PENGENALAN POLA SIDIK JARI MENGGUNAKAN JARINGAN SYARAF *BACKPROPAGATION*

SKRIPSI

IFAN PRADHANA

DEPARTEMEN MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS AIRLANGGA
SURABAYA
2011

PENGENALAN POLA SIDIK JARI MENGGUNAKAN JARINGAN SYARAF *BACKPROPAGATION*

SKRIPSI

Sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Sains
Bidang Matematika di Fakultas Sains dan Teknologi
Universitas Airlangga

Oleh :

IFAN PRADHANA
NIM. 080610108

Tanggal Lulus : 26 Januari 2011

Disetujui Oleh :

Pembimbing I	Pembimbing II
AULI DAMAYANTI, S.Si.,M.Si	**EVA HARIYANTI, S.Si.,M.T**
NIP. 19751107 200312 2 004	**NIP. 19810508 200501 2 001**

LEMBAR PENGESAHAN NASKAH SKRIPSI

Judul : **Pengenalan Pola Sidik Jari Menggunakan Jaringan Syaraf *Backpropagation***

Penyusun : **IFAN PRADHANA**

NIM : **080610108**

Tanggal Ujian : **26 Januari 2011**

Disetujui oleh :

Pembimbing I	**Pembimbing II**
AULI DAMAYANTI, S.Si.,M.Si	**EVA HARIYANTI, S.Si.,M.T**
NIP. 19751107 200312 2 004	**NIP. 19810508 200501 2 001**

Mengetahui :

Ketua Departemen Matematika Fakultas Sains dan Teknologi Universitas Airlangga	**Ketua Program Studi Matematika Fakultas Sains dan Teknologi Universitas Airlangga**
Dr. Miswanto, M.Si	**Dra. Inna Kuswandari, M. Si.**
NIP. 19680204 199303 1 002	**NIP. 19660905 199102 2 001**

PEDOMAN PENGGUNAAN SKRIPSI

Skripsi ini tidak dipublikasikan, namun tersedia di perpustakaan dalam lingkungan Universitas Airlangga. Diperkenankan untuk dipakai sebagai referensi kepustakaan, tetapi pengutipan harus seijin penulis dan harus menyebutkan sumbernya sesuai kebiasaan ilmiah.

KATA PENGANTAR

Puji syukur alhamdulillah senantiasa penulis ucapkan kepada Allah SWT yang telah melimpahkan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul **"** *Pengenalan Pola Sidik Jari Menggunakan Jaringan Syaraf Backpropagation ".*

Dalam penyusunan skripsi ini, penulis memperoleh banyak bantuan dari berbagai pihak, karena itu penulis mengucapkan terima kasih yang sebesar-besarnya kepada.

1. Kedua orang tuaku, bapakku (Subandi) dan ibuku (Lis Indrayani). Akhirnya aku lulus, terima kasih atas semua yang telah ayah dan ibu berikan kepadaku. Pasti akan aku balas semua rasa kasih sayang & kepercayaan yang telah diberikan.
2. Adikku, Ferry Firmansyah, terima kasih atas semuanya doanya.
3. Seluruh keluarga yang banyak memberikan nasehat.
4. Auli Damayanti, S.Si, M.Si dan Eva Hariyanti, S.Si, M.T, selaku dosen pembimbing yang telah memberikan banyak nasehat, arahan dan masukan.
5. Penghuni Labkom 204 (Mas Aziz, Mas Koni, Happy Ramanja P., Adit, Iqbal, Dadang, Adi Slamet K., R. Diptya, Napoli, Debi Firlandi, Furqon S., Syaifullah Abbas, Khusnun, Ayu, Indra Kurniawan, Hidayat Dwi P., Abah, dan Arif Kurniawan) yang telah memberikan nasehat, semangat dan doanya.
6. Thank's to Diyyah Ayu dan Ratih Frastanty.
7. ML community (Novan, Jo, Seta, Sigit, Dean, Dimas, Andre, Ricky, Krisna, Arindha, dan Emak).

8. Teman-teman Math ’06 yang tidak bisa saya sebutkan satu persatu.

Penulis menyadari bahwa penyusunan skripsi ini tak lepas dari kekurangan dan kesalahan. Untuk itu penulis mengharapkan kritik dan saran yang membangun sehingga dapat dijadikan acuan untuk penyusunan tugas-tugas selanjutnya.

Surabaya, 26 Januari 2011

Ifan Pradhana

Ifan Pradhana, 2011, Fingerprint Pattern Recognition using Artificial Neural Network Backpropagation**,** This final project was supervised by Auli Damayanti, S.Si, M.Si and Eva Hariyanti, S.Si, M.T. Mathematics Department, Faculty of Sciences and Technology, Airlangga University.

ABSTRACT

Fingerprint pattern recognition is a way knowing someone's identity trough fingerprint observation and research, which being used for a lot of needs, evidence, identification or as a replacement of signature. One of Fingerprint pattern recognition is artificial neural network, this method using human's brain principal which consists of neuron as input progression to produce output from weights exist. The purpose of this final project is applying artificial neural network to fingerprint pattern recognition and create a program simulating this method using Visual Basic 6.0 software language with supporting operation system. Artificial neural network architecture used is *multilayer neural network* with backpropagation algorithm.

Data used are fingerprint images with 120 x 120 pixel size which transformed into numeric with image processing. Steps used on image processing such as grayscale process, binary image, and segmentation. From image processing, 24 x 24 matrix were been produced, then with normalization process the matrix changed into 576 x 1 seized vector for each image. From the normalization progress would be the fingerprint recognition artificial neural network input. After normalization progress, the input will progressed for training and testing.

Network training using 90 fingerprints data with 0,9 learning rate and 0,01 error, iteration stopped at 257^{th} iteration. Validation result for 30 images, can be recognized with 76,67% accuracy to those fingerprint patterns.

Keywords: fingerprint pattern recognition, artificial neural network, backpropagation

Ifan Pradhana, 2011, Pengenalan Pola Sidik Jari Menggunakan Jaringan Syaraf *Backpropagation***,** Skripsi ini dibawah bimbingan Auli Damayanti, S.Si, M.Si dan Eva Hariyanti, S.Si, M.T. Departemen Matematika, Fakultas Sains dan Teknologi, Universitas Airlangga.

ABSTRAK

Pengenalan pola sidik jari merupakan suatu sarana dan upaya pengenalan identitas diri seseorang melalui suatu proses pengamatan dan penelitian sidik jari, yang dipergunakan untuk berbagai keperluan/kebutuhan, tanda bukti, tanda pengenal ataupun sebagai pengganti tanda tangan (cap Jempol). Salah satu teknik pengenalan pola sidik jari adalah jaringan syaraf tiruan, dimana metode ini menggunakan prinsip dari otak manusia yang terdiri dari *neuron* sebagai pemrosesan *input* untuk menghasilkan *output* berdasarkan bobot yang ada. Skripsi ini bertujuan untuk menerapkan jaringan syaraf tiruan pada pengenalan pola sidik jari dan membuat program yang mensimulasikan metode ini menggunakan bahasa *software* Visual Basic 6.0 dengan sistem operasi yang mendukung. Arsitektur jaringan syaraf tiruan yang digunakan adalah *multilayer neural network* dengan algoritma pembelajaran *backpropagation*.

Data yang digunakan berupa gambar sidik jari berukuran 120 x 120 *pixel* yang telah dirubah menjadi numerik dengan proses pengolahan citra. Langkah-langkah yang dilakukan pada pengolahan citra antara lain proses *grayscale*, citra *biner*, dan segmentasi. Dari proses pengolahan citra diperoleh numerik berupa matriks berukuran 24 x 24, kemudian dengan proses normalisasi matriks tersebut dirubah menjadi vektor berukuran 576 x 1 untuk setiap gambar. Dari proses normalisasi tersebut akan menjadi *input* jaringan syaraf tiruan pengenalan pola sidik jari. Setelah proses normalisasi, *input* akan diproses untuk *training* dan *testing*.

Pelatihan jaringan menggunakan data sebanyak 90 gambar sidik jari dengan learning rate 0,9 dan error sebesar 0,01, iterasi berhenti pada iterasi ke-257. Hasil validasi untuk 30 gambar dapat dikenali dengan keakuratan sebesar 76,67% terhadap pola-pola sidik jari tersebut.

Kata kunci : pengenalan pola sidik jari, jaringan syaraf tiruan, *backpropagation*.

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

Nomor Judul Lampiran

BAB I

PENDAHULUAN

1.1 Latar Belakang

Sidik jari adalah pola dua dimensi yang diciptakan dari pergesekan bukit-bukit yang terdapat pada jari manusia (Prabhakar, 2001). Sidik jari dapat digunakan untuk mengenali identitas orang yang memilikinya karena bentuknya khas untuk setiap orang. Antara satu orang dengan yang lainnya tidak mungkin memiliki gambaran yang sama persis, bahkan pada saudara kembar sekalipun. Pada seseorang juga tidak mungkin ditemukan pola yang sama satu dengan yang lain di antara kesepuluh jarinya sendiri. Sidik jari manusia juga tidak akan pernah berubah seumur hidup seseorang dan sangat sulit membuat tiruannya. Oleh karena itu, sidik jari sebagai salah satu masalah kompleks yang tidak memiliki model matematis yang jelas akan dijadikan sebagai bahan masukan Jaringan Syaraf Tiruan (JST). Salah satu metode yang digunakan untuk proses pengenalan pola sidik jari dalam JST adalah metode *backpropagation*.

Jaringan saraf tiruan (*artificial neural network*) adalah salah satu cabang dari bidang kecerdasan buatan (*artificial intelligence*) dan merupakan alat untuk memecahkan masalah terutama di bidang-bidang yang melibatkan pengelompokan dan pengenalan pola (*pattern recognition*).

Jaringan saraf tiruan adalah suatu teknologi yang diilhami dari jaringan saraf biologis pada manusia, dapat dilatih untuk mengenali suatu obyek yang memiliki

pola tertentu dan spesifik. Dengan pelatihan yang terstruktur untuk mengenali suatu obyek tertentu sehingga dengan pasti dapat mengenali, maka jaringan saraf tiruan yang telah terbimbing itu bisa mengenali ataupun menemukan kembali obyek tersebut sekalipun diacak dengan obyek lain. (Haykin, S., 1994).

Salah satu contoh aplikasi pengenalan pola yang cukup kompleks adalah pengenalan pola sidik jari. Pola sidik jari merupakan salah satu teknologi yang dapat digunakan dalam mengidentifikasi seseorang. Bahkan sidik jari menjadi teknologi yang dirasa cukup handal, karena terbukti relatif akurat, aman, mudah, dan nyaman untuk dipakai sebagai identifikasi bila dibandingkan dengan sistem *biometric* yang lainnya seperti retina mata atau DNA.

Jaringan syaraf tiruan mulai dilirik banyak kalangan karena mempunyai banyak kelebihan dibandingkan sistem konvensional. Jaringan syaraf tiruan mewakili pikiran manusia untuk mendekatkan diri dengan komputer, maksudnya jaringan syaraf tiruan dirancang agar komputer dapat bekerja seperti/layaknya otak manusia. Adapun keunggulan dari jaringan syaraf tiruan antara lain adalah *Adaptive learning* yaitu kemampuan untuk melakukan suatu kegiatan yang didasarkan atas data yang diberikan pada saat pembelajaran atau dari pengalaman sebelumnya, *Self-Organisation* yaitu jaringan syaraf tiruan dapat membuat organisasi sendiri atau me-representasikan informasi yang didapat pada saat pembelajaran, dan *Real Time Operation* yaitu dapat menghasilkan perhitungan secara paralel serta dengan *device hardware* khusus yang dibuat. Keunggulan lainnya dari jaringan syaraf tiruan yaitu *Fault Tolerance* melalui *Redundant Information Coding*. *Fault Tolerance* merupakan kerusakan pada bagian tertentu

dari jaringan akan mengakibatkan penurunan kemampuan. Beberapa jaringan mempunyai kemampuan untuk menahan kerusakan besar pada jaringan.

Berdasarkan uraian diatas penulis tertarik menggunakan metode *Backpropagation* untuk diaplikasikan pada pengenalan pola sidik jari dengan fungsi aktivasi yang digunakan adalah fungsi sigmoid biner. Data yang digunakan merupakan *image* yang berukuran 120x120 *pixel*. Program bantu yang digunakan penulis untuk mensimulasikan metode ini adalah *software Visual Basic* 6.0 dengan sistem operasi yang mendukung.

Hal yang perlu mendapat perhatian istimewa adalah bahwa jaringan saraf tiruan tidak diprogram untuk menghasilkan keluaran tertentu. Semua keluaran atau kesimpulan yang ditarik oleh jaringan saraf tiruan didasarkan pada pengalamannya selama mengikuti proses pembelajaran. Pada proses pembelajaran, ke dalam jaringan saraf tiruan dimasukkan pola-pola *input* (dan *target*) lalu jaringan akan diajari untuk memberikan jawaban yang benar.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalahnya adalah sebagai berikut :

1. Bagaimana cara menyusun algoritma dengan metode pengolahan citra pada *image* sidik jari manusia yang meliputi proses *greyscale*, konversi ke citra *biner*, dan segmentasi sehingga diperoleh pola sidik jari?

2. Bagaimana cara menyusun algoritma jaringan saraf tiruan dengan metode *backpropagation* untuk pengenalan pola sidik jari manusia?
3. Bagaimana membuat program untuk mengimplementasikan pengolahan citra dan jaringan syaraf tiruan metode *backpropagation* dengan menggunakan bahasa pemprograman Visual Basic untuk pengenalan pola sidik jari manusia?

1.3 Tujuan

Tujuan dari penulisan ini adalah sebagai berikut :

1. Menyusun algoritma dengan metode pengolahan citra pada *image* sidik jari manusia yang meliputi proses *greyscale*, konversi ke citra *biner*, dan segmentasi sehingga diperoleh pola sidik jari.
2. Menyusun algoritma jaringan saraf tiruan dengan metode *backpropagation* untuk pengenalan pola sidik jari manusia.
3. membuat program untuk mengimplementasikan pengolahan citra dan jaringan syaraf tiruan metode *backpropagation* dengan menggunakan bahasa pemprograman Visual Basic untuk pengenalan pola sidik jari manusia.

1.4 Manfaat

Manfaat dari penulisan ini adalah sebagai berikut :

1. Mengetahui penerapan atau aplikasi penggunaan jaringan syaraf *Backpropagation*.
2. Memperdalam pengetahuan tentang penerapan jaringan syaraf *Backpropagation*, terutama penerapan dalam permasalahan pengenalan pola sidik jari.
3. Menambah khasanah ilmu pengetahuan tentang teori-teori matematika khususnya di bidang terapan matematika, mengenai aplikasi kecerdasan buatan.
4. Menjadi bahan pertimbangan dan perbandingan untuk penerapan algoritma lainnya yang dapat mendukung perkembangan ilmu pengetahuan dan teknologi pada masa mendatang.

1.5 Batasan Masalah

Batasan masalah dari penulisan ini adalah sebagai berikut :

1. Data yang digunakan adalah berupa *image* sidik jari manusia berukuran 120x120 *pixel* dengan *file* berupa JPG.
2. Sidik jari manusia yang digunakan adalah 5 (lima) jari pada jari tangan kiri.
3. Sidik jari yang diambil, dimulai dari jari kelingking sampai ibu jari.
4. Posisi semua sidik jari vertikal.

BAB II

TINJAUAN PUSTAKA

2.1 Identifikasi Sidik Jari

Identifikasi sidik jari biasa juga disebut *Daktiloskopi* yang merupakan suatu sarana dan upaya pengenalan identitas diri seseorang melalui suatu proses pengamatan dan penelitian sidik jari, yang dipergunakan untuk berbagai keperluan/kebutuhan, tanda bukti, tanda pengenal ataupun sebagai pengganti tanda tangan (cap Jempol).

(Anonim 1, 2010)

2.2 Pengenalan Pola

Pengenalan pola (*pattern recognition*) sebenarnya telah lama ada, dan pengenalan pola mengalami perkembangan terus menerus dimulai dari pengenalan pola tradisional kemudian menjadi pengenalan pola modern. Pada mulanya pengenalan pola berbasis pada kemampuan alat indra manusia, dimana manusia mampu mengingat suatu informasi pola secara menyeluruh hanya berdasarkan sebagian informasi pola yang tersimpan di dalam ingatannya.

Inti dari pengenalan pola adalah proses pengenalan suatu objek dengan menggunakan berbagai mode dimana dalam proses pengenalannya memiliki tingkat akurasi yang tinggi. Tingkat akurasi yang tinggi memiliki pengertian

bahwa suatu objek yang secara manual (oleh manusia) tidak dapat dikenali tetapi bila menggunakan salah satu metode pengenalan yang diaplikasikan pada komputer masih dapat dikenali.

(Suta Wijaya, 2004)

2.3 Pengolahan Citra

Pengolahan citra merupakan suatu proses dengan masukan berupa citra dan hasilnya juga berupa citra. Pengolahan citra ini bertujuan untuk memperbaiki kualitas citra, dimana citra yang dihasilkan dapat memberikan informasi secara jelas dan informasi ciri citra tersebut sudah berupa numerik. Adapun metode pengolahan citra yang digunakan pada penelitian ini adalah sebagai berikut:

2.3.1 Greyscale

Citra sebagai keluaran suatu sistem pengambilan atau perekaman data dapat bersifat optik berupa foto, bersifat analog berupa sinyal-sinyal video seperti gambar pada monitor TV. Proses pengolahan citra dikenal sebagai *image processing*. *Image processing* merupakan proses yang berhubungan dengan cara untuk meningkatkan kualitas suatu image sehingga representasi image tersebut mudah dimengerti oleh manusia.

Secara digital suatu *greyscale* dapat direpresentasikan dalam bentuk array dua dimensi. Tiap elemen pada array tersebut menunjukkan intensitas (*greyscale*) dari image pada posisi koordinat yang bersesuaian. Apabila suatu image direpresentasikan dalam 8 bit maka berarti pada image terdapat 2^8 atau 256 level *greyscale.* Level *grayscale* tersebut diperoleh dengan mengambil

rata-rata dari nilai warna *red, green, blue* dengan cara sebagai berikut :

$$Grey = \frac{r + g + b}{3} \qquad (2.1)$$

Dimana :

r = nilai matriks *red*

g = nilai matriks *green*

b = nilai matriks *blue*

dengan nilai 0-255. Dengan 0 menunjukkan level intensitas yang paling gelap dan 255 menunjukkan intensitas paling terang. Tiap elemen pada array diatas disebut sebagai *picture elemen* atau sering dikenal sebagai pixel. Dengan melakukan perubahan pada intensitas masing-masing *pixel* maka representasi image secara keseluruhan akan berubah. Image yang dinyatakan dengan NxM matriks mempunyai intensitas tertentu pada *pixel* tertentu, dengan N merupakan jumlah baris dan M merupakan jumlah kolom. Posisi *picture elemen* (x, y) dan koordinat *pixel* (r, c) berbeda.

(Basuki, Palandi, Fatchurrochman, 2005)

2.3.2 Citra *Biner*

Citra *biner* (hitam-putih) merupakan citra yang banyak dimanfaatkan untuk keperluan *pattern recognition* yang sederhana seperti pengenalan angka atau pengenalan huruf. Untuk citra dengan derajat keabuan 256, maka nilai tengahnya adalah 128 sehingga untuk mengubah menjadi citra *biner* dapat dituliskan :

Jika x<128 maka x=0, jika tidak maka x=255

Menurut Basuki, Palandi, dan Fatchurrochman, untuk hasil citra *biner* yang sempurna maka dihitung nilai rata-rata derajat keabuan dan kemudian ditentukan batas-batasnya, dapat digunakan rumus :

$$xr = \frac{1}{m.n}\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} \qquad (2.2)$$

Dimana :

xr = nilai citra *biner*

X_{ij} = nilai *pixel* pada posisi (i,j)

Jika x<xr maka x=0, jika tidak maka x=255

(Basuki, Palandi, Fatchurrochman, 2005)

2.3.3 *Segmentasi*

Secara umum segmentasi citra dapat diartikan membagi citra menjadi bagian-bagian penyusunnya atau segmen-segmen yang lebih kecil sehingga diharapkan untuk pengolahan datanya dapat menjadi lebih cepat. Hasil dari tahap segmentasi citra ini berupa data piksel yang menyusun batas dari suatu

daerah atau semua titik dalam suatu daerah. Tahapan dalam proses segmentasi adalah, tiap segmen dari suatu citra dicari rata segmen yaitu jumlah dari intensitas atau tingkat *greyscale* dari keseluruhan piksel dalam satu segmen dibagi banyaknya piksel dari segmen tersebut. Sehingga tiap segmen terdiri dari piksel yang mempunyai tingkat *greyscale* yang sama. Proses segmentasi citra dapat dilihat pada Gambar 2.1.

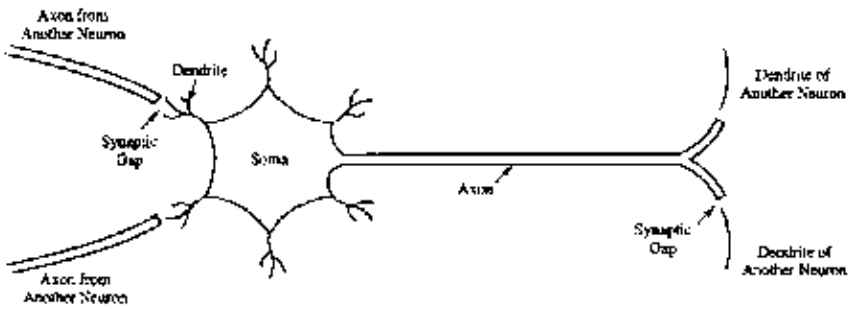
Gambar 2.1 : Proses Segmentasi 4 piksel diwakili 1 piksel

(Basuki, Palandi, Fatchurrochman, 2005)

2.4 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan sama halnya seperti jaringan syaraf manusia yang terdiri atas sel-sel yang disebut *neuron*. Ada tiga komponen utama *neuron*, yaitu *dendrit*, *soma*, dan *akson*. *Dendrit* akan menerima sinyal-sinyal dari *neuron* lain. Sinyal tersebut merupakan *impuls* listrik yang ditransmisikan melalui *sypnatic gap* melalui proses kimia. Sedangkan, *soma* atau badan sel akan menjumlah sinyal-sinyal *input* yang masuk. Jika ada *input* yang masuk, sel akan aktif dan akan mentransmisikan sinyal ke sel lain melalui *akson* dan *sypnatic gap*. Untuk lebih

jelasnya susunan *neuron* biologis ini dapat dilihat pada Gambar 2.2 berikut :



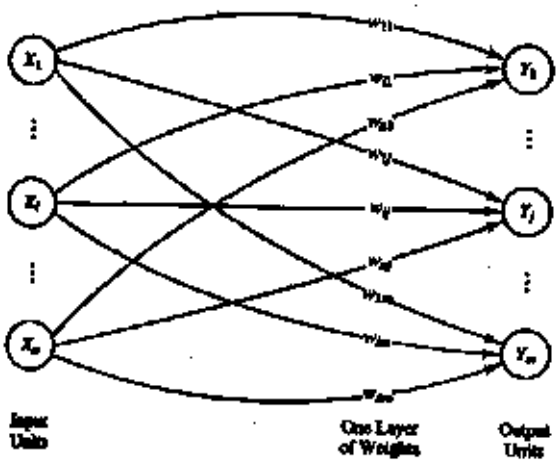
Gambar 2.2 : Susunan *neuron* biologis

Pada jaringan syaraf tiruan, juga terdapat istilah *neuron* atau lebih sering disebut unit, sel, atau *node*. Setiap *neuron* terhubung dengan *neuron-neuron* lain melalui *layer* dengan bobot tertentu. Bobot di sini melambangkan informasi yang digunakan oleh jaringan untuk menyelesaikan persoalan. Pada jaringan syaraf biologis, bobot tersebut fungsinya sama dengan aksi pada proses kimia yang terjadi pada *synaptic gap*. Sedangkan, setiap *neuron* mempunyai *internal state* yang disebut aktivasi. Aktivasi tersebut merupakan fungsi dari *input* yang diterima. Suatu *neuron* akan mengirim sinyal ke *neuron-neuron* lain, tetapi pada suatu saat, hanya ada satu sinyal yang dapat dikeluarkan walaupun sinyal tersebut ditransmisikan pada beberapa *neuron* lain.

(Anang, 2009)

2.4.1 Arsitektur Jaringan Syaraf Tiruan

Pada jaringan syaraf, *neuron-neuron* tersusun dalam *layer*. *Neuron* yang terletak dalam *layer* yang sama, biasanya mempunyai hubungan yang sama antara satu dengan lainnya.

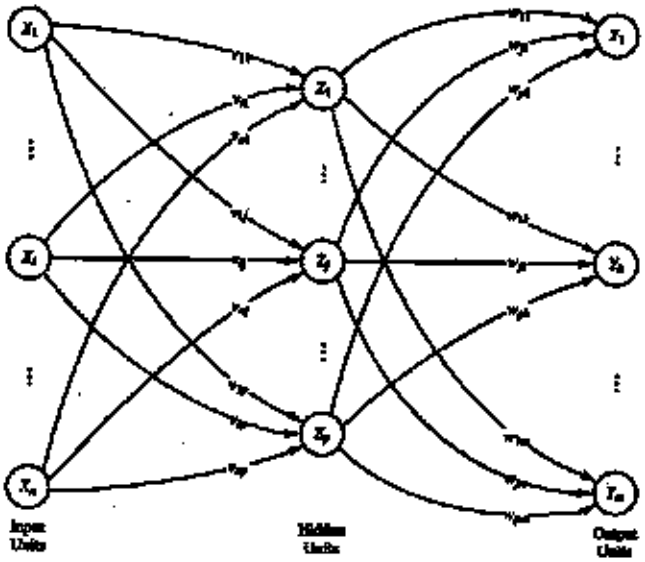


Gambar 2.3 : Jaringan syaraf *single layer*

Pengaturan *neuron* dalam *layer* dan hubungan-hubungannya disebut arsitektur jaringan.

Jaringan syaraf dapat diklasifikasikan menjadi dua jenis, yaitu *single layer* dan *multilayer*. Untuk lebih jelasnya, susunan *single layer* dan *multilayer* terdapat pada Gambar 2.3 dan Gambar 2.4.

Dalam jaringan *single layer*, *neuron-neuron* dapat dikelompokkan menjadi dua bagian, yaitu unit-unit *input* dan unit-unit *output*. Unit-unit *input* menerima masukan dari luar sedangkan unit-unit *output* akan mengeluarkan respon dari jaringan sesuai dengan masukannya.



Gambar 2.4 : Jaringan syaraf *multilayer*

Sedangkan, dalam jaringan *multilayer*, selain ada unit-unit *input* dan *output*, juga terdapat unit-unit yang tersembunyi (*hidden*). Jumlah unit *hidden* tersebut tergantung pada kebutuhan. Semakin kompleks jaringan, unit *hidden* yang dibutuhkan makin banyak, demikian pula jumlah *layer*nya. Pada Gambar 2.4, terdapat tiga buah *layer* dengan bobot v dan w. Jaringan *multilayer* sering menyelesaikan persoalan yang lebih rumit karena pelatihan untuk hal yang kompleks akan lebih berhasil jika menggunakan jaringan *multilayer*.

(Anang, 2009)

2.4.2 Pelatihan Jaringan Syaraf Tiruan

Pelatihan jaringan syaraf dimaksudkan untuk mencari bobot-bobot yang terdapat dalam tiap *layer*. Ada dua jenis pelatihan dalam sistem jaringan

syaraf yaitu proses belajar terawasi dan proses belajar tak terawasi.

a. Proses Belajar Terawasi

Dalam proses belajar yang terawasi, seolah-olah ada "guru" yang mengajari jaringan. Cara pelatihan jaringan tersebut adalah dengan memberikan data-data yang disebut data *training*. Data *training* terdiri atas pasangan *input-output* yang diharapkan dan merupakan *associative memory*. Data-data itu biasanya didapat dari pengalaman atau pengetahuan seseorang dalam menyelesaikan persoalan. Setelah jaringan dilatih, *associative memory* akan mengingat suatu pola. Jika jaringan diberi *input* baru, jaringan dapat mengeluarkan *output* seperti yang diharapkan (*target*) berdasarkan pola yang sudah ada.

Ada beberapa metode dalam proses belajar terawasi, diantaranya *Delta Rule*, *Backpropagation* atau *Generalized Delta Rule* dan *Counterpropagation*.

(Anang, 2009)

b. Proses Belajar Tak Terawasi

Dalam proses belajar tak terawasi, tidak ada "guru" yang mengajari jaringan. Jaringan hanya diberi data *input*, tanpa *target*. Jaringan akan memodifikasi bobot sehingga untuk *input* yang hampir sama, *output* yang dihasilkan sama (*cluster units*). Jaringan akan menghasilkan contoh-contoh vektor untuk setiap *cluster* yang terbentuk.

Metode yang dipakai dalam proses belajar tak terawasi ini antara lain *Kohonen Self Organizing Maps* dan *Learning Vector Quantization*.

(Anang, 2009)

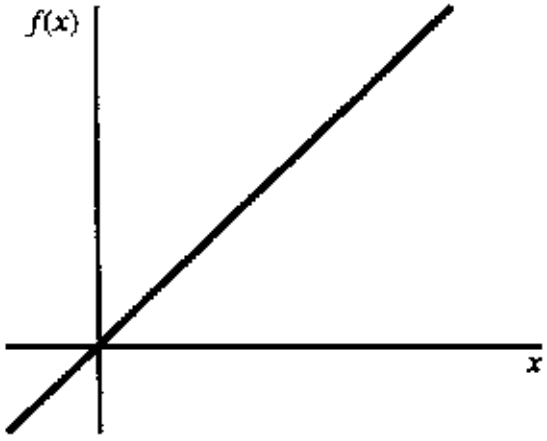
2.4.3 Fungsi Aktivasi

Berikut fungsi-fungsi aktivasi yang biasanya digunakan dalam sistem jaringan syaraf.

a. Fungsi Identitas

$$f(x) = x \text{, untuk semua } x \qquad (2.3)$$

Fungsi identitas merupakan fungsi aktivasi untuk semua unit *input*. Bentuk fungsi identitas terdapat pada Gambar 2.5.



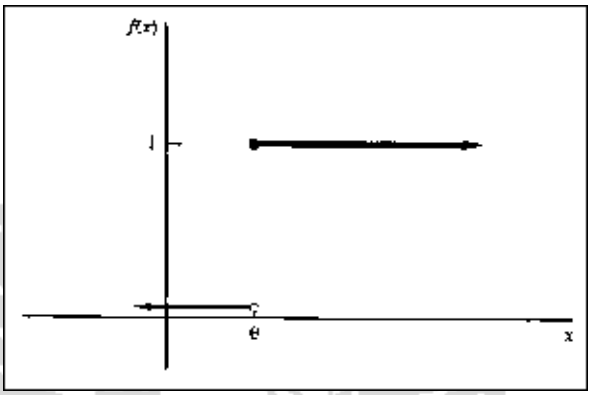
Gambar 2.5 : Fungsi identitas

(Fausett, 2003)

b. Fungsi *Step Biner*

$$f(x) = \begin{cases} 1 & \text{jika } x \geq \theta \\ 0 & \text{jika } x < \theta \end{cases} \qquad (2.4)$$

Fungsi *step biner* sering dipakai pada jaringan *single layer*. Bentuk fungsi *step biner* terdapat pada Gambar 2.6.



Gambar 2.6 : Fungsi *step biner*

(Fausett, 2003)

c. Fungsi *Sigmoid Biner*

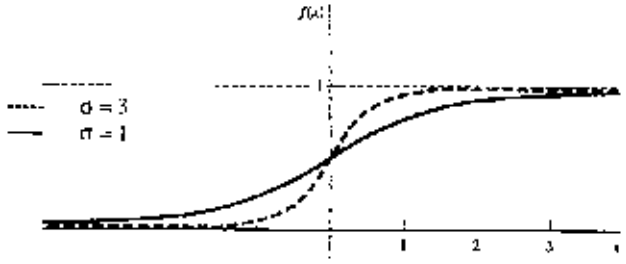
$$f(x) = \frac{1}{1 + \exp(-x\sigma)} \qquad (2.5)$$

$$f'(x) = \sigma f(x)[1 - f(x)] \qquad (2.6)$$

di mana σ adalah parameter kecuraman fungsi.

Fungsi *sigmoid biner* berbentuk kurva S dan merupakan fungsi yang paling umum. Biasanya *sigmoid biner* digunakan dalam jaringan yang menggunakan metode pelatihan *backpropagation*, karena bentuk fungsi aktivasi dan turunan fungsinya sederhana sehingga mudah dihitung. Fungsi

sigmoid biner mempunyai nilai *range* [0,1]. Fungsi tersebut digunakan jika *output* yang diinginkan (*target*) terletak dalam interval [0,1]. Pada Gambar 2.7 berikut adalah bentuk dari fungsi *sigmoid biner* dengan $\sigma = 1$ dan $\sigma = 3$. Biasanya σ yang digunakan $= 1$.



Gambar 2.7 : Fungsi *sigmoid biner*

(Fausett, 2003)

d. Fungsi *Sigmoid Bipolar*

$$g(x) = 2f(x) - 1 = \frac{2}{1 + \exp(-\sigma x)} - 1$$

$$= \frac{1 - \exp(-\sigma x)}{1 + \exp(-\sigma x)} \qquad (2.7)$$

$$g'(x) = \frac{\sigma}{2}[1 + g(x)][1 - g(x)] \qquad (2.8)$$

Fungsi *sigmoid bipolar* adalah fungsi *sigmoid biner* yang mempunyai nilai *range* [-1,1]. Fungsi *sigmoid bipolar* berhubungan erat dengan fungsi tangen hiperbolik. Fungsi tangen hiperbolik juga dapat digunakan sebagai fungsi aktivasi jika *output* yang diinginkan dari jaringan terletak dalam interval [-1,1]. Persamaan berikut menunjukkan bentuk fungsi *sigmoid bipolar* dan fungsi tangen hiperbolik untuk $\sigma = 1$.

Untuk fungsi *sigmoid bipolar* :

$$g(x) = \frac{1 - \exp(-x)}{1 + \exp(-x)} \qquad (2.9)$$

Sedangkan, untuk fungsi tangen hiperbolik :

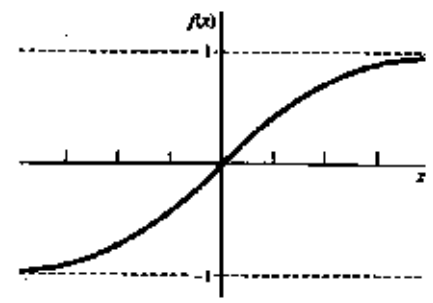
$$h(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

$$= \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \qquad (2.10)$$

dengan turunan fungsi tangen hiperbolik :

$$h'(x) = [1 + h(x)][1 - h(x)] \qquad (2.11)$$

Bentuk dari *sigmoid bipolar* terdapat pada Gambar 2.8.



Gambar 2.8 : Fungsi *sigmoid bipolar*

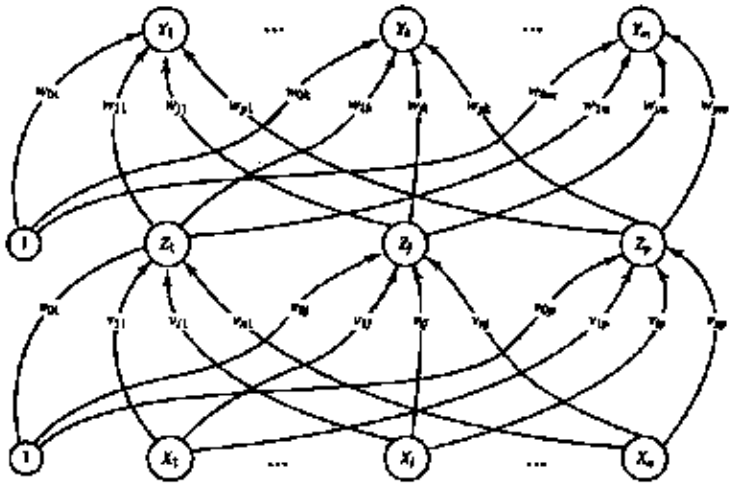
(Fausett, 2003)

2.5 Jaringan Syaraf *Backpropagation*

2.5.1 Arsitektur Jaringan

Salah satu dan metode pelatihan terawasi pada jaringan syaraf adalah metode *backpropagation*. Ciri metode tersebut adalah meminimalkan *error* pada *output* yang dihasilkan oleh jaringan.

Dalam metode *backpropagation*, biasanya digunakan jaringan *multilayer*. Sebagai contoh, pada Gambar 2.9 digambarkan jaringan dengan sebuah *hidden layer*. Dalam jaringan, selain terdapat unit-unit *input*, unit-unit tersembunyi (*hidden*) dan *output* juga terdapat bias yang diberikan pada unit-unit tersembunyi dan *output*.



Gambar 2.9 : Jaringan syaraf *backpropagation* dengan satu *hidden layer*

Pada Gambar 2.9 dan selanjutnya, unit *input* akan dilambangkan dengan X, unit *hidden* dilambangkan dengan Z, dan unit *output* dilambangkan dengan Y. Sedangkan, untuk bobot antara X dan Z dilambangkan dengan v

dan bobot antara Z dan Y dilambangkan dengan w. Untuk bias, biasanya dipakai indeks v_{01} dan w_{01}.

(Fausett, 2003)

2.5.2 Algoritma Metode *Backpropagation*

Pada intinya, pelatihan dengan metode *backpropagation* terdiri atas tiga langkah, yaitu sebagai berikut :

a. Umpan maju (*feedforward*).

b. Propagasi balik error (*backpropagation of error*)

c. Pembaharuan (*update*) bias dan bobot.

Saat umpan maju (*feedforward*), setiap unit *input* (X_i) akan menerima sinyal *input* dan akan menyebarkan sinyal tersebut pada tiap unit *hidden* (Z_j). Setiap unit *hidden* kemudian akan menghitung aktivasinya dan mengirim sinyal (z_j) ke tiap unit *output*. Kemudian, setiap unit *output* (Y_k) juga akan menghitung aktivasinya (y_k) untuk menghasilkan respon terhadap *input* yang diberikan jaringan.

Saat proses pelatihan, setiap unit *output* membandingkan aktivasinya (y_k) dengan nilai *target* untuk menentukan besarnya *error*. Berdasarkan *error* tersebut, dihitung faktor δ_k. Faktor δ_k digunakan untuk mendistribusikan *error* dan *output* kembali ke *layer* sebelumnya. Dengan cara yang sama, faktor β_j juga dihitung pada unit *hidden* Z_j. Faktor δ_k digunakan untuk

memperbaharui bobot antara *hidden layer* dan *input layer*.

Setelah semua faktor δ dan β ditentukan, bobot untuk semua *layer* diperbaharui secara bersamaan. Pembaharuan bobot w_{jk} (dari unit *hidden* Z_j ke unit *output* Y_k) dilakukan berdasarkan faktor δ_k dan aktivasi z_j dari unit *hidden* Z_j. Sedangkan, pembaharuan bobot v_{ij} (dari unit *input* X_i ke unit *hidden* Z_j) dilakukan berdasarkan faktor β_j, dan aktivasi x_i, dari *input*.

Selengkapnya, notasi-notasi yang akan digunakan pada algoritma pelatihan dan pengujian yaitu sebagai berikut.

x Data *training* untuk *input.*

$$x = (x_1,...,x_i,...,x_n)$$

t Data *training* untuk *output* (*target*).

$$t = (t_1,...,t_k,...,t_m)$$

α *Learning rate* yaitu parameter yang mengontrol perubahan bobot selama pelatihan. Jika *learning rate* besar, jaringan semakin cepat belajar, tetapi hasilnya kurang akurat. *Learning rate*, biasanya dipilih antara 0 dan 1.

X_i Unit *input* ke-i . Untuk unit *input*, sinyal yang masuk dan keluar pada suatu unit dilambangkan dengan variabel yang sama, yaitu x_i.

Z_j Unit *hidden* ke-j . Sinyal *input* pada Z_j dilambangkan dengan

z_in_j. Sinyal *output* (aktivasi) untuk Z_j dilambangkan dengan z_j.

v_{0j} Bias untuk unit *hidden* ke- j.

v_{ij} Bobot antara unit *input* ke- i dan unit *hidden* ke- j.

Y_k Unit *output* ke- k. Sinyal *input* ke Y_k dilambangkan y_in_k. Sinyal *output* (aktivasi) untuk Y_k dilambangkan dengan y_k.

w_{0k} Bias untuk unit *output* ke- k.

w_{jk} Bobot antara unit *hidden* ke- j dan unit *output* ke- k.

δ_k Faktor koreksi *error* untuk bobot w_{jk}.

β_j Faktor koreksi *error* untuk bobot v_{ij}.

(Fausett, 2003)

2.5.2.1 Prosedur Pelatihan

Langkah-langkah pelatihan *backpropagation* sebagai berikut :

Langkah 0. Inisialisasi bobot (sebaiknya diatur pada bilangan acak yang kecil).

Langkah 1. Jika *error* atau *maximum epoch* belum terpenuhi.

Umpan maju

Langkah 2. Setiap unit *input* $(X_i, i = 1,...,n)$ menerima sinyal *input* x_i dan

menyebarkan sinyal tersebut pada seluruh unit lapisan di atasnya (unit *hidden*).

Langkah 3. Setiap unit *hidden* $(Z_j, j = 1,..., p)$, akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot, termasuk biasnya :

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \qquad (2.12)$$

dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output*, dan unit *hidden* yang bersangkutan :

$$z_j = f(z\_in_j) \qquad (2.13)$$

lalu mengirim sinyal *output* tersebut ke seluruh unit pada unit pada lapisan atasnya (unit *output*).

Langkah 4. Setiap unit *output* $(Y_k, k = 1,..., m)$ akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot, termasuk biasnya :

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \qquad (2.14)$$

dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output* dan unit *output* yang bersangkutan :

$$y_k = f(y\_in_k) \qquad (2.15)$$

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

Propagasi balik error

Langkah 5. Setiap unit *output* $(Y_k, k = 1,...,m)$ menerima suatu *target* yang sesuai dengan *input training* untuk menghitung kesalahan (*error*) antara *target* dengan *output* yang dihasilkan jaringan :

$$\delta_k = (t_k - y_k) f'(y\_in_k) \qquad (2.16)$$

Faktor δ_k digunakan untuk menghitung koreksi *error* (Δw_{jk}) yang akan dipakai untuk memperbaharui w_{jk}, di mana :

$$\Delta w_{jk} = \alpha \delta_k z_j \qquad (2.17)$$

Selain itu, juga dihitung koreksi bias Δw_{0k} yang akan dipakai untuk memperbaharui w_{0k}, di mana :

$$\Delta w_{0k} = \alpha \delta_k \qquad (2.18)$$

Faktor δ_k kemudian dikirimkan ke *lapisan* yang berada di bawahnya.

Langkah 6. Setiap unit *hidden* $(Z_j, j = 1,..., p)$ menjumlah *input* delta yang sudah berbobot :

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \qquad (2.19)$$

Kemudian, hasilnya dikalikan dengan turunan dari fungsi aktivasi yang digunakan jaringan untuk menghasilkan faktor koreksi *error* (β_j), di mana :

$$\beta_j = \delta\_in_j f'(z\_in_j) \qquad (2.20)$$

Faktor β_j digunakan untuk menghitung koreksi *error* (Δv_{ij}) yang akan dipakai untuk memperbaharui v_{ij}, di mana :

$$\Delta v_{ij} = \alpha \delta_j x_i \qquad (2.21)$$

Selain itu, juga dihitung koreksi bias Δv_{0j} yang akan dipakai untuk memperbaharui v_{0j}, di mana :

$$\Delta v_{0j} = \alpha \delta_j \qquad (2.22)$$

Langkah 7. (Pembaharuan bias dan bobot)

Setiap unit *output* $(Y_k, k = 1,...,m)$ akan memperbaharui bias dan bobotnya dari setiap unit *hidden* $(Z_j, j = 1,..., p)$,

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \qquad (2.23)$$

Demikian pula, setiap unit *hidden* $(Z_j, j = 1,..., p)$ akan memperbaharui bias dan bobotnya dari setiap unit *input* $(X_i, i = 1,...,n)$,

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \qquad (2.24)$$

Langkah 8. Memeriksa *stop condition*.

Untuk menentukan *stopping condition* terdapat dua cara, yaitu :

- Membatasi iterasi (*epoch*) yang ingin dilakukan.

- Membatasi *error*.

Cara menghitung *error* dengan *Mean Square Error* (*MSE*) adalah sebagai berikut :

- Dengan bobot yang ada saat itu, lakukan langkah umpan maju (langkah ke-3 sampai dengan langkah ke-5 di mana *input*nya diambil dari *input training set*, jika yang ingin dihitung adalah *training set error* maka yang dihitung adalah *training set error*. Langkah tersebut dilakukan untuk semua data *training*.
- Kemudian, dicari selisih antara *target output* (t_k) dengan *output* jaringan (y_k) dan dihitung pada persamaan *Mean Square Error*. Jika terdapat m data *training*, maka :

$$MSE = 0{,}5 \times \{(t_{k1} - y_{k1})^2 + (t_{k2} - y_{k2})^2 + ... + (t_{km} - y_{km})^2\} \quad (2.25)$$

(Fausett, 2003)

2.5.2.2 Prosedur Pengujian

Setelah pelatihan, jaringan syaraf *backpropagation*, diaplikasikan dengan hanya menggunakan umpan maju dari algoritma pelatihan. Prosedur pengujiannya adalah sebagai berikut :

Langkah 0. Inisialisasi bobot (dari algoritma pelatihan).

Langkah 1. Untuk $i = 1,...,n$, atur aktivasi unit *input* x_i.

Langkah 2. Untuk $j = 1,..., p$:

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

$$z_j = f(z\_in_j)$$

Langkah 3. Untuk $k = 1,..., m$:

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$$

$$y_k = f(y\_in_k)$$

Langkah 4. Jika data *input* sudah mencapai *n*, maka proses pengujian berhenti.

(Fausett, 2003)

BAB III

METODE PENELITIAN

Adapun langkah-langkah penyelesaian pengenalan pola sidik jari dalam penelitian ini adalah sebagai berikut :

1. Pengambilan data berupa *image* 5 (lima) sidik jari pada jari tangan kiri dari 6 mahasiswa.

   Sampel yang digunakan berupa *image* sidik jari dari mahasiswa yang berbeda. Dimensi citra yang digunakan adalah matriks 120 x 120 *pixel*. Dipilihnya dimensi tersebut, agar *input* yang digunakan ke jaringan saraf tiruan tidak terlalu banyak, yang nantinya mempengaruhi proses dari sistem yang dibuat. Jumlah *image* sidik jari yang digunakan adalah sebanyak 120 *image* sidik jari yang diambil dari 5 sidik jari pada tangan kiri 6 mahasiswa yang masing-masing diambil sebanyak 4 kali.

2. Pengolahan citra hasil dari *image* sidik jari.

   Langkah-langkah pengolahan citra antara lain :

   a. Pembacaan file *image* sidik jari berukuran 120 x 120 *pixel.*
   b. Proses *Grayscale*, yaitu mengubah citra berwarna (red, green, blue) menjadi citra *grayscale* dengan mengambil rata-rata dari rgb menggunakan persamaan 2.1 yang menghasilkan nilai antara [0,255].

c. Proses konversi ke citra *biner*, yaitu proses mengubah kuantisasi citra dengan mengatur jumlah derajat keabuan yang ada pada citra yang menghasilkan nilai 0 dan 255.

d. Proses segmentasi pada citra yakni menjumlahkan *greyscale* dari keseluruhan *pixel* dalam satu segmen dibagi banyaknya *pixel* dari segmen tersebut, sehingga tiap segmen terdiri dari *pixel* yang mempunyai tingkat *greyscale* yang sama. Tiap segmen berisi 5 x 5 *pixel*, sehingga citra berukuran 120 x 120 *pixel* menjadi 24 x 24 segmen atau 576 segmen.

e. Proses normalisasi pada citra yakni proses pada nilai intensitas tiap segmen dari citra agar bernilai antara 0-1 dengan cara rata segmen dibagi dengan tingkat *greyscale* yang paling tinggi. Matrik berukuran 24 x 24 segmen menjadi matrik berukuran 576 x 1 segmen yang bernilai antara 0-1 ini akan menjadi *input* jaringan syaraf tiruan pengenalan pola sidik jari.

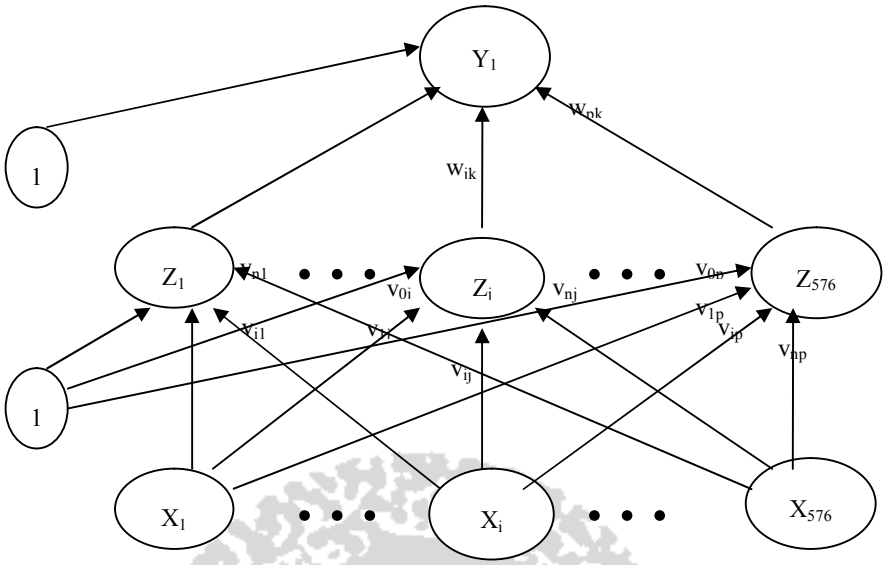
Untuk mempermudah pembacaan langkah-langkah penyelesaian pada proses pengolahan citra bisa dilihat pada Gambar 3.2.

3. Membuat rancangan data *output* yang akan digunakan sebagai pelatihan (*training*) dan pengujian (*testing*).

Unit *output* ditentukan sebanyak 1 *node*, yang bernilai *range* (0,1). Dengan pertimbangan bahwa pelambangan 5 sidik jari dari 6 mahasiswa dalam bentuk *y_output* yang bernilai antara 0 sampai 1.

4. Mendesain arsitektur jaringan.

Dalam skripsi ini hanya terdiri dari 3 *layer*, yaitu sebuah *input layer*, sebuah *hidden layer*, dan sebuah *output layer*.



Gambar 3.1 : Rancangan arsitektur *backpropagation* untuk pengenalan pola sidik jari

Rancangan jumlah unit pada tiap *layer* :

a. *Input layer* sebanyak 576 *node*, sesuai dengan dimensi yang digunakan yaitu 24 x 24.

b. *Hidden layer* sebanyak 576 *node*.

   Penentuan jumlah *node* bervariasi, jumlah *node* yang terlalu sedikit akan menyebabkan proses pelatihan tidak akan menghasilkan bobot yang stabil, namun jumlah *node* yang terlalu banyak akan menyebabkan proses pelatihan menjadi lebih lambat (Budhi, G.S., 2003).

c. *Output layer* sebanyak 1 *node*, dimana pelambangan 5 sidik jari dari 6 mahasiswa dalam bentuk *y_output* yang bernilai antara 0 sampai 1.

5. Menentukan fungsi aktivasi yang akan digunakan.

   Fungsi aktivasi yang akan digunakan adalah fungsi aktivasi *sigmoid biner*

dengan alasan *output* yang diinginkan (*desired output*) terletak antara 0 dan 1. Selain itu, fungsi aktivasi ini paling umum digunakan untuk metode *backpropagation*. (Puspitaningrum, 2006).

6. Algoritma *backpropagation* yang digunakan

6.1 Prosedur Pelatihan

Langkah 0. Inisialisasi bobot dan bias (sebaiknya diatur pada bilangan acak yang kecil).

a. Bobot-bobot antara lapisan *input* dengan lapisan tersembunyi, disebut v_{ij} dengan $i = 1,...,576$ dan $j = 1,...,576$

b. Bobot-bobot bias antara lapisan *input* dengan lapisan tersembunyi, disebut v_{0j} dengan $j = 1,...,576$

c. Bobot-bobot antara lapisan tersembunyi dengan lapisan *output*, disebut w_{jk} dengan $j = 1,...,576$ dan $k = 1,...,90$

d. Bobot-bobot bias antara lapisan tersembunyi dengan lapisan *output*, disebut w_{0k} dengan $k = 1,...,90$

Keseluruhan bobot di atas diinisialisasi dengan menggunakan bilangan *random* antara -0,5 dan 0,5.

Langkah 1. Jika *stopping condition* belum terpenuhi (kondisi berhenti apabila toleransi error terpenuhi atau ketika iterasi mencapai iterasi maksimum).

Umpan maju

Langkah 2. Setiap unit *input* $(X_i, i = 1,...,576)$ menerima sinyal *input* x_i dan menyebarkan sinyal tersebut pada seluruh unit lapisan di atasnya (unit *hidden*).

Langkah 3. Setiap unit *hidden* $(Z_j, j = 1,...,576)$, akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot, termasuk biasnya menggunakan persamaan (2.12) dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output*, dan unit *hidden* yang bersangkutan menggunakan persamaan (2.13), lalu mengirim sinyal *output* tersebut ke seluruh unit pada unit pada lapisan atasnya (unit *output*).

Langkah 4. Setiap unit *output* $(Y_k, k = 1,...,90)$ akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot, termasuk biasnya menggunakan persamaan (2.14) dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output* dan unit *output* yang bersangkutan menggunakan persamaan (2.15), lalu mengirim sinyal *output* ini ke seluruh unit pada unit *output*.

Propagasi balik *error*

Langkah 5. Setiap unit *output* $(Y_k, k = 1,...,90)$ menerima suatu target yang sesuai dengan *input training* untuk menghitung kesalahan (*error*) antara target dengan *output* yang dihasilkan jaringan

menggunakan persamaan (2.16). Faktor δ_k digunakan untuk menghitung koreksi *error* (Δw_{jk}) menggunakan persamaan (2.17) yang akan dipakai untuk memperbaharui w_{jk}. Selain itu, juga dihitung koreksi bias Δw_{0k} menggunakan persamaan (2.18) yang akan dipakai untuk memperbaharui w_{0k}. Faktor δ_k kemudian dikirimkan ke *lapisan* yang berada di bawahnya.

Langkah 6. Setiap unit *hidden* $(Z_j, j = 1,...,576)$ menjumlah *input* delta yang sudah berbobot menggunakan persamaan (2.19). Kemudian, hasilnya dikalikan dengan turunan dari fungsi aktivasi yang digunakan jaringan untuk menghasilkan faktor koreksi *error* (δ_j) menggunakan persamaan (2.20). Faktor δ_j digunakan untuk menghitung koreksi *error* (Δv_{ij}) menggunakan persamaan (2.21) yang nantinya akan dipakai untuk memperbaharui v_{ij}. Selain itu, juga dihitung koreksi bias Δv_{0j} menggunakan persamaan (2.22) yang nantinya akan dipakai untuk memperbaharui v_{0j}.

Langkah 7. (Pembaharuan bobot dan bias)

Setiap unit *output* $(Y_k, k = 1,...,90)$ akan memperbaharui bias dan bobotnya dari setiap unit *hidden* $(Z_j, j = 1,...,576)$ menggunakan persamaan (2.23). Demikian pula, setiap unit *hidden* $(Z_j, j = 1,...,576)$ akan memperbaharui bias dan bobotnya dari

setiap unit *input* $(X_i, i = 1,...,576)$ menggunakan persamaan (2.24).

Langkah 8. Memeriksa *stop condition*.

Apabila toleransi error belum terpenuhi atau ketika iterasi belum mencapai iterasi maksimum, kembali ke langkah 2.

Untuk mempermudah pembacaan langkah-langkah penyelesaian pada proses pelatihan *backpropagation* bisa dilihat pada Gambar 3.3.

6.2 Prosedur Pengujian

Setelah pelatihan, jaringan syaraf *backpropagation*, diaplikasikan dengan hanya menggunakan umpan maju dari algoritma pelatihan. Prosedur aplikasinya adalah sebagai berikut :

Langkah 0. Inisialisasi bobot (dari algoritma pelatihan).

Langkah 1. Untuk $i = 1,...,576$, atur aktivasi unit *input* x_i.

Langkah 2. Untuk setiap unit *hidden* $(Z_j, j = 1,...,576)$, akan dijumlahkan sinyal-sinyal *input* yang sudah berbobot, termasuk biasnya menggunakan persamaan (2.12) dan memakai fungsi aktivasi *sigmoid biner* untuk menghitung sinyal *output*, dan unit *hidden* yang bersangkutan menggunakan persamaan (2.13), lalu mengirim sinyal *output* tersebut ke seluruh unit pada unit pada lapisan atasnya (unit *output*).

Langkah 3. Untuk setiap unit *output* $(Y_k, k = 1,...,m)$ akan dijumlahkan

sinyal-sinyal *input* yang sudah berbobot, termasuk biasnya menggunakan persamaan (2.14) dan memakai fungsi aktivasi *sigmoid biner* untuk menghitung sinyal *output* dan unit *output* yang bersangkutan menggunakan persamaan (2.15), lalu mengirim sinyal *output* ini ke seluruh unit pada unit *output*.

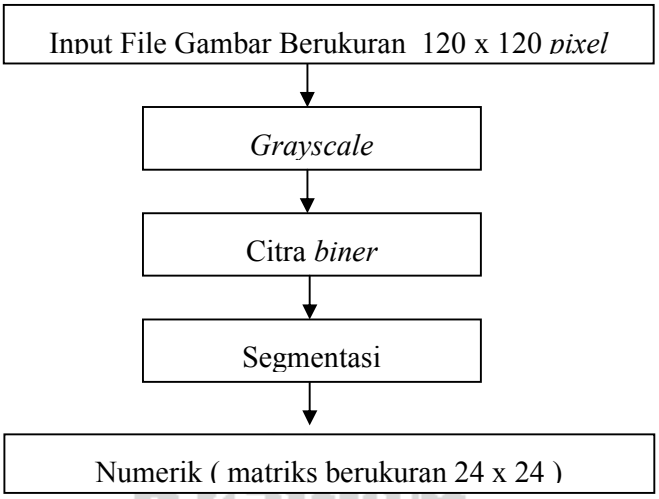
Langkah 4. Jika data *input* sudah mencapai *n*, maka proses pengujian berhenti.

Untuk mempermudah pembacaan langkah-langkah penyelesaian pada proses pengujian *backpropagation* bisa dilihat pada Gambar 3.4.

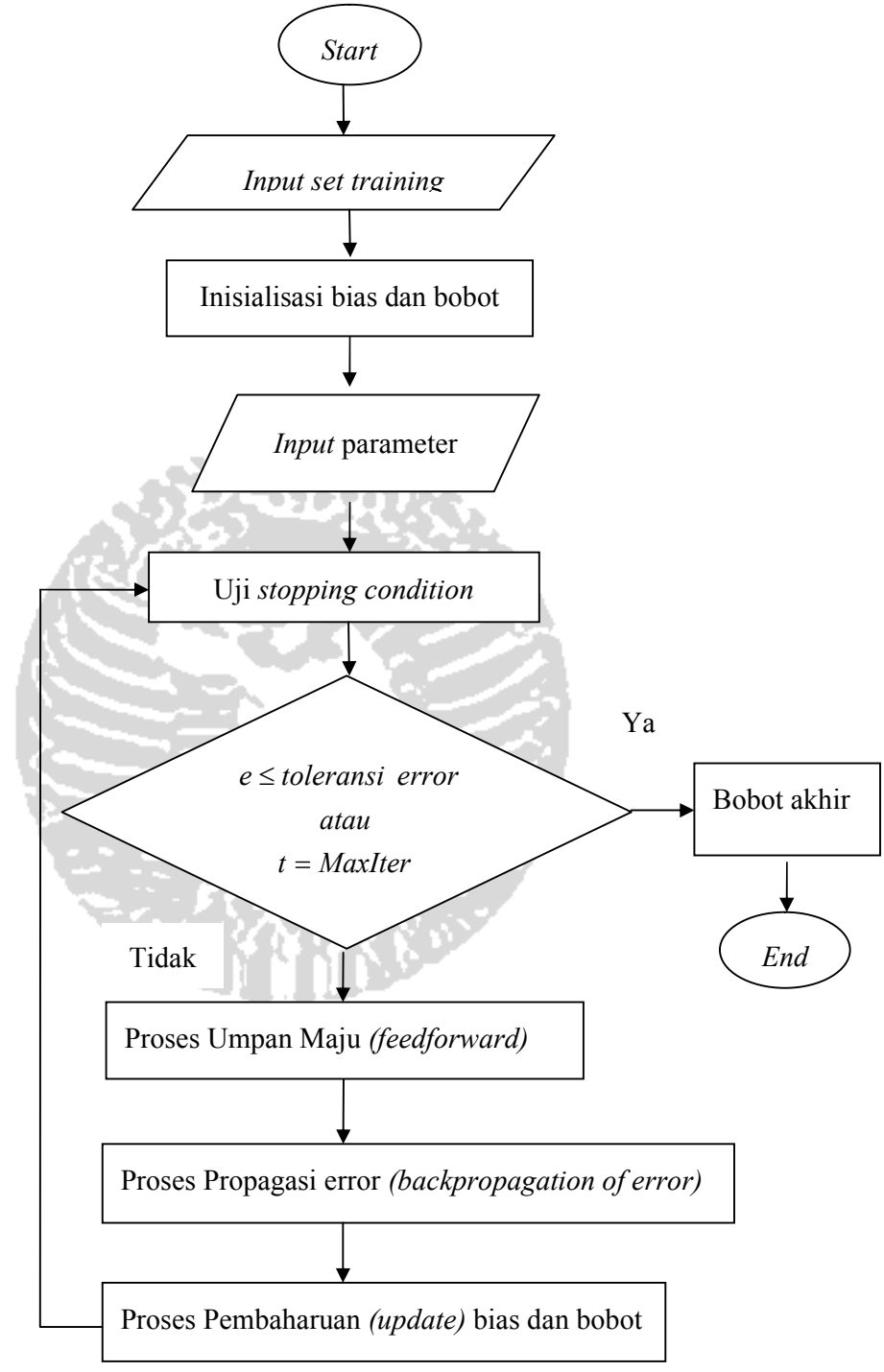
7. Melakukan uji validasi pada model jaringan saraf tiruan dengan metode *backpropagation* dengan memberikan data *input* baru pada jaringan, dan apakah model tersebut menghasilkan *output* yang sesuai dengan hasil yang di targetkan.

Untuk mempermudah pembacaan langkah-langkah penyelesaian pada proses pengenalan pola sidik jari bisa dilihat pada Gambar 3.5.

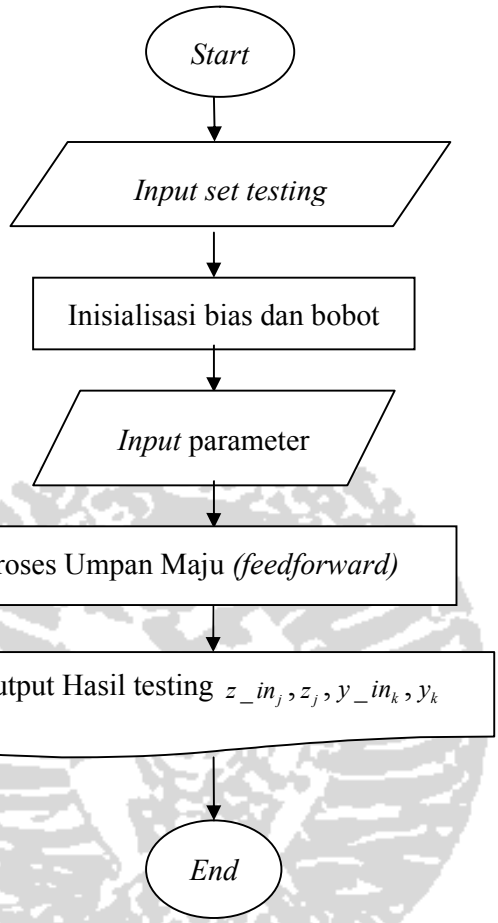
8. Membuat program untuk pengenalan pola sidik jari dengan menggunakan *software* visual basic 6.0.



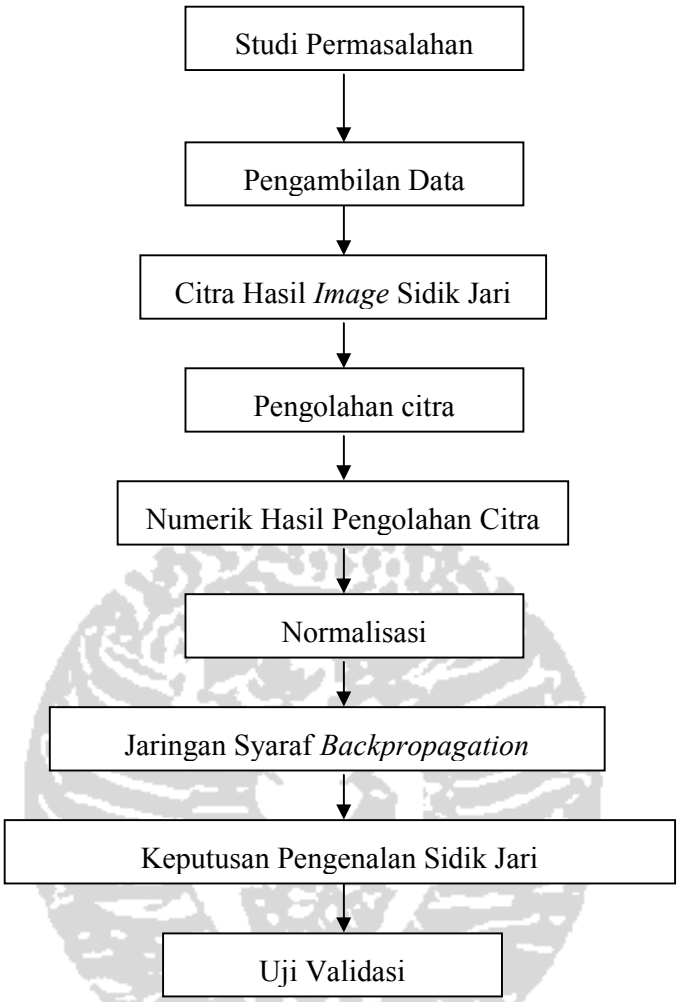
Gambar 3.2 : Diagram alir pengolahan citra



Gambar 3.3 : *Flowchart* dari algoritma *training backpropagation*



Gambar 3.4 : *Flowchart* dari algoritma *testing backpropagation*



Gambar 3.5 : Diagram alir pengenalan pola sidik jari

BAB IV

PEMBAHASAN

Pada bab ini akan diberikan beberapa penjelasan mengenai pengenalan pola sidik jari dengan menggunakan jaringan syaraf tiruan metode *backpropagation*.

4.1 Pengolahan Citra

Sampel acuan atau referensi yang akan diujikan pada skripsi ini adalah sampel berupa *image* sidik jari dengan jenis file JPEG dari beberapa mahasiswa. Jumlah *image* yang digunakan adalah 120 *image* sidik jari yang diambil dari 5 sidik jari tangan kiri dari 6 mahasiswa yang masing-masing diambil sebanyak 4 kali. Dari 120 *image* sidik jari tersebut dibagi penggunaannya untuk sampel pelatihan dan pengujian. Banyaknya sampel yang digunakan untuk pelatihan yaitu sebanyak 90 *image* sidik jari, sedangkan untuk validasi menggunakan 30 *image* sidik jari. Keseluruhan *image* sidik jari yang digunakan dalam skripsi ini dapat dilihat pada Lampiran 1.

Untuk mendapatkan data input pada proses pelatihan jaringan syaraf tiruan, gambar-gambar yang digunakan diubah menjadi data numerik dengan menggunakan proses pengolahan citra.

Seluruh hasil *image* sidik jari yang akan digunakan untuk pengenalan pola sidik jari harus berukuran sama. Dalam skripsi ini, ukuran *image* sidik jari yang digunakan ukurannya sama yaitu 120 x 120 *pixel*. Setelah seluruh *image* berukuran seragam, langkah selanjutnya yaitu *image* akan memasuki proses pengolahan citra.

4.1.1 Proses *Greyscale*

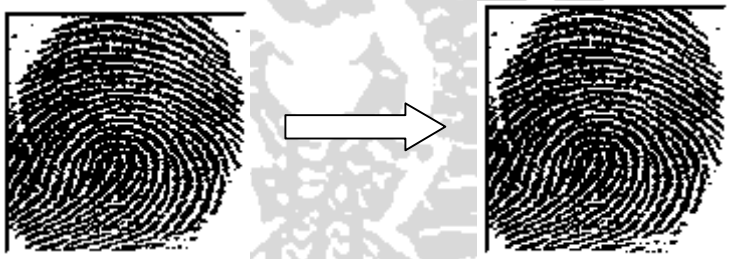
Proses pertama pada pengolahan citra adalah proses *grayscale* yaitu proses mengubah citra berwarna menjadi citra abu-abu (*grey*). Proses ini bertujuan untuk mengubah gambar yang berwarna sehingga gambar yang digunakan pada pengolahan citra hanya berwarna antara hitam sampai putih. Dari hasil grayscale ini, nilai numerik yang akan diperoleh yaitu berada pada interval [0-255] dengan 0 merupakan nilai numerik dengan intensitas warna paling gelap, sedangkan 255 merupakan nilai numerik dengan intensitas warna paling terang. Sedangkan numerik yang terletak diantara 0 dan 255 merupakan representasi nilai numerik dari warna *grey* yang dihasilkan. Proses pengubahan warna tersebut dilakukan dengan menggunakan *software Visual Basic 6.0*. Gambar 4.1 adalah prosedur program untuk mendapatkan citra *grayscale*.

```
Sub Prosedur_Greyscale ()

    For i = 1 To jumlah baris piksel ' gambar adalah untuk menampung gambar asli
        For j = 1 To jumlah kolom piksel
            warna = gambar.Point(i,j)        ' untuk mengembalikan warna gambar asli
            r = warna And RGB (255,0,0)              ' untuk menghitung warna merah
            g = Int(warna And RGB (0,255,0)/256)    ' untuk menghitung warna hijau
            b =Int( Int(warna And RGB(0,0,255)/256)/256) ' untuk menghitung warna biru
            x = (r + g + b) / 3        ' untuk menghitung warna greyscale
            ' gambar_grey adalah untuk menampung gambar hasil greyscale
            gambar_grey(i,j) = RGB (x,x,x)    ' untuk mengatur warna hasil greyscale
        Next j
    Next i
End Sub
```

Gambar 4.1 : Prosedur Program *Greyscale*

Perubahan citra dari citra asli menuju proses *greyscale* dapat dilihat pada Gambar 4.2 dibawah ini :



Gambar 4.2 : Perubahan gambar asli menuju citra *grayscale*

4.1.2 Proses Citra *Biner*

Pada pengolahan citra, proses selanjutnya setelah proses *grayscale* adalah proses citra *biner* yaitu citra yang banyak dimanfaatkan untuk keperluan *pattern recognition* yang sederhana seperti pengenalan angka atau pengenalan huruf. Citra yang digunakan pada proses citra *biner* merupakan citra hasil *grayscale*. Perhitungan

nilai citra *biner* ini dilakukan dengan menggunakan persamaan (2.2). Gambar 4.3 adalah prosedur program untuk mendapatkan citra dari hasil proses citra *biner*.

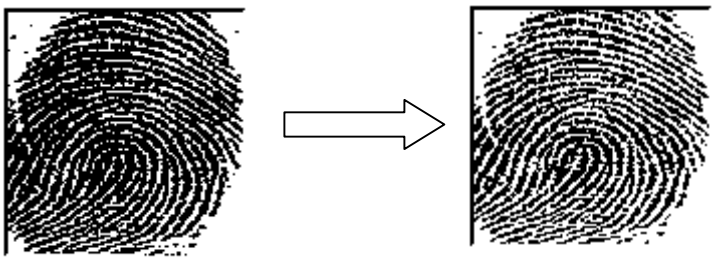
```
Sub Prosedur_Citra Biner ()
    Prosedur_Greyscale      ' memanggil prosedur grayscale

    For i = 1 To jumlah baris piksel  ' gambar adalah untuk menampung gambar asli
        m = m + 1  ` untuk menghitung ukuran piksel pada baris matriks image
        n = 0  ` untuk menginisialkan nilai pada kolom matriks image
        For j = 1 To jumlah kolom piksel
            warna = gambar.Point(i,j)        ' untuk mengembalikan warna gambar asli
            r = warna And RGB (255,0,0)              ' untuk menghitung warna merah
            g = Int(warna And RGB (0,255,0)/256)   ' untuk menghitung warna hijau
            b =Int( Int(warna And RGB(0,0,255)/256)/256) ' untuk menghitung warna biru
            n = n + 1 ` untuk menghitung ukuran piksel pada kolom matriks image
            x = (r + g + b) / 3        ' untuk menghitung warna grayscale
            xr = xr + x   ' untuk menghitung citra biner
            wx(m,n) = x `  untuk menginisialkan nilai grayscale
            ' gambar_grey adalah untuk menampung gambar hasil greyscale
            gambar_grey(i,j) = RGB (x,x,x)    ' untuk mengatur warna hasil greyscale
        Next j
    Next i

    For i = 1 To jumlah baris piksel  ' gambar adalah untuk menampung gambar asli
        For j = 1 To jumlah kolom piksel
          If wx(I,j)<xr then x = 0 else x = 255 `  untukmenentukan hasil citra biner
          ' gambar_biner adalah untuk menampung gambar hasil citra biner
          ` gambar_biner(i.j) = (15*(i -1)+1, 15*(j - 1) + 1) RGB (x,x,x)
         ' untuk mengatur warna hasil citra biner
        Next j
    Next i
End Sub
```

Gambar 4.3 : Prosedur Program Citra *Biner*

Perubahan citra dari proses *grayscale* menuju proses citra *biner* dapat dilihat pada Gambar 4.4 dibawah ini :



Gambar 4.4 : Perubahan citra *greyscale* menuju citra *biner*

4.1.3 Proses Segmentasi

Langkah selanjutnya pada proses pengolahan citra adalah proses segmentasi yaitu membagi citra menjadi segmen-segmen yang lebih kecil sehingga diharapkan untuk pengolahan datanya dapat menjadi lebih cepat. Hasil dari tahap segmentasi citra ini berupa data *pixel* yang menyusun batas dari suatu daerah atau semua titik dalam suatu daerah. Pada proses-proses sebelumnya citra berukuran 120 x 120 *pixel*, setelah citra mengalami proses segmentasi ukurannya berubah menjadi 24 x 24 *pixel*. Hal ini bertujuan untuk mempermudah dalam pengolahan data selanjutnya karena ukuran numerik yang akan diperoleh lebih sedikit.

Ukuran citra yang diperoleh dari proses segmentasi merupakan ukuran matriks yang akan diperoleh setelah citra dirubah kedalam bentuk numerik. Gambar 4.5 adalah prosedur program untuk mendapatkan citra hasil dari proses segmentasi.

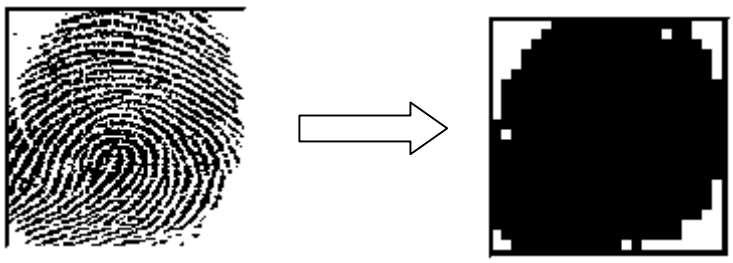
```
Sub Prosedur_Segmentasi ()
      Prosedur_Greyscale      ` memanggil prosedur grayscale
      Prosedur_Citra Biner       ` memanggil prosedur histogram equalization

      For i = 1 To n1                  ` n1 adalah lebar gambar asli
          For j = 1 To n2    ` n2 adalah panjang gambar asli
                  If ((i = 1) And (j = 1)) Or ((i = 1) And (j Mod bil_segmen = 0)) Then `
      bil.segmen adalah bilangan yang diinginkan untuk men-segmen gambar
                      If (i < n1) And (j< n2) Then
                              hitung nilai_greytotal
                              ratasegmen = Round(nilai_greytotal/bil_segmen^2)   ` menghitung
      rata-rata segmentasi
                              For a = 0 To bil_segmen – 1
                                  For b = 0 To bil_segmen – 1
                                       gambar_segmen(i.j) = (15*(i+a-1)+1),15*(j+b-1)+1),
      RGB(ratasegmen,ratasegmen,ratasegmen)   'gambar_segmen untuk menampung gambar
      hasil segmentasi
                                 Next b
                              Next a
                      End If
                  End If

                If ((i Mod bil_segmen = 0) And (j = 1)) Or ((i Mod bil_segmen = 0) And (j Mod
      bil_segmen = 0)) Then
                      If (i < n1) And (j< n2) Then
                              hitung nilai_greytotal
                              ratasegmen = Round(nilai_greytotal/bil_segmen^2)
                              For a = 0 To bil_segmen – 1
                                  For b = 0 To bil_segmen – 1
                                       gambar_segmen(i,j) = (15*(i+a-1)+1),15*(j+b-1)+1),
      RGB(ratasegmen,ratasegmen,ratasegmen)
                                 Next b
                              Next a
                      End If
                  End If
          Next j
      Next i
End Sub
```

Gambar 4.5 : Prosedur Program Segmentasi

Perubahan citra dari proses citra *biner* menuju proses segmentasi dapat dilihat pada Gambar 4.6 di bawah ini :



Gambar 4.6 : Perubahan citra *biner* menuju citra segmentasi

Setelah citra mengalami proses segmentasi, citra diubah ke bentuk numerik berupa matriks.

4.2 Proses Normalisasi

Numerik hasil segmentasi belum dapat digunakan pada proses pelatihan jaringan syaraf. Nilai numerik dari hasil segmentasi dirubah menjadi numerik dengan interval antara 0 sampai 1. Perubahan numerik ini dilakukan dengan membagi setiap numerik atau setiap elemen matriks dengan nilai tertinggi dari interval grayscale yaitu 255. Selain merubah range interval numerik, proses normalisasi juga merubah ukuran matriks menjadi satu kolom. Jadi matriks yang dihasilkan dari proses normalisasi yaitu matriks yang berukuran 576 x 1 dengan numerik yang bernilai antara 0 sampai 1. Matriks hasil normalisasi ini merupakan data *input* untuk proses pelatihan jaringan syaraf tiruan.

```
Sub Prosedur_Normalisasi ()
  Prosedur_Segmentasi
  ratasegmen = Val(ratasegmen) / 255  'nilai 255 adalah nilai derajat keabuan maksimum
End Sub
```

Gambar 4.7 : Prosedur program normalisasi

4.3 Prosedur Program Jaringan Syaraf *Backpropagation*

Pada metode *backpropagation* terdapat 2 tahap, yaitu *training* data dan *testing* data. *Training* data bertujuan untuk memperoleh bobot-bobot yang optimal, yang akan digunakan sebagai inisialisasi bobot pada *testing* data.

Prosedur program jaringan syaraf tiruan untuk *training* data dan *testing* data dengan metode *backpropagation* secara umum dapat dilihat pada Gambar 4.8 dan Gambar 4.9.

```
 Prosedur Training Data dengan Metode Backpropagation
{
            input set training();
            inisialisasi bias dan bobot();
            input parameter();
    For epoch=1 To batas_epoch
          For data=1 To jml_data
                  umpan maju();
                  propagasi error();
                  pembaharuan bobot dan bias();
          Next data
                  mean square error();
          if(error <= 0.01 or epoch = max_iterasi);
          end if
    Next epoch
  }
```

Gambar 4.8 : Prosedur program proses *training* data

```
Prosedur Testing Data dengan Metode Backpropagation

{
              input set testing;
              load bias dan bobot dari training data;
              input parameter();
                  umpan maju();

}
```

Gambar 4.9 : Prosedur program proses *testing* data

4.3.1 *Input Set Training*

Prosedur pengisian *input set training* dapat dilihat di bawah ini. *Jml_data* adalah jumlah data yang digunakan, *x_input* adalah banyak unit *input*, z_*hidden* adalah banyak unit *hidden*, *y_output* adalah banyak unit *output*. *Numerik* hasil pengolahan citra {*data_in (i,j)*} adalah data *training* untuk *input*, unit *input* ke-*i* pada data ke-*j*. Prosedur program proses *set input training* data secara umum dapat dilihat pada Gambar 4.10.

```
Prosedur Input Set Training()

{
            jml_data=jumlah data; // jumlah data
            x_input=input; // banyak unit input
            z_hidden=hidden; // banyak unit hidden
            y_output=output; // banyak unit output
            tr=target output; // target jaringan

            For n=1 To jml_data
                   For i=1 To x_input
                       data_in(i,n); // input
                   Next i
            Next n
}
```

Gambar 4.10 : Prosedur program proses *set input training* data

4.3.2 Inisialisasi Bobot dan Bias

Pada prosedur ini akan ditentukan nilai dari masing-masing bobot *V_ij* dan *W_jk*. Bobot *V_ij* adalah bobot antara unit *input* ke-*i* dan unit *hidden* ke-*j*. Sedangkan *W_jk* adalah bobot antara unit *hidden* ke-*j* dan unit *output* ke-*k*. Bobot-bobot tersebut diinisialisasi dengan rumus, *(0,5 - (-0,5)) * Rnd + (-0,5)*, dimana *Rnd* adalah bilangan *random*. Rumusan tersebut adalah fungsi untuk memperoleh bilangan *random* antara -0,5 sampai dengan 0,5. Prosedur inisialisasi bobot dan bias serta prosedur bilangan *random* dapat dilihat pada Gambar 4.11 dan Gambar 4.12.

```
Prosedur Inisialisasi Bias and Bobot()

{
          // inisialisasi bobot V
          For i=1 To x_input
               For j=1 To z_hidden+1
                   V_ij=Bilangan Random();
               Next j
          Next i

         // inisialisasi bobot W
          For j=1 To x_hidden+1
               For k=1 To 1
                   W_jk=Bilangan Random();
               Next k
         Next j
}
```

Gambar 4.11 : Prosedur program proses inisialisasi bobot dan bias

```
Prosedur Bilangan Random()

{
        Randomize
        (0.5 - (-0.5)) * Rnd + (-0.5)
}
```

Gambar 4.12 : Prosedur bilangan random

4.3.3 *Input* Parameter

Input parameter jaringan syaraf meliputi *input alpha*, *batas_epoch* dan *batas_error*. *Alpha* adalah parameter yang mengontrol perubahan bobot selama *training* data (*learning rate*), sedangkan *batas_epoch* dan *batas_error* adalah batas iterasi dan batas *error*. Keduanya adalah batas yang digunakan untuk *stopping*

condition pada *training* data. Prosedur *input* parameter dapat dilihat pada Gambar 4.13.

```
Prosedur Input Parameter()

{
                Alpha=alpha; // learning rate
                batas_epoch=epoch maksimum; // batas iterasi
                batas_error=error maksimum; // batas error
}
```

Gambar 4.13 : Prosedur program *input* parameter

4.3.4 Fungsi Aktivasi

Berdasarkan persamaan 2.5 dan 2.6, prosedur memperoleh nilai dari fungsi aktivasi *sigmoid biner* dan nilai turunannya dapat dilihat pada Gambar 4.14 dan Gambar 4.15.

```
Prosedur Sigmoid Biner()

{
    f = 1 / (1 + Exp(-X));
}
```

Gambar 4.14 : Prosedur program *sigmoid biner*

```
Prosedur Differensial Sigmoid Biner()

{
    df = f(X) * (1 - f(X));
}
```

Gambar 4.15 : Prosedur program *differensial sigmoid biner*

4.3.5 *Mean Square Error*

Besarnya nilai *error* untuk *training* data dihitung dengan menggunakan metode *Mean Square Error*. Prosedur untuk mencari besarnya *Mean Square Error* dapat dilihat pada Gambar 4.16.

```
Prosedur Mean Square Error()

{
              Proses umpan maju();
                 // hitung error
                   Error=0;
              For data=1 To jml_data
                  For n=1 To jml_data
                      Error=Error+((tr-Y(k,n))^2);
                  Next n
                      MSE=sqr(Error/data);
              Next data
}
```

Gambar 4.16 : Prosedur program *Mean Square Error*

4.3.6 Umpan Maju

Pada proses umpan maju, setiap unit *input* (*data_in(i,j)*) akan menerima sinyal input dan akan menjumlahkan sinyal tersebut pada tiap unit *hidden*. Setiap unit *hidden*, akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot termasuk biasnya, $z\_in_j = v_{0j} + \sum_{i=1}^{x\_input} x_i v_{ij}$. Kemudian dengan menggunakan fungsi aktivasi *sigmoid biner* dengan σ = 1 untuk menghitung sinyal *output* dari unit *hidden* yang bersangkutan diperoleh *Z (j,k)*. Dalam setiap unit *output* dilakukan penjumlahan

sinyal-sinyal *hidden* yang sudah berbobot termasuk biasnya, $y\_in_k = w_{0k} + \sum_{j=1}^{z\_hidden} z_j w_{jk}$. Kemudian dengan menggunakan fungsi aktivasi *sigmoid biner* σ = 1 untuk menghitung sinyal *output* dari unit *hidden* yang bersangkutan diperoleh Y_k. Keseluruhan proses umpan maju tersebut dapat dilihat pada Gambar 4.17.

```
Prosedur Umpan Maju()

{
            // hitung Z
         For j=1 To z_hidden
             Sum=0;
             For i=1 To x_input
                 Sum=Sum+data_in(i,j)*V_ij;
             Next i
                 Z_in(j,k)=V_0j+Sum;
                 Z(j,k)=f(Z_in(j,k));
         Next j
           // hitung Y
         For k=1 To y_output
             Sum=0;
             For j=1 To z_hidden
                 Sum=Sum+Z(j,k)*W_jk;
             Next j


                 Y_in(n,k)=W_0k+Sum;


                 Yk=f(Y_in(n,k));
           Next k
}
```

Gambar 4.17 : Prosedur program umpan maju

4.3.7 Propagasi Balik *Error*

Langkah selanjutnya setelah proses umpan maju yaitu proses propagasi *error* (*error of backpropagation*). Pada proses ini, setiap unit *output* Y_k menerima suatu *target pattern* yang sesuai dengan *input training pattern* untuk menghitung kesalahan (*error*) antara target dengan *output* yang dihasilkan jaringan, $\delta_k = (t_k - y_k) f'(y\_in_k)$. Faktor *ss* (sebagai δ_k) digunakan untuk menghitung koreksi error *delta_Wjk* yang akan dipakai untuk memperbaharui *W_jk*, dimana $\Delta w_{jk} = \alpha \delta_k z_j$. Selain itu, juga dihitung koreksi bias *delta_Wok* yang akan dipakai untuk memperbaharui *W_ok*, dimana $\Delta w_{0k} = \alpha \delta_k$.

Setiap unit *hidden Z(i,j)* menjumlah *input ss* atau (δ_k) yang sudah berbobot, $\delta\_in_j = \sum_{k=1}^{y\_output} \delta_k w_{jk}$. Kemudian, hasilnya dikalikan dengan turunan dari fungsi aktivasi yang digunakan jaringan untuk menghasilkan faktor koreksi error *s_j* atau (β_j) dimana $\beta_j = \delta\_in_j f'(z\_in_j)$. Faktor *s_j* atau ($\beta_j$) digunakan untuk menghitung koreksi *error delta_Vij* yang akan dipakai untuk memperbaharui *V_ij*, di mana $\Delta v_{ij} = \alpha \beta_j x_i$. Selain itu, juga dihitung koreksi bias *detal_Voj* yang nantinya akan dipakai untuk memperbaharui *V_oj*, dimana $\Delta v_{0j} = \alpha \beta_j$. Prosedur program propagasi balik *error* dapat dilihat pada Gambar 4.18.

```
Prosedur Propagasi Balik Error()

{
              // hitung Del_W
         For k=1 To y_output
                   ss(k)=(tr-Y(n,k))*df(Y_in(n,k))
             For j=0 To z_hidden
                   delta_Wjk=alpha*ss*Z(i,j);
                   delta_W0k=alpha*ss;
             Next j
         Next k
             // hitung Del_V
         For j=1 To z_hidden
                 S_j=0;
             For k=1 To y_output
                 S_j=s_j+ss*W_jk;
             Next k
                      S_j=s_j*df(z_in(i,j));
                   For i=0 To x_input
                      delta_V0j=alpha*s_j;
                      delta_Vij=alpha*s_j*data_in(i,n)
                   Next i
         Next j
}
```

Gambar 4.18 : Prosedur program propagasi balik *error*

4.3.8 Pembaharuan Bobot dan Bias

Pada proses *update* bobot dan bias, setiap unit *output* Y_k akan memperbaharui bias dan bobotnya dari setiap unit *hidden*, $w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk}$. Demikian pula, setiap unit *hidden Z(j,k)* akan memperbaharui bias dan bobotnya dari setiap unit *input*, $v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}$. Prosedur program pembaharuan bobot dan bias dapat dilihat pada Gambar 4.19.

```
Prosedur Pembaharuan bias dan bobot()

{
             // hitung V
        For i=1 to x_input
            For j=1 to z_hidden
                V_newij=V_ij+delta_Vij;
                V_ij=V_newij;
            Next j
        Next i
            // hitung W
        For j=1 to z_hidden
            For k=1 to z_hidden
                W_newjk=W_jk+delta_Wjk;
                W_jk=W_newjk;
            Next k
        Next j
}
```

Gambar 4.19 : Prosedur program pembaharuan bias dan bobot

4.4 Implementasi Program

Berikut ini adalah cara pelatihan jaringan syaraf sederhana pada pengenalan pola sidik jari dengan menggunakan metode *backpropagation*. Dimana keseluruhan tampilan program dapat dilihat pada Lampiran 3. Jaringan terdiri atas 1 *input layer* dengan 576 unit *input*, 1 *hidden layer* dengan 576 unit *hidden* dan 1 *output layer* dengan 1 unit *output*.

Training set yang digunakan sebanyak 90 *image* sidik jari. *Learning rate* (α) yang digunakan adalah 0 sampai 1. Sebelum pelatihan, *stopping condition* harus ditentukan terlebih dahulu. Pada skripsi ini, dihentikan jika *error* telah mencapai 0.01

atau *epoch* = 500. Bias dan bobot diinisialisaikan dengan bilangan *random* antara -0.5 sampai dengan 0.5.

Kemudian, hasil *training* sistem aplikasi model jaringan syaraf tiruan dengan berbagai macam kombinasi *learning rate* 0,1 sampai 0,9 dapat dilihat pada Tabel 4.1.

Tabel 4.1 : Hasil *training* dengan kombinasi *learning rate*

No.	*Learning Rate*	*Error*	*Epoch*	Hasil Persentase
1	0.1	0.01	376	100%
2	0.2	0.01	350	100%
3	0.3	0.01	339	100%
4	0.4	0.01	325	100%
5	0.5	0.01	319	100%
6	0.6	0.01	326	100%
7	0.7	0.01	309	100%
8	0.8	0.01	286	100%
9	0.9	0.01	257	100%

Keterangan : Jumlah data = 90, Max *epoch* = 500 dan Max *error* = 0,01

Dari Tabel 4.1 di atas, keberhasilan dalam mengenali pola sidik jari sebesar 100% adalah semua *learning rate*. Sehingga semua *learning rate* tersebut digunakan dalam proses uji *validasi.*

Hasil uji *validasi* sistem aplikasi model jaringan syaraf tiruan dengan berbagai macam kombinasi *learning rate* 0,1 sampai 0,9 dapat dilihat pada Tabel 4.2.

Tabel 4.2 : Hasil uji *validasi* dengan kombinasi *learning rate*

No.	*Learning Rate*	*Error*	*Epoch*	Hasil Persentase
1	0.1	0.01	376	63,33%
2	0.2	0.01	350	63,33%
3	0.3	0.01	339	63,33%
4	0.4	0.01	325	66,67%
5	0.5	0.01	319	66,67%
6	0.6	0.01	326	73,33%
7	0.7	0.01	309	73,33%
8	0.8	0.01	286	73,33%
9	0.9	0.01	257	76,67%

Keterangan : Jumlah data = 30, Max *epoch* = 500, dan Max *error* = 0,01

Setelah dilakukan pelatihan, semakin besar nilai *learning rate*, maka jumlah *epoch* semakin kecil, kecuali pada *learning rate* 0,5 dan 0,6. Dari Tabel 4.2 di atas, keberhasilan dalam memprediksi hasil pengenalan pola sidik jari sebesar 76,67% pada *learning rate* 0,9. Pengambilan penggunaan *learning rate* 0,9 dengan alasan bahwa penggunaan *learning rate* tersebut menghasilkan *error* sebesar 0,01 pada *epoch* ke-257 dan prosentase validasi sebesar 76,67%. Sedangkan penggunaan *learning rate* yang lain menghasilkan *error* sebesar 0,01 pada *epoch* lebih besar dari 257 dengan prosentase *validasi*nya yang berbeda-beda, sehingga berdasarkan dari Tabel 4.2 *learning rate* yang paling efisien digunakan adalah 0,9.

Kemudian diperoleh bias dan bobot akhir dari *input layer* ke *hidden l*ayer (v) serta bias dan bobot akhir dari *hidden layer* ke *output layer* (w). Tampilan potongan bobot optimal (matriks 15x60 terawal dari matriks 576x576) dapat dilihat pada Lampiran 2.

Setelah dilakukan pembelajaran, maka akan dilakukan pengujian terhadap 30 *image* sidik jari. Hasil pengujiannya dapat dilihat pada Tabel 4.3.

Tabel 4.3 : Hasil uji *validasi* data

No	Gambar	Nama	Target *Output*	Hasil *Output*	Pola Sidik Jari	Keterangan
1	Sidik Jari 1	Dadang D.P.P	0.98433011	0.98433011	Jari Kelingking	Benar
2	Sidik Jari 2	Dadang D.P.P	0.99666858	0.99666858	Jari Manis	Benar
3	Sidik Jari 3	Dadang D.P.P	0.9942452	0.9942452	Jari Tengah	Benar
4	Sidik Jari 4	Dadang D.P.P	0.99871769	0.99871769	Jari Telunjuk	Benar
5	Sidik Jari 5	Dadang D.P.P	0.99483951	0.99683414	Ibu Jari Kiri	Salah
6	Sidik Jari 6	Nurul Fitriyah	0.99280487	0.99280487	Jari Kelingking	Benar
7	Sidik Jari 7	Nurul Fitriyah	0.99762284	0.99808973	Jari Manis	Salah
8	Sidik Jari 8	Nurul Fitriyah	0.99092216	0.99092216	Jari Tengah	Benar
9	Sidik Jari 9	Nurul Fitriyah	0.99554423	0.99554423	Jari Telunjuk	Benar
10	Sidik Jari 10	Nurul Fitriyah	0.99585694	0.99585694	Ibu Jari	Benar
11	Sidik Jari 11	Furqon S.	0.99456086	0.99456086	Jari Kelingking	Benar
12	Sidik Jari 12	Furqon S.	0.99788837	0.99788837	Jari Manis	Benar
13	Sidik Jari 13	Furqon S.	0.99725381	0.99725381	Jari Tengah	Benar
14	Sidik Jari 14	Furqon S.	0.98860274	0.98860274	Jari Telunjuk	Benar
15	Sidik Jari 15	Furqon S.	0.98834631	0.98241349	Ibu Jari	Salah
16	Sidik Jari 16	Ifan Pradhana	0.99555022	0.99555022	Jari Kelingking	Benar
17	Sidik Jari 17	Ifan Pradhana	0.99604109	0.98925009	Jari Manis	Salah
18	Sidik Jari 18	Ifan Pradhana	0.99802244	0.99802244	Jari Tengah	Benar
19	Sidik Jari 19	Ifan Pradhana	0.99925273	0.99923118	Jari Telunjuk	Salah
20	Sidik Jari 20	Ifan Pradhana	0.97274169	0.97274169	Ibu Jari	Benar
21	Sidik Jari 21	Napoli Andita	0.9993365	0.98316422	Jari Kelingking	Salah
22	Sidik Jari 22	Napoli Andita	0.99666659	0.99666659	Jari Manis	Benar
23	Sidik Jari 23	Napoli Andita	0.99563247	0.99563247	Jari Tengah	Benar
24	Sidik Jari 24	Napoli Andita	0.99321518	0.99321518	Jari Telunjuk	Benar
25	Sidik Jari 25	Napoli Andita	0.99463196	0.99463196	Ibu Jari	Benar
26	Sidik Jari 26	Novan E.A	0.99052182	0.99052182	Jari Kelingking	Benar
27	Sidik Jari 27	Novan E.A	0.99230378	0.99230378	Jari Manis	Benar
28	Sidik Jari 28	Novan E.A	0.98392588	0.99213292	Jari Tengah	Salah

No	Gambar	Nama	Target *Output*	Hasil *Output*	Pola Sidik Jari	Keterangan
29	Sidik Jari 29	Novan E.A	0.99785605	0.99785605	Jari Telunjuk	Benar
30	Sidik Jari 30	Novan E.A	0.97179019	0.97179019	Ibu Jari	Benar

Dari Tabel 4.3 di atas dapat dilihat bahwa 76,67% *test* terhadap pola-pola sidik jari tersebut dapat dikenali pola sidik jari (akurat). Ketepatan jaringan syaraf tiruan mengenali pola-pola tersebut ditentukan oleh bobot optimal yang diperoleh pada *training* data.

BAB V

PENUTUP

5.1 Kesimpulan

1. Telah dapat disusun metode pengolahan citra yang digunakan untuk pengenalan pola sidik jari dari hasil *image* sidik jari dengan jenis file berupa JPG. Pengolahan citra pada hasil *image* sidik jari dapat dilakukan pada gambar/citra berukuran 120x120 *pixel*. Dimana hasil akhir dari pengolahan citra berupa matriks berukuran 576 x 1.
2. Telah dapat disusun metode *backpropagation* yang digunakan untuk melakukan pengenalan pola sidik jari dari hasil *image* sidik jari, dengan data berupa gambar JPG. Dalam skripsi ini hanya terdiri dari 3 *layer*, yaitu sebuah *input layer*, sebuah *hidden layer*, dan sebuah *output layer*. Dimana jumlah unit *input layer* sebanyak 576 node, *hidden layer* sebanyak 576 node, dan *output layer* sebanyak 1 node.
3. Dengan menggunakan program, pada proses *training* menggunakan *image* sidik jari sebanyak 90 *image*, dengan *learning rate* 0,9, max epoch sebanyak 500 dan max error sebesar 0,01 diperoleh hasil training sukses pada epoch ke-257 dengan error sebesar 0,01. Kemudian validasi untuk 30 *image* sidik jari, diperoleh prosentase sebesar 76,67% gambar sesuai dengan *target*.

5.2 Saran

Untuk penelitian selanjutnya, penulis berharap bahwa pengenalan pola sidik jari menggunakan jaringan syaraf tiruan *backpropagation* ini dapat dikembangkan untuk pengenalan pola sidik jari yang mungkin lebih teliti atau dengan menggunakan metode yang lebih baik dari penelitian ini agar dapat digunakan langsung oleh instansi yang memerlukan.

DAFTAR PUSTAKA

Anang, S., 2009, *Pengenalan Pola Angka Menggunakan Jaringan Saraf Tiruan Metode Backpropagation*, Skripsi, Departemen Matematika, Universitas Airlangga, Surabaya.

Anonim 1, 2010, *Permohonan Perumusan Dan Identifikasi Sidik Jari (Daktiloskopi) ,* http://www.Depkumham.go.id, akses Maret 2008.

Basuki, A., Palandi, J., Fatchurrochman, 2005, *Pengolahan Citra Menggunakan Visual Basic*, Penerbit Graha Ilmu, Yogjakarta.

Budhi,G., Gunawan,I., Jaory,S., 2004, *Metode Jaringan Saraf Tiruan Back Propagation pada Citra Digital,* http://www.petra.ac.id, akses: 22 Juni 2009.

Fausett, L., 2003, *Fundamental of Neural Network Architecture, Algorithm, and Application*, Printice-Hall, Inc, London.

Haykin, S., 1994, *Neural Network A Comprehensive Foundation*, Macmillan College Publishing Co., New York.

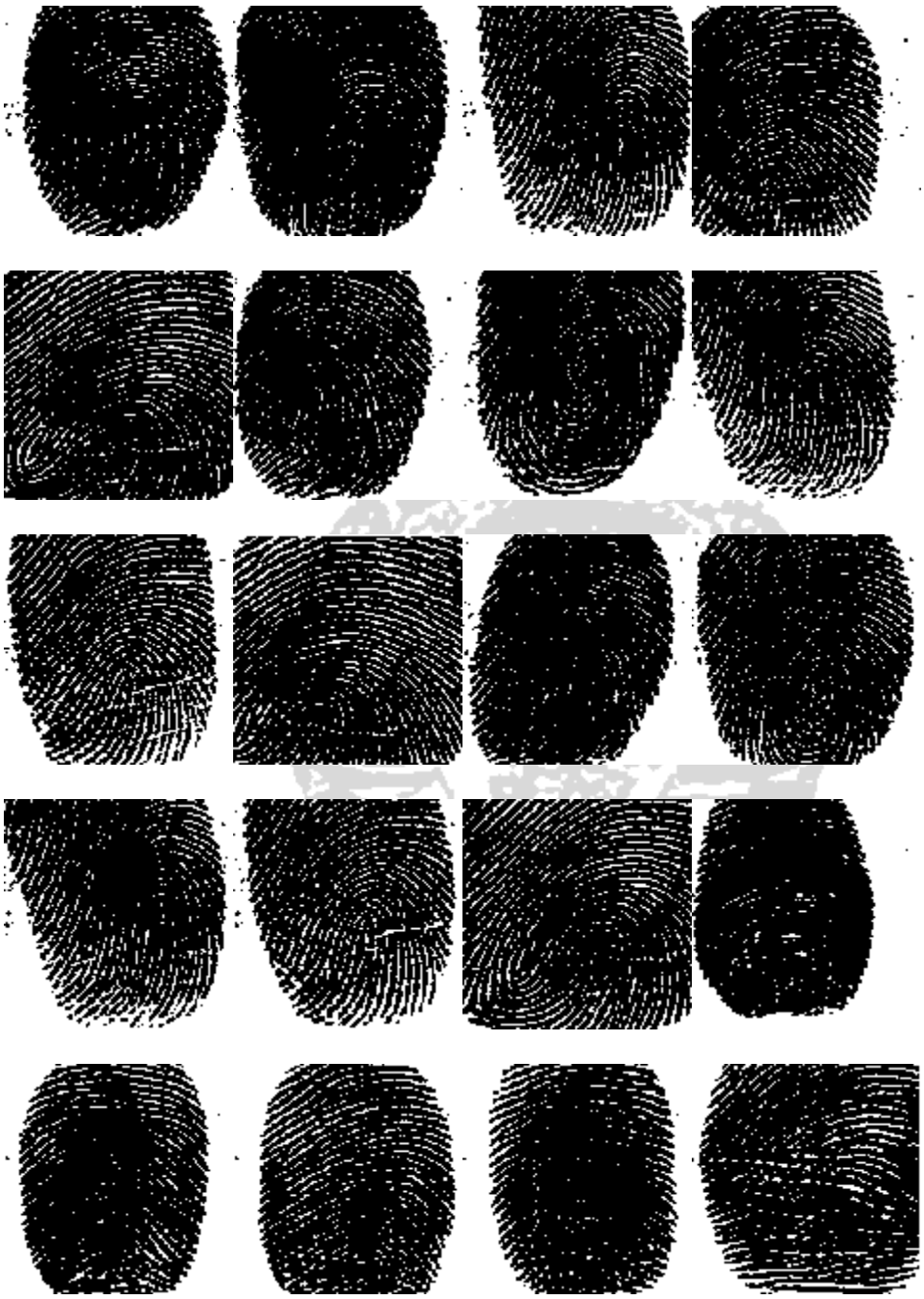
Kusumadewi, S., 2006, *Jaringan Syaraf Tiruan*, Penerbit Graha Ilmu, Jogjakarta.

Puspitaningrum, Diyah, 2006, *Pengantar Jaringan Saraf Tiruan*, Penerbit Andi, Yogyakarta.

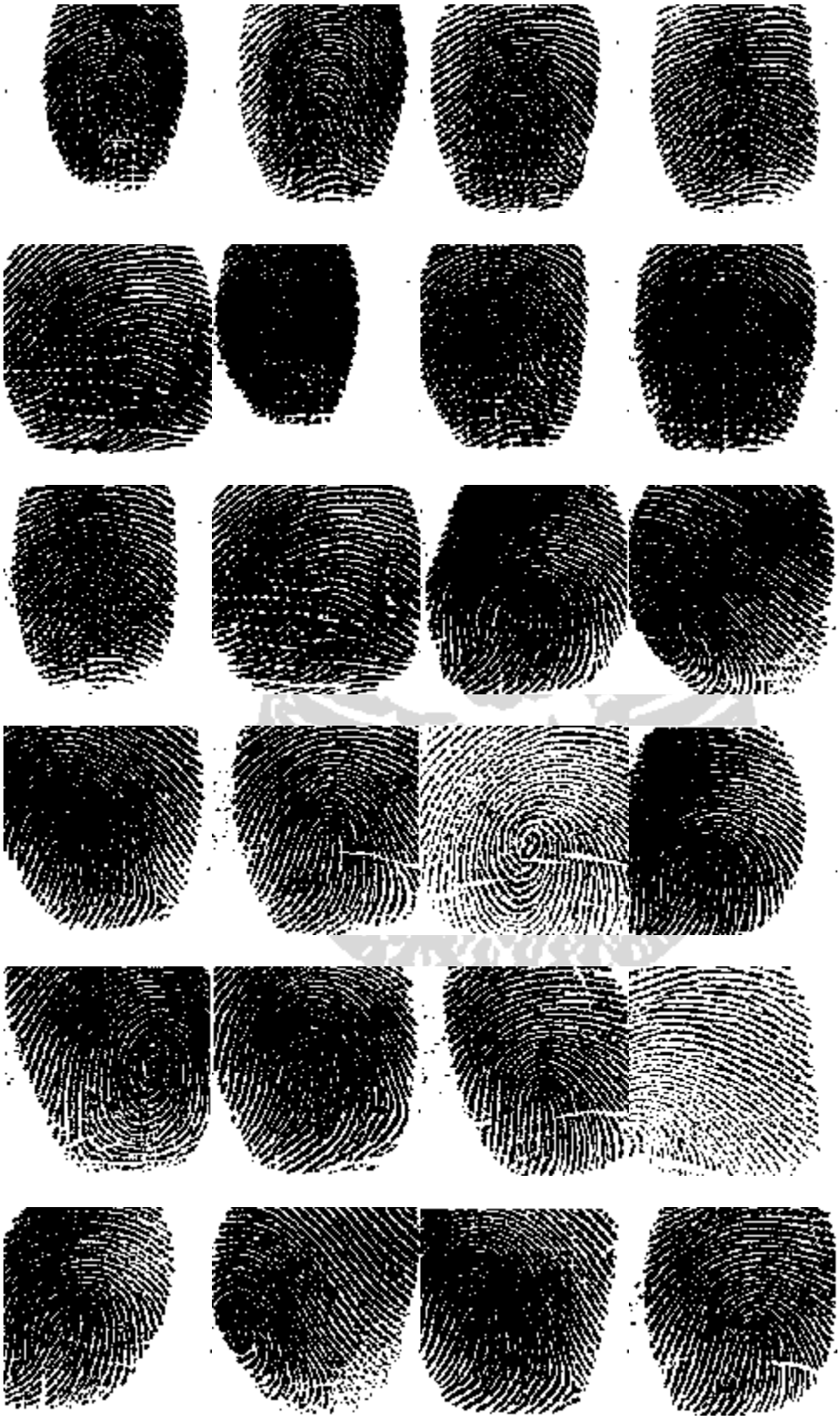
Suta Wijaya, Gede Pasek, 2004, *Pengenalan Citra Sidik Jari Berbasis Transformasi Wavelet dan Jaringan Syaraf Tiruan,* Jurnal Teknik Elektro, Universitas Mataram.

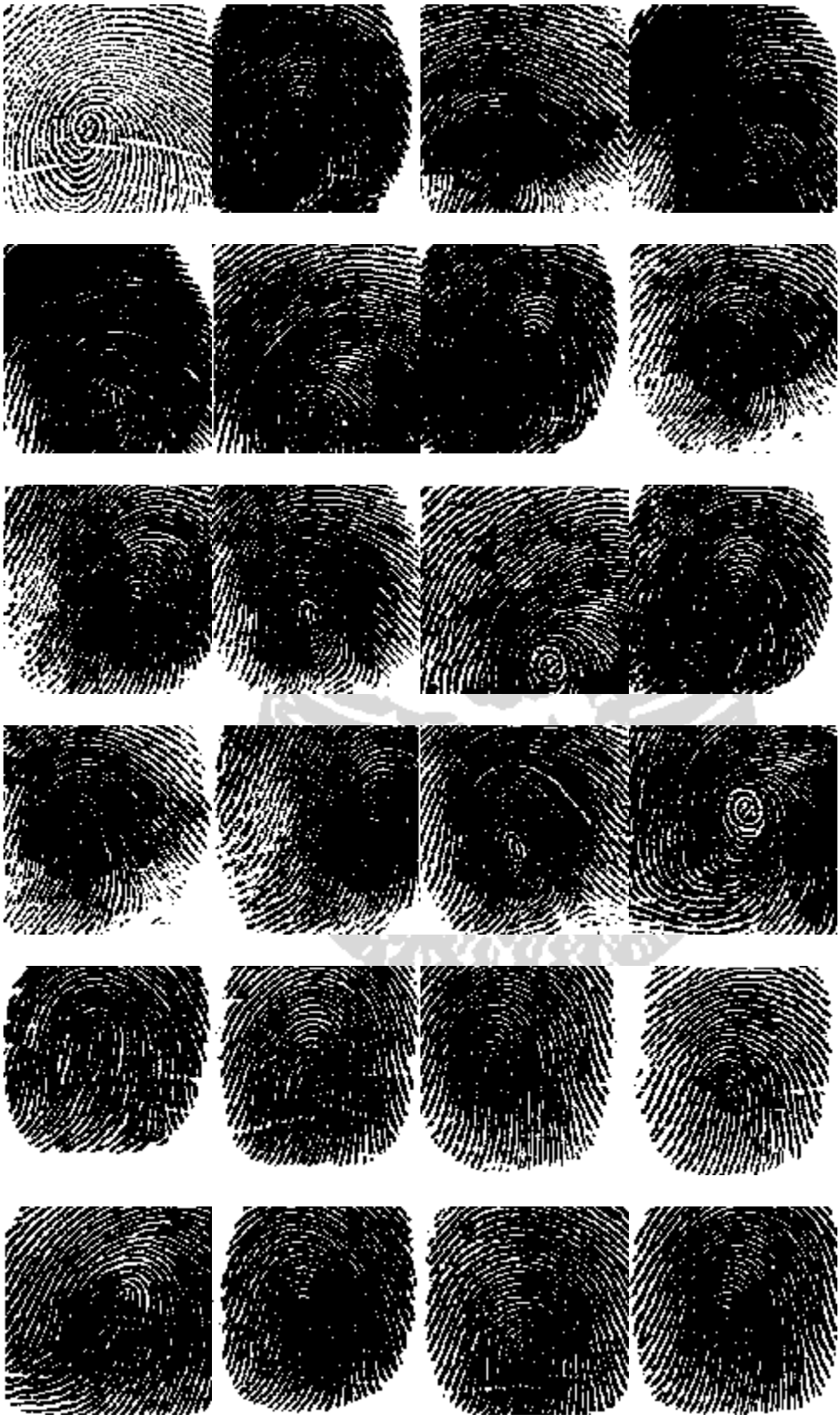
Lampiran 1 : *Image* sidik jari yang digunakan dalam pengenalan pola sidik jari

Image Sidik Jari Untuk Training



Lampiran 1- 2





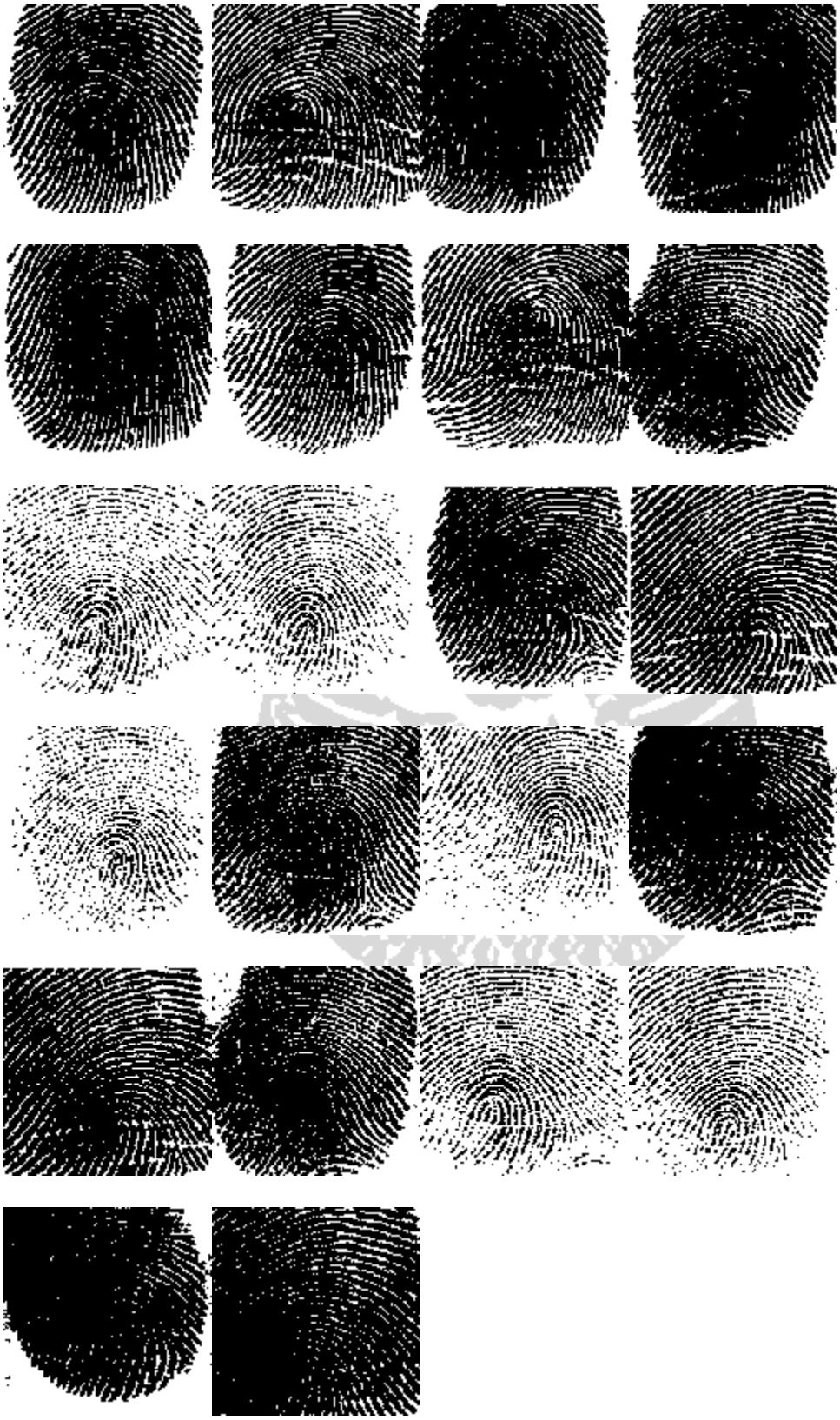
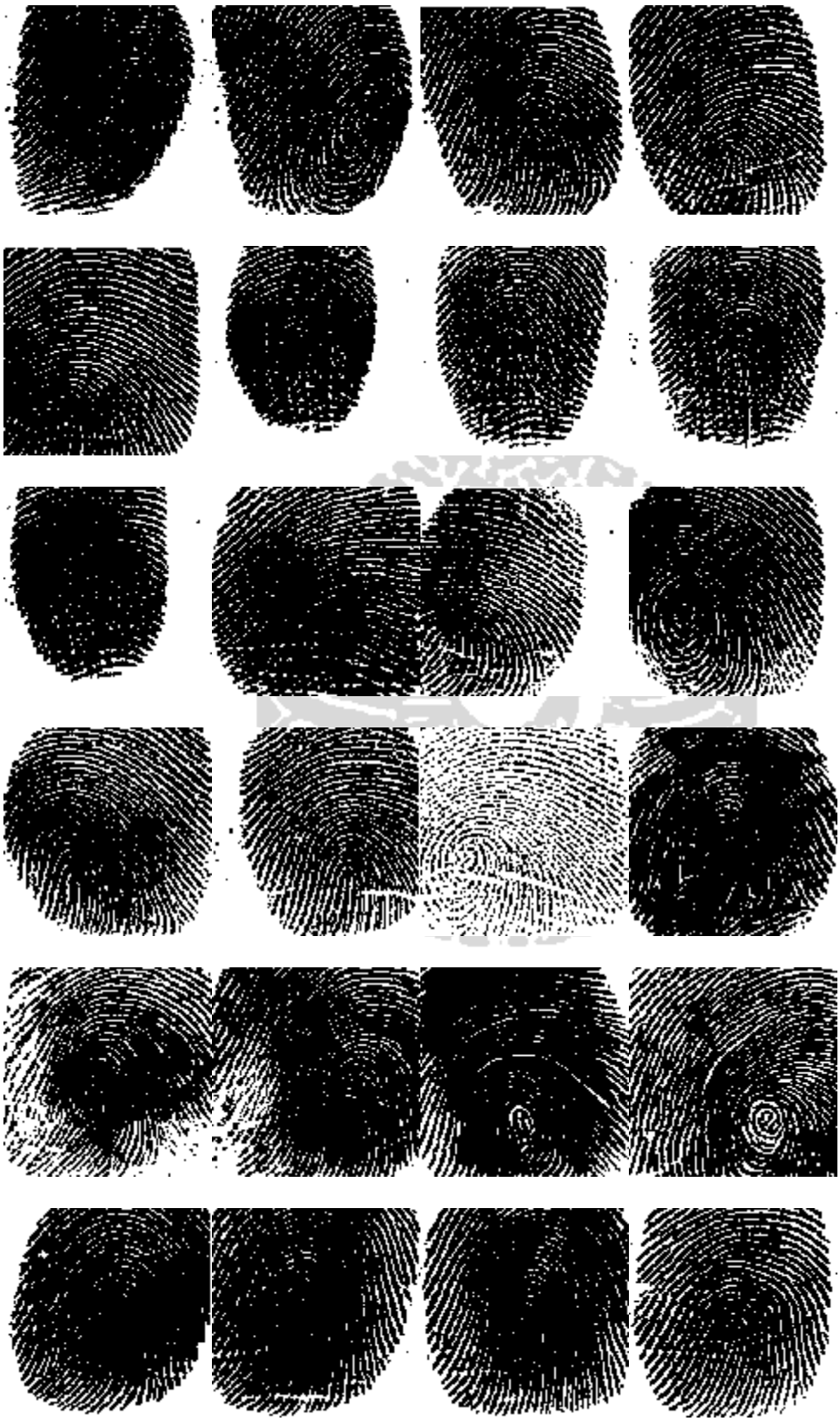


Image Sidik Jari Untuk Validasi



Lampiran 2 : Sebagian tampilan dari bobot optimal (matriks 15 x 60 terawal dari matriks berukuran 576 x 576)

V_ij	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.4787	-0.4884	0.4246	-0.3535	0.1102	0.2959	0.1137	0.2194	0.174	-0.3524	-0.2425	0.2064	0.4105	0.4858	0.1772
2	0.4327	-0.3512	-0.2379	0.1492	-0.2282	-0.3424	0.4226	-0.1904	-0.2445	0.1398	-0.3085	-0.1903	0.4748	0.4007	-0.082
3	0.2255	0.0701	-0.139	0.4792	-0.2428	0.2088	-0.0539	-0.3894	-0.4692	0.4002	-0.212	-0.0235	0.3361	-0.4595	-0.0983
4	0.3204	-0.0044	0.168	0.1625	0.328	0.3336	-0.0374	-0.3194	0.0826	0.0954	-0.2591	-0.2665	0.0273	0.1553	0.2703
5	0.3031	-0.4118	-0.3766	-0.4698	-0.2779	-0.4382	0.3393	0.4886	0.4819	-0.4821	0.3427	0.1283	-0.1839	0.1462	0.3933
6	0.4899	-0.11	-0.4346	-0.1384	0.158	0.2567	0.3682	-0.3025	0.0577	-0.0862	-0.4064	-0.3658	-0.1996	-0.1445	-0.2249
7	-0.2878	0.1836	-0.4979	-0.4987	0.1828	0.4715	0.4922	0.1101	0.1969	0.1731	0.3595	-0.0171	-0.0173	0.1189	0.0652
8	0.1348	0.3935	0.3416	-0.1322	-0.4516	-0.3841	0.4253	-0.0581	-0.3831	-0.258	0.4438	-0.475	0.4677	-0.1578	-0.1742
9	0.2859	0.4291	0.0616	0.349	-0.1812	-0.2469	-0.1012	-0.3251	-0.4683	-0.0671	0.2323	-0.4287	-0.3955	0.069	-0.1616
10	-0.4864	-0.4858	-0.3769	-0.1492	-0.1563	-0.1824	0.0972	0.4787	0.2004	-0.1453	0.1798	0.1022	0.3562	0.386	0.3549
11	-0.0245	0.1209	0.4619	0.4469	0.4483	-0.3662	0.3329	-0.0038	0.1421	0.0432	0.0853	0.4374	-0.1983	-0.1199	-0.1703
12	-0.4077	0.4021	0.3129	-0.4591	0.0346	-0.1557	0.0703	-0.1555	-0.0885	0.3037	0.0625	-0.1643	-0.1992	-0.2498	0.2698
13	0.4214	0.1915	-0.1545	0.0496	-0.237	0.1728	-0.288	-0.4653	-0.4926	-0.0185	0.2377	-0.1329	0.4842	0.0201	-0.4563
14	-0.1672	0.1119	0.429	-0.1612	0.0535	0.4945	-0.3373	0.4465	-0.0555	-0.242	-0.1712	-0.2184	0.3299	-0.2423	0.377
15	0.0903	-0.0774	-0.178	0.4626	-0.058	0.0456	-0.129	-0.2986	0.1464	0.436	0.3925	-0.0507	0.2931	0.3403	-0.4002

Lampiran 2-2

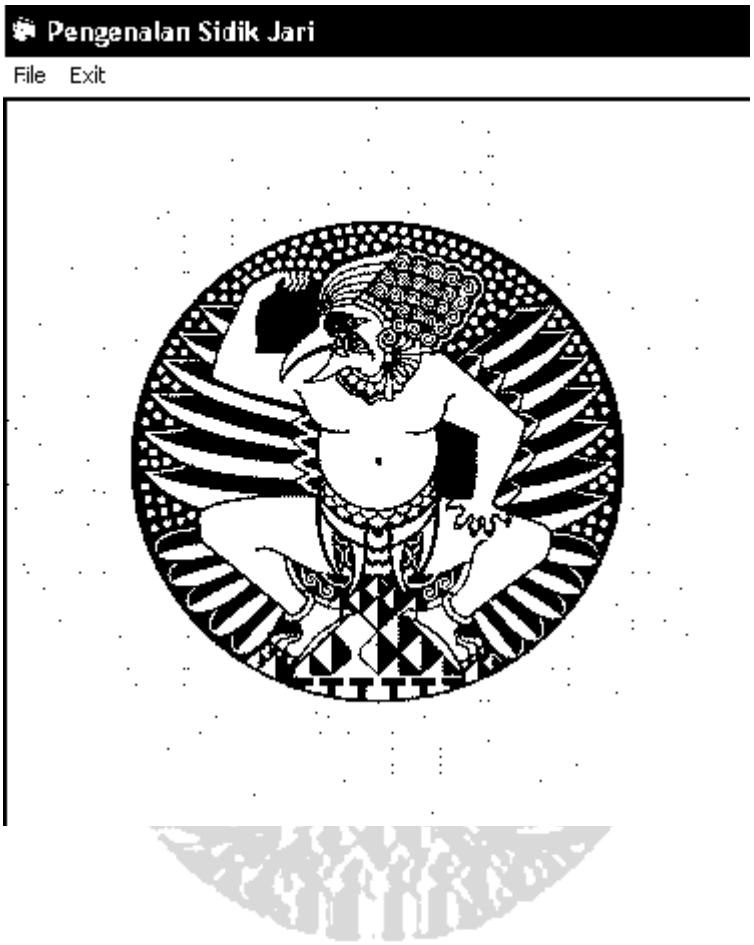
V_ij	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0.4098	-0.3839	-0.1874	0.1949	-0.121	0.0957	0.3535	-0.2884	-0.324	0.3661	0.4485	0.0359	-0.1088	0.1094	0.1923
2	0.3053	-0.1465	0.4969	0.4484	-0.2895	0.2891	0.3139	0.3916	0.4018	-0.4565	-0.1056	-0.3652	-0.3296	-0.1168	0.0167
3	0.4227	0.464	0.2727	-0.1892	0.0187	-0.2157	-0.2473	-0.249	-0.052	0.0743	-0.3354	-0.3769	-0.3507	-0.4036	0.4865
4	-0.1539	0.102	0.1463	0.2922	-0.1271	-0.4906	0.1637	0.4122	0.3254	-0.2695	-0.2827	0.2405	-0.0495	-0.2096	-0.0645
5	-0.3139	0.4673	-0.1403	0.3297	0.4115	0.3898	-0.3141	0.3979	-0.4174	-0.2968	0.407	-0.171	-0.229	-0.105	-0.0491
6	-0.4764	0.4711	-0.0954	0.2092	0.1795	-0.4808	0.0416	-0.0292	0.3306	-0.1013	-0.018	0.1784	-0.1759	0.4593	0.0049
7	-0.1235	0.3477	-0.3476	0.2826	-0.338	0.0066	-0.3549	0.3497	0.4655	0.1593	-0.2374	0.473	0.0586	0.3194	-0.0199
8	0.409	0.491	0.4491	-0.1356	0.0429	0.3555	-0.4897	-0.1424	0.1111	0.2735	-0.4084	0.4987	0.3615	-0.4829	0.3299
9	0.4752	0.4292	-0.1236	-0.0995	-0.1479	-0.4374	-0.1752	-0.4127	0.4612	-0.0701	-0.4704	-0.119	-0.1285	0.2775	0.3921
10	-0.0273	0.448	0.3294	-0.1152	0.2565	0.1173	0.3379	0.0756	0.238	-0.0987	-0.3719	-0.2068	-0.227	-0.1266	0.0584
11	0.3291	0.0795	0.2036	0.3419	0.4849	0.1129	-0.1332	0.2658	0.3639	0.3528	0.0324	-0.4336	-0.2889	-0.3581	-0.0619
12	-0.099	-0.4414	-0.431	-0.3222	-0.3377	-0.4152	-0.2367	-0.3328	-0.195	0.3389	0.4076	-0.3447	0.156	0.1056	0.3508
13	-0.3003	-0.3123	0.4458	-0.2639	-0.0205	0.4805	-0.4386	-0.2653	0.4628	0.0466	0.4559	0.3508	0.4812	0.0943	-0.105
14	0.4479	-0.461	0.0844	-0.199	0.1675	-0.2036	-0.108	0.154	0.2335	-0.137	-0.0237	-0.1853	0.2138	-0.0738	0.1028
15	-0.2926	0.457	-0.2008	-0.4258	0.1223	0.0268	0.1627	-0.2817	-0.3735	-0.2176	-0.1437	0.4592	-0.1704	0.2238	0.396

V_ij	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
1	0.4614	-0.091	0.3756	-0.1747	0.0955	-0.0767	0.0362	0.3738	-0.063	-0.2969	0.487	-0.4589	0.2044	-0.3915	-0.3987
2	0.1094	-0.0741	0.4704	0.0223	0.3059	0.1489	0.0035	-0.1055	0.0406	-0.4053	-0.3924	-0.4138	0.4701	-0.254	-0.0251
3	0.1958	0.2064	-0.0733	0.006	-0.4406	-0.0103	0.0327	0.086	-0.1716	-0.4539	-0.3983	-0.3197	0.0949	-0.2602	-0.381
4	-0.2168	-0.4414	-0.2265	0.0178	0.0823	0.1015	-0.4105	-0.0619	0.1066	0.001	-0.0109	0.3979	-0.1076	-0.2733	-0.1708
5	-0.332	0.3583	0.2741	0.0314	-0.149	0.301	-0.3262	0.0393	-0.4658	0.1466	0.405	0.1826	-0.3286	0.2988	0.4592
6	-0.4895	0.0782	-0.4131	0.0312	-0.0295	-0.2146	0.094	-0.0489	-0.0973	-0.1338	0.3606	0.203	0.2578	-0.3287	-0.2755
7	0.1801	0.4989	0.392	0.348	0.0231	0.0208	0.0018	0.0074	-0.4685	0.1762	0.0715	0.4559	0.4157	0.2249	-0.3501
8	-0.4827	-0.3165	0.1835	0.1367	-0.4304	0.4363	0.2108	0.1509	0.3795	0.027	-0.3566	0.4422	0.1286	-0.2615	0.1479
9	-0.3444	0.227	-0.1238	0.4906	0.0046	0.2628	0.4744	-0.2794	-0.1807	-0.4299	-0.499	0.1856	-0.3828	0.1124	-0.4811
10	0.2285	-0.0082	-0.4664	0.4566	-0.0866	0.252	-0.0507	0.1985	-0.014	0.342	0.146	0.036	-0.2933	-0.0659	-0.2522
11	0.2333	0.4486	-0.4575	-0.0874	0.2337	-0.1843	-0.0853	-0.3588	-0.2847	-0.2403	-0.2553	-0.319	0.4596	-0.4235	0.2228
12	-0.0142	0.0121	0.0105	0.1695	0.0698	-0.2822	0.2857	0.3708	-0.2233	0.4936	-0.2149	-0.1545	0.0741	-0.4859	-0.2495
13	-0.3844	-0.0429	-0.1906	0.4616	0.3578	-0.3306	0.0585	-0.053	0.0324	-0.2309	-0.1583	-0.0093	-0.2015	-0.4552	-0.0035
14	-0.4978	-0.0966	-0.2589	-0.4153	-0.4576	-0.4716	0.0206	0.1113	0.383	0.1983	0.0763	-0.174	0.3595	-0.3902	-0.2911
15	0.2015	-0.3417	0.1485	-0.3454	-0.4695	-0.2524	0.4803	0.3768	-0.3347	0.0807	0.3468	-0.0552	0.1617	-0.1194	0.408

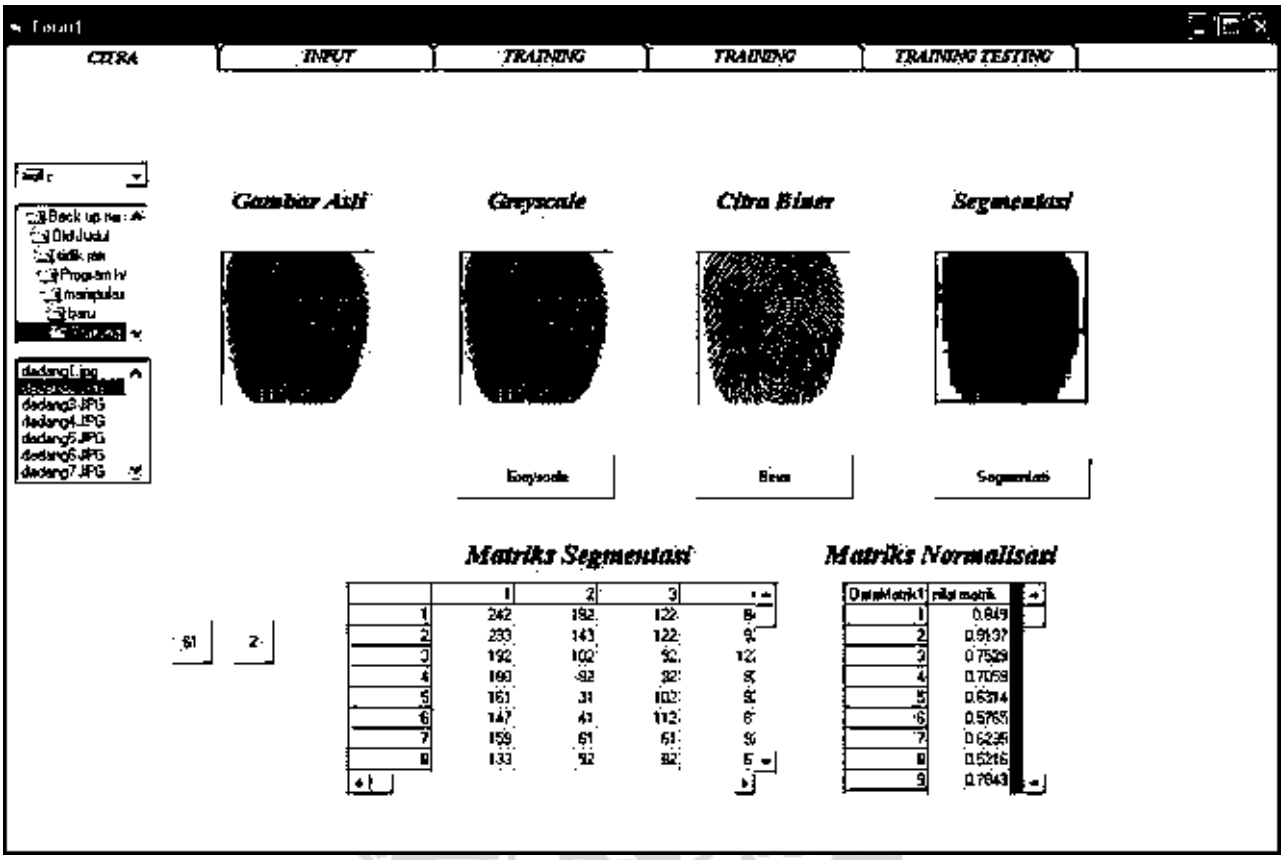
V_ij	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	-0.1522	-0.0858	-0.1443	-0.0669	0.0223	-0.2913	0.0333	0.1702	-0.358	-0.0656	-0.1327	-0.1626	0.0233	-0.0298	0.2372
2	-0.4019	0.3576	-0.4837	-0.2439	-0.1929	-0.4114	-0.2974	0.2002	-0.456	-0.064	-0.3109	0.0455	-0.078	0.0761	0.3998
3	0.0988	-0.3683	0.1432	0.1956	-0.4762	0.3453	-0.3227	-0.3532	-0.2249	0.4792	-0.4977	-0.1799	0.3261	-0.4332	0.1583
4	0.0294	-0.0663	-0.3759	-0.4112	-0.1516	-0.1766	-0.0073	0.1348	-0.0422	0.1437	0.4145	-0.2628	-0.4995	-0.3935	0.3316
5	-0.1226	-0.1574	-0.4662	-0.0382	0.3885	0.3318	-0.235	-0.1252	-0.2803	0.4552	0.0595	-0.0493	-0.2451	0.1393	-0.0969
6	-0.3988	0.1086	0.4355	0.1224	0.4141	0.4157	-0.0196	-0.0198	-0.3078	0.0429	-0.3325	0.3535	-0.1535	0.0229	-0.3983
7	0.0633	0.0197	-0.2247	-0.0105	-0.1631	0.0244	-0.3247	-0.1245	0.4374	0.4009	0.1523	0.203	0.4445	-0.3867	0.2045
8	0.4641	0.0972	-0.1919	-0.2214	-0.2431	-0.4451	-0.245	-0.0069	-0.0991	-0.0987	-0.041	-0.0822	-0.2803	0.0562	0.4882
9	-0.162	-0.3802	-0.0092	-0.0722	0.3692	0.0809	0.4599	-0.0825	0.3505	-0.1356	0.1929	-0.3292	-0.3763	0.2729	-0.2021
10	0.1747	0.2235	0.2873	-0.132	-0.397	-0.4122	0.2007	0.2498	-0.3308	0.0197	0.2167	-0.3429	-0.0898	0.4242	0.0738
11	0.2268	-0.4148	-0.1986	0.0714	0.1268	-0.4995	0.3494	0.1234	0.1483	0.1073	-0.0937	0.1499	0.493	0.2534	-0.4314
12	-0.4284	-0.0807	0.2361	-0.173	-0.433	0.3992	-0.023	0.3287	0.0926	0.1212	-0.2577	-0.2459	0.1058	-0.1459	0.3043
13	-0.0282	-0.1529	0.4384	0.15	0.1492	-0.02	0.3108	-0.0774	0.0521	0.4371	-0.0424	0.3811	0.121	-0.0248	-0.1257
14	-0.3456	-0.2892	-0.0004	0.3493	-0.1175	-0.1553	0.3473	0.4487	-0.0155	-0.1466	-0.1708	0.1194	-0.2983	0.2796	-0.4793
15	0.3944	-0.1342	-0.2503	0.4343	-0.3197	0.1168	0.0611	0.1345	0.1917	0.2002	-0.4008	0.2339	-0.482	0.0508	0.4499

Lampiran 3 : *Form* program pengenalan pola sidik jari dengan menggunakan jaringan syaraf *backpropagation*

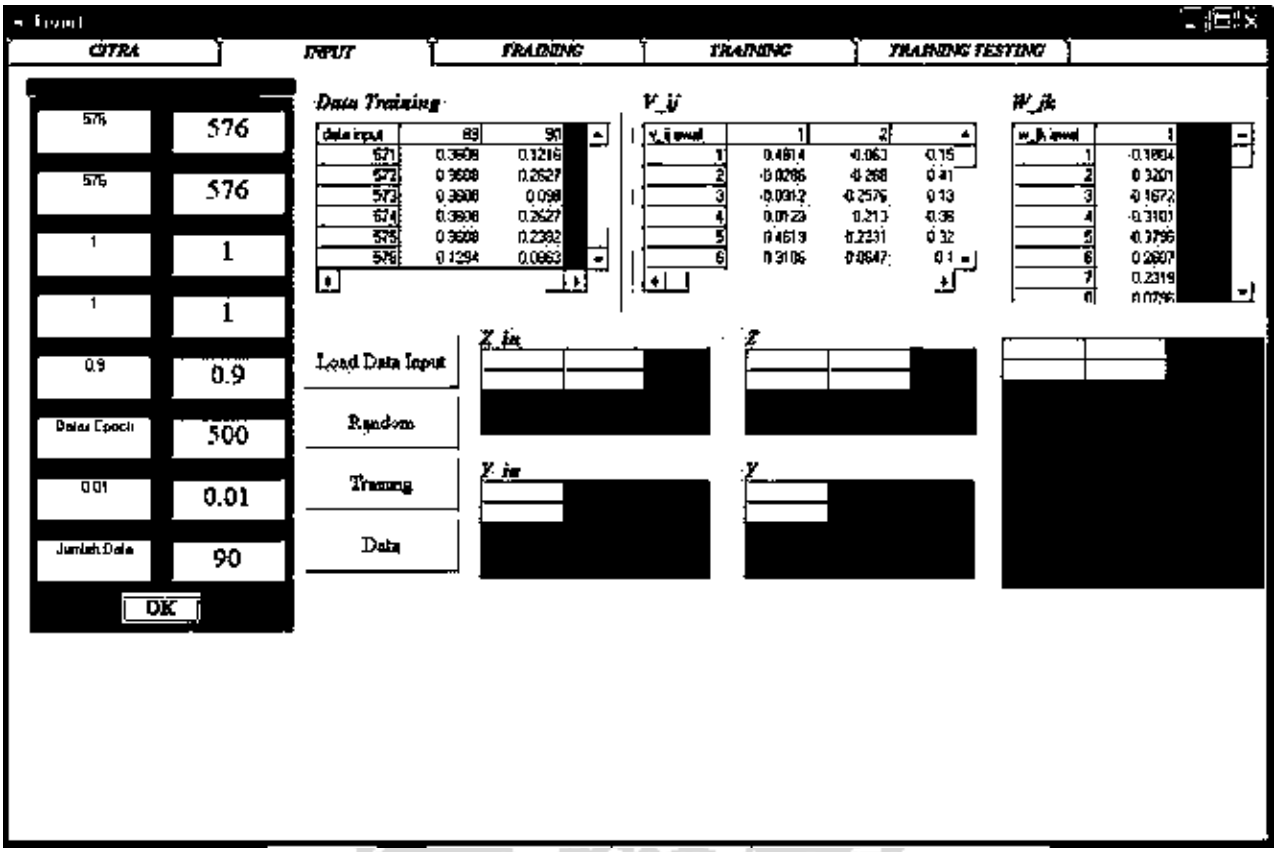
Form_Menu



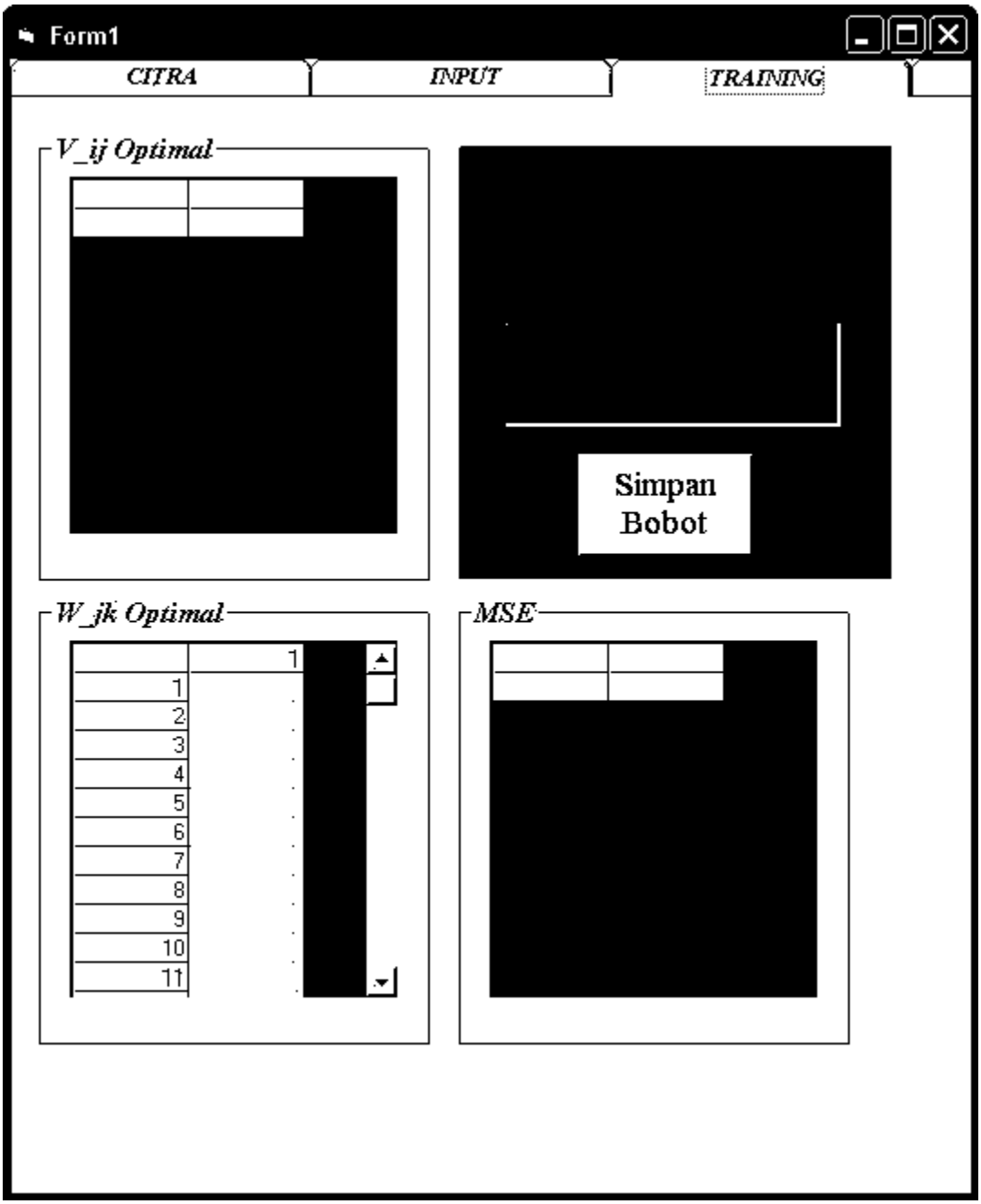
Form1 *SSTab*_Citra



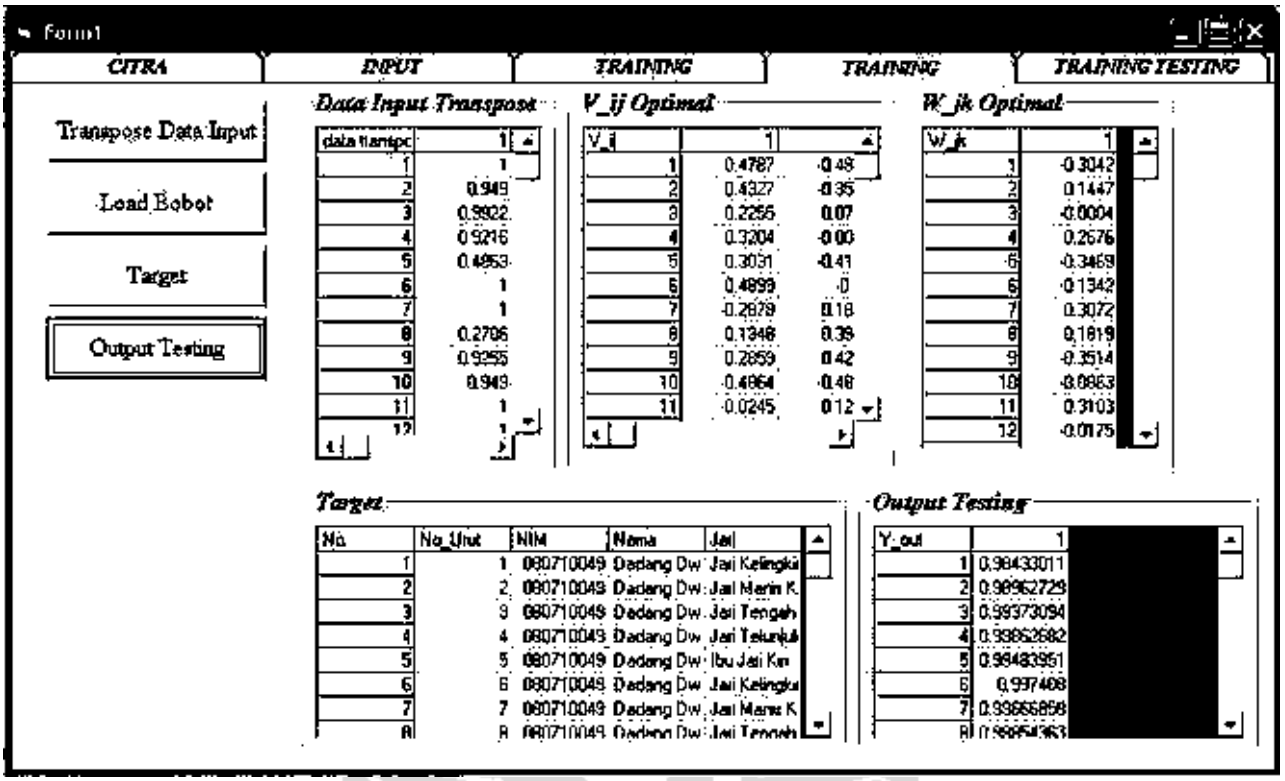
Form1 *SSTab_* Input



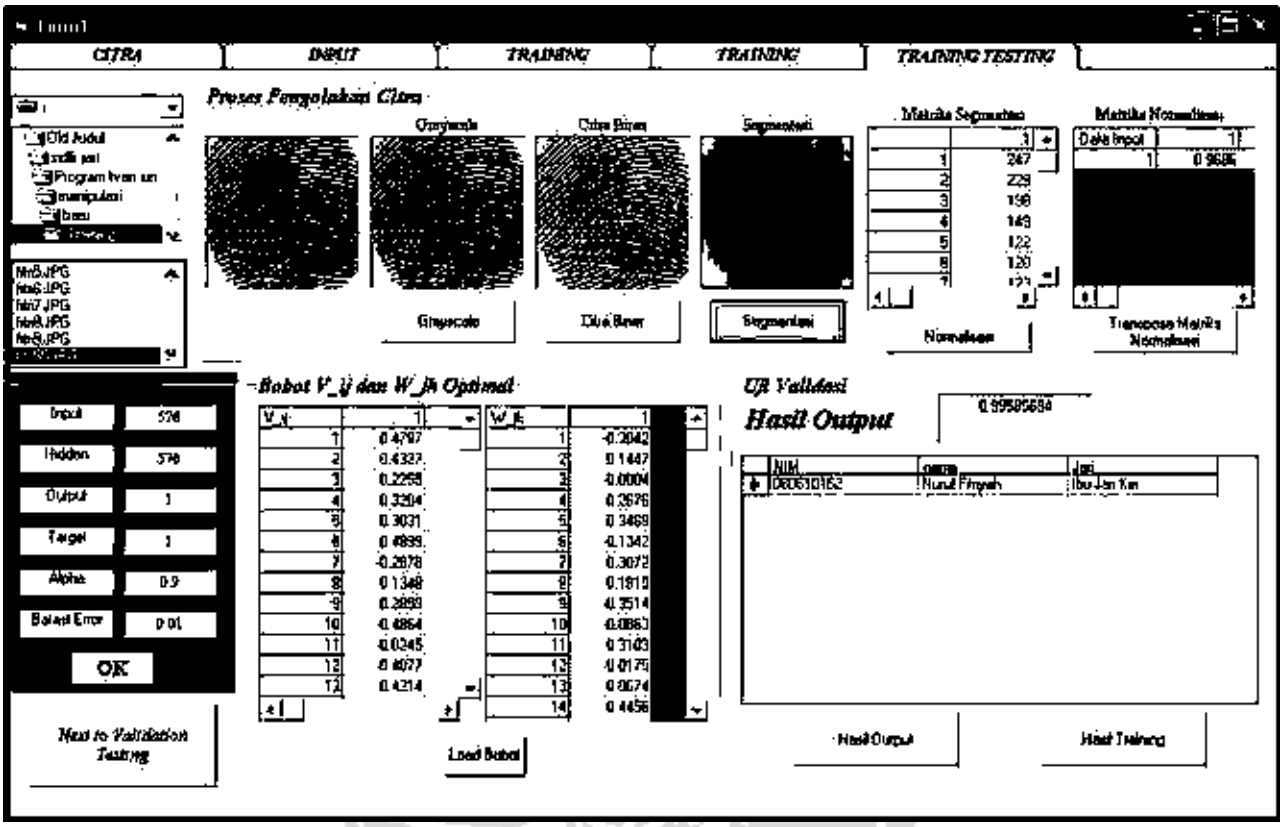
Form1 *SSTab_ Learning*



Form1 *SSTab_ Training*



Form1 *SSTab_ Testing Training*



Form2 *SSTab_ Validasi*