석 사 학 위 논 문

메모리 트랜스포머 Q-학습을 활용한 카트-폴 시스템 제어

한 병 찬

제주대학교 대학원

전 자 공 학 전 공

2024년 2월

메모리 트랜스포머 Q-학습을 활용한 카트-폴 시스템 제어

이 논문을 공학석사 학위논문으로 제출함

한 병 찬

제주대학교 대학원

전 자 공 학 전 공

지 도 교 수 김 호 찬

한병찬의 공학 석사 학위논문을 인준함

2023년 12월

심사위원장 김 경 연 인

위 원 강 민 제 인

위 원 김 호 찬 인

목 차

그 림 목 차

표 목 차

메모리 트랜스포머 Q-학습을 활용한 카트-폴 시스템 제어

한 병 찬

제주대학교 대학원 전 자 공 학 전 공

요약

본 논문은 기존 심층강화학습 알고리즘을 개선하기 위하여 Memory Transformer Q-learnig Network(MTQN)을 제안하였다. MTQN은 sequence 시스템을 보다 효율적으로 모델링하기 위하여 기존 심층 강화학습 모델에 transformer를 결합하여 구성하였으며, 또한 transformer 사용을 위하여 부수적으로 LSTM의 gating mechanism이 이용되었다.

제안한 알고리즘은 대표적인 강화학습 benchmark 환경인 Cart-Pole 시스템에서 비교 분석하였으며, Cart-Pole 시스템은 OpenAI에서 제공하는 gym 라이브러리를 사용하였으며, 강화학습 알고리즘은 pytorch와 numpy로 구현되었다.

제안한 알고리즘의 성능분석을 위하여 대표적인 심층 강화학습 알고리즘인 DQN과 DQN의 변형 알고리즘들을 비교 분석하였다. 실험은 Cart-Pole 시스템의 evaluation score, cart position 그리고 pole angle을 추출하여 비교 분석하였다. Evaluation score를 확인해본 결과 제안된 알고리즘이 가장 빠르고 안정적으로 학습함을 보였다. 또한 cart position과 cart의 누적 이동 거리를 확인해본 결과 제안된 알고리즘에서 cart position이 다른 알고리즘에 비하여 원점을 기준으로 균형적으로 수렴하며 움직였고, 총 누적 거리는 다른 알고리즘에 비해 확연히 짧은 모습을 확인할 수 있었다. Pole angle 또한 제안한 알고리즘에서 다른 알고리즘과 비교하여 빠르게 원점 근처로 수렴하였으며 그리고 학습된 알고리즘에서 수행한 결과를 추출한 자료에서도 원점을 기준으로 대칭적인 분포한다는 사실을 확인하였다.

1. 서론

1.1 연구 배경 및 목적

강화학습은 다양한 유형의 게임, 로봇 제어, 주식 거래, 자원 할당, 추천 시스템, 자연어 처리 등 현실 세계의 다양한 문제에 적용될 수 있다[1-4]. 강화학습을 적용할 수 있는 응용 분야는 무궁무진하지만, 한동안 연구자들은 강화학습을 이용해 실제 문제를 해결하는 데 어려움을 겪어왔다. 단순한 상태 공간과 행동 공간을 가진 문제의 경우 딥러닝을 사용하지 않는 알고리즘으로도 좋은 결과를 얻을 수 있었지만, 현실 세계의 문제는 매우 복잡하므로 주어진 시간 내에 답을 도출하는 것이 불가능하다는 근본적인 한계가 있었다. 이후 이미지 처리를 위한 신경망 학습 기술과 Convolutional Neural Network가 개발되었고[5-6], 딥러닝과 강화학습을 결합한 DQN이 등장하였다. DQN은 강화학습을 experience replay 기법과 target network 생성을 통해 지도학습의 형태로 학습시킬 수 있게 만들었다[7]. 이후 연구를 통해 DQN의 학습 방식은 Q-value에 overestimate가 발생한다는 문제점이 존재한다는 사실이 밝혀졌고 이러한 문제를 해결하기 위하여 DDQN이 등장하였다[8]. 이후 DDQN의 신경망 구조가 비효율적이라는 사실이 밝혀졌고 Dueling DDQN이 등장하게 되었다. Dueling DDQN은 DDQN의 신경망 구조가 Q 함수를 직접 학습하지 않고 V 함수와 A 함수를 통해 간접적으로 Q 함수를 근사할 수 있도록 만들었다[9]. 이 3가지 알고리즘은 심층 강화학습의 대표적인 알고리즘이 되었고, 현재 이를 적용하여 현실 세계의 다양한 문제를 해결하는 많은 연구가 주목받고 있다.

Transformer[10]는 본래 기계 번역을 위한 Seq2Seq 모델로, RNN을 대체할 모델로 제안되었다. 이후 이미지를 패치 단위로 나누어 시퀀스 데이터로 만들어 vanilla transformer 형태를 이미지 처리 분야에 적용한 vision transformer가 등장하였고[11], 많은 후속 연구들을 통해 긴 sequence를 잘 모델링 할 수 있고 확장성이 뛰어나다는 점 덕분에 여러 task에서 CNN 및 RNN보다 성능이 우수함이 증명되어 자연어 처리(NLP), 컴퓨터 비전(CV), 음성 처리 등 다양한 task에서 널리 채택되고 있다

[12-16]. 특히 최근 폭발적인 관심을 받고 있는 chatGPT[17]의 경우도 transformer를 변형하여 학습시킨 모델로, transformer는 딥러닝 분야에서 매우 중요한 위치를 차지하였다.

본 논문은 transformer의 장점을 이용한 강화학습 에이전트인 MTQN을 제안하고, 대표적인 가치 기반 심층 강화학습 알고리즘인 DQN과 DQN의 변형 알고리즘들과 비교 분석하여 MTQN이 DQN과 DQN의 변형 알고리즘보다 대표적인 강화학습 benchmark 환경인 Cart-Pole 시스템[18-19]을 더 잘 제어할 수 있음을 보인다.

1.2 논문의 구성

본 논문은 총 5장으로 구성되어 있으며 각 장의 구성은 다음과 같다.

제 1장은 본 논문의 서론에 해당하며 연구 배경 및 목적과 논문의 구성에 대하여 설명한다.

제 2장은 Cart-Pole 시스템에 대하여 설명한다.

제 3장은 기존에 Cart-Pole 시스템 제어에 사용하던 테이블 기반 Q-Learning과 딥러닝 기반의 Q-Learning에 대하여 설명하고 이를 개선 시킬 신경망 구조를 제안한다.

제 4장은 기존 알고리즘을 사용한 제어 결과와 제안한 구조를 사용한 제어 결과를 3가지 평가 기준을 통해 비교 분석한다.

제 5장은 실험 결과를 바탕으로 결론을 제시한다.

2. Cart-Pole 시스템

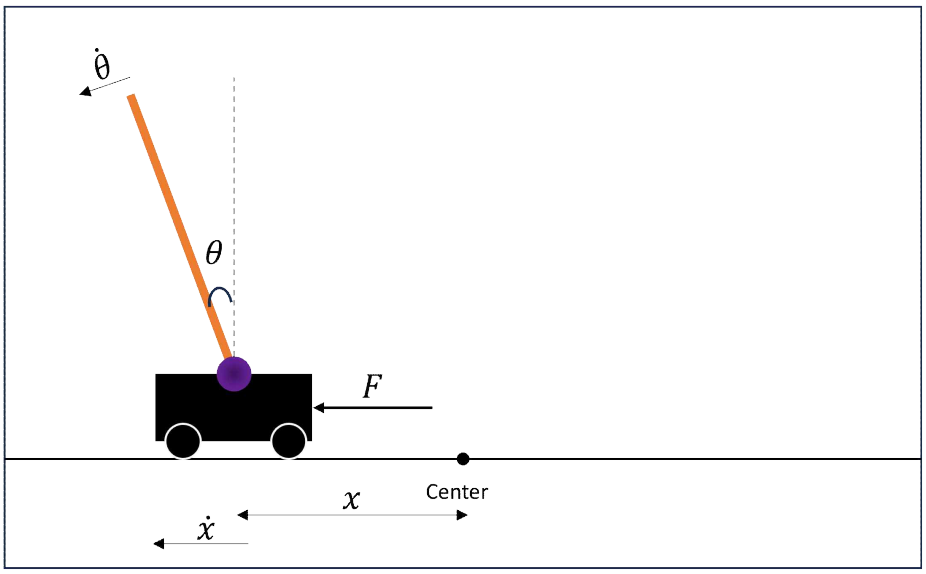


Fig. 1 The Cart-Pole system.

Cart-Pole 시스템은 강화학습 알고리즘의 성능을 평가하는 데 널리 사용되는 Benchmark 환경이다[18-19]. Cart-Pole 시스템은 cart와 cart에 수직으로 세워지는 pole로 구성되는데, cart는 마찰 없이 수평선 좌우로 자유롭게 움직일 수 있고 pole이 cart의 움직임에 따라 기울기가 변한다. 강화학습 에이전트는 cart를 좌우로 움직이는 행동을 취하여 cart가 정해진 구역을 벗어나지 않고 pole이 균형을 유지하도록 하여 보상을 얻게 된다. Cart-Pole 시스템은 식 2.1, 식 2.2와 같이 표현할 수 있다[20].

$$\ddot{\theta} = \frac{(M+m)gsin\theta - \cos\theta[F + ml\dot{\theta}^2\sin\theta]}{(\frac{4}{3})(M+m)l - ml\cos^2\theta} \tag{2.1}$$

$$\ddot{x} = \frac{F + ml[\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta]}{M+m} \tag{2.2}$$

여기서 M은 cart의 질량으로 $0.711\,kg$, m은 pole의 질량으로 $0.209\,kg$이다. 또한 g는 중력 가속도로 $9.8\,m/s^2$이고 F는 cart에 적용되는 힘의 크기로 $\pm 10\,Newtons$, l은 cart의 중앙에서 pole의 무게중심까지의 거리 $0.326\,m$이다. 시뮬레이션에 사용된 타임 스텝 사이의 간격 τ는 $0.02\,seconds$로 설정하였다.

3. Transformer를 활용한 Cart-Pole 시스템 제어

3.1 강화학습

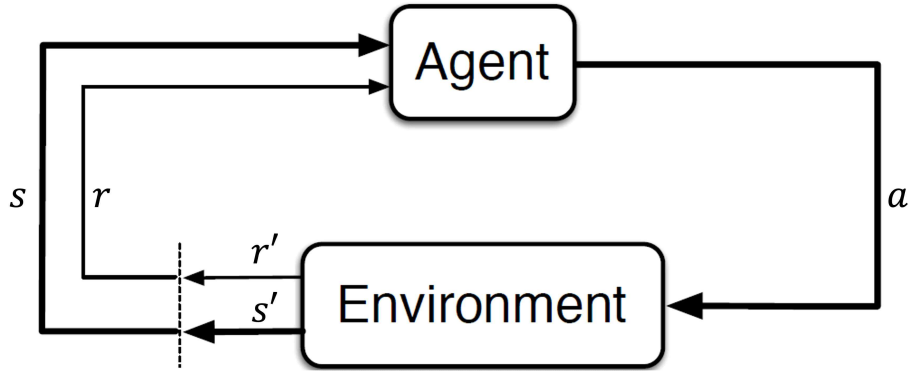


Fig. 2 The agent-environment interaction in reinforcement learning.

강화학습은 순차적인 의사결정 문제를 해결하는 방법이다. 강화학습은 환경과 에이전트로 구성되며 Markov Decision Process(MDP)로 모델링 된다[21]. MDP에서 t시점의 상태(s, $s \in S$)와 행동(a, $a \in A$)이 주어졌을 때, 다음 상태(s')로 갈 확률은 t시점 이전에 상호작용을 하며 얻은 값과는 무관하다는 특성을 Markov Property라 하고 이를 식 3.1과 같이 나타낼 수 있다.

$$P(s_{t+1}|S_t,A_t) = P(s_{t+1}|S_t,A_t,S_{t-1},A_{t-1},\dots) \tag{3.1}$$

환경은 s를 나타내는 정보를 만들어내고, 에이전트는 s를 관측하고 이를 바탕으로 a를 선택하여 환경과 상호작용하게 된다. 이때 에이전트가 a를 선택하는 방법을 정책($\pi(a|s)$)이라 한다. 에이전트가 환경에 a를 취하면 환경은 이 a를 바탕으로 식 3.2를 통해 s'으로 전이하게 되고, 식 3.3에 의해 이에 따른 보상(r)이 에이전트에게 주어진다.

$$p(s'|s,a) = P(S_t = s'|S_{t-1} = s, A_{t-1} = a) \tag{3.2}$$

$$r(s,a) = \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a] \tag{3.3}$$

s, a, r이 주어지는 한 사이클을 타임 스텝이라 하고 (s, a, r, s') 쌍을 경험 튜플이라 한다. 강화학습 문제는 에이전트가 매 타임 스텝에서 좋은 s를 선택하고 한 에피소드에서 받는 감가된 r의 총합인 반환 값(G)을 최대화하여야 하며, 식 3.4와 같이 나타낼 수 있다.

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-1} R_T \\ &= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned} \tag{3.4}$$

감가율($\gamma \in [0, 1]$)은 현재 타임 스텝에서의 가치를 계산할 때 미래 타임 스텝에서의 r을 작게 만들어 예측된 G에 대한 분산을 줄이는 역할을 한다.

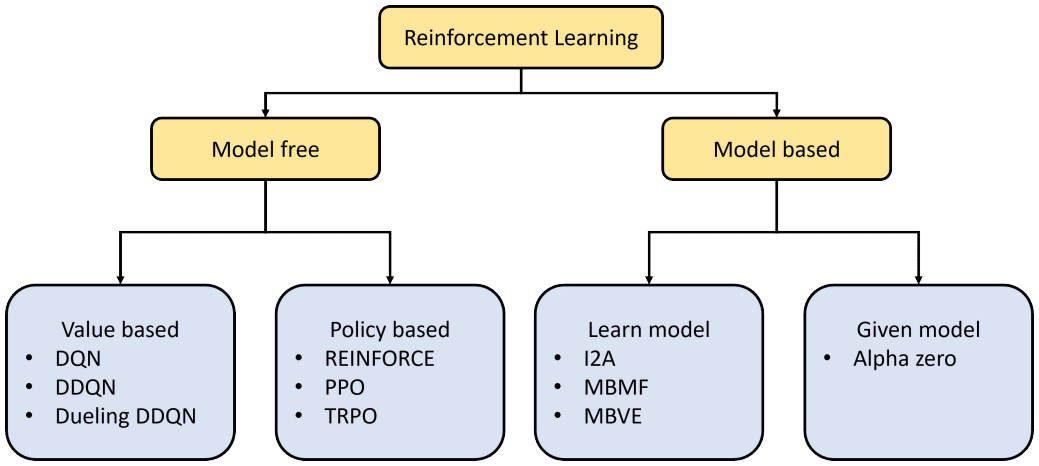


Fig. 3 Classification of reinforcement learning based on learning function.

강화학습은 여러 기준으로 구분할 수 있는데 알고리즘이 학습하는 주요 함수가 무

엇인지 혹은 알고리즘이 현재 정책만을 사용하여 학습을 진행하는 지로 구분할 수 있다. 알고리즘이 학습하는 주요 함수가 무엇인지를 기준으로 하면 정책 기반 학습, 가치 기반 학습, 모델 기반 학습으로 구분할 수 있다.

정책 기반 학습은 정책을 학습하는 방법인데 상태를 바탕으로 좋은 행동을 도출한다. 대표적인 알고리즘으로는 REINFORCE[22]가 있다. 정책 기반 학습은 에이전트의 목적을 직접 최적화하므로 매우 일반적인 최적화 방법이고, 이산적 행동 공간을 가진 환경과 연속적 행동 공간을 가지는 환경 모두에 적용할 수 있다는 장점이 있다. 하지만 정책의 분산이 크고 표본 비효율적이라는 단점도 존재한다.

가치 기반 학습은 학습을 통해 가치 함수를 추정하고 이를 바탕으로 상태-행동 쌍을 평가하고 정책을 생성한다. 대표적인 알고리즘으로는 DQN, DDQN, Dueling DDQN 등이 있다. 가치 기반 학습은 정책 기반 알고리즘보다 분산이 작아 표본 효율적이라는 장점이 있지만, 최적 정책으로의 수렴을 보장할 수 없고 이산적 행동 공간을 가진 환경에만 적용할 수 있다는 단점이 존재한다.

모델 기반 학습은 환경의 전이 역학에 대한 모델을 학습하여 에이전트가 환경 모델의 궤적을 예측하게 하는 방법이다. 하지만 큰 상태 공간과 행동 공간을 가지는 환경 모델을 학습하기가 매우 어렵기 때문에 3가지 학습법 중 가장 제약이 큰 학습법이다.

그 외에도 2가지의 학습법의 장점만을 취하여 결합한 형태의 학습법도 존재한다. 대표적으로 정책 기반 학습과 가치 기반 학습을 결합한 Actor-Critic[23], A3C, A2C[24] 가 존재한다.

알고리즘이 현재 정책만을 사용하여 학습을 진행하는지에 따라 구분하면 on-policy 방법과 off-policy 방법으로 구분할 수 있다. 알고리즘이 현재의 정책으로부터 생성된 데이터만을 가지고 학습을 진행하면 on-policy 알고리즘이라 한다. On-policy 알고리즘은 현재 정책만을 가지고 학습을 진행하기 때문에 이전 타임 스텝에서 생성된 데이터는 사용하지 않는다. 따라서 더 많은 데이터를 필요로 하기 때문에 표본 비효율적이다. 이와 반대인 off-policy 알고리즘은 표본 효율적이기는 하지만 데이터를 저장하기 위한 메모리가 훨씬 많이 필요하여 메모리 효율이 떨어진다.

가치 기반 학습은 상태-가치함수 또는 행동-가치함수를 학습하는 방법으로, 학습을 통해 가치함수를 추정하고 이를 바탕으로 (s,a) 쌍을 평가하고 정책을 생성한다.

상태-가치함수는 $V^{\pi}(s)$ 또는 V 함수라 부르며 s에서 모든 a에 대하여 에이전트가 얻을 수 있는 G에 대한 기대치를 말하고 식 3.5와 같이 나타낼 수 있다.

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}[G_t|S_t = s] \\ &= \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...|S_t = s] \\ &= \mathbb{E}_{\pi}[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V^{\pi}(s')], \quad \forall s \in S \end{aligned} \tag{3.5}$$

행동-가치함수는 $Q^{\pi}(s,a)$ 또는 Q 함수로 부르며 s에서 특정 a를 취했을 때, 에이전트가 얻을 수 있는 G에 대한 기대치를 말하고 식 3.6과 같이 나타낼 수 있다.

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}_{\pi}[G_t|S_t = s, A_t = a] \\ &= \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...|S_t = s, A_t = a] \\ &= \mathbb{E}_{\pi}[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a] \\ &= \sum_{s',r} p(s',r|s,a)[r + \gamma V^{\pi}(s')], \quad \forall s \in S, \ \forall a \in A(s) \end{aligned} \tag{3.6}$$

식 3.6과 식 3.7에 의해 $V^{\pi}(s)$는 식 3.7과 같이 재정의될 수 있다. 즉, $V^{\pi}(s)$가 s에서 가능한 모든 $Q^{\pi}(s,a)$의 기댓값임을 의미한다.

$$V^{\pi}(s) = \sum_a \pi(a|s) Q^{\pi}(s,a) \tag{3.7}$$

또한 $V^{\pi}(s)$와 $Q^{\pi}(s,a)$를 이용하면 행동-이점 함수를 구할 수 있다. $A^{\pi}(s,a)$ 또는 A 함수라 불리는 행동-이점 함수는 정책 π를 수행하는 동안 s에서 a를 취했을 때 얻을 수 있는 이점으로, 동일한 정책 π에서 a에 대한 가치와 s에 대한 가치의 차이를 의미한다. $A^{\pi}(s,a)$는 식 3.8과 같이 나타낼 수 있다.

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \tag{3.8}$$

V 함수와 Q 함수는 모두 행동에 대해서 표현된다. 이 두 함수가 최적의 상태일 때를 이상성이라 부른다. 즉 모든 s에 대하여 다른 정책들보다 반환값의 기댓값이 더 큰 상태를 의미한다. 최적 V 함수($V^{*}(s)$)는 모든 상태에 걸쳐, 모든 정책에 대한 최대의 가치를 가지는 V 함수를 말하며 식 3.9와 같이 표현할 수 있다. 최적 Q 함수($Q^{*}(s,a)$)는 모든 (s,a) 쌍에 걸쳐 모든 정책에 대한 최대의 가치를 가지는 Q 함수를 말하며 식 3.10과 같이 표현할 수 있다.

$$\begin{aligned} V^{*}(s) &= \max_{\pi} V^{\pi}(s), \qquad (\forall\, s \in S) \\ &= \max_{a} \sum_{s',r} p(s',r|s,a)[r + \gamma V^{*}(s')] \end{aligned} \tag{3.9}$$

$$\begin{aligned} Q^{*}(s,a) &= \max_{\pi} Q^{\pi}(s,a), \qquad (\forall\, s \in S,\ \forall\, a \in A) \\ &= \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} Q^{*}(s',a')] \end{aligned} \tag{3.10}$$

식 3.9와 식 3.10을 이용하면 이상적인 정책을 얻을 수 있다.

가치 기반 학습에서 에이전트가 Q 함수에 대해 탐욕적으로 행동하는 것에는 한가지 문제가 있다. 바로 에이전트가 전체 상태-행동 공간을 충분히 탐험하지 못할 수도 있다는 점이다. 예를 들어 에이전트가 s에서 항상 동일한 a를 선택한다면 에이전트가 경험하지 못하는 (s,a)쌍들이 무수히 많이 존재할 것이다. 이러한 경우 경험하지 못한 (s,a) 쌍에 대한 Q 함수의 추정값이 임의로 초기화되기 때문에 정확하지 않을 것이다. 이러한 문제를 해결하기 위하여 학습 시에는 완전한 탐욕적 정책이 아닌 ε-탐욕적 정책을 사용한다. ε-탐욕적 정책이란 에이전트가 $1-\varepsilon$의 확률로는 탐욕적 정책을 택하고 ε의 확률로는 임의의 행동을 택하는 정책이다. 여기서 ε은 탐험 확률이라 하고 무작위성을 주어 에이전트가 다양한 (s,a) 쌍을 경험하게 해준다. 하지만 이러한

방식에는 단점도 존재한다. 확률성이 존재하기 때문에 탐욕적 정책보다 좋지 않을 수 있다. 이러한 관계를 탐험-활용 균형이라 한다. 탐험-활용 균형을 적절히 조절하기 위해서는 학습 초반에는 적당히 큰 ε 값을 설정하여 학습을 진행하여 에이전트가 빠르게 환경을 탐험할 수 있도록 하고 점점 ε을 감소시켜 많은 타임 스텝이 지난 후에는 활용에 초점을 맞추도록 하는 방식을 사용하면 된다[25].

3.2 Deep Learning 기반 Q-Learning

심층 강화학습 이전의 Q-Learning은 Q 함수를 테이블 형식으로 저장하여 학습하는 테이블 기반 방식을 사용하였다. 이러한 방식은 s 공간과 a 공간이 커지게 되면 테이블의 크기 또한 함께 커지게 되어 실제 환경처럼 매우 복잡한 환경이 주어질 경우, 학습하는 것이 거의 불가능에 가까웠다. 하지만 이후 딥러닝의 발전으로 테이블 대신 Q 함수 자체를 근사하는 DQN과 같은 알고리즘들이 등장하며 이러한 문제를 해결할 수 있게 되었다.

3.2.1 테이블 기반 Q-Learning

Q-Learning은 강화학습 에이전트가 시행착오를 통해 최적의 행동을 학습할 수 있도록 하는 알고리즘이다. 이 알고리즘은 주어진 상태에서 특정 행동을 취했을 때 예상되는 G를 나타내는 Q 함수를 사용한다. Q 함수를 반복적으로 업데이트함으로써 Q-Learning은 에이전트 시간이 지남에 따라 더 나은 의사결정을 내릴 수 있도록 한다.

Q 함수는 일반적으로 행이 상태에 해당하고 열이 행동에 해당하는 2차원 Q 테이블 구조를 사용하여 표현되며, 초기 Q 테이블은 임의의 값이나 0으로 초기화된다. Q 테이블에서 상태-행동 쌍에 대한 Q 값은 에이전트가 s에서 a를 수행하여 얻을 수 있는 G를 나타낸다.

Q-Learning 알고리즘은 탐색 및 활용 프로세스를 통해 작동한다. 탐색 단계에서는 에이전트가 무작위 a를 취하여 환경에 대한 정보를 수집하고 그에 따라 Q 테이블을 업데이트한다. 에이전트가 더 많은 탐색을 수행하면 학습된 Q 값을 활용하여 의사결정을 내리고 누적 보상을 극대화하는 활용 단계로 점차 전환된다.

$$r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \tag{3.11}$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{3.12}$$

식 3.11을 temporal difference라고 하고 이를 이용하여 식 3.12와 같은 Q-Learning 업데이트 수식을 얻을 수 있다. 이렇게 temporal difference를 사용하는 학습 방식을 TD 학습이라고 한다. 많은 타임 스텝 동안 temporal difference를 점진적으로 Q 테이블에 더해주어 업데이트하면 결과적으로 최적의 값으로 수렴하게 되는데, 이는 에이전트가 주어진 환경에서 G를 극대화하는 전략인 최적의 정책을 학습했음을 의미한다.

이후 Q-Learning은 실제 환경의 문제를 해결하기 위해 수많은 확장 및 개선이 이루어졌다. 실제 환경의 상태 공간은 매우 복잡하여 테이블 구조로 표현할 수 없기 때문이다. 따라서 이러한 문제를 해결하기 위하여 Q-Learning에 딥러닝을 결합한 형태인 DQN이 등장하였다. DQN은 대규모 상태 및 행동 공간을 처리하기 위해 Q 테이블 대신 신경망을 사용하여 함수 근사화를 하였다. 이러한 접근 방식은 복잡한 게임과 같은 영역에서 대단한 성공을 거두었다.

3.2.2 DQN

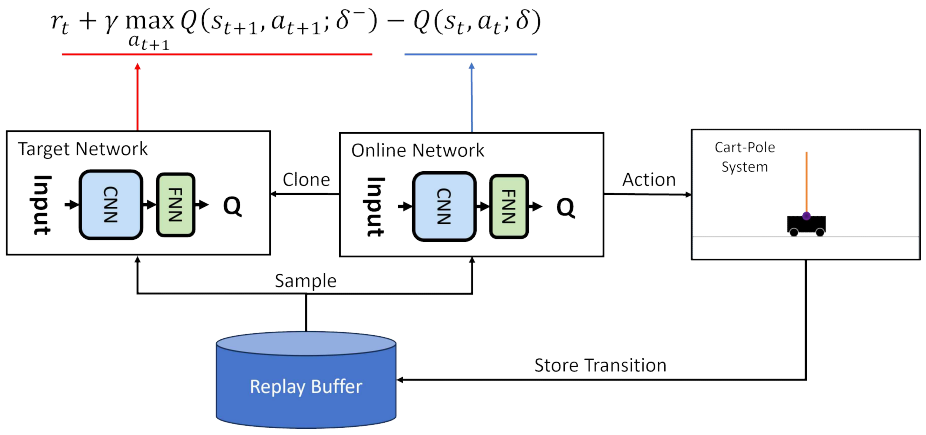


Fig. 4 The schematic diagram of DQN.

DQN은 강화학습을 지도학습처럼 만들려는 시도에서 출발하였다. 심층 강화학습에서 나타나는 문제는 두 가지가 있는데, 첫 번째 문제는 데이터에 대한 IID 가정의 불

일치성이다[7]. 지도학습에 이용되는 최적화 기법은 보통 학습 데이터의 샘플들이 독립적이면서 동일한 분포를 가져야 하는데, 강화학습의 데이터 샘플은 순차적인 성향을 지니기 때문에 $t+1$ 시점에서의 샘플이 t 시점에서의 샘플과 연관되어 독립적이지 않다. 또한 강화학습에서 행동을 생성하는 정책은 시간에 따라 계속 변화하기 때문에 이에 따라 샘플들의 분포도 따라 변화하여 샘플들이 동일한 분포를 띄지 않게 된다. 두 번째 문제는 target이 고정되어 있지 않다는 점이다.

$$r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \delta) - Q(s_t, a_t; \delta) \tag{3.13}$$

지도학습은 target label이 고정된 값으로 주어지는데, 식 3.13과 같이 Q 테이블을 θ를 가중치로 가진 신경망으로 나타낼 경우, 학습 스텝마다 예측 Q의 가중치가 변하면서 같은 가중치를 사용하는 target 또한 계속 변화하게 된다.

DQN은 이러한 문제들을 해결하기 위하여 1) Experience Replay 기법을 사용하였다. Experience Replay는 여러 타임 스텝에 대한 경험 샘플을 Replay Buffer에 저장해 두었다가 샘플링 하는 방식이다. 이때 Replay Buffer의 크기가 적당히 크면 여기서 뽑아낸 데이터 샘플이 독립적이면서 동일한 분포를 띄게 되므로 첫 번째 문제를 해결할 수 있다. 2) Target Network를 만들어 사용하였다. Target이 고정되지 않고 계속 변화하므로 target을 고정하기 위하여 target network를 만들어 target을 고정한다. 여기서 target network와 최적화하려는 online network는 같은 구조를 사용하고, 일정 타임 스텝마다 online network와 target network를 동기화 시키는 방식을 사용한다. 이러한 기법을 사용하면 target이 변화하기 전에 online network가 조금씩 target을 향해 움직일 수 있게 된다. 이를 식 3.14와 같이 표현할 수 있으로 DQN의 구조는 Fig. 4와 같다.

$$r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \delta^{-}) - Q(s_t, a_t; \delta) \tag{3.14}$$

3.2.3 DDQN

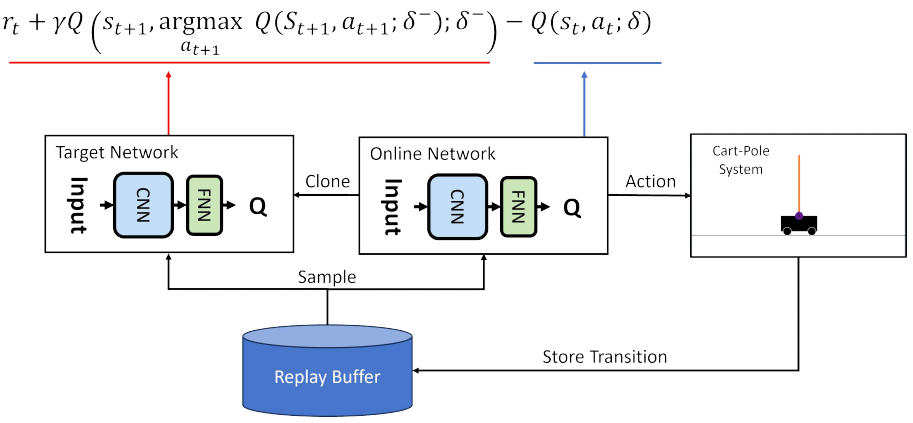


Fig. 5 The schematic diagram of DDQN.

DQN은 off-policy TD 학습 방식을 사용하여 추정 Q-value의 최대를 찾는 연산을 수행한다. 이때 특정 Q-value에 오차가 존재하면 Q-value의 최댓값에 편향이 발생하여 overestimate가 이루어지기 때문에 학습에 좋지 않은 영향을 끼칠 수 있다[8]. Q-value의 오차가 생기는 이유는 1) 신경망을 사용하기 때문에 함수의 완벽한 근사가 이루어지지 않을 수 있고, 2) 환경 자체에 노이즈가 포함될 수 있다. 또한 3) 에이전트가 환경을 완전히 탐험하지 않았을 수도 있다는 이유가 존재할 수 있다.

이러한 점을 개선하기 위하여 DDQN 알고리즘이 나왔다. DDQN은 Q-value의 overestimate 문제를 해결하기 위하여 각기 다른 경험을 사용하여 2개의 Q 함수를 학습한다. 첫 번째 Q 함수는 Q-value를 최대로 만드는 행동을 선택하고, 두 번째 Q 함수는 이 행동을 이용하여 Q-value를 계산한다. 이처럼 각기 다른 경험을 사용하여 학습된 Q 함수는 Q-value의 편향을 줄일 수 있다. 이와 같은 방식으로 학습할 경우, 학습을 위한 online network와 target network가 각각 2개씩 필요하게 된다. 하지만 총 4개의 신경망을 학습시키는 것은 많은 시간이 소요되므로 실제 학습 시 online-network는 최적의 Q-value를 갖게 하는 행동을 찾고 target-network를 사용하여 이 행동을 평가하는 방식으로, 총 2개의 신경망만을 사용한다. 이러한 방식은 target-network를 Q-value 평가에 사용하여 안정적으로 target을 고정할 수 있게 한다. DDQN의 구조는 Fig. 5와 같다.

또 한 가지 개선점은 손실함수로 Huber Loss를 사용한 것이다. DQN은 손실함수로 MSE를 사용하였는데, MSE는 손실이 0으로 갈수록 경사가 감소하여 최적화를 할 때 유리하다는 장점이 있기도 하지만, 작은 오차보다는 큰 오차에 많은 페널티를 부여하기 때문에 target이 지속적으로 변화하고 신경망을 통해 Q 함수를 근사하는 심층 강화학습에는 적절하지 않을 수 있다. Huber Loss는 MSE와 MAE를 합친 함수인데, 앞서 언급한 MSE의 장점과 모든 구간에서의 경사가 일정하다는 MAE의 장점을 모두 사용할 수 있다는 이점이 있다.

3.2.4 Dueling DDQN

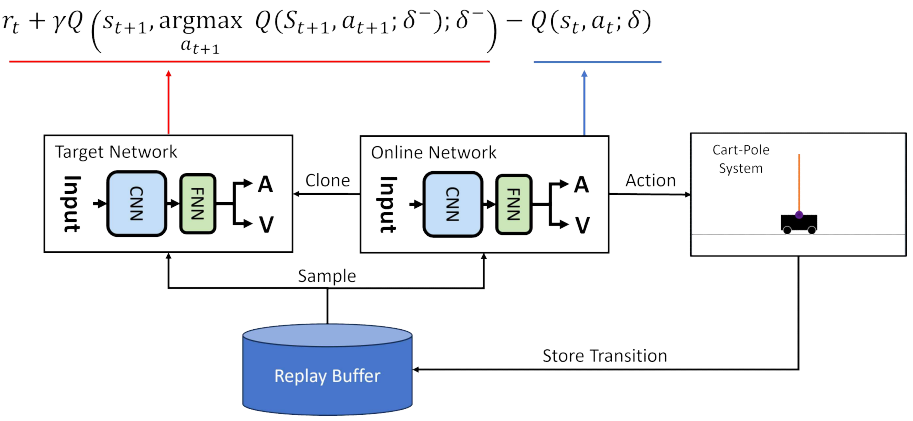


Fig. 6 The schematic diagram of Dueling DDQN.

Dueling DDQN은 알고리즘이 아닌 신경망 구조를 변형하여 앞서 언급한 알고리즘들을 개선한 알고리즘이다. Dueling DDQN의 구조는 Fig. 6과 같다. Q 함수는 모든 행동이 공유하는 가치인 V 함수와 각 행동이 갖는 가치인 A 함수로 분리할 수 있다. 기존 DQN과 DDQN의 경우 Q 함수를 행동별로 분리하여 학습하지만 사실 V 함수는 모든 행동에 대하여 같은 값을 가지기 때문에 비효율적이다[9]. 이러한 비효율성을 해결하기 위하여 Dueling Network가 등장하였다. Dueling Network는 두 개의 분류기를 가지는데, 하나는 V 함수에 대한 분류기이고, 다른 하나는 A 함수에 대한 분류기이다. 여기서 나온 두 개의 출력을 이용하면 효율적으로 Q 함수를 근사할 수 있다.

Q 함수를 근사하는 방법은 식 3.7과 같지만, 실제로는 V 함수와 A 함수를 더한 값에서 A 함수의 평균을 빼는 방식으로 Q 함수를 근사하여 안정적인 최적화 과정을 수행할 수 있게 한다.

Dueling Network는 A 함수를 이용하여 Q 함수를 추정하므로 유사한 가치를 가진 행동들이 많을 때 효과적이다. 신경망은 근사할 때 오차를 가지게 되는데 DQN과 DDQN의 신경망 구조는 상태-행동 쌍이 각자 나뉘어 있으므로 그에 따른 오차들도 잠재적으로 다른 성향을 띈다. V 함수는 특정 상태에서 모든 행동에 대한 Q 함수의 일부이기 때문에 dueling network를 사용하게 되면 발생하는 함수의 오차와 분산을 줄일 수 있다.

3.3 Transformer 기반 Q-Learning

Transformer는 다양한 분야에서 사용되는 딥러닝 모델이며 긴 sequence를 잘 모델링 할 수 있고 확장성이 뛰어나다는 장점을 가진 모델이다. 강화학습에서 신경망의 입력은 타임 스텝마다의 경험 튜플로 구성되므로 sequence가 있는 데이터라 볼 수 있기 때문에 본 논문에서는 DQN의 신경망 구조를 Transformer와 결합하여 긴 sequential 데이터를 잘 모델링 할 수 있게 설계한 MTQN을 제안하였다.

3.3.1 Transformer

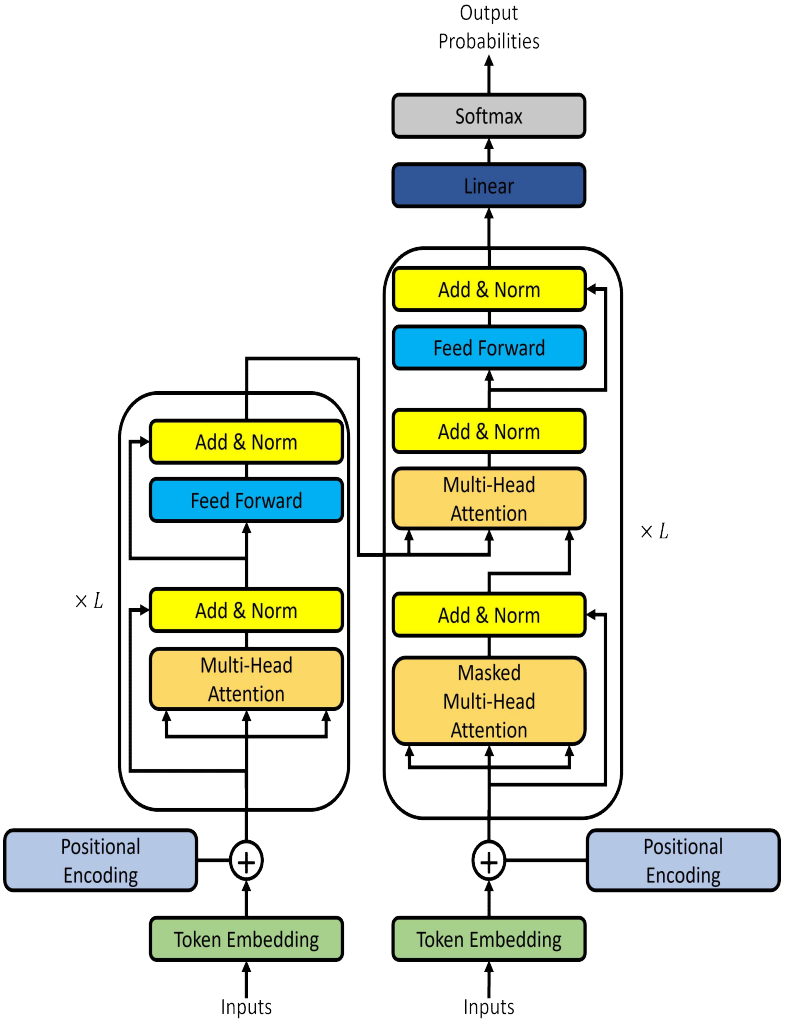
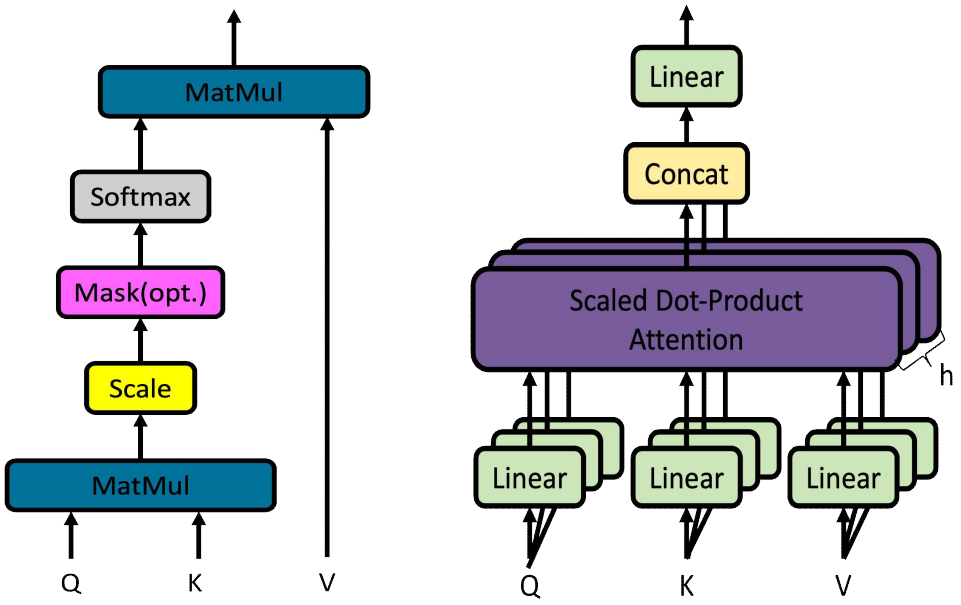


Fig. 7 The architecture of transformer.

Transformer는 다양한 분야에서 사용되는 딥러닝 모델이며 자연어 처리(NLP), 컴퓨터 비전(CV), 음성 처리 등 다양한 분야에서 널리 채택되고 있다. Transformer는 본래 기계 번역을 위한 Seq2Seq 모델로, RNN을 대체할 모델로 제안되었으며[10] 더 나아가 여러 연구를 통해 이미지 처리에서도 CNN보다 성능이 우수함이 입증되었다[26-30]. Transformer의 가장 큰 장점은 긴 sequence를 잘 모델링 할 수 있고 확장성이 뛰어나다는 점이다. Vanilla Transformer는 인코더와 디코더로 구성되어 있으며 각각은 Multi-Head Self-Attention 모듈과 FFN 모듈로 구성된다. Transformer의 인코더는 소스 시퀀스를 인코딩하여 디코더로 보내주는 역할을 하고 디코더는 인코더에서 받은 정보를 바탕으로 타켓 시퀀스를 생성하는 역할을 한다. Transformer의 구조는 Fig. 7과 같다.

기존의 RNN은 단어의 위치에 따라 단어를 순차적으로 입력받아 처리하여 각 단어의 위치 정보를 가질 수 있었다[31]. 하지만 transformer의 경우 단어를 순차적으로 받지 않고 문장 전체를 입력받기 때문에 순서에 대한 정보를 가지고 있지 않으므로 단어의 순서에 대한 정보를 제공하기 위하여 임베딩 벡터에 positional encoding을 더하여 준다.

Transformer의 main 블록 내부의 연결은 residual 연결로 구성되어 있는데, 이는 모듈의 입력과 출력을 더하는 구조를 말한다. 이러한 연결 구조를 사용할 경우 여러 개의 계산 경로가 생기므로 다양한 관점에서 블록 계산을 수행하여 입력의 feature를 잘 추출할 수 있게 한다.



(a) Scaled dot-product attention. (b) Multi-head attention.

Fig. 8 The scaled dot-product attention and multi-head attention.

Multi-Head Self-Attention 모듈은 Transformer의 핵심 구조로 Fig. 8(b)와 같다. Self-Attention 모듈은 입력으로 받은 Sequence에서 어느 부분이 가장 중요한지를 판단하는 모듈로, 모델이 긴 Sequence에서 feature를 더욱 잘 추출할 수 있게 한다. 입력 시퀀스 $X \in \mathbb{R}^{n \times d} (n = sequence\ length,\ d = embedding\ dimension)$가 주어질 때, Self-Attention 모듈은 입력에 각각 W_Q, W_K, W_V를 행렬곱하여 $Q \in \mathbb{R}^{n \times d_q}$, $K \in \mathbb{R}^{n \times d_k}$, $V \in \mathbb{R}^{n \times d_v}$로 매핑한다. Q는 분석의 대상이 되는 단어에 대한 가중치 벡터, K는 각 단어가 쿼리에 해당하는 단어와 얼마나 연관이 되어있는지, V는 K의 의미를 나타내는 가중치 벡터라고 해석할 수 있다. Fig. 8(a)는 식 3.15와 같이 나타낼 수 있다. Fig. 8(a)의 Mask는 디코더 파트에서 사용하는 방식으로, 순차적 입력값이 아닌 전체 입력값을 한 번에 받는 transformer의 특성상 미래 시점의 데이터를 예측 과정에서 제외해주어야 하므로 masking 방식을 사용하여 미래 시점의 데이터에 대한 attention score가 0이 되도록 하는 방식이다.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{D_k}})V \tag{3.15}$$

이때 $softmax(\frac{QK^T}{\sqrt{D_k}})$를 attention score라 한다. Attention score는 한 문장 내에서 특정 단어와 모든 단어와의 관련성의 정도를 나타내며 이 attention score가 transformer 모델이 긴 sequence 입력을 잘 모델링 할 수 있는 핵심 부분이다. 이러한 self-attention 모듈을 동시에 여러 번 수행하는 방식을 multi-head-attention이라 한다.

Vision transformer는 처음으로 이미지 처리에 CNN 대신 transformer 모델을 적용한 구조로 이미지 처리의 판도를 바꾼 계기가 되었다. Vision transformer의 구조는 Fig. 9와 같다.

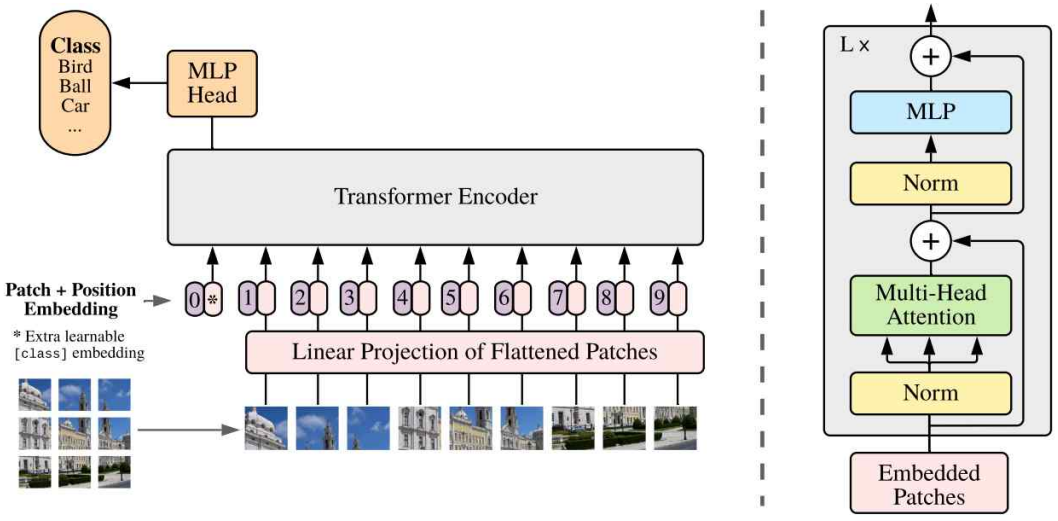
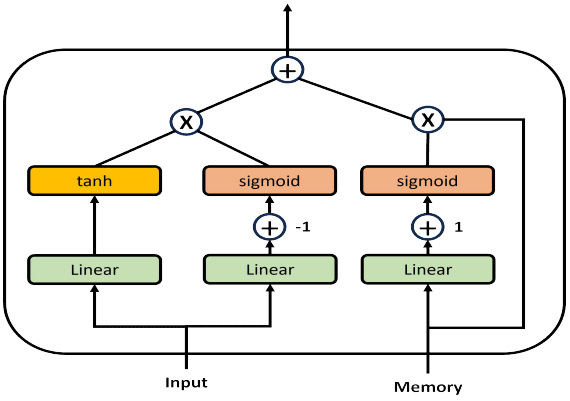


Fig. 9 The architecture of Vision Transformer.

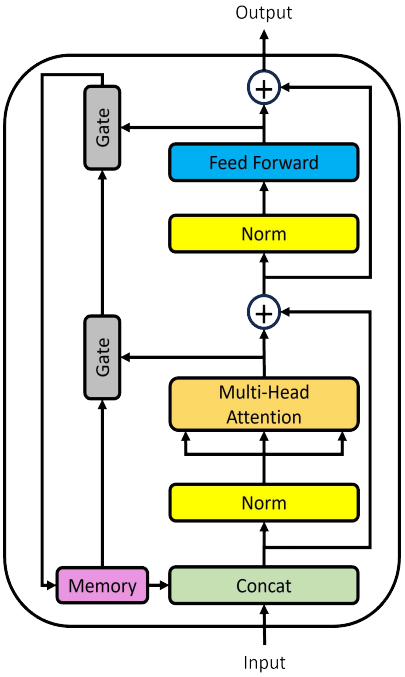
Vision Transformer는 이미지를 텍스트 시퀀스처럼 사용하여 기존 텍스트 시퀀스에 사용되던 transformer의 인코더 부분을 그대로 가져와 사용하였다 한 가지 차이점은 기존에는 attention 모듈과 FFN 모듈을 거친 후 정규화를 진행하였지만 vision transformer의 경우 정규화를 먼저 진행 후 나머지 모듈의 입력으로 사용하였다[11]. 이미지 분류의 과정은 다음과 같다. 먼저 입력 이미지 $x \in \mathbb{R}^{H \times W \times C}$를 고정된 크기의 패치 $x_p \in \mathbb{R}^{N \times (P^2 C)},\ N = HW/P^2$로 나누어 준다. 이때 H는 이미지의 높이, W는 이미지의

너비, C는 채널 수를 의미하고, P는 패치의 높이와 너비, N은 패치의 총 개수를 의미한다. 이렇게 이미지를 패치 단위로 나누어 시퀀스가 있는 데이터로 만들어 주는 작업이 vision transformer의 핵심 기법이었다. 나머지 작업은 기존과 거의 동일하다. 각 패치를 선형 임베딩하고 vanilla transformer와 마찬가지로 position embedding을 더해주는데 position embedding을 더해주는 이유는 transformer의 경우 CNN처럼 inductive bias가 존재하지 않기 때문에 이를 해결하기 위함이다. 또한 이미지 분류에 사용되므로 학습 가능한 CLS 토큰을 만들어 더하여 주고 이후 transformer 인코더를 거쳐 인코딩된 CLS 값을 이용하여 이미지 분류 작업을 진행하게 된다.

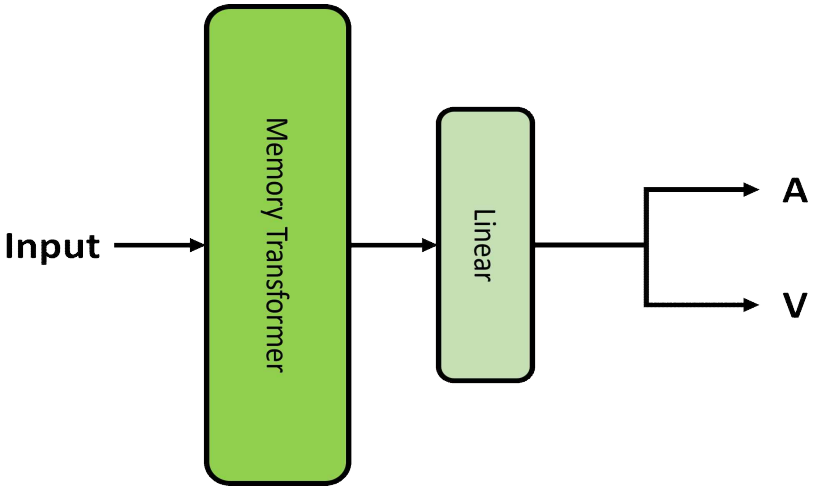
3.3.2 Memory Transformer Q-Learning



(a) Gate Module.



(b) Memory Transformer.



(c) Memory Transformer Q-Network.

Fig. 10 The architecture of memory transformer.

본 논문에서는 DQN의 신경망 구조를 Transformer와 결합하여 성능을 개선하였다. 강화학습에서 신경망의 입력은 타임 스텝마다의 경험 튜플로 구성되므로 sequence가 있는 데이터라 볼 수 있다. 입력 데이터는 이미지이므로, vision transformer가 사용한 방식을 통해 시퀀스가 있는 데이터로 만들어 주었다. 강화학습은 무수한 타임 스텝 동안 시행착오를 겪으며 학습하는 방식이므로 이러한 긴 sequential 데이터를 잘 모델링 할 수 있는 transformer 구조를 사용하여 효과적이고 안정적으로 학습을 진행할 수 있도록 구성하였다. 또한 추가로 transformer의 residual connection 대신 LSTM, GRU와 유사한 gating mechanism을 사용하여 [32-33] 모델이 가지고 있는 memory 벡터를 갱신할 수 있게 구성하였다. Gate는 LSTM과 동일하게 구성[34]하였고, 식 3.16과 같이 표현된다.

$$\begin{aligned} z_t &= \tanh(W_z h_t + b_z) \\ i_t &= \sigma(W_i h_t + b_i - 1) \\ f_t &= \sigma(W_f h_t + b_f + 1) \\ c_{t+1} &= c_t \odot f_t + z_t \odot i_t \end{aligned} \tag{3.16}$$

식 3.11은 총 4개의 gate로 구성되어 있는데, 먼저 i_t는 새로 들어오는 정보들 중 어떤 정보를 memory에 업데이트할 것인가를 정하는 input gate이다. z_t gate는 어떤 정보를 얼만큼 업데이트 할 것인지 tanh 함수를 사용하여 정하고 이 i_t와 z_t gate가 합쳐져 memory를 업데이트할 준비를 한다. f_t는 forget gate로, sigmoid 함수를 사용하여 이전 메모리 벡터에서 삭제할 부분을 정해준다. 마지막으로 c_{t+1}은 나머지 3개의 gate에서 나온 값들을 조합하여 메모리를 적절히 업데이트하는 gate이다.

Memory 벡터는 초기에 0으로 초기화 되고 gate를 통과하며 계속 갱신된다. Gate의 구조는 Fig. 10(a)와 같다. 이러한 gating mechanism을 사용하였을 때 입력받은 경험 튜플뿐만 아니라 모델의 memory(hidden state)를 타임 스텝마다 적절히 갱신하며 모델이 긴 sequence를 더욱 잘 모델링 할 수 있게 도와준다는 것을 실험적으로 보였다. 본 논문에서 제안하는 Memory Transformer의 구조는 Fig. 10(b)와 같고, MTQN의 구조는 Fig. 10(c)와 같다.

4. 실험 및 결과 분석

강화학습 알고리즘은 pytorch와 numpy로 구현하였고, Cart-Pole 시스템은 OpenAI에서 제공하는 gym 라이브러리를 사용하였다. Gym 라이브러리[35]가 제공하는 Cart-Pole 시스템의 observation space는 Table 1과 같다.

Table 1 The observation space of Cart-Pole system.

Observation	Min	Max
Cart Position	$+4.8\ m$	$-4.8\ m$
Cart Velocity	$-\infty\ m/s$	$+\infty\ m/s$
Pole Angle	$-0.418\ rad$	$+0.418\ rad$
Pole Angular Velocity	$-\infty\ rad/s$	$+\infty\ m/s$

공정하고 정확한 결과 분석을 위하여 모델의 구조와 같은 부분은 제외하고 나머지 hyper-parameter는 Table 2와 같이 통일하여 실험을 진행하였다.

Table 2 The fixed hyper-parameters for experiments.

Train	
Optimizer	Adam
Learning rate	0.0005
Update target step	10
τ	0.1
Agent	
γ	0.99
Goal evaluation score	195
policy	
ε_{init}	1.0
ε_{min}	0.3
ε_{decay_step}	20000

Table 2에서 update target step은 online network와 target network의 parameter를 동기화하는 타임 스텝의 수를 의미하고, τ는 동기화를 진행할 때 어느 정도의 비율로 target network를 업데이트할지 정하는 parameter이다. 또한 학습

정책으로는 ε-탐욕적 정책을 사용하였고 ε은 ε_{init}부터 ε_{min}까지 ε_{decay_step}에 걸쳐 식 4.1과 같이 감소하도록 설계하였다.

$$\varepsilon_t = \varepsilon_{min} + (\varepsilon_{init} - \varepsilon_{min}) \times e^{-\frac{t}{\varepsilon_{decay\_step}}}, \quad (t = current\ \ time\ \ step) \tag{4.1}$$

또한 알고리즘의 일반화 성능을 확인하기 위하여 5개의 random seed를 사용하여 [36-38] 매 알고리즘을 5번 학습시켜 evaluation score를 비교 분석하였다. 추가로 Cart-Pole 시스템은 cart를 지정된 범위를 벗어나지 않고 pole이 쓰러지지 않게 하는 것이 목표이므로 cart position과 pole angle을 그래프로 나타내어 이러한 상태가 잘 유지되는지를 확인하였다.

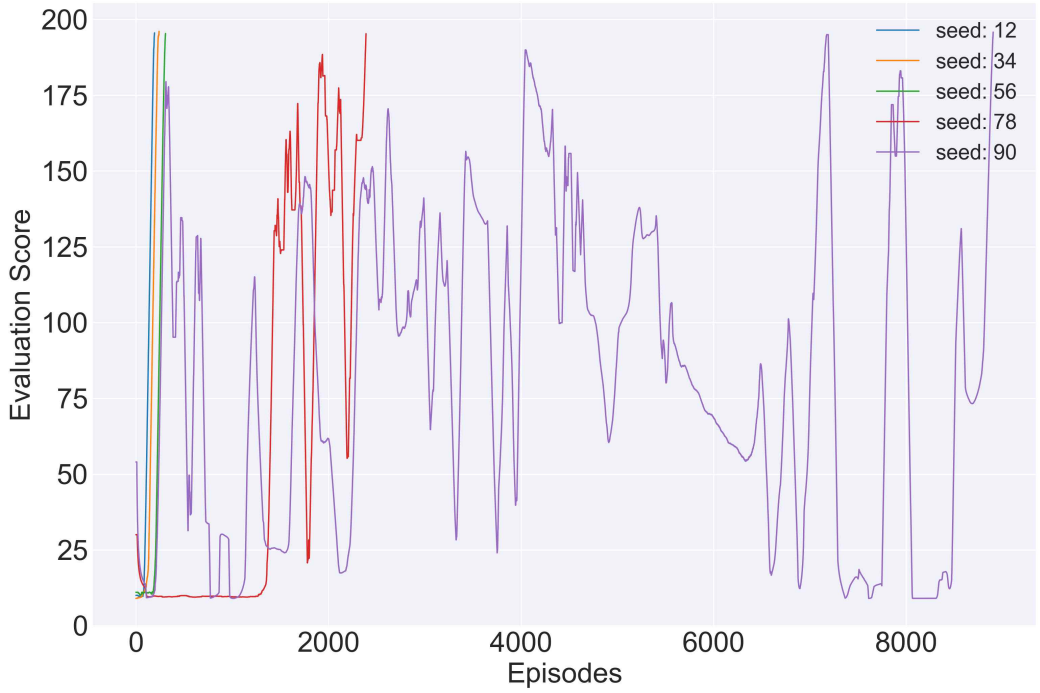
4.1 Evaluation Score

학습은 매 seed 별로 evaluation score가 Cart-Pole 시스템의 최대 score인 200 중 195를 달성하면 종료되도록 설정하였다. 결과 그래프에는 evaluation score가 목표 score를 달성할 때까지의 에피소드 score를 나타내었다.

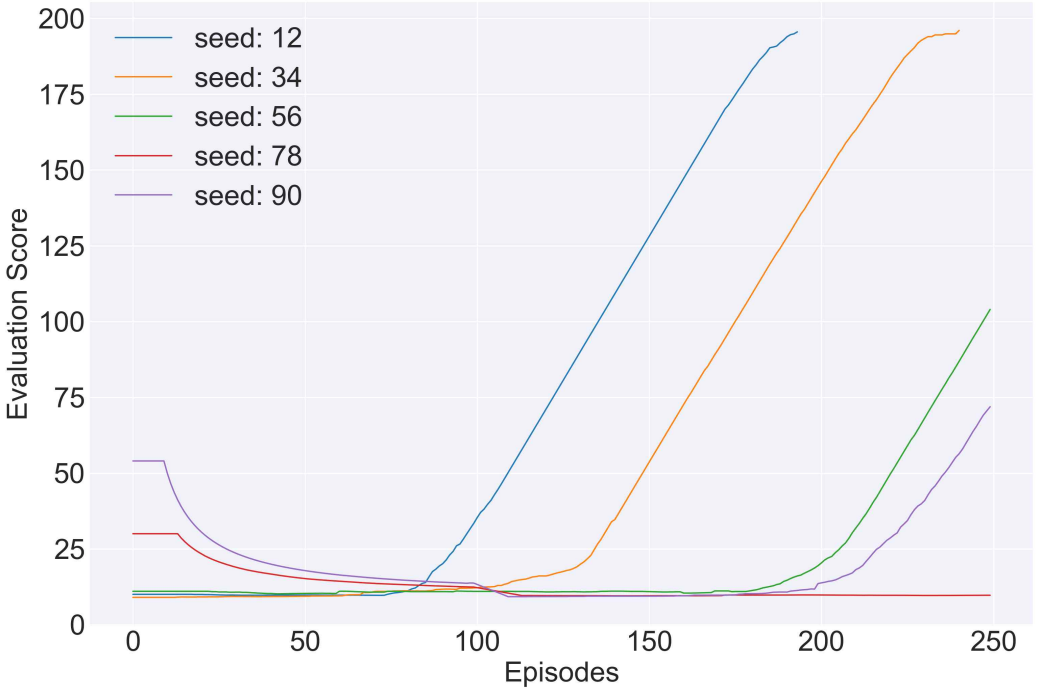
실험 결과 MTQN을 사용하였을 때 평균적으로 적은 에피소드에서 학습이 종료되었고, 학습이 진행됨에 따라 evaluation score가 지속적으로 상승하는 경향을 볼 수 있었다. 이는 본 논문에서 제안하는 MTQN 구조가 다른 알고리즘에 비해 빠르고 안정적으로 학습이 가능하다는 사실을 말한다.

Fig. 11(a)는 DQN을 사용하였을 때 evaluation score가 목표 score를 달성할 때까지의 에피소드 score를 나타내었으며, Fig. 11(b)에는 다른 알고리즘과의 비교가 쉽도록 250 에피소드까지의 결과를 나타내었다. Fig. 11(a)를 살펴보면 seed를 12, 34, 56으로 설정하여 학습한 결과 비교적 빠른 에피소드에 목표 evaluation score를 달성하였지만, seed를 78로 설정한 경우에는 2000 에피소드 이후, 90으로 설정한 경우는 8000 에피소드가 넘어서야 목표 evaluation score에 도달하는 모습을 확인할 수 있다. Seed 값에 따라 학습 시간에 확연한 차이를 보이고, evaluation score 곡선이 매우 요동치는 것으로 미루어 보아 DQN은 학습 안정성이 떨어지고 일반화가

잘되지 않는다고 볼 수 있다. Fig. 12는 DDQN을 사용했을 때 evaluation score가 목표 score를 달성할 때까지의 에피소드 score를 나타내었다. Fig 12를 살펴보면 DDQN의 경우 DQN에 비해 비교적 빠른 에피소드 안에 evaluation score를 달성하지만, seed 별로 목표 evaluation score를 달성하는 에피소드가 차이가 나는 것으로 보아 여전히 학습 안정성이 떨어지는 것을 확인할 수 있다. Fig. 13은 Dueling DDQN을 사용했을 때 evaluation score가 목표 score를 달성할 때까지의 에피소드 score를 나타내었다. Fig. 13을 살펴보면 Dueling DDQN은 DQN, DDQN에 비하여 학습이 빠르게 종료된다. 하지만 seed를 90으로 설정한 경우 evaluation score 그래프가 크게 하락했다가 다시 상승하는 모습을 보아 여전히 학습이 불안정함을 볼 수 있다. Fig. 14는 MTQN을 사용했을 때 evaluation score가 목표 score를 달성할 때까지의 에피소드 score를 나타내었다. Fig. 14를 보면 MTQN의 경우 다른 알고리즘에 비하여 확연히 빠르게 목표 evaluation score를 달성하고, 그래프가 지속적으로 상승하는 형태를 보이므로 안정적으로 학습이 됨을 확인할 수 있다. 물론 seed 34를 사용한 경우 evaluation score 그래프가 학습이 불안정한 모습을 확인할 수 있긴 하지만, 다른 알고리즘을 사용할 때보다는 미세한 수준이다. 이는 나머지 알고리즘에 비하여 본 논문에서 제안하는 MTQN이 빠르고 안정적으로 학습시킬 수 있는 모델임을 말한다.



(a) Evaluation score of DQN per episode.



(b) Evaluation score of DQN up to 250 episode.

Fig. 11 Evaluation score of DQN.

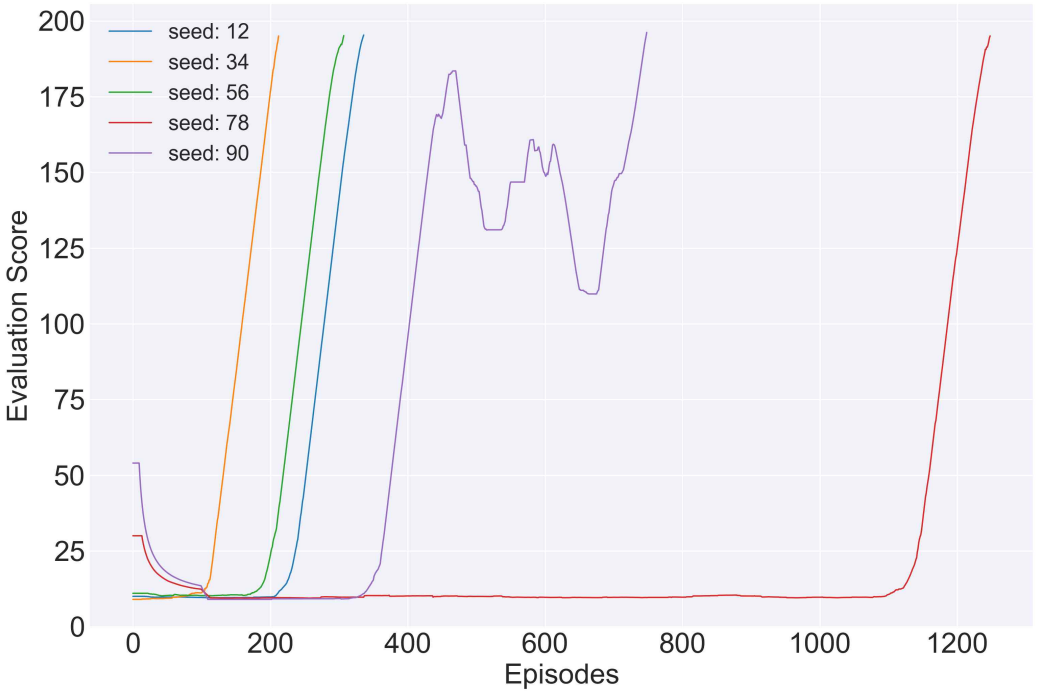


Fig. 12 Evaluation score of DDQN per episode.

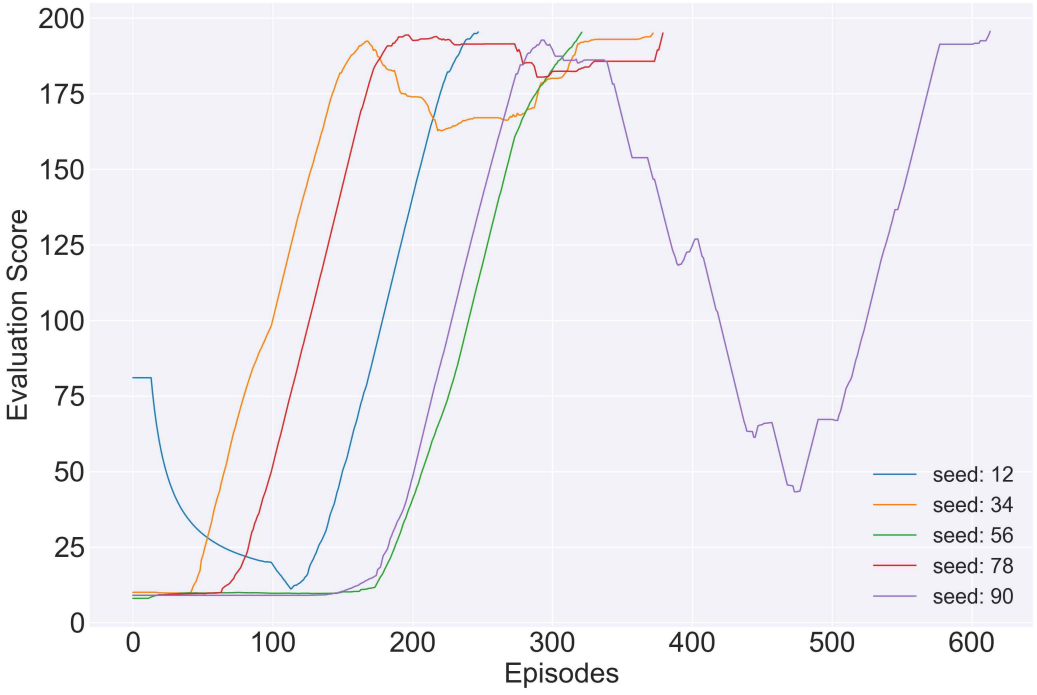


Fig. 13 Evaluation score of Dueling DDQN per episode.

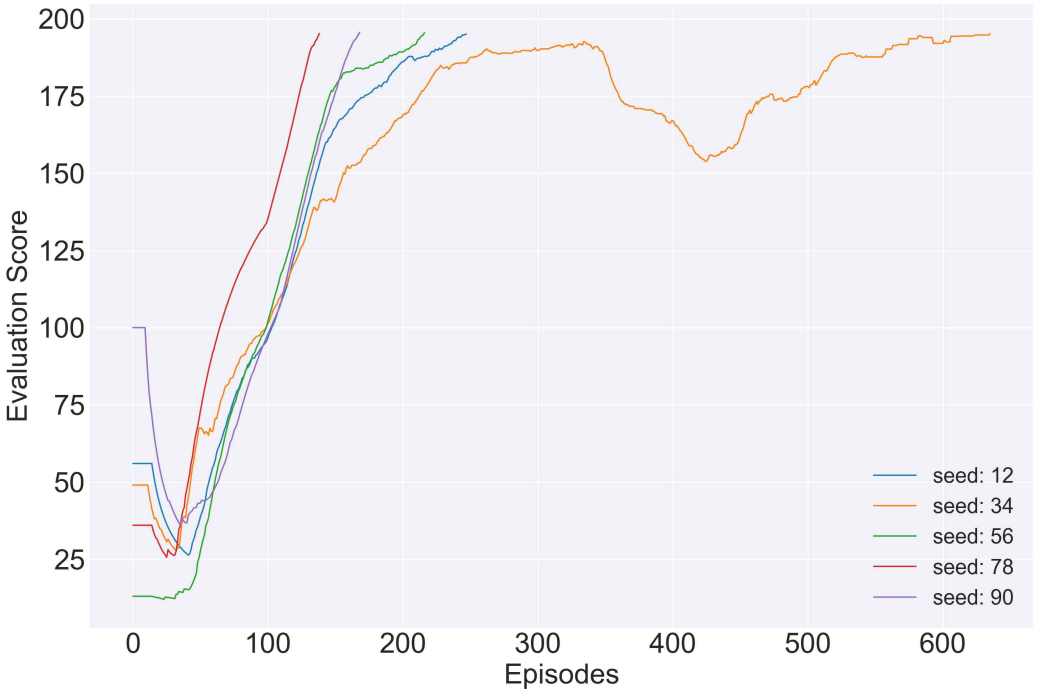


Fig. 14 Evaluation score of MTQN per episode.

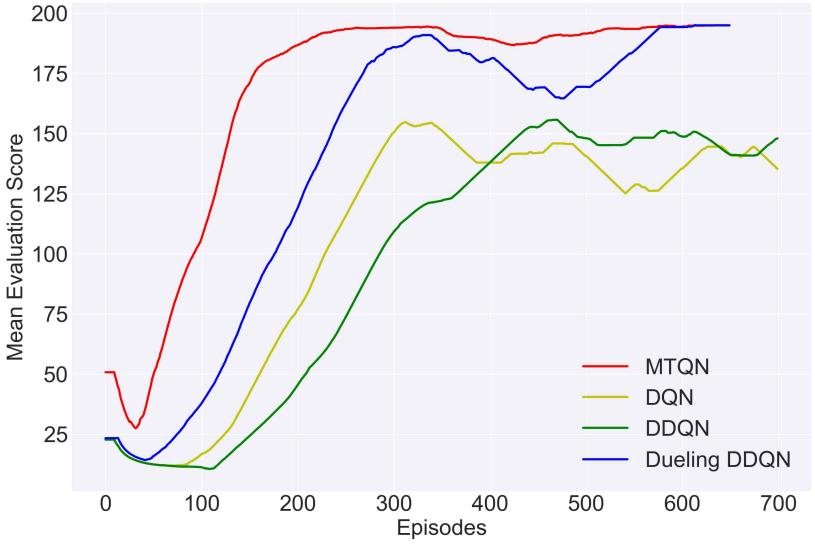


Fig. 15 Comparison of mean evaluation score.

Fig. 15는 모든 seed의 평균 evaluation score값을 비교한 그래프이다. Fig. 15에서도 마찬가지로 볼 수 있듯이 다른 알고리즘을 사용했을 때보다 MTQN의 평균 evaluation score가 지속적으로 상승하고 빠르게 목표값에 도달하는 것을 확인할 수 있다.

4.2 Cart Position

Cart-Pole 시스템은 cart를 좌우로 움직이며 pole의 균형을 잡는 환경이므로 cart가 진동폭이 크게 움직이지 않게 제어할수록 이상적인 알고리즘이라 볼 수 있다. 따라서 타임 스텝에 따른 cart position과 타임 스텝에 따른 cart의 누적 이동 거리를 그래프로 나타내어 관찰하였다. Cart position과 누적 이동 거리는 학습 시 가장 빠르게 최대 목표 evaluation score를 달성한 모델을 에이전트로 사용하여 추출하였다.

Fig. 16은 DQN을 사용했을 때 타임 스텝에 따른 cart position을 나타내었다. Fig. 16을 살펴보면 DQN으로 학습한 결과 cart position이 비교적 오른쪽 치우친 경향을 볼 수 있다. Fig. 17은 DDQN을 사용했을 때 타임 스텝에 따른 cart position을 나타내었다. Fig. 17을 살펴보면 DDQN의 경우 DQN에 비하여 진동폭이 작게 움직이지만 약간 왼쪽으로 치우친 경향을 확인할 수 있다. Fig. 18은 Dueling DDQN을 사용했을 때 타임 스텝에 따른 cart position을 나타내었다. Fig. 18을 보면 Dueling DDQN의 경우 오히려 DQN보다도 cart를 우측으로 치우치게 움직이는 모습을 확인할 수 있다. Fig. 19는 MTQN을 사용했을 때 타임 스텝에 따른 cart position을 나타내었다. Fig. 19를 보면 cart position이 나머지 알고리즘에 비하여 원점을 기준으로 균형적으로 수렴하며 움직임을 확인할 수 있다.

실험 결과 DDQN과 MTQN 구조를 사용했을 때 cart position이 다른 알고리즘에 비하여 비교적 진동폭이 크지 않게 움직이는 모습을 확인할 수 있었다.

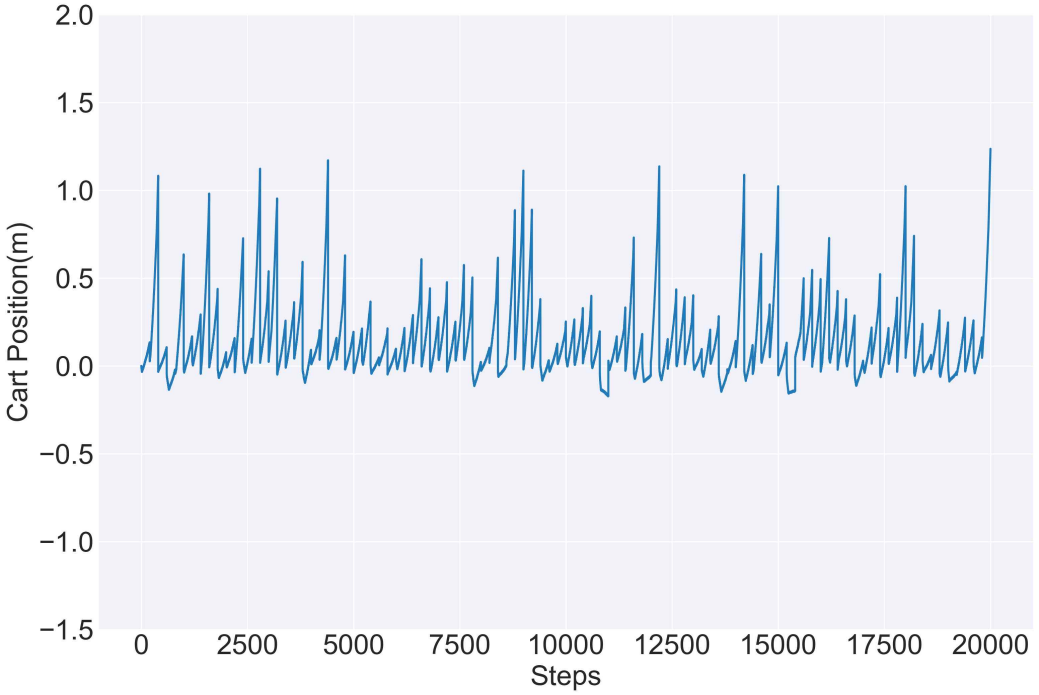


Fig. 16 Cart position per time step using DQN.

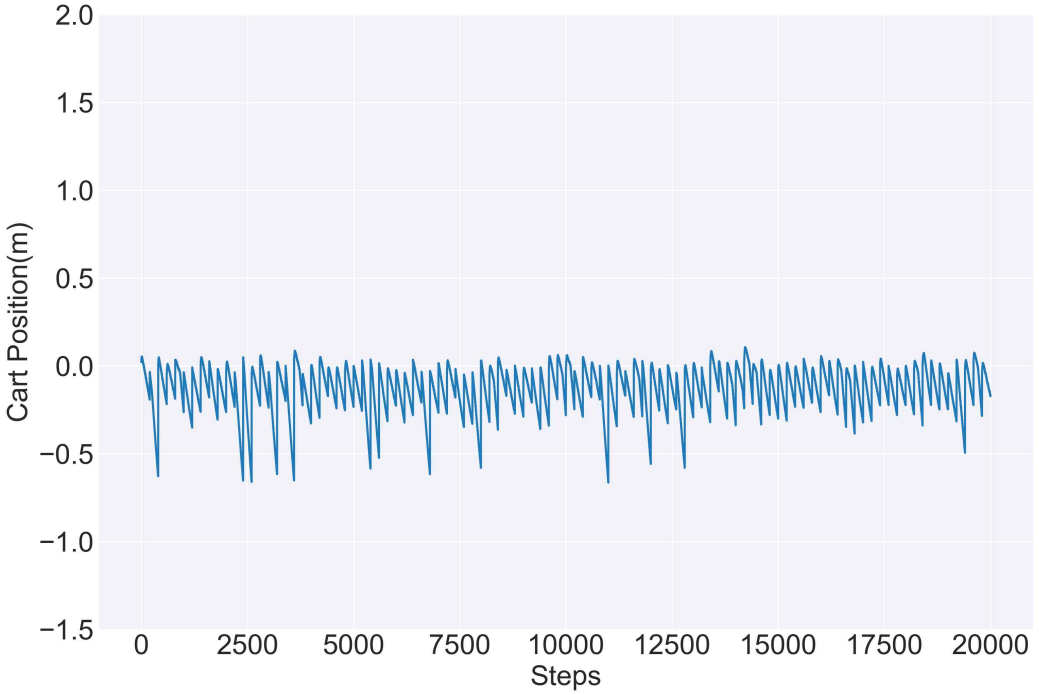


Fig. 17 Cart position per time step using DDQN.

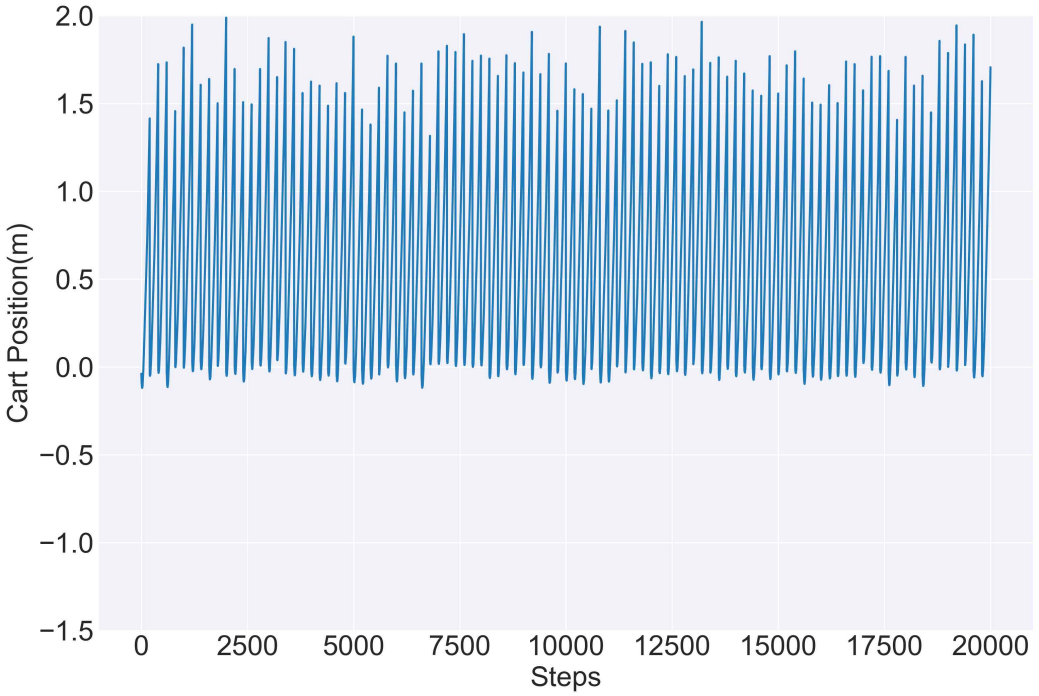


Fig. 18 Cart position per time step using Dueling DDQN.

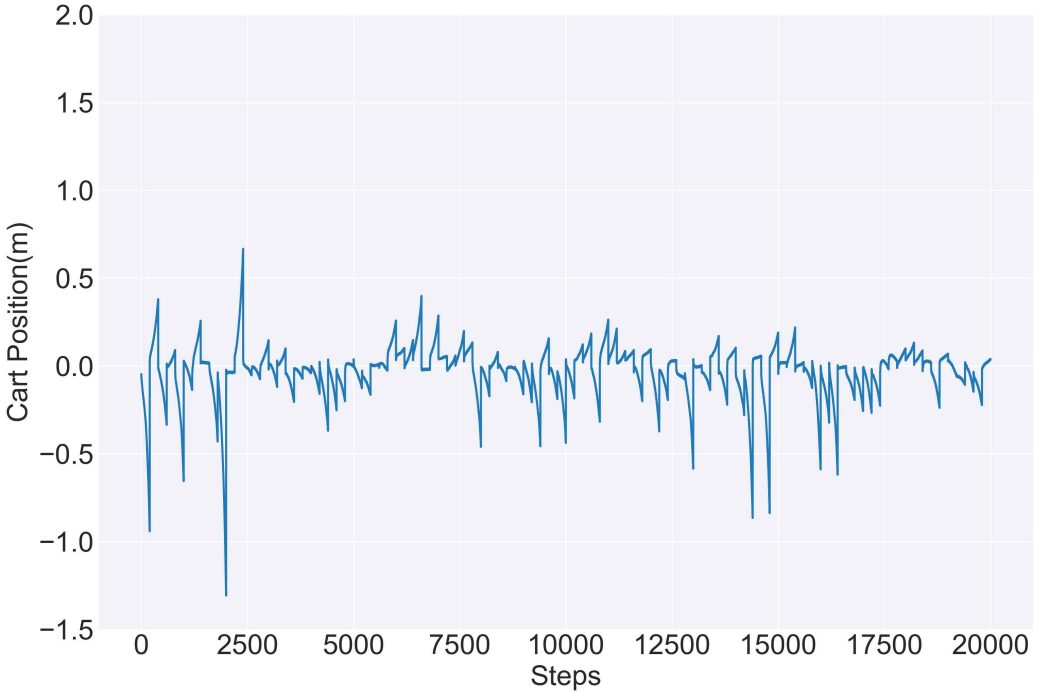


Fig. 19 Cart position per time step using MTQN.

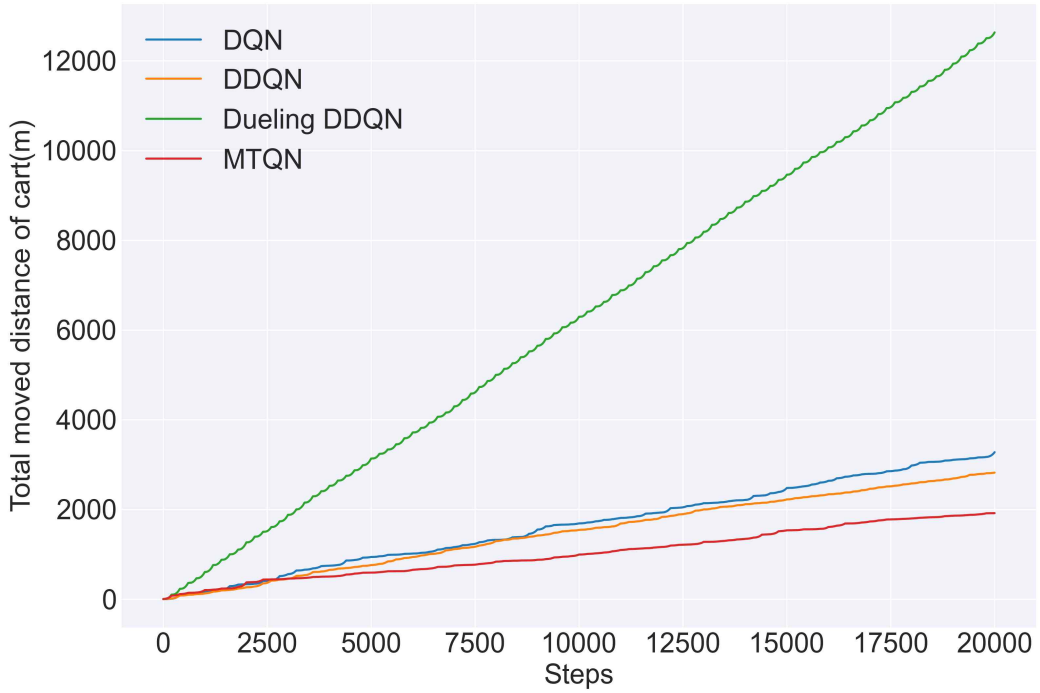


Fig. 20 Total cumulative distance of Cart.

Fig. 20은 알고리즘별 cart의 총 누적 이동 거리를 나타내었다. Fig. 20을 살펴보면 MTQN을 사용했을 때 누적거리가 $2000\,m$ 이하로 가장 적게 움직였고, DDQN을 사용했을 때 약 $2700\,m$로 두 번째, DQN을 사용했을 때 총 $3000\,m$ 이상으로 세 번째, Dueling DDQN을 사용했을 때 누적거리가 총 $12000\,m$ 이상으로 가장 많이 움직였음을 확인할 수 있었다. 따라서 본 논문에서 제안하는 MTQN을 사용했을 때 cart의 총 누적 이동 거리가 다른 알고리즘에 비하여 짧고 진동폭이 작게 움직이는 것으로 보아 MTQN이 다른 알고리즘에 비해 효율적으로 cart position을 제어한다고 말할 수 있다.

4.3 Pole Angle

Cart-Pole 시스템은 pole의 균형을 잡는 것이 목표인 환경이므로 타임 스텝에 따른 pole angle과 pole angle의 분포를 나타낸 histogram을 관찰하였다. Pole angle은 학습 시 가장 빠르게 최대 목표 evaluation score를 달성한 모델을 에이전트로 사용하여 추출하였다. 또한 추가로 학습 과정 중 pole angle의 변화를 관찰하여 원점으로의 수렴 여부를 확인하였다.

4.3.1 학습 과정 중 pole angle

Fig. 21-24는 학습 과정 중의 pole angle을 추출하여 나타낸 그림이다. Fig. 21과 Fig. 22를 살펴보면 DQN과 DDQN 학습 시 pole angle이 15000 타임 스텝 이후 수렴하여 매우 느리게 수렴하는 모습을 확인할 수 있는 반면, Fig. 23과 Fig. 24를 살펴봤을 때, Dueling DDQN과 MTQN 학습 시 pole angle이 비교적 빠르게 원점 근처로 수렴하는 모습을 확인할 수 있다. 특히 MTQN 학습 시 더욱 빠르고 원점에 가깝게 수렴하는 모습을 확인할 수 있다.

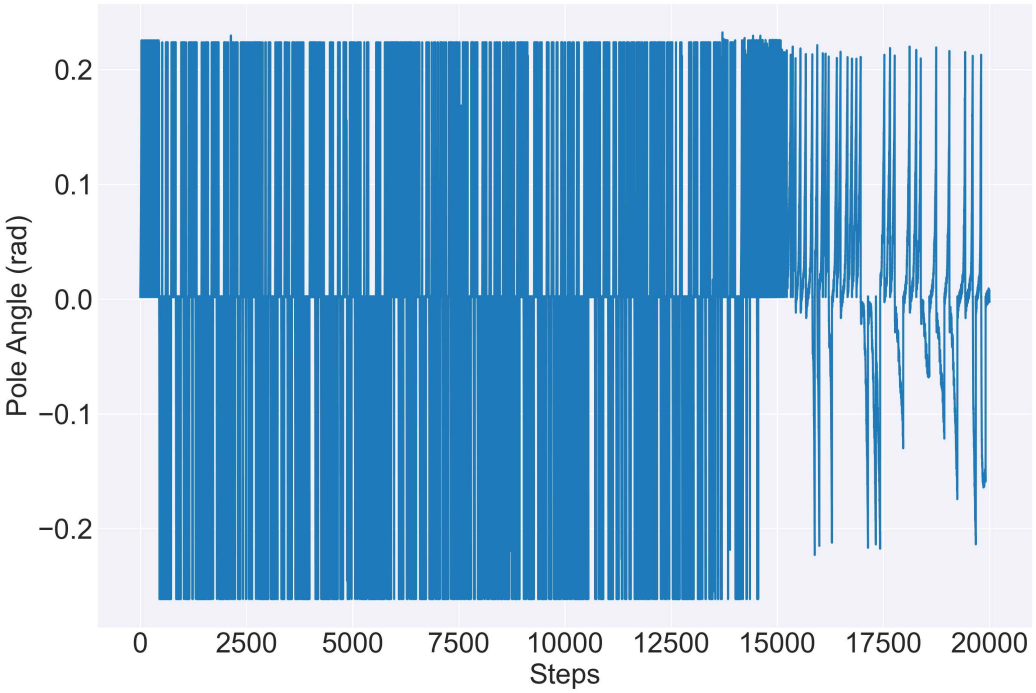


Fig. 21 Pole angle per time step on training DQN.

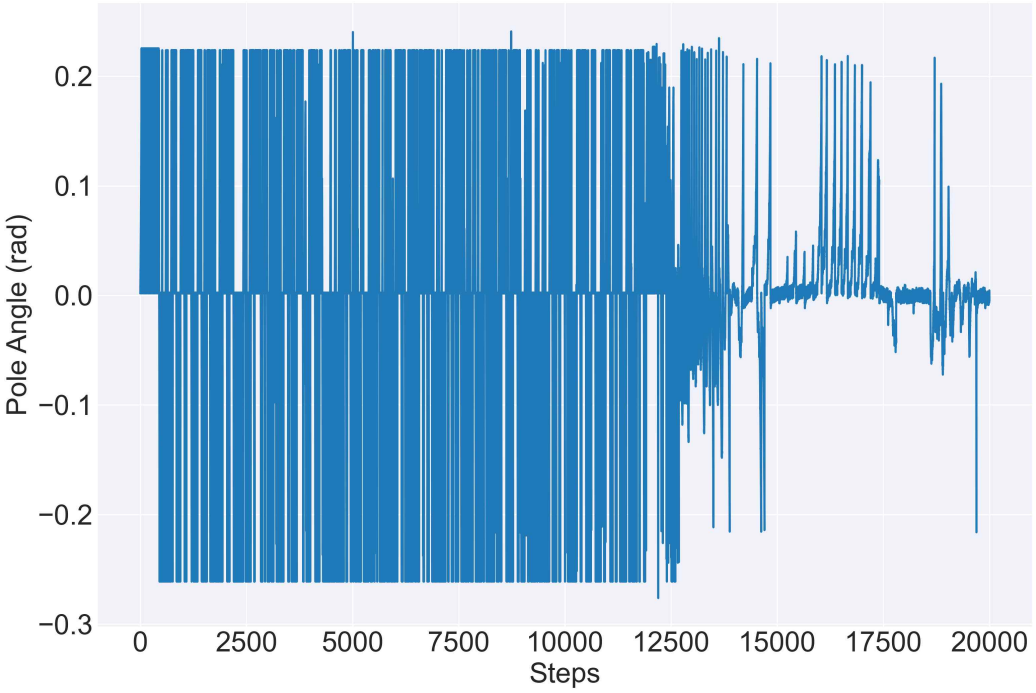


Fig. 22 Pole angle per time step on training DDQN.

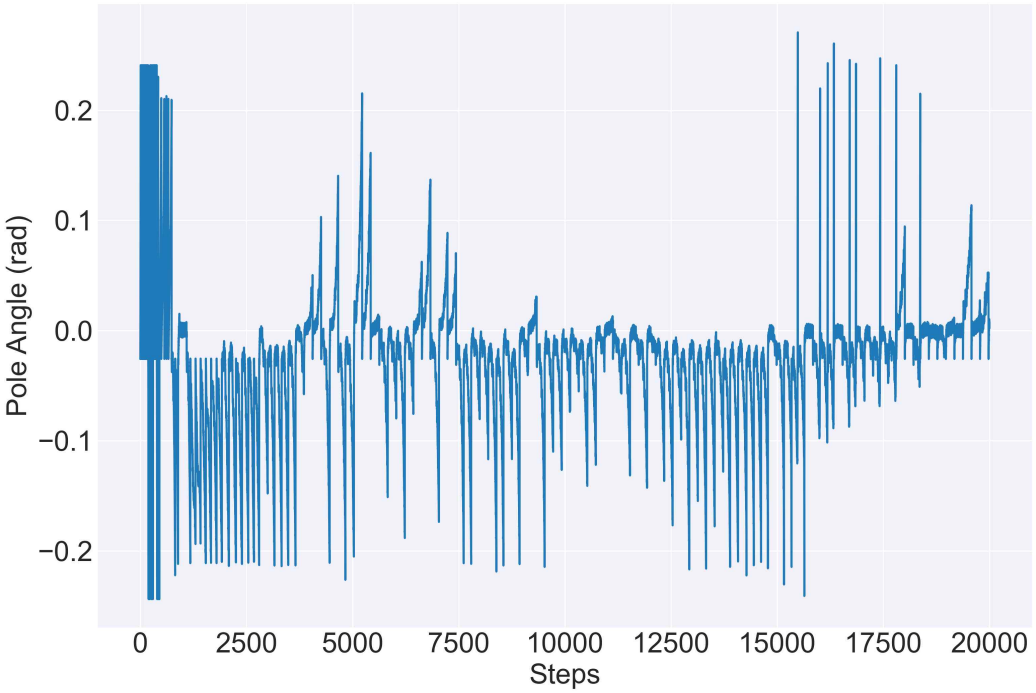


Fig. 23 Pole angle per time step on training Dueling DDQN.

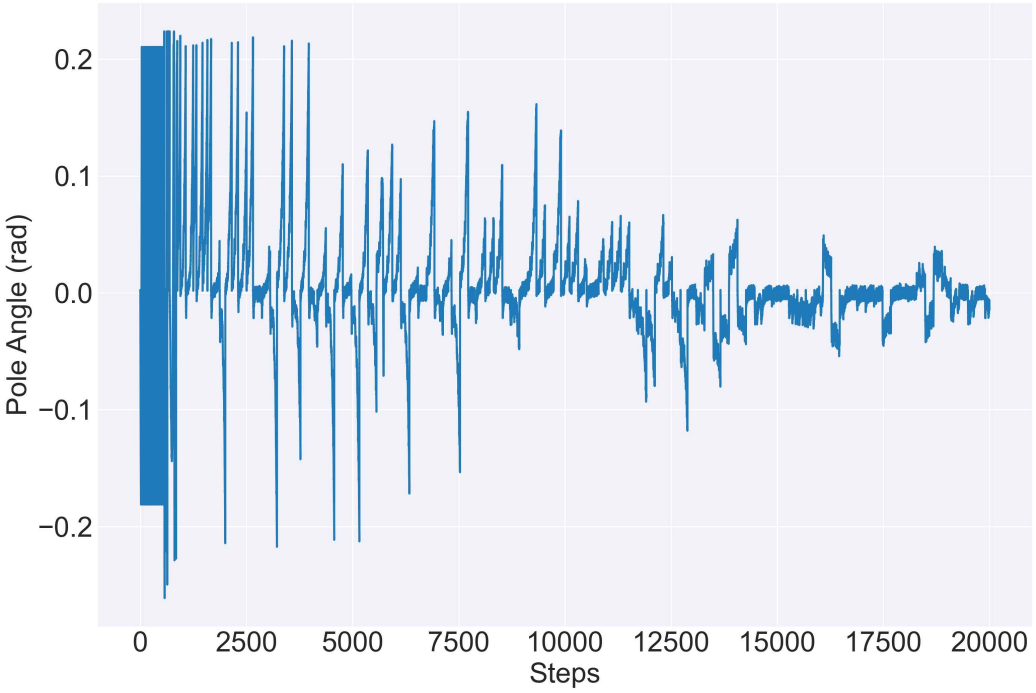
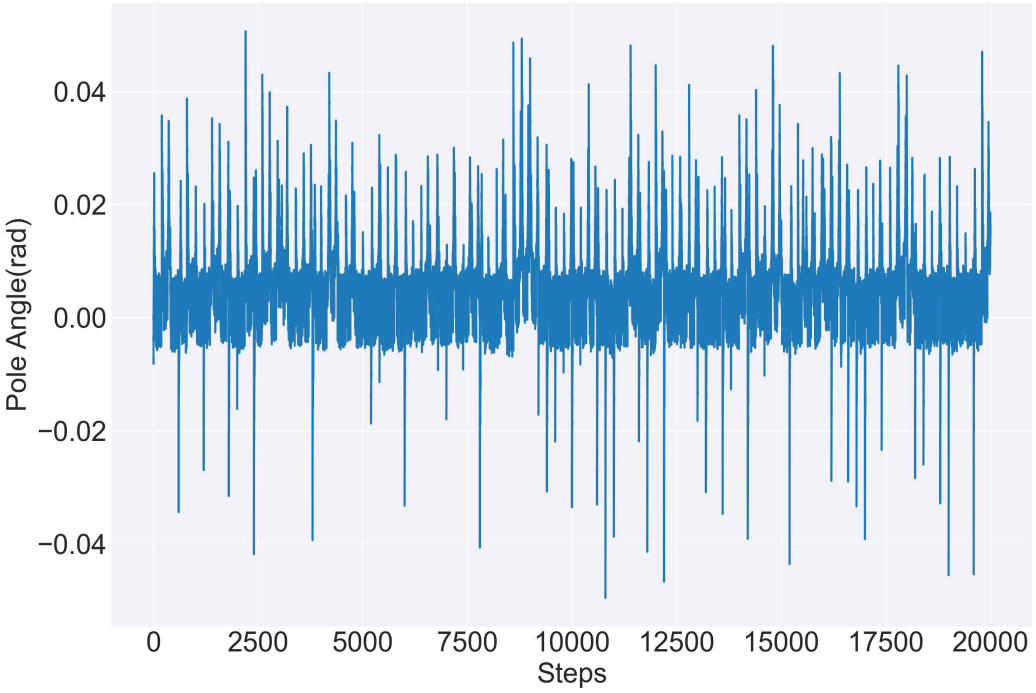


Fig. 24 Pole angle per time step on training MTQN.

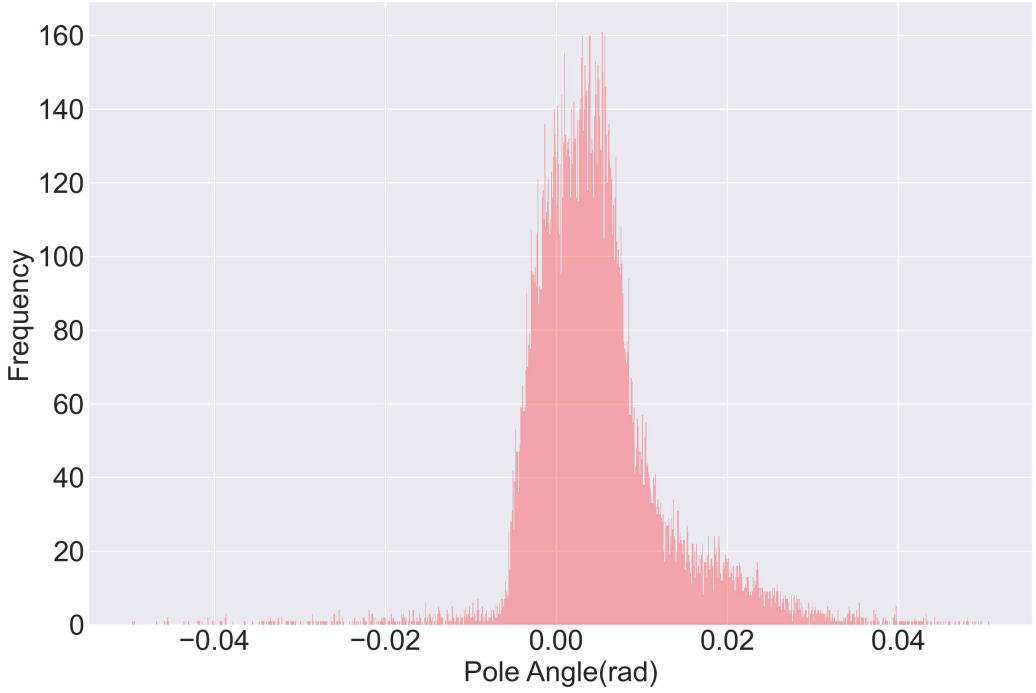
4.3.2 학습 완료 후 pole angle

실험 결과 Dueling DDQN을 제외한 알고리즘 모두 pole angle을 0 rad근처에 대칭적으로 유지하는 모습을 보여주며 특히 MTQN을 사용한 경우 pole angle이 정확히 0rad에 가장 많이 분포하는 모습을 확인할 수 있었다. 이는 MTQN 알고리즘이 가장 이상적으로 pole angle을 제어한다고 할 수 있다.

Fig. 17(a)를 보면 DQN을 사용하여 학습한 모델의 경우 pole angle이 대체적으로 0 rad 근처로 수렴하기는 하나, Fig. 17(b)를 봤을 때 0 rad보다 근소하게 우측 분포가 쏠린 경향을 확인할 수 있다. Fig. 18(a)를 살펴보면 DDQN 알고리즘을 사용한 경우 DQN과 마찬가지로 pole angle이 0 rad 근처로 잘 수렴하는 모습을 확인할 수 있다. Fig. 18(b)를 통해 분포를 살펴보면 pole angle이 DQN보다 훨씬 대칭적으로 분포하는 모습을 확인할 수 있다. Fig. 19(a)를 보면 Dueling DDQN의 경우 0rad보다 큰 각으로 수렴하고, 다른 알고리즘에 비해 수렴 범위가 넓음을 확인할 수 있다. 마찬가지로 Fig. 19(b)를 확인해보면 0rad을 기준으로 좌측은 거의 분포하지 않고 우측으로 넓게 분포함을 확인할 수 있다. 이는 이상적인 pole angle의 분포와는 거리가 멀다. Fig. 20(a)을 살펴보면 pole angle이 0rad으로 수렴하는 모습을 볼 수 있고, Fig. 20(b)을 봤을 때 pole angle이 0rad일 때가 가장 많았고 분포 역시 대칭적임을 확인할 수 있다.

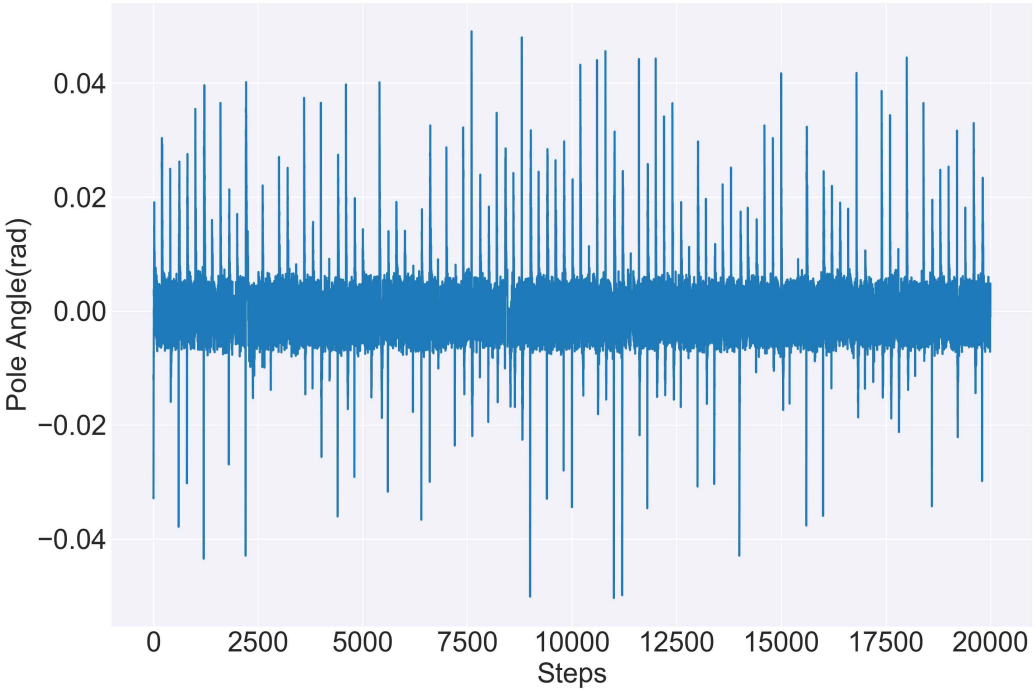


(a) Pole angle per time step using DQN.

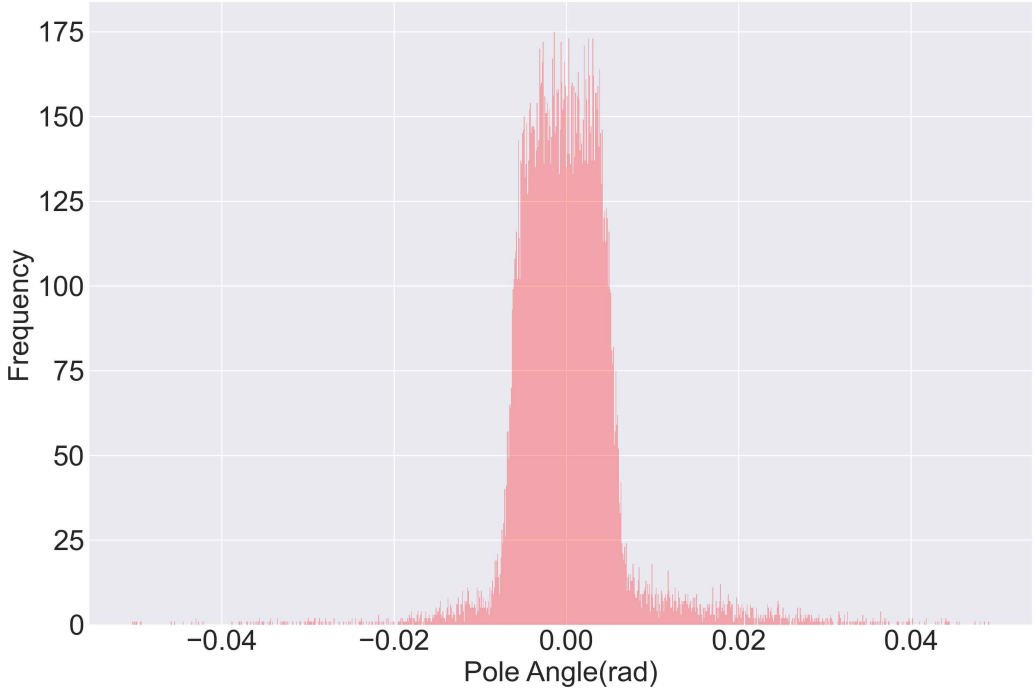


(b) Pole angle distribution using DQN.

Fig. 25 Pole angle per time step using DQN.

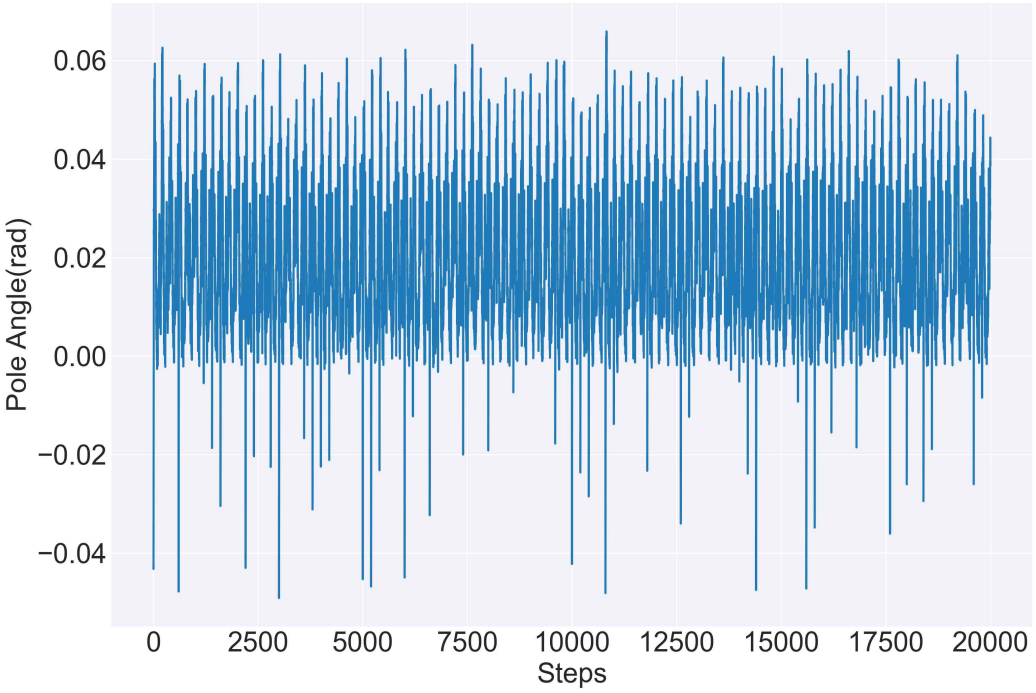


(a) Pole angle per time step using DDQN.

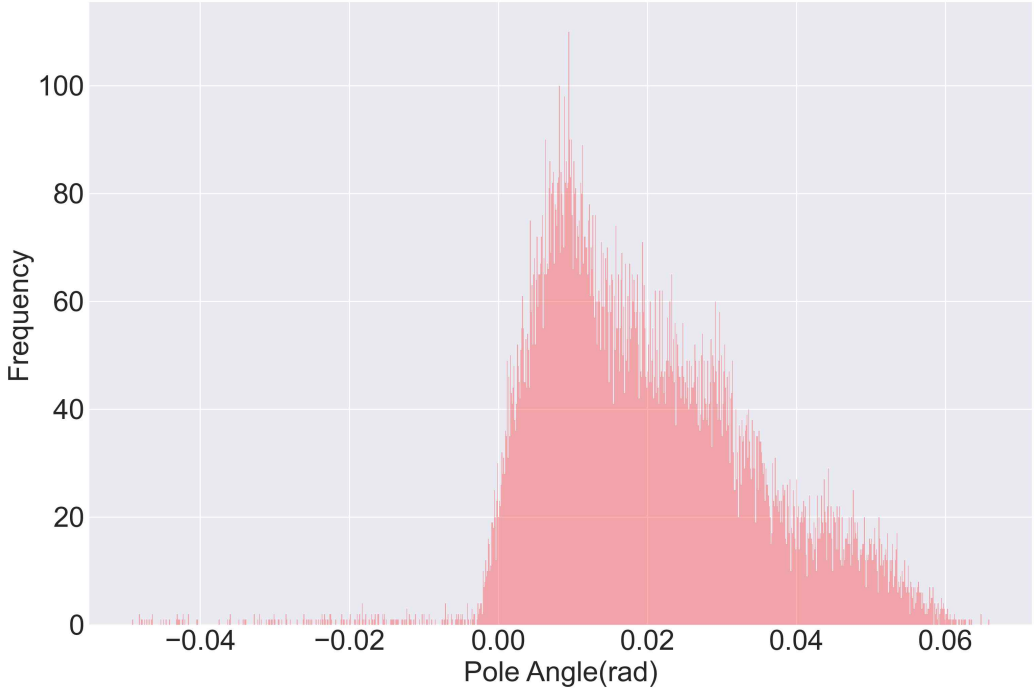


(b) Pole angle distribution using DDQN.

Fig. 26 Pole angle per time step using DDQN.

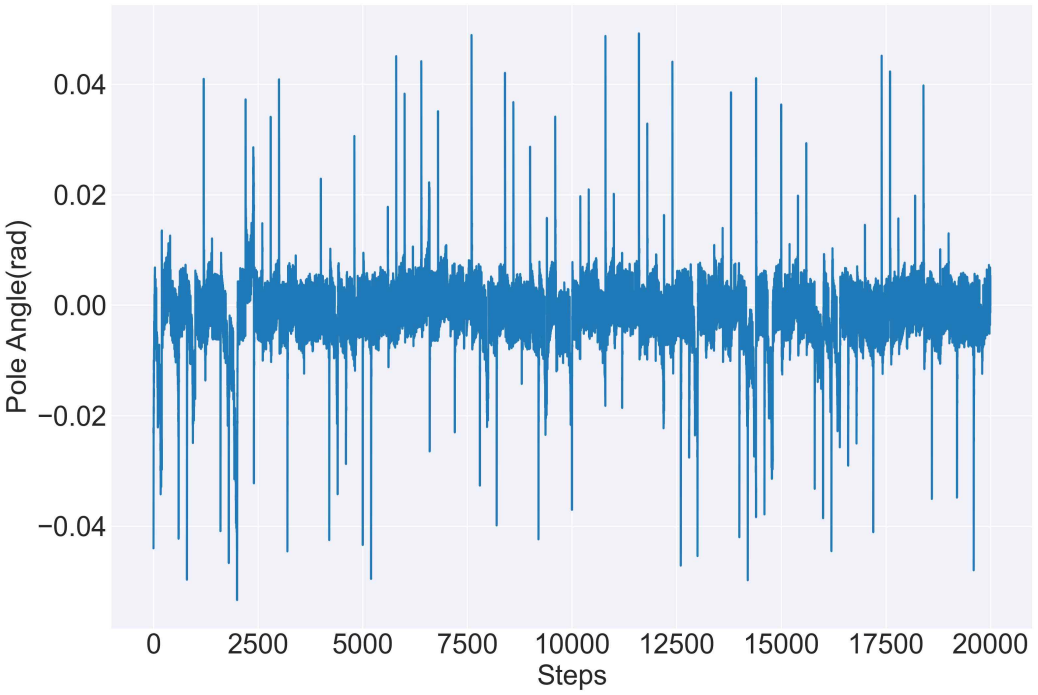


(a) Pole angle per time step using Dueling DDQN.

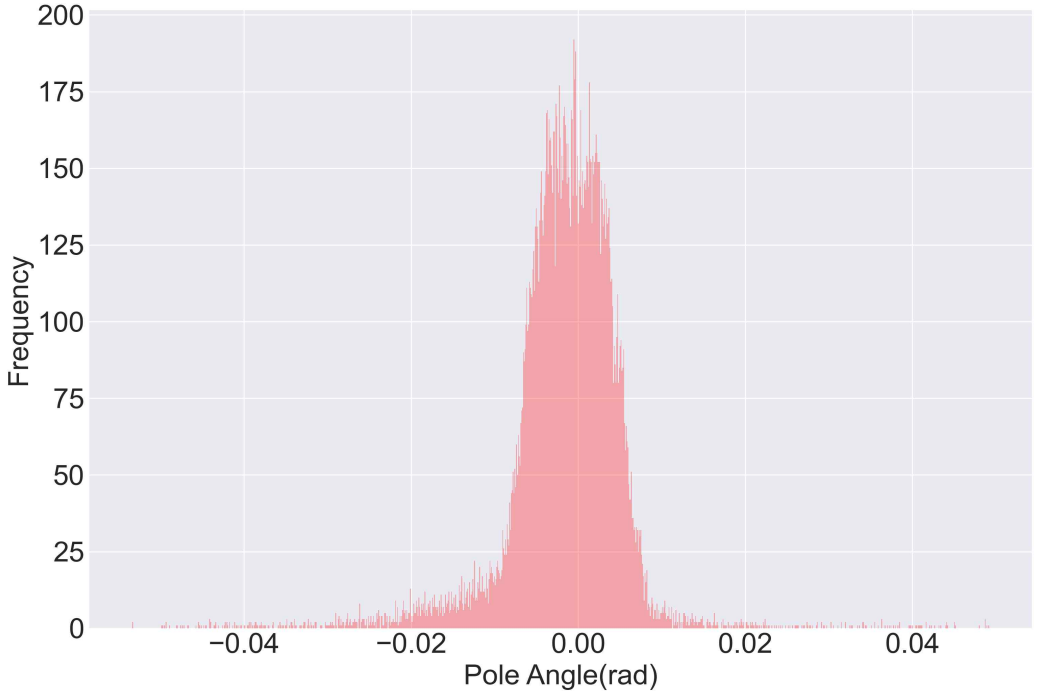


(b) Pole angle distribution using Dueling DDQN.

Fig. 27 Pole angle per time step using Dueling DDQN.



(a) Pole angle per time step using MTQN.



(b) Pole angle distribution using MTQN.

Fig. 28 Pole angle per time step using MTQN.

5. 결론

본 논문에서는 transformer를 심층 강화학습 모델에 결합하여 긴 sequence 시스템을 효율적으로 제어할 수 있도록 개선하였다. 또한 transformer를 심층 강화학습 모델에 효율적으로 결합하기 위하여 LSTM의 gating mechanism을 이용하였다.

제안한 알고리즘은 성능분석을 위하여 대표적인 강화학습 benchmark 환경인 Cart-Pole 시스템에서 대표적인 심층 강화학습 알고리즘인 DQN, DQN의 변형 알고리즘들과 비교 분석하였다.

실험은 Cart-Pole 시스템의 evaluation score, cart position 그리고 pole angle을 비교 분석하였으며, 모든 항목에서 제안한 알고리즘이 기존 심층 강화학습 알고리즘들에 비해 우수함을 보였다.

Evaluation score는 매 seed 별로 Cart-Pole 시스템의 최대 score인 200중 195를 달성하면 종료되도록 설정하였다. 제안한 알고리즘이 다른 심층 강화학습 알고리즘과 비교하여 평균적으로 적은 에피소드에서 학습이 종료되었고, 학습이 진행됨에 따라 evaluation score가 지속적으로 상승하는 경향을 보여주었다. Cart position도 제안한 알고리즘에서 다른 알고리즘에 비하여 진동폭이 크지 않게 움직였고, cart의 총 누적 이동 거리가 다른 알고리즘에 비하여 짧게 나타났다. 따라서 제안된 알고리즘은 다른 알고리즘에 비해 효율적으로 cart position을 제어한다고 말할 수 있다. pole angle의 분석은 학습 중과 학습 이후에 타임 스텝에 따른 pole angle과 pole angle의 분포를 나타낸 히스토그램을 사용하였다. 학습 전 pole angle을 추출해 봤을 때 DQN과 DDQN의 경우 pole angle이 일정하게 수렴하지 않는 모습을 확인할 수 있는 반면, Dueling DDQN과 제안한 알고리즘은 학습 시 pole angle이 빠르게 원점 근처로 수렴하는 모습을 확인할 수 있었다. 특히 제안한 알고리즘이 학습 시 더욱 빠르고 일정하게 수렴하는 모습을 확인할 수 있었다. 학습이 완료된 후 pole angle을 추출해봤을 때 대체적으로 모든 알고리즘에서 $0\,rad$ 근처에 대칭적으로 유지하는 모습을 보여주었지만, 제안한 알고리즘을 사용했을 때 pole angle이 $0\,rad$에 가장 많이 분포하는 모습을 확인할 수 있었다. 이러한 사실로 봤을 때, 제안한 알고리즘을 사용하면 Cart-Pole 시스템의 데이터를 잘 모델링할 수 있고, 효율적으로 제어

할 수 있음을 알 수 있었다.

참 고 문 헌

[1] Kober, Jens, J. Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey." The International Journal of Robotics Research 32.11 (2013): 1238-1274.

[2] Shao, Kun, et al. "A survey of deep reinforcement learning in video games." arXiv preprint arXiv:1912.10944 (2019).

[3] Hambly, Ben, Renyuan Xu, and Huining Yang. "Recent advances in reinforcement learning in finance." Mathematical Finance 33.3 (2023): 437-503.

[4] Ouyang, Long, et al. "Training language models to follow instructions with human feedback." Advances in Neural Information Processing Systems 35 (2022): 27730-27744.

[5] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[6] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).

[7] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[8] Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning." Proceedings of the AAAI conference on artificial intelligence. Vol. 30. No. 1. 2016.

[9] Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." International conference on machine learning. PMLR, 2016.

[10] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[11] Dosovitskiy, Alexey, et al. "An image is worth 16x16 words:

Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).

[12] Liu, Ze, et al. "Swin transformer: Hierarchical vision transformer using shifted windows." Proceedings of the IEEE/CVF international conference on computer vision. 2021.

[13] Dong, Linhao, Shuang Xu, and Bo Xu. "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition." 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2018.

[14] Chen, Hanting, et al. "Pre-trained image processing transformer." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021.

[15] Verma, Prateek, and Jonathan Berger. "Audio transformers: Transformer architectures for large scale audio understanding. adieu convolutions." arXiv preprint arXiv:2105.00335 (2021).

[16] Gong, Yuan, et al. "Ssast: Self-supervised audio spectrogram transformer." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 36. No. 10. 2022.

[17] OpenAI. (2022). GPT-3.5: Language Models for Natural Language Understanding. https://openai.com

[18] A. Dutech, et al. "Reinforcement learning benchmarks and bake- offs II." Advances in Neural Information Processing Systems (NIPS) 17 (2005).

[19] C. W. Anderson "Learning to control an inverted pendulum using neural networks." IEEE Control Systems Magazine 9.3 (1989): 31-37.

[20] Barto, Andrew G., Richard S. Sutton, and Charles W. Anderson. "Neuronlike adaptive elements that can solve difficult learning control problems." IEEE transactions on systems, man, and cybernetics 5 (1983): 834-846.

[21] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An

introduction. MIT press, 2018.
[22] Sutton, Richard S., et al. "Policy gradient methods for reinforcement learning with function approximation." Advances in neural information processing systems 12 (1999).

[23] Konda, Vijay, and John Tsitsiklis. "Actor-critic algorithms." Advances in neural information processing systems 12 (1999).
[24] Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." International conference on machine learning. PMLR, 2016.
[25] Dann, Chris, et al. "Guarantees for epsilon-greedy reinforcement learning with function approximation." International conference on machine learning. PMLR, 2022.
[26] Zhang, Hao, et al. "Dino: Detr with improved denoising anchor boxes for end-to-end object detection." arXiv preprint arXiv:2203.03605 (2022).
[27] Petit, Olivier, et al. "U-net transformer: Self and cross attention for medical image segmentation." Machine Learning in Medical Imaging: 12th International Workshop, MLMI 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, September 27, 2021, Proceedings 12. Springer International Publishing, 2021.
[28] Touvron, Hugo, et al. "Llama: Open and efficient foundation language models." arXiv preprint arXiv:2302.13971 (2023).
[29] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
[30] Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).
[31] Schmidt, Robin M. "Recurrent neural networks (rnns): A gentle introduction and overview." arXiv preprint arXiv:1912.05911 (2019).
[32] Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural

networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).
[33] Dai, Zihang, et al. "Transformer-xl: Attentive language models beyond a fixed-length context." arXiv preprint arXiv:1901.02860 (2019).
[34] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.
[35] Brockman, Greg, et al. "Openai gym." arXiv preprint arXiv:1606.01540 (2016).
[36] Haarnoja, Tuomas, et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." International conference on machine learning. PMLR, 2018.
[37] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).
[38] Andrychowicz, Marcin, et al. "Hindsight experience replay." Advances in neural information processing systems 30 (2017).

Cart-Pole System Control with Memory Transformer Q-Learning

Byeong-Chan Han

Department of Electronic Engineering
The Graduate School
Jeju National University

Abstract

This paper proposes a Memory Transformer Q-learnig Network (MTQN) to improve the existing deep learning algorithm. The MTQN is constructed by combining the existing deep learning model with a transformer to model the sequence system more efficiently, and the gating mechanism of LSTM is used.

The proposed algorithm is compared and analyzed on the Cart-Pole system, which is a representative reinforcement learning benchmark environment. The Cart-Pole system uses the gym library provided by OpenAI, and the reinforcement learning algorithm is implemented with pytorch and numpy.

To analyze the performance of the proposed algorithm, DQN, a representative deep reinforcement learning algorithm, and DQN's variants were compared and analyzed. The experiments were conducted by extracting the evaluation score, cart position, and pole angle of the Cart-Pole system. The evaluation score shows that the proposed algorithm learns the fastest and most stable. Also, by checking the cart position and the cumulative distance by the cart, it can be seen that the cart position in the proposed algorithm moves symmetrically around the origin compared to other

algorithms, and the total cumulative distance is significantly shorter than other algorithms. Pole angle also converged to the origin quickly in the proposed algorithm compared to the other algorithms, and the extracted results from the trained algorithm also confirmed that it was symmetrically distributed around the origin.