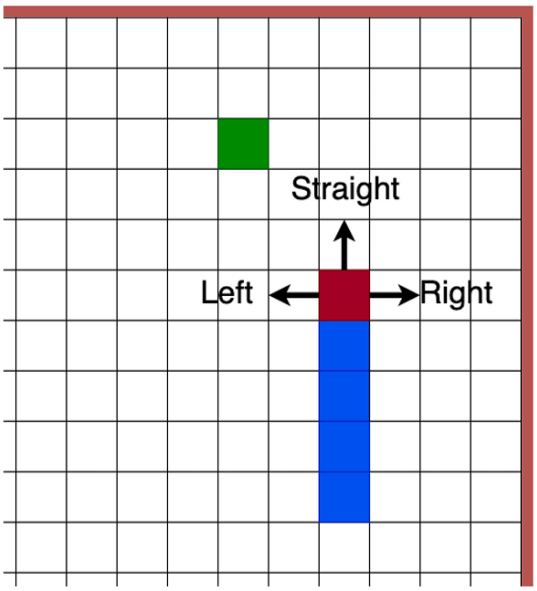
BAB II

TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 Permainan Ular Klasik



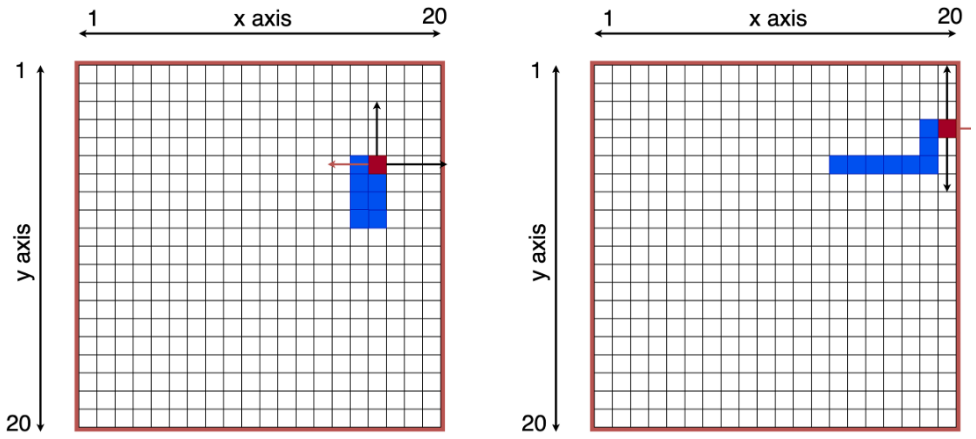
Gambar 2.1 Ilustrasi permainan ular (Sebastianelli dkk., 2021)

Blockade, atau dikenal juga sebagai Snake, adalah permainan *arcade* tradisional yang pertama kali diterbitkan pada tahun 1976 di platform Gremlin. Pada awalnya, permainan ini hanya tersedia untuk konsol seperti Atari 2600. Popularitasnya mulai meningkat secara global pada awal abad ke-21, terutama dengan munculnya ponsel Nokia. Tujuan utama permainan ini adalah untuk mengumpulkan sebanyak mungkin "apel" dalam batas papan yang diberikan guna mencapai skor tertinggi. Namun, semakin tinggi skornya, semakin panjang tubuh ularnya, dan papan permainan memberikan ruang yang semakin sempit untuk

bergerak, menjadikannya semakin menantang (Brown dkk., 2021). Ditunjukkan pada Gambar 2.1.

Aturan dari permainan pada Snake AI Competition di Universitas Innopolis (Brown dkk., 2021):

a. Ular hanya bisa bergerak lurus, ke kanan, dan ke kiri
b. Ular mencari makanan untuk mendapatkan skor dan skor yang didapat membuat panjang ular betambah 1 frame atau piksel
c. Jika ular menabrak badannya sendiri atau mencapai sudut dari *board*, maka game selesai. Ditunjukkan pada Gambar 2.2
d. Posisi makanan pada *board* berjumlah 4 dan posisinya *spawn* secara random.



Gambar 2.2 Keadaan yang berbahaya bagi ular (Sebastianelli dkk., 2021b)

Biasanya ada dua fase berbahaya saat bermain game Snake: pertama terjadi pada awal permainan ketika tubuh ular masih pendek dan bergerak lambat, yang dapat menyebabkan gangguan bagi pemain manusia dan mengakibatkan kematian ular; kedua terjadi kemudian saat jari manusia mulai sedikit lelah dan perlu istirahat.

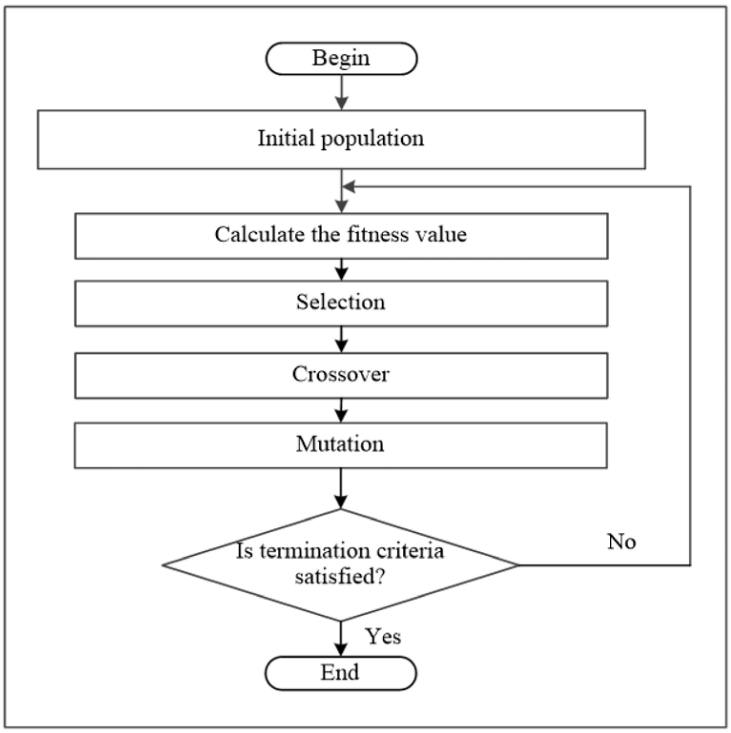
Kemudian ular menabrak tubuhnya sendiri, yang mengakibatkan akhir dari permainan (Pan dkk., 2023).

2.1.2 Algoritma Genetika

Algoritma genetika adalah jenis algoritma optimasi, yang berarti algoritma ini digunakan untuk menemukan nilai maksimum atau minimum dari suatu fungsi (Carr, 2014). Algoritma genetika biasanya digunakan untuk menghasilkan solusi berkualitas tinggi untuk masalah pencarian dan pengoptimalan dengan bergantung pada operator yang berorientasi pada biologi seperti seleksi, *crossover*, dan mutasi (Albadr dkk., 2020). Algoritma genetika telah digunakan secara luas dalam machine learning, adaptive control, combinatorial optimization, dan signal processing area. GA memiliki kemampuan pencarian global yang baik dan juga dianggap sebagai salah satu teknologi penting yang berkaitan dengan perhitungan cerdas modern (Yu dkk., 2017).

Parameter sederhana yang umum pada hampir semua algoritma genetika menurut (Carr, 2014), sebagai berikut:

a. Fungsi *fitness* untuk optimalisasi
b. Populasi dari kromosom
c. Seleksi dari kromosom yang akan di-*reproduce*
d. *Crossover* untuk membuat generasi selanjutnya pada kromosom
e. Random *mutation* pada pada kromosom untuk generasi yang baru



Gambar 2.3 *Flowchart* dari algoritma genetika (Yu dkk., 2017)

Penjelasan sederhana terkait prosedur dari algoritma genetika pada Gambar 2.3 dijelaskan oleh (Yu dkk., 2017) pada penelitiannya. *Initial population* memerlukan solusi yang mungkin untuk himpunan P, yaitu, serangkaian generasi acak dengan nilai yang nyata pada persamaan 2.1,

$$P = \{p_1, p_2, \dots, p_s\} \quad (2.\ 1)$$

Evaluasi atau menghitung nilai *fitness* merupakan tahapan kedua sebelum masuk ke tahapan biologi. Fungsi *fitness* harus digambarkan untuk mengevaluasi setiap kromosom dalam populasi, yang ditandai pada persamaan 2.2,

$$Fitness = g(P) \quad (2.\ 2)$$

Setelah tahapan evaluasi, dilakukanlah seleksi. Kromosom-kromosom akan disusun berdasarkan nilai *fitness*-nya. Seleksi *parents* dilakukan dengan melibatkan dua *parents* untuk tahapan *crossover* dan *mutation*.

Setelah proses seleksi selesai, *parent* dari kedua kromosom baru atau *offspring* (C1, C2) dibentuk menggunakan operator genetik. Kromosom baru (C1, C2) kemudian disimpan dalam *children population* C. Proses ini melibatkan operasi *crossover* dan *mutasi*. Operasi *crossover* digunakan untuk menukar informasi antara dua *parents* yang telah dipilih sebelumnya. Ada beberapa metode operator *crossover* yang umum, seperti *single-point, two-point*, *k-point crossover,* dan *arithmetical crossover*. Sementara itu, dalam operasi *mutation*, gen-gen dari kromosom *offspring* yang telah disilangkan dimodifikasi menjadi bentuk baru dengan nilai acak. Langkah selanjutnya, *children population* C telah sepenuhnya dibentuk dan akan ditransfer ke populasi berikutnya (P). P kemudian digunakan dalam iterasi yang selanjutnya, di mana seluruh proses dijalankan lagi. Iterasi akan berhenti jika terjadi konvergensi hasil, atau jika jumlah iterasi melebihi ambang batas maksimum (Yu dkk., 2017).

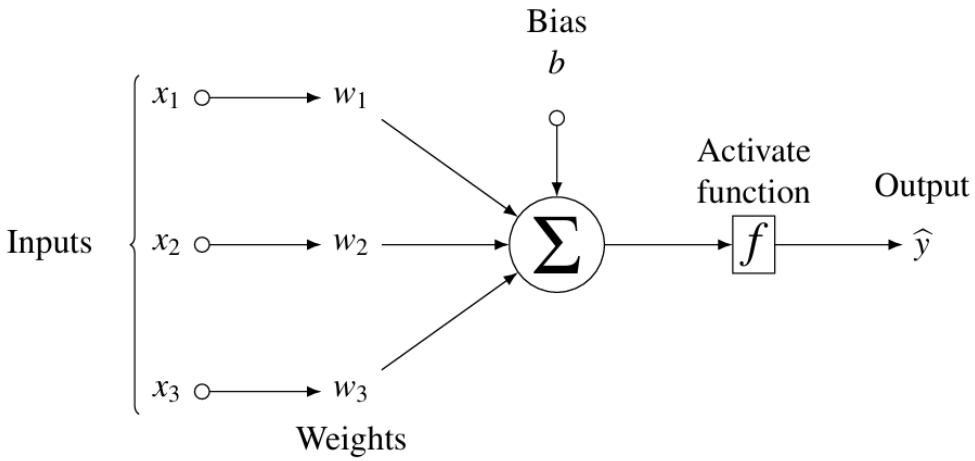
2.1.3 *Neural networks*

Neural networks dapat dijelaskan menggunakan sebuah kumpulan data, sebuah model, *loss function*, dan prosedur optimisasi. Kumpulan data mengacu pada total sampel yang tersedia untuk pelatihan, validasi, dan pengujian algoritma. Biasanya, kumpulan data ini dibagi menjadi tiga bagian: kumpulan data pelatihan, validasi, dan pengujian. Algoritma belajar dari kumpulan data pelatihan. Kemudian, kumpulan data validasi digunakan untuk mengevaluasi dan mengoptimalkan

algoritma pembelajaran. Terakhir, kumpulan data pengujian menjadi unseen data, dan digunakan untuk menguji algoritma yang telah dilatih. Model mengacu pada struktur yang menyimpan semua informasi yang menjelaskan fungsi yang telah dipelajari oleh algoritma. *Loss function* adalah metrik yang harus diminimalkan selama pelatihan. Prosedur optimisasi digunakan untuk menemukan parameter optimal dari model yang meminimalkan *loss function* (Goodfellow dkk., 2016).

Penelitian yang telah dilakukan oleh (Aldakheel dkk., 2021) memberikan penjelasan terkait arsitektur *neural networks* biasa dan *feed-forward neural networks*:

a. Standard *neural networks*



Gambar 2.4 Arsitektur dari *neural networks* (Aldakheel dkk., 2021)

Pada Gambar 2.4, *Neural networks* terdiri dari neuron-neuron yang memiliki satu output skalar dan beberapa input. Terdiri dari empat bagian: (i) nilai-nilai input, (ii) *weight* dan bias, (iii) *weighted total* (sum), dan (iv) fungsi aktivasi (unit ambang). Diagram skematis dari sebuah neuron buatan diilustrasikan pada Gambar 2.4, di mana x^1 hingga x^3 adalah input, w^1 hingga w^3 adalah *weight* yang sesuai, b

adalah bias, f adalah fungsi aktivasi yang diterapkan pada *weighted total* dari input, dan y adalah output dari neuron tersebut.

Secara matematika, bisa ditulis seperti pada persamaan 2.3,

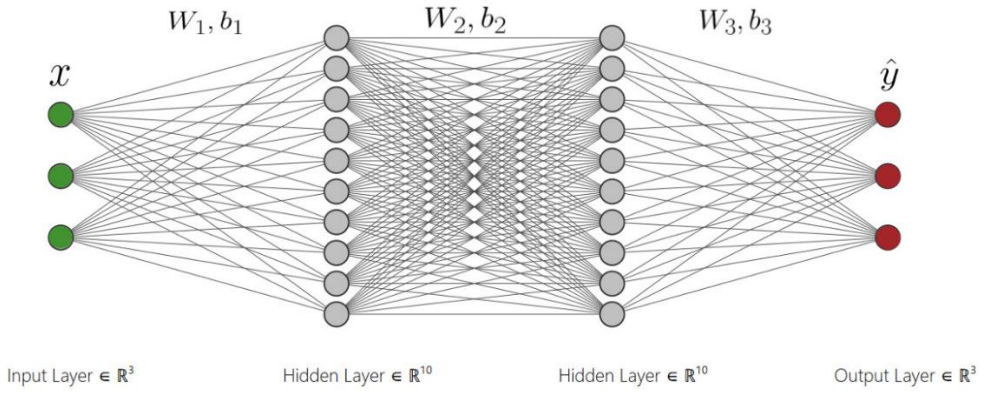
$$\hat{y} = f(\sum_{i=1}^{N} x_1 w_1 + b) \qquad (2.3)$$

N adalah bentuk sampel. Persamaan 2.3 juga dapat ditulis dalam bentuk yang sederhana, ditunjukkan pada persamaan di bawah (2.4),

$$\hat{y} = f(x^T W + b)\ dengan\ x = [x_1, \dots, x_N]^T\ dan\ W = [w_1, \dots, w_N]^T \qquad (2.4)$$

Fungsi aktivasi (f) menentukan output dari neuron dalam model *neural networks*, efisiensinya dalam komputasi, dan kemampuannya untuk berlatih dan konvergen setelah beberapa iterasi. Hal ini memberikan sifat-sifat non-linear pada network dan bertujuan untuk mengubah sinyal masukan dari sebuah node dalam *neural networks* menjadi *output signal*, yang kemudian digunakan sebagai input di lapisan berikutnya dari *neural networks*.

b. *Feed-forward Neural networks* (FFNN)



Gambar 2.5 Arsitektur dari *Feed-forward neural networks* (Aldakheel dkk., 2021)

Dalam sebuah FFNN, jumlah input dan output telah ditetapkan dan pengetahuan variabel yang sebelumnya diabaikan. Sesuai dengan kompleksitas data latihan, arsitektur FFNN (jumlah *hidden layer* dan neuron di setiap layer) harus ditentukan.

Arsitektur dari FFNN pada Gambar 3, bisa didefinisikan seperti pada persamaan 2.5,

$$\hat{y} = f(H_2 W_3 + b_3) \ \ dengan \ \ H_2 = f(H_1 W_2 + b_2) \ \ dan \ H_1 = (xW_1 + b_1) \qquad (2.5)$$

Vektor ŷ adalah output, vektor input x berisi fitur-fitur sampel, W adalah matriks bobot, dan b adalah vektor bias untuk masing-masing lapisan. Arsitektur FFNN dapat disederhanakan seperti pada persamaan 2.6, mengingat bahwa y adalah fungsi sebenarnya:

$$\hat{y} = \hat{y}(x, W, b) \ \ dan \ \ \{b^1, W^1\} = Arg\left\{\min_{bW} \mathcal{L}(\hat{y}, y)\right\} \qquad (2.6)$$

Loss function $\mathcal{L}$ diminimalkan untuk menemukan weight yang optimal W′ dan bias b′ dari model *neural networks* yang telah dilatih.

Jumlah dari input layer, neuron pada hidden layer, dan output layer pada Gambar 2.5 merupakan gambaran umum dari FFNN. Jumlah pasti dari arsitektur FFNN pada penelitian ini akan dijelaskan lebih detail pada bagian metodologi penelitian.

2.2 State-of-the-Art

State of the art yang disajikan pada Tabel 2.1 berisi tentang hasil penelitian yang sudah dilakukan sebelumnya, berkaitan dengan penerapan kecerdasan buatan pada permainan ular. Selain itu, terdapat juga gap penelitian yang membedakan dari penelitian yang sudah dilakukan sebelumnya.

Tabel 2.1 State-of-the-Art penelitian

No	Nama Peneliti/Journal		Hasil Penelitian
1	**Peneliti**	(B. HalmosiC. Sik-Lányi, 2019)	Parameter yang digunakan: Selection rate 0,5% yang berarti 50% populasi dengan nilai *fitness* terburuk akan dibuang, *mutation rate* 0,0001%. Menggunakan 2 tipe *feed-forward neural networks* pada eksperimennya. Pertama, 4-4-4 *neural networks* yang terdiri dari input layer, satu hidden layer, dan output layer dengan masing-masing 4 neurons. Kedua, 8-8-4 yang memiliki 8-8 neurons pada input layer, hidden layer, dan 4 neuron pada output layer. Hasil eksperimen menunjukkan skor yang didapatkan lebih dari 25 secara rata-rata. Namun, generasi yang digunakan pada penelitian ini hanya dibatasi sampai 500 generasi saja.
	Judul	Learning to play snake using genetic *neural networks*	
	Algoritma/ Metode	Algoritma genetika, *neural networks*	
2	**Peneliti**	(Białas, 2019)	Penelitian dilakukan dengan 2 kali eksperimen. Eksperimen pertama skor tertinggi adalah 60 dengan menggunakan 2000 populasi. Eksperimen kedua mendapat 40-45 skor tertinggi dengan menggunakan 1000
	Judul	Implementation of artificial intelligence in Snake game	

No	Nama Peneliti/Journal		Hasil Penelitian
		using genetic algorithm and *neural networks*	generasi. Penelitian ini menggunakan *mutation rate* 0.5, hanya 1 hidden layer, dan top population 5. Tidak terdapat GUI dari permainan ular pada pernelitian ini.
	Algoritma/ Metode	Algoritma genetika, *neural networks*	
3	**Peneliti**	(Chi Yuen dkk., 2021)	Penelitian ini dilakukan dengan 4 kali eksperimen. Eksperimen pertama mendapat nilai terbaik dengan menggunakan *mutation rate* 0.05. *Mutation rate* 0.00 memiliki nilai yang terlalu rendah sehingga ular tidak dapat belajar apa-apa, sedangkan *mutation rate* 0.08 terlalu tinggi untuk ular. Eksperimen kedua mendapat skor tertingi pada 20000 populasi. Menggunakan 20 populasi hanya unggul dalam kecepatan komputasi, tapi skor yang didapatkan sangat rendah. Ketika ukuran populasinya 200, mendapatkan skor yang lebih baik dibandingkan dari 20 populasi dan waktu komputasi yang lebih cepat dibandingkan menggunakan 20000 populasi. Ukuran populasi 20000 mendapatkan skor tertinggi meskipun memakan waktu yang cukup lama. Hasil akhir yang diberikan hanya berupa grafik hasil kinerja dari ular tanpa ada GUI dari permainan ular. Selain itu, eksperimen dari parameter masing-masing hanya dilakukan 1 kali.
	Judul	Investigating parameters of genetic algorithm and neural network on classic snake game	
	Algoritma/ Metode	Algoritma genetika, *neural networks*	
4	**Peneliti**	(Hau Hor dkk., 2022)	Peneliti melakukan analisis terhadap kinerja ular dalam berbagai situasi perubahan parameter. Setiap parameter memiliki pengaruh tertentu terhadap kinerja ular. Terbukti bahwa memberikan hasil yang baik ketika parameter seperti panjang blok, persentase mutasi, dan intensitas mutasi diatur pada nilai-nilai menengah. Mengenai persentase parameter
	Judul	Snake Game: A genetic neural network approach	
	Algoritma/ Metode	Algoritma genetika, *neural networks*	

No	Nama Peneliti/Journal		Hasil Penelitian
			terbaik/terburuk yang dihasilkan, mendapat rasio tertinggi sebesar 18:2 and hasil tersebut akan memberikan skor terbaik bagi ular. Output yang dihasilkan hanya berupa grafik yang menunjuka skor dari ular tanpa ada GUI dari permainan ular.
5	**Peneliti**	(Pan dkk., 2023)	Hasil penelitian berupa perbandingan skor dari algoritma DQN original, DQN final, dan PPO. DQN final mendapatkan skor yang paling unggul dengan mendapat 20 sampai 25, PPO mendapat 10 sampai 20, dan DQN original mendapat 5 sampai 15. Peneliti menyebutkan bahwa dari hasil akhir yang didapat, DQN dan PPO tidak lebih baik dari evolutionary algorithm.
	Judul	Playing the Snake Game with Reinforcement Learning	
	Algoritma/ Metode	Deep Q-Network, Proximal Policy Optimization	
6	**Peneliti**	(Jipong dkk., 2016)	Dalam eksperimennya, langkah awal berupa memeriksa kebutuhan untuk menggabungkan beberapa fungsi penilaian, seperti smoothness, ruang, dan makanan. Kemudian, membandingkan empat varian EA dan mengidentifikasi operator *crossover* yang baik serta metode lingkungan yang efektif. Manfaat bobot terkait panjang dalam pengontrol juga dikonfirmasi oleh hasil optimasi EA dan permainan acak. Hasil akhirnya membandingkan antara pengontrol yang telah berevolusi dengan pengontrol heuristik. Meskipun pengontrol yang dibuat dapat bermain lebih baik dalam permainan tertentu, namun dalam kondisi tertentu tidak dapat bermain dengan baik.
	Judul	Snake Game AI: Movement Rating Functions and Evolutionary Algorithm-based Optimization	
	Algoritma/ Metode	Evolutionary Algorithm	

No	Nama Peneliti/Journal		Hasil Penelitian
7	**Peneliti**	(Kong & Mayans, 2021)	Penelitian ini melakukan 100 kali percobaan dengan ukuran *board* 10 x 10 pada masing-masing algoritma untuk membandingkan hasil yang terbaik. Hasilnya almighty move mendapat *score* yang paling tinggi 100, A* search with forward checking mendapatkan posisi kedua 40 sampai 80, dan dilanjutkan dengan A* dengan rentang skor 22-40, BFS mendapat 19 sampai 24, dan random move dibawah 10. Semua algoritma menampilkan kelebihan dan kekurangan dibandingkan dengan random move. Informed search algorithms menghasilkan skor yang sangat baik di awal tetapi tidak konsisten dan skor yang didapat semakin menurun. Almighty move merupakan algoritma yang lambat, tetapi skor yang dihasilkan cukup maksimal sampai akhir dari permainan.
	Judul	Automated Snake Game Solvers via AI Search Algorithms	
	Algoritma/ Metode	Best first search, A* search, A* search with forward checking, random move, almighty move	
8	**Peneliti**	(Sebastianelli dkk., 2021b)	Deep neural network yang digunakan memilih 5 layer yang terdiri dari layer pertama sebagai input layer yang memiliki 11 nodes. Ketiga layer selanjutnya digunakan untuk ekstraksi fitur untuk memprediksi pergerakan ular dari masing-masing state yang dikunjungi. Ketiga layer ini memiliki 100 nodes yang terhubung satu sama lain. Layer terakhir adalah output layer. Hasil dari pengujian mendapat skor yang bertahap dari generasi yang digunakan. Generasi 25 mendapat skor 2, generasi 46 mendapat skor 4, generasi 75 mendapat skor 21, generasi 77 mendapat skor 35, dan generasi 79 mendapat skor 52.
	Judul	A deep Q-learning based approach applied to the snake game	
	Algoritma/ Metode	Deep Q-Learning	
9	**Peneliti**	(Khunger dkk., 2021)	Fokus dari penelitian ini tidak hanya untuk mendapatkan skor tertinggi yang bisa didapat oleh ular. Namun, terdapat juga catatan terkait penyebab matinya ular. Penelitian ini menggunakan Deep Q-Learning
	Judul	Reinforcement Learning in Game Playing	

No	Nama Peneliti/Journal		Hasil Penelitian
	Algoritma/ Metode	Deep Q-Learning, deep *neural networks*	dan deep *neural networks* yang memiliki 5 layer yang terdiri dari 1 input layer, 3 hidden layer, dan 1 output layer. Hasil eksperimen menunjukan reward yang didapatkan yaitu, +500 dari makanan, -100 karena menabrak tembok, -100 menabrak dirinya sendiri, dan -10 untuk hal lainnya.
10	**Peneliti**	(Wei dkk., 2018)	Deep Q-Network digunakan dalam penelitian ini untuk memainkan game ular. CNN disertakan untuk melatih Q-learning. Dari eksperimen yang dilakukan terdapat perbandingan antara manusai asli, DQN biasa, dan refined DQN dalam meraih skor yang tinggi. Rata-rata manusia mendapat skor 1.98 dengan skor terbaik yang didapat adalah 15. DQN biasa mendapat skor rata-rata 0.26 sedangkan skor terttinggi yang didapat adalah 2. Refined DQN mendapat skor rata-rata 9.04 dan skor terbaik adalah 17. Hasil eksperimen menunjukkan bahwa model DQN yang disempurnakan mengungguli model dasar dari DQN. Selain itu, refined DQN juga dapat melampaui hasil dari manusia.
	Judul	Autonomous agents in snake game via deep reinforcement learning	
	Algoritma/ Metode	Deep Q-Networks	
11	**Peneliti**	(Zhang & Cai, 2020)	Penelitian yang dilakukan bukan hanya sekedar implementasi algoritma. Namun, terdapat perbandingan antara algoritma double DQN dan PPO dalam mendapatkan skor tertinggi. Dari eksperimen yang sudah dilakukan, DQN mendapat skor tertinggi 13 sedangkan PPO mendapat skor tertinggi 18. Hasil dari penelitian menunjukan bahwa DQN mendapatkan skor yang lebih stabil dibandingkan dengan PPO. Namun, PPO mendapatkan skor tertinggi meskipun tidak konsisten.
	Judul	Train a snake with reinforcement learning algorithms	
	Algoritma/ Metode	Double deep Q-Networks, Proximal Policy Optimization	

No	Nama Peneliti/Journal		Hasil Penelitian
12	**Peneliti**	(B. Ma dkk., 2016)	Penelitian ini berfokus pada perbandingan performa dari algoritma heuristic, reinforcement learning – SARSA, dan Q-Learning. Eksperimen dilakukan sebanyak 1000 kali pengujian dengan hasil algoritma heuristk (Approximation algorithm) stabil mendapat 77.504, SARSA mendapat skor tertinggi 61.830, dan Q-Learning mendapat skor tertinggi 36.567. Hasil ini menunjukan bahwa SARSA dan Q-Learning masih belum bisa melebihi skor dari algoritma heuristic. Namun, SARSA lebih diunggulkan dibandingkan dengan Q-Learning yang mendapat skor terkecil dan tidak stabil.
	Judul	Exploration of Reinforcement Learning to SNAKE	
	Algoritma/ Metode	Convolutional *neural networks*, reinforcement learning – SARSA, Approximation Algorithm, Q-Learning	
13	**Peneliti**	(Tushar & Siddique, 2023)	Peneliti melakukan modifikasi pada algoritma DRL yang mereka gunakan. DRL standar dikembangkan dengan menggunakan CNN dari Q-Networks agar dapat menghasilkan model yang lebih baik. Hasil dari penelitian berupa perbandingan model yang dibuat oleh peneliti dengan model yang dibuat oleh peneliti lain. Penelitian yang sebelumnya dilakukan oleh (Wei dkk., 2018) membandingkan modelnya refined DQN yang lebih unggul dibandingkan standar DQN dan human agen dengan mendapat skor rata-rata 9.04 dan tertinggi 17. Eskperimen yang sudah dilakukan oleh peneliti membuktikan bahwa model yang dibuatnya jauh lebih baik dibandingkan dengan refined DQN. Model yang dibuat menghasilkan skor rata-rata 9.53 dan skor tertinggi 20.
	Judul	A Memory Efficient Deep Reinforcement Learning Approach For Snake Game Autonomous Agents	
	Algoritma/ Metode	Modified deep reinforcement learning, convolutional *neural networks*	

No	Nama Peneliti/Journal		Hasil Penelitian
14	**Peneliti**	(Chindarkar dkk., 2020)	Penelitian ini membahas bagaimana reinforcement learning dapat digunakan untuk melatih agen yang akan belajar cara memainkan permainan ular klasik saat dilatih menggunakan DQN. Peneliti mengusulkan mekanisme reward yang dirancang dengan cermat untuk menyelesaikan masalah reward yang jarang dan tertunda. Selain itu, peneliti menggunakan strategi training gap untuk menghilangkan training experience yang tidak tepat dan menerapkan metode *dual experience replay* untuk lebih meningkatkan efektivitas pelatihan. Hasil dari penelitian ini tidak terdapat kejelasan dari skor yang didapat oleh model yang digunakan peneliti.
	Judul	Training an AI agent to play a Snake Game via Deep Reinforcement Learning	
	Algoritma/ Metode	Deep Q-Learning, deep *neural networks*	
15	**Peneliti**	(Sagar dkk., 2020)	Penelitian ini berfokus pada perbandingan 5 algoritma yang akan diterapkan dalam permainan ular. Hasilnya menunjukkan bahwa AI Bot terampil untuk mencapai skor maksimum dalam jumlah langkah yang minimal. Almighty move unggul dengan mendapatkan skor paling tinggi dibandingkan algoritma yang lain, yaitu 100. Posisi kedua ditempati oleh A* with forward checking dengan skor 40 sampai 80. Selanjutnya diikuti oleh A* dengan skor 24 sampai 40, BFS mendapat 14 sampai 26, dan random move mendapat nilai paling rendah, yaitu 3 sampai 7.
	Judul	Solving the Classic Snake Game Using AI for Training Electronic Sport Players	
	Algoritma/ Metode	Best first search, A* search, A* search with forward checking, random move, almighty move	

2.3 Matriks Penelitian

Tabel 2.2 Matriks penelitian

No	Penelitian	Tujuan penelitian			Algoritma									
		Pengujian Parameter	GUI Permainan Ular	Perbandingan Algoritma	Algoritma Genetika	Neural Network	Deep Reinforcement Learning	Proximal Policy Optimization	Evolutionary Algorithm	Best First Search	A* Search	A* Search with Forward Cehcking	Almighty Move	Approximation Algorithm
1	(B. HalmosiC. Sik-Lányi, 2019)		✓		✓	✓								
2	(Białas, 2019)	✓			✓	✓								
3	(Chi Yuen dkk., 2021)	✓			✓	✓								
4	(Hau Hor dkk., 2022)	✓			✓	✓								
5	(Pan dkk., 2023)			✓			✓	✓						

No	Penelitian	Tujuan penelitian			Algoritma									
		Pengujian Parameter	GUI Permainan Ular	Perbandingan Algoritma	Algoritma Genetika	Neural Network	Deep Reinforcement Learning	Proximal Policy Optimization	Evolutionary Algorithm	Best First Search	A* Search	A* Search with Forward Cehcking	Almighty Move	Approximation Algorithm
6	(Jipong dkk., 2016)	✓							✓					
7	(Kong & Mayans, 2021)			✓						✓	✓	✓	✓	
8	(Sebastianelli dkk., 2021b)	✓	✓			✓	✓							
9	(Khunger dkk., 2021)	✓	✓			✓	✓							
10	(Wei dkk., 2018)	✓	✓	✓			✓							
11	(Zhang & Cai, 2020)		✓	✓			✓	✓						
12	(B. Ma dkk., 2016)			✓		✓	✓							✓

No	Penelitian	Tujuan penelitian			Algoritma									
		Pengujian Parameter	GUI Permainan Ular	Perbandingan Algoritma	Algoritma Genetika	Neural Network	Deep Reinforcement Learning	Proximal Policy Optimization	Evolutionary Algorithm	Best First Search	A* Search	A* Search with Forward Cehcking	Almighty Move	Approximation Algorithm
13	(Tushar & Siddique, 2023)	✓	✓	✓		✓	✓							
14	(Chindarkar dkk., 2020)						✓							
15	(Sagar dkk., 2020)			✓						✓	✓	✓	✓	

Berdasarkan penelitian terkait yang sudah disajikan sebelumnya, dapat disimpulkan dalam bentuk matriks penelitian yang ada pada Tabel 2.2. Pada matriks penelitian, sudah banyak peneliti yang menerapkan berbagai algoritma pada permainan ular. Namun, baru ada 4 penelitian yang menggunakan algoritma genetika dan *neural networks*. Selain itu, dari keempat penelitian hanya 1 penelitian

yang menampilkan GUI dari permainan ular. Penelitian yang akan dilakukan ini akan menampilkan GUI dari permainan ular beserta dengan pengujian parameter, yang sebelumnya belum dilakukan.