# PENGENALAN KATA TULISAN TANGAN MENGGUNAKAN JARINGAN SYARAF TIRUAN OPTICAL BACKPROPAGATION

**SKRIPSI**

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Oleh:

**ISNAWAN TRI WIDODO**

**0510960033-96**


**PROGRAM STUDI ILMU KOMPUTER**

**JURUSAN MATEMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS BRAWIJAYA**

**MALANG**

**2011**

**LEMBAR PENGESAHAN SKRIPSI**

**Pengenalan Kata Tulisan Tangan Menggunakan Jaringan Syaraf Tiruan Optical Backpropagation**

**Oleh :**

**ISNAWAN TRI WIDODO**

**0510960033-96**

**Setelah dipertahankan di depan Majelis Penguji**
**Pada tanggal 20 Juli 2011**

**dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Komputer dalam bidang Ilmu Komputer**

| Pembimbing I, | Pembimbing II, |
|---|---|
| **Lailil Muflikhah, S.Kom., M.Sc.** | **Drs. Marji, MT.** |
| **NIP. 197411132005012001** | **NIP. 196708011992031001** |

**Mengetahui,**

**Ketua Jurusan Matematika**

**Fakultas MIPA Universitas Brawijaya**

**Dr. Abdul Rouf Alghofari A., M.Sc**
**NIP.196709071992031001**


UNIVERSITAS BRAWIJAYA

**LEMBAR PERNYATAAN**

**Saya yang bertanda tangan di bawah ini :**

**Nama : Isnawan Tri Widodo**
**NIM : 0510960033-96**
**Jurusan : Matematika**
**Program Studi : Ilmu Komputer**
**Penulis skripsi berjudul : Pengenalan Kata Tulisan Tangan Menggunakan Jaringan Syaraf Tiruan Optical Backpropagation**

**Dengan ini menyatakan bahwa :**

1. **Isi dari Skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Skripsi ini.**
2. **Apabila dikemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

**Demikian pernyataan ini dibuat dengan segala kesadaran.**

**Malang, 20 Juli 2011**

**Yang menyatakan,**

**Isnawan Tri Widodo**
**NIM. 0510960033-96**


UNIVERSITAS BRAWIJAYA

# PENGENALAN KATA TULISAN TANGAN MENGGUNAKAN JARINGAN SYARAF TIRUAN OPTICAL BACKPROPAGATION

## ABSTRAK

Sistem pengenalan tulisan tangan telah banyak diimplementasikan untuk aplikasi pengenalan tulisan tangan, antara lain sistem pemilahan surat otomatis, pengolahan cek di bank dan banyak aplikasi lainnya. Aplikasi pengenalan tulisan tangan banyak menggunakan metode jaringan syaraf tiruan. Hal ini karena tingkat akurasi yang dimiliki cukup tinggi. *Optical Backpropagation* adalah salah satu algoritma dalam jaringan syaraf tiruan yang didesain untuk menyelesaikan beberapa kendala yang berkaitan dengan backpropagation standar. Salah satu aspek penting dari algoritma ini adalah kemampuannya untuk menghindari *local minima* dalam proses pelatihannya dengan kecepatan yang cukup tinggi untuk mencapai konvergensi selama proses pelatihan. Dalam penelitian ini, sampel kata yang didapat akan disegmentasi menjadi karakter mengunakan algoritma *Modified Vertical Histogram.* Dari karakter yang tersegmentasi tersebut, kemudian dilakukan proses pelatihan dan pengenalan. Dari penelitian ini diperoleh struktur jaringan syaraf tiruan Optical Backpropagation dengan jumlah neuron pada *input layer* sebanyak 400 unit, jumlah neuron pada *hidden layer* sebanyak 100 unit, jumlah neuron pada *output layer* sebanyak 400 unit, nilai *learning rate* sebesar 0.004 dengan *max epoch* sebesar 5000. Jaringan syaraf tiruan yang terbentuk mampu mengenali citra tulisan tangan sebesar 96,70% untuk citra yang berisi huruf tunggal yang pernah dilakukan pembelajaran dan hasil pengenalan terhadap citra yang berisi huruf tunggal yang belum dilakukan pembelajaran sebesar 56.41%. Sedangkan tingkat keakuratan sistem secara keseluruhan terhadap kata yang diujikan sebesar 72.45%.

Kata kunci : tulisan tangan latin, jaringan syaraf tiruan, Optical Backpropagation

**HANDWRITING WORDS RECOGNITION USING OPTICAL BACKPROPAGATION NEURAL NETWORK**


ABSTRACT

Handwriting recognition system has been implemented for handwriting applications such as automatic mail voting system, signature validation, processing check at the bank and many others. Some of handwriting application systems used artificial neural network method. That was because of its high level of accuracy. *Optical Back propagation* is one of the algorithms in artificial neural networks designed to overcome several problems associated with standard *back propagation*. One of the important aspects of this algorithm was its ability to avoid *local minima* in the training process with high speed level to achieve convergence during the training process. In this research, handwriting samples obtained will be segmented into characters using a *Modified Vertical Histogram algorithm.* From this segmented character, the training and recognizing process will be done. From this study was obtained the structure of *Optical Back propagation* neural network with the number of neurons in the input layer were 400 units, the number of neurons in the hidden layer were 100 units, the number of neurons in the output layer were 400 units, the value of learning rate were 0004 with the max epoch were 5000. Artificial neural network that has formed was capable to recognize handwriting image were 96.70% for images containing a single letter has ever done learning and recognition results to images that contain single letter that has ever done learning were 56.41%. While, the overall level of accuracy of the system is tested with the word were 72.45%.



Keyword: cursive handwriting, neural networks, Optical Back propagation

## KATA PENGANTAR

Puji syukur kehadirat Allah SWT, hanya dengan rahmat dan karunia yang telah diberikan kepada penulis, sehingga dapat menyelesaikan skripsi yang berjudul "*Pengenalan Kata Tulisan Tangan Menggunakan Jaringan Syaraf Tiruan Optical Backpropagation*".

Skripsi ini merupakan salah satu syarat untuk memenuhi persyaratan akademis untuk menyelesaikan studi di program Sarjana Ilmu Komputer Universitas Brawijaya.

Pada kesempatan ini penulis mengucapkan banyak terima kasih atas segala bantuan dan dedikasi moral maupun material dalam rangka penyusunan skripsi ini.

1. Lailil Muflikhah, S.Kom., M.Sc., dan Drs. Mardji, MT., selaku dosen pembimbing yang telah membimbing dengan bijaksana dan sabar dalam penyusunan skripsi ini.
2. Dr. Abdul Rouf Alghofari, M.Sc., selaku Ketua Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Brawijaya.
3. Drs. Mardji, MT., selaku Ketua Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Brawijaya.
4. Agus Wahyu Widodo, ST., selaku dosen pembimbing akademik.
5. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Brawijaya.
6. Segenap staf dan karyawan Jurusan Matematika Universitas Brawijaya yang telah membantu penyusunan skripsi ini.
7. Ayah, Ibu, dan saudara-saudara, serta keluarga besarku yang tersayang, terima kasih atas dukungan dan doanya.
8. Riesky, Isna sekeluarga, Kholis, dan seluruh teman-teman Prodi Ilmu Komputer serta teman-teman lain yang selalu memberi dukungan dan doanya
9. Seluruh pihak yang tidak dapat disebut secara langsung yang telah memberikan bantuan demi terselesaikannya skripsi ini.

Penulis menyadari bahwa masih banyak kekurangan dalam penulisan laporan ini yang disebabkan oleh keterbatasan kemampuan

dan pengalaman. Oleh karena itu, Penulis sangat menghargai saran dan kritik yang sifatnya membangun.

Penulis berharap semoga laporan ini dapat memberikan manfaat kepada pembaca dan bisa diambil manfaatnya untuk pengembangan selanjutnya.

Malang, 20 Juli 2011

Penulis

## DAFTAR ISI

## DAFTAR GAMBAR

# DAFTAR TABEL

Halaman

## DAFTAR *SOURCECODE*

Halaman


UNIVERSITAS BRAWIJAYA

## DAFTAR LAMPIRAN

UNIVERSITAS BRAWIJAYA

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Tulisan tangan latin (*Cursive*) adalah gaya tulisan tangan yang digunakan untuk menulis secara cepat. Kata "*cursive*" berasal dari bahasa Latin yaitu *cursivus*, yang berarti mengalir. Huruf pada tulisan latin tersambung pada setiap katanya. Tulisan latin sering digunakan dalam kegiatan sehari-hari, seperti mencatat, pengisian formulir dan surat-menyurat. Untuk menulis kita sering memakai pena dan kertas dari pada langsung memakai papan ketik (*keybord*). Setelah catatan tersebut selesai ditulis, maka baru kemudian diketik. Demikian juga untuk mereproduksi naskah-naskah lama dalam tulisan tangan latin sehingga dibutuhkan waktu dan tenaga khusus untuk mengetik ulang. Untuk mengurangi waktu dan tenaga pengetikan tersebut maka diperlukan suatu perangkat lunak pengenal tulisan tangan latin (Sediyono, 2000).

Hasil dari penelitian tentang pengenalan tulisan tangan latin telah banyak diaplikasikan untuk memenuhi kebutuhan manusia. Beberapa aplikasi pengenalan tulisan tangan antara lain sistem pemilahan surat otomatis, validasi tanda tangan, pengolahan cek di bank dan aplikasi lain. Karena manfaat dari pengenalan tulisan tangan inilah kebutuhan akan proses pengenalan tulisan tangan secara akurat semakin diperlukan, dengan menerapkan berbagai metode yang ada.

Aplikasi pengenalan pola, khususnya pengenalan tulisan tangan, banyak menggunakan metode jaringan syaraf tiruan. Hal ini karena tingkat akurasi yang dimiliki cukup tinggi. Dalam penelitian yang dilakukan Hassin (2003), yang membuat pengenalan pola terhadap huruf-huruf arab, menghasilkan tingkat keakuratan 96% untuk pengenalan yang hanya menggunakan jaringan syaraf tiruan saja, dan 92,2 % untuk sistem yang menggunakan *structural clasifiers*.

Meskipun memiliki akurasi yang cukup bagus, algoritma Backpropagation memiliki kelemahan, yaitu pelatihan sering membutuhkan waktu yang lama untuk mencapai konvergensi, dan memungkinkan untuk terjebak ke dalam *local minima* (suatu kondisi dimana pelatihan terhenti pada suatu nilai *error*/galat minimum, namun bukan nilai *error*/galat yang terendah) (Salameh, 2005).

Optical Backpropagation adalah salah satu algoritma dalam jaringan syaraf tiruan yang didesain untuk menyelesaikan beberapa kendala yang berkaitan dengan backpropagation standar, yang

menggunakan fungsi nonlinier, yang diterapkan pada unit keluaran. Salah satu aspek penting dari algoritma ini adalah kemampuannya untuk menghindari *local minima* dalam proses pelatihannya, serta memiliki kecepatan yang cukup tinggi untuk mencapai konvergensi selama proses pelatihan (Otair, 2005).

Dalam penelitian ini akan dilakukan pengujian terhadap tulisan yang tidak hanya berisi satu huruf saja, melainkan beberapa huruf yang membentuk kata dengan tipe tulisan latin bersambung/*cursive*. Oleh karena itu, sebelum proses jaringan syaraf tiruan dilakukan, akan ada proses segmentasi yang akan memisahkan tiap-tiap huruf dalam kata menjadi karakter.

Berdasarkan latar belakang yang telah dipaparkan, maka judul yang diambil dalam skripsi ini adalah "Pengenalan Tulisan Tangan menggunakan Jaringan Syaraf Tiruan Optical Backpropagation"

## 1.2 Perumusan Masalah

Permasalahan yang akan dijadikan objek penelitian pada tugas akhir ini adalah :

1. Bagaimana mengimplementasikan algoritma jaringan syaraf tiruan Optical Backpropagation pada tulisan tangan latin (*cursive handwriting*)
2. Berapa tingkat kebenaran atau akurasi pengenalan tulisan tangan dengan algoritma Optical Backpropagation terhadap sejumlah sampel yang diberikan

## 1.3 Batasan Masalah

Dari permasalahan yang ada di atas, akan diberikan batasan-batasan agar pembahasan masalah ini tidak melebar. Batasan-batasan tersebut yaitu :

1. Citra yang diproses adalah citra berekstensi *Bitmap* (.BMP)
2. Proses pengenalan bersifat *offline*
3. Tulisan yang akan dikenali berupa huruf-huruf latin yang tegak bersambung, tidak ditulis miring (*slant*).
4. Huruf yang dikenali berupa huruf alphabet.

## 1.4 Tujuan

Ada beberapa tujuan yang ingin dicapai dalam melaksanakan penelitian untuk penulisan tugas akhir, yaitu :

1. Mengimplementasikan algoritma jaringan syaraf tiruan Optical Backpropagation pada tulisan tangan latin (*cursive handwriting*)
2. Mengetahui tingkat kebenaran atau akurasi pengenalan tulisan tangan dari sistem yang dilatih menggunakan jaringan syaraf tiruan Optical Backpropagation.

## 1.5 Manfaat

Manfaat yang dapat diperoleh dari penelitian ini antara lain:

1. Sebagai referensi mengenai pengenalan tulisan tangan dengan algoritma Optical Backpropagation pada tulisan tangan latin (*cursive handwriting*) untuk penelitian lebih lanjut

## 1.6 Sistematika penulisan

BAB I : PENDAHULUAN

Bab I berisi tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan penulisan, manfaat penulisan, serta sistematika penulisan skripsi.

BAB II : TINJAUAN PUSTAKA

Dalam bab ini akan dijelaskan mengenai konsep Jaringan Syaraf Tiruan, fungsi aktivasi, proses pembelajaran JST, Algoritma Backpropagation, algoritma Optical Backpropagation.

BAB III : METODOLOGI DAN PERANCANGAN

Dalam bab ini akan dijelaskan mengenai kebutuhan sistem, metode dan langkah yang digunakan dalam pengenalan tulisan tangan, serta contoh perhitungan dengan Jaringan Syaraf tiruan Backpropagation.

BAB IV : IMPLEMENTASI DAN UJI COBA

Pada bab ini dijelaskan implementasi aplikasi, ujicoba, dan analisa hasil.

BAB V : KESIMPULAN DAN SARAN

Bab V berisi tentang kesimpulan dari pembahasan yang telah dilakukan di bab IV serta saran pengembangan dari keseluruhan tahapan pembuatan skripsi ini.

# BAB II
# TINJAUAN PUSTAKA

## 2.1 Pengenalan Tulisan tangan

Pengenalan tulisan tangan adalah proses untuk mengenali tulisan yang ditulis menggunakan alat tulis biasa kedalam betuk digital teks. Penelitian dalam bidang pengenalan tulisan tangan sudah cukup banyak dilakukan, dan mempunyai banyak metode dan strategi. Salah satu faktor penentu adalah mengenai jumlah kata dan penulis yang mampu dikenali. Pada sistem *online*, umumnya pengenalan dibatasi untuk satu orang saja (*mono scriptor*), yaitu si pengguna, sedang pada sistem *offline* umumnya bersifat *multi scriptor*, dimana tulisan yang di-*scanning* berasal dari lebih dari satu orang. Hal ini menunjukkan betapa dibutuhkan sistem yang mampu mengenali berbagai gaya penulisan dari masing-masing orang.

Menurut jenis tulisan yang dikenali, sistem dapat dibagi menjadi Pengenalan Karakter (*Optical Character Recognition*) dan Pengenalan Tulisan Tangan Latin (*Cursive Handwriting)*. Pada pengenalan tulisan tangan bersambung dikenal metode holistik dan analitis. Metode holistik mengenali tulisan per-kata, sedangkan metode analitis mencoba membagi kata menjadi bagian yang lebih kecil, kemudian baru masing-masing bagian tersebut dikenali (Lecolinet, 1994)).

## 2.2 *Preprocessing*

### 2.2.1 *Grayscaling*

Proses *Greyscaling* mengubah gambar menjadi berwarna hitam putih dengan mengubah setiap komponen RGB menjadi bernilai sama.

Untuk mengubah RGB menjadi citra *grayscale* dapat digunakan persamaan 2.1 atau persamaan 2.2 :

$$Grayscale = 0{,}299R + 0{,}587G + 0{,}114B \quad (2.1)$$

Atau

$$Grayscale = \frac{R + G + B}{3} \quad (2.2)$$

Perhitungan nilai *grayscale* yang sebenarnya adalah dengan menggunakan persamaan 2.1. Namun persamaan yang umum

digunakan adalah persamaan 2.2 karena lebih mudah untuk digunakan dan diingat (Budhi, 2006).

### 2.2.2 *Tresholding*

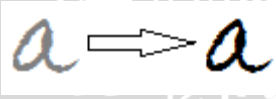
Proses *tresholding* mengubah gambar menjadi gambar biner, dimana ditentukan sebuah nilai *treshold* kemudian piksel yang memiliki nilai di bawah level *treshold* diubah menjadi nilai warna hitam (0 pada nilai biner) dan nilai di atas level *threshold* diset menjadi nilai warna putih (1 pada nilai biner). Proses *treshold* digunakan untuk mengekstrak tulisan (*foreground*) dari latar belakang (*background*) dan menjadikan gambar menjadi biner (Guillevic,1995).

Fungsi pengambangan secara global dinyatakan pada persamaan 2.3

$$f_B(i,j) \begin{cases} 0, & f_g(i,j) \leq T \\ 1, & lainnya \end{cases} \quad (2.3)$$

Keterangan :
$f_B(i,j)$ = citra biner
$f_g(i,j)$ = citra hitam putih
$T$ = nilai ambang yang dispesifikasikan (*threshold*)

Dengan spesifikasi pengambangan, obyek dibuat berwana gelap sedangkan latar belakang berwarna terang. Gambar 2.1 menunjukkan contoh proses *Tresholding.*

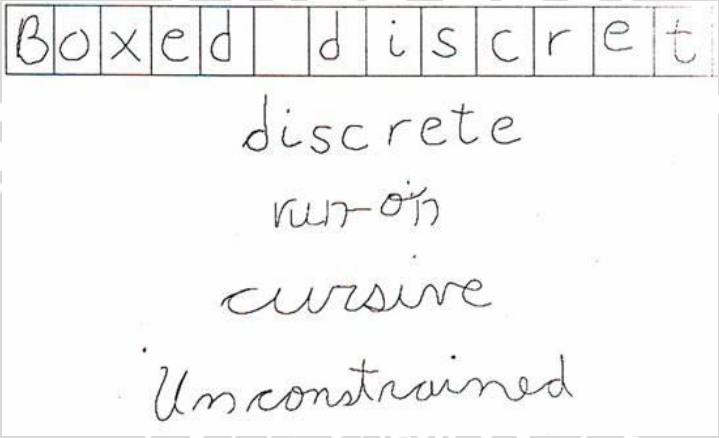


**Gambar 2.1** Contoh proses *Tresholding*

### 2.2.3 Segmentasi

Setelah input data telah dinormalisasi pada derajat tertentu melalui proses sebelumnya, tahap berikutnya dari *preprocessing* adalah segmentasi. Proses segmentasi ini adalah proses untuk memisahkan bagian-bagian tulisan menjadi huruf-huruf. Proses segmentasi tidak kalah pentingnya dengan proses pengenalan, karena kesalahan dalam melakukan segmentasi akan sangat berpengaruh

terhadap benar tidaknya hasil pengenalan huruf. Kesalahan segmentasi biasanya terjadi karena salah menentukan bagian mana dari tulisan yang merupakan huruf selanjutnya atau sebelumnya. Kesalahan yang lain adalah karena suatu goresan pena kadang dapat diartikan lebih dari satu huruf. Secara umum, gaya tulisan tangan dibedakan menjadi lima tipe, yaitu *boxed discrete, discrete, run-on, cursive* dan *unconstrained* (Munawir, 2008). Contoh dari masing-masing gaya tulisan tersebut seperti tampak pada Gambar 2.2.




**Gambar 2.2** Gaya tulisan tangan

Tulisan tipe *box discrete* merupakan yang paling mudah disegmentasi karena tulisan akan terpisah satu sama lain dalam jarak yang seragam. Tipe tulisan *discrete* juga cenderung agak mudah disegmentasi karena huruf-hurufnya yang terpisah. Kesulitan tulisan *discrete* dibanding *box discrete* adalah dalam hal jarak antar huruf yang tak seragam, namun hal ini biasanya diatasi dengan melihat adanya celah diantara masing-masing huruf.

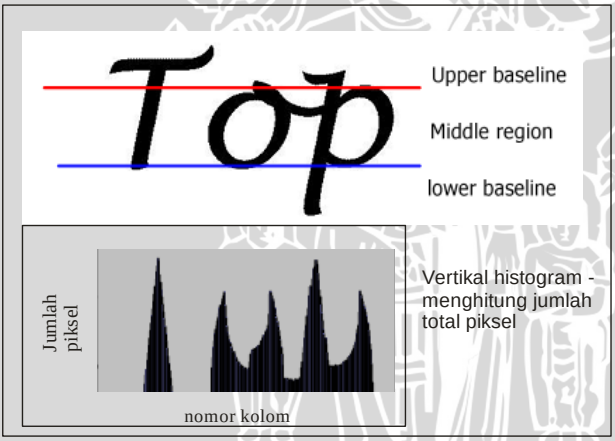
Pada gaya tulisan *run-on*, segmentasi huruf menjadi sukar dan bahkan bisa diartikan berbeda-beda. Bagian huruf sebelumnya kadang menjorok ke huruf berikutnya sehingga hampir tak ada celah sama sekali. Satu hal yang menolong untuk tulisan model *run-on* adalah terangkatnya pena untuk beberapa saat setiap pergantian huruf , namun hal ini hanya bisa dideteksi pada pengenalan huruf secara *online*. Gaya tulisan *cursive* juga sulit disegmentasi karena tulisan tersambung dan tidak terdapat celah sama sekali. Pena hanya terangkat sebentar setiap pergantian kata. Kebanyakan orang bahkan

menggunakan campuran dari *cursive* dan *run-on* sehingga hal ini menimbulkan tantangan tersendiri untuk poses segmentasinya.

Segmentasi bertujuan mengelompokkan piksel-piksel objek menjadi wilayah (*region*) yang mempresentasikan objek. Dalam sistem ini, metode segmentasi yang digunakan untuk memisahkan antara karakter yang satu dengan yang lain adalah *modified vertical histogram*, dengan menjadikan huruf atau objek ke dalam bentuk garis-garis histogram secara vertikal.

### 2.2.3.1 Modified Vertical Histogram

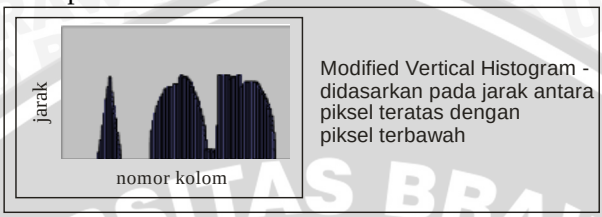
Salah satu pendekatan untuk menganalisis huruf dan sambungan antar huruf adalah dengan menggunakan analisis histogram vertikal. Algoritma ini didasarkan pada distribusi piksel secara vertikal. Histogram dibentuk dengan memproyeksikan dari jumlah total piksel pada tiap-tiap kolom dari citra. Area dengan kepadatan piksel rendah akan diidentifikasi sebagai titik segmentasi. Gambar 2.3 menunjukkan contoh dari histogram vertikal.




**Gambar 2.3** *Vertical Histogram*

Gambar 2.3 menunjukkan bahwa cukup banyak area yang memiliki kepadatan piksel rendah. Hal ini dikarenakan histogram vertikal tidak bisa membedakan perbedaan antara huruf dan sambungan antar huruf. Oleh karena itu, *modified vertical histogram* dikembangkan untuk meningkatkan akurasi dari penentuan sambungan antar huruf.

Konsep dari *modified vertical histogram* dibentuk dengan menghitung jarak antara piksel teratas dan terbawah dari masing-masing kolom pada citra.




**Gambar 2.4** *Modified vertical histogram*

Seperti dapat dilihat pada Gambar 2.4, daerah sambungan antar huruf cukup jelas sehingga mudah untuk disegmentasi. Salah satu kelemahan dari *modified vertical histogram* adalah algoritma ini tidak cocok untuk karakter-karakter yang penulisannya saling bertumpuk (*overlapped*) (Cheng, 2005 )

### 2.2.4. Normalisasi

Normalisasi pada pengolahan citra berarti mentransformasikan citra ke bentuk citra normal yang sesuai dengan kebutuhan. Besar dan kecil ukuran pada citra normalisasi tidak sesuai ukuran yang diambil dari citra semula. Dalam sistem ini, penulis menggunakan penskalaan dari citra semula ke bentuk citra normalisasi. Penskalaan ini tergantung besar dan kecil ukuran pada citra semula artinya tidak berarti apakah citra membesar atau mengecil tergantung ukuran citra semula. Penskalaan citra dirumuskan dengan persamaan 2.4 dan 2.5 (Munir,2004)

$$x' = S_x . x \quad (2.4)$$

$$y' = S_y . y \quad (2.5)$$

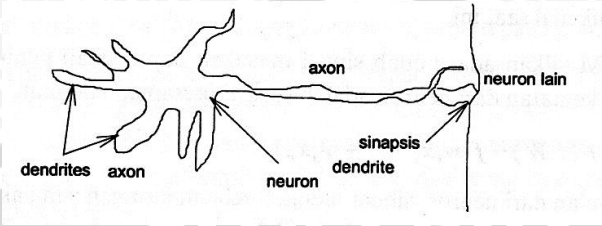
## 2.3 Jaringan Syaraf Tiruan

### 2.3.1 Pengertian Jaringan Syaraf Tiruan

Jaringan syaraf tiruan adalah merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan di sini digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran (Kusumadewi,2003).

Neuron adalah satuan unit pemroses terkecil pada otak. Bentuk sederhana sebuah neuron yang oleh para ahli dianggap

sebagai satuan unit pemroses tersebut seperti digambarkan pada Gambar 2.5.

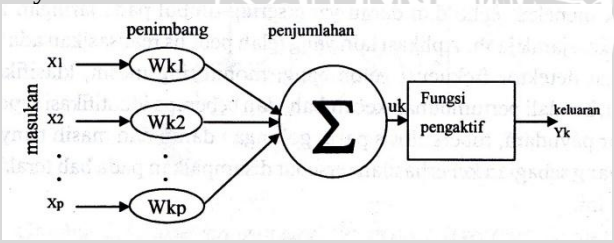



**Gambar 2.5** Struktur sederhana sebuah neuron

Jaringan otak manusia tersusun tidak kurang dari $10^{13}$ buah neuron yang masing-masing terhubung oleh sekitar $10^{15}$ buah dendrite. Fungsi dendrite adalah sebagai penyampai sinyal dari neuron tersebut ke neuron yang terhubung dengannya. Sebagai terusan keluaran, setiap neuron memiliki axon, sedangkan bagian penerima sinyal disebut sinapsis (*synapse*). Secara umum jaringan syaraf terbentuk dari satu trilyun (bahkan lebih) struktur dasar neuron yang terinterkoneksi dan terintegrasi antara satu dengan yang lain oleh satu trilyun sinapsis sehingga dapat melakukan aktifitas menyimpan(*memorize*) pengetahuan secara teratur dan terus menerus sesuai dengan kebutuhan (Purnomo, 2006).

### 2.3.2 Model Dasar Jaringan Syaraf Tiruan

Tiruan neuron dalam struktur jaringan syaraf tiruan adalah elemen pemroses seperti pada Gambar 2.6 yang dapat berfungsi seperti halnya sebuah neuron.




**Gambar 2.6** Struktur neuron jaringan syaraf

Sejumlah sinyal masukan x dikalikan dengan masing-masing penimbang yang bersesuaian W. Kemudian dilakukan penjumlahan dari seluruh hasil perkalian tersebut dan keluaran yang dihasilkan dilalukan kedalam fungsi pengaktif untuk mendapatkan tingkatan derajat sinyal keluarannya F(x,W). Walaupun masih jauh dari

sempurna, namun kinerja dari tiruan neuron ini identik dengan kinerja dari sel biologi yang dikenal saat ini.

Misalkan ada n buah sinyal masukan dan n buah penimbang, fungsi keluaran dari neuron adalah seperti Persamaan 2.6

$$F(x,W) = f\ (w_1 x_1 + ... + w_n x_n) \qquad (2.6)$$

Kumpulan dari neuron dibuat menjadi sebuah jaringan yang akan berfungsi sebagai alat komputasi. Jumlah neuron dan struktur jaringan untuk setiap masalah yang akan diselesaikan adalah berbeda. Demikian pula dengan penimbang diantara masing-masing neuron yang terhubung, besarnya akan ditentukan pada saat jaringan dilatih dengan sekumpulan sampel data.

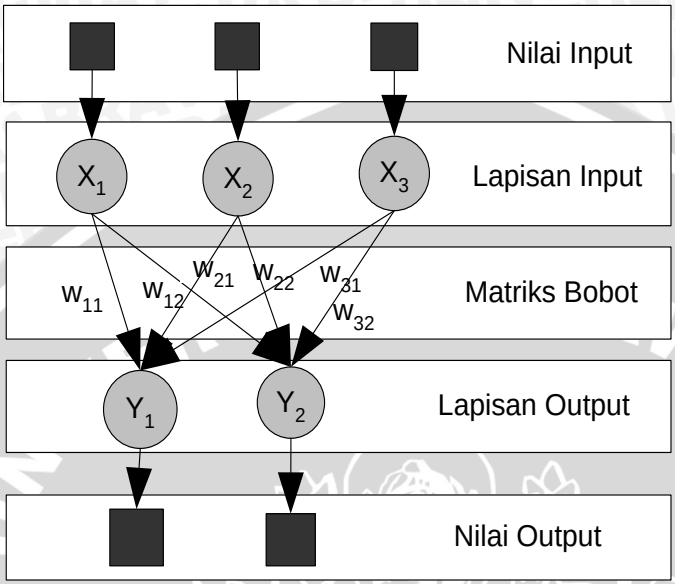
Struktur pada Gambar 2.6 adalah sebuah model tiruan neuron sederhana yang dinamakan elemen pemroses ADALINE (*Adaptive Linear Neuron*). Neuron tunggal tersebut dinyatakan sebagai neuron/unit logika ambang (*treshold logic neuron/unit*) dengan keluaran bipolar (-1 dan +1). Masukan unit tersebut jua bipolar, tanpa ada batasan apakah bernilai biner atau riil. Model jaringan yang terbentuk oleh ADALINE disebut MADALINE (Purnomo, 2006).

### 2.3.3 Arsitektur Jaringan Syaraf Tiruan

Hubungan antara neuron dalam jaringan syaraf mengikuti pola tertentu tergantung pada arsitektur jaringan syarafnya. Menurut bentuknya, ada 3 macam arsitektur jaringan syaraf, yaitu (Kusumadewi, 2006) :

a. Jaringan syaraf dengan lapisan tunggal (*single layer net*).

   Jaringan dengan lapisan tunggal hanya memiliki satu lapisan dengan bobot bobot terhubung. Jaringan ini hanya menerima *input* kemudian secara langsung akan mengolahnya menjadi *output* tanpa harus melalui lapisan tersembunyi, seperti yang terlihat pada Gambar 2.7.



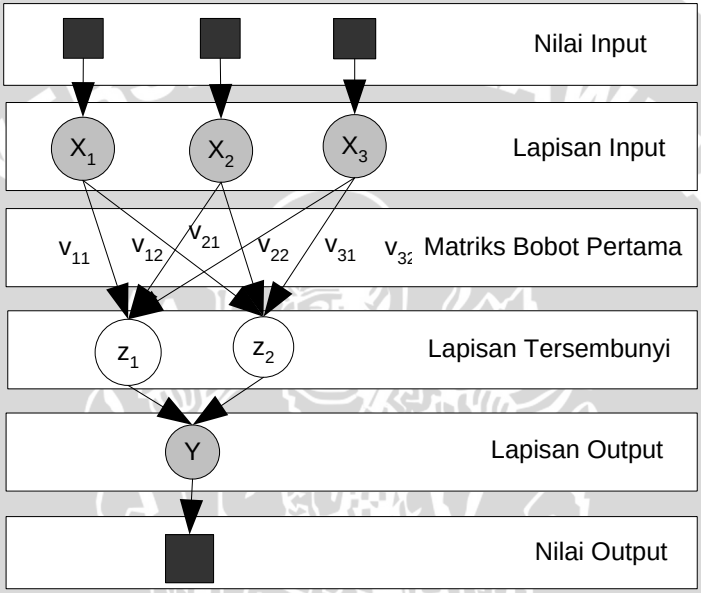

**Gambar 2.7** Jaringan saraf dengan lapisan tunggal

Pada Gambar 2.7 tersebut, lapisan *input* memiliki 3 neuron, yaitu X1, X2 dan X3. Sedangkan pada lapisan *output* memiliki 2 neuron yaitu Y1 dan Y2. Neuron-neuron pada kedua lapisan saling berhubungan. Seberapa besar hubungan antara 2 neuron ditentukan oleh bobot yang bersesuaian. Semua unit *input* akan dihubungkan dengan setiap unit *output*.

b. Jaringan syaraf dengan banyak lapisan (*multilayer net).*

Jaringan dengan banyak lapisan memiliki 1 atau lebih lapisan yang terletak diantara lapisan *input* dan lapisan *output* (memiliki 1 atau lebih lapisan tersembunyi). Umumnya, ada lapisan bobot-bobot yang terletak antara 2 lapisan yang bersebelahan. Setiap nilai yang diinputkan akan dikalikan dengan bobot yang terhubung ke tiap neuron pada lapisan tersembunyi, lalu dijumlah. Hasil penjumlahannya diinputkan pada fungsi aktivasi yang berlaku pada neuron lapisan tersembunyi tersebut untuk mendapatkan hasilnya. Kemudian, nilai hasil dari tiap neuron lapisan tersembunyi dikalikan dengan bobot yang terhubung ke masing-masing

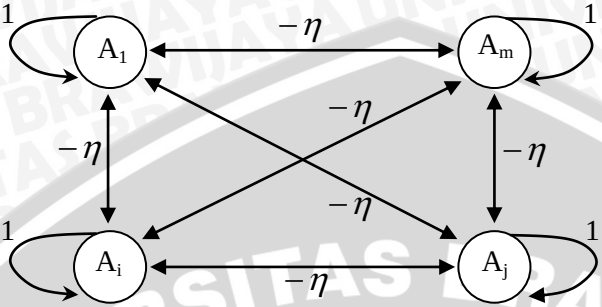
neuron pada sisi *output*. Hasil penjumlahannya dimasukkan pada fungsi aktivasi yang berlaku untuk mendapatkan nilai keluarannya. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit daripada lapisan dengan lapisan tunggal, tentu saja dengan pembelajaran yang lebih rumit.




**Gambar 2.8** Jaringan syaraf dengan banyak lapisan

c. Jaringan syaraf dengan lapisan kompetitif (*competitive layer net*).

Jaringan syaraf ini merupakan jenis jaringan saraf yang memiliki bobot yang telah ditetapkan dan tidak memiliki proses pelatihan, dan digunakan untuk mengetahui neuron pemenang dari sejumlah neuron yang ada. Nilai bobot untuk diri sendiri tiap neuron adalah 1, sedangkan untuk neuron lain adalah bobot random negatif. Gambar 2.9 menunjukkan salah satu contoh arsitektur jaringan dengan lapisan kompetitif yang memiliki bobot - $\eta$.




**Gambar 2.9** Jaringan syaraf dengan lapisan kompetitif

### 2.3.4 Proses Pembelajaran

Umumnya, jika menggunakan jaringan syaraf tiruan, hubungan antara *input* dan *output* harus diketahui secara pasti dan jika hubungan tersebut telah diketahui maka dapat dibuat suatu model. Hal lain yang penting adalah proses belajar hubungan *input/output* dilakukan dengan pembelajaran. Ada dua tipe pembelajaran yang dikenal yaitu pembelajaran terawasi dan pembelajaran tak terawasi.

Metode pembelajaran terawasi, digunakan jika *output* yang diharapkan telah diketahui sebelumnya. Biasanya pembelajaran dilakukan dengan menggunakan data yang telah ada. Sedangkan pada metode pembelajaran yang tidak terawasi, tidak memerlukan target *output*. Pada metode ini tidak dapat ditentukan hasil seperti apa yang diharapkan selama proses pembelajaran. Selama proses pembelajaran, nilai bobot disusun dalam suatu *range* tertentu tergantung pada nilai *input* yang diberikan. Tujuan pembelajaran ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran seperti ini biasanya sangat cocok untuk pengelompokkan (klasifikasi) pola (Kusumadewi, 2003).

### 2.3.5 Fungsi Aktivasi

Fungsi aktivasi digunakan untuk menentukan nilai keluaran pada neuron. Jika net=$\sum x_i y_i$, maka fungsi aktivasinya f(net)=f($\sum x_i y_i$).

Beberapa fungsi aktivasi yang sering digunakan adalah (Purnomo, 2006) :

a. Fungsi sigmoid biner (*binary sigmoid*)

Fungsi sigmoid biner memiliki nilai pada range 0 sampai 1. Oleh karena itu, fungsi ini sering digunakan pada jaringan syaraf yang membutuhkan nilai output yang terletak pada interval 0 sampai 1. Namun fungsi ini bisa juga digunakan oleh jaringan syaraf yang nilai outputnya 0 atau 1. Fungsi sigmoid biner dirumuskan pada persamaan

$$y = f(x) = \frac{1}{1+e^{-\sigma x}} \quad (2.7)$$

Dengan :

$$f'(x) = \sigma f(x)[1 - f(x)] \quad (2.8)$$

b. Fungsi sigmoid bipolar (*bipolar sigmoid*)

Fungsi sigmoid bipolar hampir sama dengan fungsi sigmoid biner, hanya saja output dari fungsi ini memiliki *range* antara -1 sampai 1.

Fungsi sigmoid bipolar dirumuskan pada persamaan

$$y = f(x) = \frac{1-e^{-x}}{1+e^{-x}} \quad (2.9)$$

Dengan :

$$f'(x) = \frac{\sigma}{2}[1 + f(x)][1 - f(x)] \quad (2.10)$$

Fungsi ini sangat dekat dengan fungsi *hyperbolic tangen*. Keduanya memiliki *range* antara -1 sampai 1. Fungsi *hyperbolic tangen* dirumuskan pada persamaan

$$y = f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad (2.11)$$

atau

$$y = f(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \quad (2.12)$$

Dengan

$$f'(x) = [1 + f(x)][1 - f(x)] \quad (2.13)$$

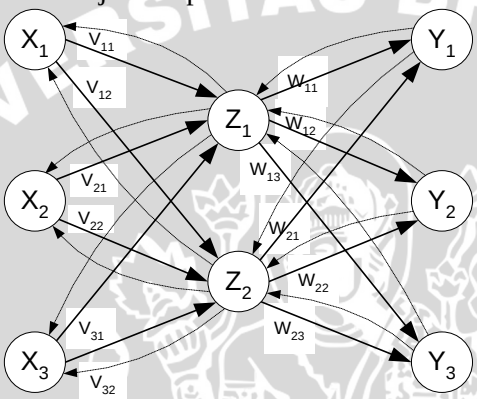
### 2.3.6 Algoritma Backpropagation

#### 2.3.6.1 Arsitektur backpropagation

Kelemahan jaringan syaraf tiruan yang terdiri dari lapisan tunggal membuat perkembangannya menjadi terhenti pada sekitar tahun 1970 an. Penemuan backpropagation yang terdiri dari beberapa lapisan membuka cakarawala, terlebih setelah berhasil ditemukannya berbagai aplikasi yang dapat diselesaikan dengan backpropagation, membuat jaringan syaraf tiruan semakin diminati orang.

Backpropagation merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan oleh perceptron dengan banyak

lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang ada pada bagian tersembunyi. Algoritma backpropagation yang menggunakan error output untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan nilai error ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, neuron-neuron diaktifkan dengan menggunakan fungsi aktivasi sigmoid atau bipolar sigmoid. Arsitektur jaringan syaraf tiruan Backpropagation ditunjukkan pada Gambar 2.10.




**Gambar 2.10** Arsitektur jaringan Backpropagation

Pada dasarnya, algoritma backpropagation terbagi menjadi tiga tahap, yaitu langkah maju (*feedforward*), propagasi balik (*backpropagation*), dan perubahan nilai bobot. Pada saat *feed forward*, perhitungan nilai bobot neuron hanya didasarkan pada vektor masukan. Sedangkan pada backpropagation, nilai bobot diperhalus dengan memperhitungkan nilai target atau keluaran (Kusumadewi, 2003).

Nilai *mean square error* (MSE) pada satu perulangan pelatihan dihitung sebagai jumlah kuadrat dari error (error = nilai keluaran ($t_k$) – nilai masukan ($y_k$)), kemudian dibagi dengan jumlah data. Secara matematis dapat dituliskan pada persamaan berikut (Purnomo, 2006) :

$$MSE = \frac{1}{N} \sum_{k=1}^{N} (t_k - y_k)^2 \qquad (2.14)$$

### 2.3.6.2 Algoritma pelatihan backpropagation

Struktur algoritma pelatihan *Backpropagation* adalah:

1. Inisialisasi bobot-bobot
   Tentukan angka pembelajaran ($\alpha$).
   Tentukan pula nilai toleransi *error* yang diinginkan dan set maksimal *epoch* jika ingin membatasi jumlah *epoch* yang digunakan.
2. Selama kondisi berhenti tidak terpenuhi, lakukan langkah ke-3 sampai langkah ke-10.
3. Untuk setiap pasangan pola pelatihan, lakukan langkah ke-4 sampai langkah ke-9.

   *Tahap maju (Feed Forward)*
4. Tiap-tiap unit masukan ($X_i$, i = 1, 2, 3, ..., n) menerima sinyal masukan dan meneruskan sinyal tersebut ke tiap-tiap unit pada lapisan tersembunyi.
5. Tiap-tiap unit di lapisan tersembunyi (*zj*, *j* = 1, 2, 3, ..., *p*) dikalikan dengan penimbang dan dijumlahkan serta ditambah dengan biasnya:
   $$Z\_in_j = V_{0j} + \sum_{i=1}^{n} X_i V_{ij} \qquad (2.15)$$
   Kemudian dihitung sesuai dengan fungsi aktivasi yang digunakan :
   $$Z_j = f(Z\_in_j) \qquad (2.16)$$
   dan mengirimkan sinyal tersebut ke semua unit pada lapisan di lapis keluaran (unit *output*).
6. Tiap-tiap unit di lapisan keluaran (*output*) (*Yk*, *k* = 1, 2, 3, ..., *m*) dikalikan dengan penimbang dan dijumlahkan serta ditambah dengan biasnya:
   $$Y\_in_k = W_{0k} + \sum_{j=1}^{p} Z_j W_{jk} \qquad (2.17)$$
   Kemudian dihitung kembali sesuai dengan fungsi aktivasi
   $$Y_k = f(Y\_in_k) \qquad (2.18)$$

   *Tahap mundur (Backpropagation)*
7. Tiap-tiap unit keluaran ($Y_k$, k=1,...,m) menerima pola target sesuai dengan pola masukan saat pelatihan(*training*) dan dihitung galatnya ($\delta k$), yaitu:
   $$\delta_k = (t_k - y_k)\, f'(y\_in_k) \qquad (2.19)$$
   Kemudian hitung nilai koreksi bobot yang nantinya digunakan untuk memperbaiki nilai bobot antara lapisan tersembunyi dan lapisan keluaran (*wjk*), yaitu:

$$\Delta w_{jk} = \alpha \delta_k z_j \quad (2.20)$$

Hitung juga koreksi bias yang digunakan untuk memperbaiki nilai bias antara lapisan tersembunyi dan lapisan keluaran (*Wk*0), yaitu:

$$\Delta w_{k0} = \alpha \delta_k \quad (2.21)$$

8. Masing-masing penimbang yang menghubungkan unit-unit lapis keluaran dengan unit-unit pada lapis tersembunyi (Z*j*, *j* = 1, 2, 3, ..., *p*) dikalikan delta (*δk*) dan dijumlahkan sebagai masukan ke unit-unit lapis berikutnya.

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k W_{jk} \quad (2.22)$$

Selanjutnya dikalikan dengan turunan dari fungsi aktivasi untuk menghitung galatnya

$$\delta_k = \delta\_in_j \, f'(y\_in_j) \quad (2.23)$$

Kemudian hitung koreksi bobot untuk memperbaiki nilai bobot antara lapisan *input* dan lapisan tersembunyi (*vji*), yaitu:

$$\Delta v_{ji} = \alpha \delta_j x_i \quad (2.24)$$

Kemudian hitung koreksi bias untuk memperbaiki nilai bobot antara lapisan *input* dan lapisan tersembunyi (*vj*0), yaitu:

$$\Delta v_{j0} = \alpha \delta_j \quad (2.25)$$

*Tahap pengoreksian bobot*

9. Tiap-tiap unit keluaran (Yk, *k* = 1, 2, 3, ..., *m*) diperbaiki bobot dan biasnya ($W_{kj}$), yaitu:

$$w_{kj}(baru) = w_{kj}(lama) + \Delta w_{kj} \quad (2.26)$$

tiap-tiap unit tersembunyi ($Z_j$, j=1,...,p) diperbaiki bias dan bobotnya (j=0,...,n), yaitu:

$$v_{ji}(baru) = v_{ji}(lama) + \Delta v_{ji} \quad (2.27)$$

10. Tes kondisi berhenti

Keterangan

| Notasi | Keterangan |
|---|---|
| Xp | Pola masukan ke-p, |
| xi | Unit ke i pada lapiss masukan |
| $t_k$ | Pola target keluaran dari pelatihan |
| Xi | Nilai pengaktif dari unit xi |

| | |
|---|---|
| Zj | Unit ke-j pada lapis tersembunyi |
| $Z\_in_j$ | Keluaran untuk unit Zj |
| $z_j$ | Nilai pengaktif di unit Zj |
| Yk | Unit ke-k pada lapis keluaran |
| $Y\_in_k$ | Keluaran untuk unit Yk |
| $y_k$ | Nilai pengaktif dari unit Yk |
| $W_{k0}$ | Nilai bobot pada bias untuk unit Yk |
| $W_{kj}$ | Nilai penimbang dari $Z_{ij}$ ke unit $Y_k$ |
| $\Delta W_{kj}$ | Selisih antara $W_{kj}(t)$ dengan $W_{kj}(t+1)$ |
| $V_{j0}$ | Nilai penimbang pada bias untuk unit $Z_j$ |
| $V_{ij}$ | Nilai penimbang dari unit Xi ke unit Zj |
| $\Delta V_{ij}$ | Selisih antara $V_{ij}(t)$ dengan $V_{ij}(t+1)$ |
| $\delta_k$ | Faktor pengendalian nilai penimbang pada lapis keluaran |
| $\delta_j$ | Faktor pengendalian nilai penimbang pada lapis tersembunyi |
| $\alpha$ | Konstanta laju pelatihan |

### 2.3.7 Optimalitas arsitektur algoritma BackPropagation

#### 2.3.7.1. Inisialisasi bobot

Bobot sebagai interkoneksi jaringan syaraf tiruan yang akan dilatih biasanya diinisialisasi dengan nilai nyata kecil secara acak (*random*) (Purnomo, 2006).

Banyak studi empiris membuktikan bahwa meneruskan pelatihan pada saat galat mencapai nilai yang kecil dan stabil akan menghasilkan nilai-nilai bobot yang tidak diinginkan. Hal ini berpengaruh pada peningkatan galat dan kualitas *mapping* menurun.

Hirose (1991) dalam Purnomo (2006), menyatakan bahwa dalam banyak penelitian menunjukkan bahwa konvergensi tidak akan dicapai bila bobot kurang bervariasi, juga jika acaknya terlalu kecil. Konvergensi hampir selalu tercapai untuk inisialisasi acak pada -0.5 sampai 0.5 atau -1 sampai 1.

#### 2.3.7.2. Penentuan jumlah hidden layer

Banyaknya hidden layer dan neuron hidden layer sangat penting dalam menentukan lamanya proses pelatihan. Apabila pengalaman percobaan sangat minim, Widrow (1987) dalam Patterson (1996) mengemukakan "The Rule of Thumb" sebagai acuan kisaran jumlah neuron hidden layer pada layer pertama dengan formula :

$$h = \frac{Ntraining}{10(o+i)} = \frac{Ntraining*\varepsilon}{(0+i)} \quad (2.28)$$

dimana
Ntraining : banyaknya data yang ditraining dalam training set
h: banyaknya neuron hidden layer
o: banyaknya neuron output layer
i: banyaknya neuron input layer
$\varepsilon$: toleransi kesalahan

Patterson (1996) mengemukakan bahwa satu hidden layer sudah cukup untuk memenuhi beberapa kasus aplikasi.

#### 2.3.7.3. Parameter laju pelatihan (*learning rate*)

Paremeter laju pelatihan (learning rate) sangat berpengaruh pada intensitas proses pelatihan. Begitu pula terhadap efektifitas dan kecepatan mencapai konvergensi dari pelatihan. Nilai opotimum dari laju pelatihan (η) tergantung dari masalah yang diselesaikan, prinsipnya dipilih sedemikian rupa sehingga tercapai konvergensi yang optimal dalam proses pelatihan. Nilai laju pelatihan (η) yang cukup kecil menjamin penurunan gradien terlaksana dengan baik, namun ini berakibat bertambahnya jumlah iterasi. Pada umumnya, besar nilai laju pelatihan dipilih mulai 0,001 sampai 1 selama proses pelatihan (Purnomo, 2006).

### 2.3.8 Algoritma Optical Backpropagation

Algoritma Backpropagation adalah algoritma pelatihan terbimbing yang paling banyak digunakan untuk jaringan syaraf tiruan lapis jamak (*multilayer feedforward*). Kendala besar yang dihadapi dalam algoritma ini adalah lambatnya proses pelatihan, pemilihan nilai yang tepat untuk parameternya, dan bagaimana menghindari *local minima*. *Local minima* adalah suatu kondisi dimana pelatihan terhenti pada suatu nilai *error*/galat minimum, namun bukan nilai *error*/galat yang terendah. Untuk menghindari *local minima*, salah satu langkah yang digunakan adalah dengan memperkecil nilai laju pembelajaran (*learning rate*). Akan tetapi, langkah ini akan memperlambat proses pelatihan. Selain itu, dalam beberapa permasalahan, algoritma backpropagation standar tersebut membutuhkan waktu yang cukup lama untuk mengadaptasi bobot

antar unit dalam jaringan untuk meminimalkan *mean square error* (MSE) antara keluaran (*output*) yang diinginkan dengan keluaran (*output*) jaringan sebenarnya (Salameh, 2005).

Optical Backpropagation adalah salah satu algoritma dalam jaringan syaraf tiruan yang didesain untuk menyelesaikan beberapa kendala yang berkaitan dengan backpropagation standar yang menggunakan fungsi nonlinier, yang diterapkan pada unit keluaran. Salah satu aspek penting dari algoritma ini adalah kemampuannya untuk menghindari *local minima* dalam proses pelatihannya, serta memiliki kecepatan yang cukup tinggi untuk mencapai konvergensi selama proses pelatihan (Otair, 2005).

Pada Optical Backpropagation(OBP), kecepatan konvergensi dari proses pembelajaran dapat ditingkatkan secara signifikan dengan menyesuaikan nilai dari *Mean Square Error*(MSE), yang akan ditransmisikan mundur dari lapisan keluaran untuk setiap unit di lapisan tersembunyi(*hidden layer*).

Pada algoritma Backpropagation, nilai galat pada tiap-tiap unit keluaran didefinisikan dengan :

$$\delta_{pk} = (Y_{pk} - O_{pk}) \qquad (2.29)$$

Dimana p adalah vektor pelatihan, dan k adalah unit keluaran. Dalam hal ini, $Y_{pk}$ adalah nilai keluaran yang diinginkan, dan $O_{pk}$ adalah nilai keluaran sebenarnya dari unit ke-k, sementara $\delta_{pk}$ kemudian akan mempropagasi mundur (*backward*) untuk memperbarui bobot dari lapisan keluaran (*output layer*) dan lapisan tersembunyi (*hidden layer*).

Pada Optical Backpropagation, nilai galat pada tiap-tiap unit keluaran adalah :

$$New\delta_{pk} = (1 + e^{(Y_{pk} - O_{pk})^2}), \text{ jika } (Y_{pk} - O_{pk}) \geq 0 \qquad (2.30)$$

$$New\delta_{pk} = -(1 + e^{(Y_{pk} - O_{pk})^2}), \text{ jika } (Y_{pk} - O_{pk}) < 0. \qquad (2.31)$$

Optical backpropagation menggunakan dua bentuk galat karena fungsi eksponen selalu menghasilkan nilai 0 atau positif. Nilai dari $New\delta_{pk}$ ini akan meminimalkan galat dari tiap-tiap unit keluaran dengan lebih cepat bila dibandingkan dengan $\delta_{pk}$, dan bobot pada unit tertentu akan menjadi sangat besar dari nilai awalnya (Otair,2005).

# BAB III
# METODOLOGI DAN PERANCANGAN

## 3.1 Deskripsi Umum Sistem

Sistem pengenal tulisan merupakan sebuah sistem yang dapat melakukan pengenalan terhadap citra yang berisi tulisan tangan manusia. Pada dasarnya, sistem yang akan dibuat terdiri dari dua tahap yaitu, tahap pelatihan (*training*) dan tahap pengenalan.

Tahap pelatihan atau pembelajaran merupakan tahapan pertama sebelum dapat melakukan pengenalan, dimana dalam tahapan ini sistem akan melakukan pembelajaran terhadap huruf yang dimasukkan ke dalam sistem. Huruf yang dimasukkan ke dalam sistem berupa huruf tunggal hasil tulisan tangan manusia.

Pada tahapan pengenalan, sistem akan melakukan pengenalan terhadap citra yang dimasukkan oleh *user*. Dalam tahapan ini data tulisan yang dimasukkan oleh *user* akan dicocokkan dengan data pengetahuan hasil dari tahapan sebelumnya yaitu proses pembelajaran.

Dalam setiap tahapan pelatihan dan pengenalan, terdapat satu proses yang sangat penting terhadap hasil pengenalan maupun hasil pembelajaran. Proses yang dimaksud adalah proses pengolahan citra (*image processing*). Pemrosesan citra dikatakan sangat penting karena sebuah citra harus diproses dengan baik terlebih dahulu sebelum dilakukan perhitungan-perhitungan yang ada pada proses pembelajaran maupun pada proses pengenalan.

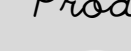
## 3.2 Data Uji

Pada penelitian ini data yang digunakan adalah citra dengan format *bitmap* yang berisi tulisan tangan. Data tulisan tangan diperoleh dari pengambilan tulisan tangan 10 orang. Setiap orang diminta untuk menuliskan kata-kata yang telah ditentukan dalam lembaran kertas yang kemudian dijadikan *file* citra dengan cara dipindai dengan *scanner*.

Data yang terkumpul akan digunakan untuk melakukan proses pelatihan dan proses pengujian sistem. Khusus untuk pelatihan, data yang digunakan merupakan huruf tunggal. Misalkan akan melakukan pelatihan terhadap huruf ”A”, maka data masukan berupa citra berisi huruf ”A” tunggal yang tidak tergabung dalam sebuah suku kata, kata, maupun kalimat. Sedangkan untuk pengujian, data berupa citra yang berisi huruf tunggal dan citra yang berisi

kumpulan beberapa huruf yang membentuk kata. Data pelatihan dan pengujian digambarkan pada Gambar 3.1 dan Gambar 3.2.

**Gambar 3.1** Contoh data untuk pelatihan



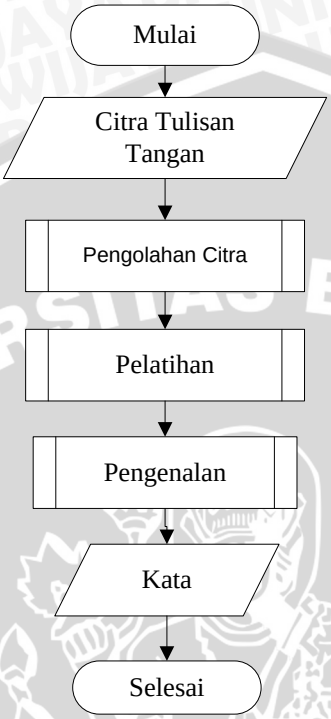
**Gambar 3.2** Contoh data untuk pengujian

### 3.3 Perancangan Sistem

Berdasarkan analisa yang dilakukan, berikut akan dibahas mengenai arsitektur dan proses yang terjadi pada perangkat lunak yang akan dibangun.

#### 3.3.1. Alur Sistem

Secara garis besar, alur sistem dapat digambarkan seperti yang terlihat dalam Gambar 3.3.

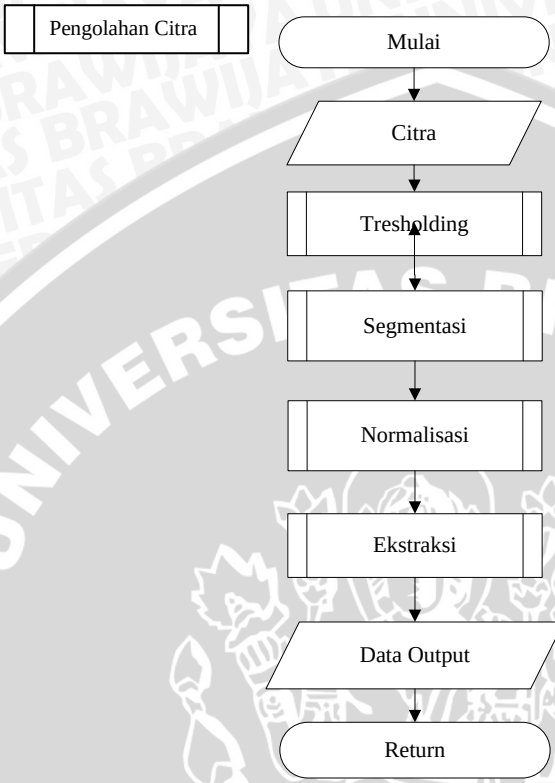



**Gambar 3.3** Diagram alir sistem

Dari gambar diagram alir sistem (Gambar 3.3) dapat dijelaskan bahwa sistem pertama kali akan menerima *input* dari *user*. *Input* disini adalah sebuah citra yang berisi tulisan tangan. Citra yang masuk ke dalam sistem kemudian akan diproses dalam tahap pengolahan citra, kemudian ekstraksi ciri yang kemudian akan dijadikan data *input* yang akan diproses pada jaringan syaraf tiruan. Keluaran dari pelatihan adalah bobot hasil pelatihan, dan keluaran dari hasil pengenalan adalah kata.

### 3.3.2. Pengolahan Citra

Seperti yang telah dijelaskan pada bahasan deskripsi umum sistem (sub bab 3.2), pengolahan citra merupakan satu proses yang sangat berpengaruh terhadap hasil pengenalan maupun pembelajaran huruf. Secara garis besar, proses pengolahan citra dapat digambarkan seperti pada Gambar 3.4.




**Gambar 3.4** Diagram alir pengolahan citra

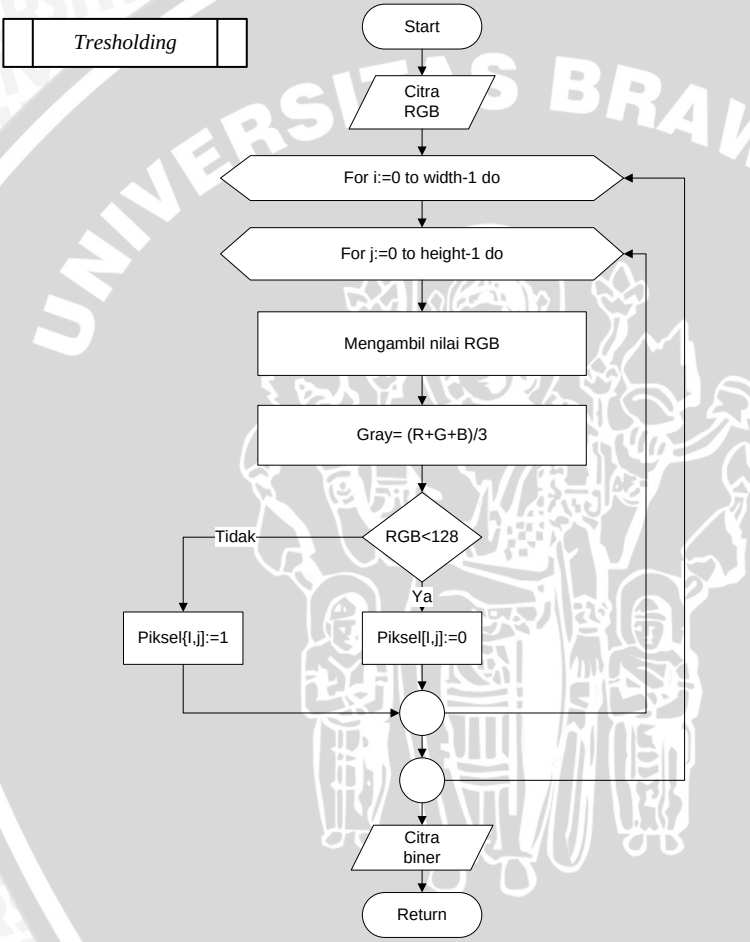
### 3.3.2.1 *Thresholding*

Pada tahap ini, akan dilakukan proses *thresholding* terhadap citra input, yaitu proses perbandingan antara nilai *grayscale* setiap *pixel* dengan nilai *threshold*. Jika nilai *grayscale* lebih besar atau sama dengan nilai *threshold*, *pixel* tersebut diberi nilai 1 (warna putih), sedangkan jika nilai *grayscale* kurang dari nilai *threshold*, *pixel* tersebut diberi nilai 0 (warna hitam). Berikut merupakan langkah-langkah binerisasi :

1. Masukkan citra berwarna atau *grayscale*.
2. Mangambil nilai RGB (*Red*, *Green*, *Blue*) pada setiap piksel yang membentuk citra.
3. Mencari nilai *grayscale* dengan menerapkan persamaan 2.2.
4. Memeriksa apakah nilai *grayscale* lebih dari nilai *treshold* atau sebaliknya.

5. Jika nilai *grayscale* lebih dari nilai treshold maka piksel tersebut diubah menjadi citra biner bernilai 1 (putih), begitu juga sebaliknya, jika nilai *grayscale* kurang dai 127 maka piksel tersebut diubah menjadi citra biner bernilai 0 (hitam).

Proses *tresholding* citra RGB ditunjukkan oleh Gambar 3.5




**Gambar 3.5** Diagram alir proses *tresholding*

### 3.3.2.2 Segmentasi

Untuk membagi citra ke dalam karakter pada setiap katanya dilakukan proses segmentasi. Proses segmentasi karakter dilakukan dengan melakukan penelusuran citra. Proses ini akan mencari nilai koordinat minimum dan koordinat maksimum pada setiap karakter,

sehingga akan diperoleh nilai lebar pada setiap karakter yang ditemukan, dimana *local minima* dianggap sebagai batas pemisah antar karakter.

Untuk melakukan segmentasi per karakter pada baris teks menggunakan *modified vertical histogram*. Gambar 3.6 menunjukkan ilustrasi proses *character* segmentation.

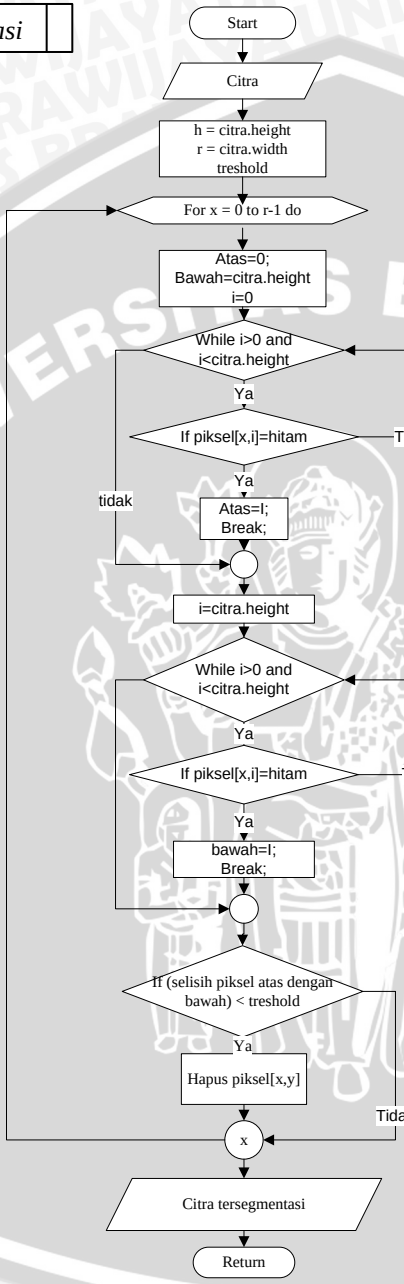

**Gambar 3.6** *Ilustrasi proses character segmentation*

Langkah-langkah segmentasi karakter yaitu :

1. Menginisialisasi nilai *treshold*, h menunjukkan ukuran tinggi dari citra dan r menunjukkan ukuran lebar dari citra.
2. Mencari nilai posisi piksel hitam teratas dan terbawah pada tiap kolom x pada citra.
3. Apabila nilai MapX (jaral piksel berwarna hitam teratas dengan terbawah) pada kolom x bernilai kurang dari *treshold*, maka piksel pada kolom tersebut dihapus.
4. Proses dilakukan secara berulang hingga x < r-1.

Gambar 3.7 menunjukkan urutan proses dari segmentasi karakter

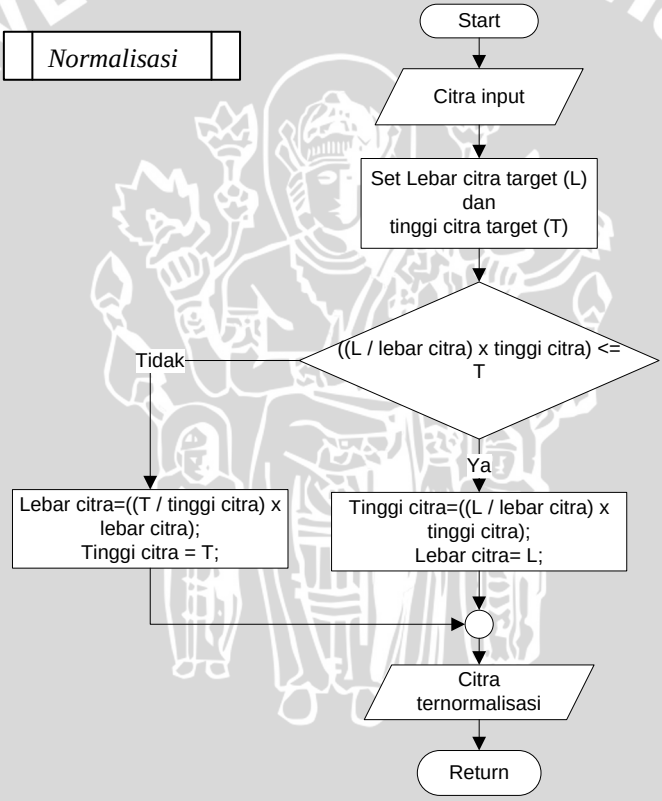
*Segmentasi*




**Gambar 3.7** Diagram alir proses segmentasi karakter

### 3.3.2.3 Normalisasi Ukuran Citra

Proses normalisasi merupakan proses untuk mengubah ukuran citra menjadi ukuran M x N untuk setiap karakternya. Hal ini dilakukan karena karakter-karakter yang dihasilkan dari proses segmentasi belum tentu mempunyai ukuran yang sama. Hasil perkalian M x N ini nantinya menjadi jumlah neuron yang ada pada *input layer*. Proses normalisasi dilakukan dengan membandingkan antara ukuran citra asal dengan ukuran citra target. Hal ini bertujuan agar walaupun ukuran citra berubah, proporsi antara lebar dan tinggi citra tetap terjaga. Proses normalisasi ukuran citra ditunjukkan pada Gambar 3.8




**Gambar 3.8** Diagram alir proses normalisasi citra

### 3.3.2.4 Ekstraksi

Ekstraksi merupakan proses untuk mengubah citra menjadi kode tertentu yang nantinya akan dimasukkan dalam proses jaringan

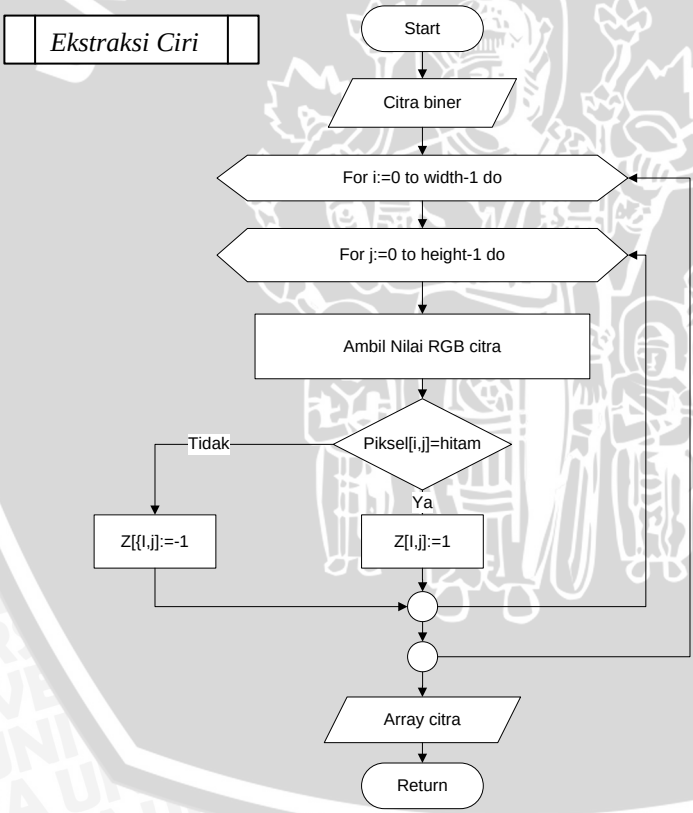
syaraf. Pengkodean ini berdasarkan nilai dari setiap *pixel* dalam citra. Jika *pixel* bernilai hitam maka akan dikodekan menjadi angka 1, apabila tidak, maka akan dikodekan dengan angka -1. Penggunaan angka 1 dan -1 ini berdasarkan fungsi aktivasi bipolar sigmoid yang nantinya akan digunakan pada proses jaringan syaraf.

(a)

1 1 1 1 1 1 1 1 1 1
-1 -1 -1 -1 -1 -1 -1 -1 1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 1 1 -1 -1
-1 -1 -1 -1 -1 -1 -1 1 1 -1 -1 -1
-1 -1 -1 -1 -1 -1 1 1 -1 -1 -1 -1
-1 -1 -1 1 1 -1 -1 -1 -1 -1 -1 -1
-1 -1 1 1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1
1 1 1 1 1 1 1 1 1 1 -1

(b)

**Gambar 3.9**
(a) Citra sampel, (b) Kode hasil ekstraksi




**Gambar 3.10** Diagram alir proses ekstraksi ciri

Keterangan
Z[i,j] = Array citra hasil ekstraksi ciri

### 3.3.3 Proses Jaringan Syaraf Tiruan

Proses jaringan syaraf tiruan dilakukan setelah proses pengolahan citra selesai dilakukan. Proses ini dapat dibagi menjadi proses membuat struktur jaringan, proses pelatihan, dan proses pengenalan.

#### 3.3.3.1. Struktur Jaringan Syaraf Tiruan

Jaringan yang akan dibangun meliputi tiga buah *layer*, yaitu *input layer*, satu *hidden layer*, dan *output layer* seperti yang tampak pada Tabel 3.1

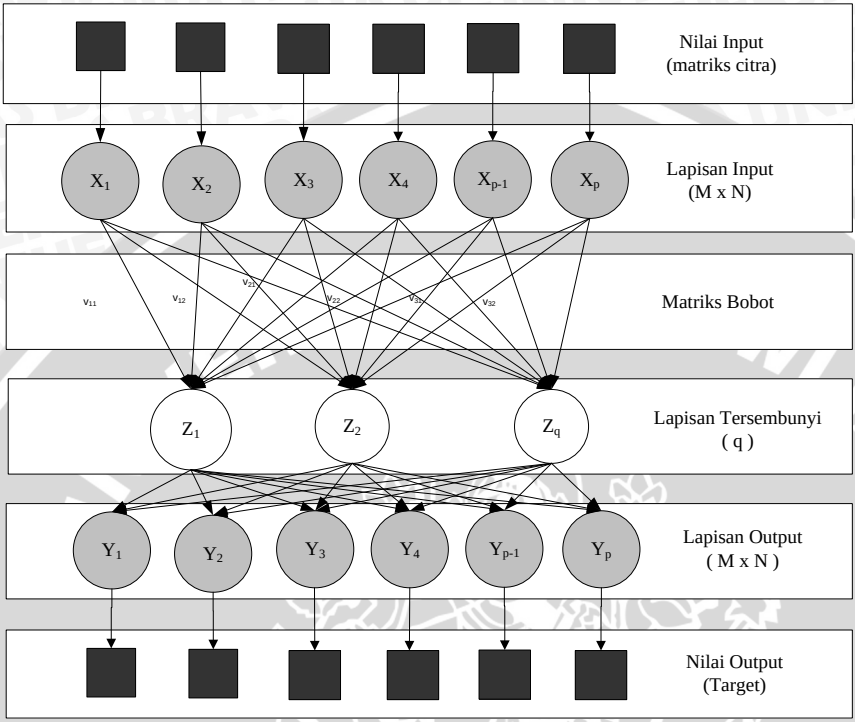
**Tabel 3.1** Distribusi neuron

| ***Layer*** | **Jumlah Neuron** |
|---|---|
| *Input* | M x N |
| *Hidden* | q |
| *Output* | M x N |

*Input layer* dan *output layer* terdiri dari M x N buah neuron yang merupakan hasil perkalian ukuran panjang dan lebar citra pada saat normalisasi. Misalkan nilai M x N adalah 20 x 20, maka terdapat 400 buah neuron pada *input layer* dan *output layer*. Jadi, pada saat pelatihan pertama dilakukan, nilai *input* dan *output layer* akan bernilai sama.

Sedangkan jumlah neuron pada *hidden layer* adalah q, dimana nilai q nantinya akan diujicobakan beberapa nilai. Hal ini dilakukan untuk mengetahui bagaimana pengaruh jumlah *hidden layer* terhadap hasil dari pengenalan tulisan tangan.

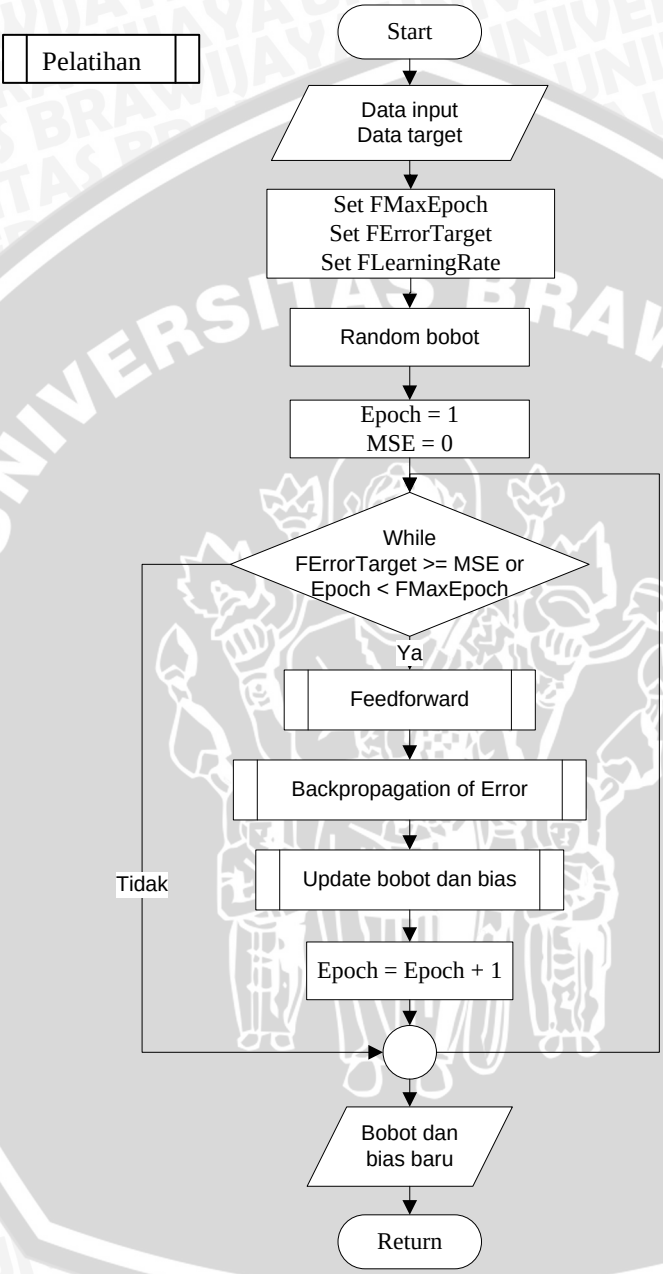
Pada Gambar 3.11 ditampilkan jaringan syaraf tiruan yang terbentuk dengan spesisifikasi *input layer*, 1 *hidden layer* dan *output layer*. Jumlah neuron pada *input* dan *output layer* adalah hasil perkalian M x N, ukuran yang digunakan pada saat normalisasi ukuran citra.




**Gambar 3.11** Jaringan syaraf tiruan pengenalan huruf

### 3.3.3.2. Proses Pelatihan

Pada proses pelatihan, terjadi beberapa proses yang digambarkan dalam diagram alir seperti yang tampak pada Gambar 3.12.




**Gambar 3.12** Diagram alir proses pelatihan

Urutan langkah proses pelatihan :

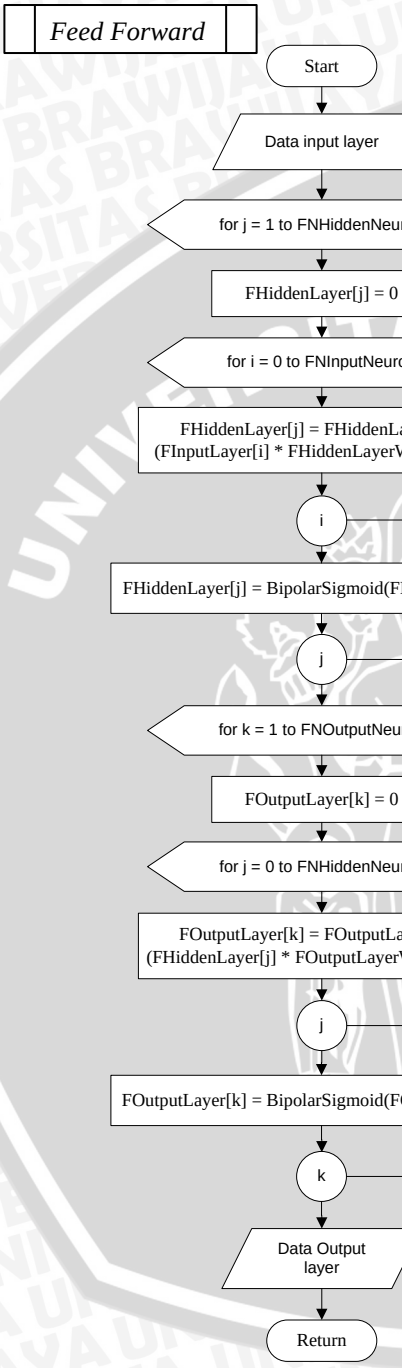
1. Mulai
2. Masukkan data *input* yang diperoleh dari proses pengolahan citra serta data target yang akan dilatihkan.
3. Tentukan nilai maksimum iterasi (*FMaxEpoch*), target kesalahan yang ingin dicapai (*FErrorTarget*), dan nilai laju pembelajaran (*FLearningRate*)
4. Proses pembobotan awal.
5. Atur nilai iterasi (*Epoch*) sama dengan 1 dan MSE sama dengan 0.
6. Selama *FerrorTarget* lebih besar atau sama dengan MSE atau *Epoch* kurang dari *FmaxEpoch* lakukan langkah 7, jika tidak, lakukan langkah 10.
7. Lakukan proses *feedforward* (persamaan 2.15 sampai 2.18). Proses ini akan dijelaskan lebih rinci dalam Diagram alir *feedforward* (Gambar 3.13).
8. Lakukan proses penghitungan informasi *error*. Proses ini akan dijelaskan lebih rinci dalam diagram alir *backpropagation of error* (Gambar 3.14).
9. Lakukan perhitungan bobot baru. Proses ini akan dijelaskan lebih rinci dalam diagram alir *update* bobot dan bias (Gambar 3.15).
10. Proses berhenti

Gambar 3.13 merupakan diagram alir yang menjelaskan langkah-langkah yang dilakukan dalam proses perambatan maju (*feedforward*). Urutan langkah yang terjadi selama proses *feedforward* adalah sebagai berikut:

1. Mulai
2. Kalikan seluruh data *input layer* (*FInputLayer*) dengan bobot pada masing-masing koneksi yang menuju ke *hidden layer* (*FHiddenLayerWeights*). Kemudian jumlahkan seluruh vektor bobot yang menuju neuron *hidden layer* yang sama.
3. Lakukan aktivasi hasil penjumlahan (langkah 2) tersebut pada masing-masing neuron. Aktivasi yang digunakan adalah bipolar sigmoid (persamaan 2.9).
4. Kalikan seluruh data hasil aktivasi masing-masing neuron *hidden layer* pada *FhiddenLayer[j]* dengan bobot pada masing-masing koneksi bobot *FhiddenLayerWeights[k,j]* yang terhubung dengan neuron pada *hidden layer*. Kemudian

jumlahkan seluruh vektor bobot yang menuju *output layer* yang sama.

5. Lakukan aktivasi hasil penjumlahan (langkah 4) tersebut pada masing-masing neuron di *output layer*.
6. Selesai. Hasil pada *output layer* akan digunakan pada proses *backpropagation*.




**Gambar 3.13** Diagram alir proses *feedforward*

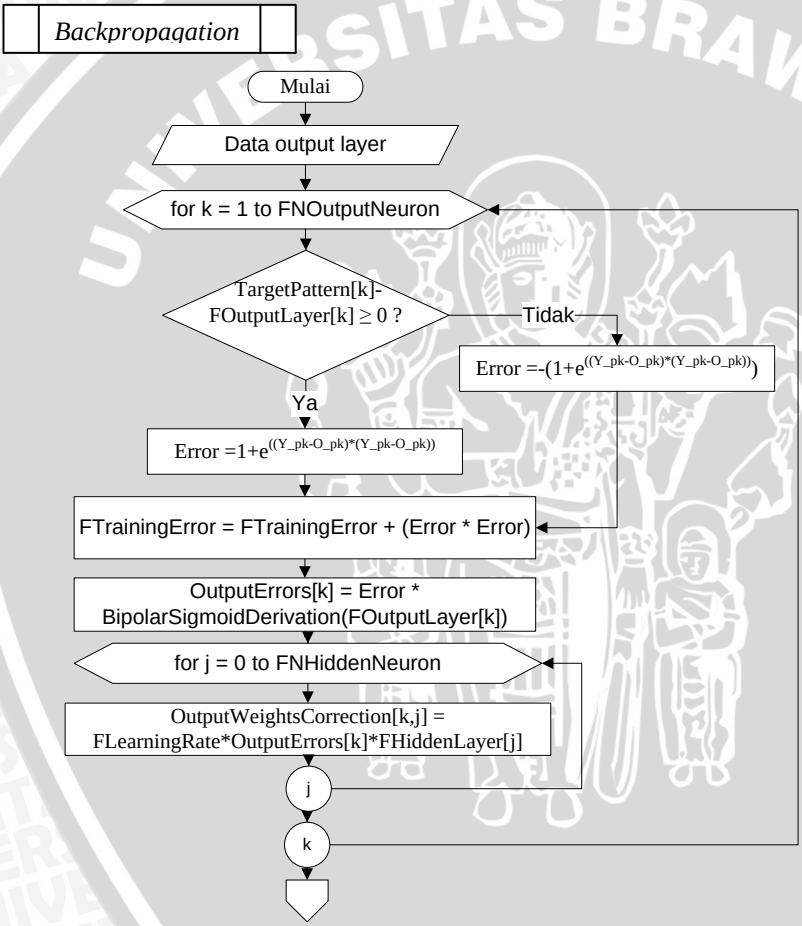
Setelah *feedforward* selesai dilakukan, proses yang terjadi selanjutnya adalah proses *backpropagation*. Proses *backpropagation* merupakan proses perhitungan informasi kesalahan (*error*) pada tiap neuron pada masing-masing *layer*. Perhitungan informasi kesalahan dimulai dari kesalahan yang berada pada *output layer* kemudian merambat mundur ke *hidden layer*. Informasi kesalahan berguna untuk menghitung faktor peubah bobot yang akan digunakan untuk perbaikan bobot lama.

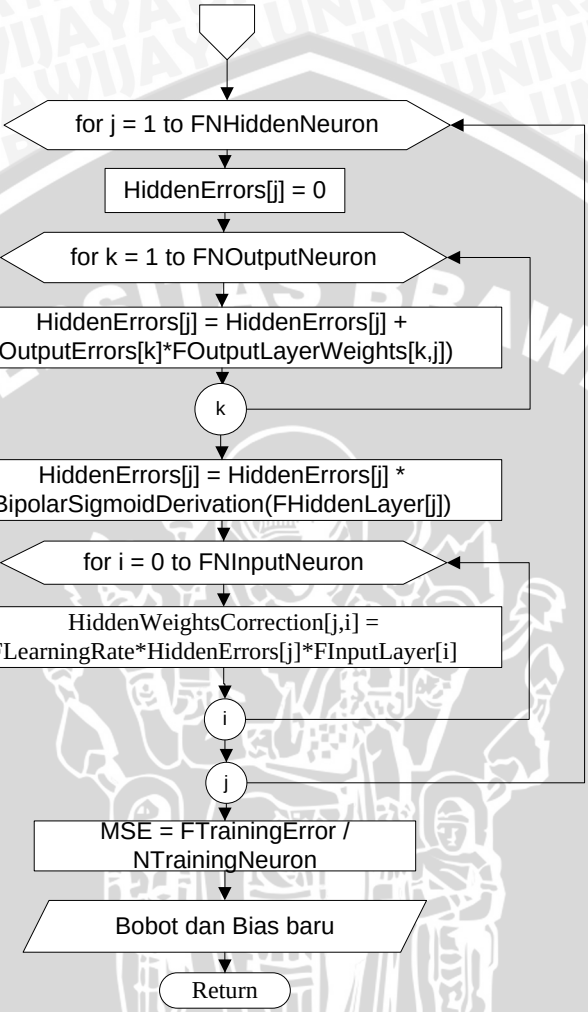
Adapun langkah-langkah yang terjadi selama proses *backpropagation* digambarkan pada diagram alir Gambar 3.14 dengan penjelasan sebagai berikut:

1. Mulai
2. Pada lapisan output. Hitung selisih antara target pengenalan (*TargetPattern[k]*) dengan *output* pengenalan (*FoutputLayer[k]*). Kemudian dengan menggunkan persamaan 2.30 dan 2.31, hitung nilai *Error*. Hitung nilai *FTrainingError* dengan cara menambahkan nilai *FTrainingError* sebelumnya dengan kuadrat *Error*. Kalikan nilai *Error* dengan *output* pengenalan (*FoutputLayer[k]*) yang telah diaktivasi dengan fungsi turunan aktivasi. Hasil perkalian ini merupakan faktor kesalahan pada *output layer* (*OutputErrors[k]*) yang akan digunakan untuk menghitung faktor kesalahan pada *hidden layer* dan untuk menghitung faktor peubah bobot pada vektor bobot yang menuju *output layer*.
3. Hitung nilai faktor peubah bobot baru pada tiap vektor yang menuju lapisan *output* (*OutputWeightsCorrection[k,j]*) dengan cara mengalikan *FLearningRate* dengan *OutputErrors[k]* dan *FHiddenLayer[j]*.
4. Pada lapisan *hidden*. Untuk menghitung faktor kesalahan masing-masing neuron *hidden layer* dilakukan: masing-masing faktor kesalahan di *output* (*OutputErrors[k]*) dikalikan dengan bobot lama yang terkoneksi dengan neuron *output layer* (*FoutputLayerWeights[k,j]*). Hasil perkalian pada seluruh koneksi yang terhubung dengan masing-masing neuron pada lapisan *hidden* akan dijumlahkan. Faktor kesalahan pada neuron lapisan *hidden* akan digunakan untuk menghitung peubah bobot pada koneksi dari lapisan *input* menuju lapisan *hidden*.
5. Hitung nilai faktor peubah bobot baru pada tiap vektor yang menuju lapisan *hidden* (*HiddenWeightsCorrection[j,i]*) dengan

cara mengalikan *FLearningRate* dengan *HiddenErrors[k]* dan *FInputLayer[j]*.

6. Hitung nilai MSE dengan cara membagi FTrainingError dengan 2
7. Selesai. Faktor peubah dan MSE akan digunakan dalam proses selanjutnya.





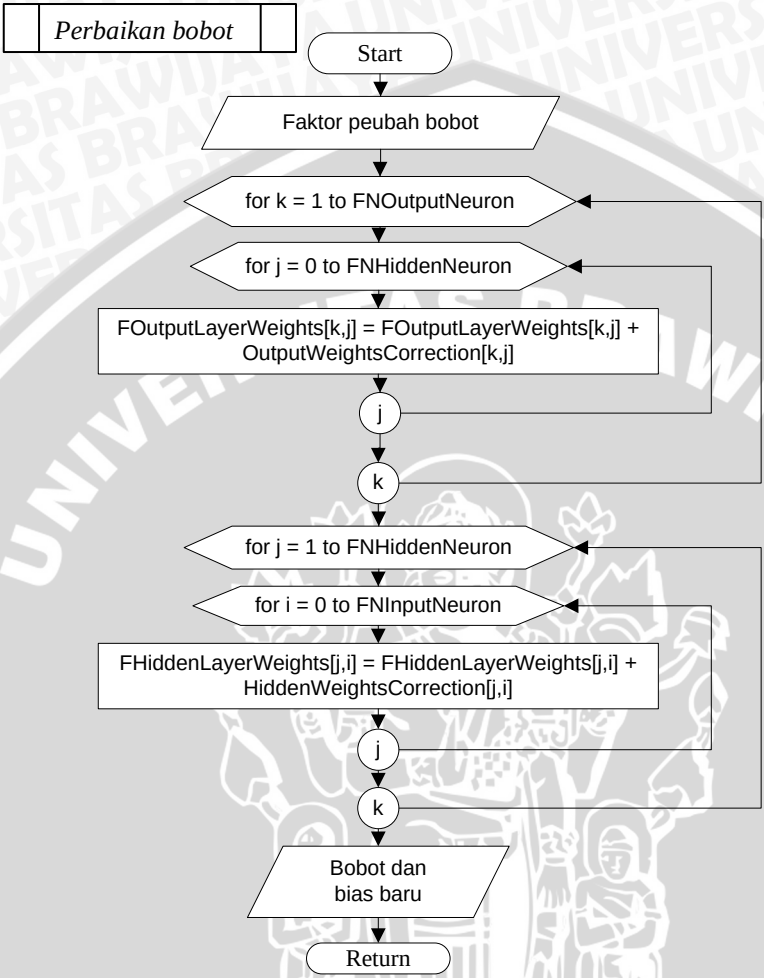

**Gambar 3.14** Diagram alir proses *backpropagation of error*

Hasil dari proses *backpropagation* adalah nilai faktor peubah bobot yang digunakan untuk melakukan perubahan bobot dan nilai MSE.

Proses perubahan bobot dilakukan untuk mendapatkan nilai bobot baru. Proses ini digambarkan dengan diagram alir pada Gambar 3.15.

Langkah-langkah yang dilakukan dalam proses perubahan bobot adalah :

1. Mulai
2. Perbaiki nilai bobot untuk setiap koneksi yang menuju ke *output layer* dengan cara menjumlahkan nilai bobot lama (*FOutputLayerWeights[k,j]*) dengan faktor peubah bobot (*OutputWeightsCorrection[k,j]*) yang telah dihitung pada proses *backpropagation*.
3. Perbaiki nilai bobot untuk setiap koneksi yang menuju ke *hidden layer* dengan cara menjumlahkan nilai bobot lama (*FHiddentLayerWeights[j,i]*) dengan faktor peubah bobot (*HiddenWeightsCorrection[j,i]*) yang telah dihitung pada proses *backpropagation*.
4. Selesai.




**Gambar 3.15** Diagram alir proses perubahan bobot

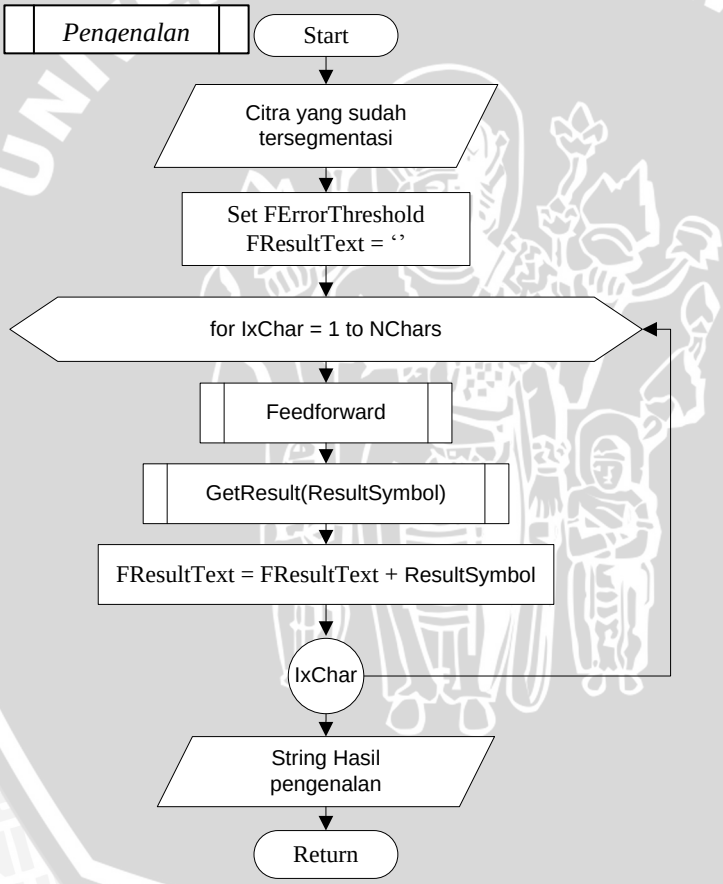
### 3.3.3.3. Proses Pengenalan

Setelah proses pelatihan selesai dilakukan, maka proses pengenalan dapat dilakukan. Proses ini digambarkan pada diagram alir Gambar 3.16. Masukan dalam proses pengenalan ini adalah huruf yang sudah disegmentasi dalam proses pengolahan citra. Langkah-langkah pengenalan secara menyeluruh diterangkan sebagai berikut :

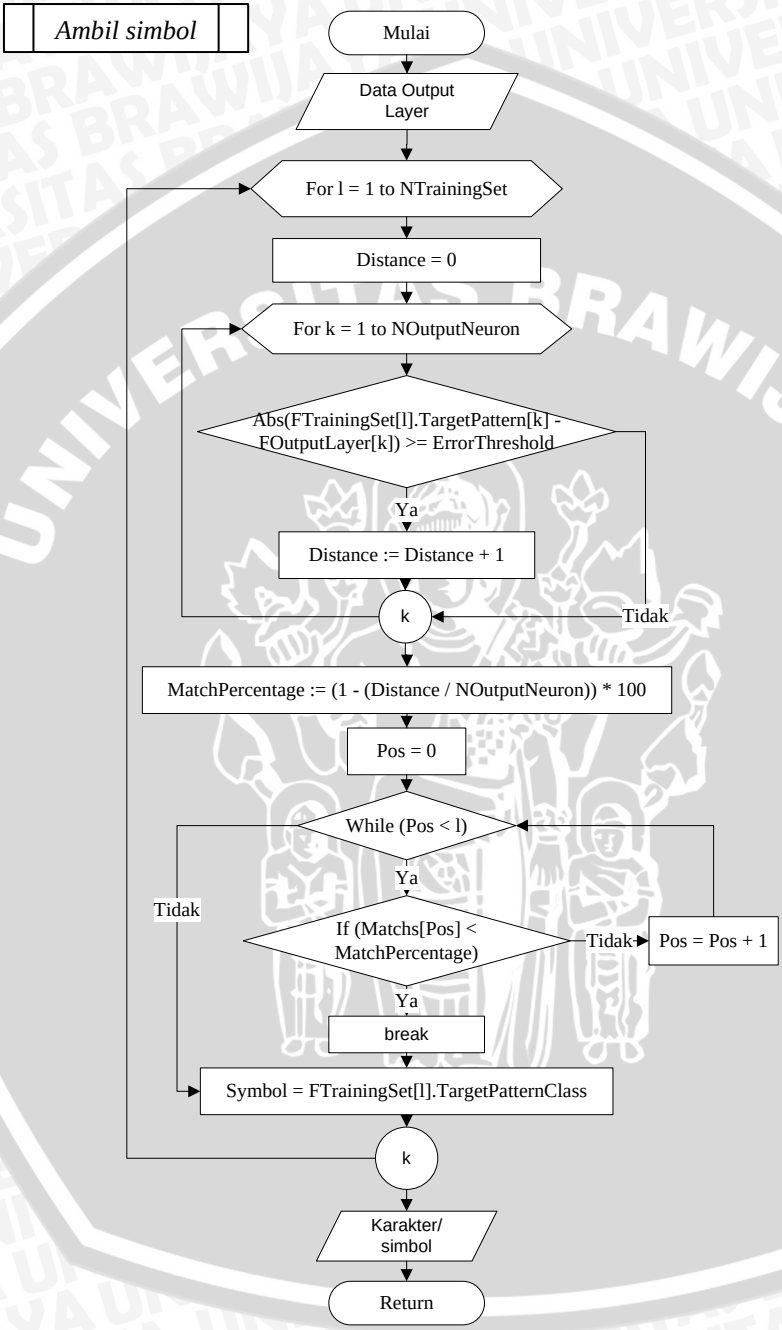
1. Mulai
2. Tetapkan nilai ambang kesalahan (*FErrorThreshold*), inisialisasikan hasil pengenalan (*FResultText*)

3. Untuk setiap karakter huruf dalam kata lakukan langkah 4.
4. Lakukan *feedforward* seperti pada proses pelatihan.
5. Dapatkan simbol hasil pengenalan. Proses mendapatkan simbol ini akan dijelaskan lebih rinci pada pada diagram alir proses ambil simbol pada Gambar 3.17
6. *FResultText* sama dengan *FResultText* ditambah dengan *ResultSymbol*.
7. Selesai.

Keluaran dari proses pengenalan ini adalah *FResultText* yang berisi string hasil pengenalan.




**Gambar 3.16** Diagram alir proses pengenalan




**Gambar 3.17** Diagram alir proses ambil simbol

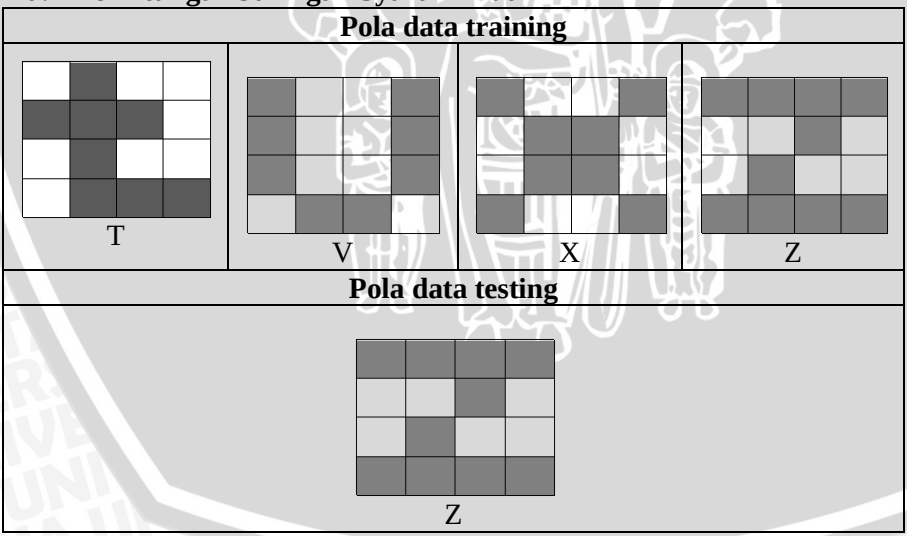
Langkah-langkah dalam mendapatkan simbol huruf yang dikenali digambarkan pada diagram alir Gambar 3.17 dengan penjelasan sebagai berikut:

1. Mulai
2. Inisialisasi nilai *Symbol*
3. Untuk setiap data dalam *TrainingSet* lakukan langkah 4
4. Inisialisasi nilai *Distance*
5. Jika nilai mutlak kesalahan pada tiap-tiap *output layer* lebih besar atau sama dengan *ErrorThreshold*, tambahkan nilai *Distance* dengan nilai satu.
6. Hitung nilai persentase kesamaan (*MatchPercentage*)
7. Ganti nilai *Symbol* dengan *TargetPatternClass*
8. Urutkan nilai kesamaan dari yang paling besar
9. Selesai. Data hasil pengenalan adalah yang mempunyai prosentase kesamaan yang paling besar.

## 3.4. Perhitungan manual

Pada contoh perhitungan ini tidak menggunakan perhitungan asli secara manual, namun mengunakan ukuran sederhana sehingga mudah dalam memahami maksud.

### a. Perhitungan Jaringan Syaraf Tiruan




**Gambar 3.18** Contoh data untuk *training* dan *testing*

1. Data input matriks
   Pada contoh perhitungan ini, digunakan citra input berukuran 4 x 4 pixel. Matriks citra ini akan diubah kedalam array 1 dimensi. Contoh huruf yang digunakan adalah huruf Z. Dari Gambar 3.17, piksel yang berwarna hitam akan dikodekan dengan angka 1, sementara piksel yang berwarna putih akan dikodekan dengan -1. Hasilnya seperti pada Tabel 3.2.

**Tabel 3.2** Data Input pelatihan

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| -1 | -1 | 1 | -1 |
| -1 | 1 | -1 | -1 |
| 1 | 1 | 1 | 1 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | -1 | -1 | 1 | -1 | -1 | 1 | -1 | -1 | 1 | 1 | 1 | 1 |

**Tabel 3.3** Data target pelatihan

| Target pelatihan |
|---|
| 1 |
| 1 |
| 1 |
| 1 |

Tentukan parameter awal untuk pelatihan, yang meliputi :
Neuron *input* = 16
Neuron *Hidden* =8
Neuron *Output* = 4
$\alpha = 0.01$
MSE = 0.001

**Inisialisasi bobot**

**Tabel 3.4** Bobot awal lapisan tersembunyi (Vij)

| i \ j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | -0.2 | -0.35 | 0.34 | -0.49 | -0.38 | -0.5 | 0.44 | 0.1 |
| 2 | -0.27 | 0.04 | 0.02 | -0.18 | -0.23 | -0.07 | -0.11 | 0.31 |
| 3 | 0.37 | -0.22 | -0.05 | -0.24 | 0.1 | -0.42 | -0.12 | 0 |
| 3 | -0.39 | -0.43 | 0.12 | -0.33 | 0.12 | 0.4 | -0.3 | -0.18 |
| 4 | 0.38 | -0.01 | -0.25 | 0.06 | -0.38 | 0.2 | -0.08 | -0.14 |
| 5 | 0.25 | 0.24 | -0.1 | -0.23 | 0.23 | -0.41 | -0.19 | -0.32 |
| 6 | 0.48 | -0.34 | 0.18 | 0.34 | -0.43 | -0.37 | -0.23 | -0.31 |
| 7 | -0.27 | 0.08 | 0.46 | -0.17 | -0.08 | 0.37 | -0.11 | -0.27 |
| 8 | 0.28 | -0.02 | -0.28 | -0.2 | -0.48 | 0.14 | -0.46 | 0.07 |
| 9 | -0.18 | 0.49 | -0.09 | -0.31 | 0.39 | -0.31 | 0 | -0.32 |
| 10 | -0.2 | 0.13 | -0.14 | -0.48 | -0.31 | 0.15 | 0.2 | -0.2 |
| 11 | 0.39 | 0.48 | 0.32 | -0.1 | -0.29 | -0.23 | -0.5 | -0.18 |
| 12 | -0.1 | -0.5 | -0.01 | 0.25 | 0.36 | 0.47 | -0.5 | -0.44 |
| 13 | -0.12 | -0.17 | -0.35 | -0.49 | 0.26 | 0.27 | -0.01 | -0.3 |
| 14 | -0.3 | -0.19 | 0.36 | 0.01 | 0.34 | 0.43 | -0.12 | -0.33 |
| 15 | -0.29 | 0.3 | -0.31 | -0.29 | 0.1 | 0.49 | 0.27 | -0.21 |
| 16 | 0.03 | 0.37 | 0.04 | -0.2 | 0.35 | 0.08 | 0.48 | -0.29 |

**Tabel 3.5** Bobot awal lapisan keluaran (Wjk)

| k \ j | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | -0.25 | -0.23 | 0.48 | -0.49 |
| 2 | -0.49 | -0.45 | 0.27 | 0.21 |
| 3 | -0.11 | 0.23 | -0.19 | 0.32 |
| 4 | -0.47 | 0.41 | 0.01 | 0.25 |
| 5 | -0.28 | -0.49 | -0.16 | 0.44 |
| 6 | 0.34 | -0.41 | -0.02 | 0.39 |
| 7 | -0.15 | 0.37 | 0.15 | 0.38 |
| 8 | -0.17 | -0.48 | -0.28 | 0.49 |

**Iterasi 1**

2. Tiap-tiap unit di lapisan tersembunyi ($zj$, $j$ = 1, 2, 3, ..., $p$) dikalikan dengan penimbang dan dijumlahkan serta ditambah dengan biasnya sesuai dengan Persamaan 2.18.

$Z\_in_1 = V_{01} + \sum_{i=1}^{n} X_i V_{i1}$

= -0.2 + ((1*-0.2)+(1*-0.27)+(1*0.37)+(1*-0.39)+
(1*0.38)+(-1*0.25)+(-1*0.48)+(1*-0.27) +(-1*0.28)+
(-1*-0.18)+(1*-0.2)+(-1*0.39)+(-1*-0.1)+(1*-
0.12)+(1*-0.3)+(1*-0.29)+(1*0.03) )
= -2.45

Hasil perhitungan ditunjukkan pada Tabel 3.6.

**Tabel 3.6** Tabel operasi pada layer tersembunyi

| j | $Z\_in_j$ |
|---|---|
| 1 | -2.45 |
| 2 | -0.41 |
| 3 | -0.10 |
| 4 | -2.24 |
| 5 | 0.26 |
| 6 | 2.54 |
| 7 | 1.87 |
| 8 | 0.20 |

Kemudian dihitung sesuai dengan fungsi aktivasi yang digunakan, yaitu fungsi sigmoid bipolar, sesuai dengan persamaan 2.19

$Z_j = f(Z\_in_j)$

$= f'(x) = [1 + f(x)][1 - f(x)]$

$=(1+\frac{e^{-2.45}-e^{2.45}}{e^{-2.45}+e^{2.45}})(1-\frac{e^{-2.45}-e^{2.45}}{e^{-2.45}+e^{2.45}})$

$= (1+\frac{-11.502}{11.0674})(1-\frac{-11.502}{11.0674}) = -0.84112$

**Tabel 3.7** Tabel hasil aktivasi tiap unit pada unit tersembunyi

| j | $Z_j$ |
|---|---|
| 1 | -0.84112 |
| 2 | -0.20218 |
| 3 | -0.04994 |
| 4 | -0.80757 |
| 5 | 0.12927 |
| 6 | 0.08538 |
| 7 | 0.73292 |
| 8 | 0.09968 |

3. Tiap-tiap unit di lapisan keluaran (*output*) (*Yk*, *k* = 1, 2, 3, ..., *m*) dikalikan dengan penimbang dan dijumlahkan serta ditambah dengan biasnya, sesuai dengan persamaan 2.20. hasil perhitungan ditunjukkan pada Tabel 3.8

$$Y\_in_k = W_{0k} + \sum_{j=1}^{p} Z_j W_{jk}$$

$Y\_in_1 = W_{01} + \sum_{j=1}^{p} Z_j W_{j1}$

= -0.25+((0.84112*-0.25)+(-0.20218*-0.49)+(-0.04994*-0.11)+(-0.80757*-0.47)+(0.12927*-0.28)+(0.08538* 0.34) + ( 0.73292*-0.15)+( 0.09968*-0.17) )

= -0.04861

**Tabel 3.8** Tabel operasi perhitungan pada lapisan *output*

| k | $Y\_in_k$ |
|---|---|
| 1 | -0.04861 |
| 2 | 0.19524 |
| 3 | 0.14125 |
| 4 | -0.28814 |

Kemudian dihitung kembali sesuai dengan fungsi aktivasi sesuai dengan persamaan 2.21. Hasil perhitungan ditunjukkan pada Tabel 3.9

$Y_k = f(Y\_in_k)$

$= f'(x) = [1 + f(x)][1 - f(x)]$

$= (1+\frac{e^{-0.04861}-e^{0.04861}}{e^{-0.04861}+e^{0.04861}})(1-\frac{e^{-0.04861}-e^{0.04861}}{e^{-0.04861}+e^{0.04861}})$

= -0.02430

**Tabel 3.9** Tabel hasil aktivasi pada lapisan *output*

| k | $Y_k$ |
|---|---|
| 1 | -0.02430 |
| 2 | 0.09731 |
| 3 | -0.07051 |
| 4 | -0.14308 |

4. Tiap-tiap unit keluaran ($Y_k$, k=1,...,m) menerima pola target sesuai dengan pola masukan saat pelatihan(*training*) dan dihitung galatnya ($\delta k$) sesuai dengan persamaan 2.22. Hasil perhitungan ditunjukkan pada Tabel 3.10.

$$\delta_k = -\left(1 + e^{(Y_{pk} - O_{pk})^2}\right) f'(y\_in_k)$$

$$= -\,(1 + e^{(-0.02430-1)^2})\,(1+\frac{e^{-0.04861}-e^{0.04861}}{e^{-0.04861}+e^{0.04861}})(1-\frac{e^{-0.04861}-e^{0.04861}}{e^{-0.04861}+e^{0.04861}})$$

$$= 2.0173$$

**Tabel 3.10** Tabel perhitungan galat tiap unit *output*

| j | $\partial_j$ |
|---|---|
| 1 | 2.0173 |
| 2 | 2.0173 |
| 3 | 2.0173 |
| 4 | 2.0173 |

Kemudian hitung nilai koreksi bobot yang nantinya digunakan untuk memperbaiki nilai bobot antara lapisan tersembunyi dan lapisan keluaran (*wjk*) sesuai dengan persamaan 2.23. Hasil perhitungan ditunjukkan pada Tabel 3.11.

$$\Delta w_{jk} = \alpha \delta_k z_j$$

$$= 0.01 * -\,(1 + e^{(-0.02430-1)^2}) * 1 = -0.162$$

**Tabel 3.11** Tabel hasil perhitungan koreksi bobot unit *ouput*

| k / j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | -0.162 | -0.003 | -0.0096 | -0.1555 | 0.249 | 0.164 | 0.141 | 0.192 |
| 2 | -0.136 | -0.0323 | -0.0080 | -0.1303 | 0.209 | 0.138 | 0.118 | 0.161 |
| 3 | -0.141 | -0.0339 | -0.0083 | -0.1355 | 0.022 | 0.143 | 0.123 | 0.167 |
| 4 | -0.193 | -0.0464 | -0.0011 | -0.1856 | 0.297 | 0.196 | 0.168 | 0.229 |

Hitung juga koreksi bias yang digunakan untuk memperbaiki nilai bias antara lapisan tersembunyi dan lapisan keluaran (*Wk*0) dengan menggunakan persamaan 2.24. Hasil perhitungan ditunjukkan pada Tabel 3.12

$\Delta w_{0k} = \alpha\ \delta_k$

$= 0.01 * 2.0173 = 0.02173$

**Tabel 3.12** Tabel hasil perhitungan koreksi bias (*Wk*0)

| k | $W_{k0}$ |
|---|---|
| 1 | 0.02173 |
| 2 | 0.16140 |
| 3 | 0.16779 |
| 4 | 0.22988 |

5. Masing-masing penimbang yang menghubungkan unit-unit lapis keluaran dengan unit-unit pada lapis tersembunyi (Z*j*, *j* = 1, 2, 3, ..., *p*) dikalikan delta ($\delta k$) dan dijumlahkan sebagai masukan ke unit-unit lapis berikutnya, sesuai dengan persamaan 2.25. Hasil perhitungan ditunjukkan pada Tabel 3.13

$\delta\_in_j = \sum_{k=1}^{m} \delta_k W_{jk}$

$\delta\_in_1 = \sum_{k=1}^{m} \delta_k W_{1k}$

= (2.0173*0.25)+ (2.0173*-0.23)+ (2.0173*0.48)+ (2.0173*-0.49)

= -0.10743

**Tabel 3.13** Tabel hasil perhitungan informasi galat ($\delta\_in_j$)

| j | $\delta\_in_j$ |
|---|---|
| 1 | -0.10743 |
| 2 | 0.27629 |
| 3 | 0.17344 |
| 4 | -0.10213 |
| 5 | 0.42098 |
| 6 | -0.42507 |
| 7 | -0.10312 |
| 8 | -0.11043 |

Selanjutnya dikalikan dengan turunan dari fungsi aktivasi, sesuai persamaan 2.26, untuk menghitung galatnya. Hasil perhitungan ditunjukkan pada Tabel 3.14

$\delta_1 = \delta\_in_1 \, f'(y\_in_1)$

= -0.10743* $(1+\frac{e^{-2.45}-e^{2.45}}{e^{-2.45}+e^{2.45}})(1-\frac{e^{-2.45}-e^{2.45}}{e^{-2.45}+e^{2.45}})$

= 1.9265

**Tabel 3.14** Tabel hasil perhitungan aktivasi ($\delta_k$)

| k | $\delta_k$ |
|---|---|
| 1 | 1.9265 |
| 2 | 1.6140 |
| 3 | 1.6779 |
| 4 | 2.2988 |

Kemudian hitung koreksi bobot untuk memperbaiki nilai bobot antara lapisan *input* dan lapisan tersembunyi (*vji*) dengan persamaan 2.27. Hasil perhitungan ditunjukkan pada Tabel 3.15.

$\Delta v_{ji} = \alpha \delta_j x_i$

$\Delta v_{11} = \alpha \delta_1 x_1$

= 0.01*1.9265*-1

= -0.01

**Tabel 3.15** Tabel hasil perhitungan koreksi bobot

| j \ i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | -0.01 | 0.0276 | 0.0173 | -0.01 | 0.042 | -0.004 | -0.01 | 0.0278 |
| 2 | -0.01 | 0.0276 | 0.0173 | -0.01 | 0.042 | -0.004 | -0.01 | 0.0278 |
| 3 | -0.01 | 0.0276 | 0.0173 | -0.01 | 0.042 | -0.004 | -0.01 | 0.0278 |
| 4 | -0.01 | 0.0276 | 0.0173 | -0.01 | 0.042 | -0.004 | -0.01 | 0.0278 |
| 5 | 0.0107 | -0.027 | -0.017 | 0.0102 | -0.042 | 0.0042 | 0.0103 | -0.027 |
| 6 | 0.0107 | -0.027 | -0.017 | 0.0102 | -0.042 | 0.0042 | 0.0103 | -0.027 |
| 7 | -0.01 | 0.0276 | 0.0173 | -0.01 | 0.042 | -0.004 | -0.01 | 0.0278 |
| 8 | 0.0107 | -0.027 | -0.017 | 0.0102 | -0.042 | 0.0042 | 0.0103 | -0.027 |

Kemudian hitung koreksi bias untuk memperbaiki nilai bobot antara lapisan *input* dan lapisan tersembunyi (*vj*0) dengan menggunakan persamaan 2.28. Hasil perhitungan ditunjukkan pada Tabel 3.16.

$\Delta v_{0j} = \alpha \delta_j$

$\Delta v_{01} = \alpha \delta_1$

= 0.01 *-0.10743 = -0.010743

**Tabel 3.16** Tabel hasil perhitungan perbaikan bobot ($V_{j0}$)

| j | $V_{j0}$ |
|---|---|
| 1 | -0.010743 |
| 2 | 0.027629 |
| 3 | 0.017344 |
| 4 | -0.010213 |
| 5 | 0.042098 |
| 6 | -0.004206 |
| 7 | -0.010312 |
| 8 | 0.027871 |

6. Tiap-tiap unit keluaran (Yk, $k$ = 1, 2, 3, ..., $m$) diperbaiki bobotnya ($W_{kj}$) dengan persamaan 2.29. Hasil perbaikan bobot ditunjukkan pada Tabel 3.17.

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

   $w_{11}(baru) = w_{11}(lama) + \Delta w_{11}$

   $= - 0.25 + - 0.162$

   $= -0.412$

**Tabel 3.17** Tabel hasil perbaikan bobot ($W_{kj}$)

| j \ k | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | -0.4120 | 0.16917 | 0.41821 | 0.22202 |
| 2 | -0.2668 | 0.032274 | -0.51775 | 0.38492 |
| 3 | 0.51924 | 0.38015 | 0.45573 | 0.11044 |
| 4 | -0.36664 | -0.41545 | 0.09086 | -0.10941 |
| 5 | -0.57212 | -0.25661 | -0.47451 | -0.17624 |
| 6 | 0.25156 | -0.49448 | -0.14048 | -0.50356 |
| 7 | 0.2833 | -0.46069 | -0.21358 | -0.28574 |
| 8 | 0.48151 | 0.32668 | -0.24741 | 0.41839 |

Tiap-tiap unit tersembunyi ($Z_j$, j=1,...,p) diperbaiki bias dan bobotnya ($V_{ji}$) (j=0,...,n) dengan persamaan 2.30. Hasil perhitungan ditunjukkan pada Tabel 3.18.

$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}$

$v_{11}(baru) = v_{11}(lama) + \Delta v_{11}$

$= -0.2 + -0.0129$

$= -0.2129$

**Tabel 3.18** Tabel hasil perbaikan bias ($V_{ij}$)

| j \ i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | -0.21290 | -0.35216 | 0.092241 | -0.16941 | -0.32377 | -0.19867 | 0.14122 | 0.15324 |
| 2 | -0.35373 | 0.29584 | 0.16624 | -0.19941 | -0.37577 | 0.0093338 | 0.38722 | 0.43524 |
| 3 | 0.54627 | -0.10016 | -0.10976 | -0.019406 | -0.54377 | -0.17067 | -0.080783 | 0.19524 |
| 4 | -0.59373 | -0.14816 | -0.61376 | -0.37941 | 0.58423 | 0.17733 | 0.62722 | 0.075242 |
| 5 | 0.078266 | 0.42784 | -0.45776 | -0.53541 | 0.17623 | 0.18933 | 0.61522 | -0.24876 |
| 6 | 0.44973 | -0.28384 | 0.15776 | 0.18741 | -0.51223 | 0.55467 | 0.14078 | 0.10476 |
| 7 | -0.006266 | -0.043835 | -0.21424 | 0.54741 | 0.54377 | 0.098666 | 0.48878 | -0.31524 |
| 8 | -0.54573 | 0.12784 | -0.42176 | -0.31941 | -0.33977 | -0.38667 | -0.52478 | 0.57924 |
| 9 | -0.37827 | -0.64384 | -0.40624 | 0.42741 | 0.57977 | 0.17067 | 0.14078 | 0.16476 |
| 10 | 0.46173 | -0.091835 | -0.13024 | 0.43941 | 0.19577 | 0.038666 | 0.0087827 | -0.38724 |
| 11 | -0.52173 | 0.13984 | 0.45424 | -0.36741 | 0.068228 | -0.29067 | -0.21278 | 0.56724 |
| 12 | 0.053734 | -0.007835 | 0.38576 | 0.055406 | 0.23177 | -0.26133 | -0.42322 | 0.53676 |
| 13 | -0.34227 | 0.11216 | -0.55024 | -0.32859 | 0.50777 | -0.29733 | -0.59122 | -0.59124 |
| 14 | 0.18627 | 0.28384 | -0.19376 | -0.019406 | -0.11177 | 0.41733 | 0.087217 | -0.20076 |
| 15 | -0.60573 | 0.019835 | -0.50576 | 0.13659 | -0.087772 | 0.15333 | 0.027217 | 0.35124 |
| 16 | -0.52173 | 0.24784 | -0.18176 | -0.53541 | -0.13577 | -0.48267 | -0.52478 | 0.051242 |

## 3.5. Evaluasi

Proses evaluasi merupakan tahapan akhir setelah perangkat lunak selesai dikembangkan. Evaluasi yang nantinya akan dilakukan meliputi uji coba terhadap sistem yang telah dibangun dengan melakukan proses segmentasi serta pembelajaran dan proses pengenalan terhadap citra yang berisi tulisan tangan. Dari 10 macam tulisan tangan dari orang yang berbeda, 7 diantaranya akan digunakan untuk data pelatihan dan sisanya akan digunakan untuk data pengujian pengenalan tulisan tangan.

Proses pengujian segmentasi dilakukan dengan dengan menulis kata dengan masing-masing 5 variasi tulisan yang berbeda dan akan dilakukan uji segmentasi pada tiap kata tersebut. Pada

Tabel 3.19 akan ditunjukkan prosentase keberhasilan dari tiap-tiap huruf dibandingkan dengan jumlah total dari huruf yang didapat. Maka dari tabel tersebut akan diperoleh huruf yang berhasil dan tidak berhasil disegmentasi.

**Tabel 3.19** Hasil Uji Segmentasi

| No | Kata | Jml Huruf | Orang ke-1 | Orang ke-2 | Orang ke-3 | Orang ke-4 | Orang ke-5 | Berhasil (%) |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | | | | | | |

Dari hasil pada Tabel 3.19, dapat diketahui tingkat akurasi per huruf yang disegmentasi. Proses selanjutnya yang dilakukan dalam proses pengujian adalah pengujian terhadap aplikasi untuk menemukan nilai parameter yang optimal. Percobaan dilakukan terhadap beberapa parameter yang mempengaruhi, yang meliputi ukuran citra, ukuran hidden layer, nilai laju pembelajaran (*learning rate*), serta nilai MSE (*Mean Square Error*).

**Tabel 3.20** Hasil percobaan pengaruh ukuran citra

| **Percobaan** | **Ukuran citra (pixel)** | ***Epoch*** | **MSE** |
|---|---|---|---|
| 1 | | | |
| 2 | | | |
| 3 | | | |

**Tabel 3.21** Hasil percobaan pengaruh jumlah unit hidden layer

| **Percobaan** | **Jumlah unit *Hidden Layer*** | ***Epoch*** | **MSE** |
|---|---|---|---|
| 1 | | | |
| 2 | | | |
| 3 | | | |

**Tabel 3.22** Hasil percobaan pengaruh *learning rate*

| **Percobaan** | ***Learning rate*** | ***Epoch*** | **MSE** |
|---|---|---|---|
| 1 | | | |
| 2 | | | |
| 3 | | | |

Uji coba selanjutnya adalah proses pengenalan karakter dengan menggabungkan proses segmentasi dengan proses pengenalan dengan algoritma Optical Backpropagation. Uji coba dilakukan dengan menggunakan huruf-huruf yang memiliki tingkat keberhasilan pengujian yang tinggi. Perancangan hasil uji coba proses pengenalan karakter dapat dilihat pada Tabel 3.23. Tingkat keakuratan (*Recognition Rate*) dari huruf yang dikenali dihitung dengan menggunakan Persamaan 3.1

$$keberhasilan = \frac{jumlah\ huruf\ dikenali\ dengan\ benar}{jumlah\ huruf\ yang\ diujikan}\ x\ 100\% \quad (3.1)$$

**Tabel 3.23** Hasil Uji Sistem Keseluruhan

| No | Kata | Jml Huruf | Orang ke-1 | Orang ke-2 | Orang ke-3 | Orang ke-4 | Orang ke-5 | Berhasil (%) |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

# BAB IV
# IMPLEMENTASI DAN PEMBAHASAN

## 4.1. Lingkungan Implementasi

Perangkat keras yang digunakan dalam pengembangan sistem pengenalan tulisan tangan ini adalah sebagai berikut :

1. Prosesor Intel Dual Core 3.0 Ghz
2. Memori 2 GB
3. Hardisk dengan kapasitas 500 GB
4. Monitor 15”
5. Keyboard
6. Mouse
7. Scanner Canon MP258

Perangkat lunak yang digunakan dalam pengembangan sistem pengenalan tulisan tangan ini adalah :

1. Sistem Operasi Windows 7
2. Borland Delphi 2007
3. Editor gambar Microsoft Paint

## 4.2 Implementasi Program

### 4.2.1. Implementasi Pemrosesan Citra

Implementasi pemrosesan citra terdiri dari beberapa subproses seperti yang akan dijelaskan sebagai berikut.

#### 4.2.1.1 Struktur Data Pemrosesan Citra

Struktur data yang digunakan dalam pemrosesan citra seperti pada *Sourcecode* 4.1.

```
1   DEFAULT_BW_THRESHOLD = 127;
2   DEFAULT_NOISE_THRESHOLD = 10;
3   DEFAULT_SPACE_WIDTH = 22;
4
5   TImageArray = array of array of Byte;
6   TChar = record
7     Pixels: array of TPoint;
8     MinPixel: TPoint;
9     MaxPixel: TPoint;
10    Noise: Boolean;
11    CharCode: Char;
12  end;
```

```
13  TWord = record
14    Left: Integer;
15    Right: Integer;
16    Chars: array of TChar;
17  end;
18  TLine = record
19    Top: Integer;
20    Bottom: Integer;
21    Words: array of TWord;
22  end;
23  TLines = array of TLine;
```

***Source code* 4.1** Struktur data pemrosesan citra

*TImageArray* adalah sebuah *array* dua dimensi yang berfungsi untuk menandai piksel yang aktif dari citra yang dimasukkan. Piksel aktif adalah piksel citra *input*an yang bila di-*grayscale*-kan memiliki nilai kurang dari nilai *threshold* (*DEFAULT_BW_THRESHOLD*) yang telah ditentukan. *TImageArray* memiliki ukuran dimensi yang sama dengan citra *input*an. Dalam *TImageArray* piksel aktif akan diwakili oleh nilai 1 dan piksel non aktif akan diwakili dengan nilai 0. *TimageArray* akan menjadi representasi dari citra *input*an dan *TImageArray* inilah yang akan digunakan dalam proses segmentasi.

Proses segmentasi terhadap citra *input* terbagi ke dalam tiga langkah yaitu menentukan batas atas dan bawah (segmentasi baris), menentukan batas kiri dan kanan dari masing-masing kata (segmentasi kata), dan segmentasi karakter. *TLine* adalah komponen dasar dalam proses segmentasi baris, sedangkan *TWord* dan *TChar*, masing-masing menjadi komponen penting yang berperan dalam proses segmentasi kata dan segmentasi huruf.

*TLine* adalah sebuah *record* yang berfungsi untuk menyimpan satu baris tulisan dalam citra *input*an. Dari satu atau beberapa baris tulisan yang ditemukan kemudian disimpan dalam *TLines* yakni sebuah *array* dari *TLine*. *TLine* akan menyimpan koordinat y piksel aktif yang menjadi batas atas (*TLine.Top*) dan batas bawah (*TLine.Bottom*) dari baris tulisan yang ditemukan.

Untuk menentukan apakah sebuah piksel aktif menjadi batas bawah dari baris tulisan pertama dan piksel aktif yang lain menjadi batas atas dari baris tulisan selanjutnya adalah dengan menghitung selisih antara koordinat y antara kedua piksel tersebut. Bila selisih lebih besar dari batas pemisah baris yang telah ditentukan sebelumnya, maka piksel aktif yang satu akan menjadi batas bawah

dari baris tulisan ke-i dan piksel yang lain akan menjadi batas atas baris tulisan ke-(i+1).

*TWord* adalah *record* yang digunakan untuk menyimpan batas kiri dan kanan dari tiap-tiap kata, dari tiap baris tulisan yang sebelumnya telah disimpan pada *TLine*. Untuk memisahkan kata yang satu dengan kata yang lain, akan dihitung selisih koordinat x dari piksel aktif yang satu dengan piksel aktif yang lain. Bila selisihnya lebih besar atau sama dengan dari nilai batas pemisah kata (*DEFAULT_SPACE_WIDTH*), maka piksel aktif yang satu akan menjadi kanan dari kata ke-i dan piksel aktif yang lain akan menjadi batas kiri kata ke-(i+1).

#### 4.2.1.2 *Thresholding*

Proses *thresholding* dilakukan dengan membandingkan nilai *grayscale* dari tiap piksel dengan nilai yang digunakan untuk proses *thresholding*. Piksel yang memiliki nilai *grayscale* lebih dari nilai *threshold* (putih), maka akan diwakili dengan nilai 0. Piksel yang memiliki nilai *grayscale* kurang dari *threshold* (hitam) akan diwakili dengan nilai 1. Proses ini ditunjukkan pada *Sourcecode* 4.2

```
1   procedure TOBPFrame.FilterBnW;
2   var
3     x, y, R, G, B:  Integer;  Grayscale: Real;
4   begin
5     SetLength(FImageArray, Image.Width, Image.Height);
6     for x := 0 to High(FImageArray) do
7       for y := 0 to High(FImageArray[0]) do
8       begin
9         GetRGBColor(Image.Canvas.Pixels[x, y], R, G, B);
10        Grayscale := (0.299 * R) + (0.587 * G) + (0.114
11  * B);
12        if GrayScale >= FBWThreshold then
13          FImageArray[x, y] := 0 // putih
14        else
15          FImageArray[x, y] := 1;  // hitam
16      end;
17  end;
```

***Sourcecode* 4.2** *Tresholding*

Proses *thresholding* ini dilakukan dengan cara melakukan iterasi ke semua piksel yang dimulai dari koordinat (0,0). Untuk mendapatkan nilai *RGB* dari setiap piksel, maka digunakan fungsi *GetRGBColor* seperti pada *Source code* 4.3.

```
1  procedure TOBPFrame.GetRGBColor(Color: TColor; out R:
2  Integer; out G: Integer; out B: Integer);
3  begin
4    R :=getrvalue(color);
5    G :=getrvalue(color);
6    B :=getrvalue(color);
7   end;
```

***Sourcecode* 4.3** Fungsi untuk mendapatkan nilai RGB

#### 4.2.1.3 Segmentasi

Proses segmentasi terbagi menjadi 3 langkah, yaitu segmentasi baris, segmentasi kata, dan segmentasi huruf (karakter). Segmentasi baris digunakan untuk mendapatkan koordinat atas dan bawah pada setiap baris tulisan. Proses ini ditunjukkan pada *Sourcecode* 4.4. Segmentasi kata akan menghasilkan batas kiri dan batas kanan kata untuk setiap baris tulisan yang telah ditemukan pada proses segmentasi baris. Proses ini ditunjukkan pada *Sourcecode* 4.5. Segmentasi yang terakhir adalah segmentasi huruf (karakter).

Pada segmentasi karakter, proses awal yang dilakukan adalah dengan menghitung jarak piksel aktif paling atas dengan piksel aktif terbawah untuk tiap-tiap kolom citra dan nilainya disimpan dalam sebuah array. Pada setiap kolom dari citra yang jarak piksel teratas dengan terbawahnya kurang dari batas, maka piksel pada kolom dianggap sebagai sambungan antar huruf dan akan dihapus. Proses ini ditunjukkan pada *Sourcecode* 4.6.

Proses berikutnya dari segmentasi karakater adalah mencari piksel-piksel yang berdekatan yang menyusun suatu karakter. Setiap piksel yang menyusun suatu kata terlebih dahulu dicek apakah merupakan piksel yang bernilai 1 atau bukan. Bila piksel tersebut tidak bernilai 1 (bukan hitam) maka piksel tersebut tidak diberi label atau dinotasikan dengan *UNLABELED*. Akan tetapi, bila piksel bernilai 1 (hitam) maka dilakukan pengecekan terlebih dahulu apakah piksel bagian atas, kiri atas, kiri, dan kiri bawah dari piksel tersebut berlabel atau tidak. Bila salah satu dari piksel terdekat tersebut berlabel, maka piksel tersebut diberi label dengan nilai yang sama dengan nilai piksel terdekat yang berlabel. Dengan cara yang demikian, maka piksel- piksel yang berdekatan yang menyusun suatu karakter akan memiliki label atau nilai yang sama. Label atau nilai akan bertambah setiap kali ditemukan karakter baru. Dari setiap karakter yang ditemukan kemudian disimpan koordinat minimum dan maksimumnya. Proses ini ditunjukkan pada *Sourcecode* 4.7.

```
1   procedure TOBPFrame.SegmentLines;
2   var
3     BlankRow: Boolean;
4     SearchFor: (sfTop, sfBottom);
5     x, y: Integer;
6   begin
7     y := 0;
8     SearchFor := sfTop;
9     SetLength(FLines, 0);
10    repeat
11      BlankRow := True;
12      for x := 0 to High(FImageArray) do
13        if FImageArray[x, y] = 1 then
14        begin
15          BlankRow := False;
16          if SearchFor = sfTop then
17          begin
18            SetLength(FLines, Length(FLines) + 1);
19            FLines[High(FLines)].Top := y;
20            SearchFor := sfBottom;
21          end;
22          Break;
23        end;
24      if (SearchFor = sfBottom) and BlankRow then
25      begin
26        FLines[High(FLines)].Bottom := y - 1;
27        SearchFor := sfTop;
28      end;
29      y := y + 1;
30    until y = Length(FImageArray[0]);
31    if       SearchFor       =       sfBottom       then
32  FLines[High(FLines)].Bottom := y - 1;
33  end;
```

***Source code* 4.4** Proses segmentasi baris

```
1   procedure TOBPFrame.SegmentWords;
2   var
3     IxLine: Integer;
4     NBlankColumn: Integer;
5     SearchFor: (sfLeft, sfRight);
6     x, y: Integer;
7   begin
8     for IxLine := 0 to High(FLines) do
9       with FLines[IxLine] do
10      begin
11        x := 0;
12        SearchFor := sfLeft;
13        NBlankColumn := 0;
```

```
14        repeat
15          NBlankColumn := NBlankColumn + 1;
16          for y := Top to Bottom do
17            if FImageArray[x, y] = 1 then
18            begin
19              NBlankColumn := 0;
20              if SearchFor = sfLeft then
21              begin
22                SetLength(Words, Length(Words) + 1);
23                Words[High(Words)].Left := x;
24                SearchFor := sfRight;
25              end;
26              Break;
27            end;
28          if (SearchFor = sfRight) and (NBlankColumn >=
29  FSpaceWidth) then
30          begin
31            Words[High(Words)].Right     :=     x     -
32  NBlankColumn;
33            SearchFor := sfLeft;
34          end;
35          x := x + 1;
36        until x = Length(FImageArray);
37        if      SearchFor      =      sfRight      then
38  Words[High(Words)].Right := x - 1;
39      end;
40  end;
```

***Sourcecode* 4.5** Proses segmentasi kata

```
1   procedure TOBPFrame.ModVHist;
2   var I,x,y,count,atas,bawah:integer;
3     ixline,tres: Integer;
4     ixword: Integer;
5     searchVert:(sVtop,sVbottom);
6   begin
7   tres:=StrToInt(FMain.JvSpinEdit1.Text);
8   for ixline := 0 to High(FLines) do
9     for ixword := 0 to High(FLines[ixline].Words) do
10    with FLines[ixline]do
11    begin
12
13  SetLength(Words[ixword].Akarakter,Words[ixword].Right-
14  Words[ixword].Left);
15      for    x    :=    0    to    Words[ixword].Right-
16  Words[ixword].Left-1 do
17        begin
18        count:=0;
19        atas:=0;bawah:=0;
```

```
20        searchVert:=sVtop;
21        for y := top to Bottom do
22          begin
23           if FImageArray[x+Words[ixword].Left,y]=1 then
24            begin
25              if searchVert=sVtop then
26              begin
27                atas:=y;
28                bawah:=y;
29                searchVert:=sVbottom;
30              end
31              else
32              begin
33                bawah:=y;
34              end;
35            end;
36          end;
37          count:=bawah-atas;
38          Words[ixword].Akarakter[x]:=count;
39          if count<tres then
40          begin
41            for I := top to Bottom do
42            begin
43              FImageArray[x+Words[ixword].Left,i]:=0;
44            end;
45          end;
46        end;
47    end;
48  end;
```

***Sourcecode* 4.6** Proses menghapus sambungan antar huruf dengan *modified vertical histogram*

```
1   for IxLine := 0 to High(FLines) do
2    for IxWord := 0 to High(FLines[IxLine].Words) do
3    begin
4     with FLines[IxLine] do
5     begin NewLabel := 0;
6      SetLength(LabeledPixels, Words[IxWord].Right -
7                Words[IxWord].Left + 2, Bottom - Top + 3);
8      for x := 0 to High(LabeledPixels) do
9      begin LabeledPixels[x, 0] := UNLABELED;
10      LabeledPixels[x,     High(LabeledPixels[0])]     :=
11  UNLABELED;
12     end;
13      for y := 0 to High(LabeledPixels[0]) do
14       LabeledPixels[0, y] := UNLABELED;
15      for x := 1 to High(LabeledPixels) do
16       for y := 1 to High(LabeledPixels[0]) - 1 do
```

```
17      begin
18      if  FImageArray[x+Words[IxWord].Left-1,y+Top-1]  <>
19 0 then       LabeledPixels[x, y] := UNLABELED
20      else
21      begin
22       PixelLabels[drTop] := LabeledPixels[x, y - 1];
23       PixelLabels[drTopLeft]:=LabeledPixels[x-1, y-1];
24       PixelLabels[drLeft] := LabeledPixels[x - 1, y];
25       PixelLabels[drBottomLeft]:=LabeledPixels[x-
26 1,y+1];
27        if (PixelLabels[drTop]=UNLABELED) and
28           (PixelLabels[drTopLeft] = UNLABELED) and
29           (PixelLabels[drLeft] = UNLABELED) and
30           (PixelLabels[drBottomLeft] = UNLABELED) then
31       begin
32        LabeledPixels[x,y]:=NewLabel;
33        SetLength(IdChars,NewLabel+1);
34        IdChars[NewLabel]  :=  NewLabel;  NewLabel  :=
35 NewLabel+1;
36       end
37       else
38       begin ReplacementLabel := UNLABELED;
39        for Dir := drTop to drBottomLeft do
40         if PixelLabels[Dir] <> UNLABELED then
41          if ReplacementLabel = UNLABELED then
42          begin
43         LabeledPixels[x,y]:=IdChars[PixelLabels[Dir]];
44           ReplacementLabel:=IdChars[PixelLabels[Dir]];
45          end
46          else
47           IdChars[PixelLabels[Dir]]:=ReplacementLabel;
48      end;
49     end;
50    end;    end;
51    for IxChar := 0 to High(IdChars) do
52     while IdChars[IdChars[IxChar]]<>IdChars[IxChar] do
53          IdChars[IxChar] := IdChars[IdChars[IxChar]];
54  end;
55 end;
```

***Sourcecode* 4.7** Proses segmentasi karakter

#### 4.2.1.4 Normalisasi Ukuran Citra

Proses normalisasi dilakukan untuk setiap karakter yang telah ditemukan pada saat proses segmentasi. Proses ini akan menjadikan ukuran panjang dan lebar setiap citra karakter sesuai dengan ukuran yang dibutuhkan. Proses ini ditunjukkan seperti pada *Sourcecode* 4.8

```
1   function GetAutoFitBitmap(Bitmap: TBitmap;
2                             Width: Integer; Height:
3   Integer): TBitmap;
4   var
5     BoundRect: TRect;
6     FitRect,Myrect: TRect;
7     TmpBitmap: TBitmap;
8     x, y: Integer;tinggi, lebar :integer;
9   begin
10    Assert((Width > 0) and (Height > 0));
11
12    TmpBitmap := TBitmap.Create;
13    with Bitmap do
14    begin
15      BoundRect := Rect(Bitmap.Width - 1, Bitmap.Height
16  - 1, 0, 0);
17      for y := 0 to Bitmap.Height - 1 do
18        for x := 0 to Bitmap.Width - 1 do
19          if Canvas.Pixels[x, y] = clBlack then
20          begin
21       if x < BoundRect.Left then BoundRect.Left := x;
22       if y < BoundRect.Top then BoundRect.Top := y;
23       if x > BoundRect.Right then BoundRect.Right := x;
24       if y>BoundRect.Bottom then BoundRect.Bottom := y;
25          end;
26    end;
27    BoundRect.Right := BoundRect.Right + 1;
28    BoundRect.Bottom := BoundRect.Bottom + 1;
29    tinggi:= BoundRect.Bottom - BoundRect.Top;
30    lebar :=BoundRect.Right - BoundRect.Left;
31    Myrect:=Rect(0, 0, lebar, tinggi);
32    with TmpBitmap do
33    begin
34      Width := BoundRect.Right - BoundRect.Left;
35      Height := BoundRect.Bottom - BoundRect.Top;
36      Canvas.CopyRect(Myrect, Bitmap.Canvas, BoundRect);
37    end;
38    Result := TBitmap.Create;
39    Result.Width := Width;
40    Result.Height := Height;
41    Result.Canvas.FillRect(Rect(0, 0, Width, Height));
42    FitRect := Rect(0, 0, BoundRect.Right -
43  BoundRect.Left,BoundRect.Bottom -  BoundRect.Top);
44    if (Width / FitRect.Right) * FitRect.Bottom <=
    Height then
      begin
        FitRect.Bottom := Ceil((Width / FitRect.Right) *
    FitRect.Bottom);
        FitRect.Right := Width;
```

```
    end
    else
    begin
      FitRect.Right := Ceil((Height / FitRect.Bottom) *
FitRect.Right);
      FitRect.Bottom := Height;
    end;
    Result.Canvas.StretchDraw(FitRect, TmpBitmap);
    TmpBitmap.Free;
end;
```

***Sourcecode* 4.8** Normalisasi ukuran citra

#### 4.2.1.5 Ekstraksi

Setelah proses normalisasi selesai dilakukan, kemudian dilakukan proses ekstraksi data. Proses ini dilakukan mengkodekan piksel menjadi nilai 1 atau -1. Jika piksel[x,y] berwarna hitam, maka akan dikodekan menjadi angka 1, sedangkan jika tidak, akan dikodekan dengan angka -1. Penggunaan angka 1 dan -1 ini berdasarkan fungsi aktivasi bipolar sigmoid yang nantinya akan digunakan pada proses jaringan syaraf. Proses ini seperti ditunjukkan pada *Sourcecode* 4.9.

```
1  class procedure TOBP.BitmapToLayer(Bitmap: TBitmap;
2  Layer: TLayer);
3  var
4    i: Integer;
5    x, y: Integer;
6  begin
7    Assert(Length(Layer) = (Bitmap.Width *
8  Bitmap.Height) + 1);
9    i := 1;
10   for y := 0 to Bitmap.Height - 1 do
11     for x := 0 to Bitmap.Width - 1 do
12     begin
13       if Bitmap.Canvas.Pixels[x, y] = clBlack then
14         Layer[i] := 1
15       else
16         Layer[i] := -1;
17       i := i + 1;
18     end;
19 end;
```

***Sourcecode* 4.9** Proses ekstraksi fitur

### 4.2.2. Implementasi Jaringan Syaraf Tiruan

#### 4.2.2.1 Struktur Data Implementasi Jaringan Syaraf Tiruan

Dalam implementasi jaringan syaraf tiruan, setiap pasangan data pelatihan akan disimpan dalam record TTrainingPairs yang akan menyimpan data-data neuron input (InputPatterns) dan neuron target (*TargetPatterns*) serta karakter huruf yang dikenali (*TargetPatternClass*) berdasarkan target tersebut. Implementasi struktur data jaringan syaraf tiruan ditunjukkan pada *Sourcecode* 4.10.

```
1   //- Deklarasi Neuron dan bobot -
2   TNeuron = Real;
3   TLayer = array of TNeuron;
4   TWeight = Single;
5   TLayerWeights = array of array of TWeight;
6
7   //- Training set -
8   TPatternClass = string;
9   TTrainingPairs = record
10    InputPatterns: array of TLayer;
11    TargetPattern: TLayer;
12    TargetPatternClass: TPatternClass;
13  end;
14  TTrainingSet = array of TTrainingPairs;
```

**Sourcecode 4.10** Struktur data jaringan syaraf tiruan

#### 4.2.2.2 Proses Pelatihan

##### 4.2.2.2.1 Inisialisasi Bobot

Proses inisialisasi bobot dilakukan pada fungsi *InitWeights.* Fungsi ini bertujuan untuk menginisialiasai nilai awal bobot dari *unit-unit* yang berada pada *input layer* ke *unit-unit* yang berada pada *layer hidden,* dan bobot dari *unit-unit* yang berada pada *layer hidden* ke *unit-unit* yang berada pada *layer output.* Bobot diinisialisasi secara acak dengan nilai berkisar antara -0,5 sampai 0,5. Implementasi dari inisialisasi bobot ditunjukkan pada *Sourcecode* 4.11.

```
1   procedure TOBP.InitWeights;
2   var
3     i, j, k: Integer;
4     maks,mins:Single;
5   begin
6   maks:=0.5;
```

```
7   mins:=-0.5;
8     Randomize;
9       for j := 0 to FNHiddenNeuron do
10      for i := 0 to FNInputNeuron do
11        FHiddenLayerWeights[j, i] := Random(100)*(maks-
12  mins)*0.01+mins;
13      for k := 1 to FNOutputNeuron do
14      for j := 0 to FNHiddenNeuron do
15        FOutputLayerWeights[k, j] := Random(100)*(maks-
16  mins)*0.01+mins;
17   end;
```

***Sourcecode* 4.11** Inisialisasi bobot awal

#### 4.2.2.2.2 *Feedforward*

Proses feedforward merupakan proses yang bertujuan untuk mengaktifkan *unit-unit* yang berada pada *layer hidden* dan *layer output* dengan menggunakan fungsi aktivasi bipolar sigmoid pada persamaan 2.12. Implementasi dari proses *Feedforward* ditunjukkan pada *Sourcecode* 4.12 dan implementasi dari fungsi aktivasi bipolar sigmoid ditunjukkan pada *Sourcecode* 4.13.

```
1   procedure TOBP.FeedForward;
2   var
3     i, j, k: Integer;          begin
4     for j := 1 to FNHiddenNeuron do
5     begin
6       FHiddenLayer[j] := 0;
7       for i := 0 to FNInputNeuron do
8         FHiddenLayer[j] := FHiddenLayer[j] +
9                            (FInputLayer[i] *
10  FHiddenLayerWeights[j, i]);
11      FHiddenLayer[j] :=
12  BipolarSigmoid(FHiddenLayer[j]);
13    end;
14    for k := 1 to FNOutputNeuron do
15    begin
16      FOutputLayer[k] := 0;
17      for j := 0 to FNHiddenNeuron do
18        FOutputLayer[k] := FOutputLayer[k] +
19                           (FHiddenLayer[j] *
20  FOutputLayerWeights[k, j]);
21      FOutputLayer[k] :=
22  BipolarSigmoid(FOutputLayer[k]);
23    end;
24  end;
```

***Sourcecode* 4.12** Proses *feedforward*

```
function TOBP.BipolarSigmoid(x: Single): Single;
begin
  try
    Result := (2 / (1 + Exp(-x))) - 1;
  except
    on EOverflow do Result := 0;
  end;
end;
```

***Sourcecode* 4.13** Fungsi pengaktif bipolar sigmoid

#### 4.2.2.2.3 Proses perhitungan *error* pada *output layer*

Nilai *error* pada output layer ini digunakan untuk mendapatkan nilai koreksi bobot pada *output layer* yang nantinya digunakan untuk menghitung nilai bobot baru pada *output layer*. Proses perhitungan nilai *error* ini mengimplementasikan Optical Backpropagation. Implementasi dari proses perhitungan *error* pada *output layer* ditunjukkan pada *Sourcecode* 4.14.

```
for k := 1 to FNOutputNeuron do
begin
  Error := FTrainingSet[l].TargetPattern[k]-
FOutputLayer[k];
  if Error>=0 then
tempError:=1+Exp(error*Error)
  else
  tempError:=-(1+Exp(Error*Error));
  if abs(Error) >= FErrorThreshold then
     FNNeuronError := FNNeuronError + 1;
  FTrainingError := FTrainingError + (Error * Error);
  OutputErrors[k] := tempError *
BipolarSigmoidDerivation(FOutputLayer[k]);
  for j := 0 to FNHiddenNeuron do
   OutputWeightsCorrection[k,j]:= FLearningRate *
OutputErrors[k] * FHiddenLayer[j];
end;
```

***Sourcecode* 4.14** Proses perhitungan nilai error pada output layer

#### 4.2.2.2.4 Proses perhitungan *error* pada *hidden layer*

Nilai *error* yang terdapat pada *hidden layer* dihitung seperti tampak pada *Source code* 4.15. Nilai *error* ini digunakan untuk menghitung nilai koreksi bobot *hidden layer* yang kemudian digunakan untuk menghitung nilai bobot baru pada *hidden layer*.

```
1   //- Hitung informasi error bobot hidden layer
2   for j := 1 to FNHiddenNeuron do
3   begin
4     HiddenErrors[j] := 0;
5
6     for k := 1 to FNOutputNeuron do
7       HiddenErrors[j] := HiddenErrors[j] +
8         (OutputErrors[k] * FOutputLayerWeights[k, j]);
9
10    HiddenErrors[j] := HiddenErrors[j] *
11        BipolarSigmoidDerivation(FHiddenLayer[j]);
12
13    for i := 0 to FNInputNeuron do
14      HiddenWeightsCorrection[j, i] := FLearningRate *
15      HiddenErrors[j] * FInputLayer[i];
16  end;
```

***Sourcecode* 4.15** Proses perhitungan nilai error pada hidden layer

Implementasi perhitungan *error* pada hidden layer menggunakan fungsi aktivasi turunan dari bipolar sigmoid seperti ditunjukkan pada *Source code* 4.16

```
1   // Aktivasi fungsi turunan bipolar sigmoid
2   function TBackProp.BipolarSigmoidDerivation(Fx:
3   Single): Single;
4   begin
5     Result := ((1 + Fx) * (1 - Fx)) / 2;
6   end;
```

***Source code* 4.16** Turunan fungsi Bipolar Sigmoid

#### 4.2.2.2.5 Proses *update* bobot pada *output layer*

Proses perhitungan bobot baru pada *output layer* diimplementasikan seperti pada *Sourcecode* 4.17.

```
1   //- Update bobot output layer and bias
2   for k := 1 to FNOutputNeuron do
3     for j := 0 to FNHiddenNeuron do
4       FOutputLayerWeights[k, j] :=
5   FOutputLayerWeights[k, j]
6         + OutputWeightsCorrection[k, j];
```

***Source code* 4.17** Proses menghitung bobot baru pada *output layer*

#### 4.2.2.2.6 Proses *update* bobot pada *hidden layer*

Perhitungan bobot baru pada *hidden layer* diimplementasikan seperti pada *Sourcecode* 4.18.

```
//- Updates bobot hidden layer dan bias
for j := 1 to FNHiddenNeuron do
  for i := 0 to FNInputNeuron do
    FHiddenLayerWeights[j, i] := FHiddenLayerWeights[j, i]
      + HiddenWeightsCorrection[j, i];
```

***Source code* 4.18** Proses menghitung bobot baru pada *hidden layer*

### 4.2.2.3 Proses pengenalan

Proses pengenalan tulisan diawali dengan proses pengolahan citra (*tresholding*, segmentasi baris, segmentasi kata, dan segmentasi karakter) sehingga didapat karakter-karakter huruf yang telah tersegmentasi. Tiap-tiap karakter tersebut kemudian akan dikenali satu per satu.

Proses pengenalan diawali dengan melakukan normalisasi ukuran citra menjadi ukuran yang telah ditentukan, kemudian dilakukan ekstraksi fitur. Berikutnya dilanjutkan dengan proses *feedforward*. Implementasi dari rangkaian proses ini ditunjukkan pada *Source code* 4.19. Proses berikutnya adalah proses mendapatkan simbol yang ditunjukkan pada *Source code* 4.20. Proses mendapatkan simbol ini dilakukan dengan mencari data yang memiliki selisih nilai terkecil dari *output layer* pada data yang tersimpan dengan *output layer* citra.

```
1   procedure TOBPFrame.Recognize;
2   var
3     IxLine, IxWord, IxChar: Integer;
4     i: Integer;
5     OrgBitmap: TBitmap;
6     StretchBitmap: TBitmap;
7     ResultSymbol: TPatternClass;
8   begin
9     Screen.Cursor := crHourglass;
10    FilterBnW;
11    SegmentLines;
12    SegmentWords;
13    ModVHist;
14    SegmentChars;
15    FResultText := '';
16    OrgBitmap := TBitmap.Create;
17    for IxLine := 0 to High(FLines) do
```

```
18    for IxWord := 0 to High(FLines[IxLine].Words) do
19     begin
20       for IxChar := 0 to
21  High(FLines[IxLine].Words[IxWord].Chars) do
22         with
23  FLines[IxLine].Words[IxWord].Chars[IxChar] do
24           if not Noise then
25           begin
26             OrgBitmap.Width := MaxPixel.x -
27  MinPixel.x + 1;
28             OrgBitmap.Height := MaxPixel.y -
29  MinPixel.y + 1;
30             OrgBitmap.Canvas.FillRect(
31               Rect(0, 0, OrgBitmap.Width,
32  OrgBitmap.Height));
33             for i := 0 to High(Pixels) do
34               OrgBitmap.Canvas.Pixels[
35                 Pixels[i].x - MinPixel.x,
36                 Pixels[i].y - MinPixel.y] :=
37  clBlack;
38   StretchBitmap := GetAutoFitBitmap(OrgBitmap,
39  OBP.InputPatternWidth, OBP.InputPatternHeight);
40   TOBP.BitmapToLayer(StretchBitmap, OBP.InputLayer);
41   StretchBitmap.Free;
42   OBP.FeedForward;
43   OBP.GetResult(ResultSymbol);
44   FResultText := FResultText + ResultSymbol;
45         end;
46       FResultText := FResultText + ' ';
47     end;
48    OrgBitmap.Free;
49    Screen.Cursor := crDefault;
50  end;
```

***Source code* 4.19** Proses pengenalan

```
1   procedure TOBP.GetResult(var Symbols: TStringList; var
2   Matchs: TStringList;var Unmatchs: TStringList);
3   var
4     HammingDistance: Integer;
5     k, l: Integer;
6     MatchPercentage: Single;
7     Pos: Integer;
8   begin
9     for l := 0 to High(FTrainingSet) do
10    begin
11      HammingDistance := 0;
12      for k := 1 to FNOutputNeuron do
13        if Abs(FTrainingSet[l].TargetPattern[k] -
```

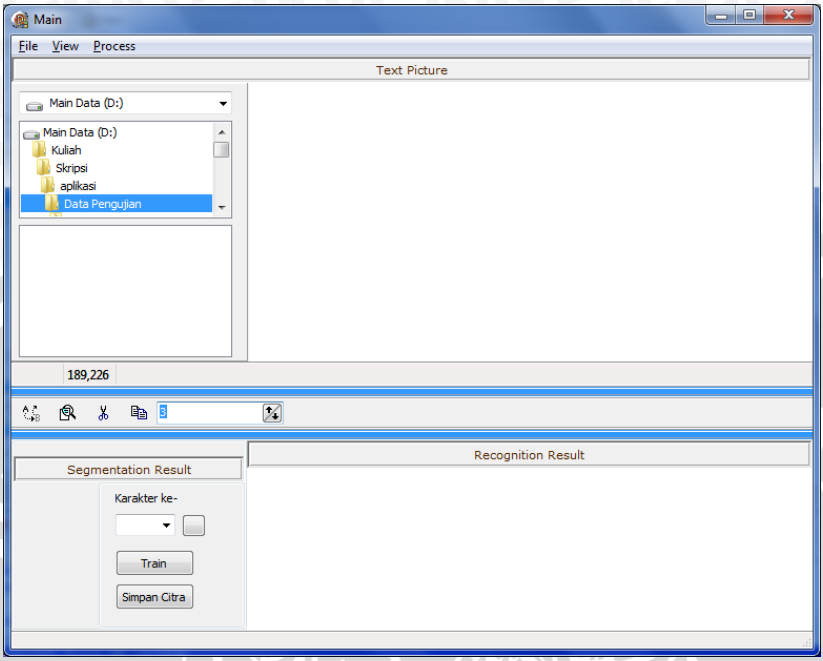
```
14  FOutputLayer[k])  >= FErrorThreshold then
15          HammingDistance := HammingDistance + 1;
16      MatchPercentage := (1 - (HammingDistance /
17  FNOutputNeuron)) * 100;
18      Pos := 0;
19
20      while (Pos < l) do
21   if (StrToFloat(Matchs[Pos]) < MatchPercentage) then
22          Break
23        else
24          Pos := Pos + 1;
25
26  Symbols.Insert(Pos,FTrainingSet[l].TargetPatternClass);
27  Matchs.Insert(Pos, FloatToStrF(MatchPercentage,
28  ffFixed, 4, 2));
29  Unmatchs.Insert(Pos, IntToStr(HammingDistance) + ' / '
30  +   IntToStr(FNOutputNeuron) + ' ');
31    end;
32    for Pos := 0 to Matchs.Count - 1 do
      Matchs[Pos] := Matchs[Pos] + '%';
    end;
```

***Source code* 4.20** Proses mendapatkan simbol

## 4.3 Implementasi antar muka

### 4.3.1 Form utama

Tampilan utama dari program pengenalan tulisan tangan menggunakan jaringan syaraf tiruan Optical Backpropagation dapat dilihat pada Gambar 4.1




**Gambar 4.1** Form utama

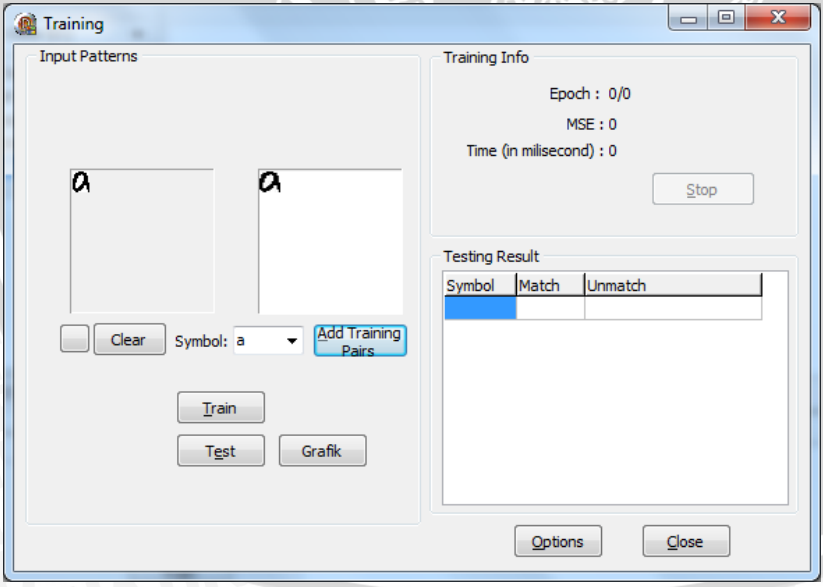
DalamForm utama tersebut terdiri atas beberapa bagian yang akan dijelaskan sebagai berikut :

1. Menu utama, terdiri dari 3 menu yaitu :
   a. File, terdiri dari 10 sub menu yaitu :
      - new knowledge, digunakan untuk membuat data pelatihan baru
      - open knowledge, digunakan untuk membuka data pelatihan yang sudah disimpan
      - saveknowledge dan save knowledge as : digunakan untuk menyimpan data pelatihan
      - new picture, digunakan untuk membersihkan citra utama
      - open picture, digunakan untuk membuka citra
      - save picture dan save picture as, digunakan untuk menyimpan citra
      - save result dan save result as, digunakan untuk menyimpan teks hasil pengenalan
      - exit, digunakan untuk keluar dari program

b. View, terdiri dari 4 sub menu yang digunakan untuk menampilkan *toolbar* pada form utama
c. Process, terdiri dari 3 submenu yaitu :
   - Training, digunakan untuk memanggil form training
   - Recognize, digunakan untuk melakukan proses pengenalan tulisan
   - Option, digunakan untuk memanggil form untuk mengatur parameter-parameter jaringan syaraf tiruan.
d. Toolbar, yang berisi tombol untuk memanggil form *training,* tombol untuk proses pengenalan, tombol untuk proses segmentasi, serta *statusbar* untuk mengatur nilai *treshold* pada segmentasi karakter.
e. Frame 1, berisi PanelListbox dan FileListBox yang digunakan untuk membuka file citra yang akan dikenali serta citra inputan yang akan dikenali
f. Frame Teks hasil, yang akan menampilkan teks hasil pengenalan serta citra hasil segmentasi.

### 4.3.2 Form Training

Tampilan dari form training dapat dilihat pada Gambar 4.2




**Gambar 4.2** Form *training*

Pada form *training*, terdapat beberapa bagian, yaitu :

a. Tombol Clear, digunakan untuk menghapus citra
b. Combobox simbol, yang berisi huruf-huruf yang sudah dilatih
c. Tombol AddTrainingPairs, digunakan untuk menambah data pelatihan
d. Tombol Train, digunakan untuk melakukan proses pelatihan
e. Tombol Test digunakan untuk menampilkan hasil tes terhadap citra yang telah dilatih
f. Tombol Grafik, digunakan untuk menampilkan grafik pelatihan
g. Tombol Option, digunakan untuk mengatur parameter pada jaringan syaraf tiruan
h. Tombol Close, digunakan untuk menutup form training

## 4.4 Implementasi ujicoba

Aplikasi perangkat lunak yang telah diimplementasikan akan dilakukan beberapa pengujian dan analisa. Pengujian tersebut dilakukan meliputi :

1. Pengujian terhadap proses segmentasi karakter
2. Pengujian terhadap ukuran citra
3. Pengujian terhadap jumlah unit hidden layer
4. Pengujian terhadap nilai *learning rate*
5. Pengujian terhadap pengenalan huruf tunggal
6. Pengujian terhadap sistem keseluruhan

Dalam pengujian ini, sampel yang digunakan sebagai data uji akan dibagi menjadi tiga kelompok. Sampel 1merupakan kumpulan kata yang akan digunakan untuk pengujian proses segmentasi karakter. Sampel 2 merupakan kumpulan huruf yang berhasil disegmentasi dengan prosentase 100%. Sampel 3 merupakan kumpulan kata yang dibentuk dari sampel 2.

### 4.4.1 Pengujian terhadap proses segmentasi karakter

Pengujian segmentasi dilakukan dengan kata dari 5 orang penulis. masing-masing orang menulis 2 kali untuk kata yang sama. Dengan demikian akan terdapat 10 jenis tulisan untuk tiap-tiap kata.

Gambar 4.3 dan Gambar 4.4 menunjukkan contoh hasil dari segmentasi karakter.

**Gambar 4.3** Hasil segmentasi kata officially pada sampel 1

**Gambar 4.4** Hasil segmentasi kata officially pada sampel 4

Dalam pengujian, terlihat bahwa dalam 10 kali percobaan segmentasi pada tiap kata mempunyai hasil yang berbeda-beda. Hal ini disebabkan oleh model/gaya tulisan yang berbeda-beda dari masing-masing orang.

Pada Tabel 4.1 ditunjukkan prosentase keberhasilan dari tiap-tiap huruf dibandingkan dengan jumlah total dari huruf yang didapat.

**Tabel 4.1** Hasil pengujian segmentasi karakter

| No | Huruf | Jumlah huruf | Jumlah berhasil | prosentase (%) |
|---|---|---|---|---|
| 1 | a | 220 | 220 | 100 |
| 2 | b | 30 | 30 | 100 |
| 3 | c | 40 | 40 | 100 |
| 4 | d | 30 | 29 | 96.6666667 |
| 5 | e | 110 | 110 | 100 |
| 6 | f | 30 | 30 | 100 |
| 7 | g | 40 | 40 | 100 |
| 8 | h | 30 | 6 | 20 |
| 9 | i | 110 | 109 | 99.0909091 |
| 10 | j | 30 | 30 | 100 |
| 11 | k | 30 | 29 | 96.6666667 |
| 12 | l | 130 | 130 | 100 |
| 13 | m | 30 | 7 | 23.3333333 |
| 14 | n | 70 | 6 | 8.57142857 |
| 15 | o | 100 | 100 | 100 |
| 16 | p | 50 | 42 | 84 |
| 17 | q | 30 | 30 | 100 |
| 18 | r | 80 | 8 | 10 |

| | | | | |
|---|---|---|---|---|
| 19 | s | 30 | 30 | 100 |
| 20 | t | 60 | 60 | 100 |
| 21 | u | 60 | 0 | 0 |
| 22 | v | 40 | 17 | 42.5 |
| 23 | w | 40 | 0 | 0 |
| 24 | x | 20 | 19 | 95 |
| 25 | y | 20 | 12 | 60 |
| 26 | z | 30 | 30 | 100 |
| Rata-rata | | | | 78 |

Dari Tabel 4.1, diketahui huruf-huruf yang memiliki prosentase keberhasilan segmentasi 100% adalah huruf a,b,c,e,f,g,j,l,o,q,s,t, dan z. Huruf-huruf ini akan menjadi data untuk sampel 2.

### 4.4.2 Pengujian terhadap ukuran citra

Pengujian terhadap ukuran citra dilakukan terhadap sampel sebanyak 5 huruf dari sampel 2, yaitu a,b,e,l, dan q. Percobaan dilakukan untuk mengetahui pengaruh ukuran citra terhadap hasil pelatihan jaringan. Pengujian ini dilakukan menggunakan *hidden layer* sebanyak 20 dan *learning rate* sebesar 0,001. Hasil dari pengujian terhadap ukuran citra ditunjukkan pada Tabel 4.2.

**Tabel 4.2** Hasil pengujian terhadap ukuran citra

| No | Ukuran citra (piksel) | *Epoch* | MSE Rata-rata |
|---|---|---|---|
| 1 | 10 x 10 | 5000 | 0.00002358634 |
| 2 | 15 x 15 | 5000 | 0.00002359478 |
| 3 | 20 x 20 | 5000 | 0.00002352742 |
| 4 | 25 x 25 | 5000 | 0.00002354358 |

Dari hasil percobaan tersebut MSE paling kecil terjadi pada citra yang berukuran 20 x 20 piksel. Dengan demikian, percobaan berikutnya dilakukan dengan menggunakan citra *input* yang berukuran 20 x 20 *pixel*.

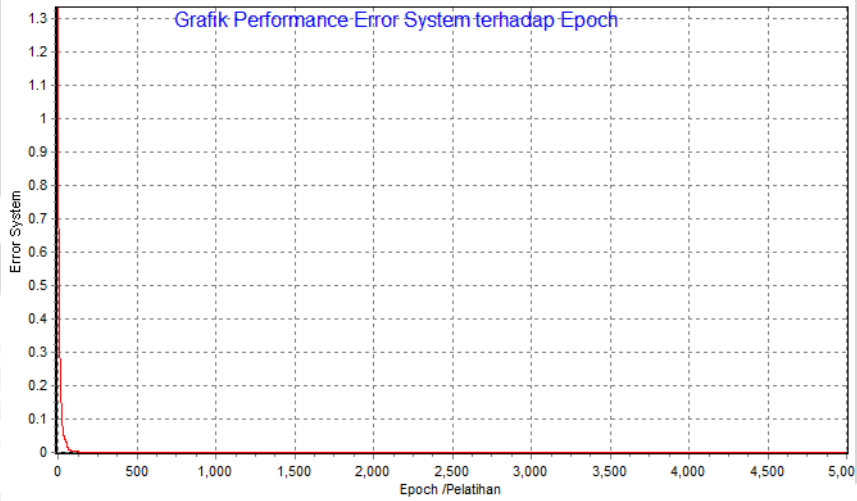
### 4.4.3 Pengujian terhadap jumlah unit hidden layer

Pengujian terhadap jumlah unit hidden layer dilakukan dengan menggunakan nilai *learning rate* yang sama dengan percobaan sebelumnya, yaitu 0,001. Ukuran citra inputan yang digunakan adaah 20 x 20 piksel. Pengujian dilakukan terhadap huruf a,b,e,l, dan q. Hasil dari pengujian terhadap jumlah unit hidden layer ditunjukkan pada Tabel 4.3.

**Tabel 4.3** Pengujian terhadap jumlah unit hidden layer

| No | Jumlah unit *Hidden Layer* | *Epoch* | Rata-rata MSE |
|---|---|---|---|
| 1 | 20 | 5000 | 0.0000236600 |
| 2 | 40 | 5000 | 0.0000060792 |
| 3 | 60 | 5000 | 0.0000027570 |
| 4 | 80 | 5000 | 0.0000015456 |
| 5 | 100 | 5000 | 0.0000010004 |

Dari pengujian terhadap jumlah unit *hidden layer* diperoleh MSE terkecil yang terjadi pada simulasi ke-5, yaitu dengan jumlah *hidden layer* sebanyak 100 buah. Gambar 4.5 menunjukkan grafik penurunan MSE untuk *hidden layer* sebanyak 100 buah.




**Gambar 4.5** Grafik penurunan MSE hidden layer=100

### 4.4.4 Pengujian terhadap nilai *learning rate*

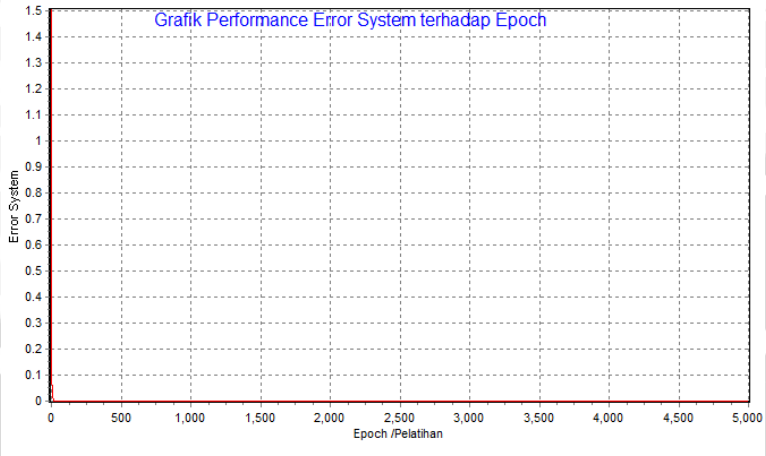
Pengujian terhadap nilai *learning rate* dilakukan dengan jumlah unit *hidden layer* sebanyak 100 unit dengan ukuran citra 20 x 20 piksel. Pengujian dilakukan terhadap masing-masing individu huruf a,b,e,l, dan q.

**Tabel 4.4** Pengujian terhadap nilai *learning rate*

| **Percobaan ke-** | ***Learning rate*** | ***Epoch*** | **MSE** |
|---|---|---|---|
| 1 | 0,001 | 5000 | 0.00000102470 |
| 2 | 0,002 | 5000 | 0.00000025208 |
| 3 | 0,003 | 5000 | 0.00000011058 |
| 4 | 0,004 | 5000 | 0.00000006258 |
| 5 | 0,005 | 5000 | 0.00000003906 |
| 6 | 0,006 | 5000 | 0.00000002790 |
| 7 | 0,007 | 5000 | 0.00000002020 |
| 8 | 0,008 | 5000 | 0.00000001524 |
| 9 | 0,009 | 5000 | 0.00000001216 |
| 10 | 0,010 | 5000 | 0.00000000966 |

Dari pengujian terhadap nilai *learning rate*, didapatkan nilai *learning rate* yang menghasilkan nilai MSE terendah adalah 0.01.

Gambar 4.6 menunjukkan grafik penurunan MSE untuk *learning rate* sebesar 0.01, dengan hidden layer sebanyak 100 unit dan ukuran citra 20 x 20 piksel.




**Gambar 4.6** Grafik penurunan MSE *learning rate* =0.01, hidden layer = 100 unit

Pengujian berikutnya terhadap *learning rate* dilakukan dengan jumlah unit hidden layer sebanyak 100 unit dengan ukuran citra 20 x 20 piksel. Pengujian dilakukan dengan menggunakan seluruh huruf yang digunakan dalam pelatihan (sampel 2).

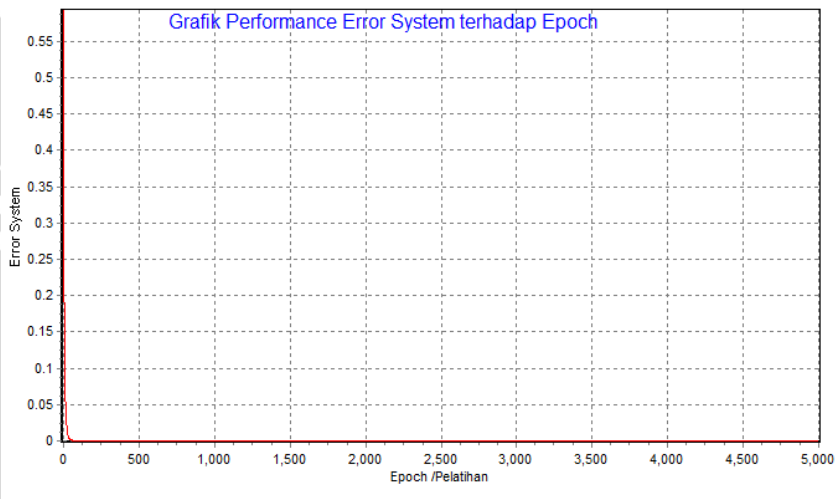
**Tabel 4.5** Pengujian terhadap nilai *learning rate* terhadap seluruh data pelatihan

| **Percobaan ke-** | ***Learning rate*** | ***Epoch*** | **MSE** |
|---|---|---|---|
| 1 | 0,001 | 5000 | 0.00000018116 |
| 2 | 0,002 | 5000 | 0.00000005659 |
| 3 | 0,003 | 5000 | 0.00000003962 |
| 4 | 0,004 | 5000 | 0.00000003323 |
| 5 | 0,005 | 5000 | 0.000769268 |
| 6 | 0,006 | 5000 | 0.00153847 |
| 7 | 0,007 | 5000 | 0.075937 |
| 8 | 0,008 | 5000 | 0.072392 |
| 9 | 0,009 | 5000 | 0.111380 |
| 10 | 0,010 | 5000 | 0.153569 |

Dari pengujian terhadap nilai *learning rate* dengan menggunakan seluruh data pelatihan, didapatkan nilai *learning rate* yang menghasilkan nilai MSE terendah pada percobaan ke-4, yaitu

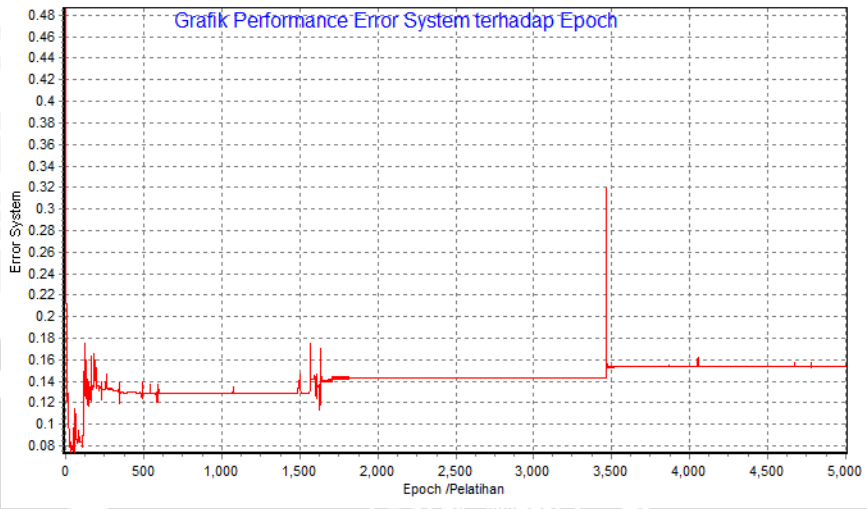
0.004. Hal ini menunjukkan bahwa arsitektur jaringan yang optimal untuk seluruh data pelatihan didapat pada jumlah unit hidden layer sebanyak 100 unit dengan ukuran citra 20 x 20 piksel dan nilai *learning rate* sebesar 0.004.

Gambar 4.7 menunjukkan grafik penurunan MSE untuk *learning rate* sebesar 0.004, dengan hidden layer sebanyak 100 unit dan ukuran citra 20 x 20 piksel untuk pelatihan dengan seluruh data pelatihan.




**Gambar 4.7** Grafik penurunan MSE *learning rate* =0.004, hidden layer = 100 unit

Gambar 4.8 menunjukkan grafik penurunan MSE untuk *learning rate* sebesar 0.01, dengan hidden layer sebanyak 100 unit dan ukuran citra 20 x 20 piksel untuk pelatihan dengan seluruh data pelatihan.




**Gambar 4.8** Grafik penurunan MSE *learning rate* =0.01, hidden layer = 100 unit

### 4.4.5 Pengujian terhadap pengenalan huruf tunggal

Setelah mendapatkan arsitektur jaringan yang optimal, maka dilakukan pelatihan terhadap data. Data yang digunakan adalah huruf yang memiliki prosentase keberhasilan segmentasi sebesar 100% sebagaimana dapat dilihat pada Tabel 4.1. Proses pelatihan dilakukan terhadap 7 jenis tulisan yang berbeda untuk setiap jenis huruf.

Setelah pelatihan selesai dilakukan, pengujian dilakukan terhadap citra yang berisi huruf tunggal (tidak tergabung dalam kata) yang sudah pernah dilakukan pembelajaran dan yang belum pernah dilakukan pembelajaran sebelumnya. Bobot yang dipakai untuk pengujian adalah bobot dari hasil pembelajaran dengan jumlah unit pada *hidden layer* sebesar 100 unit, nilai *learning rate* sebesar 0.004. Hasil dari pengujian terhadap huruf tunggal yang pernah dilakukan pembelajaran ditunjukkan pada Tabel 4.6 dan hasil dari pengujian terhadap huruf tunggal yang belum dilakukan pembelajaran ditunjukkan pada Tabel 4.7.

**Tabel 4.6** Hasil pengujian terhadap citra yang berisi huruf tunggal yang pernah dilakukan pembelajaran

| Percobaan | Benar | Salah | Keakuratan (%) |
|---|---|---|---|
| 1 | 13 | 0 | 100 |
| 2 | 13 | 0 | 100 |
| 3 | 13 | 0 | 100 |
| 4 | 13 | 0 | 100 |
| 5 | 12 | 1 | 92.3 |
| 6 | 12 | 1 | 92.3 |
| 7 | 12 | 1 | 92.3 |
| Rata-rata | | | 96.70 |

**Tabel 4.7** Hasil pengujian terhadap citra yang berisi huruf tunggal yang belum dilakukan pembelajaran

| Percobaan | Benar | Salah | Keakuratan (%) |
|---|---|---|---|
| 1 | 8 | 5 | 61.53 |
| 2 | 6 | 7 | 46.15 |
| 3 | 8 | 5 | 61.53 |
| Rata-rata | | | 56.41 |

Dari hasil pengujian terhadap citra yang berisi huruf tunggal yang sudah pernah dilakukan pembelajaran, sistem mampu mengenali dengan benar data yang diujikan dengan tingkat keakuratan sebesar 96.70%. Sedangkan hasil dari pengujian terhadap citra yang berisi huruf tunggal yang belum pernah dilakukan pembelajaran menghasilkan nilai rata-rata keakuratan sebesar 56.41%.

### 4.4.6 Proses pengujian terhadap sistem keseluruhan

Proses pengujian terhadap sistem keseluruhan dilakukan terhadap 22 kata yang terbentuk dari sampel 2. Masing-masing kata ditulis sebanyak 5 kali. Hasil dari pengujian terhadap sistem secara keseluruhan ditunjukkan pada Tabel 4.8.

**Tabel 4.8** Hasil pengujian terhadap sistem keseluruhan

| No | Kata | 1 | | 2 | | 3 | | 4 | | 5 | | Berhasil (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | dikenali | jml benar | dikenali | jml benar | dikenali | jml benar | dikenali | jml benar | dikenali | jml benar | |
| 1 | bagal | batat | 3 | fasal | 3 | laqaf | 2 | logol | 2 | laqal | 3 | 52 |
| 2 | bossa | fotsa | 3 | lossa | 4 | fossa | 4 | fogga | 2 | lossa | 4 | 68 |
| 3 | cegat | cesat | 4 | cesat | 4 | eeqat | 3 | cegal | 4 | ccgal | 3 | 72 |
| 4 | jegal | teqat | 2 | jegae | 4 | jegaf | 4 | jeqct | 2 | jegal | 5 | 68 |
| 5 | joglo | jcslo | 3 | jogfo | 4 | joglo | 5 | logfo | 3 | joglo | 5 | 80 |
| 6 | logat | logat | 5 | logat | 5 | fogat | 4 | foqcl | 1 | logal | 4 | 76 |
| 7 | selat | selat | 5 | selat | 5 | selat | 5 | setal | 3 | selat | 5 | 92 |
| 8 | tossa | tecsa | 3 | tossa | 5 | lossa | 4 | lossa | 4 | lossa | 4 | 80 |
| 9 | total | tatel | 3 | tetal | 4 | tofsl | 3 | lofol | 2 | tofal | 4 | 64 |
| 10 | zotaq | zotas | 4 | zotaq | 5 | zolbq | 3 | ooloq | 2 | zolbq | 3 | 68 |
| 11 | alfa | alfa | 4 | alfa | 4 | alfa | 4 | ollo | 1 | alfa | 4 | 85 |
| 12 | algo | algo | 4 | algo | 4 | algo | 4 | atqo | 2 | algo | 4 | 90 |
| 13 | azab | aacl | 1 | azab | 4 | azal | 3 | ooal | 1 | azal | 3 | 60 |
| 14 | baca | bbea | 2 | faea | 2 | laca | 3 | laco | 2 | faca | 3 | 60 |
| 15 | bola | bola | 4 | fola | 3 | lola | 3 | folo | 2 | lola | 3 | 75 |
| 16 | calo | colc | 2 | calo | 4 | calo | 4 | cofo | 2 | eaqo | 2 | 70 |
| 17 | cola | eala | 2 | cota | 3 | cola | 4 | sota | 2 | cola | 4 | 75 |
| 18 | laba | lolo | 1 | lala | 3 | laba | 4 | jact | 1 | lasa | 3 | 60 |
| 19 | logo | lcqo | 2 | taqo | 1 | loqo | 3 | fego | 2 | loqo | 3 | 55 |
| 20 | qola | qolo | 3 | qola | 4 | qcla | 3 | qoto | 2 | qola | 4 | 80 |
| 21 | tega | tssa | 2 | tega | 4 | lega | 3 | fesa | 2 | tega | 4 | 75 |
| 22 | zola | zsla | 3 | zola | 4 | zola | 4 | zota | 3 | zola | 4 | 90 |

Dari pengujian terhadap sistem secara keseluruhan, didapat nilai rata-rata akurasi keberhasilan pengenalan karakter sebesar 72.45%.

## 4.5 Analisis hasil pengujian

### 4.5.1 Analisis pengujian terhadap segmentasi karakter

Berdasarkan prosentase keberhasilan dari segmentasi karakter yang ditunjukkan pada Tabel 4.1, tingkat keberhasilan secara keseluruhan dari kata-kata yang diujikan mencapai 78%. Dengan demikian, tidak semua huruf dapat disegmentasi dengan benar. Hal ini disebabkan oleh gaya/model penulisan masing-masing orang yang berbeda-beda, dimana tulisan tersebut ada berhasil disegmentasi dengan benar dan ada yang tidak. Selain itu, gaya penulisan huruf juga berpengaruh terhadap lokasi titik potong/segmentasi terhadap sebuah karakter. Hal ini nantinya akan berpengaruh terhadap proses pengenalan.

**Gambar 4.9** Gaya penulisan huruf "h" yang dapat disegmentasi dengan benar

**Gambar 4.10** Gaya penulisan huruf "h" yang tidak dapat disegmentasi dengan benar

Gambar 4.9 menunjukkan gaya penulisan huruf "h" yang dapat disegmentasi dengan benar. Gambar 4.10 menunjukkan gaya penulisan huruf "h" yang tidak dapat disegmentasi dengan benar. Karakter "h" tersebut tersegmentasi menjadi dua bagian (*oversegmented*) karena jarak piksel teratas dan terbawah pada tengah-tengah karakter kurang dari *threshold* yang ditetapkan.

Selain itu, bentuk/morfologi dari huruf berpengaruh terhadap keberhasilan proses segmentasi karakter. Huruf-huruf yang memiliki bentuk terbuka ke atas atau ke bawah, seperti huruf m,n,v, dan w tidak berhasil disegmentasi dengan benar. Gambar 4.11 menunjukkan contoh karakter huruf m yang tidak berhasil disegmentasi dengan benar dan terbagi menjadi tiga bagian.

**Gambar 4.11** Karakter huruf "m" yang tidak berhasil disegmentasi dengan benar

### 4.5.2 Analisis pengujian terhadap ukuran citra

Berdasarkan hasil pengujian terhadap ukuran citra yang ditunjukkan pada Tabel 4.2, diketahui bahwa tiap-tiap ukuran yang diujicobakan dengan ukuran *hidden layer* dan nilai *learning rate* yang diberikan, menghasilkan MSE yang berbeda. Ukuran citra sebagai ukuran *input layer* dan *output layer* ini nantinya akan berpengaruh terhadap jumlah iterasi (*computation time*/*computation cost*) yang dibutuhkan. Dengan demikian, penentuan ukuran citra sebagai ukuran *input layer* dan *output layer* yang tepat akan berpengaruh terhadap kecepatan untuk mencapai target *error* yang telah ditetapkan.

### 4.5.3 Analisis pengujian terhadap jumlah unit hidden layer

Berdasarkan hasil pengujian terhadap jumlah unit hidden layer pada arsitektur yang terbentuk yang ditunjukkan pada Tabel 4.3, dapat diketahui bahwa pada jumlah hidden layer yang semakin besar, maka nilai MSE yang dihasilkan semakin kecil. Dari grafik yang ditunjukkan pada Gambar 4.5, pada jumlah hidden layer 100, proses pelatihan mencapai konvergensi pada jumlah epoch yang kecil. Dengan demikian, dari jumlah unit hidden layer yang diujikan, dapat disimpulkan bahwa semakin besar jumlah hidden layer maka semakin semakin cepat pula MSE mencapai target *error* yang diharapkan

### 4.5.4 Analisis pengujian terhadap nilai *learning rate*

Berdasarkan hasil dari pengujian terhadap nilai *learning rate* pada data tunggal dengan arsitektur yang telah terbentuk yang ditunjukkan pada Tabel 4.4, diketahui bahwa semakin besar nilai *learning rate*, maka nilai MSE semakin kecil. Dari gambar 4.6 diketahui bahwa konvergensi tercapai pada *epoch* yang kecil. Dengan demikian, maka semakin besar nilai *learning rate*, maka semakin cepat pula MSE mencapai target *error* yang diharapkan. Akan tetapi, pada pelatihan dengan jumlah data yang cukup banyak, hal ini tidak berlaku. Hasil pengujian menunjukkan bahwa justru pada pengujian dengan jumlah data yang cukup banyak, justru pelatihan tidak bisa mencapai nilai MSE yang diharapkan. Dengan nilai *learning rate* yang besar (lebih dari 0.005) yang diharapkan pelatihan berlangsung lebih cepat, proses pelatihan justru tidak bisa mencapai konvergensi. Faktor yang berpengaruh dalam hal ini adalah pemilihan nilai bobot awal. Nilai bobot awal yang kurang bervariasi dan nilai acaknya terlalu kecil menyebabkan konvergensi tidak akan tercapaicapai. Oleh karena itu, pemilihan nilai bobot awal dan nilai *learning rate* yang tepat akan sangat berpengaruh terhadap keberhasilan proses pelatihan.

### 4.5.5 Analisis pengujian terhadap pengenalan huruf tunggal

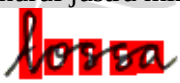
Berdasarkan hasil dari proses pengenalan terhadap karakter yang sudah dilatihkan sebelumnya pada Tabel 4.6, ternyata tingkat keberhasilan rata-ratanya tidak mencapai 100 %, yaitu hanya mencapai 96,70 %. Begitu pula hasil dari pengenalan terhadap karakter yang belum dilatihkan yang ditunjukkan pada Tabel 4.7, tingkat keberhasilannya hanya mencapai 56,41 %. Hal ini

dimungkinkan karena adanya faktor kemiripan bentuk antar huruf karena gaya penulisan yang mirip antara satu huruf dengan yang lainnya, sehingga proses pengenalan menjadi salah. Gambar 4.12 menunjukkan contoh huruf "t" yang dikenali sebagai huruf "f" karena gaya penulisan yang kurang benar.

**Gambar 4.12** Penulisan huruf "t" yang kurang benar

### 4.5.6 Analisis pengujian terhadap sistem keseluruhan

Proses ini meliputi proses segmentasi karakter, proses pelatihan, dan proses pengenalan. Hasil dari pengujian terhadap sistem secara keseluruhan ditunjukkan pada Tabel 4.8. Dari Tabel 4.8 tersebut dapat diketahui bahwa tingkat akurasi keberhasilan pengenalan karakter sebesar 72.45%. Dari Tabel 4.8, dapat diketahui bahwa proses segmentasi karakter berhasil untuk mensegmentasi citra tulisan menjadi karakter. Akan tetapi, setelah melalui proses pengenalan, ternyata gagal untuk dikenali dengan baik. Hal ini dikarenakan gaya penulisan huruf yang mirip antara huruf yang satu dengan huruf yang lainnya. Selain itu, gaya penulisan juga berpengaruh terhadap lokasi titik potong/segmentasi huruf, sehingga setelah disegmentasi suatu huruf justru mirip dengan huruf yang lain.



**Gambar 4.13** Kata "tossa" pada dataset ke-4

**Gambar 4.14** Kata "tossa" pada dataset ke-2

Pada Gambar 4.13, kata "tossa" berhasil disegmentasi dengan benar. Akan tetapi gaya penulisan karakter "t" pada kata "tega" tersebut memiliki kemiripan dengan huruf "l", sehingga justru dikenali sistem sebagai huruf "l". Gambar 4.14 menunjukkan kata "tossa" berhasil disegmentasi dengan baik. Karakter "t" pada kata tersebut juga berhasil dikenali sistem dengan baik. Dengan demikian, faktor gaya penulisan sangat berpengaruh terhadap keberhasilan pengenalan tulisan tangan.

# BAB V
# PENUTUP

## 5.1 Kesimpulan

Dari pengujian yang telah dilakukan, dapat disimpulkan menjadi beberapa hal antara lain:

1. Dari hasil pelatihan, terbentuk struktur jaringan syaraf tiruan terbaik yang terdiri dari 400 unit neuron *input layer*, 100 unit neuron *hidden layer*, 400 unit neuron *output layer*, nilai *learning rate* sebesar 0,004 dengan *max epoch* sebesar 5000.
2. Tingkat keakuratan pengenalan tulisan tangan sebesar 96,70% untuk citra yang berisi huruf tunggal yang pernah dilakukan pembelajaran dan keakuratan hasil pengenalan terhadap citra yang berisi huruf tunggal yang belum dilakukan pembelajaran sebesar 56,41%. Sedangkan tingkat keakuratan sistem secara keseluruhan terhadap kata yang diujikan sebesar 72,45%.

## 5.2 Saran

Ada beberapa hal yang dapat digunakan sebagai saran untuk proses pengembangan sistem selanjutnya, yaitu:

1. Proses segmentasi karakter dapat disempurnakan sehingga dapat melakukan segmentasi karakter pada seluruh karakter huruf dengan baik.
2. Inisialisasi bobot awal perlu diperbaiki dan perlu dicoba untuk menggunakan koefisien momentum untuk mendapatkan konvergensi yang lebih cepat selama proses pelatihan.

## DAFTAR PUSTAKA


Akbar, Ali. 2007. *Jaringan Syaraf Tiruan Untuk Menilai Aransemen Musik*. Laporan tugas akhir. Program Studi Informatika, Sekolah Teknik Elektro Dan Informatika, Institut Teknologi Bandung. Bandung

Arifin, Muh Syaiful. 2008. *Pengenalan Tulisan Tangan Menggunakan Algoritma Jaringan Syaraf Tiruan Propagasi Balik*. Laporan Tugas Akhir. Program Studi Ilmu Komputer, Universitas Brawijaya. Malang.

Budhi, G.S, Gunawan, I, Jaowry S,. 2006. *Metode Jaringan Syaraf Tiruan Backpropagation untuk Pengenalan Huruf Cetak pada Citra Digital*

Cheng, Chun Ki, Michael Blumenstein. 2005. *The Neural-based Segmentation of Cursive Words using Enhanced Heuristics*. icdar, pp.650-654. Eighth International Conference on Document Analysis and Recognition (ICDAR'05).

Guillevic,D. 1995. *Unconstrained handwriting recognition applied to the recognition of bank cheques*. Concordia University, Montreal. Canada

Hassin, Abbas h.,Huang J Hua, dan Tang Xiang Long. 2003. *Offline Arabic Character Recognition System*. Journal of Harbin Institute of Technology Vol 10, No 1.

Kusumadewi, Sri. 2003. *Artificial Intelligent teknik dan aplikasinya.* Penerbit Graha Ilmu. Yogyakarta.

Kusumo Putro, E. philipus, dan Rahmat Widyanto.1999. *Pengenalan huruf tulisan tangan menggunakan logika fuzzy dan jaringan syaraf tiruan*. Seminar on air- PPI tokyo institute of technology No. 1.

Lecolinet, E dan O. Baret. 1994. *Fundamentals in Handwriting Recognition*. S. Impedovo Ed., pages 235-263, NATO ASI Series F: Computer and Systems Sciences, Vol. 124, Springer Verlag.

Munawir. 2008. *Pattern recognition pada pengenalan tulisan*. Politeknik Negeri Lhokseumawe

Munir, Rinaldi. 2004. *Pengolahan Citra Digital dengan Pendekatan Algoritmik*. Informatika, Bandung.

Otair, Mohammed, dan Walid Salameh. 2005. *Speeding Up Back-Propagation Neural Networks*. Proceeding of Informing Science and Information Technology Education Joint Conference.

Patterson, D.W. 1996. *Artificial neural network theory and application*. Prentice Hall. Singapore.

Purnomo, Mauridhi Heri, dan Agus Kurniawan. 2006. *Supervised Neural Networks dan aplikasinya*. Penerbit Andi, Yogyakarta.

Rodrigues, Roberto dan gay Thome. 2000. *Cursive character recognition – a character segmentation method using projection profile histogram*. International Conference on Information Systems, analysis andSynthesis on Image and Multidimention Signal Processing

Salameh, Walid dan Mohammed A. Otair. 2005. *Online Handwritten Character Recognition Using an Optical Backpropagation Neural Network*. Proceeding of Informing Science and Information Technology Education Joint Conference.

Sediyono, Eko. 2000. *Metoda Jaringan Saraf Tiruan dengan Arsitektur Jaringan Radial Basis Kooperatif dan Kompetitif untuk Pengenalan Tulisan Tangan*. Fakultas Teknik Jurusan Teknik Elektro, Univ. Kristen Satya Wacana Salatiga. Jakarta.

Siang, J.J. 2005. *Jaringan Syaraf Tiruan dan Pemrogramannya menggunakan MATLAB*. Penerbit Andi, Yogyakarta.

Zhang, Y.Y. dan Wang, P.S.P. 1994. *Design of Parallel Thinning Algorithm*. Proceding 3rd Int. Workshop on Parallel Image Analysis

**Lampiran 1**

Tabel hasil pengujian terhadap sampel karakter dengan nilai learning rate 0.001 dan jumlah unit hidden layer 20 pada ukuran citra 10x10 piksel

| Huruf | Ukuran citra (pixel) | *Epoch* | MSE |
|---|---|---|---|
| a | 10 * 10 | 5000 | 0.000023700 |
| l | 10 * 10 | 5000 | 0.000023517 |
| b | 10 * 10 | 5000 | 0.000023583 |
| q | 10 * 10 | 5000 | 0.000023564 |
| e | 10 * 10 | 5000 | 0.000023568 |
| Rata-rata | | | 0.00002358 |

Tabel hasil pengujian terhadap sampel karakter dengan, nilai learning rate 0.001 dan jumlah unit hidden layer 20 pada ukuran citra 15x15 piksel

| Huruf | Ukuran citra (pixel) | *Epoch* | MSE |
|---|---|---|---|
| a | 15 * 15 | 5000 | 0.0000235449 |
| l | 15 * 15 | 5000 | 0.0000236182 |
| b | 15 * 15 | 5000 | 0.0000235709 |
| q | 15 * 15 | 5000 | 0.0000235484 |
| e | 15 * 15 | 5000 | 0.0000236915 |
| Rata-rata | | | 0.000023595 |

Tabel hasil pengujian terhadap sampel karakter dengan nilai learning rate 0.001 dan jumlah unit hidden layer 20 pada ukuran citra 20x20 piksel

| Huruf | Ukuran citra (pixel) | *Epoch* | MSE |
|---|---|---|---|
| a | 20 * 20 | 5000 | 0.000023572 |
| l | 20 * 20 | 5000 | 0.000023598 |
| b | 20 * 20 | 5000 | 0.000023466 |
| q | 20 * 20 | 5000 | 0.000023464 |
| e | 20 * 20 | 5000 | 0.000023538 |
| Rata-rata | | | 0.000023527 |

Tabel hasil pengujian terhadap sampel karakter dengan nilai learning rate 0.001 dan jumlah unit hidden layer 20 pada ukuran citra 25x25 piksel

| **Huruf** | **Ukuran citra (pixel)** | ***Epoch*** | **MSE** |
|---|---|---|---|
| a | 25 * 25 | 5000 | 0.0000235795 |
| l | 25 * 25 | 5000 | 0.0000234749 |
| b | 25 * 25 | 5000 | 0.0000235584 |
| q | 25 * 25 | 5000 | 0.0000235983 |
| e | 25 * 25 | 5000 | 0.0000235068 |
| Rata-rata | | | 0.0000235436 |

**Lampiran 2**

Tabel hasil pengujian terhadap jumlah unit hidden layer dengan ukuran citra 20 x 20 piksel dan learning rate 0.001 pada karakter huruf "a"

| Percobaan ke- | Jumlah unit *Hidden Layer* | *Epoch* | MSE |
|---|---|---|---|
| 1 | 20 | 5000 | 0.000023521 |
| 2 | 40 | 5000 | 0.000006162 |
| 3 | 60 | 5000 | 0.000002684 |
| 4 | 80 | 5000 | 0.000001576 |
| 5 | 100 | 5000 | 0.000000992 |

Tabel hasil pengujian terhadap jumlah unit hidden layer dengan ukuran citra 20x20 piksel dan learning rate 0.001 pada karakter huruf "l"

| Percobaan ke- | Jumlah unit *Hidden Layer* | *Epoch* | MSE |
|---|---|---|---|
| 1 | 20 | 5000 | 0.000023484 |
| 2 | 40 | 5000 | 0.000006022 |
| 3 | 60 | 5000 | 0.000002772 |
| 4 | 80 | 5000 | 0.000001511 |
| 5 | 100 | 5000 | 0.000000980 |

Tabel hasil pengujian terhadap jumlah unit hidden layer dengan ukuran citra 20x20 piksel dan learning rate 0.001 pada karakter huruf "b"

| Percobaan ke- | Jumlah unit *Hidden Layer* | *Epoch* | MSE |
|---|---|---|---|
| 1 | 20 | 5000 | 0.000023480 |
| 2 | 40 | 5000 | 0.000006080 |
| 3 | 60 | 5000 | 0.000002700 |
| 4 | 80 | 5000 | 0.000001533 |
| 5 | 100 | 5000 | 0.000000964 |

Tabel hasil pengujian terhadap jumlah unit hidden layer dengan ukuran citra 20x20 piksel dan learning rate 0.001 pada karakter huruf "b"

| **Percobaan ke-** | **Jumlah unit *Hidden Layer*** | ***Epoch*** | **MSE** |
|---|---|---|---|
| 1 | 20 | 5000 | 0.000023780 |
| 2 | 40 | 5000 | 0.000006240 |
| 3 | 60 | 5000 | 0.000002775 |
| 4 | 80 | 5000 | 0.000001533 |
| 5 | 100 | 5000 | 0.000001023 |

Tabel hasil pengujian terhadap jumlah unit hidden layer dengan ukuran citra 20x20 piksel dan learning rate 0.001 pada karakter huruf "e"

| **Percobaan ke-** | **Jumlah unit *Hidden Layer*** | ***Epoch*** | **MSE** |
|---|---|---|---|
| 1 | 20 | 5000 | 0.000023890 |
| 2 | 40 | 5000 | 0.000006011 |
| 3 | 60 | 5000 | 0.000002853 |
| 4 | 80 | 5000 | 0.000001521 |
| 5 | 100 | 5000 | 0.000000976 |

**Lampiran 3**

Tabel hasil pengujian terhadap nilai learning rate dengan ukuran citra 20x20 piksel dan jumlah unit hidden neuron 100 pada karakter huruf "a"

| Percobaan ke- | *Learning rate* | *Epoch* | MSE |
|---|---|---|---|
| 1 | 0,001 | 5000 | 0.000000999 |
| 2 | 0,002 | 5000 | 0.000000251 |
| 3 | 0,003 | 5000 | 0.000000109 |
| 4 | 0,004 | 5000 | 0.000000064 |
| 5 | 0,005 | 5000 | 0.000000039 |
| 6 | 0,006 | 5000 | 0.000000028 |
| 7 | 0,007 | 5000 | 0.000000020 |
| 8 | 0,008 | 5000 | 0.000000015 |
| 9 | 0,009 | 5000 | 0.000000012 |
| 10 | 0,010 | 5000 | 0.000000009 |

Tabel hasil pengujian terhadap nilai learning rate dengan ukuran citra 20x20 piksel dan jumlah unit hidden neuron 100 pada karakter huruf "l"

| Percobaan ke- | *Learning rate* | *Epoch* | MSE |
|---|---|---|---|
| 1 | 0,001 | 5000 | 0.0000010120 |
| 2 | 0,002 | 5000 | 0.0000002484 |
| 3 | 0,003 | 5000 | 0.0000001122 |
| 4 | 0,004 | 5000 | 0.0000000625 |
| 5 | 0,005 | 5000 | 0.0000000385 |
| 6 | 0,006 | 5000 | 0.0000000285 |
| 7 | 0,007 | 5000 | 0.0000000206 |
| 8 | 0,008 | 5000 | 0.0000000151 |
| 9 | 0,009 | 5000 | 0.0000000127 |
| 10 | 0,010 | 5000 | 0.0000000098 |

Tabel hasil pengujian terhadap nilai learning rate dengan ukuran citra 20x20 piksel dan jumlah unit hidden neuron 100 pada karakter huruf "b"

| **Percobaan ke-** | ***Learning rate*** | ***Epoch*** | **MSE** |
|---|---|---|---|
| 1 | 0,001 | 5000 | 0.0000009996 |
| 2 | 0,002 | 5000 | 0.0000002640 |
| 3 | 0,003 | 5000 | 0.0000001089 |
| 4 | 0,004 | 5000 | 0.0000000611 |
| 5 | 0,005 | 5000 | 0.0000000383 |
| 6 | 0,006 | 5000 | 0.0000000273 |
| 7 | 0,007 | 5000 | 0.0000000206 |
| 8 | 0,008 | 5000 | 0.0000000152 |
| 9 | 0,009 | 5000 | 0.0000000120 |
| 10 | 0,010 | 5000 | 0.0000000099 |

Tabel hasil pengujian terhadap nilai learning rate dengan ukuran citra 20x20 piksel dan jumlah unit hidden neuron 100 pada karakter huruf "q"

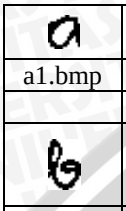
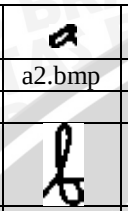
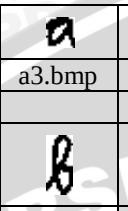
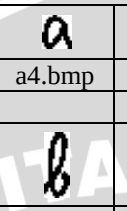
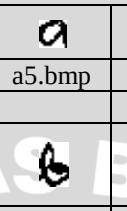
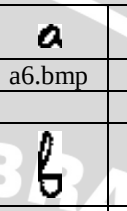
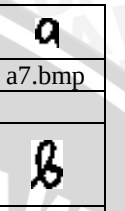
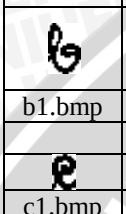
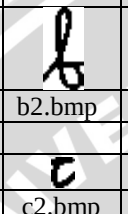
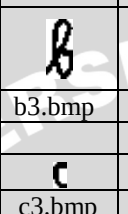
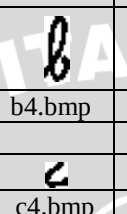
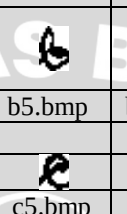
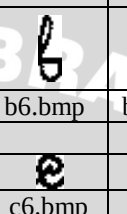
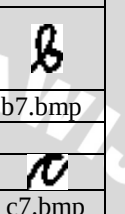
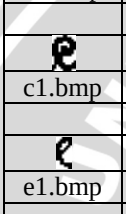
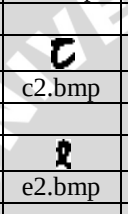
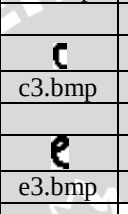
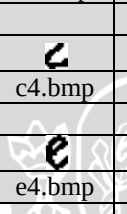
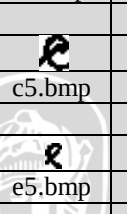
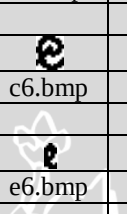
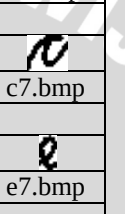
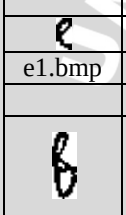
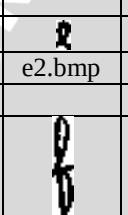
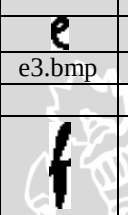
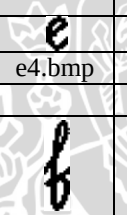
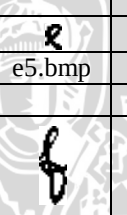
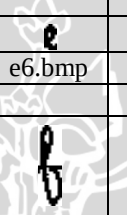
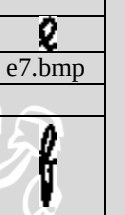
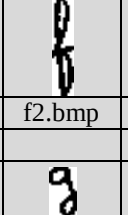
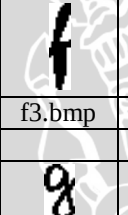
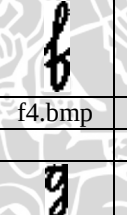
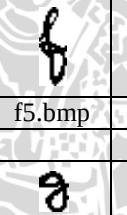
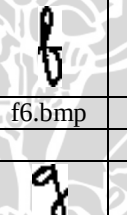
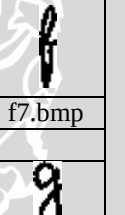
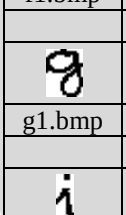
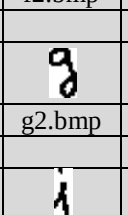
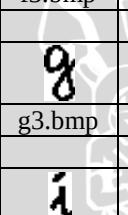
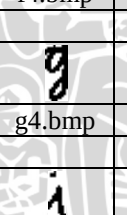
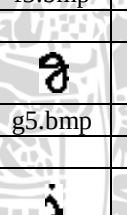
| **Percobaan ke-** | ***Learning rate*** | ***Epoch*** | **MSE** |
|---|---|---|---|
| 1 | 0,001 | 5000 | 0.0000009864 |
| 2 | 0,002 | 5000 | 0.0000002496 |
| 3 | 0,003 | 5000 | 0.0000001105 |
| 4 | 0,004 | 5000 | 0.0000000619 |
| 5 | 0,005 | 5000 | 0.0000000404 |
| 6 | 0,006 | 5000 | 0.0000000283 |
| 7 | 0,007 | 5000 | 0.0000000199 |
| 8 | 0,008 | 5000 | 0.0000000156 |
| 9 | 0,009 | 5000 | 0.0000000120 |
| 10 | 0,010 | 5000 | 0.0000000097 |

Tabel hasil pengujian terhadap nilai learning rate dengan ukuran citra 20x20 piksel dan jumlah unit hidden neuron 100 pada karakter huruf "e"

| Percobaan ke- | *Learning rate* | *Epoch* | MSE |
|---|---|---|---|
| 1 | 0,001 | 5000 | 0.0000011265 |
| 2 | 0,002 | 5000 | 0.0000002474 |
| 3 | 0,003 | 5000 | 0.0000001123 |
| 4 | 0,004 | 5000 | 0.0000000634 |
| 5 | 0,005 | 5000 | 0.0000000391 |
| 6 | 0,006 | 5000 | 0.0000000274 |
| 7 | 0,007 | 5000 | 0.0000000199 |
| 8 | 0,008 | 5000 | 0.0000000153 |
| 9 | 0,009 | 5000 | 0.0000000121 |
| 10 | 0,010 | 5000 | 0.0000000099 |

**Lampiran 4**

Karakter huruf yang digunakan dalam pelatihan

| | | | | | | |
|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |
| a1.bmp | a2.bmp | a3.bmp | a4.bmp | a5.bmp | a6.bmp | a7.bmp |
|  |  |  |  |  |  |  |
| b1.bmp | b2.bmp | b3.bmp | b4.bmp | b5.bmp | b6.bmp | b7.bmp |
|  |  |  |  |  |  |  |
| c1.bmp | c2.bmp | c3.bmp | c4.bmp | c5.bmp | c6.bmp | c7.bmp |
|  |  |  |  |  |  |  |
| e1.bmp | e2.bmp | e3.bmp | e4.bmp | e5.bmp | e6.bmp | e7.bmp |
|  |  |  |  |  |  |  |
| f1.bmp | f2.bmp | f3.bmp | f4.bmp | f5.bmp | f6.bmp | f7.bmp |
|  |  |  |  |  |  |  |
| g1.bmp | g2.bmp | g3.bmp | g4.bmp | g5.bmp | g6.bmp | g7.bmp |
|  |  |  |  |  |  |  |
| j1.bmp | j2.bmp | j3.bmp | j4.bmp | j5.bmp | j6.bmp | j7.bmp |
|  |  |  |  |  |  |  |
| l1.bmp | l2.bmp | l3.bmp | l4.bmp | l5.bmp | l6.bmp | l7.bmp |
|  |  |  |  |  |  |  |
| o1.bmp | o2.bmp | o3.bmp | o4.bmp | o5.bmp | o6.bmp | o7.bmp |
|  |  |  |  |  |  |  |

| q1.bmp | q2.bmp | q3.bmp | q4.bmp | q5.bmp | q6.bmp | q7.bmp |
|---|---|---|---|---|---|---|
| | | | | | | |
|  |  |  |  |  |  |  |
| s1.bmp | s2.bmp | s3.bmp | s4.bmp | s5.bmp | s6.bmp | s7.bmp |
| | | | | | | |
|  |  |  |  |  |  |  |
| t1.bmp | t2.bmp | t3.bmp | t4.bmp | t5.bmp | t6.bmp | t7.bmp |
| | | | | | | |
|  |  |  |  |  |  |  |
| z1.bmp | z2.bmp | z3.bmp | z4.bmp | z5.bmp | z6.bmp | z7.bmp |

Daftar karakter huruf yang digunakan dalam pengujian

| a8.bmp | a9.bmp | a10.bmp | b8.bmp | b9.bmp | b10.bmp |
|---|---|---|---|---|---|
| | | | | | |
|  |  |  |  |  |  |
| c8.bmp | c9.bmp | c10.bmp | e8.bmp | e9.bmp | e10.bmp |
| | | | | | |
|  |  |  |  |  |  |
| f8.bmp | f9.bmp | f10.bmp | g8.bmp | g9.bmp | g10.bmp |
| | | | | | |
|  |  |  |  |  |  |
| j8.bmp | j9.bmp | j10.bmp | l8.bmp | l9.bmp | l10.bmp |
| | | | | | |
|  |  |  |  |  |  |
| o8.bmp | o9.bmp | o10.bmp | q8.bmp | q9.bmp | q10.bmp |
| | | | | | |

| | | | | | |
|---|---|---|---|---|---|
|  |  |  |  |  |  |
| s8.bmp | s9.bmp | s10.bmp | t8.bmp | t9.bmp | t10.bmp |
| | | | | | |
|  |  |  | | | |
| z8.bmp | z9.bmp | z10.bmp | | | |

**Lampiran 5**

Tabel hasil uji segmentasi

| No | Kata | Jumlah huruf | data ke-1 | data ke-2 | data ke-3 | data ke-4 | data ke-5 | data ke-6 | data ke-7 | data ke-8 | data ke-9 | data ke-10 | % berhasil |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | albuquerque | 11 | albqeqe | albqeqe | albqeqe | albqeqe | albqeqe | albqeqe | albqeqe | albqerqe | albqeqe | albuqeqe | 65.45 |
| 2 | alhamdulillah | 13 | alalilla | alamdlillah | aladllla | aladlilla | aladlilla | aladlilla | aladlilla | alhamdlilla | alhamdlilla | alhadlilla | 73.08 |
| 3 | artificial | 10 | atificial | atificial | atificial | atificial | atificial | atificial | atificial | atificial | atificial | atificial | 90.00 |
| 4 | bertanggungjawab | 16 | bertagggjaab | betagggjaab | bertagggjaab | betagggjaab | betagggjaab | betagggjaab | betagggjaab | betagggjaab | betagggjaab | bertagggjaab | 70.63 |
| 5 | development | 11 | deelopment | developmet | deeloet | deeloet | deelopet | deelopet | deelopet | deelopet | development | developet | 78.18 |
| 6 | jejaring | 8 | jejaig | jejaig | jejaig | jejarig | jejaig | jejaig | jejaig | jejaig | jejaing | jejaing | 77.50 |
| 7 | ketaqwaan | 9 | ketaqaa | ketaqaa | ketaqaa | ketaqaa | ketaqaa | ketaqaa | ketaqaa | ketaqaa | ketaqaa | ketaqaa | 77.78 |
| 8 | localhost | 9 | localost | localost | localost | localhost | localost | localost | localost | localhost | localhost | localost | 92.22 |
| 9 | mandrake | 8 | adake | andae | adake | adake | adake | adake | adake | adake | adake | mandake | 65.00 |
| 10 | officially | 10 | officially | officially | officiall | officially | officiall | officiall | officially | officially | officiall | officially | 96.00 |
| 11 | picokavazi | 10 | picokavazi | picokaazi | icokavazi | picokaazi | picokavazi | picokavazi | picokavazi | picokaazi | picokaazi | picokavazi | 95.00 |
| 12 | protozoa | 8 | protozoa | protozoa | otozoa | potozoa | potozoa | potozoa | potozoa | potozoa | potozoa | potozoa | 88.75 |
| 13 | swallow | 7 | sallo | sallo | sallo | sallo | sallo | sallo | sallo | sallo | sallo | sallo | 71.43 |
| 14 | vivipar | 7 | ivipar | ivipa | iia | viia | vivipa | viipa | viipa | iipa | iipa | vivipar | 70.00 |
| 15 | xenopus | 7 | xeops | xeops | eops | xeos | xeops | xeops | xeops | xeops | xeops | xenops | 70.00 |
| 16 | zyrex | 5 | zyex | zyex | zyex | zyex | zex | zex | zex | zyex | zyex | zex | 72.00 |