# BAB 2
# LANDASAN TEORI

Landasan teori menjelaskan tentang teori dasar yang digunakan dalam penelitian ini. Teori dasar tersebut meliputi kondisi berpikir, sinyal elektroensephalogram (EEG), segmentasi, *wavelet, wavelet transform***,** ekstraksi fitur menggunakan *Discrete Wavelet Transform* (DWT), jaringan syaraf tiruan (JST), arsitektur jaringan, fungsi aktivasi, algoritma *backpropagation*, metode analisis terstruktur dan perangkat lunak pendukung.

## 2.1 Kondisi Berpikir

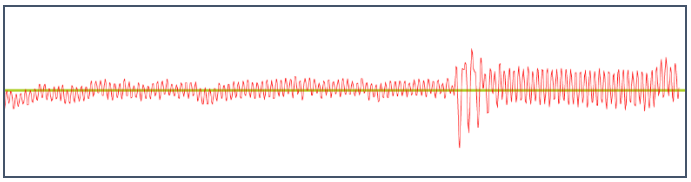
Berpikir adalah suatu kegiatan mental yang melibatkan kerja otak. Walaupun tidak bisa dipisahkan dari aktivitas kerja otak, pikiran manusia lebih dari sekedar kerja organ tubuh yang disebut otak. Kegiatan berpikir juga melibatkan seluruh pribadi manusia dan juga melibatkan perasaan dan kehendak manusia. Memikirkan sesuatu berarti mengarahkan diri pada obyek tertentu, menyadari secara aktif dan menghadirkannya dalam pikiran kemudian mempunyai wawasan tentang obyek tersebut.

Berpikir pada dasarnya memiliki dua kondisi yaitu rileks dan non-rileks. Rileks yang merupakan kondisi psikis dimana seseorang mampu masuk pada kondisi yang tenang, kondisi dimana seseorang bisa fokus pada suatu objek tanpa menambahkan hal lain dan dengan rileks seseorang akan mampu memahami apa yang betul-betul dipikirkannya. Seseorang yang mampu masuk pada kondisi rileks, maka akan mudah menganalisa hal yang rumit dan tepat menghasilkan solusi yang terbaik. Berpikir non rileks juga berarti bekerja keras secara mental untuk memahami sesuatu yang dialami atau mencari jalan keluar dari permasalahan yang sedang dihadapi. Dalam berpikir juga termuat kegiatan meragukan dan memastikan, merancang, menghitung, mengukur, mengevaluasi, membandingkan, menggolongkan, memilah-

milah atau membedakan, menghubungkan, menafsirkan, melihat kemungkinan-kemungkinan yang ada, membuat analisis dan sintesis menalar atau menarik kesimpulan dari premis-premis yang ada, menimbang, dan memutuskan.

## 2.2 Sinyal Elektroensephalogram (EEG)

Elektroensephalogram (EEG) merupakan aktivitas otak berupa gelombang otak yang direkam melalui kulit kepala [7]. EEG diperoleh dengan menggunakan elektroda yang ditempatkan pada kulit kepala. Dalam otak terdapat jutaan *neuron* yang masing-masing menghasilkan tegangan listrik yang kecil, dan tegangan ini membuat listrik yang terdapat pada elektroda dapat mendeteksi dan menerima sinyal EEG dari kulit kepala. Pada gambar 2.1 merupakan contoh grafik perekaman sinyal EEG.



**Gambar 2.1 Grafik Sinyal EEG**

Sinyal EEG dijelaskan dalam hal frekuensi dan amplitudo, yang perbedaannya tergantung pada rangsangan dari luar maupun dari dalam mental seseorang. Jenis-jenis frekuensi EEG berhubungan dengan berbagai kondisi dari otak, yang pertama gelombang Delta yaitu berfrekuensi rendah di antara 0 – 4 Hz, gelombang ini muncul apabila seseorang dalam keadaan tidur nyenyak, yang kedua gelombang Teta yang berfrekuensi di antara 4 – 8 Hz, gelombang ini terjadi apabila seseorang dalam kondisi tidur ringan, mengantuk atau stres emosional, yang ketiga gelombang Alfa yang berfrekuensi di antara 8 – 13 Hz, gelombang ini muncul apabila seseorang dalam keadaan sadar, yang keempat yaitu gelombang Beta, gelombang ini memiliki frekuensi di antara 14 – 30 Hz, gelombang ini muncul apabila seseorang sedang dalam beraktivitas, dan yang terakhir adalah gelombang Gamma yang memiliki

frekuensi lebih dari 30 Hz, biasanya gelombang ini berkaitan dengan aktivitas otak untuk mengintegrasikan berbagai macam rangsangan [7].

Pengolahan sinyal EEG ini dilakukan karena pengenalan secara visual terhadap sinyal EEG sulit dilakukan [7]. Informasi yang terkandung di dalam sinyal EEG ini tidak dapat langsung diamati dikarenakan sinyal EEG memiliki frekuensi dan amplitudo yang lemah, mereka dapat dengan mudah terkontaminasi oleh sumber lain. Sinyal EEG yang berasal dari *biolistrik* lain disebut sebagai *noise* atau pada sinyal EEG dinamakan artefak. Artefak pada sinyal EEG dibagi menjadi dua kategori yaitu, fisiologis dan non-fisiologis. Fisiologis yang berasal dari dalam tubuh manusia seperti detak jantung, mata, otot, gerakan tubuh dan lidah. Sedangkan sumber dari non-fisiologis seperti rangsangan terhadap suara dan lingkungan disekitarnya. Sumber lain seperti berkeringat juga dapat mengubah impedansi pada antarmuka elektroda kulit kepala dan menghasilkan artefak.

### 2.3 *Wavelet*

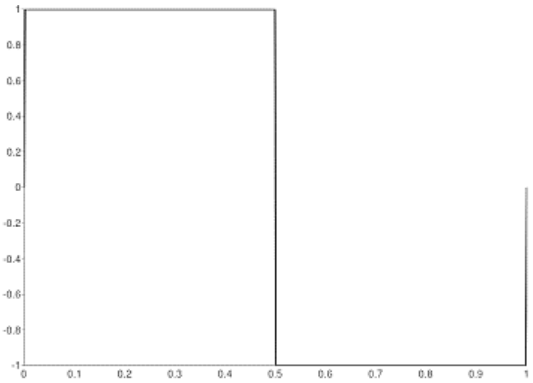
*Wavelet* adalah gelombang osilasi dengan amplitudo yang dimulai dari nol, meningkat dan kemudian menurun ke nol kembali. Kata *Wavelet* ditemukan oleh Morlet dan Grossman pada tahun 1980, dalam bahasa Prancis yaitu *ondelette* yang berarti gelombang kecil [8]. *Wavelet* digunakan untuk mengekstrak informasi dari berbagai macam data seperti sinyal atau citra. Kumpulan dari *wavelet* akan diuraikan atau diekstraksi tanpa adanya informasi yang hilang. Data yang telah diekstraksi dapat dipulihkan dengan mendapatkan informasi yang asli, ukuran yang kecil serta kerugian yang minimal.

Keluarga *wavelet* yang berbeda dapat digunakan untuk analisis *wavelet*. Terdapat berbagai jenis keluarga *wavelet seperti haar*, *daubechies*, *symlets* dan lainnya. Perbedaan ini didasarkan pada kesimetrikan atau keteraturan data yang digunakan. Selain itu, keluarga *wavelet* yang bekerja dengan baik pada suatu aplikasi tertentu mungkin tidak bisa dikatakan lebih unggul dibandingkan dengan aplikasi

lainnya, karena kinerjanya tergantung pada aplikasi atau data yang digunakan. Beberapa penjelasan pada setiap keluarga *wavelet* adalah sebagai berikut.

1. *Haar Wavelet*

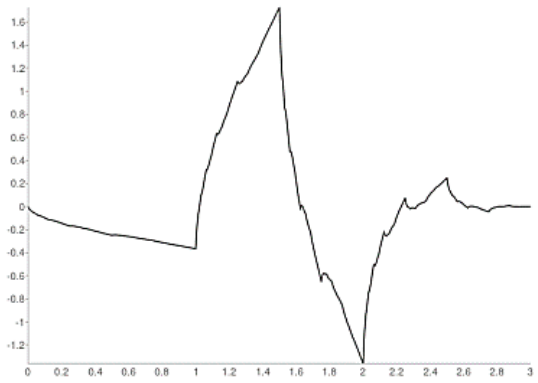
*Haar wavelet* adalah jenis keluarga *wavelet* yang paling sederhana dan tertua diantara semua keluarga *wavelet* [9]. Pada gambar 2.1 merupakan merupakan fungsi *square-shaped* yang membentuk *haar*. Fungsi ini cocok digunakan untuk analisis sinyal dengan transisi tiba-tiba dan tidak terus-menerus berubah.




**Gambar 2.2 Jenis *Haar Wavelet***

2. *Daubechies Family*

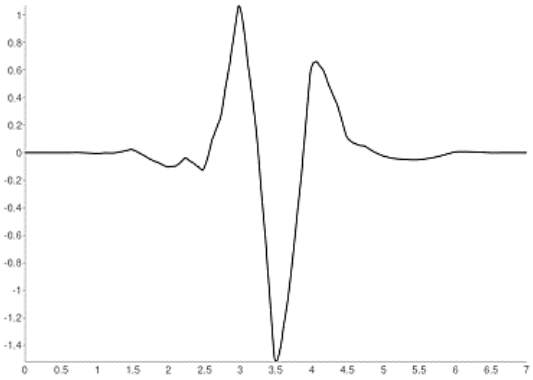
*Daubechies wavelet* merupakan keluarga *wavelet* yang paling sering digunakan untuk pemrosesan sinyal *digital*. Banyak penelitian pada sinyal menggunakan *daubechies* ini dikarenakan sinyal memiliki ciri gelombang yang asimetris [9]. Pada gambar 2.2 merupakan beberapa jenis *wavelet daubechies* dengan beberapa karakteristik gelombang yang berbeda. Pada penelitian ini, sinyal EEG yang digunakan memiliki ciri gelombang yang asimetris. Maka, keluarga *wavelet* yang cocok digunakan untuk sinyal EEG adalah keluarga dari *daubechies wavelet*. Pada penelitian terdahulu, dengan jenis db4 memiliki akurasi yang baik pada *filtering* sinyal EEG [3].



**Gambar 2.3 Jenis *Daubechies Wavelet***

3. *Symlets Family*

*Symlets wavelet* merupakan *wavelet* simetris yang ditemukan untuk memodifikasi *wavelet* simetris. Keluarga ini hampir simetris dan orthogonal [9]. Keluarga *symlet* memiliki beberapa jenis seperti pada gambar 2.3.



**Gambar 2.4 Jenis *Symlets Wavelet***

## 2.4 *Wavelet Transform*

*Wavelets Transform* adalah fungsi *transform* yang digunakan untuk menguraikan data atau fungsi atau operator menjadi komponen frekuensi yang berbeda-beda dengan resolusi yang disesuaikan dengan skalanya. *Wavelet Transform* dibagi menjadi dua jenis yaitu *continuous wavelet transform* (CWT) dan *discrete wavelet transform* (DWT). Pada dasarnya CWT handal dalam menganalisis data

dengan frekuensi yang tinggi, sedangkan pada frekuensi rendah DWT relatif lebih handal [9]. Cara kerja *wavelet transform s*ecara garis besar adalah sebagai berikut.

1. *Continuous Wavelet Transform* (CWT)

Cara kerja CWT adalah dengan menghitung konvolusi sebuah sinyal dengan sebuah jendela modulasi pada setiap waktu dengan setiap skala yang diinginkan. Jendela modulasi yang mempunyai skala fleksibel inilah yang biasa disebut dengan *mother of wavelet* (induk *wavelet*) atau fungsi dasar *wavelet.*

2. *Discrete Wavelet Transform* (DWT)

Cara kerja DWT relative lebih mudah dibandingkan dengan CWT dalam pengimplementasiannya. Prinsip dasar dari DWT adalah cara untuk mendapatkan representasi waktu dan skala dari sebuah sinyal dengan menggunakan teknik *digital filter* dan operasi *sub-sampling*.

Pada penelitian ini, sinyal EEG yang digunakan memiliki ciri frekuensi dan amplitudo yang rendah. Maka untuk kasus ini, *discrete wavelet transform (DWT)* merupakan metode yang cocok dalam mereduksi sinyal EEG.

## 2.4 Ekstraksi Fitur Menggunakan *Discrete Wavelet Transform* (DWT)

*Discrete wavelet transform* (DWT) berhasil digunakan pada penelitian dalam mengekstraksi fitur pada pasien epilepsi dengan mengambil fitur pada domain waktu dan frekuensi. Ekstraksi fitur dengan DWT pada sinyal f(n) dilakukan dengan dekomposisi sinyal ke dalam aproksimasi dan detil informasi menggunakan dua fungsi yaitu *scaling function* dan *wavelet function*, yang masing-masing berhubungan dengan *low-pass filter* g(n) dan *high-pass filter* h(n). Berikut merupakan koefisien dari db4 *low-pass filter* dan *high-pass filter* [8].

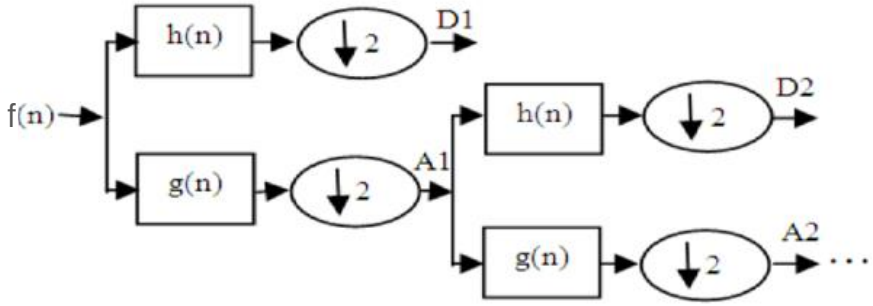
a. Koefisien *scaling function* (*low-pass filter*)

$$g_0 = \frac{1+\sqrt{3}}{4\sqrt{2}},\ g_1 = \frac{3+\sqrt{3}}{4\sqrt{2}},\ g_2 = \frac{3-\sqrt{3}}{4\sqrt{2}},\ g_3 = \frac{1-\sqrt{3}}{4\sqrt{2}} \qquad \ldots(2.1)$$

b. Koefisien *wavelet function* (*high-pass filter)*

$$h_0 = \frac{1-\sqrt{3}}{4\sqrt{2}},\ h_1 = -\frac{3-\sqrt{3}}{4\sqrt{2}}, h_2 = \frac{3+\sqrt{3}}{4\sqrt{2}}, h_3 = -\frac{1+\sqrt{3}}{4\sqrt{2}} \quad \ldots(2.2)$$

Pada gambar 2.4 merupakan gambaran untuk proses dekomposisi sinyal menggunakan DWT.




**Gambar 2.5 Dekomposisi Sinyal DWT**

Pada gambar 2.5 terdapat dua proses dalam satu kali proses dekomposisi yaitu konvolusi dan *sub-sampling*. Konvolusi merupakan bagian dari proses dekomposisi yang melewatkan sinyal pada *filter low-pass* dan *high-pass*. Perhitungan konvolusi yaitu perkalian dua buah fungsi menggunakan *inner product* dengan persamaan sebagai berikut.

a. Konvolusi terhadap *low-pass filter*

$$Y(j)_{low} = \sum_{i=0}^{n} f(j,i) * g(i) \quad \ldots(2.3)$$

b. Konvolusi terhadap *high-pass filter*

$$Y(j)_{high} = \sum_{i=0}^{n} f(j,i) * h(i) \quad \ldots(2.4)$$

Dimana:

$Y(j)_{low}$ : konvolusi terhadap *low-pass filter*

$Y(j)_{high}$ : konvolusi terhadap *high-pass filter*

f(j,i) : sinyal masukan

g(i) : koefisien *low-pass filter*

h(i) : koefisien *high-pass filter*

j : pergeseran ke-j = 0, 1, …, n

i : 0, 1, …, n

Langkah-langkah dalam proses konvolusi adalah sebagai berikut [4].

1. Menyamakan koefisien *filer* dengan panjang sinyal dengan memberi nilai 0 pada elemen lainnya.
2. Samakan posisi sinyal ke-0 dengan koefisien yang sudah disamakan, lalu lakukan perkalian sampai sinyal ke-*n*.
3. Jumlahkan seluruh nilai hasil perkalian untuk mendapatkan nilai hasil ke-0.
4. Lakukan pergeseran ke kanan terhadap koefisien sehingga koefisien ke-0 tepat berada pada sinyal ke-1.
5. Lakukan langkah 2, 3 dan 4 sampai koefisien pertama sejajar dengan sinyal yang terakhir ke-n.

Kemudian setelah melalui proses konvolusi, sinyal akan melewatkan *sub-sampling* yang dalam hal ini adalah *downsampling* yaitu proses mengabaikan faktor dua pada sinyal.

## 2.5 Jaringan Syaraf Tiruan (JST)

Jaringan syaraf tiruan (JST) merupakan salah satu representasi dari otak manusia untuk mensimulasikan proses pembelajaran otak manusia. Otak pada dasarnya belajar dari pengalaman, ini menunjukan bahwa otak menyimpan informasi sebagai pola. Jaringan syaraf tiruan memecahkan masalah pada komputer yaitu bisa melakukan hal-hal yang belum diketahui untuk mengerjakan suatu permasalahan.

Permasalahan yang belum diketahui ini dapat dikerjakan dengan menggunakan pola yang berasal dari pelatihan data masukan. Beberapa kelebihan pada jaringan syaraf tiruan yaitu [10].

1. Pembelajaran Adaptif

   Kemampuan untuk belajar bagaimana melakukan tugas-tugas berdasarkan data yang diberikan atau pada pengalaman awal.

2. *Self-organizing learning*

   JST dapat membuat organisasi sendiri atau representasi untuk menerima informasi selama waktu belajar, yaitu belajar untuk mengklasifikasikan tanpa adanya pelatihan.

3. Operasi Langsung

   Perhitungan JST mungkin dilakukan secara parallel, perangkat keras khusus yang sedang dirancang dan diproduksi memanfaatkan kemampuan ini.

4. Toleransi Kesalahan

   Kerusakan sebagian dari JST mengarah kepada kinerja itu sendiri. Namun, beberapa kemampuan jaringan dapat dipertahankan tanpa mempengaruhi kinerja lainnya.

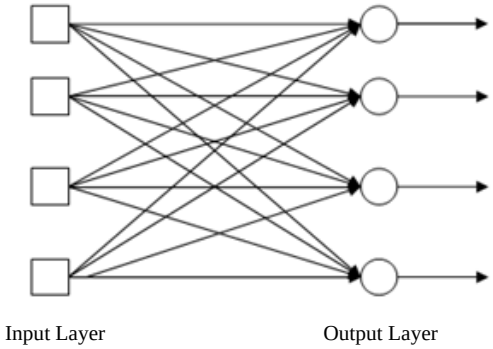
Seperti halnya pada otak manusia, jaringan syaraf tiruan terdiri dari beberapa *neuron*, *di mana* setiap *neuron* memiliki keterkaitan satu sama lainnya. Informasi yang diterima dari *neuron* masukan akan dilanjutkan ke *neuron* lainnya. Pada jaringan syaraf tiruan, hubungan ini dikenal sebagai bobot. Informasi tersebut akan disimpan pada suatu nilai tertentu di dalam bobot dan kemudian akan dilanjutkan ke *neuron* lainnya.

## 2.6 Arsitektur Jaringan

Arsitektur jaringan merupakan pola *di mana neuron-neuron* pada JST disusun berhubungan erat dengan algoritma yang digunakan untuk melatih jaringan [10]. Beberapa arsitektur jaringan yang sering dipakai dalam jaringan syaraf tiruan yaitu sebagai berikut.

1. *Single-Layer Feedforward Networks*

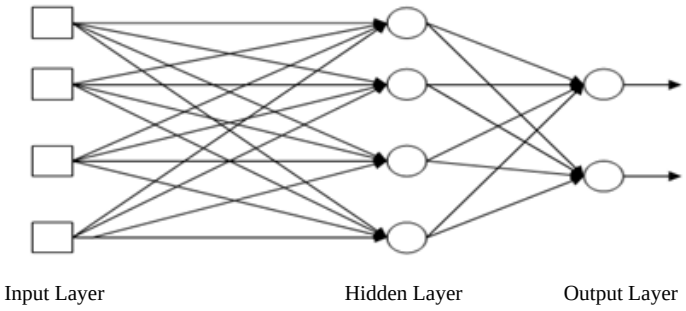
Dalam jaringan ini, sekumpulan *input neuron* dihubungkan langsung dengan sekumpulan *output*nya. Dalam beberapa model (misal *perceptron*), hanya ada sebuah unit *neuron output*. Pada gambar 2.5 merupakan arsitektur dari *single-layer feedforward networks.*




**Gambar 2.6 *Single-Layer Feedforward Networks***

2. *Multi-Layer Feedforward Networks*

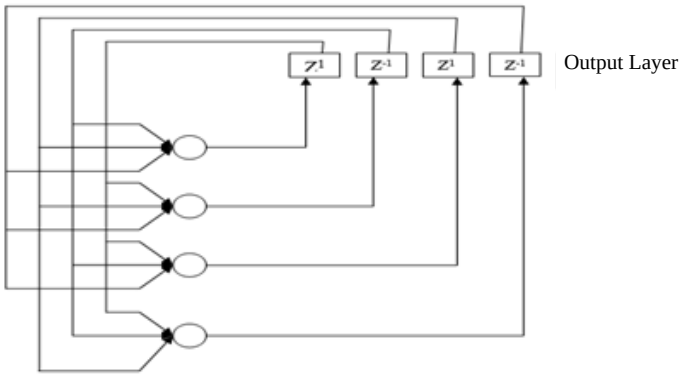
Jaringan *layer* jamak merupakan perluasan dari *layer* tunggal. Dalam jaringan ini, selain unit *input* dan *output*, ada unit-unit lain (sering disebut *layer* tersembunyi/ *hidden layer*). Dimungkinkan pula ada beberapa *layer* tersembunyi. Sama seperti pada unit *input* dan *output*, unit-unit dalam satu *layer* tidak saling berhubungan. Pada gambar 2.6 merupakan arsitektur dari *Multi-Layer Feedforward Networks*




**Gambar 2.7 *Multi-Layer Feedforward Networks***

3. *Recurrent Networks*

Jaringan *recurrent* mirip dengan jaringan *layer* tunggal ataupun ganda. Hanya saja, ada *neuron output* yang memberikan sinyal pada unit *input* (sering disebut *feedback loop*). Gambar 2.7 merupakan arsitektur dari *recurrent*




**Gambar 2.8 *Recurrent Networks***

## 2.7 Pembelajaran dalam JST

Pembelajaran dalam JST merupakan suatu proses dimana parameter-parameter bebas JST diadaptasikan melalui proses perangsangan berkelanjutan oleh lingkungan *di mana* jaringan berada [10]. Berdasarkan cara memodifikasi bobotnya, terdapat dua macam pembelajaran di dalam JST yaitu *supervised learning* (pembelajaran terawasi) dan *unsupervised learning* (pembelajaran tidak terawasi). Berikut merupakan penjelasan dari kedua jenis pembelajaran di dalam JST:

1. *Supervised Learning* (pembelajaran terawasi)

Proses pembelajaran terawasi pada JST adalah proses belajar dengan memberikan latihan untuk mencapai suatu target keluaran yang telah ditentukan. JST mendapatkan latihan untuk mengenal pola-pola tertentu. Dengan memberikan target keluaran, perubahan *input* akan diadaptasi oleh *output* dengan mengubah bobot-bobot

mengikuti aturan algoritma yang ditentukan. Sebelum pelatihan, terdapat beberapa hal yang perlu di perhatikan yaitu sebagai berikut [10].

a. Pemberian urutan pola yang akan diajarkan
b. Kriteria perhitungan *error*
c. Kriteria proses belajar
d. Jumlah iterasi yang h/arus dilalui (*epoch)*
e. Inisialisasi bobot dan parameter awal

Salah satu proses pembelajaran terawasi adalah menggunakan algoritma *backpropagation.* Jaringan *backpropagation* ini sering dikenal dengan *multilayer perceptron* (MLP) dikarenakan arsitektur jaringan ini memiliki *layer* selain *input* dan *output*.

2. *Unsupervised Learning* (pembelajaran tidak terawasi)

Pembelajaran tidak terawasi merupakan pembelajaran pada JST yang tidak memiliki *target output.* Pembelajaran tanpa pengawasan akan mengelompokkan pola-pola masukan ke dalam kelompok yang berbeda. *Output* yang dihasilkan tidak dapat ditentukan pada proses pelatihan. Selama proses pelatihan, nilai bobot disusun dalam suatu *range* tertentu tergantung pada nilai *input* yang diberikan. Tujuan dari pembelajaran ini adalah mengelompokkan *unit-unit* yang hampir sama ke dalam suatu *area* tertentu.

## 2.8 Fungsi Aktivasi

Fungsi aktivasi merupakan fungsi yang digunakan pada jaringan syaraf untuk mengaktifkan atau tidak mengaktifkan *neuron*. Fungsi aktivasi pada JST memiliki berbagai macam jenis, tetapi fungsi aktivasi yang umum digunakan adalah sebagai berikut.

1. Fungsi *threshold*

Fungsi undak biner sering digunakan pada jaringan dengan lapisan tunggal untuk mengkonversi input dari suatu variabel yang bernilai kontinu ke suatu output biner (0 atau 1).

$$f(x)=\begin{cases}0 & if \quad x \le 0\\ 1 & if \quad x > 0\end{cases} \qquad \ldots(2.5)$$

2. Fungsi *sigmoid biner*

Fungsi *sigmoid biner* memiliki nilai pada *range* 0 sampai 1. Oleh karena itu, fungsi ini sering digunakan untuk jaringan syaraf yang membutuhkan nilai output yang terletak pada *interval* 0 sampai 1. Pada penelitian ini menggunakan fungsi aktivasi *sigmoid biner* untuk mengaktivasi nilai keluaran pada *hidden* dan *output layer*.

$$f(x) = \frac{1}{1+e^{-x}} \qquad \ldots(2.6)$$

## 2.9 Algoritma *Backpropagation*

*Backpropagation* merupakan algoritma *supervised learning* dengan arsitektur *multi-layer feedforward networks.* Algoritma *backpropagation* menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan *error* ini, tahap perambatan maju (*feedforward*) harus dikerjakan terlebih dahulu. Pada perambatan maju, *neuron-neuron* diaktifkan dengan menggunakan fungsi aktivasi sigmoid biner yang merupakan persamaan dari (2.6). Sebelum pelatihan dilakukan inisialisasi *parameter* yang digunakan dalam pelatihan dengan menentukan nilai laju pembelajaran (α), toleransi kesalahan dan maksimal iterasi. Setelah proses inisialisasi, peroses selanjutnya adalah menentukan bobot-bobot yang menghubungkan semua lapisan dan juga bobot biasnya. Nilai yang tersimpan di dalam bobot dilakukan secara acak dengan rentang diantara -0.5 sampai 0.5.

Pada proses pelatihan dibagi atas dua tahap yaitu proses *feedforward* dan *backward* untuk memperbaharui nilai pada bobot. Pada pelatihan, tahap *feedforward* menghasilkan *output* yang kemudian dibandingkan terhadap terhadap target tiap data *input*, apabila selisih lebih besar dari toleransi kesalahan maka dilakukan koreksi

bobot dengan *backpropagation*. Penjelasan proses pelatihan adalah sebagai berikut [15].

Proses *feedforward*:

1. Setiap unit *input* ($x_i$, i = 1, 2, 3, …, n) mengirimkan sinyal *input* ke setiap *input* yang berada pada *hidden layer*.
2. Masing-masing unit di lapisan tersembunyi ($z_j$, j = 1, 2, 3, …, p) dikalikan dengan bobotnya dan jumlahkan serta tambahkan dengan biasnya:

$$z_in_j = V_{0j} + \sum_{i=1}^{n} X_i V_{ij} \quad \ldots(2.7)$$

menghitung sinyal *hidden* dengan menggunakan fungsi aktivasi (2.6):

$$z_j = f(z_in_j) \quad \ldots(2.8)$$

3. Masing-masing unit *output* ($y_k$, k = 1, 2, 3, …, m) dikalikan dengan bobot dan dijumlahkan dengan biasnya.

$$y_in_k = W_{0k} + \sum_{j=1}^{p} Z_j W_{jk} \quad \ldots(2.9)$$

menghitung sinyal *output* dengan menggunakan fungsi aktivasi (2.6):

$$y_k = f(y_in_k) \quad \ldots(2.10)$$

Proses *backward*:

1. Masing-masing unit *output* ($y_k$, k = 1, 2, 3, …, m) menerima target pola $t_k$ sesuai dengan pola *input* saat pelatihan dan kemudian informasi *error* lapisan *output* ($\delta_k$) dihitung.

$$\delta_k = (t_k - y_k)\, f'(y_in_k) = (y_k)(1-y_k) \quad \ldots(2.11)$$

kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai $W_{jk}$):

$$\Delta W_{jk} = \alpha \delta_k Z_j \quad \ldots(2.12)$$

Hitung koreksi bias untuk memperbaiki nilai $W_{0k}$:

$$\Delta W_{0k} = \alpha \delta_k \quad \ldots(2.13)$$

2. Pada setiap *hidden layer* ($z_j$, j = 1, 2, 3, …, p) menjumlahkan delta *input* (dari unit-unit yang berada pada lapisan sebelumnya)

$$\delta_\text{in}_j = \sum_{k=1}^{m} \delta_k W_{jk} \quad \ldots(2.14)$$

$$\delta_j = \delta_\text{in}_j f'(z_in_j) \text{ yang berarti:} \quad \ldots(2.15)$$

$$\delta_j = (\delta_\text{in}_j)(z_in_j)(1 - z_in_j) \quad \ldots(2.16)$$

kemudian hitung koreksi bobot untuk memperbaiki bobot V.

$$\Delta V_{ij} = \alpha \delta_j x_i \quad \ldots(2.17)$$

Hitung koreksi bobot bias $V_{0j.}$

$$\Delta V_{0j} = \alpha \delta_j \quad \ldots(2.18)$$

3. Masing-masing unit keluaran ($y_k$, k = 1, 2, 3, …, m) dilakukan untuk memperbaiki bias dan bobotnya (j = 0, 1, 2, …p) sehingga menghasilkan bobot dan bias baru.

$$W_{jk}(baru) = W_{jk}(lama) + \Delta W_{jk} \quad \ldots(2.19)$$

Demikian juga untuk bobot pada *input layer* ke *hidden layer*, diperbaiki dengan menggunakan koreksi bobot yang sudah dihitung sebelumnya.

$$V_{ij}(baru) = V_{ij}(lama) + \Delta V_{ij} \quad \ldots(2.20)$$

Terdapat dua kondisi yang dapat menghentikan pelatihan, yang pertama jika nilai rata-rata kesalahan sudah lebih kecil dari toleransi kesalahan, yang kedua jika jumlah iterasi sudah lebih dari maksimal iterasi/*epoch* yang ditentukan. Untuk persamaan rata-rata eror atau *Mean Square Error* (MSE) direpresentasikan pada persamaan 2.21.

$$MSE = \frac{1}{n} \sum_{k=1}^{n} (t_{k-} y_k)^2 \quad \ldots(2.21)$$

Dimana:

$t_k$ = target keluaran

$y_k$ = nilai keluaran (*neuron output*)

n = jumlah *neuron*

k = 1, 2, …, n

Pada tahap pengujian, algoritma yang digunakan pada *backpropagation* hanya proses perambatan maju (*feedforward)* saja. Bobot-bobot hasil dari pelatihan akan digunakan pada tahap pengujian. Kemudian hasil *output* dari pengujian akan dibandingkan menggunakan persamaan (2.5).

## 2.10 Metode Analisis Sistem Terstruktur

Metode analisis yang digunakan dalam pemrograman sistem klasifikasi ini adalah *Data Flow Diagram* (DFD). Sedangkan untuk pemodelan data digambarkan dengan *Entity Relationship Diagram* (ERD).

### 2.10.1 *Data Flow Diagram* (DFD)

Pemodelan proses yang digunakan untuk menggambarkan proses yang terjadi di da Algoritma dan Flowchart Flowchart, Dian Palupi Rini Analisis dan Perancangan Sistemlam sistem klasifikasi ini yaitu menggunakan DFD. *Data Flow Diagram* (DFD) adalah alat pembuatan model proses yang memungkinkan profesional sistem untuk menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur data, baik secara manual maupun komputerisasi. DFD ini sering disebut juga dengan nama *Bubble chart*, *Bubble diagram*, model proses, diagram alur kerja, atau model fungsi.

Diagram konteks adalah diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. Diagram konteks merupakan level tertinggi dari DFD yang menggambarkan seluruh *input* ke sistem atau output dari sistem. Ia akan memberikan gambaran tentang keseluruhan sistem. Dalam diagram konteks hanya ada satu proses, tidak boleh ada *store* di dalam diagram konteks.

Aliran data yang masuk kedalam dan keluar dari suatu proses harus sama dengan aliran data yang masuk kedalam dan keluar dari rincian proses pada level/tingkatan di bawahnya. Hal yang perlu diperhatikan didalam menggambarkan proses DFD adalah sebagai berikut [20].

a. Harus terdapat keseimbangan *input* dan *output* dari satu level dan level berikutnya.
b. Keseimbangan antara level 0 dan level 1 dilihat pada *input/output* dari aliran data ke atau dari terminal pada level 0, sedangkan keseimbangan antara level 1 dan level 2 dilihat pada *input/output* dari aliran data ke/dari proses yang bersangkutan.
c. Nama aliran data, data *store* dan terminal pada setiap level harus sama apabila objeknya sama.

### 2.10.2 *Entity Relationship Diagram* (ERD)

Pemodelan data yang digunakan dalam pembangunan sistem klasifikasi ini menggunakan pemodelan ERD. *Entity Relationship Diagram* (ERD) adalah sebuah konsep yang mendeskripsikan hubungan antara penyimpanan dan didasarkan pada persepsi dari sebuah dunia nyata yang terdiri dari sekumpulan objek, disebut entiti dan relasi diantar objek-objek tersebut. Atau juga bisa disebut suatu model jaringan (network) yang menggunakan susunan data yang disimpan dari sistem secara abstrak. Elemen-elemen diagram hubungan entitas akan dijelaskan sebagai berikut [11].

1. *Entity*

   *Entity* adalah sesuatu apa saja yang ada dalam sistem, nyata maupun abstrak dimana data tersimpan atau dimana terdapat data. Entitas diberi nama dengan kata benda dan dapat dikelompokan dalam empat jenis nama, yaitu orang, benda, lokas dan kejadian.

2. *Relationship*

   *Relationship* adalah hubungan alamiah yang terjadi antara entitas. Pada umumnya penghubung (*Relationship*) diberi nama dengan kata kerja dasar, sehingga memudahkan untuk melakukan pembacaan relasinya (bisa dengan kalimat aktif atau kalimat pasif). Derajat Relationship yang sering dipakai dalam ERD:

a. *Unary Relationship*

*Unary Relationship* adalah model *Relationship* yang terjadi diantara *entity* yang berasal dari *entity* set yang sama.

b. *Binary Relationship*

*Binary Relationship* adalah model Relationship antara *instance-instance* dari satu tipe entitas (dua *entity* yang berasal dari *entity* yang sama). *Relationship* ini paling umum digunakan dalam pembuatan model data.

c. *Tenary Relationship*

*Tenary Relationship* merupakan *relationship* antara *instance-intance* dari tiga tipe entitas secara sepihak.

3. Atribut

Atribut adalah sesuatu yang menjelaskan apa sebenarnya yang dimaksud entitas maupun *relationship,* sehingga sering dikatakan atribut adalah elemen dari setiap entitas dan *relationship*. Atriburt terdiri dari dua jenis yaitu *Identifier* (*key*) digunakan untuk menentukan suatu *entity* secara unik (Primary *key*) dan *Descriptor* (nonkey *attribute*) digunakan untuk menspesifikasikan karakteristik dari suatu *entity* yang tidak unik.

4. Kardinalitas (*Cardinality*)

Kardinalitas relasi menunjukan jumlah maksimum yang dapat berelasi dengan entitas pada entitas yang lain. Jenis kardinalitas pada *Entity Relational Diagram* terdiri menjadi 3 yaitu:

a. *One to One*

Tingkat hubungan satu ke satu, dinyatakan dengan satu kejadian pada entitas pertama, hanya mempunyai satu hubungan dengan satu kejadian pada entitas kedua dan sebaliknya.

b. *One to Many* atau *Many to One*

Tingkat hubungan satu ke banyak adalah sama dengan banyak ke satu. Tergantung dari arah mana hubungan tersebut dilihat. Untuk satu kejadian

pada entitas yang pertama dapat mempunyai banyak hubungan dengan kejadian pada entitas yang kedua.

c. *Many to Many*

Tingkat hubungan banyak ke banyak terjadi jika tiap kejadian pada sebuah entitas akan mempunyai banyak hubungan dengan kejadian pada entitas lainnya.

### 2.10.3 *Flowchart*

*Flowchart* dapat diartikan sebagai suatu alat atau sarana yang menunjukkan langkah-langkah yang harus dilaksanakan dalam penyelesaian suatu permasalahan untuk komputasi dengan cara mengekspresikannya ke dalam serangkaian simbol-simbol grafis khusus [17].

Tujuan utama dari penggunaan *flowchart* adalah untuk menggambarkan suatu tahapan penyelesaian masalah secara sederhana, terurai, rapi, dan jelas dengan menggunakan simbol-simbol yang standar. Tahap penyelesaian masalah yang disajikan harus jelas, sederhana, efektif, dan tepat. Dalam penulisan *flowchart* dikenal dua model, yaitu sistem *flowchart* dan program *flowchart* [18].

a. Sistem *flowchart*

Sistem *flowchart* merupakan diagram alir yang menggambarkan susatu sistem peralatan komputer yang digunakan dalam proses pengolahan data serta hubungan antar peralatan tersebut. Sistem *flowchart* ini tidak digunakan untuk menggambarkan urutan langkah untuk memecahkan masalah, tetapi hanya untuk menggambarkan prosedur dalam sistem yang dibentuk. Dalam menggambarkan *flowchart* biasanya digunakan simbol-simbol yang standar, tetapi pemrogram juga dapat membuat simbol-simbol sendiri apabila simbol-simbol yang telah tersedia dirasa masih kurang. Dalam kasus ini pemrogram harus melengkapi gambar *flowchart* tersebut dengan kamus simbol untuk menjelaskan arti dari masing-masing simbol yang digunakan agara pemrogram lain dapat mengetahui maksud dari simbol-simbol tersebut [18].

b. Program *flowchart*

Program *flowchart* merupakan diagram alir yang menggambarkan urutan logika dari suatu prosedur pemecahan masalah. Untuk menggambarkan program *flowchart* telah tersedia simbol-simbol standar, namun demikian seperti halnya pada sistem *flowchart*, pemrogram dapat menambah khasanah simbol-simbol tersebut asalkan pemrogram melengkapinya dengan penggambaran program *flowchart* dengan kamus simbol [18].

Manfaat menggunakan *flowchart* dalam mengembangkan prosedur pemecahan masalah komputasi.

1. Akan terbiasa berpikir secara sistematis dan terstruktur dalam setiap kesempatan
2. Akan lebih mudah mengecek dan menemukan bagian-bagian prosedur yang tidak valid dan bertele-tele.
3. Prosedur yang dikembangakan akan lebih mudah dipahami oleh orang lain, sehingga tidak menimbulkan kesalahan interpretasi apabila mau menerapkan prosedur yang dikembangkan [17].

Secara umum, proses penyelesaian suatu permasalahan terdiri dari lima langkah utama, yaitu:

1. Dimulainya suatu proses
2. Membaca data masukkan
3. Proses penyelesaian permasalahan
4. Mencetak hasil pengolahan/informasi
5. Diakhirinya semua proses pengolahan.

## 2.11 Perangkat Lunak Pendukung

Perangkat lunak pendukung yang digunakan di dalam pembangunan sistem klasifikasi ini menggunakan MySQL sebagai *database*, HTML dan PHP sebagai bahasa pemrograman, XAMPP sebagai *web service* yang akan dijelaskan lebih detail sebagai berikut.

### 2.11.1 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang *multithread, multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus di mana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak sama dengan proyek-proyek seperti Apache, dimana perangkat lunak dikembangkan oleh komunitas umum dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing. MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB yang memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius.

MySQL digunakan oleh sebagian besar Web Server yang ada di jagat internet. Di samping karena dianggap simple, juga dapat di-porting pada berbagai sistem operasi sekelas server, seperti Windows, Linux, Solaris, Mac OS, BSD, Unix, IBM-AIX. Walaupun *relative simple*, MySQL memiliki fitur-fitur yang sangat baik, sehingga sangat cocok untuk digunakan dalam implementasi aplikasi basis data, khususnya yang berbasis Web. Setelah beberapa kali ganti pemilik, saat ini MySQL dimiliki oleh Oracle Corporation, sebuah perusahaan skala besar di bidang basis data yang sebelumnya juga sudah memiliki DBMS yang paling popular yaitu Oracle [12]. Pada sistem klasifikasi ini, versi MySql yang digunakan MySql 6.8.3.

### 2.11.2 HTML

*Hypertext Markup Language* (HTML) adalah bahasa dari *World Wide Web* (www) yang dipergunakan untuk menyusun dan membentuk dokumen agar dapat ditampilkan pada program browser. HTML merupakan protokol yang digunakan untuk mentransfer data atau dokumen dari web server ke browser.

Fungsi dari HTML adalah membentuk tata letak dokumen, dalam hal ini menentukan jenis huruf, gambar, dan dokumen-dokumen lainnya. Menentukan hubungan ke dokumen lain, HTML merupakan suatu bahasa computer yang termasuk dalam kategori SGML (*Standard Generalized Markup Language*) dimana bentuknya merupakan file standart ASCII yang berisi kode-kode untuk mengatur dokumen [13].

### 2.11.3 PHP

PHP adalah kependekan dari *PHP Hypertext Preprocessor*, bahasa *interpreter* yang mirip dengan bahasa C dan Perl yang memiliki kesederhanaan perintah [14]. PHP dapat digunakan bersama dengan HTML sehingga memudahkan dalam pembangunan aplikasi web dengan cepat. Php dapat digunakan untuk meng-*update* basis data dan menciptakan basis data. *Interpreter* adalah sebuah program yang digunakan untuk membaca *file* yang berisi kode program yang akan dijalankan, kemudian *interpreter* tersebut akan meminta CPU untuk melakukan perintah yang diterimanya. Adapun keunggulan yang dimiliki oleh PHP (*Personal Home Page*) sebagai berikut [14].

1. *Life Cycle* yang sangat singkat, sehingga PHP selalu *up to date* mengikuti perkembangan teknologi internet. *Cross Platform*, yakni PHP dapat dipakai di hampir semua *webserver* yang ada di pasaran (terutama Apache dan Microsoft IIS) dan dijalankan pada berbagai sistem operasi (Linux, Windows, FreeBSD).
2. PHP mendukung koneksi ke banyak database baik yang gratis maupun komersil, seperti *MySQL*, *mSQL*, *Oracle*, *Microsoft SQL Server*, *Interbase*, dan banyak lagi.
3. PHP bersifat *open source* dan gratis. Kemudahan dalam mendapatkan dokumentasi. di Internet, kita tidak akan sulit untuk mencari baik itu referensi, kode-kode PHP yang sudah jadi dan juga mengajukan pertanyaan pada grupgrup diskusi yang di dalamnya banyak sekali para master PHP.

### 2.11.4 XAMPP

XAMPP merupakan sebuah perangkat lunak atau software yang bebas (maksudnya *open source*) yang dapat digunakan di berbagai sistem operasi. XAMPP merupakan kompilasi dari beberapa program. XAMPP berfungsi sebagai *server offline* yang berdiri sendiri (seringkali disebut sebagai localhost). XAMPP terdiri dari beberapa program yaitu Apache HTTP Server, MYSQL Database, PHP, dan Pearl. Program ini dapat di *download* secara gratis karena tersedia dalam lisensi GNU (General Public License). XAMPP terkenal sebagai program *Web Service* (localhost) yang mudah digunakan *user friendly*. Versi XAMPP yang digunakan pada sistem klasifikasi ini yaitu XAMPP v.3.2.1 [19].

### 2.11.5 Apache *Web Service*

Apache merupakan turunan dari *Web Service* yang dikembangkan oleh NCSA (*National Center for Supercomputing Application*) sekitar tahun 1995 yang dikenal dengan NCSA HTTP Daemon (NCSA HHTPd) yang pada RedHat Linux 7.1 telah digunakan *patch* untuk mengganti NCSA HHTPd tersebut. Pada RedHat Linux 7.1 [16]. Apache yang digunakan adalah versi 1.3.19 dengan release 5 atau lebih dikenal dengan paket apache-1.3.19-5.i386.rpm.

Apache *Web Service* memiliki program pendukung cukup banyak yang dapat memberikan layanan yang cukup bagi penggunanya. Berikut ini adalah beberapa program pendukung dari Apache *Web Service*.

1. Kontrol Akses.
2. *Common Gateway Interface* (CGI), yang paling terkenal dan sangat sering digunakan adalah Perl (*Practical Extraction dan Report Language*).
3. PHP (*Personal Home Page*).
4. SSI (*Server Slide Include*).