

Laporan Praktikum 4 Kontrol Cerdas

Nama : Rafli May Sandy

NIM : 224308020

Kelas : TKA-6A

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

Student Lab Assistant : Rizky Putri Ramadhani (214308092)

1. Judul Percobaan

Melatih dan Menguji Agen Reinforcement Learning Untuk Mengontrol Lingkungan Secara Otonom

2. Tujuan Percobaan

Tujuan dari percobaan pada praktikum kali ini, sebagai berikut:

1. Mahasiswa dapat memahami Konsep dasar dari Reinforcement Learning (RL) dalam sistem kendali.
2. Mahasiswa dapat mengetahui cara mengimplementasikan RL menggunakan algoritma Deep Q-Network (DQN).
3. Mahasiswa dapat mengetahui bagaimana melatih dan menguji agen RL untuk mengontrol lingkungan secara otonom.

3. Landasan Teori

- Reinforcement Learning (RL) merupakan salah satu cabang dari Machine Learning yang berbeda dari supervised learning dan unsupervised learning. Metode ini berfokus pada pengembangan agen cerdas yang dapat belajar dari interaksi dengan lingkungan (environment). Dalam RL, agen bertujuan mencapai suatu target dengan menerima feedback berupa reward atau penalty atas setiap tindakan yang dilakukan. Agen kemudian belajar untuk memaksimalkan akumulasi reward atau meminimalkan penalty guna mencapai tujuan dengan lebih optimal.
- Environment merupakan elemen penting dalam pengembangan Reinforcement Learning (RL). Environment berperan dalam memberikan observasi dari lingkungan simulasi kepada agen RL, menerima dan mengeksekusi aksi yang dilakukan oleh agen, serta mengevaluasi aksi tersebut dengan memberikan reward feedback kepada agen. Dalam pengembangan algoritma RL, keberadaan environment sangat krusial, karena tidak memungkinkan untuk langsung menguji algoritma dalam lingkungan fisik. Implementasi langsung pada sistem nyata, seperti robot, menghadapi berbagai tantangan, termasuk biaya tinggi, banyaknya variabel tak terkontrol yang dapat mempengaruhi hasil, serta kendala fisik yang dapat memperlambat proses pelatihan agen. Oleh karena itu, simulasi environment digunakan untuk menguji dan mengoptimalkan algoritma sebelum diterapkan di dunia nyata.
- OpenAI Gym adalah pustaka open-source yang dikembangkan oleh tim OpenAI untuk mendukung pengembangan algoritma Reinforcement Learning (RL). OpenAI Gym ditulis dalam bahasa pemrograman Python, yang banyak digunakan oleh peneliti machine learning dan data science dalam mengembangkan serta mengimplementasikan berbagai algoritma. Gym dirancang untuk

mempermudah pengembangan algoritma RL, mengatasi kendala utama yang sering dihadapi oleh peneliti, yaitu ketiadaan standar dalam pengujian dan perbandingan algoritma. Tanpa environment standar, sulit untuk menguji dan mereproduksi hasil penelitian yang dilakukan oleh peneliti lain. Selain itu, membangun environment sendiri akan memakan waktu dan sumber daya yang signifikan, sehingga OpenAI Gym menyediakan berbagai environment siap pakai untuk mempercepat proses penelitian.

- Python adalah bahasa pemrograman yang populer. Python sering dimanfaatkan dalam pengembangan web, perangkat lunak, penelitian, dan system scripting. Python dapat digunakan untuk 39 menangani data besar dan melakukan operasi matematika yang kompleks. Python bekerja di berbagai platform seperti Windows, Mac, Linux, Raspberry Pi, dan lain-lain. Python dirancang untuk mudah dibaca, yaitu memiliki sintaks yang sederhana dan menggunakan bahasa inggris.
- NumPy (Numerical Python) merupakan library Python yang fokus pada scientific computing. NumPy memiliki kemampuan untuk membentuk objek N-dimensional array, yang mirip dengan list pada Python. Struktur data NumPy lebih membutuhkan ukuran yang lebih kecil dibandingkan dengan list lainnya tetapi mempunyai performa yang lebih cepat
- TensorFlow adalah salah satu library software yang dikembangkan oleh Google Brain dalam organisasi penelitian Google Smart Machine. Teknologi ini dirancang untuk pembuatan model machine learning serta penelitian jaringan saraf tiruan. TensorFlow menggabungkan aljabar komputasi dengan teknik optimasi kompilasi, sehingga dapat mempercepat perhitungan ekspresi matematis yang kompleks, terutama dalam mengatasi masalah efisiensi waktu komputasi. Berikut beberapa fitur utama TensorFlow (Abadi et al., 2016):
 1. Memungkinkan definisi, optimasi, dan perhitungan efisien terhadap ekspresi matematis yang melibatkan array multidimensi (tensors).
 2. Mendukung pemrograman berbasis kecerdasan buatan (AI) dan machine learning.
 3. Memanfaatkan GPU secara optimal, dengan otomatisasi manajemen memori dan data. TensorFlow dapat menjalankan kode yang sama di CPU maupun GPU, serta menentukan secara otomatis bagian perhitungan yang perlu diproses di GPU.
 4. Skalabilitas tinggi, mampu menangani komputasi dalam skala besar di berbagai mesin dan dataset yang besar.

TensorFlow memiliki berbagai fitur yang mendukung training, testing, dan deployment model machine learning dengan lebih efisien.

4. Analisis dan Diskusi

Analisis:

Dalam percobaan pada minggu ke-4 ini, saya melakukan percobaan melatih dan menguji agen RL untuk mengontrol lingkungan secara otonom. Dalam percobaan ini saya menggunakan algoritma Deep Q-Network (DQN) yang berfungsi untuk mengimplementasikan RL, selain itu saya juga menggunakan TensorFlow untuk membuat dan menjalankan RL. Dalam proses pembuatan, program dibagi menjadi 2. Program pertama digunakan untuk melatih agen RL hingga mencapai 1000 episode sedangkan pada program kedua digunakan untuk menguji agen RL tersebut. Ketika kedua program dijalankan, pada program pertama menghasilkan data latih sebanyak 1000 episode dengan keterangan score dan epsilon disetiap episode, sedangkan pada program kedua menghasilkan data uji berupa score dan juga gambar animasinya. Dari kedua data tersebut dapat disimpulkan bahwa ketika agen dilatih sebanyak 100 episode, ia belum mampu mengontrol CartPole dengan baik dikarenakan score pelatihan tidak meningkat secara signifikan dan tidak stabil sehingga pada skor uji menjadi rendah. Sedangkan ketika agen dilatih sebanyak 1000 episode menghasilkan score yang lebih tinggi walaupun masih kurang stabil atau naik turun. Namun hal ini dapat diatasi dengan melakukan perubahan parameter seperti gamma, epsilon, dan learning rate. Ketika nilai gamma tinggi maka agen lebih focus pada reward jangka Panjang dan dapat meningkatkan stabilitas pembelajaran, sedangkan jika nilai gamma rendah agen akan lebih focus pada reward yang sudah jadi atau instan dan dapat mempercepat pembelajaran. Pada epsilon jika diberi nilai tinggi agen akan lebih sering mencoba aksi acak (eksplorasi) sehingga sangat bagus digunakan untuk awal training. Jika epsilon diberi nilai rendah

agen akan lebih sering memilih aksi terbaik (eksploitasi) sehingga bagus digunakan untuk akhir training. Pada learning rate jika diberi nilai tinggi membuat agen dapat belajar dengan cepat tetapi bisa tidak stabil dan rentan terhadap overfitting, sedangkan jika diberi nilai rendah dapat membuat agen menjadi lambat dalam belajar tetapi bisa membuat lebih stabil dan dapat meningkatkan performa jangka panjang.

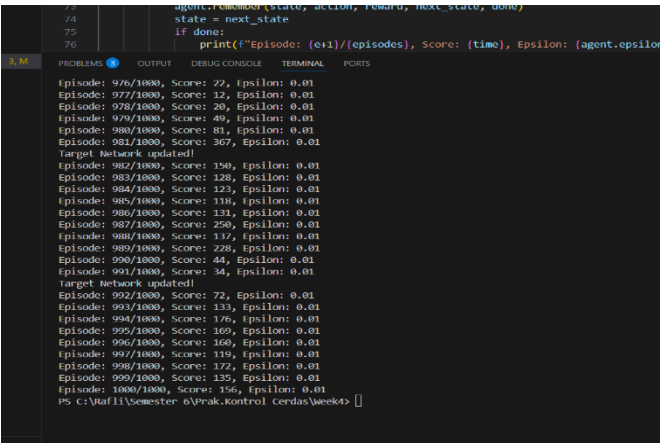
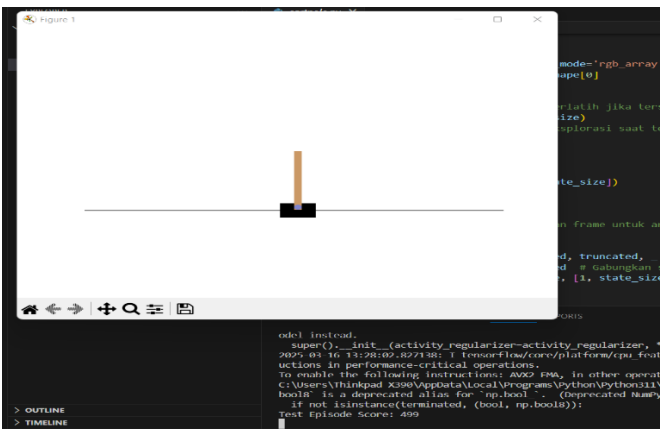
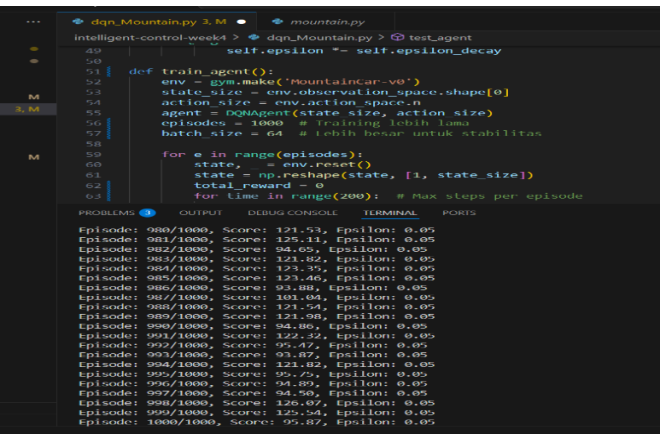
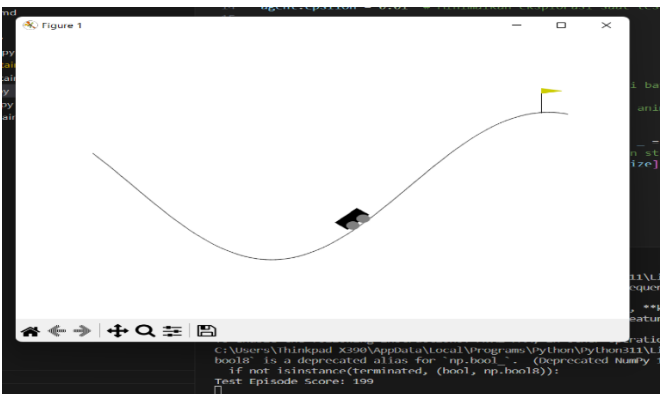
Diskusi:

1. Perbedaan utama antara RL dan Supervised Learning dalam sistem kendali dapat dilihat dari cara belajar dan sumber data. Pada RL dalam melakukan pembelajaran melalui trial dan error dengan mengeksplorasi lingkungan dan menerima reward dan RL tidak memerlukan dataset berlabel karena dapat mengumpulkan data sendiri dengan berinteraksi dengan lingkungan. Sedangkan Supervised Learning dalam melakukan pembelajaran melalui data berlabel yang telah diberikan sebelumnya sehingga membutuhkan dataset berlabel besar untuk melatih model dengan akurasi tinggi.
2. Cara mengoptimalkan keseimbangan antara eksplorasi dan eksploitasi yaitu, pada awal pelatihan gunakan lebih banyak eksplorasi dengan menggunakan ϵ -Greedy, Boltzmann, atau UCB untuk mengeksplorasi lingkungan. Ketika sudah berjalan seiring waktu gunakan eksploitasi bisa dengan menggunakan ϵ -decay atau Dueling DQN untuk menemukan strategi terbaik. Jika kondisi masalah dengan reward jarang gunakan curiosity-driven exploration.
3. Aplikasi lain dari RL dalam sistem kendali nyata yang dapat diimplementasikan yaitu:
 - Kendali kendaraan otonom (Self-Driving Cars & Autonomous Rail).
 - Kendali Robotik dan otomasi industry.
 - Kendali pergerakan kereta api.

5. Assignment

Pada assignment, program latih dan juga program uji agen dimodifikasi dengan mengubah environmentnya yang awalnya menggunakan CartPole-v1 menjadi MountainCar-v0. Selain itu pada program latih ditambahkan fitur target network agar dapat meningkatkan stabilitas pelatihan. Kemudian, untuk menjalankan programnya sama dengan menjalankan program sebelumnya, yaitu jalankan dulu program latih baru program ujinya. Untuk hasil data latih maupun uji pada program MountainCar berbanding terbalik dengan hasil data dengan menggunakan CartPole. Pada program dengan menggunakan CartPole semakin besar nilai score yang dihasilkan maka semakin bagus environment yang dihasilkan. Sedangkan mountain semakin rendah nilai data score pada pelatihan agen maka akan semakin cepat agen mencapai puncak. Selain itu perbedaan selanjutnya yaitu pada tujuannya, CartPole bertujuan untuk Menjaga keseimbangan tiang di atas troli dengan menggerakkan troli ke kiri atau kanan sedangkan MountainCar Membantu mobil mencapai puncak bukit dengan mengayun bolak-balik untuk mendapatkan momentum. Dari hasil kedua data dengan menggunakan MountainCar dapat disimpulkan bahwa agen belum mampu mengontrol MountainCar dengan baik dikarenakan skor pelatihan tidak menurun secara signifikan, sehingga skor uji masih terlalu tinggi yang membuat agen belum bisa mencapai puncak.

6. Data dan Output Hasil Pengamatan

NO	Variabel	Hasil Pengamatan
1	Data hasil latih agen environment CartPole	 <pre> 75 agent.reset(state, action, reward, next_state, done) 76 state = next_state 77 if done: 78 print(f"Episode: {e+1}/{episodes}, Score: {time}, Epsilon: {agent.epsilon}") 79 80 Episode: 976/1000, Score: 22, Epsilon: 0.01 81 Episode: 977/1000, Score: 12, Epsilon: 0.01 82 Episode: 978/1000, Score: 20, Epsilon: 0.01 83 Episode: 979/1000, Score: 49, Epsilon: 0.01 84 Episode: 980/1000, Score: 81, Epsilon: 0.01 85 Episode: 981/1000, Score: 307, Epsilon: 0.01 86 Target Network updated! 87 Episode: 982/1000, Score: 150, Epsilon: 0.01 88 Episode: 983/1000, Score: 128, Epsilon: 0.01 89 Episode: 984/1000, Score: 123, Epsilon: 0.01 90 Episode: 985/1000, Score: 118, Epsilon: 0.01 91 Episode: 986/1000, Score: 131, Epsilon: 0.01 92 Episode: 987/1000, Score: 250, Epsilon: 0.01 93 Episode: 988/1000, Score: 117, Epsilon: 0.01 94 Episode: 989/1000, Score: 228, Epsilon: 0.01 95 Episode: 990/1000, Score: 44, Epsilon: 0.01 96 Episode: 991/1000, Score: 34, Epsilon: 0.01 97 Target Network updated! 98 Episode: 992/1000, Score: 72, Epsilon: 0.01 99 Episode: 993/1000, Score: 133, Epsilon: 0.01 100 Episode: 994/1000, Score: 176, Epsilon: 0.01 101 Episode: 995/1000, Score: 169, Epsilon: 0.01 102 Episode: 996/1000, Score: 160, Epsilon: 0.01 103 Episode: 997/1000, Score: 119, Epsilon: 0.01 104 Episode: 998/1000, Score: 172, Epsilon: 0.01 105 Episode: 999/1000, Score: 135, Epsilon: 0.01 106 Episode: 1000/1000, Score: 156, Epsilon: 0.01 107 ps c:\Uatili\Semester 6\Prak.Kontrol cerdas\Week4> </pre>
2	Hasil envirointment CartPole.	
3	Data hasil latih agent envirointment MountainCar	 <pre> ... dqn_Mountain.py 3.8M mountain.py Intelligent-control-week4 > dqn_Mountain.py - test_agent 49 self.epsilon -= self.epsilon_decay 50 51 def train_agent(): 52 env = gym.make("MountainCar-v0") 53 state_size = env.observation_space.shape[0] 54 action_size = env.action_space.n 55 agent = DQNAgent(state_size, action_size) 56 episodes = 1000 # training lebih lama 57 batch_size = 64 # lebih besar untuk stabilitas 58 59 for e in range(episodes): 60 state = env.reset() 61 total_reward = 0 62 for time in range(200): # Max steps per episode 63 64 Episode: 980/1000, Score: 121.53, Epsilon: 0.05 65 Episode: 981/1000, Score: 125.11, Epsilon: 0.05 66 Episode: 982/1000, Score: 94.65, Epsilon: 0.05 67 Episode: 983/1000, Score: 121.82, Epsilon: 0.05 68 Episode: 984/1000, Score: 123.35, Epsilon: 0.05 69 Episode: 985/1000, Score: 123.46, Epsilon: 0.05 70 Episode: 986/1000, Score: 93.88, Epsilon: 0.05 71 Episode: 987/1000, Score: 101.64, Epsilon: 0.05 72 Episode: 988/1000, Score: 121.54, Epsilon: 0.05 73 Episode: 989/1000, Score: 121.90, Epsilon: 0.05 74 Episode: 990/1000, Score: 94.86, Epsilon: 0.05 75 Episode: 991/1000, Score: 122.32, Epsilon: 0.05 76 Episode: 992/1000, Score: 95.47, Epsilon: 0.05 77 Episode: 993/1000, Score: 93.87, Epsilon: 0.05 78 Episode: 994/1000, Score: 121.82, Epsilon: 0.05 79 Episode: 995/1000, Score: 95.75, Epsilon: 0.05 80 Episode: 996/1000, Score: 94.89, Epsilon: 0.05 81 Episode: 997/1000, Score: 94.50, Epsilon: 0.05 82 Episode: 998/1000, Score: 126.07, Epsilon: 0.05 83 Episode: 999/1000, Score: 125.54, Epsilon: 0.05 84 Episode: 1000/1000, Score: 95.87, Epsilon: 0.05 </pre>
4	Hasil envirointment MountainCar	

7. Kesimpulan

Dari percobaan yang telah dilakukan dapat disimpulkan bahwa:

1. Percobaan ini dilakukan dengan menggunakan TensorFlow untuk membangun model Deep Q-Network (DQN) yang nantinya digunakan untuk mengimplementasikan agen RL dan menggunakan OpenAI Gym sebagai simulasi lingkungan untuk pelatihan RL.
2. Pada percobaan ini terdiri dari 2 tahapan, yaitu tahap pelatihan agen yang digunakan untuk melatih agen RL dan tahap pengujian agen yang digunakan untuk menguji agen RL yang sebelumnya sudah dilatih.
3. Hasil percobaan menunjukkan bahwa, jika agen dilatih sebanyak 100 episode maka agen belum mampu mengontrol CartPole dengan baik, yang ditunjukkan dengan skor pelatihan yang tidak meningkat secara signifikan dan tidak stabil serta skor uji yang rendah. Sedangkan ketika agen dilatih sebanyak 1000 episode agen mampu mengontrol CartPole dengan baik, yang ditunjukkan dengan skor pelatihan yang meningkat walaupun masih belum stabil hal ini disebabkan oleh pemilihan parameter yang kurang optimal, seperti gamma, epsilon, dan learning rate.

8. Saran

Berdasarkan hasil percobaan yang telah dilakukan, terdapat beberapa saran yang penulis ajukan, sebagai berikut:

1. Agar dapat meningkatkan performa agen dalam mengontrol CartPole disarankan untuk menyesuaikan parameter, menambah jumlah episode pelatihan, dan menggunakan teknik optimasi agar agen RL dapat belajar lebih baik dan mencapai kontrol optimal pada CartPole.
2. Implementasikan algoritma Deep Q-Network (DQN) dalam permasalahan yang ada di dunia nyata.

9. Daftar Pustaka

Irvansyah, Muhammad Adhitya. (2020). *IMPLEMENTASI METODE REINFORCEMENT LEARNING PADA SIMULASI PERGERAKAN ROBOT MULTI-SENDI*
