BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauan Pustaka

Penulis menggunakan beberapa referensi sebagai bahan perbandingan dengan penelitian yang akan penulis lakukan. Dalam perbandingan ini penulis berfokus pada segi objek, metode yang digunakan, serta kesimpulan yang didapat dari hasil penelitian tersebut.

Nicolaus(2016) melakukan penelitian tentang "Pengenalan Pola Huruf Jepang *Hiragana* Menggunakan Algoritma *Backpropagation".* Mendapat hasil terbaik pada percobaan ke 7 sebesar 86.63% dengan menggunakan 2 neuron lapisan tersembunyi yaitu 120 dan 100 dan 1 lapisan keluaran yaitu 46. Iterasi yang terjadi sebesar 511 kali.

Penelitian yang lain "Penerapan Metode Jaringan Syaraf Tiruan *Backpropagation* Dalam Pengenalan Pola Aksara *Hancanaraka"*. Penelitian dilakukan oleh Sugeng Winardi dan Hamzah (2014). Hasil yang diperole sistem aplikasi ini berhasil diimplementasikan dengan baik, meskipun masih ada kekurangan yaitu belum bisa mengenali apabila masih terdapat noise yang terlalu besar. Hasil terbaik didapat dari percobaan huruf/aksara "JA" akurasi yang didapat 98%.

Pujiatus Syahara (2017) melakukan penelitian "Indentifikasi Huruf *Hijaiyah* Tulisan Tangan Menggunakan Jaringan Syaraf Tiruan *Backpropagation".* Nilai akurasi tertinggi yang didapat sebesar 51.33% dari 150

sampel data uji pada huruf "Ta" dengan jumlah *input* layer 300, *hidden* layer 60, dan *output* layer 30.

Donny Avianto (2016) meneliti tentang "Pengenalan Pola Karakter Plat Nomor Kendaraan Menggunakan Algoritma Momentum *Backpropagation Neural Network".* Sistem mampu melakukan pengenalan pola dengan baik, dengan melakukan percobaan dari 276 karakter yang terdiri dari huruf dan angka, mampu mengenali sebanyak 268. Sehingga akurasi yang didapat mencapai 97,01%. Penulis mengharapkan pada masa yang akan datang ada penelitian selanjutnya yang dapat mengenali karakter palat nomor pada kendaraan bergerak bahkan dalam kecepatan tinggi.

Farqi dan Erwien (2015) melakukan penelitian tentang aksara *carakan* dengan judul "Aplikasi Aksara *Carakan* Madura Dengan Menggunakan Metode *Backpropagation"*. Berdasarkan penelitian tersebut diperoleh hasil bahwa metode ini dapat mengenali huruf aksara *carakan* Madura sebesar 7,333%, dengan jumlah input = 256 neuron, hidden layer = 3 neuron, output layer = 1 neuron, dan learning rate = 0,2.

Sedangkan penelitian yang diusuilakan adalah Identifikasi aksara *lontaraq* tulisan tangan mengunakan metode ekstraksi fitur binerisasi pola huruf dan metode pengenalan pola menggunakan *backpropagation*

Untuk mempudahkan membandingkan tinjauan pustaka dengan penelitian sebelumnya, maka dibuat tabel 2.1.

Tabel 2.1 Perbandingan Penilitian Sebelumnya

Penulis dan Tahun	Objek	Metode Ekstraksi Fitur	Metode Pengenalan Pola	Hasil
Nicolaus Euclides Wahyu Nugroho (2016)	Huruf Jepang	Binerisasi Pola Huruf	*Backpropagation*	Tingkat akurasi tertinggi didapat pada percobaan 7 sebesar 86.63%.
Sugeng Winardi & Hamzah (2014)	Aksara *Hanaca-raka*	Binerisasi Pola Huruf	*Backpropagation*	Sistem aplikasi ini berhasil diimplementasikan dengan baik, meskipun masih ada kekurangan yaitu belum bisa mengenali apabila masih terdapat noise yang terlalu besar.
Pujiatus Syahara (2017)	Huruf Hijaiyah Tulisan Tangan	Binerisasi Pola Huruf	*Backpropagation*	Nilai akurasi yang diperole sebesar 51.33% dari 150 sampel data uji.
Donny Avianto (2016)	Plat Nomor Kendaraan	*Citra biner*	*Momentum Backpropagation Neural Network*	sistem mampu melakukan pengenalan pola dengan baik. Hasilnya, dari total 276 karakter yang terdiri dari huruf dan angka, sistem

				mampu mengenali 268 karakter diantaranya. Sehingga akurasi sistem pada penelitian ini mencapai 97,01%.

Tabel 2.1 Perbandingan Penilitian Sebelumnya

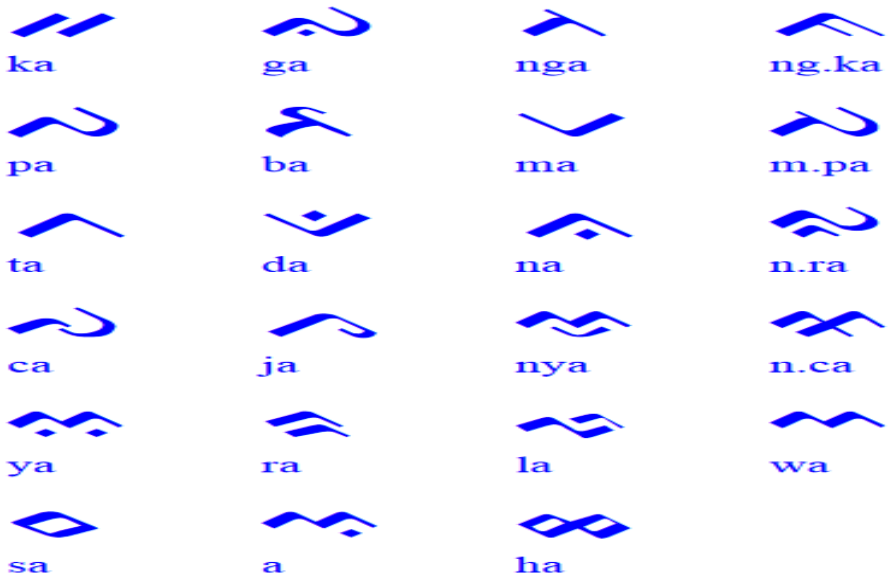
Penulis dan Tahun	**Objek**	**Metode Ekstraksi Fitur**	**Metode Pengenalan Pola**	**Hasil**
Erwien Tjipta W & Iswan Wahyu Al Farqi (2015)	Aksara *Carakan* Madura	Binerisasi Pola Huruf	*Backpropagation*	Berdasarkan pengujian sistem, dengan jumlah input = 256 neuron, *hidden layer* = 3 neuron, *output layer* = 1 neuron, dan *learning rate* = 0,2 dapat mengenali huruf aksara *carakan* Madura sebesar 78,333 %.
Usulan Peneliti (2018)	Aksara *Lontara-q* tulisan tangan	Binerisasi Pola huruf	*Backpropagation*	Nama aksara dan akurasi hasil identifikasi

2.2. Dasar Teori

2.2.1. Aksara *Lontaraq*

Menurut anggapan masyarakat Makassar, huruf *lontaraq* dilatar belakangi oleh suatu kepercayaan atau falsafah "Sulapa' Appa" (empat persegi alam semesta), yakni: Butta (tanah), Je'ne (air), Anging (angin), dan Pepe' (api). Demikian pula, kemungkinan besar Daeng Pammate menciptakan huruf lontara karena berangkat dari kepercayaan tersebut (Nuah, 2017).

Sementara menurut budayawan Prof Mattulada (alm), bentuk lontara berasal dari "sulapa eppa wala suji". Wala suji berasal dari kata wala yang artinya pemisah/pagar/penjaga dan suji yang berarti putri. Wala Suji adalah sejenis pagar bambu dalam acara ritual yang berbentuk belah ketupat. Sulapa eppa (empat sisi) adalah bentuk mistis kepercayaan Bugis-Makassar klasik yang menyimbolkan susunan semesta, api-air-angin-tanah (Nuah, 2017). Berikut adalah bentuk-bentuk dari aksara *lontaraq* bisa dilihat pada Gambar 2.1



Gambar 2.1 Aksara *Lontaraq* (sumber: www.unhas.ac.id)

2.2.2. Pengolahan Citra Biner dan Ekstaksi Fitur

Pengolahan citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsi visual. Proses ini mempunyai ciri data masukan dan informasi keluaran yang berbentuk citra. Istilah pengolahan citra digital secara umum didefinisikan sebagai pemrosesan citra dua dimensi dengan komputer. Pengolahan citra adalah pemrosesan citra, khususnya dengan menggunakan komputer, menjadi citra yang kualitasnya lebih baik (Fennisya, 2017). Jadi pengolahan citra ini dilakukan untuk membuat citra yang sebelumnya kualitasnya kurang bagus setelah citra tersebut diolah akan menghasilkan keluaran citra dengan kualitas yang lebih baik.

Feature Extraction atau ekstraksi fitur merupakan suatu pengambilan ciri / *feature* dari suatu bentuk yang nantinya nilai yang didapatkan akan dianalisis untuk proses selanjutnya. *Feature extraction* dilakukan dengan cara menghitung jumlah titik atau pixels yang ditemui dalam setiap pengecekan, dimana pengecekan dilakukan dalam berbagai arah *tracing* pengecekan pada koordinat kartesian dari citra digital yang dianalisis, yaitu vertikal, horizontal, diagonal kanan, dan diagonal kiri (Ashari, 2016).

2.2.3. Citra Biner

Citra biner adalah citra dimana piksel-pikselnya hanya memiliki dua buah nilai derajat keabuan (*grayscale*) yaitu hitam dan putih. Pixel-pixel (picture elements) suatu objek akan bernilai 1 sedangkan pixel–pixel latar belakang bernilai 0. Pada waktu menampilkan gambar, 0 adalah putih dan 1 adalah hitam. Jadi, pada citra biner, latar belakang berwarna putih sedangkan objek berwarna hitam (Fennisya, 2017). Penerapan citra biner sudah sering kita temui dalam

kehidupan sehari-hari salah satu contoh adalah *bar code* yang sering terdapat pada kemasan suatu produk, bar code termasuk citra biner, gambar/objek(hitam) sebagai bernilai 1 dan latar belakang(putih) bernilai 0.

Membutuhkan memori yang kecil dikarenakan nilai derajat keabuan hanya membutuhkan representasi 1 bit bisa sebagai alasan untuk menggunakan citra biner, selain itu keuntungan yang lain bisa menghemat waktu pemrosesan lebih cepat dengan citra hitam-putih karena banyak operasi pada citra biner yang dilakukan sebagai operasi logika (AND, OR, NOT).

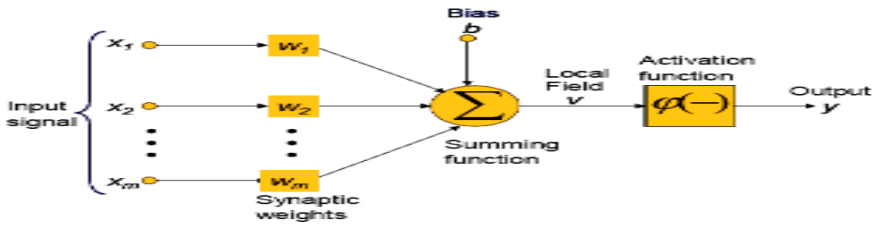
2.2.4. Jaringan Syaraf Tiruan (JST)

Menurut Fu (1994) sebuah jaringan syaraf tiruan merupakan simulasi dari otak biologis yang bertujuan untuk belajar mengenali pola data dan mensimulasikan proses belajar adaptif biologis, walau dalam skala yang sangat sederhana (Halim, 2012). Jaringan syaraf tiruan dibuat untuk bisa mengenali pola suatu data seperti kerja otak manusia karena jaringan syaraf tiruan adalah salah satu representasi otak manusia.

Pada jaringan syaraf tiruan, terdapat beberapa neuron-neuron yang kemudian akan di satukan dalam satu layer yang disebut dengan lapisan neuron. Neuron-neuron yang sudah disatukan akan dihubungkan dengan layer-layer yang lain. Jadi, semua informasi yang didapat dari neuron yang satu akan disampaikan dengan semua layer yang ada, proses penyampaian informasi akan dimulai dari lapisan *input* kemudian melewati lapisan *hidden* untuk menuju lapisan *output*.

Prinsip jaringan syaraf tiruan (Riadi. 2016) ditentukan oleh tiga elemen dasar model saraf, yaitu:

1. Satu set dari sinapsis, atau penghubung yang masing-masing digolongkan oleh bobot atau kekuatannya.
2. Sebuah penambah untuk menjumlahkan sinyal-sinyal input. Ditimbang dari kekuatan sinaptik masing-masing neuron.
3. Sebuah fungsi aktivasi untuk membatasi amplitudo output dari neuron. Fungsi ini bertujuan membatasi jarak amplitude yang diperbolehkan oleh sinyal output menjadi sebuah angka yang terbatas.



Gambar 2.2 Prinsip Dasar JST (Sumber: https://media.neliti.com)

Pada gambar 2.2, Y menerima masukan dari neuron x1, x2, dan x3, dengan bobot hubungan masing-masing adalah w1, w2, dan w3. Ketiga impuls neuron yang ada dijumlahkan dengan (persamaan 2.1)

Net = x1w1 + x2w2 + x3w3..... (2.1)

Besarnya impuls yang diterima oleh Y mengikuti fungsi aktivasi y = f(net). Apabila nilai fungsi aktivasi cukup kuat, maka sinyal akan diteruskan. Nilai fungsi aktivasi (keluaran model jaringan) juga dapat dipakai sebagai dasar untuk mengubah bobot (Siang, 2004).

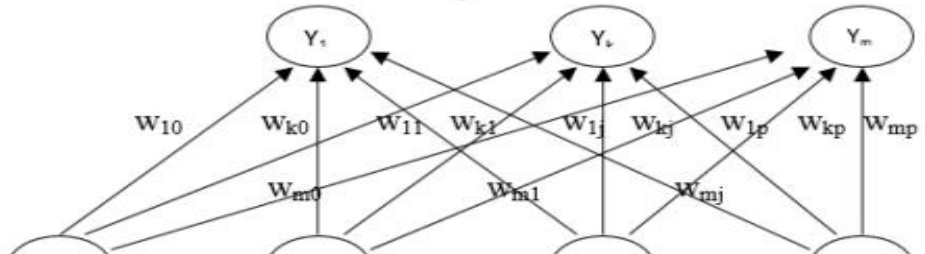
2.2.5. *Backpropagation*

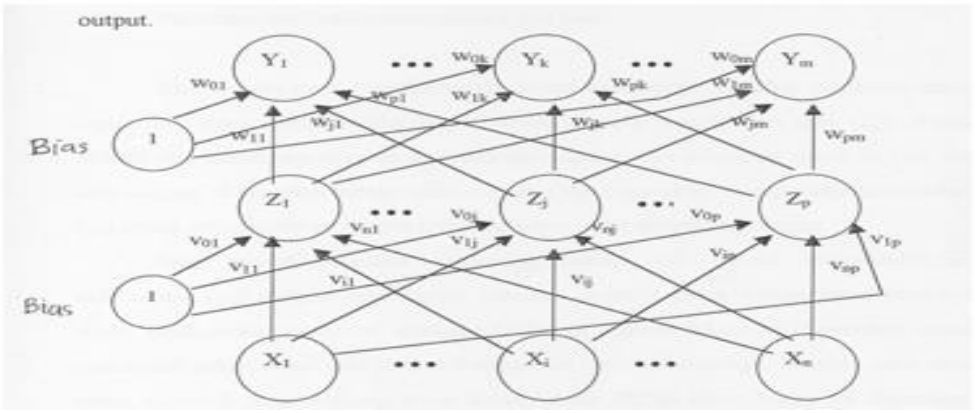
Salah satu metode pelatihan terawasi pada jaringan syaraf adalah metode *Backpropagation*. Algoritma *Backpropagation* merupakan sebuah algoritma yang dapat memperkecil tingkat suatu error dengan menyesuaikan bobot yang berdasarkan target serta output yang diharapkan. Secara umum, *Backpropagation* memiliki langkah-langkah utama diantaranya pengambilan input, penelusuran *error*, dan kemudian penyesuaian bobot. Dalam metode *backpropagation* algoritma yang dikerjakan adalah inisialisasi bobot, komputasi *forward* dan *backpropagation* dan inisialisasi kondisi *stopping* berdasarkan nilai batas error atau jumlah batas *epoch*. Apa yang dimaksud dengan *epoch? Epoch* sendiri merupakan rangkaian langkah-langkah dari dalam pembelajaran ANN (*Artificial neural network*) atau dalam bahasa indonesia disebut Jaringan saraf tiruan. Kemudian satu *epoch* dapat diartikan sebagai satu pembelajaran ANN itu sendiri (Fauzia, 2018).

Dalam algoritma *backpropgation* menggunakan lapisan-lapisan untuk menghasilkan sebuah keluaran atau *output,* yaitu:

1. Lapisan masukan (layer *input*)
2. Lapisan tersembunyi (layer *hidden*)
3. Lapisan keluaran (layer *output*)

Layar *input* tidak melakukan proses apapun, pada layer *input* hanya memasukkan ke layer *hidden,* layer *hidden* berfungsi sebagai ruang untuk menyesuaikan bobot yang untuk mendapatkan nilai yang mendekati target *output* yang diinginkan. Arsitektur jaringan syaraf tiruan *backpropagation* bisa dilihat pada gambar 2.3.





Gambar 2.3 Arsitektur JST *Backpropagation*(Sumber: jurnal.unpad.ac.id**)**

Penggunaan metode *backpropagation* memiliki 2 tahap yaitu tahap pelatihan atau tahap pembelajaran dimana pada tahap ini digunakan untuk melatih dengan berbagai data yang diberikan. Selanjutnya tahap pengujian atau penggunaan pada tahap ini dilakukan untuk menguji pada tahap pelatihan. Namun pada tahap pelatihan terdiri dari tiga langkah, yaitu:

1. Data dimasukkan ke input jaringan (*feedforward*)
2. Perhitungan dan propagasi balik dari error yang bersangkutan
3. Pembaharuan (*adjustment*) bobot bias

Pada fase *feedforward*, pola yang dimasukkan akan dihitung maju yang mulai dari lapisan *input* sampai lapisan *output* menggunakan fungsi aktivasi. Dalam pelatihan ini fungsi aktifasi yang digunakan adalah fungsi sigmoid biner, fungsi ini memiliki range nilai 0 sampai 1 karena. Dalam fase *backpropagation* (langkah kedua) menghitung nilai kesalahan, kemudian akan di propagasikan mundur mulai dari garis yang berhubungan langsung dengan unit-unit di layer *output*. Fase ke tiga (*adjustment*) bertujuan untuk memodifikasi bobot dengan

tujuan menurunkan kesalahan yang terjadi. Fase-fase ini akan terus berlanjut dan hanya akan berhenti jika kondisi untuk berhenti terpenuhi.

Fungsi sigmoid biner

$f_1(x)=1/(1+e^{-x})$ ..... (2.2)

dengan turunan

$f_1'(x)=f_1(x)(1-f_1(x))$ ..... (2.3)

Algoritma *Backpropagation:*

Step 0 : Inisialisasi bobot dan bias baik bobot maupun bias dapat diset dengan sembarang angka (acak) dan biasanya angka di sekitar 0 dan 1 atau -1 (bias positif atau negatif)

Step 1 : Jika *stopping* condition masih belum terpenuhi, jalankan step 2-9.

Step 2 : Untuk setiap data training, lakukan step 3-8.

1. Umpan maju (feedforward)

Step 3 : Setiap unit yang dipakai di sini adalah *input* training data yang sudah diskalakan.

Step 4 : Setiap *hidden* unit (Z_j ,j=1,…,p) akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot, termasuk biasnya dengan (persamaan 2.4)

$$z_{in_j} = vo_j + \sum_{i=1}^{n} x_i v_{ij} \quad \text{..... (2.4)}$$

dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal output dari *hidden* unit yang bersangkutan, melalui (persamaan 2.5),

$$z_j = f(z_{in_j}) \text{..... (2.5)}$$

lalu mengirim sinyal *output* ini ke seluruh unit pada unit output

Keterangan:

x = *input* vektor pelatihan x= (x_1,…., x_i,….., x_n)

v_{0j} = bias pada unit tersembunyi j

Z_j = unit tersebunyi j *input* jaringan Z_j disimbolkan dengan Z_in_j

∑ = sigma

n = Jumlah data latih

Step 5 : Setiap unit *output* (Y_k ,k=1,…,m) akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot, termasuk biasnya dengan (persamaan 2.6),

$$y_{in_k} = w_{ok} + \sum_{j=1}^{p} z_j w_{jk} \quad \textbf{(2.6)}$$

dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output* dari unit *output* yang bersangkutan, melalui (persamaan 2.7):

$$y_k = f(y_{in_k}) \quad \textbf{(2.7)}$$

Keterangan:

Y_k = unit *output input* jaringan ke Y_k disimbolkan dengan y_in_k

w_{ok} = nilai bias

2. Propagasi balik error (backpropagation o f error)

Step 6 : Setiap unit *output* (Y_k ,k=1,…,m) menerima suatu target (*output* yang diharapkan) yang akan dibandingkan dengan *output* yang dihasilkan dengan (persamaan 2.8).

$$\delta_k = (t_k - y_k) f'(y_{in_k}) \quad \textbf{(2.8)}$$

Faktor δ_k ini digunakan untuk menghitung koreksi error (ΔW_{jk}) yang nantinya akan dipakai untuk memperbaharui W_{jk}, dengan (persamaan 2.9):

$$\Delta w_{jk} = \alpha \delta_k z_j \quad \textbf{(2.9)}$$

Selain itu juga dihitung koreksi bias ΔW_{0k} yang nantinya akan dipakai untuk memperbaharui W_{0k}, dengan (persamaan 2.10):

$$\Delta w_{ok} = \alpha \delta_k \quad \textbf{(2.10)}$$

Faktor δ_k ini kemudian dikirimkan ke layer di depannya

Keterangan:

y = *output neuron*

t = target data latih

δ_k = informasi tentang kesalahan pada unit Y_k yang disebarkan kembali ke unit tersembunyi.

α = *learning rate*

Δ = delta

ΔW_{jk} = koreksi error

Step 7 : Setiap *hidden* unit (Zj,j=1,…,p) menjumlah *input* delta (yang dikirim dari layer pada step 6) yang sudah berbobot, (persamaan 2.11):

$$\delta_{in_j} = \sum_{k=1}^{m} \delta_k w_{jk} \quad \textbf{(2.11)}$$

Hasilnya dikalikan dengan turunan dari fungsi aktivasi yang digunakan jaringan untuk menghasilkan faktor koreksi error δ_j, (persamaan 2.12):

$$\delta j = \delta_{in_j} f'\left(z_{in_j}\right) \quad \textbf{(2.12)}$$

Faktor δ_j ini digunakan untuk menghitung koreksi error (ΔV_{ij}) yang nantinya akan dipakai untuk memperbaharui V_{ij}, (persamaan 2.13):

$$\Delta v_{ij} = \alpha \delta_j x_i \quad \textbf{(2.13)}$$

Selain itu juga dihitung koreksi bias ΔV_{0j} yang nantinya akan dipakai untuk memperbaharui V_{0j}, (persamaan 2.14):

$$\Delta v_{oj} = \alpha \delta_j \text{ ..... (2.14)}$$

Keterangan:

δ_j = informasi tentang kesalahan dari lapisan *output* ke unit tersembunyi Z_j

X_i = unit *input* I

ΔV_{ij} = koreksi error bias pada unit tersembunyi j

3. Pembaharuan bobot dan bias:

Step 8 : a. Setiap unit output (Y_k ,k=1,…,m) akan memperbaharui bias dan bobotnya dengan setiap hidden unit. (persamaan 2.15):

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \text{ ..... (2.15)}$$

b. Demikian pula untuk setiap hidden unit akan memperbaharui bias dan bobotnya dengan setiap unit input. (persamaan 2.16):

$$v_{ij}(baru) = v_{ij} + \Delta v_{ij} \text{ ..... (2.16)}$$

Keterangan:

v_{0j} =bias pada unit tersembunyi j

Step 9 : Memeriksa stopping condition, jika stop condition telah terpenuhi, maka pelatihan jaringan dapat dihentikan.

4. *Stopping Condition*

Untuk menentukan stopping condition terdapat dua cara yang biasa dipakai, yaitu:

a. Membatasi iterasi yang ingin dilakukan.

Misalnya jaringan akan dilatih sampai iterasi yang ke-500. Yang dimaksud dengan satu iterasi adalah perulangan step 3 sampai step 8 untuk semua training data yang ada.

b. Membatasi error.

Misalnya menentukan besar Mean Square Error antara output yang dikehendaki dan output yang dihasilkan oleh jaringan. Jika terdapat sebanyak m training data, maka untuk menghitung Mean Square Error digunakan (persamaan 2.17)::

$$\mathbf{MSE} = \mathbf{0,5X}\{(\boldsymbol{t_{k1}} - \mathbf{y_{k1}})^{\mathbf{2}} + (\boldsymbol{t_{k2}} - \mathbf{y_{k2}})^{\mathbf{2}} + \cdots + (\boldsymbol{t_{km}} - \mathbf{y_{km}})^{\mathbf{2}}\} \textbf{....(2.17)}$$

Keterangan:

MSE = Mean Square Error

t = *output* vector target t= (t_1,…., t_k,….., t_m)

Y_k = unit *output*

5. Tahap pengujian & Penggunaan

Mengimplementasikan metode backpropagation pada tahap pengujian sama seperti proses belajar, tetapi hanya pada bagian umpan majunya saja :

Step 0 : Inisialisasi bobot sesuai dengan bobot yang telah dihasilkan pada proses pelatihan di atas.

Step 1 : Untuk setiap input, lakukan step 2-4.

Step 2 : Untuk setiap input i=1,…,n skalakan bilangan dalam range fungsi aktivasi seperti yang dilakukan pada proses pelatihan di atas.

Step 3 : Untuk j=1,…,p:

$$\boldsymbol{z_{in_j}} = \boldsymbol{v_{oj}} + \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x_i}\, \boldsymbol{v_{ij}} \textbf{ ..... (2.18)}$$

$$z_j = f(z_{in_j}) \quad \text{(2.19)}$$

Step 4 : Untuk k=1,…,m:

$$y_{in_k} = w_{ok} + \sum_{j=1}^{p} z_j w_{jk} \quad \text{(2.20)}$$

$$y_k = f(y_{in_k}) \quad \text{(2.21)}$$

2.2.6. Akurasi dan Error Rate

Pengukuran terhadap kinerja suatu sistem klasifikasi merupakan hal yang penting. Kinerja sistem klasifikasi menggambarkan seberapa baik sistem dalam mengklasifikasikan data. *Confusion matrix* merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode klasifikasi. Pada dasarnya *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang seharusnya.

Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat 4 (empat) istilah sebagai representasi hasil proses klasifikasi. Keempat istilah tersebut adalah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN). Nilai *True Negative* (TN) merupakan jumlah data negatif yang terdeteksi dengan benar, sedangkan *False Positive* (FP) merupakan data negatif namun terdeteksi sebagai data positif. Sementara itu, *True Positive* (TP) merupakan data positif yang terdeteksi benar. *False Negative* (FN) merupakan kebalikan dari *True Positive*, sehingga data posifit, namun terdeteksi sebagai data negative(Achmad, 2017).

Table 2.2 *Confusion Matrix*

Kelas	Terklasifikasi Positif	Terklasifikasi Negatif
Positif	TP (True Positive)	FN (False Negative)
Negatif	FP (False Positive)	TN (True Negative)

Berdasarkan nilai *True Negative* (TN), *False Positive* (FP), *False Negative* (FN), dan *True Positive* (TP) dapat diperoleh nilai akurasi, presisi dan *recall*. Nilai akurasi menggambarkan seberapa akurat sistem dapat mengklasifikasikan data secara benar. Dengan kata lain, nilai akurasi merupakan perbandingan antara data yang terklasifikasi benar dengan keseluruhan data. Sedangkan *Error Rate* adalah kasus yang diidentifikasi salah dengan sejumlah semua kasus. Untuk memperoleh akurasi dapat menggunakan rumus berikut:

$$\boldsymbol{Akurasi} \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+TN+FP+FN}} \boldsymbol{x\ 100\%} \quad \textbf{(2.22)}$$

$$\boldsymbol{Error\ Rate} \frac{\boldsymbol{FP+TN}}{\boldsymbol{TP+TN+FP+FN}} \boldsymbol{x\ 100\%} \quad \textbf{(2.23)}$$

2.2.7. Matlab

Matlab adalah sebuah bahasa dengan kinerja tinggi untuk komputasi masalah teknik. MATLAB mengintegrasikan komputasi, visualisasi, dan pemrograman dalam suatu model yang sangat mudah untuk pakai dimana masalah-masalah dan penyelesaiannya diekspresikan dalam notasi matematika yang familiar(Febri, 2016). Penggunaan Matlab meliputi

1. Matematika dan komputasi
2. Pembentukan algoritma
3. Akusisi data
4. Pemodelan, simulasi, dan pembuatan *prototype*
5. Analisa data, explorasi, dan visualisasi
6. Grafik keilmuan dan bidang rekayasa

Nama MATLAB merupakan singkatan dari *matrix laboratory*. Dalam lingkungan perguruan tinggi teknik, MATLAB merupakan perangkat pilihan untuk penelitian dengan produktifitas yang tingi, pengembangan dan analisanya. Fitur-fitur MATLAB sudah banyak dikembangkan, dan lebih kita kenal dengan nama *toolbox*. Sangat penting bagi seorang pengguna MATLAB, *toolbox* mana yang mandukung untuk *learn* dan *apply technology* yang sedang dipelajarinya. *Toolbox* ini merupakankumpulan dari fungsi-fungsi MATLAB (*M-files*) yang telah dikembangkan ke suatu lingkungan kerja MATLAB untuk memecahkan masalah dalam kelas particular. Area-area yang sudah bisa dipecahkan dengan toolbox saat ini meliputi pengolahan sinyal, system kontrol, *neural networks, fuzzy logic, wavelets*, dan lain-lain(Febri, 2016).

Sebagai sebuah sistem, MATLAB tersusun dari 5 bagian utama:

1. **Development Environment,** merupakan sekumpulan perangkat dan fasilitas yang membantu kita untuk menggunakan fungsi-fungsi dan file-file MATLAB. Beberapa perangkat ini merupakan sebuah **Graphical User Interfaces (GUI).** Termasuk didalamnya adalah *MATLAB desktop* dan *Command Window, Command History*, sebuah editor dan *debugger*, dan *browsers* untuk melihat *help, workspace, files*, dan *search path.*
2. **Matlab Mathematical Function Library,** merupakan sekumpulan algoritma komputasi mulai dari fungsi-fungsi dasar sepertri: *sum, sin, cos,* dan *complex arithmetic,* sampai dengan fungsi-fungsi yang lebih komplek seperti *matrix inverse, matrix eigenvalues, Bessel functions,* dan *fast Fourier transforms.*

3. **Matlab Language**, merupakan suatu *high-level matrix/array language* dengan *control flow statements, functions, data structures, input/output,* dan *fitur-fitur object-oriented programming*. Ini memungkinkan bagi kita untuk melakukan kedua hal baik "pemrograman dalam lingkup sederhana" untuk mendapatkan hasil yang cepat, dan "pemrograman dalam lingkup yang lebih besar" .
4. **Graphics**, matlab memiliki fasilitas untuk menampilkan vector dan matrices sebagai suatu grafik. Didalamnya melibatkan *high-level functions* (fungsi-fungsi level tinggi) untuk visualisasi data dua dikensi dan data tiga dimensi, *image processing, animation*, dan *presentation graphics*. Ini juga melibatkan fungsi level rendah yang memungkinkan bagi kita untuk membiasakan diri untuk memunculkan grafik mulai dari bentuk yang sederhana sampai dengan tingkatan graphical user interfaces pada aplikasi matlab.
5. **Matlab Application Program Interface (API),** merupakan suatu *library* C yang memungkinkan program yang telah kita tulis dalam bahasa dan *Fortran* mampu berinterakasi dengan matlab.