import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.optimizers import Adam

class DQNAgent:

def \_\_init\_\_(self, state\_size=5, action\_size=2):

self.state\_size = state\_size

self.action\_size = action\_size

self.gamma = 0.95 # Discount factor

self.epsilon = 1.0 # Exploration rate

self.epsilon\_min = 0.01

self.epsilon\_decay = 0.995

self.learning\_rate = 0.001

self.model = self.\_build\_model()

def \_build\_model(self):

model = Sequential()

model.add(Dense(24, input\_dim=self.state\_size, activation="relu"))

model.add(Dense(24, activation="relu"))

model.add(Dense(self.action\_size, activation="linear"))

model.compile(loss="mse", optimizer=Adam(learning\_rate=self.learning\_rate))

return model

def act(self, state):

if np.random.rand() <= self.epsilon:

return np.random.choice(self.action\_size)

q\_values = self.model.predict(np.array([state]), verbose=0)

return np.argmax(q\_values[0])

def train(self, state, action, reward, next\_state, done):

target = reward

if not done:

target = reward + self.gamma \* np.amax(self.model.predict(np.array([next\_state]), verbose=0)[0])

target\_f = self.model.predict(np.array([state]), verbose=0)

target\_f[0][action] = target

self.model.fit(np.array([state]), target\_f, epochs=1, verbose=0)

if self.epsilon > self.epsilon\_min:

self.epsilon \*= self.epsilon\_decay