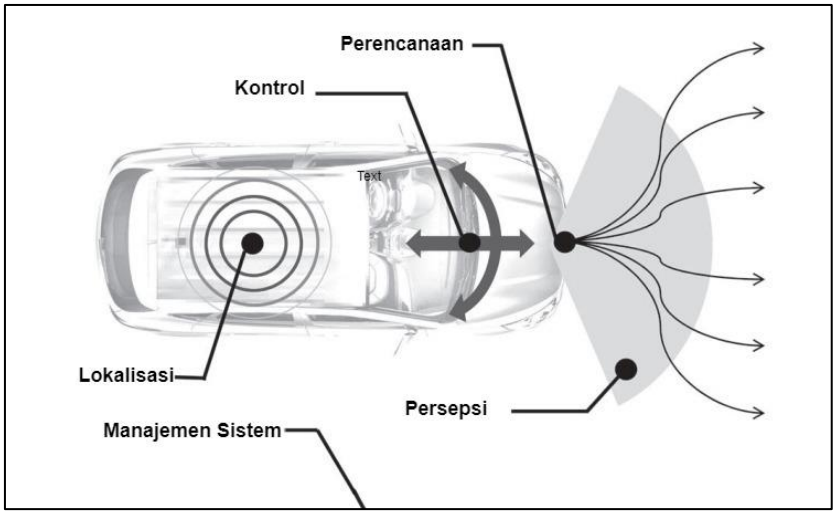
BAB 2
TINJAUAN PUSTAKA

2.1 Mobil Balap Otonom

Mobil otonom adalah kendaraan yang memiliki kemampuan untuk bernavigasi dengan mendeteksi atau mengobservasi lingkungan sekitarnya tanpa input atau operasi manual manusia. Mobil otonom belajar dan memahami lingkungannya sendiri serta membuat keputusan mengemudi seperti menghindari rintangan dan merencanakan rute [1]. Tidak hanya diterapkan pada mobil biasa, sistem otonom juga dapat diterapkan pada mobil balap yang dapat menjelajahi lintasan balap dan membuat keputusan tentang bagaimana menghindari rintangan dan pesaing lainnya secara *real-time*. Pada kebanyakan kasus, terutama untuk pengujian, lingkungan simulasi digunakan. Mobil balap otonom sering kali digunakan pada kompetisi balap robotik dan juga pada *video game* bergenre balap [2] [5].

Dalam pengaplikasianya di dunia nyata, terdapat 5 fungsi utama pada yang ada pada mobil otonom, diantaranya, persepsi, lokalisasi, perencanaan, kontrol, dan manajemen sistem [13] yang dapat dilihat pada Gambar 2.1 Fungsi Utama Mobil Otonom.



Gambar 2.1 Fungsi Utama Mobil Otonom.

Persepsi adalah proses yang mendeteksi lingkungan sekitar mobil otonom menggunakan berbagai teknik observasi seperti menggunakan berbagai macam sensor, contohnya RADAR, LIDAR, *Ultra Sonic* ataupun menggunakan Kamera yaitu *Computer Vision*. Lokalisasi berfungsi menemukan posisi mobil otonom dengan menggunakan teknik *Global Positioning System* (GPS), dan peta jalan raya. Fungsi perencanaan, menentukan perilaku dan gerakan mobil otonom berdasarkan informasi dari persepsi dan lokalisasi. Fungsi kontrol kemudian mengikuti perintah yang diinginkan dari fungsi perencanaan dengan mengendalikan kemudi, akselerasi, dan pengereman mobil otonom. Terakhir, manajemen sistem mengawasi keseluruhan sistem mengemudi otonom. Contoh fungsi dari manajemen sistem ini meliputi sistem manajemen kesalahan, sistem pencatatan [13].

2.2 Lingkungan Dinamis

Lingkungan dinamis dalam sistem otonom mengacu pada lingkungan di mana kondisi atau keadaannya dapat berubah secara tidak terduga atau dinamis seiring waktu. Agen yang melakukan proses *training* harus mampu menyesuaikan strateginya berdasarkan informasi baru yang diperoleh dari lingkungan tersebut. Perubahan dalam lingkungan dapat termasuk perubahan dalam transformasi sebuah objek, aturan permainan, munculnya atau menghilangnya rintangan, atau perubahan dalam kondisi fisik lingkungan [14] . Oleh karena itu, agen harus dapat mengambil keputusan yang tepat dalam menghadapi situasi yang berubah-ubah ini. Menurut studi pada penelitian [15] tantangan utama dalam lingkungan dinamis meliputi:

1. Penginderaan (*Sensing*)

   Sistem otonom harus memiliki sensor yang mampu mendeteksi perubahan lingkungan dengan akurat dan cepat.

2. Perencanaan (*Planning*)

   Algoritma perencanaan harus fleksibel dan cepat untuk menyesuaikan dengan perubahan yang terdeteksi.

3. Pengambilan Keputusan (*Decision Making*)

   Sistem harus mampu membuat keputusan yang optimal meskipun informasi yang tersedia tidak lengkap atau tidak pasti.

2.3 *Dynamic Obstacle avoidance* (DOA)

Dynamic Obstacle avoidance adalah kemampuan kendaraan untuk secara aktif menghindari atau mengatasi hambatan atau rintangan yang muncul secara tiba-tiba atau bergerak disepanjang jalur perjalanan. Hal ini mencakup kemampuan untuk mendeteksi, memprediksi, dan merespons terhadap perubahan lingkungan secara *real-time* untuk menjaga keamanan dan kelancaran perjalanan kendaraan. Teknik yang umum digunakan untuk mencapai *Dynamic Obstacle avoidance* termasuk penggunaan sensor-sensor seperti lidar, radar, kamera, dan pengolahan data yang cepat untuk menghasilkan keputusan navigasi yang optimal dalam waktu singkat. Dengan adanya *Dynamic Obstacle avoidance*, kendaraan otonom dan semi-otonom dapat beroperasi dengan lebih aman dan efisien di lingkungan yang kompleks dan dinamis.

2.4 *Artificial Intelligence* (AI)

Artificial Intelligence (AI) atau kecerdasan buatan merujuk pada kemampuan sistem komputer untuk melakukan tugas-tugas yang biasanya membutuhkan kecerdasan manusia. Ini mencakup berbagai disiplin ilmu seperti pembelajaran mesin (ML), pemrosesan bahasa alami (*natural language processing*), pengenalan pola (pattern recognition), dan sistem pakar (*expert systems*). Menurut Russell dan Norvig AI adalah studi tentang agen cerdas, yaitu sistem yang dapat mempersepsikan lingkungan mereka dan mengambil tindakan yang memaksimalkan peluang mereka untuk mencapai tujuan tertentu. Pembelajaran mesin memungkinkan sistem belajar dari data dan meningkatkan kinerjanya seiring waktu tanpa pemrograman eksplisit [16]. Pembelajaran mendalam (*Deep learning*), sebagai subset dari pembelajaran mesin, menggunakan jaringan saraf tiruan dengan banyak lapisan untuk memodelkan dan memahami data yang sangat kompleks [17].

AI memainkan peran penting dalam memungkinkan sistem otonom untuk beroperasi dalam lingkungan yang dinamis dan kompleks. Misalnya, dalam navigasi dan penghindaran hambatan, teknik seperti *reinforcement learning* dan *vision-based navigation* memungkinkan kendaraan otonom atau robot bergerak dengan aman di lingkungan yang tidak terstruktur [18]. Dalam pengambilan keputusan *real-time*, sistem otonom menggunakan algoritma pembelajaran mesin

untuk membuat keputusan cepat berdasarkan data sensor yang diterima [19]. Selain itu, jaringan saraf konvolusional atau *Convolutional Neural networks* (CNN) digunakan untuk pengenalan dan klasifikasi objek dalam gambar atau video, yang sangat penting untuk navigasi dan interaksi dengan lingkungan. Studi kasus dalam berbagai bidang menunjukkan penerapan AI yang luas. Kendaraan otonom menggunakan teknik AI seperti pembelajaran mendalam untuk pengenalan objek dan *reinforcement learning* untuk perencanaan jalur dan penghindaran penghalang [15] dalam robotika industri, robot menggunakan AI untuk meningkatkan efisiensi dan fleksibilitas dalam proses manufaktur, misalnya melalui sistem visi komputer untuk inspeksi kualitas otomatis [20].

2.5 Pembelajaran Mesin

Pembelajaran mesin adalah cabang dari kecerdasan buatan yang berfokus pada pengembangan sistem komputer yang dapat belajar dari data dan pengalaman untuk meningkatkan kinerjanya seiring waktu tanpa perlu pemrograman eksplisit. Tujuan utama dari pembelajaran mesin adalah untuk memungkinkan komputer untuk mengidentifikasi pola-pola yang tersembunyi dalam data dan menggunakan pola-pola ini untuk membuat keputusan atau prediksi yang berguna. Pembelajaran mesin dibagi menjadi beberapa bagian diantaranya *supervised learning*, *unsupervised learning*, dan *reinforcement learning*.

Dalam pembelajaran mesin, tujuan utama adalah membuat model yang dapat mempelajari pola-pola yang tersembunyi dalam data dan membuat prediksi yang akurat atau mengambil keputusan yang tepat berdasarkan pola-pola ini. Proses pembelajaran ini sering kali melibatkan pengoptimalan model melalui iterasi berulang, di mana model diperbarui berdasarkan evaluasi kinerja terhadap data pelatihan. Salah satu komponen kunci dari proses ini adalah fungsi kerugian (*loss function*), yang mengukur seberapa baik model saat ini memprediksi *output* yang benar berdasarkan data pelatihan yang diberikan [21].

2.6 *Reinforcement learning*

Reinforcement learning (RL) adalah salah satu sub kategori dari pembelajaran mesin di mana agen belajar untuk mengambil tindakan dalam lingkungan tertentu berdasarkan informasi yang diberikan. dengan tujuan untuk memaksimalkan sebuah nilai yang di sebut *reward*. Dalam RL Imbalan adalah variabel bilangan bulat atau skalar yang terkait dengan beberapa *state* bagian atau pasangan *state* dan *action*. Dalam prosesnya, agen menerima informasi berupa data dari lingkungan pada satuan waktu (*step*) yang disebut *state*. Data pada *state* bisa berbentuk data citra data pada sensor, koordinat suatu objek, atau apapun data yang ada pada lingkungan dimana agen dilatih. Data pada *state* selanjutnya akan digunakan untuk agen dalam mempertimbangkan pemilihan *action* dan setiap agen memilih sebuah *action* maka itu akan merubah pula *state* pada lingkungan dan juga dilakukan proses pemberian *reward* atau dapat disebut *reward* function yang merupakan umpan balik (*feedback*) objektif dari lingkungan [22].

Tujuan utama dari agen dalam *reinforcement learning* adalah untuk mempelajari kebijakan (*policy*) yang optimal, yaitu serangkaian aturan yang menentukan tindakan apa yang harus diambil dalam setiap keadaan untuk memaksimalkan jumlah hadiah yang diperoleh dari lingkungan. Proses belajar di RL terjadi melalui *trial and error*, dimana agen melakukan tindakan, mengamati hasilnya, dan menggunakan umpan balik ini untuk memperbarui strateginya seiring waktu. Salah satu konsep kunci dalam RL adalah *reward signal*, yang memberikan umpan balik positif atau negatif kepada agen berdasarkan tindakan yang diambilnya. Agar agen dapat belajar dengan baik, *reward signal* harus dirancang sedemikian rupa sehingga mencerminkan tujuan yang diinginkan dari sistem tersebut. Dengan memanfaatkan *reward signal* ini, agen dapat menggunakan teknik-teknik seperti *Q-learning* atau metode Monte Carlo untuk belajar kebijakan yang optimal.

terdapat dua proses pada *reinforcement learning* yaitu Eksplorasi dan Eksploitasi. Eksplorasi adalah proses di mana agen mencoba tindakan yang belum dieksplorasi dengan harapan memperoleh informasi baru atau memperbaiki perkiraannya tentang lingkungannya. Eksplorasi diperlukan untuk menghindari

jebakan lokal dan untuk mendapatkan pemahaman yang lebih baik tentang lingkungan secara keseluruhan. dalam hal ini dapat di simpulkan bahwa eksplorasi adalah proses *training* agen pada lingkunganya. Sedangkan Eksploitasi adalah proses di mana agen memilih tindakan yang diyakini akan memberikan hasil terbaik berdasarkan pengetahuan saat ini [23] . Eksploitasi diperlukan untuk memanfaatkan informasi yang sudah diketahui untuk memaksimalkan keuntungan atau mencapai tujuan segera. yang dapat disimpulkan bahwa eksploitasi adalah proses mensimulasikan kemampuan agen setelah dilatih .

2.6.1 *Cumulative Reward*

Cumulative reward adalah konsep dalam RL yang merujuk pada jumlah total *reward* yang diperoleh agen selama periode waktu tertentu atau sepanjang episode tertentu dari interaksi dengan lingkungan. Agen dalam RL berusaha memaksimalkan *cumulative reward* ini dengan memilih tindakan optimal berdasarkan kebijakan (*policy*) yang dipelajari.

Cumulative reward dihitung sebagai jumlah dari semua *reward* yang diterima dari setiap tindakan yang diambil oleh agen sampai titik waktu tertentu, dapat di gambarkan dengan persamaan berikut :

$$R = \frac{1}{T}\sum_{t=0}^{t-1} rt \tag{2.1}$$

di mana R adalah *cumulative reward* yang dimulai dari waktu t = 0 ke t - 1, dan rt adalah *reward* yang diterima pada waktu t dan T adalah total *step*.

Pengukuran *cumulative reward* biasanya tergantung pada konteks dan tujuan spesifik dari tugas yang sedang dikerjakan sehingga tidak ada standar universal pada pengukuranya. *Cumulative reward* digunakan karena metode ini secara langsung terkait dengan tujuan utama agen dalam RL, yaitu memaksimalkan *reward* yang diterima selama interaksi dengan lingkungan. Pengukuran *cumulative reward* memberikan cara yang jelas dan kuantitatif untuk menilai seberapa baik agen mencapai tujuan. Namun, meskipun begitu beberapa penelitian terkait dengan *reinforcement learning* menggunakan metode ini untuk mengukur peformansi metriknya [1] [2] [12]. sehingga untuk pengujianya biasa dilakukan dengan standarisasi dengan penelitian yang melakukan penelitian dan pengukuran terkait,

caranya dengan menggunakan parameter yang sama dengan penelitian yang akan dibandingkan seperti *leaning rate*, *discount factor*, dan lain-lain agar konsisten dengan penelitian yang dibandingkan.

2.7 Q learning

Q-learning adalah salah satu teknik pembelajaran penguatan (*reinforcement learning*) yang digunakan untuk mempelajari kebijakan (*policy*) optimal untuk pengambilan keputusan di lingkungan yang dinamis. Tujuan dari *Q-learning* adalah untuk mempelajari fungsi nilai aksi (*action-value* function) yang dikenal sebagai Q-function, yang mengestimasi nilai dari setiap pasangan keadaan-tindakan (*state-action pair*) dalam lingkungan. Dalam *Q-learning*, agen secara iteratif memperbarui estimasi *Q-value* berdasarkan pengalaman yang diperoleh dari berinteraksi dengan lingkungan. Algoritma ini mengikuti prinsip *Bellman equation*, di mana nilai Q dari sebuah pasangan keadaan-tindakan didefinisikan sebagai imbalan (*reward*) yang segera diterima ditambah dengan nilai maksimal dari Q pada keadaan berikutnya yang dihasilkan dari tindakan tersebut.

Selama proses belajar, agen melakukan eksplorasi dan eksploitasi. Eksplorasi dilakukan untuk mengeksplorasi lingkungan dan mencari tindakan baru yang dapat meningkatkan pemahaman agen tentang lingkungan. Eksploitasi dilakukan untuk memanfaatkan pengetahuan yang sudah ada dan mengambil tindakan yang dianggap optimal berdasarkan nilai Q yang sudah dipelajari.

Q-learning telah diterapkan dalam berbagai aplikasi, termasuk pengendalian robot, permainan video, pengelolaan jaringan, dan sebagainya. Meskipun *Q-learning* memiliki keuntungan dalam kemampuannya untuk menangani lingkungan yang kompleks dan dinamis, namun proses pembelajarannya seringkali membutuhkan waktu yang lama dan memerlukan penyesuaian parameter yang cermat untuk mencapai hasil yang diinginkan.

Pada *Q-learning reward* (imbalan) merupakan sinyal umpan balik yang diberikan kepada agen sebagai hasil dari tindakan yang diambilnya dalam lingkungan. *Reward* ini memberi tahu agen seberapa baik atau buruk tindakan

tersebut, sehingga agen dapat belajar untuk memilih tindakan yang lebih baik di masa depan.

2.8 *Deep learning*

Deep learning adalah salah satu sub-bidang dari pembelajaran mesin yang berfokus pada penggunaan arsitektur jaringan saraf tiruan yang dalam (*Deep Neural networks*) untuk mempelajari representasi hierarkis dari data. *Deep learning* mencoba untuk meniru struktur dan fungsi otak manusia dengan menggunakan jaringan saraf tiruan yang terdiri dari banyak lapisan (*layer*) yang saling terhubung [17]. Arsitektur *Deep learning* biasanya terdiri dari beberapa lapisan yang berturut-turut mengolah data dari lapisan input hingga menghasilkan *output* yang diinginkan. Setiap lapisan dalam jaringan ini mempelajari representasi yang semakin abstrak dari data, yang memungkinkan jaringan untuk menangkap pola-pola yang kompleks dan tersembunyi dalam data. *Deep learning* memiliki beberapa proses perhitungan seperti perhitungan bobot dengan persamaanya

$$z\_in_i = \Sigma x_j \cdot v_{j,i} + bv_i \tag{2.2}$$

Deep learning telah menjadi sangat populer dalam beberapa tahun terakhir karena kemampuannya untuk mencapai hasil yang sangat baik dalam berbagai tugas, termasuk pengenalan gambar, pengenalan suara, pemrosesan bahasa alami, dan lain-lain. Keberhasilan *Deep learning* sebagian besar didorong oleh kemajuan dalam perangkat keras yang lebih kuat dan ketersediaan data yang besar, yang memungkinkan jaringan saraf yang lebih dalam dan kompleks untuk dilatih dengan efektif. Meskipun *Deep learning* telah mencapai kesuksesan besar dalam banyak aplikasi, namun ada beberapa tantangan yang terkait dengan penggunaannya, termasuk kebutuhan akan data yang besar, interpretabilitas model yang rendah, dan kecenderungan terhadap *overfitting* pada *dataset* kecil. Namun demikian, dengan penelitian dan pengembangan yang terus berlanjut, *Deep learning* diharapkan dapat terus memberikan kontribusi besar dalam memecahkan berbagai masalah dalam bidang-bidang seperti ilmu komputer, kedokteran, keuangan, dan banyak lagi.

2.8.1 *Gradient Descent*

Gradient Descent adalah salah satu algoritma optimisasi yang paling umum digunakan dalam pembelajaran mesin dan *Deep learning*. Tujuannya adalah untuk menemukan nilai minimum dari suatu fungsi yang bisa menjadi fungsi kerugian (*loss function*) atau fungsi biaya (*cost function*) dengan mengiterasikan parameter model ke arah yang menurunkan nilai fungsi tersebut.

Konsep utama di balik *Gradient Descent* adalah menggunakan gradien atau turunan parsial dari fungsi tersebut untuk menentukan arah dan besarnya perubahan yang harus dilakukan pada parameter model agar nilai fungsi tersebut berkurang. Dalam konteks pembelajaran mesin, ini berarti menemukan nilai parameter model yang menghasilkan prediksi yang paling dekat dengan data yang sesungguhnya [9].

2.8.2 Fungsi Aktivasi

Fungsi aktivasi adalah fungsi matematis yang diterapkan pada keluaran setiap neuron dalam jaringan saraf tiruan (*neural network*) atau dalam algoritma pembelajaran mesin lainnya. Tujuan dari fungsi aktivasi adalah untuk memperkenalkan non-linear ke dalam model, yang memungkinkan jaringan saraf untuk mempelajari pola-pola yang kompleks dan non-linear dalam data.

Fungsi aktivasi yang akan digunakan pada penelitian ini diantaranya

1. ReLU (*Rectified Linear Unit*) : ReLU adalah salah satu fungsi aktivasi yang paling umum digunakan. Rumusnya sederhana:

$$F(x) = \max(0, x) \tag{2.3}$$

Fungsi ini menghasilkan keluaran nol jika inputnya negatif, dan input itu sendiri jika positif.

2. Sigmoid : Sigmoid adalah fungsi yang menghasilkan keluaran dalam rentang (0, 1). Persamaanya adalah sebagai berikut :

$$F(x) = \frac{1}{1+e^{x}} \tag{2.4}$$

Fungsi sigmoid sering digunakan pada lapisan tersembunyi (*hidden layer*s) dari jaringan saraf, atau sebagai fungsi aktivasi untuk *output layer* dalam model klasifikasi biner.

Pemilihan fungsi aktivasi yang tepat sangat penting dalam desain jaringan saraf, karena dapat mempengaruhi kinerja dan kecepatan konvergensi model. Dengan menggunakan fungsi aktivasi yang sesuai, jaringan saraf dapat belajar dengan lebih efisien dan menghasilkan hasil yang lebih baik [24].

2.9 *Deep Reinforcement learning*

Deep Reinforcement learning (DRL) adalah perpaduan antara *Deep learning* dan *Reinforcement learning* yang menggabungkan keunggulan dari kedua paradigma tersebut untuk mempelajari tindakan yang optimal dalam lingkungan yang kompleks. DRL menggunakan jaringan saraf tiruan yang dalam (*Deep Neural networks*) untuk mempelajari kebijakan (*policy*) atau fungsi nilai aksi (*action-value* function) yang optimal dari pengalaman yang diperoleh dari berinteraksi dengan lingkungan [11] .

Pada dasarnya, DRL menggunakan struktur jaringan saraf yang dalam untuk merepresentasikan dan memproses informasi dari lingkungan yang kompleks, dan menggunakan algoritma *reinforcement learning* untuk mengatur cara bagaimana agen memperbarui kebijakan atau nilai aksi berdasarkan pengalaman yang diperoleh. Dengan memanfaatkan kekuatan dari kedua paradigma ini, DRL dapat mempelajari tindakan yang optimal dalam berbagai tugas yang sulit dan kompleks.

2.10 Deep Q-Network (DQN)

Deep Q-Network adalah sebuah algoritma dalam bidang *Deep Reinforcement learning* yang digunakan untuk mempelajari kebijakan (*policy*) optimal untuk pengambilan keputusan dalam lingkungan yang dinamis dan kompleks. DQN merupakan pengembangan dari algoritma *Q-learning* klasik dengan memanfaatkan jaringan saraf tiruan yang dalam (*Deep Neural networks*) untuk memodelkan fungsi nilai aksi (*action-value function*) yang kompleks. Algoritma DQN telah dikembangkan pada penelitian [25] untuk memainkan *game* atari. Yang saat ini dapat pula diimplementasikan pada berbagai macam kasus.

2.10.1 Algoritma DQN

Berikut adalah algoritma DQN yang dapat dilihat pada Gambar 2.1 Algoritma DQN

Algoritma DQN

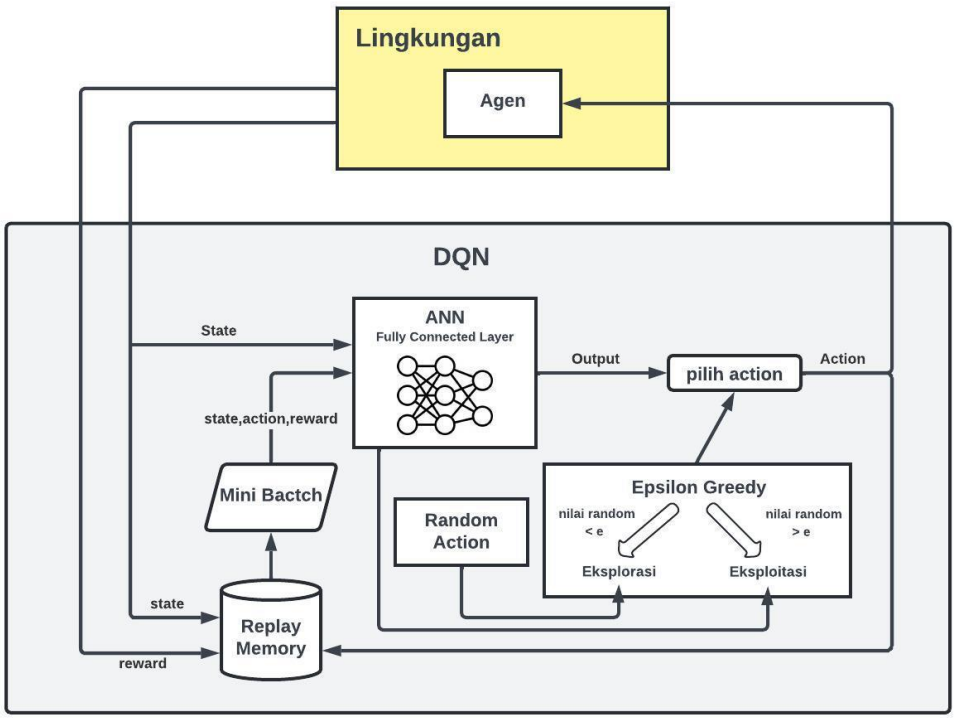
```
inisialisasi replay memory (M)
inisialisasi neural network (ANN) dengan bobot random untuk
menghasilkan Q Value
while (!selesai)
    if pada random
        aksi = aksi random
    else
        state = lingkungan/agen state
        aksi = max(ANN.KalkulasiLuaran(state))
if aksi gagal
    reward -1
else
    reward +1
Simpan(state,aksi,reward) pada M
secara periodic ambil batch memori pada dari M
    while bukan pada akhir batch
        QualityValueLama = max(ANN.KalkulasiLuaran(M.state))
        QualityValueSelanjutnya = max(ANN.KalkulasiLuaran(M + 1.state))
        umpanbalik = M.reward + diskon * QualityValueSelanjutnya
        ANN.Latih(state, umpanbalik)
```

Gambar 2.2 Algoritma DQN

Pada dasarnya, DQN menggunakan jaringan saraf tiruan atau ANN untuk mengaproksimasi fungsi nilai aksi, yang memungkinkan agen untuk mengambil tindakan optimal berdasarkan keadaan saat ini dari lingkungan. DQN dilatih dengan menggunakan algoritma *Q-learning* dan menggunakan pengalaman yang diperoleh dari berinteraksi dengan lingkungan untuk memperbarui estimasi nilai aksi.

2.10.2 Arsitektur DQN

Untuk lebih jelasnya arsitektur DQN dapat dilihat pada Gambar 2.2 Arsitektur DQN



Gambar 2.3 Arsitektur DQN

Salah satu fitur utama dari DQN adalah penggunaan pengalaman berulang (*Replay Memory*) yang memungkinkan agen untuk menyimpan dan mengulangi pengalaman yang telah diperoleh selama proses pelatihan. Hal ini membantu dalam mengurangi korelasi antara pengalaman yang berurutan dan memungkinkan agen untuk lebih efisien mempelajari pola-pola dalam data.

jika pada algoritma *Q learning* penentuan maksimum *reward* dari setiap *state* dan *action* digunakan Q tabel namun pada *Deep Q-Network* digunakan *Neural network* sebagai penggantinya [1] dengan menggunakan persamaan belmans sebagai berikut :

$$Q(s,a) = Q(s,a)+\alpha[r+\gamma \max a'Q(s',a') - Q(s,a)] \quad (2.5)$$

Dimana :

1. Q (s,a): Nilai Q (*action-value*) untuk pasangan keadaan, aksi, yaitu keadaan s dan aksi a.

2. r : Imbalan segera yang diperoleh dari mengambil tindakan a saat berada dalam keadaan s.
3. α: Tingkat pembelajaran (*leaning rate*), yang mengontrol laju di mana nilai Q diperbarui.
4. γ : Faktor diskon (*discount factor*), yang menentukan pentingnya imbalan di masa depan.
5. max$a'Q(s',a')$: Nilai Q maksimum dari semua kemungkinan aksi a' di keadaan berikutnya, s'.

untuk estimasi nilai Q. Dengan dilengkapi *Deep Neural network* (DNN) , *Q-learning* menunjukkan keunggulannya pada aplikasi-aplikasi yang berhasil di Atari *Game*s [25] , dengan versi dalamnya yang disebut *Deep Q-Network* (DQN) [26]. Pada DQN.

2.10.3 *Replay Memory*

Pada dasarnya, gagasan di balik konsep dari *Replay Memory* adalah menyimpan pengalaman berisi *state*, *action* yang diambil dan *reward* yang diterima dalam satu daftar panjang. Daftar ini disebut *Replay Memory*, karena berisi 'memori' semua pengalaman yang dimiliki agen. data dari *Replay Memory* digunakan untuk pelatihan dengan mengambil sampel pengalaman dari memori secara acak.

Alasan dibuatnya *Replay Memory* adalah untuk menghilangkan korelasi yang muncul saat kami melatih pengalaman yang terjadi satu demi satu. Dengan melatih secara acak pada situasi yang berbeda pada waktu yang berbeda membuat agen dapat mempelajari secara lebih objektif tindakan apa yang sebenarnya berkorelasi dengan *Q-value* yang tinggi. berikut adalah perhitungan pada proses penambahan data *Replay Memory*

$$e_t = (s_t, a_t, r_{t+1}) \tag{2.6}$$

Kemudia setiap satuan *Replay Memory* ditambahkan pada data dengan persamaan berikut

$$D_t = \{e_1, \dots, e_t\} \tag{2.7}$$

2.10.4 *Epsilon Greedy*

Tujuan agen adalah mencoba mengumpulkan *reward* maksimum di masa depan yang juga dikenal sebagai *Q-value* untuk menyelesaikan suatu tugas. Sangat penting untuk menemukan keseimbangan antara terus mengeksplorasi dan mengeksploitasi nilai-nilai Q yang diketahui secara maksimal. Solusi yang baik untuk masalah ini adalah dengan menerapkan kebijakan ε-*greedy* dimana ε adalah probabilitas antara 0 (Eksploitasi) hingga 1 (Eksplorasi). Agen diperbolehkan melakukan tindakan acak dengan probabilitas ε alih-alih selalu memilih tindakan 'serakah' (tindakan dengan nilai Q terbesar). Merupakan praktik yang baik untuk menetapkan ε pada 1 di awal sehingga terdapat peluang 100% bahwa agen akan bertindak secara acak. Ketika agen terus melakukan eksplorasi, ε secara bertahap terurai menjadi semakin eksploitatif. Peluruhan ini dapat dilakukan dengan menggunakan laju peluruhan, dimana ε dikurangi dengan laju peluruhan pada setiap langkah waktu [1].

2.11 *Unity Engine*

Unity Engine adalah sebuah platform pengembangan permainan (*game development*) yang sangat populer yang digunakan oleh para pengembang untuk membuat permainan video dan aplikasi interaktif lainnya. Ini adalah salah satu platform pengembangan permainan terkemuka di dunia dan dikenal karena kesederhanaan penggunaannya, fleksibilitas, dan dukungan lintas-platform yang kuat. *Unity Engine* menyediakan beberapa *tools* yang berguna untuk penelitian ini seperti physic sistem untuk mobilm, pembuatan 3d *Environtment* dan yang lain, *Unity Engine* juga telah banyak dipakai untuk melatih berbagai *agent Deep reinforcement learning* untuk simulasinya. *Unity Engine* juga memiliki *tools* yang akan diperlukan pada penelitian ini yaitu Raycats sistem yang mana dapat mendeteksi jarak antara objek satu dengan yang lainnya.

2.12 Bahasa Pemrograman C#

Pemrograman C# adalah keterampilan yang sangat berharga bagi para pengembang perangkat lunak. Bahasa ini, dikembangkan oleh Microsoft, menawarkan sintaksis yang bersih dan jelas, membuatnya mudah dipahami oleh para pemula sekalipun. Selain itu, C# adalah bahasa berorientasi objek, yang berarti

ia mendukung konsep seperti kelas, objek, dan pewarisan, memungkinkan pengembang untuk mengorganisir dan menyusun kode mereka dengan baik. C# juga menawarkan tipe data yang kuat dan berbagai jenis tipe data, baik tipe data-nilai maupun tipe data-referensi. Keunggulan lainnya adalah bahwa C# dapat digunakan secara lintas-platform, tidak hanya dalam lingkungan Windows, tetapi juga dalam lingkungan Linux dan MacOS melalui implementasi seperti .NET Core dan Mono. Berkat dukungan untuk berbagai jenis aplikasi, mulai dari permainan video hingga aplikasi desktop dan web, C# menjadi salah satu bahasa pemrograman yang sangat populer. Dengan bantuan lingkungan pengembangan terintegrasi seperti Visual Studio, pengembang dapat meningkatkan produktivitas mereka dengan fitur-fitur seperti *IntelliSense, debugging,* dan *refactoring*. Dengan semua fitur dan kemampuannya, pemrograman C# merupakan keterampilan yang sangat berharga untuk dipelajari bagi siapa pun yang tertarik dalam pengembangan perangkat lunak.

Dalam *field* pembelajaran mesin memang banyak digunakan bahasa pemrograman python untuk implementasinya karena lebih cepat dan optimal, namun karena penelitian ini hanya berfokus untuk mengimplementasikan dan bukan pada proses optimisasi maka digunakan c# untuk bahasa pemrogramannya. Namun untuk metodennya dipastikan dapat diimplementasikan pula menggunakan bahasa lain seperti python.

2.13 Studi Literatur

Terdapat beberapa penelitian terdahulu tentang implementasi mobil otonom Beberapa pendekatan menggunakan lingkungan nyata dan simulasi untuk melatih dan meningkatkan model dalam lingkungan mengemudi otonom. seperti pada penelitian [2] yang mengimplementasikan algoritma PPO dan POCA pada mobil balap kart di *Unity Engine*. penelitian ini berfokus untuk mengetahui performansi model dari kedua algortima dengan mengukur *loss value* dan *qumulative reward*-nya. Penelitian yang serupa juga telah dilakukan [3] dengan membandingkan 3 algortima termasuk PPO dan 2 lainnya yaitu DQN dan *Deep Deterministic Policy Gradient* (DDPG). Berbeda dengan penelitian sebelumnya, penelitian ini berfokus meningkatkan performansi dari kendaraan otonom. Peneliti membandingkan studi

yang dibahas berdasarkan domain aplikasinya, algoritma yang digunakan pada Tabel 2.1 Studi Literatur.

Tabel 2.1 Studi Literatur

Penelitian	**Pengaplikasian**	**Metode**	**Pengujian performansi**	**Hasil**
[1] Quek *et al*	Navigasi kendaraan pada mobil otonom menggunakan Unity Engine	DQN dengan CNN sebagai input data	*Cumulative reward* per *step*	reward meningkat dengan tren naik yang menunjukkan bahwa agen terus belajar hingga mencapai kondisi stabil dengan nilai akhirnya yaitu 0,956
[2] Savid *et al*	Navigasi pada mobil balap kart 3D dengan menghindari penghalang menggunakan Unity Engine	PPO dan POCA	*Cumulative reward per step* dan nilai *loss* per *step* pada tahap training, dan jumlah tabrakan pada tahap testing (20 percobaan)	PPO mendapatkan peforma yang lebih optimal dari POCA. Pada grafik *cumulative reward* menunjukkan penurunan pada awal step namun setelah beberapa step meningkat secara bertahap dan menunjukkan tren naik yang artinya agen mampu belajar dengan baik. Didapatkan *cumulative reward* sebesar 0,761 pada akhir step lalu nilai *loss* sebesar 0,0013 **,** Jumlah tabrakan paling sedikit didapat sebanyak **2** kali pada model yang di latih

				sebanyak 6.000.000 step
[3] Tammewar *et al*	Kontrol berkendara pada mobil balap 2D	DQN, DDPG dan PPO	*Cumulative reward* per *episode* dan jumlah percobaan latihan	Hasil dari Penelitian ini menunjukan bahwa DQN unggul dibandingkan dengan algoritma lainnya. Grafik menunjukan tren naik yang stabil hingga akhir episode dengan menunjukkan nilai *cumulative reward* 0.2075 pada episode terakhir yaitu episode ke-2000
[12] Siboo1 *et al*	Kontrol kendaraan	DDPG dan PPO	Cummulative *reward* per *step*	DDPG unggul pada penelitian ini, *Reward* maksimum keseluruhan yang diperoleh oleh DDPG lebih besar daripada PPO. Untuk 100 episode, *reward* maksimum keseluruhan untuk DDPG adalah 0,245.