

# Laporan Praktikum Kontrol Cerdas

Nama : Aureyza Pandu Qinara  
NIM : 224308078  
Kelas : TKA - 7D  
Akun Github (Tautan) : <https://github.com/aureyzapandu>  
Student Lab Assistant :-

Judul Percobaan

Week 4: *Reinforcement Learning for Autonomous Control*

## 1. Tujuan Percobaan

Tujuan dari praktikum “*Reinforcement Learning for Autonomous Control*”, mahasiswa diharapkan mampu:

- a) Memahami konsep dasar *Reinforcement Learning* (RL) dalam sistem kendali.
- b) Mengimplementasikan agen RL menggunakan algoritma *Deep Q-Network* (DQN).
- c) Menggunakan *OpenAI Gym* sebagai simulasi lingkungan untuk pelatihan RL.
- d) Melatih dan menguji agen RL untuk mengontrol lingkungan secara otonom.
- e) Menggunakan *GitHub* untuk *version control* dan dokumentasi praktikum.

## 2. Landasan Teori

- *Reinforcement Learning*

*Reinforcement Learning* (RL) adalah cabang pembelajaran mesin di mana agen belajar melalui interaksi dengan lingkungan. Agen mengambil suatu aksi, lalu lingkungan memberikan *feedback* berupa reward (hadiah) atau penalty (hukuman). Tujuan utama RL adalah menemukan *policy* (strategi) terbaik agar agen dapat memaksimalkan *reward* kumulatif sepanjang waktu (Sutton and Barto, 2015). Berbeda dengan *supervised learning* yang memerlukan data berlabel, RL bersifat *trial-and-error* sehingga sangat

cocok digunakan dalam masalah pengendalian dinamis dan sistem otonom.

- *Convolutional Neural Network (CNN)*

DQN merupakan pengembangan dari algoritma *Q-Learning* yang menggabungkan *Deep Neural Network* untuk mengaproksimasi fungsi nilai (*Q-value*). *Q-value* merepresentasikan seberapa baik suatu aksi diambil dalam kondisi tertentu. Dengan adanya *neural network*, DQN dapat menangani *state space* yang besar dan kompleks yang sulit ditangani *Q-Learning* biasa (Mnih et al., 2015).

Beberapa fitur penting DQN adalah:

- a. *Experience Replay*: menyimpan pengalaman agen di memori untuk dilatih ulang, sehingga mengurangi korelasi antar data.
- b. *Epsilon-Greedy Policy*: menjaga keseimbangan antara eksplorasi (mencoba aksi baru) dan eksploitasi (memilih aksi terbaik).
- c. *Target Network* (opsional): meningkatkan stabilitas pembelajaran dengan memisahkan jaringan prediksi dan target.

- *OpenAI Gym*

*OpenAI Gym* adalah pustaka *open-source* yang menyediakan berbagai *environment* untuk menguji algoritma RL. *Environment* ini memiliki standar API sederhana yang memudahkan interaksi agen dengan lingkungan melalui fungsi *reset()*, *step()*, dan *render()* (Brockman et al., 2016). *OpenAI Gym* banyak digunakan untuk penelitian dan praktikum RL, karena mencakup berbagai skenario mulai dari kontrol klasik, permainan Atari, hingga simulasi robot.

- *CartPole*

*CartPole* adalah salah satu *environment* klasik di *OpenAI Gym*. Masalahnya adalah mengendalikan tiang (*pole*) yang dipasang di atas sebuah kereta (*cart*) agar tetap tegak seimbang. Agen hanya bisa memilih dua aksi diskret: menggerakkan *cart* ke kiri atau kanan. *State space* terdiri dari 4 variabel kontinu: posisi *cart*, kecepatan *cart*, sudut *pole*, dan kecepatan *angular pole*. *Reward* diberikan +1 setiap langkah selama *pole* masih seimbang, dan episode berakhir jika *pole* jatuh atau *cart* keluar dari batas. Masalah ini sering digunakan sebagai benchmark awal untuk menguji algoritma RL

(Lillicrap et al., 2019).

- *MountainCar*

*MountainCar* adalah *environment* kontrol klasik lain di *OpenAI Gym*. Tugas agen adalah menggerakkan mobil kecil di lembah pegunungan untuk mencapai puncak di sisi kanan. Agen hanya dapat memilih 3 aksi diskret: mendorong ke kiri, diam, atau mendorong ke kanan. *State space* terdiri dari 2 variabel kontinu: posisi mobil dan kecepatannya. *Reward* default adalah -1 untuk setiap langkah hingga mobil mencapai puncak, sehingga agen didorong untuk menemukan strategi seefisien mungkin. Tantangan utama pada *MountainCar* adalah agen harus belajar menggunakan momentum (bergerak maju-mundur) agar bisa mencapai puncak (Kumar et al., 2022).

### 3. Analisis dan Diskusi

Pada percobaan ini digunakan algoritma *Deep Q-Network* (DQN) untuk mengendalikan *environment* CartPole. Jumlah episode pelatihan yang seharusnya 1000 dikurangi menjadi 100 agar proses lebih cepat, sehingga performa agen masih beragam dan belum stabil optimal. Pada 1000 episode, agen berpeluang besar mencapai strategi optimal sehingga nilai reward mendekati maksimum (500). Namun dengan 100 episode, agen belum sepenuhnya mengeksplorasi semua kemungkinan strategi, sehingga ada episode uji coba yang berhenti lebih cepat. Nilai gamma ( $\gamma$ ) mempengaruhi fokus agen terhadap reward jangka panjang, nilai epsilon ( $\epsilon$ ) mengatur keseimbangan eksplorasi dan eksploitasi, sedangkan learning rate menentukan seberapa cepat bobot jaringan saraf diperbarui. Keseimbangan eksplorasi dan eksploitasi menjadi kunci agar agen tidak hanya mencoba strategi baru, tetapi juga mengeksplorasi strategi terbaik yang sudah dipelajari.

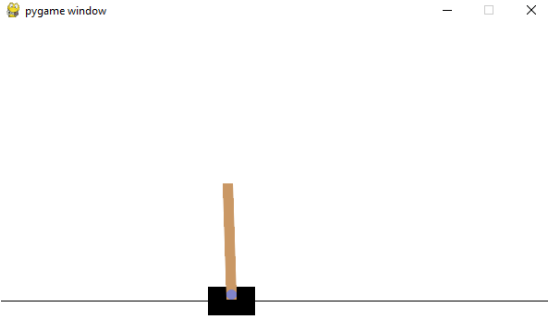
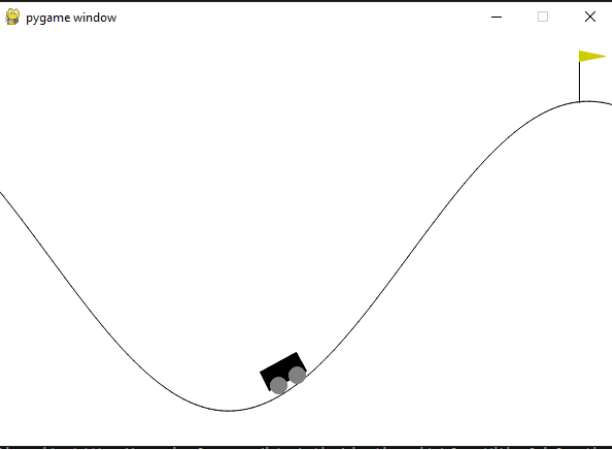
Berbeda dengan supervised learning yang menggunakan data berlabel, reinforcement learning mengandalkan feedback berupa reward dari interaksi dengan lingkungan. Tantangan utama adalah menyesuaikan parameter dan jumlah episode agar agen benar-benar mampu mencapai reward maksimum. Meskipun sederhana, percobaan CartPole ini mencerminkan potensi RL dalam sistem kendali nyata, seperti robotika, transportasi rel, kendaraan

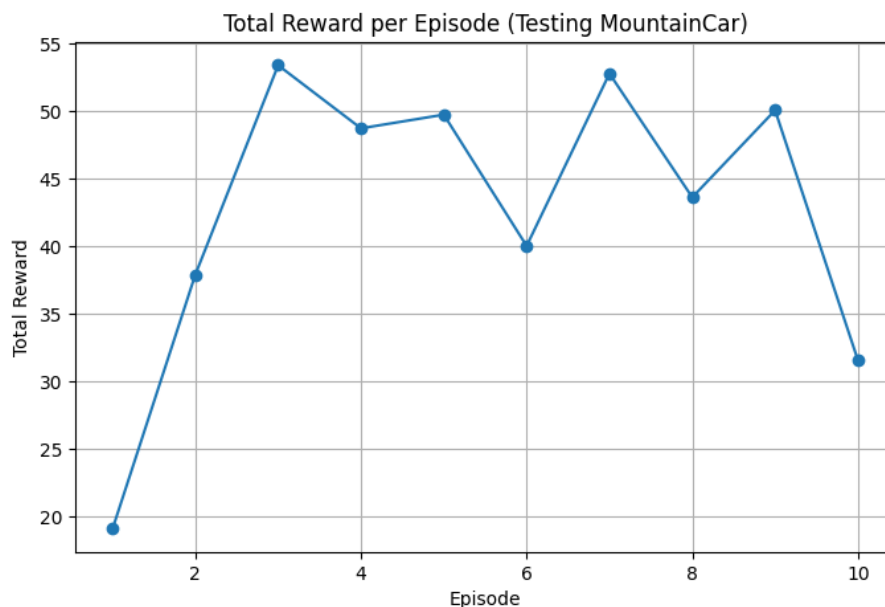
otonom, maupun optimasi proses industri.

#### 4. Assignment

1. Melakukan studi literatur mengenai *Reinforcement Learning* (RL), algoritme dasar seperti *Q-Learning*, DQN, dan PPO, serta penerapannya pada kendali otonom.
2. Membuat akun *GitHub*, *repository*, dan *clone* ke laptop untuk version control.
3. Menginstal *software* pendukung (*Python*, *VSCode*, *Git*) serta *library* (*gym/gymnasium*, *numpy*, *matplotlib*, *stable-baselines3*).
4. Menyiapkan *environment* *CartPole-v1* dan *MountainCar-v0* dari *OpenAI Gym* untuk eksperimen.
5. Mengimplementasikan algoritma RL (DQN untuk *CartPole*, DQN/PPO untuk *MountainCar*).
6. Menjalankan proses *training* dengan pengaturan *hyperparameter* (*learning rate*, *gamma*, *epsilon*, dll.).
7. Mencatat reward per episode dan membuat grafik learning curve untuk evaluasi performa.
8. Mengevaluasi model terlatih pada beberapa episode tanpa eksplorasi untuk melihat stabilitas dan akurasi kontrol.
9. Menyimpan hasil eksperimen (model, log, grafik, dan video episode) sebagai dokumentasi.
10. Melakukan *commit* dan *push* ke *repository GitHub* sebagai arsip akhir.

## 5. Data dan Output Hasil Pengamatan

No	Variabel	Hasil Pengamatan
1	Data hasil uji <i>environment CartPole</i>	<pre> Episode: 992/1000, Score: 499, Epsilon: 0.01 Episode: 993/1000, Score: 216, Epsilon: 0.01 Episode: 994/1000, Score: 263, Epsilon: 0.01 Episode: 995/1000, Score: 499, Epsilon: 0.01 Episode: 996/1000, Score: 264, Epsilon: 0.01 Episode: 997/1000, Score: 218, Epsilon: 0.01 Episode: 998/1000, Score: 210, Epsilon: 0.01 Episode: 999/1000, Score: 499, Epsilon: 0.01 Episode: 1000/1000, Score: 171, Epsilon: 0.01 </pre>
2	Hasil <i>enviromtent</i> <i>CartPole</i>	
3	Data hasil uji <i>Environment MountainCar</i>	<pre> PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL Episode: 989/1000, Score: 970, Epsilon: 0.01 Episode: 990/1000, Score: 1118, Epsilon: 0.01 Episode: 991/1000, Score: 962, Epsilon: 0.01 Episode: 992/1000, Score: 1045, Epsilon: 0.01 Episode: 993/1000, Score: 332, Epsilon: 0.01 Episode: 994/1000, Score: 935, Epsilon: 0.01 Episode: 995/1000, Score: 962, Epsilon: 0.01 Episode: 996/1000, Score: 1063, Epsilon: 0.01 Episode: 997/1000, Score: 388, Epsilon: 0.01 Episode: 998/1000, Score: 1131, Epsilon: 0.01 Episode: 999/1000, Score: 1144, Epsilon: 0.01 Episode: 1000/1000, Score: 942, Epsilon: 0.01 PS C:\&gt; </pre>
4	Hasil <i>enviromtent</i> <i>MountainCar.</i>	



Perbedaan hasil grafik pada *environment CartPole* dan *MountainCar* terutama dipengaruhi oleh struktur *reward*, tingkat kesulitan lingkungan, serta strategi pembelajaran yang dijalankan agen. Pada *CartPole*, agen menerima *reward* positif setiap langkah selama tiang tetap seimbang, sehingga sinyal pembelajaran jelas dan memotivasi agen untuk mempertahankan kondisi stabil lebih lama. Dengan demikian, setelah melalui 1000 episode, agen pada *CartPole* dapat mencapai skor yang tinggi dan relatif stabil, menunjukkan bahwa proses konvergensi sudah tercapai. Sebaliknya, pada *MountainCar*, agen justru mendapat penalti  $-1$  setiap langkah hingga mencapai *goal*, sehingga tanpa adanya *reward shaping* agen cenderung kesulitan menemukan strategi optimal. Lingkungan yang lebih

kompleks dan menuntut agen untuk mengumpulkan momentum dengan cara berayun bolak-balik juga menyebabkan proses pembelajaran berlangsung lebih lambat. Akibatnya, meskipun telah dilakukan *training* sebanyak 1000 episode, grafik pada *MountainCar* masih menunjukkan skor yang rendah dan cenderung fluktuatif, menandakan bahwa agen belum sepenuhnya berhasil mencapai konvergensi.

## 6. Kesimpulan

- a) Percobaan ini berhasil menunjukkan bagaimana agen belajar melalui interaksi dengan lingkungan menggunakan mekanisme *trial and error*, di mana keputusan yang tepat diperkuat melalui *reward*.
- b) Agen berbasis *Deep Q-Network* (DQN) berhasil dibuat dengan memanfaatkan jaringan saraf tiruan untuk memperkirakan nilai aksi, sehingga agen dapat belajar strategi kendali optimal pada *environment* yang diberikan.
- c) *OpenAI Gym* terbukti menjadi *platform* yang efektif untuk menyediakan *environment* standar (*CartPole-v1* dan *MountainCar-v0*) sehingga mempermudah proses eksperimen dan evaluasi performa agen.
- d) Hasil training menunjukkan bahwa pada *CartPole-v1*, agen mampu mencapai performa tinggi dan stabil setelah 1000 episode, sedangkan pada *MountainCar-v0*, agen membutuhkan strategi tambahan seperti *reward shaping* agar dapat belajar secara efektif. Hal ini memperlihatkan bagaimana agen RL dapat mengontrol sistem secara otonom dengan tingkat keberhasilan yang berbeda sesuai kompleksitas lingkungan.

## 7. Saran

- a) Disarankan untuk mengimplementasikan variasi algoritma RL yang lebih stabil seperti *Double DQN*, *Dueling DQN*, atau *Prioritized Experience Replay* agar performa agen meningkat, terutama pada *environment* yang kompleks seperti *MountainCar*.
- b) Disarankan untuk menambahkan visualisasi berupa grafik perbandingan *loss function*, *epsilon decay*, dan *average reward* agar analisis performa

agen lebih komprehensif.

- c) Perlu diterapkan strategi *reward shaping* yang lebih terkontrol agar agen tidak menghasilkan skor yang terlalu tinggi dan tidak realistis, misalnya dengan normalisasi atau *reward clipping*.

## 8. Daftar Pustaka

- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W., 2016. OpenAI Gym. <https://doi.org/10.48550/arXiv.1606.01540>
- Kumar, A., Levine, A., Feizi, S., 2022. Policy Smoothing for Provably Robust Reinforcement Learning. <https://doi.org/10.48550/arXiv.2106.11420>
- Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2019. Continuous control with deep reinforcement learning. <https://doi.org/10.48550/arXiv.1509.02971>
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. *Nature* 518, 529–533. <https://doi.org/10.1038/nature14236>
- Sutton, R.S., Barto, A.G., 2015. Reinforcement Learning: An Introduction.