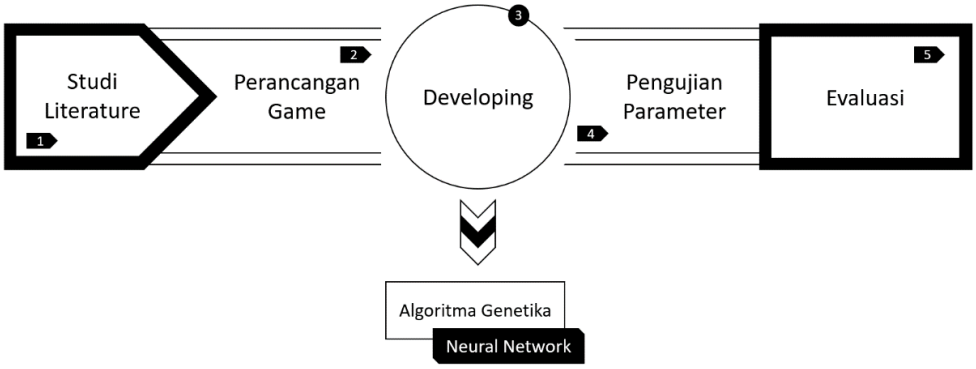
BAB III

METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Pada metodologi penelitian dijelaskan bagaimana tahapan dari penelitian yang akan dilakukan. Tahapan penelitian disajikan pada Gambar 3.1



Gambar 3.1 Tahapan penelitian

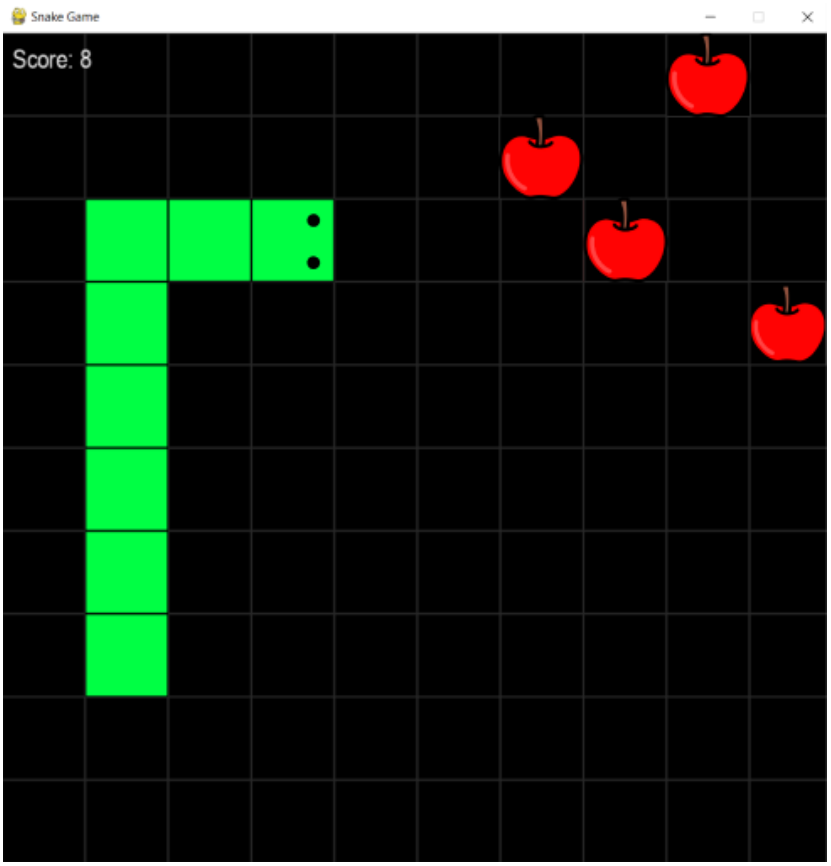
3.2 Studi Literature

Studi literature dilakukan untuk mempelajari penelitian-penelitian yang sebelumnya sudah dilakukan. Hal ini dilakukan untuk mencari kekurangan dan kelebihan dari penelitian sebelumnya untuk diaplikasikan pada penelitian yang akan dilakukan. Selain itu, menciptakan gap antara penelitian yang akan dilakukan dan penelitian sebelummya menjadi tujuan utama dari dilakukannya studi literature. Tahapan studi literature terdapat pada Tabel 2.1 dan Tabel 2.2.

3.3 Perancangan Game

Membuat konsep dari permainan ular merupakan tahapan kedua dari penelitian. Bagaimana permainan ular akan berjalan dan aturan-aturan yang akan

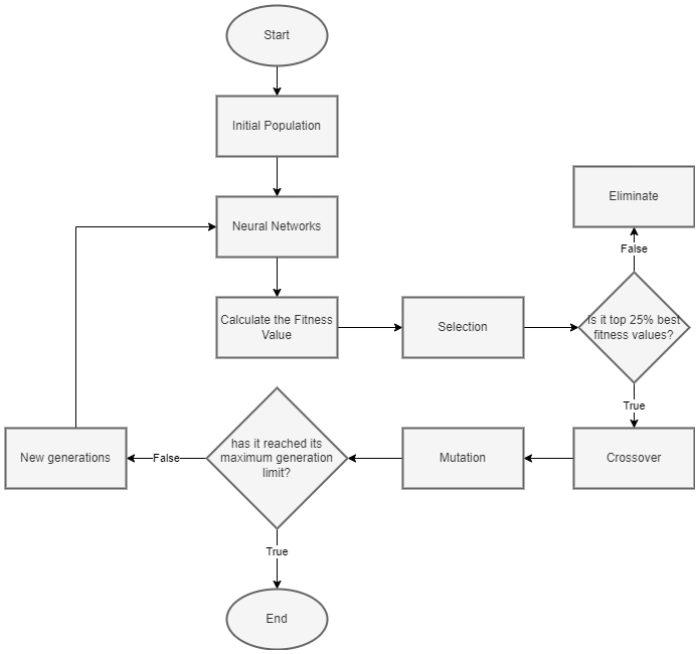
diberlakukan di dalamnya, dibuat pada tahapan ini. Gambar 3.2 merupakan konsep dari permainan ular yang akan dibangun:



Gambar 3.2 GUI dari permainan ular

a. GUI dari permainan ular hanya sebesar 10x10 pada Gambar 3.2
b. Terdapat 4 makanan yang akan muncul secara acak pada permainan. Ketika ular berhasil mendapatkan 1 makanan, makanan lainnya akan muncul
c. Pergerakan ular 4 arah: UP, DOWN, LEFT, RIGHT
d. Permainan selesai ketika ular menabrak badannya sendiri atau ular menabrak batas permainan pada ujung *board* dan ketika ular mencapai *max turn*-nya

3.4 Developing



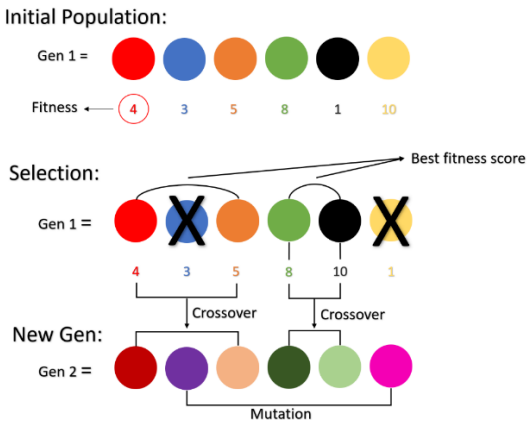
Gambar 3.3 Flowchart dari kombinasi algoritma genetika dan *neural networks*

Gambar 3.3 merupakan flowchart dari implementasi algoritma genetika dan *neural networks* pada permainan ular. Sederhananya, populasi akan dibuat dengan nilai matriks yang acak sebelum masuk ke *neural networks*. Selanjutnya, pada penghitungan nilai fitness masing-masing ular akan bermain sampai kalah sehingga mereka akan mendapatkan skor. Ketika semua ular sudah memiliki skor, dilakukanlah seleksi. Populasi yang terpilih untuk melanjutkan ke tahapan *crossover* dan mutasi hanyalah sebesar 25% terbaik, sedangkan sisanya akan dihapus. Jika pada pengujian jumlah generasi belum sampai batas maksimum yang ditentukan, tahapan tersebut akan dilakukan pengulangan dengan menggunakan generasi yang baru.

3.4.1 Algoritma Genetika

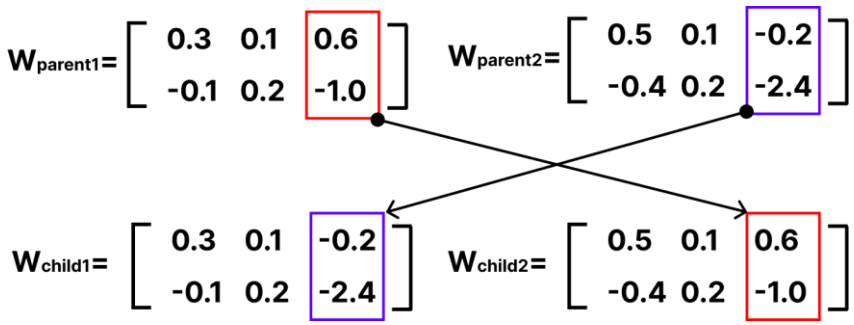
Dalam algoritma genetika, *crossover* dan mutasi adalah dua operator penting yang memainkan peran krusial dalam pengembangan algoritma yang efisien (Kumar dkk., 2021). Tahapan ini yang memainkan peran penting dalam mencari solusi terbaik pada permainan ular. Ilustrasi bagaimana proses *crossover* dan mutasi ditunjukkan pada Gambar 3.4. dan Gambar 3.5.

Bentuk lingkaran dapat digambarkan sebagai ular yang masing-masing sudah memiliki nilai *fitness*. Dari keenam ular tersebut dilakukanlah seleksi dengan mengeliminasi 2 ular dengan nilai fitness terburuk. Ular yang memiliki nilai *fitness* terbaik akan dilakukan *crossover*, yang dimana menggabungkan informasi dari 2 ular pada generasi 1 sehingga melahirkan ular yang baru pada generasi 2. Warna ular hasil dari *crossover* digambarkan memiliki kesamaan dengan induknya. Ini merupakan tanda bahwa ular hasil dari *crossover* membawa informasi campuran dari kedua induknya. Selain itu, ular hasil dari mutasi memiliki warna yang jauh berbeda dengan induknya. Ini terjadi karena informasi yang dimiliki induk sebelumnya dirubah secara acak untuk memperkenalkan variasi baru ke dalam populasi.

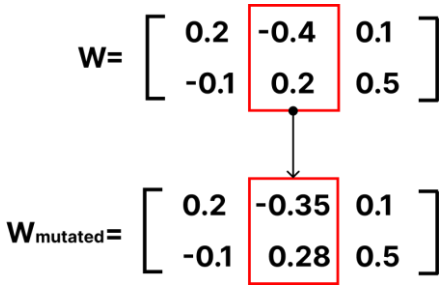


Gambar 3.4 Proses dari algoritma genetika

Jika digambarkan ke dalam matriks bobot, hasil dari operasi *crossover* dan *mutation* akan menjadi seperti pada Gambar 3.5 dan Gambar 3.6



Gambar 3.5 Operasi *crossover* pada matriks bobot



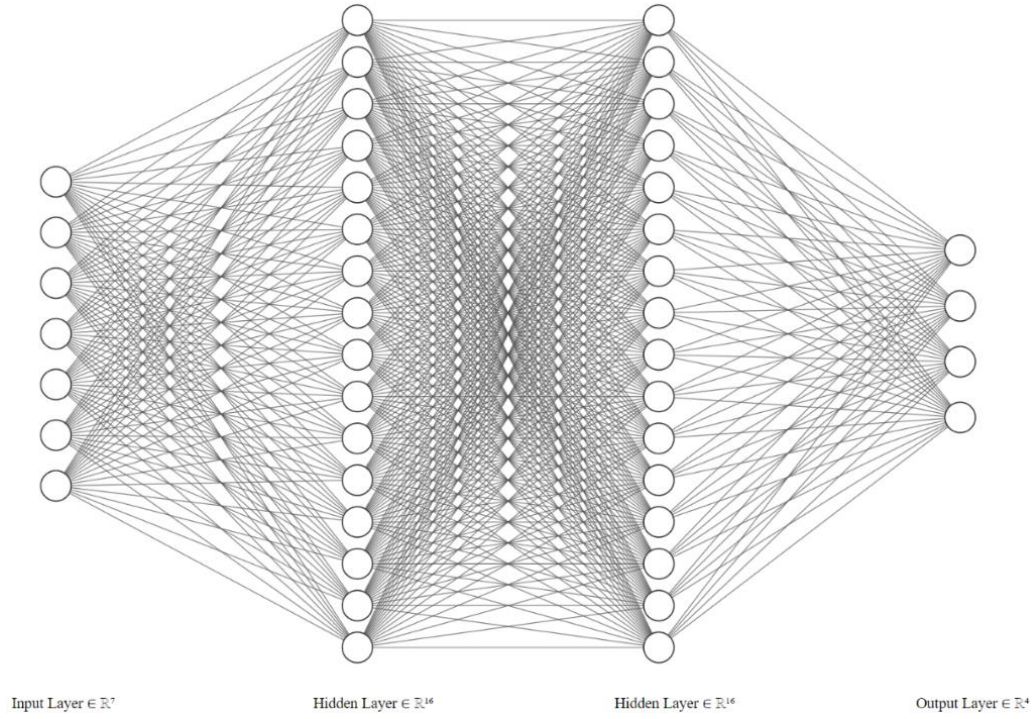
Gambar 3.6 Operasi mutasi dengan matriks bobot

Pada Gambar 3.5 kedua parent saling bertukar informasi untuk menciptakan bobot child yang baru. Bobot child yang pertama akan mengambil dua digit terakhir bobot dari parent kedua dan begitu juga sebaliknya. Gambar 3.6 menggambarkan bagaimana mutasi dari bobot terjadi. Berbeda dengan *crossover* yang hanya bertukar informasi, pada mutasi ini informasinya berubah sesuai dengan *mutation chance* dan *mutation size* yang di-*set*. Semakin besar *mutation size*-nya, maka semakin besar juga perubahan informasinya.

3.4.2 *Feed-forward Neural networks*

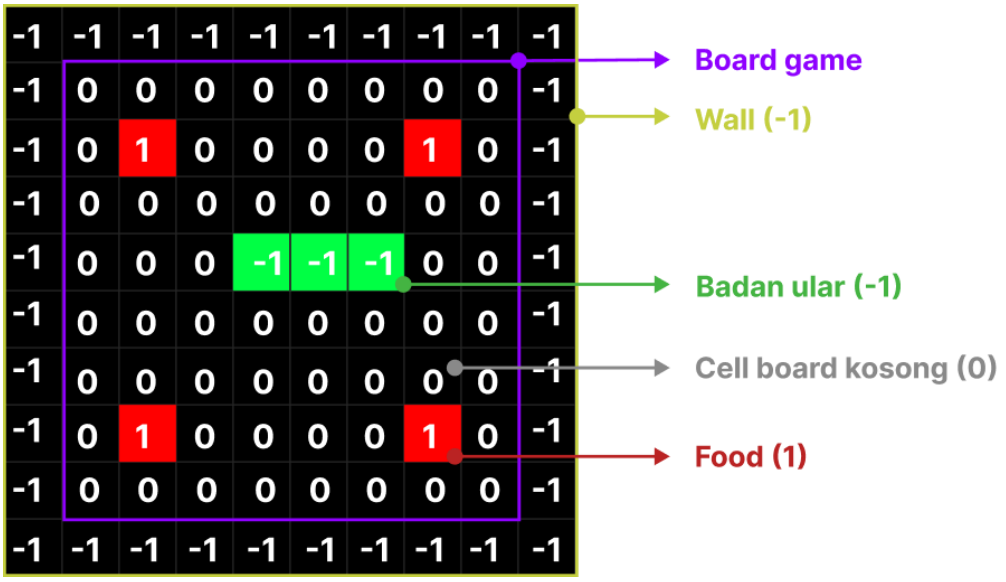
Jenis *neural networks* yang akan digunakan pada penelitian ini adalah *Feed-forward Neural networks* (FFNN). *Neural networks* yang akan dibangun nantinya memiliki 4 layer, yang terdiri dari 1 input layer, 2 hidden layer, dan 1 output layer.

Input layer memiliki 7 neuron, masing-masing hidden layer memiliki 16 neuron, dan output layer memiliki 4 neuron. Arsitektur dari *neural networks* disajikan pada Gambar 3.7



Gambar 3.7 Arsitektur dari *feed-forward neural networks*

Tujuh neuron pada input layer di-set berdasarkan dari window size dari permainan ular. Jumlah neuron pada hidden layer dapat berubah sesuai dengan pengujian parameter yang akan dilakukan. Default yang di-set adalah 16 neuron. Pada output layer berisi 4 neuron berdasarkan dari arah pergerakan ular yang di mana hanya 4 arah: UP, DOWN, LEFT, RIGHT.



Gambar 3.8 Vektor input pada *board* permainan

Berdasarkan window size pada Gambar 3.8, banyaknya neuron pada input layer adalah 49 (window_size^2) + 1 (bias), yang di mana masing-masing *cell* memiliki informasi sebagai berikut:

a. -1 untuk keadaan yang berbahaya seperti dinding dan tubuh ular

b. 1 untuk makanan

c. 0 untuk bagian *cell* yang kosong

Vektor input yang digambarkan pada Gambar 3.8 akan dimasukan sebagai nilai x pada perhitungan hidden layer 1. Hal inilah yang membuat *neural networks* dapat membuat pergerakan ularnya menjadi efisien.

3.4.3 Peran dari masing-masing algoritma

a. Algoritma Genetika

Algoritma genetika adalah teknik optimasi yang terinspirasi oleh proses evolusi biologis. Berikut adalah bagaimana perannya dalam permainan ular:

(1) Inisasi populasi: Algoritma dimulai dengan populasi awal dari berbagai solusi acak (individu). Dalam permainan ular, setiap individu mewakili strategi atau urutan langkah tertentu yang harus diikuti oleh ular.

(2) Evaluasi: Setiap individu dievaluasi berdasarkan kinerjanya dalam permainan yang sebelumnya sudah dimainkan oleh *neural networks*. Misalnya, seberapa jauh ular dapat bergerak tanpa menabrak dinding atau tubuhnya sendiri, dan berapa banyak makanan yang bisa dikumpulkan.

(3) Selection: Individu dengan kinerja terbaik dipilih untuk menjadi *parent* dari generasi berikutnya untuk dilanjutkan pada proses *crossover* dan *mutation*. Ini seperti memilih yang paling kuat untuk bertahan hidup dan bereproduksi.

b. Feed-forward neural networks

Sebagai vision dari ular yang menentukan arah gerak yang efisien, *neural networks* memiliki peran sebagai berikut:

(1) Pembelajaran berdasarkan data: Neural networks menerima data input, seperti posisi ular, makanan, dan rintangan di sekitar. Data ini diproses melalui dua *hidden layer* yang mencoba memahami pola atau fitur dalam data. *Neural networks* menghasilkan keputusan, seperti arah yang harus diambil oleh ular (atas, bawah, kiri, kanan)

(2) Pelatihan: *Neural networks* dilatih menggunakan data dari permainan. Selama pelatihan, *networks* mencoba mengurangi kesalahan antara prediksi (keputusan arah) dan hasil yang diinginkan (tidak menabrak dan mengumpulkan makanan). Data yang digunkan merupakan data yang

sudah diolah oleh algoritma genetika untuk mengurangi kesalahan prediksinya.

3.5 Pengujian Parameter

Pengujian ini dilakukan untuk mengetahui interaksi dari masing-masing parameter yang dapat mempengaruhi performa dari ular. Misalnya pada algoritma genetika, bagaimana peran dari generasi dan populasi dalam meningkatkan performa dan juga mempengaruhi efisiensi dari *runtime* programnya. Berbeda dengan penelitian sebelumnya (Białas, 2019) yang hanya melakukan dua kali eksperimen, pengujian parameter pada penelitian ini terdiri dari 4 pengujian dengan 60 kali eksperimen, sebagai berikut:

a. Pengujian 1 terdiri dari 10 eksperimen dengan jumlah generasi yang besar, yaitu 1000. Namun memiliki populasi yang kecil, yaitu 100.
b. Pengujian 2 terdiri dari 10 eksperimen dengan jumlah generasi yang kecil, yaitu 100. Namun memiliki populasi yang besar, yaitu 1000
c. Pengujian 3 terdiri dari 10 eksperimen dengan jumlah generasi dan populasi sama sama besar, yaitu 1000
d. Pengujian 4 terdiri dari 30 eksperimen: 20 eksperimen menguji masing-masing *mutation chance* 5 kali dari 0.1% sampai 0.5% dan 10 eksperimen menguji jumlah neuron pada hidden layer (16 dan 24)

Banyaknya eksperimen untuk masing-masing pengujian didasarkan pada gap yang dihasilkan. Misalnya untuk pengujian 1, 2, dan 3 hanya dilakukan sebanyak 10 kali eksperimen karena perbedaan hasil dari masing-masing pengujian sudah sangat

jauh sekali sehingga dapat diambil kesimpulan mana yang lebih baik. Menambah jumlah eksperimen tidak dapat merubah hasilnya.

3.6 Evaluasi

Evaluasi merupakan tahapan akhir dari penelitian ini. Pada tahapan ini mulai diklasifikasikan parameter-parameter yang menghasilkan performa terbaik. Parameter yang sudah ditetapkan ini nantinya dapat digunakan sebagai rekomendasi settingan terbaik untuk kombinasi algoritma genetika dan *neural networks* kedepannya.