STUDI PERBANDINGAN PENGENALAN KARAKTER AKSARA LAMPUNG DENGAN METODE DETEKSI TEPI ROBERTS DAN SOBEL

(Skripsi)

Oleh

HALIM ABDILLAH SHOLEH

FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2018

ABSTRACT

A COMPARATIVE STUDY LAMPUNG SCRIPT CHARACTER RECOGNITION BASED EDGE DETECTION METHOD OF ROBERTS AND SOBEL

By

HALIM ABDILLAH SHOLEH

The culture of Lampung script is now rarely used and will get extinct. In order to preserve the culture, this digital research about Lampung script with image processing method and pattern recognition is conducted. This research is divided into two stages, training and testing. Training stage conducted image processing and training the artificial neural network (ANN) backpropagation. One past of image processing is edge detection. The edge detection method is used to analyze the Lampung script character to find the line pattern of the letters. The image processing methods used in image edge detection are Roberts edge detection method and Sobel edge detection method. The result of image processing will get into testing stage. Sobel method has better performance on recognizing. Lampung script character, in which the result from testing Lampung script character gives 28.5 % of error for Roberts detection method and 14.5 % of error for Sobel detection method.

Keywords **:** *ANN Backpropagation, Edge Detection, Error Value, Image Processing, Lampung character, Roberts, Sobel.*

ABSTRAK

STUDI PERBANDINGAN PENGENALAN KARAKTER AKSARA LAMPUNG DENGAN METODE DETEKSI TEPI ROBERTS DAN SOBEL

Oleh

HALIM ABDILLAH SHOLEH

Budaya aksara Lampung pada saat ini sudah jarang digunakan sehingga terancam punah. Untuk melestarikan budaya tersebut, maka dilakukan penelitian mengenai huruf aksara Lampung secara digital dengan metode pengolahan citra dan pengenalan pola. Pada penelitian ini, terbagi menjadi dua tahap yaitu tahap pelatihan dan tahap pengujian. Pada tahap pelatihan dilakukan pengolahan citra dan pelatihan jaringan saraf tiruan (JST) backpropagation. Salah satu tahapan dalam pengolahan citra adalah deteksi tepi. Dalam menganalisis karakter huruf aksara Lampung menggunakan metode deteksi tepi untuk melihat pola garis huruf tersebut. Metode pengolahan citra yang digunakan dalam mendeteksi tepi citra adalah metode deteksi tepi Roberts dan metode deteksi tepi Sobel. Hasil pengolahan citra akan masuk ke tahap pelatihan JST. Bobot yang didapat dari pelatihan kemudian akan digunakan pada tahap pengujian. Metode Sobel dapat mengenali karakter aksara Lampung lebih baik, dimana hasil dari pengujian karakter aksara Lampung memberikan nilai presentase error sebesar 28.5% dengan metode deteksi Roberts dan 14.5% dengan metode deteksi Sobel.

Kata kunci: *Aksara Lampung, deteksi tepi, JST backpropagation, nilai error, pengolahan citra, Roberts, Sobel*

STUDI PERBANDINGAN PENGENALAN KARAKTER AKSARA LAMPUNG DENGAN METODE DETEKSI TEPI ROBERTS DAN SOBEL

Oleh

HALIM ABDILLAH SHOLEH

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA TEKNIK

Pada

Jurusan Teknik Elektro
Fakultas Teknik Universitas Lampung

FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2018

Judul Skripsi	: **STUDI PERBANDINGAN PENGENALAN KARAKTER AKSARA LAMPUNG DENGAN METODE DETEKSI TEPI ROBERTS DAN SOBEL**
Nama Mahasiswa	: Halim Abdillah Sholeh
Nomor Pokok Mahasiswa	: 1215031037
Program Studi	: Teknik Elektro
Fakultas	: Teknik

MENYETUJUI

1. Komisi Pembimbing

Yessi Mulyani, S.T., M.T.	**Ing. Hery Dian Septama, S.T.**
NIP 19731226 200012 2 001	NIP 19850915 200812 1 001

2. Ketua Jurusan Teknik Elektro

10/10-2018

Dr. Ing. Ardian Ulvan, S.T., M.Sc.
NIP 19731128 199903 1 005

Disahkan Tanggal:

MENGESAHKAN

1. Tim Penguji

Ketua : **Yessi Mulyani, S.T., M.T.** .....

Sekretaris : **Ing. Hery Dian Septama, S.T.** .....

Penguji
Bukan Pembimbing : **Dr. Agus Trisanto, S.T., M.T.** .....

2. Dekan Fakultas Teknik

Prof. Suharno, M.Sc., Ph.D.
NIP 19620717 198703 1 002

Tanggal Lulus Ujian Skripsi : **7 Juni 2018**

viii

SURAT PERNYATAAN

Yang bertanda tangan dibawah ini, saya:

Nama	: Halim Abdillah Sholeh
NPM	: 1215031037
Jurusan	: Teknik Elektro
Fakultas	: Teknik
Judul	: Studi Perbandingan Pengenalan Karakter Aksara Lampung dengan Metode Deteksi Tepi Roberts dan Sobel

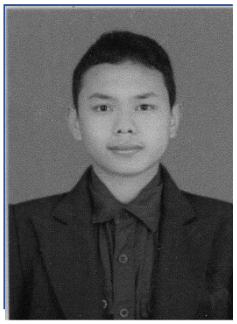
Dengan ini saya menyatakan bahwa skripsi ini dibuat oleh saya sendiri. Adapun karya orang lain yang terdapat dalam skripsi ini telah dicantumkan sumbernya pada daftar pustaka.

Apabila pernyataan saya tidak benar maka saya bersedia dikenai sanksi sesuai hukum yang berlaku.

Bandar Lampung, 7 Juni 2018

Halim Abdillah Sholeh
1215031037

RIWAYAT HIDUP



Penulis dilahirkan di Bandung, Jawa Barat pada tanggal 15 Februari 1995 sebagai anak pertama dari enam bersaudara dari Bapak Fitriadi dan Ibu Syifa.

Riwayat Pendidikan penulis dimulai dari Taman Kanak-kanak (TK) Pelita Antapani, Bandung diselesaikan tahun 2000, Sekolah Dasar (SD) diselesaikan di SDN Griba 14 Antapani, Bandung pada tahun 2006, Sekolah Menengah Pertama (SMP) di SMPN 14 Bandung pada tahun 2009, dan Sekolah Menengah Atas (SMA) di SMAN 23 Bandung pada tahun 2012.

Tahun 2012, Penulis terdaftar di Universitas Lampung sebagai mahasiswa Jurusan Teknik Elektro Fakultas Teknik melalui jalur SNMPTN (Seleksi Nasional Masuk Perguruan Tinggi Negeri) Undangan. Selama menjadi mahasiswa penulis pernah menjadi asisten Laboratorium Pemodelan dan Simulasi Universitas Lampung pada tahun 2013 sampai 2015 dan aktif di organisasi Himpunan Mahasiswa Teknik Elektro (HIMATRO) Universitas Lampung pada tahun 2013 sampai 2014. Pada Agustus 2015 Penulis melakukan kerja praktek di PT. Wincor Nixdorf Indonesia.

Motto

"Sesungguhnya sesudah kesulitan itu ada kemudahan, sesungguhnya sesudah kesulitan itu ada kemudahan" (Q.S. Al-Insyirah: 5-6)

Biar tidak capek itu harus ikhlas. Ikhlas itu harus sama antara hati, perbuatan dan pikiran (Jusuf Kalla)

Perubahan tidak akan pernah terjadi jika kita terus menunggu waktu atau orang yang tepat. Kita adalah perubahan itu sendiri (Barrack Obama)

Dengan mengucapkan syukur kepada Allah SWT dan dengan rendah hati dan rasa hormat skripsi ini penulis persembahkan kepada :

Kedua Orang tua dan kelima adikku tersayang serta keluarga besar dan sahabat-sahabat yang selalu memberikan semangat motivasi, nasihat dan doa didalam proses menyelesaikan skripsi ini.

SANWACANA

Puji syukur kepada Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga skripsi ini dapat diselesaikan dengan baik dan tepat waktu.

Skripsi dengan judul "Studi Perbandingan Pengenalan Karakter Aksara Lampung dengan Metode Deteksi Tepi Roberts dan Sobel" merupakan salah satu syarat untuk memperoleh gelar sarjana Teknik di Universitas Lampung.

Dalam kesempatan ini penulis mengucapkan banyak terima kasih kepada:

1. Bapak Prof. Suharno, M.Sc., Ph.D., selaku Dekan Fakultas Teknik Universitas Lampung
2. Bapak Irza Sukmana, S.T., M.T., Ph.D., selaku Wakil Dekan (WD) 1 FT Unila
3. Bapak Dr. Ing. Ardian Ulvan, S.T., M.Sc., selaku Ketua Jurusan Teknik Elektro
4. Ibu Yessi Mulyani, S.T., M.T., selaku Ketua Jurusan Teknik Informatika Universitas Lampung sekaligus dosen pembimbing utama skripsi atas ketersediaanya meluangkan waktu, motivasi, dukungan, bimbingan dan arahannya
5. Bapak Ing. Hery Dian Septama, S.T., selaku pembimbing kedua yang telah meluangkan waktu untuk memberika bimbingan, saran dan kritik dalam proses penyelesaian skripsi ini.

6. Bapak Dr. Agus Trisanto, S.T., M.T., selaku dosen penguji utama skripsi yang telah memberikan dukungan, saran dan kritik yang sangat membangun.
7. Ibu Dr. Endah Komalasari, S.T., M.T., selaku pembimbing akademik atas bimbingan yang telah diberikan selama penulis menjadi mahasiswa.
8. Segenap dosen dan pegawai Jurusan Teknik Elektro yang telah memberi materi dan pengalaman selama penulis menduduki bangku perkuliahan dari awal hingga akhir.
9. Segenap keluarga dirumah dan keluarga besar, terima kasih atas doa, dukungan baik secara moril maupun materil selama ini.
10. Penghuni Puskom UNILA dan Laboratorium Terpadu Teknik elektro, yang telah senantiasa bekerja sama dalam berbagai hal baik itu pengalaman, pengetahuan, candaan dan lain sebagainya.
11. Teman-teman elektro 2012 yang telah memberikan semangat dalam menjalankan Pendidikan di Teknik Elektro Universitas Lampung, semoga kita bisa sukses Bersama.

Akhir kata, penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan, akan tetapi sedikit harapan semoga skripsi yang sederhana ini dapat berguna dan bermanfaat bagi kita semua. Dan semoga ALLAH SWT membalas kebaikan semua pihak yang telah membantu dalam penyelesaian skripsi ini. Aamiin.

Bandar Lampung, 7 Juni 2018
Penulis,

Halim Abdillah Sholeh

DAFTAR ISI

Halaman

DAFTAR TABEL`

Tabel Halaman

DAFTAR GAMBAR

I. PENDAHULUAN

1.1 Latar Belakang

Aksara Lampung merupakan bahasa yang digunakan oleh penduduk asli Lampung dalam melakukan komunikasi. Aksara Lampung merupakan ciri sebagai identitas diri bagi provinsi Lampung. Sebagai identitas diri, maka penggunaan aksara Lampung tidak boleh hilang dan harus dilestarikan. Karena banyaknya pendatang dari luar Lampung sehingga aksara Lampung menjadi tidak terlalu banyak dipakai dalam komunikasi sehari-hari. Perlu upaya-upaya agar aksara Lampung tetap terjaga kelestariannya. Salah satu upaya yang dilakukan adalah memanfaatkan teknologi yang sering digunakan sehari-hari oleh masyarakat seperti komputer, *tablet* dan *smartphone*. Perangkat tersebut sudah sangat banyak digunakan oleh masyarakat dengan berbagai macam kelebihannya.

Untuk penggunaan perangkat seperti komputer dibutuhkan agar penelitian mengenai karakter huruf menjadi lebih efisien. Penelitian ini menggunakan pengolahan citra. Sebagai informasi visual, peran citra sangatlah penting. Penglihatan secara matematis pada citra adalah fungsi kontinu dari intensitas cahaya didalam bidang dua dimensi dengan x dan y adalah koordinat spasial serta amplitude f pada pasangan koordinat (x,y) biasa disebut dengan intensitas atau level keabuan di titik itu. Jika x, y dan f adalah berhingga kesemuanya dan diskrit citra merupakan sebuah citra digital [1].

Karena banyaknya bentuk dan variasi aksara maka jaringan saraf tiruan (JST) selalu berperan membuat pelatihan karakter aksara menjadi lebih efisien dan simpel. JST telah membuktikan untuk memiliki kemampuan memodelkan hubungan yang tidak spesifik antara bentuk *input* dan bentuk *output* [2].

Teknik pengolahan citra yang digunakan salah satunya adalah deteksi tepi. Deteksi tepi merupakan deteksi yang dapat menentukan titik tepi yang memiliki sebuah nilai level warna yang drastis pada tepi dari objek. Disana ada beberapa metode yang digunakan didalam mendeteksi tepi, Yakni Metode *Laplacian of Gaussian* (LoG), Roberts, Sobel, Prewitt, Canny dan sebagainya. Dalam skripsi ini, akan dibahas dua buah metode deteksi tepi yaitu metode *Roberts* dan *Sobel*, yang mana hasilnya akan dibandingkan untuk memilih yang lebih baik dalam pengenalan karakter aksara Lampung. Berikut adalah penelitian sebelumnya mengenai aksara Lampung:

Tabel 1.1 Penelitian Aksara Lampung

No	Nama (NPM)	Tahun	Judul Penelitian	Keterangan
1	Hendri Setiawan	2014	Rancang Bangun Aplikasi Pengenalan Tulisan Tangan Aksara Lampung dengan Masukan Layar Sentuh Menggunakan JST *Backpropagation*	Tidak menggunakan deteksi tepi
2	Eliza Hara	2016	Sistem Pengenalan Tulisan Tangan Aksara Lampung dengan Metode Deteksi Tepi (*Canny*) Berbasis JST *Backpropagation*	Menggunakan deteksi tepi

1.2 Tujuan Penelitian

Adapun tujuan penelitian ini adalah sebagai berikut:

1. Menganalisis hasil pelatihan JST *Backpropagation* pada karakter huruf aksara Lampung.
2. Menganalisis hasil perbandingan deteksi tepi *Roberts* dan *Sobel* pada karakter huruf aksara Lampung.

1.3 Manfaat Penelitian

Manfaat yang diharapkan dari hasil penelitian ini adalah sebagai berikut:

1. Mengetahui kinerja JST *Backpropagation* pada karakter huruf aksara Lampung.
2. Dengan adanya studi perbandingan deteksi tepi dapat memberikan informasi tentang kinerja metode deteksi tepi dalam melatih karakter huruf aksara Lampung.

1.4 Rumusan Masalah

Rumusan masalah yang dapat ditarik dari penjelasan latar belakang adalah sebagai berikut:

1. Bagaimana melatih karakter aksara Lampung dengan JST *Backpropagation*.
2. Bagaimana proses perbandingan hasil pengujian deteksi tepi *Roberts* dan *Sobel*.

1.5 Batasan Masalah

Rumusan masalah diatas, dibatasi dengan beberapa hal sebagai berikut:

1. JST yang digunakan adalah JST *Backpropagation*.
2. Membahas deteksi tepi *Roberts* dan *Sobel* pada pengolahan citra.
3. Membahas JST *Backpropagation*.
4. Data sampel berupa karakter aksara Lampung.

5. Aplikasi yang digunakan : Matlab R2009a.

1.6 Hipotesis

Pada penelitian ini diharapkan metode deteksi tepi *Roberts* dan *Sobel* mampi mengenali karakter aksara Lampung.

1.7 Sistematika Penulisan Laporan

Untuk memudahkan penulisan dan pemahaman mengenai materi tugas akhir ini maka tulisan dibagi menjadi lima bab, yaitu:

BAB I Pendahuluan

Memuat latar belakang, tujuan, manfaat, perumusan masalah, Batasan masalah, hipotesis dan sistematika penulisan.

BAB II Tinjauan Pustaka

Menjelaskan landasan teori yang digunakan dalam penelitian yang telah dan akan dilakukan berhubungan dengan penelitian.

BAB III Metode Penelitian

Menjelaskan mengenai metode penelitian yang digunakan dimana berisi waktu dan tempat penelitian, alat dan bahan dan tahap-tahap perancangan.

BAB IV Hasil dan Pembahasan

Membahas pengujian dan hasil penelitian dan kinerja alat atau sistem yang telah dirancang

BAB V Simpulan dan Saran

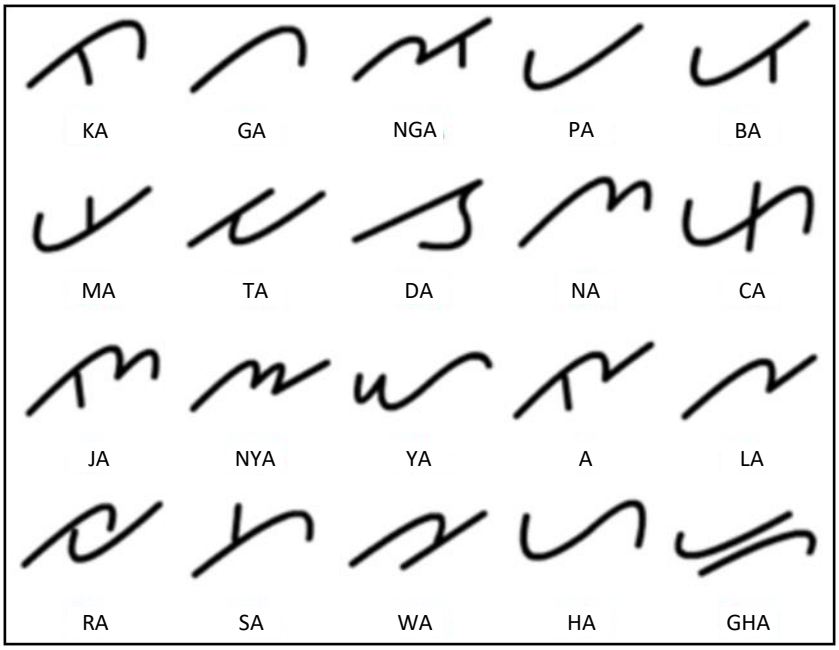
Memuat simpulan dan saran dari penelitian yang telah dilakukan

II. TINJAUAN PUSTAKA

2.1 Aksara Lampung

2.1.1 Induk Huruf (*Kelabai Sukhat*)

Pada Penulisan aksara Lampung didalamnya terdapat 20 huruf induk, yaitu: ‘ka’, ‘ga’, ‘nga’, ‘pa’, ‘ba’, ‘ma’, ‘ta’, ‘da’, ‘na’, ‘ca’, ‘ja’, ‘nya’, ‘ya’, ‘a’, ‘la’, ‘ra’, ‘sa’, ‘wa’, ‘ha’ dan ‘gha’ [3]. Bentuk induk huruf tersebut dapat diperlihatkan pada gambar 2.1.



Gambar 2.1 Induk huruf aksara Lampung

2.1.1 Anak Huruf (*Anak Sukhat*)

Dalam aksara Lampung terdapat 12 anak huruf yakni di atas, di bawah dan samping kanan induk huruf.

a. Anak Huruf yang berada di Atas Induk Huruf

Anak huruf yang terletak di atas induk huruf terdapat 6 (enam) anak huruf yang ditunjukkan pada Tabel 2.1.

Tabel 2.1 Anak huruf yang berada di atas induk huruf

Nama 'bunyi'	Symbol
Ulan 'I'	----------
Ulan 'e'	----------
Bicek 'e'	----------
Rejunjung 'r'	----------
Tekelubang 'ng'	----------
Datas 'an'	----------

b. Anak Huruf yang berada di Bawah Induk Huruf

DalamAnak huruf yang terletak di bawah induk huruf terdapat 3 (tiga) anak huruf seperti ditunjukkan pada Tabel 2.2.

Tabel 2.2 Anak Huruf yang berada di Bawah Induk Huruf

Nama 'bunyi'	Symbol
Bitan 'o'	----------
Bitan 'u'	----------
Tekelungau 'au'	----------

c. Anak Huruf yang berada di Depan Induk Huruf

Pada Anak huruf yang terletak di depan induk huruf terdapat 3 (tiga) anak huruf sepertiditunjukkan pada Tabel 2.3.

Tabel 2.3 Anak Huruf yang berada di Depan Induk Huruf

Nama 'bunyi'	Symbol
Tekelingai 'ai'	---------- ı
Keleniah 'ah'	---------- ᔕ
nengen	---------- ∫

2.2 Pengolahan Citra

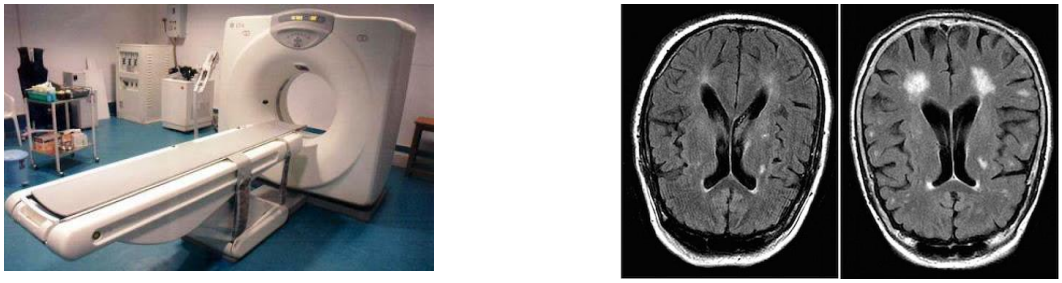
Penggunaan citra digital sangat popular pada masa sekarang. Peralatan peralatan elektronik yang diaplikasikan dalam kegiatan sehari-hari, misalnya *fingerprint reader* (pembaca sidik jari), *scanner*, kamera digital dan mikroskop digital, yang sudah menghasilkan citra digital. Perangkat lunak (*software*) sudah digunakan oleh pengguna dalam mengolah citra yang diinginkan. Contohnya adalah *Adobe Photoshop* dan GIMP (GNU *Image Manipulation program*) dengan menyajikan fitur-fitur untuk memanipulasi citra digital juga sebagai *editor* yang baik untuk sebuah citra.

Pengolahan citra adalah pengertian umum untuk berbagai teknik yang keberadaanya sebagai manipulasi dan modifikasi citra dengan cara yang variatif. Pengolahan citra digital disebut sebagai pemrosesan gambar berdimensi dua melalui komputer digital. Contoh gambar yang dapat diolah dengan mudah pada bentuk dimensi dua adalah foto. Ketika foto tersebut diubah menjadi bentuk digital, berbagai macam proses pengolahan dapat diterapkan terhadap citra tersebut. Selain citra digital terdapat citra analog. Salah satu contoh citra analog adalah seperti foto

mahasiswa yang terdapat pada kartu mahasiswa. Agar foto tersebut bisa menjadi digital maka dilakukan *scan* sehingga bisa ditampilkan di layer komputer [4].

Pengolahan citra merupakan bagian dari penerapan aplikasi yang penting dalam kebutuhan pengguna. Penerapannya adalah seperti pengenalan pola, penginderaan jarak jauh menggunakan satelit atau pesawat udara serta *machine vision*. Penerapan pada pengenalan pola adalah seperti segmentasi yaitu pemisahan antara latar belakang dengan objek. Kemudian, objek tersebut dilakukan klasifikasi pola. Sebagai contoh, sebuah objek buah dapat dikenali sebagai mangga, jeruk atau anggur. Dalam identifikasi untuk objek yang didapatkan didalam citra dapat digunakan untuk penginderaan jarak jauh, tekstur atau warna. Pengolahan citra memiliki peran penting dalam mengenali bentuk variasi mesin yang terdapat pada *machine vision*. Penggunaan kamera dalam ruangan sperti cctv merupakan salah satu bagian dari aplikasi pemrosesan citra. Perubahan gerakan yang tertangkap dari citra tersebut dapat menjadi acuan dalam melaporkan situasi yang terekam saat itu.

Aplikasi pengolahan citra sudah digunakan secara luas. Seperti pada dunia kedokteran, pengolahan citra berperan sangat penting pada berbagai alat kedokteran. Salah satu contoh alatnya adalah CT *Scan* (*Computed Tomography Scan*) atau sebutan lainnya adalah *Computerized Axial Tomography Scan* yang juga bagian dari contoh aplikasi pengolahan citra, yang penggunaannya untuk melihat bagian-bagian dalam dari tubuh manusia. Tomografi merupakan suatu proses yang mana bisa menghasilkan citra berdimensi dua dari potongan objek berdimensi tiga yang mana proses sebelumnya berasal dari hasil pemindaian satu dimensi [4].



Gambar 2.2 CT *Scan* dan contoh hasil CT *Scan*

Aplikasi yang diterapkan pada CT *Scan* menggunakan prinsip dasar dalam pengolahan citra yakni peningkatan kontras dan kecerahan, penghilangan derau pada citra, dan pencarian bentuk objek. Seringkali menjumpai citra yang tidak jelas akibat kekurangan sinar ketika objek dibidik dengan kamera. Dengan pengolahan citra, hal tersebut diperbaiki dengan peningkatan kecerahan dan kontras. Citra Biasanya sering dalam keadaan terdistorsi atau memiliki derau. Jika memang diperlukan maka derau wajib dibersihkan. Untuk keperluan-keperluan tersebut, beberapa metode digunakan untuk mengolah citra. Salah satu caranya adalah dengan melakukan penggunaan filter. Agar tercapainya didalam mengenali objek citra tersebut, maka objek perlu dipisahkan dari latar belakangnya. Salah satu pendekatan yang umum dipakai adalah pendekatan batas objek. Dalam hal ini batas objek adalah bagian dari tepi objek. Tepi objek yang telah didapatkan dilakukanlah pencarian ciri dari objek tersebut, pencarian ciri terhadap objek dan melakukan pemisahan objek tersebut dengan latar belakang yang digunakan. Pemisahahan antara objek citra dengan latar belakang yang digunakan pada objek dikenal dengan sebutan segmentasi [4].

Pengolahan citra adalah proses awal dalam pembuatan aplikasi pengenalan aksara Lampung. Dalam penelitian ini, metode pengolahan citra yang digunakan adalah

binerisasi, *slicing*, deteksi tepi *Roberts* dan *Sobel*, dilasi, *fill*, *cropping*, *resizing* dan vektorisasi.

2.2.1 Binerisasi

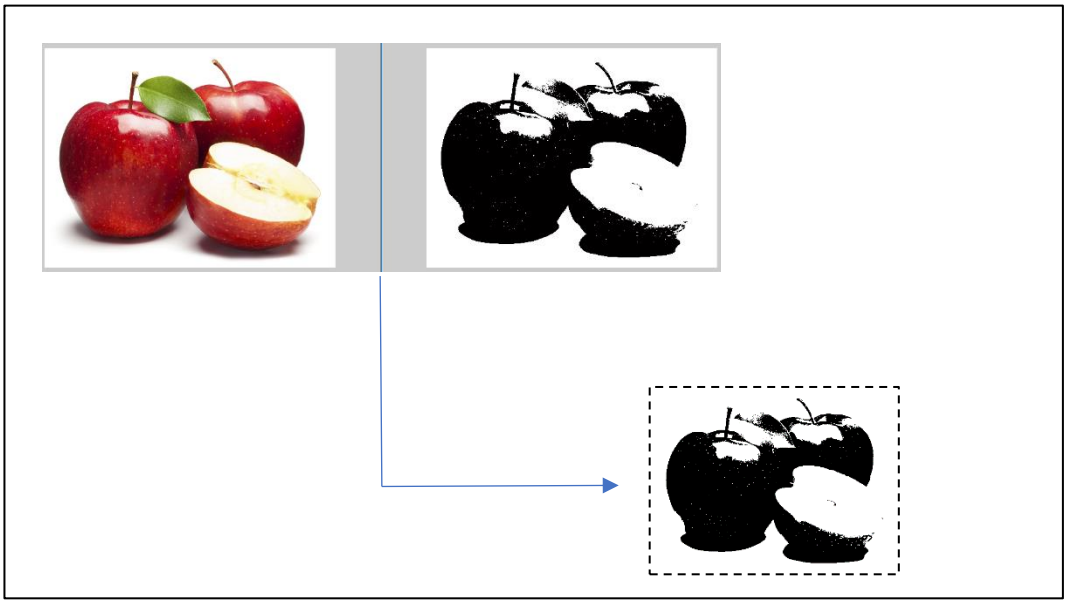
Binerisasi adalah salah satu proses yang mana citra tersebut akan dikenali dengan warna yang telah ada ketentuannya sehingga menjadi bagian penting dalam pengolahan citra. Binerisasi citra merupakan proses perubahan warna kedalam bentuk biner. Adapun biner yang telah diketahui adalah 0 dan 1. Dari proses tersebut didapatkan 2 warna saja yaitu warna hitam dan putih. Sebuah citra biner adalah citra dengan nilai piksel seperti salah satu dari dua nilai diskrit yaitu "off" dan "on". Penggunaan citra biner memudahkan untuk membedakan fitur-fitur struktural, seperti membedakan objek dari latar belakang. Umumnya suatu citra biner disimpan sebagai matrik dua dimensi yang berisi nilai satu menyatakan piksel "on" dan nol menyatakan piksel "off"[5].

Gambar 2.3 Citra Hasil Binerisasi

2.2.2 *Slicing*

Slicing merupakan salah satu proses pengolahan citra yang melakukan proses pemotongan yang dibutuhkan sebagai informasi yang berupa citra untuk keperluan penelitian. Sebagai contohnya adalah buah apel.Pada gambar apel, proses *slicing* dilakukan untuk memisahkan apel tersebut dari latar belakang. Dapat dikatakan

bahwa proses ini merupakan bagian dari segmentasi yaitu pemisahan antara objek yang ingin diambil bagiannya dengan latar belakang.



Gambar 2.4 Citra hasil *slicing*

Dari gambar tersebut memperlihatkan proses *slicing*, dimana suatu objek diambil dan dipisakan dari objek lain atau latar belakang.

2.2.3 Deteksi Tepi

Deteksi tepi salah satu Teknik pengolahan citra yang berfungsi untuk memperoleh tepi-tepi objek yang menjadi keperluan dalam penelitian. Deteksi tepi dilakukan dengan memanfaatkan perubahan warna yang signifikan pada area yang berbeda. Definisi tepi pada objek disini adalah “himpunan piksel yang saling terhubung antara area pertama dan area kedua”. Tepi yang diambil merupakan tepi yang sangat penting yaitu informasi yang diperlukan sebagai bahan penelitian. Informasi yang didapat dari tepi bisa berupa bentuk maupun ukuran objek tergantung keperluan dan kebutuhan. Umumnya deteksi tepi menggunakan dua macam detector, yaitu

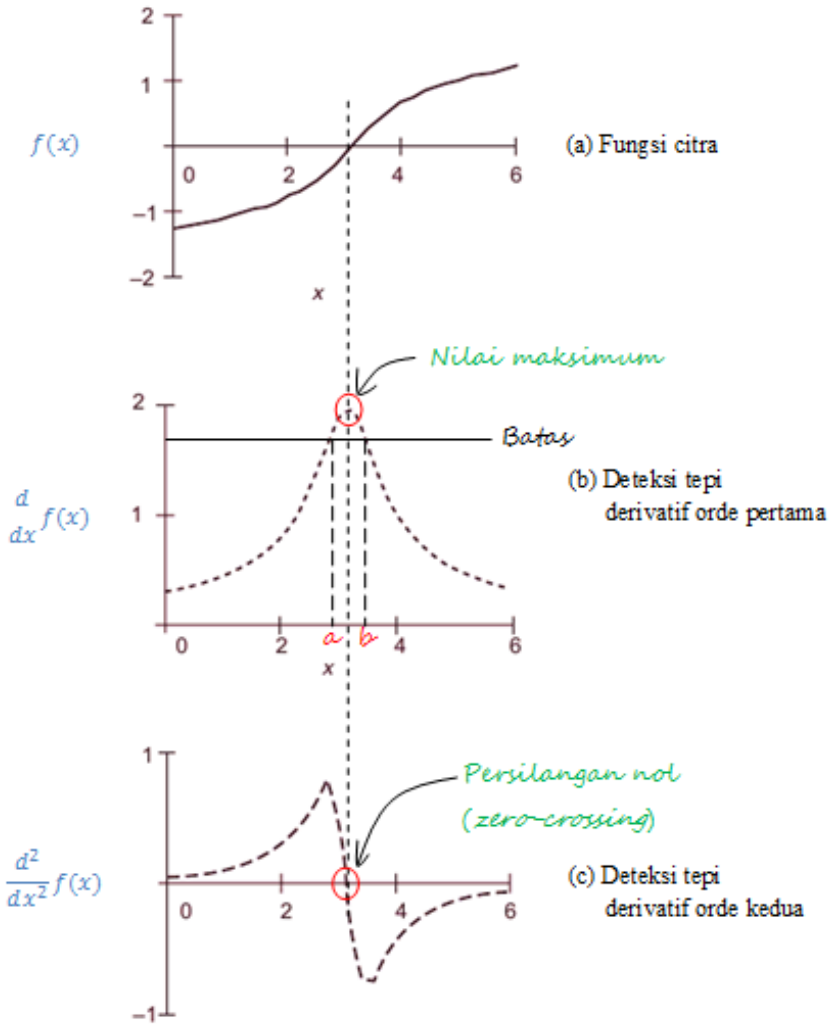
detector baris (H_y) dan detector kolom (H_x). Yang tergolong jenis ini adalah operator *Roberts*, *Prewitt*, *Sobel*, *Frei-chen*.

Deteksi tepi memiliki dua golongan. Golongan pertama disebut deteksi tepi orde pertama, yang bekerja dengan menggunakan turunan atau differensial orde pertama. Termasuk kelompok ini adalah operator *Roberts*, *Prewitt*, *Sobel*. Golongan kedua disebut sebagai deteksi tepi orde kedua, yang menggunakan turunan orde kedua. Contoh yang termasuk kelompok ini adalah *Laplacian of Gaussian* [4].

Tabel 2.4 memberikan definisi turunan orde pertama dan kedua baik itu kontinu maupun diskrit. Bentuk diskrit sangat berguna untuk melakukan deteksi tepi. Gambar 2.4 menunjukan hubungan antara fungsi citra dan deteksi tepi orde pertama dan orde kedua. Turunan tepi yang perlu diketahui biasanya terletak pada nilai absolut maksimum pada turunan pertama dan persilangan nol (*zero-crossing*) pada turunan kedua [4].

Tabel 2.4 orde pertama dan kedua pada bentuk kontinu dan diskrit

Turunan	Bentuk kontinu	Bentuk diskret
$\frac{df}{dx}$	$\lim_{\Delta x \to 0} \frac{f(y, x+\Delta x) f(y,x)}{\Delta x}$	f(y,x+1)f(y,x)
$\frac{df}{dy}$	$\lim_{\Delta x \to 0} \frac{f(y+\Delta y, x) f(y,x)}{\Delta y}$	f(y+1,x)f(y,x)
$\nabla f(y,x)$	$\left[\frac{df}{dy}, \frac{df}{dx}\right]$	[f(y, x+1)f(y,x), f(y+1,x)f(y,x)]
$\frac{d^2 f}{dx^2}$	$\lim_{\Delta x \to 0} \frac{\left(\frac{df}{dx}\right)(y, x+\Delta x)\left(\frac{df}{dx}\right) f(y,x)}{\Delta x}$	f(y,x+1)- 2f(y,x)+f(y,x-1)
$\frac{d^2 f}{dy^2}$	$\lim_{\Delta x \to 0} \frac{\left(\frac{df}{dy}\right)(y+\Delta y, x)\left(\frac{df}{dy}\right) f(y,x)}{\Delta y}$	f(y+1,x) -2f(y,x)+f(y-1,x)
$\nabla^2 f(y,x)$	$\frac{d^2 f}{dx^2} + \frac{d^2 f}{dy^2}$	f(y,x+1)+f(y,x-1) - 4f(y,x)+f(y+1,x)+f(y-1,x)



Gambar 2.5 Deteksi orde pertama dan kedua pada arah x

Contoh di gambar 2.4 (a) menunjukkan keadaan fungsi intensitas citra f(y,x) pada arah x dengan bentuk tepi tanjakan landau. Gambar 2.4 (b) menunjukkan keadaan turunan pertama sedangkan persilangan digambar 2.4 (c) menyatakan letak tepi pada turunan kedua. Apabila nilai batas dikenakan pada turunan pertama, puncak tidak lagi menjadi tepi. Akibatnya, terdapat dua nilai yang memenuhi (yaitu a dan b). Kedua nilai tersebut akan menjadi piksel-piksel tepi. Berbeda halnya pada turunan kedua, tepi akan selalu berupa satu piksel. Hal ini terlihat pada perpotongan

fungsi turunan kedua dengan sumbu x. Akibatnya, ketebalan tepi akan selalu berupa satu piksel.

2.2.3.1 Metode *Roberts*

Metode *Roberts* merupakan metode dengan operator *Roberts* berbasis gradien yang menggunakan dua buah *kernel* yang berukuran 2x2 piksel. Operator ini mengambil arah diagonal untuk penentuan arah dalam perhitungan nilai gradien sehingga banyak dikenal sebagai operator silang. Perhitungan dalam operator *Roberts* sebagai berikut [6]:

$$G = \sqrt{R_x^2 + R_y^2} \qquad (2.1)$$

Dengan G = besar gradient operator Roberts

R_x = gradien Roberts arah horizontal

R_y = gradien Roberts arah vertikal

$$R_+ = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad R_- = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Gambar 2.6 Operator *Roberts*

Berikut adalah contoh *source code* pengolahan citra dengan deteksi tepi *Roberts* pada buah apel [7].

```
%pengolahan citra dengan deteksi tepi roberts
MyImg=double(imread('D:\final\task\reference\image\apelbiner.png'));
Gx=[1 0; 0 -1];
Gy=[0 1; -1 0];
fGx=imfilter(MyImg, Gx);
fGy=imfilter(MyImg, Gy);
G=sqrt((fGx.^2+fGy.^2));
imshow([MyImg uint8(G)])
```

Berikut adalah gambar buah apel sebelum dan setelah menggunakan deteksi tepi *Roberts*

Gambar 2.7 Citra hasil deteksi tepi *Roberts*

2.2.3.2 Metode *Sobel*

Metode *Sobel* merupakan metode dengan menggunakan dua buah kernel berukuran 3x3 piksel untuk perhitungan gradien sehingga perkiraan gradien berada tepat di tengah jendela. Besar gradien yang dihitung menggunakan operator *sobel* adalah sebagai berikut [6]:

$$G = \sqrt{S_x^2 + S_y^2} \qquad (2.2)$$

Dengan G = besar gradien operator *Sobel*

S_x = gradien *Sobel* horizontal

S_y = gradien *Sobel* vertikal

Dimana G adalah besar gradien di titik tengah kernel dan turunan parsial dihitung menggunakan persamaan sebagai berikut:

$S_x = (a_2 + ca_3 + a_4) – (a_0 + ca_7 + a_6)$ (2.3)

$S_y = (a_0 + ca_1 + a_2) – (a_6 + ca_5 + a_4)$ (2.4)

Dimana c adalah konstanta yang bernilai 2. S_x dan S_y diimplementasikan menjadi *kernel* berikut:

$$s_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad s_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Gambar 2.8 Operator *Sobel*

Berikut adalah contoh *source code* pengolahan citra dengan deteksi tepi *Sobel* pada buah apel [7].

```
%pengolahan citra dengan deteksi tepi sobel
MyImg=double(imread('D:\final task\reference\image\apelbiner.png'));
Gx=[-1 0 1; -2 0 2; -1 0 1];
Gy=[1 2 1; 0 0 0;-1 -2 -1];
fGx=imfilter(MyImg, Gx);
fGy=imfilter(MyImg, Gy);
G=sqrt((fGx.^2+fGy.^2));
imshow([MyImg uint8(G)])
```

Berikut adalah gambar buah apel sebelum dan setelah menggunakan deteksi tepi *Sobel*.

Gambar 2.9 Citra hasil deteksi tepi *Sobel*

2.2.4 Dilasi dan *Fill*

Operasi Dilasi biasa dipakai untuk mendapatkan efek pelebaran terhadap piksel bernilai 1. Maksudnya adalah penambahan piksel pada batas antar objek. Sehingga nantinya jika dilakukan operasi ini maka ukuran citra hasilnya lebih besar dari ukuran aslinya. Sedangkan operasi *fill* diterapkan sebagai pengisi *holes* di tengah karakter aksara [5].

2.2.5 *Cropping* dan *Resizing*

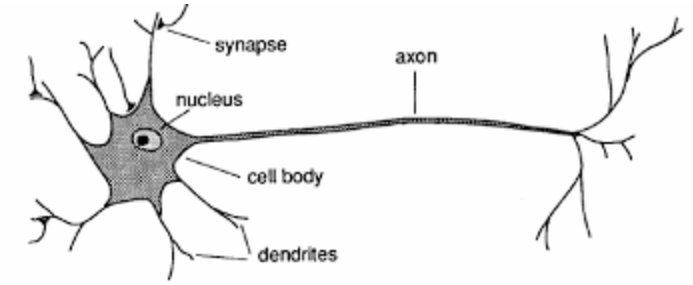
Cropping adalah proses pemotongan citra pada koordinat tertentu pada area citra. Untuk melakukan pemotongan citra perlu ada dua koordinat penting yaitu koordinat awal sebagai titik awal dilakukannya pemotongan dan titik akhir sebagai akhir dari pemotongan suatu citra. Dan hasil akhirnya akan terbentuk sebuah bangun segiempat sebagai informasi akurat dengan piksel yang belum ditentukan.

Pada saat kita melakukan proses pengolahan citra, ukuran citra akan menjadi lebih kecil atau lebih besar dibanding sebelumnya. *Resizing* merupakan perubahan pada citra di bagian sisi ukuran sehingga citra yang satu dengan citra yang lain memiliki ukuran yang sama atau bisa dikatakan piksel yang telah ditentukan sama rata. Pengubahan ukuran citra dilakukan dengan metode interpolasi. Interpolasi adalah metode dari sebuah citra yang bekerja dengan cara meningkatkan dan mengurangi jumlah piksel pada suatu citra digital. Interpolasi bekerja dengan menggunakan data yang diketahui untuk memperkirakan nilai-nilai pada titik yang tidak diketahui.

2.3 Jaringan Saraf Tiruan

Artificial Neural Network (ANN) atau Jaringan Saraf Tiruan (JST) adalah paradigma pengolah informasi yang memiliki kinerja oleh sistem saraf secara biologis, seperti proses informasi pada otak manusia. Hal yang paling penting dari paradigma ini adalah pemrosesan informasi yang saling berhubungan dan menyelesaikan masalah tertentu secar serentak dan bersamaan. Cara kerja JST memang meniru cara kerja manusia yang mana keduanya memiliki contoh. Ada yang namanya proses pembelajaran yang sangat penting didalam melakukan

pengenalan pola dan klasifikasi data. Belajar dalam sistem biologis melibatkan penyelesaian terhadap koneksi sinapsis yang ada antara neuron [8].

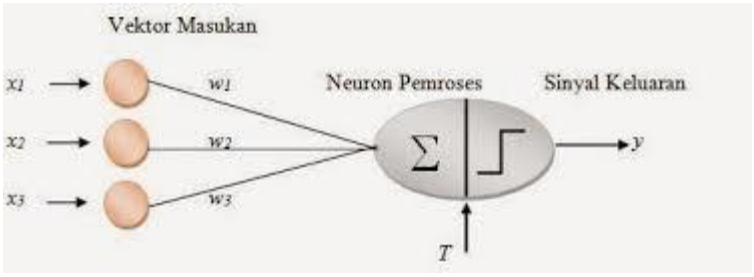


Gambar 2.10 Model Jaringan Saraf Biologi

JST merupakan salah satu bagian dari metode *soft computing. Soft computing* adalah kumpulan Teknik-teknik komputasi dalam ilmu komputer yang melakukan kegiatan analisis dan model suatu fenomena tertentu dengan tujuan untuk melakukan eksploitasi adanya toleransi terhadap ketidakpastian dan kebenaran parsial agar penyelesaiannya bisa dengan mudah serta penghematan biaya bisa murah. *Soft computing* berbeda dengan *conventional computing* yang memungkinkan toleransi terhadap *input,* proses, *output* bersifat tidak akurat, tidak pasti dan setengah benar. *Soft computing* berusaha untuk melakukan proses penggabungan dari beberapa model perhitungan. [8].

JST adalah sistem yang memproses informasi dengan cara melihat karakteristik-karakteristik yang mirip dengan jaringan saraf biologi. JST termasuk salah satu mesin yang memodelkan cara kerja otak yang dirancang fungsi tersebut dapat bekerja sesuai dengan tugasnya atau tugas-tugas tertentu. Dengan menganalogikan sistem kerja otak manusia tersebut, JST memiliki sebuah unit pemroses yang disebut neuron (akson jika dalam otak manusia) yang berisi penambah dan fungsi aktivasi, sejumlah bobot (sinapsis dalam otak manusia), sejumlah vektor masukan

(dendrit dalam otak manusia). Fungsi aktivasi berguna untuk mengatur keluaran yang diberikan oleh neuron. Desain JST secara umum ditunjukan pada gambar 2.11.



Gambar 2.11 Model Jaringan Saraf Tiruan

Pada gambar tersebut vektor masukan terdiri dari sejumlah nilai (fitur) yang diberikan sebagai nilai masukan pada JST, vektor masukan tersebut ada tiga nilai (x_1, x_2, x_3) sebagai fitur dalam vektor yang akan diproses dalam JST masing-masing nilai masukan melewati sebuah hubugnan berbobot 2, kemudian semua nilai digabungkan. Nilai gabungan tersebut kemudian diproses sebagai fungsi aktivasi untuk menghasilkan sinyal y sebagai keluaran. Fungsi aktivasi memiliki nilai ambang yang memiliki fungsi sebagai batas nilai agar selalu dalam batasan nilai tersebut.

Diibaratkan sebagai manusia yang belajar dari lingkungan agar manusia bisa mengelola lingkungan dengan baik karena belajar dan terus belajar sehingga menjadi pengalaman dan akhirnya bisa mengenali berdasarkan pengalaman tersebut. JST sebagai model yang tugasnya melakukan pelatihan terhadap suatu data latih maupun data uji dalam proses apapun dengan tujuan mendapatkan nilai bobot baru yang diperlukan dalam pengujian nantinya. Proses pelatihan dalam JST dapat menggunakan algoritma-algoritma seperti, *Perceptron, Backpropagation,*

Self-Organizing Map (SOM), Delta, Associative Memory, Learning Vector Quantization, dan sebagainya.

2.3.1 Mengapa Menggunakan JST

Untuk menjawab pertanyaan "mengapa menggunakan JST?" JST mempunyai kemampuan yang luar biasa untuk mendapatkan informasi dari data yang rumit atau tidak pas ketepatannya, mampu menjawab permasalahan yang tidak terstruktur dan terdefinisi, JST mampu belajar dari pengalaman, walaupun tidak ada kepastian dengan JST dapat mengakuisisi pengetahuan yang tidak pasti tersebut, generalisasi dan ekstraksi dilakukan bila data tersebut ingin dikenali pola-pola tertentu saja, terciptanya pola pengetahuan melalui pengembangan kemampuan belajar (*self-organizing*), memilih *input* yang sudah bisa ditentukan sehingga bisa diakukan pengenalan (klasifikasi), Jika diberikan data hanya sebagian namun dengan JST data tersebut bisa dikenali secara keluruhan karena memiliki keterkaitan antar data (asosiasi), mempunyai kemampuan mengolah data-data *input* tanpa harus mempunyai target (*self-organizing*) dan mampu menemukan jawaban terbaik sehingga mampu meminimalisasi jumlah biaya (optimasi) [8].

JST adalah representasi buatan dari otak manusia yang mencoba mensimulasikan dan memodelkan proses pembelajaran pada otak manusia tersebut. Buatan dari otak manusia tersebut memiliki makna didalam implementasinya menggunakan program komputer dalam proses perhitungan selama pembelajaran.

JST dibentuk sebagai generalisasi model matematika dengan asumsi bahwa [8]:

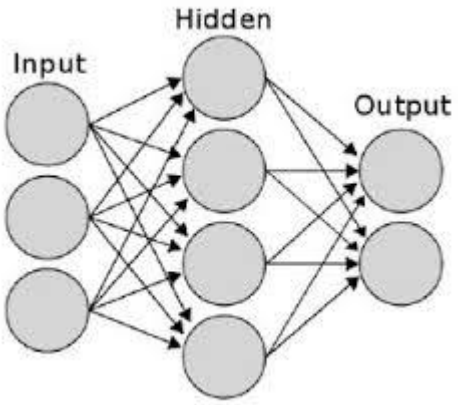
1. Pemrosesan informasi terjadi pada banyak elemen sederhana (neuron).
2. Sinyal dikirimkan diantara neuron-neuron melalui penghubung dendrit dan akson
3. Penghubung antar elemen memiliki bobot yang akan menambah atau mengurangi sinyal.
4. Untuk menentukan *output*, setiap neuron memiliki fungsi aktivasi yang dikenakan pada jumlah semua *input*-nya. Besar *output* akan dibandingkan dengan nilai *threshold* tertentu.

Dengan meniru sistem jaringan saraf biologis maka, model JST memiliki tiga karakterisitik utama [9]:

1. Arsitektur jaringan merupakan pola keterhubungan antara neuron. Keterhubungan neuron-neuron inilah yang membentuk suatu jaringan.
2. Algoritma jaringan merupakan metode untuk menentukan nilai bobot hubungan. Ada dua jenis metode, yaitu metode pelatihan atau pembelajaran (memorisasi) dan metode pengenalan atau aplikasi.
3. Fungsi aktivasi merupakan fungsi untuk menentukan nilai *output* berdasarkan nilai total masukan pada neuron. Fungsi aktivasi suatu algoritma jaringan dapat berbeda dengan fungsi aktivasi algoritma jaringan yang lain.

2.3.2 Arsitektur Jaringan

JST salah satunya ditentukan oleh hubungan antar-neuron disebut juga sebagai arsitektur jaringan. Neuron-neuron yang terkumpul dalam lapisan-lapisan disebut neuron-layer [4]. Arsitektur JST dapat dilihat pada gambar berikut :



Gambar 2.12 Arsitektur JST

Lapisan-lapisan penyusun JST terbagi menjadi tiga yaitu:

1. Lapisan *Input* (*Input Layer*)

   Lapisan *input* ini menerima sinyal dari luar kemudian masuk kedalam neuron *input* dan sinyal tersebut diteruskan ke neuron-neuron lain dalam jaringan tersebut. Ciri dari lapisan *input* tersebut dengan melihat cara kerja sel saraf sensor pada jaringan saraf biologi.

2. Lapisan Tersembunyi (*Hidden Layer*)

   Lapisan hidden merupakan lapisan yang berfungsi membuat pelatihan menjadi lebih lama karena didalam hidden melakukan kegiatan pemecahan masalah. Konsekuensi yang diterima apabila lapisan tersebut terdapat hidden adalah pelatihan tersebut membutuhkan waktu yang lama dalam prosesnya dan pelatihannya menjadi lebih sulit.

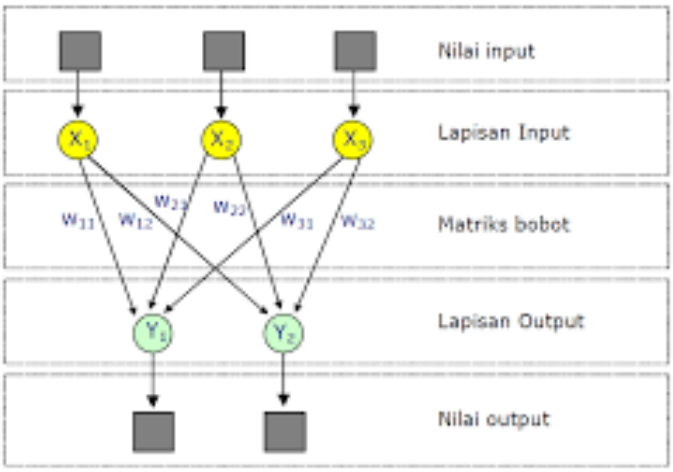
3. Lapisan *Output* (*Output* Layer)

   Lapisan *output* berfungsi untuk menyalurkan sinyal-sinyal yang ada dalam jaringan berupa bobot baru dan bias baru yang digunakan dalam perangkat uji. Pada jaringan saraf biologi lapisan ini juga terdiri dari beberapa neuron.

Beberapa arsitektur jaringan yang sering digunakan dalam jaringan saraf tiruan antara lain:

1. Jaringan Lapisan Tunggal

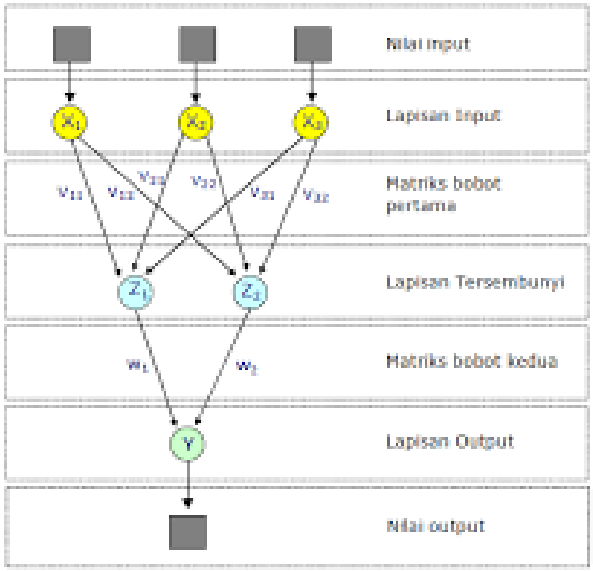
   Jaringan dengan lapisan tunggal memiliki satu lapisan *input* dan satu lapisan *output*. Didalam data yang terdapat dalam lapisan *input* dilangsungkan terhubung dengan lapisan *output* tanpa melewati lapisan hidden. Contoh JST: *perceptron*.



Gambar 2.13 Jaringan dengan Lapisan Tunggal

2. Jaringan Lapisan Banyak

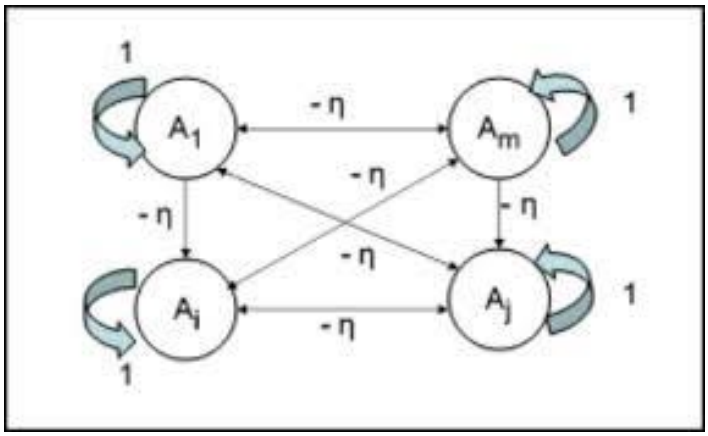
   Jaringan lapisan banyak mempunyai tiga jenis lapisan yakni lapisan *input*, lapisan hidden dan lapisan *output*. Keuntungan Jaringan lapisan banyak ini dibandingkan dengan jaringan lapisan tunggal adalah didalam penyelesaian masalah yang lebih spesifik. Contoh JST: *Backpropagation*



Gambar 2.14 Jaringan dengan Lapisan Banyak

3. Jaringan dengan Lapisan Kompetitif

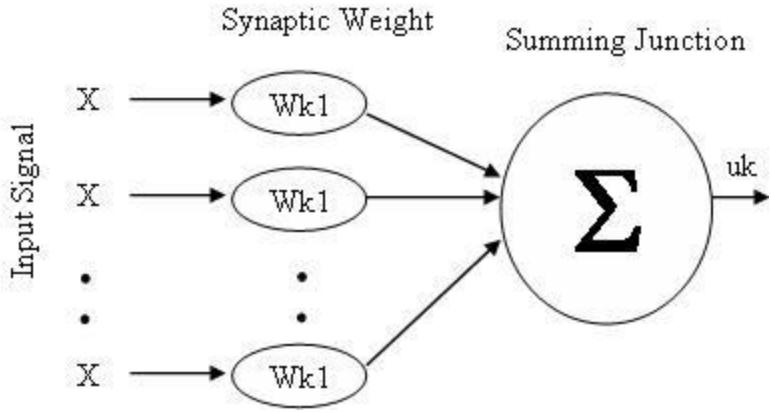
Jaringan ini sudah ada target yang ditentukan dan kelas serta arsitektur yang telah ditetapkan. Sehingga mengakibatkan neuron neuron tersebut saling berkompetisi untuk menjadi yang lebih aktif. Nilai bobot dari tiap-tiap neuron adalah satu, sedangkan neuron lainnya bernilai acak negatif. Dan diharapkan bisa lebih cepat melakukan pelatihan dibanding dengan Backpropagation. Contoh JST; LVQ (*Learning Vector Quantization*).



Gambar 2.15 Jaringan dengan Lapisan Kompetitif

2.3.3 Neuron Buatan

Neuron buatan (*artificial neuron*) dirancang untuk menirukan karakteristik neuron biologis orde pertama. Secara prinsip, serangkauan *input* (masukan) yang masing-masing menggambarkan keluaran yang lain akan dibangkitkan. Setiap masukan dikalikan dengan suatu faktor bobot tertentu yang analog dengan tegangan sinapsis, dan kemudian semua *input* terboboti itu dijumlahkan untuk menentukan tingkat aktivasi suatu neuron [9].

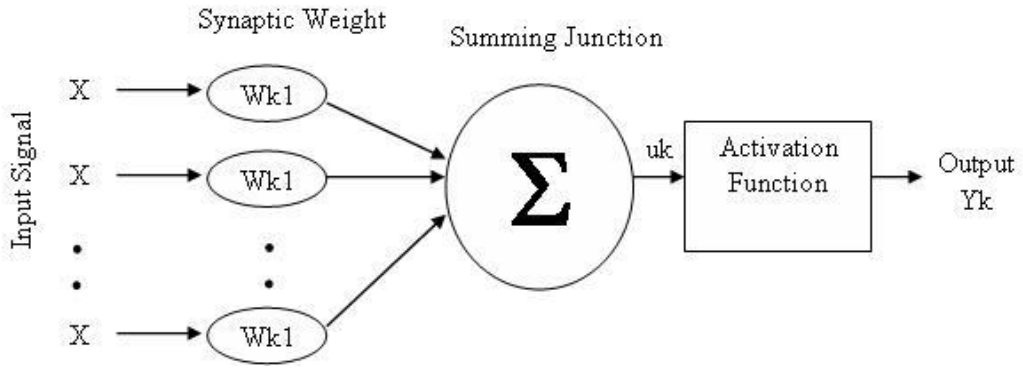


Gambar 2.16 Neuron buatan sederhana

Dari gambar tersebut, serangkaian *input* dengan vektor X yang bersesuaian dengan sinyal yang masuk ke dalam sinapsis neuron biologis. Setiap sinyal dikalikan ke suatu penimbang w_{k1} sebelum masuk ke blok penjumlah S_j. Setiap faktor pembobot bersesuaian dengan tegangan penghubung sinapsis biologis tunggal. Faktor pembobot ini secara Bersama-sama teracusebagai vektor W. Blok penjumlahan yang kurang lebih bersesuaian dengan badan sel biologis, menjumlahkan semua *input* terboboti secara aljabar dan menghasilkan sebuah *output* yang dinotasikan dengan variavel U_j. Dalam unit keluaran variable X dimasukkan dalam suatu dungsi *f* tertentu untuk menghasilkan keluaran akhir. Fungsi *f* yang digunakan merupakan

fungsi linier atau fungsi-fungsi lain yang lebih kompleks. Fungsi yang sering digunakan adalah fungsi sigmoid.

Hasil penjumlahan tersebut akan dibandingkan dengan suatu nilai ambang (threshold) tertentu melalui fungsi aktivasi setiap neuron. Apabila *input* tersebut melewati suatu nilai ambang tertentu maka neuron tersebut akan diaktifkan. Pengaktifan tersebut akan membuat neuron tersebut mengirimkan *output* melalui bobot-bobot *output*-nyake semua neuron yang berhubungan dengannya, demikian seterusnya. Model matematisnya seperti gambar berikut:



Gambar 2.17 Model matematis non-linier dari suatu Neuron

Sinyal x_1, x_2, x_3, …, x_n adalah sinyal *input*; w_{k1}, w_{k2}, …, w_{kn} adalah bobot-bobot sinaptik dari neuron k; u_k adalah *linier combiner output*; fungsi aktivasi; dan y_k adalah sinyal *output* dari neuron. Penggunaan *threshold* pada fungsi aktivasi memberikan pengaruh adanya *offline transformation* terhadap *output* u_k dari *linear combiner.*

Satu sel saraf dapat dimodelkan secara matematis seperti diilustrasikan seperti gambar 2.9. Satu sel saraf terdiri dari tiga bagian, yaitu fungsi penjumlah, fungsi aktivasi dan keluaran. Secara matematis, kita bias menggambarkan sebuah neuron j dengan menuliskan pasangan persamaan sebagai berikut:

$netj = \sum_{j=1}^{p} w_{nj} x_n$ dan $y_k = \varphi(netj - \theta_k)$ (2.5)

Gambar 2.18 dan 2.19 dikatakan sebagai suatu struktur jaringan yang dihitung dan informasi dapat diproses didalamnya. Beberapa konsep yang berhubungan dengan pemrosesan informasi tersebut seperti yang sudah disebutkan diatas, yaitu [9]:

1. *Input*

Setiap *input* dihubungkan ke satu atribut tunggal. Sebagai contoh, pada kasus pemberian keputusan untuk setuju atau tidaknya memberikan pinjaman, beberapa atribut yang digunakan adalah tingkatan, umur dan kepemilikan rumah. Nilai numerik atau representasi suatu atribut merupakan *input* dari jaringan. Beberapa tipe data seperti teks, gambar dan suara dapat juga digunakan untuk mengubah data menjadi *input* yang bermakna dari data simbolik atau ke data skala.

2. *Output*

Output dari jaringan berisi solusi untuk permasalahan. Sebagai contoh dalam aplikasi peminjaman solusi yang diinginkan adalah ya atau tidak. Pada JST ditetapkan nilai numerik seperti nilai satu untuk ya dan nol untuk tidak. Tujuan dari jaringan itu adalah menghitung adanya keluaran. Kerap kali proses terakhir dari *output* dibutuhkan.

3. Bobot (*weights*)

Unsur kunci dari JST adalah bobot. Bobot menunjukkan suatu kekuatan relatif (*relative strength*) atau nilai matematik dari *input* data atau banyaknya koneksi yang memindahkan data dari satu lapisan ke lapisan lainnya. Dengan kata lain, bobot menunjukkan kepentingan relatif dari setiap *input* ke pemrosesan elemen (PE) dan akhirnya menghasilkan *output*. Bobot adalah hal yang penting sekali dimana mereka menyimpan pola pembelajran dan informasi.

4. Fungsi Penjumlahan

Fungsi Penjumlahan menghitung bobot jumlah dari semua elemen *input* yang dimasukkan pada setiap pemrosesan elemennya. Fungsi Penjumlahan merupakan perkalian setiap nilai *input* dengan bobotnya. Apabila total bila jumlah bobotnya dimasukkan sebagai Y maka untuk n *input* dalam satu pemrosesan elemen adalah:

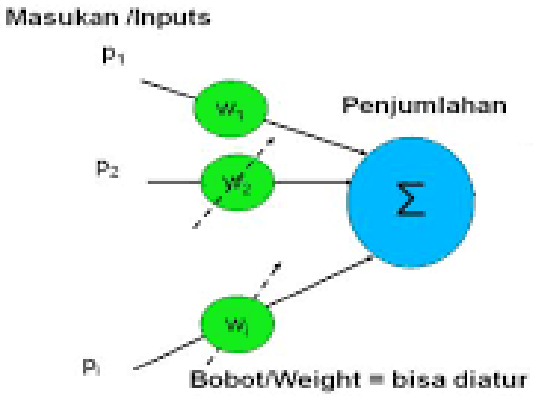
$$Y = \sum_{i=1}^{n} X_i W_i \qquad (2.6)$$

Dan untuk neuron ke-j dari beberapa pemrosesan neuron dalam suatu lapisan maka dapat dituliskan formulanya:

$$Y = \sum_{i=1}^{n} X_i W_{ij} \qquad (2.7)$$

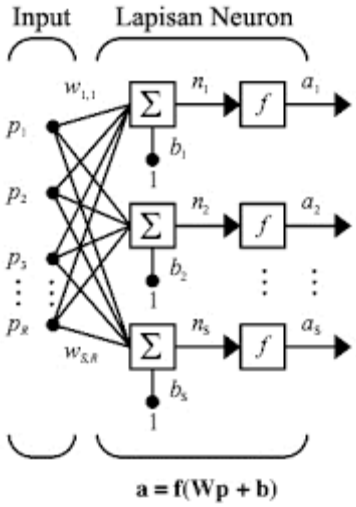
Kedua formula tersebut diilustrasikan dalam gambar berikut ini:

a. Neuron Tunggal



Gambar 2.18 Fungsi penjumlahan untuk neuron tunggal

b. Beberapa Neuron



Gambar 2.19 Fungsi penjumlahan untuk beberapa neuron

5. Fungsi Transfer

Fungsi penjumlahan menghitung simulasi internal atau tingkat aktivasi dari neuron. Berdasarkan tingkat ini, neuron bias menghasilkan suatu *output* dan bias juga tidak. Hubungan antara tingkat aktivasi internal dan *output* dapat berupa linier atau non-linier. Hubungan tersebut dinamakan fungsi transfer. Pemilihan terhadap fungsi yang spesifik berpengaruh kepada operasi jaringan. Fungsi sigmoid sangat popular dan menggunakan fungsi transfer non-*linier*.

2.3.4 Fungsi Aktivasi

Unit terpenting pada struktur JST adalah *input* jaringan dengan menggunakan fungsi scalar ke scalar disebut fungsi aktivasi atau fungsi *threshold* atau fungsi transfer. Hasil dari nilai keluaran adalah aktivasi unit. Beberapa fungsi aktivasi yang digunakan adalah untuk memecahkan masalah *non-linear* [10]. Perilaku JST ditentukan oleh bobor dan *input-input* fungsi aktivasi yang ditetapkan. Beberapa jenis fungsi aktivasi yang sering digunakan dalam JST adalah fungsi undak biner

hard limit, fungsi undak biner *threshold*, fungsi bipolar *symmetric hard limit*, fungsi fungsi bipolar dengan *threshold*, fungsi linear, fungsi *saturating linear*, fungsi symmetric *saturating linear*, fungsi sigmoid biner, dan fungsi sigmoid bipolar [8]. Beberapa contoh fungsi aktivasi yang dipakai adalah:

a. Fungsi undak biner *hard limit*

Fungsi undak biner ini biasanya digunakan oleh lapisan tunggal untuk mengonversi nilai *input* dari suatu variabel yang bernilai kontinu ke suatu nilai *output* biner (0 dan 1). Secara matematis, fungsi undak biner (*hard limit*) dituliskan sebagai berikut:

$$y = \begin{cases} 0, & jika\ x \leq 0 \\ 1, & jika\ x > 0 \end{cases} \qquad (2.8)$$

b. Fungsi undak biner *threshold*

berbeda dengan fungsi undak biner *hard limit*, fungsi undak biner *threshold* menggunakan nilai ambang θ sebagai batasnya. Secara matematis fungsi undak biner *threshold* dituliskan sebagai berikut.

$$y = \begin{cases} 0, & jika\ x < \theta \\ 1, & jika\ x \geq \theta \end{cases} \qquad (2.9)$$

c. Fungsi bipolar *symmetric hard limit*

Fungsi bipolar *symmetric hard limit* mempunyai *output* yang bernilai 1, 0 atau -1. Secara matematis, fungsi *symmetric hard limit* dituliskan sebagai berikut.

$$y = \begin{cases} 1, & jika\ x > 0 \\ 0, & jika\ x = 0 \\ -1, & jika\ x < 0 \end{cases} \qquad (2.10)$$

d. Fungsi bipolar dengan *threshold*

Fungsi bipolar dengan *threshold* mempunyai *output* yang bernilai 1, 0 atau -1 untuk batas nilai ambang θ tertentu. Secara matematis, fungsi bipolar dengan *threshold* dituliskan sebagai berikut.

$$y = \begin{cases} 1, & jika\ x > \theta \\ 0, & jika\ x = \theta \\ -1, & jika\ x < \theta \end{cases} \quad (2.11)$$

e. Fungsi Linear (identitas)

Nilai *input* dan nilai *output* pada fungsi linear adalah sama. Secara matematis, fungsi linear dituliskan sebagai berikut.

$$y = x \quad (2.12)$$

f. Fungsi *saturating linear*

Fungsi ini akan bernilai satu jika *input*-nya lebih dari $^1/_2$. Jika nilai *input* terletak antara $-^1/_2$ dan $^1/_2$ maka *output*-nya akan bernilai sama dengan nilai *input* ditambah $^1/_2$. Jika nilai *input*-nya kurang dari $-^1/_2$ maka fungsi bernilai nol. Secara matematis fungsi *saturating linear* sebagai berikut.

$$y = \begin{cases} 1, & jika\ x \geq 0.5 \\ x + 0.5, & jika -0.5 \leq x \leq 0.5 \\ 0, & jika\ x \leq 0.5 \end{cases} \quad (2.13)$$

g. Fungsi *symmetric saturating linear*

Fungsi ini akan bernilai 1 jika *input*-nya lebih dari 1. Jika nilai *input*-nya terletak diantara -1 dan 1 maka *output*-nya akan bernilai sama dengan nilai *input*-nya. Sedangkan jika *input*-nya kurang dari -1 maka fungsi akan bernilai -1. Secara matematis, fungsi *symmetric saturating linear* dituliskan sebagai berikut.

$$y = \begin{cases} 1, & jika\ x \geq 1 \\ x, & jika -1 \leq x \leq 1 \\ 0, & jika\ x \leq 1 \end{cases} \quad (2.14)$$

h. Fungsi Sigmoid Biner

Biasanya fungsi ini digunakan untuk JST yang dilatih menggunakan metode *backpropagation.* JST yang membutuhkan nilai *output* yang terletak pada interval 0 sampai 1 sering kali menggunakan fungsi sigmoid binerkarena dungsi ini

memiliki nilai pada *range* 0 sampai 1. Secara matematis, fungsi sigmoid biner dituliskan sebagai berikut.

$$y = f(x) = \frac{1}{1+e^{-\sigma x}} \quad (2.15)$$

Dengan: $f'(x) = \sigma f(x)[1 - f(x)]$ (2.16)

i. Fungsi sigmoid bipolar

Output dari fungsi digmoid bipolar memiliki *range* antara 1 sampai -1. Secara matematis, fungsi sigmoid bipolar dirumuskan sebagai berikut.

$$y = f(x) = \frac{1-e^{-x}}{1+e^{-x}} \quad (2.17)$$

Dengan $F'(x) = \frac{\sigma}{2}[1 + f(x)][1 - f(x)]$ (2.18)

Fungsi ini hamper sama dengan fungsi *hyperbolic tangent*. Keduanya memiliki *range* antara -1 sampai 1.

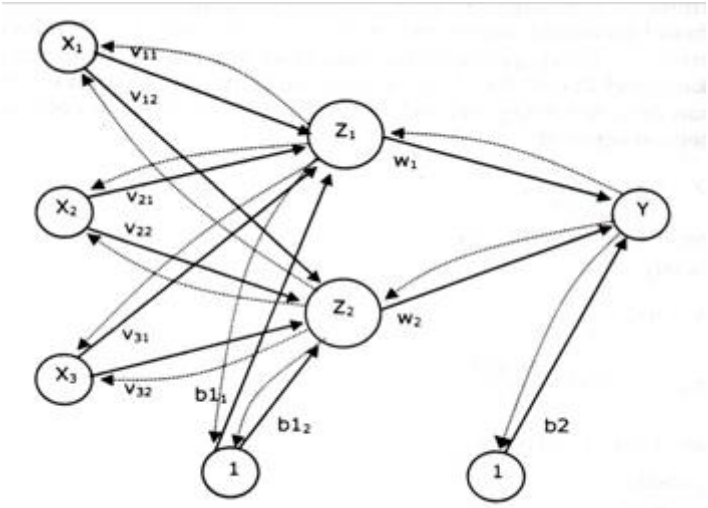
2.3.5 Proses Pembelajaran JST

Dilihat dari memodifikasi bobotnya, pelatihan JST dibagi menjadi dua yaitu [8]:

1. *Supervised learning* (pembelajaran terawasi), suatu *input* diberikan ke jaringan. Jaringan melakukan proses sehingga menghasilkan *output*. *Output* dari jaringan tersebut akan diselisihi dengan target sehingga didapatkan error. Dilatih hingga mencapai error yang ditargetkan sehingga mendapatkan bobot yang sempurna. Kemudian, Jaringan akan memodifikasi bobot sesuai dengan kesalahan tersebut.. Contoh dari tipe ini adalah metode *Backpropagation*, *Perceptron*, Hebbian dan ADALINE.

Supervised Learning didasarkan pada pelatihan sampel data dari sumber data dengan klasifikasi yang sudah ditetapkan. Teknik semacam itu digunakan dalam

Feedforward aatau *Multi Layer Perceptron* (MLP). MLP merupakan ANN turunan dari *perceptron* dengan satu atau lebih *hidden* layer atau bisa disebut juga sebagai JST *backpropagation*. JST *backpropagation* terdiri dari satu layer masukan, satu *hidden* layer dan satu layer keluaran. Sinyal masukan dipropagasikan dengan arah maju pada layer per layer. Contoh arsitektur diberikan pada gambar berikut:



Gambar 2.20 Arsitektur JST *Backpropagation*

Pada gambar 2.22, ada satu *hidden* layer dengan dua neuron dan satu layer keluaran dengan satu neuron. Sebenarnya ada satu layer lagi dalam *backpropagation*, yaitu layer masukan berupa vektor masukan. Akan tetapi, karena dalam layer ini tidak ada komputasi, yang dilakukan hanya meneruskan sinyal/vektor masukan yang diterima ke layer di depannya.

Setiap layer dalam *backpropagation*, masing-masing mempunya fungsi khusus. Layer masukan berfungsi menerima sinyal/vektor masukan dari luar dan mendistribusikannya ke semua neuron dalam *hidden* layer. Layer keluaran menerima sinyal keluaran (atau dengan kata lain stimulus pola) dari *hidden* layer dan memunculkan sinyal/nilai/kelas keluaran dari keseluruhan jaringan.

Dalam neuron hidden dapat mendeteksi fitur-fitur tersembunyi sedangkan bobot dari hidden layer didapat dari antara vektor masukan dengan hidden layer. Dalam penentuan kelas *output* menggunakan fitur-fitur tersembunyi yang didapat dari neuron dalam hidden layer tersebut. *Hidden* layer menyembunyikan keluaran yang diinginkan. Neuron dalam *hidden* layer tidak dapat diamati melalui perilaku mauskan/keluaran jaringan secara keseluruhan. Selain itu, juga tidak ad acara yang jelas untuk mengetahui apa keluaran yang diinginkan oleh *hidden* layer. Dengan kata lain, keluaran yang diinginkan *hidden* layer ditentukan oleh layer itu sendiri.

Pada proses selama pelatihan dilakukan *input* yang sudah dilakukan vektorisasi dimasukkan kedalam jaringan. Kemudian jaringan *input* melewati hidden dan memproses hingga akhirnya mengeluarkan suatu *output Output* tersebut nantinya akan dibandingkan dengan target. Jika *output* tidak sesuai dengan target maka terus dilakukan pelatihan hingga mendapatkan bobot yang memiliki nilai error yang kecil. Tujuan dari pelatihan ini mendapatkan bobot baru yang bisa membuat *output* jaringan sama dengan target yang diharapkan. Dengan pelatihan yang supervise, jaringan diarahkan oleh *input* dan target untuk bisa melatih jaringan hingga dicapai bobot-bobot terbaik. Pelatihan dilakukan dengan memberikan pasangan pola-pola *input* dan *output*.

2. *Unsupervised Learning* (Pembelajaran tidak terawasi), perubahan bobot di dalam proses pelatihannya berdasarkan parameter tertentu dan jaringan dimodifikasi menurut ukuran parameter tersebut. Jaringan ini hanya menggunakan sejumlah

pasangan data masukan tanpa ada contoh keluaran yang diharapkan. Dalam proses penyusunan diri, *cluster* yang dipilih sebagai pemenang adalah *cluster* yang mempunyai vektor bobot paling cocok dengan pola *input*. Contoh model yang masuk dalam kategori ini adalah model kompetitif, Kohonen, *neocognition* dan LVQ (*Learning Vector Quantization*).

Dalam pelatihan tanpa supervisi, tidak ada pembimbing yang digunakan untuk memandu proses pelatihan. Artinya, jaringan hanya diberi *input*, tetapi tidak mendapatkan target yang diinginkan sehingga modifikasi bobot pada jaringan dilakukan menurut parameter-parameter tertentu. Sebagai contoh, pola-pola masukan yang tersedia diklasifikasikan ke dalam kelompok-kelompok yang berbeda [4].

2.3.6 Cara Pelatihan JST *Backpropagation*

Neuron dalam MLP *backpropagation* menggunakan cara yang sama untuk menghitung v seperti pada *perceptron*, formula yang digunakan adalah:

$$v = \sum_{i=1}^{r} x_i w_i \qquad (2.19)$$

Nilai r adalah jumlah masukan (fitur) data masukan, x merupakan nilai fitur/vektor, w adalah bobot vektor. Nilai v tersebut kemudian diaktivasi untuk menghasilkan sinyal keluaran. Fungsi aktivasi yang digunakan adalah sigmoid biner atau sigmoid bipolar. Untuk parameter *slope* (a) bernilai 1, maka fungsi aktivasi sigmoid biner menjadi

$$y = \frac{1}{1+e^{-v}} \qquad (2.20)$$

Dan untuk bipolar menjadi

$$y = \frac{1}{1+e^{-v}} - 1 \qquad (2.21)$$

Secara prosedural, algoritma pelatihan *backpropagation* dijelaskan di bawah ini. Untuk menjelaskan algoritma pelatihan *backpropagation*, digunakan acuan pada yaitu MLP dengan tiga layer (dua *hidden* layer). Untuk indeks x, z, y masing-masing menyatakan indeks neuron dalam layer masukan, *hidden*, dan keluaran. Sinyal masukan x_1, x_2, x_3 dirambatkan dari kiri ke kanan. Sinyal v_{ij} menyatakan bobot untuk koneksi dari *input* layer ke *hidden* layer. Simbol w_{jk} menyatakan bobot dari *hidden* layer ke *output* layer. Asumsi ini menggunakan satu layer tersembunyi (dengan fungsi aktivasi sigmoid biner) (1)[9]:

Langkah 1: Inisialisasi

Melakukan inisialisasi semua bobot pada *hidden* layer dan layer keluaran, lalu terapkan fungsi aktivasi yang digunakan untuk setiap layer. Menetapkan laju pembelajaran. Untuk inisialisasi bobot bias menggunakan bilangan acak dalam jangkauan (-0.5, 0.5) atau menggunakan distribusi seragam dalam jangkauan kecil.

Langkah 2: Aktivasi

Mengaktivasi jaringan dengan menerapkan *input* (x_i, I = 1,2, 3…, n) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit pada lapisan *hidden*.

- Menghitung *output* yang didapatkan dari neuron dalam *hidden* later. Setiap unit *hidden* (Z_j, j = 1,2, 3…, p) menjumlahkan bobot sinyal *input* dengan persamaan berikut,

  $$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \qquad (2.22)$$

  Dan menerapkan fungsi aktivasi untuk menghitung sinyal *output*-nya.

  $$z_j = f(z\_in_j) = \frac{1}{1+e^{z\_in_j}} \qquad (2.23)$$

- Menghitung *output* yang didapatkan dari neuron dalam layer *output*. Setiap unit *output* (Y_k, k = 1,2, 3…, m) menjumlahkan bobot sinyal *input*

  $y\_in_k = w_{0k} + \sum_{i=1}^{p} z_i w_{jk}$ (2.24)

  Dan menerapkan fungsi aktivasi untuk menghitung sinyal *output*-nya:

  $y_k = f(y\_in_k) = \frac{1}{1+e^{-vk}}$ (2.25)

Langkah 3: Perbarui Bobot

Bobot diperbarui pada saat dirambatkan balik dalam JST, *error* yang dikembalikan sesuai dengan arah keluarnya sinyal *output*.

- Setiap unit *output* (Y_k, k = 1,2, 3…, m) menerima pola target yang sesuai dengan pola *input* pelatihan, kemudian hitung *error* dengan persamaan berikut.

  $\delta_k = (t_k - y_k) f'(y\_in_k) = (t_k - y_k) y_k (1 - y_k)$ (2.26)

  f' adalah turunan dari fungsi aktivasi

  kemudian hitung koreksi bobot dengan persamaan berikut.

  $\Delta w_{jk} = \alpha\ \delta_k\ z_j$ (2.27)

  Dan menghitung koreksi bias dengan persamaan berikut,

  $\Delta w_{0k} = \alpha\ \delta_k$ (2.28)

  Sekaligus mengirimkan δ_k ke unit-unit yang ada di lapisan paling kanan

- Setiap unit tersembunyi (Z_j, j = 1,2, 3…, p), menjumlahkan delta *input*-nya (dari unit-unit yang berada pada lapisan di kanannya):

  $\delta\_in_j = \sum_{k=1}^{p} \delta_k\ w_{jk}$ (2.29)

  Karena jaringan hanya memiliki sebuah *output* maka:

  $\delta\_in_j = \delta_k\ w_{j1}$ (2.30)

- Kemudian hitung *error* dengan persamaan berikut.

$$\delta_j = \delta\_in_j f'(y\_in_j) = \delta\_in_j z_j(1 - z_j) \quad (2.31)$$

Kemudian hitung koreksi bobot dengan persamaan berikut.

$$\Delta v_{jk} = \alpha\ \delta_j\ z_i \quad (2.32)$$

Setelah itu, hitung juga koreksi bias dengan persamaan berikut:

$$\Delta v_{0k} = \alpha\ \delta_j \quad (2.33)$$

- Tahap perubahan bobot dan bias

  Setiap unit *output* (Y_k, k = 1,2, 3…, m), dilakukan perubahan bobot dan bias (j = 1,2, 3…, p), dengan persamaan berikut.

  $$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (2.34)$$

  Setiap unit tersembunyi (Z_j, j = 1,2, 3…, p), dilakukan perubahan bobot dan bias (i = 1,2, 3…, n), dengan persamaan berikut.

  $$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \quad (2.35)$$

Langkah 4: Iterasi

Naikan satu untuk iterasi, kembali ke langkah aktivasi dan ulangi proses tersebut sampai kriteria *error* tercapai.

2.3.7 Implementasi JST *Backpropagation* di MATLAB

MATLAB menyediakan dua fungsi yang berkaitan dengan MLP, yaitu newff() untuk membuat JST MLP dan train() untuk melakukan pelatihan pada JST yang dibuat pada fungsi newff(). Perintah yang dipakai untuk membentuk jaringan adalah newff dengan format berikut (1).

net = newff (PR,[S1 S2 …SN], {TF1 TF2 …TFN}, BTF, BLF, PF) (2.36)

Dengan:

Net = jaringan *backpropagation* yang terdiri dari n layer

PR = matriks ordo Rx2 yang berisi nilai minimum dan maksimum R buah elemen masukannya

Si (I = 1,2, …n,) = jumlah unit pada layer ke-I (I = 1,2, …n,)

TFi (I = 1,2, …n,) = fungsi aktivasi yang dipakai pada layer ke-I (I = 1,2, …n,).

Default = tansig (sigmoid bipolar)

BTF = fungsi pelatihan jaringan. Defaultnya = traingdc

BLF = fungsi perubahan bobot/bias. Default = learngdm

PF = fungsi perhitungan *error*. Default = mse

Beberapa fungsi aktivasi yang dipakai matlab dalam pelatihan *backpropagation* adalah:

- Tansig (sigmoid bipolar). F(net) = $\frac{2}{1+e^{-net}} - 1$. Fungsi ini adalah default yang dipakai. Fungsi sigmoid bipolar memiliki *range* [1,-1]
- Logsig (sigmoid biner). F(net) = $\frac{1}{1+e^{-net}}$. Fungsi sigmoid biner memiliki bentuk serupa dengan sigmoid bipolar, hanya *range* nya adalah [0,1]
- Purelin (Fungsi identitas) F(net) = net

Pelatihan di matlab menggunakan fungsi yang bervariasi dengan tujuan untuk akselerasi pelatihan. Fungsi umumnya yang digunakan matlab adalah traingdx. Dalam fungsi ini, perubahan didalam bobot dibawa keluar dengan menambahkan momentum dan memperhatikan perubahan didalam bobot dalam pengulangan sebelumnya. Disamping itu, rate pemahaman (learnin rate = α) adalah bukan konstanta yang tetap, tetapi dapat berubah selama pengulangan.

Secara umum, pelatihan backpropagation di matlab dijalankan didalam batch. Semua pola dimasukkan pertama, kemudian bobot diubah. Dalam pelatihan berkelompok, semua *input* harus ditempatkan dalam sebuah matrix. Ini adalah sebuah perbedaan kecil dari perceptron yang peltihannya dapat menggunakan pelatihan berkelompak atau pelatihan yang terpola.

Setiap kali bentukan sebuah jaringan *backpropagation*, matlab akan memberikan bobot dan inisial bias dengan nomor acak yang kecil. Bobot dan bias ini akan mengubah setiap waktunya dari bentukan jaringan. Tetapi jika kamu mau untuk memberikan sebuah bobot tertentu, kemudian memberikan nilai ke net.IW, net.LW dan net.b.

Perhatikan perbedaan antara net.IW dan net.LW. net.IW {j,i} digunakan sebagai variabel untuk menyimpan bobot dari layer *input* l ke unit tersembunyi (*hidden*), layer j. Karena didalam backpropagation , unit *input* hanya terhubung dengan layer tersembunyi yang palinng rendah. Bobot disimpan didalam net.IW {1,1}. Sebaliknya, net.LW {k,j} digunakan untuk menyimpan bobot dari unit yang ada didalam unit *hidden*-j ke unit *hidden*-k. Sebagai contoh, net LW {2,1} adalah penyimpanan bobot dari *hidden* paling bawah (*hidden* 1) ke *hidden* diatasnya (*hidden* 2)(1).

Ada beberapa parameter pelatihan yang dapat kita atur sebelum pelatihan dilakukan. Dengan memberi nilai yang kita inginkan pada paremeter-parameter tersebut kita dapat memperoleh hasil yang lebih optimal (1).

Net.trainParamshow : digunakan untuk menunjukkan perubahan dari frekuensi (Default: setiap 25 epoch).

Net.trainParam.epochs: digunakan untuk menenukan batas nilai mse sehingga pengulangan diberhentikan.

Net.trainParam.goal : digunakan untuk menentukan batas nilai mse sehingga pengulangan diberhentikan. Pengulangan akan berhenti jika mse kurang dari batas *limit* yang spesifik didalam net.traniParam.goal atau nomor dari epoch mencapai batas spesifik dalam net.trainParam.epoch.

Net.trainParam.lr : digunakan untuk menentukan *rate* dari pemahaman (α = *learning rate*). Biasanya secara otomatis dari MATLAB akan memberikan nilai 0.01. Semakin besar nilai α-nya maka proses pelatihan yang semakin cepat akan terjadi. Tetapi jika nilai α-nya terlalu besar, algoritmanya menjadi tidak stabil dan mencapai titik minimum local.

Net.trainParam.time : digunakan untuk membatasi durasi pelatihan (dalam *seconds*). Pelatihan akan diberhentikan jika durasi melebihi nilai yang spesifik dari net.trainParam.time.

Pelatihan akan lebih cepat apabila laju pemahaman dapat diubah-ubah besarnya selama proses pelatihan. Jika *error* sekarang lebih besar dibandingkan *error* sebelumnya, maka laju pemahaman diturunkan. Jika sebaliknya, maka laju pemahaman diperbesar. Dengan demikian laju pemahaman dapat dibuat sebesar-besarnya dengan tetap mempertahankan kestabilan proses. Dalam matlab

penggunaan dari variabel laju pemahaman diselesaikan dengan menggunakan ‘traingda’ dan ‘traingdx’ diatas parameterfungsi pelatihan newff [8].

2.3.6 Pelatihan Standar *Backpropagation*

Pelatihan standar *backpropagation* terdiri dari 3 fase. Fase pertama adalah fase maju (*the advanced phase*). Pola *input* dikalkulasikan maju dari layer *input* ke layer *output* menggunakan fungsi aktivasi yang spesifik. Fase kedua adalah fase mundur (*the backward phase*). Perbedaan antara *output* dan target yang ditentukan adalah *error* yang terjadi. *Error* dipropagasikan mundur, dimulai dari garis yang dihubungkan secara langsung ke unit *output*. Fase ketiga adalah modifikasi dari bobot-bobot untuk mengurangi *error* yang terjadi [11].

Fase I: Propagasi Maju

Selama propagasi maju, sinyal *input* (=xi) dipropagasikan ke layer tersembunyi menggunakan fungsi aktivasi yang spesifik. *Output* dari tiap unit layer tersembunyi (=zj) dipropagasikan maju lagi ke layer tersembunyi diatasnya menggunakan fungsi aktivasi yang spesifik. Dan seterusnya sampai menghasilkan jaringan *output* (=yk). Selanjutnya, Jaringan *ouput* (=yk) dibandingkan ke target untuk dicapai. Perbedaan tk – yk adalah *error* yang terjadi. Jika *error* ini lebih kecil dari batas toleransi yang spesifik, kemudian pengulangan diberhentikan. Bagaimanapun, Jika *error* masih lebih baik dari batas toleransi, bobot dari setiap garis dalam jaringan akan dimodifikasi untuk mengurangi *error*.

Fase II: Propagasi Maju

Berdasarkan dari *error* tk - yk, dihitung faktor δ k (k = 1, 2, …, m) yang digunakan untuk menyalurkan *error* di unit yk ke semua unit tersembunyi yang menghubungkan dengan yk. Δk juga digunakan untuk perubahan bobot garis yang terhubung dengan unit keluaran. Dengan jalan yang sama, dihitung faktor δ j di setiap unit di dalam layer tersembunyi di dalam bawah layer. Demikian selanjutnya sampai keseluruhan faktor δ yang ada didalam unit tersembunyi yang secara langsung dihubungkan ke unit *input*.

Fase III: Perubahan bobot

Setelah semua faktor δ dihitung, bobot dari semua garis dimodifikasi secara bersamaan. Perubahan bobot dari sebuah garis didasarkan pada faktor δ neuron di layer atasnya. Sebagai contoh, perubahan didalam bobot dari garis yang menuju ke layer *output* didasarkan atas faktor δ yang ada di unit *output*. Ketiga fase ini dilakukan berulang-ulang dan terus menerus. Kondisi penghentian fase biasanya lebih sering menggunakan capaian maks iterasi atau toleransi kesalahan. Jika sudah mencapai maksimum iterasi, maka iterasi akan dihentikan, atau nilai kesalahan tidak melebihi *tolerance limit* (batas toleransi) yang *default* atau yang sudah ditentukan.

III. METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian dilakukan di Laboratorium Terpadu Teknik Elektro Universitas Lampung. Penelitian dimulai pada bulan Juni 2017 sampai dengan bulan Juli 2018.

3.2 Alat dan Bahan

Kebutuhan alat dan bahan untuk melakukan penelitian ini adalah sebagai berikut:

a. Kebutuhan *Hardware*

Sebuah *netbook* untuk melakukan perancangan dan pembangunan sistem dengan spesifikasi sebagai berikut:

Processor : Intel(R) Celeron(R) CPU 847 @ 1.10 Ghz

RAM : 4 GB

b. Kebutuhan *Software*

Software digunakan untuk melakukan perancangan dan pembuatan sistem.

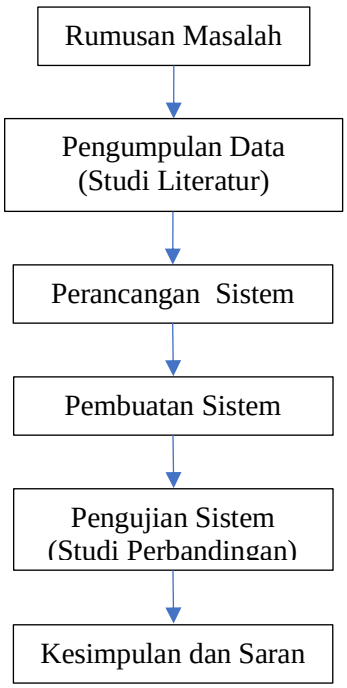
Adapim *software* tersebut adalah:

- MATLAB7.8.0 R2009a untuk penulisan *source code* program
- Ms. Office 2016 untuk membuat dokumentasi dan laporan penelitian

c. Kebutuhan Data

Data yang diambil berupa aksara Lampung dari penelitian yang dilakukan oleh Hendri Setiawan.

3.3. Tahap - tahap dalam penelitian



Gambar 3.1 Tahapan Penelitian

3.3.1 Rumusan Masalah

Rumusan masalah merupakan dasar pemikiran dan acuan untuk penelitian ini. Rumusan masalah dalam penelitian ini mengacu pada latar belakang penelitian yang telah dijelaskan pada bab satu.

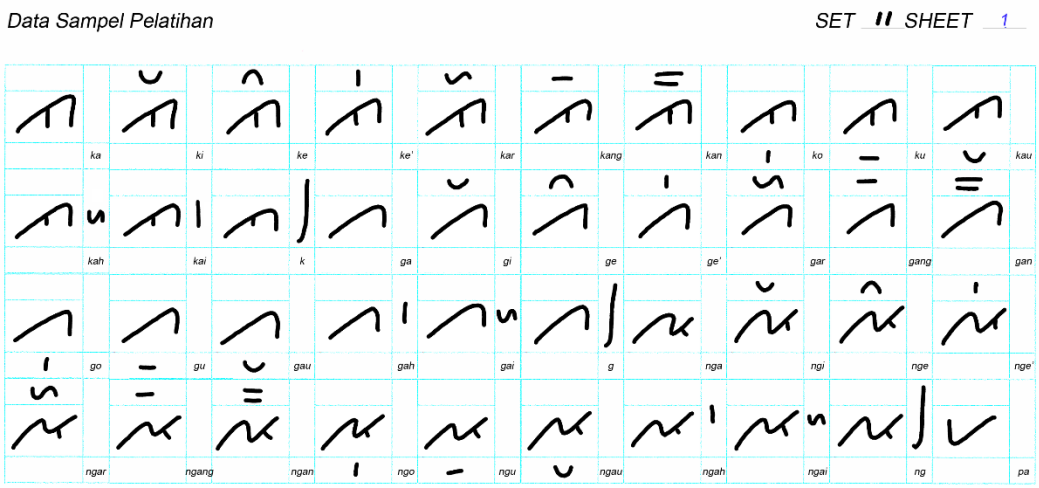
3.3.2 Pengumpulan Data

Tahap ini dilakukan sebagai langkah untuk mendalami masalah dan memberi solusi terhadap masalah yang dirumuskan. Studi literatur (kajian pustaka) merupakan penelusuran literatur yang bersumber dari buku, media, jurnal ataupun hasil dari

penelitian orang lain yang bertujuan untuk menyusun dasar teori yang kita gunakan untuk penelitian. Studi literatur dilakukan untuk mendapatkan informasi mengenai hal berikut:

- Aksara Lampung
- Pengolahan Citra
- Jaringan Saraf Tiruan (JST)
- Pemrograman MATLAB

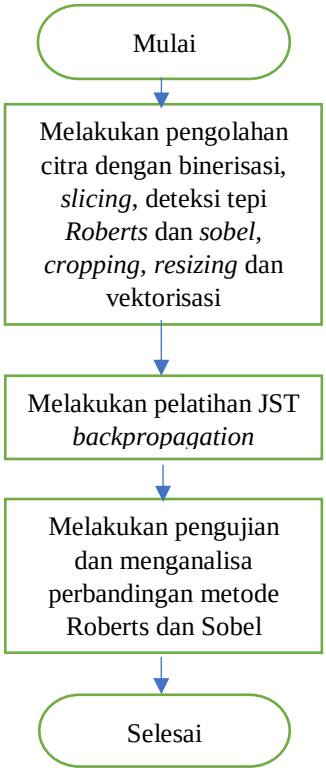
Pengumpulan data sampel telah dikumpulkan oleh peneliti sebelumnya yaitu Hendri Setiawan. Penelitian data sampel dibutuhkan untuk melatih JST. Data sampel berupa data karakter aksara Lampung yang terdiri dari 10 set yang berisi karakter aksara tunggal (tanpa anak huruf) dan karakter aksara campuran (dengan anak huruf). Dalam aksara Lampung terdapat 20 karakter induk huruf atau karakter tunggal dan 12 anak huruf (6 anak huruf atas, 3 anak huruf bawah dan 3 anak huruf depan). Pada tiap set data pelatihan, satu karakter aksara ditulis sebanyak 13 kali yaitu satu karakter tanpa anak huruf dan 12 lainnya dengan anak huruf.



Gambar 3.2 Contoh data karakter aksara Lampung

3.3.3 Perancangan Sistem

Dalam penelitian ini terdapat dua jenis perangkat yang digunakan dalam sistem, yaitu perangkat keras dan perangkat lunak. Perangkat keras yang digunakan adalah *netbook*. Sedangkan untuk perangkat lunak, MATLAB sebagai perangkat pemrograman untuk pengolahan citra dan pelatihan JST. Perangkat yang dibuat dalam penelitian ini adalah perangkat lunak pada perangkat pelatihan dan pengujian. Sistem yang dibuat pada perangkat tersebut untuk studi perbandingan dengan melihat hasil *error* setelah dilakukan pengujian.

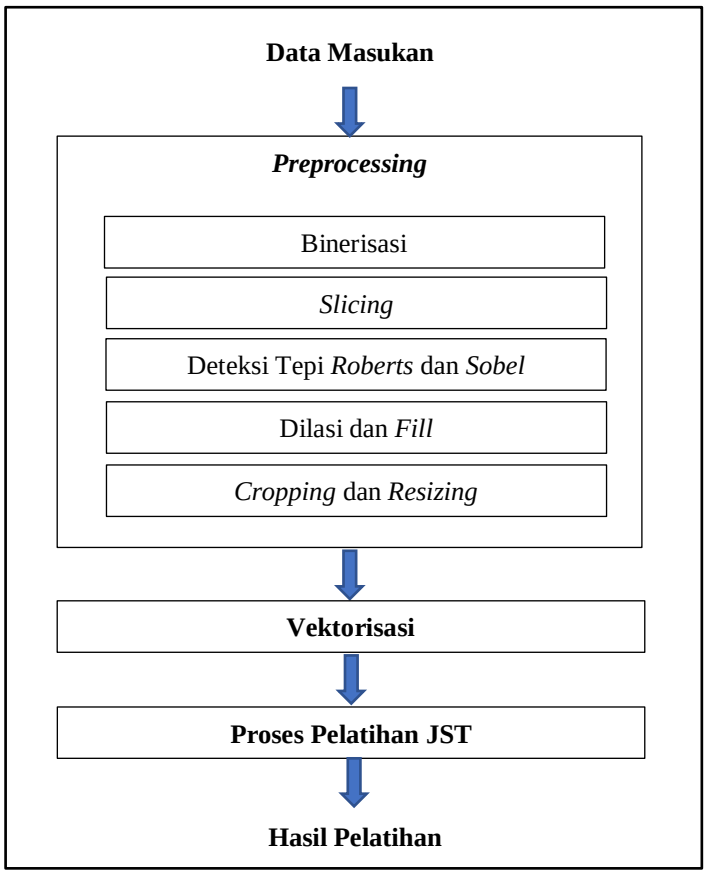


Gambar 3.3 Diagram Alir Perancangan Sistem

3.3.4 Pembuatan Sistem

Pada tahap pembuatan sistem ini yaitu pembuatan perangkat pelatihan dengan metode deteksi *Roberts* dan *sobel*. Hasil pelatihan tersebut akan dilihat metode terbaik dalam pengenalan karakter aksara Lampung.

Pada perangkat pelatihan data masukan yang digunakan adalah angket data pelatihan yang terdiri dari 10 set angket data. Penentuan jumlah set sebanyak 10 set dimaksudkan untuk memudahkan sistem melakukan pengenalan karakter aksara. Apabila data masukan terlalu variatif sedangkan jumlah karakter yang akan dikenali cukup banyak (20 karakter aksara) maka akan menyulitkan sistem untuk melakukan pengenalan. Gambar 3.4 menunjukkan proses kerja yang terjadi dalam perangkat pelatihan.



Gambar 3.4 Proses kerja perangkat pelatihan

Dalam proses kerja perangkat pelatihan, data masukan akan melewati beberapa tahapan. Tahap awal adalah tahap pengolahan citra yang meliputi binerisasi, *slicing* (pemotongan), deteksi tepi *Roberts* dan *Sobel*, dilasi dan *fill*, *cropping* dan *resizing* dan vektorisasi.

Binerisasi asalah proses untuk mengubah citra warna menjadi citra hitam putih atau citra biner dengan nilai *threshold* 0.05. Proses binerisasi ini juga dilakukan untuk menghilangkan garis bantu yang ada pada angket data pelatihan. Berikut ini adalah program untuk mengubah citra warna menjadi citra biner pada matlab.

```
I = imread('D:\final task\reference\image\*.jpg');
binaryimage = im2bw (I);
```

Setelah citra diubah ke dalam citra biner, selanjutnya dilakukan pemotongan terhadap citra tersebut untuk mendapatkan karakter induk huruf maupun anak huruf. Sebelum dilakukan pemotongan, terlebih dahulu ditentukan batas-batas pemotongan yang akan dilakukan. Batas pemotongan ditentukan dengan memperhatikan garis bantu yang terdapat pada angket data pelatihan. Berikut ini adalah batas-batas pemotongan citra pada perangkat pelatihan.

```
% batas pemotongan karakter induk huruf
startX = 40;                            % Titik awal X
startY = 230;                           % Titik awal Y
width = 185;                            % Panjang citra hasil slicing
height = 123;                           % Tinggi citra hasil slicing
deltaX = 295 - startX;                  % Jarak antar citra di titik X
deltaY = 488 - startY;                  % Jarak antar citra di titik Y

% batas pemotongan karakter anak huruf atas
startX = 29;                            % Titik awal X
startY = 165;                           % Titik awal Y
width = 180;                            % Panjang citra hasil slicing
height = 67;                            % Tinggi citra hasil slicing
deltaX = 287 - startX;                  % Jarak antar citra di titik X
deltaY = 423 - startY;                  % Jarak antar citra di titik Y
```

```
% batas pemotongan karakter anak huruf bawah
startX = 29;                    % Titik awal X
startY = 361;                   % Titik awal Y
width = 196;                    % Panjang citra hasil slicing
height = 67;                    % Tinggi citra hasil slicing
deltaX = 285 - startX;          % Jarak antar citra di titik X
deltaY = 619 - startY;          % Jarak antar citra di titik Y

% batas pemotongan karakter anak huruf depan
startX = 225;                   % Titik awal X
startY = 165;                   % Titik awal Y
width = 65;                     % Panjang citra hasil slicing
height = 191;                   % Tinggi citra hasil slicing
deltaX = 481 - startX;          % Jarak antar citra di titik X
deltaY = 423 - startY;          % Jarak antar citra di titik Y
```

Setelah dilakukan pemotongan citra maka langkah selanjutnya adalah mencari tepi batas citra pada karakter aksara dengan menggunakan metode deteksi tepi. Pada penelitian ini menggunakan dua metode deteksi tepi yaitu metode *Roberts* dan *Sobel*, Kemudian dilanjutkan dengan penebalan karakter dan *fill*.

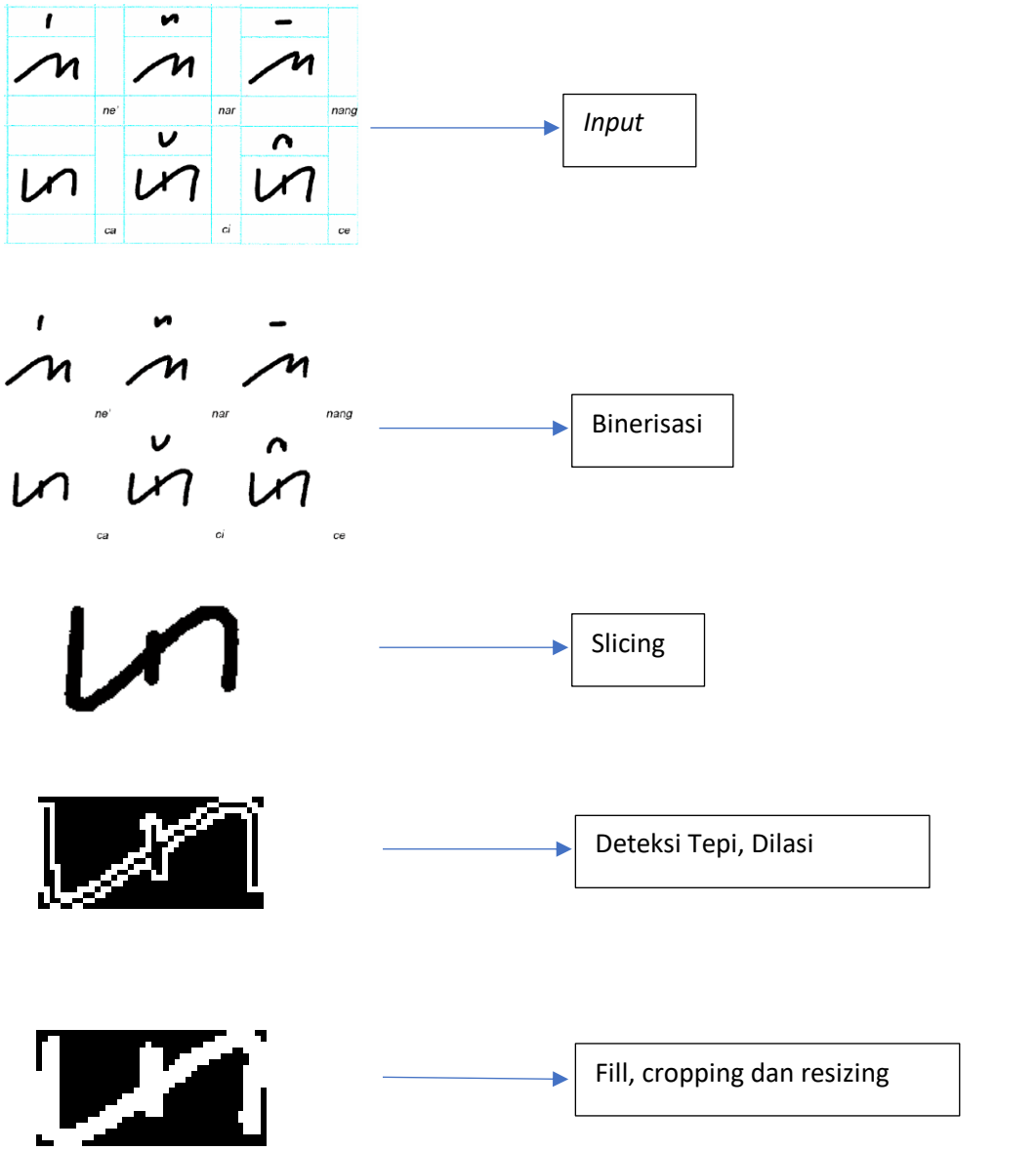
```
% deteksi tepi, dilasi dan fill
ed = edge (I,'roberts');
se = (strel('square',4);
A = imdilate (ed,se);
B = imfill (A,'holes');
```

Proses *cropping* akan menghasilkan sebuah citra yang hanya berisi data karakter saja, sedangkan data di sekeliling data karakter dihilangkan. Berikut ini merupakan kode program untuk melakukan proses *cropping*.

```
if m == size(aksara, 2)             % Jika m = character terakhir
    O = B;
  else
    mmY = [size(B, 1) 0];           % Jumlah baris dalam matriks I
    mmX = [size(B, 2) 0];           % Jumlah kolom dalam matriks I
    for y = 1:size(B, 1)            % Loop baris (dari baris ke 1 s.d ke baris terakhir)
      for x = 1:size(B, 2)          % Loop kolom (dari kolom ke 1 s.d ke kolom terakhir)
        if B(y, x) == 1
          if y < mmY(1)             % jika y kurang dari jumlah baris
            mmY(1) = y;
          end
          if y > mmY(2)             % jika y lebih dari 0
            mmY(2) = y;
          end
          if x < mmX(1)             % jika x kurang dari jumlah kolom
            mmX(1) = x;
          end
          if x > mmX(2)             % jika x lebih dari 0
            mmX(2) = x;
          end
        end
    end
    O = B(mmY(1):mmY(2), mmX(1):mmX(2)); % Citra hasil cropping
  end
```

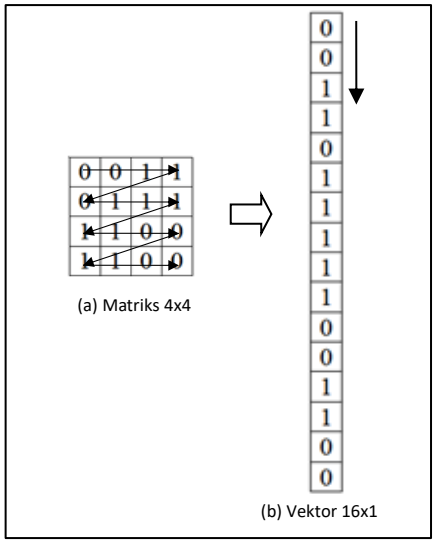
Langkah terakhir pada *preprocessing* adalah melakukan perubahan terhadap ukuran citra atau *resizing*. Pada proses ini citra disamakan ukurannya menjadi 20x40 piksel. Dengan memperkecil ukuran citra, maka data yang masuk ke dalam JST dapat dibatasi sehingga dapat menyederhanakan sisi masukan.

```
%Proses resizing citra
O = imresize(O, [20 40]);          % Rubah ukuran citra menjadi 20 x 40
```



Gambar 3.5 Proses pengolahan citra aksara ‘ca’

Karena data yang diproses di JST adalah data vektor, maka data citra yang sudah dilakukan *resizing* harus diubah ke dalam bentuk matriks vektor. Matriks vektor merupakan matriks yang terdiri dari satu baris (1xn) dan satu kolom (nx1). Matriks vektor pada penelitian ini merupakan data matriks yang terdiri dari 800 baris dengan satu kolom (800x1). Untuk mendapatkan data vektor perhatikan gambar 3.6



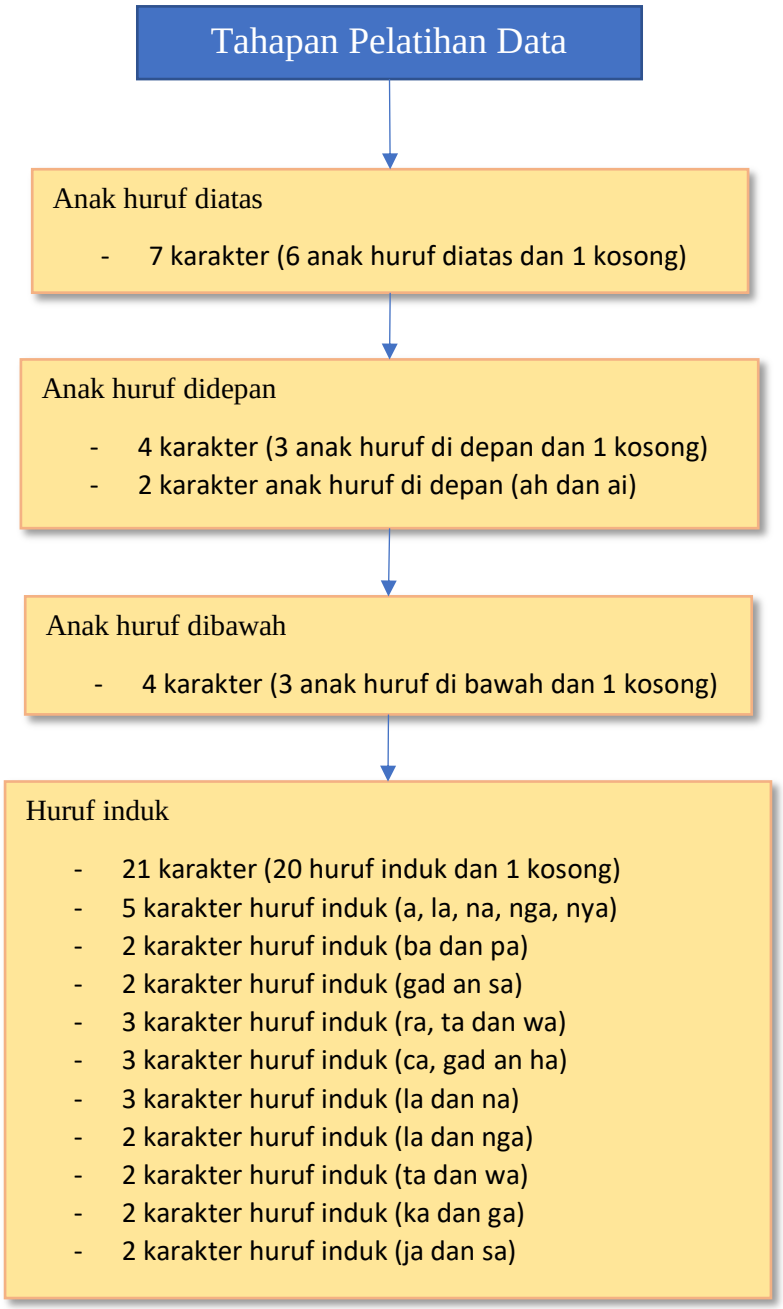
Gambar 3.6 Contoh vektorisasi Citra

Seperti yang terlihat pada gambar, data vektor bisa didapatkan dari matriks yang berukuran 4x4. Cara untuk mengubah matriks 4x4 pada gambar 3.6(a) ke dalam bentuk vektor yaitu diawali dengan langkah mengambil nilai pada baris pertama, kemudian dilanjutkan dengan mengambil nilai pada baris kedua, baris ketiga dan baris keempat. Kalau misalkan ada 800 baris berarti dilanjutkan sampai 800 baris. Nilai-nilai tersebut dibuat dalam satu garis lurus dengan arah dari atas ke bawah. Sehingga didapat matriks vektor dengan ukuran 16x1 seperti pada gambar 3.6(b).

Proses selanjutnya adalah melakukan pelatihan menggunakan jaringan saraf tiruan dengan algoritma *backpropagation*. Dalam pelatihan jaringan terdapat parameter-

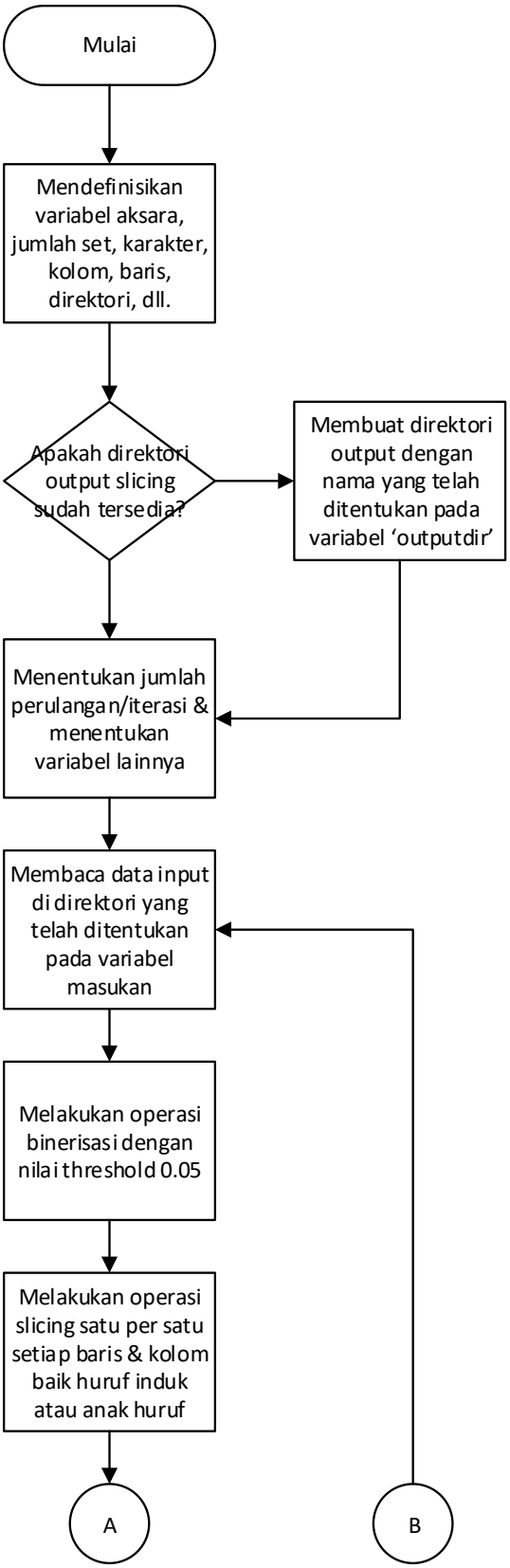
parameter yang dibutuhkan oleh jaringan supaya didapatkan hasil pelatihan yang baik. Parameter-parameter yang dibutuhkan tersebut adalah seperti bobot awal, bias awal, parameter epoch, parameter goal. Parameter epoch dan parameter goal merupakan syarat yang harus dicapai untuk menghentikan proses pelatihan. Proses pelatihan akan berhenti ketika salah satu diantaranya telah tercapai.

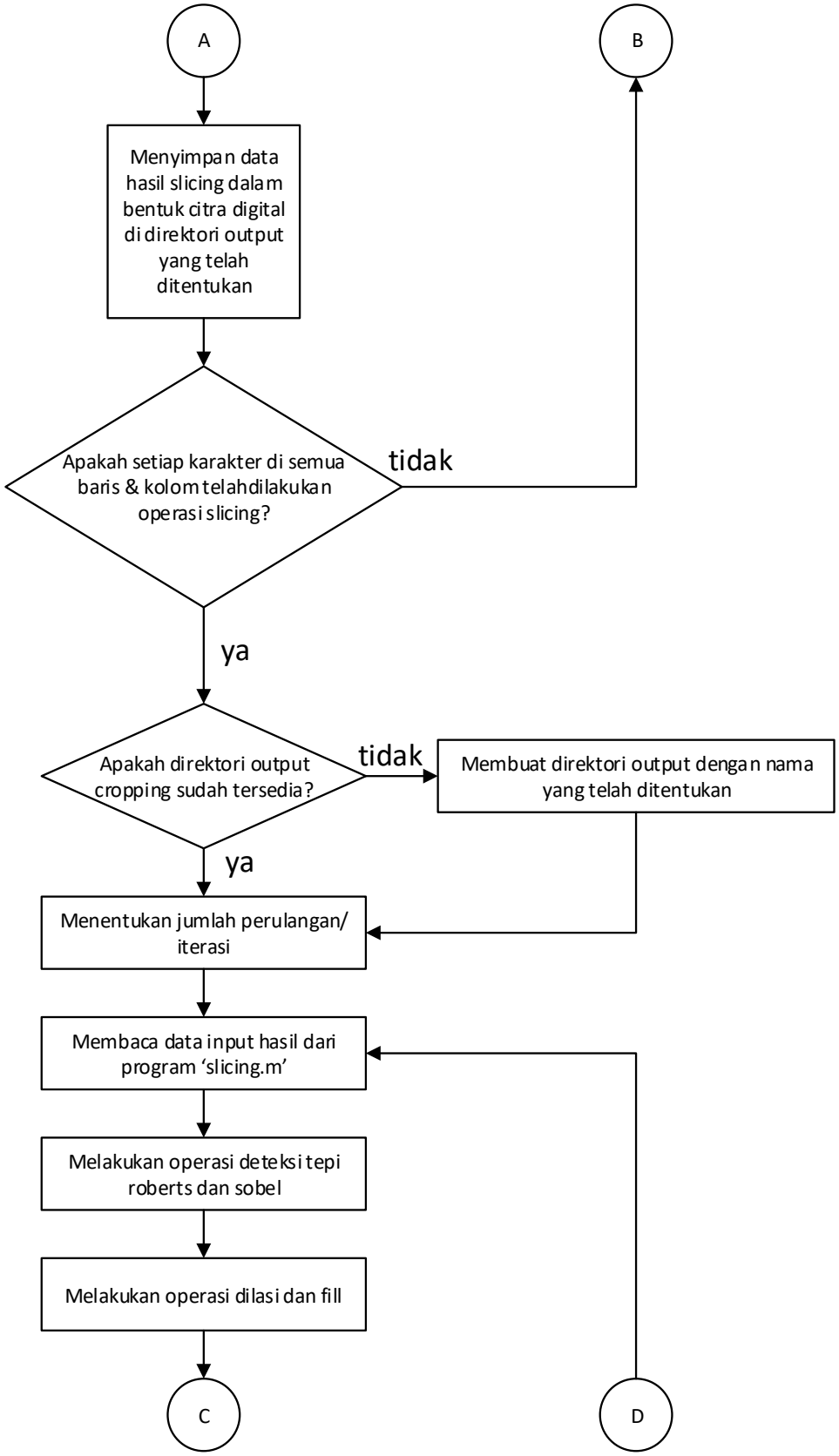
Dalam aksara Lampung dikarenakan terdapat beberapa karakter aksara yang mempunyai bentuk hampir serupa maka diperlukan proses *training* atau pelatihan bertingkat (ganda) untuk melatih karakter-karakter tersebut. Dari gambar 3.7, proses tersebut terdiri dari: tahapan penelitian 21 karakter huruf induk (20 karakter huruf induk dan 1 karakter kosong), 7 karakter anak huruf di atas (6 karakter anak huruf di atas dan 1 karakter kosong), 4 karakter anak huruf di bawah (3 karakter anak huruf di bawah dan 1 karakter kosong) dan 4 karakter anak huruf di depan (3 karakter anak huruf di depan dan 1 karakter kosong), 5 karakter huruf induk (a, la, na, nga, nya), 2 karakter huruf induk (ga & sa), 3 karakter huruf induk (ra, ta & wa), 3 karakter huruf induk (ca, gha & ha), 2 karakter huruf induk (ka & ga), dan 2 karakter huruf induk (ja & sa) yang kesemuanya menggunakan *input* data sebanyak 10 set angket data. Hasil yang diperoleh dari JST adalah bobot dan bias.

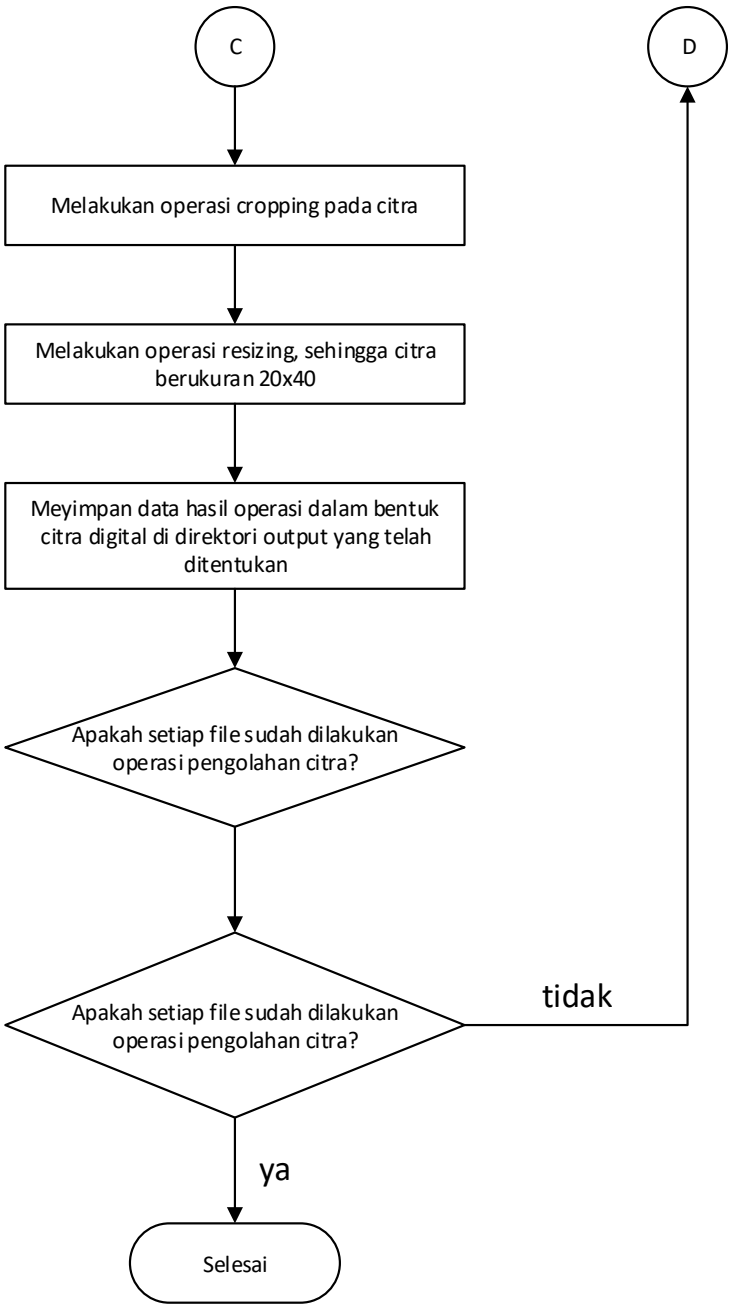


Gambar 3.7 Tahapan pelatihan bertingkat (ganda)

- Program Pengolahan Citra




A
B
Menyimpan data hasil slicing dalam bentuk citra digital di direktori output yang telah ditentukan
Apakah setiap karakter di semua baris & kolom telahdilakukan operasi slicing?
tidak
ya
Apakah direktori output cropping sudah tersedia?
tidak
Membuat direktori output dengan nama yang telah ditentukan
ya
Menentukan jumlah perulangan/ iterasi
Membaca data input hasil dari program 'slicing.m'
Melakukan operasi deteksi tepi roberts dan sobel
Melakukan operasi dilasi dan fill
C
D



Gambar 3.8 Flowchart proses kerja program pengolahan citra

Gambar 3.8 menunjukkan diagram alir proses kerja program pengolahan citra. Dari gambar tersebut tahapan yang terjadi pada proses ini diantaranya adalah:

1. Program mendefinisikan variabel-variabel yang dibutuhkan, seperti jumlah data, batas antar karakter, panjang, lebar dan lain-lain.

2. Program akan membuat direktori penyimpanan untuk hasil *slicing* apabila belum tersedia.

3. Program akan menentukan jumlah iterasi atau perulangan proses *slicing* yang akan dilakukan.

4. Program akan melakukan operasi binerisasi terhadap data masukan dengan nilai *threshold* sebesar 0.05.

5. Program akan melakukan operasi *slicing* satu per satu pada tiap baris dan kolom sebanyak perulangan yang telah ditentukan kemudian menyimpan hasil *slicing* ke dalam direktori yang telah dibuat sebelumnya.

6. Program akan membuat direktori keluaran untuk hasil *cropping*, melakukan pembacaan terhadap citra hasil *slicing* serta menentukan jumlah iterasi yang akan dilakukan pada proses *cropping*.

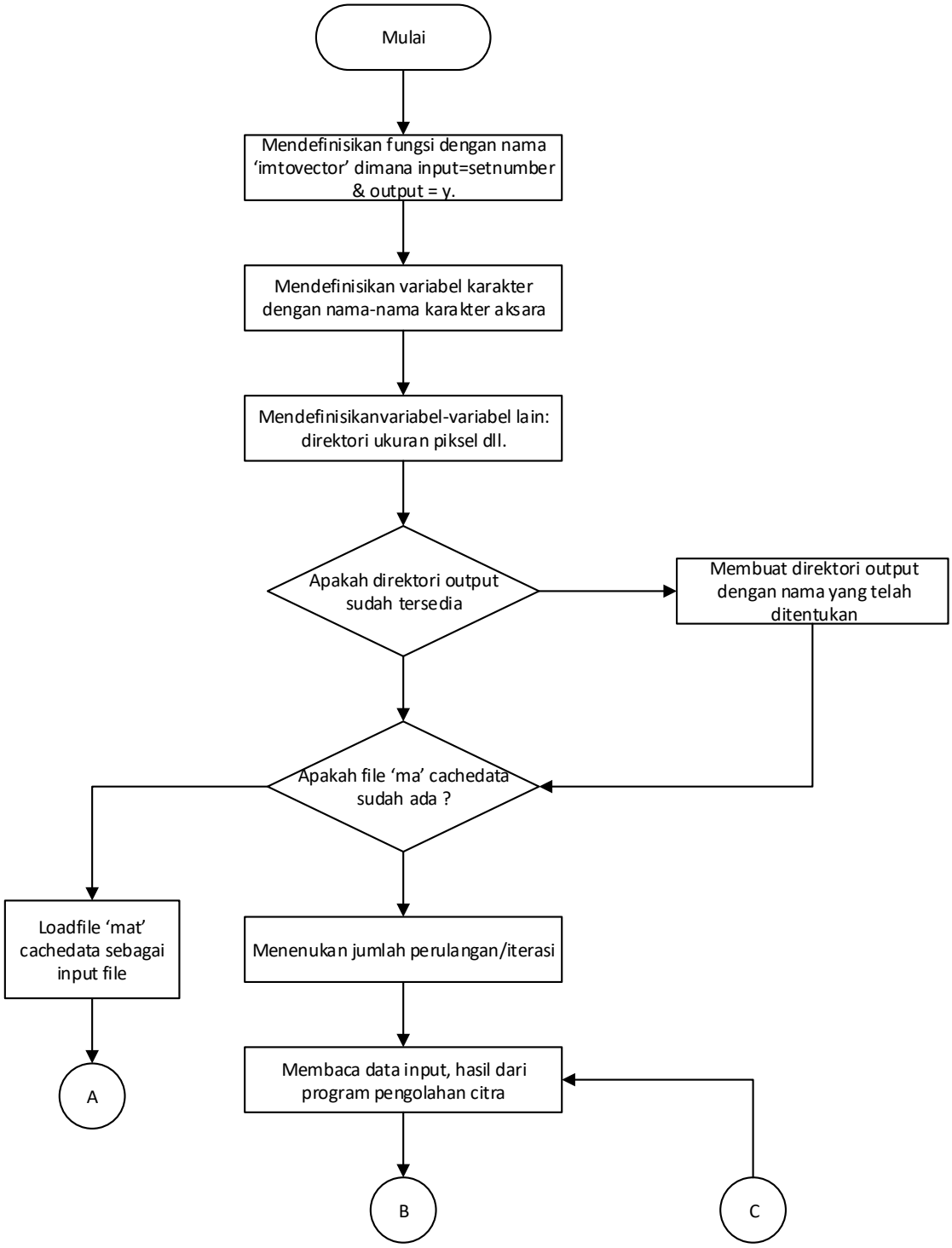
7. Program akan melakukan operasi deteksi tepi *Roberts* dan *sobel*, dilasi dan *fill*

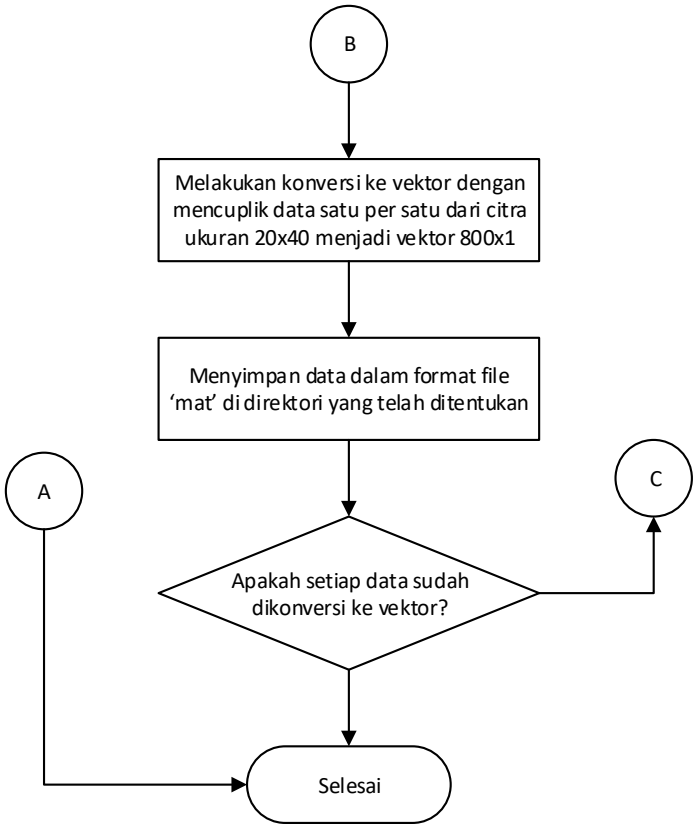
8. Program akan menyimpan hasil akhir berupa citra berukuran 20x40 piksel ke dalam direktori yang telah dibuat sebelumnya.

Beberapa tahapan untuk pembuatan program pengolahan citra pada Matlab terlampir pada lampiran B.

- Pembuatan Program Vektorisasi Citra

Program vektorisasi citra dibuat untuk melakukan perubahan data citra hasil *cropping* ke dalam bentuk vektor. Program vektorisasi adalah sebuah fungsi yang akan bekerja jika digunakan oleh program lain. Dalam hal ini, program fungsi vektorisasi citra digunakan oleh program jaringan saraf tiruan. Apabila fungsi vektorisasi citra belum dibuat, maka program jaringan saraf tiruan tidak akan bisa dieksekusi. Diagram alir proses kerja program vektorisasi citra ditunjukkan pada Gambar 3.9.


Mulai
Mendefinisikan fungsi dengan nama 'imtovector' dimana input=setnumber & output = y.
Mendefinisikan variabel karakter dengan nama-nama karakter aksara
Mendefinisikanvariabel-variabel lain: direktori ukuran piksel dll.
Apakah direktori output sudah tersedia
Membuat direktori output dengan nama yang telah ditentukan
Apakah file 'ma' cachedata sudah ada ?
Loadfile 'mat' cachedata sebagai input file
Menenukan jumlah perulangan/iterasi
A
Membaca data input, hasil dari program pengolahan citra
B
C



Gambar 3.9 Flowchart proses kerja program vektorisasi citra

Dari Gambar 3.9 proses kerja program vektorisasi citra terdiri dari beberapa tahapan kerja diantaranya:

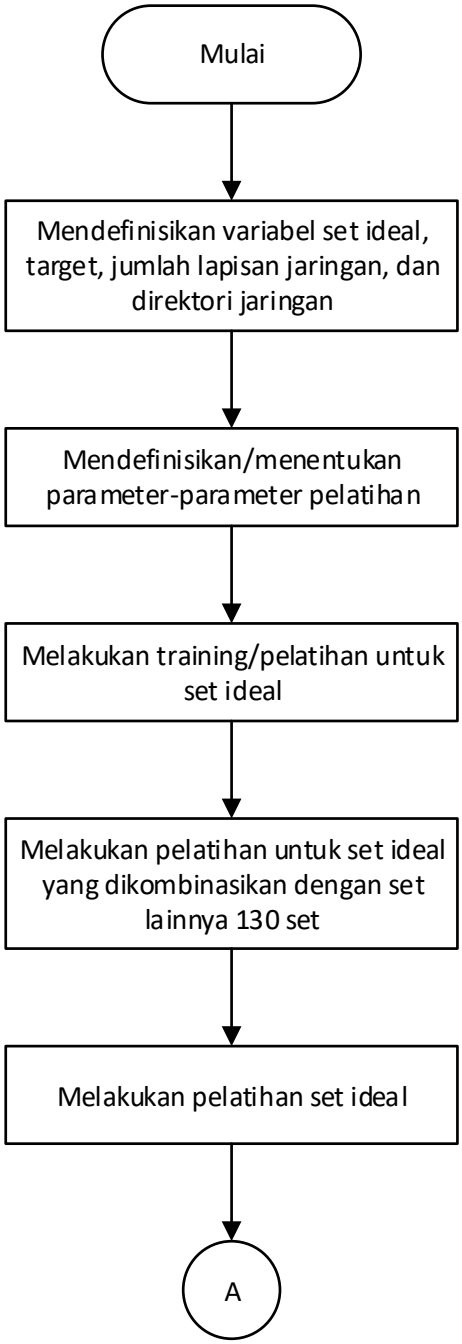
1. Program akan mendefinisikan fungsi dengan nama 'imtovector'.
2. Program akan mendefinisikan variabel-variabel yang dibutuhkan program seperti nama karakter, direktori, jumlah set, dll.
3. Program akan membuat direktori keluaran jika direktori tersebut belum tersedia.
4. Program akan melakukan *checking*, apakah *file* '.mat' *chacedata* telah tersedia. Jika telah tersedia maka program akan menjalankan *file* '.mat'

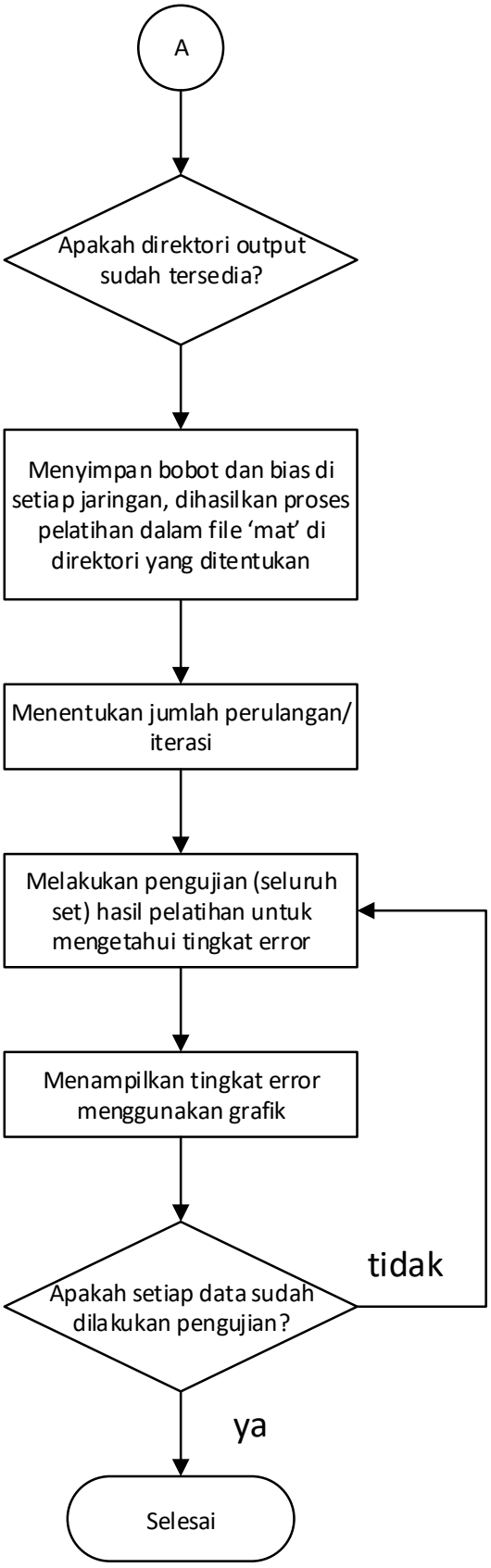
chacedata, tetapi jika belum tersedia maka program akan menjalankan instruksi berikutnya.

5. Program menentukan jumlah iterasi atau perulangan proses konversi vektor.
6. Program membaca data masukan dari direktori keluaran hasil dari hasil program pengolahan citra.
7. Program melakukan konversi vektor dari data citra berukuran 20x40 menjadi data vektor berukuran 800x1.
8. Program akan menyimpan data hasil konversi vektor di direktori keluaran yang telah ditentukan dan memastikan bahwa semua data telah dilakukan operasi konversi ke vektor.

Langkah-langkah dalam pembuatan program vektorisasi citra menggunakan pemrograman Matlab *Editor* terlampir pada lampiran B.

- Pembuatan Program Jaringan Saraf Tiruan





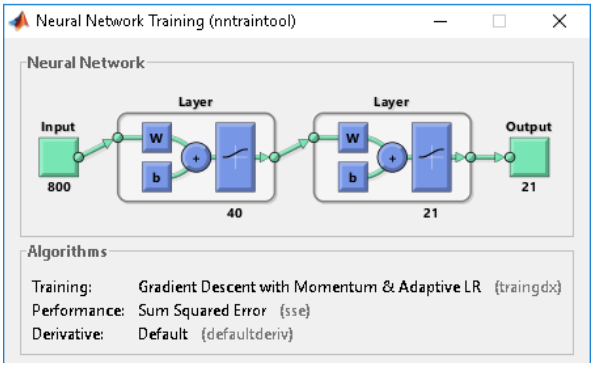
Gambar 3.10 Diagram alir proses kerja program JST

Gambar 3.10 merupakan proses pelatihan menggunakan jaringan saraf tiruan. Pada proses ini, tahapan yang dilakukan diantaranya:

1. Program akan mendefinisikan variabel target, set ideal dan membentuk jaringan serta menentukan parameter-parameter pelatihan yang dibutuhkan oleh jaringan.

```
% 1. Membuat Data dan Mendefinisikan Variabel
% ------------------------------------------------------------------------

masterset = imtovector(1);              % matriks berukuran 800 x 21
targets = eye(size(masterset, 2));      % matriks identitas 21 x 21

[R, Q] = size(masterset);               % 800 x 21
[S2, Q] = size(targets);                % 21 x 21
S1 = 20;                                % jumlah unit pada layar tersembunyi I

% ------------------------------------------------------------------------
% 2. Network Parameter
% ------------------------------------------------------------------------

% Proses inisialisasi jaringan
net = newff(repmat([0 1], R, 1), [S1 S2], {'logsig', 'logsig'},'traingdx');
       % bentuk jaringan backpropagation dengan 2 layer (S1 (20) dan S2 (21))
       % matriks 800 x 2 (berisi 0 dan 1)
       % fungsi pelatihan & perubahan sigmoid biner

net.LW{2,1} = net.LW{2,1}*0.01;         % Nilai bobot diperkecil 100 kali
net.b{2} = net.b{2}*0.01;
net.performFcn = 'sse';                 % Set fungsi performance ke sse
net.trainParam.goal = 0.001;            % Batas nilai error agar iterassi dihentikan
net.trainParam.show = 50;               % menampilkan frekuensi perubahan epochs
net.trainParam.epochs = 10000;          % Jumlah maksimum epoch pelatihan
net.trainParam.mc = 0.7;                % Momentum constant
```

2. Program akan melakukan pelatihan dengan tiga tahap pelatihan, yaitu pelatihan set ideal, pelatihan set ideal dikombinasikan dengan set lainnya dan pelatihan set ideal kembali.
3. Program akan membuat direktori keluaran. Direktori ini adalah sebagai tempat untuk menyimpan bobot serta bias yang dihasilkan dari pelatihan. Berikut adalah contoh pelatihan dari JST.



4. Program akan menyimpan nilai bobot dan bias ke dalam direktori yang telah dibuat sebelumnya.
5. Program melakukan pengujian terhadap 10 % dari jumlah data yang dilatih untuk mengetahui tingkat kesalahan (*error*) rata-rata.

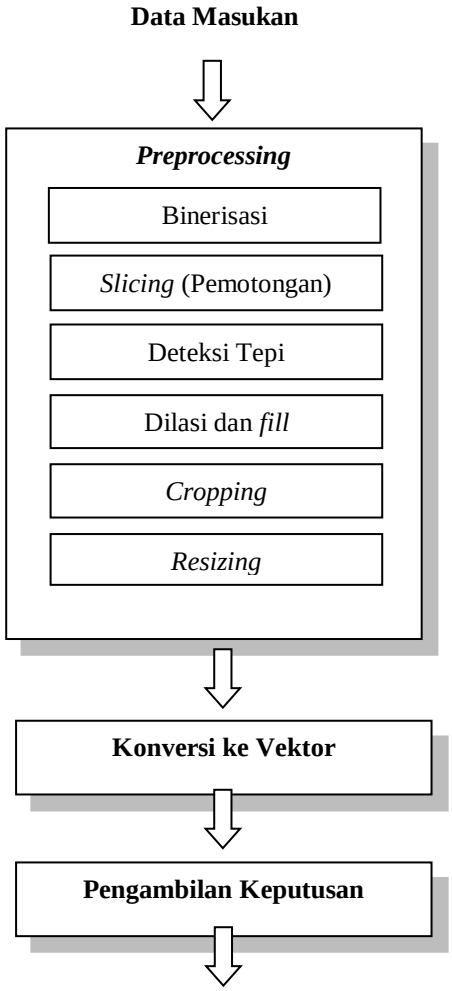
Program akan menyimpan data tingkat kesalahan dalam format *excel* dan menampilkannya dalam bentuk grafik. Hasil yang diperoleh dari jaringan saraf tiruan adalah berupa bobot dan bias jaringan yang akan digunakan pada perangkat pengujian.

3.3.6 Perangkat Pengujian

Proses yang terjadi pada perangkat pengujian hampir sama dengan proses yang ada pada perangkat pelatihan. Pada perangkat pengujian juga dilakukan pengolahan citra seperti halnya pada perangkat pelatihan, yaitu binerisasi, *slicing*, deteksi tepi Roberts dan Sobel, dilasi dan *fill*, *cropping* dan *resizing* hingga didapat citra akhir berukuran 20x40 piksel. Kemudian citra tersebut diubah ke dalam bentuk vektor yang terdiri dari 800 baris (800x1). Tidak dilakukan kembali pelatihan JST karena sudah memiliki bobot dan bias yang baru yang didapat dari pelatihan sebelumnya. Sehingga, karakter akan langsung

diolah untuk dikenali oleh jaringan dengan memanfaatkan bobot dan bias yang telah dihasilkan dari proses pelatihan sebelumnya. Perangkat uji akan langsung melakukan pengambilan keputusan untuk mengenali data masukan dan menghasilkan keluaran berupa hasil pengenalan karakter tersebut.

Tahapan dari proses kerja perangkat aplikasi ditunjukkan pada Gambar 3.11



Gambar 3.11 Proses kerja perangkat pengujian

Langkah-langkah yang perlu dilakukan dalam menampilkan hasil pembacaan pada perangkat pengujian adalah sebaga berikut:

1. Program akan membaca data masukan yang telah dituliskan aksara Lampung oleh pengguna.

2. Program akan membuat direktori keluaran yang dibutuhkan untuk menyimpan hasil pengolahan citra dan hasil vektorisasi citra.

3. Program akan melakukan proses pengolahan citra yang meliputi binerisasi, *slicing* (pemotongan), deteksi tepi, dilasi, *fill, cropping* dan *resizing* kemudian menyimpan hasil pengolahan citra ke dalam direktori yang telah dibuat.

4. Program akan membentuk matriks vektor berdasarkan data citra yang dihasilkan dari proses pengolahan citra kemudidan menyimpannya ke dalam direktori hasil vektorisasi yang telah dibuat.

5. Jaringan akan menguji data vektor untuk dapat menentukan karakter tersebut. Jaringan terlebih dahulu akan menguji karakter induk huruf kemudian diikuti pengujian terhadap anak huruf atas, bawah dan depan. Untuk anak huruf sendiri bernilai kosong karena yang diuji berupa karakter induk huruf aksara Lampung.

6. Program akan melakukan pengkodean karakter terhadap induk huruf dan anak huruf yang dikenali untuk dapat menampilkan hasil pembacaan. Tabel 3.1 menunjukkan pengkodean untuk karakter induk huruf dan anak huruf.

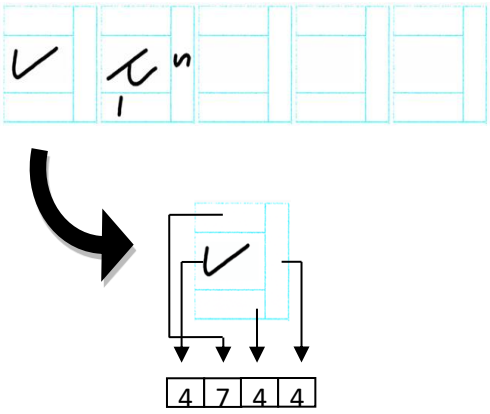
7. Program akan mencocokkan kode karakter tersebut pada *database* sehingga didapatkan hasil pembacaan dan menampilkannya pada kolom ‘Alphabet’.

Database yang berisi data pengkodean karakter dapat dilihat pada Lampiran C.

Tabel 3.1 Pengkodean karakter induk huruf dan anak huruf

Karakter induk huruf		**Karakter anak huruf**					
		Anak huruf atas		**Anak huruf bawah**		**Anak huruf depan**	
karakter	**kode**	**karakter**	**kode**	**karakter**	**kode**	**karakter**	**kode**
ka	1	*i*	1	*o*	1	*ah*	1
ga	2	*e*	2	*u*	2	*ai*	2
nga	3	*ee*	3	*au*	3	*nengen*	3
pa	4	*ar*	4	kosong	4	kosong	4
ba	5	*ang*	5				
ma	6	*an*	6				
ta	7	kosong	7				
da	8						
na	9						
ca	10						
ja	11						
nya	12						
ya	13						
a	14						
la	15						
ra	16						
sa	17						
wa	18						
ha	19						
gha	20						

Untuk menjelaskan penggunaan pengkodean karakter pada Tabel 3.1, perhatikan Gambar 3.12.



Gambar 3.12 Pengkodean karakter ‘pa’

Dengan melihat pada gambar 3.12, untuk kolom karakter pa dikodekan menjadi 4744 disesuaikan dengan *database* pengkodean karakter. Untuk anak huruf diatas dikodekan dengan angka 4, untuk huruf utama ‘pa’ dikodekan dengan angka 7. Untuk bawah dan depan karena sama-sama kosong dikodekan dengan angka 4.

3.3.7 Pengujian Sistem

Keandalan sistem ditentukan dari persentase kesalahan yang dihasilkan dari pengujian. Jika persentasi kesalahan masih berada diatas spesifikasi yang telah ditentukan, maka akan dilakukan analisis teoritis atau optimasi rancangan untuk menghasilkan keandalan sistem yang maksimal.

Hasil dari *output* berupa presentase kesalahan yang dihasilkan dari pengujian. Variasi dari *output* tersebut akan dilakukan studi perbandingan antara metode *Roberts* dan *Sobel* untuk melihat variasi tingkat *error*.

$$\%\ kesalahan = \frac{jumlah\ kesalahan}{jumlah\ pengujian} \times 100\% \qquad (3.1)$$

3.3.8 Kesimpulan dan Saran

Tahapan ini adalah tahapan akhir dari pelaksanaan tugas akhir. Berisi tentang kesimpulan dari hasil penelitian yang diperoleh serta saran yang membangun sehingga menjadi panduan untuk penelitian selanjutnya.

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah dilakukan pembuatan, pengujian dan Analisa dari penelitian ini, dapat diambil beberapa kesimpulan diantaranya:

1. Berdasarkan hasil dari pelatihan JST pada huruf utama dan bertingkat, deteksi tepi dengan menggunakan metode *sobel* lebih baik dari metode *Roberts.* Pada metode *Roberts* didapatkan presentase *error* sebesar 28.5 % sedangkan pada metode *sobel* didapatkan presentase sebesar 14.5 %.
2. Pelatihan bertingkat (ganda) atau pelatihan berulang terhadap karakter yang memiliki kemiripan bentuk, membuat tingkat *error* pelatihan menjadi lebih kecil sehingga pengenalan karakter menjadi lebih akurat.
3. Kesalahan pengenalan pada perangkat pengujian dipengaruhi oleh beberapa hal yaitu bentuk karakter aksara, proses pengolahan citra, keberagaman bentuk tulisan tangan dan penentuan parameter jaringan saraf tiruan.

5.2 Saran

Diperlukan studi lebih lanjut untuk pemilihan metode deteksi tepi yang terbaik dalam pengenalan aksara Lampung agar diperoleh hasil deteksi yang lebih akurat.

DAFTAR PUSTAKA

[1] Prasetyo, Eko. 2014. *DATA MINING-Mengolah Data Menjadi Informasi Menggunakan MATLAB*. Yogyakarta: Andi.

[2] Dung Do, Viet; & Woo, Dong-Min. (2015). Handwritten Character Recognition Using Feedforward Artificial Neural Network. *7th International Conference on Latest Trends in Engineering & Technology (ICLTET)*, 2, 142-144.

[3] Noeh, M dan Harisfadilah. 1979. *Kamus Umum Bahasa Lampung-Indonesia*. Universitas Lampung. Bandar Lampung.

[4] Kadir, Abdul; Susanto, Adhi. 2013. *Teori dan Aplikasi Pengolahan Citra.* Yogyakarta: Andi.

[5] Wijaya, M.C. dan Prijono, A. 2007. *Pengolahan Citra Digital Menggunakan MATLAB.* Informatika. Bandung.

[6] Sutoyo,T,et al. 2009. *Teori Pengolahan Citra Digital*. Yogyakarta: Andi.

[7] Hery Purnomo, Mauridhi; Muntasa, Arif. 2010. *Konsep Pengolahan Citradan Ekstraksi Fitur*. Yogyakarta: Graha Ilmu.

[8] Sutoyo, T; Mulyanto, Edy; Suhartono, Vincent. 2011. *Kecerdasan Buatan*. Yogyakarta: Andi.

[9] Deisani, Anita; Arhami, Muhammad. 2009. *Konsep Kecerdasan Buatan.* Yogyakarta: Andi.

[10] Karlik, Bekik; & Olagac, A Vehbi. (2010). Performance Analysis of Various Functions in Generalized MLP Architectures of Neural Networks. *International Journal of Artificial Intelligence and Expert Systems (IJAE)*, 1, 111-122.

[11] Siang, JJ. 2009. *Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan MATLAB*. Yogyakarta: Andi.

[12] Setiawan, Hendri. "Rancang Bangun Aplikasi Pengenalan Tulisan Tangan Aksara Lampung dengan Masukan Layar Sentuh Menggunakan Jaringan Syaraf Tiruan *Backpropagation*". Skripsi. FT, Teknik Elektro, Universitas Lampung, Bandar Lampung.

[13] Hara, Eliza. "Sistem Pengenalan Tulisan Tangan Aksara Lampung dengan Metode Deteksi Tepi (*Canny*) Berbasis Jaringan Syaraf Tiruan *Backpropagation*". Skripsi. FT, Teknik Elektro, Universitas Lampung, Bandar Lampung.