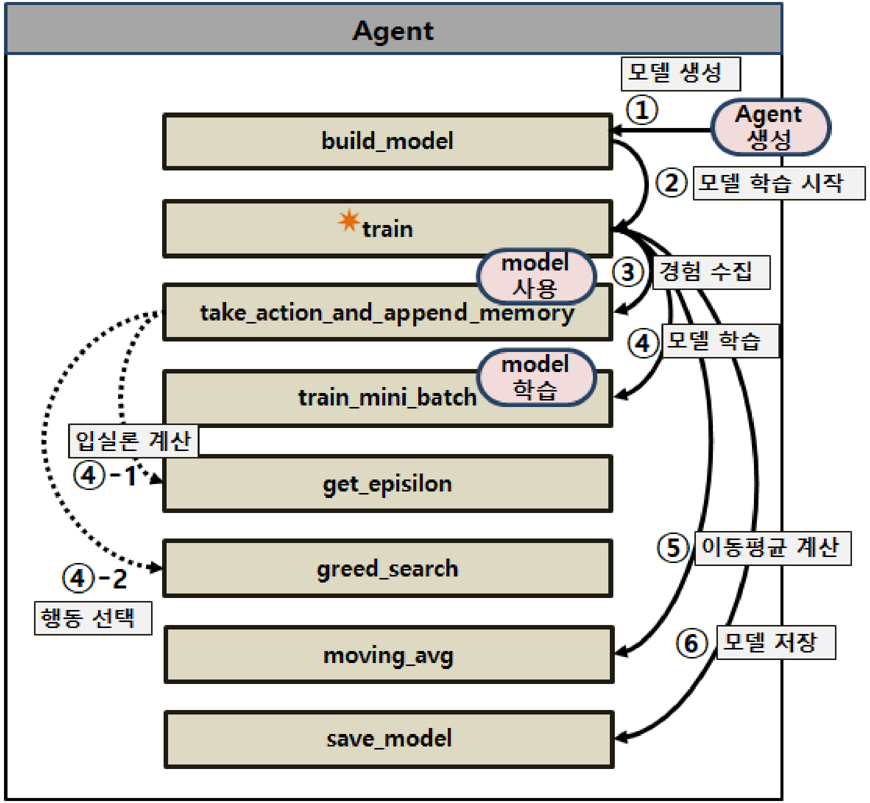
06

DQN 알고리즘

2. 프로그래밍

DQN 프로그래밍

프로그램 구조



DQN 프로그래밍

전체 코드 리뷰

코드 리뷰

DQN 프로그래밍 코드분석

Agent 클래스 속성

(1) 프로그램 동작 설정

(2) 모델 설정

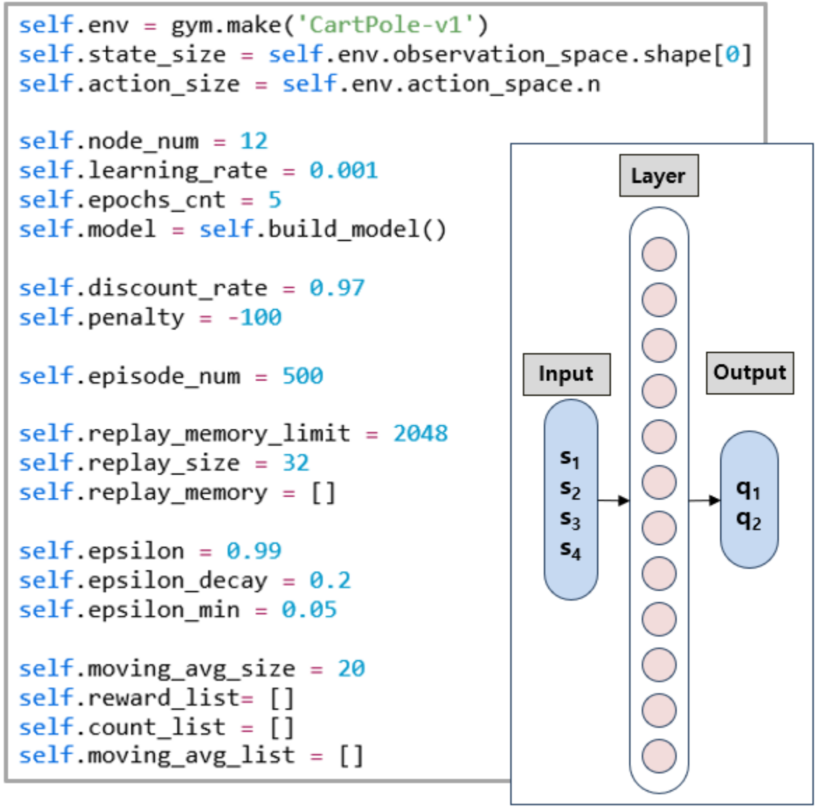
(3) 학습 설정

(4) 반복 설정

(5) 데이터 수집 환경

(6) 탐험 환경 설정

(7) 학습 모니터링 설정



```
self.env = gym.make('CartPole-v1')
self.state_size = self.env.observation_space.shape[0]
self.action_size = self.env.action_space.n

self.node_num = 12
self.learning_rate = 0.001
self.epochs_cnt = 5
self.model = self.build_model()

self.discount_rate = 0.97
self.penalty = -100

self.episode_num = 500

self.replay_memory_limit = 2048
self.replay_size = 32
self.replay_memory = []

self.epsilon = 0.99
self.epsilon_decay = 0.2
self.epsilon_min = 0.05

self.moving_avg_size = 20
self.reward_list= []
self.count_list = []
self.moving_avg_list = []
```

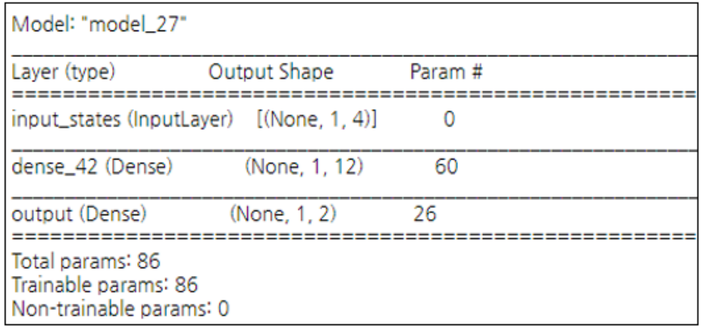
DQN 프로그래밍 코드분석

build_model 함수

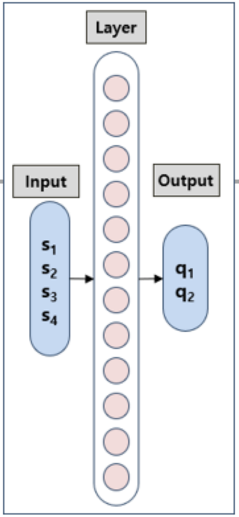
(1) Input
(2) Layers
(3) Output
(4) 모델구성
(5) 환경설정

(6) 모델정보

```
def build_model(self):
        input_states = Input(shape=(1,self.state_size), name='input_states')
        x = (input_states)
        x = Dense(self.node_num, activation='relu')(x)
        out_actions = Dense(self.action_size, activation='linear', name='output')(x)
        model = tf.keras.models.Model(inputs=[input_states], outputs=[out_actions])
        model.compile(optimizer=Adam(lr=self.learning_rate),
                      loss='mean_squared_error'
                     )
        model.summary()
        return model
```

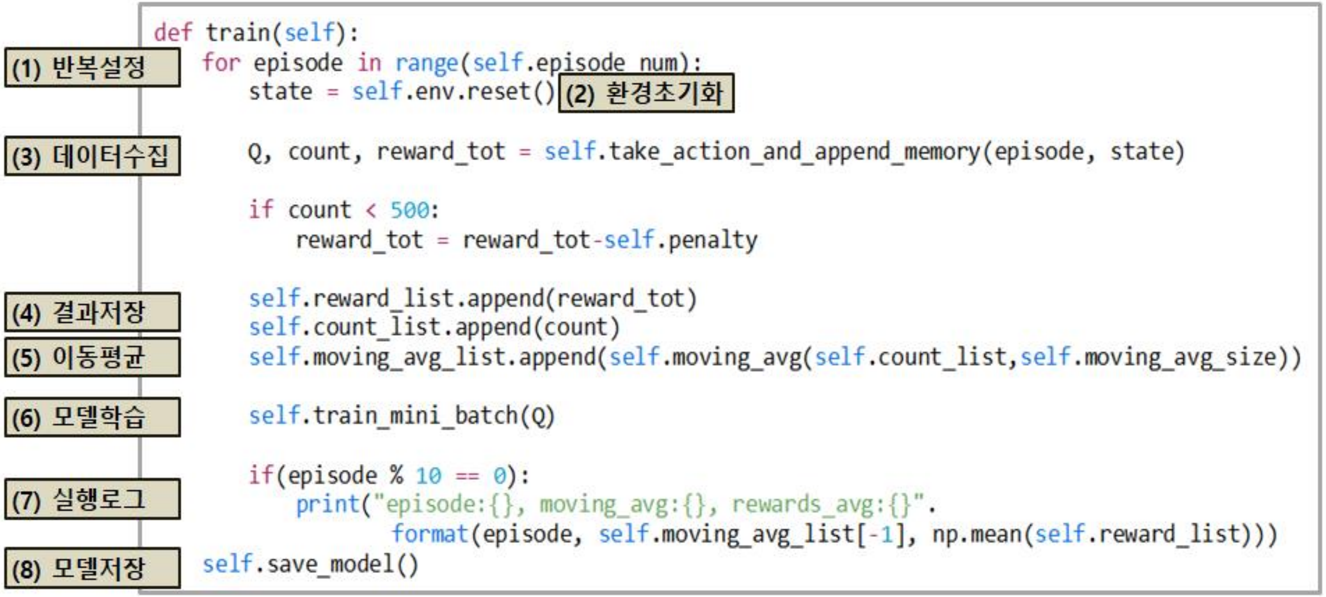


```
Model: "model_27"
_________________________________________________________________
Layer (type)              Output Shape            Param #
=================================================================
input_states (InputLayer)  [(None, 1, 4)]          0
_________________________________________________________________
dense_42 (Dense)           (None, 1, 12)           60
_________________________________________________________________
output (Dense)            (None, 1, 2)            26
=================================================================
Total params: 86
Trainable params: 86
Non-trainable params: 0
```



DQN 프로그래밍 코드분석

train 함수



```
def train(self):
    for episode in range(self.episode_num):
        state = self.env.reset()

        Q, count, reward_tot = self.take_action_and_append_memory(episode, state)

        if count < 500:
            reward_tot = reward_tot-self.penalty

        self.reward_list.append(reward_tot)
        self.count_list.append(count)
        self.moving_avg_list.append(self.moving_avg(self.count_list,self.moving_avg_size))

        self.train_mini_batch(Q)

        if(episode % 10 == 0):
            print("episode:{}, moving_avg:{}, rewards_avg:{}".
                  format(episode, self.moving_avg_list[-1], np.mean(self.reward_list)))
    self.save_model()
```

DQN 프로그래밍 코드분석

take_action_and_append_memory 함수

(1) 입실론 계산

(2) 반복 설정

(3) 데이터 모양 변경

(4) 모델 사용 Q 예측

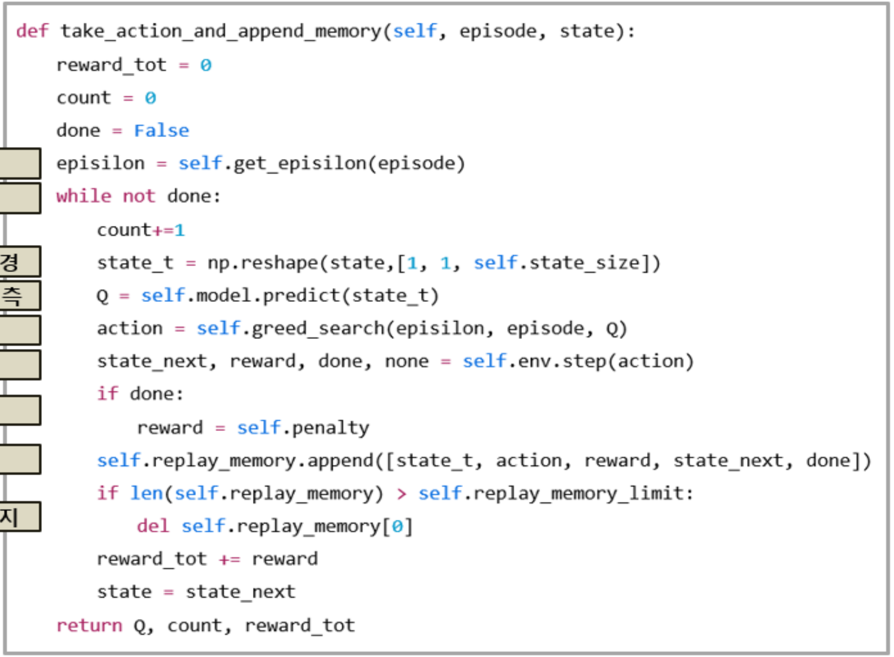
(5) 행동 선택

(6) 수레 이동

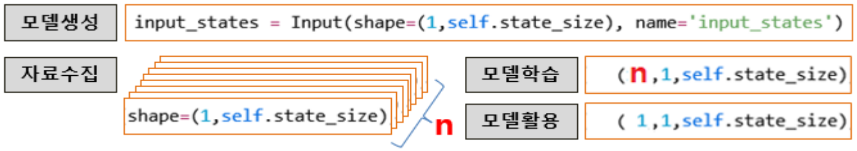
(7) 페널티 설정

(8) 실행기록 저장

(9) 메모리 크기 유지



```
def take_action_and_append_memory(self, episode, state):
    reward_tot = 0
    count = 0
    done = False
    episilon = self.get_episilon(episode)
    while not done:
        count+=1
        state_t = np.reshape(state,[1, 1, self.state_size])
        Q = self.model.predict(state_t)
        action = self.greed_search(episilon, episode, Q)
        state_next, reward, done, none = self.env.step(action)
        if done:
            reward = self.penalty
        self.replay_memory.append([state_t, action, reward, state_next, done])
        if len(self.replay_memory) > self.replay_memory_limit:
            del self.replay_memory[0]
        reward_tot += reward
        state = state_next
    return Q, count, reward_tot
```



모델생성 input_states = Input(shape=(1,self.state_size), name='input_states')

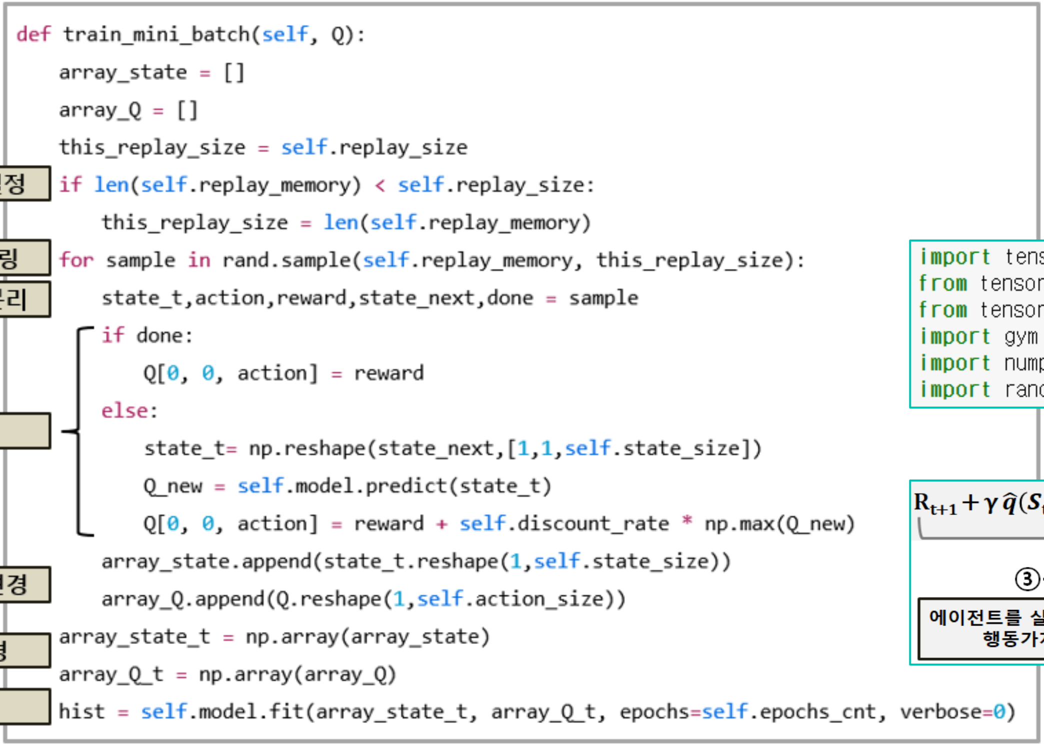
자료수집 shape=(1,self.state_size) n

모델학습 (n,1,self.state_size)

모델활용 (1,1,self.state_size)

DQN 프로그래밍 코드분석

train_mini_batch 함수



```
def train_mini_batch(self, Q):
    array_state = []
    array_Q = []
    this_replay_size = self.replay_size
    if len(self.replay_memory) < self.replay_size:
        this_replay_size = len(self.replay_memory)
    for sample in rand.sample(self.replay_memory, this_replay_size):
        state_t,action,reward,state_next,done = sample
        if done:
            Q[0, 0, action] = reward
        else:
            state_t= np.reshape(state_next,[1,1,self.state_size])
            Q_new = self.model.predict(state_t)
            Q[0, 0, action] = reward + self.discount_rate * np.max(Q_new)
        array_state.append(state_t.reshape(1,self.state_size))
        array_Q.append(Q.reshape(1,self.action_size))
    array_state_t = np.array(array_state)
    array_Q_t = np.array(array_Q)
    hist = self.model.fit(array_state_t, array_Q_t, epochs=self.epochs_cnt, verbose=0)
```

(1) replay 크기 설정

(2) Random 샘플링

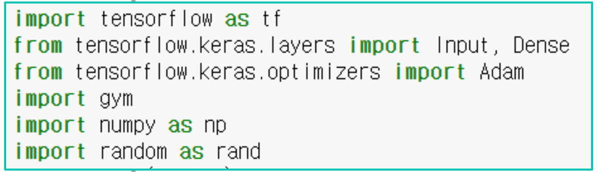
(3) 학습 데이터 분리

(4) Q값 계산

(5) 데이터 모양 변경

(6) Numpy로 변경

(7) 모델 학습



```
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.optimizers import Adam
import gym
import numpy as np
import random as rand
```

$$\underbrace{R_{t+1}+\gamma\,\hat{q}(S_{t+1},\ A_{t+1},\ w)}_{③\text{-}1}-\underbrace{\hat{q}(S_t,\ A_t,\ w)}_{③\text{-}2}$$

③-1 에이전트를 실행해서 얻은 행동가치함수

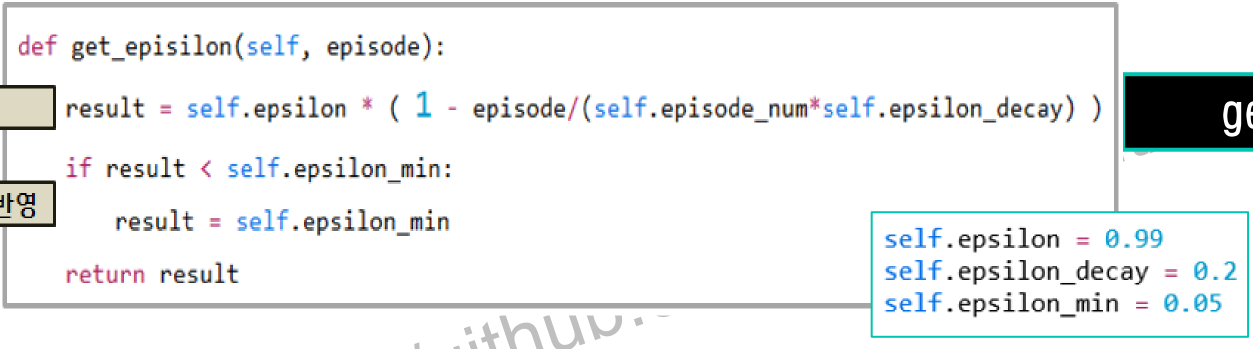
③-2 인공신경망에서 예측한 행동가치함수

DQN 프로그래밍 코드분석

get_episilon 함수

(1) 입실론 계산

(2) 최소 입실론 값 반영



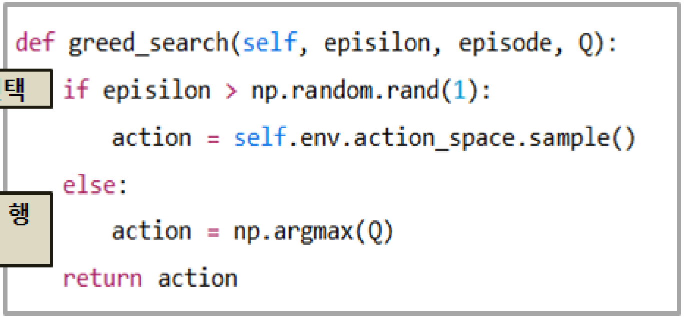
```
def get_episilon(self, episode):
    result = self.epsilon * ( 1 - episode/(self.episode_num*self.epsilon_decay) )
    if result < self.epsilon_min:
        result = self.epsilon_min
    return result
```

```
self.epsilon = 0.99
self.epsilon_decay = 0.2
self.epsilon_min = 0.05
```

greed_search 함수

(1) 랜덤하게 행동 선택

(2) Q 값을 기준으로 행동 선택



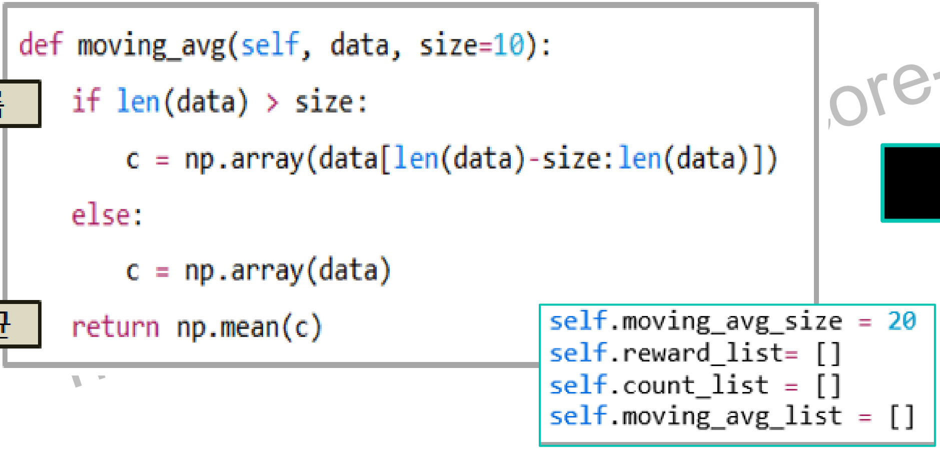
```
def greed_search(self, episilon, episode, Q):
    if episilon > np.random.rand(1):
        action = self.env.action_space.sample()
    else:
        action = np.argmax(Q)
    return action
```

DQN 프로그래밍 코드분석

get_episilon 함수

(1) size 크기만큼 자름

(2) 잘라진 데이터 평균

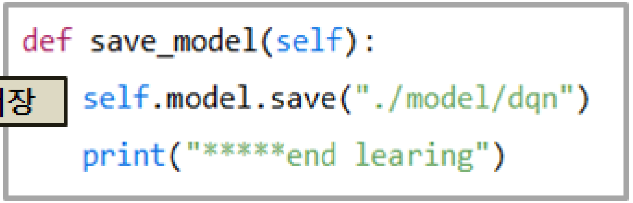


```
def moving_avg(self, data, size=10):
    if len(data) > size:
        c = np.array(data[len(data)-size:len(data)])
    else:
        c = np.array(data)
    return np.mean(c)
```

```
self.moving_avg_size = 20
self.reward_list= []
self.count_list = []
self.moving_avg_list = []
```

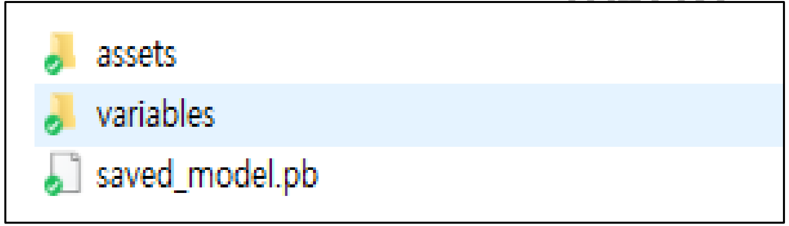
DQN 프로그래밍 코드분석

(1) 모델 저장



```
def save_model(self):
    self.model.save("./model/dqn")
    print("*****end learing")
```

save_model 함수



assets
variables
saved_model.pb