https://doi.org/10.7840/kics.2024.49.3.333

어뢰위협 회피를 위한 강화학습 기반 기만 전술

정 재 현◆, 김 규 선*, 박 수 현°, 김 중 헌*, 윤 원 혁**

Reinforcement Learning-Based Counter Measure Tactics to Avoid Torpedo Threats

Jaehyun Chung◆, Gyu Seon Kim*, Soohyun Park°, Joongheon Kim*, Wonhyuk Yun**

요 약

최근 잠수함 체계에 있어서 미래 첨단 기술 기반의 전투성능 극대화를 위하여 잠수함용 지능형 임무지원시스템 통합 자동화 기술에 대한 연구가 활발히 진행되고 있다. 특히, 잠수함 어뢰 기만 전술 고도화 기술은 적 어뢰 및 기만기 성능변화에 따른 기만전술을 연구한다. 일반적으로 어뢰의 속도는 잠수함보다 훨씬 빠르므로 긴급한 상황이기 때문에 신속하고 정확한 계산이 필요하다. 이에, 본 논문에서는 기계학습 방법 중 하나인 강화학습 기법을 적용하여 어떤 상황에서도 대응할 수 있는 적 어뢰 기만 전술을 산출하고, 자함의 회피 침로를 선택할 수 있는 어뢰 기만 전술 알고리즘을 제안한다. 해상 환경에서 발생할 수 있는 다양한 변수에 적응할 수 있는 강화학습 알고리즘을 적용하고, 이를 통해 해당 알고리즘의 우수성을 평가함과 동시에 실제 해양 환경에서 어뢰 기만 전술에 적용 가능성을 제시한다.

키워드 : MDP, 심층 강화학습, 어뢰 기만 전술
Key Words : MDP, Deep Reinforcement Learning, Torpedo Counter Measure Strategies

ABSTRACT

Recently, research on integrated automation technology for intelligent mission support systems for submarines is being actively conducted in order to maximize combat performance based on future cutting-edge technologies in the submarine system. In particular, advanced technology for submarine torpedo counter measure tactics studies counter measure tactics according to changes in the performance of enemy torpedoes and decoy aircraft. In general, the speed of a torpedo is much faster than that of a submarine, so quick and accurate calculations are necessary because it is an emergency situation. Accordingly, in this paper, we apply reinforcement learning, one of the machine learning methods, to calculate enemy torpedo counter measure tactics that can respond to any situation and propose a torpedo counter measure tactic algorithm that can select an evasive course for the ship. We apply a reinforcement learning algorithm that can adapt to various variables that may occur in the maritime environment, and through this, we evaluate the excellence of the algorithm and present its applicability to torpedo counter measure tactics in an actual maritime environment.

※ 본 연구는 2022년 정부(방위사업청)의 재원으로 국방기술진흥연구소의 지원을 받아 수행된 연구임. (No. KRIT-CT-22-023, 21-107-D00-012, 잠수함 어뢰기만전술 고도화 기술) 본 논문의 교신저자는 박수현임.
◆ First Author : Korea University School of Electrical Engineering, rupang1234@korea.ac.kr, 학생회원
° Corresponding Author : Korea University School of Electrical Engineering, soohyun828@korea.ac.kr, 정회원
* Korea University School of Electrical Engineering, joongheon@korea.ac.kr, 종신회원
** LIG Nex1, wonhyuk.yun2@lignex1.com, 정회원
논문번호 : 202311-158-A-RU, Received November 28, 2023; Revised December 8, 2023; Accepted December 8, 2023

Ⅰ. 서 론

해상 전투에 있어 수상함과 잠수함은 중요한 무기체계이다. 수상함과 잠수함은 다양한 무장을 탑재하여 임무를 수행한다. 특히, 주력 무기인 어뢰는 적군을 무력화시키는 역할을 한다. 하지만 이러한 어뢰는 반대로 잠수함에게 가장 위협적인 존재이기도 하다. 어뢰의 공격으로부터 보호하는 어뢰대항책은 오히려 어뢰의 성능을 향상시키는 기반이 되었다. 어뢰의 발전으로 인해 잠수함에 대한 장거리 정밀 타격이 가능해짐에 따라 주요한 전력인 잠수함이 이 위험에 노출되었다. 따라서 다양한 제약 조건이 존재하는 상황에서 제한된 자원을 가지고, 잠수함에게 장거리에서 큰 타격을 입힐 수 있는 빠르게 접근하는 어뢰에 대해 실패없이 확실한 대응이 가능한 자율적이고 효과적인 잠수함의 어뢰 회피 전술에 대한 행동을 권고하는 체계가 필요하다. 이에 어뢰에 대항할 수 있도록 기만기 체계가 탑재되었지만, 해양 환경에 따른 기만기의 상태와 자함 및 적 어뢰의 변동성이 적중률과 효용성을 감소시키는 큰 원인이 된다.

본 논문에서는 효과적으로 기만기 제어 및 잠수함 회피 침로를 권고하는 체계를 위해 인공지능을 활용하는 방법을 제안한다. 인공지능을 활용하면 기만기의 효용성을 최대화하기 위해 최적화된 기만기의 발사 시점 및 회피 침로를 계산하여 빠르게 변화하는 환경에 적응하고 지속적인 대응 전술을 수립하여 자함이 생존할 수 있도록 한다. 한편, 어뢰 기만이 실패하여도 자함이 최적 회피 침로를 산출하여 어뢰를 회피할 수 있도록 보조한다. [그림 1]은 어뢰 기만 전술의 개념을 설명하는 그림으로, 기만기 발사 시 어뢰의 표적이 잠수함에서 기만기로 바뀌어 어뢰가 기만기를 추적하고 잠수함은 기만기의 반대 방향으로 기동하여 잠수함이 생존하는 모습을 나타낸다[1]. 본 논문에서는 사용할 인공지능으로 강화학습을 택하여, 강화학습의 알고리즘인 Deep Q-Network (DQN)을 접목시켜 효율적인 어뢰 기만 전술을 제안한다.

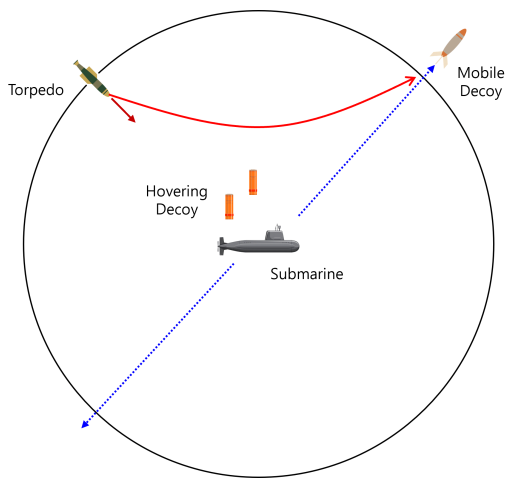


그림 1. 어뢰 기만 전술의 개략도
Fig. 1. Schematic diagram of torpedo deception tactics

본 논문은 2장에서 어뢰 기만 전술에 대해 서술한다. 3장에서는 강화학습에 대한 기본 개념을 소개하고 잠수함의 생존율을 높이기 위한 최적 기만 전술 알고리즘을 제안한다. 이어 4장에서 본 알고리즘의 성능을 평가하고, 5장에서 결론을 맺는다.

Ⅱ. 어뢰 기만 전술

어뢰에 대응하기 위한 잠수함의 어뢰대항책에 대해 많은 연구가 활발히 이루어지고 있다. 오늘날 어뢰에 대응하기 위해 가상 시뮬레이션 기반 기만기 성능 극대화 솔루션이 활발히 연구되고 있다[2],[3]. 전반적으로는 어뢰에 대한 잠수함의 회피 시뮬레이션에 대한 연구가 많다. 기만기는 어뢰를 유도하여 잠수함이 회피할 수 있는 시간을 증가시켜 잠수함이 생존할 수 있도록 하는 장치이다. 어뢰 회피가 어려운 상황에서 잠수함은 회피 기동이 불가능하다고 판단되면 기만기를 발사한다. 이때, 기만기는 소음을 발생시켜 어뢰가 잠수함 대신 기만기 쪽으로 향하여 폭발하도록 유도한다. 기만기의 종류에는 부유식 기만기(Hovering Decoy)와 자항식 기만기(Mobile Decoy)로 두 가지가 있다[4].

부유식 기만기(Hovering Decoy)는 움직이지 않는 기만기이다. 부유식 기만기는 수중의 일정한 위치를 유지하면서 고정된 상태로 어뢰를 기만하는 역할을 한다. 이는 일정한 위치에서 운용되기 때문에 복수 개의 기만기를 동시에 운용하기도 한다.

자항식 기만기(Mobile Decoy)는 부유식 기만기와 달리 이동하는 기만기이다. 이는 들어오는 어뢰의 방위에 수직인 경로에서 발사되도록 Modeling 되었다. 자항식 기만기는 잡음 신호를 생성하는 기능과 어뢰의 위치를 감지하여 표적 신호를 모의하는 기능이 있다. 이는 부유식 기만기의 단점을 보완하여 자함의 생존성을 더 높이게 되었다. 한 연구에 따르면, 기만기는 잠수함에서 10,000m 떨어진 위치까지 17노트의 일정한 속도로 이동이 가능하다[5].

기존 어뢰 기만 전술에 관한 연구는 앞서 말한 바와 같이 시뮬레이션을 이용한 연구가 많다. 성균관대학교에서는 Discrete Event System Specification (DEVS) 시뮬레이션 시스템을 통해 기만기가 접근하는 어뢰를

기만하는 데까지 걸리는 대응시간으로 측정하여 어뢰 기만을 성공시키기 위해 필요한 조건들을 찾아낸다. 대만 Shih Chien 대학에서는 시뮬레이션과 Evolutionary 알고리즘을 통해 기만기를 사용한 전술의 성공률을 측정한다. 이는 주어진 기준에 따라 여러 번의 변이를 통해 진화된 최적 행동 집합이 나오도록 한다. 해군사관학교에서는 Monte-Carlo 시뮬레이션을 통해 어뢰회피전술 체계의 효과도를 분석한다. 이 전술은 여러 번의 시뮬레이션을 진행하여 나온 결과들을 통해 잠수함 생존율을 극대화시키는 요소를 확인하고 이를 최우선으로 행동하는 방식이다.

[표 1]은 기존 어뢰 기만 전술과 본 논문에서 사용할 강화학습을 적용한 알고리즘을 비교한 것을 나타낸다. 현재 어뢰 기만 전술은 부유식 기만기와 자항식 기만기를 복합적으로 사용하여 시뮬레이션을 통한 전술을 사용하고 있다. 그러나 이러한 솔루션은 기만기의 성능을 높여 어뢰에 대응하는 방법이므로, 이는 어뢰를 향상시키는 밑거름이 된다. 따라서 기만기의 성능을 높이면, 당장은 어뢰에 대응할 수 있을지 모르지만, 어느 순간 어뢰의 성능이 향상될 가능성은 여전히 존재한다. 따라서 어뢰의 성능이 지속적으로 향상되어도 이에 실시간으로 신속하게 대응 가능한 알고리즘이 필요하다. 또한, Monte-Carlo시뮬레이션의 경우 그 전까지 대응하지 못했던 선유도 중어뢰에 대해서도 대응이 가능했으나, 학습 시 불안정한 경우가 생길 수 있다. 이 문제를 해결하기 위해 기계 학습의 한 종류인 강화학습 중 인공신경망을 활용한 알고리즘은 좋은 방안이 된다.

Ⅲ. 강화학습 개요

3.1 강화학습과 MDP

강화학습(Reinforcement Learning)은 기계학습의 한 유형으로, 정적인 데이터셋에 의존하지 않고 동적인 환경에서 작동하며 수집된 경험을 통해 학습하는 점에서 지도학습(Supervised Learning) 및 비지도학습(Unsupervised Learning)과는 차이가 있다[6]. 이를 통해 강화학습은 지도학습과 비지도학습에서 중요한 과정인 데이터 수집, 처리 및 Labeling 과정이 필요 없다는 장점이 있다. 강화학습에서 경험은 에이전트(Agent)가 주어진 환경(Environment)에서 상호작용하고 시행착오 과정을 통해 얻어진다. 에이전트는 결정을 내리고 행동하는 주체로서 이러한 경험을 통해 학습한다. 에이전트는 현재 상태(State)를 기반으로 자신의 행동(Action)을 선택하고, 그 행동으로 인해 주어진 환경으로부터 얻는 보상(Reward)을 통해 다음 상태로 진행한다. 에이전트는 이러한 과정을 반복함으로써, 보상을 통해 스스로 학습하고 최적의 정책(Policy)을 찾는다. 이렇듯 결정을 시간 순서대로 연속적으로 내려야 하는 문제를 순차적 의사결정(Sequential Decision Making) 문제라고 한다[7]. 또한, 정책(Policy)은 에이전트의 현재 상태를 입력으로 받아 해당 상태에서 에이전트가 어떤 행동을 수행할지 결정하는 함수이다. 에이전트는 자신의 정책에 따라 행동을 선택하기 때문에, 최고의 보상을 얻기 위해 최적의 정책을 학습하는 것이 필수적이다.

강화학습은 순차적 의사결정 문제를 해결하기 위한 방법으로, 마르코프 결정 프로세스(Markov Decision Process, MDP)를 통해 수학적으로 다양한 상태와 행동을 가지는 시스템의 문제를 모델링할 수 있다[8]. MDP는 “미래는 오로지 현재에 의해 결정된다”는 마르코프 성질(Markov Property)을 따른다. 이를 기반으로 하여 식 (1)과 같이 구성된다.

$$MDP \equiv (S,\ A,\ P,\ R,\ \gamma) \tag{1}$$

여기서 S는 상태(State), A는 행동(Action), P는 전이 확률 행렬(Transition Probability Matrix), R은 보상(Reward), γ는 감쇠 인자(Discount Factor)이다. 상태 집합 (S)은 에이전트인 잠수함이 주어진 해상 환경에서 가능한 상태들을 모두 모아 놓은 집합으로 3장에서 자세히 정의한다. 행동 집합 (A)은 에이전트인 잠수함이 해당 환경에서 취할 수 있는 모든 행동을 포함하는 집합이다. 본 논문에서는 에이전트인 잠수함의 행동 집합은 어뢰를 피하기 위한 회피기동과 기만기 제어로 구성되어 있다. 전이 확률 행렬 (P)는 현재 상태에서 다음 상태

표 1. 기존 어뢰 기만 전술 및 제안한 어뢰 기만 전술 비교
Table 1. Comparison of existing torpedo deception tactics and proposed torpedo deception tactics

어뢰 기만 전술 방법	방식
DEVS 시뮬레이션	시뮬레이션을 통해 어뢰 기만 대응시간 측정하여 기만 성공 요소 탐색
Evolutionary 알고리즘	시뮬레이션 기반 여러 번의 변이를 통해 진화된 최적 행동 집합 탐색
Monte-Carlo 시뮬레이션	시뮬레이션 기반 회피전술 효과도 분석 및 잠수함 생존율 극대화 요소 파악
인공신경망 기반 강화학습	인공신경망 기반 실시간으로 상황 분석 및 최적의 행동 제시

로 도착할 확률을 가리킨다. 즉, 특정 상태 s에 있을 경우, 에이전트가 특정 행동을 취함으로써 다음 상태(s')로 전이될 확률을 나타낸다. 보상 (R)은 어떤 상태 s에 도착했을 때 받게 되는 값을 의미한다. 이를 수식으로 표현하면 식 (2)와 같다. 여기서, R_t, S_t는 각각 t 시점에서의 보상과 상태를 나타낸다.

$$R = \mathbb{E}[R_t|S_t = s] \quad (2)$$

감쇠 인자(γ)는 미래에 얻을 보상에 비해 당장 얻는 보상을 얼마나 더 중요하게 여길 것인지 나타내는 요소로, 0에서 1사이의 값으로 설정된다. 감쇠 인자는 에피소드를 진행하면서 지속적으로 곱해지는 가중치이다. 따라서 가중치의 값이 0에 가까울수록 미래에 대한 보상이 0으로 수렴하는 속도가 빨라진다. 다시 말하면, γ 값이 0에 가까울수록 현재 상태에 얻을 수 있는 보상에 더 중점을 두며 1에 가까울수록 미래 보상에 더 큰 가중치를 두고 행동하는 에이전트가 된다.

위 내용을 통해, 리턴(Return, G_t)이라는 보상의 확장된 개념을 이해할 수 있다. 리턴은 특정 시간 t 이후의 미래에 대한 가중치 값이 곱해져서, 감쇠된 보상의 합으로 정의된다. 이것은 식 (3)과 같이 계산된다. 강화학습에서의 목표는 모든 에이전트가 누적 보상의 합인 리턴을 최대화하기 위한 것을 알아야 한다.

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (3)$$

앞서 언급한대로 감쇠 인자(γ)는 0과 1 사이의 값을 가진다. 따라서 이러한 γ의 제약은 미래에 가까워질수록 γ를 계속 곱해주기 때문에 미래에 받을 보상은 0에 가까워지게 된다. 이를 통해 미래에 얻을 보상 및 현재 얻는 보상의 가중치를 조절할 수 있다. 한편, γ가 0과 1 사이 값이기 때문에 리턴이 무한대로 발산하는 것을 방지한다. 만약 γ값이 1보다 크면 리턴이 무한으로 발산할 수 있어서 리턴들 간의 비교가 어려워지고, 학습이 원활하게 진행되지 않을 수 있다. 그래서 항상 γ값을 1보다 작게 설정하여 리턴이 수렴하도록 할 수 있다. 즉, 감쇠 인자는 수학적 안전장치의 역할을 한다.

한편, 리턴은 에이전트의 정책 함수 (π)에 의해 결정된다. 정책 함수는 각 상태에서 어떤 액션을 선택할지 정해주는 함수이다. 즉, 특정 상태 s에서 에이전트가 취할 행동 a를 정해주는 함수이며, 이는 식 (4)와 같다. 여기서, A_t는 t 시점에서의 행동을 나타낸다.

$$\pi(a|s) = P[A_t = a \mid S_t = s] \quad (4)$$

에이전트는 환경 내에서 보상 값에 따라 학습하며, 최종적으로 보상을 최대화하기 위한 방향으로 학습한다. 이 과정에서 정책을 수정해 나가며 최종적으로 리턴의 기댓값을 최대화하는 최적 정책 (π^*)을 학습하게 된다. 이를 위해 보상을 최대화할 수 있는 행동을 선택하려면 각 상태의 가치를 계산해야 한다. 가치는 식 (5)의 벨만 최적 방정식(Bellman Optimally Equation)을 통해 계산된다[9].

$$q_*(s_t|a_t) = R + \gamma \sum_{s' \in S} P_{ss'}^{a} \max\left[q_*(s', a')\right] \quad (5)$$

여기서 $q(s_t|a_t)$는 상태-액션 가치 함수로, 현재 상태 s_t에서 취할 수 있는 행동 a_t의 가치를 평가하는 함수이며, 일반적으로 Q 가치(Q-Value)로 표현된다. Q 가치 함수에서는 상태 s와 행동 a가 입력으로 동시에 들어가게 된다. Q가치 함수에서는 수식처럼 상태와 행동을 모두 고려하는데, 이는 상태에 따라 에이전트가 선택한 액션의 결과가 달라지기 때문이다. 즉, 동일한 행동 a를 선택했어도, 에이전트가 있는 상태가 s인지 s'인지에 따라 달라진다는 것이다. 이때, 에이전트는 벨만 최적 방정식의 최대 연산자를 이용하여 정책을 기반으로 Q가치를 최대화하는 가장 높은 보상을 얻을 수 있는 방향으로 행동을 선택하게 된다. 그러므로 학습이 반복되면서 최적의 가치를 가지는 식 (6)의 최적 정책을 발견할 수 있다.

$$\pi_* = arg \max_a q_*(s_t|a_t) \quad (6)$$

이와 같은 방식의 강화학습은 효과적으로 순차적 의사결정 문제를 해결할 수 있는 방법이다. 일반적으로 순차적 의사결정 문제를 다루는 강화학습의 대표적인 방법은 동적 계획법(Dynamic Programming)이다[10]. 동적 계획법은 임의로 초기화된 가치 값을 보다 실제 값으로 근사화하기 위해 벨만 방정식을 반복적으로 사용하는 방법이다. 하지만 동적 계획법은 Pseudo-Polynomial의 연산 복잡도를 가지기 때문에 비교적 간단한 상태 정보를 가진 문제만 해결할 수 있으며 복잡하고 상태 정보가 많은 실제 환경과 유사한 문제를 다루기에는 적합하지 않다[11]. 이 같은 문제를 해결하기 위해 본 논문에서는 Q-Learning을 소개한다.

3.2 Monte-Carlo

Monte-Carlo (MC)는 특정 MDP에서 가치를 측정하

는 강화 학습 알고리즘이다. 이는 '직접 측정하기 어려운 통계량에 대해 여러 차례 샘플을 추출하여 해당 값을 추측하는 기술'이다. 이는 여러 번의 리턴 계산을 통해 얻은 평균값이 실제 가치에 수렴할 것이라는 원리를 활용한 것이다. 즉, 대수의 법칙(Law of Large Numbers)을 통해 샘플링 횟수가 증가함에 따라 에이전트는 더 많은 경험을 쌓아 가치 예측 값이 점점 정확해진다. MC를 통해 업데이트하는 식은 식 (7)과 같다.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(G_t - Q(s_t, a_t)) \quad (7)$$

MC는 에이전트가 리턴 G_t를 따라가도록 학습하여 G_t를 MC의 목표(Target)라 한다. MC에서 에이전트는 각 에피소드가 끝날 때마다 리턴을 저장한 후, 값을 평균 내어 업데이트한다.

3.3 SARSA

MC는 학습 업데이트를 수행하기 위해 리턴이 필요하며, 이 리턴은 에피소드가 완료되어야 얻을 수 있다. 따라서, MC는 종료 조건이 명확한 경우의 MDP에서만 사용할 수 있다. 이때, 종료 조건이 명확하지 않은 MDP의 경우는 Temporal Difference (TD) 학습을 사용하여 진행한다. TD는 에피소드가 종료되기 전에 Value 값을 업데이트한다. 이는 명칭 그대로, 시간적 차이를 활용하여 과거의 추정치를 업데이트하기 위해 미래의 추정치를 사용한다. 다시 말하면, 환경에서 에이전트가 한 Step 더 진행할 시 과거보다 더 정확하게 추정할 수 있으며, 이를 통해 학습 업데이트를 수행한다. 이처럼 TD를 사용하여 Q 가치를 근사하는 알고리즘을 State-Action-Reward-State-Action (SARSA)라고 한다. SARSA는 식(8)에 의해 업데이트가 진행된다[12].

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \quad (8)$$

3.4 Q-Learning

Q-Learning은 현재 상태에서 특정 행동을 취할 때 얻는 보상의 기댓값을 학습하여 최적의 정책을 찾아내는 방법으로 Q 가치가 가장 큰 값을 가지는 행동을 선택하여 문제를 해결한다[13]. Q-Learning은 학습을 진행할 때 식 (9)와 같이 벨만 최적 방정식을 기반으로 한다.

$$Q(s, a) \leftarrow Q(s, a) + \alpha(R + \gamma max Q(s', a') - Q(s, a)) \quad (9)$$

이때 a는 학습률(Learning Rate)로 학습을 어느 정도 빠르게 진행할지 결정하는 상수이다. 이로써 Q-Learning은 현재 상태 s에서 행동 a를 선택할 때 얻을 수 있는 보상의 기댓값을 출력하는 Q 가치 함수를 통해 최적 정책이 결정된다.

Q-Learning은 Off-Policy 학습법에 속한다. Off-Policy는 학습에 사용되는 타깃 정책과 행동 정책이 서로 다른 학습 방법을 말한다. 여기서 타깃 정책은 학습의 대상이 되는 정책이며 지속적인 업데이트를 통해 강화되는 정책이다. 행동 정책은 실제로 환경과 상호작용하여 경험을 쌓고 있는 정책을 말한다. Q-Learning에서 행동 정책에는 에이전트의 탐험(Exploration)을 위한 확률 값인 ε을 포함하고 타깃 정책에는 Q 가치가 높은 행동만을 선택하는 Greedy 정책을 따르는데, 이는 벨만 최적 방정식에 최대 연산자가 있고 Q-Learning이 벨만 최적 방정식을 대상으로 학습하기 때문이다. 이를 통해 Q-Learning은 과거의 경험을 재사용할 수 있고, 사람의 데이터로부터 학습할 수 있다.

3.5 Deep Q-Network (DQN)

위에서 설명한 것처럼, Q-Learning은 Off-Policy를 통해 여러가지 이점을 누릴 수 있다. 그러나, Q-Learning 알고리즘에도 치명적인 단점이 있다. 그것은 Q-Learning이 테이블을 이용한 학습이라는 점이다. 상태집합이 굉장히 커지면, 수많은 상태 정보들을 테이블에 담아 만들기가 힘들어지기 때문이다. 특히, 어뢰 기만 전술은 실제로 많은 상태 정보가 필요하기 때문에 Q-Learning을 직접 적용하는 것은 어렵다. 따라서 일일이 모든 데이터를 테이블에 저장할 수 없기 때문에 인공신경망(Neural Network)이라는 새로운 개념을 알고리즘에 도입한다. 이처럼 Q-Learning에 인공신경망을 적용하여 값을 일반화하여 학습을 진행하는 알고리즘을 Deep Q-Network (DQN)이라고 한다[14], [15].

[그림 2]는 본 논문에서 진행할 실험 환경을 도입한 인공신경망 구조의 간단한 예시를 나타낸다. DQN은 기존 강화학습과 달리 인공신경망을 통해 파라미터 θ를 학습한다. 위 그림에서는 길이가 6인 벡터를 입력으로 받아 8개의 출력을 얻는다. 여기서 가운데에 있는 여러 개의 노드로 구성된 레이어들을 Hidden 레이어라고 한다. 이때, 노드는 이전 노드로부터 들어오는 값을 선형결합 후 ReLU와 Sigmoid 같은 비선형 함수로의 입력으로 사용된다. DQN에서는 Q-Learning과 비슷하게 $R + \gamma maxQ(s', a')$을 타겟으로 하고, 이에 대한 손실함수를 식 (10)과 같이 정의한다.

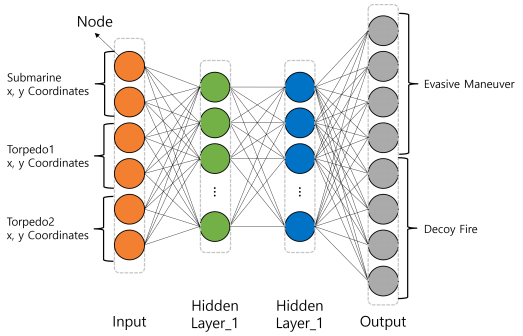


그림 2. 인공신경망 구조
Fig. 2. Structure of artificial neural networks

$$L(\theta) = \mathbb{E}[R + \gamma \max Q_\theta(s', a') - Q_\theta(s, a))^2] \quad (10)$$

파라미터 θ를 업데이트하는 과정에서는, DQN은 식 (11)과 같이 편미분을 사용하여 업데이트한다.

$$\theta' \leftarrow \theta + \alpha(R + \gamma \max_{a'} Q_\theta(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t)) \times \nabla_\theta Q_\theta(s_t, a_t) \quad (11)$$

하지만, Q-Learning에 인공신경망을 도입하면 샘플들 간의 상관관계가 생기고 타겟 네트워크에 계산과 학습이 진행되어 학습의 안정성이 떨어지는 문제가 생긴다. DQN은 안정적인 학습 수렴을 얻기 위해 경험 재사용(Experience Replay) 및 별도의 타겟 네트워크(Target Network)를 활용하였다.

3.5.1 경험 재사용(Experience Replay)

경험 재사용은 샘플들 간의 상관관계를 줄이기 위해 나온 개념이다. 이를 위해 DQN에서는 리플레이 버퍼(Replay Buffer)를 사용한다. 이는 버퍼에 가장 최근 데이터 n개를 저장하는 아이디어다. 버퍼에서 특정 미니배치 크기만큼 데이터를 무작위로 뽑아 모든 데이터를 여러 번 재사용이 가능해진다. 이러한 무작위 데이터로 학습하면 데이터 간의 상관성이 작아져서 데이터 효율성이 증가한다. 따라서 학습이 효율적으로 이루어지게 된다.

3.5.2 별도의 타겟 네트워크(Target Network)

손실 함수 $L(\theta)$의 직관적인 의미는 정답과 모델의 예측 값 간의 차이로 해석할 수 있다. 그래서 θ는 학습하는 과정에서 이 차이를 줄이는 방향으로 업데이트 된다. 이때 Q-Learning에서는 타겟으로 $R + \gamma maxQ_\theta(s', a')$가 사용되는데, 이는 θ가 변함에 따라 같이 변하게 되어 학습 시 안정성이 떨어진다. 이를 해결하기 위해 DQN은 타겟 네트워크를 별도로 두었다. 이는 손실 함수를 계산할 때 네트워크의 파라미터는 변하지 않게 고정시켜 타겟 값이 변하지 않도록 하는 역할을 한다. 또한 일정 주기마다 고정시킨 파라미터를 최신 파라미터로 교체하여 학습의 안정성을 증가시켰다.

Ⅳ. 강화학습 기반 어뢰 기만 전술 실험 및 평가

4.1 실험 설계

본 장에서는 어뢰 기만 전술을 사용할 수 있는 상황을 만들기 위해 [그림 3]의 환경에서 서로 다른 위치에 있는 두 어뢰가 존재하는 상황에서 실험을 진행하였다. 정사각형 그리드를 이용하여 환경을 구성하였으나, 따로 그리드의 크기를 정해두진 않았다. 본 논문에서는 복잡한 환경을 축소하여 인공신경망을 통해 안정적인 학습을 보장할 수 있는 DQN을 제안한다. DQN의 대조군으로는 무작위로 행동을 선택하는 1) Random, 테이블 기반으로 학습을 진행하는 2) SARSA, 3) Monte-Carlo, 그리고 Q 네트워크를 활용한 4) Q-Learning의 4가지가 있다. 다음은 이와 같은 환경에서 구현하기 위해 필요한 상태, 행동, 보상, ε 및 다른 변수들을 정의함으로써 실험 설계에 관해 논의한다. 초기 잠수함과 어뢰의 위치는 [표 2]와 같다. 기존 어뢰 기만 전술에서 가장 많이 쓰였던 방식은 Monte-Carlo 방식이었다. 본 논문에서는 기존 어뢰 기만 전술에 쓰인 Monte-Carlo 방식과 집중적으로 비교하여 인공신경망에 기반한 전술의 성능이 더 좋다는 것을 확인해보았다.

한편, 실험에 관한 시뮬레이션을 진행하는 데 사용했

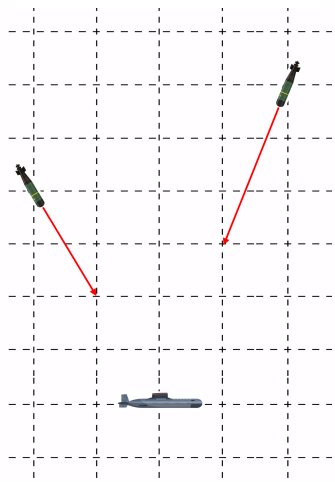


그림 3. 어뢰 2개를 이용한 실험 환경
Fig. 3. Experimental environment using two torpedoes

던 컴퓨터 사양 및 Tool은 [표 3]과 같다. 본 논문에서는 AMD Ryzen 5 5600X 6-Core의 프로세서와 Windows 11이 있는 컴퓨터에서 VS Code에서 PyTorch 모듈로 해당 환경을 구현하여 실험을 진행하였다. 이때 사용된 RAM의 용량은 64.0 GB이다.

표 2. 초기 실험 환경 설정
Table 2. Experimental environment setup

초기 환경 설정	값
잠수함 위치	(6, 6)
어뢰 1 위치	(2, 4)
어뢰 2 위치	(0, 8)

표 3. 실험 환경 시뮬레이션에 쓰이는 Tool 및 컴퓨터 사양
Table 3. Tools and computer specifications used to simulate the experimental environment

종류	이름
프로세서	AMD Ryzen 5 5600X 6-Core
Windows 사양	Windows 11 Education
프로그램	VS Code
모듈	PyTorch
RAM	64.0GB

4.1.1 상태

실험은 앞서 말한 바와 같이 그리드 크기가 정해져 있지 않은 상태에서 진행된다. 이는 잠수함이 바다에서 어뢰를 회피하기 때문에 바다라는 환경은 공간의 제약을 줄 수 없을 정도로 넓은 공간이므로 그리드의 크기에 제약을 두지 않았다. 즉, 만약 시뮬레이션 실행 시 주어진 그리드의 가장 왼쪽 점에서 에이전트가 Left 행동을 취해도 에피소드가 종료되지 않고 계속 왼쪽으로 어뢰에 맞을 때까지 이동하게 설정하였다.

강화학습의 에이전트인 잠수함의 상태 정보는 각 단계 t에 따라 식 (12)와 같이 정의했다.

$$S_t = \{s_{xt}, s_{yt}, t1_{xt}, t1_{yt}, t2_{xt}, t2_{yt}\} \quad (12)$$

이때 s_{xt}, s_{yt}는 현재 잠수함의 x, y 좌표 정보를, $t1_{xt}$, $t1_{yt}$, $t2_{xt}$, $t2_{yt}$는 각각 두 어뢰의 x, y 좌표 정보를 나타낸다. 주어진 환경에서 잠수함은 어뢰에서 멀어지는 방향으로, 어뢰는 잠수함에게 가까워지는 방향으로 움직이도록 설정하였다. 이때, 상태 전이가 일어나는 경우 잠수함은 1칸, 어뢰는 2칸씩 움직이도록 설정하여 잠수함과 어뢰의 속도 차이를 표현하였다. 이때, 어뢰의 경우 상태가 바뀔 때마다 움직일 수 있는 방향이 4가지가 아닌 8가지이다. 만약 어뢰와 잠수함 사이 거리가 1칸 남았을 경우, 어뢰의 기동 방향이 4가지라면 어뢰는 2칸씩 움직이기 때문에 격추되어야 할 잠수함이 생존해 있는 경우가 생기기 때문이다. 또한, 어뢰 두 개가 같은 좌표에 있는 경우는 두 어뢰의 깊이가 다르기 때문에 어뢰의 충돌이 일어나지 않는 것으로 하였다. 이는 [그림 4]를 통해 더 쉽게 알 수 있다. 각각의 상태 정보는 현재의 에이전트가 어떤 상황에 있는지를 표현하며, 에이전트는 이 정보를 바탕으로 행동을 선택한다.

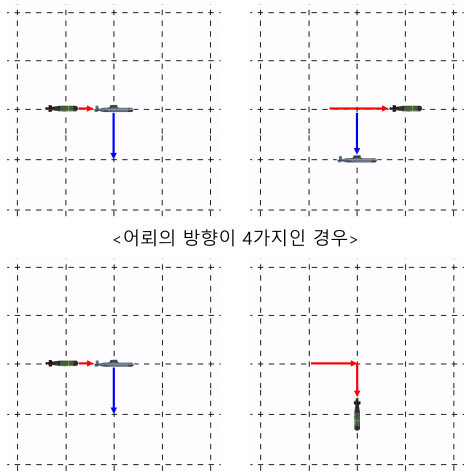


그림 4. 어뢰의 방향이 8가지인 이유
Fig. 4. The reason why torpedoes have eight directions

4.1.2 행동

잠수함은 어뢰의 위치에 따라 회피기동을 수행할지 아니면 기만기를 발사할지를 선택할 수 있다. 이에 따라 행동 집합은 식 (13)과 같이 정리할 수 있다.

$$A = \begin{Bmatrix} Up, Right, Down, Left \\ Decoy:\ Up, Right, Down, Left \end{Bmatrix} \quad (13)$$

어뢰 위치에 따라 잠수함은 주어진 그리드 환경에서 각 네 방향으로 기동할 수 있다. 또한 잠수함은 각 방향마다 발사할 수 있는 기만기 수를 제한하였는데, 이는 실제 기만기의 수량이 제한된 상황에서도 잠수함이 오래 생존해야 하는 상황을 표현한 것이다. 초기 어뢰와 잠수함의 위치를 고려하여 각 방향 당 기만기 수는 한 발로 제한하였다. 기만기 수량 제한은 보상 함수로 표현하였으며, 이는 다음 장에서 더 자세히 설명한다. 또한, 기만기 발사 시에 잠수함은 전에 움직인 방향 그대로 1번 더 기동한다. 예를 들어, 잠수함이 현 상태에서 오

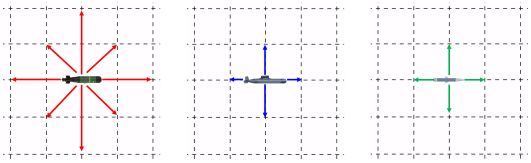


그림 5. 어뢰, 잠수함, 기만기의 이동 방향 및 크기
Fig. 5. Direction and size of movement of torpedoes, submarines, and decoys

른쪽으로 기동하였고 다음 상태에서 기만기를 위로 발사할 예정이면, 다음 상태에서 잠수함은 위로 기만기를 발사함과 동시에 오른쪽으로 기동한다. [그림 5]는 어뢰, 잠수함 그리고 기만기의 기동 방향과 방향 크기를 나타낸다.

4.1.3 보상 함수

잠수함의 효율적인 회피기동 경로를 생성하기 위해 식 (14)와 같이 보상 함수를 설계하였다. 위에서 말한대로, 기만기 수량은 보상 함수를 통해 제한하였다. 따라서 보상 함수는 두 가지 경우에 대해 다른 식을 얻는다.

$$R = \begin{cases} -25 & (if\ \ n > 1) \\ \frac{\min(d_{t1}, d_{t2})}{d_0} & (otherwise) \end{cases} \quad (14)$$

여기서 n은 기만기를 발사한 수량으로, 두 발 이상 발사하면 -25의 보상을 주어 학습 수렴 시 기만기를 각 방향마다 한 발 이내로 발사할 수 있도록 제한하였다. d_{t1}, d_{t2}는 각각 두 어뢰와 잠수함 사이 거리를 나타내며, d_0는 초기 상황에서 두 어뢰와 잠수함 사이 최소 거리를 나타낸다. 주어진 환경에서 $|\overrightarrow{d_0}| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$으로 계산되며, 그 값은 $\sqrt{20}$이다. 이 보상 함수 식은 기만기를 각 방향마다 한 발 이내로 발사하였을 때 적용한 경우로, 현 상태의 두 어뢰와 잠수함 사이 최소 거리를 d_0로 나누어 매 단계마다 상대적 거리로 보상을 얻게끔 설계하였다.

4.1.4 기타 실험 변수 설정

학습 초기에는 예측한 값이 불확실하기 때문에 충분히 학습할 수 있게 하기 위하여 ε - *Greedy* 방법을 도입했다[15]. ε - *Greedy* 방법은 최적의 행동을 선택하는 동시에 특정 확률로 무작위 행동을 선택함으로써 새로운 정보를 탐험하는 방법이다. 이 방법을 통해 강화학습에서의 탐험(Exploration)과 활용(Exploitation) 사이의 적절한 균형을 맞출 수 있다.

DQN 알고리즘을 이용하여 에이전트인 잠수함을 학습할 때, 인공신경망 학습 시 사용된 파라미터는 [표 4]와 같다. 잠수함 학습 시 배치 크기인 32의 데이터를 추출하여 학습 데이터로 활용한다. 학습 시 에피소드를 100 단위로 이루어지며, 감쇠 인자는 0.98이고, 학습률은 0.0005로 설정하였다. 본 실험에서 사용한 인공신경망은 128개의 노드로 이루어진 Hidden 레이어 2개가 있는 인공신경망을 사용한다. 이때, 입력 레이어와 출력 레이어의 노드 수는 각각 6, 8개이다. 주어진 실험 환경에서 학습을 진행하였고, 다음 장에서는 해당 실험 환경에서 얻은 성능평가에 대해 서술한다.

표 4. 학습 시 사용된 파라미터
Table 4. Parameters used during training

파라미터	값
감쇠 인자, γ	0.98
학습률, α	0.0005
초기 입실론, ε	0.3
최소 입실론, ε_{min}	0.01
배치 크기	32
활성함수	ReLU
옵티마이저	Adam

4.2 실험 결과

4.2.1 평균 보상

강화학습에서 에이전트는 리턴이라고 불리는누적 보상의 합을 극대화하는 방향으로 학습을 진행한다. 따라서 에피소드 별로 누적 보상의 합이 어떻게 변하는지 추적함으로써 해당 에이전트의 성능을 평가할 수 있다.

[그림 6]은 총 10000 에피소드가 진행됨에 따라 DQN, Q-Learning, Monte-Carlo, SARSA, Random 알

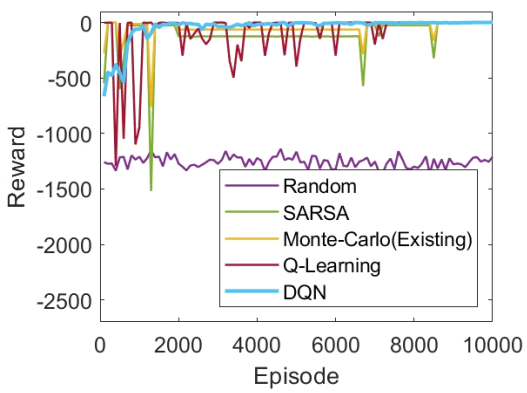


그림 6. 각 에피소드에 따른 평균 보상 (Random, SARSA, Q-Learning, DQN)
Fig. 6. Average rewards for each episode (Random, SARSA, Q-Learning, DQN)

고리즘에서의 평균 보상 값을 비교한 그래프이다. DQN의 경우, 약 2000 에피소드에 수렴 값에 도달하여 에피소드가 진행됨에 따라 보상 값의 변동이 적어 안정적이고 일정 값으로 수렴하는 것을 알 수 있다. 반대로, Monte-Carlo, SARSA와 Q-Learning은 보상 값의 변동이 심하며 약 8000 에피소드(SARSA와 Monte-Carlo의 경우 약 9000 에피소드)에 수렴 값에 도달하는 것을 알 수 있다. 특히 Monte-Carlo의 경우 기존 어뢰 기만 전술에 사용되고 있는 기법임에도 불구하고, Q 네트워크에 기반한 강화학습보다 성능이 떨어지는 것을 알 수 있다. Random의 경우, 다른 강화학습 알고리즘과 달리 보상 값 그래프 추세가 상승하지 않음을 확인할 수 있다. 각 알고리즘의 최종 수렴된 값 또한 DQN이 가장 높은 것을 알 수 있다. [표 5]는 각 알고리즘의 최종 수렴된 값을 나타낸다. 잠수함이 추적하는 어뢰에 대응하기 위해 제한된 수량의 기만기 범위 내에서 기만기를 발사하고 효율적인 회피 경로로 어뢰를 회피할 수 있도록 보상 함수를 설계했기 때문에 보다 높은 보상 값을 가진 DQN 알고리즘이 SARSA, Q-Learning보다 목표를 더욱 잘 달성했음을 알 수 있다.

표 5. 최종 수렴 값 (SARSA, Q-Learning, DQN)
Table 5. Final Convergence Value (SARSA, Q-Learning, DQN)

알고리즘	수렴값
Random	-1254.641
SARSA	-1.039
Monte-Carlo (Original tactic)	-0.398
Q-Learning	1.141
DQN	1.751

4.2.2 잠수함 생존율

[그림 7]은 제안한 알고리즘과 대조군인 SARSA, Monte-Carlo, Q-Learning, Random의 잠수함의 생존율을 나타낸다. 잠수함의 생존율은 매 에피소드마다 잠수함이 격추를 당했는지에 대한 유무를 바탕으로 격추당하지 않았던 에피소드의 수를 전체 에피소드의 수로 나눈 값으로 측정하였다. [그림 7]을 통해, 테이블 기반으로 학습이 이루어지는 SARSA, Q-Learning, Monte-Carlo와 무작위하게 행동을 선택하는 Random에 비해 DQN의 잠수함 생존율이 더 높은 것을 알 수 있다. 이는 또한 기존 어뢰 기만 전술에 쓰인 Monte-Carlo 방식보다 인공신경망 기반 강화학습의 생존율이 확실히 차이나는 것을 알 수 있다. [표 6]은 각 알고리즘 별 정확한 잠수함 생존율 수치를 나타낸다.

표 6. 잠수함 생존율 비교
Table 6. Submarine survival rate

알고리즘	생존율 (%)
Random	38.30
SARSA	76.04
Monte-Carlo (Original tactic)	77.63
Q-Learning	78.09
DQN	97.22

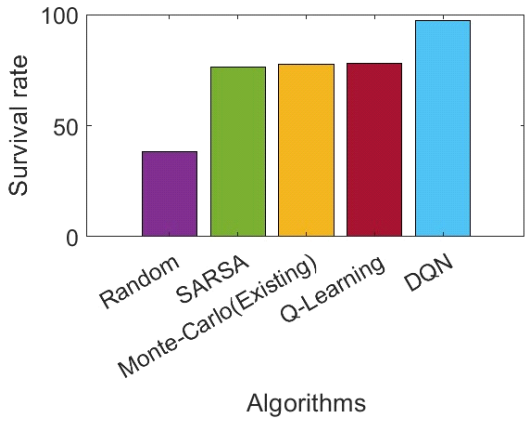


그림 7. 잠수함 생존율(Random, SARSA, Q-Learning, DQN)
Fig. 7. Submarine survival rate (Random, SARSA, Q-Learning, DQN)

정확한 수치를 측정한 결과, DQN은 SARSA, Q-Learning, Monte-Carlo보다 약 18 ~ 20% 더 높은 잠수함 생존율을 보였 다. Random과 비교하였을 때는 약 60% 더 생존율이 높았음을 알 수 있다. 이는 본 논문에서 인공신경망을 활용한 알고리즘을 잠수함 어뢰 기만 전술 및 회피기동에 적용하였을 때, Monte-Carlo를 적용한 기존 어뢰 기만 전술보다 우수한 성능을 보였음을 입증한다. 이에 따라, 다음 장에서는 시각화한 그림에 대한 비교는 DQN과 DQN의 대조군으로 Monte-Carlo보다 좀 더 성능이 좋은 Q-Learning과 비교하였다.

4.2.3 어뢰 기만 전술 기반 잠수함 기동 제어

[그림 8]과 [그림 9]는 DQN과 Q-Learning을 통해 학습하였을 때 나온 잠수함 기동의 한 예시를 시각화한 그림을 나타낸다. T는 각 에피소드의 단계 수를 나타내고, 초기 환경은 T = 0으로 설정하였다. 또한, 파란색 점과 빨간색 점은 각각 잠수함과 어뢰를 나타내며, 그 외의 색깔을 가진 작은 점은 기만기를 나타낸다. [그림 8]에서 DQN은 기만기를 두 발 이상 기만기를 발사하

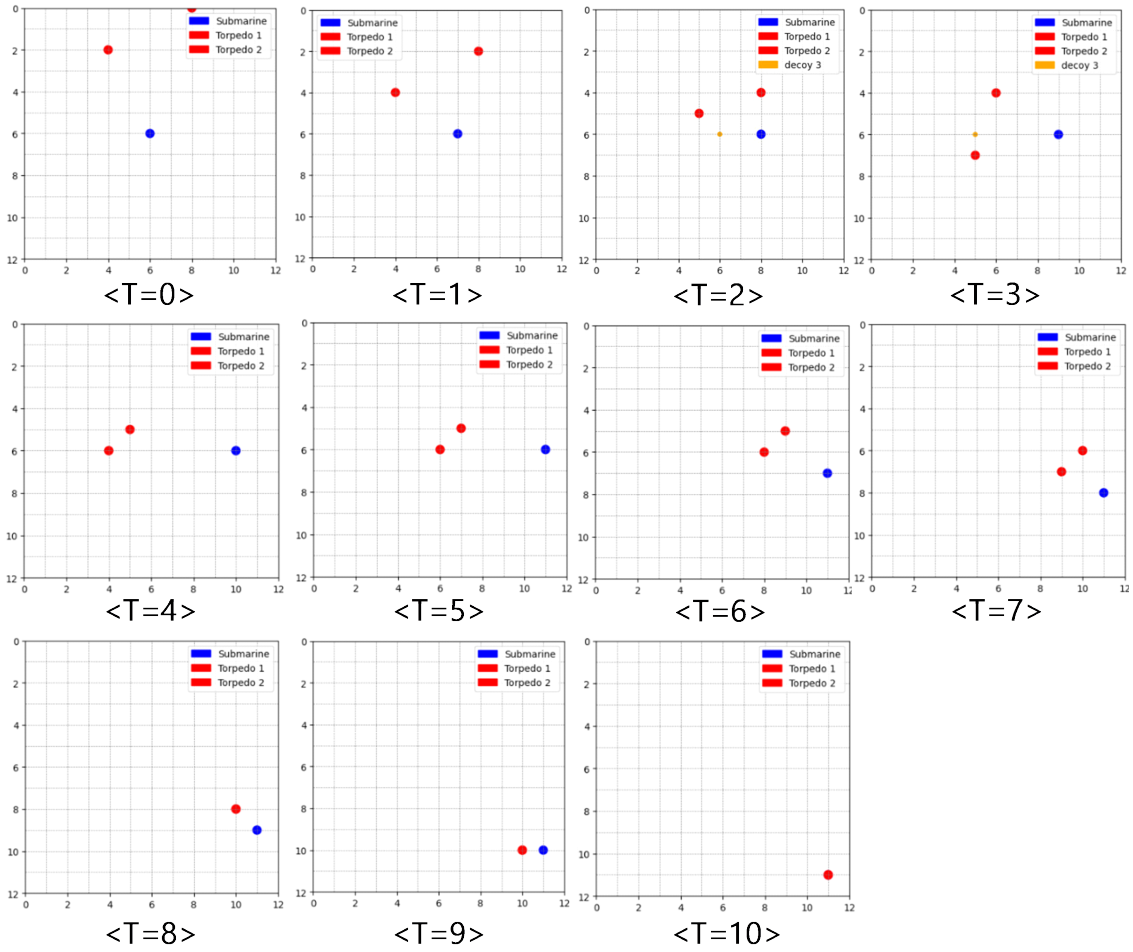


그림 8. DQN으로 학습한 잠수함 기동의 한 예시
Fig. 8. An example of submarine maneuver (DQN Algorithm)

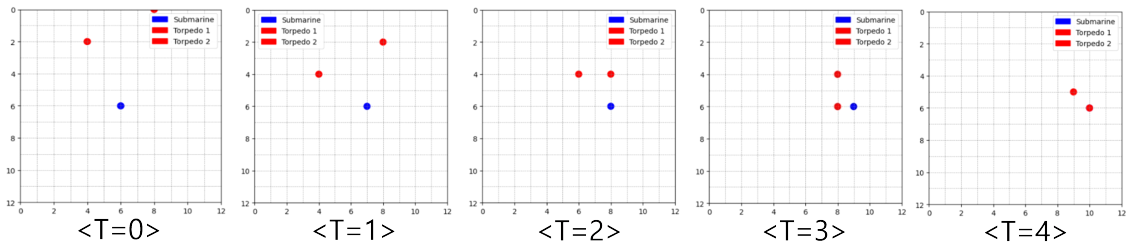


그림 9. Q-learning으로 학습한 잠수함 기동의 한 예시
Fig. 9. An example of submarine maneuver (Q-learning Algorithm)

여 어뢰를 유도하면서 회피하고 잠수함의 생존 시간을 높여주는 모습을 나타낸다. 그러나 [그림 9]에서 Q-Learning은 기만기 발사에 관한 행동은 하지 않으며, 회피 기동 또한 최적화된 방향으로 진행되지 않고 있음을 보여준다. [그림 6, 7, 8, 9]를 통해 제시한 알고리즘이 어뢰를 회피하고 잠수함의 생존율을 높여주는 데 큰 영향을 미쳤음을 알 수 있다.

Ⅴ. 향후 논의 방향

실제 잠수함이 존재하는 해양 환경에서는 다양한 변수가 존재한다. 해양 환경에서는 해양 생물, 암초, 난류, 난파선 잔해 등의 다양한 장애물이 있으며, 이는 잠수함의 경로와 속력에 영향을 미친다. 뿐만 아니라, 잠수함을 공격하는 어뢰 또한 물의 장력 등을 고려하면 어뢰가 탐지하는 범위에서 잠수함이 벗어나는 경우도 생길 수 있다. 이렇듯 실제로는 다양한 현실적 제약을 고려해야 한다. 이는 아무리 복잡한 환경에 맞는 강화학습 알고리즘을 적용하여 학습 시 현실적 제약에 의해 수렴 문제가 발생할 수 있다. 본 논문에서는 해상 환경을 추상화시켜 실험을 진행한 것으로 인공신경망을 활용한 강화학습의 성능이 가장 좋은 것으로 나타났다. 추후에는 사람에 의해 판단을 내리는 기존 어뢰 기만 전술에 현실적 제약에 영향을 많이 받지 않는 인공신경망을 활용한 강화학습을 적용하여 신속한 판단에 의해 잠수함의 생존율을 높이는 방향으로 연구가 진행되어야 한다. 또한, 본 논문에서 진행한 실험보다 현실성 있는 환경 Setting을 통하여 실전에 활용가능한 알고리즘을 좀 더 고도화되는 방향으로 연구가 진행될 필요가 있다.

Ⅵ. 결 론

본 논문은 두 어뢰를 회피하기 위해 잠수함의 어뢰 기만 전술과 자함 회피 침로를 찾기 위한 강화학습을 기반으로 하는 잠수함 기동 방법을 제안한다. 이 연구에서는 어뢰와 잠수함의 기동할 수 있는 칸 수를 다르게 하여 실제 어뢰의 속도가 잠수함보다 빠르다는 사실을 함께 고려하였다. 본 논문에서는 해당 문제에 다양한 강화학습 알고리즘을 적용하여 잠수함 회피기동 및 어뢰 기만 전술 성능을 평가하였다. 그 결과, 일반적인 Random과 비교하였을 때는 강화학습 기반 알고리즘이 우수한 성능을 보였다. 특히, SARSA, Monte-Carlo, Q-Learning 알고리즘과 비교하여 인공신경망을 활용한 DQN은 우수한 성능과 안정적인 성공률을 보였으며, 잠수함 생존 비율도 높은 것을 확인하였다. 따라서 본 논문에서는 기존에 사용하였던 Monte-Carlo 방식과 달리 인공신경망을 이용한 강화학습 알고리즘이 우수한 성능을 보였으며, 이를 통해 좀 더 복잡한 실제 환경에 대해서도 잠수함이 효율적인 어뢰 기만 전술과 회피기동을 하여 어뢰에 의한 피해율이 낮아지는데 기여할 것으로 기대된다.

References

[1] K. H. Liang and K. M. Wang, “Using simulation and evolutionary algorithms to evaluate the design of mix strategies of decoy and jammers in anti-torpedo tactics,” in *Proc. IEEE Winter Simulation Conf.*, pp. 1299-1306, Monterey, CA, USA, Dec. 2006. (https://doi.org/10.1109/WSC.2006.323228)

[2] K. Zhan, B. Yu, and J. Wang, “Simulations of the anti-torpedo tactic of the conventional submarine using decoys and jammers,” *Applied Mechanics and Mater.*, vol. 65, pp. 165-168, Jun. 2011. (https://doi.org/10.4028/www.scientific.net/AMM.65.165)

[3] D. Y. Cho, M. J. Son, J. H. Kang, S. J. Lee, J. H. Cha, S. J. Yoo, and Y. S. Ko, “Analysis of a submarine’s evasive capability against an antisubmarine warfare torpedo using DEVS modeling and simulation,” in *Proc. Spring Simulation Multi Conf.*, vol. 2, pp. 307-315, Norfolk, Virginia, USA, Mar. 2007. (https://doi.org/10.1145/1404680.1404728)

[4] M. Shin, H. Cho, J. Lee, J. S. Lim, S. Lee, W. J. Kim, and W. Hong, “Effectiveness analysis for survival probability of a surface warship considering static and mobile decoys,” *J. Korea Soc. Simulation*, vol. 25, no. 3, pp. 53-63, Aug. 2016. (https://doi.org/10.9709/JKSS.2016.25.3.053)

[5] K. R. Armo, “The relationship between a submarine’s maximum speed and its evasive capability,” *Naval Postgraduate School*, Monterey, California, Jun. 2000.

[6] R. Nian, J. Liu, and B. Huang, “A review on reinforcement learning: Introduction and applications in industrial process control,” *Computers & Chemical Eng.*, vol. 139, pp. 106886, Aug. 2020. (https://doi.org/10.1016/j.compchemeng.2020.106886)

[7] W. B. Powell, *From reinforcement learning to optimal control: A unified framework for sequential decisions*, Cham, Switzerland:

Springer International Publishing, pp.29-74, 2021.
(https://doi.org/10.1007/978-3-030-60990-0_3)

[8] F. Garcia and E. Rachelson, "Markov decision processes," *Markov Decision Processes in Artificial Intell.*, vol. 12, no. 1, pp. 1-38, Feb. 2013.
(https://doi.org/10.1002/9781118557426.ch1)

[9] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, pp. 449-458, Sydney, NSW, Australia, Aug. 2017.
(https://doi.org/10.48550/arXiv.1707.06887)

[10] R. Bellman, "The theory of dynamic programming," *Bulletin of the Am. Math. Soc.*, vol. 60, no. 6, pp. 503-515, Jul. 1954.
(https://doi.org/10.1090/S0002-9904-1954-09848-8)

[11] S. A. Cook, "The complexity of theorem-proving procedures," *Logic, Automata, and Computational Complexity: The Works of Stephen A. Cook*, pp. 143-152, May 2023.
(https://doi.org/10.1145/3588287.3588297)

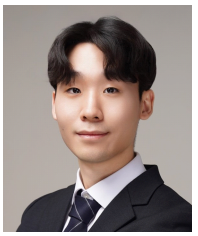
[12] D. Zhao, H. Wang, K. Shao, and Y. Zhu, "Deep reinforcement learning with experience replay based on SARSA," in *Proc. IEEE SSCI*, pp. 1-6, Athens, Greece Dec. 2016.
(https://doi.org/10.1109/SSCI.2016.7849837)

[13] C. J. Watkins and P. Dayan, "Q-Learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279-292, May 1992.
(https://doi.org/10.1007/BF00992698)

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, Dec. 2013.
(https://doi.org/10.48550/arXiv.1312.5602)

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, Feb. 2015.
(https://doi.org/10.1038/nature14236)

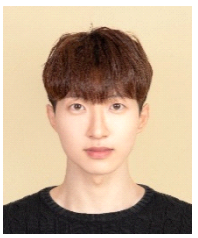
정 재 현 (Jaehyun Chung)



2023년 8월 : 고려대학교 공과대학 전기전자공학부 졸업 (공학사)
2023년 9월~현재 : 고려대학교 공과대학 전기전자공학과 석사과정

<관심분야> Reinforcement Learning, Torpedo Counter Measure Tactics
[ORCID:0009-0008-2202-7200]

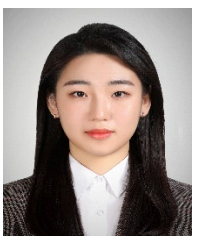
김 규 선 (Gyu Seon Kim)



2023년 2월 : 인하대학교 공과대학 기계항공공학부 항공우주공학과 졸업 (공학사)
2023년 9월~현재 : 고려대학교 공과대학 전기전자공학과 석박사통합과정

<관심분야> Reinforcement Learning, Aerial/ Satellite Communication, Aircraft Embedded System
[ORCID:0000-0002-5559-9749]

박 수 현 (Soohyun Park)



2019년 2월 : 중앙대학교 소프트웨어대학 컴퓨터공학과 졸업 (공학사)
2023년 8월 : 고려대학교 공과대학 전기전자공학과 졸업 (공학박사)

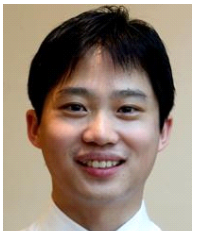
2023년 9월~현재 : 숙명여자대학교 공과대학 소프트웨어학부 조교수

<관심분야> Deep Learning Theory, Network/ Mobility Applications, Quantum Machine Learning, AI-based Autonomous Control
[ORCID:0000-0002-6556-9746]

김 중 헌 (Joongheon Kim)

2004년 2월 : 고려대학교 컴퓨터학과 졸업
2006년 2월 : 고려대학교 컴퓨터학과 석사
2014년 8월 : University of Southern California Computer Science 박사
2016년 3월~2019년 8월 : 중앙대학교 소프트웨어대학 조교수
2019년 9월~현재 : 고려대학교 전기전자공학부 부교수
<관심분야> Stochastic Optimization, Mobility, Reinforcement Learning, Quantum
[ORCID:0000-0003-2126-768X]

윤 원 혁 (Wonhyuk Yun)



2007년 2월 : 한동대학교 전산전자 공학부 학사 졸업
2023년 2월~현재 : 아주대학교 정보통신대학원 석사 재학 중
2007년 2월~현재 : LIG넥스원 해양연구소 수석연구원
<관심분야> Reinforcement Learning, Mobility, Submarine Combat System
[ORCID:0009-0002-0216-1986]