DQN의 2048 게임 적용에 대한 한계 분석 및 개선 가능성 탐색

타일 해커 팀

장예원, 최상아, 한사랑

GITHUB : https://github.com/KanghwaSisters/25-2-2048

[초록]

본 연구는 강화학습 기반의 심층 Q-네트워크(Deep Q-Network, DQN)를 적용하여 2048 게임을 효과적으로 학습하는 모델을 구축하는 것을 목표로 한다. 이를 위해 기본 DQN 구조에 더하여 Double DQN, Dueling DQN, Prioritized Experience Replay(PER)와 같은 다양한 성능 개선 기법을 적용하였으며, ε-탐욕 정책의 수렴 속도와 보상 함수 설계(Reward Shaping)가 학습 성능에 미치는 영향을 분석하였다. 학습 과정은 상태 관측, 행동 선택, 환경 반응, 경험 저장, 네트워크 학습, 타깃 네트워크 업데이트의 6단계로 정의하여 강화학습의 전형적인 절차에 따라 진행하였다.

실험 결과, 기본 DQN에 비해 개선 기법을 적용한 모델이 더 빠른 수렴 속도와 높은 평균 보상을 기록하였으며, 특히 ε-탐욕 정책의 적절한 파라미터 설정과 병합 점수·빈 칸 수를 반영한 보상 설계가 안정적인 학습 곡선을 형성하는 데 효과적임을 확인하였다. 최종적으로 제안된 모델은 다른 설정 대비 높은 성능을 나타내었으며, 이는 2048과 같은 퍼즐형 게임 환경에서도 강화학습 기법의 확장 가능성을 보여준다. 본 연구는 DQN 성능 개선 요인을 체계적으로 검증하였다는 점에서 학술적 의의를 가지며, 향후 다양한 게임 및 실제 응용 문제에 강화학습을 적용하는 데 중요한 시사점을 제공한다.

핵심 주제어: 강화학습, DQN, Prioritized Experience Replay(PER), ε-탐욕 정책, 보상 설계, 2048 게임

목차

그림 • 표 목차

Ⅰ. 서론

A. 연구 배경 및 목적

강화 학습 방법 중 특정 상태에 따른 행동의 누적 보상 값, 즉 Q 값(Q-value)으로 학습하는 방법인 Q-learning은 강화 학습 역사의 초기부터 많이 사용되었다. 전통적인 Q-learning 학습은 Q 값을 테이블에 저장하여 사용하기 때문에 작은 크기의 상태공간에서만 사용이 가능하다는 근본적인 단점이 존재했다. 하지만 이미지, 동영상, 센서 데이터와 같은 비정형·고차원 데이터를 딥러닝으로 처리하는 기술이 발전하면서, 21세기 초부터 딥러닝으로 데이터를 처리하는 과정에 강화 학습을 활용하는 방식이 급확산되었다. 이에 따라 새로운 Q-learning 방식의 필요성이 대두되었고, 고차원 데이터를 처리하기 위해서 Q 값을 근사하고 경험을 저장하는 인공 신경망이 필수로 자리 잡았다. 본 연구에서는 게임 '2048'을 강화 학습을 통해 풀어보고 4x4의 상태 공간 안에서 DQN 알고리즘을 적용한 모델이 어떤 성능을 보이는지 실험해 보고자 한다.

B. 2048 게임의 강화학습 적용

'2048' 퍼즐 게임에 강화학습을 적용할 때, 4x4 격자 공간 위에 놓인 타일들의 배치는 '상태(State)'로, 상하좌우 네 방향으로 타일을 움직이는 행위는 '행동(Action)'으로 정의된다. 타일이 합쳐질 때 발생하는 점수는 '보상(Reward)'으로 주어지며, 에이전트는 누적 보상을 최대화하는 방향으로 정책을 학습한다.

그러나 2048 게임은 단순한 규칙에도 불구하고 강화학습 에이전트가 최적 정책을 학습하기에 여러 도전적인 특성을 내포하고 있다. 첫째, 16개의 타일이 가질 수 있는 값의 조합으로 인해 상태 공간(State Space)이 매우 방대하다. 이는 A절에서 언급한 전통적인 Q-러닝의 테이블 방식으로는 해결이 불가능하며, Q-함수를 근사하는 심층 신경망의 사용이 필수적인 이유가 된다. 둘째, 행동 후 새로운 타일이 '무작위' 위치에 등장하는 확률적(Stochastic) 특성은 학습의 불확실성을 증대시킨다. 동일한 상태에서 동일한 행동을 하더라도 다음 상태가 달라질 수 있기 때문에, 에이전트는 다양한 변수에 강건한 정책을 학습해야 한다. 마지막으로, 보상이 즉각적으로 주어지지 않는 희소 보상(Sparse Reward) 문제와 장기적인 전략의 중요성이다. 당장의 보상을 얻는 근시안적인 움직임보다 보드 전체의 구조를 유리하게 유지하는 것이 최종 점수에 더 큰 영향을 미치므로, 에이전트는 지연된 보상(Delayed Reward)을 효과적으로 평가하고 학습할 수 있어야 한다.

이처럼 거대한 상태 공간, 환경의 확률성, 그리고 보상의 지연이라는 복합적인 문제들은 기본적인 DQN 알고리즘만으로는 최적의 성능을 달성하기 어렵게 만드는 요인이다. 따라서 이러한 문제들을 효과적으로 해결하고 학습 효율을 높이기 위해서는 DQN의 핵심 구조와 학습 메커니즘에 대한 다각적인 개선 방안을 모색할 필요가 있다.

C. 연구 목표

따라서 본 연구의 목표는 퍼즐 게임 2048 환경에 심층 Q-네트워크(DQN)를 적용하여 강화학습 기반의 최적 정책을 학습하는 것이다. 이를 위해 Double DQN, Dueling DQN, Prioritized Experience Replay(PER), ε-탐욕 정책의 수렴 속도, 보상 설계 등 다양한 성능 개선 요인을 실험적으로 검증하였다. 궁극적으로는 여러 방법론을 비교·분석함으로써 2048 게임에서 가장 안정적이고 효율적인 학습 성능을 달성하는 DQN 구조를 제안하고자 한다.

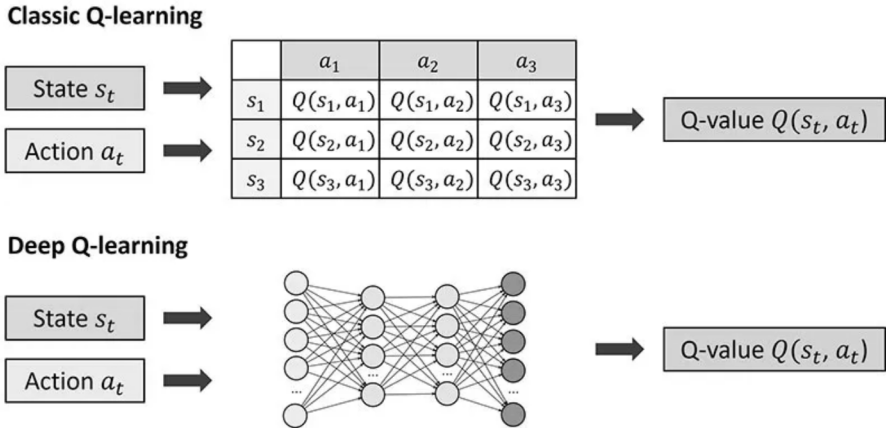
Ⅱ. 관련 연구

A. DQN 구조

1. DQN 개념 및 등장 배경

기존의 Q-learning은 상태-행동 가치 함수(Q-value)를 테이블 형태로 저장하여 학습한다. 이러한 방식은 상태와 행동의 개수가 제한적인 단순 환경에서는 효과적으로 작동하나, 상태공간과 행동공간의 규모가 커지는 복잡한 환경에서는 심각한 한계를 드러낸다. 또한, 상태와 행동의 모든 조합에 대한 Q-value를 저장해야 하므로 막대한 메모리 자원을 요구하며, 상태 차원이 높아질수록 학습 속도 저하와 일반화 성능의 저하가 발생한다. 나아가, 환경 상호작용을 통해 수집되는 데이터 간의 높은 상관성은 학습의 불안정성을 초래하여, 수렴 속도를 늦추거나 최적 정책에 도달하지 못하게 하는 요인이 된다.

[그림 1] Classic Q-learning과 Deep Q-learning의 차이점



이를 해결하고자 Google DeepMind는 2013년에 NIPS, 2015년에 Nature 두 번의 논문을 통해 DQN 알고리즘을 발표했다. DQN은 Deep Q-Network을 줄인 말로, 딥러닝을 활용하여 Q-value를 근사하는 알고리즘이다. 기존에 테이블 형태로 Q-value를 저장하던 방식과는 달리, 신경망으로 복잡한 상태를 입력받아 각 행동에 대한 Q-value를 직접 예측하게 만드는 방식인 것이다.

2. DQN의 핵심 구성 요소

DQN은 CNN, Experience Replay, Target Network, Double DQN, Dueling DQN의 다섯가지 핵심 요소로 구성되어 있다.

1) CNN

DQN은 고차원 상태 공간을 처리하기 위해 합성곱 신경망(Convolutional

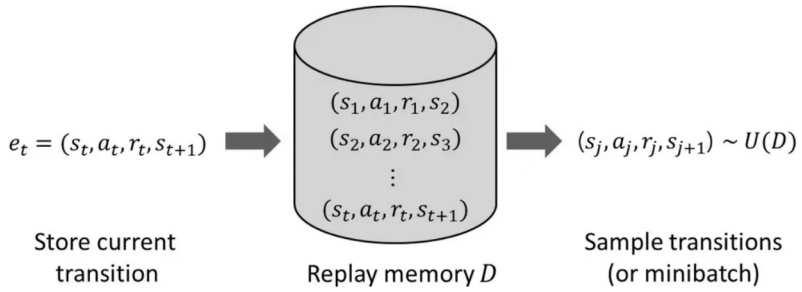
Neural Network, CNN)을 사용한다. CNN은 입력 상태로부터 패턴과 특성을 추출하여, 합성곱 계층(Convolution Layer), 비선형 활성화 함수(ReLU), 풀링(Pooling) 과정을 거쳐 압축된 형태의 특징 맵(Feature Map)을 생성한다. 이와 같은 과정을 통해 생성된 특징 백터는 완전 연결층(Fully Connected Layer)에 전달되어 각행동에 해당하는 Q-value로 변환된다. 해당 구조는 복잡한 시각적 정보를 효과적으로 학습하여 DQN의 정책 성능 향상에 크게 기여한다는 특징이 있다.

2) Experience Replay

DQN에서 경험 재생(Experience Replay)은 Q-learning 기반 강화학습의 안정성과 효율성을 향상시키는 핵심 기법 중 하나이다. 강화학습의 특성상 에이전트는 환경과의 상호작용을 통해 시간 순서에 따른 경험을 수집하며, 이때 연속된 경험 샘플 간에는 높은 상관관계가 존재한다. 이러한 상관관계는 Q-value 추정의 편향을 유발하여 학습의 불안정성을 초래할 수 있다. 이를 해결하고자 Experience Replay는 다음과 같은 과정을 거친다.

a) 매 스텝마다 추출된 샘플 e(t) = (s(t), a(t), r(t), s(t+1))을 replay memory D에 저장한다.
b) Replay Memory D에 저장된 샘플들을 uniform하게 랜덤 추출하여 Q-update 학습에 이용한다.

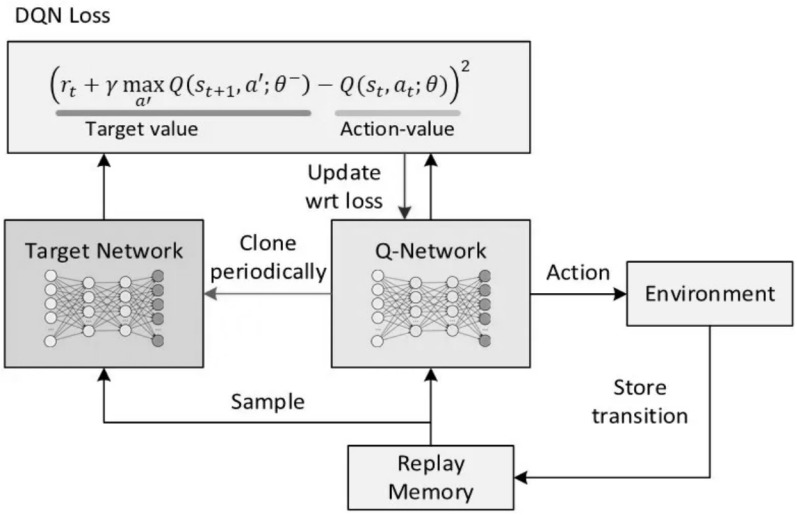
[그림 2] Experience Replay



위 과정은 연속된 데이터 간의 상관성을 약화시키고, 데이터의 분포를 다양화하여 전체 경험 분포를 더 반영한 Q-value 학습을 가능하게 한다.

3) Target Network

[그림 3] DQN의 target network 구조



4) Double DQN

또한, 본 연구에서는 기존 DQN 구조의 한계를 보완하기 위하여 Double DQN 기법을 적용하였다. 일반적인 DQN은 타깃 값을 계산할 때 동일한 네트워크를 사용하여 다음 상태에서의 최적 행동을 선택하고 그 행동의 가치를 평가하기 때문에, Q값이 과대 추정(overestimation)되는 문제가 발생한다. Double DQN은 이를 완화하기 위해 두 네트워크를 분리하여 사용한다. 즉, 현재 Q-네트워크(Q(s,a;θ))로 다음 상태에서의 최적 행동을 선택하고, 그 행동의 가치는 타깃 네트워크(Q(s,a;θ^-))로 평가한다. 이때 손실 함수의 목표값은 다음과 같이 정의된다.

$$y_t^{\text{DoubleDQN}} = r_t + \gamma\, Q\left(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta); \theta^-\right)$$

이와 같이 행동 선택과 평가를 분리함으로써 과대 추정 문제를 줄이고, 보다 안정적이고 신뢰성 있는 학습을 가능하게 한다.

5) Dueling DQN

아울러, 본 연구에서는 Dueling DQN 구조를 적용하여 상태 가치와 행동 가치를 분리한 학습을 수행하였다. 전통적인 DQN은 모든 상태-행동 쌍에 대해 Q값을 직접 추정하지만, 일부 상태에서는 특정 행동이 결과에 큰 영향을 미치지 않는다. 이러한 한계를 극복하기 위해 Dueling 구조는 네트워크의 출력부를 상태 가치(Value function) V(s)와 이득(Advantage function) A(s,a)로 분리한다. 최종 Q값은 다음과 같이 두 함수를 결합하여 계산된다.

$$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a')\right)$$

이 방식은 특정 행동의 영향력이 크지 않은 상태에서도 의미 있는 상태 가치를 학습할 수 있게 하며, 전반적인 학습 효율성과 일반화 성능을 높인다.

3. DQN의 동작 과정

DQN의 동작 과정은 다음 단계들을 거쳐 진행된다.

1) 상태 관측

에이전트는 매 스텝마다 환경으로부터 현재 상태를 입력받는다. 본 연구에서는 2048 보드의 상태를 원-핫 인코딩 형태의 4×4×16 텐서로 변환하여 Q-네트워크의 입력으로 사용하였다. 이러한 상태 표현은 보드 내 타일 값의 분포를 명확히 반영하며, 신경망이 고차원 상태공간을 효과적으로 학습할 수 있도록 한다.

2) 행동 선택

Q-네트워크는 주어진 상태를 입력받아 가능한 모든 행동에 대한 Q값을 출력한다. 에이전트는 이 중 최대 Q값을 갖는 행동을 선택하거나, 일정 확률로 무작위 행동을 수행한다. 이러한 ϵ-탐욕 정책(ϵ-greedy policy)은 학습 초기에 탐험(exploration)을 유도하고, 시간이 지남에 따라 활용(exploitation)을 강화함으로써 균형 있는 학습을 가능하게 한다.

3) 환경 반응

선택된 행동이 환경에 적용되면 게임은 한 스텝 진행되며, 그 결과로 다음 상태와 보상이 반환된다. 예를 들어, 2048 게임에서는 타일이 병합될 경우 양의 보상이 주어지고, 불법적인 이동의 경우 보상은 0으로 처리된다. 이 과정은 에이전트가 환경과의 상호작용을 통해 학습에 필요한 경험을 축적하는 단계이다.

4) 경험 저장

각 스텝에서 얻은 전이 (s(t), a(t), r(t), s(t+1))는 경험 재생 메모리에 저장된다. 이 메모리는 일정 용량을 가지며, 오래된 경험은 새로운 경험에 의해 순환적으로 대체된다. 저장된 경험은 샘플링 과정을 통해 무작위로 추출되어 학습 데이터로 활용되며, 데이터 간 상관관계를 줄이고 분포의 다양성을 확보하는 데 기여한다.

5) 학습(Training)

에이전트는 일정 주기마다 경험 재생 메모리에서 무작위로 추출한 미니배치를 이용해 신경망을 학습한다. 학습의 목표는 현재 Q-네트워크가 출력한 Q값과 타깃 Q값 간의 차이를 최소화하는 것이다. 손실 함수는 평균제곱오차(MSE)로 정의되며, 다음과 같이 표현된다.

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim D}\left[\left(y - Q(s,a;\theta)\right)^2\right]$$

여기서 (s, a, r, s')는 경험 메모리 D로부터 샘플링된 전이며, 타깃 값 y는

$$y = r + \gamma \max_{a'} Q_{\text{target}}(s', a'; \theta^-)$$

으로 계산된다. 여기서 γ는 할인율, θ는 현재 Q-네트워크의 파라미터, θ^-는 타깃 네트워크의 파라미터를 의미한다.

네트워크의 파라미터는 손실 함수의 그래디언트를 계산하여 경사하강법 방식으로 갱신된다. 구체적으로,

$$\theta \leftarrow \theta - \alpha \nabla_\theta L(\theta)$$

로 업데이트가 이루어지는데, 여기서 α는 학습률을 나타낸다. 이러한 반복적인 학습 과정을 통해 Q-네트워크는 상태-행동 쌍에 대한 가치 함수를 점진적으로 근사하게 된다.

6) 타겟 네트워크 업데이트

학습의 안정성을 확보하기 위하여 별도의 타깃 네트워크를 유지하며, 일정 주기마다 주 네트워크의 가중치를 타깃 네트워크로 복사한다. 이러한 구조는 학습 목표와 추정치가 동시에 변동하여 발생하는 불안정성을 완화하고, 발산을 방지하며, 더 안정적인 수렴을 가능하게 한다.

B. DQN 장단점 및 한계

1. DQN 장점

앞서 말한 것과 같이 DQN 알고리즘은 CNN을 이용해 Q-value를 추산하기 때문에, 기존의 테이블 형태 입력 데이터에서 벗어나서 입력 데이터의 공간적 정보의 복잡성을 효과적으로 줄이고 더 효율적으로 학습할 수 있게 됐다. CNN이 필터와 풀링을 통해 특징을 뽑아내면, 복잡한 환경에서도 안정적으로 학습이 가능하다.

2. DQN 단점 및 한계, 그리고 연구의 방향성

그럼에도 불구하고 위 알고리즘은 두 가지의 단점을 가지고 있다. 첫째, 수렴 속도가 느리다는 단점을 가지고 있다. DQN은 딥러닝과 강화학습의 결합이다. 강화학습이 그러하듯 처음 몇 단계 동안 에이전트들은 주어진 환경을 오롯이 혼자 탐색하고 외부의 과대한 개입 없어 스스로 학습해야 한다. 본 실험에 적용하면 1개의 에이전트가 미지의 4x4 형태의 공간 상태 안의 가능한 모든 경로를 탐색해야 한다는 것이고, 이는 기존의 간단한 Q-learning 알고리즘과 달리 계산량을 크게 증폭시키고 수렴 속도를 늦출 수 밖에 없다.

둘째, 탐험 정도에 따른 매 학습 결과의 차이가 존재한다. 상태-행동 가치함수를 테이블 형태로 저장하는 기존의 Q-learning 알고리즘에서도, 인공신경망을 이용해서 Q-value를 근사하는 DQN 알고리즘에서도 탐색과 활용(utilization)은 중요한 개념으로 작용한다. 탐색이 무작위로 행동하며 좋은 성과와 나쁜 성과 중 한 가지를 내는 것이라면, 활용은 좋은 성과를 낸 탐색 행동을 통해 끊임없이 최대 보상을 얻기 위해 이를 이용하는 것을 의미한다. 활용도가 높다는 것은 곧 탐색 비율 대비 성과가 좋다는 것을 의미한다. 따라서 에이전트는 최적의 점수를 달성하기 위해 항상 최대 보상 값을 얻는 선택을 해야 하지만, 실제로 실험을 해봤을 때 샘플이 충분하지 않거나, 경험이 부족해서 환경에 대한 정보가 부족할 때는 최적 전략을 도출하지 못한 것을 확인했다.

따라서 본 연구에서는 DQN 알고리즘을 적용한 에이전트의 수렴 속도를 늦추지 않으면서도 최대 보상을 추구하는 최적 전략을 유지할 수 있도록 000, 000, 000, 그리고 PER 알고리즘을 한 번씩 적용해 보며, DQN 알고리즘 성능을 최적화하고자 한다.

C. 2048 게임의 강화학습 연구 사례

앞선 2-b 절에서 DQN이 가지는 느린 수렴 속도와 탐험-활용의 딜레마라는 이론적 한계를 논의했다. 본 절에서는 이러한 한계를 극복하기 위해 다양한 알고리즘적 개선 사항을 종합적으로 적용한 구체적인 공개 프로젝트 사례(YangRui2015, 2048_env)를 심층적으로 탐구하고자 한다. 이 사례는 단순히 높은 점수를 달성하는 것을 넘어, 어떻게 DQN의 문제점들을 완화하고 2048이라는 복잡한 환경에 성공적으로 안착시키려 시도했는지 그 과정과 방법을 상세히 보여준다는 점에서 본 연구에 시사점을 제공한다.

1. 문제 접근 방식:

1) CNN 기반 상태 표현과 로그 변환 보상 설계

해당 프로젝트는 2048 게임의 4x4 격자 보드를 CNN(Convolutional Neural Network)의 입력으로 직접 사용한다. 이는 2-b에서 언급한 DQN의 장점과 같이, CNN이 타일의 위치와 값에 따른 공간적 특징을 자동적으로 추출하게 하여 복잡한 게임 상태를 에이전트가 효과적으로 학습하도록 만든다.

특히 주목할 점은 보상 설계(Reward Shaping) 방식이다. 일반적인 접근법은 타일이 합쳐졌을 때 얻는 점수(r)를 그대로 보상으로 사용하지만, 이 경우 4점에서 2048점까지 보상의 편차가 매우 커 학습을 불안정하게 만들 수 있다.

이 프로젝트에서는 log2(1+r)라는 로그 변환을 보상 함수로 사용하였다. 이는 점수가 높을수록 보상을 더 주되, 그 증가폭을 점차 줄여줌으로써 특정 큰 보상 값이 Q-value 추정에 과도한 영향을 미치는 것을 방지하고 학습 과정을 안정화시키는 효과적인 전략이다.

2) DQN 한계 극복을 위한 알고리즘적 개선 적용

이 프로젝트의 핵심은 바닐라 DQN(Vanilla DQN)을 그대로 사용한 것이 아니라, 그 한계를 극복하기 위해 알려진 여러 기법들을 적극적으로 도입했다는 점이다.

- 학습 효율성 및 수렴 속도 개선: 가장 두드러지는 특징은 PER(Priority Experience Replay)의 도입이다. 이는 모든 경험을 동일한 가치로 취급하는 기존 방식과 달리, TD-오차가 큰, 즉 에이전트가 예측하지 못했던 '놀라운' 경험을 우선적으로 학습에 사용한다. 이를 통해 에이전트는 자신의 실수를 더 빠르고 효율적으로 교정할 수 있어 전반적인 학습 속도와 성능 개선을 꾀할 수 있다. 제공된 학습 결과 이미지의 제목이 'CNN입력 + Priority'인 점에서 이 기법이 핵심적인 역할을 했음을 유추할 수 있다.
- 학습 안정성 확보: DQN의 고질적인 문제인 Q-value의 과대평가(Overestimation) 문제를 완화하기 위해 Double DQN이 적용되었다. 또한, 목표 네트워크(Target Network)를 일정 주기마다 통째로 복사하는 대신 가중치를 서서히 업데이트하는 소프트 타겟 업데이트(Soft target replacing) 방식을 사용하고, 경사도 폭주(Exploding Gradients)를 막기 위해 그래디언트 클리핑(Clip gradient norm)을 적용하는 등 학습 과정 전반의 안정성을 높이기 위한 다각적인 장치를 마련했다.

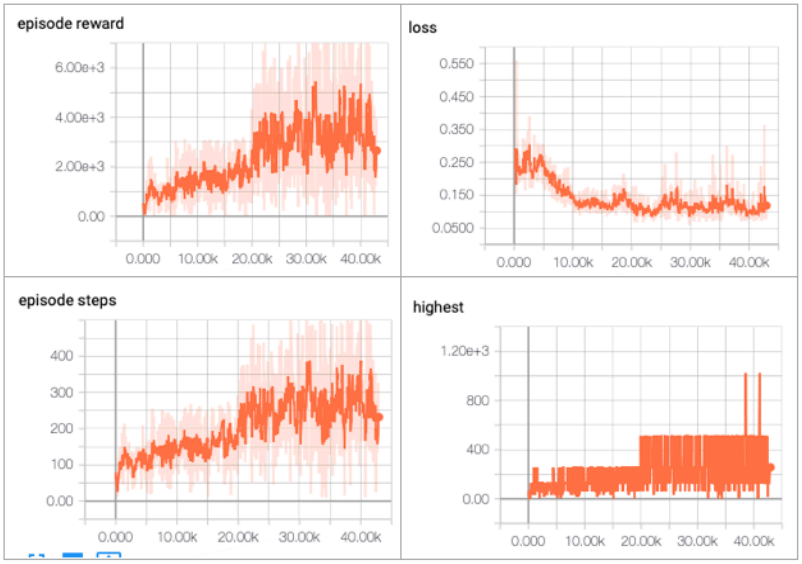
2. 결과 분석

[그림 4] YangRui2015의 DQN 학습 결과 종합 그래프

CNN输入+Priority

效果最好

参数	episodes	卷积层	全连接	dqn parm	other	result
值	3e4	(1,32) (32,64) kernel_size=3 stride=1 padding=1 保持输入大小不变	(64*16,512) (512,128) (128,4)	buffer size: 1e4 learning rate: 1e-4 epsilon:0.1 batch size:128 train interval step:1 gamma:0.99 clip norm:1 soft update:0.1 target replacing:300	epsilon decay:100 episodes reward:log2(1+r)	training mean:3300 eval mean:5100 eval max:11500 highest:1024



학습 결과를 보면, 약 40,000 에피소드에 걸쳐 학습이 진행됨에 따라 episode reward와 episode steps이 꾸준히 우상향하는 것을 볼 수 있다. 손실(loss) 값 또한 초기에 급격히 감소한 후 안정적으로 수렴하며, 이는 앞서 언급한 여러 안정화 기법들이 효과적으로 작동했음을 시사한다.

인상적인 부분은 최고 타일(highest) 그래프이다. 에이전트는 오랜 기간 512 타일의 벽을 넘지 못하다가, 40,000 에피소드 부근에서 마침내 1024 타일을 달성하는 '퀀텀 점프'를 보여준다. 평가 단계에서의 평균 점수(eval mean)가 5100점, 최대 11500점에 도달한 것 역시 이러한 성공적인 학습을 뒷받침한다.

3. 종합 및 시사점

YangRui2015의 프로젝트 사례는 2048 게임에 DQN을 성공적으로 적용하기 위해서는 단일 기법의 적용을 넘어, 알고리즘 자체의 내재적 한계를 다각적으로

극복하려는 노력이 필요함을 명확하게 보여준다. PER을 통한 학습 효율 개선, Double DQN을 통한 과대평가 문제 완화, 그리고 다양한 안정화 기법들이 조합되었을때, 강력한 시너지를 발휘하여 2048과 같은 복잡한 문제에 대한 해결 가능성을 제시한다.

Ⅲ. 연구 방법

A. 환경

1. 환경 버전

본 연구에서는 직접 구현한 Gym 스타일의 2048 환경을 사용하였다. 환경은 OpenAI Gym의 규약을 따르도록 설계되었으며, reset, step, render 메서드를 지원한다. 이를 통해 다양한 DQN 기반 알고리즘을 동일한 인터페이스에서 비교할 수 있도록 하였다.

2. 상태 표현

상태(state) 표현은 4×4 크기의 보드를 원-핫 인코딩 방식으로 변환하여 사용하였다. 각 칸의 값 {2,4,⋯,2^16}}을 16개의 채널에 투영하여 4×4×16 형태의 텐서로 구성하였다. 이러한 방식은 CNN 기반 네트워크에서 고차원 입력을 효과적으로 처리할 수 있도록 하며, 상태 표현의 일반화 성능을 높인다.

3. 행동 정의

행동(action) 공간은 상·하·좌·우 네 가지 방향으로 정의되었다. 합법적 이동 시 타일 병합과 무작위 타일 추가가 이루어지며, 타일은 90% 확률로 2, 10% 확률로 4가 생성된다. 불법적 이동의 경우 보드 상태가 변하지 않으므로 보상은 0으로 주어지며, 새로운 타일도 생성되지 않는다. 다만, 스텝 수는 정상적으로 증가하여 학습 과정에서 불필요한 시도가 누적되도록 처리하였다. 또한 환경은 현재 가능한 행동만을 표시하는 액션 마스크(action mask)를 제공하여 무효 행동의 선택을 방지한다.

4. 보상 함수 설계

보상 함수 설계는 단순성을 유지하면서도 학습을 유도할 수 있도록 고안되었다. 보상은 두 가지 항목으로 구성된다. 첫째, 이동 과정에서 병합된 타일의 총합을 로그 스케일로 변환한 병합 보상이다. 둘째, 현재 보드의 빈 칸 개수에 로그 함수를 적용한 빈 칸 보상이다. 이 두 보상은 각각 가중치를 곱하여 합산되며, 단조성(monotonicity)이나 평탄성(smoothness) 보상과 같은 추가 휴리스틱은 포함하지 않았다. 이러한 단순화는 과적합을 방지하고 학습 안정성을 높이는 데 목적이 있다.

B. 모델 구조

본 연구에서는 2048 게임 환경에 최적화된 DQN 계열 신경망 구조를 사용하였다. 기본 구조는 합성곱 신경망(Convolutional Neural Network, CNN)을 기반으로 하였으며, 선택적으로 Double DQN, Dueling DQN, 그리고 우선순위 경험 재생(Prioritized Experience Replay, PER)을 적용할 수 있도록 설계하였다. 이를 통해 다양한 변형 기법들이 성능에 미치는 영향을 실험적으로 비교하였다.

1. 기본 CNN 네트워크

기본 네트워크 구조는 다음과 같다. 입력은 4×4×16 크기의 원-핫 인코딩 상태이며, 첫 번째 차원(16채널)은 타일 값을 나타낸다. 세 개의 합성곱 계층(conv1, conv2, conv3)과 배치 정규화 계층이 연속적으로 연결되어 있으며, 각 합성곱 계층은 128개의 채널과 3×3 커널을 사용한다. 활성화 함수로는 ReLU를 적용하였다. 합성곱 계층을 통과한 특성 맵은 전결합층(fully connected layer)으로 전달되며, 은닉층의 차원은 512로 설정되었다. 전결합층 두 개를 거친 후 출력층에서 네 개의 Q-value(상·하·좌·우)가 산출된다. 과적합을 방지하기 위해 은닉층 사이에는 드롭아웃(0.1)이 적용되었다. 가중치 초기화는 He 초기화를 사용하였다.

2. Dueling 구조

Dueling DQN 구조는 기본 CNN 네트워크를 기반으로, 마지막 은닉층 출력을 Value stream과 Advantage stream으로 분리하였다. Value stream은 상태의 가치를 단일 스칼라 값으로 추정하며, Advantage stream은 각 행동의 상대적 우위를 추정한다. 최종 Q-value는 다음과 같이 두 출력을 결합하여 계산한다.

$$Q(s,a) = V(s) + A(s,a) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a')$$

여기서 V(s)는 Value 함수, A(s,a)는 Advantage 함수이며, |A|=4는 행동 공간의 크기를 의미한다.

3. Double DQN

Double DQN 기법은 타겟 Q-value 계산 시 온라인 네트워크와 타겟 네트워크를 분리하여 사용한다. 온라인 네트워크는 다음 상태에서의 최적 행동을 선택하고, 타겟 네트워크는 해당 행동의 Q-value를 평가한다. 이 과정을 통해 일반 DQN이 가지는 과추정(overestimation) 문제를 완화하였다.

4. Replay Buffer

경험 재생 메모리는 두 가지 방식으로 구현하였다. 기본적인 균등 샘플링을 제공하는 Replay Buffer와, TD-error 크기에 비례하여 샘플의 확률을 조정하는 Prioritized Replay Buffer를 지원하였다. 후자의 경우 중요도 가중치(importance sampling weight)를 적용하여 샘플링 편향을 보정하였다.

이와 같은 신경망 구조는 CNN의 지역적 패턴 추출 능력을 활용하여 보드 상태를 효과적으로 인코딩하며, Double DQN, Dueling DQN, PER을 통해 DQN의 기존 한계를 단계적으로 보완할 수 있도록 구성되었다.

C. 학습 설정

1. 학습 반복

본 연구에서는 총 3,000에피소드 동안 학습을 수행하였다. 각 에피소드의 최대 스텝 수는 5,000으로 제한하였으며, 이때 환경은 2048 보드 게임에서의 상호작용을 통해 상태 – 행동 – 보상 전이를 축적하였다.

2. 하이퍼파라미터

하이퍼파라미터 설정은 다음과 같다. 경험 재생 버퍼의 크기는 100,000으로 설정하였고, 미니배치 크기는 64를 사용하였다. 할인율 γ는 0.99로 두어 장기 보상을 고려하도록 하였으며, 학습률은 1×10^{-4}로 설정하였다. 탐험 – 활용 균형을 위한 ϵ-greedy 정책은 ϵ_start=0.9, ϵ_end=0.01, decay 파라미터는 30,000으로 두어 학습이 진행됨에 따라 점진적으로 무작위 탐색을 줄였다. 타깃 네트워크는 1,000 스텝마다 동기화되도록 설정하였다.

3. 최적화 기법

최적화 기법으로는 Adam 옵티마이저를 사용하였다. 그래디언트 폭발을 방지하기 위해 학습 과정에서 그래디언트 클리핑을 적용하였다. 또한, Double DQN, Dueling 구조, 우선순위 경험 재생(PER)을 활성화하여 각각 과추정 완화, 상태 가치 분리, 샘플 효율성 개선 효과를 기대하였다.

4. 평가 및 로딩

Soft update 계수 τ는 별도로 사용하지 않고, 하드 업데이트 방식으로 타깃 네트워크를 주기적으로 갱신하였다. 학습 과정 동안 에이전트의 성능은 일정

간격(eval interval=100)마다 10에피소드의 평가를 통해 확인하였으며, 결과는 저장 및 시각화되었다.

D. 평가 방법

1. 평가 지표

본 연구에서는 학습된 에이전트의 성능을 다각적으로 측정하기 위하여 다양한 평가 지표를 설정하였다. 주요 지표는 에피소드 평균 점수, 최고 타일, 그리고 에피소드 길이이다. 에피소드 평균 점수는 각 에피소드에서 획득한 최종 점수를 평균한 값으로, 정책의 전반적인 성능을 나타낸다. 최고 타일은 게임 종료 시 보드에 남은 가장 큰 타일의 값으로 정의되며, 2048 게임의 특성을 고려할 때 정책의 질적 수준을 직관적으로 보여준다. 또한 에피소드 길이는 종료까지 소요된 스텝 수를 기록하여, 에이전트의 생존 능력과 전개 능력을 보조적으로 평가하는 지표로 활용하였다.

2. 평가 절차

평가는 학습 과정과 별도로 이루어졌으며, 탐험 요소를 배제한 순수 탐욕 정책(greedy policy)을 사용하였다. 이를 위해 평가 시에는 ε -탐색을 비활성화하고, 액션 마스킹을 적용하여 무효한 행동을 제거하였다. 주기적 평가의 경우, 학습 중 100에피소드마다 10회의 독립된 에피소드를 실행하고 평균 점수를 기록하였다. 학습 완료 후에는 100게임을 독립적으로 수행하여 최종 성능을 종합적으로 측정하였다. 이때 각 게임에 대해 최종 점수, 최고 타일, 스텝 수, 그리고 최종 보드 상태를 저장하여 후속 분석에 활용하였다.

3. 비교 방식

베이스라인과 개선 모델의 비교는 동일한 환경, 동일한 하이퍼파라미터, 동일한 학습 에피소드 수를 전제로 수행되었다. 비교 대상에는 기본 DQN뿐 아니라 Double DQN, Dueling DQN, 우선순위 경험 재생(Prioritized Replay, PER), 그리고 액션 마스킹 기법이 포함되었다. 각 기법은 독립적으로 또는 조합하여 적용되었으며, 이를 통해 개선 효과를 정량적으로 평가하였다. 비교는 주기적 평가에서의 점수 추세를 통해 수렴 속도와 안정성을 분석하고, 최종 100게임 평가에서의 평균 및 표준편차, 최고 타일 분포를 통해 성능의 절대적 수준과 변동성을 분석하는 방식으로 이루어졌다.

4. 분석 방식

마지막으로 학습 과정의 모니터링은 보상, 점수, 최고 타일, 스텝 수, 그리고 손실 값을 시계열로 기록하고, 이동평균을 적용하여 시각화하였다. 이를 통해 각 개선 기법이 수렴 속도, 안정성, 데이터 효율성에 미치는 정성적 영향을 관찰하였다. 추가적으로 최종 평가에서는 최고 타일 달성률을 분포 형태로 제시하고, 대표적인 최고 성능 게임과 최저 성능 게임의 최종 보드를 시각화하여 정책의 전략적 패턴을 해석하였다.

Ⅳ. DQN 성능 개선 요인

A. Double DQN & Dueling DQN

본 절에서는 DQN의 대표적인 개선 아키텍처인 Double DQN과 Dueling DQN을 각각 적용하고, 두 기법을 함께 사용했을 때의 성능 변화를 수치적으로 비교 분석하여 2048 게임에 가장 효과적인 네트워크 구조를 탐색하고자 한다.

1. 실험 설계

성능 비교를 위해 다음과 같이 네 가지 조합으로 실험을 구성하였다. 모든 실험은 동일한 하이퍼파라미터를 사용하여 3000 에피소드 동안 학습을 진행했으며, 학습된 모델의 성능은 별도의 100개 에피소드에 대한 평가를 통해 측정하였다.

실험 1: 기본 DQN (Double DQN: False, Dueling DQN: False)

실험 2: Double DQN 적용 (Double DQN: True, Dueling DQN: False)0

실험 3: Dueling DQN 적용 (Double DQN: False, Dueling DQN: True)

실험 4: Double DQN + Dueling DQN 적용 (Double DQN: True, Dueling DQN: True)

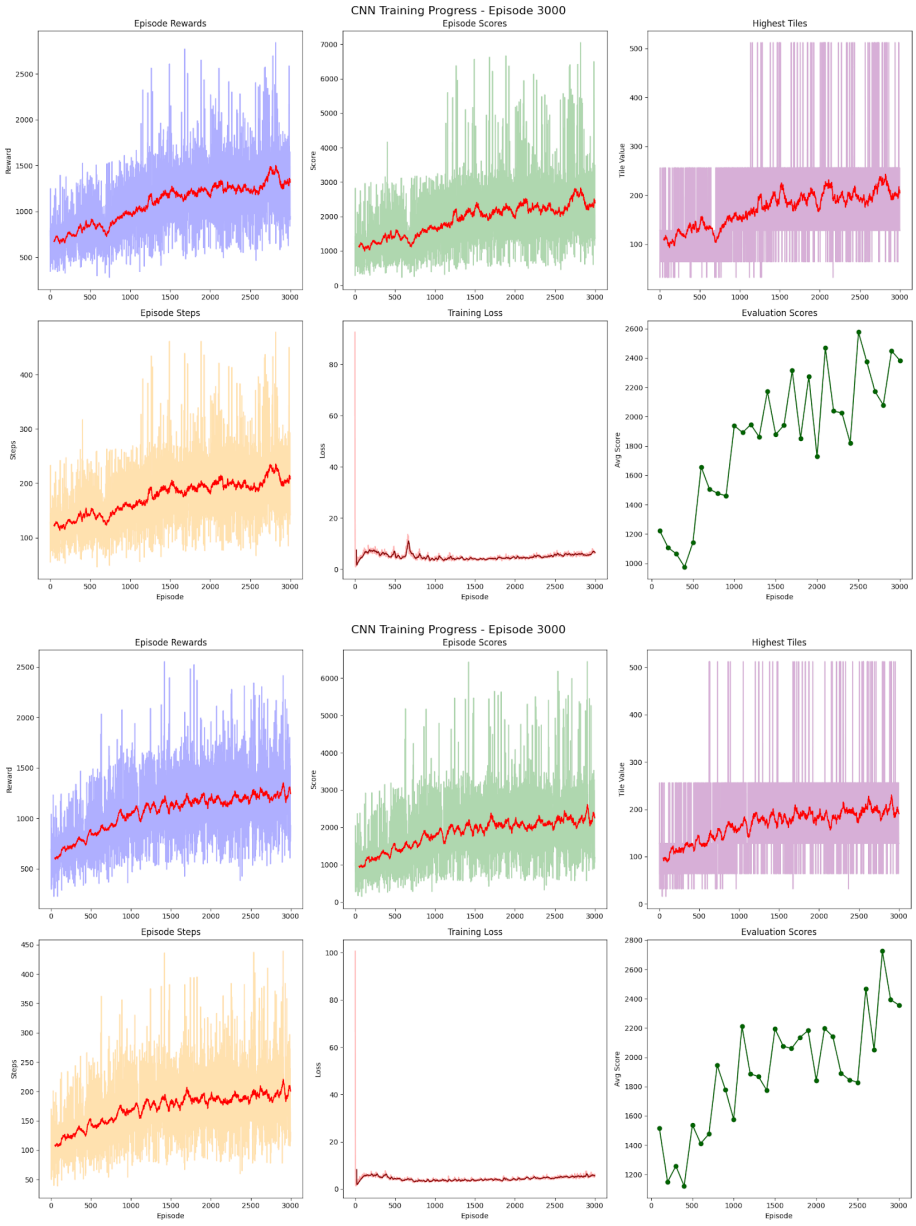
2. 실험 결과 및 분석

실험 결과, 전반적인 성능은 **실험 4 〉 실험 3 〉 실험 2 〉 실험 1** 순으로 나타났다. 각 실험의 주요 지표를 정리하면 아래 표와 같다.

[표1] 아키텍처 별 성능 결과 요약 표

구분	실험 1 (Base)	실험 2 (Double)	실험 3 (Dueling)	실험 4 (combined)
평균 점수	2241.0	2270.2	6007.2	**7403.1**
최고 점수	7048	6440	15776	**17456**
평균 최고 타일	198.4	202.9	426.2	**506.9**
달성 최고 타일	512	512	1024	1024

[그림 5] 학습 진행 그래프 (실험 1, 실험 2)



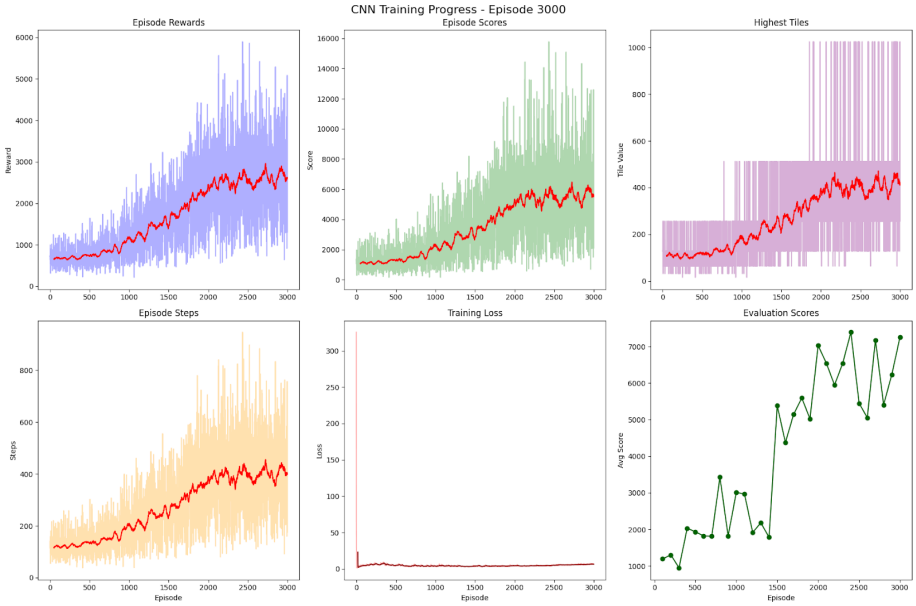
1) Double DQN의 제한적 성능 향상 (실험 1 vs 실험 2)

기본 DQN (실험 1)에 Double DQN만을 적용한 실험 2의 경우, 평균 점수 (2241.0 → 2270.2) 와 평균 최고 타일 (198.4 → 202.9) 에서 근소한 성능 향상을 보였으나 그 차이는 유의미하지 않았다. 이는 Double DQN의 역할과 2048 게임의 특성에서 기인한 것으로 분석된다.

Double DQN은 Q-value의 과대평가(Overestimation) 문제를 완화하여 학습 안정성을 높이는 데 기여하지만, 2048 게임과 같이 어떤 행동을

선택하는 것보다 현재 상태(State)의 가치를 정확히 평가하는 것이 더 중요한 환경에서는 그 효과가 제한적일 수 있다. 즉, 잘못된 가치 추정 문제를 일부 보정해주기는 하나, 상태 자체에 대한 평가 능력을 근본적으로 개선하지는 못하기에 단독 사용 시 성능의 극적인 향상을 이끌어내기는 어려웠다.

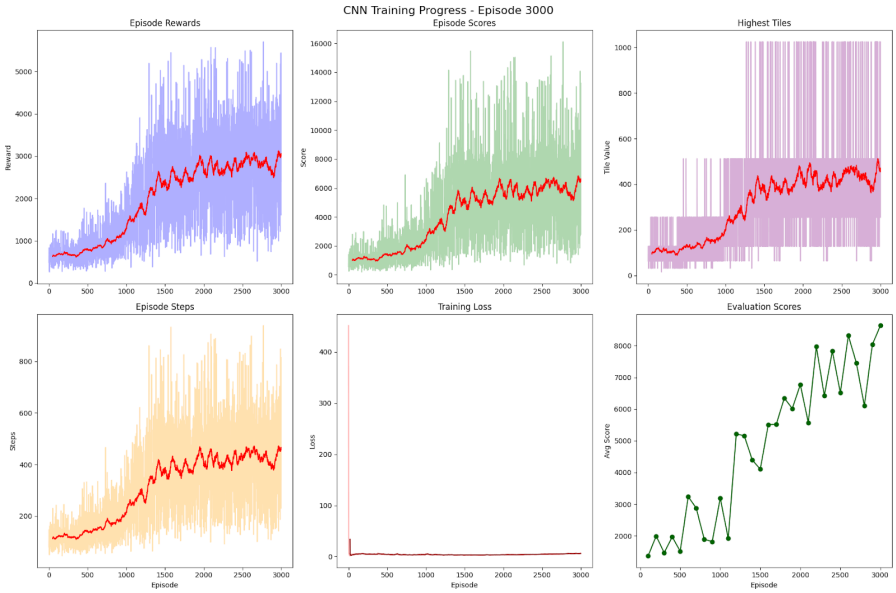
[그림 6] 학습 진행 그래프 (실험 3)



2) Dueling DQN의 폭발적인 성능 향상 (실험 1 vs 실험 3)

반면, Dueling DQN을 단독으로 적용한 실험 3은 모든 지표에서 괄목할 만한 성능 향상을 보였다. 평균 점수는 2241.0점에서 6007.2점으로 약 2.7배 증가했으며, 평균 최고 타일 또한 198.4에서 426.2로 2배 이상 상승했다. 특히, 최초로 1024 타일을 안정적으로 달성하기 시작했다는 점이 가장 큰 차이점이다. 이는 상태 가치를 평가하는 V(s)와 행동의 이점을 평가하는 A(s,a)를 분리하는 Dueling 아키텍처의 특징 덕분이다. 2048 게임에서는 특정 행동(e.g., '위로 이동')이 항상 좋은 것이 아니라, 보드 상태(State) 자체가 '좋은 상태' 혹은 '나쁜 상태'로 구분되는 경우가 많다. Dueling DQN은 이러한 상태의 가치를 직접적으로 학습하기 때문에, 어떤 행동을 해도 점수 획득이 어려운 절망적인 상태와 높은 점수를 기대할 수 있는 좋은 상태를 더 효율적으로 구분하고 학습할 수 있다.

[그림 7] 학습 진행 그래프 (실험 4)



3. 시너지를 통한 최적 성능 달성 (실험 3 vs 실험 4)

실험 4는 Dueling DQN의 뛰어난 상태 평가 능력 위에 Double DQN의 학습 안정성을 더한 조합이다. 실험 3 대비 평균 점수는 6007.2점에서 7403.1점으로 약 23% 추가 상승했으며, 평균 최고 타일 역시 426.2에서 506.9로 눈에 띄게 증가했다.

이는 Dueling 아키텍처가 효율적으로 상태와 행동의 가치를 분리하여 학습하는 과정에서 발생할 수 있는 미세한 과대평가 경향을 Double DQN이 효과적으로 억제해주었기 때문으로 분석된다. 즉, Dueling DQN이 '무엇을 배워야 하는지'에 대한 올바른 방향을 제시하면, Double DQN은 그 학습 과정이 '더 안정적이고 정확하게' 이루어지도록 돕는 시너지 효과를 발휘한 것이다.

4. 종합 및 시사점

네 가지 실험을 통해 2048 게임 환경에서는 *Dueling DQN* 아키텍처가 성능 향상에 가장 결정적인 역할을 하며, Double DQN은 학습 안정성을 더해주는 보조적인 역할을 수행함을 확인했다. Dueling DQN의 상태 가치(V)와 행동 우위(A) 분리 메커니즘은 복잡한 게임 국면을 평가하는 데 탁월한 성능을 보였으며, 이는 본 모델의 성능을 끌어올리는 핵심 요인으로 작용한다.

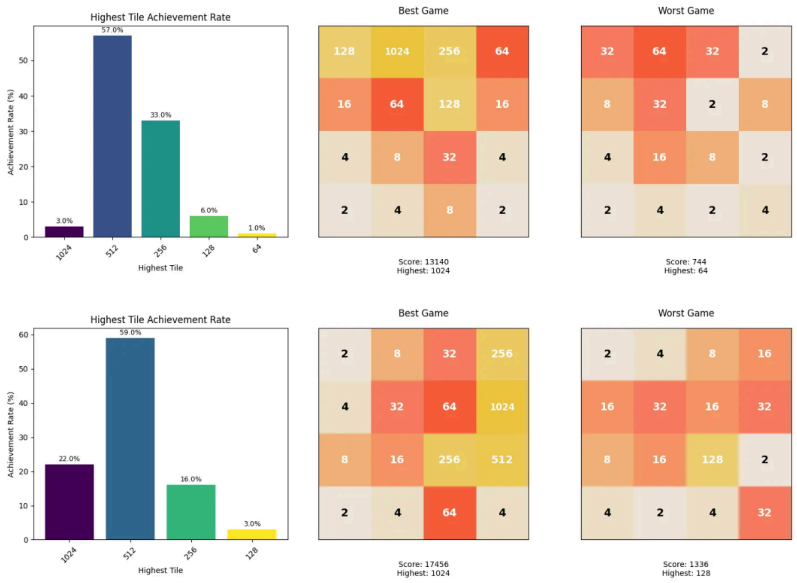
이러한 가장 효율적인 네트워크 아키텍처 조합(Double + Dueling)을 확립한 것을 바탕으로, 다음 4-B절에서는 에이전트의 학습 방식, 즉 경험 데이터를 어떻게 활용하는지에 따라 성능이 어떻게 달라지는지 분석하고자 한다. 이를 위해 표준 경험 리플레이(Replay Buffer)와 우선순위 경험 리플레이(PER)의 성능을 비교 분석할 것이다.

B. Replay Buffer vs Prioritized Experience Replay (PER)

1. Replay Buffer

심층 Q-러닝을 비롯하여 전반적인 강화학습 에이전트는 보상을 최대화할 수 있도록 정책을 학습하는데, 특히 상태, 행동, 보상 그리고 다음 상태를 경험으로 묶어 버퍼(buffer)에 저장해뒀다가 학습을 진행하면서 다시 꺼내쓰는 Replay Buffer를 자주 사용한다. 이때 오래된 경험을 지우고 새로운 경험을 순서대로 채운 버퍼에서 무작위로 꺼내 쓰는 방식과, 우선순위를 매겨 경험을 골라 학습하는 방식인 Prioritized Experience Replay(PER)가 존재하는데, 이 때 각 모델에서 도출된 결과값을 통해 모델의 성능을 비교해 보고자 한다. 연속적인 경험만을 바탕으로 한 학습은 시간적인 상관관계가 높기 때문에 예측이 가능해지고 이는 학습을 불안정하게 만들 수 있다. 아래는 경험을 무작위로 골라 학습했을 때와 특정 조건에 따라 우선순위를 매기고 경험을 골라서 학습했을 때의 성능을 비교한 표이다.

[그림 8] 훈련 모델의 경험 추출 방식에 따른 성능 차이



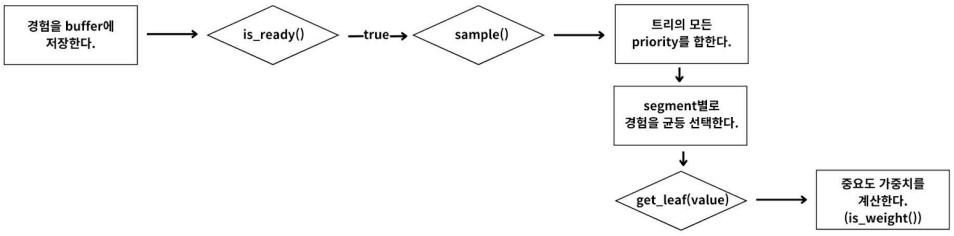
본 실험은 Replay Buffer로 3000 에포크 훈련한 모델을 100 에피소드 씩 진행했을 때의 결과와 Prioritized Experience Replay로 동일하게 훈련한 후 결과를 낸 DQN 모델의 모습이다. 전자의 경우 모델이 100회 동안 평균적으로 5831.4점을 내고 평균 최대 타일 *성적(매회 기록한 숫자 타일값 중 최댓값)*이 415.4점을 웃돈 반면에 후자는 8224.6점을 내고 약 572.2점을 기록하며 더 우수한 성능을 증명했다. 두 경우 모두 최대 타일 성적이 1024인 만큼 각각의 점유율을 통해 성능을 비교할 수

있는데, PER를 이용했을 때의 점유율이 그렇지 않았을 때의 점유율보다 약 7배 더 컸다.

2. Replay Buffer Prioritized Experience Replay(PER)

PER는 무작위로 경험을 샘플링하는 방식과 다르게 경험의 중요도에 따라 선택하는 가치 기반 방식이다. 모델은 중요도를 TD-error(Temporal-Difference Error) 값을 통해 인식하고, 이 값이 클수록 좀 더 집중해서 학습 효율을 높인다.

[그림 9] 경험을 추출하는 과정



모델의 학습 결과인 경험은 가상의 이진 트리에 저장되고, 자식 노드의 우선순위 합이 부모 노드의 우선순위가 되는 sum tree에 저장되어 우선순위가 클수록 루트 노드에 가까운 형태로 존재하게 된다. 각 경험 안에는 우선순위가 존재한다. 실험은 TD-error를 기댓값(target_q_values)에서 실제값(current_q_values)의 차로 표현했으며, 이는 각각 보상과 최대 예상 Q 값의 합, 그리고 현재 Q 값을 의미한다. 따라서 중요한 경험일수록 손실을 증가시켜 우선순위(priority)를 높여야 하므로, TD-error의 절댓값을 사용하고 파라미터(alpha)를 곱해서 반영할 중요도를 결정해야 한다. 이 값들은 경험을 샘플링할 때 사용하기 위해 배열 형태의 중요도 가중치(is_weight)에 넣어지고 최종적으로 경험을 추출한다.

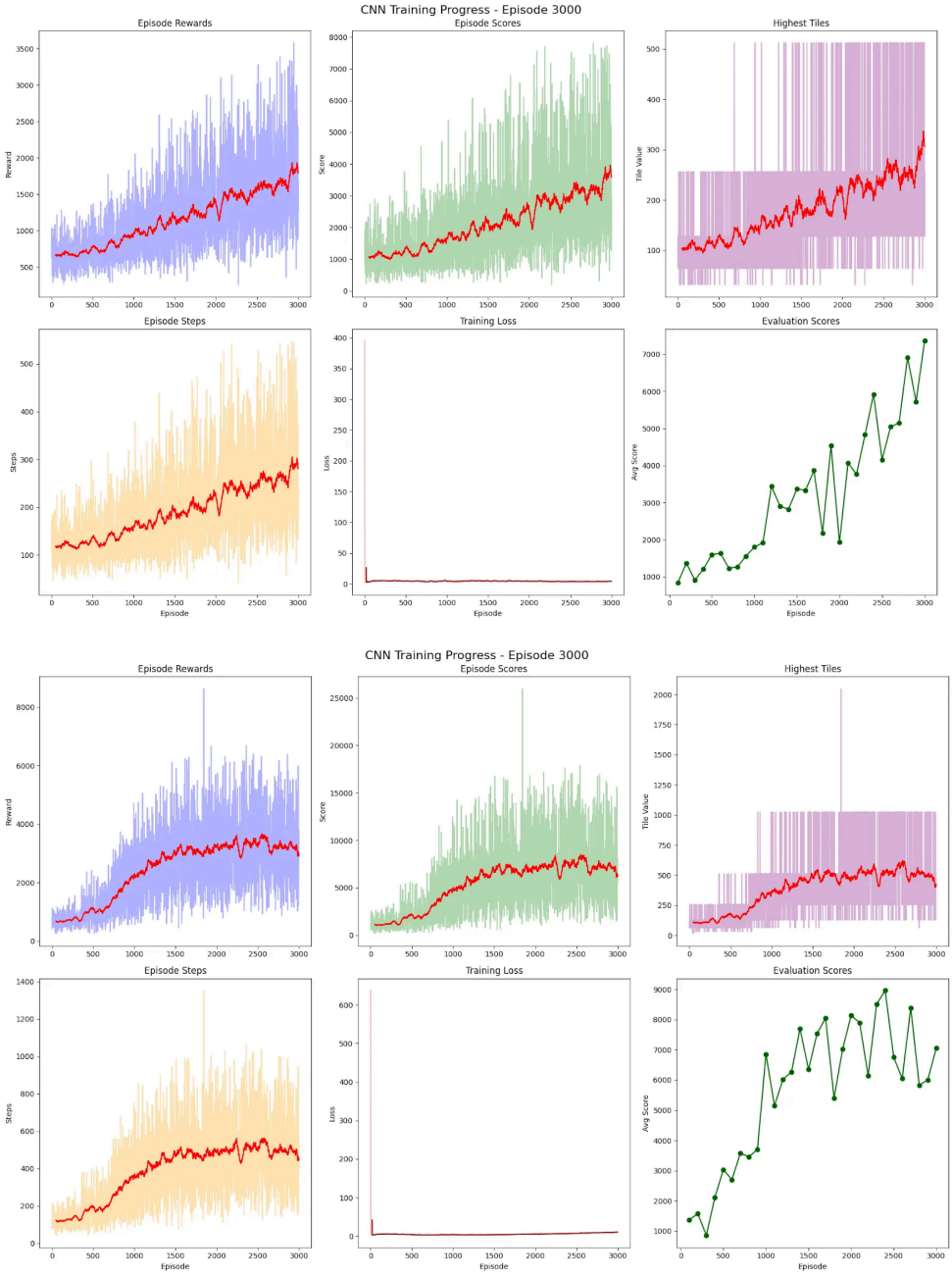
C. ε (탐욕 정책) 수렴 속도 (epsilon decay rate)

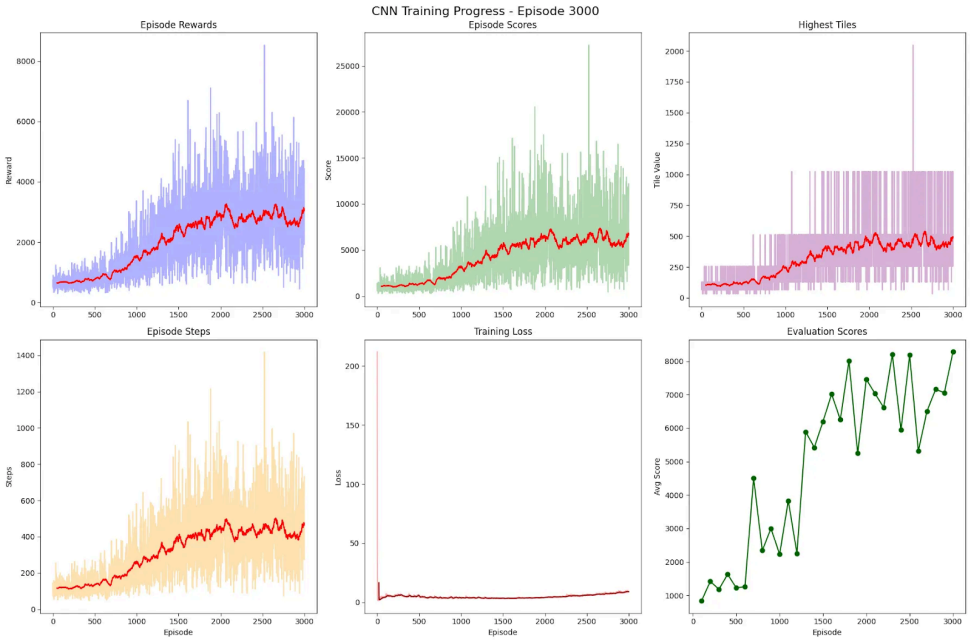
ε-탐욕 정책의 수렴 속도는 DQN 학습 과정에서 탐험과 활용의 균형을 결정하는 핵심 요인이다. 본 연구에서는 세 가지 설정을 비교하였다. 먼저, 기본 코드 설정으로 ε_start=0.9, ε_end=0.01, ε_decay=30000을 사용하였으며, 이를 기준 성능으로 삼았다. 추가로, 초기 탐험을 강화하기 위해 ε_start=1.0, ε_end=0.05, ε_decay=50000을 적용한 실험(A), 또한 빠른 수렴을 유도하기 위해 ε_start=0.8, ε_end=0.005, ε_decay=15000을 적용한 실험(B)을 수행하였다.

실험 결과, ε 수렴 속도가 지나치게 느린 경우(A)에는 학습 초반의 탐험이 과도하게

길어져 성능 향상 속도가 둔화되었다. 반대로, ε 수렴 속도를 급격히 낮춘 경우(B)에는 빠르게 결정론적 정책에 근접하면서 학습 안정성은 확보되었으나, 새로운 전략을 탐험할 기회가 줄어들어 장기적인 성능은 저하되었다. 이에 비해, 기본 설정(ε_start=0.9, ε_end=0.01, ε_decay=30000)은 적절한 탐험과 수렴 간 균형을 유지하여 가장 우수한 성능을 보였다.

[그림 10] (A), (B), 기본 코드의 학습 곡선





구체적으로, 위 그래프에서는 학습 곡선을 비교하였는데, 실험 A는 후반부에도 보상이 불안정하게 변동하는 반면, 실험 B는 비교적 일찍 평탄화되는 양상을 보였다. 기본 설정의 경우 학습 후반부까지 꾸준히 보상이 상승하며 안정적인 성능에 도달하였다.

[그림 11] (A), (B), 기본 코드의 평가 모드 검증 결과

```
최종 모델 성능 평가 시작...
Final Model Evaluation Started (100 games)
  Progress: 25/100 games completed
  Progress: 50/100 games completed
  Progress: 75/100 games completed
  Progress: 100/100 games completed

 Tile Achievement Rates:
  1024 tile:   7 games (  7.0%)
   512 tile:  63 games ( 63.0%)
   256 tile:  26 games ( 26.0%)
   128 tile:   4 games (  4.0%)
```

또한, 학습된 모델을 평가 모드(탐험적 행동 제외)에서 100회의 에피소드로 검증한 결과에서도 동일한 경향이 나타났다. 실험 A는 여전히 평균 보상이 낮게 형성되었고, 실험 B는 단기적으로는 안정적이지만 최고 성능은 기본 설정에 미치지 못하였다. 반면, 기본 설정은 높은 보상과 일관된 성능을 달성하여 최적의 ε-탐욕 정책 설정임을 확인할 수 있었다.

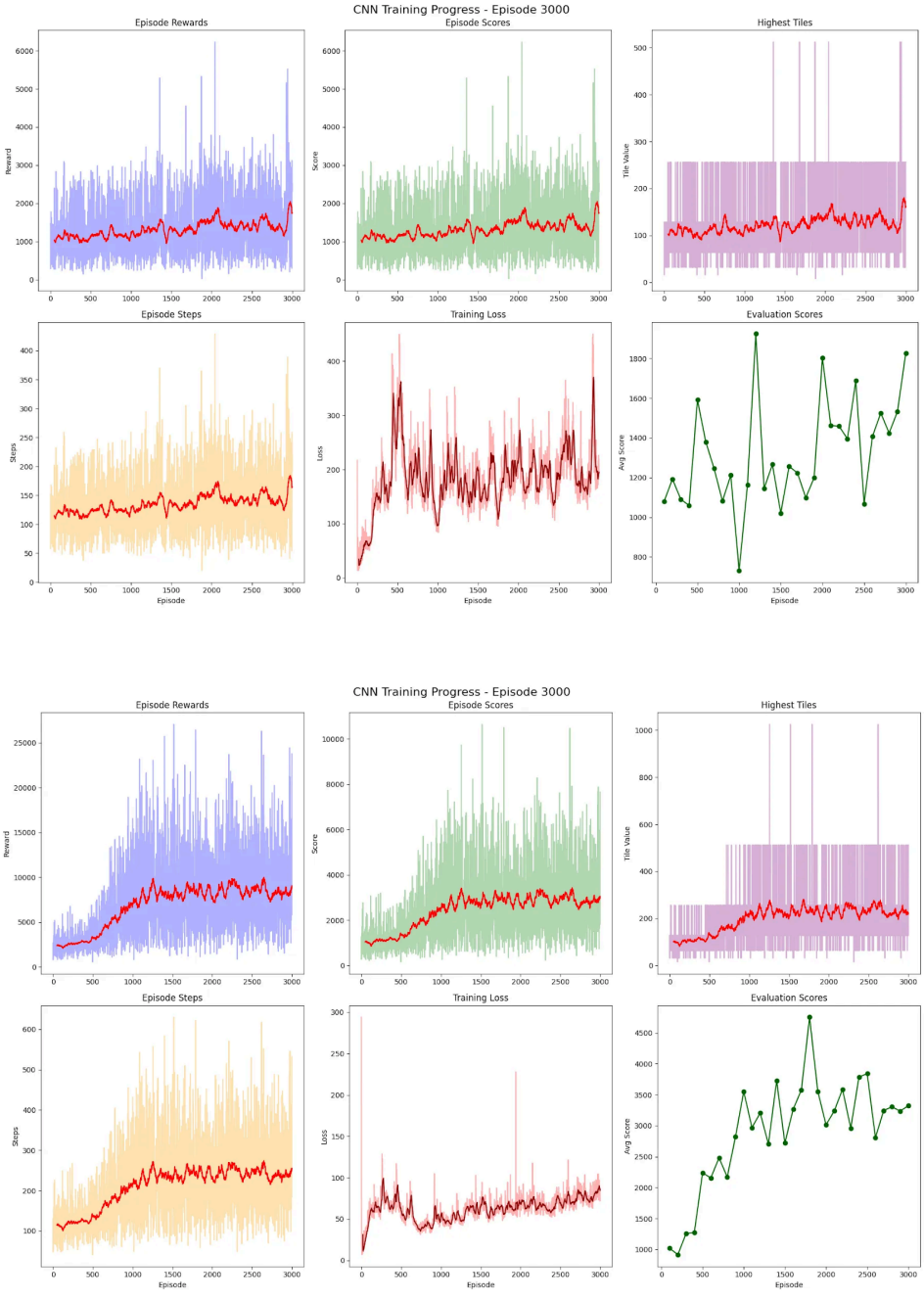
따라서 ε-탐욕 정책의 수렴 속도는 학습 성능에 직접적인 영향을 미치며, 본 연구에서는 기본 설정(ε_start=0.9, ε_end=0.01, ε_decay=30000)이 탐험과 활용의 균형 측면에서 가장 효과적임을 확인하였다.

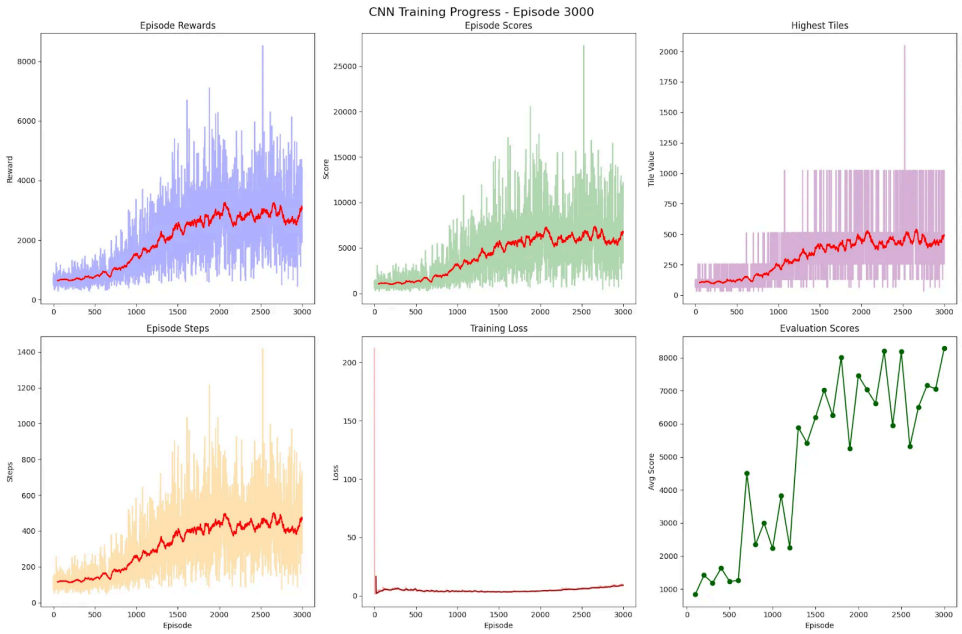
D. 보상 설계 (Reward Shaping)

DQN에서 보상 함수는 학습의 수렴 속도와 정책의 질에 직접적인 영향을 미치기

때문에, 보상 설계는 성능 개선을 위한 핵심 요소 중 하나로 꼽힌다. 본 연구에서는 2048 게임 환경에서 세 가지 단계적 보상 함수를 설계하여 그 효과를 비교하였다.

[그림 12] (1), (2), 기본 코드의 학습 곡선





첫 번째 단계에서는 가장 단순한 방식으로, 게임 내에서 발생한 누적 점수만을 보상으로 사용하는 방법을 적용하였다. 이 경우 에이전트는 점수와 직접적으로 연계된 학습을 수행하지만, 게임 진행 초기에 보상이 희소하게 주어지는 한계가 나타났다. 실제 실험 결과에서도 에피소드가 진행됨에 따라 학습이 불안정하게 수렴하고, 최종 성능 또한 낮게 관찰되었다.

두 번째 단계에서는 기존 누적 점수 기반 보상에 더해, 단조성(monotonicity)과 평탄성(smoothness) 보상을 추가하였다. 이는 타일 배치가 일정한 패턴을 유지하고, 보드 상태가 지나치게 불규칙하지 않도록 유도하는 목적을 가진다. 그러나 해당 방식은 보상 요소 간의 상충 효과(trade-off)로 인해 학습이 불안정하게 진행되었으며, 최종 성능 역시 뚜렷한 개선을 보이지 못했다. 이는 보드 구조적 특성이 점수와 직접적으로 연결되지 않는 경우가 많아, 학습 방향성이 왜곡되었기 때문으로 해석할 수 있다.

세 번째 단계에서는 단조성과 평탄성 보상을 제거하고, 병합 점수(merge score)와 빈 칸 수(empty cell)를 결합한 보상 함수를 설계하였다. 특히 로그 함수와 가중치를 함께 적용하여 보상이 과도하게 치우치지 않도록 조정하였다. 이러한 설계는 게임의 핵심 전략인 타일 병합과 공간 확보라는 두 가지 목표를 동시에 학습하게 함으로써, 안정적인 수렴을 유도하였다. 실제 실험 결과에서도 학습 곡선이 가장 안정적으로 나타났으며, 최종 평가 지표(평균 점수, 최대 타일, 생존 스텝 수 등)에서도 가장 우수한 성능을 기록하였다.

[그림 13] (1), (2), 기본 코드의 평가 모드 검증 결과

또한, 학습된 모델을 평가 모드(탐험적 행동 제외)에서 100회의 에피소드로 검증한 결과, 위 그래프에서도 동일한 경향이 확인되었다. 첫번째 보상 구조는 여전히 낮은 평균

보상을 기록하였으며, 두번째 보상 구조는 단기적으로 안정적이었으나 최대 성능에서는 기본 설정에 미치지 못하였다. 반면, 기본 설정은 높은 평균 보상과 일관된 성능을 달성하여, 2048 환경에 최적화된 보상 설계임을 입증하였다.

Ⅴ. 결론 및 시사점

본 연구에서는 DQN을 2048 게임 환경에 적용하여 다양한 성능 개선 요인들을 실험적으로 분석하였다. 먼저, Double DQN과 Dueling DQN 구조를 적용한 결과, 일반 DQN에 비해 학습 과정의 과적합을 줄이고 행동 가치 추정의 안정성을 확보할 수 있음을 확인하였다. 또한, Prioritized Experience Replay(PER)를 활용함으로써 중요한 경험에 더 높은 학습 비중을 두어 학습 효율성을 향상시킬 수 있었다.

탐험 전략과 관련하여, ε-탐욕 정책의 수렴 속도(ε decay rate)를 조절한 실험에서는 ε 감소가 지나치게 빠르면 학습 후반부의 탐험 기회가 부족해 성능이 저하되었으며, 반대로 너무 느릴 경우 불필요한 무작위 탐험이 지속되어 수렴 속도가 늦어지는 한계가 나타났다. 이로부터 기본 설정(ε_start=0.9, ε_end=0.01, ε_decay=30000)이 가장 안정적이고 우수한 성능을 달성하는 최적의 조합임을 확인하였다.

마지막으로, 보상 설계(Reward Shaping)의 차이를 분석한 결과, 단순 점수 기반 보상보다는 병합 점수와 빈 칸 수 보상을 함께 고려하는 방식이 학습 성능 향상에 긍정적인 영향을 주었다. 이는 에이전트가 단순히 고타일만을 추구하는 것이 아니라, 장기적인 게임 생존 가능성을 고려하는 전략을 학습하도록 유도했기 때문이다.

종합적으로, 본 연구는 2048 게임에서 강화학습 기반 DQN의 성능을 향상시키기 위해 구조적 개선, 메모리 활용, 탐험 전략, 보상 설계의 네 가지 측면이 중요한 역할을 한다는 점을 실험적으로 입증하였다. 이는 단순히 게임 환경을 넘어, 다양한 상태-행동 공간을 갖는 복잡한 문제에서도 DQN 성능 개선의 실질적 가이드라인을 제공할 수 있을 것이다.

다만, 본 연구는 단일 퍼즐 환경(2048)에 한정되어 실험이 진행되었으며, 연산 자원의 제약으로 인해 충분히 긴 학습 및 대규모 실험을 수행하지 못한 한계가 있다. 향후 연구에서는 보다 다양한 게임 환경과 실제 응용 문제에 DQN 및 성능 개선 요인을 확장 적용하고, 다른 최신 강화학습 알고리즘(예: Rainbow DQN, PPO, A3C 등)과의 비교 연구를 통해 보다 보편적인 성능 개선 전략을 도출할 필요가 있다.