

# Laporan Praktikum Kontrol Cerdas

Nama : Nefi Afif Sujatjana

NIM : 224308093

Kelas : TKA 7D

Akun GitHub (Tautan) : <https://github.com/nefiafif612-cmd>

## 1. Judul Percobaan :

Reinforcement Learning for Autonomous Control

## 2. Tujuan Percobaan

- Mahasiswa mampu memahami konsep dasar Reinforcement Learning (RL) dalam sistem kendali.
- Mahasiswa dapat mengimplemtasikan agen RL menggunakan algoritma *Deep Q-Network* (DQN).
- Mahasiswa mampu menggunakan OpenAI Gym sebagai simulasi lingkungan untuk pelatihan RL.
- Mahasiswa dapat melatih dan menguji agen RL untuk mengontrol lingkungan secara otonom.
- Mahasiswa dapat menggunakan GitHub untuk version control dan dokumentasi praktikum.

## 3. Landasan Teori

### 1. Reinforcement Learning (RL)

Reinforcement Learning (RL) merupakan salah satu paradigma dalam machine learning yang berfokus pada bagaimana suatu agen (agent) dapat belajar mengambil keputusan secara otonom melalui interaksi dengan lingkungannya (environment). Dalam RL, proses pembelajaran terjadi berdasarkan prinsip trial and error, di mana agen mencoba melakukan aksi (action) tertentu pada suatu kondisi (state), kemudian menerima umpan balik berupa reward atau hukuman (penalty). Tujuan utama RL adalah menemukan strategi pengambilan keputusan (policy) yang dapat memaksimalkan total reward jangka panjang (cumulative reward) (Sutton & Barto, 2018).

Secara umum, RL dapat dimodelkan sebagai Markov Decision Process (MDP) yang terdiri dari:

- a. State (S): kondisi lingkungan yang diamati oleh agen.
- b. Action (A): keputusan atau langkah yang dapat diambil agen.
- c. Reward (R): nilai yang diberikan oleh lingkungan sebagai umpan balik.
- d. Policy ( $\pi$ ): strategi agen dalam memilih aksi berdasarkan state.
- e. Value Function (V): ekspektasi reward jangka panjang dari suatu state.

## 2. Q-Learning

Salah satu algoritma RL klasik adalah Q-Learning, yang menggunakan tabel Q (Q-table) untuk menyimpan estimasi nilai (value) dari setiap pasangan state-action.

Nilai ini dikenal sebagai Q-value dan dihitung menggunakan persamaan Bellman:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s, a') - Q(s, a)]$$

Keterangan:

$s$  : state saat ini

$a$  : action yang diambil  $r$  : reward yang diterima  $\gamma$  : discount factor ( $0 < \gamma \leq 1$ ), menunjukkan seberapa penting reward jangka Panjang

$\alpha$  : learning rate, menentukan seberapa besar pembaruan nilai Q

Q-Learning efektif untuk permasalahan dengan jumlah state terbatas. Namun, ketika jumlah state sangat besar atau kontinu (misalnya input berupa gambar atau sensor kompleks), Q-table menjadi tidak efisien dan sulit diterapkan.

## 3. Deep Q-Network (DQN)

Untuk mengatasi keterbatasan Q-Learning, Mnih et al. (2015) memperkenalkan Deep Q-Network (DQN), yang menggabungkan Q-Learning dengan Deep Neural Network (DNN). Alih-alih menyimpan nilai Q pada tabel, DQN menggunakan jaringan saraf untuk memperkirakan fungsi Q. Dengan cara ini, DQN mampu memproses input berukuran besar, termasuk data visual.

Beberapa teknik penting pada DQN:

- a. Experience Replay: menyimpan pengalaman agen (state, action, reward, next state) ke dalam memori replay buffer. Data tersebut diambil secara acak untuk melatih model, sehingga mengurangi korelasi data dan meningkatkan stabilitas pembelajaran.
- b. Target Network: salinan dari jaringan utama (primary network) yang diperbarui secara periodik. Hal ini mencegah perubahan bobot yang terlalu cepat dan mengurangi osilasi dalam training.

- c. Epsilon-Greedy Policy: strategi eksplorasi di mana agen terkadang memilih aksi acak (eksplorasi) dengan probabilitas  $\epsilon$ , dan sisanya memilih aksi terbaik (eksploitasi). Nilai  $\epsilon$  biasanya menurun seiring waktu ( $\epsilon$ -decay).
  - d. Dengan kombinasi ini, DQN terbukti mampu mencapai performa setara manusia dalam beberapa permainan Atari klasik.
4. Implementasi RL dalam Simulasi

Dalam konteks pembelajaran, OpenAI Gym banyak digunakan sebagai platform simulasi untuk eksperimen RL. Dua environment yang sering digunakan adalah:

a. CartPole

CartPole adalah simulasi di mana agen harus menjaga tiang tetap seimbang di atas kereta. Agen mendapat reward jika berhasil menjaga tiang tetap tegak selama mungkin. Environment ini sering digunakan untuk menguji algoritma RL karena memiliki state yang bersifat kontinu, seperti posisi dan kecepatan kereta, serta sudut dan kecepatan tiang. Dengan menggunakan DQN, agen bisa belajar strategi yang tepat agar tiang tidak jatuh, meskipun awalnya sering gagal.

b. Mountain Car

Mountain Car adalah environment di mana mobil kecil harus mencapai puncak bukit, tetapi mesinnya terlalu lemah untuk langsung naik. Agar berhasil, mobil harus mengayun bolak-balik untuk membangun momentum. Reward hanya diberikan saat mobil mencapai puncak, dan setiap langkah lain mendapat penalti. Tantangan utama adalah reward yang jarang, sehingga agen perlu banyak eksplorasi untuk belajar.

Dari simulasi ini, mahasiswa bisa memahami bahwa setiap environment punya karakteristik berbeda. CartPole memberi reward lebih cepat sehingga agen bisa belajar dengan mudah, sedangkan MountainCar butuh strategi eksplorasi yang lebih karena reward hanya muncul jika tujuan tercapai.

5. Relevansi DQN dalam Praktikum

Dalam praktikum ini, penerapan DQN memberikan pengalaman langsung tentang bagaimana sebuah agen belajar melalui interaksi dengan lingkungan yang berbeda. Eksperimen dilakukan dengan memvariasikan parameter penting seperti learning rate, discount factor, dan epsilon decay untuk melihat pengaruhnya terhadap kecepatan konvergensi, stabilitas training, dan nilai reward yang diperoleh.

Dari hasil uji coba, terlihat bahwa penerapan DQN dapat mengatasi keterbatasan Q-Learning tradisional, meskipun tetap diperlukan teknik tambahan seperti target

network dan tuning hyperparameter agar performa agen optimal di berbagai environment.

#### 4. Analisis dan Diskusi

a) Bagaimana performa agen dalam mengontrol environment CartPole?

Pada environment CartPole, agen dengan algoritma DQN menunjukkan performa yang cukup baik setelah melalui sejumlah episode pelatihan. Agen mampu belajar untuk menjaga keseimbangan tiang (pole) dalam waktu relatif singkat, karena reward diberikan hampir setiap langkah ketika tiang tetap tegak. Hal ini memudahkan agen untuk mengidentifikasi pola aksi yang mengarah pada reward lebih tinggi. Dari hasil pengamatan, skor agen meningkat secara bertahap seiring bertambahnya episode, hingga mampu mempertahankan tiang selama batas maksimum episode (1000 langkah). Hal ini membuktikan bahwa DQN efektif digunakan pada permasalahan dengan feedback reward yang kontinu.

b) Bagaimana perubahan parameter (misal: gamma, epsilon, learning rate) mempengaruhi kinerja agen?

- Gamma ( $\gamma$ ): Semakin tinggi nilai gamma, agen cenderung mempertimbangkan reward jangka panjang, sehingga strategi yang dipelajari lebih stabil. Namun, jika terlalu tinggi mendekati 1, proses konvergensi menjadi lebih lambat.
- Epsilon ( $\epsilon$ ): Nilai epsilon mengatur keseimbangan antara eksplorasi dan eksploitasi. Pada awal pelatihan, epsilon tinggi memaksa agen lebih banyak mencoba aksi acak. Seiring waktu, epsilon menurun ( $\epsilon$ -decay) sehingga agen lebih fokus pada aksi terbaik yang telah dipelajari. Jika epsilon terlalu cepat turun, agen bisa terjebak pada solusi suboptimal.
- Learning Rate ( $\alpha$ ): Learning rate yang besar mempercepat pembaruan bobot jaringan, namun bisa menyebabkan instabilitas dan osilasi nilai reward. Sebaliknya, learning rate yang terlalu kecil memperlambat proses pembelajaran.
- Dari eksperimen, kombinasi parameter yang seimbang memberikan hasil terbaik, sedangkan parameter ekstrem (misalnya epsilon langsung sangat kecil atau learning rate terlalu besar) membuat agen sulit mencapai performa optimal.

c) Apa tantangan yang muncul selama pelatihan agen RL?

Tantangan utama yang muncul selama pelatihan agen RL adalah:

- Stabilitas Training: Kadang reward berfluktuasi tajam akibat pembaruan bobot jaringan yang tidak stabil.

- **Sparse Reward:** Pada environment seperti MountainCar, reward hanya diperoleh jika mobil berhasil mencapai puncak. Hal ini membuat proses pembelajaran lebih sulit karena agen harus menemukan strategi eksplorasi yang tepat.
- **Tuning Hyperparameter:** Pemilihan parameter seperti gamma, epsilon decay, dan learning rate sangat berpengaruh terhadap hasil akhir. Perlu dilakukan eksperimen berulang untuk menemukan kombinasi terbaik.
- **Overfitting terhadap Environment:** Jika agen terlalu lama dilatih dengan parameter tertentu, agen bisa terjebak hanya pada strategi spesifik tanpa kemampuan generalisasi.

## B. Diskusi

1. Apa perbedaan utama antara Reinforcement Learning dan metode supervised learning dalam sistem kendali?

Perbedaan utama terletak pada cara agen menerima umpan balik. Pada supervised learning, model belajar dari dataset yang berisi pasangan input-output yang benar, sehingga umpan balik bersifat langsung dan eksplisit. Sementara pada reinforcement learning, agen tidak diberikan label benar-salah, melainkan harus mengeksplorasi sendiri lingkungan dan menerima reward sebagai umpan balik. Dengan kata lain, supervised learning bersifat passive learning, sedangkan RL bersifat interactive learning di mana agen harus aktif mencoba berbagai aksi.

2. Bagaimana strategi eksplorasi (exploration) dan eksploitasi (exploitation) dapat dioptimalkan pada agen RL?

Strategi eksplorasi-eksploitasi merupakan tantangan penting dalam RL. Optimasi dapat dilakukan dengan:

- Menggunakan  $\epsilon$ -greedy policy dengan  $\epsilon$ -decay, di mana agen pada awalnya banyak bereksperimen (eksplorasi), lalu secara bertahap lebih sering mengeksploitasi strategi terbaik.
  - Menerapkan metode eksplorasi adaptif, misalnya Boltzmann exploration atau Upper Confidence Bound (UCB) untuk menyeimbangkan trade-off eksplorasi dan eksploitasi.
  - Menambahkan intrinsic reward seperti curiosity-driven exploration, sehingga agen tetap terdorong mencoba hal baru meskipun reward eksternal jarang.
3. Potensi aplikasi lain dari RL dalam sistem kendali nyata apa saja yang dapat diimplementasikan?

- Reinforcement Learning memiliki potensi luas dalam berbagai bidang sistem kendali nyata, di antaranya:
- Robotika: pengendalian robot otonom untuk navigasi, manipulasi objek, atau kerja kolaboratif di pabrik.
- Kendaraan Otonom: pengendalian mobil self-driving dalam menghadapi kondisi lalu lintas kompleks.
- Smart Grid & Energi: pengaturan distribusi energi listrik agar lebih efisien.
- Industri Manufaktur: optimasi jalur produksi dan pemeliharaan prediktif.
- Kendali Jaringan Komunikasi: optimasi routing dan manajemen sumber daya pada jaringan 5G/IoT.

Hal ini menunjukkan bahwa RL bukan hanya teori dalam simulasi, tetapi juga berpotensi besar untuk diimplementasikan dalam sistem kendali nyata yang kompleks.

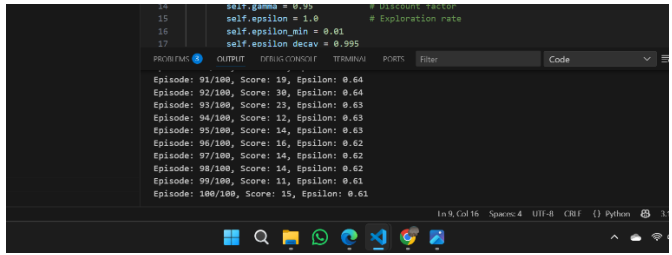
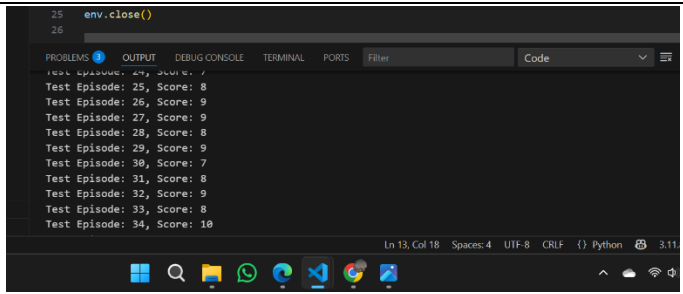
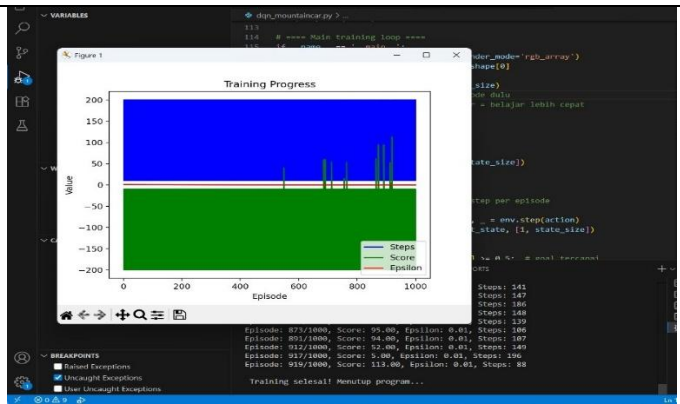
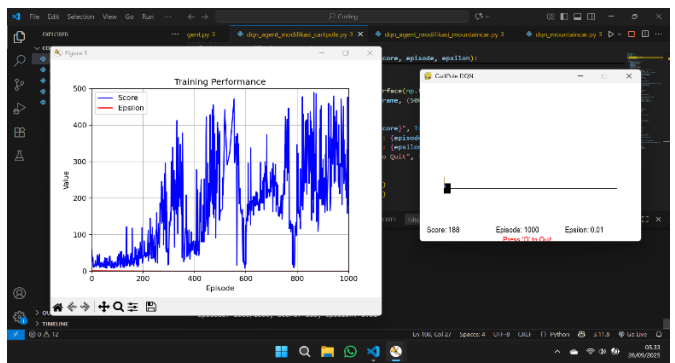
## 5. Assignment

Pada praktikum ke 4 ini kami mempelajari penerapan pada Deep Q-Network (DQN), sebuah pengembangan dari algoritma Q-Learning yang memanfaatkan jaringan saraf untuk mengatasi keterbatasan Q-table pada masalah dengan state yang besar atau kontinu. Tidak seperti Q-Learning klasik yang menyimpan nilai aksi dalam tabel, DQN memperkirakan fungsi Q menggunakan Deep Neural Network, sehingga lebih efisien dalam menangani input kompleks. Kami melakukan modifikasi terhadap DQN, salah satunya dengan menambahkan *target network* untuk meningkatkan stabilitas pelatihan. Target network memperbarui bobot secara berkala dari model utama, guna mengurangi osilasi dalam proses pembelajaran. Pengujian dilakukan pada beberapa environment seperti LunarLander-v2 dan MountainCar-v0, serta disertai eksperimen dengan memvariasikan parameter seperti *learning rate*, *discount factor* ( $\gamma$ ), dan *epsilon decay* untuk mengamati pengaruhnya terhadap performa agen.

Hasil eksperimen divisualisasikan dalam grafik reward, jumlah langkah, dan perubahan epsilon per episode. Seluruh kode, hasil, dan laporan didokumentasikan dalam repository GitHub. Praktikum ini menunjukkan bahwa DQN mampu mengatasi kelemahan Q-Learning tradisional, meskipun tetap memerlukan teknik tambahan dan penyesuaian parameter agar dapat bekerja optimal di berbagai kondisi.

## 6. Data dan Output Hasil Pengamatan

Data yang diperoleh pada praktikum minggu ke-4 dengan tabel dibawah ini:

No	Variabel	Hasil Pengamatan
1.	Percobaan pertama dilakukan men training 100 Episode dan didapatkan hasil dengan Score 15 dan Epsilon 0.61.	
2.	Hasil dari training 100 Espisode sebelum dimodifikasi ke 1000 Episode (test_agent.py) didapatkan test Episode 34 dan Score 10.	
3.	Di dapatkan hasil dari program mountain car setelah dimodifikasi menjadi 1000 Episode, dengan menunggu hasil sekitar 6 jam.	
4.	Di dapatkan hasil berupa grafik training CartPole DQN setelah di modifikasi menggunakan 1000 Episode, dan di dapatkan Score 188 dan Episode 0.01.	

## 7. Kesimpulan

Berdasarkan hasil praktikum Reinforcement Learning menggunakan algoritma Deep Q-Network (DQN), dapat disimpulkan bahwa:

1. Penggunaan DQN mampu mengatasi kelemahan Q-Learning klasik dengan menggunakan jaringan saraf untuk memprediksi fungsi Q, sehingga agen bisa belajar pada environment yang memiliki state yang rumit atau kompleks
2. Pada environment CartPole, agen menunjukkan performa sangat baik dengan cepat mencapai nilai reward maksimum setelah sejumlah episode, membuktikan bahwa DQN efektif pada kasus dengan feedback reward yang padat (dense reward).
3. Pada environment MountainCar, proses belajar menjadi lebih menantang karena reward hanya diberikan saat tujuan tercapai, sehingga agen perlu waktu lebih lama untuk mengeksplorasi dan menemukan strategi yang efektif. Ini menunjukkan bahwa karakter environment sangat berpengaruh terhadap tingkat kesulitan dalam pembelajaran.
4. Variasi parameter seperti gamma ( $\gamma$ ), epsilon decay, dan learning rate memiliki pengaruh signifikan terhadap performa agen. Kombinasi parameter yang tepat mempercepat konvergensi, sedangkan pengaturan ekstrem dapat menyebabkan pembelajaran tidak stabil atau gagal.
5. Penggunaan target network terbukti meningkatkan stabilitas pelatihan dengan mengurangi osilasi nilai Q dan memperbaiki konsistensi reward yang diperoleh agen.

## **8. Saran**

1. Untuk penelitian lanjutan, dapat dilakukan eksplorasi dengan algoritma RL lain seperti Double DQN, Dueling DQN, atau Prioritized Experience Replay agar performa agen semakin optimal.
2. Dilakukannya tuning hyperparameter yang lebih sistematis, sehingga didapatkan kombinasi parameter yang lebih optimal.
3. Dapat dilakukan eksperimen pada environment yang lebih kompleks, agar pemahaman tentang keterbatasan dan keunggulan DQN lebih mendalam.
4. Implementasi dapat diperluas ke sistem kendali nyata (misalnya robot otonom sederhana) untuk menguji performa RL di luar simulasi.
5. Dokumentasi dan visualisasi hasil pelatihan (reward, loss, epsilon decay) sebaiknya dibuat lebih lengkap agar memudahkan analisis dan perbandingan antar eksperimen.

## **9. Daftar Pustaka**

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.



- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.
- Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.
- François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. (2018). An introduction to deep reinforcement learning. *Foundations and Trends in Machine Learning*, 11(3–4), 219–354.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *arXiv preprint arXiv:1606.01540*.
- Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Silver, D., & Sutton, R. S. (2018). Rainbow: Combining improvements in deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).
- Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6), 26–38.