PENERAPAN DEEP Q NETWORK PADA RAMBU LALU LINTAS UNTUK MENGATUR KEMACETAN DALAM SIMULASI TRAFFIC JAM

SKRIPSI

Oleh:

RIDWANULLAH

NIM. 19650066

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM

MALANG

2025

PENERAPAN DEEP Q NETWORK PADA RAMBU LALU LINTAS UNTUK MENGATUR KEMACETAN DALAM SIMULASI TRAFFIC JAM

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
RIDWANULLAH
NIM. 19650066

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

HALAMAN PERSETUJUAN

PENERAPAN DEEP Q NETWORK PADA RAMBU LALU LINTAS UNTUK MENGATUR KEMACETAN DALAM SIMULASI TRAFFIC JAM

SKRIPSI

Oleh :
RIDWANULLAH
NIM. 19650066

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 13 November 2025

Pembimbing I,	Pembimbing II,
Dr. Ir. Fachrul Kurniawan ST., M.MT ., IPU	Dr. Yunifa Miftachul Arif, M. T
NIP. 19771020 200912 1 001	NIP. 19830616 201101 1 004

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Supriyono, M. Kom
NIP. 19841010 201903 1 012

HALAMAN PENGESAHAN

PENERAPAN DEEP Q NETWORK PADA RAMBU LALU LINTAS UNTUK MENGATUR KEMACETAN DALAM SIMULASI TRAFFIC JAM

SKRIPSI

Oleh :
RIDWANULLAH
NIM. 19650066

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 4 Desember 2025

Susunan Dewan Penguji

Ketua Penguji	: Fatchurrochman, M.Kom NIP. 19700731 200501 1 002	( )
Anggota Penguji I	: Shoffin Nahwa Utama, M.T NIP. 19860703 202012 1 003	( )
Anggota Penguji II	: Dr. Ir. Fachrul Kurniawan ST., M.MT ., IPU NIP. 19771020 200912 1 001	( )
Anggota Penguji III	: Dr. Yunifa Miftachul Arif, M. T NIP. 19830616 201101 1 004	( )

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Supriyono, M. Kom
NIP. 19841010 201903 1 012

HALAMAN PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:
Nama : Ridwanullah
NIM : 19650066
Fakultas / Program Studi : Sains dan Teknologi / Teknik Informatika
Judul Skripsi : Penerapan Deep Q Network Pada Rambu Lalu Lintas Untuk Mengatur Kemacetan Dalam Simulasi Traffic Jam

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 4 Desember 2025
Yang membuat pernyataan,

Ridwanullah
NIM.19650066

KATA PENGANTAR

Segala puji bagi Allah Subhanahu wa Ta'ala, Tuhan semesta alam, atas karunia rahmat, nikmat, dan kesehatan yang telah diberikan sehingga penulis dapat menyelesaikan skripsi ini dengan baik. Shalawat dan salam senantiasa kita panjatkan kepada Nabi Muhammad Shallallahu 'Alaihi Wasallam yang telah membawa agama penuh kebaikan hingga senantiasa bisa kita nikmati ini yaitu agama Islam.

Adapun penyusunan skripsi ini tidak lepas dari bantuan, dukungan, dan doa dari berbagai pihak, baik secara langsung maupun tidak langsung. Sebab itu, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Prof. Dr. Hj. Ilfi Nur Diana, M.Si., CAHRM., CRMP selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang beserta seluruh jajaran pimpinan universitas.
2. Dr. Agus Mulyono, M.Kes selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang beserta jajarannya.
3. Supriyono, M.Kom selaku Ketua Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Dr. Ir. Fachrul Kurniawan ST., M.MT., IPU selaku Dosen Pembimbing I dan Dr. Yunifa Miftachul Arif, M. T selaku Dosen Pembimbing II yang telah membimbing penulis dengan penuh kesabaran dan memberikan arahan selama proses penyusunan skripsi ini.

5. Fatchurrochman, M.Kom selaku Ketua Penguji dan Shoffin Nahwa Utama, M.T selaku Penguji I yang telah memberikan masukan, kritik, dan saran yang sangat membangun demi penyempurnaan skripsi ini.
6. Seluruh Dosen dan Staf di Program Studi Teknik Informatika, dengan ikhlas memberikan ilmu, bantuan, serta dorongan semangat selama perkuliahan.
7. Kedua orang tua yang telah memberikan dukungan dan kepercayaan sampai terwujudnya skripsi ini.
8. Saudara dan teman-teman seangkatan 2019 yang telah memberikan motivasi dalam proses penulisan ini.
9. Semua pihak yang telah membantu dan memberikan dukungan dalam bentuk apapun, yang tidak dapat disebutkan satu per satu.

Dengan segala kerendahan hati penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, penulis berharap karya ini dapat memberikan manfaat, baik bagi pembaca maupun bagi penulis sendiri sebagai proses pembelajaran dan pengembangan diri.

Malang, 4 Desember 2025

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Ridwanullah. 2025. **Penerapan Deep Q Network Pada Rambu Lalu Lintas Untuk Mengatur Kemacetan Dalam Simulasi Traffic Jam**. Skripsi. Program Studi Teknik Informatika. Fakultas Sains dan Teknologi. Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (1) Dr. Ir. Fachrul Kurniawan ST., M.MT., IPU (2) Dr. Yunifa Miftachul Arif, M. T.

Kata Kunci : *Deep Q-Network, Kemacetan, Kontrol Sinyal, Persimpangan, Simulasi*

Pengelolaan lalu lintas menggunakan kontrol sinyal persimpangan masih menjadi tantangan disebabkan respon kontrol sinyal persimpangan kurang adaptif terhadap perkembangan dinamis situasi lalu lintas. Kontrol sinyal persimpangan merupakan krusial dalam pengelolaan lalu lintas dan penguraian kemacetan. Riset ini meneliti apakah lampu kontrol sinyal persimpangan menggunakan *Deep Q-Network (*DQN*)* dapat meningkatkan efektifitas pengelolaan lalu lintas persimpangan. Kami mengembangkan simulasi lalu lintas berbasis giliran dengan dasar durasi yang sama pada simpang tiga dan empat lalu menguji kinerja DQN terhadap kontrol sinyal statis menggunakan metrik waktu penyelesaian, volume lalu lintas, kepadatan persimpangan, dan waktu tunggu kendaraan. Dengan pengujian tersebut model *agent* meraih rata-rata waktu penyelesaian lalu lintas 16,5 detik lebih cepat namun rata-rata waktu tunggu 1,2 detik lebih lama dari kontrol lampu statis pada keseluruhan skenario. Hasil ini disebabkan metrik kepadatan memiliki pembobotan *reward* lebih besar sehingga *agent* lebih unggul untuk melakukan penguraian tetapi kurang optimal untuk memberikan waktu tunggu lebih efisien.

ABSTRACT

Ridwanullah. 2025. Deep Q-Network Implementation on Traffic Signal Control for Traffic Congestion Alleviation in Traffic Jam Simulation. Thesis. Department of Informatics Engineering, Faculty of Science and Technology, Maulana Malik Ibrahim Islamic State University Malang. Supervisor: (1) Dr. Ir. Fachrul Kurniawan ST., M.MT., IPU (2) Dr. Yunifa Miftachul Arif, M. T.

Keywords: *Deep Q-Network, Traffic Congestion, Traffic Signal Control, Intersection, Simulation*

Traffic management using intersection signal control remains a challenge due to the lack of adaptive response of intersection signal control to the dynamic development of traffic situations. Intersection signal control is crucial in traffic management and congestion alleviation. This research examines whether intersection signal control lights using Deep Q-Network (DQN) can improve the effectiveness of intersection traffic management. We developed a sequence-based traffic simulation with the same duration basis at three- and four-way intersections and then tested the performance of DQN against static signal control using the metrics of completion time, traffic volume, intersection density, and vehicle waiting time. With this test, the agent model achieved an average traffic completion time 16.5 seconds faster but an average waiting time 1.2 seconds longer than static light control across all scenarios. This result caused by the density metric which has weighted with a higher reward so the agent is superior in alleviation but is less optimal in providing more efficient waiting time.

مستخلص البحث

رضوان الله. 2025. **تطبيق شبكة Deep Q-Network على إشارات المرور لتنظيم الازدحام في محاكاة الاختناقات المرورية**. رسالة جامعية. برنامج دراسة هندسة المعلوماتية، كلية العلوم والتكنولوجيا، جامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف: (1) د.إير. شارع فخرول كورنيوان، م.م.ت.، الاتحاد البرلماني الدولي (2) د. يونيفة مفتاح عارف، م.ت.

الكلمات المفتاحية : *Deep Q-Network، الازدحام، التحكم بالإشارة، التقاطع، المحاكاة*

لا يزال إدارة حركة المرور باستخدام التحكم في الإشارات عند التقاطعات تحديًا بسبب ضعف استجابة التحكم بالإشارة للتطورات الديناميكية في وضع حركة المرور. ويُعدّ التحكم بالإشارة عند التقاطعات عنصرًا بالغ الأهمية في إدارة المرور وتخفيف الازدحام. تبحث هذه الدراسة فيما إذا كانت أضواء التحكم بالإشارة في التقاطعات باستخدام شبكة *Deep Q-Network (DQN)* قادرة على تحسين فعالية إدارة حركة المرور عند التقاطعات. قمنا بتطوير محاكاة مرور تعتمد على نظام الدور وبالمدة الزمنية نفسها عند التقاطعات الثلاثية والرباعية، ثم اختبرنا أداء شبكة *DQN* مقابل التحكم بالإشارة الثابت باستخدام مقاييس :وقت الإكمال، حجم المرور، كثافة التقاطع، ووقت انتظار المركبات . ومن خلال هذا الاختبار، حقق نموذج الوكيل متوسط وقت إكمال للحركة أسرع بـ 16.5 ثانية، ولكنه سجل متوسط وقت انتظار أطول بـ 1.2 ثانية مقارنة بالتحكم الضوئي الثابت في جميع السيناريوهات. ويعود هذا الاختلاف إلى أن مقياس الكثافة مُنح وزنًا أعلى في المكافأة، مما جعل الوكيل أفضل في تفكيك الازدحام لكنه أقل كفاءة في توفير وقت انتظار أكثر اقتصادية.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kemacetan lalu lintas di Kota Malang merupakan masalah yang serius dimana kemacetan secara langsung mempengaruhi mobilitas masyarakat dan perekonomian kota. Statistik data kemacetan yang tercatat pada Kota Malang menunjukkan rata-rata lebih dari 29 jam waktu terbuang akibat penyumbatan terjadi di jalan. Ini menetapkan malang sebagai kota nomor 4 termacet se-Indonesia pada tahun 2021 dengan rata-rata kecepatan arus jalan 22,5 km/jam tiap harinya (inrix.com).

Berbagai upaya telah dilakukan oleh pihak terkait, seperti pengaturan lampu lalu lintas dan pemberlakuan sistem ganjil-genap seperti di Kota Jakarta pada tahun 2022 berperingkat 29 kota termacet di dunia (TomTom Traffic Index), ini membuktikan sistem ganjil-genap masih belum efektif mengatasi masalah kemacetan lalu lintas secara keseluruhan (Rahadian et al., 2022).

Salah satu faktor yang mengakibatkan terjadinya hal tersebut adalah meningkatnya jumlah kendaraan secara eksponensial yang tidak sejalan dengan perkembangan kapasitas jalan, sehingga kondisi arus lalu lintas bervariasi atau dengan kata lain tidak mudah ditebak (Niken Ekawati et al., 2014). Faktor lain yang mempengaruhi terjadinya kemacetan adalah waktu sinyal lampu lalu lintas persimpangan yang tidak sesuai atau tidak terkoordinasi dengan baik dengan persimpangan lain di sekitarnya dan jalan yang terlalu sempit atau berkelok-kelok (Atmadji & Syukron, 2022).

Selama beberapa dekade terakhir, untuk mengatasi masalah kemacetan penelitian dalam bidang ini telah berfokus pada pengembangan berbagai metode optimasi untuk mengatur durasi pengontrolan sinyal lampu lalu lintas. Dimana pengembangan pengontrolan sinyal ini diterapkan melalui uji coba simulasi lalu lintas. Salah satu pendekatan yang menjanjikan adalah menggunakan algoritma *Reinforcement Learning* (RL), khususnya *Deep Q-Network* (DQN).

DQN adalah *Deep Q-Network* (DQN) jenis algoritma pembelajaran penguatan yang menggunakan *Neural Network* untuk memperkirakan fungsi nilai-Q, yang mewakili hadiah yang diharapkan di masa depan untuk mengambil tindakan tertentu dalam kondisi tertentu (Kumar, 2020). Algoritma DQN menggunakan pemutaran ulang pengalaman untuk menyimpan dan mengambil sampel secara acak dari pengalaman masa lalu untuk melatih jaringan saraf, yang membantu menstabilkan proses pembelajaran. DQN telah digunakan dalam berbagai aplikasi, termasuk mengendalikan *hyperparameter* dalam upaya pengoptimasian pengontrol sinyal rambu lalu lintas (Ni et al., 2023).

(Bouzidi et al., 2021) dalam penelitiannya menggunakan metode DQN dan algoritma *Policy Gradient* untuk mensimulasikan pengontrolan pada satu persimpangan lalu lintas dengan pertimbangan parameter kendaraan yaitu kecepatan, akselerasi, posisi kendaraan, dan fase sinyal. Ini menghasilkan pelatihan *agent* RL dua skenario dan 6000 episode pelatihan *agent* dengan perbandingan pembelajaran signifikan lebih cepat yaitu skenario pertama dalam *agent* pembelajaran simulasi 70% dan kontrol selisih waktu fase sinyal lampu ada di persimpangan sebanyak 76% kemudian skenario kedua dengan pembelajaran simulasi 32% dan kontrol selisih waktu

tiap persimpangan 42% sehingga *agent* RL lebih mudah beradaptasi dengan skenario dan lebih cepat pengontrolan arus lalu lintas.

(Bouktif et al., 2023) dalam penelitiannya menggunakan *Double Deep Q-Network* (DDQN) dan *Markov Decision Process* (MDP) dengan pertimbangan parameter yang diteliti yaitu kehadiran kendaraan di persimpangan, kecepatan kendaraan, dan fase sinyal. Dengan 501 episode pelatihan *agent* RL pada tiap skenarionya dan *Prioritized Experience Replay* (PER) menghasilkan pembelajaran *agent* RL yang lebih stabil dan efektif dalam pengaturan sinyal persimpangan.

Dari penelitian diatas penerapan DQN dalam mengoptimalkan sinyal lampu lalu lintas memiliki potensi untuk menghasilkan solusi yang lebih adaptif dan dinamis daripada metode konvensional. Dalam konteks ini, penggunaan *game* simulasi untuk melatih *agent* DQN dalam mengatur kemacetan lalu lintas di Kota Malang dapat memberikan alternatif solusi untuk mengatasi padatnya kemacetan. Dalam *game* simulasi ini, pemain akan berperan penyedia skenario lalu lintas untuk melatih *agent* DQN.

Tujuan penelitian ini berhubungan dengan Al-Qur’an Ayat Al-Asr (QS. Al-Asr: 1-3) yang berisikan pesan yang kuat tentang pentingnya memanfaatkan waktu dengan sebaik-baiknya. Meskipun ayat ini tidak secara khusus berkorelasi dengan pembagian kemacetan, namun maknanya mengenai penggunaan waktu dapat diterapkan pada situasi apa pun, termasuk dalam mengatasi kemacetan. Berikut adalah bunyinya:

وَالْعَصْرِ. إِنَّ الْإِنْسَانَ لَفِي خُسْرٍ. إِلَّا الَّذِينَ آمَنُوا وَعَمِلُوا الصَّالِحَاتِ وَتَوَاصَوْا بِالْحَقِّ وَتَوَاصَوْا بِالصَّبْرِ

"Demi masa. Sesungguhnya manusia itu benar-benar dalam kerugian, kecuali orang-orang yang beriman dan mengerjakan amal saleh dan nasehat menasehati supaya mentaati kebenaran dan nasehat menasehati supaya menetapi kesabaran."

Dalam buku yang disusun oleh (Ghoffar, 2004) terjemahan Tafsir Ibnu Katsir menyatakan bahwa ayat Al-Asr mengandung makna bahwa waktu manusia sangat terbatas dan berharga, sehingga manusia harus memanfaatkannya dengan sebaik-baiknya agar berhasil. Dalam konteks penanggulangan kemacetan, hal ini dapat diartikan bahwa setiap orang yang berada dalam situasi kemacetan harus memanfaatkan waktunya dengan sebaik-baiknya untuk mencari solusi yang efektif dan efisien untuk mengatasi keadaan tersebut.

1.2 Identifikasi Masalah

1. Faktor apa yang mempengaruhi kepadatan kemacetan pada persimpangan?
2. Bagaimana merepresentasikan kemacetan pada sebuah *game* simulasi?
3. Seberapa baik *Deep Q-Network* dalam selisih rata durasi tunggu dan kepadatan untuk mengatur persimpangan daripada pengontrol sinyal statis?

1.3 Batasan Masalah

1. Pengembangan *game* simulasi hanya terbatas pada penerapan DQN pada pengontrol sinyal persimpangan jalan 3 lajur dan 4 lajur.
2. Penelitian ini berfokus pada pengembangan *agent Reinforcement Learning* (RL) pengatur sinyal lalu lintas.

1.4 Tujuan Penelitian

1. Menganalisis faktor-faktor penyebab utama kemacetan lalu lintas.

2. Membuat sarana uji coba *prototype* untuk membantu riset pengembangan intelijen sistem pengurai kemacetan.

1.5 Manfaat Penelitian

1. Bagi masyarakat umum: penelitian ini diharapkan dapat membantu mengurangi waktu perjalanan dan meningkatkan kenyamanan berkendara pada Kota Malang.
2. Bagi pemerintah setempat: memberikan ide informasi yang berguna untuk perencanaan infrastruktur dan manajemen lalu lintas yang lebih baik.
3. Bagi peneliti lain: menyediakan *model* dasar sebagai referensi dalam penggunaan teknologi untuk pengendalian kemacetan di wilayah perkotaan.

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Berikut adalah kajian pustaka untuk usulan penggunaan metode *Deep Q-Network* untuk simulasi kemacetan. (Xu et al., 2016) mendesain *model* transisi keadaan Markov pada lalu lintas perkotaan dan merumuskan kasus kontrol sinyal lalu lintas dengan *Markov Decision Process* (MDP). Untuk mengoptimalkan beban algoritma komputasi, algoritma MDP *Policy Iteration* (PI) berbasis sensitivitas digunakan untuk memecahkan beban tersebut. Model yang didesain adalah tahapan bervariasi sesuai faktor arus lalu lintas pada persimpangan, dan matriks status transisi dan matriks biaya diperbarui untuk memberikan *policy* optimal baru menggunakan algoritma PI. Model ini juga mudah diperluas dari persimpangan ke jaringan lalu lintas berdasarkan ruang-waktu dan karakteristik distribusi arus lalu lintas. Hasil percobaan numerik dari jaringan lalu lintas kecil menunjukkan pengurangan jumlah kendaraan sebesar 30% secara substansial dibandingkan dengan kontrol waktu statis dan efisien secara komputasi terutama saat lalu lintas padat.

(Abdoos et al., 2011) menggunakan teknik dari pembelajaran penguatan multi-*agent* untuk *model* jaringan lalu lintas besar sebagai sistem *multi-agent*. Dalam penelitiannya *Q-learning* digunakan pada rata-rata panjang dalam pendekatan *link* untuk memperkirakan keadaan. Representasi dimensi aksi membuat metode ini dapat digunakan pada beberapa jenis persimpangan berbeda. Hasil *Q-learning* yang diteliti mengungguli metode lalu lintas statis pada berbagai kondisi dimana pengujian ini

diatur dalam 2 skenario, kemacetan pertama 18000 kendaraan/jam diselesaikan dengan rata-rata selisih waktu 6 detik lebih cepat dan skenario kedua 27000 kendaraan/jam diselesaikan dengan rata-rata selisih waktu 50 detik lebih cepat tiap fase sinyalnya.

(Bálint et al., 2022) dalam penelitiannya menggunakan pendekatan skema *reward* dan Deep *Reinforcement Learning* dengan *Neural Network* pada kontrol sinyal rambu lalu lintas. Dalam *model* penelitiannya *agent* dapat memformulasikan *state* baru yang ter-generalisir dari konvergensi pelatihan episode sebelumnya. Dalam pengujiannya Balint membuat 2 skenario sama yang baru untuk pengujian RL *agent* dan pengujian *Time-gap Based Controller*. RL *agent* menunjukkan pengurangan kepadatan lalu lintas lebih baik sebesar 70% pada skenario pertama dan 32% pada skenario kedua sedangkan *Time-gap Based Controller* mengurangi kepadatan lalu lintas sebesar 76% pada skenario pertama dan 42% pada skenario kedua.

(Hurtado-Gómez et al., 2021) dalam peneltiannya menggunakan algoritma RL *Q-Learning* pada kontrol sinyal rambu lalu lintas dalam sistem simulasi untuk menghitung kendaraan tiap lajur persimpangan. Untuk mendukung Penelitiannya, Hurtado-Gomez menerapkan desain sistem deteksi kendaraan dengan adaptasi algoritma *Computer Vision*. Sistem ini berhasil menurunkan kendaraan sebanyak 29%, waktu jeda menjadi 50%, dan waktu terbuang turun 50% dibandingkan fase kontrol sinyal rambu lalu lintas statis.

(Kim & Jeong, 2020) dalam penelitiannya menggunakan kontrol sinyal rambu lalu lintas DQN dengan prediksi arus kemacetan untuk persimpangan. Pada penelitian ini tiap persimpangan didesain menjadi 4 × 4 dan dimodelkan sebagai *agent* yang di-*training* untuk menentukan aksi terbaik dengan menerima kondisi kemacetan dari

environment. Pada prediksi arus kemacetan ini *output* nilai-Q optimal *agent* lokal ditransfer sebagai pesan untuk menghitung nilai-Q optimal tiap *agent* secara *global* untuk memprediksi arus kemacetan persimpangan secara menyeluruh. Pada hasilnya prediksi arus kemacetan mengalahkan *model Q-learning* dan DQN dengan rata-rata waktu tunggu 25 detik lebih rendah, rata-rata jeda 23 detik lebih rendah.

(Bouzidi et al., 2021) dalam penelitiannya menggunakan metode DQN dan algoritma *Policy Gradient* untuk mensimulasikan pengontrolan pada satu persimpangan lalu lintas dengan pertimbangan parameter kendaraan yaitu kecepatan, akselerasi, posisi kendaraan, dan fase sinyal. Ini menghasilkan pelatihan *agent* RL dua skenario dan 6000 episode pelatihan *agent* dengan perbandingan pembelajaran signifikan lebih cepat yaitu skenario pertama dalam *agent* pembelajaran simulasi 70% dan selisih kontrol waktu fase sinyal lampu di persimpangan sebanyak 76% kemudian skenario kedua dengan pembelajaran simulasi 32% dan kontrol selisih waktu tiap persimpangan 42% sehingga *agent* RL lebih mudah beradaptasi dengan skenario dan lebih cepat pengontrolan arus lalu lintas.

(Bouktif et al., 2023) dalam penelitiannya menggunakan *Double Deep Q-Network* (DDQN) dan *Markov Decision Process* (MDP) dengan pertimbangan parameter yang diteliti yaitu kehadiran kendaraan di persimpangan, kecepatan kendaraan, dan fase sinyal. Dengan 501 episode pelatihan *agent* RL pada tiap skenarionya dan *Prioritized Experience Replay* (PER) menghasilkan pembelajaran *agent* RL yang lebih stabil dan efektif dalam pengaturan sinyal persimpangan.

2.2 Studi Pustaka

Tabel 2. 1 Studi Pustaka

No	Judul (Penulis, Tahun)	Metode	Hasil
1	*Traffic Signal Control based on Markov Decision Process*, (Xu et al, 2016)	*Markov Decision Process*	Model 22 jumlah kendaraan secara substansial sebesar 30% dibandingkan dengan kontrol waktu tetap terutama untuk permintaan lalu lintas tinggi, dan efisien secara komputasi
2	*Traffic light control in non-stationary environments based on multi agent Q-learning, (Abdoos et al, 2011)*	*Q-Learning*	Model Q- Learning mengungguli lalu lintas statis dengan 2 skenario rata-rata *delay* waktu 6 detik dan rata 50 detik lebih cepat tiap fasenya
3	*Deep Q-Network and Traffic Prediction based Routing Optimization in Software Defined Networks, (Bouzidi et al, 2021)*	*Deep Q-Network*	Pelatihan *agent* RL menggunakan algoritma *policy gradient* lebih mudah beradaptasi dan menghasilkan pembelajaran simulasi 70% dan selisih waktu fase 76% pada skenario pertama serta 32% pembelajaran simulasi dan selisih waktu fase 42% pada skenario kedua
4	*Deep reinforcement learning for traffic signal control with consistent state and reward design approach (Bouktif et al, 2023)*	Double *Deep Q-Network*	Dengan 501 episode pelatihan *agent* RL pada tiap skenarionya dan *Prioritized Experience Replay* (PER) menghasilkan pembelajaran *agent* RL yang lebih stabil dan efektif dalam pengaturan sinyal persimpangan

2.2.1 Faktor Penentu Kemacetan

Menurut (Song et al., 2019), faktor penentu kemacetan dapat dipengaruhi oleh faktor berikut:

1. Lebar jalan Kapasitas jalan yang dinilai tidak terlalu besar mengakibatkan pada jam tertentu jalan dipadati dengan jumlah kendaraan bermotor yang tidak seimbang dengan kapasitas lebaran jalan, hal ini menyebakan kemacetan.
2. Volume kendaraan Pada waktu tertentu volume kendaraan bermotor di jalan ini sangat tinggi, terutama pagi hari, siang hari, dan sore hari.
3. Lampu lalu lintas yang dipasang cukup banyak. Banyaknya lampu lalu lintas yang terpasang di sepanjang jalan ini seringkali meyebabkan kendaraan, sehingga secara tidak langsung ini menyebabkan kemacetan. Tidak menutup

kemungkinan ketika melintas di jalur ini pengendara dihentikan dan diantrikan oleh lampu merah beberapa kali.

4. Persimpangan jalan dan gang Persimpangan jalan adalah pertemuan atau percabangan jalan, baik sebidang maupun yang tidak sebidang, persimpangan jalan pertama yang menjadi penyebab di jalan.

2.2.2 Aturan Manajemen Lalu Lintas

Manajemen Rambu lalu lintas pada dunia modern memiliki aturan dasar yang perlu diperhatikan. Berdasarkan (Guberinic et al., 2013) terdapat *state*, variabel kontrol, dan kebutuhan pengoptimasian beberapa kriteria optimasi untuk menghasilkan metode manajemen kontrol rambu sinyal lalu lintas yang baik,

1. State

State didefinisikan sebagai jumlah minimal informasi mengenai Riwayat pemrosesan yang dibutuhkan untuk menentukan *output* dan *state* yang akan datang. Bagian dari proses lalu lintas dinamis pada persimpangan adalah jumlah kendaraan kendaraan dimana ini ditentukan oleh volume dan waktu.

A. Jumlah kendaraan arus.
B. Volume arus.
C. Kepadatan aliran volume arus.
D. Grup pengontrol sinyal (Overview sinyal).

2. Variabel Kontrol
    A. Tiap variabel kontrol (komponen dalam satu grup sinyal) harus memenuhi kondisi dimana grup sinyal bisa mendapatkan satu kesempatan lajurnya untuk berjalan dalam satu siklus.
    B. Durasi waktu rambu lampu hijau untuk setiap grup sinyal harus lebih lama dari minimal durasi lampu hijau yang sudah ditentukan.
    C. Durasi waktu lampu hijau grup sinyal harus memenuhi sebanyak kapasitas kendaraan dalam satu lajur.
3. Kriteria Optimasi
    A. Siklus waktu.
    B. Tinjauan durasi lampu hijau semua grup sinyal.
    C. Durasi lampu hijau grup sinyal.
    D. Banyaknya vektor kontrol (fase) dalam satu rencana sinyal.

2.2.3 Rambu Sinyal Lalu Lintas

Rambu sinyal lalu lintas dalam (Kementrian PUPR, 2017) disebut dengan Alat Pemberi Isyarat Lalu Lintas (APILL). Alat ini berfungsi sebagai pengatur lalu lintas orang dan/atau kendaraan di persimpangan atau ruas jalan. APILL terdiri dari dua jenis sebagai berikut:

1. APILL Otonom

APILL yang memiliki satu warna yang hanya digunakan sebagai peringatan bahaya pada pengguna jalan seperti lampu satu warna berwarna kuning atau merah kelap kelip. Lampu kuning kelap-kelip menyatakan berhati-hati pada pengguna jalan. Lampu merah kelap-kelip menyatakan bahaya pada pengguna jalan.

2. APILL Terkoordinasi

APILL yang memiliki dua sampai tiga warna yang digunakan untuk mengatur orang dan/atau kendaraan. APILL bersusun dua warna terdiri dari merah berkelanjutan dengan hijau secara berurutan baik secara vertikal maupun horizontal. APILL bersusun tiga warna terdiri dari merah berkelanjutan dengan kuning berkelanjutan dengan hijau secara berurutan baik secara vertikal maupun horizontal. Warna lampu tersebut berisyaratkan sebagai berikut:

A. Lampu merah yaitu isyarat menyatakan kendaraan harus berhenti dan tidak boleh melewati garis henti.
B. Lampu kuning yaitu isyarat memberikan peringatan pada pengemudi.
C. Lampu hijau menyatakan kendaraan berjalan atau boleh melintas.

2.2.4 Persimpangan

Persimpangan adalah Tempat bertemunnya dua atau lebih dari lengan/ruas jalan (Kementrian PUPR, 2021). Persimpangan dikategorikan sebagai dua jenis sebagai berikut.

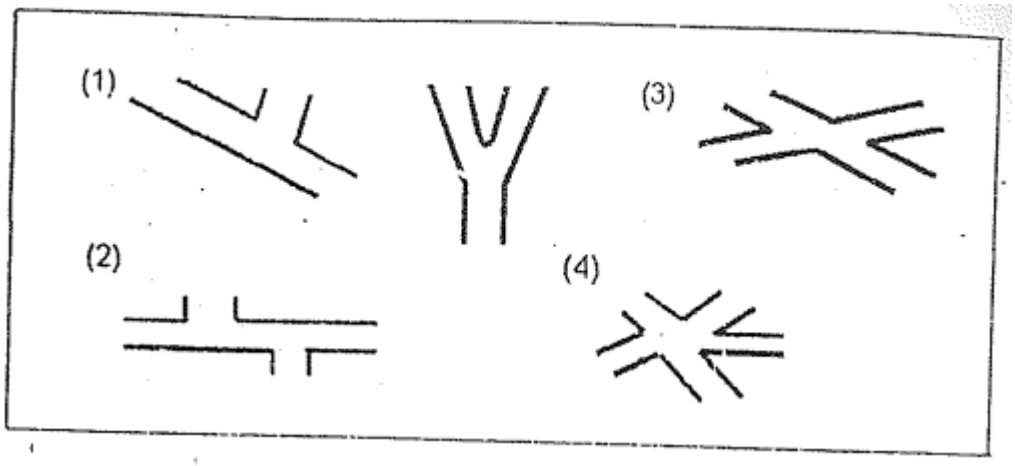
1. Persimpangan Sebidang

Persimpangan sebidang adalah persimpangan dimana pertemuan lengan dengan lengan saling tegak lurus (⊥) dengan toleransi sudut sampai ± 20°. Persimpangan ini meliputi simpang tiga dan simpang empat.

2. Persimpangan Tak Sebidang

Persimpangan tak sebidang adalah persimpangan tidak bisa tegak saling lurus dimana kondisi medan yang sangat sulit disebabkan faktor topografi atau lahan terbatas. Persimpangan ini meliputi;

A. Simpang tiga tidak tegak
B. Simpang empat tidak tegak
C. Simpang tiga ganda
D. Simpang lima



Gambar 2. 1 Persimpangan Tak Sebidang

Terdapat ketentuan umum pada persimpangan sebagai pertimbangan manajemen APILL meliputi bagaimana lajur yang ditentukan. Lajur adalah bagian dari jalur yang

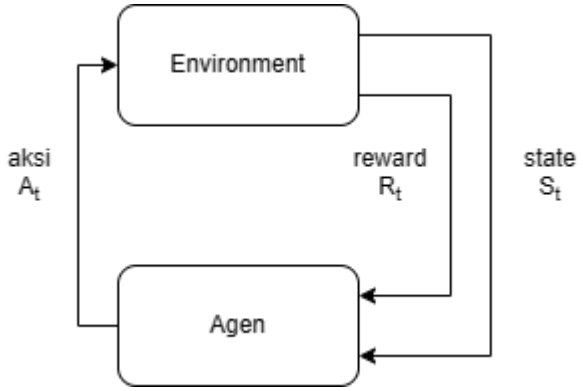
memanjang, memiliki lebar yang cukup untuk kendaraan bermotor berjalan selain sepeda motor. Ketentuan lajur sebagai berikut.

1. Lajur belok kanan sebaiknya diadakan kecuali ada larangan belok kanan jalan dua jalur dimana kecepatan rencana kurang dari 40 km/jam
2. Lebar lajur disesuaikan dengan kecepatan rencana dan kendaraan rencana dengan memperhatikan manuver pergerakan membelok.
3. Lebar lajur dapat ditambah dengan ketetapan antara 2,27 sampai dengan 3,50 meter.
4. Lengan persimpangan berada pada satu lintasan/poros garis lurus untuk lajur masuk dan lajur keluar.
5. Jumlah lajur di persimpangan mengacu pada MKJI.
6. Jika diperlukan pergeseran poros lajur tambahan harus pada standar lengkung/taper yang tepat.
7. Panjang lajur belok kiri ditentukan dengan cara yang sama dengan penentuan lajur untuk belok kanan.

2.2.5 Reinforcement Learning

Reinforcement Learning (RL) adalah teknik pembelajaran mesin yang bekerja melalui kerangka *Markov Decision Process (MDP)* dimana *MDP* adalah fungsi matematis berbasis *feedback* di mana *agent* belajar berperilaku di lingkungan dengan melakukan tindakan dan melihat hasil tindakan (Pol & Oliehoek, 2016). Untuk setiap tindakan baik, *agent* mendapat *feedback* positif, dan untuk setiap tindakan buruk, *agent* mendapat *feedback* negatif atau penalti. *RL* memecahkan jenis masalah tertentu dengan

pengambilan keputusan berurutan dengan tujuan jangka panjang seperti bermain *game* dan robotika (Bouzidi et al., 2021). Proses interaksi *agent* dan mekanisme dalam RL dapat digambarkan seperti gambar 2.2.



Gambar 2. 2 Interaksi dan Mekanisme Reinforcement Learning

Adapun komponen *Reinforcement Learning* sebagai berikut:

1. Agent

Agent dalam *Reinforcement Learning* pada dasarnya merupakan pengambil keputusan. agent dilatih untuk membuat serial keputusan (aksi) dalam simulasi yang telah ditentukan *Environment* (lingkungan)-nya dengan tujuan memaksimalkan *Reward* secara kumulatif.

2. Environment

Environment pada *Reinforcement Learning* merupakan semua hal diluar lingkup *agent*. *Environment* melingkupi semua aspek yang diinteraksikan oleh agent namun tidak dapat dikontrol secara langsung. Ini menyangkup State (keadaan) dalam

Environment, Policy (aturan) yang dapat dioperasikan, dan mekanisme didalamnya yang menyajikan Reward serta feedback pada agent.

3. State

State dalam *Reinforcement Learning* merupakan deskripsi formal untuk keadaan *Environment* yang relevan untuk proses pembuatan keputusan *agent*. State menyajikan konteks yang sesuai pada *agent* yang dapat diperiksa *agent* untuk membuat aksi dan mengambil keputusan selanjutnya agar tujuannya tercapai. Secara ideal, State memuat semua informasi diperlukan dalam mempengaruhi keputusan mendatang, membuat lingkungan *Markovian*- yang artinya *state* mendatang hanya bergantung pada *state* kini dan aksi yang diambil, bukan dari susunan kejadian terdahulu. Sebagai contohnya aksi dapat meliputi posisi kendaraan, kecepatan, jarak antar kendaraan, status sinyal rambu, dan input sensor relevan lainnya.

4. Action (Aksi)

Aksi di *Reinforcement Learning* adalah operasi apapun atau pilihan tersedia untuk *agent* yang dapat digunakan untuk berinteraksi atau merespon *environment.* Tiap aksi yang diambil oleh *agent* mempengaruhi perubahan keadaan terkini ke keadaan baru, sesuai dinamika *environment.* transisi ini melibatkan pendapatan *reward*, yang menentukan nilai aksi yang ditentukan selanjutnya dalam kondisi spesifik berikutnya.

5. Reward

Reward di *Reinforcement Learning* adalah nilai numerik yang diberikan pada *agent* setelah membuat aksi pada keadaan tertentu. *Reward* menggambarkan seketika

keuntungan atau kerugian aksi tersebut dalam *Environment.* Tujuan *agent* dalam RL untuk memaksimalkan *Reward* kumulatif yang didapat seiring waktu.

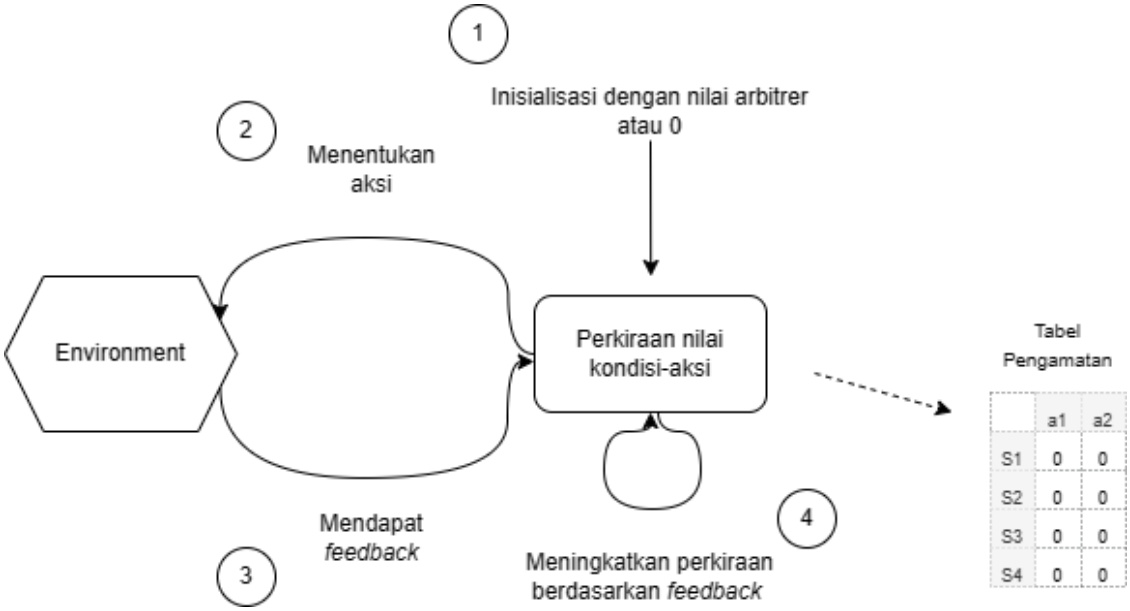
2.2.6 Q-Learning

Menurut (Ni et al., 2023) *Q-Learning* merupakan algoritma dasar DQN *reinforcement learning* yang diprogram secara asinkronous, dimana algoritma *Q-Learning* digunakan untuk memecahkan masalah pengambilan keputusan oleh *agent* learning untuk bertindak secara optimal dalam *Environment* yang ditentukan. *Q-Learning* merupakan algoritma *off-policy, model-free* yang mempelajari nilai aksi yang diambil pada keaadaan tertentu tanpa memerlukan *model Environment.*

Model-free digunakan sebagai solusi karena kasus RL tidak dapat dipecahkan secara *algebra* sehingga perlu menggunakan algoritma iteratif seperti algoritma *model-free.* Algoritma *model-free* mengimplementasikan *Policy* sebagai Tabel Pengamatan. Tabel pengamatan ini berlaku sebagai pemetaan strategi yang dapat disusun oleh *agent* untuk mendapatkan *reward* kumulatif tertinggi sepanjang episode dengan menerapkan fungsi *Q-Learning*.

Konsep dasar *Q-Learning* mempelajari fungsi nilai-Q (S, a), yang merepresentasikan harapan jangka panjang yang dihasilkan dari tindakan "a" pada keadaan "s" dengan mematuhi *policy* (aturan) tertentu. *Policy* π adalah strategi yang dipatuhi *agent* untuk menentukan tindakan yang diambil pada keadaan yang ada (Bouzidi et al., 2021).

Algoritma *Q-Learning* beroperasi dengan *meng-update* nilai-Q untuk tiap serangkaian kondisi-aksi menggunakan persamaan *Bellman* secara iteratif seperti gambar 2.3.



Gambar 2. 3 Desain Algoritma Q-Learning

1. Inisialisasi nilai-Q: Nilai-Q (s, a) diinisialisasikan secara arbitrer untuk semua rangkaian kondisi-aksi. Q (s, a) sering diinisialisasikan 0.
2. Penentuan aksi: dari keadaan *s* kini, *agent* menentukan aksi *a*. Pilihan aksi ini dapat ditentukan menggunakan *policy* turunan dari nilai-Q, seperti ϵ*-greedy*, dimana aksi terbaik (yang memiliki nilai-q tertinggi) dipilih, beserta dengan kemungkinan ϵ aksi acak ditentukan agar mendorong eksplorasi keadaan yang ada.

3. Pelaksanaan aksi dan pengamatan *feedback*: *agent* menjalankan aksi a, mengamati r*eward* r, dan menandai perubahan pada *state* baru s' yang terjadi pada e*nvironment* setelah pengambilan aksi.
4. Peng-*update*-an nilai-Q: *agent* meng-*update* **nilai-Q kini** menggunakan *reward* dan **nilai-Q target (nilai-Q maksimum)** untuk tiap pasang kondisi-aksi (s, a) menggunakan rumus:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)] \qquad (2.\ 1)$$

   Dimana:

   - α merupakan *learning rate* (kecepatan pembelajaran) ($0 < \alpha \leq 1$) – ini menentukan berapa banyak informasi baru *meng-update* informasi lama.
   - γ merupakan *discount factor* (faktor pengabaian) ($0 < \gamma \leq 1$) – ini merepresentasikan perbedaan dalam seberapa pentingnya antara *reward* mendatang dan *reward* kini.
5. Pengulangan: pengulangan langkah b sampai d untuk tiap episode, sampai *agent* mencapai kondisi terantuk dan melakukan pengulangan episode hingga Agen mendapat konvergensi untuk *policy* teroptimal.

2.2.7 Deep Q-Network

Deep Q-Network (DQN) merupakan teknik dalam RL yang mengimplementasikan *Neural Network* untuk menghitung fungsi nilai aksi optimal $Q*$ yang merupakan fungsi *Q-Learning*. Fungsi ini diartikan sebagai nilai perkiraan yang didapatkan dari menentukan aksi *a* dalam keadaan *s* (Kumar, 2020).

Tujuan utama DQN adalah untuk memahami kebijakan optimal $\pi*$, yang dapat memaksimalkan jumlah total *reward* yang diharapkan yang diperoleh mulai dari keadaan awal *s* dengan mengadopsi kebijakan π.

Dalam konteks pengembangan *game* simulasi lalu lintas, konsep DQN diimplementasikan untuk mengatur rambu lalu lintas secara adaptif dari kompleksitas kondisi lalu lintas yang ada. Sistem akan menggunakan *Neural Network* untuk memperkirakan durasi waktu lampu sinyal dari setiap kombinasi keadaan lalu lintas sebagai tindakan pengaturan rambu lalu lintas. Durasi waktu ini akan digunakan untuk memilih tindakan yang paling optimal dari hasil nilai fungsi-Q terbaik DQN *agent* untuk diterapkan pada setiap saat, dengan tujuan untuk mengoptimalkan aliran lalu lintas secara keseluruhan. Secara umum arsitektur DQN terdiri dari berikut:

1. Q-network

Q-network adalah *agent* yang dilatih untuk menghasilkan nilai kondisi-aksi optimal. *Q-network* sendiri merupakan neural *network* sederhana seperti linear *network* dengan beberapa *hidden layer* yang merepresentasikan *state* berisikan variabel numerik. Arsitektur *Q-network* dapat berupa CNN atau RNN jika *state* yang di-inputkan merupakan gambar atau teks. Dalam *Q-network* tabel serangkaian kondisi-

aksi dimasukkan sebagai *input layer*, *hidden layer* sebagai perhitungan fungsi Q(s, a), dan nilai fungsi Q tiap aksi adalah *output* layer.

2. Target-network

target-network merupakan jaringan saraf dengan arsitektur yang sama dengan *Q-network* namun *target network* tidak dilatih melainkan di-*update* bobotnya dengan menyalin *Q-network* tiap (t) langkah yang ditentukan. *Target network* digunakan untuk menstabilkan *output* nilai-Q dari *Q-network* sehingga mendapatkan konvergensi fungsi nilai-Q kedua *network.*

3. Experience Replay

Experience Replay merupakan data pengamatan dan pengalaman interaksi *agent* dengan *environment* yang dikumpulkan untuk membuat data untuk melatih *Q-network*. Pengalaman dalam *Experience Replay* pada langkah waktu t direpresentasikan dengan elemen berikut:

$$(s_t, a_t, r_t, s_{t+1}, \textbf{done}) \tag{2. 2}$$

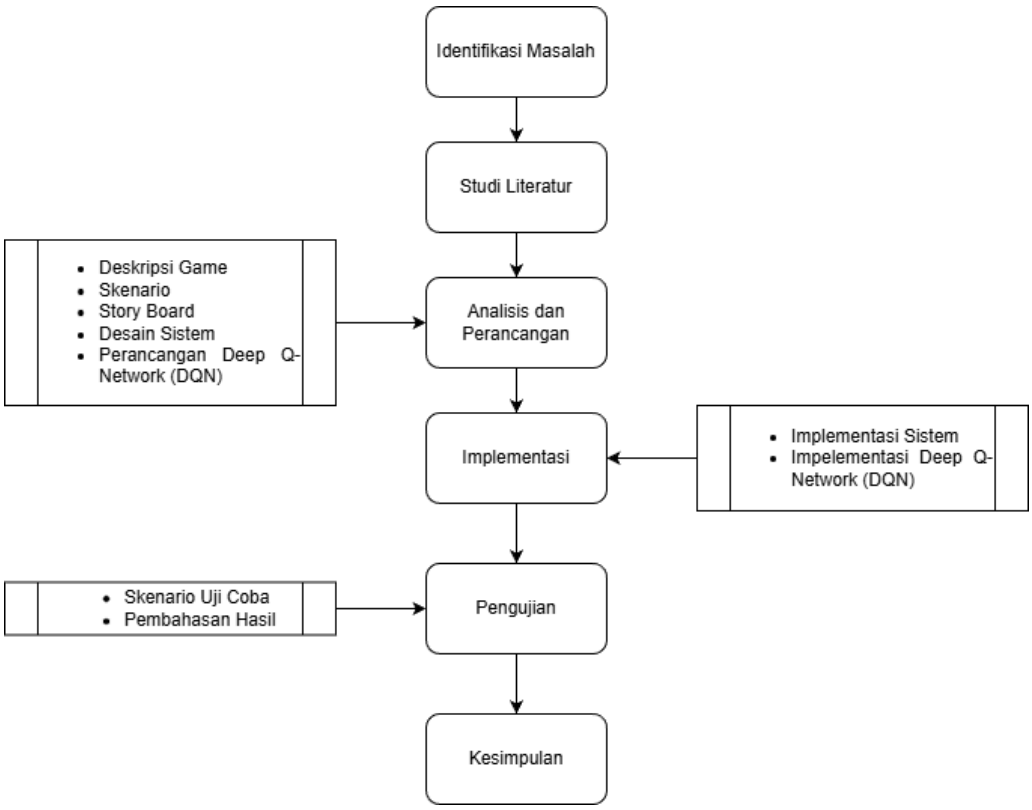
Dimana s_t merupakan kondisi pada waktu *t*, a_t adalah aksi yang ditentukan, r_t merupakan *reward* yang diperoleh, s_{t+1} merupakan kondisi baru setelah aksi ditentukan, dan **done** merupakan indikasi episode selesai.

BAB III

DESAIN DAN PERANCANGAN

3.1 Tahapan Penelitian

Pada Tahapan Penelitian ini diuraikan langkah-langkah penelitian yang diambil untuk melakukan penelitian dan menjawab permasalahan yang diangkat. Langkah-langkah tersebut diuraikan pada gambar 3. 1 berikut.



Gambar 3. 1 Tahapan Penelitian

Tahap pertama adalah identifikasi masalah yaitu langkah untuk mengurai masalah dan memahami permasalahan yang berkaitan pengontrolan sinyal rambu lalu lintas optimal dalam menangani kemacetan dalam sebuah *game* simulasi.

Tahap kedua merupakan studi literatur, yaitu analisis dan pencarian sumber-sumber literatur relevan yang dapat digunakan dalam penelitian. Tahap ketiga merupakan analisis dan perancangan. Tahap ini memuat uraian desain sistem, deskripsi *game*, skenario *game*, rancangan *storyboard*, dan penerapan metode *Deep Q-Network* (DQN) dalam penelitian.

Tahap keempat adalah implementasi, yaitu penjelasan penerapan sistem dan elemen-elemen permainan pada *game*. Tahap ini meliputi implementasi *environment game, user interface* (UI) *game,* dan *model* DQN. Selanjutnya, tahap kelima, yaitu tahap pengujian yang meliputi *data training* dan pengujian *policy decision-making agent* serta pengujian sistem dan pengujian UI.

Terakhir, tahap keenam adalah pengambilan kesimpulan dari hasil penelitian yang dilakukan. Dengan demikian, penelitian ini diharap dapat membantu dalam mengatasi dan memahami permasalahan kemacetan dalam konteks simulasi permainan.

3.1.1 Deskripsi Game

Game simulasi merupakan model permainan yang didesain untuk menirukan situasi dunia nyata secara aktivitas, sistem, atau *environment,* agar *player* dapat membuat keputusan bermain dan pengalaman dengan konsekuensi yang realistis. (Irawan et al., 2019)

Game ini merupakan *game* mensimulasikan kondisi lalu lintas beserta masalah kemacetan dengan fokus permainan berada pada pengendalian lampu sinyal

persimpangan. Sehingga, permainan ini ditujukan untuk menyelesaikan permasalahan kemacetan yang terjadi pada jalan raya.

3.1.2 Skenario Game

Skenario permainan merupakan seluruh elemen yang berkontribusi pada pengalaman *gameplay* dalam permainan baik interaktif maupun non-interaktif. Elemen ini meliputi *setting, event,* karakter, dan *goal*/tujuan dalam permainan. (Y. Li et al., 2024)

Pada penelitian ini akan dikembangkan *game* simulasi kemacetan pada *environment* persimpangan perkotaan dan sistem kontrol sinyal rambu lalu lintas.

Gameplay pada permainan ini yaitu *player* memainkan *game* untuk mengatur skenario simulasi lalu lintas untuk melatih *agent* DQN. Skenario ini ditentukan dengan pemilihan *environment* persimpangan untuk melatih *agent* pengatur rambu lalu lintas. Kemudian dilengkapi dengan penentuan banyaknya kendaraan, durasi waktu pelatihan pada skenario simulasi lalu lintas ini.

Tujuan dari *game* ini adalah mensimulasikan kemacetan pada persimpangan dan mengurai kemacetan tersebut melalui kontrol sinyal rambu lalu lintas menggunakan *agent* DQN. *agent* DQN mengatur laju-henti tiap lajur persimpangan dengan mengontrol sinyal lampu secara langsung. *agent* dapat memberikan durasi sinyal yang tepat berdasarkan pengalaman mengatasi situasi kemacetan yang ditandai dengan deteksi kendaraan melintas, Jumlah kendaraan, dan kepadatan persimpangan. Deteksi tersebut seiring waktu dan pengalaman *agent* yang terkumpul membuat pengontrolan

sinyal menjadi adaptif memberikan durasi sinyal optimal tiap lajur pada tiap situasi dalam mengurai kepadatan persimpangan.

Skenario *game* pada penelitian ini dapat diuraikan sebagai berikut:

1. Persiapan Environment

   Environment pada *game* ini terdiri dari asset gedung, lampu lalu lintas, jalan, persimpangan, dan mobil. Pada game ini disediakan 4 tempat dengan jenis persimpangan *environment* masing-masing yaitu:

   A. Persimpangan simpang tiga JL. MT Haryono.
   B. Persimpangan simpang tiga JL. Gajayana Dinoyo.
   C. Persimpangan simpang empat JL. Laksada Adi Sucipto.
   D. Persimpangan simpang tiga JL. Borobudur.

2. Desain Level

   Simulasi *Traffic Jam* memiliki desain level utama dengan 4 skenario persimpangan tempat kemacetan yang telah disebutkan sebelumnya. Pada tiap levelnya dibuat persimpangan dengan skenario simulasi yang dapat ditentukan sendiri oleh *player* di menu pembuatan skenario *game* dengan pengaturan jumlah kendaraan dan lokasi persimpangan. Setelah *player* membuat skenario, simulasi dijalankan sesuai dengan skenario desain yang telah ditentukan oleh *player* di menu pembuatan skenario *game*. Ketika simulasi selesai, *game* akan menunjukkan menu hasil dari pelatihan *agent* yang berupa durasi episode, kendaraan berhasil melintas, lokasi persimpangan.

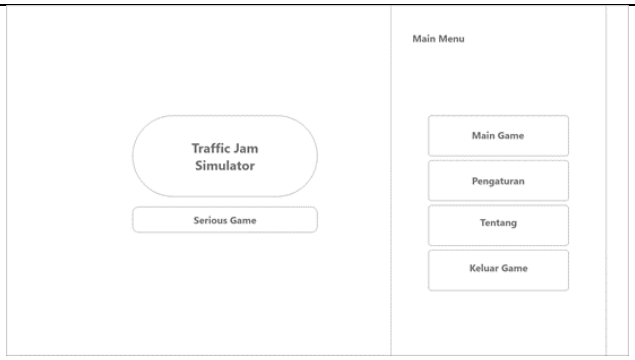
3. Pengembangan Teknologi

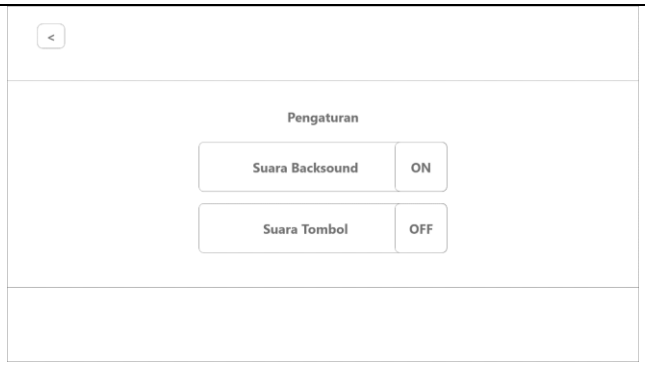
Game simulasi *Traffic Jam* dikembangkan menggunakan Unity *game engine* dengan pemrogaman C#. Sedangkan pengembangan *Reinforcement Learning* sistem kontrol sinyal *agent* menggunakan python Tensorflow. Script *reinforcement learning* ini kemudian diintegrasikan pada Unity menggunakan *plugin* ML Agents untuk berkomunikasi data dan mengeksekusi *model* DQN di unity.

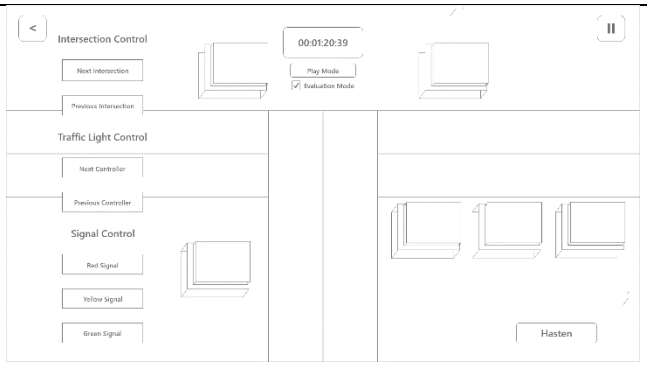
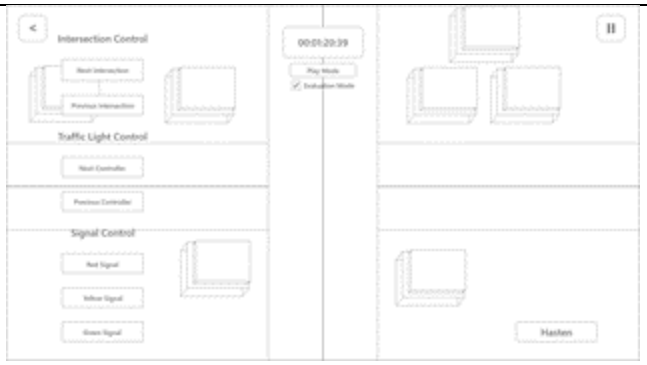
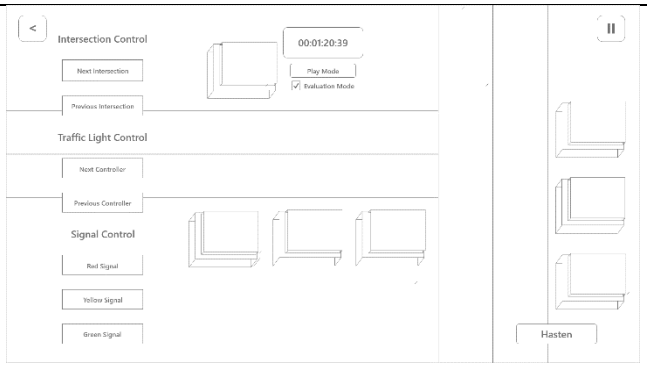
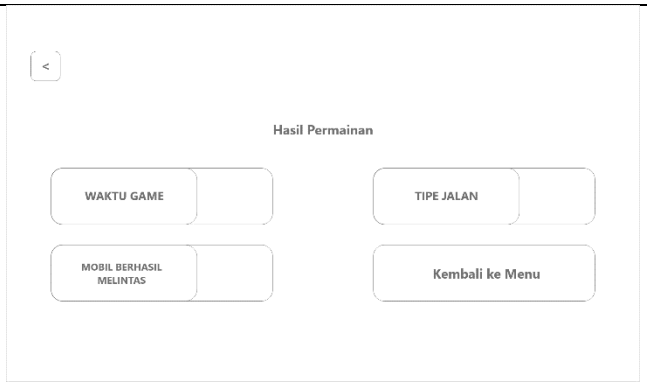
3.1.3 Storyboard Game

Storyboard Game yaitu kumpulan gambar untuk menjelaskan urutan *event* atau peristiwa yang terjadi pada *game*. Tiap gambar pada *storyboard* menggambarkan tiap adegan yang akan terjadi pada *game* yang sedang dikembangkan. Tabel 3.1 merupakan pemaparan *storyboard* Traffic Jam:

Tabel 3. 1 Storyboard Game

No	Konsep Storyboard	Keterangan
1		Berikut adalah Main menu merupakan tampilan awal pada *game* ini. Terdapat 4 tombol pada main menu yaitu main *game* untuk memulai *game*, menu pengaturan umtuk mengatur permainan, menu tentang untuk menjelaskan deskripsi

		game dan keluar *game* untuk menghentikan *game*.
2		Pada menu setting, user dapat mematikan background music dan mematikan sound effect tombol.
3		Pada menu ini user dapat memilih tipe jalan yang akan dimainkan. Pada jalan 1 terletak pada Jalan MT Haryono, pada jalan 2 terletak pada Jalan Laksada Adi Sucipto dan jalan 3 terletak pada jalan Borobudur. Pada menu ini juga dapat mengatur jumlah mobil yang spawn pada *game*. Dan pada menu mulai untuk memulai permainan. Untuk Kembali ke main menu dapat tekan pada tombol kanan atas.

4		Berikut merupakan tampilan *gameplay* jalan 1 pada jalan MT Haryono.
5		Berikut merupakan tampilan *gameplay* jalan 2 pada jalan Laksada Adi Sucipto.
6		Beikut merupakan tampilan *gameplay* jalan 3 terletak pada jalan Borobudur.
7		Pada Tampilan hasil permainan ini terdapat 3 text field dan satu tombol yaitu waktu *game* permainan yang diinputkan diawal permainan, tipe jalan yang dipilih, mobil yang berhasil melintas pada rute

		akhir jalan dan tombol Kembali ke main menu. Tujuan dari hasil permainan ini untuk menyimpulkan hasil akhir *game* yang telah selesai dimainkan.

3.2 Desain Sistem

Untuk melakukan penelitian ini diperlukan dua desain sistem *environment*, yaitu sistem *environment training* dan sistem *environment* evaluasi pada *game* ini untuk melakukan pengujian dan mengetahui perbandingan hasil kinerja DQN daripada kontrol lampu sinyal statis.

Sistem *environment* pada *game* ini memiliki *behavior environment* berbeda sesuai dengan mode permainan, dimana *behavior* ini dikembangkan berdasarkan kebutuhan skenario uji coba. Kedua *behavior environment* tersebut selanjutnya akan berinteraksi dengan mekanisme kontrol lampu persimpangan. Kedua sistem *environment* dapat dijelaskan lebih lanjut sebagai berikut.

3.2.1 Sistem *Environment* Training

Sistem *environment training* ini merupakan sistem simulasi lalu lintas yang digunakan sebagai *training environment* DQN dimana *agent* akan dilatih untuk mempelajari pengontrolan lampu sinyal lalu lintas untuk mengurai kemacetan persimpangan. Pada sistem *environment training* mekanisme simulasi lalu lintas yang

berjalan adalah simulasi dinamis, dimana kondisi lalu lintas akan semakin berkembang seiring dengan lamanya *training*.

Sistem *environment training* memiliki *behavior* perulangan prosedur dimana kendaraan dengan jumlah yang telah ditentukan pada menu pengaturan skenario pada titik-titik lokasi yang ditentukan secara acak. Sistem ini akan terus menerus men-*generate* dan menjalankan kendaraan serta mengecek *environment* yang sedang berlangsung untuk menangani dengan menghapus kendaraan tersebut dan me-*respawn* kendaraan ketika terjadi kesalahan pada *environment* seperti terjadinya tabrakan dan kesalahan *behavior* pada kendaraan yang menyebabkan kebuntuan pada lalu lintas. Setiap interval waktu satu jam akan terjadi *reset environment* agar mengulang skenario acak lalu lintas untuk mengurangi penggunaan *RAM* dan *CPU* dari kompleksitas perkembangan *environment* yang telah terjadi.

3.2.2 Sistem *Environment* Evaluasi

Sistem *environment* evaluasi merupakan sistem *environment* yang digunakan untuk menguji keefektivitasan kedua kontrol, kontrol *agent* dan kontrol statis berdasarkan parameter uji yang dilakukan. Sistem ini memberikan kondisi lalu lintas yang telah ditentukan pada *environment* lalu memberikan hasil penilaian berdasarkan *output state* yang dihasilkan.

3.2.2.1 Behavior *Environment* Evaluasi

Behavior pada sistem *environment* evaluasi akan memunculkan kendaraan dengan kondisi seperti letak pada lajur dan jumlah kendaraan yang sudah disiapkan untuk menguji kemampuan lampu kontrol sinyal persimpangan. Kemudian sistem akan

mencatat hasil penyelesaian kontrol yang sedang diuji beserta parameter pengujian dan penilaiannya.

3.2.2.2 Parameter Penilaian Evaluasi

Penilaian efektivitas *agent* diambil dengan alasan *behavior environment training* simulasi di dalam *game* terus berkembang secara dinamis. Ini menyebabkan kondisi lalu lintas terus berubah dan menjadi sulit seiring waktu, sehingga perhitungan hasil kinerja *reinforcement learning* konvensional (seperti *loss*, nilai-Q, *reward* tiap episode) menunjukkan variasi yang tinggi dan terbatas untuk di-interpretasikan.

Oleh karena itu, kami juga menyajikan statistik parameter perhitungan efektivitas berbasis permintaan persimpangan untuk menilai keberhasilan pembelajaran *agent*. Perhitungan ini mendemonstrasikan *behavior* yang lebih stabil dan ter-generalisir untuk kesulitan yang terus berkembang pada *environment* simulasi lalu lintas yang dikembangkan dalam sistem *environment* evaluasi ini.

Parameter penilaian sistem *environment* evaluasi ini digolongkan menjadi dua berdasarkan pengaruhnya dengan penguraian kemacetan dan konteks simulasi yaitu parameter penilaian utama dan parameter pendukung.

1. Parameter Penilaian Utama

Parameter penilaian utama yang kami ambil adalah waktu penyelesaian skenario lalu lintas. Waktu penyelesaian skenario menjadi parameter penilaian utama untuk menginterpretasikan penyelesaian kontrol sinyal agent terhadap skenario lalu lintas yang diujikan.

Parameter ini menyimpulkan kinerja kontrol sinyal dengan mengukur waktu kontrol sinyal persimpangan menyelesaikan skenario lalu lintas yang telah ditetapkan, dimana penyelesaian dengan waktu yang singkat menandakan pengontrolan yang baik.

2. Parameter Pendukung.

Parameter pendukung mendemonstrasikan kinerja kontrol sinyal terhadap kondisi lalu lintas yang dihadapi secara langsung. Kondisi lalu lintas ini adalah *behavior* kendaraan saat berkendara ketika simulasi, dimana masing-masing kendaraan memiliki rute berkendara yang berbeda dan bergerak dinamis.

A. *Reward* Kumulatif

Reward kumulatif adalah **jumlah total *reward*** yang diperoleh selama satu episode simulasi (satu siklus penuh pengaturan lalu lintas oleh *agent* RL). Semakin tinggi nilai ini, semakin baik pengaturan sinyal karena menunjukkan keputusan yang menghasilkan kondisi lalu lintas optimal (waktu tunggu rendah, *throughput* tinggi, kemacetan rendah). (Bouzidi et al., 2021)

$$R_{total} = \sum_{t=1}^{T} r_t \tag{3. 1}$$

Dimana:

- R_{total} : *Reward* kumulatif selama satu episode.
- r_t : *Reward* yang diterima pada waktu atau langkah ke-t.
- T : Jumlah total langkah waktu (episode simulation time steps).

B. *Average Waiting Time*

Average waiting time atau rata waktu tunggu merupakan rata-rata waktu kendaraan menunggu dalam satu fase siklus manajemen lalu lintas. Semakin rendah nilai parameter ini mengindikasikan berkembangnya arus lalu lintas semakin baik. (L. Li et al., 2016)

$$AWT = \frac{\sum_{i=1}^{n} total\ waktu\ tunggu\ tiap\ kendaraan}{jumlah\ kendaraan} \quad (3.\ 2)$$

Dimana:

- W_i : Total waktu tunggu kendaraan ke-i (misalnya dalam detik).
- n : Jumlah total kendaraan yang diamati dalam satu episode.
- AWT dinyatakan dalam satuan waktu (detik, menit).

C. Jumlah kendaraan

Menunjukkan banyaknya jumlah kendaraan pada jalur persimpangan. Parameter ini digunakan untuk mengukur tingkat kemacetan atau arus lalu lintas. (Song et al., 2019)

$$N_{veh} = \sum_{l=1}^{L} n_l \quad (3.\ 3)$$

Dimana:

- N_{veh} : Total jumlah kendaraan di seluruh lajur.
- n_l : Jumlah kendaraan di lajur ke-l.
- L : Jumlah lajur di persimpangan.

Nilai ini bisa dinyatakan rata-rata atau maksimum selama simulasi.

D. Throughput

Throughput dalam konteks pengaturan lalu lintas mengacu pada jumlah kendaraan yang berhasil melewati persimpangan dalam satu episode. *Throughput* menunjukkan seberapa baik *agent* memproses kendaraan. (Zhang et al., 2022)

$$Efisiensi_{Throughput} = \frac{N_{lolos}}{N_{permintaan}} \quad (3.\ 4)$$

Dimana:

- N_{lolos} : Jumlah kendaraan yang berhasil melewati persimpangan (keluar).
- $N_{permintaan}$: Jumlah kendaraan yang datang (masuk) selama episode.
- *Throughput* sering dinyatakan sebagai rasio (0–1) atau dalam persen. Semakin tinggi nilai *throughput* menunjukkan semakin baik performa kontrol.

E. Volume Lalu Lintas Persimpangan

Volume lalu lintas menunjukkan seberapa lancar arus jalan tersebut untuk dilewati kendaraan dalam satu episode. (Way, 1985)

$$Volume\ Lalu\ lintas\ (kendaraan\ /\ menit) = \frac{N}{T}$$

Dimana:

- N : Jumlah kendaraan.

- T : Periode perhitungan waktu (menit).

F. Kepadatan Persimpangan

Mengukur jumlah kendaraan per unit disekitar area persimpangan. Untuk mengindikasikan kemacetan. Semakin tinggi kepadatan, mengindikasikan potensi besar kemacetan. (Kim & Jeong, 2020)

$$Density = \frac{N_{veh} \times L_{veh}}{\sum L_{jalur}} \quad (3.5)$$

Dimana:

- N_{veh} : Jumlah kendaraan di area persimpangan.
- L_{veh} : Panjang rata-rata satu kendaraan (misalnya 4,5 meter).
- $\sum L_{jalur}$: Total panjang seluruh jalur atau lajur yang diamati.
- Hasilnya biasanya tanpa satuan (rasio), menunjukkan tingkat kepadatan relatif.

G. Reward Tiap Kendaraan

Menunjukkan efektivitas rata-rata *reward* yang diperoleh untuk tiap kendaraan. Digunakan untuk menilai seberapa efisien sistem memberikan *reward* terhadap setiap kendaraan. (Liang et al., n.d.)

$$RewardNorm = \frac{reward}{\sum kendaraan} \quad (3.6)$$

Dimana:

- $Reward_{total}$: Total *reward* selama satu episode.

- $\sum$ *Kendaraan*: Jumlah total kendaraan yang diamati.

H. Kepadatan Tiap Kendaraan

Menunjukkan skala tingkat kepadatan relatif per kendaraan, atau skala kemacetan dibandingkan jumlah kendaraan yang ada. Nilai kecil menunjukkan kepadatan rendah per kendaraan (lalu lintas lancar). (Song et al., 2019)

$$Kepadatan_{norm} = \frac{Kepadatan_{persimpangan}}{\sum Kendaraan} \quad (3.\ 7)$$

Dimana:

- $Kepadatan_{persimpangan}$: Nilai kepadatan dari perhitungan sebelumnya.
- $\sum$ *Kendaraan*: Jumlah total kendaraan.

3.3 Analisis dan Perancangan Agent DQN pada Game

Melalui desain sistem simulasi *environment* berdasarkan pengujian yang sudah dibahas pada sub bab 3.2, maka pengontrolan sinyal persimpangan dapat dikembangkan menjadi tiga sistem kontrol yaitu kontrol lampu statis, kontrol *agent*, dan kontrol *player*.

Untuk dapat mengembangkan *agent* Pengontrol Sinyal Rambu Lalu Lintas yang cerdas menggunakan *reinforcement learning*, perlu dikembangkan sebuah *environment game* yang sesuai untuk melatih dan menyimulasikan AI secara *real time* dan berkala. Maka dari itu *game* Traffic Jam dibangun dengan *environment* berupa area perkotaan menyerupai situasi jalan berdasarkan lokasi penelitian yang sering terjadi kemacetan yaitu Kota malang pada area Kecamatan Lowokwaru, persimpangan JL. M.T. Haryono

(Jembatan Soekarno-Hatta), persimpangan JL. Dinoyo dan persimpangan JL. Laksada Adi Sucipto, dan persimpangan JL. Borobudur.

3.3.1 Alur Perancangan DQN pada Game

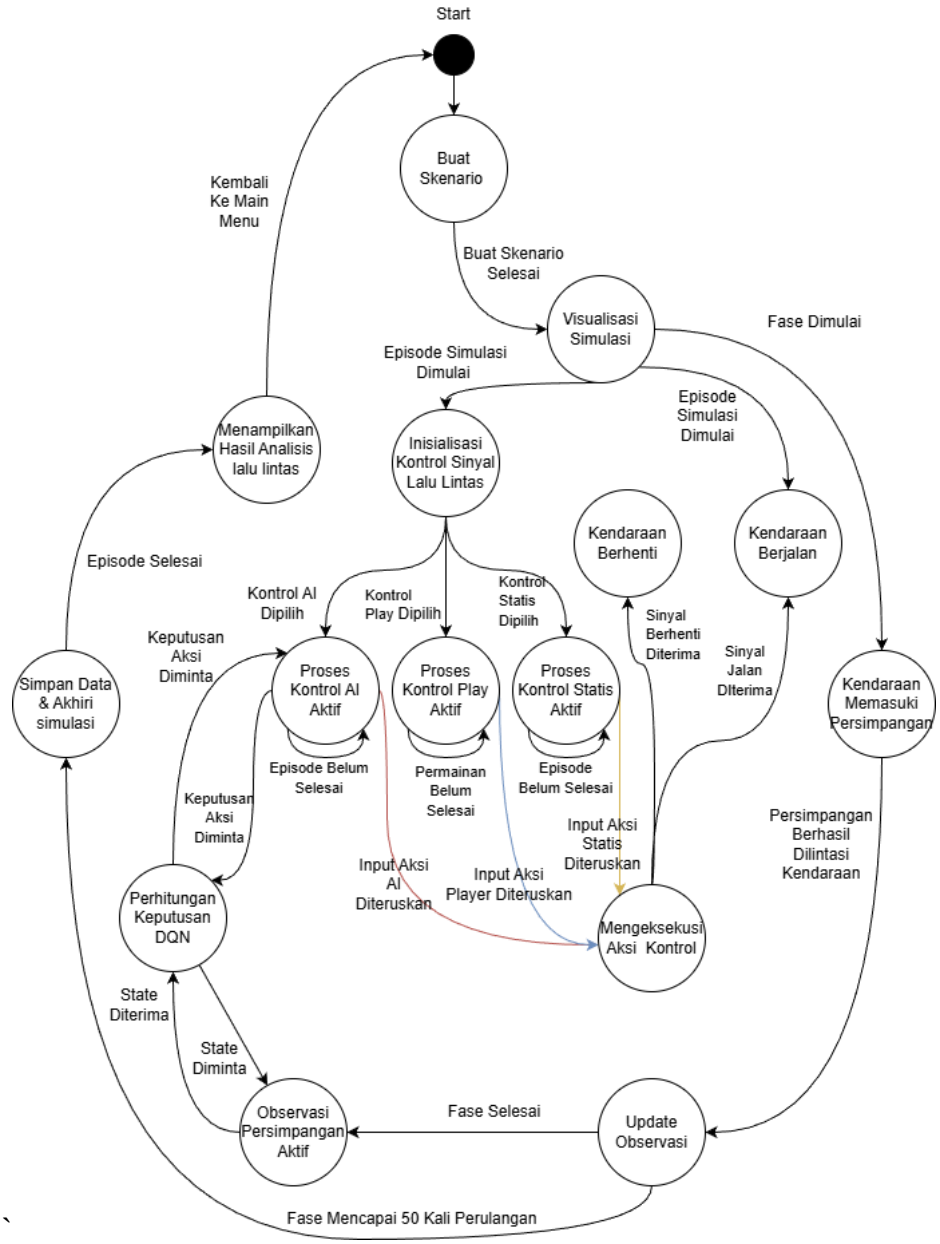
Pada *game* ini alur kerja *game* simulasi Traffic Jam ini didasari oleh interaksi kontrol sinyal dengan sistem *environment* simulasi yang telah dibahas pada sub bab 3.2. Mekanisme interaksi ini dipaparkan pada gambar diagram *Finite State Machine* 3.2.

Alur mekanisme interaksi simulasi permainan pada gambar 3.2 dapat dijelaskan seperti berikut:

1. Pada awalnya *player* akan dimulai dengan membuat skenario lalu lintas pada menu pembuatan skenario dengan memasukkan jumlah mobil dan lokasi persimpangan lalu menekan tombol mulai.
2. Ketika permainan dimulai maka sistem *environment* akan men-*generate* mobil dan situasi lalu lintas sesuai dengan pembuatan skenario yang telah ditentukan.
3. Mobil akan melaju menuju persimpangan dimana ini akan memicu sistem untuk melakukan cek kontrol lampu persimpangan untuk mengatur persimpangan untuk menjalankan kendaraan. Disini *player* dapat memilih kontrol yang akan dilakukan, yaitu kontrol lampu statis, kontrol *agent*, dan kontrol *player*.
4. Jika *player* ingin melakukan *training*, *player* menekan tombol mode kontrol persimpangan dan mengarahkan ke *AI Mode*,

5. Ketika *game* dimulai, *player* menekan tombol mode kontrol persimpangan dan mengarahkan ke penggunaan AI, ini akan menjeda permainan guna memulai proses penyalaan *agent* pada *console command*.
6. Selanjutnya sistem akan menjalankan ML Agents dan DQN untuk memulai *training*, setelah selesai siap *player* akan diarahkan untuk melanjutkan jeda *game*.
7. Kemudian pada tahap ini proses *training* dimulai bersamaan dengan simulasi lalu lintas. *Progress training* berupa langkah *agent* DQN dapat dilihat pada *console command*, untuk *progress* simulasi dapat dilihat pada *debug console unity*.
8. Pada proses ini, sistem akan mengambil data pengamatan *state* lajur pada *game* sebagai inputan *agent* kemudian *agent* akan memproses inputan tersebut untuk memberikan *output* berupa aksi yang akan dieksekusi oleh sistem kontrol *game*.
9. Ketika simulasi *training* selesai maka *player* akan diarahkan pada menu hasil yang berisikan hasil simulasi *training* AI yang telah dilakukan. Untuk detail hasil *training* secara keseluruhan dapat dilihat pada *tensorboard* dan lampiran *data training* dalam *format csv*.
10. Jika *player* ingin menjalankan kontrol statis, *player* menekan dan mengarahkan tombol kontrol ke *Auto Mode*. Ini akan menjalankan mekanisme pengendalian lampu statis persimpangan dengan memberikan waktu 10 detik sinyal jalan yakni lampu hijau pada tiap lajur secara bergantian.
11. Jika *player* ingin mengendalikan kontrol persimpangan secara langsung, *player* dapat menekan dan mengarahkan tombol kontrol ke *Play Mode*.

12. Untuk melakukan pengujian, maka *player* menyalakan tombol *boolean Evaluation Mode* dan mengarahkan tombol kontrol sinyal ke *AI Mode* atau *Auto Mode*. Ini akan menjalankan sistem *environment* evaluasi dengan kondisi lalu lintas berupa yang sudah ditentukan dari skenario uji coba penelitian.
13. Ketika pengujian selesai maka akan ditampilkan menu hasil juga yang memuat isi evaluasi yang dilakukan.



Gambar 3. 2 Diagram perancangan agent DQN pada Traffic Jam

3.3.2 Perancangan Kontrol Sinyal pada Game

Perancangan kontrol sinyal pada *game* ini merujuk kepada pengembangan sistem kontrol sinyal persimpangan seperti yang sudah dijelaskan pada gambar 3.2. Kontrol sinyal ini dibutuhkan sebagai elemen interaksi pada *gameplay* simulasi lalu lintas.

Dimana terdapat dua mekanisme kontrol yang dikembangkan yaitu mekanisme kontrol *agent* dan mekanisme kontrol sinyal statis.

3.3.2.1 Kontrol Lampu Statis

Kontrol lampu statis memiliki mekanisme perubahan sinyal lampu secara berurutan dari merah, kuning ke hijau. Lampu lalu lintas kemudian dikontrol bergantian dengan durasi sinyal lampu hijau yang sudah ditentukan yaitu 10 detik, meninggalkan lampu lain menjadi sinyal merah seperti logika pada gambar 3.3.

```
Function LightManagement(requestLight):
    If isInTransition is True:
        Exit function
    Increment totalPhase
    Set isInTransition to True
    Set selectedLight status to YELLOW
    Set requestLight status to YELLOW
    Set ignoreYellow to False
    For each light in lights:
        For each vehicle in light.vehiclesQueueIntersection:
            If vehicle is null or not active:
                Continue loop
            Get vehicleAI from vehicle
            If vehicle is STOPPED and on intersection:
                Set ignoreYellow to True
    Wait until:
        For each vehicle in vehiclesQueueIntersection:
            If vehicle is null or not active:
                Remove vehicle from list
        Continue waiting until:
            vehiclesCount is 0 OR ignoreYellow is True
    Set selectedLight status to RED
    Set selectedLight to requestLight
    Set selectedLight status to GREEN

    If intersectionType is not AI AND evaluationMode is True:
        Call EvaluateReward()
        Add totalPassedVehiclePhase to totalPassedVehiclesEpisode
        Add new demand vehicles to totalDemandVehiclesEpisode
        Call GetAllLanesTrafficStatistics()
    Call CheckEnd()
    Set isInTransition to False
    Panggil PeriodicLights setelah detik lightsDuration
End Function
```

Gambar 3. 3 Algoritma kontrol sinyal statis

3.3.2.2 Kontrol Agent

Kontrol *agent* merupakan penerapan penggunaan AI pada kontrol sinyal persimpangan, penerapan ini dilakukan dengan logika perulangan permintaan pembuatan keputusan pada *agent* ketika terdapat kendaraan pada area persimpangan. logika ini kemudian diteruskan pada GRPC *Unity* sebagai penguhubung ke python untuk menjalankan *model* DQN.

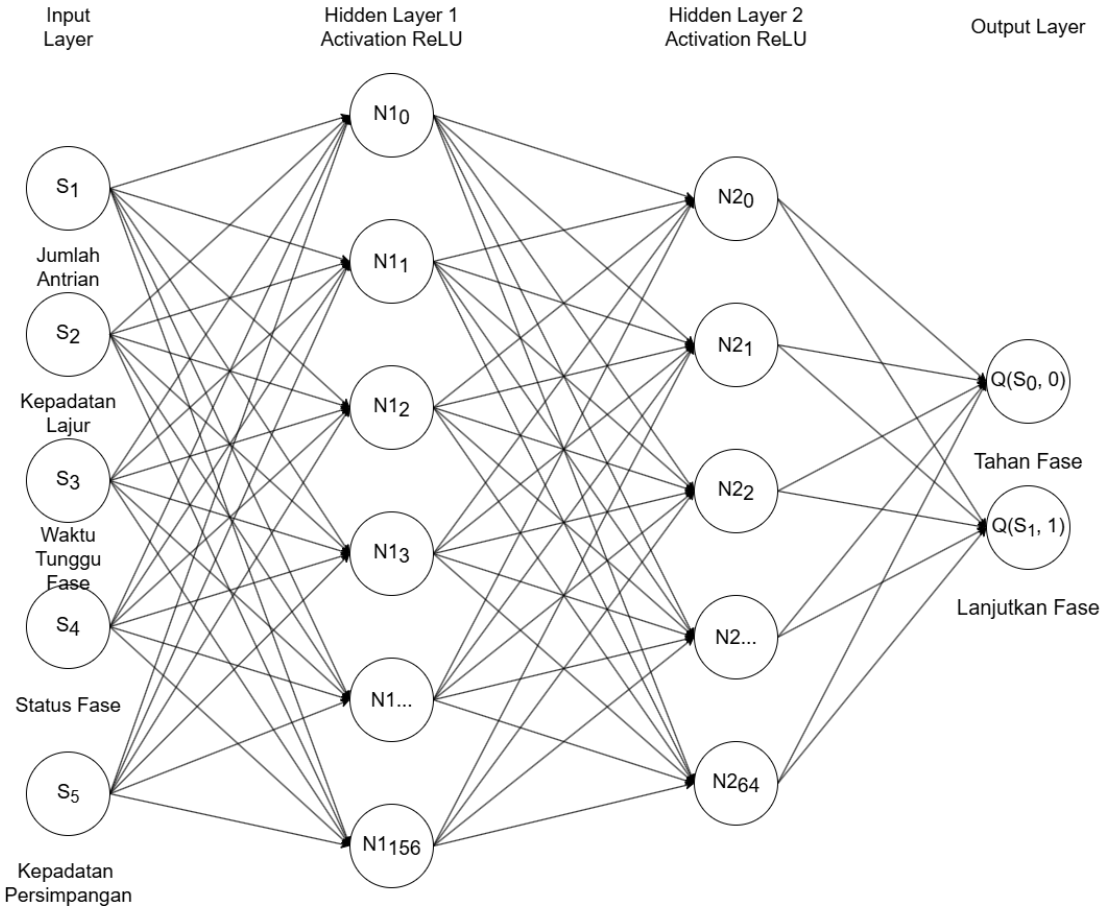
Beberapa hal yang perlu dipersiapkan untuk menerapkan *agent* DQN pada kontrol sinyal ini adalah dengan membuat arsitektur DQN yang terdiri dari Q-Network dan Target Network, *replay buffer*, kemudian menentukan *hyperparameter*. Persiapan tersebut dapat diuraikan sebagai berikut.

3.3.2.2.1 Q-Network

Q-Network pada DQN ini merupakan Jaringan perhitungan utama yang menggunakan *Neural Network* untuk menentukan aksi kontrol sinyal. Aksi kontrol sinyal ditentukan dengan menghitung dan memberikan nilai-Q pada aksi. Nilai-Q ini akan diperingkat dari yang terkecil hingga terbesar kemudian aksi yang memiliki nilai-Q terbaik akan dieksekusi. *Q-Network* pada simulasi Traffic Jam dirancang seperti berikut.

A. Arsitekur Q-Network

Arsitektur *q-network* terdiri dari 4 *layer* dengan arsitektur *neural network* pada gambar 3.4.



Gambar 3. 4 Desain Neural Network Dalam DQN

B. Input Layer

Terdapat 1 input layer dengan state dimension berupa data numerik yang didapat dari environment yaitu jumlah kendaraan, kepadatan lajur, waktu tunggu, kepadatan lajur yang dikalikan dengan banyaknya lajur (lokal) serta fase (menunjukkan giliran lampu yang sedang beroperasi) dan kepadatan zona persimpangan (global).

C. Hidden Layer

Pada penelitian ini menggunakan fully connected layer yang menghubungkan setiap neuron. Terdapat 2 hidden layer dengan detail arsitektur hidden layer pada tabel 3.2 dibawah.

Tabel 3. 2 Detail Arsitektur Hidden Layer DQN

Hidden Layer	Jumlah Neuron	Fungsi Aktivasi
Hidden Layer 1	156	ReLU
Hidden Layer 2	156	ReLU

Pada tiap *hidden layer l* memiliki perhitungan untuk dengan persamaan berikut:

$$h^l = \sigma\,(W^l h^{l-1} + b^l) \tag{3. 8}$$

Dimana:

- h^{l-1} merupakan *output* layer sebelumnya (jika $l = 1$ maka input layer).
- W^l merupakan bobot matriks *hidden layer l.*
- b^l merupakan bias vektor *hidden layer l.*
- σ merupakan aktivasi fungsi ReLU.

Pada *model* ini menggunakan *Rectified Linear Unit* (ReLU) sebagai aktivasi fungsi dengan persamaan berikut:

$$f(x) = \max(0, x) \tag{3. 9}$$

dimana input akan di-*output*kan langsung jika positif; selain itu di-*output*-kan nol.

D. Output Layer

Output layer terdiri dari 2 neuron yang merepresentasikan aksi pengoperasian persimpangan yaitu, Tahan Fase dan Lanjutkan Fase. *Output* yang diperoleh dari *q-network* adalah vektor nilai-q untuk tiap *neuron output layer* berupa $Q(s,a;\theta)$ untuk tiap kemungkinan aksi ***a*** pada *state* ***s*** dengan persamaan berikut:

$$Q(s, a\cdot;\theta) = W^{out}h^{L} + b^{out} \quad (3.\ 10)$$

Dimana:

- h^{L} merupakan *output hidden layer* sebelumnya.
- W^{out} dan b^{out} merupakan bobot dan bias *output layer*.
- Hasil vektor $Q(s, a\cdot;\theta)$ berisi nilai-q untuk semua kemungkinan aksi pada *state* ***s***.

3.3.2.2.2 Target Network

Pada pelatihan *agent*, DQN menggunakan *Target-network* sebagai pendukung untuk membandingkan kalkulasi nilai-Q *Target-network* dengan *Q-network* untuk mendapatkan nilai *Loss*. Selisih *loss* kedua *network* ini membantu DQN mencapai konvergensi lebih cepat.

Target-network sendiri memiliki arsitektur yang sama dengan *Q-network* namun *target network* tidak dilatih melainkan di-*update* bobotnya dengan menyalin *Q-network* tiap (t) langkah yang ditentukan, peng-*update*-an ini dinamakan dengan *learning rate*.

Target-network digunakan untuk menstabilkan *output* nilai-Q dari *Q-network* dengan membandingkan hasil *output* nilai-Q dari *Q-network* dan Target *network* kemudian menghitungnya menggunakan metode *Mean Squared Error* (MSE) untuk

mendapatkan *network loss* berupa selisih *output* kedua *network* tersebut kemudian meng-*update* kedua bobot *network* tersebut dan melanjutkan *training* sehingga mendapatkan konvergensi fungsi nilai-Q kedua *network*. Perhitungan yang digunakan oleh *target network* merupakan persamaan *Bellman* seperti berikut:

$$y = r + \gamma \max_{a'} Q_{\text{target}}(s', a') \quad (3.\ 11)$$

Dimana:

- y = *target nilai-q.*
- a' = aksi yang ditentukan.
- r = *reward* yang diperoleh.
- s' = *state* selanjutnya yang akan terjadi.
- γ = *discount factor.*
- $\max_{a'} \boldsymbol{Q_{target}}(s', a')$ = representasi nilai-q maksimum untuk *state* selanjutnya s'.

3.3.2.2.3 Replay Buffer

Replay Buffer digunakan sebagai pengumpulan dan augmentasi data pengalaman yang telah dilakukan *agent*. Pengalaman ini selanjutnya akan digunakan untuk melatih *Q-network* untuk menentukan aksi yang lebih baik dan sesuai pada *environment*.

Struktur *replay buffer* terdjrj darj penambahan pengalaman dan penyampelan *mini-batch* saat *training.* Komponen *replay buffer* dapat diimplementasikan dengan menerapkan persamaan sebagai berikut:

1. Storage Pengalaman

Storage pengalaman merupakan penyimpanan pengalaman interaksi yang telah dilakukan pada *environment game*. Struktur penyimpanan tersebut dapat direpresentasikan dengan ***D*** yang merepresentasikan *replay buffer*, dimana ***D*** memiliki kapasitas tetap ***N***. Tiap pengalaman $\boldsymbol{e_t}$ merupakan *tuple* berisikan:

$$\boldsymbol{e_t} = (s_t, a_t, r_t, s_{t+1}, \textbf{done}) \quad (3.\ 12)$$

Dimana:

- s_t, = *state* pada waktu *t*,
- a_t = aksi ditentukan pada waktu *t*,
- r_t, = *reward* yang diterima setelah menentukan aksi a_t,
- s_{t+1}, = *state* selanjutnya setelah menentukan aksi $a_{t,}$
- **done** = penanda *boolean* mengindikasikan akhir episode.

Buffer menyimpan kumpulan *tuple* ini untuk *sampling* berikutnya.

2. Operasi Penambahan (Memasukkan Pengalaman)

Replay buffer memiliki ukuran **N** tetap. Ketika *buffer* penuh, pengalaman lama akan digantikan oleh yang baru secara sirkular.

$$\boldsymbol{D}[i] = e_t \tag{3. 13}$$

Dimana:

- $i = t$ **mod** $\boldsymbol{N}$, merepresentasikan indeks dalam *buffer* (dengan operator *modulus* untuk memastikan Ketika $t \geq \boldsymbol{N}$, *buffer* mulai menggantikan pengalaman lama).
- t merupakan langkah waktu sekarang.

Jadi, untuk menambahkan pengalaman $\boldsymbol{e_t}$ ke dalam buffer:

$$\boldsymbol{D}[\boldsymbol{t} \textbf{ mod } \boldsymbol{N}]=(s_t,a_t,r_t,s_{t+1}, \textbf{done}) \tag{3. 14}$$

Persamaan ini memastikan pengalaman lama digantikan secara sirkular.

3. Penyampelan Pengalaman

Sampling pengalaman diperlukan untuk *training*, dengan menyampel *mini-batch* dengan ukuran $\boldsymbol{K}$ dari *replay buffer*. Untuk memastikan stabilitas, pengalaman disampel acak untuk memutuskan korelasi *temporal*.

Misalkan $\boldsymbol{K}$ merupakan *mini-batch*. *Mini-batch* pengalaman yang disampel dinotasikan menjadi:

$$\boldsymbol{B} = \{e_{i1},e_{i2},\ldots,e_{i\boldsymbol{K}}\}, \text{ dimana } i_1,i_2,\ldots,i_K \in \{1,2,\ldots,\min(t, \boldsymbol{N})\} \tag{3. 15}$$

Ini memilih $\boldsymbol{K}$ pengalaman dari *buffer* secara acak. Pemilihan indeks $\boldsymbol{i_j}$ dibentuk secara acak dari jarak pilihan yang tersedia dalam *buffer*.

4. Sampel Batch untuk *Training*

Ketika *mini-batch* ***B*** telah disampel, tiap komponen(*state*, aksi, *reward*, *state* selanjutnya, done *flag*) digunakan untuk meng-*update network.* Sampel *batch* ini berisikan:

$$\boldsymbol{B} = \{(s_i, a_i, r_i, s_{i+1}, \text{done}_i)\}_{i=1}^{\boldsymbol{K}} \quad (3.\ 16)$$

Dimana ***K*** merupakan ukuran *mini-batch*, dan tiap *tuple* dalam *mini-batch* diambil secara acak dari *replay buffer*.

5. Kapasitas Replay Buffer

Ukuran *buffer* biasanya sudah dikonfigurasi sebelumnya sehingga hanya perlu menentukan kapasitas maksimum ***N.*** Pada *buffer replay* terdapat fungsi untuk mencegah *memory overflow* (kelebihan memori untuk disimpan), karena hanya pengalaman ***N*** terbaru yang akan tetap disimpan. Fungsi ini berlaku dengan logika berikut:

if | D | ≥ **N**, timpa pengalaman terlama. (3. 17)

3.3.2.2.4 Hyperparameter Setup

Hyperparameter merupakan penyesuaian yang dilakukan secara eksperimental untuk *training network* dan memiliki peran penting dalam menentukan *model* seberapa baik *agent* belajar.

1. Learning Rate

Learning rate digunakan untuk mengontrol penyesuaian pembobotan jaringan terhadap *error gradient* pada tiap langkah selama perhitungan *loss function* agar jaringan tetap bergerak meminimalisir *error* dengan banyaknya langkah dikontrol oleh ***α***. DQN *model* menggunakan *gradient descent* untuk mengupdate bobot *q-network* dengan aturan berikut.

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla \theta L(\theta_t) \quad (3.\ 18)$$

Dimana:

- θ_t = bobot pada waktu *t*.
- α = *learning rate.*
- $\nabla \theta L(\theta)$ = gradien *loss function* terhadap bobot pada waktu *t*.

2. Discount Factor (γ)

Discount factor menentukan pentingnya *reward* mendatang. Nilai γ mendekati 1 mementingkan *reward* mendatang dan nilai γ mendekati 0 membuat *agent* mementingkan *reward* kini.

3. Replay Buffer Size

Replay buffer size merupakan jumlah pengalaman yang daptat disimpan dalam *replay buffer*. Semakin besar ukuran *replay buffer* membantu mencegah *overfitting* jaringan. *Replay buffer* menyimpan pengalaman dalam bentuk *tuple* (*s, a, r, s',* **done**). Dimana *s* merupakan *state, a* merupakan aksi *r*, merupakan *reward, s'* merupakan *state* selanjutnya, **done** merupakan penanda indikasi episode selesai. Umumnya ukuran replay *buffer* adalah 10.000 sampai 1.000.000 pengalaman. Peneliti akan menyesuaikan ukuran berdasarkan kemampuan hardware pengembangan.

4. Batch Size

Batch size merupakan ukuran jumlah pengalaman yang dapat disampel dari replay buffer untuk training pada tiap langkah. Semakin besar peng-update-an bobot semakin stabil namun membutuhkan banyak memori.

5. Frekuensi Update Target Network

Seberapa sering peng-*update*-an bobot *target network* dilakukan untuk menyamakan bobot *q-network*. Frekuensi *update target network* berlaku dengan:

$$\theta^{-} \leftarrow \theta \tag{3. 19}$$

yaitu untuk tiap C langkah, bobot target network akan di-update. Semakin banyak frekuensi update menyebabkan ketidakstabilan sedangkan semakin sedikit frekuensi update meningkatkan kestabilan. Kisaran update yang digunakan berada pada tiap 1000 sampai 10000 langkah.

6. Epsilon (ε)

Epsilon menentukan kemungkinan *agent* melakukan aksi acak untuk eksplorasi *environment* dibanding aksi yang disarankan oleh *q-network*. *Epsilon* menyeimbangkan eksplorasi dan eksploitasi. Semakin mendekati 1 berarti semakin banyak kemungkinan eksplorasi dan mendekati 0 semakin banyak kemungkinan eksploitasi. Pengaturan nilai **ε** pertama kali akan diatur menjadi mendekati 1 untuk mengeksplorasi *environment* kemudian berkurang sejalan dengan banyaknya episode yang terselesaikan

3.3.3 Desain Input

Desain input merupakan bagaimana sistem mendapatkan inputan dari *player* untuk dikelola dalam pelatihan *agent*. Pada *game* ini *player* memberikan inputan pada sistem sebagai skenario episode pelatihan *agent* yang berupa:

1. Lokasi perimpangan
2. Jumlah mobil.

Dalam perancangan kontrol sinyal *agent* desain input diwakilkan dengan inisialisasi *Markov Decision Process* (MDP) pada sistem *environment* game. *Markov Decision Process* merupakan penerjemahan kondisi *environment* pada *game* menjadi fungsi matematis yang dapat diperhitungkan oleh *agent* untuk membuat keputusan dan mempelajari *game* tersebut. Sedangkan inisialisasi MDP bertumpu pada penentuan kondisi-kondisi lalu lintas, penentuan aksi atau tindakan, dan aturan permainan yang berpengaruh terhadap kelancaran lalu lintas. Komponen inisialisasi tersebut ditentukan sebagai berikut.

1. Penentuan State (s)

State merupakan inputan yang diterima *agent* untuk berinteraksi dengan *environment.* Ketika simulasi berjalan *state* tersebut akan diambil dari persimpangan dan diperhitungkan untuk menentukan aksi, tabel 3.3 merupakan faktor kelancaran arus kemacetan yang diimplemetasikan:

Tabel 3. 3 Input State

State 1	Jumlah kendaraan
State 2	Kepadatan Arus
State 3	Waktu Tunggu Tiap Fase
State 4	Overview Grup Sinyal
State 5	Kepadatan Persimpangan

2. Aksi (a)

Aksi merupakan hal yang bisa dilakukan *agent* dalam *game*. Dalam *game* simulasi ini aksi *agent* dapat melakukan hal berikut:

A. Memilih menahan fase.

B. Melanjutkan fase.

Dimana pengaturan fase sudah dilakukan dengan pemrograman *behavior environment* termasuk pengubahan sinyal dan manajemen lampu yang bergilir maju secara bergantian untuk memastikan semua lajur mendapatkan giliran berjalan.

3. *Reward* (r)

Reward pada DQN berjalan dengan 2 hal yaitu:

A. Immediate *Reward*

Immediate *reward* merefleksikan keuntungan dari melakukan aksi yang dilakukan. *agent* mendapatkan *reward* langsung r_t dari *Environment* setelah melakukan aksi a_t dalam *state* s_t.

B. Objektif

Agent memiliki tujuan yaitu untuk memaksimalkan *reward* secara kumulatif seiring berjalannya *training*. Dimana *reward* digunakan sebagai pendorong *agent* untuk menentukan *behavior* yang terbaik untuk menyelesaikan masalah pada situasi kemacetan sekaligus menentukan kemampuan pembelajaran *agent*.

Kemampuan pembelajaran *agent* dipengaruhi oleh penentuan *reward* dan *penalty* yang didapatkan dari tiap aksi yang dilakukan oleh *agent*. Penentuan *reward* dan *penalty* ini disebut dengan *policy* berperan untuk membuat aturan *reward* yang bijak agar *agent* tidak mengeksploitasi dan membuat *agent* belajar dengan benar. Aturan *reward* didasarkan pada logika syarat-syarat yang mendorong *agent* untuk mengevaluasi faktor-faktor yang berpengaruh pada kemacetan.

Dikarenakan pengembangan *agent* ditujukan untuk penguraian kemacetan maka *policy reward* dengan dibentuk dengan pembobotan yang lebih diberatkan pada keberhasilan penyebrangan persimpangan dan penguraian kepadatan. Pembobotan *reward* didistribusikan pada tabel 3.4 berikut.

Tabel 3. 4 Aturan Sistem Reward

Reward	**Policy**	**Skor**
Positif	Kendaraan berhasil melewati persimpangan tiap fase	+ 0.5
Positif	Menjaga kepadatan keseluruhan lajur	+ 0.2
Positif	Menjaga waktu tunggu lajur	+ 0.3
Negatif	Menyalakan lampu hijau pada lajur yang kosong	- 0.1

4. State mendatang ($s^{'}$)

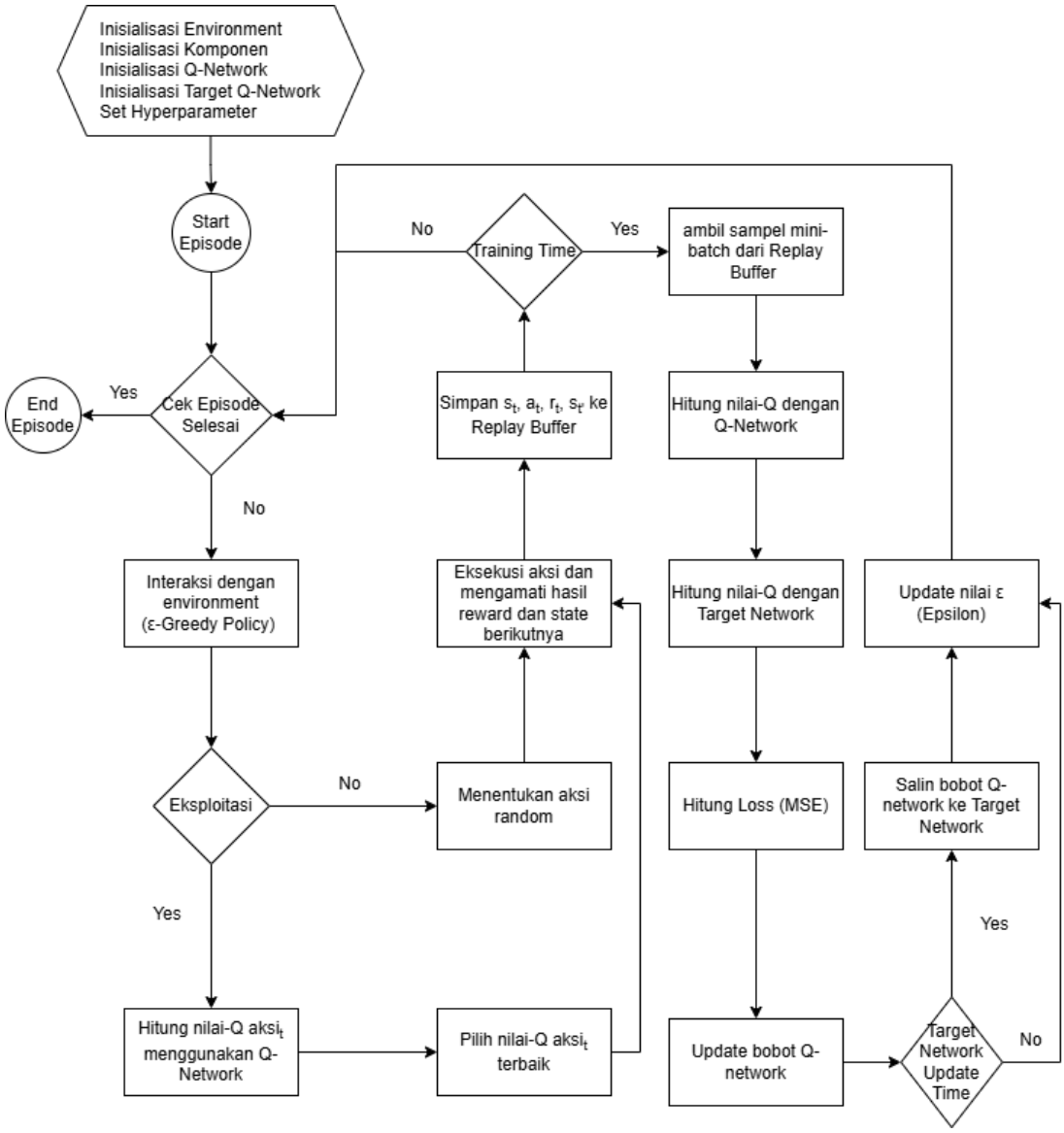
State mendatang merupakan perubahan keadaan *environment* setelah mengeksekusi aksi yang telah ditentukan. s' digunakan sebagai acuan yang diamati *agent* untuk pembelajaran pengambilan keputusan aksi yang lebih baik berikutnya.

3.3.4 Desain Proses

Dengan inisialisasi MDP yang sudah diterapkan, maka desain proses *game* Traffic Jam adalah integrasi kontrol *agent* dengan *model* DQN untuk menghasilkan *output* berupa *agent* dengan *decision-making* yang optimal untuk mengontrol durasi lampu lalu lalu lintas.

3.3.4.1 Mekanisme Interaksi Agent dengan Environment

Proses yang dikelola pada *game* ini merupakan semua *state*, aksi dan *reward* pada *environment* yang dapat diinteraksi oleh *agent* untuk menghitung nilai-q menggunakan *Neural Network*. Gambar 3.5 merepresentasikan bagaimana mekanisme pengelolaan interaksi *environment* dengan model *Deep Q-Network* yang digunakan pada penelitian ini.



Gambar 3. 5 Desain Sistem DQN

1. Inisialisasi Environment

Tahap pertama yang dilakukan untuk membangun mekanisme ini adalah Inisialisasi *environment.* Selain menginisialisasi dimensi *state*, dimensi aksi dan *reward game* diperlukan juga inisialisasi *hyperparameter* yang dibutuhkan *model* DQN untuk diinteraksi. Pada inisialisasi ini terdapat 2 proses yaitu:

A. Reset Environment

Membuat fungsi memulai ulang *environment* dengan kondisi semula dan menghapus data *state* sebelumnya. Ini ditandai dengan inisialiasi episodeBegin() dan episodeEnd() pada *environment*. Pada *game* ini *reset environment* diterapkan dalam pengontrolan lampu persimpangan dengan 50 kali langkah pengubahan fase lampu yang dilakukan *agent*. Ketika pengubahan mencapai 50 fase maka akan episodeEnd() akan di-*trigger* dan memanggil fungsi episodeBegin() serta memulai ulang langkah ke 1 dengan mengosongkan area persimpangan dan men-*deploy* ulang kendaraan.

B. Initial State

Berisikan kondisi parameter pada *environment* ketika *training* dimulai. Pada simulasi ini *initial state* berupa fase persimpangan, kendaraan.

2. Epsilon-Greedy Policy

Tahap selanjutnya yaitu penerapan *epsilon-greedy*. Epsilon-greedy (*ε-greedy*) merupakan *behavior agent* sekaligus strategi *agent* untuk mempelajari *environment* dan menggunakan pengetahuannya untuk mendapatkan *reward* sebanyak mungkin dengan mengurai kemacetan pada persimpangan. *Behavior* ini bekerja dengan fungsi matematis nilai parameter yang disebut dengan *epsilon* untuk menentukan peluang *agent* dalam berinteraksi dengan *environment*.

Terdapat dua interaksi yang dapat dilakukan oleh *agent* yaitu eksplorasi dan eskploitasi. Eksplorasi merupakan interaksi *agent* untuk melakukan aksi acak demi mempelajari *environment*. Eksploitasi merupakan interaksi *agent* menggunakan DQN untuk menghitung kemungkinan aksi terbaik yang dapat dilakukan.

Penentuan interaksi ini dipengaruhi oleh nilai *epsilon* dimana semakin rendah *epsilon* maka peluang *agent* untuk melakukan eskploitasi *environment* menggunakan DQN akan semakin besar. *ε-greedy* bekerja dengan persamaan berikut:

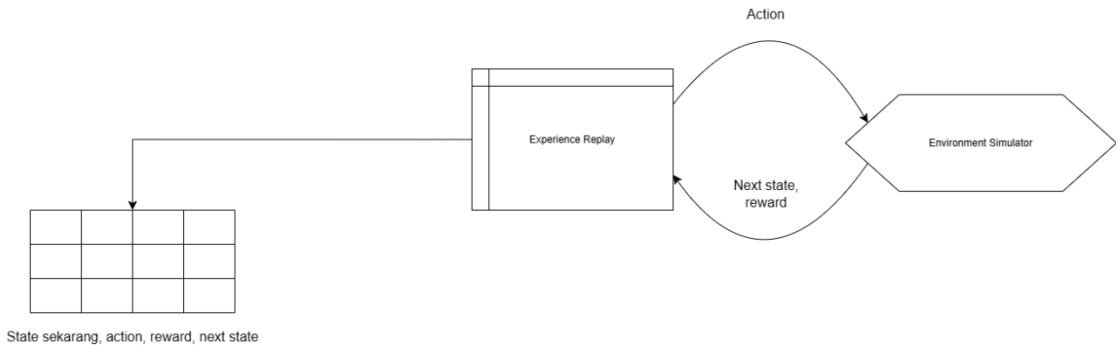
$$a_t = \begin{cases} \text{aksi acak} & \text{dengan kemungkinan } \epsilon \\ \arg\max_a Q(s_t, a) & \text{dengan kemungkinan } 1 - \epsilon \end{cases} \quad (3.\ 20)$$

Dimana:

- a_t = aksi yang ditentukan pada langkah waktu *t.*
- s_t = *state* sekarang pada langkah waktu *t.*
- $Q(s_t, a)$ = nilai-q (harapan *reward* yang akan didapat) dengan menentukan aksi *a* pada *state* s_t.
- ε = parameter yang mengatur kemungkinan penentuan aksi acak (eksplorasi).
- 1 – ε = kemungkinan menentukan aksi dengan nilai-q tertinggi (eksploitasi).

3.3.4.2 Koleksi Experience

Interaksi yang telah dilakukan melalui *ε-greedy* selanjutnya akan melalui tahap koleksi *experience,* yakni penyimpanan hasil interaksi. Pada tahap ini *agent* akan melakukan 3 proses yang dijelaskan pada gambar 3.6 berikut.



Gambar 3. 6 Proses interaksi dengan environment dan koleksi experience

1. Eksekusi Aksi

   agent akan mengeksekusi aksi yang telah ditentukan oleh kemungkinan ε-*greedy* ke dalam *environment*. Aksi ini bisa berupa aksi acak untuk eksplorasi atau aksi eksploitasi (menggunakan *q-network*).

2. Observasi Reward

   Pada proses ini *agent* akan mengamati *reward* yang diperoleh dan *state* (s_{t+1}) setelah melakukan aksi *a* dengan *state* (s_t).

3. Simpan Experience

   Setelah mengamati hasil *agent* akan menyimpan pengamatannya dalam bentuk pengalaman tuple (s_t, a_t, r_t, s_{t+1}, **done**) ke dalam *Replay Buffer*

3.3.4.3 Fase Training

Setelah pengalaman interaksi *agent* dengan *environment* terkumpul maka selanjutnya *agent* akan melalui fase *training*. Fase *training* ini akan menjelaskan bagaimana proses *training* terjadi dan penjelasan bagaimana implementasi perhitungan manual *training* DQN dengan contoh skenario berikut.

- Dimensi *state* (jumlah kendaraan, volume arus, kepadatan arus, status sinyal), Contoh *state* (15, 50, 0.7, [merah, hijau, merah, merah]).
- Dimensi aksi (lampu merah, lampu kuning, lampu hijau, controller sebelumnya, controller selanjutnya).
- DQN *network*: *hidden layer* 1 = 128 - ReLU, *hidden layer* 2 = 64 - ReLU (keduanya disederhanakan menjadi 2 neuron untuk mempermudah perhitungan), *output* layer = 5 neuron (untuk tiap aksi).
- Untuk menyederhanakan status sinyal dalam bentuk vektor menjadi merah = 0, hijau = 1, kuning = 2.
- Status sinyal [merah, hijau, merah, merah] → [0, 1, 0, 0].
- Jadi *input layer* DQN dapat direpresentasikan dengan vektor 7 dimensi seperti berikut:

$$s_t = \begin{bmatrix} 15 \\ 50 \\ 0.7 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

1. Pengambilan Sampel Batch

Pada tahap ini *agent* akan mengambil sampel pengalaman (s_t, a_t, r_t, s_{t+1}, **done**) dalam bentuk *mini-batch* dari *replay buffer* secara acak untuk *training*.

2. Prediksi Nilai-Q

Dalam *training* DQN, *q-network* dan *target network* akan digunakan untuk memprediksi nilai-q aksi. Dimana pada *target network* nilai-q disebut dengan target nilai-q. Masing-masing *network* bekerja dengan berikut:

A. *Q-Network*

Untuk memprediksi nilai-q tiap aksi ***a*** pada *state* ***s*** *q-network* menggunakan persamaan berikut:

$$\boldsymbol{Q}\ (s_t, a_0; \theta) \tag{3. 21}$$

$$\boldsymbol{Q}\ (s_t, a_1; \theta) \tag{3. 22}$$

$$\boldsymbol{Q}\ (s_t, a_2; \theta) \tag{3. 23}$$

$$\boldsymbol{Q}\ (s_t, a_3; \theta) \tag{3. 24}$$

$$\boldsymbol{Q}\ (s_t, a_4; \theta) \tag{3. 25}$$

Dimana:

- s_t = *state* yang sedang terjadi pada *agent.*
- a = aksi yang ditentukan.
- θ = bobot/parameter *neural network.*
- $\boldsymbol{Q}\ (s_t, a; \theta)$ = prediksi nilai-q

Untuk implementasi contoh skenario pertama akan memperhitungkan *hidden layer* 1 (2 neuron untuk menyederhanakan) dengan memisalkan bobot dan bias seperti berikut:

$$w_1 = \begin{bmatrix} 0.1 & -0.2 & 0.05 & 0.1 & 0.03 & 0.2 & 0.4 \\ -0.1 & 0.15 & -0.25 & 0.05 & 0.2 & -0.05 & 0.1 \end{bmatrix}, b_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Hitung jumlah bobot pada hidden layer 1:

$$z_1 = W_1^T + b_1 = \begin{bmatrix} 0.1 & -0.1 \\ 0.2 & 0.15 \\ 0.05 & -0.25 \\ 0.1 & 0.05 \\ 0.3 & 0.2 \\ 0.2 & -0.05 \\ 0.4 & 0.1 \end{bmatrix} \cdot \begin{bmatrix} 15 \\ 50 \\ 0.7 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.1 \\ -0.1 \end{bmatrix}$$

$$z_1 = \begin{bmatrix} -8.865 \\ 5.925 \end{bmatrix}$$

Terapkan aktivasi ReLU:

$$a_1 = ReLU(z_1) = \begin{bmatrix} 0 \\ 5.925 \end{bmatrix}$$

Selanjutnya perhitungan *hidden layer* 2 (2 neuron), dengan misal bobot dan bias berikut:

$$w_2 = \begin{bmatrix} 0.2 & -0.1 \\ -0.3 & 0.15 \end{bmatrix}, \; b_2 = \begin{bmatrix} 0.05 \\ 0.05 \end{bmatrix}$$

Hitung *output hidden layer 2:*

$$z_2 = w_1^T a_1 + b_2 = \begin{bmatrix} 0.2 & -0.3 \\ -0.1 & 0.15 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 5.925 \end{bmatrix} + \begin{bmatrix} 0.05 \\ 0.05 \end{bmatrix}$$

$$z_2 = \begin{bmatrix} -0.3 \times 5.925 + 0.05 \\ 0.15 \times 5.925 + 0.05 \end{bmatrix} = \begin{bmatrix} -1.7775 + 0.5 \\ 0.88875 + 0.05 \end{bmatrix}$$

$$z_2 = \begin{bmatrix} -1.7275 \\ 0.93875 \end{bmatrix}$$

Terapkan ReLU:

$$a_2 = ReLU(z_2) = \begin{bmatrix} 0 \\ 0.93875 \end{bmatrix}$$

Perhitungan *output layer* (nilai-q) dengan bobot dan bias misalkan berikut:

$$w_3 = \begin{bmatrix} 0.1 & -0.2 & 0.3 & -0.4 & 0.5 \\ -0.1 & 0.2 & -0.3 & 0.4 & -0.5 \end{bmatrix}, b_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Hitung nilai-q:

$$z_3 = w_3^T a_2 + b_3 = \begin{bmatrix} 0.1 & -0.1 \\ -0.2 & 0.2 \\ 0.3 & -0.3 \\ -0.4 & 0.4 \\ 0.5 & -0.5 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0.93875 \end{bmatrix}$$

$$z\_3 = \begin{bmatrix} -0.1 \times 0.93875 \\ 0.2 \times 0.93875 \\ -0.3 \times 0.93875 \\ 0.4 \times 0.93875 \\ -0.5 \times 0.93875 \end{bmatrix} = \begin{bmatrix} -0.094875 \\ 0.18775 \\ -0.281625 \\ 0.3755 \\ -0.469375 \end{bmatrix}$$

Diatas merupakan prediksi nilai-q untuk tiap aksi.

Kemudian *agent* akan menentukan aksi dengan nilai-q terbaik dengan persamaan berikut:

$$a^* = \arg max_a \, \mathrm{Q}(\mathrm{s}_t, \mathrm{a}) \qquad (3.\ 26)$$

Namun pada contoh skenario misalkan *agent* menentukan eksplorasi dengan memilih -0.094875 yaitu aksi lampu merah.

B. Target Network

Target network akan memprediksi nilai-q target untuk tiap aksi ***a'*** pada *state* ***s'*** yang telah disampel dari *mini-batch* akan dikalkulasi *target network* menggunakan persamaan *Bellman*. Untuk tiap pengalaman dalam *mini-batch*:

$$y = \begin{cases} r & \text{jika episode berakhir pada } s' \\ r + \gamma \max_{a'} \boldsymbol{Q}_{\text{target}}(s', a') & \text{selain itu} \end{cases} \qquad (3.\ 27)$$

- Jika episode berakhir *s*', target nilai-q adalah *reward*
- Selain itu,
    - y = target nilai-q
    - r = *immediate reward*
    - $\gamma \max_{a'}$ = maksimum *discount factor* nilai-q yang memboboti *reward* jangka panjang untuk *state* berikutnya *s'*
    - $\boldsymbol{Q}_{\text{target}}(s', a')$ = perhitungan *target network*.

Target network berfungsi untuk memberikan estimasi stabil *reward* jangka panjang karena bobot *target network* lebih sedikit di-*update* daripada *q-network*. Pada contoh skenario yang terjadi misalkan:

- *Reward*: $r = 10$
- *Discount Factor*: $\gamma = 0.99$
- Prediksi target nilai-q untuk *state* selanjutnya *s'*: $Q'(s',a') = [-1.2, 0.5, 0.9, 0.3, 0.7]$.

Untuk mencari target nilai-q maka mencari nilai-q' maksimum:

$$\max\big(Q'(s', a')\big) = max([-1.2, 0.5, 0.9, 0.3, 0.7]) = 0.9$$

Maka target nilai-q:

$$Q_{target} = 10 + 0.99 \cdot 0.9 = 10 + 0.891 = 10.891$$

Hasil nilai-q target 10.891 akan digunakan untuk mengupdate bobot *network* selama *training* menggunakan *Mean Squared Error*.

3. Perhitungan Loss Function menggunakan MSE

Perhitungan *loss function* untuk mengukur seberapa meleset prediksi *q-network* dengan prediksi target-*network* perhitungan ini dilakukan menggunakan *Mean Squared Error* (MSE)

$$\text{Loss} = \frac{1}{2}\ (y - \boldsymbol{Q}_{main}\ (s,a))^2 \qquad (3.\ 28)$$

Dimana:

- y = target nilai-q.
- $\boldsymbol{Q}_{main}\ (s,a)$ = perhitungan *target network*.

Loss ini mengukur secara kuantitas perbedaaan prediksi nilai-q dan target nilai-q. loss besar mengindikasikan jauhya kemelesetan prediksi q-network dari target network, sedangkan semakin kecil berarti network lebih akurat.

Untuk implementasi perhitungan pada contoh skenario dengan:

- Nilai-q: $\widehat{\mathrm{Q}}(\mathrm{s},\mathrm{a}) = -0.093875.$
- Target nilai-q $Q_{target} = 10.891.$

Maka:

$$Loss = \frac{1}{2}\left(10.981 - (-0.093875)\right)^2$$

$$Loss = \frac{1}{2}(10.981 + 0.093875)^2 = \frac{1}{2}(10.984875)^2$$

$$Loss = \frac{1}{2} \times 120.666 = 60.333$$

Setelah *loss* ditemukan, *network* menggunakan *backpropagation* untuk menghitung *loss* gradien untuk bobot θ dari *q-network*. *Backpropagation* menghitung

seberapa banyak kontribusi bobot dalam *q-network* menyebabkan *error* (*loss*) dan menentukan bobot yang perlu disesuaikan untuk mengurangi *error* tersebut. Fungsi gradien *loss* ke nilai-q yaitu:

$$\nabla_\theta \text{ Loss} = (\text{Q}_{\text{main}}\ (s,a;\theta) - \text{y})\ \nabla_\theta\ \text{Q}_{\text{target}}(s',a';\theta')$$

$$\nabla_\theta \text{ Loss} = -0.093875 - 10.981 = -10.984875$$

Backpropagate gradien ke bobot:

$$\hat{Q}(s,a) = W_3^T a_2 + b_3 \qquad (3.\ 29)$$

Dimana:

- W_3 merupakan bobot matriks yang terhubung pada *hidden layer 2* ke *output*.
- a_2 merupakan aktivasi fungsi pada *hidden layer 2* ($\begin{bmatrix} 0 \\ 0.93875 \end{bmatrix}$).

Hitung gradien *loss* bobot W_3:

$$\frac{\partial Loss}{\partial W_3} = \frac{\partial Loss}{\partial \hat{Q}(s,a)} \cdot \frac{\partial \hat{Q}(s,a)}{\partial W_3} = -10.984875 \cdot a_2$$

Subsitusi a_2:

$$\frac{\partial Loss}{\partial W_3} = -10.984875 \cdot \begin{bmatrix} 0 \\ 0.93875 \end{bmatrix} = \begin{bmatrix} 0 \\ -10.30955 \end{bmatrix}$$

Persamaan ini akan memberikan seberapa banyak tiap bobot θ dalam *network* yang perlu dirubah untuk meminimalisir *loss*.

4. Update (Gradient Descent) dan Pengulangan

Setelah gradien *loss* ditemukan, bobot *q-network* akan diupdate menggunakan *Gradien Descent* dimana bobot *q-network* disesuaikan dengan arah untuk meminimalisir *loss*:

$$w \leftarrow w - \alpha \cdot \frac{\partial Loss}{\partial w} \tag{3. 30}$$

Dimana:

- w = bobot *q-network.*
- α = *learning rate.*
- $\frac{\partial Loss}{\partial w}$ = *loss* gradien terhadap bobot.

Bobot akan disesuaikan secara bertahap dimana ini akan meningkatkan kemampuan *network* memprediksi nilai-q secara akurat. Pada contoh skenario misal learning rate ($\alpha = 0.01$) maka:

$$W_3 = W_3 - \alpha \cdot \frac{\partial Loss}{\partial W_3}$$

Bobot awal untuk aksi lampu merah dalam W_3 yaitu:

$$W_3 = \begin{bmatrix} 0.1 \\ -0.2 \end{bmatrix}$$

Bobot W_3 yang diupdate akan menjadi:

$$W_3 = \begin{bmatrix} 0.1 \\ -0.2 \end{bmatrix} - 0.01 \cdot \begin{bmatrix} 0 \\ -10.30955 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.2 + 0.1031 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.0969 \end{bmatrix}$$

Selanjutnya proses meng-*update* bobot W_2 pada *hidden layer* 2, yaitu dengan menghitung gradien *hidden layer* 2 bobot W_2 yaitu:

$$\frac{\partial Loss}{\partial a_3} = \frac{\partial Loss}{\partial \hat{Q}(s,a)} \cdot W_3$$

$$\frac{\partial Loss}{\partial a_2} = -10.984875 - \begin{bmatrix} 0.1 \\ -0.0969 \end{bmatrix} = \begin{bmatrix} -1.0985 \\ 1.064 \end{bmatrix}$$

Hitung gradien *loss hidden layer* 2 bobot W_2:

$$a_2 = W_3^T a_2 + b_3$$

Dimana a_1 aktivasi dari *hidden layer* 1 adalah $a_1 = \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix}$, gradien *loss*nya adalah:

$$\frac{\partial Loss}{\partial W_2} = \frac{\partial Loss}{\partial a_2} \cdot \frac{\partial a_2}{\partial W_2} = \begin{bmatrix} -10.985 \\ 1 \end{bmatrix} \cdot a_1^T$$

$$\frac{\partial Loss}{\partial W_2} = \begin{bmatrix} -1.0985 \cdot 0.5 & -1.0985 \cdot 0.25 \\ 1.064 \cdot 0.5 & 1.064 \cdot 0.25 \end{bmatrix}$$

$$\frac{\partial Loss}{\partial W_2} = \begin{bmatrix} -0.54925 & -0.274625 \\ 0.532 & 0.266 \end{bmatrix}$$

Update hidden layer 2 bobot W_2 menggunakan *gradient descent*:

$$W_2 = W_2 - \alpha \cdot \frac{\partial Loss}{\partial W_2}$$

Jika $\alpha = 0.01$, dan bobot awal yaitu:

$$W_2 = \begin{bmatrix} 0.2 & -0.3 \\ -0.1 & 0.15 \end{bmatrix}$$

Bobot terupdatenya yaitu:

$$W_2 = \begin{bmatrix} 0.2 & -0.3 \\ -0.1 & 0.15 \end{bmatrix} - 0.01 \cdot \begin{bmatrix} -0.54925 & -0.274625 \\ 0.532 & 0.266 \end{bmatrix}$$

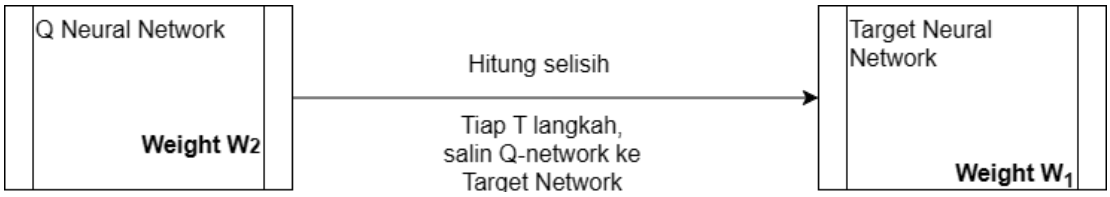
$$W_2 = \begin{bmatrix} 0.2 + 0.0054925 & -0.3 + 0.00274625 \\ -0.1 - 0.00532 & 0.15 - 000.266 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0.20549 & -0.29725 \\ -0.10532 & 0.14734 \end{bmatrix}$$

Selanjutnya yaitu mengulang proses untuk tiap sampel *mini-batch*. Bobot *q-network* akan di-*update* setelah tiap penyelesaian batch, meningkatkan prediksi seiring waktu. Semakin banyak *network* dilatih, semakin baik *network* dalam memprediksi nilai-q untuk berbagai jenis pasang aksi-kondisi.

3.3.4.4 Target Network Update

Setelah fase *training* selesai selanjutnya *target network* akan diupdate secara berkala. Update ini dilakukan umumnya tiap ribuan langkah *training* dan mengupdate dengan cara menyamakan bobot *q-network* dengan *target network* seperti pada gambar 3.7.



Gambar 3. 7 Update Target Network

Dengan *update* secara periodik memastikan *target network* menyediakan prediksi nilai-q yang lebih stabil untuk dipelajari *q-network* bersamaan dengan membuat *q-network* lebih adaptif terhadap pengalaman terbaru *agent.*

3.3.4.5 Epsilon Decay

Tahap terakhir adalah penerapan *epsilon decay*. Epsilon decay ini merupakan penurunan nilai *epsilon* seiring berjalannya *training.* Penunrunan nilai *epsilon* ini berfungsi untuk mengontrol keseimbangan antara eksplorasi dan eksploitasi seiring *agent* belajar. *Epsilon decay* menurunkan nilai ϵ seiring waktu untuk mendorong *agent* mengekspoitasi pengetahuannya. Ini sejalan dengan semakin *agent* mengertinya tentang *environment.* Pada penelitian ini menggunakan *exponential decay* dimana *epsilon* berkurang berdasarkan faktor ϵ_{decay} tiap langkah waktu:

$$\epsilon_{t+1} = \epsilon_{min} + (\epsilon_0 - \epsilon_{min}) \cdot e^{-\lambda t} \quad (3.\ 31)$$

Dimana:

- ϵ_0 = nilai awal *epsilon*.
- ϵ_{min} = nilai minimum *epsilon*, memastikan *agent* memiliki sedikit kemungkinan eksplorasi.
- λ = faktor *decay*, mengontrol kecepatan penurunan *epsilon*.
- *t* = langkah waktu atau episode sekarang.

3.3.5 Desain Output

Dari desain proses yang dirancang terdapat dua *output* yang dihasilkan. Mengikuti alur perancangan DQN desain *output* pertama adalah aksi *agent* yang dapat memainkan *game* dengan memberikan *decision making* secara optimal untuk mengatur sinyal lampu lalu lintas. Dimana *agent* memberikan durasi sinyal berdasarkan kondisi lalu lintas dan pengalaman *agent* pada tiap lampu lajur persimpangan dengan tiap jenis persimpangannya yaitu persimpangan tiga dan persimpangan empat.

Berdasarkan mekanisme interaksi *agent* dengan *environment*, desain *output* kedua merupakan data statistik pengamatan persimpangan. Data statistik ini menunjukkan dampak lalu lintas dari kinerja kontrol sinyal yang diterapkan. Data statistik ini kemudian diproses menjadi data penilaian evaluasi yang telah dibahas bada sub bab 3.2.2 yang selanjutnya dapat dianalisa, dievaluasi dan disimpulkan hasilnya. Data statistik pengamatan tersebut didistribusikan pada tabel 3.5 berikut:

Tabel 3. 5 Statistik Pengamatan

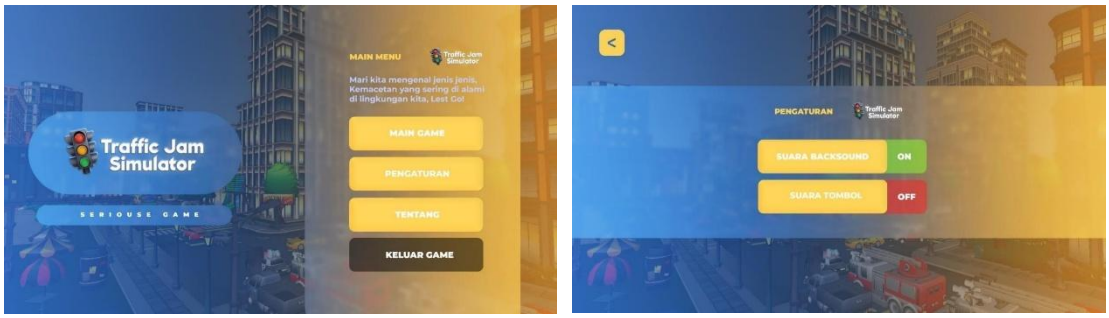
Tingkat Persimpangan	**Tingkat Jalur**
Episode	*lane{i}_avgVehicles*
Time	*lane{i}_avgDensity*
Reward	*lane{i}_avgVolume*
totalDemandVehicles	*lane{i}_avgWaitTime*
passedVehicles	
totalAvgVehiclesPerPhase	
totalAvgTrafficVolumePerPhase	
totalAvgTrafficDensityPerPhase	
lowestTrafficVolume	
highestTrafficVolume	
troughputEfficiency	
totalWaitTime/vehicle	
reward/vehicle	
density/vehicle	

BAB IV

UJI COBA DAN PEMBAHASAN

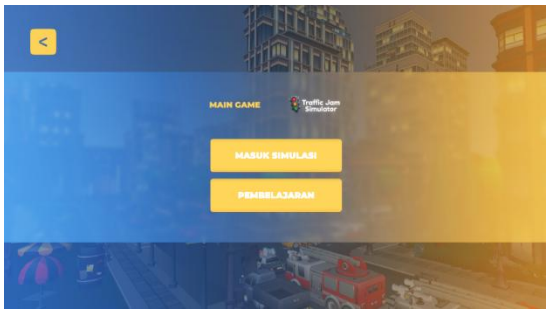
4.1 Implementasi Sistem

Sistem yang diterapkan pada *game* terdiri Antarmuka dan sistem *gameplay*. Antarmuka menjelaskan tampilan menu *game* dan fungsinya, sedangkan sistem *gameplay* yaitu menjelaskan sistem yang bekerja dalam permainan. Bagian ini menjelaskan seluruh antarmuka yang ada pada *game* dimulai dengan main menu dan pengaturan seperti gambar 4.1 berikut.



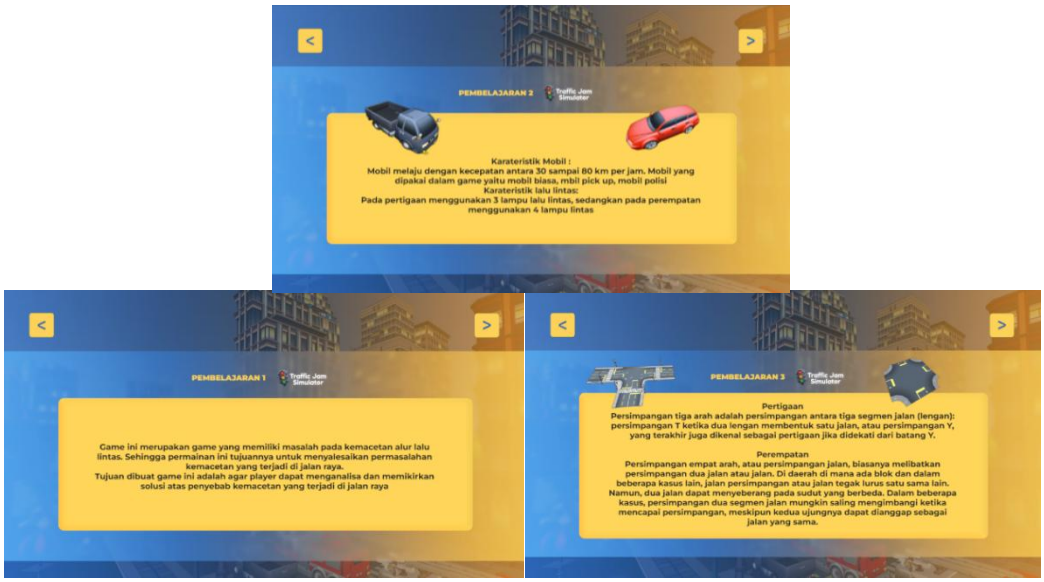
Gambar 4. 1 Antarmuka Main menu dan Pengaturan

Terdapat pilihan serta pengaturan *game* yang bisa disesuaikan *player* seperti pada gambar 4.2. Dilanjutkan ke menu mulai permainan terdapat menu dan pilihan untuk melanjutkan ke pembuatan skenario *game* atau menu pengenalan permainan.



Gambar 4. 2 Antarmuka Main Game

Jika *player* memilih menu pengenalan permainan akan masuk pada menu yang isinya menjelaskan pengertian dasar tentang lalu lintas dan aturan permainan yaitu untuk membuat skenario lalu lintas dan menyelesaikannya dengan mengontrol APILL secara langsung atau menggunakan *model* lampu statis atau *model* lampu AI, seperti pada gambar 4.3. Ketika selesai pada menu ini *player* akan dialihkan ke menu pembuatan skenario lalu lintas.



Gambar 4. 3 Antarmuka Panel Pembelajaran

Jika *player* memilih menu pembuatan skenario, *player* akan disajikan dengan pengaturan jumlah skenario kendaraan yang diinginkan dan pilihan lokasi

persimpangan yang ingin dimainkan seperti pada gambar 4.4, Ketika selesai maka *player* akan melanjutkan ke *scene gameplay*.

Gambar 4. 4 Antarmuka Panel Pembuatan Skenario

Pada *scene gameplay* terdapat beberapa tombol sebagai interaksi *player* untuk mengatur permainan. Pada deretan kiri terdapat kumpulan tombol untuk navigasi dan tombol pengontrolan sinyal lampu. Kumpulan tombol navigasi diantaranya tombol untuk memilih persimpangan selanjutnya dan sebelumnya lalu pemilihan kontrol lampu selanjutanya dan berikutnya. Dilanjutkan kumpulan tombol pengontrolan sinyal diantaranya adalah tombol menyalakan sinyal lampu merah, kuning, hijau seperti pada gambar 4.5.

Gambar 4. 5 Gameplay Traffic Jam Simulation

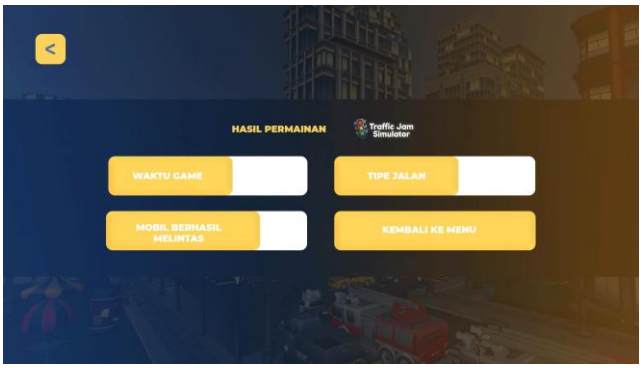
Kemudian pada atas layar terdapat kumpulan antarmuka untuk opsi pengaturan *game* yaitu tombol penggantian mode permainan, tombol *pause*, tombol kembali dan teks waktu yang telah berlalu. Tombol penggantian mode berfungsi untuk mengganti mode pengontrolan lampu ke Auto Mode untuk penggunaan kontrol lampu sinyal statis, AI Mode untuk penggunaan kontrol lampu sinyal AI, dan Play Mode untuk memainkan lampu lalu lintas secara langsung oleh *player*. Ketika AI mode ditekan akan mengalihkan *player* pada menu pause untuk memulai prosedur sistem dalam menyalakan AI pada *environment* disertai dengan panel pop up notifikasi prosedur tersebut.

Selanjutnya terdapat sistem *training* dan evaluasi yang diimplementasikan pada tombol *boolean* Evaluation Mode. Dimana *true* menandakan penggunaan sistem *environment training* dan *false* menandakan penggunaan sistem *environment* evaluasi.

Tombol *pause* akan menghentikan permainan dan memunculkan menu pause *game* yang berisikan tombol untuk melanjutkan dan keluar permainan. Untuk tombol

kembali digunakan sebagai *shortcut player* untuk keluar dari permainan dan kembali ke main menu. Terakhir terdapat teks waktu untuk mengamati berapa lama simulasi yang sedang berlangsung dan tombol *hasten* untuk mempercepat skala waktu permainan menjadi dua kali lipat yang akan berguna untuk simulasi *training* AI.

Ketika simulasi berakhir selanjutnya sistem akan menampilkan panel hasil akhir simulasi tersebut yang menjelaskan tentang waktu yang berjalan, persimpangan yang dilatih dan mobil yang berhasil melintasi persimpangan seperti gambar 4.6 berikut.



Gambar 4. 6 Panel Hasil Akhir Simulasi

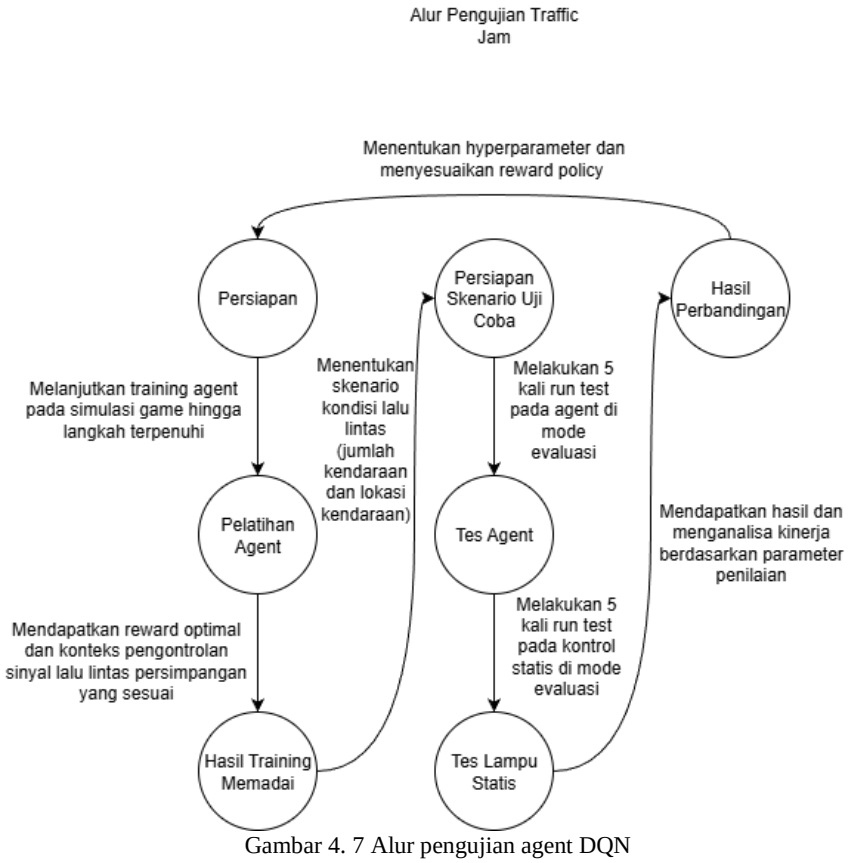
4.2 Skenario Uji Coba

Untuk melakukan uji coba dibuat dua skenario simulasi untuk empat *agent* pada masing-masing persimpangan. Dua skenario simulasi tersebut, yaitu *environment* skenario simulasi untuk *training agent* dan *environment* untuk evaluasi kinerja kedua sistem yang diuji.

Hal ini dilakukan karena sifat *environment* utama pada *game* ini dibuat dengan mekanisme kendaraan yang terus-menerus berjalan pada jalan raya serta rute yang teracak pada masing-masing kendaraan. Ini menyebabkan terjadinya kontinuitas simulasi lalu lintas menyerupai kenyataan dimana *environment* berkembang secara

dinamis dengan situasi yang tak menentu. Dari desain *environment* seperti ini akan menghasilkan *environment training* dinamis untuk melatih konvergensi pembelajaran *agent* yang ter-generalisir untuk situasi yang luas dan beragam sehingga menghasilkan keputusan yang bersifat adaptif.

Pengujian pada penelitian ini dilakukan dengan skenario pelatihan *agent* terlebih dahulu pada *environment training* agar *agent* dapat mempelajari pengontrolan sinyal lalu lintas pada simulasi yang dilakukan sebagai persiapan. Setelah pelatihan *agent* selesai akan dilanjutkan dengan skenario evaluasi untuk menguji dan menganalisa hasil dari kedua kontrol lampu sinyal persimpangan. Alur pengujian *agent* diuraikan pada gambar 4.7 berikut.



Gambar 4. 7 Alur pengujian agent DQN

4.2.1 Persiapan Simulasi Training

Skenario simulasi *training* merupakan simulasi untuk melatih *model agent* dimana perkembangan pada *environment* lebih dilepas agar memberikan variasi situasi untuk dipecahkan *agent,* mengikuti alur pengujian DQN sebelumnya. Model *agent* ini ditentukan dengan persiapan *default hyperparameter* yang ditetapkan pada tabel 4.1 berikut:

Tabel 4. 1 Penetapan Hyperparameter Agent

Hyperparameter	**Value**
learning_rate	0.0005
learning_rate_schedule	Constant
batch_size	128
buffer_size	100000
Tau	0.005
steps_per_*update*	10.0
exploration_initial_eps	0.8
exploration_final_eps	0.05
Normalize	True
hidden_units	156
num_layers	2
keep_checkpoints	5
max_steps	100000
summary_freq	100

Persipaan selanjutnya adalah penyesuaian *learning rate agent* berdasarkan jenis persimpangan. Dimana banyaknya lampu berbanding lurus dengan banyak dimensi *state* yang dianalisa *agent* untuk belajar, maka dari itu perlu pembobotan *learning rate* yang lebih kecil agar *training agent* sesuai dengan kondisi *environment* sehingga

menghasilkan konvergensi yang efektif. Tabel 4.2 berikut merupakan penentuan *learning rate* untuk tiap *agent* persimpangan:

Tabel 4. 2 Pengaturan Learning Rate Berdasarkan Persimpangan

Agent	**Jenis Persimpangan**	**Learning Rate**
agent 1	Persimpangan tiga, dua lampu	0.0005
agent 2	Persimpangan tiga, tiga lampu	0.0003
agent 3	Persimpangan empat, empat lampu	0.00015
agent 4	Persimpangan tiga, tiga lampu	0.0003

4.2.2 Persiapan Simulasi Evaluasi

Pada skenario evaluasi, persiapan skenario pengukuran kinerja antara *agent* dan lampu statis ditentukan dengan durasi sinyal 10 detik, jumlah kendaraan pada tabel 4.3 dan panjang jalur pada tabel 4.4 berikut ini:

Tabel 4. 3 Jumlah Kendaraan Tiap Jalur

Persimpangan	**Jalur 1**	**Jalur 2**	**Jalur 3**	**Jalur 4**
Tiga, dua lampu	26	39	-	-
Tiga, tiga lampu	16	12	35	-
Empat, empat lampu	22	10	14	8
Tiga, empat lampu	10	10	12	-

Tabel 4. 4 Panjang Jalur Tiap Persimpangan

Persimpangan	**Jalur 1**	**Jalur 2**	**Jalur 3**	**Jalur 4**
Tiga, dua lampu	120m	280m	-	-
Tiga, tiga lampu	120m	194m	122m	-
Empat, empat lampu	148m	80m	253m	231m
Tiga, empat lampu	253m	231m	80m	-

4.3 Hasil Uji Coba

Pada bagian ini akan membahas analisa kinerja dan hasil uji coba *agent* melalui perbandingan dengan skenario percobaan yang telah ditentukan sebelumnya.

4.3.1 Hasil Skenario Simulasi Training

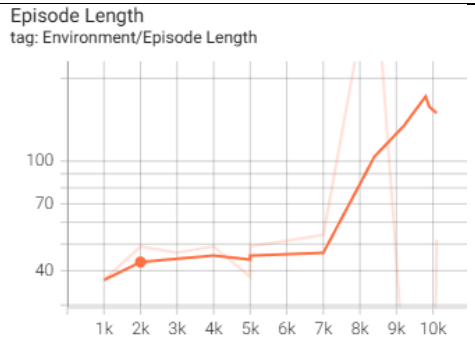
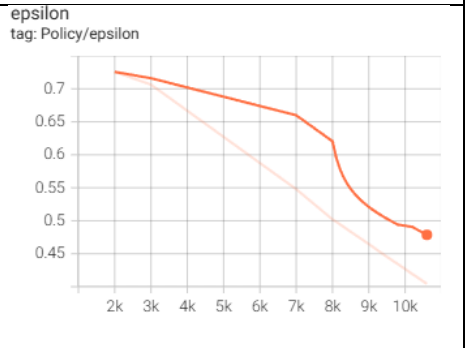
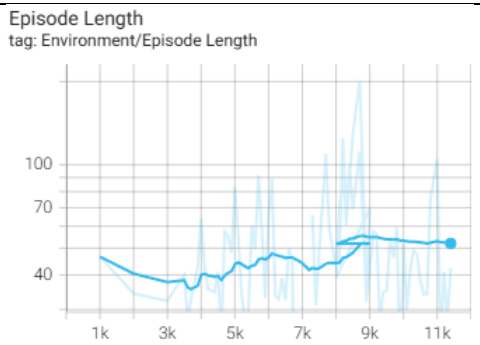
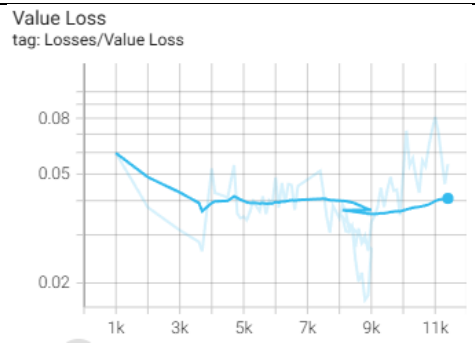
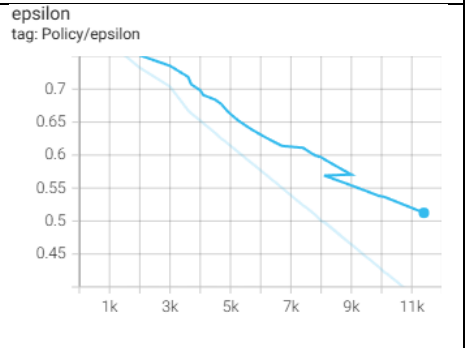
Tabel 4.5 menunjukkan hasil simulasi selama *training* hanya dengan penggunaan *extrinsic reward* yang ditentukan pada *script* Unity. Nilai ukur *training agent* direpresentasikan dengan perbandingan nilai terhadap langkah yang ditempuh. Parameter nilai tersebut yaitu *reward per episode length* dimana menunjukkan perolehan *reward agent* pada tiap episode nya kemudian *loss* merepresentasikan merepresentasikan kesalahan prediksi *agent* terhadap aksi yang benar, terakhir adalah *epsilon* merepresentasikan kecenderungan *agent* melakukan eksploitasi pengetahuannya menggunakan DQN. Pada awalnya, *reward per* episode tetap rendah untuk 2000 langkah pertama, mencerminkan fase eksplorasi *agent*, yang dapat dilihat pada kolom *epsilon* dengan nilai masih tinggi sebanyak 0.72 untuk *agent* 1, 0.75 untuk *agent* 2, 0.76 untuk *agent* 3, 0.77 untuk *agent* 4. Kebuntuan ini sudah diduga, dikarenakan kebingungan *agent* dalam menyusun strategi terhadap perubahan kondisi lalu lintas simulasi *environment* terus menerus dan feedback yang beragam pada permulaan simulasi *training*.

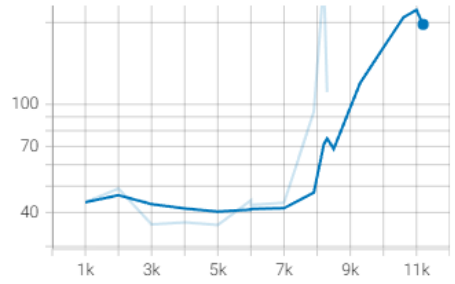
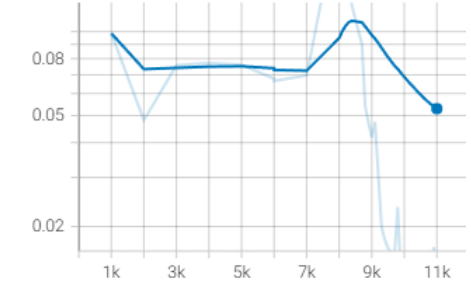
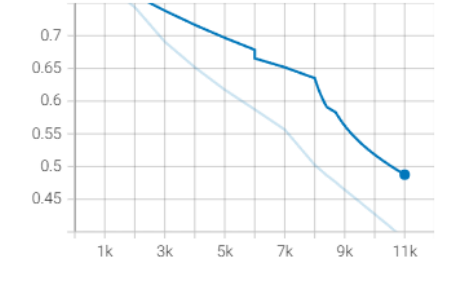
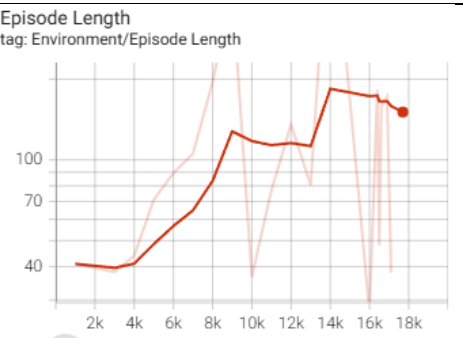
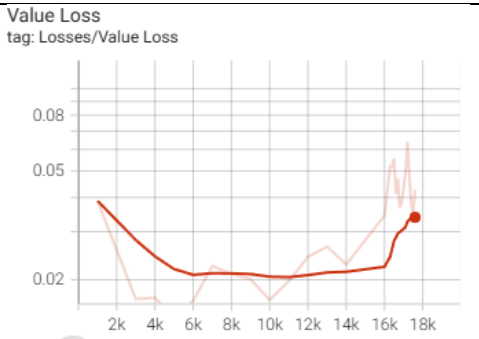
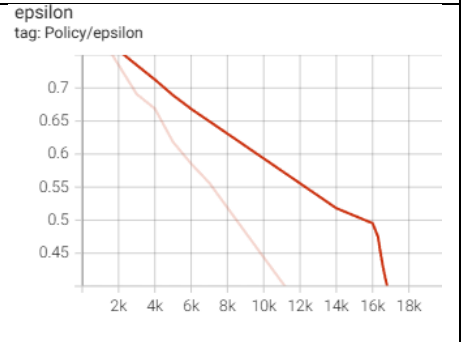
Setelah fase ini, beberapa *reward agent* bertahap mulai naik menunjukkan *agent* menemukan strategi untuk meningkatkan kinerja hingga pada puncaknya *agent* 1 mendapatkan *reward* 172 pada langkah ke 9800, *agent* 2 mendapatkan 55 pada langkah ke 9800, *agent* 3 mendapatkan *reward* 222.6 pada langkah ke 11000 sedangkan *agent*

4 mencapai puncaknya hingga 184.1 *reward* pada langkah ke 14000. Meskipun demikian kurva *reward* tetap tidak stabil disebabkan penyesuaian terhadap perkembangan *environment simulasi training* dan keragaman skenario lalu lintas yang dihadapi.

Setelah 10000 langkah, semua parameter *loss agent* tetap rendah dengan rata-rata (~0.05), walaupun nilai *loss* masih berfluktuasi saat *training*. Ini menunjukkan bahwa *agent* tetap beradaptasi dan menyesuaikan strategi untuk mengurai skenario lalu lintas.

Tabel 4. 5 Hasil Training Tiap agent

agent	***Reward* Episode**	**Loss**	**Epsilon**
***agent* 1**			
***agent* 2**			

***agent* 3**			
***agent* 4**			

Meskipun statistik yang disajikan pada tabel 4.5 berguna untuk memeriksa berjalannya *agent*, data statistik tersebut kurang dalam menentukan hasil kesuksesan pembelajaran *agent* yang disebabkan oleh struktur dinamis simulasi. Untuk itu, kami lebih bersandar pada statistik evaluasi berdasarkan skenario dan tindakan *agent* yang dipresentasikan pada sub bab 4.3.2.

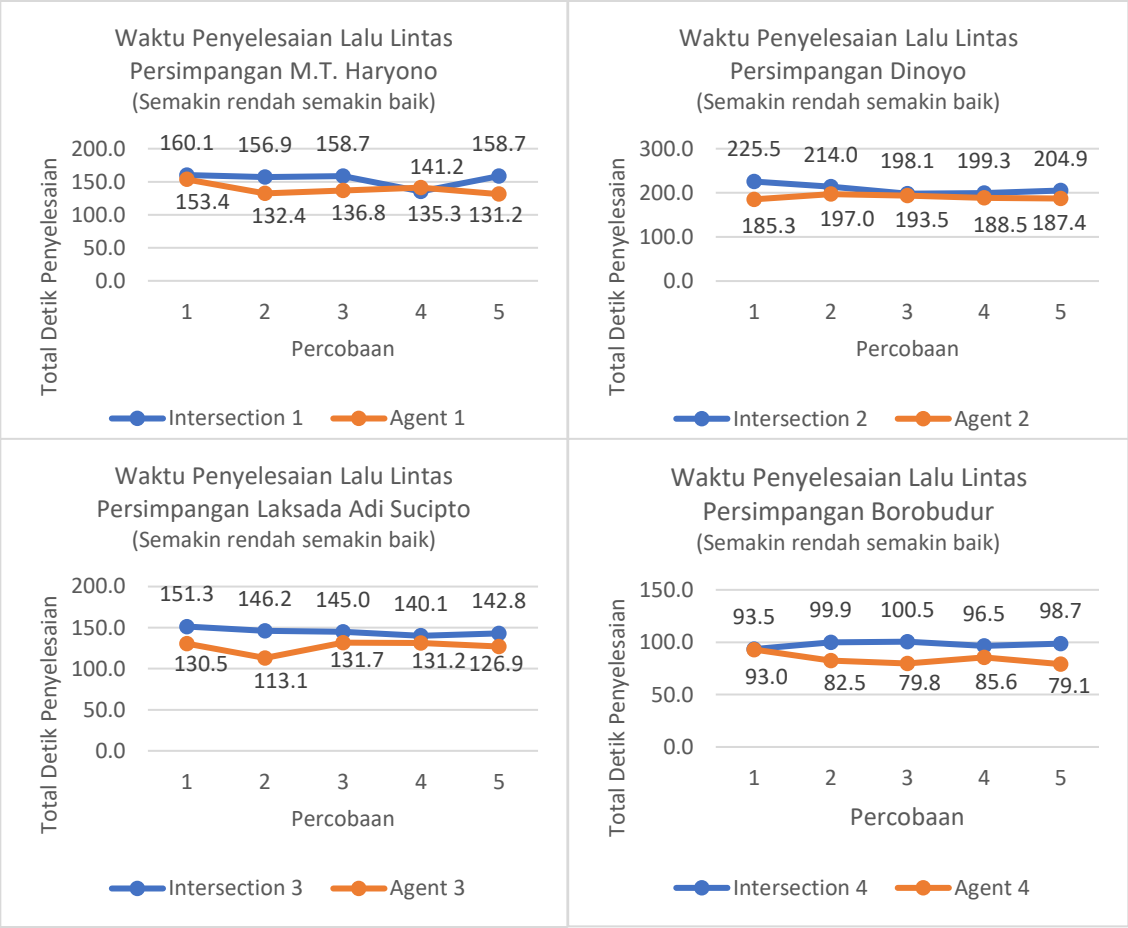
4.3.2 Hasil Skenario Simulasi Evaluasi

Hasil skenario simulasi evaluasi ini ditentukan berdasarkan perbandingan antara *agent* dan lampu statis pada persimpangan yang sama. Skenario tersebut terbagi berdasarkan lokasi menjadi 4 yaitu skenario M.T. Haryono meliputi *Intersection 1* dan *agent 1*, skenario Dinoyo meliputi *Intersection 2* dan *agent 2*, skenario Laksada Adi Sucipto meliputi *Intersection 3* dan *agent 3*, skenario Borobudur meliputi *Intersection 4* dan *agent 4*. Berikut merupakan hasil perbandingan tiap skenario berdasarkan parameternya.

1. Waktu Penyelesaian Lalu Lintas Tiap Persimpangan

Parameter waktu penyelesaian lalu lintas merepresentasikan seberapa cepat *agent* dan lampu statis dengan skenario percobaan yang diberikan. Parameter ini merupakan bagian terpenting dari statistik pengukuran kinerja yang menentukan kesuksesan pembelajaran *agent* dan seberapa efektif *agent* dalam mengurai kemacetan. Dapat dilihat pada gambar 4.8 menunjukkan hasil yang beragam dengan rata-rata waktu penyelesaian skenario lalu lintas didominasi *agent* pada setiap percobaan dan seluruh skenario. Namun masih terdapat fluktuasi yang disebabkan oleh nilai *epsilon* atas hasil *training* yang belum mencapai langkah maksimal yaitu 100000 langkah

sehingga terjadi kemungkinan *agent* untuk melakukan eksplorasi daripada eksploitasi meskipun rendah.

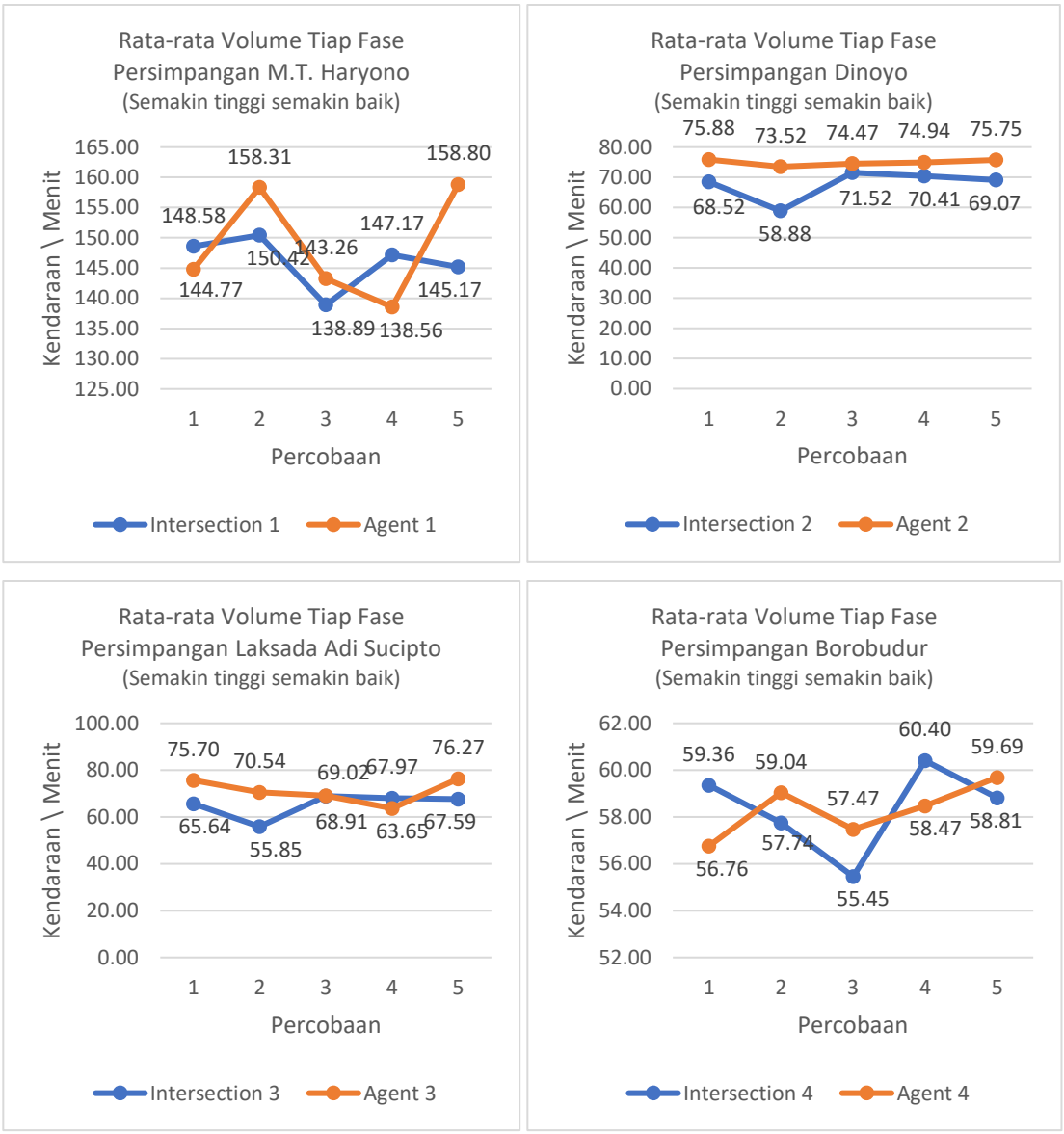


Gambar 4. 8 Perbandingan Waktu Penyelesaian Persimpangan

2. Volume Lalu Lintas Tiap Persimpangan

Parameter ini menunjukkan seberapa lancar arus persimpangan untuk dilalui kendaraan. Dimana hasil percobaan yang didapat beragam pada tiap skenario, yaitu skenario M.T. Haryono sebanyak 3 kali percobaan menunjukkan lebih baik agent, skenario Dinoyo agent lebih unggul keseluruhan percobaan, skenario Laksada Adi

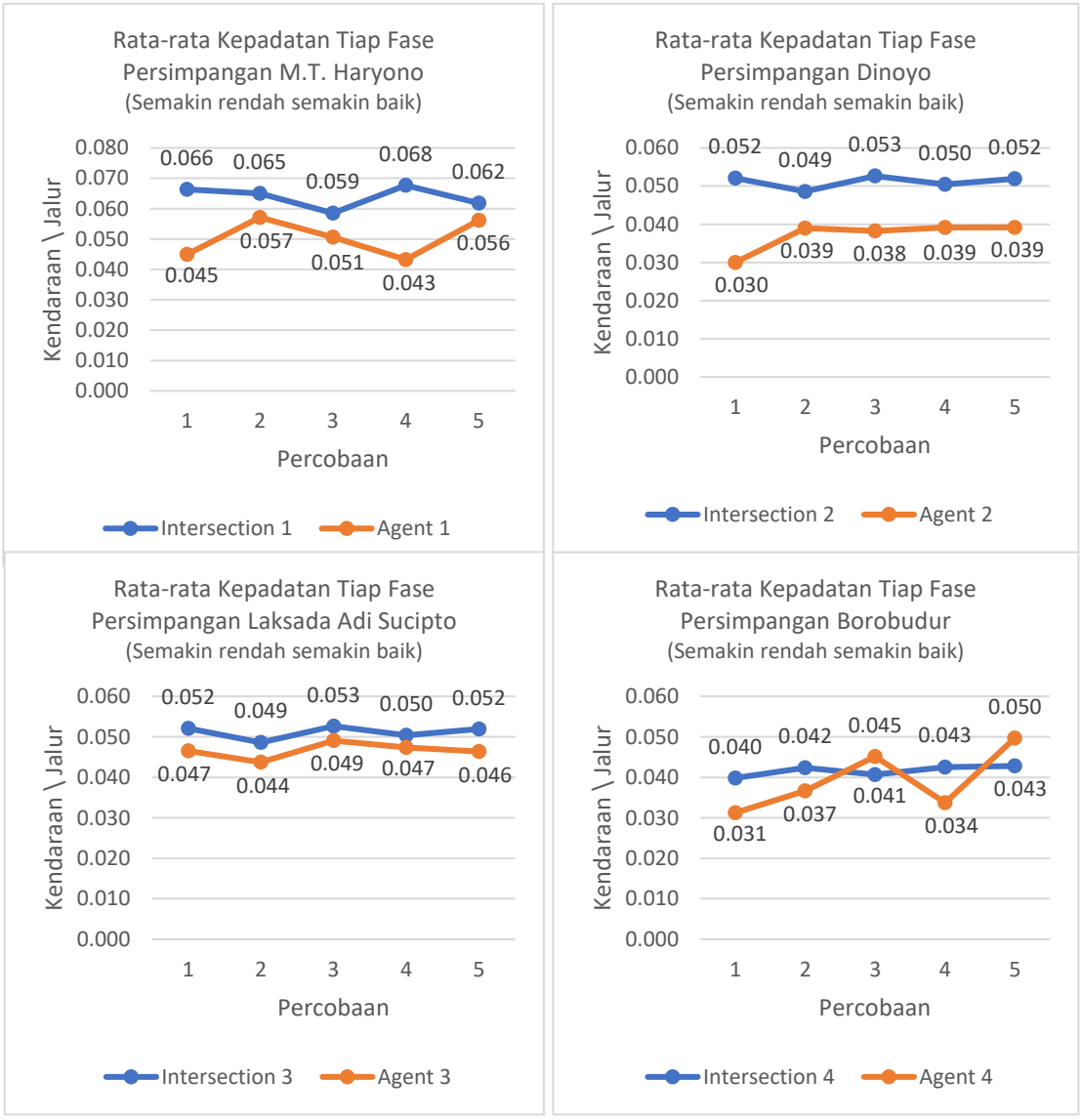
Sucipto agent lebih unggul 4 percobaan, skenario Borobudur agent lebih rendah dengan 2 percobaan lebih unggul.



Gambar 4. 9 Perbandingan Rata Volume Persimpangan

3. Kepadatan Lalu Lintas Tiap Persimpangan

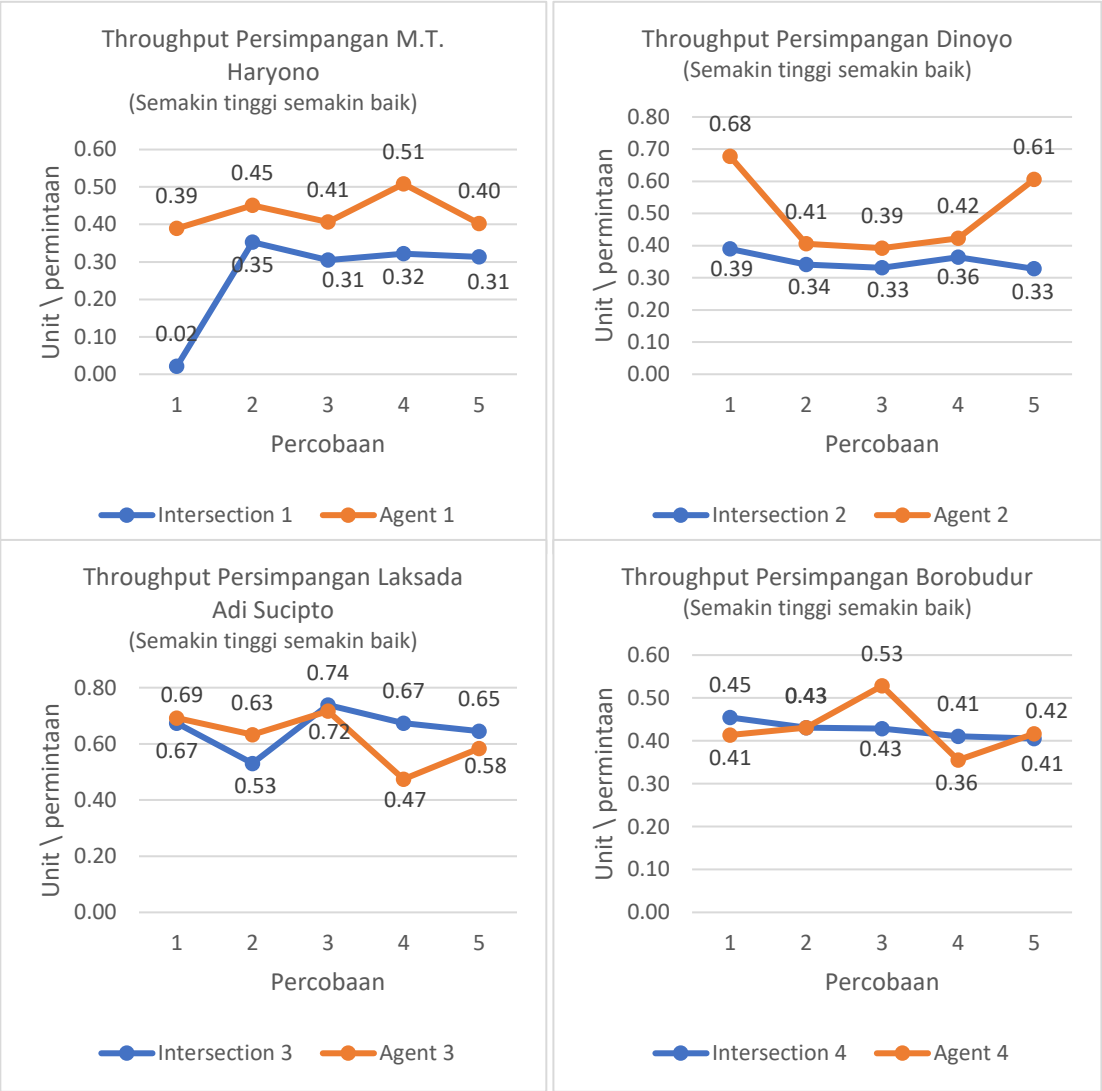
Rata kepadatan persimpangan keseluruhan percobaan pada skenario M.T. Haryono, skenario Dinoyo, dan Laksada Adi Sucipto menunjukkan agent lebih unggul daripada lampu statis sedangkan pada skenario Borobudur agent hanya unggul dengan 3 kali percobaan.



Gambar 4. 10 Perbandingan Rata Kepadatan Persimpangan

4. Throughput Lalu Lintas Tiap Persimpangan

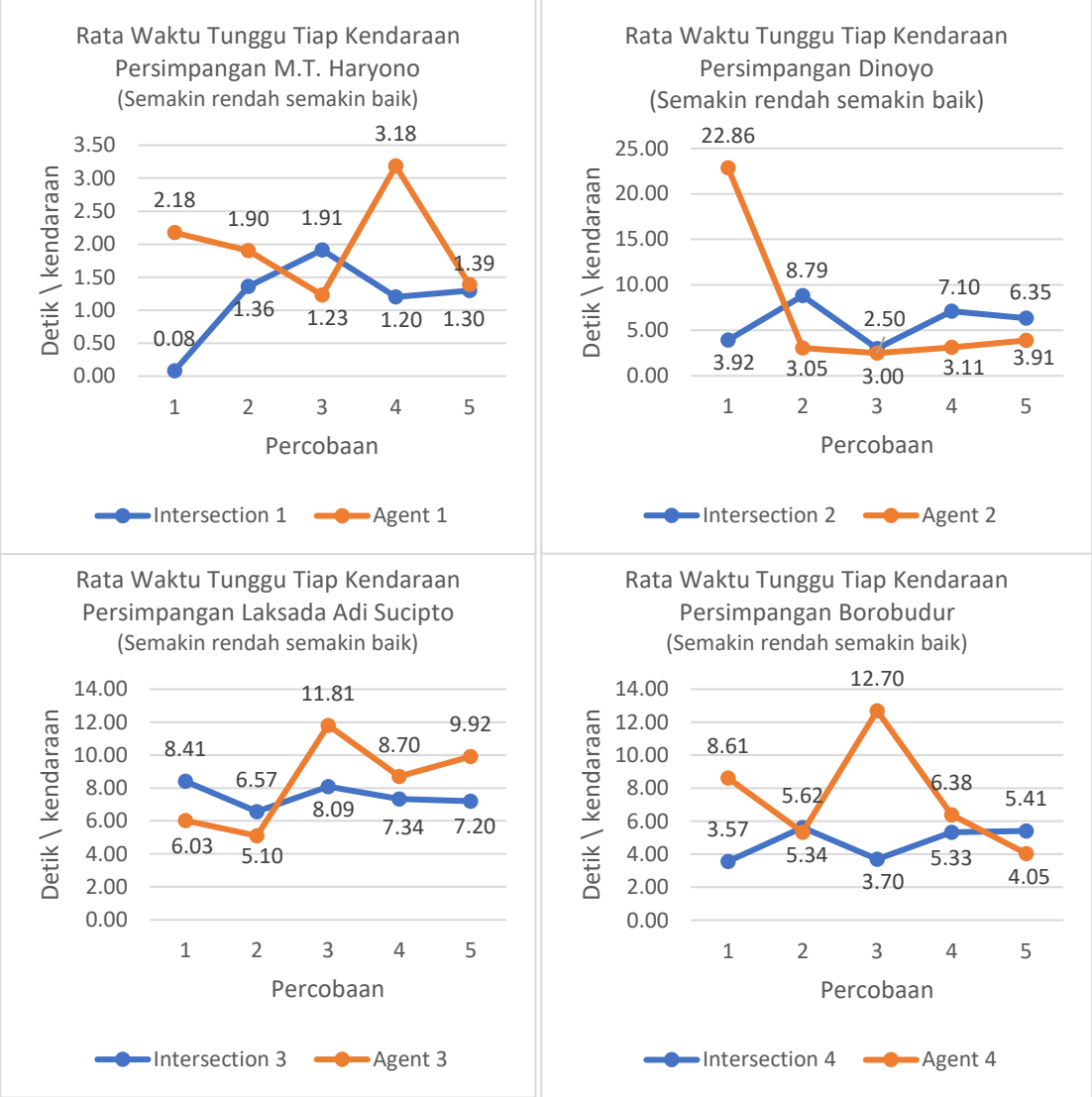
Pada statistik perbandingan *throughput,* skenario M.T Haryono dan skenario Dinoyo *agent* menunjukkan konsistensi strategi yang efektif dalam penguraian kemacetan dengan *throughput* keseluruhan percobaan yang lebih baik. Namun pada skenario Laksada Adi Sucipto dan skenario Borobudur lampu statis masih unggul dengan 3 percobaan yang lebih baik.



Gambar 4. 11 Perbandingan Throughput Persimpangan

5. Rata-rata Waktu Tunggu Kendaraan Tiap Persimpangan

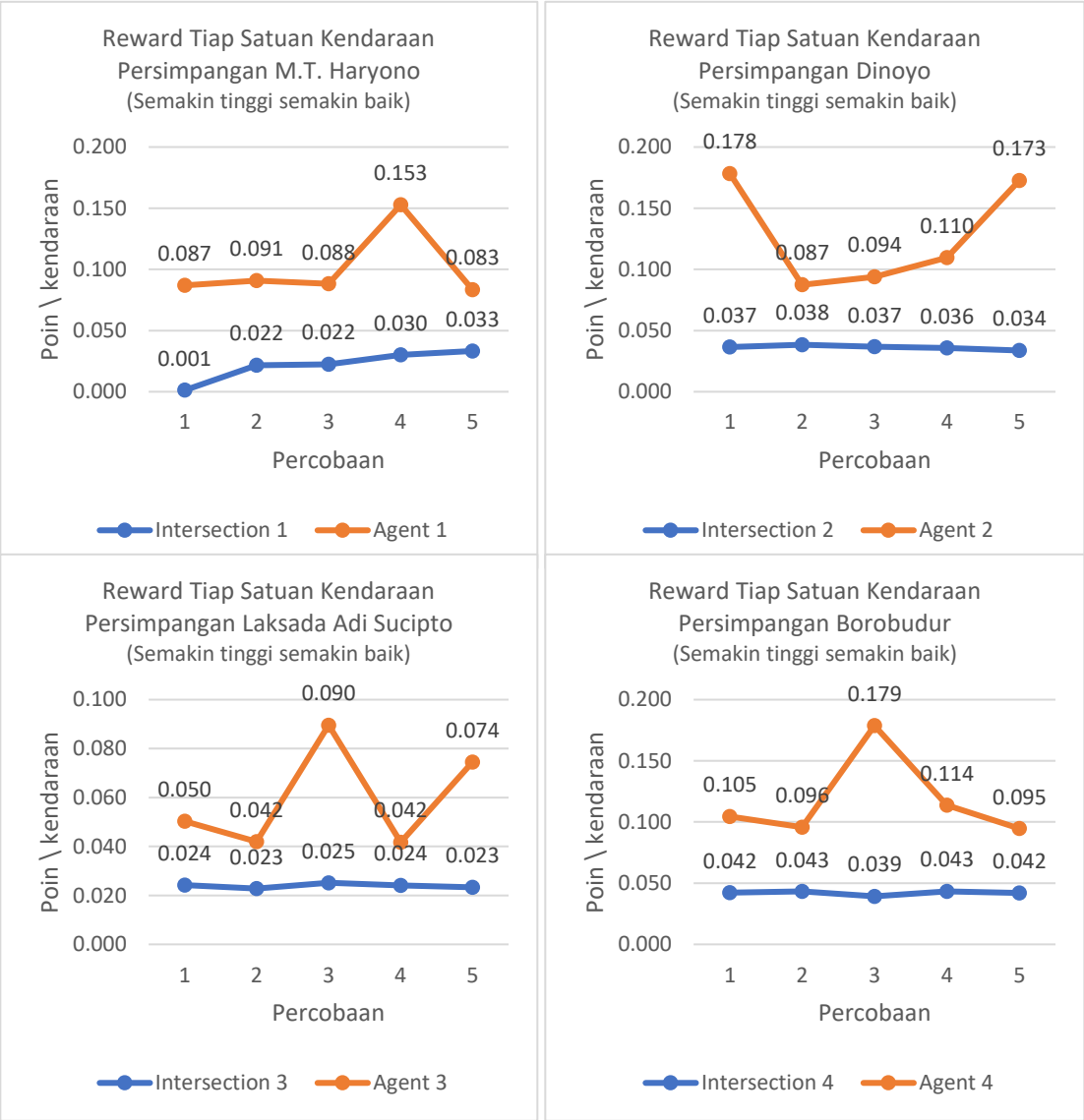
Rata waktu persimpangan pada perbandingan ini rata-rata lampu statis lebih unggul dengan skenario M.T. Haryono 4 percobaan lampu statis lebih unggul, skenario Laksada Adi Sucipto 3 percobaan lebih unggul, dan skenario Borobudur 3 percobaan lebih unggul, sedangkan agent hanya lebih unggul pada skenario Dinoyo dengan 4 percobaan lebih unggul.



Gambar 4. 12 Perbandingan Rata Waktu Tunggu Persimpangan

6. *Reward* Satuan Kendaraan Tiap Persimpangan

Pada perbandingan *reward* satuan kendaraan menunjukkan penguasaan strategi *agent* dalam penguraian kemacetan dimana keseluruhan skenario dan percobaan *agent* lebih unggul sedangkan lampu statis cenderung stagnan.

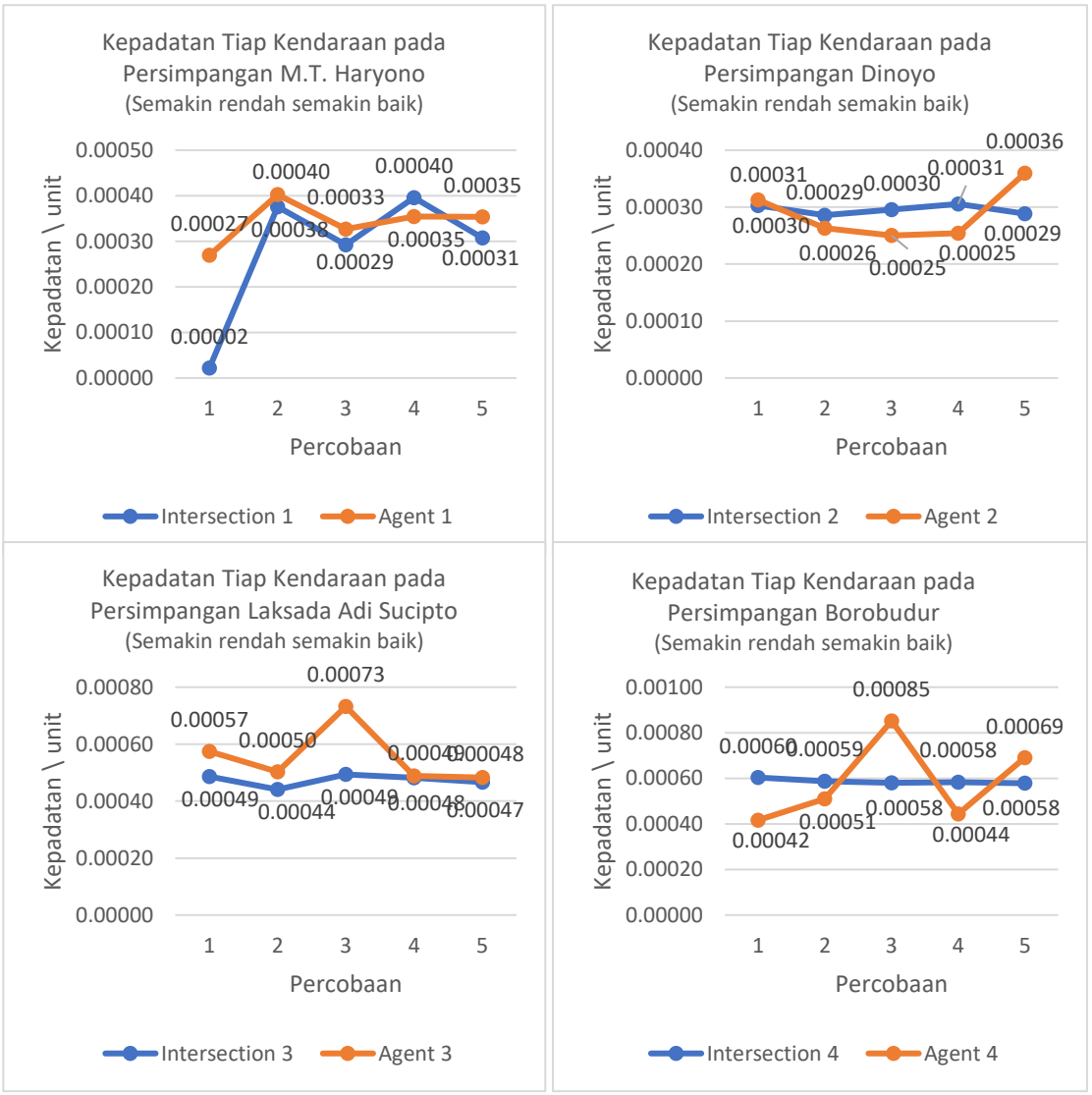


Gambar 4. 13 Perbandingan Reward per Kendaraan Persimpangan

7. Kepadatan Satuan Kendaraan Tiap Persimpangan

Hasil perbandingan pada parameter ini menunjukkan rata-rata keseluruhan kepadatan yang dialami kendaraan. Dimana pada perbandingan ini lampu statis dan *agent* masih seimbang dengan skenario M.T. Haryono dimana lampu statis lebih baik dengan 4 percobaan yang lebih unggul dan skenario Laksada Adi Sucipto keseluruhan

lebih baik, sedangkan skenario Dinoyo *agent* unggul dengan 3 percobaan yang lebih baik, dan skenario Borobudur yaitu 3 percobaan yang lebih baik.

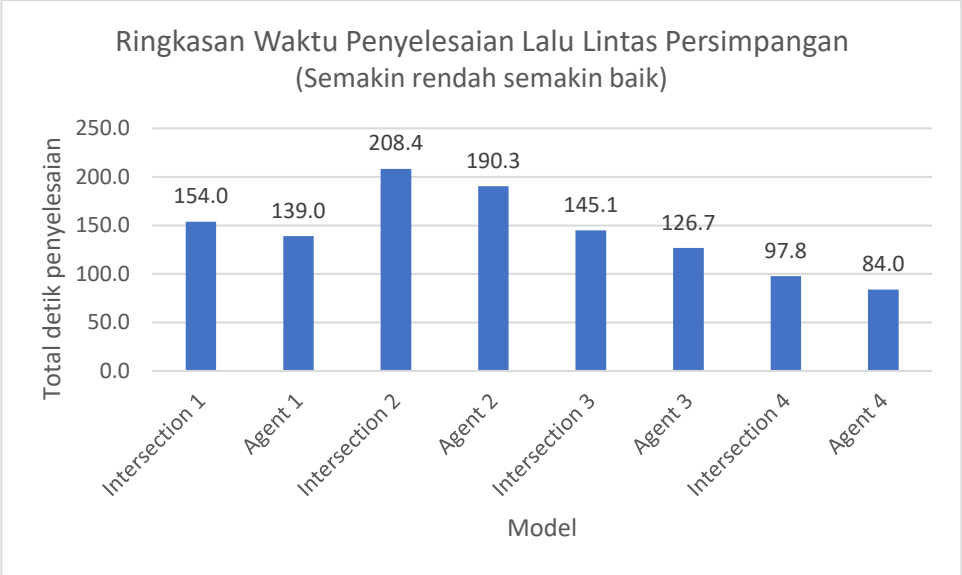


Gambar 4. 14 Perbandingan Kepadatan Tiap Kendaraan Persimpangan

8. Ringkasan Analisa Evaluasi

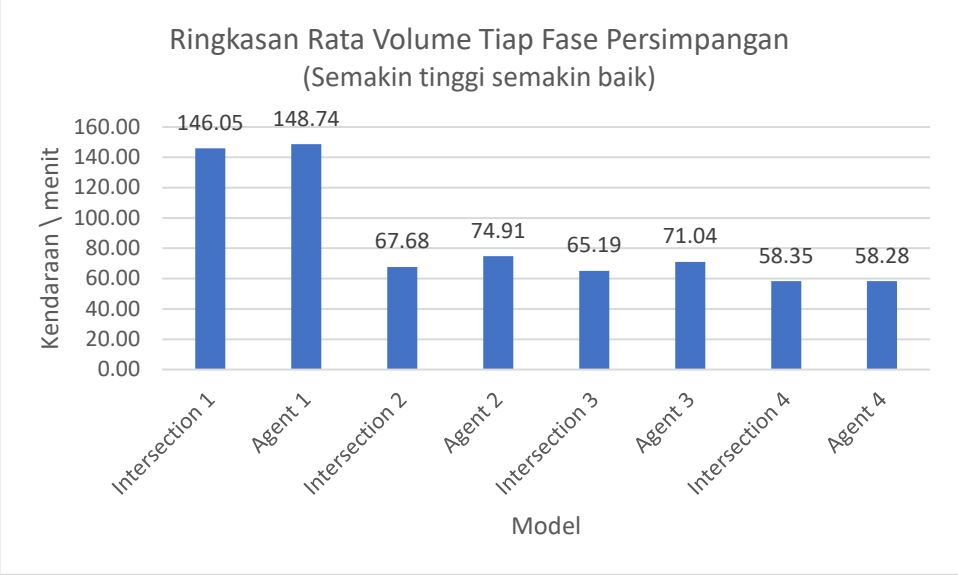
Dapat dilihat pada gambar 4.15 menunjukkan *agent* menyelesaikan semua skenario persimpangan lebih cepat daripada lampu statis dengan rata-rata penyingkatan

waktu keseluruhan sebanyak 16,5 detik lebih cepat. Dimana skenario M.T. Haryono menghasilkan selisih 15 detik, skenario Dinoyo menghasilkan selisih 18 detik, skenario Laksada Adi Sucipto menghasilkan selisih 19 detik, skenario Borobudur menghasilkan selisih 13 detik.



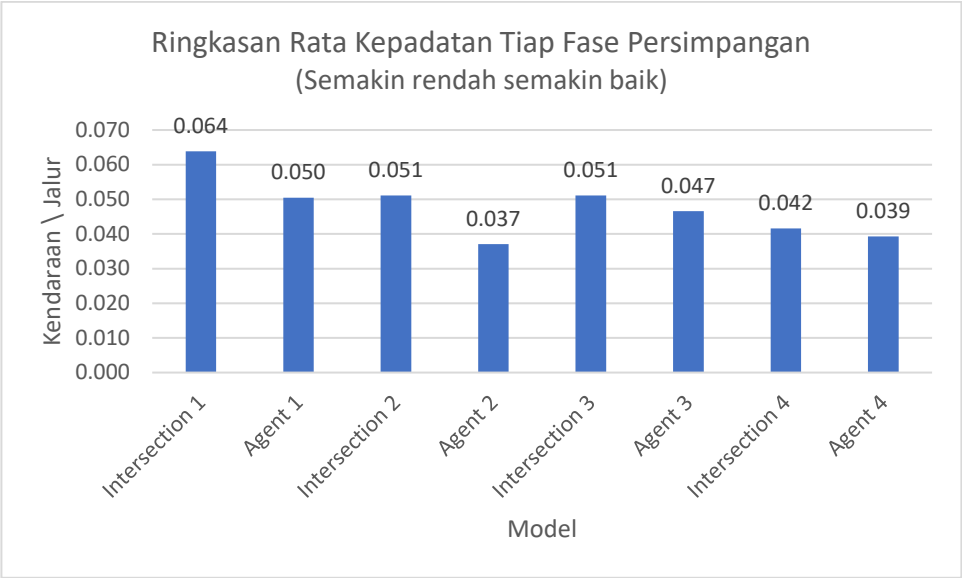
Gambar 4. 15 Ringkasan Perbandingan Waktu Penyelesaian

Untuk gambar 4.16, perbandingan rata volume, *agent* menunjukkan kelancaran volume yang lebih tinggi yaitu dari skenario M.T Haryono *agent* menghasilkan volume sedikit lebih lancar dengan 2 unit/menit lebih tinggi. Pada skenario Dinoyo *agent* menang dengan 7 unit/menit lebih tinggi. Skenario Laksada Adi Sucipto dimenangkan oleh *agent* dengan 6 unit/menit. Sedangkan pada skenario persimpangan Borobudur *model agent* dan statis perbedaan yang sangat kecil yang dimenangkan *model* statis sebanyak 0.07 unit/menit.



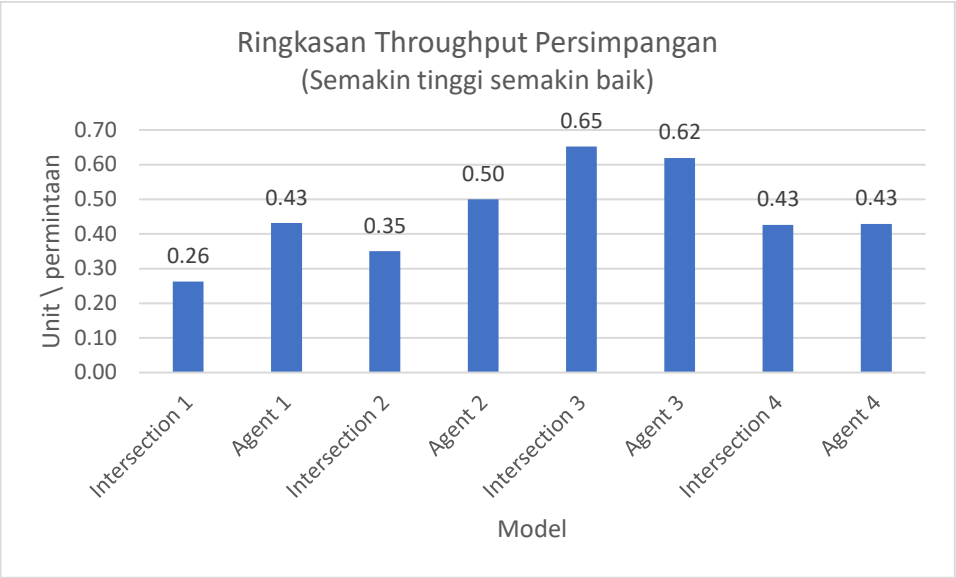
Gambar 4. 16 Ringkasan Perbandingan Rata Volume

Rata kepadatan tiap fase dapat dilihat bahwa *agent* merata lebih unggul pada seluruh skenario persimpangan dimana skenario M.T. Haryono *agent* unggul dengan selisih 0.014, skenario Dinoyo selisih 0.014, Laksada Adi Sucipto selisih 0.004, skenario Borobudur 0.003 lebih sedikit.



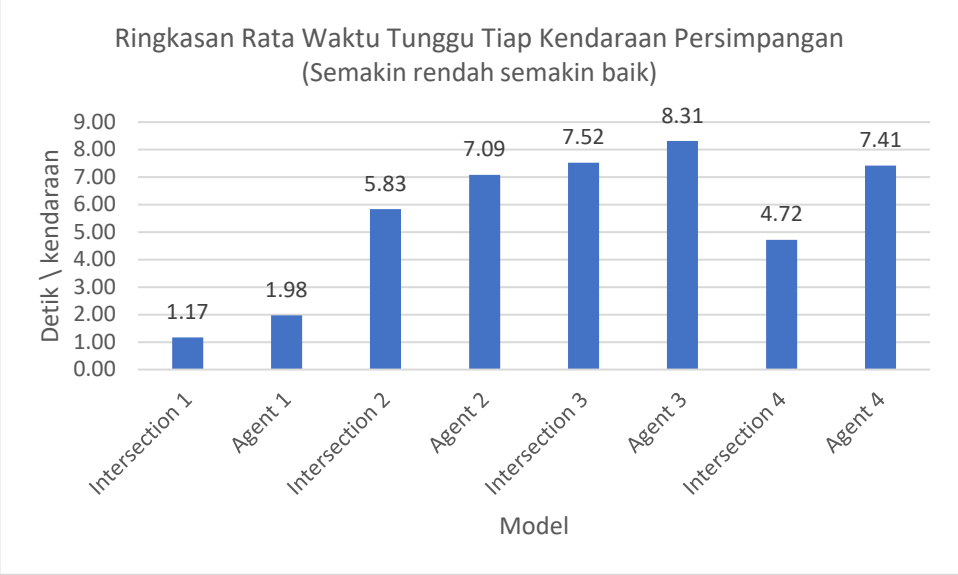
Gambar 4. 17 Ringkasan Perbandingan Rata Kepadatan

Pada gambar 4.18, *throughput* tidak lagi didominasi oleh *agent* dikarenakan perbedaan waktu penyelesaian yang mempengaruhi total permintaan kendaraan yang disimulasikan dari skenario. Pada persimpangan M.T Haryono *agent* lebih unggul dengan selisih 0.17 dan persimpangan Dinoyo sebanyak 0.15, sedangkan pada persimpangan Laksada Adi Sucipto dimenangkan lampu statis sebanyak 0.03 dan persimpangan Borobudur menunjukkan hasil yang seimbang.



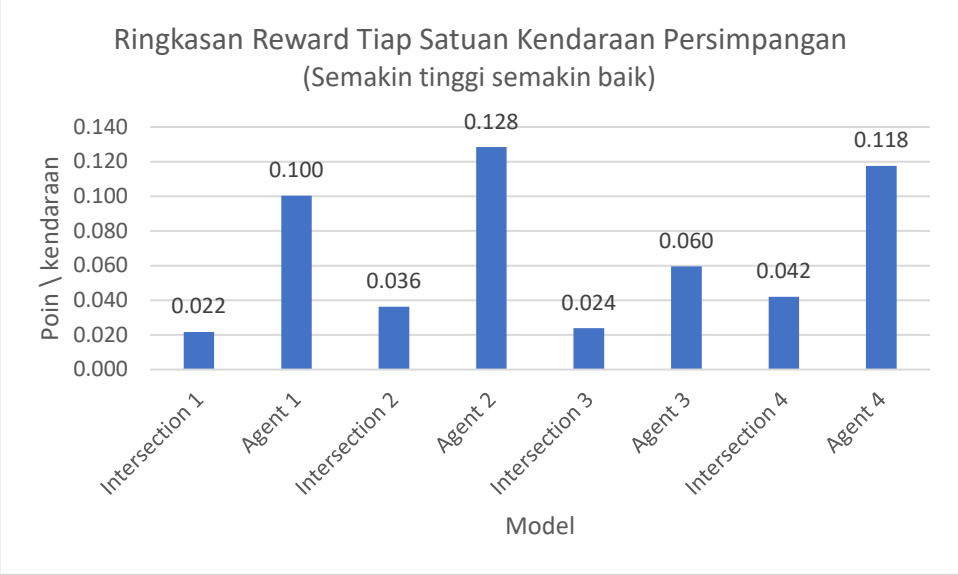
Gambar 4. 18 Ringkasan Perbandingan Throughput

Pada gambar 4.19, rata waktu tunggu dimenangkan oleh lampu statis secara keseluruhan. Dimana skenario M.T. Haryono lampu statis selisih 0.81 detik, skenario Dinoyo selisih 1.16 detik, Laksada Adi Sucipto selisih 0.81 detik, skenario Borobudur selisih 2.31 detik lebih cepat.



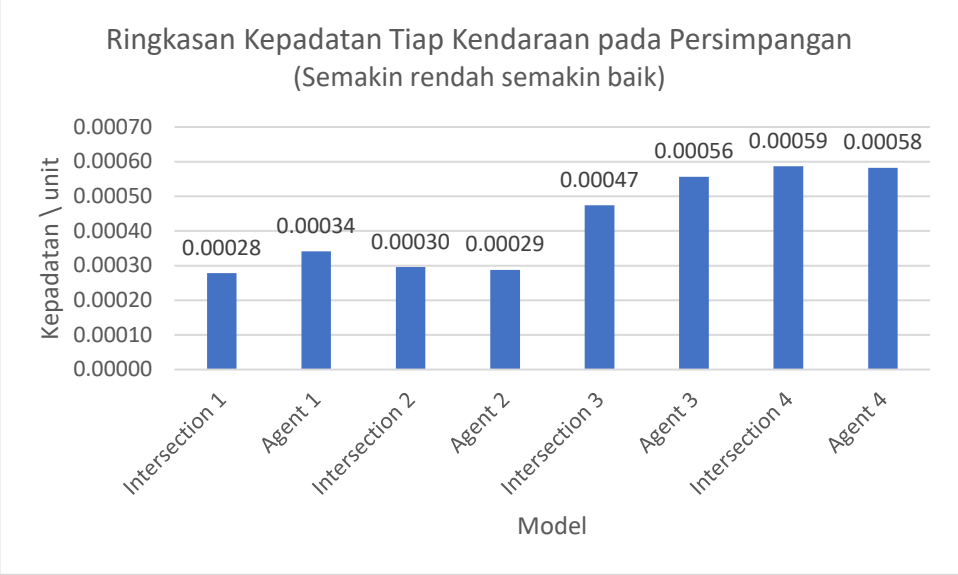
Gambar 4. 19 Ringkasan Perbandingan Waktu Tunggu per Kendaraan

Pada gambar 4.20 tiap satuan kendaraan *agent* memiliki selisih *reward* tiap kendaraan yang jauh lebih besar kerena pembelajaran *agent* yang dituntun oleh *policy*. Dimana skenario M.T. Haryono *agent* unggul dengan selisih 0.078, skenario Dinoyo selisih 0.092, Laksada Adi Sucipto selisih 0.036, skenario Borobudur 0.076.



Gambar 4. 20 Ringkasan Reward Tiap Kendaraan

Pada gambar 4.21 kepadatan tiap kendaraan mayoritas dimenangkan oleh lampu statis dengan skenario M.T. Haryono selisih 0.00006 lebih baik lampu statis, skenario Dinoyo selisih 0.00001 lebih baik *agent*, skenario Laksada Adi Sucipto selisih 0.00009 lebih baik lmapu statis, skenario Borobudur selisih 0.00001 lebih baik *agent.*



Gambar 4. 21 Ringkasan Kepadatan Tiap Kendaraan

4.4 Pembahasan Hasil

Berdasarkan tujuan dari penelitian ini yaitu penguraian kemacetan dan perbandingan kontrol *agent* terhadap lampu statis, maka kami lebih menitikberatkan kesimpulan penelitian pada pengurangan kepadatan dan penyelesaian lalu lintas. Untuk itu parameter yang kami utamakan adalah penyelesaian lalu lintas, volume lalu lintas, kepadatan persimpangan tiap fase, *throughput*, dan kepadatan tiap kendaraan.

Melihat hasil penyelesaian lalu lintas pada skenario evaluasi yang telah dilakukan terhadap semua skenario persimpangan dan semua aspek penilaian, *agent* 2 dengan skenario Dinoyo merupakan *model agent* terbaik yang mampu mengungguli keseluruhan parameter penilaian evaluasi yaitu penyelesaian lalu lintas dengan 18 detik lebih cepat, volume lalu lintas 7 unit/menit lebih lancar, kepadatan 0.015 lebih rendah, *throughput* 0.15 lebih banyak, dan kepadatan tiap kendaraan 0.00001 lebih ringkas.

Kedua disusul oleh *agent* 1 yaitu skenario M.T. Haryono dengan penyelesaian lalu lintas dengan 15 detik lebih cepat, volume lalu lintas 2 unit/menit lebih lancar, kepadatan 0.014 lebih rendah, *throughput* 0.17 lebih banyak, dan kepadatan tiap kendaraan 0.00006 lebih ringkas.

Ketiga yaitu *agent* 4 skenario Borobudur dengan penyelesaian lalu lintas dengan 13 detik lebih cepat, namun volume lalu lintas 0.07 unit/menit lebih lambat, kepadatan 0.003 lebih rendah, *throughput* 0.15 lebih banyak, dan kepadatan tiap kendaraan 0.00001 lebih ringkas.

Terakhir, keempat yaitu *agent* 3 Laksada Adi Sucipto dengan penyelesaian lalu lintas dengan 19 detik lebih cepat, volume lalu lintas 6 unit/menit lebih lancar, kepadatan 0.004 lebih rendah, namun *throughput* 0.03 lebih rendah, dan kepadatan tiap kendaraan 0.00009 lebih padat.

Berdasarkan skenario uji coba yang telah dilakukan dengan mendistribusikan bobot *reward policy* menjadi 0.5 bobot *throughput*, 0.4 bobot kepadatan dan 0.1 bobot untuk waktu tunggu membuat *agent* lebih mementingkan penguraian kepadatan persimpangan daripada waktu tunggu kendaraan.

Akibat pembobotan ini dapat dilihat *policy reward* yang telah ditentukan pada tabel 3.3 ini menyebabkan rata waktu tunggu berfluktuasi pada setiap percobaan guna menyesuaikan penguraian lalu lintas yang dapat dilihat pada gambar 4.12. Namun pada gambar 4.8 dapat dilihat bahwa *agent* lebih unggul pada rata-rata semua percobaan dengan waktu penyelesaian lalu lintas tercepat yaitu *agent* 3 skenario Laksada Adi Sucipto merupakan *model* penguraian terbaik dengan selisih rata-rata waktu 19 detik lebih cepat.

Dalam skenario simulasi dapat disimpulkan bahwa pembobotan *reward* inilah yang menentukan kecenderungan prilaku *agent* dalam mengatasi kemacetan persimpangan lalu lintas dimana tiap besaran parameternya mempengaruhi nilai *reward* dan prioritas aksi *agent*.

Adapun pendeknya jarak jalan persimpangan antar satu dengan lain dibanding banyaknya kendaraan mempengaruhi hasil skenario percobaan. Dimana kesempitan jarak persimpangan ini menyebabkan kendaraan yang telah menyebrang dapat kembali ke skenario percobaaan persimpangan sehingga terjadi perkembangan *environment* simulasi lebih besar dari skenario awal yang ditentukan. Ini dapat dibuktikan dengan meningkatnya skala jumlah *throughput* pada gambar 4.11 jika dibandingkan dengan skenario jumlah kendaraan yang diuji. Hal ini juga yang menyebabkan statistik hasil *training dqn network* tetap tidak stabil meskipun telah melakukan puluhan ribu langkah *training.*

Meskipun begitu dengan skenario uji coba dan statistik evaluasi yang kami sajikan cukup untuk mendemonstrasikan kinerja *agent* dalam menangani situasi lalu lintas secara keseluruhan dimana *agent* lebih efektif mengurai kemacetan dengan memprioritaskan strategi untuk penguraian kemacetan berdasarkan kepadatan kemacetan karena memiliki poin *reward* yang lebih besar daripada poin waktu tunggu.

4.5 Integrasi Penelitian dalam Islam

Model *agent* rambu lalu lintas yang telah dikembangkan menggunakan metode *Deep Q-Network* dapat membantu penanganan kemacetan berdasarkan kepadatan dan waktu tunggu kendaraan pada persimpangan melalui lampu rambu lalu lintas yang

dikendalikan oleh *agent.* Dimana sistem ini dapat dibuktikan pada percobaan yang telah dilakukan unggul dalam mengurai lalu lintas persimpangan. Dengan hasil *training* yang lebih banyak dan penyesuaian desain *policy* yang efektif, *agent* mampu mengoptimalkan waktu penguraian kemacetan untuk memudahkan pengendara pada saat terjadi kemacetan lalu lintas. Manfaat ini sesuai dengan makna Al-Qur'an Surat Al-Asr ayat 1 sampai 3 untuk memanfaatkan waktu sebaik-baiknya:

وَالْعَصْرِ. إِنَّ الْإِنْسَانَ لَفِي خُسْرٍ. إِلَّا الَّذِينَ آَمَنُوا وَعَمِلُوا الصَّالِحَاتِ وَتَوَاصَوْا بِالْحَقِّ وَتَوَاصَوْا بِالصَّبْرِ

Demi masa. Sesungguhnya manusia itu benar-benar dalam kerugian, kecuali orang-orang yang beriman dan mengerjakan amal saleh dan nasehat menasehati supaya mentaati kebenaran dan nasehat menasehati supaya menetapi kesabaran. (QS. Al-Asr:1-3)

Dalam buku yang disusun oleh (Ghoffar, 2004) terjemahan Tafsir Ibnu Katsir menyatakan bahwa ayat Al-Asr mengandung makna bahwa waktu manusia sangat terbatas dan berharga, sehingga manusia harus memanfaatkannya dengan sebaik-baiknya agar berhasil. Dalam konteks penanggulangan kemacetan, hal ini dapat diartikan bahwa setiap orang yang berada dalam situasi yang merugikan seperti halnya kemacetan harus memanfaatkan waktunya dengan sebaik-baiknya untuk mencari solusi yang efektif dan efisien untuk mengatasi keadaan tersebut. Manfaat ini juga didukung dengan hadits berikut;

نِعْمَتَانِ مَغْبُونٌ فِيهِمَا كَثِيرٌ مِنَ النَّاسِ، الصِّحَّةُ وَالْفَرَاغُ

"Ada dua nikmat yang banyak manusia tidak bisa memanfaatkan dengan baik, yaitu nikmat sehat dan waktu luang." (HR. Bukhari no. 6412).

Menurut Shaikh al-‘Utsaimīn dalam buku (Riyāḍuṣ Ṣāliḥīn) manusia mengalami kerugian dalam dua bentuk nikmat yang dimilikinya; dengan maksud mereka rawan terlena oleh keduanya: kesehatan jasmani dan waktu luang. Ini dikarenakan ketika seseorang sehat, ia mampu menjalankan apa yang Allah perintahkan dan menjauhi larangan-Nya, karena ia sehat, bahagia, tenang. Demikian pula dengan waktu luang; ketika ia memiliki kecukupan baginya, ia menjadi sepenuhnya bebas.

Dalam konteks ini waktu luang merupakan komoditas yang penting untuk melakukan produktivitas bagi manusia sebagaimana diperintahkan oleh Allah *Subhanahu wa Ta’ala* dimana lampu lalu lintas penguraian kemacetan ini dapat memberikan keleluasaan pengguna jalan untuk memfokuskan hal yang lebih produktif.

Disisi lain, penelitian yang dilakukan memanfaatkan teknologi kecerdasan buatan pada bidang infrastruktur lalu lintas mampu membantu kelancaran penggunaan jalan untuk publik pada umumnya sehingga produktivitas yang terjadi saat berkendara menjadi lebih mudah dan efektif. Hal ini sejalan dengan tema tolong menolong yang disebutkan dalam Al-Qur’an Surat Al-Maidah Ayat 2 dimana dalam firman-Nya:

يَا أَيُّهَا الَّذِينَ آمَنُوا لَا تُحِلُّوا شَعَائِرَ اللَّهِ وَلَا الشَّهْرَ الْحَرَامَ وَلَا الْهَدْيَ وَلَا الْقَلَائِدَ وَلَا آمِّينَ الْبَيْتَ الْحَرَامَ يَبْتَغُونَ فَضْلًا مِنْ رَبِّهِمْ وَرِضْوَانًا ۚ وَإِذَا حَلَلْتُمْ فَاصْطَادُوا ۚ وَلَا يَجْرِمَنَّكُمْ شَنَآنُ قَوْمٍ أَنْ صَدُّوكُمْ عَنِ الْمَسْجِدِ الْحَرَامِ أَنْ تَعْتَدُوا ۘ وَتَعَاوَنُوا عَلَى الْبِرِّ وَالتَّقْوَىٰ ۖ وَلَا تَعَاوَنُوا عَلَى الْإِثْمِ وَالْعُدْوَانِ ۚ وَاتَّقُوا اللَّهَ ۖ إِنَّ اللَّهَ شَدِيدُ الْعِقَابِ

"Hai orang-orang yang beriman, janganlah kamu melanggar syi'ar-syi'ar Allah, dan jangan melanggar kehormatan bulan-bulan haram, jangan (mengganggu) binatang-binatang had-ya, dan binatang-binatang qalaa-id, dan jangan (pula) mengganggu orang-orang yang mengunjungi Baitullah sedang mereka mencari kurnia dan keridhaan dari Tuhannya dan apabila kamu telah menyelesaikan ibadah haji,

maka bolehlah berburu. Dan janganlah sekali-kali kebencian(mu) kepada sesuatu kaum karena mereka menghalang-halangi kamu dari Masjidilharam, mendorongmu berbuat aniaya (kepada mereka). Dan tolong-menolonglah kamu dalam (mengerjakan) kebajikan dan takwa, dan jangan tolong-menolong dalam berbuat dosa dan pelanggaran. Dan bertakwalah kamu kepada Allah, sesungguhnya Allah amat berat siksa-Nya." (QS. Al-Maidah:2)

Ayat 2 pada Surat Al-Maidah memerintahkan manusia untuk saling tolong-menolong dalam melakukan kebaikan, dimana pada konteks penelitian yang telah dilakukan dapat berguna untuk efektifitas infrastruktur kota untuk mengatur wilayah dan produktivitas penduduknya. Tidak hanya itu, melalui *game* simulasi ini dapat digunakan sebagai contoh dan melakukan penelitian lain untuk mencari strategi penanggulangan kemacetan lalu lintas.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil pengujian perbandingan kontrol lampu *agent Deep Q-Network* dan lampu statis pada simulasi kemacetan persimpangan yang dilakukan di *game* Traffic Jam dimana faktor uji kemacetan meliputi kepadatan lalu lintas, volume lalu lintas, dan waktu tunggu kendaraan. Dapat disimpulkan bahwa kontrol *agent* dapat mengurai kemacetan dengan prioritas kepadatan lalu lintas dimana rata-rata dari keseluruhan waktu penguraian lalu lintas *agent* dapat mengurai 16,5 detik lebih cepat daripada kontrol lampu statis. Dimana *agent* dengan model terbaik adalah *agent* 2 yang berada pada skenario simulasi persimpangan Dinoyo dengan waktu penguraian 18 detik lebih cepat daripada lampu statis. Kemudian hasil dari 4 rangkaian perbandingan dengan skenario-skenario berbeda tersebut dapat disimpulkan pada poin-poin berikut.

1. Dengan pengujian skenario yang telah dilakukan, aspek penilaian terhadap kepadatan lalu lintas diketahui *agent* 2 dan *agent* 1 adalah model terbaik dengan selisih rata 0.014 kepadatan terhadap lampu statis.
2. Dengan pengujian skenario yang telah dilakukan, aspek penilaian terhadap arus volume lalu lintas lalu lintas diketahui *agent* 2 adalah model terbaik dengan rata-rata selisih arus volume lalu lintas 7 unit/menit lebih tinggi dibanding lampu statis.

3. Dengan pengujian skenario yang telah dilakukan, aspek penilaian terhadap waktu tunggu kendaraan pada lalu lintas, semua *agent* dikalahkan oleh lampu statis dimana skenario Borobudur merupakan model terburuk dengan rata-rata selisih waktu tunggu kendaraan 2.31 detik lebih lambat daripada lampu statis.
4. Peningkatan efektifitas *throughput agent* dari percobaan yang telah dilakukan, penerapan kepadatan lalu lintas dan volume arus pada tiap lajur dapat dikelola *agent* dengan baik sehingga dapat merepresentasikan kemacetan pada simulasi.
5. Faktor jumlah kendaraan yang sedang melintasi persimpangan dan waktu sinyal lampu persimpangan tiap lajur dapat mempengaruhi perkembangan kepadatan dan kelancaran arus kemacetan pada persimpangan.
6. Distribusi bobot *policy reward* kontrol *agent* yaitu 0,4 poin *reward* untuk kepadatan dan 0,1 poin *reward* untuk waktu tunggu mempengaruhi kontrol cara *agent* mengurai kemacetan.

Pada akhir kesimpulan, dari skenario pelatihan dan uji coba yang telah dilakukan menghasilkan pengontrolan sinyal *agent* yang lebih unggul untuk melakukan penguraian tetapi kurang optimal untuk memberikan waktu tunggu lebih efisien.

5.2 Saran

Setelah dilakukannya uji coba terhadap penguraian kemacetan pada persimpangan menggunakan *agent Deep Q-Network* masih terdapat ketidaksempurnaan pada *agent* simulasi yang dikembangkan dengan *policy* yang ditentukan dimana *agent* masih inkonsisten dan tidak merata pada seluruh aspek dalam pengoptimalan lalu lintas. Untuk itu, berikut merupakan saran yang dihimbau oleh penulis kepada penelitian selanjutnya.

1. Diperlukan pengembangan kompleksitas arsitektur *Deep Q-Network* terutama pada pembagian *neuron* dan *layer* yang diperlukan untuk memahami pemrosesan data *training* yang lebih optimal.
2. Diperlukan pengembangan *environment* yang lebih stabil pada penggambaran dimensi *state* dan dimensi aksi untuk mensimulasikan persimpangan yang akan diuji, sehingga pengambilan data untuk pengambilan keputusan dan *training agent* lebih mudah dipahami dan lebih mudah untuk mencapai konvergensi.
3. Pada pengembangan *policy reward* diperlukan desain yang lebih optimal untuk menyeimbangkan antara penguraian kepadatan dan waktu tunggu kendaraan sehingga menghasilkan kenyamanan berkendara yang lebih efektif pada pengguna jalan.

DAFTAR PUSTAKA

Atmadji, E., & Syukron, A. (2022). Optimalisasi waktu tunggu mobil pada lampu merah di tiga persimpangan di Kota Yogyakarta. *Jurnal Kebijakan Ekonomi Dan Keuangan*, *1*(1), 126–133. https://doi.org/10.20885/jkek.vol1.iss1.art13

Bouktif, S., Cheniki, A., Ouni, A., & El-Sayed, H. (2023). Deep *reinforcement learning* for traffic signal control with consistent *state* and *reward* design approach. *Knowledge-Based Systems*, *267*, 110440. https://doi.org/10.1016/j.knosys.2023.110440

Bouzidi, E. H., Outtagarts, A., Langar, R., & Boutaba, R. (2021). *Deep Q-Network* and Traffic Prediction based Routing Optimization in Software Defined Networks. *Journal of Network and Computer Applications*, *192*(March), 103181. https://doi.org/10.1016/j.jnca.2021.103181

Ghoffar, A. (2004). *Tafsir Ibnu Katsir 1 a.pdf* (pp. 123–125).

Guberinic, S., Senborn, G., & Lazic, B. (2013). *Optimal Traffic Control Urban Intersection*. CRC Press.

Irawan, C. D., Mamahit, D. J., Sambul, A. M., Elektro, T., Teknik, F., Ratulangi, U. S., & Manado, J. K. B. (2019). *Pembuatan Game Simulasi Kewirausahaan untuk Profesi Petani*. *14*(1), 53–62.

Kementrian PUPR. (2017). *Modul 6 perencanaan perlengkapan jalan*. 11–81.

Kementrian PUPR. (2021). *tata-cara-perencanaan-geometrik-persimpangan-sebidang.pdf*.

Kim, D., & Jeong, O. (2020). Cooperative traffic signal control with traffic flow prediction in multi-intersection. *Sensors (Switzerland)*, *20*(1). https://doi.org/10.3390/s20010137

Kumar, S. (2020). *Balancing a CartPole System with Reinforcement Learning -- A Tutorial*. *0*. http://arxiv.org/abs/2006.04938

Li, L., Member, S., Lv, Y., & Wang, F. (2016). *Traffic Signal Timing via Deep Reinforcement Learning*. *3*(3), 247–254.

Li, Y., Wang, Z., Zhang, Q., & Liu, J. (2024). *Measuring Diversity of Game Scenarios*. 1–24. http://arxiv.org/abs/2404.15192

Liang, X., Du, X., Member, S., Wang, G., & Han, Z. (n.d.). *Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks*. *XX*(Xx), 1–11.

Ni, J., Li, F., & Wu, Z.-G. (2023). *Deep Reinforcement Learning Based Optimal Infinite-Horizon Control of Probabilistic Boolean Control Networks*. http://arxiv.org/abs/2304.03489

Niken Ekawati, N., Saleh Soeaidy, M., & Ribawanto, H. (2014). Kajian Dampak Pengembangan Pembangunan Kota Malang Terhadap Kemacetan Lalu Lintas (Studi Pada Dinas Perhubungan Kota Malang). *Jurnal Administrasi Publik (JAP)*, *2*(1), 129–133.

Pol, E. van der, & Oliehoek, F. A. (2016). Coordinated deep reinforcement learners for traffic light control. *30th Conference on Neural Information Processing Systems*

(NIPS), *Nips*, 1–8.

Rahadian, A. H., Saputra, M., & Ramadhanty, D. (2022). Analisis Implementasi Kebijakan Sistem Ganjil Genap Dalam Mengatasi Kemacetan Di Provinsi DKI Jakarta. *Jurnal Ilmiah Untuk Mewujudkan Masyarakat Madani*, *50*(1), 50–52. http://ojs.stiami.ac.id

Song, J., Zhao, C., Zhong, S., Nielsen, T. A. S., & Prishchepov, A. V. (2019). Mapping spatio-temporal patterns and detecting the factors of traffic congestion with multi-source data fusion and mining techniques. *Computers, Environment and Urban Systems*, *77*(June), 101364. https://doi.org/10.1016/j.compenvurbsys.2019.101364

Way, F. (1985). *Highway Capacity Manual*.

Xu, Y., Xi, Y., Li, D., & Zhou, Z. (2016). Traffic Signal Control based on *Markov Decision Process*. *IFAC-PapersOnLine*, *49*(3), 67–72. https://doi.org/10.1016/j.ifacol.2016.07.012

Zhang, L., Du, B., Telikani, A., & Wu, J. (2022). *DynamicLight : Dynamically Tuning Traffic Signal Duration with DRL DynamicLight : Dynamically Tuning Traffic Signal Duration with DRL*. *November*. https://doi.org/10.48550/arXiv.2211.01025

LAMPIRAN

LAMPIRAN 1

Hasil Evaluasi Kontrol Agent 1

	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**
episode	72	160	160	160	181
time	141.73	153.43	132.35	136.83	141.2
reward	14.01	15.36	13.8	14.5	19.43
totalDemandVehicles	216	167	142	155	122
passedVehicles	27	65	64	63	62
totalAvgVehiclesPerPhase	0.062834	0.044963	0.057129	0.050637	0.043232
totalAvgTrafficVolumePerPhase	166.8307	144.7716	158.3131	143.2621	138.5564
totalAvgTrafficDensityPerPhase	0.062834	``0.044963	0.057129	0.050637	0.043232
lowestTrafficVolume	0	0	0	0	0
highestTrafficVolume	1028.476	973.3279	1137.621	858.4792	981.509
troughputEfficiency	0.125	0.389222	0.450704	0.406452	0.508197
totalWaitTime/vehicle	2.767053	2.176934	1.901983	1.230639	3.183994
reward/vehicle	0.062456	0.086967	0.090709	0.088135	0.152755
density/vehicle	0.000291	0.000269	0.000402	0.000327	0.000354
lane0_avgVehicles	0.04847	0.026024	0.04899	0.030848	0.024853
lane0_avgDensity	0.048	0.026	0.049	0.031	0.025
lane0_avgVolume	146.425	126.789	159.89	155.933	152.275
lane0_avgWaitTime	5.48	3.53	4.12	3.61	1.68
lane1_avgVehicles	0.077199	0.063902	0.065269	0.070425	0.06161
lane1_avgDensity	0.077	0.064	0.065	0.07	0.062
lane1_avgVolume	187.236	162.755	156.736	130.591	124.838
lane1_avgWaitTime	3.02	2.87	2.74	2.4	2.2

LAMPIRAN 2

Hasil Evaluasi Kontrol Agent 2

	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**
episode	122	122	122	122	122
time	185.27	197.04	193.47	188.54	187.39
reward	18	13.51	14.92	17.66	19.62
totalDemandVehicles	96	148	153	154	109
passedVehicles	65	60	60	65	66
totalAvgVehiclesPerPhase	0.030019	0.038947	0.038276	0.039122	0.039183
totalAvgTrafficVolumePerPhase	75.8756	73.52038	74.47395	74.939	75.75002
totalAvgTrafficDensityPerPhase	0.030019	0.038947	0.038276	0.039122	0.039183
lowestTrafficVolume	0	0	0	0	0
highestTrafficVolume	858.8342	865.6964	900.2134	845.17	871.2958
troughputEfficiency	0.677083	0.405405	0.392157	0.422078	0.605505
totalWaitTime/vehicle	22.864	3.049978	2.499763	3.111773	3.907016
reward/vehicle	0.178338	0.087395	0.09401	0.109656	0.172538
density/vehicle	0.000313	0.000263	0.00025	0.000254	0.000359
lane0_avgVehicles	0.00817	0.010106	0.010778	0.013791	0.011552
lane0_avgDensity	0.008	0.01	0.011	0.014	0.012
lane0_avgVolume	39.077	37.006	36.69	41.921	43.318
lane0_avgWaitTime	0.23	3.39	3.4	3.89	1.24
lane1_avgVehicles	0.027288	0.02281	0.023103	0.023195	0.02399
lane1_avgDensity	0.027	0.023	0.023	0.023	0.024
lane1_avgVolume	51.469	51.834	51.509	48.308	48.617
lane1_avgWaitTime	6.99	5.67	5.71	5.94	6.06
lane2_avgVehicles	0.054597	0.083924	0.080948	0.08038	0.082006
lane2_avgDensity	0.055	0.084	0.081	0.08	0.082
lane2_avgVolume	137.081	131.722	135.223	134.588	135.315
lane2_avgWaitTime	3.12	3.13	3.01	2.8	2.59

LAMPIRAN 3

Hasil Evaluasi Kontrol Agent 3

	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**
episode	103	103	103	103	103
time	130.49	113.08	131.72	131.2	126.93
reward	4.8	4.54	6.41	4.62	7.95
totalDemandVehicles	81	87	67	97	96
passedVehicles	56	55	48	46	56
totalAvgVehiclesPerPhase	0.046547	0.043713	0.04906	0.047344	0.046346
totalAvgTrafficVolumePerPhase	75.70018	70.53973	69.0202	63.65054	76.27022
totalAvgTrafficDensityPerPhase	0.046547	0.043713	0.04906	0.047344	0.046346
lowestTrafficVolume	0	0	0	0	0
highestTrafficVolume	806.1504	795.3264	822.5786	782.6039	801.0602
troughputEfficiency	0.691358	0.632184	0.716418	0.474227	0.583333
totalWaitTime/vehicle	6.031909	5.101406	11.81003	8.698323	9.918477
reward/vehicle	0.050274	0.041997	0.08959	0.041728	0.074495
density/vehicle	0.000575	0.000502	0.000732	0.000488	0.000483
lane0_avgVehicles	0.014143	0.014485	0.018281	0.019653	0.017593
lane0_avgDensity	0.014	0.014	0.018	0.02	0.018
lane0_avgVolume	85.357	89.912	98.805	92.062	84.244
lane0_avgWaitTime	0	0	0.3	0.15	0.33
lane1_avgVehicles	0.089201	0.086477	0.09988	0.098094	0.091501
lane1_avgDensity	0.089	0.086	0.1	0.098	0.092
lane1_avgVolume	33.313	35.098	10.848	10.111	35.292
lane1_avgWaitTime	12.63	11.56	0	0	13.07
lane2_avgVehicles	0.027638	0.026547	0.02906	0.028946	0.029712
lane2_avgDensity	0.028	0.027	0.029	0.029	0.03
lane2_avgVolume	39.858	39.999	37.202	47.867	58.911
lane2_avgWaitTime	3.91	3.65	3.78	3.83	3.74
lane3_avgVehicles	0.055206	0.047344	0.049021	0.042684	0.046576
lane3_avgDensity	0.055	0.047	0.049	0.043	0.047
lane3_avgVolume	144.273	117.15	129.225	104.562	126.634
lane3_avgWaitTime	7.41	8.73	4.83	8.32	7.01

LAMPIRAN 4

Hasil Evaluasi Kontrol Agent 4

	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**
episode	25	25	25	25	26
time	92.98	82.49	79.81	85.61	79.06
reward	8.56	7.8	9.77	9.12	7.28
totalDemandVehicles	75	72	53	76	72
passedVehicles	31	31	28	27	30
totalAvgVehiclesPerPhase	0.031247	0.036673	0.045168	0.033707	0.049686
totalAvgTrafficVolumePerPhase	56.75644	59.04221	57.46617	58.46923	59.68723
totalAvgTrafficDensityPerPhase	0.031247	0.036673	0.045168	0.033707	0.049686
lowestTrafficVolume	0	0	0	0	0
highestTrafficVolume	663.7849	479.6654	479.5744	481.2753	672.118
troughputEfficiency	0.413333	0.430556	0.528302	0.355263	0.416667
totalWaitTime/vehicle	8.608137	5.339314	12.69581	6.382853	4.04605
reward/vehicle	0.104615	0.095854	0.17877	0.113844	0.094673
density/vehicle	0.000417	0.000509	0.000852	0.000444	0.00069
lane0_avgVehicles	0.014795	0.018982	0.010871	0.017695	0.016739
lane0_avgDensity	0.015	0.019	0.011	0.018	0.017
lane0_avgVolume	53.351	44.58	50.957	48.355	49.392
lane0_avgWaitTime	1.61	7.34	2.27	6.72	5.93
lane1_avgVehicles	0.030784	0.036662	0.035743	0.036265	0.012947
lane1_avgDensity	0.031	0.037	0.036	0.036	0.013
lane1_avgVolume	45.305	52.445	46.376	44.262	51.826
lane1_avgWaitTime	7.52	7.95	7.64	8.25	1.87
lane2_avgVehicles	0.048161	0.054376	0.08889	0.047163	0.119373
lane2_avgDensity	0.048	0.054	0.089	0.047	0.119
lane2_avgVolume	71.613	80.101	75.066	82.79	77.843
lane2_avgWaitTime	1.52	2.23	4.09	1.84	5.68

LAMPIRAN 5

Hasil Evaluasi Kontrol Statis 1

	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**
episode	1	10	29	70	179
time	167.09	160.14	156.85	158.72	135.33
reward	4.14	4.12	4.54	4.78	5.69
totalDemandVehicles	179	3024	178	200	171
passedVehicles	61	63	69	61	55
totalAvgVehiclesPerPhase	0.065917	0.066365	0.065041	0.058519	0.067686
totalAvgTrafficVolumePerPhase	144.8852	148.5834	150.4181	138.8879	147.1689
totalAvgTrafficDensityPerPhase	0.065917	0.066365	0.065041	0.058519	0.067686
lowestTrafficVolume	0	0	0	0	0
highestTrafficVolume	809.764	768.9298	724.8019	780.2733	814.1078
troughputEfficiency	0.340782	0.020833	0.352601	0.305	0.321637
totalWaitTime/vehicle	1.415769	0.082199	1.357252	1.913175	1.198599
reward/vehicle	0.020641	0.0012	0.021636	0.022341	0.030177
density/vehicle	0.000368	2.19E-05	0.000376	0.000293	0.000396
lane0_avgVehicles	0.053932	0.057178	0.051266	0.035207	0.049567
lane0_avgDensity	0.054	0.057	0.051	0.035	0.05
lane0_avgVolume	130.06	134.671	138.841	136.212	164.15
lane0_avgWaitTime	4.7	4.73	4.39	5.57	4.3
lane1_avgVehicles	0.077902	0.075552	0.078815	0.081831	0.085806
lane1_avgDensity	0.078	0.076	0.079	0.082	0.086
lane1_avgVolume	159.711	162.496	161.995	141.563	130.187
lane1_avgWaitTime	3.88	3.45	3.67	4.05	3.12

LAMPIRAN 6

Hasil Evaluasi Kontrol Statis 2

	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**
episode	119	119	119	119	119
time	225.54	213.95	198.13	199.3	204.93
reward	6.99	7.01	7.11	6.27	6.53
totalDemandVehicles	172	170	178	165	180
passedVehicles	67	58	59	60	59
totalAvgVehiclesPerPhase	0.052119	0.048592	0.052637	0.050421	0.051946
totalAvgTrafficVolumePerPhase	68.51809	58.88217	71.51566	70.40757	69.06875
totalAvgTrafficDensityPerPhase	0.052119	0.048592	0.052637	0.050421	0.051946
lowestTrafficVolume	0	0	0	0	0
highestTrafficVolume	444.1013	459.2948	429.7904	505.9819	364.6807
troughputEfficiency	0.389535	0.341177	0.331461	0.363636	0.327778
totalWaitTime/vehicle	3.920817	8.792573	3.002644	7.099568	6.345191
reward/vehicle	0.036595	0.038415	0.036742	0.035768	0.033705
density/vehicle	0.000303	0.000286	0.000296	0.000306	0.000289
lane0_avgVehicles	0.024243	0.020177	0.022159	0.02216	0.023474
lane0_avgDensity	0.024	0.02	0.022	0.022	0.023
lane0_avgVolume	49.574	33.855	45.06	49.17	49.415
lane0_avgWaitTime	7.89	6.13	7.96	4.87	6.56
lane1_avgVehicles	0.014404	0.013964	0.016615	0.013809	0.014108
lane1_avgDensity	0.014	0.014	0.017	0.014	0.014
lane1_avgVolume	30.177	34.958	44.047	35.626	39.288
lane1_avgWaitTime	11.65	3.34	8.6	3.12	18.55
lane2_avgVehicles	0.117711	0.111634	0.119136	0.115294	0.118255
lane2_avgDensity	0.118	0.112	0.119	0.115	0.118
lane2_avgVolume	125.804	107.834	125.44	126.427	118.503
lane2_avgWaitTime	6.53	7.57	6.48	6.15	7.45

LAMPIRAN 7

Hasil Evaluasi Kontrol Statis 3

	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**
episode	79	79	79	79	79
time	185.85	169.71	180.27	176.2	179.3
reward	5.89	5.84	5.84	5.84	5.82
totalDemandVehicles	203	201	200	206	198
passedVehicles	61	64	67	65	69
totalAvgVehiclesPerPhase	0.038051	0.039398	0.039752	0.039907	0.037597
totalAvgTrafficVolumePerPhase	60.44343	68.45967	67.65871	67.52843	65.88382
totalAvgTrafficDensityPerPhase	0.038051	0.039398	0.039752	0.039907	0.037597
lowestTrafficVolume	0	0	0	0	0
highestTrafficVolume	730.4706	597.9124	547.1006	559.3491	777.4272
troughputEfficiency	0.300493	0.318408	0.335	0.315534	0.348485
totalWaitTime/vehicle	4.670223	4.765624	5.063025	4.465392	4.9077
reward/vehicle	0.025669	0.025643	0.025606	0.024971	0.025771
density/vehicle	0.000187	0.000196	0.000199	0.000194	0.00019
lane0_avgVehicles	0.048947	0.050576	0.040576	0.048081	0.041584
lane0_avgDensity	0.049	0.051	0.041	0.048	0.042
lane0_avgVolume	60.04	65.869	62.902	64.063	60.52
lane0_avgWaitTime	7.79	6.64	6.5	6.89	4.51
lane1_avgVehicles	0.028762	0.02985	0.037554	0.032213	0.033775
lane1_avgDensity	0.029	0.03	0.038	0.032	0.034
lane1_avgVolume	46.668	41.541	46.907	46.643	55.298
lane1_avgWaitTime	3.07	3.73	6.83	6.04	5.29
lane2_avgVehicles	0.037842	0.035855	0.034084	0.037822	0.032493
lane2_avgDensity	0.038	0.036	0.034	0.038	0.032
lane2_avgVolume	50.26	62.847	47.572	63.499	45.738
lane2_avgWaitTime	10.55	10.77	9.63	9.25	9.78
lane3_avgVehicles	0.036652	0.041311	0.046797	0.041511	0.042538
lane3_avgDensity	0.037	0.041	0.047	0.042	0.043
lane3_avgVolume	84.806	103.582	113.254	95.909	101.98
lane3_avgWaitTime	11.78	7.29	10	9.82	8.76

LAMPIRAN 8

Hasil Evaluasi Kontrol Statis 4

	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**
episode	24	24	24	24	24
time	93.46	99.93	100.49	96.46	98.73
reward	3.52	3.79	3.47	3.83	3.76
totalDemandVehicles	66	72	70	73	74
passedVehicles	30	31	30	30	30
totalAvgVehiclesPerPhase	0.039836	0.042322	0.040652	0.042544	0.042801
totalAvgTrafficVolumePerPhase	59.36071	57.74355	55.45355	60.40258	58.81432
totalAvgTrafficDensityPerPhase	0.039836	0.042322	0.040652	0.042544	0.042801
lowestTrafficVolume	0	0	0	0	0
highestTrafficVolume	311.2174	308.1817	297.3552	312.711	302.443
troughputEfficiency	0.454546	0.430556	0.428571	0.410959	0.405405
totalWaitTime/vehicle	3.567091	5.615762	3.698748	5.326511	5.412564
reward/vehicle	0.042225	0.043243	0.039253	0.043299	0.042024
density/vehicle	0.000604	0.000588	0.000581	0.000583	0.000578
lane0_avgVehicles	0.023252	0.022513	0.022145	0.023773	0.021365
lane0_avgDensity	0.023	0.023	0.022	0.024	0.021
lane0_avgVolume	51.787	45.19	45.736	52.697	40.366
lane0_avgWaitTime	9.04	9.71	9.73	9.46	9.43
lane1_avgVehicles	0.011522	0.020221	0.013903	0.019694	0.020968
lane1_avgDensity	0.012	0.02	0.014	0.02	0.021
lane1_avgVolume	48.59	69.934	55.503	68.392	61.267
lane1_avgWaitTime	1.8	2.93	2.31	2.9	3.46
lane2_avgVehicles	0.084733	0.084233	0.085907	0.084166	0.086068
lane2_avgDensity	0.085	0.084	0.086	0.084	0.086
lane2_avgVolume	77.705	58.107	65.121	60.119	74.81
lane2_avgWaitTime	4.42	4.93	5.04	4.75	4.72