BAB 2
KAJIAN PUSTAKA

Bab ini membahas mengenai kajian teori dari permasalahan yang diteliti yaitu penerapan *Reinforcement Learning* (RL) pada *Traffic Signal Control* (TSC) menggunakan algoritma *Deep Q-Network* (DQN) dengan data yang berasal dari *Simulation of Urban Mobility* (SUMO). Selain itu pada bab ini juga menjelaskan mengenai penelitian terdahulu yang memiliki keterkaitan dan menjadi landasan maupun kerangka konsep dari penelitian ini.

2.1 TEORI PENUNJANG

2.1.1 *Traffic Signal Control*

Traffic Signal Control adalah sistem yang dirancang untuk mengatur pergerakan lalu lintas di persimpangan jalan melalui penggunaan sinyal lampu lalu lintas. Tujuan utamanya adalah mengoptimalkan arus kendaraan, mengurangi kemacetan, dan meningkatkan keselamatan bagi semua pengguna jalan, termasuk pejalan kaki dan pengendara sepeda (Huang, 2024). Sistem ini merupakan komponen penting dalam manajemen lalu lintas perkotaan dan berperan signifikan dalam sistem transportasi modern.

A. Prinsip Dasar *Traffic Signal Control*

Operasi dasar dari *traffic signal control* melibatkan pengaturan fase dan siklus sinyal untuk mengelola pergerakan lalu lintas di persimpangan. Fase merujuk pada periode waktu di mana pergerakan tertentu diizinkan, bergerak lurus atau berbelok. Siklus sinyal adalah urutan lengkap dari semua fase yang terjadi sebelum urutan tersebut dimulai kembali (Koonce dkk., 2008). Elemen penting dalam siklus sinyal dapat dilihat pada Tabel 2.1 berikut.

Tabel 2.1 Elemen siklus sinyal *Traffic Signal Controll*

Istilah	Deskripsi
Green Time	Waktu di mana pergerakan lalu lintas diizinkan.
Yellow Time	Waktu peringatan sebelum transisi ke fase merah.
Red Time	Waktu di mana pergerakan lalu lintas dilarang.

Efektivitas sistem *traffic signal control* dipengaruhi oleh beberapa faktor, termasuk volume lalu lintas, variasi kepadatan pada berbagai waktu (jam sibuk dan non-jam sibuk), kecepatan rata-rata kendaraan, dan karakteristik fisik persimpangan (Henry, 2005).

B. Klasifikasi *Traffic Signal Control*

Sistem traffic signal control dapat diklasifikasikan berdasarkan metode pengaturan sinyalnya sebagai berikut:

1. *Fixed-Time Traffic Signal Control* (FTTS)

   FTTS adalah sistem di mana durasi setiap fase sinyal ditetapkan sebelumnya berdasarkan data historis lalu lintas. Sistem ini tidak responsif terhadap perubahan kondisi lalu lintas real-time (L. Zhang, Xie, dkk., 2021).

2. *Actuated Traffic Signal* (ATS)

   ATS menggunakan sensor atau detektor kendaraan untuk menyesuaikan waktu sinyal berdasarkan permintaan aktual. Sistem ini lebih responsif dibandingkan FTTS karena dapat menyesuaikan fase sinyal secara otomatis (L. Zhang, Xie, dkk., 2021). Dalam implementasinya terdapat implementasi *semi actuated* dan *fully actuated. Semi actuated* sensor hanya ditempatkan pada lengan minor, arus utama selalu mendapat hijau minimal, *fully actuated* sedangkan setiap lengan simpang memiliki detektor (Koonce dkk., 2008). Alur kerja pada ATS ini fase hijau lalu lintas diberikan menyesuaikan permintaan kaki simpang, jika tidak ada permintaan maka fase hijau diberikan ke kaki simpang yang sudah ditentukan diawal konfigurasi, biasanya yang dianggap jalur utama (Nema, 2009). Ketika sistem sedang melayani fase hijau pada kaki simpang tertentu, dan kaki simpang tersebut sangat padat maka sensor deteksi terus memberi sinyal dan *controller* terus menambahkan durasi hijau sampai batas maksimum, akibatnya kaki simpang lain terlambat untuk dilayani (Koonce dkk., 2008).

3. *Adaptive Traffic Signal Control* (ATSC)

   ATSC adalah sistem yang menggunakan data real-time dan algoritma berbasis AI untuk menyesuaikan waktu sinyal secara dinamis. Sistem ini mengumpulkan data secara terus-menerus dan mengoptimalkan waktu sinyal untuk meminimalkan panjang antrean dan penundaan di persimpangan (Genders & Razavi, 2019).

C. Fase dan Siklus Traffic Signal Control

Fase (*phase*) adalah interval hijau, kuning, merah yang melayani satu kelompok arah pergerakan kaki simpang. Misal pada suatu kaki simpang saat ini yang diizinkan untuk jalan adalah kaki simpang dari arah utar maka ini terhitung satu fase, atau jika pada saat ini yang diizinkan untuk jalan adalah kaki simpang dari arah utara dan dari arah selatan secara bersamaan maka ini juga terhitung satu fase (Wolshon & Pande, 2009). Sedangkan siklus (*Cycle*) merupakan satu putaran penuh urutan sinyal dari awal fase pertama kembali ke awal fase yang sama. Durasi totalnya disebut *cycle length* dan diukur dalam detik. Setiap kendaraan, pejalan kaki, atau arah lalu lintas dijamin memperoleh hak hijau minimal satu kali dalam satu siklus tetap (Koonce dkk., 2008). Berdasarkan pengendalian nya, terdapat siklus yang tetap (*cyclic*) dimana setiap fase selalu mendapat giliran dan urutan yang sama dalam satu siklus, kemudian siklus acak (*acyclic*) dimana fase diberikan tanpa urutan (Koonce dkk., 2008).

2.1.2 *Reinforcement Learning*

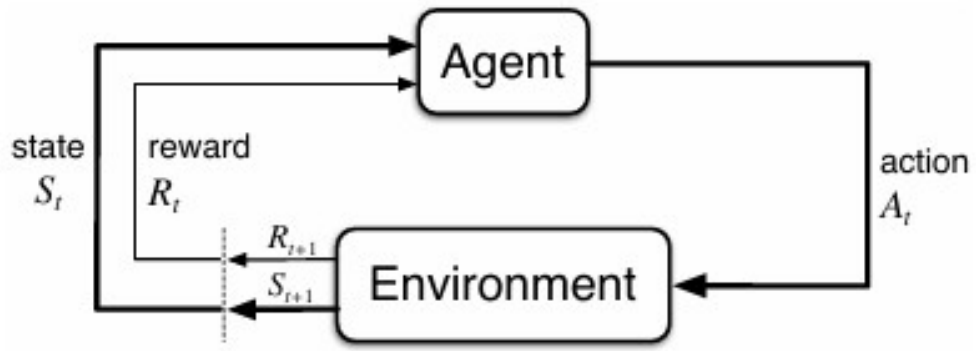
Reinforcement Learning (RL) adalah salah satu paradigma dalam pembelajaran mesin di mana agen belajar berinteraksi dengan lingkungan melalui mekanisme *trial* and *error*. Agen menerima umpan balik dalam bentuk *reward* atau penalti, dengan tujuan untuk memaksimalkan *reward* kumulatif dalam jangka panjang (Sutton & Barto, 2018). Berbeda dengan pembelajaran terawasi (*supervised learning*) yang membutuhkan pasangan *input-output* yang berlabel, RL tidak memerlukan data pelatihan yang eksplisit, melainkan belajar dari konsekuensi tindakan yang diambil (François-Lavet dkk., 2018).

Menurut Sutton dan Barto (2018), pembelajaran penguatan berjalan di atas *Markov Decission Process* (MDP), yang mencakup beberapa komponen utama berikut.

Tabel 2.2 Komponen *Reinforcement Learning*

Komponen	Deskripsi
State (S)	Kondisi lingkungan saat ini.
Action (A)	Pilihan tindakan yang dapat diambil agen dalam suatu state tertentu.
Reward (R)	Umpan balik numerik yang diberikan setelah agen mengambil suatu tindakan.
Policy (π)	Strategi yang digunakan agen untuk menentukan tindakan berdasarkan state saat ini.
Value (V)	Estimasi keuntungan jangka panjang dari suatu state atau aksi.

Agen dalam RL bertujuan untuk menemukan *policy* optimal yang memungkinkan mereka mengambil tindakan terbaik untuk memaksimalkan *reward* kumulatif (Sutton & Barto, 2018). Untuk mencapai tujuan ini, agen perlu menyeimbangkan antara eksplorasi (mencoba tindakan baru untuk menemukan hasil yang lebih baik) dan eksploitasi (memilih tindakan yang sudah diketahui memberikan reward tinggi) (François-Lavet dkk., 2018).



Gambar 2.1 Proses *Reinforcement Learning*

(sumber:Sutton & Barto, 2018)

2.1.3 *Reinforcement Learning* untuk *Traffic Signal Control*

Salah satu penerapan utama RL adalah dalam pengendalian sinyal lalu lintas, di mana sistem RL dapat menyesuaikan waktu sinyal secara *real-time* berdasarkan kondisi lalu lintas untuk mengoptimalkan efisiensi transportasi (Gao dkk., 2017). Dibandingkan dengan metode konvensional *Fixed-Time Traffic Signal* (FTTS) dan *Actuated Traffic*

Signal (ATS), RL memiliki keunggulan dalam menyesuaikan sinyal tanpa memerlukan pemodelan lalu lintas yang kompleks.

Dalam konteks pengendalian sinyal lalu lintas, agen RL berinteraksi dengan lingkungan (jaringan jalan dan kendaraan) untuk mengoptimalkan aliran lalu lintas. Proses ini dapat dimodelkan sebagai *Markov Decision Process*. Dalam model tersebut, *State* (S) mencerminkan kondisi aktual lalu lintas di persimpangan, misalnya jumlah kendaraan yang menunggu atau tingkat kepadatan pada setiap arah. *Action* (A) merepresentasikan pilihan yang tersedia bagi agen kontrol, contohnya mengubah fase sinyal atau menyesuaikan durasi lampu hijau untuk sebuah jalur tertentu. Setelah tindakan diambil, agen menerima *Reward* (R) berupa umpan balik yang dirancang untuk mendukung tujuan pengendalian, misalnya meminimalkan total waktu tunggu atau mengurangi panjang antrean. Strategi pengambilan keputusan, yaitu *Policy* (π), selanjutnya menentukan peta dari setiap kondisi lalu lintas (*state*) ke tindakan optimal yang memaksimalkan akumulasi *reward* seiring waktu (Abdulhai dkk., 2003).

Agen RL belajar melalui interaksi berulang dengan lingkungan lalu lintas, dengan menyesuaikan *policy*-nya untuk meningkatkan kinerja berdasarkan *reward* yang diterima. Pendekatan ini memungkinkan sistem pengendalian sinyal untuk beradaptasi dengan perubahan pola lalu lintas yang berubah-ubah atau tidak menentu dari waktu waktu, tanpa memerlukan pemrograman eksplisit yang dikhususkan atau didefinisikan untuk masing-masing skenario.

2.1.4 Implementasi *Reinforcement Learning*

Tahapan implementasi reinforcement learning (RL) secara konseptual berbeda dari alur kerja *machine learning supervised* dan *unsupervised* karena prosesnya diawali bukan dari pengumpulan data statis (François-Lavet dkk., 2018). Dengan membandingkan metodologi, eksperimen, dan penerapan dari berbagai publikasi,

diidentifikasi serangkaian langkah umum yang membentuk kerangka implementasi RL sebagai berikut.

1. Formulasi Masalah ke *Markov Decission Process* (MDP)

RL mensyaratkan masalah harus bisa dirumuskan sebagai *Markov Decision Process* (MDP), dimana masalah dapat ditentukan apa yang menjadi himpunan *state*, *action* dan *reward* nya (Sutton & Barto, 2018).

2. Pembangunan atau Pemilihan Lingkungan/E*nvirontment*

   Pengembang RL bergantung pada adanya *environment* untuk melakukan interaksi antara penerapan aksi yang diambil dan dampak yang diperolah, *environment* dapat berupa fisik atau simulator (François-Lavet dkk., 2018)

3. Pemilihan Algoritma & Arsitektur

   Untuk menyelesaikan permasalahan dengan RL algoritma menjadi bagian terpenting, keberhasilan algoritma bergantung pada ruang MDP yang telah ditentukan karena algoritma yang mengambil kebijakan dalam RL (Watkins & Dayan, 1992).

4. Implementasi Agen

   Berdasarkan formulasi masalah kedalam MDP yang diidentifikasi agen dibangun berdasarkan nilai yang jelas untuk setiap State, Action dan Reward untuk mencapai tujuan yang diselesaiakan (Sutton & Barto, 2018).

5. Pelatihan Agen

   Agen dijalankan dalam episode berulang untuk mempelajari nilai *state, action*, dan *reward* yang didapat dari setiap transisi pembelajaran. Kurva *reward* per episode dipantau untuk memeriksa konvergensi pembelajaran (Lin, 1992)

6. Evaluasi & Validasi

   Evaluasi kinerja RL biasanya diukur dengan *cumulative return*, *average reward*, atau metrik domain lain yang diharapkan (François-Lavet dkk., 2018).

2.1.5 *Algoritma Q-Learning*

Q-*Learning* adalah teknik *Reinforcement Learning* di mana sebuah agen belajar seberapa bagus (nilai Q) setiap aksi/*action* (A) pada suatu keadaan/*state* (S). Agen tidak membutuhkan peta transisi sistem (*model-free*). Agen cukup mencoba, mendapat hadiah (*reward*), lalu memperbaiki tabel Q-nya hingga tahu aksi mana yang paling menguntungkan di tiap keadaan. *Q-learning* terdiri dari beberapa komponen inti yang terlihat pada Tabel 2.3 berikut.

Tabel 2.3 Komponen inti *Q-Learning*

Komponen	**Makna singkat**
Q-table	Tabel 2-dimensi. Baris = keadaan, kolom = aksi. Nilai di dalamnya disebut *Q-value*: prediksi total *reward* masa depan bila aksi itu diambil.
α (*learning rate*)	Seberapa cepat *Q-value* lama diganti nilai baru (0 – 1).
γ (*discount factor*)	Seberapa penting hadiah masa depan dibanding hadiah langsung (0 – 1).
ε (*epsilon*)	Peluang agen berjalan acak (*explore*) daripada memilih aksi terbaik saat ini (*exploit*).

Cara kerja *Q-Learning* merupakan proses belajar melalui percobaan dan kesalahan (*trial-and-error*) yang dilakukan secara berulang-ulang. Mula-mula agen membuat *Q-table* berisi nol untuk setiap pasangan keadaan-aksi. Pada setiap langkah di sebuah episode, agen memilih aksi dengan strategi *ε-greedy* sebagian kecil waktu (ε) ia mencoba aksi acak guna mengeksplorasi, sisanya ia mengambil aksi dengan *Q-value* tertinggi untuk mengeksploitasi pengetahuan saat ini. Setelah aksi dijalankan, lingkungan mengembalikan reward dan keadaan baru. Agen kemudian memperbarui nilai $Q(S_t, A_t)$ menggunakan rumus:

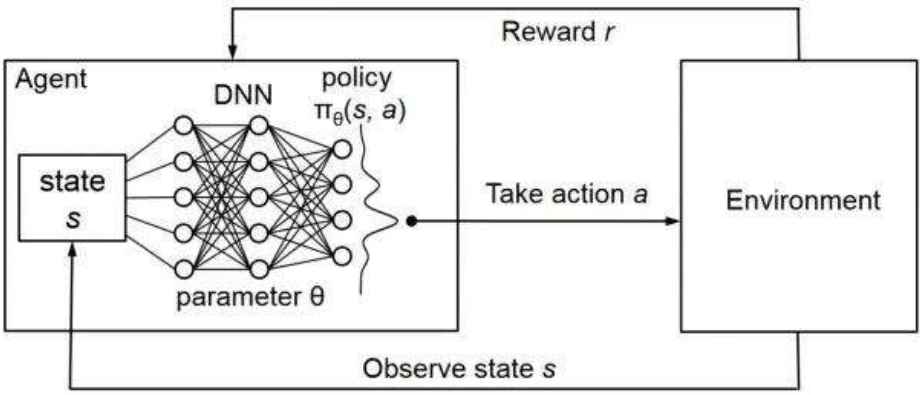
$$\boldsymbol{Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]} \qquad \textbf{(2.1)}$$

Agent mengulang proses dalam banyak episode dan secara bertahap menurunkan nilai ε untuk mengurangi eksplorasi, agen akhirnya mengisi tabel Q dengan estimasi nilai optimal, di titik ini cukup memilih aksi dengan *Q-value* tertinggi di setiap keadaan untuk mendapatkan kebijakan terbaik (Watkins & Dayan, 1992).

2.1.6 *Algoritma DQN*

Algoritma *Deep Q-Network* pertama kali diusulkan pada penelitian Mnih dkk pada tahun 2015. Penelitian ini menjadi tolak ukur dan titik awal bagi banyak penelitian *deep machine learning* (Roderick dkk., 2017). DQN mampu menggabungkan machine learning dengan kelas jaringan saraf tiruan yang dikenal dengan *deep neural network* (Mnih dkk., 2015).

Deep Q-Network (DQN) adalah algoritma *reinforcement learning* yang mengintegrasikan algoritma *Q-Learning* dengan *Deep Neural Networks* untuk memperkirakan fungsi nilai-Q secara efektif. Kombinasi ini memungkinkan DQN untuk menangani ruang keadaan dimensi tinggi yang diperjuangkan oleh *Q-Learning* tradisional (Z. Li dkk., 2021). Struktur umum dari algoritma DQN dapat dilihat pada Gambar 2.2 berikut.



Gambar 2.2 Diagram algoritma *Deep Q-Network*

(sumber: Jayakody, 2022)

Jaringan saraf dalam *Q-Learning* mengambil status sebagai masukan dan menghasilkan Nilai-Q untuk setiap tindakan di ruang tindakan (Mnih dkk., 2015). Tujuan dari jaringan saraf adalah untuk mempelajari dan melatih parameter agar dapat mendekati fungsi nilai tindakan optimal (Sutton & Barto, 2018). Selama proses prediksi, jaringan terlatih ini digunakan untuk memprediksi tindakan terbaik berikutnya yang terjadi di lingkungan dengan memilih tindakan yang memiliki nilai-Q tertinggi. Pada dasarnya, Q-Learning menentukan fungsi nilai tindakan *(action-value function*) untuk kebijakan target tertentu, yang pada akhirnya memilih tindakan dengan nilai terbaik (Watkins & Dayan, 1992).

2.1.7 *Simulation of Urban Mobility (SUMO)*

Simulation of Urban Mobility (SUMO) merupakan simulator lalu lintas jalan raya *open source*, sangat portabel, dan paket simulasi lalu lintas pada jalan raya yang didesain untuk digunakan pada skenario simulasi yang luas (Lopez dkk., 2018). SUMO

mendukung banyak aplikasi yang membantu dalam mempersiapkan model simulasi lalu lintas yang sangat mendetail, contohnya jumlah jalur jalan raya, tipe kendaraan, kecepatan kendaraan, lampu lalu lintas dan masih banyak lainnya yang dapat secara bebas dikonfigurasi. SUMO memungkinkan pengguna untuk membuat model lalu lintas yang realistis, mensimulasikan berbagai skenario lalu lintas, dan menguji berbagai strategi pengaturan lalu lintas tanpa risiko pada jalan raya sebenarnya. SUMO telah menyediakan kerangka kerja besar dengan alat yang berguna untuk pembuatan, validasi, dan evaluasi skenario lalu lintas besar (Lopez dkk., 2018). SUMO juga tersedia secara gratis dan diterbitkan di bawah *Eclipse Public License* V2. SUMO digunakan di seluruh dunia dan diunduh lebih dari 35.000 kali setiap tahun (Lopez dkk., 2018).

Simulasi memperbarui posisi kendaraan dalam langkah temporal dengan durasi yang ditentukan pengguna Δt memindahkannya dengan kenaikan posisi $\Delta x(t)$. Secara default, SUMO menggunakan skema *Euler* orde pertama $\Delta x(t) = \Delta t \cdot v(t+\Delta t)$, dimana $v(t+\Delta t) = v(t)+\alpha t \cdot a(t)$ dan $a(t)$ adalah percepatan yang dipilih model yang mengikuti mobil. Hal ini dapat diartikan sebagai kendaraan yang melaju dengan kecepatan tetap dalam satu langkah waktu dan mengubah kecepatan tersebut secara instan di antara dua langkah. Secara default, model SUMO hanya berpindah jalur dengan jalur yang searah (Lopez dkk., 2018).

A. Cara Kerja SUMO

Untuk memanfaatkan simulasi pada aplikasi SUMO, maka dapat dilakukan dengan melakukan tahapan berikut.

1. Pembuatan jaringan, dilakukan untuk membuat ruang simulasi, dengan memanfaatkan *netconvert* dapat mengubah data OSM/*Shapefile* menjadi berkas .net.xml.
2. Generasi rute, dilakukan untuk menentukan pergerakan objek simulasi contohnya kendaraan, dapat memanfaatkan *duarouter* atau *generator* lalu-lintas yang menghasilkan .rou.xml berdasarkan *volume*, *OD-matrix*, atau *demand* acak.
3. Pengaturan sinyal, dilakukan untuk dapat menentukan komponen lalu lintas diantaranya *Traffic Signal Control*, dengan memanfaatkan *netedit* atau skrip XML.
4. Simulasi, menjalankan simulasi sesuai dengan ruang simulasi, pergerakan dan komponen lalu lintas yang sudah dibangun dengan perintah *sumo/sumo-gui.* Dari

proses menjalankan simulasi ini, menghasilkan data contohnya *delay*, antrean, emisi dan data lainnya yang dibutuhkan.

5. Kontrol *real-time*, untuk mendapatkan data atau mengendalikan proses simulasi secara realtime dapat memanfaatkan integrasi SUMO dengan API TraCI. Integrasi ini mengizinkan program eksternal membaca/menulis data simulasi, sehingga memungkinkannya diintegrasikan dengan Python, C++ dan Java. (Eclips, 2025).

B. Studi Kasus Pemanfaatan SUMO

SUMO merupakan platform simulasi yang bersifat *open source, highly portable*, *microscopic* and *continuous multi-modal* (Eclips, 2025). SUMO telah digunakan dalam berbagai penelitian lintas domain, mulai dari perencanaan tata kota hingga pengujian algoritma kecerdasan buatan untuk pengendalian sinyal lalu lintas. Berikut beberapa contoh studi kasus yang memanfaatkan SUMO.

1. Perencanaan Tata Kota

   Essamlali dkk. (2025) melakukan penelitian *Impact of Urban Block Shape on Traffic and Air Quality,* peneliti memvariasikan bentuk blok (persegi, radial, segitiga) di jaringan kota SUMO, kemudian menghitung emisi menggunakan modul HBEFA. Hasilnya, blok persegi panjang mampu mengurangi CO_2 ± 14 % dibanding grid persegi karena distribusi arus lebih merata.

2. Kendaraan Otonom

   Studi co-simulation I-75 Florida memodelkan evakuasi badai Irma di SUMO, lalu menilai penetrasi 25 % CAV. Hasilnya, potensi tabrakan turun ≈ 42 % berkat interaksi ACC/CACC (Syed dkk., 2023). Pada penelitian yang lain Framework matematis + SUMO merancang penempatan SAV bagi lansia dan kelompok rentan saat evakuasi pedesaan, hasilnya dapat waktu tunggu evakuasi turun 15–20 % (Sevim dkk., 2025).

3. Environment RL untuk TSC

   SUMO sering dijadikan environment RL TSC, karena integrasi API TraCI dapat memberikan data lalu lintas antrean dan waktu tunggu secara *real time*. Selain itu dengan integrasi ini model RL dapat langsung menerapkas *action* yang dipilih dan mengobservasi dampak nya. Dari tahun 2016 sampai dengan tahun 2024 68%

penelitian RL TSC menggunakan SUMO sebagai environtment nya (Michailidis dkk., 2025).

2.1.8 Nilai *Variance*

Variance (varians) merupakan ukuran statistik yang digunakan untuk menggambarkan tingkat penyebaran data terhadap nilai rata-ratanya. *Variance* mengukur seberapa besar variasi atau ketidaksamaan nilai-nilai individual dalam suatu kumpulan data relatif terhadap nilai tengah distribusinya. Menurut Devore (2012), *variance* didefinisikan sebagai rata-rata dari kuadrat selisih antara setiap observasi dengan nilai rata-ratanya, yang menjadikan ukuran ini sangat peka terhadap penyimpangan yang besar. Penggunaan kuadrat pada selisih dimaksudkan untuk menghilangkan efek tanda negatif sekaligus memberikan bobot yang lebih besar pada observasi yang berada jauh dari rata-rata. Secara konseptual, *variance* tidak hanya mencerminkan sebaran data, tetapi juga mengindikasikan tingkat ketidakteraturan (*dispersion energy*) dalam suatu sistem data (Devore, 2012). Untuk suatu himpunan data $x_1, x_2, \dots, x_n$ dengan nilai rata-rata $\bar{x}$, variance populasi didefinisikan sebagai:

$$\boldsymbol{\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \qquad (2.2)$$

Variance memiliki karakteristik utama berupa sensitivitas tinggi terhadap nilai ekstrem (*outlier*). Karena setiap deviasi dikuadratkan, satu observasi yang menyimpang jauh dari rata-rata dapat meningkatkan nilai *variance* secara signifikan. Douglas C. Montgomery dan George C. Runger (2011) menjelaskan bahwa sifat ini menjadikan *variance* sangat efektif untuk mengidentifikasi ketimpangan struktural dalam suatu sistem, khususnya ketika terdapat elemen-elemen yang mengalami beban jauh lebih besar dibandingkan elemen lainnya. Namun demikian, kelemahan utama *variance* terletak pada satuan pengukuran yang berupa kuadrat dari satuan data asli, sehingga interpretasinya secara langsung menjadi kurang intuitif.

Dalam analisis *fairness*, variance berperan penting karena mampu menangkap ketimpangan ekstrem antar elemen sistem, memberikan indikasi kuat mengenai ketidakseimbangan distribusi beban, dan menjadi dasar matematis bagi ukuran dispersi

lain, termasuk standar deviasi. Namun, karena keterbatasan interpretasi satuan, *variance* jarang digunakan sebagai satu-satunya indikator dalam analisis praktis, melainkan dikombinasikan dengan standar deviasi (Devore, 2012; Douglas C. Montgomery & George C. Runger, 2011).

2.1.9 Nilai Standar Deviasi

Standar deviasi merupakan ukuran dispersi yang diperoleh dari akar kuadrat *variance*, sehingga memiliki satuan yang sama dengan data asli. Tujuan utama penggunaan standar deviasi adalah untuk menyediakan ukuran penyebaran yang lebih mudah diinterpretasikan secara praktis. Menurut Devore (2012), standar deviasi memberikan gambaran mengenai penyimpangan tipikal suatu observasi terhadap nilai rata-ratanya, sehingga menjadi ukuran dispersi yang paling umum digunakan dalam analisis statistik terapan. Standar deviasi populasi didefinisikan sebagai:

$$\boldsymbol{\sigma} = \sqrt{\frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{x_i} - \bar{\boldsymbol{x}})^{\mathbf{2}}} \tag{2.3}$$

Karena merupakan akar dari *variance*, standar deviasi mewarisi sifat-sifat matematis *variance*, termasuk sensitivitas terhadap nilai ekstrem, meskipun dampaknya tidak setajam *variance* itu sendiri (Dennis D. Wackerly dkk., 2008).

Standar deviasi menggambarkan besar sebaran rata-rata observasi terhadap nilai tengah distribusi. Nilai standar deviasi yang kecil menunjukkan bahwa data terkonsentrasi di sekitar rata-rata, sedangkan nilai yang besar mengindikasikan variasi yang tinggi. Devore (2012) menekankan bahwa standar deviasi sangat bermanfaat untuk membandingkan tingkat variasi antar sistem atau antar kebijakan karena interpretasinya yang langsung dan intuitif. Keunggulan utama standar deviasi meliputi interpretasi langsung dalam satuan asli data, kemudahan komunikasi hasil analisis kepada praktisi dan pengambil kebijakan, dan representasi konsistensi kinerja suatu sistem. Namun demikian, karena transformasi akar kuadrat, standar deviasi cenderung menghaluskan pengaruh nilai ekstrem, sehingga tidak selalu cukup untuk mendeteksi ketimpangan yang tajam dalam distribusi data.

Variance dan standar deviasi merupakan dua ukuran dispersi yang saling melengkapi. *Variance* memberikan sensitivitas tinggi terhadap ketimpangan ekstrem, sedangkan standar deviasi memberikan kemudahan interpretasi. Dennis D. Wackerly dkk. (Dennis D. Wackerly dkk., 2008) menegaskan bahwa penggunaan kedua ukuran secara simultan memungkinkan analisis dispersi yang lebih komprehensif, karena *variance* menekankan struktur ketimpangan, dan standar deviasi menekankan konsistensi dan interpretasi praktis. Oleh karena itu, dalam analisis sistem kompleks yang melibatkan aspek *fairness*, penggunaan *variance* dan standar deviasi secara bersamaan memberikan gambaran yang lebih lengkap dibandingkan penggunaan salah satunya saja.

2.1.10 Nilai Perubahan Relatif (*Relative Change*)

Perubahan relatif (*relative change*) merupakan cara untuk menyatakan perubahan suatu nilai terhadap nilai acuan (*reference value*) dalam bentuk proporsi. Perubahan ini sering digunakan dalam analisis kuantitatif untuk membandingkan dua kondisi atau dua metode yang menghasilkan nilai metrik tertentu, karena mampu mengekspresikan besarnya perubahan dalam bentuk persentase sehingga lebih mudah diinterpretasikan (Triola, 2012; Walpole dkk., 2017).

Misalkan terdapat suatu metrik M yang diukur pada dua kondisi, yaitu kondisi acuan (*baseline*) dan kondisi pembanding (*new method*). Maka perubahan relatif dapat didefinisikan sebagai:

$$\%\boldsymbol{\Delta M} = \frac{\boldsymbol{M_{\text{new}} - M_{\text{baseline}}}}{\boldsymbol{M_{\text{baseline}}}} \times \mathbf{100}\% \tag{2.4}$$

Berdasarkan persamaan tersebut, nilai $\%\Delta M$bernilai positif jika $M_{\text{new}} > M_{\text{baseline}}$, dan bernilai negatif jika $M_{\text{new}} < M_{\text{baseline}}$. Dengan demikian, tanda dari perubahan relatif memberikan informasi arah perubahan, sedangkan besarannya menunjukkan seberapa signifikan perubahan yang terjadi dibandingkan kondisi *baseline* (Montgomery & Runger, 2003).

Dalam kasus tertentu, sebuah metrik memiliki interpretasi "semakin kecil semakin baik" (*lower is better*), misalnya ukuran error, variasi, waktu, atau biaya. Untuk

metrik ini, evaluasi perbaikan kinerja sering dinyatakan dalam bentuk persentase penurunan relatif (*relative reduction*), yang dirumuskan sebagai berikut:

$$\%\textbf{Penurunan} = \frac{\boldsymbol{M}_{\textbf{baseline}} - \boldsymbol{M}_{\textbf{new}}}{\boldsymbol{M}_{\textbf{baseline}}} \times \mathbf{100}\% \tag{2.4}$$

Persentase penurunan relatif digunakan untuk menunjukkan tingkat perbaikan dari suatu metode atau kondisi baru dibandingkan baseline, di mana nilai persentase yang lebih besar mengindikasikan penurunan metrik yang lebih signifikan (Triola, 2012).

2.2 PENELITIAN TERKAIT

2.2.1 Implementasi RL dalam *Traffic Signal Control*

Dalam upaya mengatasi kompleksitas pengendalian sinyal lalu lintas, berbagai studi telah mengadopsi pendekatan *reinforcement learning*, khususnya *Deep Q-Network* (DQN), sebagai solusi adaptif untuk mengoptimalkan kinerja lalu lintas. Salah satu studi komprehensif dilakukan oleh Ouyang dkk. (2024), membandingkan implementasi *Q-learning* dan *Deep Q-Network* dalam skenario persimpangan jalan tol kota. Hasil penelitian menunjukkan bahwa algoritma *Deep Q-Network* secara signifikan meningkatkan performa lalu lintas dibandingkan dengan metode tanpa kontrol maupun *Q-learning* tradisional. Model *Deep Q-Network* yang diusulkan mampu meningkatkan kecepatan rata-rata kendaraan hingga 10% dan menurunkan tingkat okupansi lajur hingga 30%. Studi ini menyoroti kemampuan RL untuk mengelola lingkungan lalu lintas yang dinamis, meskipun tidak menekankan pada aspek *fairness* antar kaki simpang (Ouyang dkk., 2024).

Studi lain yang relevan adalah dari Chang dkk. (2024), yang mengembangkan model *multi-agent* RL bernama CVDMARL pengembangan dari *Deep Q-Network* untuk mengatur sinyal lalu lintas di lingkungan perkotaan berskala besar. Pendekatan ini memanfaatkan komunikasi eksplisit dan implisit antar agen dengan teknik *Centralized Training Decentralized Execution* (CTDE) untuk meningkatkan efisiensi pengambilan keputusan. Hasil eksperimen menunjukkan bahwa model ini berhasil mengurangi panjang antrean kendaraan hingga 9.12% dan waktu tunggu hingga 7.67% dibandingkan

metode baseline terbaik. Dalam penelitian ini pengembangan RL tidak mengimplementasikan aspek *fairness* melainkan berfokus pada aspek penurunan tekanan antrean.

Selanjutnya, studi oleh Khozam dan Farhi (2023) memperluas cakupan implementasi *Deep Q-Network* ke dalam domain lalu lintas angkutan massal dengan mengembangkan model *Double DQN* (DDQN) untuk mengoptimalkan berbagai aspek operasional yaitu waktu kedatangan, waktu henti di peron, kecepatan maksimum, dan arus masuk penumpang. Model ini berhasil menjaga performa sistem transportasi pada tingkat permintaan yang sangat tinggi. Meskipun fokus utama bukan pada sinyal lalu lintas persimpangan konvensional, pendekatan ini membuktikan efektivitas RL dalam menangani sistem lalu lintas kompleks dan padat secara adaptif dan terintegrasi. Namun demikian, dua studi sebelumnya, aspek *fairness* antar pengguna jalan belum menjadi perhatian utama dalam desain *reward function* mereka (Khozam & Farhi, 2023).

Ketiga studi ini menunjukkan bahwa RL, khususnya DQN, memberikan kontribusi nyata dalam meningkatkan efisiensi lalu lintas. Namun, isu *fairness* belum sepenuhnya diintegrasikan ke dalam desain algoritma saat ini, meskipun mulai diakui sebagai dimensi penting dalam pengembangan sistem kendali lalu lintas yang lebih berkelanjutan.

2.2.2 *Fairness-Aware RL* dalam *Traffic Signal Control*

Seiring meningkatnya kompleksitas sistem lalu lintas perkotaan, pendekatan berbasis *reinforcement learning* mulai dikembangkan tidak hanya untuk mengoptimalkan efisiensi lalu lintas, tetapi juga untuk mengatasi tantangan *fairness* dalam distribusi waktu tunggu. Dalam konteks ini, sejumlah penelitian telah menganggap isu *fairness* sebagai dimensi penting dalam desain sistem pengendalian sinyal lalu lintas.

Penelitian oleh Luo dan Jin (2022) memperkenalkan arsitektur *hybrid action* space bernama *Traffic signal control with hybrid Proximal Policy Optimization* (TBO). Model ini menggabungkan pengaturan tahap (*staging*) dan durasi fase secara sinkron, dengan evaluasi berbasis *Gini coefficient* untuk menilai pemerataan distribusi hak lintas jalan. Hasil penelitian menunjukkan bahwa TBO mampu mempertahankan efisiensi kontrol lalu lintas tanpa mengorbankan aspek *fairness*, meskipun pengukuran *fairness*

tersebut tidak dimunculkan secara eksplisit dalam publikasi hanya disampaikan pada kesimpulan akhir. Disisi lain secara siklus lalu lintas model ini menerapkan pemilihan fase hijau secara acak yang tidak dapat memberikan kepastian giliran (Luo & Jin, 2022).

Sementara itu, C. Li dkk. (2020) memodelkan RL dengan menerapkan fairness dalam fungsi objektif dengan mengusulkan algoritma baru bernama *Fairness Scheduling Proximal Policy Optimization* (FSPPO). Tujuan utama dari FSPPO adalah meminimalkan waktu tunggu maksimum antar kendaraan (dikaki simpang yang sama) sebagai representasi *fairness*, sekaligus mempertahankan performa rata-rata sistem. Hasil eksperimen dalam lingkungan simulasi SUMO menunjukkan bahwa model yang fokus pada *fairness* mampu mengurangi waktu tunggu rata-rata lebih baik. Meskipun demikian, sistem ini mengandalkan tekanan antrean dalam pemilihan fase hijau, sehingga siklus lalu lintas bersifat acak dan tidak terprediksi (C. Li dkk., 2020).

Kedua studi ini menegaskan pentingnya mempertimbangkan *fairness* dalam desain sistem pengendalian sinyal lalu lintas berbasis RL. Selanjutnya, belum adanya penerapan yang mempertahankan fase siklus tetap dalam arsitektur tersebut mengindikasikan adanya ruang penelitian lanjutan untuk menggabungkan *fairness* RL dengan mempertahankan fase siklus tetap yang dapat memberikan *fairness* rata-rata waktu tunggu sekaligus kepastian giliran setiap kaki simpang.

2.2.3 Tabel Perbandingan

Dari beberapa penelitian tersebut kita dapat membandingkan penelitian saat ini dengan penelitian sebelumnya yang relevan. Perbandingan tersebut dapat dilihat pada Tabel 2.1 berikut.

Tabel 2.4 Perbandingan penelitian terkait

Judul	**Peneliti**	**Tahun**	**Algoritma**	**Environment**	**Tipe Intersection**	**Fairness-Aware**	**Siklus**	**Tujuan Utama**
A Comparative Study of Traffic Signal Control Based on RL Algorithms	Ouyang dkk.	2024	DQN	SUMO	Single	Tidak	Acak	Efisiensi lalu lintas (kecepatan, okupansi lajur)
CVDMARL: *A Communication-Enhanced Multi-Agent RL Traffic Signal Control Method*	Chang dkk.	2024	DQN (Multi-agent, CTDE)	SUMO	Multi	Tidak	Acak	Reduksi antrean dan waktu tunggu
Deep Reinforcement Q-Learning for Intelligent Traffic Control in Mass Transit	Khozam dan Farhi	2023	DDQN	Metro Line Simulator	Single (Transit)	Tidak	Acak	Efisiensi operasional transportasi massal
Reinforcement Learning for Traffic Signal Control in Hybrid Action Space	Luo dan Jin	2022	Hybrid PPO	SUMO	Multi	Ya (*Gini coefficient*)	Acak	Efisiensi dan distribusi *fairness*

Judul	**Peneliti**	**Tahun**	**Algoritma**	**Environment**	**Tipe Intersection**	**Fairness-Aware**	**Siklus**	**Tujuan Utama**
Fairness Control of Traffic Light via Deep Reinforcement Learning	C. Li dkk.	2020	FSPPO	SUMO	Single	Ya (antar pengendara)	Acak	Optimasi *fairness* dan waktu tunggu maksimum
Penelitian Ini (Usulan)	Muhammad Wahyudi	2025	DQN	SUMO	Single	Ya (antar kaki simpang)	Tetap/Teratur	Menyeimbangkan efisiensi dan *fairness* dalam siklus tetap