

Laporan Praktikum Kontrol Cerdas

Minggu Ke-4

Nama : Harby Izza Ma'had Mahardika

NIM : 224308034

Kelas : TKA-6B

Akun Github (Tautan) : <https://github.com/mahardikaizza29>

Student Lab Assistant : Mas Ali

1. Judul Percobaan : *Reinforcement Learning for Autonomous Control*

2. Tujuan Percobaan :

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

3. Landasan Teori :

Reinforcement Learning (RL) adalah metode pembelajaran di mana agen belajar mengambil keputusan melalui interaksi dengan lingkungan. Agen memperoleh reward atau penalty sebagai umpan balik dari tindakannya, sehingga ia dapat mengoptimalkan strategi untuk mencapai tujuan tertentu.. Dalam konteks sistem kendali, RL digunakan untuk mengoptimalkan pengendalian suatu sistem tanpa model eksplisit, memungkinkan agen untuk belajar dari pengalaman langsung melalui interaksi dengan lingkungan (**Norman, n.d.**). Namun, RL juga memiliki tantangan, seperti waktu pelatihan yang lama, eksplorasi yang dapat menyebabkan hasil buruk di awal, serta kesulitan dalam merancang fungsi reward yang sesuai (**Andreanus & Kurniawan, 2018**).

Deep Q-Network (DQN) adalah algoritma dalam pembelajaran penguatan (*Reinforcement Learning*) yang menggabungkan *Q-Learning* dengan jaringan saraf dalam (*deep neural networks*) untuk mengatasi masalah dengan ruang keadaan yang besar dan kompleks (**Putra et al., 2024**). Dengan menggunakan DQN, agen dapat belajar dari

lingkungan yang kompleks dengan jumlah kemungkinan kondisi (state) yang sangat besar, sesuatu yang sulit dilakukan jika hanya mengandalkan Q-table seperti pada Q-Learning biasa. Tujuan utama agen adalah untuk menemukan urutan tindakan yang menghasilkan hadiah total terbesar dalam jangka panjang. Untuk itu, DQN menggunakan fungsi nilai Q yang memperkirakan seberapa baik setiap tindakan dalam suatu keadaan tertentu. Selain CartPole, DQN juga dapat digunakan pada lingkungan lain seperti LunarLander-v2, di mana agen belajar mengendalikan pendaratan pesawat luar angkasa, atau MountainCar-v0, di mana agen harus menemukan cara untuk mencapai puncak bukit dengan menggunakan momentum (Gou & Liu, 2019).

LunarLander-v2 adalah environment OpenAI Gym untuk menguji algoritma RL, di mana agen mengendalikan pendaratan wahana luar angkasa di permukaan bulan. Reward diberikan berdasarkan kualitas pendaratan dan efisiensi penggunaan bahan bakar, dengan penalti untuk pendaratan buruk atau keluar dari area. Sementara, MountainCar-v0 adalah environment di mana agen mengayunkan mobil untuk mendapatkan momentum agar mencapai puncak bukit. Agen dapat bergerak ke kiri, kanan, atau diam, dengan reward negatif setiap langkah, mendorong penyelesaian secepat mungkin sebelum batas langkah tercapai.

4. Analisis dan Diskusi :

A. Analisis

Pada Mata Kuliah Praktikum Kontrol Cerdas minggu keempat ini kami melakukan percobaan yaitu implementasi *Deep Q-Network* (DQN) untuk menyelesaikan permasalahan *MountainCar-v0* menggunakan *reinforcement learning*. Algoritma ini melatih agen agar dapat mencapai puncak bukit dengan memanfaatkan pengalaman yang dikumpulkan selama bermain. Program dimulai dengan mendefinisikan kelas *DQNAgent*, yang bertanggung jawab atas logika pembelajaran. Agen juga memiliki *memory buffer* (deque) yang digunakan untuk menyimpan pengalaman (*experience replay*), sehingga pembelajaran tidak hanya bergantung pada pengalaman terbaru, tetapi juga dari pengalaman sebelumnya. Agen mengambil keputusan berdasarkan eksplorasi dan eksploitasi, yang dikendalikan oleh parameter epsilon. Pada awalnya, agen cenderung memilih aksi secara acak (eksplorasi), tetapi seiring waktu nilai epsilon

akan berkurang secara bertahap (**epsilon_decay**), sehingga agen mulai lebih sering memilih aksi berdasarkan pengalaman sebelumnya (eksploitasi).

Dalam bagian utama kode, lingkungan *MountainCar-v0* dari *Gymnasium* diinisialisasi dengan mode *human render*, sehingga visualisasi pergerakan mobil dapat ditampilkan. Kemudian, agen dilatih selama episode yang telah ditentukan. Pada setiap episode, agen mengamati keadaan lingkungan (*state*), memilih aksi (*act()*), menerima umpan balik dalam bentuk reward, dan menyimpan pengalaman tersebut ke dalam *memory buffer*. Jika agen berhasil mencapai tujuan (*terminated*), ia mendapatkan reward yang lebih besar, sedangkan jika masih dalam perjalanan, ia diberikan penalti kecil untuk mendorong eksplorasi lebih lanjut. Selain itu, terdapat integrasi dengan *Pygame* untuk mendeteksi apakah pengguna menutup jendela permainan. Jika jendela ditutup, maka program akan berhenti dan *environment* akan ditutup dengan **env.close()**. Secara keseluruhan, kode ini menggambarkan proses pembelajaran agen menggunakan DQN, di mana agen belajar melalui pengalaman untuk mengoptimalkan pergerakan dalam lingkungan *MountainCar-v0*.

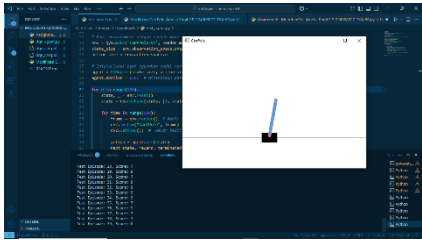
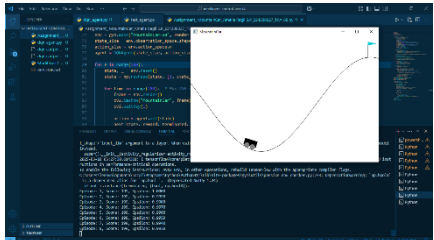
B. Diskusi

Program DQN ini melatih agen untuk menyelesaikan *MountainCar-v0* menggunakan *experience replay*, yang membantu agen belajar lebih stabil dengan mengandalkan pengalaman sebelumnya. Strategi eksplorasi dan eksploitasi diterapkan dengan nilai *epsilon* yang berkurang seiring waktu, memungkinkan agen menemukan strategi terbaik tanpa langsung terjebak pada pola tertentu. Sistem *reward* juga dirancang agar agen lebih termotivasi mencapai tujuan, meskipun bisa ditingkatkan dengan memberi hadiah tambahan saat mendekati puncak. Dari sisi pelatihan, penggunaan sampel acak membantu menjaga stabilitas, tetapi ada risiko *overfitting* yang bisa diatasi dengan metode *prioritized experience replay*. Namun, karena program ini menggunakan mode tampilan (*human render*), prosesnya bisa menjadi lebih lambat, terutama jika dijalankan dalam waktu lama atau pada perangkat dengan spesifikasi rendah. Untuk mengatasinya, saat pelatihan bisa dilakukan tanpa tampilan visual, lalu baru diaktifkan saat evaluasi. Secara keseluruhan, program ini cukup baik dalam menerapkan DQN, meskipun masih ada ruang untuk optimasi dalam eksplorasi, pemberian *reward*, dan efisiensi pembelajaran.

5. Assignment :

Tugas pada minggu keempat ini yaitu membuat sebuah program yang menerapkan algoritma Deep Q-Network (DQN) untuk menyelesaikan tantangan *MountainCar-v0* menggunakan pustaka *Gymnasium*. Program ini melatih agen agar dapat mencapai puncak bukit dengan belajar dari pengalaman sebelumnya. Dalam program ini, menerapkan konsep *experience replay*, yang memungkinkan agen menyimpan pengalaman dalam *memory buffer* dan menggunakannya kembali saat pelatihan. Hal ini membantu menjaga stabilitas pembelajaran dan mencegah agen terlalu bergantung pada pengalaman terbaru saja. Program ini memiliki beberapa fungsi penting yang mendukung proses pelatihan agen. Dalam kelas `DQNAgent`, terdapat fungsi `act()`, yang bertanggung jawab dalam menentukan aksi agen berdasarkan kebijakan *epsilon-greedy*, sehingga agen tetap memiliki peluang untuk mengeksplorasi lingkungan. Fungsi `remember()` digunakan untuk menyimpan pengalaman yang diperoleh selama pelatihan, sedangkan `replay()` berfungsi untuk mengambil sampel pengalaman secara acak dari *memory buffer* dan memperbarui nilai Q dengan cara melatih ulang model menggunakan data tersebut. Setiap episode dimulai dengan mengatur ulang lingkungan, lalu agen berulang kali memilih aksi, menerima reward, menyimpan pengalaman, dan memperbarui model berdasarkan data yang dikumpulkan. Jika jumlah pengalaman dalam *memory buffer* mencukupi, agen akan menjalankan proses pelatihan untuk memperbaiki keputusannya. Selain itu, program ini juga terintegrasi dengan *Pygame* untuk menangani input dari pengguna. Jika jendela permainan ditutup oleh pengguna, program akan berhenti secara otomatis. Namun, karena program menggunakan mode tampilan (*human render*), visualisasi ini dapat memperlambat proses pelatihan, sehingga bisa dipertimbangkan untuk menonaktifkan tampilan selama pelatihan dan hanya mengaktifkannya saat evaluasi. Beberapa parameter yang bisa diubah antara lain *gamma*, *epsilon decay*, atau arsitektur jaringan saraf untuk melihat bagaimana perubahan tersebut mempengaruhi performa agen. Selain itu, metode seperti *prioritized experience replay* dapat ditambahkan untuk meningkatkan efisiensi pembelajaran.

6. Data dan Output Hasil Pengamatan :

No.	Variabel	Hasil Pengamatan	
		CartPole-v1	MountainCar-v0
1.	Waktu Pelatihan	Lebih cepat mencapai performa optimal dalam beberapa ratus episode	Membutuhkan lebih banyak waktu untuk mencapai keberhasilan yang stabil
2.	Kondisi Awal untuk Scorenya	Skor bervariasi tergantung seberapa lama tiang tetap seimbang.	Skor awal mencapai 199 karena maksimal langkah dalam satu episode adalah 200.
3.	Kondisi Akhir	Episode berakhir jika tiang jatuh atau batas langkah maksimum tercapai	Episode berakhir jika mobil mencapai puncak bukit (bendera)
4.	Prinsip Kerja	Sistem ini mengikuti hukum fisika, di mana jika tongkat mulai miring, gerobak harus segera bergerak ke arah kemiringan untuk menyeimbangkannya. Jadi, tongkat harus tetap seimbang di atas gerobak yang bergerak ke kiri atau kanan.	Sistem ini menampilkan sebuah mobil kecil berada di lembah di antara dua bukit dan harus mencapai puncak salah satu bukit. Mobil tidak memiliki tenaga yang cukup untuk langsung naik, sehingga harus mengandalkan momentum dengan cara mengayun bolak-balik agar dapat mencapai ketinggian yang cukup.
5.	Hasil		

7. Kesimpulan :

Berdasarkan praktikum dan analisis yang telah dilakukan, maka dapat diambil kesimpulan yaitu:

- Metode *prioritized experience replay* dapat digunakan untuk membuat agen lebih cepat belajar dari pengalaman yang lebih berpengaruh terhadap keberhasilannya.
- Mode tampilan (*human render*) dalam simulasi memperlambat proses pelatihan. Oleh karena itu, saat pelatihan, sebaiknya tampilan visual dinonaktifkan dan hanya digunakan untuk evaluasi hasil.
- Lingkungan dengan *reward* yang jarang seperti *MountainCar*, membutuhkan eksplorasi yang lebih lama dan *replay buffer* yang lebih besar agar agen dapat mengumpulkan pengalaman yang cukup untuk menemukan strategi yang efektif.
- Penggunaan target *network* membantu stabilitas pembelajaran, menghindari perubahan nilai Q yang terlalu drastis.
- Dengan DQN, agen mampu belajar bahwa hanya maju terus tidak cukup untuk mencapai tujuan. Agen memahami bahwa terkadang perlu bergerak mundur terlebih dahulu untuk mengumpulkan momentum sebelum mencapai puncak.

8. Saran :

Untuk meningkatkan efektivitas pembelajaran agen dalam menyelesaikan permasalahan *MountainCar-v0*, ada beberapa hal yang dapat dioptimalkan. Salah satunya adalah penyesuaian parameter seperti *gamma*, *epsilon decay*, dan ukuran *memory buffer* agar agen dapat belajar lebih efisien dan menemukan strategi optimal dengan lebih cepat. Selain itu, penggunaan metode *prioritized experience replay* dapat dipertimbangkan agar agen lebih sering belajar dari pengalaman yang lebih relevan terhadap keberhasilannya, sehingga proses pelatihan menjadi lebih efektif. Dari segi efisiensi, karena *MountainCar-v0* membutuhkan banyak episode untuk belajar, sebaiknya tampilan visual (*human render*) dinonaktifkan selama pelatihan agar tidak memperlambat proses. Visualisasi dapat diaktifkan kembali saat evaluasi untuk melihat hasil akhir dari pembelajaran agen. Dengan berbagai perbaikan ini, diharapkan agen dapat belajar lebih cepat, lebih stabil, dan lebih efisien dalam menyelesaikan tantangan *MountainCar-v0*.

9. Daftar Pustaka :

- Andreanus, J., & Kurniawan, A. (2018). Sejarah, Teori Dasar dan Penerapan Reinforcement Learning: Sebuah Tinjauan Pustaka. *Jurnal Telematika*, 12(2), 113–118. <https://doi.org/10.61769/telematika.v12i2.193>
- Gou, S. Z., & Liu, Y. (2019). *DQN with model-based exploration: Efficient learning on environments with sparse rewards* (No. arXiv:1903.09295). arXiv. <https://doi.org/10.48550/arXiv.1903.09295>
- Norman, P. (n.d.). *Training reinforcement learning model with custom OpenAI gym for IIoT scenario*.
- Putra, R. A., Syahbana, Y. A., & Ananda. (2024). Implementasi Algoritma Deep Q-Network (DQN) pada Lampu Lalu Lintas Adaptif Berdasarkan Waktu Tunggu dan Arus Kendaraan. *The Indonesian Journal of Computer Science*, 13(5). <https://doi.org/10.33022/ijcs.v13i5.4372>