Pasinformatik, Volume 4, Issue 2, July 2025
E-ISSN : 2986-5360

Teknik Peningkatan Performa Pelatihan Berbasis Visual *Reinforcement Learning*

Faqih Firdaus Kemal Pangestu[1], Handoko Supeno[2]

Jurusan Teknik Informatika, Fakultas Teknik, Universitas Pasundan
Jln. Dr. Setiabudhi no. 193 Bandung, Jawa Barat
[1] faqihfirdaus026@gmail.com, [2] handoko@unpas.ac.id

Abstrak : Kemampuan kendaraan otonom dalam bermanuver dan menghindari tabrakan merupakan aspek penting dalam pengembangan sistem kendaraan otonom. Penelitian ini bertujuan untuk mengimplementasikan metode Reinforcement Learning (RL) berbasis *Value Based* supaya meningkatkan kemampuan agen dalam bermanuver dan menghindari tabrakan secara mandiri menggunakan hanya sensor visual berbasis kamera. Penelitian ini mencakup proses pengumpulan data dari simulasi kendaraan otonom, pengembangan dan pelatihan model menggunakan algoritma Value Based, serta evaluasi kinerja model dengan metrik seperti reward function, tingkat keberhasilan penghindaran tabrakan, dan manuver. Beberapa Teknik kemudian diusulkan dalam penelitian ini untuk menyempurnakan yang ditemukan pada algoritma value based seperti *overestimation*, penggunaan pengalaman yang tidak efisien, dan keterbatasan observasi secara parsial yang lazim menjadi masalah pada *Value Based Reinforcement Learning*. Model ini diujicobakan dan dijalankan pada dunia simulator Enduro. Hasil dari penelitian ini diharapkan dapat meningkatkan ketepatan penghindaran tabrakan kendaraan otonom dan kemampuan bermanuver. Selain itu, penelitian ini memberikan kontribusi bagi pengembangan teknologi kendaraan otonom berbasis kecerdasan buatan, yang dapat diaplikasikan dalam sistem kendaraan di masa depan.

Kata kunci: *Reinforcement Learning, Machine Learning,* kendaraan otonom, kecerdasan buatan.

I. PENDAHULUAN

Pada era saat ini, salah satu bidang teknologi yang mengalami perkembangan pesat adalah kendaraan otonom atau *Autonomous Vehicle* (AV). AV merupakan kendaraan yang mampu bergerak dan bernavigasi secara mandiri di lingkungan nyata tanpa kontrol langsung dari manusia. Sistem ini dirancang untuk mengamati dan memahami lingkungan sekitarnya melalui input visual, seperti kamera, dan kemudian mengambil keputusan berdasarkan persepsi tersebut, menyerupai cara kerja manusia saat mengemudi [1]. Dengan berkembangnya kemampuan persepsi berbasis visual, kendaraan otonom kini dapat mengenali objek, jalan, dan rambu-rambu lalu lintas menggunakan kamera dan pengolahan citra, yang kemudian digunakan dalam proses pengambilan keputusan secara *real-time*. Salah satu fitur penting dari AV yang sedang banyak dikembangkan adalah sistem navigasi, terutama dalam hal penghindaran tabrakan (*collision avoidance*). Menurut Siegwart et al. (2022), sistem navigasi yang efektif mencakup beberapa komponen utama: pengambilan data sensor, pemetaan lingkungan, pengambilan keputusan kognitif, dan kontrol gerakan [2]. Kemampuan ini sangat penting untuk meminimalkan risiko kecelakaan di jalan raya. Dikutip dari Kominfo.go.id, kecelakaan lalu lintas di Indonesia pada tahun-tahun terakhir masih didominasi oleh kelalaian manusia (human error). Berdasarkan laporan kepolisian dan data Kementerian Perhubungan, lebih dari 60% kecelakaan disebabkan oleh faktor manusia, seperti ketidaktahuan pengemudi terhadap aturan dan kurangnya konsentrasi selama berkendara, disusul oleh kondisi kendaraan yang tidak layak jalan, serta faktor lingkungan dan prasarana [3]. Seiring dengan kemajuan teknologi, *Machine Learning* (ML) berperan penting dalam pengembangan sistem AV. Salah satu pendekatan ML yang menjanjikan adalah *Reinforcement Learning* (RL), di mana agen (kendaraan) belajar dari interaksi langsung dengan lingkungannya melalui proses trial and error. Agen mencoba berbagai tindakan untuk mencapai tujuan tertentu dan mendapatkan *reward* (imbalan) atau *punishment* (hukuman) sebagai umpan balik dari lingkungan. Salah satu varian RL yang terkenal adalah Deep Q-Learning (DQN), yang memanfaatkan jaringan saraf tiruan (neural network) untuk memperkirakan nilai dari suatu tindakan berdasarkan kondisi saat ini [4]. Meskipun DQN telah menunjukkan hasil yang menjanjikan, pendekatan ini masih memiliki keterbatasan, salah satunya adalah kebutuhan terhadap banyak pengalaman untuk mencapai performa optimal. Untuk mengatasi hal ini, dikembangkan metode dan Teknik yang lebih canggih seperti *Double Deep Q Network* (DDQN), *Multi step* DQN, hingga *Prioritized Experience Replay (PER)*, yang memberikan prioritas lebih tinggi pada pengalaman-pengalaman penting saat pengambilan sampel, sehingga proses pembelajaran menjadi lebih efisien [5]. Lebih lanjut, pengembangan strategi pembelajaran bertahap seperti *Multi-Step Learning* mulai digunakan untuk meningkatkan efisiensi pembelajaran dalam algoritma RL. Berbeda dari pendekatan satu langkah (*single-step*), *Multi-Step*

Learning mempertimbangkan *reward* kumulatif dari beberapa langkah ke depan sekaligus. Pendekatan ini telah terbukti mampu mempercepat konvergensi dan meningkatkan stabilitas pembelajaran, karena agen tidak hanya belajar dari satu transisi, tetapi dari rangkaian hasil berturut-turut yang lebih bermakna secara keseluruhan. Dengan mempertimbangkan konsekuensi dari tindakan jangka pendek dan menengah, *Multi-Step DQN* menjadi lebih adaptif terhadap situasi dinamis di lingkungan berkendara.

II. METODE PENELITIAN

Metode yang digunakan dalam penelitian ini adalah *Deep Reinforcement Learning*. Metode penelitiannya adalah sebagai berikut:

1. Identifikasi Masalah
2. Perancangan *State*
3. Perancangan Aksi
4. Perancangan *reward function*
5. Perancangan model pembelajaran
6. Perancangan Algoritma
7. Perancangan simulasi
8. Pelatihan agent
9. *Fine Tuning* Metode
10. Pengujian Agent
11. Evaluasi Agent

Hasil penelitian ini akan memberikan kontribusi berupa algoritma dan *reward function* yang dibutuhkan untuk membangun agen yang dapat menghindari tabrakan hanya dengan input visual yang ditingkatkan menggunakan Teknik *Double DQN*

A. Identifikasi Masalah

Pada tahap ini, tujuan utama adalah memilih topik yang akan diteliti dan mengidentifikasi masalah yang ada dalam topik tersebut. Pada tahap ini peneliti tertarik mengembangkan teknik-teknik yang dapat memperbaiki kelemahan dari kemampuan pelatihan DQN pada konteks *reinforcement learning* berbasis visual.

B. Perancangan State

State merupakan representasi dari kondisi lingkungan yang digunakan oleh agen untuk mengambil keputusan pada setiap langkah interaksi. Dalam environment Enduro-v5, representasi *state* tidak berasal dari sensor dunia nyata seperti posisi kendaraan, kecepatan, atau jarak terhadap rintangan, melainkan berasal dari citra layar (frame visual) yang secara langsung mencerminkan kondisi lingkungan permainan. Pada tahap ini, frame (gambar) diambil langsung dari *environment* Atari Enduro yang disediakan oleh Gymnasium (*Atari Learning Environment*). Gambar hasil tangkapan tersebut merepresentasikan tampilan visual dari permainan secara penuh, termasuk jalan, kendaraan lain, cuaca, serta indikator skor. Agar gambar dapat digunakan sebagai input untuk model, dilakukan proses prapemrosesan visual. Adapun tahapan pra-pemrosesan dalam penelitian ini adalah sebagai berikut:

1. Konversi ke *grayscale*
   Gambar berwarna (RGB) dikonversi menjadi grayscale (1 kanal) untuk mengurangi kompleksitas komputasi, karena informasi warna tidak penting dalam konteks permainan ini.

2. Ubah ukurna gambar menjadi 84x84 piksel
   Ukuran asli frame Atari adalah 210x160 piksel, namun untuk efisiensi dan kompatibilitas dengan arsitektur CNN standar, gambar diubah menjadi 84x84 piksel.

3. Frame stacking sebanyak 4 frame terakhir
   Karena satu frame statis tidak mencerminkan dinamika (misalnya arah gerak kendaraan), maka digunakan 4 frame terakhir secara berurutan (*stacked*) sebagai satu state. Ini memungkinkan model menangkap informasi gerakan atau kecepatan relatif objek dalam permainan.

Dengan demikian, state akhir direpresentasikan sebagai array berdimensi (84, 84, 4), di mana 4 adalah jumlah frame yang distack (bukan kanal warna RGB). Setiap frame dalam stack merupakan gambar grayscale.

C. Perancangan Aksi

Aksi yang dapat dilakukan oleh agen pada lingkungan Enduro-v5 dirancang untuk mendukung kemampuan navigasi dan penghindaran tabrakan secara optimal dalam konteks permainan balap mobil 2D. Berbeda dengan simulasi kendaraan dunia nyata seperti pada Carla yang memungkinkan kontrol kontinu seperti throttle dan brake, Enduro menyediakan aksi dalam bentuk diskret, sebagaimana yang tersedia dalam lingkungan Atari aslinya.

Dalam penelitian ini, agen memiliki beberapa opsi aksi diskret yang mencerminkan kontrol sederhana khas permainan Atari, yaitu:

a. 0 — No Operation: Tidak melakukan apa-apa (diam)
b. 1 — Accelerate: Mempercepat mobil
c. 2 — Steer Left: Membelokkan mobil ke kiri
d. 3 — Steer Right: Membelokkan mobil ke kanan

Dengan menggunakan ruang aksi diskret seperti di atas, agen belajar untuk memilih kombinasi tindakan terbaik berdasarkan informasi visual yang diterimanya dari layar permainan (frame stack). Strategi optimal dicapai ketika agen mampu mempercepat dan bermanuver untuk menyalip kendaraan lain serta menghindari tabrakan dalam kecepatan tinggi, terutama di area lalu lintas yang padat atau saat kondisi visual berubah (seperti kabut dan malam hari).

D. Perancangan reward function

Pada penelitian ini, reward function difokuskan pada pencapaian progress dan penghindaran tabrakan dalam permainan Enduro-v5, yang merepresentasikan situasi lalu lintas dinamis dengan kecepatan tinggi dan kepadatan kendaraan.

Berikut adalah desain fungsi reward yang digunakan:

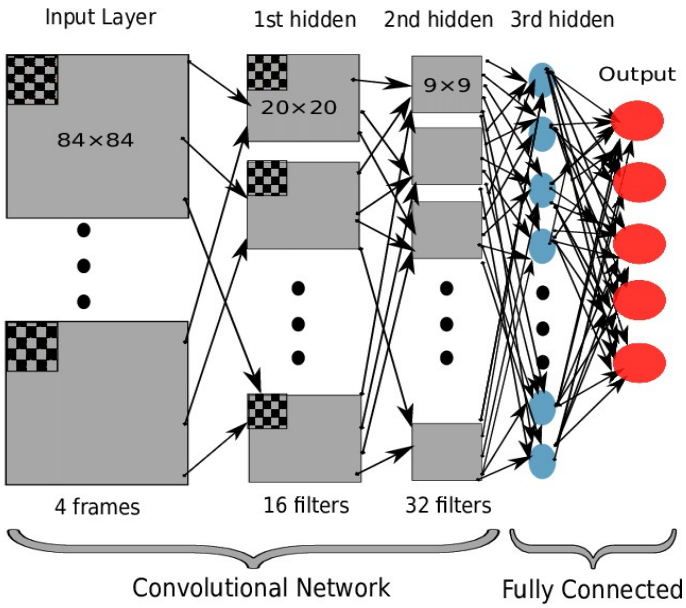
1. Reward positif diberikan saat agen berhasil menyalip kendaraan lain (*overtake*). Setiap mobil yang berhasil disalip akan memberikan *reward* sebesar +1, karena ini menunjukkan kemajuan dan efisiensi berkendara.
2. Penalti besar diberikan saat agen mengalami tabrakan. Ketika mobil agen menabrak kendaraan lain atau menyentuh batas jalan hingga simulasi selesai, agen mendapatkan *reward* sebesar -100 sebagai bentuk hukuman agar belajar menghindari tabrakan.
3. Reward kecil diberikan saat agen terus bergerak aktif (tidak diam terlalu lama). Jika agen tetap bergerak (bergerak secara visual), namun tidak menyalip kendaraan, tetap diberikan *reward* +0.1 agar menghindari perilaku pasif.
4. Penalti kecil diberikan jika agen diam terlalu lama atau tidak ada progress signifikan. Untuk mencegah perilaku diam atau tidak aktif, diberikan *reward* -1 jika dalam periode tertentu tidak ada overtake yang terjadi.

Reward function ini dirancang untuk mendorong agen bergerak aktif, menyalip kendaraan, dan bertahan hidup selama mungkin tanpa tabrakan dalam lalu lintas yang semakin padat dan cepat.

E. Perancangan model pembelajaran

Arsitektur CNN (*Convolutional Neural Network*) yang digunakan dalam penelitian ini mengacu pada struktur jaringan standar yang digunakan oleh Stable-Baselines3 (SB3) dalam menangani lingkungan berbasis citra, seperti Atari Enduro-v5**.** CNN ini bertugas mengekstraksi fitur visual dari input *frame stack* yang merepresentasikan kondisi lingkungan permainan secara visual. Input yang digunakan berupa stack 4 frame *grayscale* hasil prapemrosesan dari citra permainan, masing-masing berukuran 84 piksel x 84 piksel**.** Jadi, dimensi input-nya adalah (84, 84, 4)**.** rsitektur jaringan terdiri dari beberapa lapisan konvolusional yang dilanjutkan dengan lapisan fully connected untuk memprediksi nilai Q (*Q-values*) dari setiap aksi diskret. Rincian arsitekturnya adalah sebagai berikut:

1. Input: Stack 4 frame grayscale ukuran 84×84, sama persis dengan yang digunakan di Enduro-v5.
2. 3 Lapisan Konvolusional:
   a. Conv1: 32 filter, kernel 8×8, stride 4 → output fitur sekitar 20×20
   b. Conv2: 64 filter, kernel 4×4, stride 2 → output sekitar 9×9
   c. Conv3: 64 filter, kernel 3×3, stride 1 → output mendalam fitur map
3. Flatten: Mengubah output konvolusi menjadi vektor 1-dim.
4. Fully Connected (FC): Layer dengan 512 neuron, menggunakan aktivasi ReLU.
5. Output Layer: Menghasilkan *Q-value* untuk setiap aksi diskret (4 aksi: no-op, accelerate, left, right).



Gambar 1 Model Pembelajaran

F. Perancangan Algoritma

Pada tahap awal, digunakan metode *experience replay* yang melibatkan penyimpanan pengalaman agent pada setiap waktu dalam dataset $D_t = \{e_1,...,e_t\}$, di mana pengalaman ini direpresentasikan oleh rumus $e_t = (s_t, a_t, r_t, s_{t+1})$. Selanjutnya, dilakukan inisialisasi nilai action – value(Q) dengan menggunakan bobot acak (θ). Selain itu, juga dilakukan inisialisasi target action – value(Q) dengan menggunakan bobot (θ - = θ). Selama proses pembelajaran, nilai Q diperbarui dengan menggunakan sampel atau *minibatch* dari *experience* yang dituliskan dengan (s, a, r, s'), yang diambil secara acak dari kumpulan sampel. Algoritma ini menyimpan *tuple experience*(N) terakhir ke dalam replay memory, dan kemudian sampel dipilih secara acak dari dataset(D). Selanjutnya, untuk meningkatkan kestabilan metode, digunakan juga jaringan saraf terpisah dalam neural network untuk menghasilkan target y_j dalam proses pembaruan nilai Q. Lalu, dilakukan perhitungan loss untuk memperbaharui *neural network*, setiap pembaharuan pada langkah C, akan dikloningkan dengan q – network untuk mendapatkan target network (Q). Algoritma Double DQN dikembangkan untuk mengatasi masalah *overestimation* pada DQN klasik, yaitu saat agent terlalu optimis dalam memperkirakan nilai Q. Pada DQN standar, pemilihan dan evaluasi aksi maksimum dilakukan oleh jaringan yang sama, yang menyebabkan bias. Untuk mempercepat proses pembelajaran dan memperkaya informasi *reward*, digunakan pendekatan *Multi-Step DQN*. Alih-alih hanya menggunakan reward satu langkah ke depan, pendekatan ini menghitung akumulasi *reward* dari langkah berikutnya. Targetnya ditulis sebagai:

G. Perancangan Simulasi

Dalam penelitian ini, *environment* simulasi yang digunakan adalah *Atari Enduro-v5* dari *Gymnasium/Atari Learning Environment (ALE)*, yang merepresentasikan skenario berkendara di jalan raya dengan kecepatan tinggi. Environment ini menampilkan tampilan visual berbasis piksel dari perspektif orang pertama, menyerupai pengalaman mengemudi dalam kondisi lalu lintas padat.

Environment terdiri dari:

a. Objek Statis: Jalan raya, marka jalan, serta elemen latar belakang seperti langit, pegunungan, dan garis batas lintasan. Objek-objek ini memberikan konteks visual yang membantu agent memahami arah pergerakan dan batas jalur.

b. Objek Dinamis: Mobil-mobil NPC yang bergerak secara acak pada jalur yang berbeda. Mobil-mobil ini menjadi rintangan utama yang harus dihindari oleh agent untuk memaksimalkan reward dan menghindari tabrakan.

Agent dilatih untuk:

a. Menyalip kendaraan lain dengan berpindah jalur (belok kiri/kanan),

b. Menghindari tabrakan dengan kendaraan di depannya,

c. Mempertahankan kecepatan optimal, karena semakin jauh jarak tempuh dan semakin banyak mobil yang disalip, reward akan semakin tinggi.

Pra pemrosesan Input Visual

Untuk meningkatkan efisiensi pembelajaran dan mengurangi kompleksitas data input, digunakan teknik preprocessing sebagai berikut:

1. Grayscale Conversion:

   Citra dari lingkungan awalnya berupa RGB. Untuk menyederhanakan representasi dan mengurangi dimensi input, citra dikonversi menjadi *grayscale*. Ini mengurangi input dari (84, 84, 3) menjadi (84, 84, 1).

2. Pengubahan Ukuran:
   Ukuran asli gambar diperkecil menjadi 84x84 piksel agar sesuai dengan arsitektur *Nature CNN* dan mempercepat komputasi.

3. *Frame Stacking* (4 frame):

   Agar agent dapat memahami dinamika pergerakan objek (misalnya kecepatan dan arah mobil di depannya), digunakan teknik *frame stacking* sebanyak 4 frame terakhir. Hasilnya adalah input dengan bentuk (84, 84, 4), di mana setiap kanal mewakili satu waktu/frame. Ini memungkinkan agent memahami perubahan antar frame sebagai bentuk "memori jangka pendek".

H. Pelatihan Agent

Agen dilatih berbasis Deep Q – learning(DQN) yaitu algoritma *Deep Reinforcement Learning* yang menggabungkan Q – learning dengan neural network yang bertujuan untuk meningkatkan efisiensi pembelajaran dan menangani lingkungan yang kompleks. DQN memiliki fitur penyimpanan experience yaitu replay memory/replay buffer, di mana agent menyimpan experience dalam *replay memory* yang didefinisikan sebagai susunan tuple dan dituliskan sebagai berikut (s , a , r , s), experience ini digunakan untuk latih ulang kembali nantinya. Namun replay memory yang ada pada DQN sering kali mengulang experience dengan value yang kurang maksimal mengakibatkan value akumulasi reward tidak maksimal [6]. DQN menggunakan *neural netwok* untuk memprediksi nilai tindakan terbaik untuk setiap keadaan. Setelah memilih tindakan, model melakukan action pada environment dan menerima reward. Kemudian, agen memperbarui dua jaringan saraf, yaitu neural network dan target network. Pembaruan ini dilakukan dengan menggunakan persamaan Bellman, yang membandingkan nilai prediksi dari target network dengan reward aktual yang diterima. Persamaan Bellman mengasumsikan bahwa agen akan selalu memilih tindakan terbaik yang diketahuinya di masa depan. Berikut rumus dari persamaan Bellman yang dirumuskan pada persamaan 3.1 [7].

$$Q(S_t, A_t) = (1 - a)\, Q(S_t, A_t) + a * \left(R_t + \lambda * max_a\, Q(S_{t+1}, a)\right)$$

Dimana :
S : Keadaan/kondisi yang dihadapi agent.
A : Action yang diambil agent dari hasil pemetaan environment.
R : Reward dari action yang dilakukan oleh agent.
t : Tahapan waktu.
α : Tahapan pembelajaran.
ƛ : Faktor diskon, digunakan untuk menyeimbangkan nilai reward di masa depan dan reward di masa sekarang.

Prioritized Experience Replay(PER) merupakan teknik yang diperkenalkan oleh Schaul et.al pada jurnal mereka "*Prioritized Experience Replay*". Teknik ini mengatasi permasalahan iterasi pengalaman yang tidak efisien dalam RL, di

mana pengalaman penting mungkin saja tidak diulang, menyebabkan pembelajaran lambat atau tidak maksimal. PER bekerja dengan memilih sampel dengan error terbesar dengan harapan *neural network* dapat meminimalkannya [8].

Oleh karena itu, pemilihan sampel dengan TD error terbesar sehingga neural network dapat meminimalkannya untuk meningkatkan kemampuan generalisasi pada model.

$$\delta_i = r_t + \underset{a \in A}{\gamma max} Q_{\theta^-}(s_{t+1}, a) - Q_\theta(s_t, a_t)$$

Sehingga *experience* yang sebelumnya dituliskan dengan (st, at, rt, st+1), setelah ditambahkan error menjadi (st, at, rt, st+1, δi). Dalam melakukan pendistribusian probability untuk sampling menggunakan persamaan 3.3 berikut ini [9].

$$P(i) = \frac{p_i^a}{\sum_k p_k^a}$$

Namun, dalam *Prioritized Replay*, terdapat bias karena pengambilan sampel tidak dilakukan secara acak, melainkan berdasarkan proporsi kesalahan TD – Error. Oleh karena itu, akan dilakukan perbaikan terhadap bias ini menggunakan bobot *Importance Sampling* seperti yang di rumuskan pada persamaan 3.4 [9].

$$w_i = \left(\frac{1}{N} . \frac{1}{P(i)}\right)^\beta$$

Yang bertujuan untuk menkompensasi *non – uniform probabilities*, ketika ß = 0, tidak ada proses koreksi sama sekali, dan sebaliknya, ketika ß = 1 akan memberikan koreksi pada setiap bias.

Untuk mempercepat konvergensi dan memungkinkan agent belajar dari konsekuensi jangka menengah, algoritma ini juga mengimplementasikan *Multi-Step DQN*. Tidak seperti DQN standar yang hanya mempertimbangkan reward satu langkah ke depan, Multi-Step DQN menghitung akumulasi reward selama n langkah ke depan, yang dituliskan sebagai berikut:

$$R_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k}$$

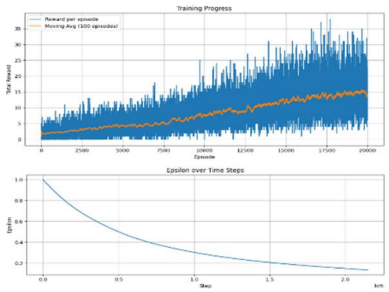
Transisi yang disimpan dalam buffer kemudian menjadi:

$$(s_t, a_t, R_t^{(n)}, s_{t+n})$$

Dengan menggunakan akumulasi *reward* dari beberapa langkah ke depan, target Q menjadi lebih stabil dan representatif, sehingga mengurangi delay antara aksi dan *reward* yang dapat terjadi pada lingkungan seperti Enduro. Ini juga membantu mengurangi *overestimation* bias dan mempercepat pembelajaran terutama pada awal pelatihan.

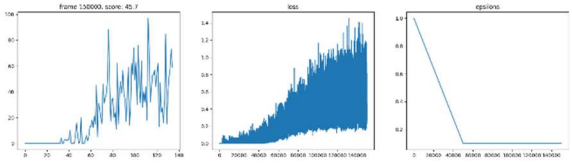
III. HASIL DAN PEMBAHASAN

Bagian ini menyajikan hasil pengujian dari pelatihan agent. Model yang dihasilkan dari proses *training* kemudian diuji dengan tujuan untuk menilai kinerja model terhadap data yang belum pernah dilihat sebelumnya. Pada pengujian pertama akan dilakukan dengan kondisi environment yang sama seperti pada proses training, model akan diletakan sejalur dengan obstacle, dengan harapan model dapat melakukan proses pengereman sebelum terjadinya tabrakan.



Gambar 2 Model Pembelajaran

Berdasarkan Gambar 2, pada tahap awal pengujian eksperimen ini, model yang dipilih sebagai model terbaik adalah model pada timestep ke-1.800.000. Model ini menunjukkan performa *reward* yang stabil dan tinggi, serta nilai epsilon yang telah mendekati nilai minimumnya, yang menandakan bahwa proses eksploitasi telah lebih dominan dibanding eksplorasi. Grafik *reward* menunjukkan bahwa nilai reward per episode meningkat secara konsisten hingga mencapai lebih dari 30 di beberapa episode akhir. Sementara itu, garis *moving average* selama 100 episode menunjukkan tren kenaikan yang stabil, yang menandakan bahwa agen berhasil belajar strategi yang efektif dari lingkungan. Grafik kedua (*epsilon decay*) memperlihatkan nilai epsilon yang menurun secara gradual hingga mendekati nol, menandakan bahwa agen telah melewati fase eksplorasi dan bertransisi penuh ke eksploitasi kebijakan yang telah dipelajari. Dalam penelitian ini, evaluasi terbatas pada hasil pelatihan berupa grafik *reward* dan nilai epsilon. Belum dilakukan evaluasi eksplisit terhadap kemampuan agen dalam skenario pengereman tertentu. Oleh karena itu, diperlukan pengujian lanjutan di simulator untuk mengonfirmasi kemampuan agent secara lebih mendalam terhadap aksi-aksi kritis seperti pengereman.



Gambar 3 Model Pembelajaran

Seperti yang digambarkan pada gambar 3, hasil dari pengujian eksperimen tahap kedua dengan menggunakan model yang sama hanya saja ditambah multistep DQN. pengujian tahap pertama, menunjukkan dari *Reward* per episode meningkat dan stabil, ditambah garis *moving average* → menandakan pembelajaran efektif dan konvergen dan pengujian tahap kedua juga sudah cukup baik *Reward* meningkat pesat seiring waktu Skor mulai naik perlahan setelah episode ke-40, dan meningkat drastis hingga mencapai 80–90 di beberapa episode terakhir. → agen belajar dengan baik, tren naik perlahan dan stabil dari awal hingga 150.000 langkah dan terakhir Model eksplorasi secara maksimal di awal (ε=1), lalu mengeksploitasi setelah cukup belajar. Nilai akhir 0.1 masih menjaga sedikit eksplorasi, artinya eksplorasi berjalan sesuai rencana di posisi yang sama saat ada rintangan, dapat disimpulkan bahwa model kedua hasilnya cukup baik sama dengan pengujian yang pertama.

IV. KESIMPULAN

Berdasarkan pengujian, pengamatan, dan analisis hasil yang diperoleh sejauh ini, maka dapat ditarik beberapa kesimpulan sebagai berikut:

1. Algoritma DQN dasar menunjukkan performa awal yang cukup baik dalam pembelajaran navigasi visual, namun mengalami keterbatasan dalam efisiensi pembelajaran dan kesulitan dalam mencapai generalisasi yang stabil.

2. Penggunaan Prioritized Experience Replay (PER) memberikan dampak positif terhadap efisiensi pembelajaran, tetapi hasil yang diperoleh masih belum optimal karena sensitivitas terhadap pemilihan hyperparameter seperti alpha dan beta. Hal ini menunjukkan bahwa tuning parameter yang lebih tepat masih diperlukan untuk memaksimalkan manfaat dari PER.
3. Penerapan *Multi-Step DQN* dan *Double DQN* memberikan peningkatan signifikan dalam stabilitas dan kecepatan konvergensi model. Multi-step return membantu agent memperoleh sinyal reward yang lebih informatif dalam jangka pendek, sehingga mempercepat pembelajaran pada episode-episode awal. Sementara itu, Double DQN mengurangi overestimation bias terhadap nilai Q, yang sering terjadi pada DQN klasik.
   Algoritma ini diimplementasikan secara manual menggunakan PyTorch tanpa menggunakan pustaka seperti Stable-Baselines3, agar memberikan fleksibilitas penuh dalam memodifikasi struktur jaringan, mekanisme multi-step, dan logika pemilihan aksi. Sementara itu, Stable-Baselines3 (SB3) digunakan dalam eksperimen terpisah yang memanfaatkan algoritma PPO, bukan DQN. SB3 menyediakan implementasi PPO yang efisien, stabil, dan mendukung integrasi dengan sistem logging dan evaluasi otomatis, namun tidak digunakan dalam eksperimen utama berbasis DQN ini.
4. Berdasarkan hasil eksperimen pada model terbaik di timestep ke-1.800.000, reward menunjukkan kenaikan yang konsisten dengan nilai epsilon yang mendekati minimum, menandakan bahwa proses eksploitasi telah dominan. Model mampu melakukan navigasi dan pengambilan keputusan pengereman secara tepat pada mayoritas episode, meskipun masih ditemukan beberapa anomali kecil pada skenario tertentu.

Referensi

[1] Agarwal, Rishabh, Dale Schuurmans, and Mohammad Norouzi. "An optimistic perspective on offline reinforcement learning." *International conference on machine learning*. PMLR, 2020.
[2] Fan, J., Wang, Z., Xie, Y., & Yang, Z. (2020, July). A theoretical analysis of deep Q-learning. In *Learning for dynamics and control* (pp. 486-489). PMLR.
[3] Fildes, B., Keall, M., Bos, N., Lie, A., Page, Y., Pastor, C. & Tingvall, C. (2015). Effectiveness of low speed autonomous emergency braking in real-world rear-end crashes. *Accident Analysis & Prevention*, *81*, 24-29.B.
[4] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.
[5] Pokrzywa, Jack. "SAE global ground vehicle standards." *SAE International* (2014).
[6] Quek, Y. T., Koh, L. L., Koh, N. T., Tso, W. A., & Woo, W. L. (2021). Deep Q-network implementation for simulated autonomous vehicle control. *IET intelligent transport systems*, *15*(7), 875-885.
[7] Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.
[8] Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.
[9] S. Ruder, “An overview of multi-task learning in deep neural networks,” ArXiv Prepr. ArXiv170605098, 2017.
[10] U. R. Beierholm, C. Anen, S. Quartz, and P. Bossaerts, “Separate encoding of model-based and model-free valuations in the human brain,” Neuroimage, vol. 58, no. 3, pp. 955–962, 2011.