Playing Games to Learn Robustly From Adversarial Expert Demonstrations

Prithviraj Dasgupta
Distributed Intelligent Systems Section (Code 5583)
Information Technology Division
U. S. Naval Research Laboratory
UNITED STATES OF AMERICA

raj.dasgupta@nrl.navy.mil

ABSTRACT

Modern day warfare is characterized by increasing complexity as well as by smart and technologically adept enemies. To tackle some of the complexities of modern warfare, machine learning (ML) based techniques have recently provided suitable means to automate tasks on the battlefield. However, smart enemies equipped with ML techniques not only engage in fair competition on the battlefield, but craft malicious methods using strategies like deception and covert attacks to break ML algorithms and gain an unfair advantage. To counter these threats, ML techniques used on automated battlefield systems must be made robust against adversarial attacks.

We analyse the problem of adversarial learning in a competitive scenario in the context of a reinforcement learning algorithm called Learning from Demonstrations (LfD). In LfD, a learning agent observes demonstrations of operations done by an expert to learn to perform a task quickly and effectively. LfD has been used successfully in military operations such as autonomous search and reconnaissance using teams of robots, or, autonomous grasping to deactivate improvised explosive devices. However, malicious enemies could exploit LfD by planting adversarial experts that either give incorrect demonstrations or modify legitimate demonstrations to make the learning agent fail in its task. To address this problem, we first analyse different demonstration modification strategies that could be used by an adversarial expert within the LfD framework, in terms of the modification costs expended by the adversary and the degradation in task performance effected by the modification on the learning agent. We then propose a novel concept using game-playing between the adversary and the learning agent that can be used by the learning agent to strategically learn from potential adversarial expert demonstrations via LfD without significantly degrading its task performance. We present evaluations of our proposed techniques for robust learning with adversarial modifications of expert demonstrations in an Atari-like game called LunarLander within the AI-Gym environment.

1.0 INTRODUCTION

Consider a state-of-the-art, military-grade missile defence system - precise sensors, top-of-the-line hardware components and sophisticated algorithms are used to ensure that it operates decisively and accurately to thwart enemy intrusions. The algorithms driving most modern military systems strategically employ artificial intelligence (AI) enabled machine learning (ML) algorithms to enhance their performance by making autonomous decisions in complex situations and gain advantage over the enemy. For example, a missile defence system might use ML in fog-of-war to predict the track of an incoming missile with razor sharp accuracy even when there is sparse and noisy information available about the missile's launch coordinates or when the missile's trajectory is observable intermittently or poorly. One of the popular methods used to enable ML techniques

operate autonomously is reinforcement learning (RL) and learning from demonstration (LfD)[1]. The LfD concept draws inspiration from a teacher-student setting where a teacher gives demonstrations of the correct way to perform a task to a student. The student then mimics the teacher's demonstrations, while adapting the demonstrations suitably to account for small variations in the task. The main advantage that LfD affords is that observing demonstrations from an expert teacher enables the student to learn the task more quickly and effectively as compared to learning via self-explorations of doing the task without the teacher. In AI-based LfD, a learning agent observes demonstrations of operations given by an expert in the form of trajectories (observation-action sequences) to learn to perform a task. LfD has been successfully used in military operations such as autonomous search and reconnaissance using teams of robots, and, autonomous grasping to deactivate improvised explosive devices [2].

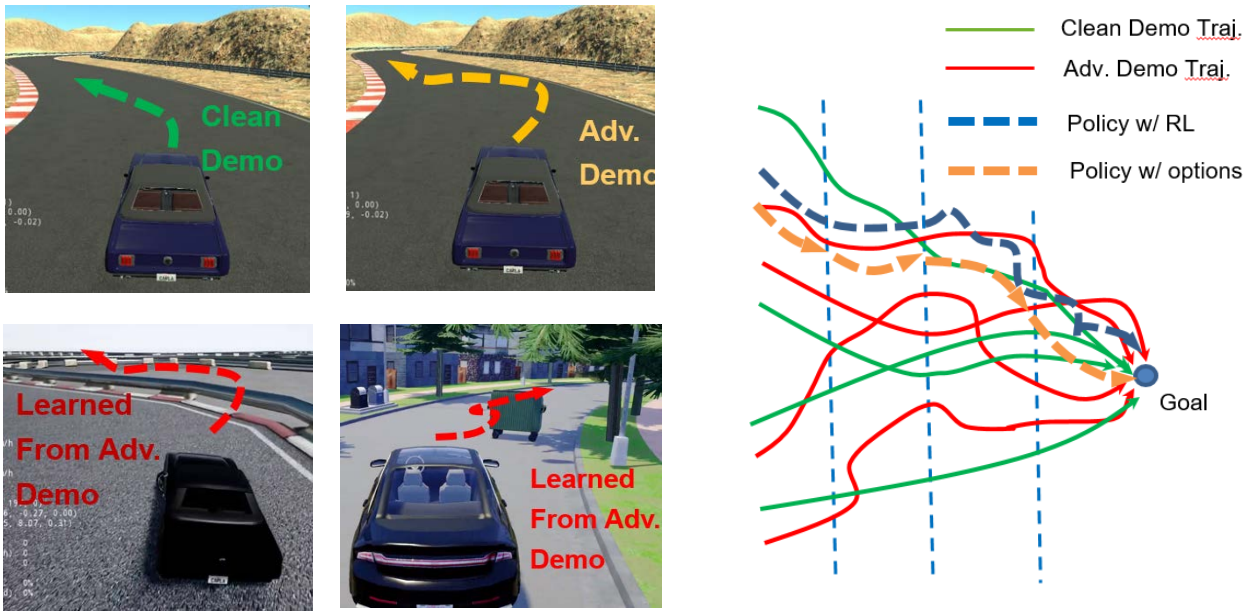

Figure 1. (Left) Effect of adversarial trajectories on policy learned using LfD for an autonomous driving setting. (Right) In our proposed approach, clean (green) and adversarial (red) trajectories are first equi-partitioned. Policies are then learned for each partition after accepting or rejecting trajectory parts using options (golden dashed lines) or using conventional reinforcement learning for the un-partitioned trajectories (blue dashed line)

In the real-world, however, adversaries pose a major threat to AI-based systems using LfD: malicious entities could exploit LfD by planting adversarial experts that either give incorrect demonstrations or modify legitimate demonstration data (for example, by man-in-the-middle attacks) to make the learning agent fail in its task. An example is illustrated in Figure 1, where adversarial demonstrations could lead to an incorrectly learned policy for a driving manoeuver and result in crashing an autonomously driven vehicle. Researchers [5, 6] have similarly shown that noisy or incorrect demonstrations could be successfully used by an adversary to misguide an LfD-based learning agent to learn and perform incorrect actions such as falling down while trying to perform rather simple tasks like walking or kicking a ball. Simultaneously, researchers have also started to investigate solutions to the adversarial LfD problem. Principal solution approaches proposed include augmenting demonstration trajectories used by the learning agent with a noise variable to represent possible deviations [3], or incorporating a risk term in the learning framework to capture the effect of incorrect trajectories on the learned task [4]. However, existing approaches to adversarial LfD have certain important limitations: adversarial demonstrations have either to be marked or labelled appropriately so that the learning agent avoids learning from them, or, for

commensurate demonstrations, the number of correct trajectories must far exceed the number of adversarial trajectories, so that the adversarial demonstration's effect on the learned task is minimal. However, both these requirements are difficult to realize in real-life military problems as it is infeasible to label demonstration data as clean versus malicious, or, acquire large quantities of demonstration data, clean or adversarial.

In this paper, we propose a novel technique that can be used by a learning agent to build robustness against adversarial demonstrations within LfD using a game-like framework between the learning agent and demonstrator, without requiring large quantities or correct versus adversarial labels on demonstration data. We have presented a preliminary evaluation of the proposed technique using an Atari-like game called LunarLander within the AI-Gym environment. Our results show that using our proposed technique a learning agent can successfully learn to land the LunarLander in the presence of adversarial demonstrations.

2.0 ADVERSARIAL EXPERT DEMONSTRATIONS FRAMEWORK

We consider a scenario where a learning agent has to perform tasks within an environment by learning via reinforcement learning from demonstrations (LfD) of the tasks given by experts. Some of the experts could be adversarial and modify the trajectory demonstrations with an intention of making the learning agent fails to perform the task correctly while following the modified demonstration. In the rest of the paper, we refer to adversarial experts as experts for the sake of legibility. The LfD framework is formalized using a Markov Decision Process (MDP) [12]. The output of the LfD algorithm is a policy that gives a state to action mapping for performing a task. RL learns a policy via a process called training during which it explores the environment, observes the state-action-reward pairings received during the exploration, and finally selects sequences of state-action-reward pairings that lead to higher expected rewards as its policy.

Demonstrations from experts are given in the form of sequences of state-action-reward tuples called trajectories. Expert trajectories could be either benign or adversarial. Benign and adversarial expert trajectories respectively demonstrate the correct and incorrect way of doing the task and either aid or hinder the learning agent to learn to perform the task. Expert demonstrations are incorporated into the agent's learning to perform the task using an LfD algorithm called DAGGER[1]. DAGGER uses supervised learning from expert-demonstrated trajectories to learn a policy but adds a weight parameters, β, that denotes the learning agent's weight or trust in incorporating a trajectory into its learned policy.

2.1 Trajectory Modification

To modify a set of trajectories, an adversarial expert adds noise at strategic locations within a clean set of trajectories. The noise could be added in the states, actions or rewards of the trajectory data. The strength, σ_{adv}, of the adversary determines the number of clean trajectories that are modified within the clean trajectory set. Evidently, a stronger adversary could distort clean trajectories quickly and misguide the learning agent more aggressively. However, highly modified trajectories are also easier to detect as malicious by the learner. Therefore, it makes sense for the adversary to modify trajectories strategically so that the modified trajectory has the desired effect of misguiding the learner, while simultaneously evading detection by the learner. To achieve this balance between modification and evasion, we propose to use a trajectory modification technique on the adversary that is inspired by adversarial text modification [14]. Here, the adversary calculates the gradient of the reward function with respect to states within a trajectory, selects the states within the trajectory that have the maximum and minimum gradient and replaces the state with the maximum reward gradient with the one that has minimum reward gradient, along with the action for the replaced state. This operation is repeated for a certain number of iterations and for a certain number of trajectories corresponding to the strength of the adversary. The

intuition behind this trajectory modification technique is that regions in the trajectory that have higher reward gradients have higher contributions towards the optimality of the policy[1] learned by the learning agent. If the states in these regions could be replaced by states that correspond to lower gradients, the resulting policy learned will be further from optimal and likely result in misguiding the learning agent towards failing to perform its task.

The adversarial trajectories are injected into the expert trajectory set used to train the learner's LfD algorithm. The problem facing the learner is to selectively learn from clean or valid trajectories while discarding adversarially modified trajectories. This task is not straightforward, as trajectory data cannot be labelled as clean versus adversarial by humans via inspection. Neither can statistical divergence between trajectories offer a reliable method for detecting adversarial trajectories. Because there might be few, albeit clean trajectories that demonstrate difficult manoeuvers in less-frequented parts of an environment – giving high divergence from frequently demonstrated, clean trajectories, yet being valid demonstrations of the task at hand.

To address the problem of detecting adversarial trajectories, we first observe that adversarial trajectories might not be adversarial in their entirety; there could be portions of an adversarial trajectory that are clean and valid demonstrations and could be used by the learner to improve its task learning. It makes sense for the learner to retain and learn from the clean portions of trajectories and discard the adversarial portions. The decision problem facing the learner can be written as: given a trajectory that is partitioned into parts, for every part, decide whether to accept and learn from it or discard it.[2] On the other hand, the problem facing the adversary is to determine an appropriate modification strength, σ_{adv}, so that it can fool the learner into making an incorrect decision to retain and learn from an adversarial trajectory, or part, thereof. To achieve this, we formulate the interaction between the expert and learner as a 2-player game, as described below.

[1] We follow the conventional definition of an RL policy that finds state-action sequences that maximize expected rewards [11].

[2] We assume that the length of the adversary's modifications to a trajectory is much smaller than the size of a partition, and, consequently, the adversarial part of a trajectory is limited within a single partition.

3.0 GAME-PLAYING TO COUNTER ADVERSARIAL DEMONSTRATIONS

The game between the expert and learner consists of rounds, at each round the expert makes the first move followed by the learner. During its move, the expert selects an adversarial strength value, σ_{adv}, corresponding to the amount of modification it plans to do on the trajectories. σ_{adv}, = 0 corresponds to no trajectory modification, while σ_{adv}, = 1 represents all data in the trajectory is modified. The exact value of σ_{adv}, is not revealed to the learner. The learner moves next: it receives the trajectories from the expert, selectively accepts of rejects trajectories based on trajectory acceptance threshold values, denoted by o_{Thr} and f_{Thr}, and updates its policy from the data in the accepted trajectories. For the policy update, the learner uses the options framework [9] that first partitions a long horizon trajectory into smaller trajectories, then uses DAGGER algorithm to learn a policy from trajectories in each partition, and finally connects the policies in successive partition using a technique called policy chaining. A schematic for the learner's policy update method is illustrated in Figure 1 (Right). The expert (learner) wins the game if the learner accepts (discards) to learn from the majority of the trajectories that the expert had modified. The game then proceeds to the next round. The expert and learner use the history of wins and losses to determine an appropriate adversarial strength value of σ_{adv} and, learner threshold values, o_{Thr} and f_{Thr}, for the next round, respectively. Algorithms 1 and 2 show the pseudo-code of the algorithms used by the learner and expert for playing the game.

3.1 Measuring Trajectory Distances

A critical aspect of the learner's decision is its ability to measure distances between a known, clean set of trajectories and a newly demonstrated trajectory. We propose to use two metrics for this: (a) **Occupancy Measure,** which gives the fraction of times different states are visited in a set of trajectories. Note that the occupancy measure only relates to the states that were visited by a trajectory but does not account for the order in which they are visited; and, (b) **Fréchet Distance,** which gives a real-valued distance or similarity metric between a pair of curves, while accounting for the location and ordering of the points along the curves The Fréchet distance measures the maximum distance between possible couplings of pairs of points along two curves. The minimum possible Fréchet distance value is 0 when two curves coincide with each other higher values denote the degree of divergence between the curves.

Input: T_C: set of known, clean trajectories
T_d: set of demonstrated trajectories
Output: Policy learned from demonstrated trajectories

$\tau_{Acc} = \{\}$ // set of accepted trajectory partitions
$\{T_{c,i}\}$ = Divide T_C into N equal partitions
// learn policy for partition i
Π_i = policy learned from $T_{C,i}$ using DAGGER algorithm

Update o_{Thr}, f_{Thr} based on game outcomes history
for each demo trajectory τ_d in T_d :
  $\{\tau_{d,i}\}$ = Divide trajectory τ_d into N equal partitions
  For each partition $\tau_{d,i}$:
    o = occupancy measure between $\tau_{d,i}$ and $T_{C,i}$
    f = average Fréchet distance between $\tau_{d,i}$ and $T_{C,i}$
    if ($o < o_{Thr}$ AND $f < f_{Thr}$)
      add $<\tau_{d,i}, \beta_{hi}>$ to τ_{Acc}
    elif ($o < o_{Thr}$ OR $f < f_{Thr}$)
      add $<\tau_{d,i}, \beta_{mid}>$ to τ_{Acc}
    else
      discard $\tau_{d,i}$

Update learned policy Π_t using $<\tau_{d,i}, \beta_*>$ using DAGGER algorithm
Chain partitioned policies Π_t into single policy Π
return Π

Algorithm 1. Algorithm used by the learner to accept or reject trajectory demonstrations.

Input: T_C: set of known, clean trajectories
Output: T_d: set of demonstrated trajectories

Update adversarial strength parameter σ_{adv}, $0 <= \sigma_{adv} <= 1$, based on game outcomes history
T_d = Select fraction σ_{adv} of trajectories from T_C
for each trajectory τ_d in T_d :
  Δ = Reward gradients w.r.t. state in τ_d
  for n_{flips} iterations:
    $s_{min}, s_{max} = arg\ min_S(\Delta), arg\ max_S(\Delta)$
    $s_{max} = s_{min}$ in τ_d
    overwrite action for replaced state s_{max} with $a' \neq \Pi(s_{max})$
return T_d

Algorithm 2. Algorithm used by the expert to modify clean trajectories

4.0 EXPERIMENTAL RESULTS

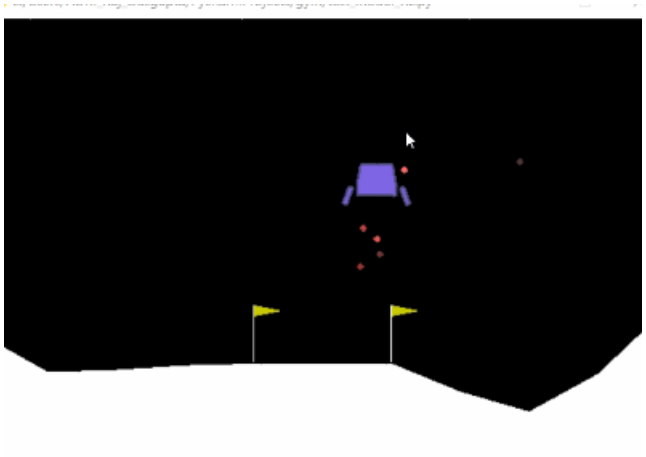

Figure 2. A snapshot of the LunarLander environment used for our experiments.

We report some preliminary results from our proposed game-based technique for learning with adversarial trajectories. For our evaluations, we used the discrete LunarLander environment available within AI Gym. The problem consists of landing an airborne two-legged spacecraft at a specific location called the landing pad within a 2D environment akin to the surface of the Moon. A snapshot of the game environment is shown in Figure 2. The state space consists of an 8-dimension vector given by the 2-D coordinates of the center of the spacecraft, 2-D linear velocity, orientation and angular velocity and whether both legs of the spacecraft are on the ground. The initial state of the spacecraft consists of random coordinates towards the top of the environment and random initial velocity. The action space of the spacecraft consists of four actions: to fire its main, left or right engines or do nothing (no-op). The agent receives a reward of 320 points of landing on both legs on the landing pad, a penalty of -100 points for crashing, while manoeuvring the spacecraft incurs a penalty of -0.3 for using the main engine and -0.03 for the left or right engine For our baseline reinforcement learning algorithm we used the deep Q -network (DQN) algorithm available via stable baselines. The algorithms were implemented using the following open source libraries: Tensorflow 1.15, OpenAI Gym 0.18 and stable baselines 2.10. The clean trajectory data set consisted of 1000 trajectories generated using stable baselines – generate-expert-trajectory method, each trajectory contained about 200-300 state-action-reward sequences.

Table 1 shows the effect of different adversarial strength values, σ_{adv}, on the quality of the task performed by the learner after learning from the trajectory data, measured in terms of the reward received by the learner. As shown in the table, higher values of σ_{adv} result in poorer task performance by the learner. The learner's performance starts to deteriorate from around $\sigma_{adv} > 0.3$, and it loses its ability to correctly perform the task of landing the LunarLander on the landing pad even once, for $\sigma_{adv}, >= 0.54$.

Table 1. Learner rewards for different amounts of adversarial trajectory modifications

Adversarial Strength (σ_{adv})	*Average Reward*	*Standard Deviation*	*Median Reward*	*Maximum Reward*	*Minimum Reward*
Clean	*255.442*	*59.684*	*269.892*	*321.466*	*4.208*
0.09	*249.317*	*72.815*	*273.523*	*317.399*	*-24.531*
0.27	*223.216*	*108.883*	*269.182*	*319.449*	*-205.077*
0.36	*12.594*	*116.8456*	*10.6957*	*307.636*	*-348.0903*
0.54	*-223.1203*	*152.946*	*-197.742*	*47.818*	*-645.412*
0.81	*-492.673*	*199.096*	*-569.918*	*55.339*	*-773.953*

Figure 3 shows a comparison of the rewards received by the learner agent without (left) and after (right) using our proposed game based framework. Overall, the learner using the proposed game framework is able to recuperate from failing to perform the task due to adversarial modifications, as reflected in the higher rewards.

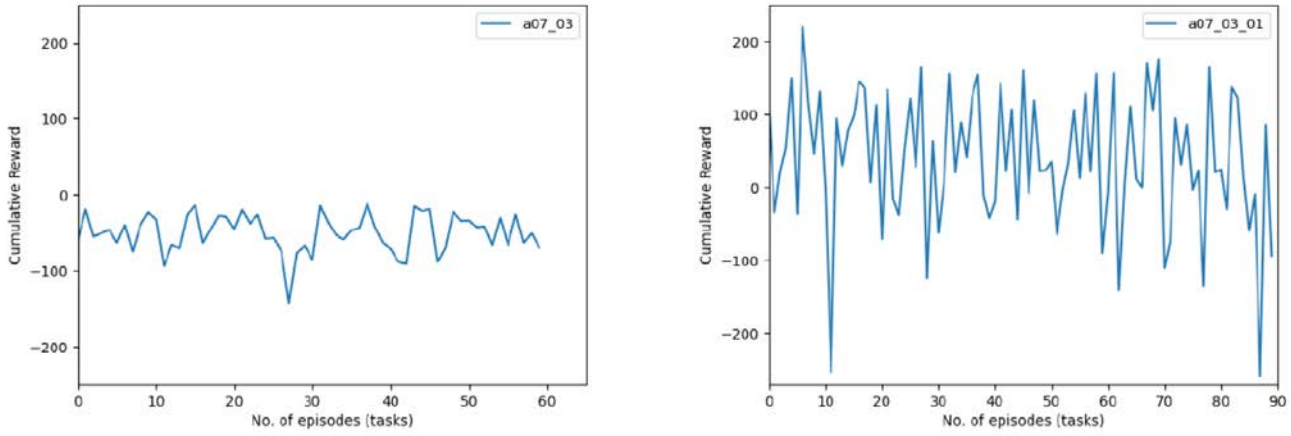

Figure 3. Rewards received by learner agent before (left) and after (right) using proposed game based framework.

5.0 CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we described a novel approach using a game based framework to build robustness against adversarial expert demonstrations that could be used by a learning agent to learn discerningly from potentially adversarial experts. This is our first step in this direction and there are several interesting directions to explore. A few planned directions include calculating the parameter updates of σ_{adv}, o_{Thr} and f_{Thr} done by the expert and learner at the end of each round using game theory based techniques like Nash equilibrium or counterfactual

regret; not revealing the game outcome – win or loss, to the expert, rather have the expert infer the game outcome by observing the learner's performance or rewards from the task. Recently, several frameworks relevant to our problem have been proposed including generalized adversarial imitation learning (GAIL) [7] and option-critic architecture [8]. These frameworks could be integrated with our proposed game-based framework for adversarial LfD to improve its performance. We believe that further exploration of these topics along the directions proposed in this paper will enable robust, reliable and effective machine learning for military applications.

REFERENCES

[1] Osa, T., Pajarinen, J., Neumann, G., Bagnell, J.A., Abbeel, P. and Peters, J., 2018. An algorithmic perspective on imitation learning. Foundations and Trends in Robotics, 7(1-2), pp.1-179.

[2] Argall, B.D., Chernova, S., Veloso, M. and Browning, B., 2009. A survey of robot learning from demonstration. Robotics and autonomous systems, 57(5), pp.469-483.

[3] Pan, X., Seita, D., Gao, Y., and Canny, J., "Risk averse robust adversarial reinforcement learning," in [2019 International Conference on Robotics and Automation (ICRA)], 8522-8528, IEEE (2019).

[4] Mandlekar, A., Zhu, Y., Garg, A., Fei-Fei, L., and Savarese, S., "Adversarially robust policy learning: Active construction of physically-plausible perturbations," in [2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)], 3932-3939, IEEE (2017).

[5] Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A., "Robust adversarial reinforcement learning," in [Proceedings of the 34th Intl. Conf. Machine Learning, ICML 2017], 2817-2826 (2017).

[6] Gleave, A., Dennis, M., Wild, C., Kant, N., Levine, S., and Russell, S., "Adversarial policies: Attacking deep reinforcement learning," arXiv preprint arXiv:1905.10615 (2019).

[7] Ho, J. and Ermon, S., "Generative adversarial imitation learning," in [Advances in neural information processing systems], 4565-4573 (2016).

[8] Bacon, P.-L., Harb, J., and Precup, D., "The option-critic architecture," in [Proceedings of the AAAI Conference on Artificial Intelligence], 31(1) (2017).

[9] Sutton, R. S., Precup, D., and Singh, S., "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," Artificial intelligence 112(1-2), 181-211. (1999).

[10] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. Advances in neural information processing systems, 27.

[11] Ebrahimi, J., Rao, A., Lowd, D. and Dou, D., 2018, July. HotFlip: White-Box Adversarial Examples for Text Classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 31-36).

[12] Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.