BAB 2

TINJAUAN PUSTAKA

2.1 *Grafologi*

Grafologi berasal dari kata *graphos* yang berarti coretan dan *logos* adalah ilmu. Berdasarkan pengertian tersebut dapat dikatakan bahwa *grafologi* merupakan sebuah cabang ilmu yang mempelajari sebuah coretan tangan. Tulisan tangan disini bukan selalu huruf saja yang dapat dianalisis, namun sebuah coretan- coretan tangan juga dapat memiliki arti tersendiri. *Grafologi* juga dikatakan sebagai seni membaca tulisan tangan, karena tulisan setiap orang memiliki ciri- ciri yang khusus[2].

Grafologi adalah seni dan ilmu yang mempelajari tentang tulisan tangan. Karena tulisan tangan berasal dari otak manusia, maka apa yang dituangkan dalam tulisan itu adalah buah pikirannya. Buah pikiran manusia ini dapat memberi gambaran atau mencerminkan kepribadian manusia. Meski kegiatan menulis ini tampak sebagai kegiatan yang dikendalikan oleh pikiran sadar, tetapi ternyata justru sangat dipengaruhi oleh pikiran dawah sadar manusia. Oleh karena itu, bentuk, gaya tulisan, dan karakter-karakter dalam diri masing-masing orang tidaklah sama [6].

2.1.1 Sejarah *Grafologi*

Ilmu *grafologi* telah dikenal sejak berabad-abad yang lalu. Misalnya di Tiongkok, pengetahuan tentang tulisan tangansudah digunakan sejak tahun 1000 Masehi, meskipun tudak secara ilmiah. Baru pada tahun 1622, seorang dokter dari Italia, Dr. Camillo Baldi, mengemukakan penemuan tentang ilmu pengenalan tulisan yang dibuat secara sistematik/ilmiah dan dimuat dalam bukunya. Didalam buku tersebus dijelaskan tentang adanya keterikatan antara tulisan tangan seseorang dengan karakter dan kepribadiannya [6].

Lalu pada tahun 1741-1801, pastor swis, J.C Lavanter membuat laporan yang lebih sistematik dan cermat tentang tulisan tangan. Meskipun masih berupa perkiraan dan menerangkan tentang sifat-sifat umum, tetapi telah memberikan

inspirasi bagi orang lain, khususnya di Paris. Orang yang terinspirasi adalah Louis Hocquart. Selain itu, ada juga yang berasal dari kelompok agama, seperti Kardinal Regnier, Uskup Agung Cambrai, Uskup Boudinet, dan Flandrin [6].

Grafologi kemudian mengalami perkembangan pesat di Prancis dengan memunculkan tokoh-tokoh terdepan dalam bidang *grafologi,* diantaranya Jules Depoin, Binet G. Trade, dan Assene Aruss yang tergabung dalam "Himpunan Studi *Grafologi*". Namun, pusat perkembangan kemudian berpindah ke Jerman pada pertengahan abad XVIII dengan tampilnya Adlof Hentze, Sohwiedland, Gerhard Wilhem Langen Burch, dan Rudolphine Poppee [6].

2.1.2 Manfaat *Grafologi*

Berdasarkan dari sejarahnya, ilmu *grafologi* digunakan untuk mengenali karakter seseorang dari tulisan tangannya. Berikut adalah beberapa contoh dari manfaat *grafologi*:

1. Mengetahui *lifetrap*.
2. Mengetahui karakter orang lain.
3. Membantu kemampuan kita dalam berkomunikasi.
4. Identifikasi.
5. Sarana intropeksi diri.
6. Meningkatkan keahlian.
7. Mengungkap kasus kejahatan

2.1.3 Tanda Tangan menurut ilmu *Grafologi*

Tanda tangan merupakan atribut biometrik yang penting dari individu yang dapat digunakan sebagai identitas. Penggunaan tanda tangan merupakan cara yang alami dan tradisional sebagai identitas yang sah. Hal ini membuat keberadaan tanda tangan menjadi penting, sehingga diperlukan adanya sistem yang digunakan untuk memberi pengamanan supaya tidak disalah gunakan oleh pihak yang tidak bertanggung jawab.

Tanda tangan adalah hasil proses menulis seseorang yang bersifat khusus sebagai substansi simbolik. Tanda tangan merupakan bentuk yang paling banyak digunakan untuk identifikasi seseorang [7]. Pada umumnya pengenalan tanda tangan dilakukan secara manual oleh seseorang dengan mencocokkan secara

langsung tanda tangan yang sah dengan tanda tangan yang dilakukan saat ituContoh-contoh tanda tangan setiap orang umumnya identik namun tidak sama, artinya tanda tangan seseorang sering berubah-ubah setiap waktu. Perubahan ini menyangkut posisi, ukuran maupun faktor tekanan tanda tangan [5]. Kelemahan sistem manual dalam identifikasi tanda tangan adalah si pemeriksa tanda tangan harus teliti dalam melakukan pencocokan untuk menghindari kesalahan. Oleh karena itu untuk mengatasi kelemahan pencocokan tanda tangan secara manual, proses pencocokan tanda tangan perlu dilakukan secara otomastis dengan sistem komputer sehingga diharapkan mempermudah dalam identifikasi tanda tangan seseorang.

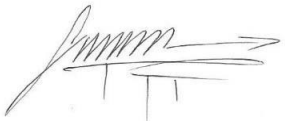
2.1.4 Fitur Tanda Tangan Dalam *Grafologi*

Analisis tanda tangan berdasarkan sembilan fitur untuk menggambarkan apakah terdapat kemunculan kepribadian tertentu dari 15 tipe yang ditinjau. Untuk lebih jelas, diperlihatkan pada Tabel 2.1, yaitu pola tanda tangan beserta kepribadian yang menyertai [1].

Tabel 2. 1 Fitur Pada Tanda Tangan

Fitur	Gambar	Ciri	Kepribadian
Awal Kurva		Lengkung mundur	Nyaman akan masalalu.
		Lengkung tajam	Mampu memformasikan pikiran secara tajam.
		Lengkung lembut	Hati-hati, ramah, diplomatis.
Coretan Akhir		Menaik	Terbuka, pandangan kedepan, keinginan maju, percaya diri.
		Menurun	Kurang semangat, berfikir realistis, kurang percaya diri, mudah putus asa.

Fitur	Gambar	Ciri	Kepribadian
Cangkang		Lengkung tertutup	Ketakutan berlebih, introvert, tidak memperdulikan sekitar, tidak suka bergaul dan tidak suka bekerja sama.
Coretan ditengah		Adanya coretan	Kurang percaya diri dan mudah depresi.
Garis bawah		Adanya garis bawah	Membutuhkan dukungan Dalam membuat keputusan, serta memiliki kemampuan dalam memimpin.
Margin ekstrim		Cendrung ke kanan	Ceroboh, kurang perhatian
		Cenderung ke kiri	Takut gagal, takut pada orang lain, kurang percaya diri, pesimis.
		Cenderung ke atas	Respek pada diri sendiri, Mencerminkan pribadi bahagia.
		Cenderung ke bawah	Depresi, pemalu, merasa asing.
Struktur titik		Tanda titik	Pendirian stabil, memiliki rasa curiga, selalu menjaga jarak tidak mudah percaya.

Fitur	Gambar	Ciri	Kepribadian
Tanda tangan terpisah		Tanda tangan terpisah	Memiliki pengalaman yang kurang menyenangkan di masa lalu.
Garis terpisah		Coretan akhir garis terpisah	Membatasi keiinginannya, tidak berani mengambil resiko, sering patah semangat dan ragu mengambil keputusan.

2.2 Pengolahan Citra *Digital*

Pengolahan citra digital adalah istilah umum untuk berbagai teknik yang keberadaannya untuk memanipulasi dan memodifikasi citra dengan berbagai cara. Setiap foto dengan bentuk citra digital dapat diolah menggunakan perangkat lunak tertentu. Foto adalah contoh gambar berdimensi dua yang dapat diolah dengan mudah.

2.2.1 Pengertian Citra

Manusia merupakan makhluk *visual* yang mengandalkan penglihatan untuk memahami sekelilingnya, dengan kemampuan tersebut manusia dapat merasakan dan mengetahui perbedaan sesuatu hal dengan cepat. Citra sendiri didefinisikan sebagai fungsi dari dua variabel misalnya $a_{x,y}$ di mana a merupakan amplitudo atau dalam hal ini ialah kecerahan, sedangkan x, y ialah koordinat yang membentuk citra [8].

Citra merupakan istilah lain untuk gambar pada salah satu komponen multimedia dengan karakteristik yang tidak dimiliki oleh data teks. Secara harfiah citra ialah gambar atau sebuah terusan cahaya pada bidang dwimatra (dua dimensi) yang ditinjau dari sudut pandang matematis. Cahaya yang menyinari suatu objek akan memberikan gambaran tentang objek tersebut, hal ini dikarenakan adanya pantulan cahaya saat sumber cahaya menyinari objek. Pantulan cahaya tersebut akan ditangkap oleh alat – alat optik, misalnya mata manusia, kamera, *scanner*, dan sebagainya [8].

Citra secara umum terbagi menjadi dua jenis, citra analog dan citra digital, dalam studi tentang pengolahan citra, citra yang digunakan pada penelitian ini merupakan citra digital yang dihasilkan dari mesin atau alat elektronik yang mampu menangkap gambar.

Pada penelitian ini, citra berperan sebagai data masukan yang akan berpengaruh terhadap proses dan hasil. Citra dengan kualitas yang buruk akan menurunkan akurasi pada klasifikasi. Oleh karena itu, guna memperoleh kualitas citra yang lebih baik, dilakukanlah pengolahan citra.

2.2.2 Pengolahan Citra

Citra atau gambar merupakan salah satu media yang dapat menyampaikan informasi lebih detil dan jelas bila dibandingkan dengan teks, penggunaannya pun dapat merepresentasikan suatu objek secara lebih nyata, namun tidak semua citra memiliki kualitas yang baik [8].

Dalam pengolahan citra digital terutama citra yang digunakan sebagai data masukan suatu proses, perlu memiliki kualitas yang baik. Hal–hal yang mengganggu kualitas citra contohnya adalah adanya bercak hitam atau putih yang bukan bagian dari objek pada citra.

Meskipun demikian, untuk memperoleh citra yang baik memerlukan alat optik yang bagus dengan harga yang cukup mahal. Alternatif dari penggunaan alat optik yang mahal adalah dilakukannya pengolahan citra dengan beragam macam metode, salah satunya ialah menggunakan konsep *smoothing.*

Pengolahan gambar atau pengolahan citra yang sering disebut *image processing.* Adalah suatu proses pengolahan gambar yang mengubah gambar menjadi gambar lain yang memiliki kualitas lebih baik untuk tujuan tertentu. Pengolahan citra memiliki dua tujuan utama, yaitu:

1. Memperbaiki kualitas citra, dimana citra yang dihasilkan dapat menampilkan informasi secara jelas. Hal ini berarti manusia sebagai pengolah informasi *(human perception).*
2. Mengekstrasi informasi ciri yang menonjol pada suatu citra, hasilnya adalah informasi citra dimana manusia mendapatkan informasi ciri dari citra secara numerik

Operasi-operasi pada pengolahan citra diterapkan pada citra bertujuan untuk:

1. *Image enhancement* adalah perbaikan citra yang dilakukan untuk meningkatkan kualitas penampakan citra atau menonjolkan beberapa aspek informasi yang terkandung dalam citra.
2. *Image restoration* adalah menghilangkan atau meminimumkan sebuah cacat yang ada pada sebuah citra.
3. *Image segmentation* adalah operasi citra yang perlu mengkelompokkan, mencocokan atau mengukur sebuah citra.
4. *Image analysis* adalah sebuah ektraksi ciri-ciri tertentu yang dimiliki pada sebuah citra untuk membantu dalam pengidentifikasian objek. Proses segementasi citra kadangkala diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya.
5. *Image reconstruction* adalah penggabungan dari beberapa citra dan menghasilkan sebuah citra baru.
6. *Image compression* adalah pemampatan sebuah citra, yaitu pengecilan ukuran citra.
7. *Steganografi* adalah penyembunyian sebuah data rahasia yang disimpan didalam sebuah citra sehingga keberadaan data tersebut tidak diketahui oleh orang lain.

Beberapa jenis operasi dari pengolahan citra adalah sebagai berikut [1]:

1. Modifikasi Kecemerlangan (*Brightness Modifikasi*)

   Mengubah nilai keabuan atau warna dari gelap menjadi terang ataupun sebaliknya mengubah citra yang terlalu terang/pucat menjadi gelap.

2. Peningkatan Kontras (*Contrast Enhancement*)

   Dengan meningkatkan kontras dari sebuah citra maka titik yang cenderung gelap menjadi lebih gelap dan yang cenderung terang menjadi lebih terang.

3. Negasi

   Operasi untuk mendapatkan citra negatif (*negative image*)+

4. Mengubah ukuran(*resize)*

   Resize atau penskalaan adalah sebuah operasi geometri yang digunakan untuk memperbesar atau memperkecil ukuran dari sebuah citra

sesuai dengan ukuran yang dibutuhkan. Pada peenskalaan apabila variabel penskalaannya bernilai lebih besar dari 1, maka ukuran citra akan diperbesar, namun apabila variabel penskalaannya bernilai lebih kecil dari 1 maka ukuran citra akan diperkecil.

Proses penskalaan dapat dilakukan dengan rumus [3]:

$$\boldsymbol{x} = \frac{\boldsymbol{pb} * \boldsymbol{pp}}{\boldsymbol{pa}} \quad (2.1)$$

Keterangan

$\boldsymbol{x}$ = Nilai piksel baris baru

$\boldsymbol{pb}$ = Ukuran panjang matriks baru

$\boldsymbol{pp}$ = Posisi piksel baris

$\boldsymbol{pa}$ = Ukuran panjang matriks lama

$$\boldsymbol{y} = \frac{\boldsymbol{lb} * \boldsymbol{lp}}{\boldsymbol{la}} \quad (2.2)$$

Keterangan

$\boldsymbol{y}$ = Nilai piksel kolom baru

$\boldsymbol{lb}$ = Ukuran lebar matriks baru

$\boldsymbol{lp}$ = Posisi piksel kolom

$\boldsymbol{la}$ = ukuran lebar matriks lama

5. Pengabuan (*Grayscale*)

Citra *grayscale* menangani gradasi warna hitam dan putih yang menghasilkan efek warna abu-abu [9]. Gradasi warna pada setiap gambar dinyatakan dengan intensitas. Dalam hal ini, intensitas berkisar diantara 0 sampai dengan 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih. Pada *grayscale* nilai intensitas tersebut dapat diseragamkan dengan suatu fungsi. Berikut ini adalah rumus konversi citra berwana (RGB) menjadi nilai intensitas *grayscale*.

$$\boldsymbol{Grayscale} = 0{,}2989 * \boldsymbol{R} + 0{,}5870 * \boldsymbol{G} + 0{,}1140 * \boldsymbol{B} \quad (2.3)$$

6. Pengambangan (*Thresholding*)

Metode thresholding adalah citra biner dengan nilai intensitas piksel sebesar 0 atau 1. Setelah citra sudah tersegmentasi atau sudah berhasil dipisahkan objeknya dengan background, maka citra biner yang diperoleh dapat dijadikan sebagai masking utuk melakukan proses cropping

sehingga diperoleh tampilan citra asli tanpa background atau dengan background yang dapat diubah-ubah [3].

7. Pencerminan (*Flipping*)

   Pencerminan merupakan proses menggambar citra ke bentuk kebalikan dari citra tersebut, seperti ketika orang sedang bercermin.

8. Rotasi (*Rotating*)

   Rotasi adalah proses memutar koordinat citra sesuai dengan derajat yang ditentukan

9. Pemotongan (*Cropping*)

   Memotong bagian yang diinginkan dari sebuah citra

10. Penskalaan (*Scaling*)

    Mengubah ukuran citra menjadi lebih besar atau lebih kecil

11. Deteksi Tepi (*Edge Detection*)

    Deteksi tepi adalah suatu proses yang menghasilkan tepi-tepi dari obyek dalam sebuah citra. Algoritma *Canny* berjalan dalam 5 langkah yang terpisah yaitu:

    a. Penghalusan (*Smoothing*): Menggabungkan gambar untuk menghilangkan noise.
    b. Mencari Gradien (*Finding gradient*): Tepian harus ditandai pada gambar memiliki gradien yang besar.
    c. *Non-maksimum-suppresion*: Hanya maxima lokal yang harus ditandai *Ende*
    d. *Double thresholding*: Tepian yang berpotensi ditentukan oleh *thresholding*.
    e. *Edge tracking by hysteresis*: Tepian final ditentukan dengan menekan semua sisi yang tidak terhubung dengan tepian yang sangat kuat

Tahap dateksi tepi canny adalah sebagai berikut:

1. Langkah pertama pada deteksi tepi Canny adalah mengkonvolusikan hasil matriks *Grayscale* dengan tapis Gaussian.

$$\frac{1}{115}\begin{bmatrix}2 & 4 & 5 & 4 & 2\\ 4 & 9 & 12 & 9 & 4\\ 5 & 12 & 15 & 12 & 5\\ 4 & 9 & 12 & 9 & 4\\ 2 & 4 & 5 & 4 & 2\end{bmatrix} \quad (2.4)$$

2. Kemudian menghitung nilai tepi menggunakan operator sobel menggunakan rumus

$$\boldsymbol{G = |Gx| + |Gy|} \quad (2.5)$$

$$\boldsymbol{Mx = \begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}, My = \begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}} \quad (2.6)$$

3. Setelah menemukan tepi dengan operator sobel, maka langkah selanjutnya adalah menentukan arah tepi dengan persamaan sebagai berikut:

$$\boldsymbol{\theta = \arctan(|Gy|/|Gx|)} \quad (2.7)$$

4. Setelah arah tepi diperoleh selanjutnya mengkonversi arah tepi kedalam salah satu kategori 0, 45, 90, 135 derajat.
5. Kemudian dilakukannya penghilangan nonmaksimum, piksel yang tidak dianggap tepi dapat diubah menjadi 0.
6. Kemudian dilakuaknnya pengambangan hysteresis dengan melibatkan dua ambang, T1 yaitu 125 dan T2 yaitu 255.

7. Kemudian dilakukan threshold dengan persamaan sebagai berikut:

$$\boldsymbol{f(x,y) = \begin{cases}1\ if(x,y\ \geq 255\\ 0\ if(x,y\ < 255\end{cases}} \quad (2.8)$$

Metode *edge detection* akan mendeteksi semua *edge* atau garis-garis yang membentuk objek gambar dan akan memperjelas kembali pada bagian-bagian tersebut. Tujuan pendeteksian ini adalah bagaimana agar objek di dalam gambar dapat dikenali dan disederhanakan bentuknya dari bentuk sebelumnya. Metode *Canny edge detection* merupakan pengembangan dari metode dasar *edge detection*. Perancangan sebuah prosedur dengan menerapkan langkah-langkah metode *Canny edge detection* akan menghasilkan sebuah tampilan gambar yang berbeda dengan menampilkan efek relief didalamnya. Efek relief adalah seperti sebuah tampilan batu kasar yang diukir, yaitu garis-garis kasar yang membentuk sebuah penggambaran objek di dalamnya. Efek relief terbentuk dari bayangan terang dan gelap. Kedua bayangan ini terjadi akibat adanya sorotan sinar mengenai gambar dari arah tertentu. Kelebihan dari metode *Canny* ini adalah kemampuan untuk mengurangi *noise* sebelum melakukan perhitungan deteksi tepi sehingga tepi-tepi yang dihasilkan lebih banyak [2].

2.2.3 Segmentasi

Dalam pengolahan citra, terkadang kita menginginkan pengolahan hanya pada obyek tertentu. Oleh sebab itu, perlu dilakukan proses segmentasi citra yang bertujuan untuk memisahkan antara objek (*foreground*) dengan *background*. Pada umumnya keluaran hasil segmentasi citra adalah berupa citra biner di mana objek (*foreground*) yang dikehendaki berwarna putih (1), sedangkan *background* yang ingin dihilangkan berwarna hitam (0). Sama halnya pada proses perbaikan kualitas citra, proses segmentasi citra juga bersifat eksperimental, subjektif, dan bergantung pada tujuan yang hendak dicapai.

Pada proses ini citra akan dipotong menjadi dua area untuk segmentasi *vertikal* dengan ukuran piksel [25x100] supaya mendapatkan fitur kurva awal dan coretan akhir, sementara dua area untuk segmentasi *horizontal* dengan piksel [50x20] supaya mendapatkan coretan tengan dan garis bawah.

2.2.4 *Two Dimensional Linear Discriminant Analysis* (2D-LDA)

Metode 2D-LDA merupakan pengembangan dari metode LDA. Alur kerja dari metode 2D-LDA hampir sama dengan metode LDA, perbedaan utama antara

metode LDA dan 2D-LDA adalah cara representasi data, jika LDA data direpresentasikan menjadi *vector* atau matriks satu dimensi maka metode 2D-LDA direpresentasikan menjadi matriks dua dimensi. Langkah pertama adalah mencari *mean* dari beberapa data, misal ada dua data citra x1 dan x2 dengan persamaan 2.9 [7] [10]

$$\boldsymbol{\mu_1} = \frac{\mathbf{1}}{\boldsymbol{N_1}} \sum_{\boldsymbol{x\epsilon\omega_1}} \boldsymbol{X}, \ \boldsymbol{\mu_2} = \frac{\mathbf{1}}{\boldsymbol{N_2}} \sum_{\boldsymbol{x\epsilon\omega_2}} \boldsymbol{X} \tag{2.9}$$

Persamaan 2.10 digunakan untuk mencari matriks *covarian* dari kedua data setelah mean didapatkan [7].

$$\boldsymbol{S_i} = \sum_{\boldsymbol{x\epsilon\omega_i}} (\boldsymbol{x} - \boldsymbol{\mu_i})(\boldsymbol{x} - \boldsymbol{\mu_i})^{\boldsymbol{\tau}} \tag{2.10}$$

Kemudian setelah mendapatkan matriks *covarian*, langkah selanjutnya mencari *within class scatter matrix* dengan persamaan 2.11 [7].

$$\boldsymbol{S_w} = \boldsymbol{S_1} + \boldsymbol{S_2} \tag{2.11}$$

Persamaan 2.12 digunakan untuk mencari *between class scatter matrix* [7].

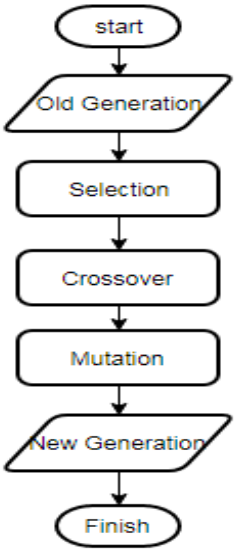
$$\boldsymbol{S_B} = (\boldsymbol{\mu_1} - \boldsymbol{\mu_2})(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})^{\boldsymbol{\tau}} \tag{2.12}$$

Kemudian membentuk matriks proyeksi (W) yang digunakan untuk proyeksi data baru (data testing) menggunakan persamaan 2.13 [7].

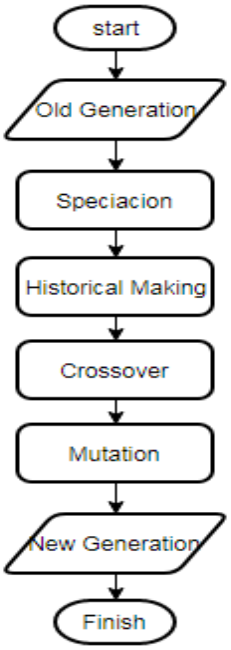
$$\boldsymbol{S_w^{-1} S_B} = \boldsymbol{\lambda W} \tag{2.13}$$

2.2.5 NeuroEvolution of Augmenting Topologies (NEAT)

Algoritma NEAT (NeuroEvolution of Augmenting Topologies) adalah algoritma neuroevolution yang dikembangkan di University of Texas di Austin 2002 [11]. Ini adalah sebuah algoritma genetik dan juga salah satu algoritma jaringan saraf konstruktif yang paling populer. Jaringan ini pada dasarnya merupakan lgortma genetik(GA) dimana beberapa struktur dan aturannya dilakukan penyempurnaan dan dikombinasikan dengan sistem JST yang terkendali sehinga meningkatkan efektifitas dari proses yang terjadi dalam JST tersebut. Secara umum perbedaan NEAT dengan GA dapat digambarkan pada *flowchart* dibawah ini.



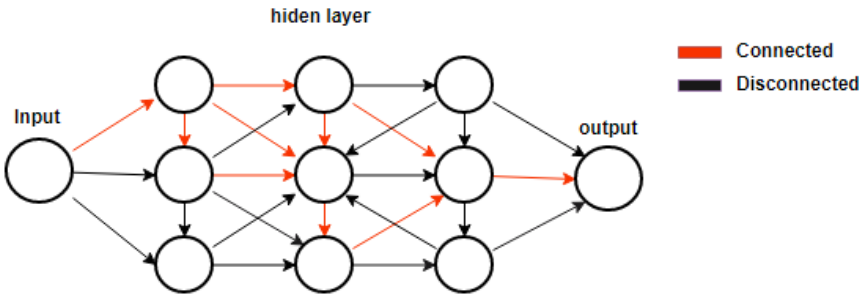
Gambar 2. 1 *Flowchart* Algoritma Genetik



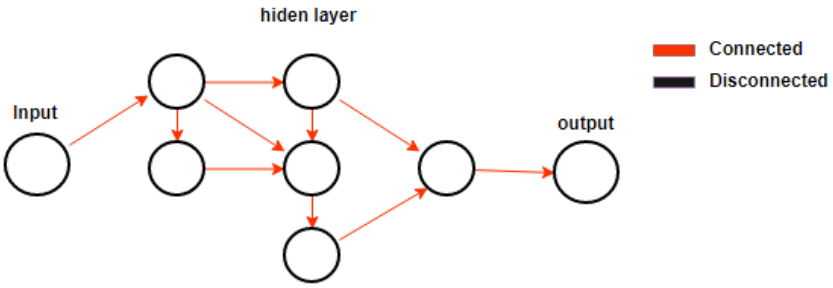
Gambar 2. 2 *Flowchart NEAT*

2.2.6 Struktur JST NEAT

Sebagaimana yang telah dijelaskan sebelumnya, NEAT dikombinasikan dengan JST yang terkendali, sktuktur koneksi yang dibentuk berdasarkan informasi dalam sebuah *Gnome*, ini memungkinkan sebuah JST terus berkembang seiring kompleksitas yang terjadi namun dengan koneksi jaringan se-efektif mungkin. berbanding terbalik dengan NEAT, JST pada GA merupakan gambaran keseluruhan kompleksitas yang mungkin terjadi sehingga tidak jarang koneksi dan *node* dalam JST tersebut tidak digunakan. pada beberapa kasus JST GA harus dilakukan perancangangan ulang seiring kompleksitas dari sistem tersebut. Dibawah ini gambaran JST pada NEAT dan GA yang mengambarkan koneksi yang terjadi pada sebuah kasus.



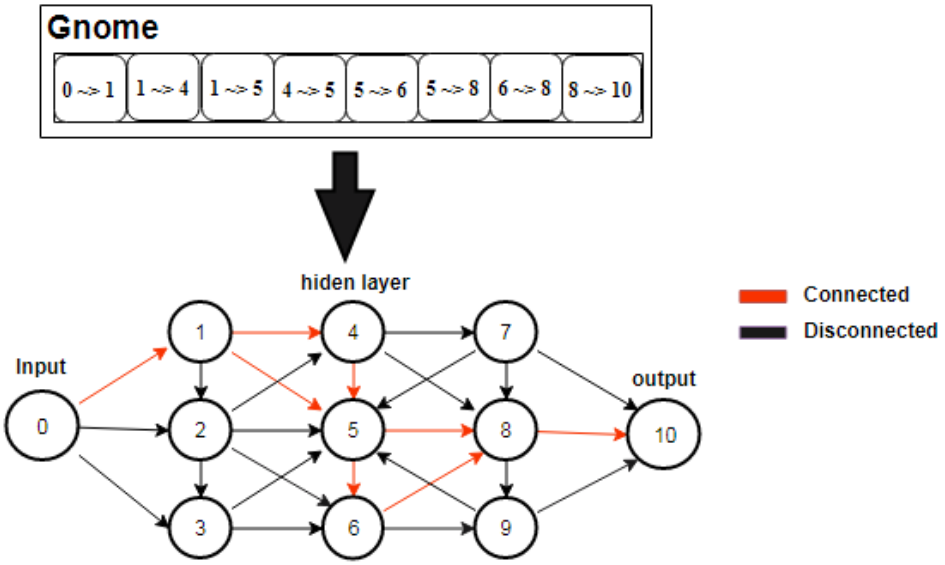
Gambar 2. 3 *JST* pada *GA*



Gambar 2. 4 *JST* pada NEAT

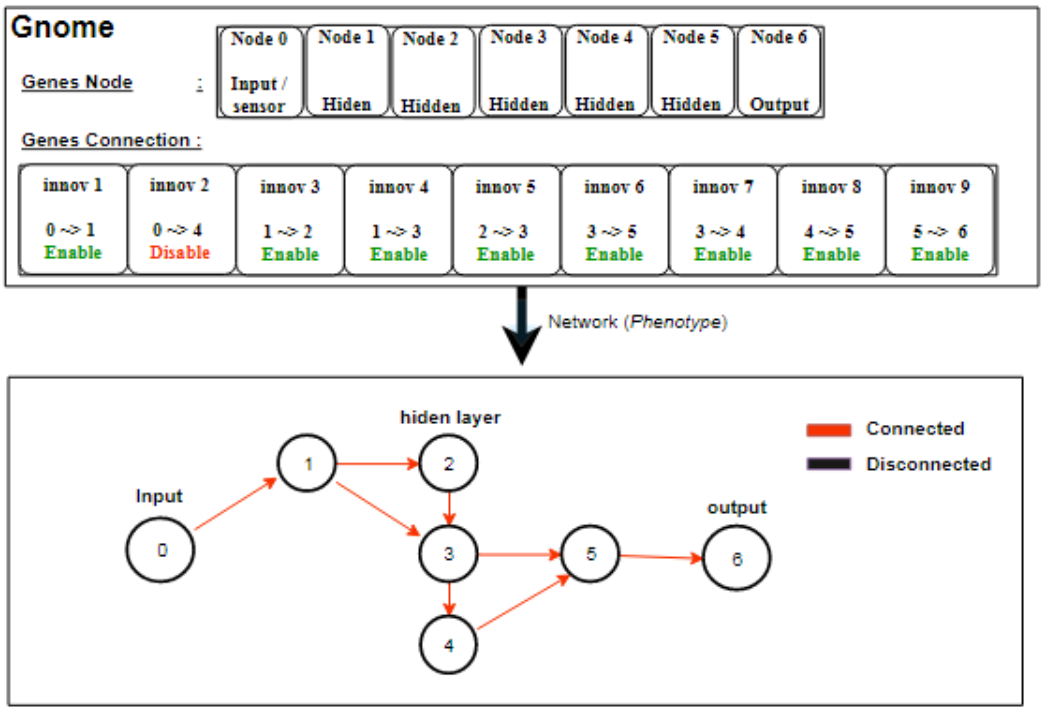
2.2.6.1 Struktur *Gnome* Pada NEAT

Gnome merupakan inti dari sebuah algoritma genetik, *Gnome* berisikan sebuah *genes* yang berisikan informasi dari setiap individu dalam sebuah populasi yang mengambarkan perubahan dan sifat setiap generasi yang ada[11]. Pada pola JST sendiri, Gnome mengambarkan koneksi (*Phenotype*) yang terjadi pada setiap layer atau *node* berdasarkan bobot yang diterima. Berikut ini gambaran umum *Gnome* dari individu pada JST GA.



Gambar 2. 5 *Gnome* Pada *JST GA*

Gnome pada NEAT memiliki dua *genes*, *genes* pertama berisikan informasi *node* yang digunakan, dan *genes* kedua berisikan informasi koneksi *gene* berdasarkan bobot, informasi ketersediaan koneksi, dan nomor inovasi . selain itu *Gnome* pada NEAT juga berfungsi sebagai pembentuk struktur JST (*Phenotype*). Dibawah ini gambaran umum dari *Gnome* individu pada salah satu generasi populasi NEAT.



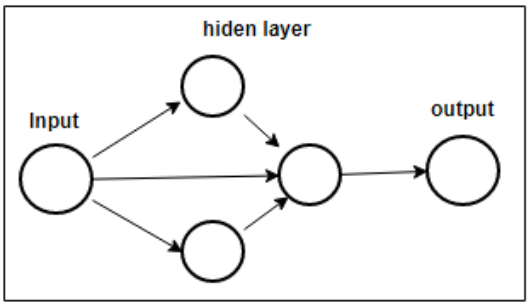
Gambar 2. 6 *Gnome* Pada JST NEAT

2.2.6.2 Struktur Koneksi Pada NEAT

Struktur jaringan NEAT dinilai sangat baik dikarenakan mampu melakukan koneksi dan juga penghitungan atau penilaian suatu bobot pada *node* secara bersamaan meskipun sistem masih dalam proses *learning* [11]. Secara umum koneksi dalam *NEAT* terbagi tiga jenis yaitu sebagai berikut [12].

1. **Forward Link**

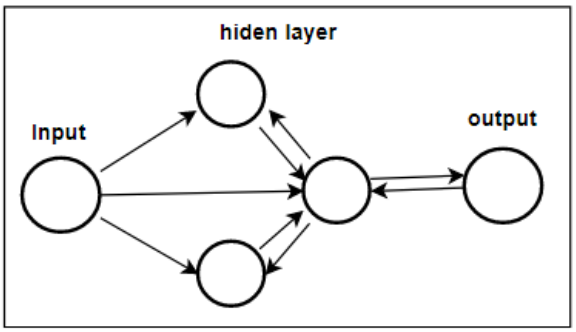
Forward Link merupakan koneksi yang terjadi antara *node* dengan *node* lain berbentuk *continue* atau lurus. Dibawah ini adalah gambaran dari jenis koneksi *Forward Link*.



Gambar 2. 7 Koneksi *Forward Link*

2. Recurrent Link

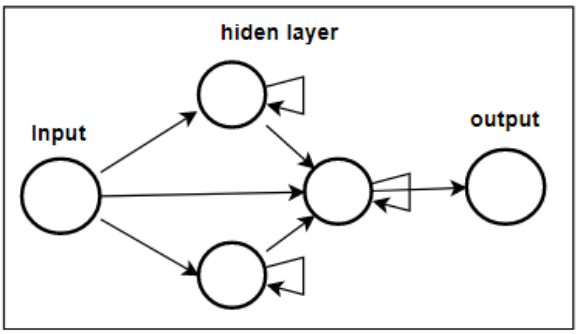
Recurrent Link merupakan koneksi yang terjadi antara *node* dapat terjadi maju dan mundur ke *node* sebelumnya. Koneksi *Recurrent Link* secara umum dapat digambarkan sebagai berikut.



Gambar 2. 8 Koneksi *Recurent Link*

3. Self-Connected

Self-Connected merupakan koneksi yang terjadi berupa koneksi dari satu *node* ke *node* itu sendiri. Koneksi *Self-Connected* secara umum dapat digambarkan sebagai berikut.



Gambar 2. 9 Koneksi *Forward Link*

2.2.6.3 Struktur Umum Proses NEAT

Seperti yang telah dijelaskan sebelumnya NEAT merupakan turunan atau pembaharuan dari algoritma *Genetic Algorithm* (GA), secara umum struktur proses dari NEAT sendiri terdari dari empat proses sebagaimana yang telah digambarkan pada gambar 3 yaitu sebagai berikut :

1. Speciation

Speciation atau spesiesisasi merupakan pengelompokan populasi dalam sebuah generasi menjadi beberapa spesies berdasarkan kesamaan jaringan JST yang diciptakan sehingga memudahkan proses dari *crossover* atau proses penyilangan

genes sehingga menghasilkan *offspring* atau keturunan dengan *Gnome* yang lebih baik dari generasi sebelumnya.Proses *speciation* sendiri mempunyai fungsi sebagai berikut:

a) Melindungi inovasi pada *genes* individu sebelumnya yang dinilai baik tidak hilang.
b) Mencegah terjadinya proses *disjoint gene* berlebih pada proses *crossover* sehingga mencegah terjadinya *less compatible* (ketidak kompatibelan sebuah *Gnome* dalam sebuah generasi).
c) Mengoptimalkan perkembangan *Gnome* sebelum dilakukan *crossover* sehinga struktur *Gnome* pada *offspring* yang dihasilkan optimal.

Proses *speciation* sendiri dapat dilakukan dengan menghitung nilai kompatibel δ (*delta*) suatu struktur *Gnome.* Secara matematis penghitungan jarak tersebut dapat digambarkan dengan rumus dibawah ini :

$$\boldsymbol{\delta = \frac{c_1 \, \mathrm{E}}{N} + \frac{c2 \, D}{N} + C_3 \, . \, \overline{W}} \qquad (2.14)$$

Keterangan

δ = Nilai kompatibel delta

E = Jumlah *excess*

D = Jumlah *disjoint*

$\overline{W}$ = perbedaan bobot rata-rata *genes* yang cocok

C_1,C_2,C_3 = Koefesien

N = Jumlah *genes* pada Gnome paling besar

Dengan didapatnya nilai δ (delta) maka memungkinkan untuk menerapkan aturan aturan ambang batas (*threshold*) sehingga daftar urutan *species* dapat dipertahankan, dimana *Gnome* akan ditempatkan secara berurutan setiap generasinya. Setiap *species* diwakili oleh sebuah Gnome dari species sebelumnya yang didapat secara acak. *Gnome* yang diturunkan sebelumnya (***g)*** ditempatkan pada *species* pertama yang menunjukan (***g)*** diangap kompatible dengan *Gnome* yang mewakili species tersebut. Maka dengan ini, sebuah populasi yang terjadi tidak saling tumpng tindih satu sama lain. Jika (***g)*** tidak kompatible dengan *species* yang ada maka species baru diciptakan dengan (***g)*** sebagai *Gnome* yang mewakili species baru tersebut.

Dalam proses reproduksi, NEAT mengunakan metode *Explicit Fitness Sharing* (Goldberg and Richardson 1987). Dimana setiap individu dengan *species* yang sama saling berbagi *fitnes*(kesesuaian) pada lingkungan populasi, sehingga sebuah *species* tidak berkembang terlalu besar meskipun memiliki banyak individu yang memiliki kemampuan yang baik, ini bertujuan agar tidak terjadinya pengambil alihan seluruh populasi oleh suatu *species* yang melanggar konsep dari evolusi itu sendiri [11].

Penyesuain nilai *fitness* f_i' untuk setiap individu i dihitung berdasarkan jarak nilai δ (delta) dari setiap individu *j* dari sebuah populasi, penghitungan nilai fitness ini dapat digambarkan dengan rumus matematika sebgai berikut:

$$\boldsymbol{f_i' = \frac{f_i}{\left(\sum_{j-1}^{n}\right) sh(\delta(i,j))}} \qquad (2.15)$$

Keterangan

f_i^l = nilai *fitnesess*

sh = nilai sh = 0 jika nilai jarak ($\delta(i,j)$) melewati nilai *threshold* jika tidak sh = 1

i = individu.

j = jarak nilai δ dari setiap individu

n = jumlah populasi

Perhitungan nilai *fitneses* diatas bertujuan untuk mengurangi jumlah individu yang secara struktur *Gnome* mirip seperti individu i , hingga secara bertahap mengilangkan individu dengan kemampuan terendah yang kemudian secara perlahan juga mengantikan seluruh angota populasi menjadi individu *offspring* atau keturunan dari generasi sebelumnya.

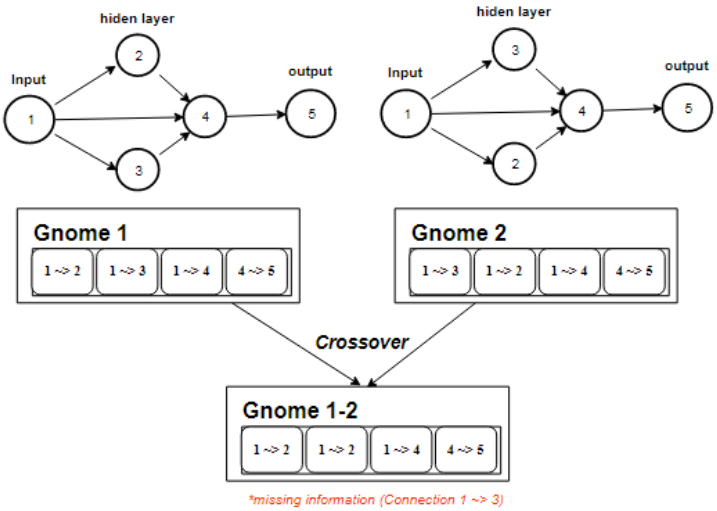
2. Historical Marking

Historical marking merupakan proses pencatatan dan juga pengecekan garis keturunan setiap individu. Dimana dua *genes* dengan *historical* yang sama harus mengambarkan struktur yang sama.

Setiap individu termasuk keturunannya (*offspring*) memiliki satu historical marking yang sama, data historical ini akan terus bertambah seiring terjadinya penambahan *inovation* pada *Gnome.* Sehingga mencegah terjadinya inovasi yang sudah pernah ada sebelumnya dibuat kembali namun dengan nomer inovasi berbeda.

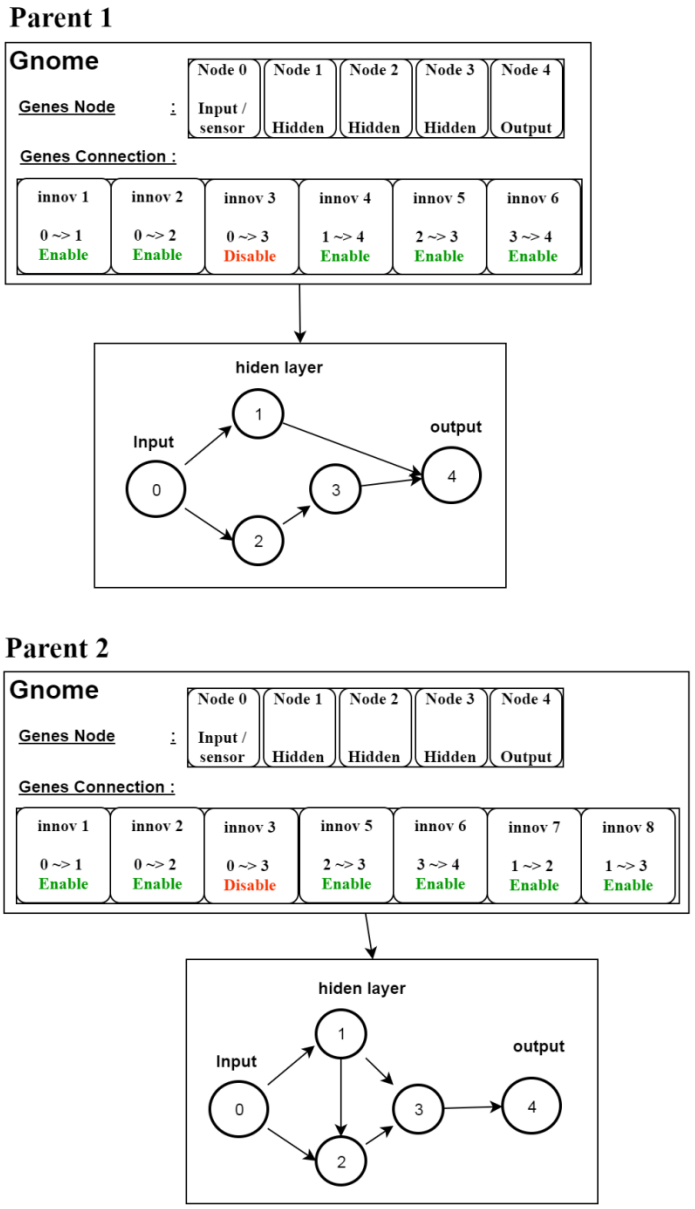
3. Crossover

Crossover merupakan proses penyilangan dua buah *Genome* yang cocok sehingga menghasilkan *offspring* atau keturunan yang merepresentasikan *gene* dari *parent.* Pada prosess algoritma GA proses *crossover* sering kali terjadi *missing information* pada *offspring* yang dihasilkan, dikarenakan terjadinya *incompatible crossover*. Hal ini terjadi bila dua *Gnome* memiliki isi *gene* yang sama namun memiliki penyusunan *gene* yang beda, dalam GA kedua *Gnome* tersebut akan dikatakan berbeda satu sama lain [7]. Salah satu contoh *error* ini adalah seperti yang terjadi proses *crossover* pada gambar dibawah ini.

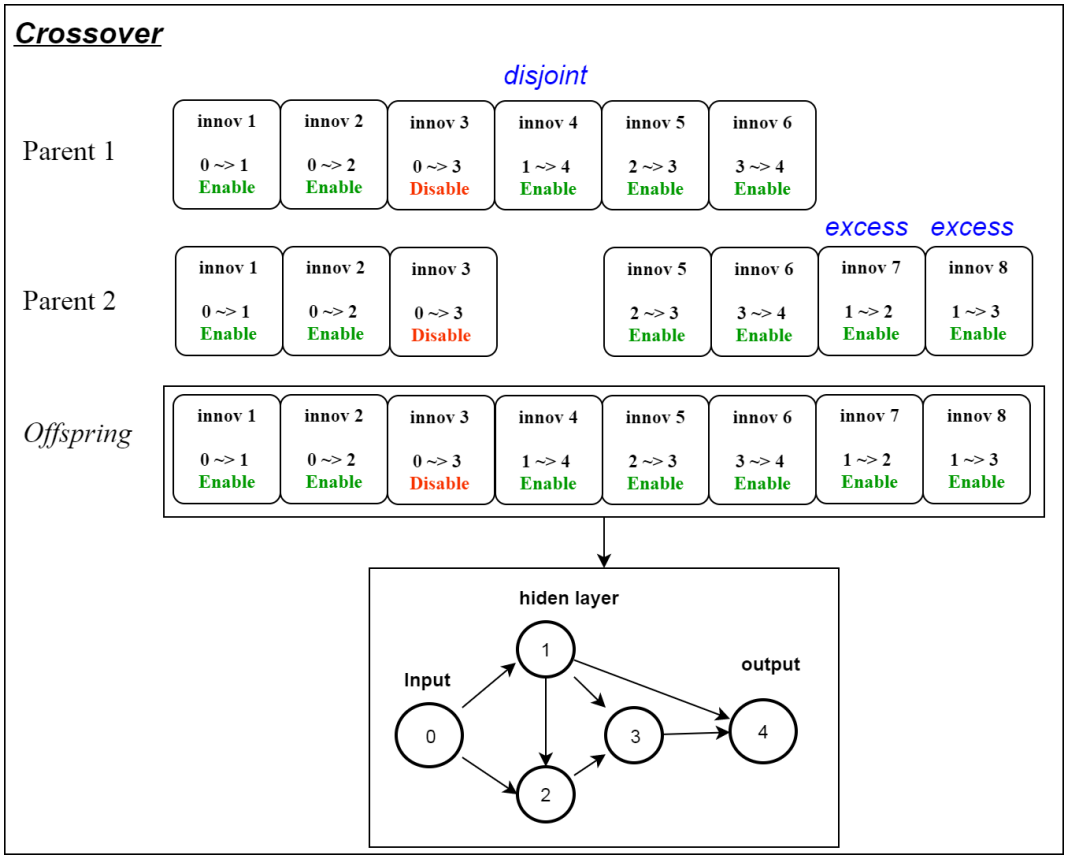


Gambar 2. 10 *Error crossover* pada GA

Pada algoritma NEAT dapat mencegah hal tersebut terjadi, dimana NEAT melakukan pengecekan *historical marking* sebelum proses *crossover* sehingga hanya dua Gnome yang benar benar berbeda dan cocoklah yang dapat dilakukan *crossover* [11]. Secara umum proses *crossover* NEAT dapat digambarkan seperti dibawah ini.



Gambar 2. 11 *Crossover* pada NEAT(1)



Gambar 2. 12 *Crossover* pada NEAT(2)

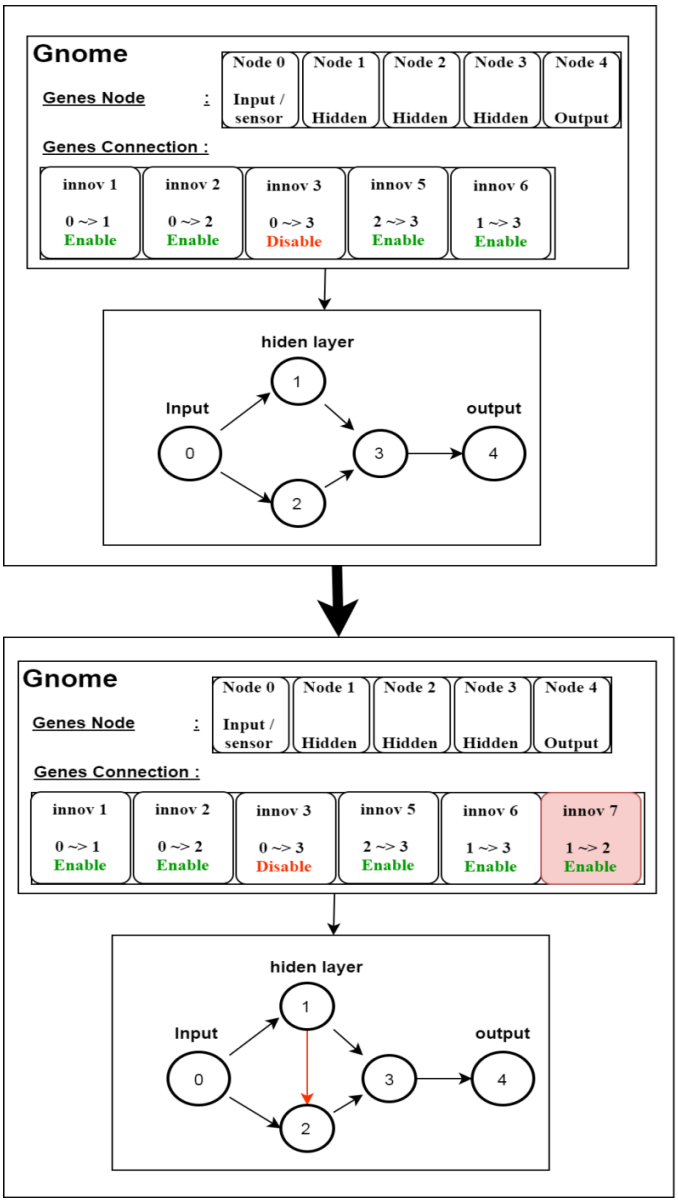
Pada gambar 2.12 proses crossover NEAT, *gene* yang cocok akan di turunkan secara acak, sedangkan *gene* yang terpisah (*disjoint*) dan juga *gene* yang berlebih (*excess*) akan di turunkan dari kedua parent. Jika kedua parent memilki nilai *fitness* yang sama atau sebanding, maka *disjoint* dan juga *excess* diturunkan secara acak [11].

4. Mutation

Mutation atau mutasi sebuah proses pada GA yang menciptakan individu baru dengan melakukan modifikasi pada *gene*. Proses mutation sendiri bertujuan untuk mengantikan gene yang terpisah(*disjoint*) dari sebuah populasi dan juga memberikan *gene* baru (*excess*) yang tidak tersedia pada generasi sebelumnya. Proses mutation dalan NEAT sendiri terjadi secara acak, dan terbagi menjadi dua yaitu sebagai berikut.

a) *Mutate Add Connection*

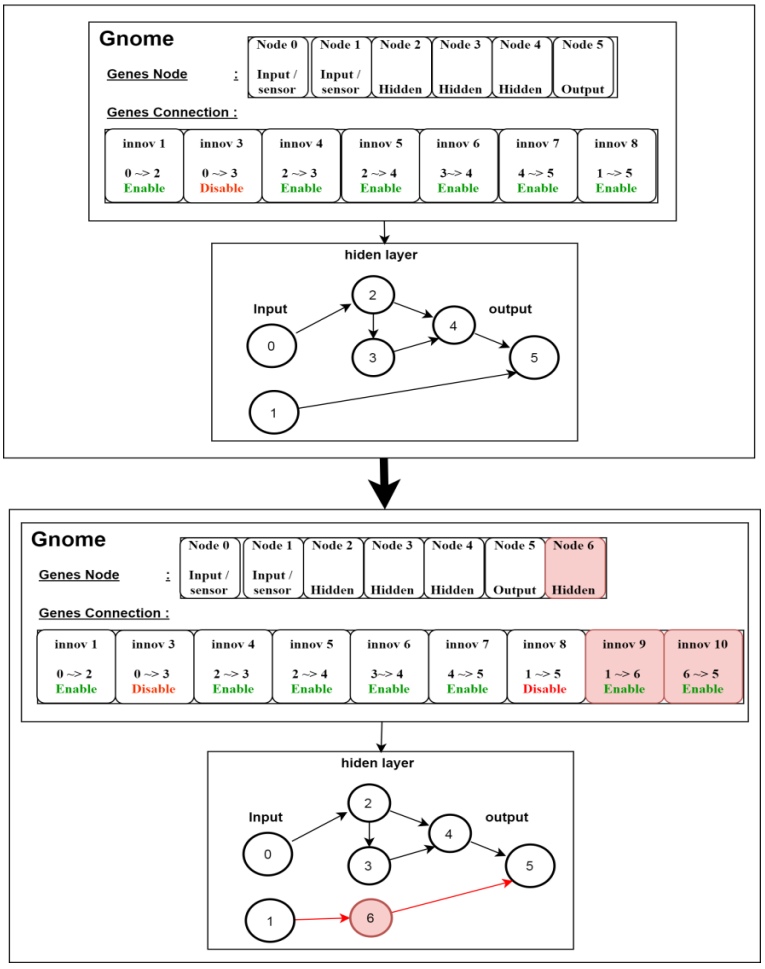
Mutate add Conection merupakan proses penambahan inovasi koneksi pada *connection genes*. *Gene* baru dibentuk dari proses koneksi node dengan *node* yang lain yang belum terhubung satu sama lain secara acak, kemudian *gene* tersebut ditambahkan pada akhir susunan *connection gene* dengan pemberian nomor *inovation* baru. Proses *mutation* ini dapat digambarkan sebagai berikut.



Gambar 2. 13 *Mutate add Conection* pada NEAT

b) *Mutate Add Node*

Mutate add Conection merupakan proses penambahan *node* pada *node genes*. Gene baru dibentuk dari pembagian sebuah *conection gene* yang aktif dengan memberikan *node* baru antara sebuah *connection gene*. Penambahan node secara umum diprioritaskan untuk *conection gene* yang berisikan koneksi langsung antara *input* dan *output node*, selain itu penambahan node dilakukan secara acak [11]. *Connection gene* yang dilakukan *split* atau pembagian akan dinon-aktifkan. *Connection gene* yang merepresentasikan koneksi *node* awal ke *node* baru diisi nilai bobot *connection gene* sebelumnya, dan *connection gene* yang merepresentasikan koneksi *node* baru ke *node* akhir diisi nilai bobot 1. Secara umum proses ini dapat digambarkan sebagai berikut:



Gambar 2. 14 *Mutate Add Node* pada NEAT

2.3 Perhitungan Akurasi

Perhitungan Akurasi digunakan untuk mengetahui akurasi terhadap hasil pengujian, dalam implementasinya pengujian Akurasi menggunakan metode *Confusion Matrix*. *Confusion matrix* merupakan salah satu metode yang digunakan untuk mengukur kinerja pada saat klasifikasi. Pengujian dilakukan pada 10 kelas, diantaranya:

1. Dataset tanda tangan berjumlah 560 data
2. Data latih sebanyak 60% dan data uji 40%

Keterangan dari data yang diujikan adalah sebagai berikut: pengujian pertama dengan menggunakan data uji 40% dari data latih yang belum dilatih sebelumnya sebanyak 336 data, sementara unt uk data latih menggunakan 60% dari data seluruhnya sebanyak 560 data.

Pada pengujian performansi menggunakan dua jenis proses klasifikasi diantaranya *True Positif* (TP) dan *False Negatif* (FN) untuk mendapatkan nilai akurasi yang digambarkan sebagai berikut:

Tabel 2. 2 *Confussion Matrix*

Perdiksi/Kelas Sebenarnya	**Kelas 1**	**Kelas 2**	**…**	**Kelas n**
Kelas 1	TP	FN	…	FN
Kelas 2	FN	TP	…	FN
…	…	…	…	…
Kelas n	FN	FN	…	TP

Berdasarkan nilai *True Positif* dan *False Negatif* yang kemudian didapatkan nilai akurasi. Nilai akurasi menggambarkan seberapa akurat sistem dapat mengklasifikasikan data secara benar, dengan kata lain, nilai akurasi merupakan perbandingan antara data yang terklasifikasi dengan benar dengan seluruh data yang diujikan. Dan berikut adalah perhitungan akurasi yang akan digunakan dalam proses pengujian akurasi, adalah sebagai berikut [13]:

$$\boldsymbol{Akurasi} = \frac{\boldsymbol{TP}}{(\boldsymbol{TP}+\boldsymbol{FN_1}+\cdots+\boldsymbol{FN_n})} \text{ x } 100\% \qquad (2.16)$$

2.4 *Flowchart*

Flowchart (Diagram Alir) adalah bagan (*Chart)* yang menunjukan alir (*flow*) di dalam program atau prosedur sistem secara logika. *Flowchart* merupakan metode untuk menggambarkan tahap-tahap pemecahan masalah dengan mempresentasikan simbol-simbol tertentu yang mudah dimengerti, mudah digunakan dan standar. Tujuan penggunaan *flowchart* adalah untuk menggambarkan suatu tahapan penyelesaian masalah secara sederhana, terurai, rapi, dan jelas dengan menggunakan simbol-simbol yang standar. Tahapan penyelesaian masalah yang disajikan harus jelas, sederhana, dan tepat.

Pada waktu akan menggambar suatu bagan alir, *programmer* dapat mengikuti pedoman-pedoman sebagai berikut:

a. Bagan alir sebaiknya digambarkan dari atas ke bawah dan mulai dari bagian kiri suatu halaman.
b. Kegiatan di dalam bagan alir harus ditunjukkan dengan jelas.
c. Harus ditunjukkan dari mana kegiatan akan di mulai dan di mana akan berakhirnya.
d. Masing-masing kegiatan di dalam bagan alir sebaiknya digunakan kata yang mewakili suatu pekerjaan.
e. Masing-masing kegiatan di dalam bagan alir harus di dalam urutan yang semestinya.
f. Kegiatan yang terpotong dan akan disambung di tempat lain harus ditunjukkan dengan jelas menggunakan simbol penghubung.
g. Gunakan simbol-simbol bagan alir standar.

2.5 Diagram Konteks

Diagram konteks adalah diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. Diagram konteks merupakan level tertinggi dari *data flow diagram* yang menggambarkan seluruh masuk ke sistem atau keluar dari sistem. Ia akan memberi gambaran tentang keseluruhan sistem. Sistem dibatasi oleh *boundary* (dapat digambarkan dengan garis putus). Dalam diagram konteks hanya ada satu proses dan tidak boleh ada *store* dalam diagram konteks.

2.6 *Data Flow Diagram* (DFD)

Diagram yang menggambarkan proses dari data *flow* diagram. *Data flow diagram* atau yang sering disebut DFD memberikan pandangan secara menyeluruh mengenai sistem yang ditangani, menunjukan fungsi-fungsi utama atau proses yang ada, aliran data, *external entity*. DFD adalah *diagram* yang menguraikan proses apa yang ada dalam diagram konteks.

2.7 Kamus Data

Kamus data berfungsi membantu pelaku sistem untuk mengartikan aplikasi secara detail dan mengorganisasi semua elemen data yang digunakan dalam sistem secara persis sehingga pemakai dan penganalisa sistem memounyai dasar pengertian yang sama tentang masukan, keluaran, penyimpanan dan proses. Kamus data sering disebut juga dengan sistem *data dictionary* adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Atribut yang berfungsi sebagai *key* juga dibedakan dengan yang bukan *key* dengan menggarisbawahi atribut tersebut

2.8 Pengujian *BlackBox*

BlackBox Testing berfokus pada spesifikasi fungsional dari perangkat lunak. Tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. *BlackBox Testing* bukanlah solusi alternatif dari *White Box Testing* tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh *White Box Testing*.