**Laporan Praktikum Kontrol Cerdas**

**Minggu ke-5**

Nama : Amalia Dwi Nurahma  
NIM : 224308075  
Kelas : TKA 7D  
Akun Github (Tautan) : <https://github.com/amaliadwinurahma>  
Student Lab Assistant :

**1. Judul Percobaan**

Deep Reinforcement Learning untuk Kontrol Kompleks

**2. coding**

import gymnasium as gym

import numpy as np

import tensorflow as tf

from tensorflow import keras

from collections import deque

import random

# Inisialisasi environment

env = gym.make("CartPole-v1")

# Parameter DRL

state\_size = env.observation\_space.shape[0]

action\_size = env.action\_space.n

learning\_rate = 0.001

gamma = 0.95

epsilon = 1.0

epsilon\_min = 0.01

epsilon\_decay = 0.995

batch\_size = 32

memory = deque(maxlen=2000)

# Membangun model Deep Q-Network (DQN)

model = keras.Sequential([

keras.Input(shape=(state\_size,)),

keras.layers.Dense(24, activation="relu"),

keras.layers.Dense(24, activation="relu"),

keras.layers.Dense(action\_size, activation="linear")

])

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

# Fungsi memilih aksi (eksplorasi vs eksploitasi)

def select\_action(state):

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

return np.random.choice(action\_size) # eksplorasi

q\_values = model.predict(state, verbose=0) # eksploitasi

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

# Proses training

for episode in range(1000):

state, \_ = env.reset()

state = np.array(state).reshape(1, state\_size)

for time in range(500):

# Pilih aksi

action = select\_action(state)

# Eksekusi aksi

next\_state, reward, terminated, truncated, \_ = env.step(action)

done = terminated or truncated

next\_state = np.array(next\_state).reshape(1, state\_size)

# Simpan ke memori

memory.append((state, action, reward, next\_state, done))

state = next\_state

print(f"Episode: {episode}, Score: {time}, Epsilon: {epsilon:.2f}")

if done:

break

# Training DQN dari pengalaman

if len(memory) > batch\_size:

minibatch = random.sample(memory, batch\_size)

for state\_mb, action\_mb, reward\_mb, next\_state\_mb, done\_mb in minibatch:

target = reward\_mb

if not done\_mb:

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

target\_f = model.predict(state\_mb, verbose=0)

target\_f[0][action\_mb] = target

model.fit(state\_mb, target\_f, epochs=1, verbose=0)

# Kurangi epsilon

if epsilon > epsilon\_min:

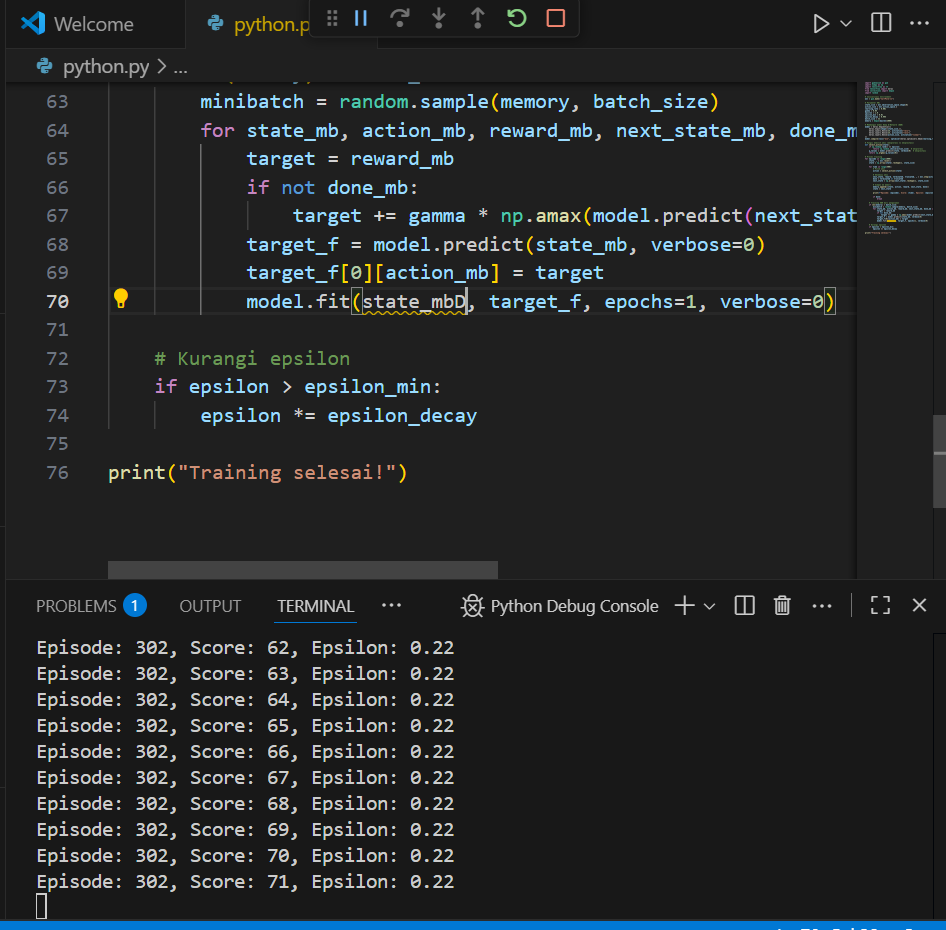
epsilon \*= epsilon\_decay

print("Training selesai!")

**3. Analisis**

Stabilitas pelatihan DQN sangat bergantung pada beberapa hal: penggunaan replay buffer, target network, ukuran batch, dan learning rate. Dalam implementasi ini, tidak ada target network sehingga nilai Q dapat berubah cepat dan tidak stabil. Implementasi per-sample model.fit di dalam loop dapat menjadi lambat; lebih efisien melakukan update dengan batch vectorized (mengumpulkan state\_mb dan target\_f ke array lalu melakukan satu kali model.fit per minibatch). Nilai epsilon awal = 1.0 dengan decay 0.995 merupakan strategi umum—memungkinkan eksplorasi luas pada awal pelatihan lalu perlahan beralih ke eksploitasi. Namun laju decay perlu disesuaikan berdasarkan observasi performa (contoh: jika agen belum mengeksplorasi cukup, perlambat decay). Gamma 0.95 membuat agen mempertimbangkan reward jangka menengah; untuk tugas CartPole yang memberikan reward setiap langkah, gamma antara 0.95–0.99 dapat bekerja baik. Tantangan umum: variabilitas skor antar episode, overestimation bias pada Q, dan kebutuhan tuning hyperparameter.mengontrol sistem.

**D. Output**



**7. Kesimpulan**

mplementasi Deep Q-Network (DQN) dapat digunakan untuk menyelesaikan permasalahan kontrol pada environment CartPole-v1.

Agen mampu belajar menyeimbangkan tiang (pole) dengan mengandalkan mekanisme exploration–exploitation melalui strategi ε-greedy.

Hasil percobaan menunjukkan bahwa agen dapat meningkatkan performanya seiring bertambahnya episode pelatihan.

Walau berhasil, implementasi dasar ini masih terbatas karena tidak menggunakan target network atau teknik stabilisasi lain, sehingga pembelajaran bisa kurang stabil.

Untuk hasil lebih optimal, diperlukan penyesuaian hyperparameter (learning rate, gamma, epsilon decay) serta pengembangan metode lebih lanjut seperti Double DQN atau Dueling DQN.