Pasinformatik, Volume 3, Issue 1, Januari 2024
E-ISSN : 2986-5360

Pengembangan Automatic Emergency Braking (AEB) pada Autonomous Driving Menggunakan Deep Q – Learning dengan PER

Kevin Anggara Putra[1] , Handoko Supeno [2]

Jurusan Teknik Informatika, Fakultas Teknik, Universitas Pasundan
Jln. Dr. Setiabudhi no. 193 Bandung, Jawa Barat
[1] vinputra.16@gmail.com , [2] handoko@unpas.ac.id

Abstrak : Penelitian ini membahas tentang kemampuan kendaraan otonom untuk menghindari rintangan, yang merupakan salah satu fitur penting dari sistem navigasi kendaraan otonom. Perancangan sistem pengereman darurat otomatis secara tradisional akan sulit karena membutuhkan biaya yang besar, kompleksitas, dan pertimbangan keamanan. Reinforcement Learning (RL) adalah salah satu metode kecerdasan buatan atau pembelajaran mesin yang dapat belajar langsung dari lingkungan. Penelitian ini menggunakan algoritma RL dengan metode deep q – learning (DQN) dan dioptimalkan dengan prioritized experience replay untuk menghindari rintangan. Simulasi dilakukan pada jalur dengan rintangan yang diposisikan satu jalan dengan model pada simulator CARLA.

Kata Kunci : Kendaraan otonom, pembelajaran penguatan mendalam, pengereman otomatis, *Deep q – learning (DQN)*, *Prioritized Experience Replay*.

I. PENDAHULUAN

Pada era saat ini ada pengembangan teknologi baru yang sedang marak dikembangkan yaitu kendaraan otonom atau *Autonomous Vehicle* (AV) sesuai dengan namanya merupakan kendaraan yang memiliki kemampuan untuk melakukan navigasi dengan mengamati environment sekitarnya dan memahaminya tanpa input atau pengoperasian manual dari manusia dengan tingkat yang sama seperti dikemudikan oleh manusia [1]. Fitur dari AV yang sedang banyak dikembangkan adalah sistem navigasi, yang merupakan bagian penting dari pengoperasian AV itu sendiri, dengan satu aspek kunci adalah kemampuan menghindari rintangan. Menurut Siegwart, sistem navigasi terdiri dari kemampuan pengambilan data sensor, pemetaan lingkungan, kognisi pengambilan keputusan, dan kontrol Gerakan [2]. Oleh karena itu, kemampuan ini sangat vital bagi AV untuk mengurangi risiko kerusakan yang serius saat terjadi kecelakaan. Dikutip dari Kominfo.go.id, kelalaian manusia (human error) menjadi faktor dominan penyebab kecelakaan lalu lintas khususnya di Indonesia. Menurut data kepolisian di Indonesia pada tahun 2017. Direktur Jenderal Perhubungan Darat, Pudji Hartanto, menyampaikan bahwa mayoritas kecelakaan lalu lintas, sebanyak 61%, disebabkan oleh faktor manusia yang meliputi kemampuan dan karakter pengemudi. Selain itu, sekitar 9% kecelakaan disebabkan oleh kondisi kendaraan yang terkait dengan pemenuhan persyaratan teknis yang sesuai dengan standar keselamatan jalan. Sedangkan sekitar 30% kecelakaan disebabkan oleh faktor prasarana dan lingkungan [3]. Penerapan *Machine Learning* (ML) dalam pengembangan kendaraan otonom memiliki dampak yang besar. *Reinforcement Learning* (RL) merupakan salat satu cabang pembelajaran dari ML yang bekerja degan cara model belajar melalui *trial and error*, di mana *agent*, mengambil tindakan untuk mencapai tujuan tertentu dan menerima imbalan(*reward*) atau hukuman(*punishment*) sebagai *feedback* atas tindakan yang diambil. Algoritma *Deep Q – Learning*(DQN) adalah salah satu teknik RL yang memanfaatkan jaringan saraf tiruan(*artificial neural network*) untuk menentukan keputusan terbaik yang harus diambil [4]. Secara teoritis, meskipun DQN telah menunjukkan kecanggihan, masih diperlukan penelitian lebih lanjut untuk memastikan keandalan teknologi tersebut. Dalam beberapa kasus, DQN memerlukan sejumlah pengalaman yang signifikan untuk mencapai hasil optimal. Agar dapat mengatasi masalah ini, digunakan metode *Prioritized Experience Replay*(PER) untuk mengatasi kekurangan tersebut. Hal ini dilakukan dengan memberikan bobot yang lebih besar pada pengalaman yang berpengaruh pada pembelajaran, sehingga pengalaman tersebut memiliki peluang lebih besar untuk dipelajari kembali oleh model. Dengan menggunakan PER, *agent* dapat memperkuat ingatan dan menggunakan kembali pengalaman yang berguna sebelumnya di mana transisi yang diamati kemudian disimpan dan diambil sampel secara terurut untuk memperbarui jaringan berikutnya [5]. Tentu pengembangan model pada AV merupakan langkah yang besar, namun, terdapat beberapa limitas atau keterbatasan pada pengembangannya, dikarenakan mustahil untuk menguji algoritma pada dunia nyata dikarenakan faktor biaya, kompleksitas, dan keamanan. Dalam mewujudkan sistem akan digunakan simulator yang mampu merepresentasikan dunia nyata, juga memiliki kemudahan perangkat lunak dan integrasi perangkat keras seperti pada Simulator CARLA. Dibekali dengan kualitas grafis yang memadai dan juga memiliki banyak fitur serta fasilitas yang mendukung untuk proses training maupun testing.

II. METODE PENELITIAN

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

1. Perancangan state
2. Perancangan aksi
3. Perancangan reward function
4. Perancangan model pembelajaran
5. Perancangan algoritma
6. Perancangan simulasi
7. Pelatihan agent
8. Evaluasi agent.

Hasil penelitian ini akan memberikan kontribusi berupa algoritma dan reward function yang dibutuhkan untuk membangun agen yang dapat melakukan *Automatic Emergency Braking* (AEB) hanya dengan input visual yang ditingkatkan menggunakan Teknik memory *Prioritized Experience Replay* (PER)

A. Perancangan State

Pada tahap ini gambar ditangkap oleh kamera lalu diubah menjadi *state*. Gambar yang diperoleh memiliki ukuran tinggi 250 piksel dan lebar 250 piksel, serta memiliki 4 nilai warna, yaitu RGB (Red, Green, Blue). Dalam penelitian ini, aspek A atau Alpha tidak digunakan. Proses prapemrosesan terdiri dari tiga langkah:

1. Mengubah gambar menjadi array data informasi dengan tipe data float. Langkah ini dilakukan untuk mempermudah pengolahan data.
2. Mengubah ukuran array menjadi (im_width, im_height, 4). Ukuran ini mewakili bentuk gambar dan nilai warna RGBA.
3. Menghilangkan kanal transparansi (alpha) dari array. Kanal alpha tidak digunakan dalam penelitian ini karena tidak ada kebutuhan untuk menampilkan gambar dengan latar belakang transparan.

Informasi *state* diperoleh dari informasi gambar yang diambil kamera depan agent saat pelatihan. Gambar tersebut memiliki ukuran tinggi 250 piksel dan lebar 250 piksel, serta memiliki 4 nilai warna, yaitu RGB (Red, Green, Blue). Dalam penelitian ini, aspek A atau Alpha tidak digunakan.

B. Perancangan Aksi

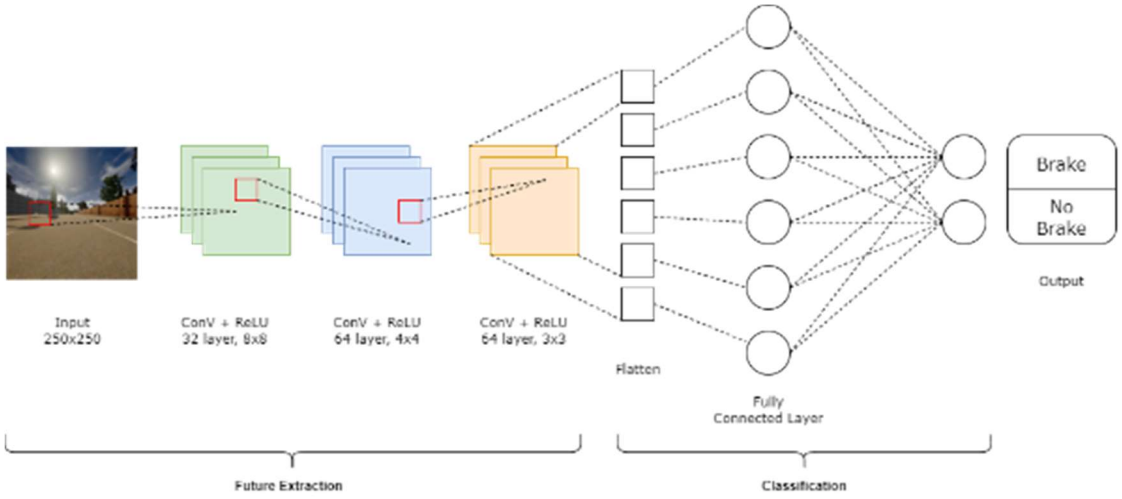
Dalam penelitian ini, terdapat dua jenis aksi yang dirancang, yaitu pengereman (brake) dan percepatan (throttle). Nilai aksi pengereman diatur sebagai variabel kontinu dengan rentang antara 0.0 hingga 1.0, sedangkan aksi percepatan nilainya tetap pada 0.5.

C. Perancangan reward function

Pada penelitian ini *reward function* akan difokuskan terhadap pengerapan dan menghindari tabrakan. Ketika *agent* sudah dapat berjalan diatas 20 km/h maka akan diberikan *reward* +1, ketika *agent* mengalami tabrakan, maka akan diberikan *reward* -100 sebagai punishment, ketika *agent* sudah mendekati objek tapi kecepatan masih diatas 0 (km>0), maka akan diberikan *reward* -1, ketika *agent* berhasil berhenti pada jarak aman akan diberikan *reward* +2, tapi jika *agent* berhenti tanpa mendeteksi adanya *obstacle* maka akan diberikan *reward* -1.

D. Perancangan model pembelajaran

Secara default lapisan hidden layer CNN pada stable – baselines3 seperti yang ditampilkan pada gambar 1 terdiri dari: *hidden layer* pertama yang menggunakan 32 filter berukuran 8x8, filter ini berinteraksi dengan gambar input dan menerapkan ReLU, untuk memperkenalkan non – linearitas yang bertujuan agar pembelajaran yang lebih baik. *Hidden layer* kedua beroperasi dengan menggunakan 32 saluran input (output dari lapisan sebelumnya), 64 filter berukuran 4x4. Filter ini juga menerapkan ReLU. *Hidden layer* ketiga beroperasi dengan menggunakan 64 saluran masukan (output dari lapisan sebelumnya), 64 filter berukuran 3x3. Filter ini juga menerapkan ReLU. Lalu dilanjutkan dengan lapisan *Flatten* untuk mengubah bentuk output menjadi vektor 1D, dan diakhiri pada lapisan *fully connected* mengambil *output* dan menggunakannya untuk memprediksi *Q – value*.



Gambar 1 Model Pembelajaran

E. 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 1. Deep Q Learning

Algorithm : Deep Q – learning with experience replay.

Initialize replay memory D to capacity N
Initialize action – value function Q with random weights θ
Initialize target action – value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For t = 1, T **do**
    With probability ε select a random action a_t
    Otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi = \phi(s_{t+1})$
    Store transition (s_t, a_t, r_t, s_{t+1}) in D
    Sample random minibatch of transitions (s_t, a_t, r_t, s_{t+1}) from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$ with respect to the Network parapemeters θ
    Every C steps reset $\hat{Q} = Q$
  End for
End for

Algoritma 2. Prioritized Experience Replay

Algorithm Double DQN with Prioritized Experience Replay (PER)

Input: minibatch k, step-size η, replay period K and size N, exponents α and β, budget T.
Initialize replay memory H = Ø, Δ = 0, p1 = 1
Observe S0 and choose A0 ~ πθ(S0)
for t = 1 **to** T **do**
  Observe St, Rt, γt
  Store transition (St−1, At−1, Rt, γt, St) in D with maximal priority pt = maxi<t pi
  If t = 0 mod K **then**
    for j = 1 to k **do**
      Sample transition $j \sim P(j) = p_j^a / \sum_i p_i^a$
      Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / max_i w_i$
      Compute TD – Error $\delta_i = R_j + \underset{a \in A}{\gamma max} Q_{\theta^-}(s_{t+1}, a) - Q_\theta(s_t, a_t)$
      Update transition priority $p_j \leftarrow |\delta_j|$
      Accumulate weight-change $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$
    end for
    Update weight $\Delta \leftarrow \Delta + \eta \cdot \Delta, reset\ \Delta = 0$
    From time to time copy weights into target network $\theta_{target} \leftarrow \theta$
  end if
  Choose action At ~ πθ(St)
end for

Prosedur PER diawali dengan memberikan prioritas kepada pengalaman yang diukur dengan besarnya temporal – difference error. Selanjutnya akan memperbaiki bias yang mana dihasilkan dari proses prioritized replay menggunakan importance sampling. Dalam konteks ini, parameter Alpha(α) digunakan untuk mengontrol tingkat prioritisas dalam algoritma. Ketika α = 0, tidak ada prioritisasi yang diterapkan, sehingga penyampelan data dilakukan secara acak tanpa mempertimbangkan tingkat kesalahan. Sebaliknya, ketika α = 1, terjadi prioritisasi penuh di mana data dengan nilai δi yang lebih tinggi akan memiliki bobot yang lebih besar dalam proses penyampelan. Dengan kata lain, data yang memiliki kesalahan(δi) yang tinggi akan lebih sering dipilih untuk dipelajari oleh algoritma. Beta(ß) adalah parameter yang akan mengukur seberapa besar bobot kita harus mengoreksi bias yang dihasilkan dari teknik penentuan prioritas, Ketika ß = 0, tidak ada proses koreksi sama sekali, dan sebaliknya, ketika ß = 1 akan memberikan koreksi pada setiap bias.

F. Perancangan Simulasi

Dalam penelitian ini, *environment* simulasi dibuat untuk menyerupai lingkungan perkotaan (urban). *Environment* dilengkapi dengan objek statis seperti jalan raya, lampu jalan, trotoar, dan bangunan. Objek statis ini membantu *agent* untuk memahami *layout environment*. Selain objek statis, *environment* juga dilengkapi dengan objek dinamis seperti *agent* dan mobil NPC. Mobil NPC berperan sebagai rintangan yang dapat bergerak. *Agent* harus belajar untuk menghindari mobil NPC ini untuk menghindari tabrakan.
.

G. Pelatihan Agen

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_t, a_t, r_t, s_{t+1}), *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].

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

Persamaan 1

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 maskimal. PER berekerja 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.

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

Persamaan 2

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].

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

Persamaan 3

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].

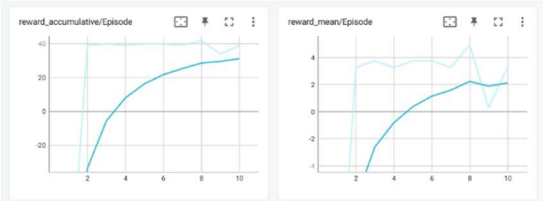
$$w_i = \left(\frac{1}{N} . \frac{1}{P(i)}\right)^\beta$$

Persamaan 4

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.

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.

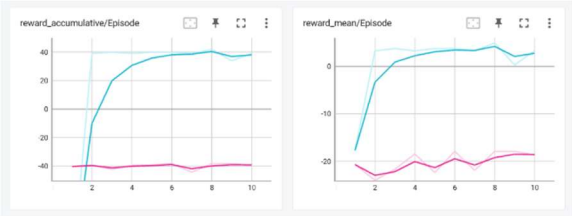


Gambar 2 Model Pembelajaran

Tabel 1 Hasil pengujian tahap pertama

Nama	Result
Model 651000	9/10

Berdasarkan pada gambar 2 dan tabel 1, pada tahap awal pengujian eksperimen ini, model yang dipilih adalah model yang memiliki kinerja terbaik, yaitu model pada timestep 651000. Model ini memiliki rata – rata *reward* yang tinggi dan telah mencapai konvergensi. Dalam 10 episode pengujian, model berhasil melakukan pengereman 9 kali dari 10 kali percobaan. Hal ini menunjukkan bahwa model dapat melakukan pengereman dengan baik. Pada pengujian kedua akan dilakukan dengan kondisi environment yang berbeda, objek *obstacle* akan dihilangkan dari jalur, dengan harapan model benar – benar berhenti karena mendeteksi objek di depannya bukan dikarenakan perhitungan jarak ataupun karena objek lain disekitarnya.



Gambar 3 Model Pembelajaran

Tabel 2 Hasil pengujian tahap kedua

Nama	Result
Model 651000	0/10

Seperti yang digambarkan pada gambar 3 dan tabel 2 hasil dari pengujian eksperimen tahap kedua yang menggunakan model yang sama dengan pengujian tahap pertama, menunjukkan dari 10 episode pengujian, model melakukan 10 kali pengereman di posisi yang sama saat ada rintangan, yang dapat disimpulkan bahwa model tidak berhenti karena mendeteksi adanya rintangan saja, melainkan juga karena dari kondisi lingkungan sekitar yang menyebabkan pengereman terlalu sensitif.

IV. KESIMPULAN

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

1. Algoritma DQN dan PER membutuhkan waktu lebih lama untuk belajar dibandingkan algoritma DQN saja. Hal ini dikarenakan masih belum menemukan *hyperparameter* yang cocok dan perlu pengkajian ulang terkait hal tersebut.
2. Algoritma DQN menggunakan *framework* stable – baseline3 menunjukkan kinerja yang sangat baik. Model mampu berhenti dengan baik tanpa menabrak. Namun, pada tahap pengujian, ditemukan anomali pada pengujian tahap kedua. Model berhenti meskipun tidak ada rintangan di depannya, yang mana ini adalah ketidaksesuaian dengan konsep *automatic emergency braking*.

Dan, berdasarkan tahapan yang telah dilakukan dalam penelitian ini, diharapkan dapat menjadi dasar untuk penelitian lanjutan. Hal ini mengingat masih banyaknya keterbatasan yang dihadapi dalam penelitian ini. Oleh karena itu, terdapat beberapa usulan pengembangan, yaitu:

1. Diperlukan kajian kembali dalam mengimplementasikan algoritma DQN dan PER karena pada penelitian ini masih belum berhasil dalam memaksimalkan perfoma dari PER.
2. Penyesuaian kembali *hyperparameter* serta model arsitektur pada model DQN stable – baseline3 yang bertujuan untuk mendapatkan model yang lebih baik dengan nilai loss yang rendah.

Ucapan Terima Kasih
Peneliti mengucapkan banyak terimakasih kepada Fakultas Teknik dan Program Studi Teknik Informatika Universitas Pasundan, Ketua Program Studi, para dosen dan pihak lain yang telah mendukung berjalannya kegiatan penelitian ini.

Referensi

[1] J. Pokrzywa, “SAE global ground vehicle standards.” 2014.
[2] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.
[3] Marroli, “Rata-rata Tiga Orang Meninggal Setiap Jam Akibat Kecelakaan Jalan,” *Kominfo*, 2018.
[4] Y. T. Quek, L. L. Koh, N. T. Koh, W. A. Tso, and W. L. Woo, “Deep Q-network implementation for simulated autonomous vehicle control,” *IET Intell. Transp. Syst.*, vol. 15, no. 7, pp. 875–885, Jul. 2021, doi: 10.1049/itr2.12067.
[5] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized Experience Replay,” 2015, *arXiv*. doi: 10.48550/ARXIV.1511.05952.
[6] V. Mnih *et al.*, “Playing Atari with Deep Reinforcement Learning,” Dec. 19, 2013, *arXiv*: arXiv:1312.5602. Accessed: Jul. 23, 2024. [Online]. Available: http://arxiv.org/abs/1312.5602
[7] R. Agarwal, D. Schuurmans, and M. Norouzi, “An Optimistic Perspective on Offline Reinforcement Learning,” 2019, doi: 10.48550/ARXIV.1907.04543.
[8] J. Fan, Z. Wang, Y. Xie, and Z. Yang, “A Theoretical Analysis of Deep Q-Learning,” 2019, *arXiv*. doi: 10.48550/ARXIV.1901.00137.
[9] T. Schaul, J. Quan, I. Antonoglou, D. Silver, and G. Deepmind, “Prioritized Experience Replay,” 2015, doi: https://doi.org/10.48550/arXiv.1511.05952.
[10] B. Fildes *et al.*, “Effectiveness of low speed autonomous emergency braking in real-world rear-end crashes,” *Accid. Anal. Prev.*, vol. 81, pp. 24–29, Aug. 2015, doi: 10.1016/j.aap.2015.03.029.