.

In Figure 6, we also found that, under the same training evaluation as Breakthrough, the antagonistic Go agent had lower and more variable win rate. This result is likely due to Go's inherent difficulty and complexity relative to Breakthrough. It also shows that an antagonist learning to minimize the victim's state-value is not a guarantee that the antagonist will learn an effective policy. That is, if an antagonist wants to cheat in a game, having access to the victim's state-values alone is not necessarily a winning strategy.

Discussion

Our work is aligned with the intent of past research in adversarial machine learning in that it exposes a problem that may arise when a malicious actor is introduced. The problem we attempt to model is when a human architect builds an antagonistic agent that can compete against a well-trained victim agent without access to environment rewards. We accomplished this by considering an idealized scenario where we formally define antagonistic behavior utilizing the victim's state-value function. From this definition, we investigated what behavior emerges by observing the change of certain metrics during the training progress of a group of antagonistic agents. We found that if we assume the number of moves represents antagonistic behavior, antagonistic agents show a significant increase in antagonistic behavior in games that encourage blocking strategies and with an appropriate level of difficulty. We also show that such an assumption is limited in games that currently have no simple, obvious metrics for antagonistic behavior (like Go). This demonstrates the inherent difficulty antagonistic architects might face when measuring progress even in an ideal case.

Our research introduces some possible future directions. One direction would be the observation of antagonistic agents in non-zero-sum games or environments consisting of more than two agents. Another direction, related to AI Safety and cybersecurity, would be to use the definition of antagonistic behavior as a more realistic model for adversarial RL than agents with access to the same reward function as its victim. Lastly, the antagonistic value function intuitively encapsulates how an architect, without knowledge of how the victim learned from the environment, would intend to create an antagonistic agent. In situations where this intuition reveals itself to be inaccurate, further research into why this intuition fails is worth investigating.

Acknowledgments We would like to thank Marc Lanctot for his assistance on OpenSpiel. This work was funded by the National Geospatial-Intelligence Agency and approved for public release as document 20-488.

References

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; and Mané, D. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* .

Bellemare, M.; Dabney, W.; Dadashi, R.; Taiga, A. A.; Castro, P. S.; Le Roux, N.; Schuurmans, D.; Lattimore, T.; and Lyle, C. 2019. A geometric perspective on optimal representations for reinforcement learning. In *Advances in Neural Information Processing Systems*, 4360–4371.

Berger, U. 2007. Brown's original fictitious play. *Journal of Economic Theory* 135(1): 572–578.

Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Debiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv preprint arXiv:1912.06680* .

Brown, N.; and Sandholm, T. 2019. Superhuman AI for multiplayer poker. *Science* 365(6456): 885–890.

Fridman, L. 2019. Gary Marcus: Toward a Hybrid of Deep Learning and Symbolic AI — Lex Fridman Podcast #43. URL https://youtu.be/vNOTDn3D_RI?t=4691.

Gleave, A.; Dennis, M.; Wild, C.; Kant, N.; Levine, S.; and Russell, S. 2020. Adversarial Policies: Attacking Deep Reinforcement Learning. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=HJgEMpVFwB.

Huang, S.; Papernot, N.; Goodfellow, I.; Duan, Y.; and Abbeel, P. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* .

Huang, Y.; and Zhu, Q. 2019. Deceptive reinforcement learning under adversarial manipulations on cost signals. In *International Conference on Decision and Game Theory for Security*, 217–237. Springer.

Lanctot, M.; Lockhart, E.; Lespiau, J.-B.; Zambaldi, V.; Upadhyay, S.; Pérolat, J.; Srinivasan, S.; Timbers, F.; Tuyls, K.; Omidshafiei, S.; et al. 2019. Openspiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453* .

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, 157–163. Elsevier.

Manheim, D. 2019. Multiparty Dynamics and Failure Modes for Machine Learning and Artificial Intelligence. *Big Data and Cognitive Computing* 3(2): 21.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533.

Pan, X.; Seita, D.; Gao, Y.; and Canny, J. 2019. Risk averse robust adversarial reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, 8522–8528. IEEE.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2817–2826. JMLR.org.

Shapley, L. S. 1953. Stochastic games. *Proceedings of the national academy of sciences* 39(10): 1095–1100.

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. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529(7587): 484.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575(7782): 350–354.

Zhang, X.; Ma, Y.; Singla, A.; and Zhu, J. 2020. Adaptive Reward-Poisoning Attacks against Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*. URL https://proceedings.icml.cc/static/paper_files/icml/2020/4819-Paper.pdf.

Appendix

Theorems and Proofs

Theorem .2. Assume a finite, turn-based, zero-sum, deterministic game with no intermediate rewards and let ν have the static, optimal value function V^*_ν. If ν wins, $0 < \gamma < 1$, ν's policy is greedy, and n is the number of time steps (or game moves) left at state s to traverse and win the game, then $n = \log_\gamma V^*_\nu(s)$.

Proof. Assume at the end of the game, the winning agent gets 1 point and the losing agent gets -1 point. From these assumptions, $V^*_\nu(s_{win}) = 1$, where s_{win} is the winning state

for ν.
If i is the number of steps left to traverse at state s, it suffices to show that $V_\nu^*(s) = \gamma^i$. This can be proven by induction on the number of steps left for the greedy agent to reach the winning terminal state using the optimal value function.
Base case: If there is 1 step left, the greedy agent will choose the action that leads to s_t such that $V_\nu^*(s_t) = 0 + \gamma * V_\nu^*(s_{win}) = \gamma * 1 = \gamma^1$.
Inductive case: Assume $V_\nu^*(s') = \gamma^i$, for all states s' such that i is the number of steps left. Let s be a state such that the greedy policy chooses the action that leads to s' from s and there are $i+1$ steps left to traverse. Since there are no intermediate rewards, we have $V_\nu^*(s) = 0 + \gamma V_\nu^*(s') = \gamma * \gamma^i = \gamma^{i+1}$.
Hence, if i is the number of steps left to traverse at state s, $V_\nu^*(s) = \gamma^i$, which implies $i = \log_\gamma V_\nu^*(s)$. □

Implementation Details

Algorithm 1: Antagonistic Deep Q-Learning

```
Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random
  weights
for episode = 1, M do
    Initialise state $s_1$
    for t = 1, T do
        Observe new victim value $V_\nu(s_t)$
        With probability $\epsilon$ select a random action $a_t$
        Otherwise select $a_t = \arg\max_a Q^*(s_t, a; \theta)$
        Execute action $a_t$ in emulator and observe
          new state $s_{t+1}$ and new victim value
          $V_\nu(s_{t+1})$
        Set $v_t := -(V_\nu(s_{t+1}) - V_\nu(s_t))$
        Store transition $(s_t, a_t, v_t, s_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions
          $(s_j, a_j, v_j, s_{j+1})$ from $\mathcal{D}$
        $y_j =$
          $\begin{cases} v_j & \text{for terminal } s_{j+1}, \\ v_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta) & \text{for non-terminal } s_{j+1}. \end{cases}$
        Perform a gradient descent step on
          $(y_j - Q(s_{j+1}, a_j, \theta))^2$
```

Hyperparameters

Neural Network Architecture Hyperparameters	
Number of Hidden Layers	6 (for Go and Connect-4) 7 (for Breakthrough and Havannah)
Number of Units	256
Activation Function Used After Each Hidden Layer	ReLU
Batch Normalization Applied After Activation Function	True

RL Training Hyperparameters	
Optimizer	Adam
Learning Rate	10^{-6}
Discount Factor γ	0.95 (for victim) 1.0 (for antagonist)
Batch Size	256
Replay Buffer Capacity	10^{5}
Gradient Update every...	10 games
Update Target Value Network every...	1000 games
Evaluate Progress every...	500 games