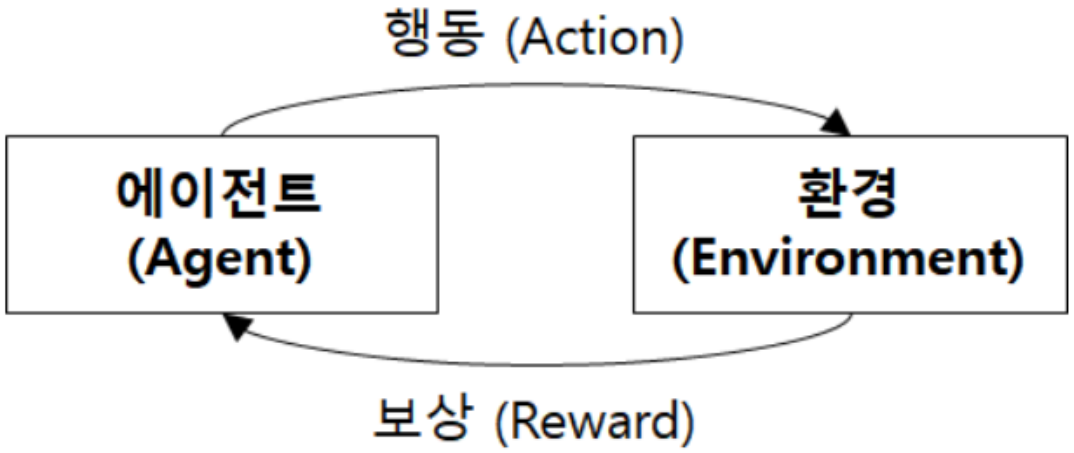
# RL

REINFORCEMENT LEARNING

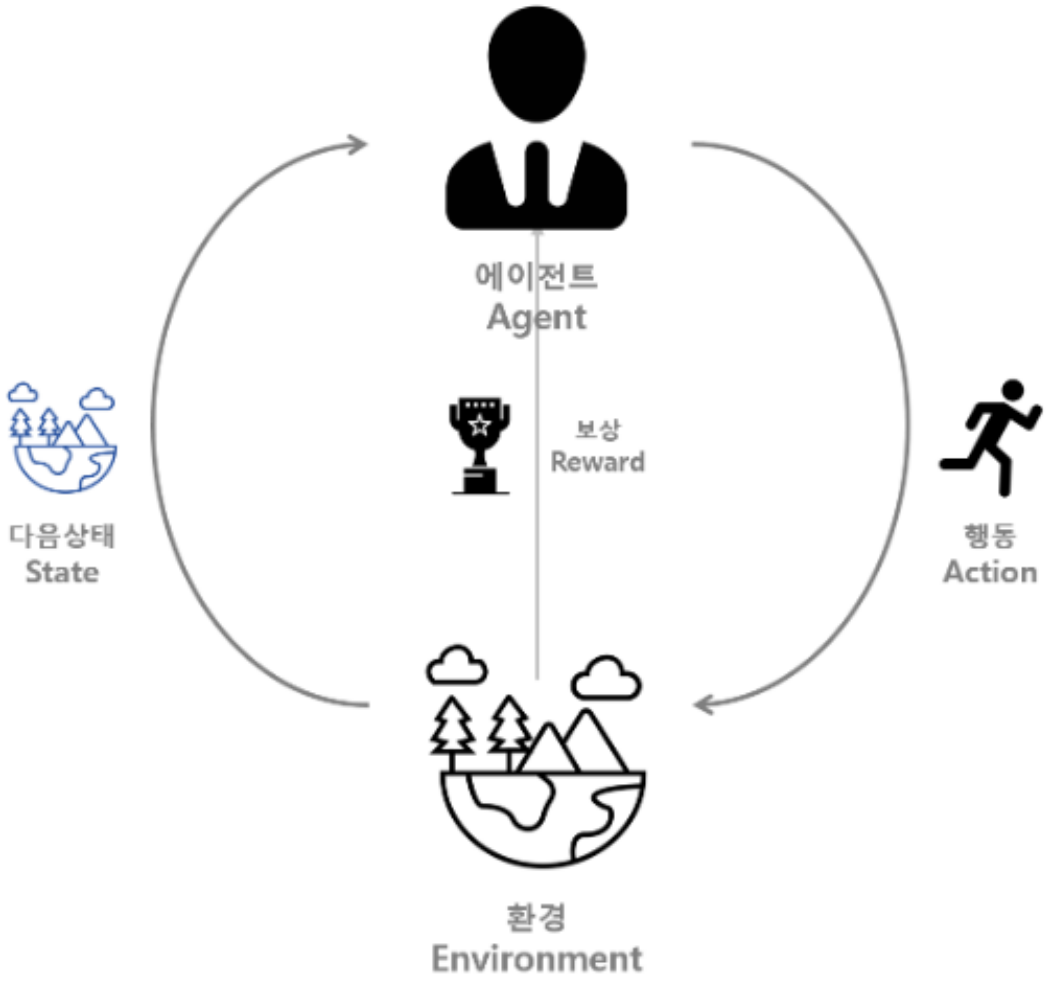
# RL이란?




❖ 기계 학습의 한 분야로, 에이전트가 환경과 상호작용을 하며 최적의 행동을 학습하는 방법을 연구하는 기술

❖ 에이전트(agent)가 하는 일

: Observation and action




# RL이란?

❖ 에이전트의 목적

: 보상의 장기간 기대치를 최대로 만드는 행동을 학습

: Agent 가 한 행동에 대해, 양 또는 음의 보상으로 피드백을 받음으로써 강화학습이 진행

# 정책(policy)

❖ 에이전트가 행동을 결정하기 위해 사용하는 알고리즘

❖ 확정적 정책

: 상태 s가 주어졌을 때, 특정 행동 a를 항상 선택하는 정책

$$\pi(s) = a$$

❖ 확률적 정책

: 상태 s가 주어졌을 때, 각 행동 a를 선택할 확률을 정의하는 정책

$\pi(a|s)$ = 행동 $a$를 상태 $s$에서 선택할 확률

# 정책 탐색

❖ 에이전트가 주어진 환경에서 최적의 정책을 찾기 위해 정책의 파라미터를 조정하거나 정책 구조를 탐색하는 과정

-> 장기적인 보상의 최대화

❖ 접근 방법

: 정책 기반 방법 & 가치 기반 방법

# 정책 탐색

❖ 정책 기반 방법

: 직접적으로 정책의 파라미터를 조정하여 최적의 정책을 찾는 방법

: REINFORCE Algorithm, Proximal Policy Optimization(PPO)

❖ 가치 기반 방법

: 상태-행동 쌍의 가치를 평가하여 최적의 행동을 선택하는 방법

: Q-Learning, Deep Q-Network(DQN)

# Environment

❖ Agent가 상호작용하는 외부시스템으로, 에이전트의 행동에 반응하여 상태와 보상을 제공하는 시스템을 의미

: 에이전트는 이와 상호작용하여 상태를 관찰하고, 행동을 취하며, 보상을 받아 정책을 학습

❖ 구성 요소

: 상태집합, 행동집합, 상태전이확률, 보상함수, 할인율

# Environment

❖ 상태집합

: 환경에서 가능한 모든 상태들의 집합

: ex) 체스판의 모든 가능한 배치

❖ 행동집합

: 에이전트가 임의의 상태에서 선택할 수 있는 모든 행동들의 집합

: ex) 체스에서 가능한 모든 수

# Environment

❖ 상태 전이 확률

: 상태 s에서 행동 a를 취했을 때, 다음 상태 s'로의 확률

$$P(s'|s,a)$$

❖ 보상 함수

: 상태 s와 행동 a의 조합에서 받는 보상

$$R(s,a)$$

# Environment

❖ Return

: reward의 합을 의미

1. Discounted Return

: 미래의 보상을 현재의 가치로 반환하기 위해 할인율을 사용하는 방법

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots$$

2. Undiscounted Return

: 미래의 모든 보상을 할인율을 사용하지 않고 합산하는 방법

$$G_t = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \cdots$$

# Environment

❖ 할인율

: 미래 보상의 중요성을 결정하는 파라미터

: 1에 가까울수록 먼 미래의 보상이 현재의 보상만큼 중요

: 0에 가까울수록 미래의 보상은 중요하게 취급되지 않음

$$0 \leq \gamma \leq 1$$

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots$$

# Q-learning

❖ Q-learning 이란?

: 가치 기반 방법으로, 상태-행동 쌍의 가치를 학습하여 최적의 정책을 찾는 알고리즘

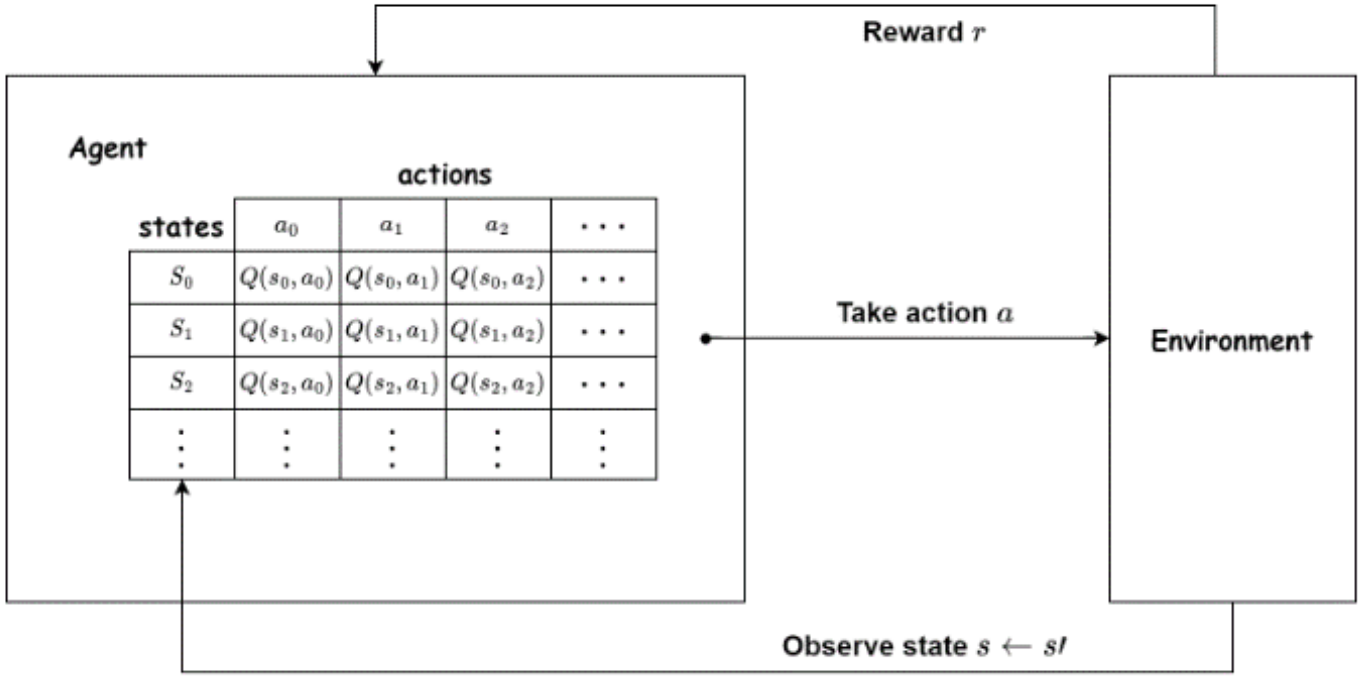
❖ The Bellman Equation

: 최종적으로 받는 모든 보상의 총합을 계산하는 방정식

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left( \underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

-> 이 방정식을 통해, Q-table 업데이트

# Q-Table




각 상태에서 행동에 대한 최대로 예상되는 미래 보상을 계산해놓는 lookup 테이블의 일종

# Q-learning

**Game Board:**

Current state (*s*): 0 0 0 / 0 1 0

**Q Table:** γ = 0.95

| | 0 0 0 / 1 0 0 | 0 0 0 / 0 1 0 | 0 0 0 / 0 0 1 | 1 0 0 / 0 0 0 | 0 1 0 / 0 0 0 | 0 0 1 / 0 0 0 |
|---|---|---|---|---|---|---|
| ↑ | 0.2 | 0.3 | 1.0 | -0.22 | -0.3 | 0.0 |
| ↓ | -0.5 | -0.4 | -0.2 | -0.04 | -0.02 | 0.0 |
| → | 0.21 | 0.4 | -0.3 | 0.5 | 1.0 | 0.0 |
| ← | -0.6 | -0.1 | -0.1 | -0.31 | -0.01 | 0.0 |

# Q-learning

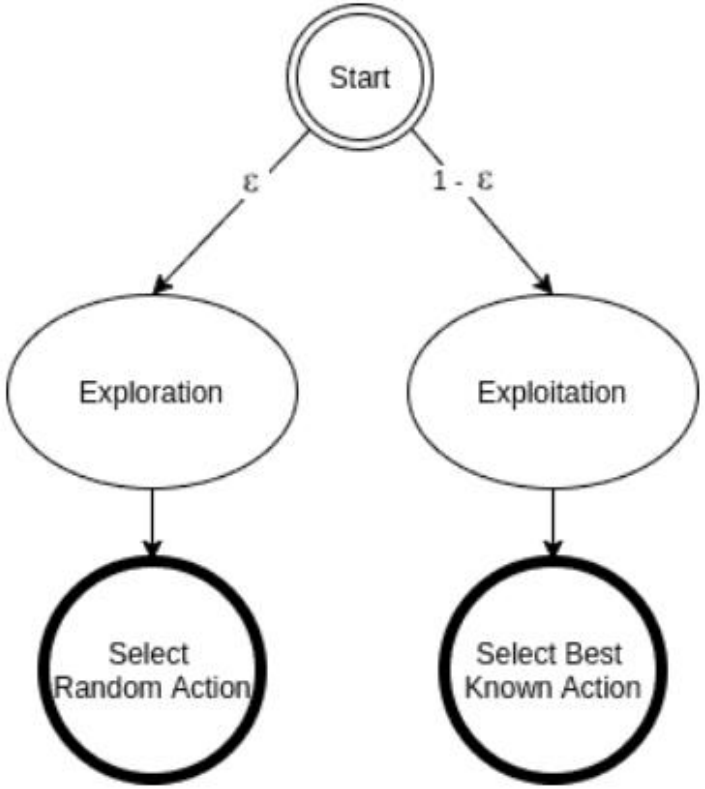
❖ Agent 의 행동 방식

: 활용과 탐험을 적절히 번갈아 가면서 행동

-> 점진적으로 활용의 비율을 늘림

❖ epsilon-greedy 전략

: 미지의 보상을 탐험하기 위한 전략

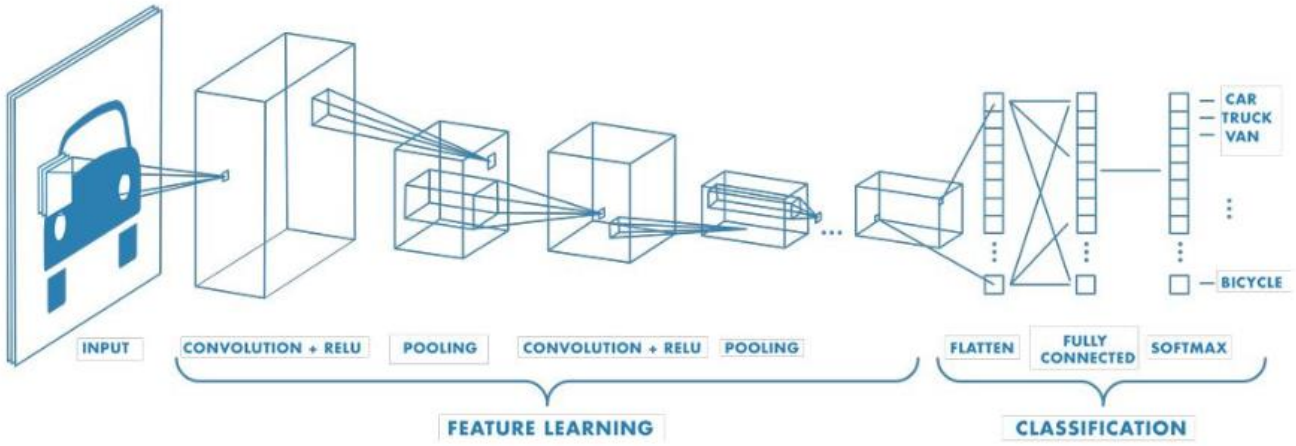


# CNN

❖ CNN (convolutional neural networks) : 합성곱 신경망

➔ 임의의 환경에서의 상태가 고차원 공간을 가질 때, 중요한 특성만을 추출하여 이를 기반으로 강화 학습을 수행하기 위한 도구

➔ input layer, convolutional layer, pooling layer, fully-connected layer 로 구성



# CNN

❖ 구조 : 입력값의 특징을 추출하는 부분 + 클래스를 분류하는 부분

1. 특징 추출 영역 : Convolutional layer + Pooling layer
2. 분류 영역 : Fully connected layer

A. Convolutional layer

: 입력값에서 특징을 추출하는 역할

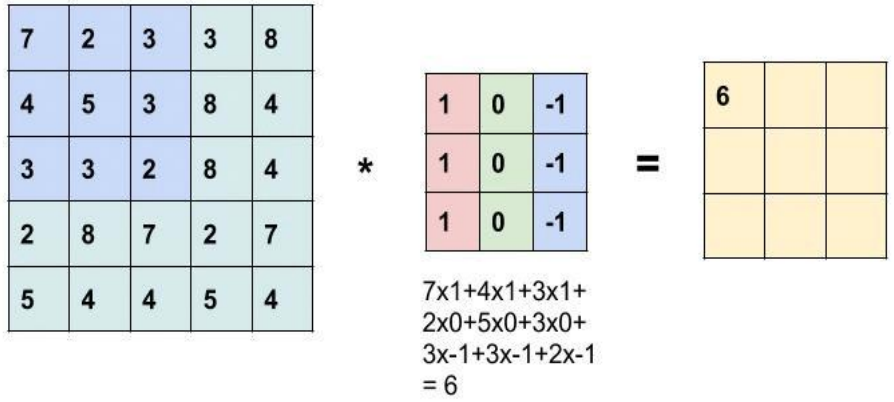
: 필터 (kernel) 라 불리는 작은 행렬이 입력값에 slidding -> 합성곱 연산 수행

: 필터가 입력값 전체를 훑으면서 특징 맵(feature map) 생성 -> 필터와 입력값의 내적값

# CNN

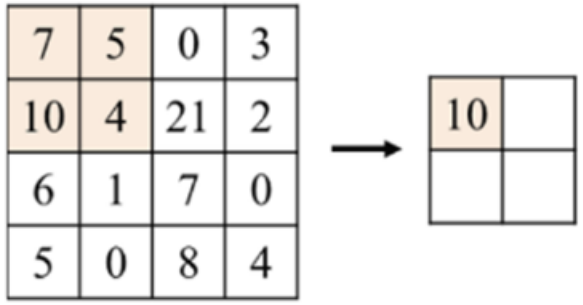
Ex) 입력값 : 5 by 5 행렬, 필터 : 3 by 3 행렬

, stride : 1 (필터의 이동량)




B. Pooling Layer : 특징 맵의 크기를 줄이고, 특정 feature 를 강조하는 역할

-> Max Pooling, Average Pooling 등이 있음

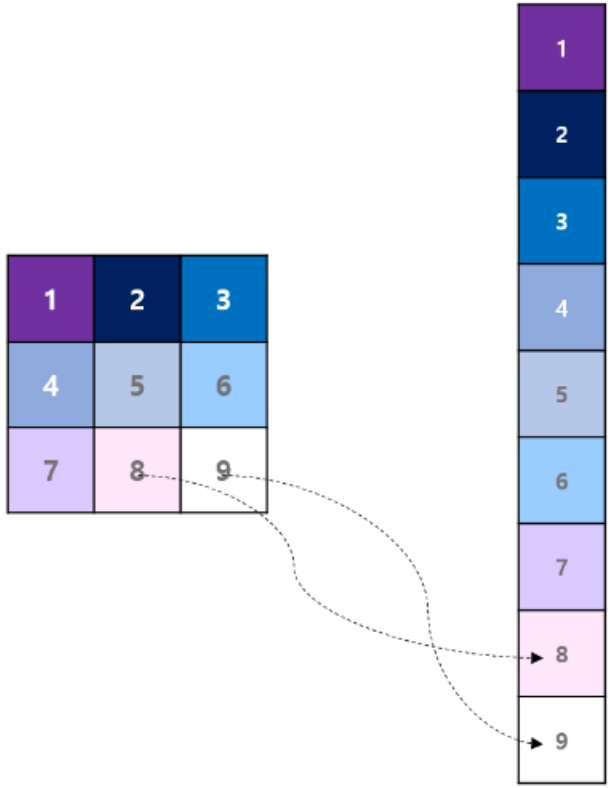


# CNN

C. Fully Connected Layer (FC Layer) :특징 맵을 1차원 벡터로 펼친 후 (Flattening), 이를 완전 연결층의 입력값으로 사용

-> 모든 노드가 연결돼 있으며, 최종적으로 각 클래스에 속할 확률을 예측

-> 출력층에서는 보통 softmax 함수를 사용해 각 클래스에 속할 확률을 계산



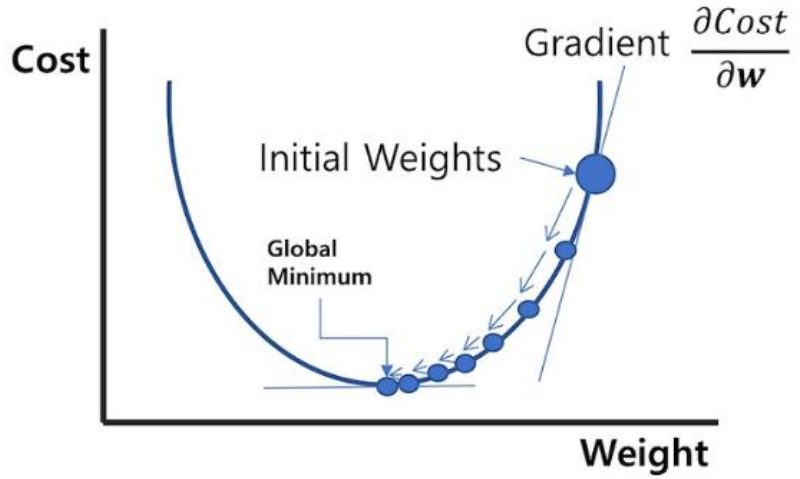
# CNN

❖ backpropagation (역전파)

: Feed-Forward 과정을 통해 손실함수 값 계산 -> back-propagation 과정을 통해 손실함수에 영향을 미친 각 노드의 가중치를 업데이트 -> 손실함수 최소화

: 기본 개념 : 손실함수 값을 각 가중치에 대해 미분한 후, 경사 하강법을 활용해 손실함수 최소화

$$w_{new} = w_{old} - \eta \cdot \frac{\partial L}{\partial w}$$



# CNN

❖ CNN 을 강화학습에 사용할 때의 이점

1. 공간적 특성 추출

: CNN 은 공간적 정보를 잘 캡처해 다양한 입력에 대한 일반화 성능이 뛰어남

-> 강화학습에서 다루는 환경의 상태가 다양할 때 매우 유용

2. 효율적 학습

: 합성곱 계층은 필터를 사용해 입력 데이터의 모든 부분에서 동일한 가중치를 공유

-> 모델의 파라미터 수를 줄이고, 궁극적으로 학습을 더 효율적으로 만듦

# Clifford Circuit Synthesis

- 아이디어 : 임의의 Clifford Circuit에 유한한 Clifford gate를 가해 회로의 Boolean matrix가 Identity matrix로 만들어짐

-> Clifford gate들을 역 gate(Hermitian)로 바꾼 후 역순으로 정렬하여 임의의 회로를 재구성 할 수 있음

$$O_0\, g_0 g_1 \cdots g_n = I \quad \Rightarrow \quad O_0 = g_n^{\dagger} g_{n-1}^{\dagger} \cdots g_0^{\dagger}$$

# Clifford Circuit Synthesis

- reward

1. Big positive reward : identity matrix 에 가까워지는 action 을 취했을 때 큰 양의 보상

2. Small penalty : 사용되는 gate 의 개수와 증가되는 회로의 depth 에 따라 작은 음의 보상

# Clifford Circuit Synthesis

- 손실 함수

$$\text{Loss} = -\mathbb{E}\left[R \cdot \log(\pi(a|s))\right]$$

$$\theta \leftarrow \theta - \alpha \cdot \nabla_\theta \text{Loss}$$

1. 높은 보상을 받는 행동에 대해서는 해당 행동을 선택할 확률이 증가됨
2. 낮은 보상을 받는 행동에 대해서는 해당 행동을 선택할 확률이 감소됨

# Ex

- 예시 : action 1,2,3이 [10, 30, 5] 의 보상값을 가진다고 가정

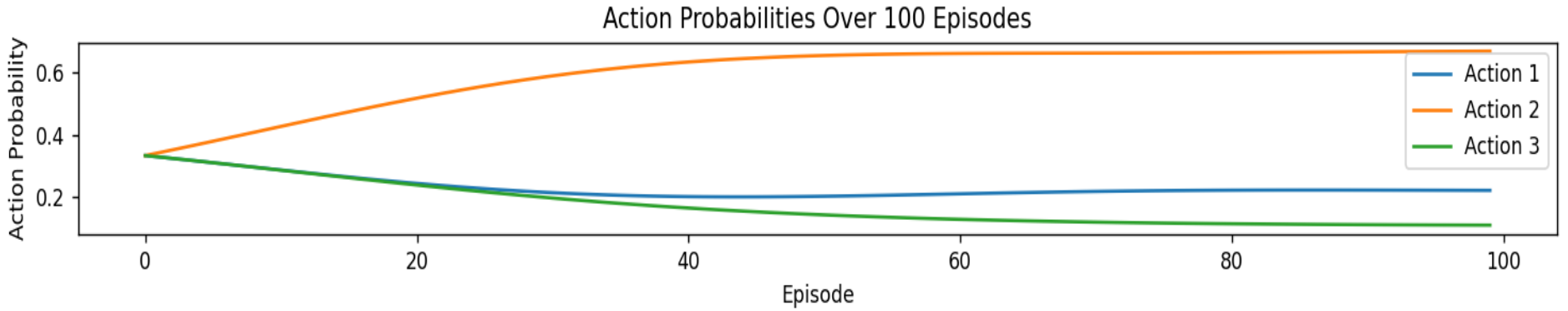
-> episode 100번 진행

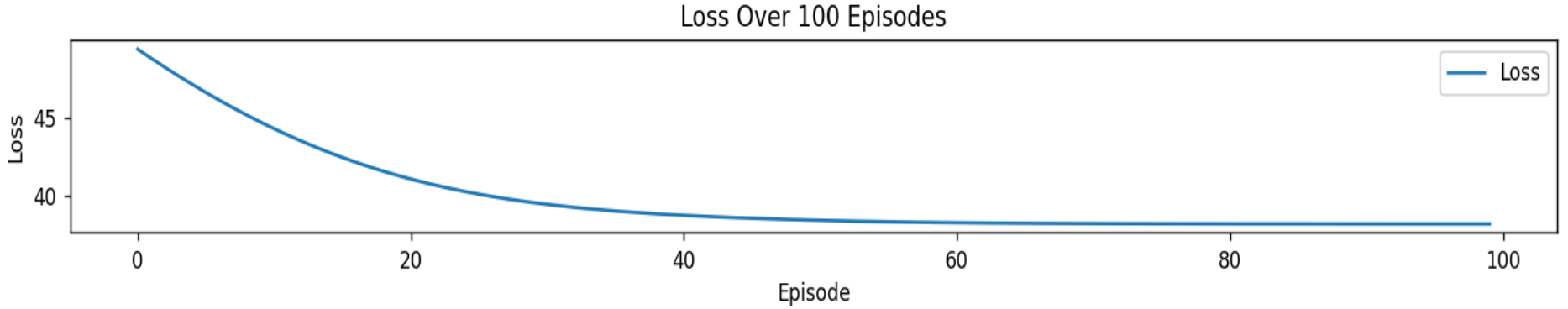
-> 초기 확률 동일

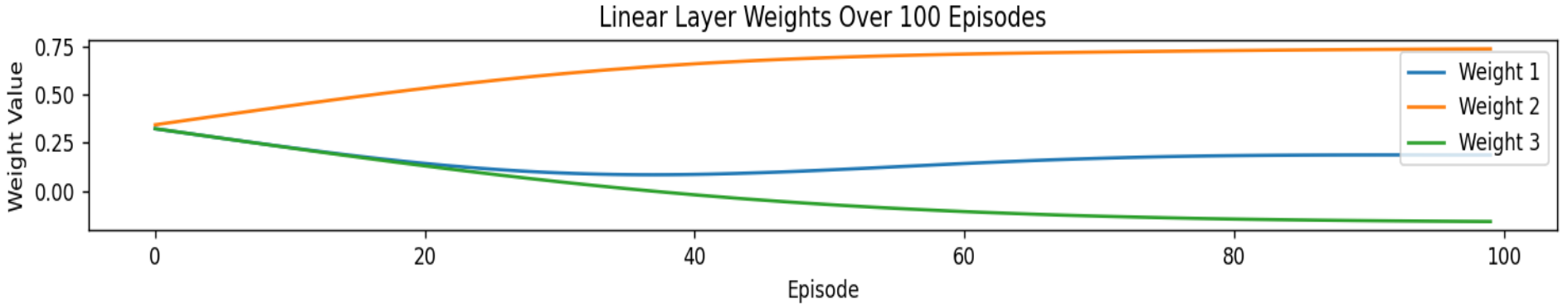
-> 입력 상태 동일

- 입력 노드 1개

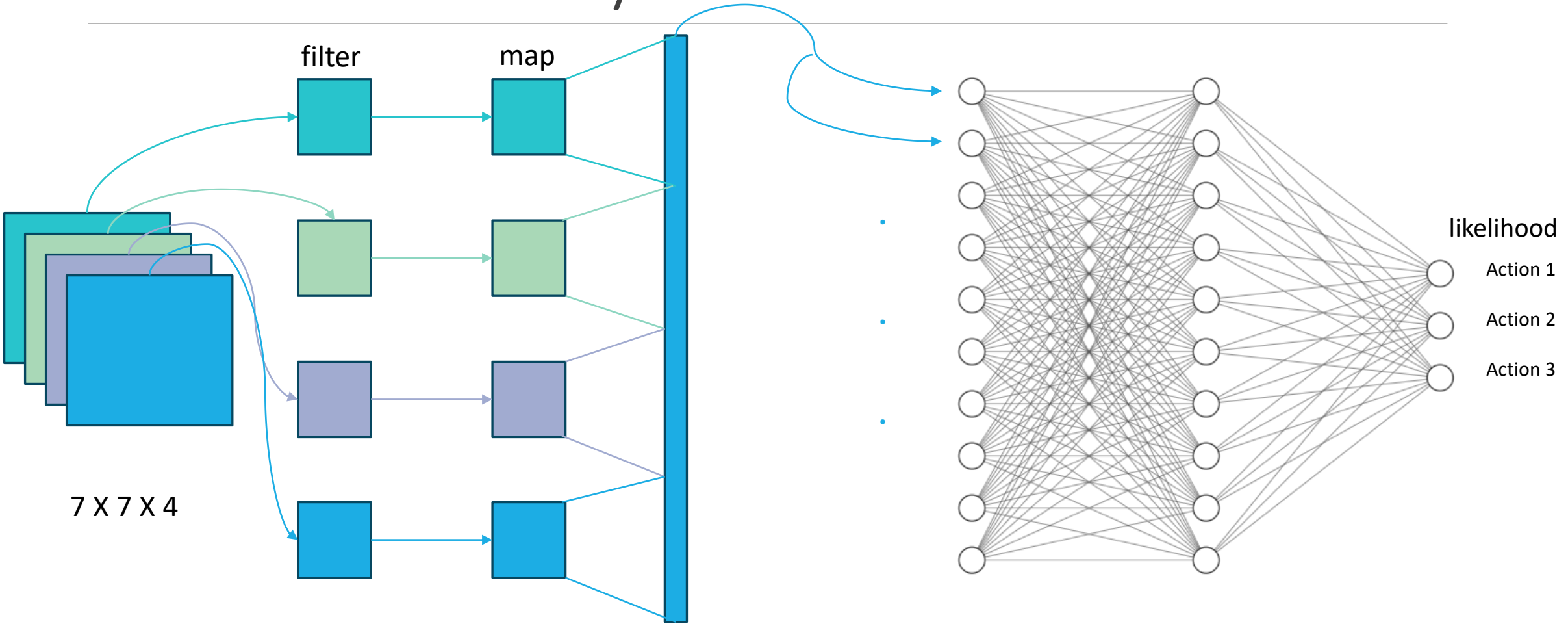
+ 출력 노드 3개



# Clifford Circuit Synthesis



# Thank you

2024.08.19