Double Deep Q-Learning for Autonomous Cyber Defense Agent Training

Ethan Morphew and Jie Gao
School of Information Technology, Carleton University, Ottawa, ON, Canada, K1S 5B6
Emails: {ethanmorphew@cmail.carleton.ca, jie.gao6@carleton.ca}

***Abstract*—This poster presents a DDQN-based approach for training an agent to defend networks against attackers of different types. By training the agent against adversaries with diverse strategies, we achieve consistently strong performance in network defense. Evaluations using the TTCP CAGE 2 challenge environment show that our approach achieves comparable or superior rewards to those on the global leaderboard of the challenge. In particular, our agent outperformed the "Ensembled DDDQN" and "PPO w/ RE3 Exploration" approaches by approximately 30% in the meander 100-step reward category and 11% in the b-line 100-step reward category, respectively.**

***Index Terms*—Autonomous Cyber Defense, Double Deep Q-Learning, Blue Agent Training, TTCP CAGE Challenge**

I. Introduction

Artificial intelligence (AI) is becoming an essential tool for improving the detection and response to cybersecurity threats [1]. Deep-reinforcement learning (DRL), which enables an agent to continuously learn from experience and dynamically adapt to evolving threats in a complex environment [2], has attracted extensive attention for autonomous cyber defense [3]. A defense (a.k.a. "blue") agent trained using DRL responds to attacks in real time, enhancing security while reducing the need for human intervention.

Double Deep Q-Learning (DDQN), an off-policy DRL method leveraging separate target and policy neural networks to decompose action selection and action evaluation, offers key advantages such as stability, sample efficiency, and low computational requirement [4]. Designed specifically for discrete action spaces, DDQN is well-suited for cyber defense applications, where decisions are typically represented as discrete variables [5]. In this poster, we present DDQN-based cyber defense agent training, evaluate it in a simulated environment, and demonstrate its performance in comparison to leading AI-based training approaches.

II. Network Scenario and Agent Design

An example network scenario is illustrated in Fig. 1. The network includes user and operational hosts, enterprise servers, and an operational server. A blue agent (defender) and a red agent (attacker) exist in the network. The red agent's objective is to compromise the operational server, while the blue agent's primary goal is to defend it. Additionally, the blue agent also protects the hosts and enterprise servers.

The DDQN algorithm is tasked with analyzing the state and choosing a course of action that will generate the highest reward. To achieve the goal, the DDQN determines Q-values, which are the expected reward for a given state-action pair. The Q-value is updated as follows [6]:

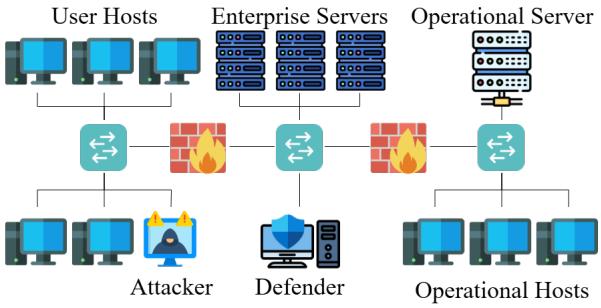

Fig. 1: Network Scenario for Cyber Defense Agent Training.

$$Q(s_t, a_t; \boldsymbol{\theta}_t) = r_{t+1} + \gamma Q\left(s_{t+1}, arg\max_a Q(s_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t^-\right)$$

where $\boldsymbol{\theta}_t$ and $\boldsymbol{\theta}_t^-$ are weights of the policy and the target neural networks, respectively, r_{t+1} is the immediate reward transitioning from state s_t into state s_{t+1} as a result of action a_t, and γ is the discount factor.

In our DDQN-based agent training, we exclude actions that rely on prior knowledge of specific red agent strategies, such as pre-placing decoys along the known path of the red agent. By excluding these actions, we can more realistically simulate an unexpected attack on the network with the additional benefit of speeding up the training by limiting the number of state-action combinations. Our agent is trained with the same DDQN on different types of red agents, randomly selecting the type of the red agent at the beginning of each episode. This creates a blue agent that can effectively defend against different types of red agents rather than specializing against one specific type.

III. Experiment Setup

To test our DDQN-based approach for training an autonomous cyber defense agent, the TTCP CAGE 2 environment designed with the purpose of testing blue agents for network defense was used [7]. As shown in Fig. 1, hosts are split into 3 networks, with subnet 1 holding the user hosts, subnet 2 holding the enterprise servers, and subnet 3 holding the operational hosts and server. The red agent in the environment is either of two types with different behaviors. The meander agent is representative of an attacker with no prior knowledge of the network and will seek to explore the network and gain access to a number of hosts. The b-line agent represents an attacker with intimate knowledge of the network topology and takes a direct path toward the operational server.

From the perspective of the blue agent, the simulated environment is represented by a 52-bit state space and 145 discrete actions representing all actions possible on all hosts.

TABLE I: Hyperparameters used for training

Hyperparameter	Value
Replay memory	25000
Discount factor (γ)	0.99
Epsilon decay	0.0002
Target network update rate (τ)	0.005
Learning rate (α)	0.0001

Fig. 2: Convergence of the DDQN in training the blue agent.

In our approach, we exclude one of the categories of actions, *decoys*, which reduces our agent's action space to 40 actions, 3 per host. The state space represents the blue agent's knowledge of each host, including current red agent action taking place on the host that the blue agent is aware of and whether or not the host has been compromised. The blue and red agents each take a single action per step, with actions being taken simultaneously at the end of the step. In each step, the blue agent chooses from the 40 actions with the following being the basic action types on each host i) *analyze*, which aims to detect the presence of the red agent, if any, within a host, ii) *remove*, which removes the red agent, with a smaller penalty, but only if the red agent has not achieved root access, or iii) *restore*, which can remove the red agent with a larger penalty. An episode, which consists of 30, 50, or 100 steps, simulates an attack on the network for a given duration.

IV. Experiment Results and Comparisons

Next, we demonstrate the performance of our DDQN-trained agent in comparison to several benchmarks. The hyperparameters used for training are given in Table I. The benchmark blue agents selected for comparison similarly avoided actions requiring prior knowledge of specific red agent policies (e.g., decoy deployment based on knowledge of the b-line agent's normal path), which is less applicable to real-world scenarios. We selected the"PPO w/RE3 Exploration" agent, ranked 16th in the CAGE 2 challenge leaderboard and the "ensemble DDDQN" agent, ranked 17th, as the main benchmark. A heuristic agent and a random agent provided by the challenge were also compared. The 100-step episode length was selected as it represents a more realistic attack scenario with a sustained red agent effort to impact the operational host. The reward is always negative, as it characterizes the negative impact of the actions of the red agent on the network.

TABLE II: Performance of agents in 100-step reward

Agent	B-line 100-step	Meander 100-step
Our Agent	-48.12	-48.10
PPO w/ RE3 Exploration	-54.01	-28.69
Ensembled DDQN	-23.96	-67.89
CCS Heuristic	-184.34	-192.63
CCS Random	-726.92	-566.41

Fig. 2 shows the convergence of the DDQN algorithm in training. The blue agent trained with the DDQN learned to take effective defense actions in a relatively short period of time. Specifically, the reward converges within only 5000 episodes of 100 steps, and the average 100-step reward improves from approximately -350 for both b-line or meander red agents before training to -48.12 and -48.10 for b-line and meander red agents, respectively.

Table II compares the performance of the blue agent trained by our DDQN algorithm with the benchmark agents. Each value in the table represents the 100-step reward averaged over 1000 episodes. Unlike the two AI-based benchmark agents, which perform significantly better for either the meander or the b-line red agent, our DDQN performs consistently for both types of the red agents. This is a result of training the same DDQN against both types of red agents. By randomly selecting the type of red agent on a per-episode basis, we force our agent to search for a strategy that performs well against both types without needing the DDQN to discern between red agent types. Notably, with the decoy actions excluded from the action space, our agent greatly outperforms the random agent even before the training.

V. Conclusions

We have demonstrated the potential of DDQN for training an cyber defense agent. In a modest number of training episodes, our agent was able to quickly discover and exploit a strategy and significantly improve the reward. The choice of randomly selecting the red agent type in training led to a more consistent performance of the trained blue agent against the meander and b-line agents, as compared to the two AI-based benchmarks. A future direction is to identify whether actions beyond decoys could be removed to further restrict the action space while improving the defense performance.

References

[1] N. E. Fard, R. R. Selmic, and K. Khorasani, "A Review of Techniques and Policies on Cybersecurity Using Artificial Intelligence and Reinforcement Learning Algorithms," *IEEE Technol. Soc. Mag.*, vol. 42, no. 3, pp. 57-68, Sept. 2023.

[2] A. A. Hammad, et al. "Deep Reinforcement Learning for Adaptive Cyber Defense in Network Security," *ACM AICCONF'24*, New York, USA, 2024, pp. 292–297.

[3] I. S. Thompson, A. Caron, C. Hicks, and V. Mavroudis, "Entity-based Reinforcement Learning for Autonomous Cyber Defence," *ACM AutonomousCyber'24 Workshop*, New York, USA, 2024, pp. 56–67.

[4] H. Kheddar, D. W. Dawoud, A. I. Awad, Y. Himeur and M. K. Khan, "Reinforcement-Learning-Based Intrusion Detection in Communication Networks: A Review," *IEEE Commun. Surv. Tutor.*, early access.

[5] N. D. L. Fuente and D. A. V. Guerra, "A Comparative Study of Deep Reinforcement Learning Models: DQN vs PPO vs A2C," arXiv:2407.14151, 2024.

[6] H. V. Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning", *AAAI'16*, Phoenix, USA, 2016, pp. 2094-2100.

[7] "TTCP CAGE Challenge 2" [Online]. Available at: https://github.com/cage-challenge/cage-challenge-2