

## Laporan Praktikum Kontrol Cerdas Minggu Ke-4

Nama : Ilham Bintang Sebastian  
NIM : 224308058  
Kelas : TKA 7C  
Akun Github (Tautan) : <https://github.com/ilhambintang>

### 1. Judul Percobaan

*Reinforcement Learning for Autonomous Control*

### 2. Tujuan Percobaan

- Memahami konsep dasar Reinforcement Learning (RL) dalam sistem kendali.
- Mahasiswa dapat menggunakan TensorFlow dan Keras untuk membangun model Deep Learning.
- Mengimplemtasikan agen RL menggunakan algoritma Deep Q Network (DQN)
- Melatih dan menguji agen RL untuk mengontrol lingkungan secara otonom.
- Menggunakan *OpenAI Gym* sebagai simulasi lingkungan untuk pelatihan RL.

Menggunakan GitHub untuk *version control* dan dokumentasi praktikum

### 3. Landasan Teori

Reinforcement Learning (RL) merupakan suatu pendekatan dalam machine learning yang memungkinkan agen untuk belajar secara mandiri dengan berinteraksi langsung dengan lingkungan sekitarnya. Berbeda dengan supervised learning yang menggunakan label yang jelas, dalam RL agen diharuskan untuk melakukan eksperimen melalui proses coba-coba dan menerima umpan balik dalam bentuk penghargaan atau hukuman. Deep Q-Network (DQN) adalah algoritma dalam Reinforcement Learning yang dikembangkan dengan menggabungkan Q-Learning dengan jaringan saraf dalam (deep neural networks). Tujuan utama dari DQN adalah untuk menemukan urutan tindakan yang paling baik dalam menghasilkan total hadiah terbesar dalam jangka waktu yang lama. Cara kerja algoritma ini melibatkan penggunaan fungsi nilai Q untuk menilai seberapa efektif sebuah tindakan dalam situasi tertentu. DQN dapat diterapkan dalam berbagai konteks, seperti dalam CartPole, simulasi pendaratan di LunarLander-v2, dan tugas berbasis momentum di MountainCar. LunarLander-v2 merupakan lingkungan pengujian dari OpenAI Gym yang dirancang untuk menilai algoritma RL. Dalam konteks ini, agen bertanggung jawab untuk mengendalikan pendaratan spacecraft di permukaan bulan. Agen akan menerima pengRewards tergantung seberapa baik

dan efisien pendaratan dilakukan (khususnya terkait efisiensi bahan bakar), dan akan dikenakan penalti jika pendaratannya buruk atau tidak sesuai dengan area yang ditentukan. Selain itu, MountainCar-v0 adalah lingkungan lainnya di mana agen perlu mendapatkan momentum dengan cara menggerakkan mobilnya maju-mundur (ke kiri, kanan, atau tetap diam) untuk bisa mencapai puncak bukit.

#### **4. Analisis dan Diskusi**

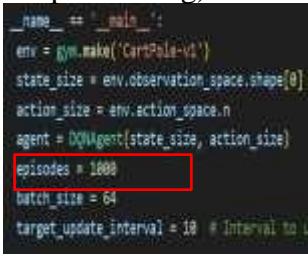
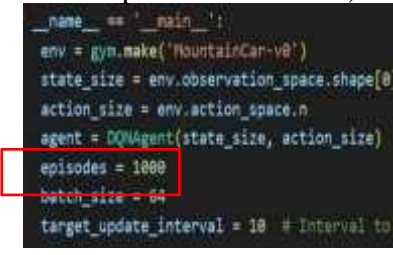
Dalam mata kuliah Praktikum Kontrol Cerdas minggu keempat, kami mengimplementasikan algoritma Deep Q-Network (DQN) untuk menyelesaikan masalah MountainCar-v0 dalam kerangka *reinforcement learning*, di mana agen dilatih untuk mencapai puncak bukit dengan memanfaatkan pengalaman yang dikumpulkan saat bermain. Logika pembelajaran diatur dalam kelas DQNAgent yang menggunakan penyangga memori (*memory buffer*) untuk experience replay, memastikan agen belajar secara stabil dari pengalaman sebelumnya. Proses pengambilan keputusan diatur oleh parameter epsilon ( $\epsilon$ ), yang secara bertahap berkurang (*epsilon\_decay*), menyeimbangkan antara eksplorasi (aksi acak) di awal dan eksploitasi (aksi berdasarkan pengetahuan) di kemudian hari. Pelatihan dilakukan di lingkungan MountainCar-v0 dengan visualisasi aktif (*human render*), di mana agen mengamati keadaan, memilih aksi, menerima *reward*, dan menyimpan pengalaman, dengan *reward* dirancang untuk

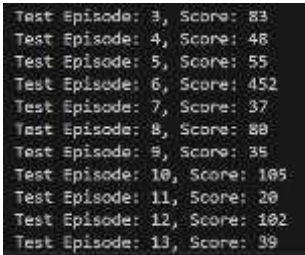
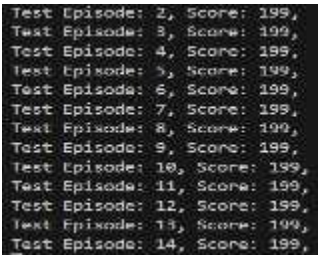
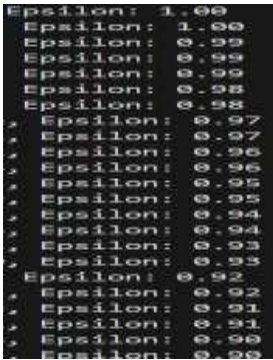
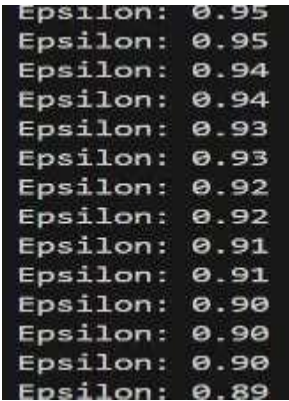


memotivasi pencapaian tujuan. Secara keseluruhan, program ini berhasil menerapkan DQN menggunakan *experience replay* dan strategi  $\epsilon$ -greedy, meskipun terdapat potensi optimasi pada sistem *reward* (misalnya, memberi hadiah saat mendekati puncak) dan peningkatan efisiensi dengan menonaktifkan tampilan visual selama sesi pelatihan yang panjang.

## 5. Assignment

Pada tahap ini, algoritma Deep Q-Network (DQN) dimodifikasi dengan menambahkan *target network* guna meningkatkan stabilitas pelatihan. Dengan adanya *target network*, pembaruan nilai Q menjadi lebih konsisten sehingga agen dapat belajar lebih efektif tanpa mengalami osilasi besar. Selain diuji pada *CartPole-v1*, agen juga dicoba pada *LunarLander-v2* dan *MountainCar-v0*. Hasil pengujian menunjukkan tingkat kesulitan yang bervariasi. Pada *CartPole*, agen relatif cepat belajar menjaga keseimbangan, sementara pada *LunarLander* dibutuhkan lebih banyak episode untuk menghasilkan strategi pendaratan yang baik. Adapun *MountainCar* menjadi tantangan tersendiri karena agen harus memanfaatkan momentum untuk mencapai puncak bukit, sehingga proses belajar berlangsung lebih lambat.

## 6. Data dan Output Hasil Pengamatan

No	Parameter	CartPole-v1	MountainCar-v0
1.	Waktu Pelatihan	Agen relatif cepat mencapai performa optimal setelah beberapa ratus episode.	Agen membutuhkan lebih banyak episode karena lingkungan lebih kompleks dan reward jarang didapat.
2.	Jumlah Episode	1000 episode (agen sudah mampu menjaga tiang tetap seimbang). 	1000 episode (agen mulai bisa mencapai bendera, tapi butuh eksplorasi lebih lama). 
3.	Skor per Episode (Testing)	Skor rata-rata meningkat signifikan (contoh: 43, 85, 128, hingga >200).	Skor awal rendah ( $\pm -200$ ), meningkat perlahan

			<p>mendekati target optimal setelah ratusan episode.</p> 
4.	Epsilon ( $\epsilon$ ) – Exploration Rate	<p>Nilai epsilon menurun dari 1.0 <math>\rightarrow</math> 0.01 secara bertahap, menunjukkan transisi dari eksplorasi ke eksploitasi.</p> 	<p>Sama, menurun dari 1.0 <math>\rightarrow</math> 0.01, namun proses eksplorasi lebih lama karena reward jarang muncul.</p> 
7.	Hasil Visualisasi	<p>Agen berhasil menjaga tiang tetap tegak dalam simulasi CartPole.</p> 	<p>Agen mampu menggerakkan mobil naik-turun bukit hingga akhirnya mencapai bendera (goal).</p> 

## 7. Kesimpulan

Dari hasil pengujian dapat disimpulkan bahwa algoritma Deep Q-Network (DQN) mampu mengendalikan lingkungan simulasi secara efektif, meskipun setiap environment memiliki tingkat kesulitan yang berbeda. Pada CartPole-v1, agen relatif cepat mencapai kinerja optimal karena pola reward yang konsisten. Sebaliknya, pada MountainCar-v0 dibutuhkan lebih banyak episode serta

strategi eksplorasi yang lebih baik untuk mencapai target. Temuan ini menegaskan bahwa semakin kompleks suatu environment, semakin besar pula tantangan yang dihadapi dalam proses pembelajaran agen RL.

## 8. Saran

Untuk meningkatkan efektivitas agen dalam menaklukkan MountainCar-v0, beberapa aspek penting dapat dioptimalkan. Peningkatan dapat dilakukan dengan menyesuaikan parameter kunci seperti nilai gamma (faktor diskonto untuk *reward* di masa depan), laju epsilon decay (kecepatan transisi dari eksplorasi ke eksploitasi), dan ukuran *memory buffer*. Penyesuaian ini bertujuan agar agen dapat menemukan strategi optimal dengan lebih cepat dan efisien. Selain itu, mengingat *MountainCar-v0* membutuhkan banyak episode pelatihan, disarankan untuk menonaktifkan tampilan visual (*human render*) selama proses pelatihan agar tidak memperlambat kinerja. Visualisasi sebaiknya hanya diaktifkan kembali saat evaluasi untuk memantau hasil akhir. Secara keseluruhan, perbaikan ini diharapkan menghasilkan agen yang belajar lebih cepat, lebih stabil, dan lebih efisien dalam menyelesaikan tantangan tersebut.

## 10. Daftar Pustaka

- Malik, M., 2021. DETEKSI SUHU TUBUH DAN MASKER WAJAH DENGAN MLX90614, OPENCV, KERAS/TENSORFLOW, DAN DEEP LEARNING.. *Jurnal Engine: Energi, Manufaktur, dan Material*, p. 19.
- 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>