BAB II
LANDASAN TEORI

2.1 Biometric

Biometric merupakan suatu metode untuk mengenali individu berdasarkan ciri fisik maupun perilaku. *Biometric* dapat digunakan dalam proses otentikasi maupun identifikasi (Shrikant, 2012). Dibandingkan dengan proses otentikasi dengan metode lama seperti *password*, PIN, dan *token*, penggunaan *biometric* lebih *reliable* karena selain tidak sulit ditiru, *biometric* tidak dapat hilang atau terlupa.

Biometric juga terbagi atas 3 tipe, *Biological, Behaviorual,* dan *Morphological* (Julien Mahier, 2009) antara lain sebagai berikut.

1. *Biological*, data dari *biological* didapatkan dari segala sesuatu yang melekat dalam tubuh manusia atau makhuk hidup seperti DNA, darah, dan bau. Secara umum, data dari *biological* sulit didapatkan secara langsung, membutuhkan alat dan teknik khusus.
2. *Behavioural*, cara untuk mendapatkan data *behavioral* dilihat dari perilaku individu dalam melakukan sesuatu atau ciri khas seseorang seperti suara, gerak langkah kaki, pola tekan *keyboard*, dan tanda tangan. Salah satu kesulitan yang ditemukan adalah dikarenakan berdasarkan sifat, data yang didapatkan mudah berubah seiring waktu.
3. *Morphological*, pendekatan paling umum dalam *biometric* dengan melihat bagian pada tubuh manusia. Mata, bentuk telinga, iris, sidik jari, dan bentuk wajah dapat dijadikan acuan dalam mengidentifikasi seseorang. Salah satu kesulitan yang ditemukan dalam tipe ini adalah perkembangan pada beberapa organ tubuh dari segi bentuk dan panjang.

Beberapa hal yang harus diperhatikan dalam proses *biometric identifier* (Shrikant, 2012) antara lain sebagai berikut.

1. *Universality* : Setiap orang memiliki ciri biometric tersebut
2. *Distinctiveness* : Antar individu satu dan lainnya dapat dibedakan berdasarkan ciri *biometric*-nya.
3. *Permanence* : Ciri *biometric* tersebut tidak berubah dalam kurun waktu tertentu.
4. *Collectability* : Ciri *biometric* dapat diukur dengan suatu metode.
5. *Performance* : Kecepatan, tingkat akurasi, dan sumber daya yang dibutuhkan selama proses pengenalan.
6. *Acceptability* : Tingkat penerimaan *user* terhadap ciri *biometric* tersebut.
7. *Circumvention* : Seberapa mudah sistem pengenal *biometric* dapat dimanipulasi dengan cara ilegal.

Dalam melakukan pengujian terhadap performa sebuah sistem biometric diperlukan perhitungan statistika antara lain sebagai berikut (Tiwari, 2012).

1. *False Acceptence Rate* (FAR)

FAR merupakan sebuah kemungkinan dari seorang user yang melakukan klaim terhadap user lain dan akan diterima atau diverifikasi oleh sistem terhadap user lain tersebut. Rumus 2.1 menunjukkan perhitungan nilai FAR terhadap satu user.

$$FAR(n) = Non\ Match\ User'(n) \ / \ Non\ Match\ User(n)$$

… Rumus 2.1

Non Match User'(n) merupakan jumlah dari user yang tidak seharusnya diterima berhasil diautentikasi sebagai user *n*, sedangkan *Non Match User(n)* adalah jumlah percobaan autentikasi terhadap user *n*.

Rumus 2.2 menunjukkan perhitungan FAR untuk keseluruhan sistem.

$$FAR = \frac{1}{N} \sum_{n=1}^{N} FAR\ (n)$$

… Rumus 2.2

Berdasarkan Rumus 2.2, nilai FAR didapat dari jumlah FAR setiap *user* dibagi dengan jumlah *user* (N).

2. *False Rejection Rate* (FRR)

FRR merupakan kemungkinan seorang *user* yang mengklaim dirinya sebagai *user* tersebut ditolak oleh sistem. Rumus 2.3 menunjukkan perhitungan FAR dari masing-masing *user*.

$$FRR(n) = Non\ Match\ User'(n)\ /\ Non\ Match\ User(n)$$

… Rumus 2.3

Non Match User'(n) merupakan jumlah *user* yang seharusnya diterima, ditolak oleh sistem, sedangkan *Non Match User(n)* adalah jumlah percobaan yang dilakukan oleh *user* tersebut.

Untuk perhitungan nilai FRR secara kesuluruhan, digunakan perhitungan yang ditunjukan pada Rumus 2.4.

$$FRR = \frac{1}{N} \sum_{n=1}^{N} FRR\ (n)$$

… Rumus 2.4

Seluruh hasil perhitungan FRR dari masing-masing *user* dijumlahkan kemudian dibagi terhadap jumlah *user* (N).

3. *Genuine Acceptance Rate* (GAR)

Menurut Christopher Grant Andrade, pada penelitian yang berjudul Investigating and comparing multimodal biometric techniques tahun 2009. GAR merupakan pengukuran tingkat akurasi dari keseluruhan sistem *biometric* dengan rumus yang ditunjukan pada Rumus 2.5.

$$GAR = 1 - FRR$$

… Rumus 2.5

2.2 Keystroke Dynamic

Keystroke dynamic adalah suatu proses menganalisa cara seseorang mengetik dengan melihat ritme pengetikan setiap detiknya (Monrose, 2000). Beberapa hal yang digunakan dalam menganalisa pola pengetikan adalah kecepatan mengetik, waktu tekan terhadap *tuts keyboard*, waktu tunggu antar penekanan satu *tuts* dan *tuts* lainnya, tingkat koreksi yang dilakukan, dan waktu tunggu antar penekanan *tuts* dibeberapa kombinasi huruf.

Verifikasi *keystroke* dapat dilakukan dengan 2 pendekatan yaitu, *static* yang mana sistem hanya akan melakukan pengecekan pola ketik pada saat tertentu seperti saat mengetik *password* ketika *login*, sedangkan *continuous* adalah pendekatan yang melakukan pengecekan sepanjang sesi *user* tersebut menggunakan sistem.

Untuk melakukan analisa pola pengetikan, Rumus 2.6 menunjukkan *metric* untuk menghitung kecepatan pengetikan (*Words per minute*) (John Paulin, 2012).

$$WPM = \frac{Number\ of\ Characters}{Time\ Spent\ for\ Typing\ (minute) \times 5}$$

… Rumus 2.6

Penghitungan waktu tekan *tuts keyboard* akan diambil dari data yang berasal dari *keyboard*. Penghitungan waktu tekan ditunjukan pada Rumus 2.7 (Douhou, 2009).

$$D_i = R_i - P_i$$

… Rumus 2.7

Dwell time didefinisikan sebagai D. Nilai Dwell time akan didapatkan dari waktu lepas yang didefinisikan sebagai R ke-i dikurang P ke-I yang merupakan waktu tekan. Untuk perhitungan waktu tunggu antar *tuts* dijelaskan pada Rumus 2.8(Douhou, 2009).

$$F_i = P_i - P_{i-1}$$

… Rumus 2.8

Flight time didefinisikan sebagai F yang akan didapat dari perhitungan Press yang didefinisikan sebagai P ke-i dikurang P ke-i-1. Untuk memperoleh nilai dari banyaknya seorang user melakukan koreksi ketika mengetik suatu kalimat digunakan rumus yang terdapat pada Rumus 2.9.

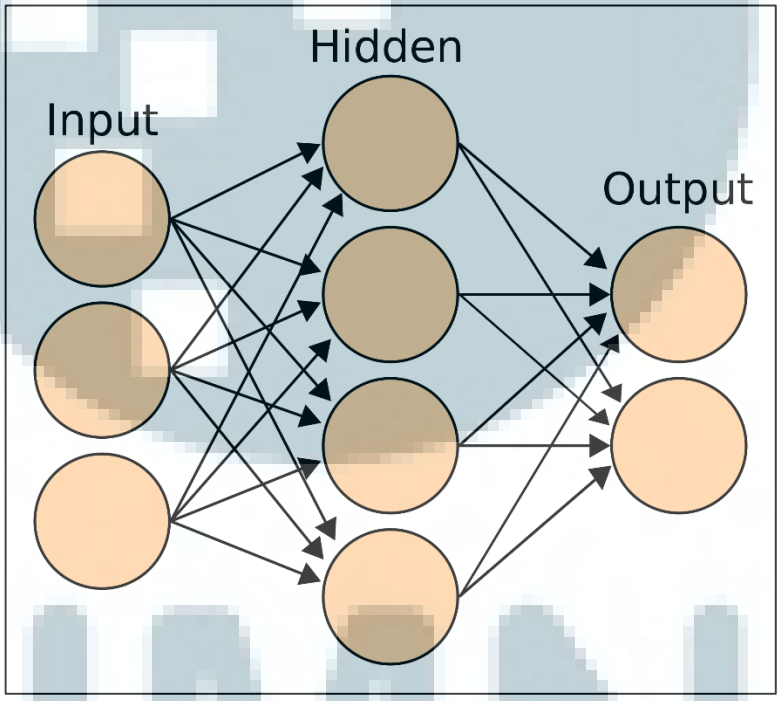
$$Rate\ of\ Backspace\ Activation = \frac{Number\ of\ Keystroke\ for\ backspace}{Number\ of\ characters\ Typed}$$

… Rumus 2.9

Jumlah dari banyaknya user menekan backspace untuk mengkoreksi kata atau kalimat dibagi dengan jumlah karakter yang diketik maka akan didapatkan nilai *backspace activation.*

2.3 Jaringan Saraf Tiruan

Pada tahun 1943, Warren S. McCulloch dan Walter Pitts membuat konseptual model pertama dari *artificial neural network* (Shiffman, 2012). Konseptual tersebut menjelaskan *neuron*, sebuah *cell* yang berada pada jaringan saraf yang menerima *input*, memproses *input* tersebut, dan menghasilkan *output*. Setiap jaringan saraf tiruan memiliki arsitektur yang berbeda, mulai dari jumlah *neuron/cell* hingga jumlah keluaran. Gambar 2.1 menunjukkan salah satu contoh dari arsitektur jaringan saraf tiruan dengan 4 *neuron* pada *input layer*, 4 *neuron* pada *hidden layer*, dan 1 *neuron* pada *output layer*.



Gambar 2.1 Arsitektur Jaringan Saraf Tiruan
(Sumber:http://natureofcode.com/book/chapter-10-neural-networks/)

Setiap arsitektur memiliki *input layer*, *output layer*, dan *hidden layer*. Masing-masing *layer* terhubung yang mana menunjukkan jalur data atau informasi. Hubungan tersebut memiliki nilai yang disebut dengan *weight*. *Weight* atau bobot berfungsi untuk mengontrol dari *neuron* satu ke *neuron* lainnya.

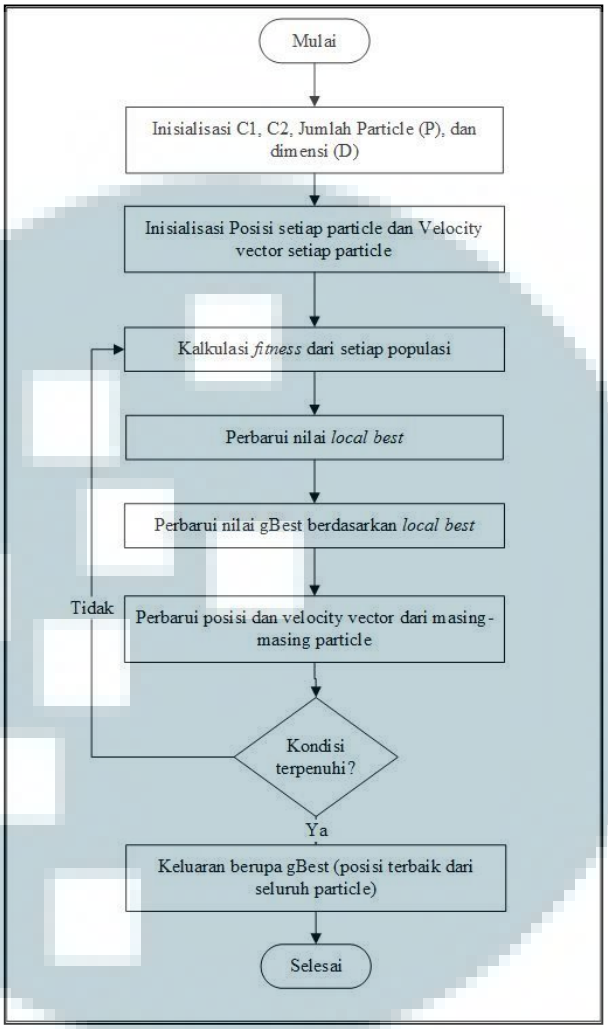
Jaringan saraf tiruan akan mengolah *input* hingga menjadi *output* yang diinginkan. Jika suatu jaringan saraf tiruan belum berhasil mendapatkan *output* yang diinginkan, maka perlu dilakukan proses *learning* untuk mengatur nilai bobot dari masing-masing koneksi. Beberapa strategi dalam melakukan *training* antara lain (Shiffman, 2012).

1. *Supervised Learning*, Jaringan saraf tiruan melakukan pencocokan *output* dengan *output* yang seharusnya.
2. *Unsupervised Learning*, proses *learning* tanpa mengetahui nilai *output* yang akan dituju.
3. *Reinforcement Learning*, proses *learning* dengan melakukan observasi terhadap output yang dihasilkan oleh jaringan saraf tiruan. Negatif atau positif hasil yang diperoleh akan mempengaruhi keputusan yang diambil selanjutnya.

2.4 Particle Swarm Optimization

Particle Swarm Optimization (PSO) adalah algoritma untuk mencari solusi terbaik dengan mensimulasikan pergerakan dan gerombolan burung (Jing-Ru Zhang, 2007). PSO memiliki populasi dari kumpulan solusi. Setiap solusi disebut *particle*, yang akan diberikan nilai awal *velocity* yang random dan terus bergerak mencari solusi dari batasan ruang atau disebut juga *problem space* (Gudise, 2003).

Setiap *particle* akan terus bergerak dengan *velocity* yang ada. Terdapat tiga jenis *velocity*, yaitu *velocity* awal saat diinisialisasi, *velocity* berdasarkan nilai terbaik dari *velocity* sebelumnya, *velocity* berdasarkan nilai terbaik dari seluruh populasi. Inisialisasi awal hingga keluaran dari algoritma PSO akan ditunjukkan pada Gambar 2.2.



Gambar 2.2 *Flowchart* Algoritma *Particle Swarm Optimization*
(sumber: Gudise, 2008)

Untuk memperbarui posisi dan *velocity* dari masing-masing particle, PSO menggunakan persamaan yang ditunjukan pada Rumus 2.10 dan Rumus 2.11 (Gudise, 2008).

$$v_i = wv_i + c_1 r_1 (P_i - x_i) + c_2 r_2 (P_g - x_i)$$

… Rumus 2.10

$$x_i = x_i + v_i$$

… Rumus 2.11

Variable v_i dan x_i adalah *velocity* dan posisi dari *particle* i. Nilai *velocity* dan posisi dari *particle* didapat dari w, yang merupakan *weight control* terhadap explorasi dari *particle* pada *search space*, sedangkan P_i adalah posisi lokal terbaik yang ditemukan oleh *particle* i dan P_g adalah posisi terbaik *global* yang ditemukan dari populasi tersebut. Parameter c_1 dan c_2 adalah konstanta yang dikenal sebagai *learning factor*. *Variable* c_1 menunjukkan seberapa dekat dengan posisi terbaik dan c_2 bernilai sama dengan posisi *global*.

PSO dikenal sebagai algoritma yang mudah digunakan karena kesederhanaan pada parameternya, karena hal tersebut juga parameter menjadi hal yang sangat penting dalam PSO. Pemilihan parameter yang optimal akan mempengaruhi hasil yang didapatkan. Pada penelitian yang berjudul *Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithm for Neural Networks* oleh Venu G. Gudise dan Ganesh K. Venayagamoorthy tahun 2003 menyebutkan bahwa parameter yang optimal dalam *training neural network* sebagai berikut.

1. Nilai *w* yang optimum didapat pada nilai 0.7 hingga 0.8.
2. Ukuran *search space* optimum didapat dengan nilai -100,100. *Search space* berpengaruh pada keleluasaan *particle* dalam mencari solusi.
3. Nilai *velocity* maksimum dari setiap *particle* yang optimum adalah 2.
4. Nilai c_1 dan c_2 yang optimal adalah 2 untuk hasil yang terbaik, begitu juga c_1 bernilai 2,5 dan c_2 bernilai 1,3 menghasilkan keluaran yang cukup baik.
5. Banyaknya *swarm* atau *paricle* yang optimal berkisar antara 20 hingga 25 *particle*.