

# Laporan Praktikum Kontrol Cerdas

Nama : Rhehan Adi Prakoso  
NIM : 224308094  
Kelas : TKA-7D  
Akun Github (Tautan) : <https://github.com/RhehanAdi>  
Student Lab Assistant : -

## 1. Judul Percobaan

*Week 4 : Reinforcement Learning for Autonomous Control.*

## 2. Tujuan Percobaan

Tujuan dari dilakukannya praktikum “*Reinforcement Learning for Autonomous Control*” adalah:

- 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.

## 3. Landasan Teori

- A. ***Reinforcement Learning* (RL)** merupakan salah satu cabang dari kecerdasan buatan yang berfokus pada pembelajaran agen melalui interaksi langsung dengan lingkungannya untuk mencapai tujuan tertentu. Dalam RL, agen mengambil keputusan dengan memilih aksi berdasarkan kondisi atau *state* saat ini, kemudian menerima umpan balik berupa reward dari lingkungan. Tujuan utama agen adalah memaksimalkan reward kumulatif jangka panjang dengan cara belajar dari pengalaman *trial-and-error*. Berbeda dengan pembelajaran terawasi (*supervised learning*), RL tidak memerlukan label jawaban yang benar untuk setiap aksi, melainkan agen harus menemukan strategi optimal sendiri. RL memiliki beberapa komponen utama, yaitu agen, lingkungan, state, aksi, reward, serta fungsi nilai (*value function*) yang digunakan untuk menilai seberapa baik suatu state atau pasangan state-aksi dalam mencapai tujuan. Metode RL dapat diklasifikasikan menjadi beberapa kategori, antara lain

*value-based*, *policy-based*, dan *actor-critic*, masing-masing memiliki pendekatan berbeda dalam menentukan kebijakan (*policy*) agen. Algoritma populer seperti *Deep Q-Network* (DQN) diterapkan secara luas pada berbagai domain.

- B. ***Deep Q-Network*** (DQN) merupakan salah satu metode dalam *Reinforcement Learning* yang menggabungkan prinsip *Q-Learning* dengan *Deep Neural Network* untuk menangani masalah lingkungan dengan *state* kontinu atau berdimensi tinggi. Pada *Q-Learning* klasik, agen menggunakan *Q-table* untuk menyimpan nilai dari setiap pasangan *state*-aksi, yang menjadi prediksi seberapa menguntungkan suatu aksi pada *state* tertentu. Namun, pendekatan ini tidak praktis ketika jumlah *state* sangat besar atau kontinu, sehingga DQN menggunakan *neural network* sebagai fungsi aproksimasi *Q*, yang disebut *Q-network*. *Neural network* ini menerima representasi *state* sebagai input dan menghasilkan nilai *Q* untuk setiap aksi sebagai output. Selama proses *training*, DQN memperbarui bobot jaringan dengan *loss function* berbasis perbedaan antara prediksi *Q* dan target *Q*, yang dihitung menggunakan *reward* aktual dan prediksi *Q* dari *state* berikutnya. Untuk meningkatkan stabilitas dan efisiensi pembelajaran, DQN biasanya menggunakan teknik *experience replay*, di mana pengalaman sebelumnya disimpan dalam *buffer* dan diambil secara acak untuk melatih jaringan, serta *target network*, yang merupakan salinan jaringan utama yang diperbarui secara periodik. Dengan kombinasi, DQN memungkinkan agen belajar strategi optimal pada lingkungan kompleks secara efektif.
- C. ***OpenAI Gym*** adalah sebuah *platform open-source* yang dikembangkan untuk memfasilitasi penelitian dan pengembangan dalam bidang *Reinforcement Learning* (RL). *Gym* menyediakan berbagai *environment* standar yang mensimulasikan tugas-tugas kontrol dan permainan, mulai dari masalah klasik seperti *CartPole*, *MountainCar*, dan *Acrobot*, hingga simulasi yang lebih kompleks seperti robotika dan permainan Atari. Setiap *environment* di *Gym* memiliki *interface* yang seragam, di mana agen dapat berinteraksi melalui tiga elemen utama: *state*, *action*, dan *reward*. Dengan struktur yang konsisten ini, peneliti dapat dengan mudah membandingkan dan menguji berbagai algoritma

RL tanpa harus membangun simulasi dari nol. *OpenAI Gym* juga mendukung integrasi dengan berbagai *library machine learning*, seperti *TensorFlow* dan *PyTorch*, sehingga memungkinkan penggunaan jaringan saraf untuk aproksimasi fungsi nilai, *policy*, atau model lingkungan. Dengan demikian, Gym berperan penting dalam mempercepat eksperimen, replikasi hasil penelitian, dan pengembangan algoritma RL yang lebih kompleks dalam skala penelitian maupun industri.

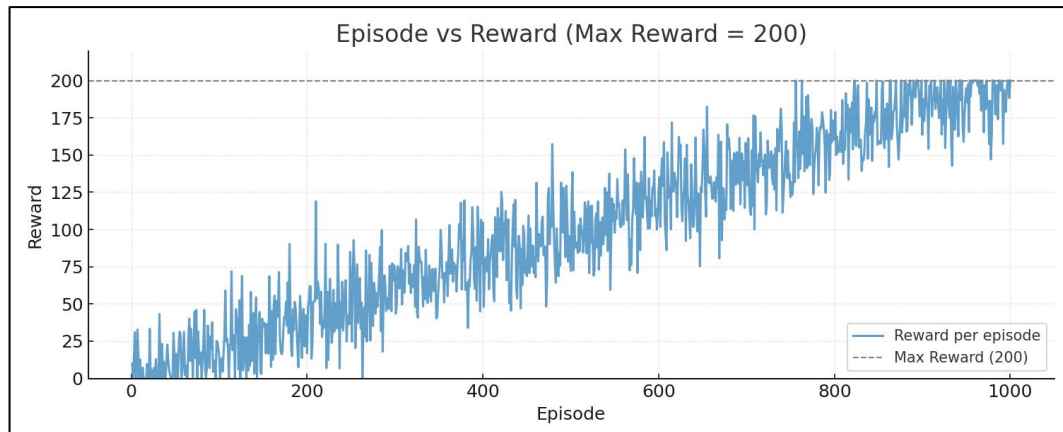
D. ***Autonomous Control*** adalah cabang dari sistem kendali yang memungkinkan suatu sistem atau perangkat untuk mengatur perilaku dan aksinya secara mandiri tanpa intervensi manusia secara langsung. Sistem ini memanfaatkan kombinasi sensor, aktuator, algoritma kontrol, dan kecerdasan buatan untuk mengamati kondisi lingkungan, memproses informasi, menentukan aksi optimal, dan mengeksekusi keputusan tersebut. Tujuan utama *autonomous control* adalah mencapai performansi optimal dan stabilitas sistem dalam menghadapi dinamika lingkungan yang berubah-ubah. Dalam implementasinya, autonomous control sering mengintegrasikan metode seperti *PID control*, *Model Predictive Control (MPC)*, *fuzzy logic*, dan *Reinforcement Learning*, sehingga memungkinkan sistem untuk belajar, beradaptasi, dan membuat keputusan *real-time*. Contoh aplikasinya meliputi kendaraan otonom, robotika, pesawat tanpa awak, dan sistem manufaktur otomatis, di mana sistem harus mampu melakukan tugas kompleks secara konsisten, aman, dan efisien tanpa pengawasan manusia.

#### 4. Analisis

Pada praktikum ini dilakukan pembuatan mode *Reinforcement Learning (RL)* dengan menggunakan *Deep Q-Network (DQN)* untuk menangani masalah lingkungan dengan *state* kontinu atau berdimensi tinggi. Terdapat 2 model yang dibuat. Pertama ada *cartpole* dan *mountain car*. *Cartpole* dan *mountain car* adalah salah satu *environment* klasik yang sering digunakan untuk menguji algoritma dari RL.

*Cartpole* adalah *environment* RL klasik yang bertugas menyeimbangkan tiang (pole) yang terpasang pada kereta (*cart*) disepanjang jalur horizontal. Agen menerima *reward* +1 untuk setiap langkah dimana tiang tetap seimbang. Episode

berakhir jika tiang miring lebih dari sudut tertentu atau jika cart bergerak keluar dari batas jalur. Untuk membuat agen ini dilakukan proses *training* menggunakan program pada python. Proses *training* dilakukan sebanyak 1000 episode yang merupakan batas minimal *training* untuk menguji *environment* RL. Banyaknya episode yang digunakan selama *training*, mempengaruhi hasil dari agen itu sendiri. Grafik terkait hal tersebut terlihat pada gambar 1 sebagai berikut.

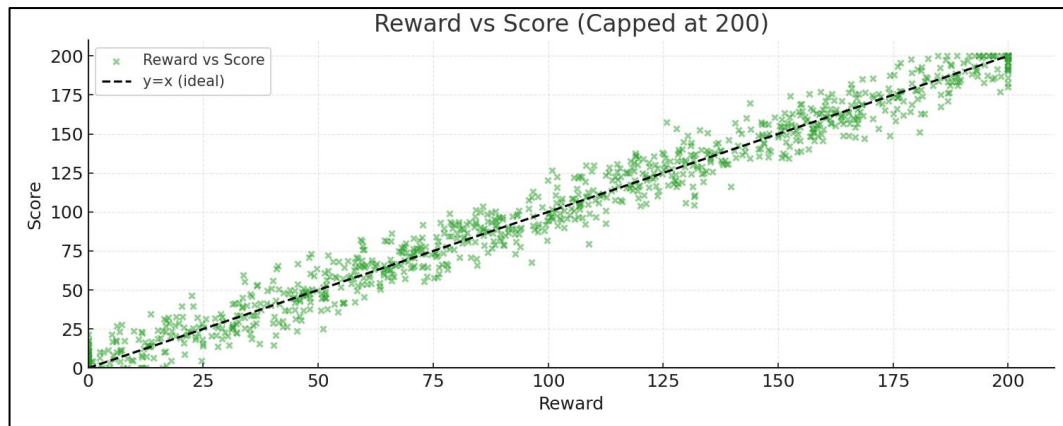


Gambar 1. Hubungan Jumlah Episode dengan *Reward* pada Agen

Pada gambar 1. terlihat semakin banyak episode yang digunakan maka *reward* semakin meningkat. Selama proses *training*, jumlah episode menentukan pola *reward* yang dicapai. Episode sendiri merupakan satu rangkaian percobaan yang dimulai dari kondisi awal hingga agen mencapai keadaan terminal atau batas langkah. Pada tahap awal *reward* yang diperoleh agen umumnya rendah dikarenakan keterbatasan pengalaman sehingga aksi yang dipilih bersifat acak dan belum optimal. Seiring bertambahnya episode, agen mengumpulkan informasi dan dari interaksi agen sebelumnya. Informasi tersebut digunakan untuk memperbaiki fungsi nilai atau kebijakan sehingga pemilihan aksi berikutnya menjadi lebih baik melalui *reward* yang meningkat.

Selanjutnya dilakukan proses uji coba atau *testing*. Pada program *training* yang sebelumnya hasil *training* disimpan dalam format *.keras*. Proses *testing* menggunakan program yang berbeda dengan memanggil data hasil *training* sebelumnya. Pengujian agen dilakukan dengan menjalankan agen hasil *training* pada lingkungan simulasi yang sama, tetapi tanpa mekanisme eksplorasi acak (nilai *epsilon* diset mendekati nol sehingga agen hanya mengeksplorasi kebijakan terbaik yang telah dipelajari). Selama program berjalan agen mampu meningkatkan performanya berdasarkan *reward* yang didapat. Grafik hubungan terhadap *score*

terlihat pada gambar 2 sebagai berikut.



Gambar 2. Hubungan *Reward* terhadap *Score* pada Agen

Pada gambar 2. Terlihat hubungan semakin banyak *reward* yang diperoleh agen maka *score* yang didapatkan agen semakin meningkat. Setiap aksi yang dilakukan agen menghasilkan *reward* sebagai bentuk umpan balik langsung dari lingkungan, baik dalam bentuk penalti maupun penghargaan. *Reward* ini kemudian dijumlahkan secara akumulatif sepanjang satu episode untuk menghasilkan *score* akhir sehingga *score* merupakan refleksi dari total *reward* yang berhasil dikumpulkan agen dari awal hingga akhir episode. Semakin sering agen memilih aksi yang sesuai dengan tujuan pembelajaran dan menghasilkan *reward* positif, maka *score* yang dicapai akan semakin tinggi. Sebaliknya, apabila agen lebih banyak menerima *reward* negatif atau gagal mempertahankan kondisi lingkungan sesuai target, maka *score* yang dihasilkan cenderung rendah.

Uji coba pada mountaincar juga berlaku dengan kondisi yang sama. Yang membedakan hanya bentuk environment RLnya saja. Jika pada mountain car agen mengendalikan mobil diantara dua bukit. Tugasnya adalah mendorong mobil ke puncak bukit disebelah kanan. Mobil tidak memiliki daya dorong yang cukup untuk langsung mencapai puncak sehingga agen harus mempelajari strategi untuk membangun momentum dalam mencapai puncak.

1. Bagaimana performa agen dalam mengontrol environment CartPole?
  - \* Agen RL belajar menyeimbangkan tiang pada CartPole melalui banyak percobaan. Awalnya, sering gagal, namun seiring waktu agen menjadi lebih baik dan mampu menyeimbangkan tiang lebih lama.
2. Bagaimana perubahan parameter (misal: gamma, epsilon, learning rate) mempengaruhi kinerja agen?

- \* a. Gamma (Discount Factor) Semakin tinggi nilai gamma, agen mempertimbangkan reward jangka panjang. Jika gamma terlalu rendah, agen hanya fokus pada reward langsung.
  - b. Epsilon (Exploration Rate) Nilai epsilon tinggi membuat agen mencoba berbagai aksi secara acak untuk menemukan strategi terbaik. Secara bertahap, nilai epsilon diturunkan agar agen lebih sering memilih aksi yang terbukti efektif.
  - c. Learning Rate menentukan seberapa cepat agen mengubah perkiraan nilainya. Nilai yang terlalu tinggi bisa membuat pembelajaran tidak stabil, sedangkan nilai yang terlalu rendah membuat proses belajar menjadi lambat.
3. Apa tantangan yang muncul selama pelatihan agen RL?
- \* Agen kadang mengalami fluktuasi *reward* yang menyebabkan pembelajaran tidak stabil. Menentukan kapan harus mencoba aksi baru (eksplorasi) dan kapan harus menggunakan aksi yang sudah diketahui efektif (eksploitasi) merupakan tantangan tersendiri. Proses pelatihan, terutama dengan algoritma seperti DQN, membutuhkan banyak waktu dan sumber daya komputasi.

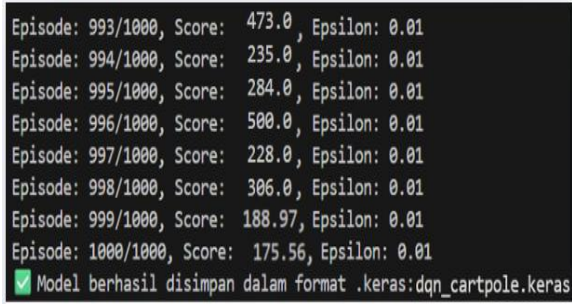
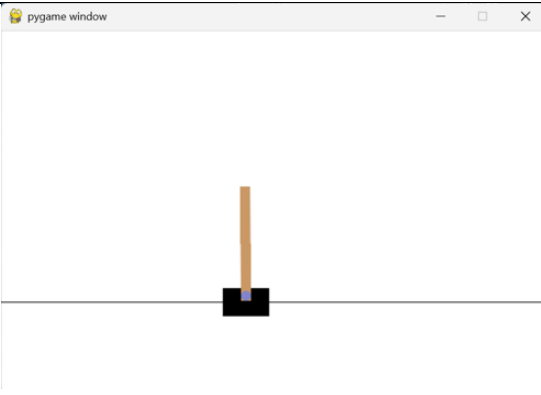
#### **Diskusi Hasil**

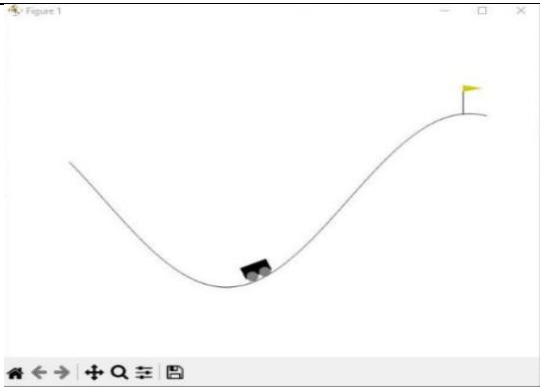
1. Apa perbedaan utama antara Reinforcement Learning dan metode supervised learning dalam sistem kendali?
  - \* Supervised learning menggunakan data yang sudah dilabeli, sedangkan RL belajar dari interaksi langsung dengan lingkungan tanpa label. Dalam supervised learning, model belajar dari contoh yang benar. Di RL, agen belajar melalui trial-and-error dengan mendapatkan reward atau hukuman. RL berfokus pada mendapatkan reward kumulatif maksimal, sedangkan supervised learning fokus pada akurasi prediksi.
2. Bagaimana strategi eksplorasi (exploration) dan eksploitasi (exploitation) dapat dioptimalkan pada agen RL?
  - \* Eksplorasi (Agen mencoba berbagai aksi untuk menemukan strategi terbaik) dan Eksploitasi (Agen menggunakan strategi yang sudah terbukti memberikan reward tinggi). Optimalisasi, Pengurangan nilai epsilon secara bertahap (epsilon decay) membantu menyeimbangkan antara eksplorasi (mencoba hal baru) dan eksploitasi (menggunakan strategi yang sudah diketahui efektif).
3. Potensi aplikasi lain dari RL dalam sistem kendali nyata apa saja yang dapat diimplementasikan?
  - \* Robotika, Untuk mengontrol pergerakan robot dalam navigasi atau manipulasi objek. Kendaraan Otonom, Mengoptimalkan keputusan pada mobil self-driving.

## 5. Assignment

- Studi literatur terkait *Reinforcement Learning* serta algoritma DQN dalam *environment Cartpole* dan *Mountaincar*
- Melakukan pemograman pada python untuk *training* agen *cartpole* dan *mountaincar*.
- Melakukan *training* pada agen *cartpole* dan *mountaincar* sebanyak 1000 episode.
- Setelah proses training selesai dilakukan pemograman yang berbeda untuk proses *testing* agen *cartpole* dan *mountaincar*.
- Melakukan *testing* agen *cartpole* dan *mountaincar*.
- Melakukan upload hasil praktikum pada repository Github.

## 6. Data dan Output Hasil Pengamatan

No	Variabel	Hasil Percobaan	Keterangan
1	<i>Proses training agen cartpole</i>	 <pre> Episode: 993/1000, Score: 473.0, Epsilon: 0.01 Episode: 994/1000, Score: 235.0, Epsilon: 0.01 Episode: 995/1000, Score: 284.0, Epsilon: 0.01 Episode: 996/1000, Score: 500.0, Epsilon: 0.01 Episode: 997/1000, Score: 228.0, Epsilon: 0.01 Episode: 998/1000, Score: 306.0, Epsilon: 0.01 Episode: 999/1000, Score: 188.97, Epsilon: 0.01 Episode: 1000/1000, Score: 175.56, Epsilon: 0.01 ✅ Model berhasil disimpan dalam format .keras:dqn_cartpole.keras </pre>	Proses <i>training</i> agen <i>cartpole</i> dilakukan sebanyak 1000 episode dengan epsiolon hampir mendekati 0 yakni 0.01
2	<i>Testing agen cartpole</i>		Selama <i>testing</i> berjalan <i>cartpole</i> berusaha untuk mempertahankan posisinya tetap seimbang. Agen menerima reward +1 untuk setiap langkah yang dihasilkan

No	Variabel	Hasil Percobaan	Keterangan
3	Hasil <i>testing</i> agen <i>cartpole</i>	<pre> Episode 1/10, Score: 473.0 Episode 2/10, Score: 235.0 Episode 3/10, Score: 284.0 Episode 4/10, Score: 500.0 Episode 5/10, Score: 228.0 Episode 6/10, Score: 306.0 Episode 7/10, Score: 500.0 Episode 8/10, Score: 304.0 Episode 9/10, Score: 293.0 Episode 10/10, Score: 500.0 </pre>	Agen mampu belajar dari setiap langkah yang dihasilkan. Pada agen Terjadi peningkatan score seiring berjalan nya episode
4	<i>Proses training</i> agen <i>mountaincar</i>	<pre> Episode: 993/1000, Score: -175.56, Epsilon: 0.01 Episode: 994/1000, Score: -175.26, Epsilon: 0.01 Episode: 995/1000, Score: -169.88, Epsilon: 0.01 Episode: 996/1000, Score: -188.97, Epsilon: 0.01 Episode: 997/1000, Score: -165.35, Epsilon: 0.01 Episode: 998/1000, Score: -177.45, Epsilon: 0.01 Episode: 999/1000, Score: -176.66, Epsilon: 0.01 Episode: 1000/1000, Score: -163.43, Epsilon: 0.01 ✓ Model berhasil disimpan dalam format .keras: dqn_mountaincar.k eras </pre>	Proses <i>training</i> agen <i>cartpole</i> dilakukan sebanyak 1000 episode dengan epsiolon hampir mendekati 0 yakni 0.01
5	<i>Testing</i> agen <i>mountaincar</i>		Selama <i>testing</i> berjalan <i>cartpole</i> berusaha untuk mempertahankan posisinya tetap seimbang. Agen menerima <i>reward</i> +1 untuk setiap langkah yang dihasilkan



No	Variabel	Hasil Percobaan	Keterangan
6	Hasil <i>testing</i> agen <i>mountaincar</i>	<div> Episode: 1/10, Score: 199.00  Episode: 2/10, Score: 199.00  Episode: 3/10, Score: 199.00  Episode: 4/10, Score: 199.00  Episode: 5/10, Score: 199.00  Episode: 6/10, Score: 199.00  Episode: 7/10, Score: 199.00  Episode: 8/10, Score: 199.00  Episode: 9/10, Score: 200.00  Episode: 10/10, Score: 200.00 </div>	Agen mampu belajar dari setiap langkah yang dihasilkan. Pada agen Terjadi peningkatan score seiring berjalan nya episode

## 7. Kesimpulan

- Agen RL menunjukkan peningkatan kemampuan di lingkungan *CartPole* setelah beberapa episode pelatihan. Parameter gamma, epsilon, dan learning rate memengaruhi stabilitas dan kinerja agen.
- Gamma yang rendah membuat agen mengabaikan reward jangka panjang.
- Epsilon yang tinggi mendorong eksplorasi berlebih.
- Learning rate yang sesuai penting untuk menjaga kestabilan pembelajaran.
- Tantangan utama meliputi fluktuasi reward, kestabilan pelatihan, dan keseimbangan eksplorasi-eksploitasi.
- Performa agen di lingkungan *MountainCar* masih memerlukan perbaikan untuk mengatasi tantangan eksplorasi yang kompleks.

## 8. Saran

- Eksplorasi teknik peningkatan stabilitas seperti *Prioritized Experience Replay* atau Double DQN.
- Sesuaikan parameter gamma, epsilon decay, dan learning rate secara dinamis sesuai kebutuhan lingkungan.
- Gunakan hardware akselerasi seperti GPU untuk mempercepat pelatihan dan memungkinkan eksperimen yang lebih kompleks.

## 9. Daftar Pustaka

Abadi, Arulkumaran, K., Deisenroth, M. P., Brundage, M. & Bharath, A. A., 2017.

Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine, 34(6), pp. 26-38.

Barto, A. G., Sutton, R. S. & Anderson, C. W., 1983. Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, and Cybernetics, SMC 13(5), pp. 834-846.

Mnih, V. et al., 2015. Human-level control through deep reinforcement learning. Nature, 518(7540), pp. 529-533.