**Utilizing open-source reinforcement learning environments for implementation of Deep Q-learning**

Colby L. Wilkinson, Mark Dalton

Naval Undersea Warfare Center, Division Newport

**Abstract**

In recent years Reinforcement Learning (RL) algorithms have been used to train autonomous agents to complete tasks in complex environments at the proficiency level of human-users. Using RL in Navy specific contexts require simulating envornmental conditions that an autonomous agent uses to learn to make optimal decisions. In this paper we utilized an open-source environment, PuckWorld, from the Python Learning Environment module, to train an autonomous agent to navigate an environment with both targets and keep-out areas. We utilized Deep Q-Learning to train our agent. This exercise demonstrated the value in using open-sourced environments to set up our training process and model structure.

**Introduction**

Recent advancements in reinforcement learning (RL) has led to the development of autonomous agents able to compete with humans at complex games such as Chess, Go, and Texas hold ‘em1–3. While these rapid developments have surprised even the most optimistic proponents of RL, practical applications in real-world settings have not materialized at the same pace [cite]. One hindrance to widespread adoption of RL in the Navy is the requisite effort needed to replicate a high-fidelity simulation environment to train an autonomous agent. In this paper we utilize an open-source environment, PuckWorld, which is part of the Python Learning Environment [cite], to implement Deep Q-Learning (DQN). Leveraging this open-source environment allows us to experiment with different model inputs and architectures, alter reward structures and tune agent training procedures before investing resources in developing a custom environment for our problem space.

**Methods**

* Environment Description
  + Description
    - Screenshot
    - Parallels to Navy context
      * UxV navigation
      * Torpedo evasion
  + Observation Space
  + Rewards
  + Action Space
* DQN
  + Model Structure
  + Explore/Exploitation balance
  + Cost Function: minimization squared difference between Q-target and intermediate/future rewards

**Results**

* Model parameters
* Model train time/CPU specs
* Rewards plot

**Conclusion**

* Advantage: establish framework to train agent on real-world environement data using existing open-source environments. Allows to feasibility analysis.
* Future work: part of 219-funded work for RL applications to tactical decision making
  + Introduce more complex environment, alter reward structure to influcne desired behaviors, scale to tactical scenarios with input from stakeholders (UWDC Tag)

References

* “Reinforcement Learning, An Introduction” by Sutton & Barto (2nd ed.)
* <https://towardsdatascience.com/qrash-course-deep-q-networks-from-the-ground-up-1bbda41d3677>

1. Silver D, Hubert T, Schrittwieser J, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science (80- )*. 2018. doi:10.1126/science.aar6404

2. Ramirez A, Reinman S, Norouzi N. PokerBot: Hand Strength Reinforcement Learning. In: *IEEE International Symposium on INnovations in Intelligent SysTems and Applications, INISTA 2019 - Proceedings*. ; 2019. doi:10.1109/INISTA.2019.8778267

3. Silver D, Huang A, Maddison CJ, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*. 2016. doi:10.1038/nature16961