# PENGHITUNG JUMLAH SEL DARAH MERAH MENGGUNAKAN JARINGAN SARAF TIRUAN DENGAN METODE BACKPROPAGATION

## SKRIPSI


Disusun oleh :

**Ihsan**

**11.18.908**



**JURUSAN TEKNIK INFORMATIKA S-1**

**FAKULTAS TEKNOLOGI INDUSTRI**

**INSTITUT TEKNOLOGI NASIONAL MALANG**


**2013**

# PENGHITUNG JUMLAH SEL DARAH MERAH MENGGUNAKAN JARINGAN SARAF TIRUAN DENGAN METODE BACKPROPAGATION

**Ihsan (11.18.908)**

**Program Studi Teknik Informatika S-1,**

**Fakultas Teknologi Industri, Institut Teknologi Nasional Malang**

**Email : ihsan.amd@gmail.com**

**Dosen Pembimbing : I. Dr. Eng. Aryuanto Soetedjo, ST, MT**

**II. Nurlaily Vendyansyah, ST**

**Abstrak**

*Pada saat ini, dunia ilmu pengetahuan memerlukan inovasi-inovasi seperti penghitung jumlah sel darah merah. Analisis yang dilakukan oleh dokter berdasarkan preparat darah tidak selalu sama antara dokter yang satu dengan dokter yang lain. Ketelitian dan konsentrasi dokter sangat menentukan hasil analisis. Di lain pihak analisis preparat darah tepi tidak menghasilkan bukti citra sehingga tidak dapat dianalisis oleh banyak dokter. Oleh karena itu perlu dibuat suatu alat yang dapat menghitung jumlah sel pada suatu citra secara cepat dan terautomatisasi, sehingga diperoleh analisis dan bukti yang akurat.*

*Analisis citra merupakan salah satu metode dalam pengolahan citra digital. Proses pengenalan citra / training pada aplikasi ini menggunakan jaringan saraf tiruan dengan metode backpropagation. Pada Aplikasi ini user yang digunakan dalam sistem disini lebih ditujukan kepada para dokter dan analis kesehatan.*

*Proses prapengolahan citra digital dimulai dari pengambilan citra sel darah merah yang telah ada, deteksi tepi citra dengan operator laplacian, Training citra sel darah merah, hingga citra siap untuk dianalisis. Analisis citra yang dilakukan dalam hal ini adalah pencacahan jumlah sel darah merah. Program yang dibuat memiliki kemampuan untuk mengenali citra sel darah merah dan menghitung jumlahnya.*

**Kata-kunci:** *pengolahan citra, jaringan saraf tiruan, metode backpropagation, deskripsi citra*

# PENGHITUNG JUMLAH SEL DARAH MERAH MENGGUNAKAN JARINGAN SARAF TIRUAN DENGAN METODE BACKPROPAGATION

**Ihsan (11.18.908)**

**Program Studi Teknik Informatika S-1,**

**Fakultas Teknologi Industri, Institut Teknologi Nasional Malang**

**Email : ihsan.amd@gmail.com**

**Dosen Pembimbing : I.** **Dr. Eng. Aryuanto Soetedjo, ST, MT**

**II.** **Nurlaily Vendyansyah, ST**

## Abstract

*At this time, the world of science require innovations such as a counter number of red blood cells. Analysis is performed by physicians based blood preparations are not always the same between one physician to another physician. Precision and concentration of physicians is critical analysis. On the other hand, analysis of peripheral blood preparations did not produce evidence that the image can not be analyzed by many doctors. It is therefore important to make a tool that can calculate the number of cells in an image quickly and terautomatisasi, in order to obtain an accurate analysis and evidence.*

*Image analysis is one of the methods in digital image processing. The process of image recognition / training on this application using artificial neural networks with back propagation method. In this application the user who used the system here is more directed at doctors and health analysts.*

*Pretreatment process digital images from the first image of the red blood cells that have been there, the image edge detection laplacian operator. Training image of red blood cells, until the image is ready to be analyzed. Image analysis performed in this case is the enumeration of the number of red blood cells. Program created has the ability to recognize the image of red blood cells and count them.*

**Keywords** : *image processing, neural networks, backpropagation method, description of the image*

## SURAT PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini :

Nama : IHSAN

NIM : 11.18.908

Program Studi : Teknik Informatika S-1

Dengan ini menyatakan bahwa Skripsi yang saya buat adalah hasil karya sendiri, tidak merupakan plagiasi dari karya orang lain. Dalam Skripsi ini tidak memuat karya orang lain, kecuali dicantumkan sumbernya sesuai dengan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat, dan apabila di kemudian hari ada pelanggaran atas surat pernyataan ini, saya bersedia menerima sangsinya.

Malang, Februari 2013

Yang membuat Pernyataan,


**Ihsan**

NIM. 11.18.908

# KATA PENGANTAR

بسم الله الرحمن الرحيم

Dengan rasa syukur kehadirat Allah swt karena penulis telah dapat menyelesaikan laporan Skripsi yang berjudul "Penghitung jumlah sel darah merah menggunakan jaringan saraf tiruan dengan metode *backpropagation*" dan menjadi salah satu syarat mutlak untuk menyelesaikan program studi Teknik Informatika jenjang Strata-1 Institut Teknologi Nasional (ITN) Malang.

Dengan segala kerendahan hati, penulis merasa bahwa dalam menyusun laporan ini masih menemui beberapa kesulitan dan hambatan, disamping itu juga menyadari bahwa penulisan laporan ini masih jauh dari sempurna dan masih banyak kekurangan-kekurangan lainnya, maka dari itu penulis mengharapkan saran dan kritik yang membangun dari semua pihak.

Terselesainya laporan Skripsi ini tidak lepas dari bantuan semua pihak, dan pada kesempatan ini penulis mengucapkan banyak terima kasih pada semua pihak yang membantu antara lain:

1. Bapak Joseph Dedy Irwawan, ST , MT selaku Ketua Program Studi Teknik Informatika S-1 Institut Teknologi Nasional Malang
2. Bapak Dr. Eng. Aryuanto Soetedjo, ST, MT selaku dosen pembimbing I dan Ibu Nurlaily Vendyansyah, S.T selaku dosen pempimbing II yang telah bersedia untuk meluangkan waktu untuk membimbing, memeriksa, serta memberikan petunjuk-petunjuk serta saran dalam penyusunan skripsi ini.
3. Ibu, Bapak serta adikku Maryani yang senantiasa memberikan do'a dan motivasi dalam penyelesaian skripsi ini.
4. Nurul Mahmudah Umar, S.Psi yang selalu memberikan do'a, semangat dan dukungan hingga terselesaikannya skripsi ini. Terima kasih sayang.
5. Bapak serta Ibu dosen Teknik Informatika S-1 Institut Teknologi Nasional Malang yang telah memberikan ilmu yang bermanfaat.
6. Ahmad Nurkhalis yang telah memberikan koreksi serta informasi tentang Jaringan Saraf Tiruan

7. Teman-teman satu perjuangan dan sahabat-sahabat penulis yang senantiasa memeberikan dorongan spiritual dalam penyelesaian skripsi ini.
8. Sahabat serta teman-teman di Institut Teknologi Nasional (ITN) Malang.
9. Semua pihak yang mungkin belum penulis sebutkan dan sahabat-sahabat yang telah membantu penulis hingga terselesaikanya skripsi ini, semoga Allah SWT. memberikan balasan yang setimpal atas jasa dan bantuan yang telah diberikan.

Penulis menyadari sepenuhnya bahwa sebagai manusia biasa tentunya tidak akan luput dari kekurangan dan keterbatasan. Maka mengharapkan saran dan kritik yang dapat menyempurnakan penulisan ini sehingga dapat bermanfaat dan berguna untuk pengembangan ilmu pengetahuan.

Malang, Februari 2013

Penulis

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang

Manusia memiliki sel darah merah yang berfungsi membawa oksigen ke jaringan-jaringan tubuh lewat darah, membawa karbon dioksida dan zat-zat sisa metabolisme menuju alat-alat ekskresi dan masih banyak lagi fungsi sel darah merah untuk manusia. Mengingat pentingnya peranan sel darah merah pada tubuh kita maka penulis tergerak untuk melakukan penelitian tentang sel darah merah.

Jumlah sel darah merah dapat diukur dengan cara konvensional. Penghitungan secara konvensional dilakukan dengan aplikasi yang menggunakan jaringan saraf tiruan. Tetapi bila sel darah yang diukur cukup banyak akan memakan banyak waktu. Hal ini menyebabkan pengukuran secara konvensional tidak efisien. Selain itu, penghitungan secara konvensional terkadang kurang akurat ketika dilakukan dengan pengamatan langsung. Hal ini disebabkan pengamatan pada sel darah sangat dipengaruhi oleh tingkat ketelitian dokter yang menganalisis. Dengan semakin majunya ilmu pengetahuan maka pengukuran akan lebih akurat dengan melakukan pengukuran dan penghitungan melalui bantuan pemprosesan citra digital.

Program simulasi komputer dapat melakukan simulasi pengolahan citra dengan cepat dan akurat. Pada citra hasil pemotretan sekelompok benda yang seragam atau hampir seragam, terdapat ciri khas pada setiap benda tersebut. Ciri khas itulah yang digunakan sebagai patokan untuk menghitung jumlahnya. Pengamatan sel-sel yang saling bertumpuk lebih sulit untuk dianalisis. Namun demikian, sepanjang ciri khas yang ditetapkan masih tampak maka masalah tersebut masih dapat dipecahkan.

Sudah ada alat di buat untuk penghitung jumlah sel darah merah yaitu Haematology tetapi harganya sangat mahal dan daya listrik yg digunakan besar. Sedangkan Pengitung jumlah sel darah merah yang akan penulis buat lebih efisien tidak menggunakan banyak biaya dan daya listrik yang digunakan pun relatif kecil.

Oleh karena itu penulis bermaksud melakukan penelitian dengan merancang dan membangun software penghitung jumlah sel darah merah dengan Metode Backpropagation.

## 1.2 Rumusan Masalah

1. Bagaimana sistem penghitung jumlah sel darah merah pada program ini
2. Bagaimana menerapakan Jaringan Saraf Tiruan dengan metode Backpropagation menggunakan software borland delphi 7.0

## 1.3 Tujuan

Pada penelitian ini akan dilakukan studi dan implementasi yang bertujuan untuk mengaplikasikan algoritma jaringan saraf tiruan dengan metode backpropagation dan merancang suatu sistem yang dapat melakukan tugas dalam menghitung jumlah sel darah merah.

## 1.4 Batasan Masalah

Skripsi ini hanya akan membahas tentang implementasi dari Software yang dirancang. Adapun pokok pembahasannya meliputi:

1. Tidak membahas proses pengambilan gambar sel darah merah.
2. Penghitungan ini menggunakan jaringan saraf tiruan (JST) dengan Metode Backpropagation.
3. Citra yang dipilih adalah citra 24 bit sehingga dikenali sebagai citra RGB.
4. Software yang digunakan untuk membangun aplikasi borland delphi 7.0
5. Tidak mengukur perbesaran citra sel darah merah yang sesungguhnya.
6. Sel darah merah yang di hitung adalah sel darah manusia.
7. User dari aplikasi ini adalah dokter dan analis kesehatan.

## 1.5 Metodologi Penelitian

Sehubungan dengan judul skripsi yaitu "Penghitung Jumlah Sel Darah Merah Menggunakan Jaringan Saraf Tiruan Dengan Metode Backpropagation" maka pembahasan skripsi ini digunakan data penelitian yang bersifat sekunder. Data sekunder adalah metode ilmiah yang berdasarkan studi pustaka atau literature dari

bahan – bahan perkuliahan, buku laporan hasil praktikum, serta konsultasi langsung dengan pembimbing maupun orang – orang yang berkecimpung didalamnya. Sehingga didapatkan hasil yang penulis rasa sudah maksimal.

## 1.6 Sistematika Penulisan

Langkah-langkah atau tahapan-tahapan yang akan dilakukan dalam menyelesaikan skripsi ini adalah sebagai berikut:

**PENDAHULUAN**

Bab ini menerangkan tentang Latar Belakang, Rumusan Masalah, Tujuan, Batasan Masalah, Metode Penelitian dan Sistematika Penulisan.

**LANDASAN TEORI**

Merupakan Dasar Teori yang mencakup : Definisi Sel Darah Merah, Pengertian Citra Digital, Definisi Pengolahan Citra, Jaringan Saraf Tiruan (JST), Metode Backpropagation, Sekilas delphi 7.0.

**ANALISA DAN PERANCANGAN SISTEM**

Berisi tentang komponen timage, program penampil citra, penyimpan citra.

**PENGUJIAN DAN HASIL**

Berisi Hasil sebuah program yang dibuat pada skripsi ini.

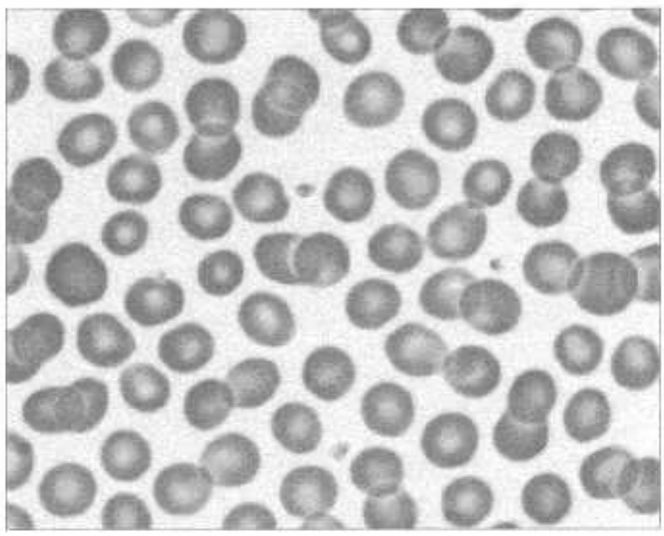
**PENUTUP**

Berisi kesimpulan dan saran-saran.

# BAB II
# LANDASAN TEORI

## 2.1 Sel Darah Merah (SDM)[3]

Di antara tiga tipe darah (sel darah merah, seldarah putih, dan trombosit), sel darah merahlah yang paling banyak jumlahnya.



**Gambar 2.1** *Sel Darah Merah*

Sel-sel darah merah mempunyai bentuk cakra (Gambar 2.1) dengan diameter 7,5 μm dengan ketebalan tepi 2 μm. Tengah-tengah cakra tersebut lebih tipis dengan ketebalan 1 μm. bentuk bikonkaf yang menarik ini mempercepat pertukaran gas-gas antara sel-sel dan plasma darah.

Proses pergantian sel darah merah dari atau olch sel darah baru terjadi setelah sirkulasi 3 sampai 4 bulan. Sel darah merah mengalami desintergrasi atau pemecahan sehingga melepas *haemoglobin* ke dalam sel dan sel darah pecah. Pembentukan sel darah merah pada orang dewasa pada sumsum tulang belakang dan pada bayi terjadi di hati, *kelenjar thymus* dan *nodula lymphatica*.

Pengaruh *haemoglobin* didalam sel darah merah menyebabkan timbulnya warna merah pada darah karena mempunyai kemampuan untuk mengangkut oksigen. Haemoglobin adalah senyawa organik yang komplek dan terdiri dari empat pigmen *forpirin* merah (*heme*) yang masing-masing mengandung *iron* dan *globin* yang

merupakan protein *globural* dan terdiri dari empat asam amino. *Haemoglobin* bergabung dengan oksigen didalam paru-paru yang kemudian terbentuk *oksihaemoglobin* yang selanjutnya melepaskan oksigen ke sel-sel jaringan didalam tubuh

Susunan dari sel darah merah adalah air (62%-72%) dan kira-kira sisanya berupa *solid* terkandung *haemoglobin* 95% dan sisanya berupa protein pada *stroma* dan membran sel, lipid, enzim, vitamin dan glukosa serta urin. Umur sel darah merah pada manusia berkisar antara 90 hingga 140 hari, rata-rata 120 hari dan pada hewan umurnya kira-kira 25 hingga 140 hari.

## 2.2 Pengertian Citra Digital[8]

Menurut Kamus Webster defenisi citra adalah "suatu represntasi, kemiripan, atau imitasi dari suatu obyek atau benda". Misalnya foto mewakili entitas diri di depan kamera. Foto sinar-X *thorax* mewakili keadaan bagian dalam tubuh seseorang, dan data dalam suatu file GIF mewakili apa yang digambarkan.

Citra dapat dikelompokkan menjadi citra tampak dan citra tak tampak. Banyak contoh citra tampak dalam kehidupan sehari-hari : foto keluarga, gambar anak, lukisan Pablo Picasso, apa yang nampak di layar monitor dan televisi, serta hologram (citra optis). Sedangkan citra tak nampak misalnya : data gambar dalam file (citra digital), dan citra yang direpresentasikan menjadi fungsi matematis. Di samping itu ada juga citra fisik tak nampak, misalnya citra distribusi panas di kulit manusia serta peta densitas dalam suatu material. Untuk dapat dilihat mata manusia, citra tak nampak ini harus diubah menjadi citra tampak, misalnya dengan menampilkannya di monitor, dicetak di atas kertas, dan sebagainya.

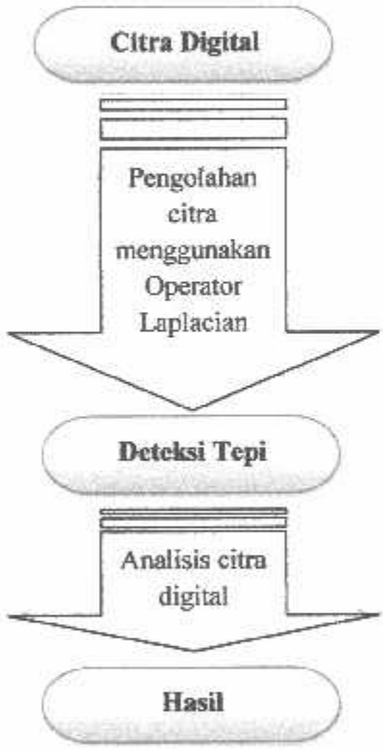
Diantara jenis-jenis citra tersebut, hanya citra *digital* yang dapat diolah menggunakan komputer. Jenis citra lain, jika hendak diolah dengan komputer, harus diubah menjadi citra digital, misalnya foto dipindah (*scan*) dengan *scanner*, persebaran panas tubuh ditangkap dengan kamera infra merah dan diubah menjadi informasi numeris, informasi densitas dan komposisi bagian dalam tubuh manusia ditangkap dengan bantuan pesawat sinar –X dan sistem deteksi radiasi menjadi

*informasi digital. Kegiatan untuk mengubah informasi citra fisik non digital menjadi* digital disebut sebagai pencitraan (*imaging*).

## 2.3 Definisi Pengolahan Citra[1]

Pengolahan Citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsi visual. Proses ini mempunyai ciri data masukan dan informasi keluaran yang berbentuk citra. Istilah pengolahan citra digital secara umum didefinisikan sebagai pemrosesan citra dua dimensi dengan komputer. Dalam definisi yang lebih luas, pengolahan citra digital juga mencakup semua data dua dimensi. Citra digital adalah barisan bilangan nyata maupun kompleks yang diwakili oleh bit-bit tertentu.

Adapun teknik-teknik pengolahan citra sebagai teknik praolah citra (*image preprocessing*)




**Gambar 2.2** *Urutan pengolahan citra digital*

Secara dimensional, citra dapat berupa citra 2 dimensi maupun 3 dimensi. Pengolahan citra 3 dimensi adalah sangat komplek dan dianggap keluar dari pembahasan ini, sehingga hanya akan dibahas pengolahan untuk citra 2 dimensi. Sedangkan citra 3 dimensi dapat didekati dengan cara dengan cara membagi citra tersebut menjadi lapisan-lapisan (*Layer*) yang masing-masing merupakan citra 2 dimensi, dengan demikian pengolahan dapat dilakukan secara 2 dimensi lapis demi lapis.

### 2.3.1 Operasi Pengolahan Citra[1]

Pengolahan citra pada dasarnya dilakukan dengan cara memodifikasi setiap titik dalam citra tersebut sesuai keperluan. Secara garis besar, modifikasi tersebut dikelompokkan menjadi :

1. Operasi titik, dimana setiap titik diolah secara tidak gayut terhadap titik-titik yang lain.
2. Operasi global, dimana karakteristik global ( biasanya berupa sifat statistik) dari citra digunakan untuk memodifikasikan nilai setiap titik.
3. Operasi temporal / berbasis bingkai, dimana sebuah citra diolah dimodifikasi secara geometris.
4. Operasi geometri, dimana bentuk, ukuran, atau orientasi citra dimodifikasi secara geometris.
5. Operasi banyak titik bertetangga, dimana data dari titik-titik yang bersebelahan (bertetangga) dengan titik yang ditinjau ikut berperan dalam mengubah nilai.
6. Operasi Morfologi yaitu operasi yang berdasarkan segmen atau bagian dalam citra yang menjadi perhatian.

Operasi-operasi tersebut akan dibahas pada pembahasan selanjutnya. Batasan yang dipakai dalam pembahasan ini adalah : Pengolahan citra yang difokuskan pada format citra skala keabuan 8 bit dengan warna hitam pekat untuk nilai minimum (0) dan warna putih cemerlang untuk nilai maksimal (255), serta citra warna *true color*. Operasi morfologi yang merupakan pengolahan citra yang sangat berhubungan erat dengan pengenalan pola.

## 2.4 Deteksi Tepi (*Operator Laplacian*) [1]

Operator yang dapat digunakan untuk mencari tepi adalah operator *Laplacian*. Berbeda dengan operator gradien yang menggunakan turunan pertama, operator *Laplacian* menggunakan turunan kedua (sering disebut *zero crossing operator*) yang didefinisikan dengan

Operator Laplace mendeteksi lokasi tepi lebih akurat khususnya pada tepi yang curam. maka diperoleh nilai mask :

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

*(a) Laplacian 5 titik*

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |

*(b) Laplacian 9 titik 1*

| | | |
|---|---|---|
| -2 | 1 | -2 |
| 1 | 4 | 1 |
| -2 | 1 | -2 |

*(c) Laplacian 9 titik 2*

**Gambar 2.3** *Operator Laplacian untuk deteksi tepi*

## 2.5 Jaringan Saraf Tiruan (JST)[6]

Definisi dari jaringan saraf tiruan yaitu "Suatu neural network (NN), adalah suatu struktur pemroses informasi yang terdistribusi dan bekerja secara paralel, yang terdiri atas elemen pemroses (yang memiliki memori lokal dan beroperasi dengan informasi lokal) yang diinterkoneksi bersama dengan alur sinyal searah yang disebut koneksi. Setiap elemen pemroses memiliki koneksi keluaran tunggal yang bercabang (fan out) ke sejumlah koneksi kolateral yang diinginkan (setiap koneksi membawa sinyal yang sama dari keluaran elemen pemroses tersebut). Keluaran dari elemen pemroses tersebut dapat merupakan sebarang jenis persamaan matematis yang diinginkan. Seluruh proses yang berlangsung pada setiap elemen pemroses harus benar-benar dilakukan secara lokal, yaitu keluaran hanya bergantung pada nilai masukan pada saat itu yang diperoleh melalui koneksi dan nilai yang tersimpan dalam memori lokal".

Jaringan Saraf Tiruan (JST) dibentuk sebagai generalisasi model matematika dari jaringan saraf biologi, dengan asumsi bahwa :

a. Pemrosesan informasi terjadi pada banyak elemen sederhana (neuron).
b. Sinyal dikirimkan diantara neuron-neuron melalui penghubung-penghubung.

c. Penghubung antar neuron memiliki bobot yang akan memperkuat atau memperlemah sinyal.

d. Untuk menentukan output, setiap neuron menggunakan fungsi aktivasi (biasanya bukan fungsi linier) yang dikenakan pada jumlahan input yang diterima. Besarnya output ini selanjutnya dibandingkan dengan suatu batas ambang.

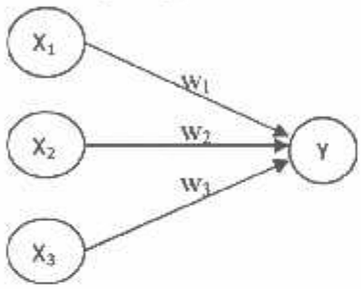
JST ditentukan oleh 3 hal :

a. Pola hubungan antar neuron (disebut arsitektur jaringan)

b. Metode untuk menentukan bobot penghubung (disebut metode *training/learning* / algoritma).

c. Fungsi aktivasi

Sebagai contoh, perhatikan neuron Y pada gambar 2.4




**Gambar 2.4** *Neuron*

Y menerima input dari neuron $X_1$, $X_2$, dan $X_3$ dengan bobot hubungan masing-masing adalah $W_1$, $W_2$, dan $W_3$. Ketiga impuls neuron yang ada dijumlahkan

$$Net = X_1W_1 + X_2W_2 + X_3W_3 \quad \text{.....(2-1)}$$

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

### 2.5.1 Kelebihan Jaringan Saraf Tiruan[7]

Kelebihan dari Sistem Jaringan Saraf Tiruan :

a. Kemampuan melakukan proses pembelajaran

b. Kemampuan beradaptasi

c. Implementasi komponen peralatan dalam bentuk paralel secara besar-besaran.

### 2.5.2 Dasar Jaringan Saraf Tiruan[7]

#### 2.5.2.1 Pengertian Neuron

Neuron dianalogikan dengan neurosikologi *(neurophysiology)* pada otak manusia. Dalam Jaringan Saraf Tiruan neuron diartikan sebagai bagian terkecil dari jaringan dari jaringan saraf tiruan yang berfungsi sebagai elemen pemproses. Dengan demikian neuron juga dapat dinyatakan sebagai prosesor sederhana dari sistem JARINGAN SARAF TIRUAN. Neuron juga dikenal dengan sebutan *percepton* atau *adaline*.

#### 2.5.2.2 Bagian-bagian Neuron

Dalam sistem Jaringan Saraf Tiruan neuron akan bekerja dengan mengumpulkan sinyal dari neuron yang terhubung sebelumnya dan memprosesnya untuk menjadi masukan bagi neuron berikutnya. Neuron tersususun dari komponen-komponen sebagai berikut:

a. Sekumpulan penghubung atau yang dikenal dengan *synapses* atau *connection link* yang dikarakterkan dengan sebuah pembobot *(weight/strength connection)*.
b. Sebuah penjumlah *(summing/adder)* yang berfungsi untuk menjumlahkan semua sinyal masukannya.
c. Sebuah fungsi tidak dinamis *(non dynamical)* yang dikenali dengan sebutan fungsi aktivasi *(activition function)*

   Persamaan dari fungsi penjumlah yang dikenal juga sebagai fungsi transformasi neuron *(neuron transfer function)* adalah:

## 2.6 Metode Backpropagation [9]

Algoritma pelatihan Backpropagasi *(Backpropagation)* atau ada yang menterjemahkannya menjadi propagasi balik pertama kali dirumuskan oleh Werbos dan dipopulerkan oleh Rumelhart dan McClelland untuk dipakai pada JST, dan selanjutnya algoritma ini biasa disingkat dengan BP. Algoritma ini termasuk metode

pelatihan superviseddan didesain untuk operasi pada jaringan feed fordward multi lapis.

Metode BP ini banyak diaplikasikan secara luas. Sekitar 90%, bahkan BP telah berhasil diaplikasikan di berbagai bidang, diantaranya diterapkan di bidang finansial, pengenalan pola tulisan tangan, pengenalan pola suara, sistem kendali, pengolahan citra medika dan masih banyak lagi keberhasilan BP sebagai salah satu metoda komputasi yang handal.

Algoritma ini juga banyak dipakai pada aplikasi pengaturan karena proses pelatihannya didasarkan pada hubungan yang sederhana, yaitu : jika keluaran memberikan hasil yang salah, maka penimbang (Weight) dikoreksi supaya galatnya dapat diperkecil dan direspon jaringan selanjutnya diharapkan akan lebih mendekati harga yang benar. BP juga berkemampuan untuk memperbaiki penimbang pada lapisan tersembunyi (hidden layer).

Secara garis besar mengapa algoritma ini disebut sebagai propagasi balik, dapat dideskripsikan sebagai berikut: ketika jaringan diberikan pola masukan sebagai pola pelatihan maka pola tersebut menuju ke unit-unit pada lapisan tersembunyi untuk diteruskan ke unit-unit lapisan keluaran. Kemudian unit-unit lapisan keluaran memberikan tanggapan yang disebut sebagai keluaran jaringan. Saat keluaran jaringan tidak sama dengan keluaran yang diharapkan maka keluaran akan menyebar mundur (backword) pada lapisan tersembunyi diteruskan ke unit pada lapisan masukan. Oleh karenanya maka mekanisme pelatihan tersebut dinamakan backpropagation/propagasi balik.

Tahap pelatihan ini merupakan langkah bagaimana suatu jaringan saraf itu berlatih, yaitu dengan cara melakukan perubahan penimbang (sambungan antar lapisan yang membentuk jaringan melalui masing masing unitnya). Sedangkan pemecahan masalah baru akan dilakukan jika proses pelatihan tersebut selesai , fase tersebut adalah fase mapping atau proses pengujian/testing.

### 2.6.1 Pelatihan Standar Backpropagation[9][11]

Pelatihan Backpropagation meliputi 3 fase. Fase pertama adalah fase maju. Pola masukan dihitung maju mulai dari layar masukan hingga layer keluaran

menggunakan fungsi aktivasi yang ditentukan. Fase kedua adalah fase mundur. Selisih antara keluaran jaringan dengan target yang diinginkan merupakan kesalahan yang terjadi. Kesalahan tersebut dipropagasikan mundur, dimulai dari garis yang berhubungan kangsung dengan unit-unit dilayar keluaran. Fase ketiga adalah modifikasi bobot untuk menurunkan kesalahan yang terjadi.

Fase I : Propagasi maju

Selama propagasi maju, sinyal masukan (=$x_i$) dipropagasikan ke layar tersembunyi menggunakan fungsi aktivasi yang ditentukan. Keluaran dari setiap unit layar tersembunyi (=$z_j$) tersebut selanjutnya dipropagasikan maju lagi ke layar tersembunyi diatasnya menggunakan fungsi aktivasi yang ditentukan. Demukian seterusnya hingga menghasilkan keluaran jaringan (=$y_k$).

Berikutnya, keluaran jaringan (=$y_k$) dibandingkan dengan target yang harus di capai (=$t_k$). Selisih $t_k - y_k$ adalah kesalahan yang terjadi. Jika kesalahan ini lebih kecil dari batas toleransi yang ditentukan, maka iterasi dihentikan. Akan tetapi apabila kesalahan masih lebih besar dari batas toleransinya, maka bobot setiap garis dalam jaringan akan dimodifikasi untuk mengurangi kesalahan yang terjadi.

Fase II : Propagasi Mundur

Berdasarkan kesalahan $t_k - y_k$, dihitung faktor $\delta_k$ (k = 1,2,...m) yang dipakai untuk mendistribusikan kesalahan di unit yk kesemua unit tersembunyi yang terhubung langsung dengan $y_k$. $\delta_k$ juga dipakai untuk mengubah bobot garis yang berhubungan langsung dengan unit keluaran.

Dengan cara yang sama, dihitung faktor δj disetiap unit di layar tersembunyi sebagai dasar perubahan bobot semua garis yang berasal dari unit tersembunyi di layar di bawahnya. Demikian seterusnya hingga semua faktor δ di unit tersembunyi yang berhubungan langsung dengan unit masukan dihitung.

Fase III : perubahan bobot

Setelah semua faktor δ dihitung, bobot semua garis dimodifikasi bersamaan. Perubahan bobot suatu garis didasarkan atas faktor δ neuron di layar atasnya. Sebagai contoh, perubahan bobot garis yang menuju ke layar keluaran didasarkan atas δk yang ada di unit keluaran.

Ketiga fase tersebut diulang-ulang terus hingga kondisi penghentian dipenuhi.

Umumnya kondisi penghentian yang sering dipakai adalah jumlah iterasi atau kesalahan. Iterasi akan dihentikan jika jumlah iterasi yang dilakukan sudah melebihi jumlah maksimum iterasi yang ditetapkan, atau jika kesalahan yang terjadi sudah lebih kecil dari batas toleransi yang diijinkan.

Algoritma pelatihan untuk jaringan dengan satu layar tersembunyi (dengan fungsi aktivasi sigmoid biner) adalah sebagai berikut:

**Langkah 0** : Inisialisasi semua bobot dengan bilangan acak kecil

**Langkah 1** : Jika kondisi penghentian belum terpenuhi, lakukan langkah 2 – 9

**Langkah 2** : Untuk setiap pasang data pelatihan, lakukan langkah 3 – 8

Fase I : Propagasi maju

**Langkah 3** : Tiap unit masukan menerima sinyal dan meneruskan ke unit tersembunyi di atasnya

**Langkah 4** : Hitung semua keluaran di unit tersembunyi zj (j = 1,2, ..., p)

$$z\_net_j = v_{jo} + \sum_{i=1}^{n} x_i v_{ji} \quad \text{.....(2-2)}$$

$$z_j = f(z\_net_j) = \frac{1}{1 + e^{-z\_net\ j}} \quad \text{.....(2-3)}$$

**Langkah 5** : Hitung semua keluaran jaringan di unit yk (k = 1,2, ...,m)

$$y\_net = w\ ko + \sum_{j=1}^{p} z_j w_{kj} \quad \text{.....(2-4)}$$

$$yk = f(z\_net_k) = \frac{1}{1 + e^{-y\_net\ k}} \quad \text{.....(2-5)}$$

Fase II : Propagasi mundur

**Langkah 6** : Hitung faktor δ unit keluaran betrdasarkan kesalahn di setiap unit keluaran yk (k = 1,2, ...,m)

$$\delta_k = (t_k - y_k) f'(y\_net_k) = (t_k - y_k)\ yk\ (1 - y_k) \quad \text{.....(2-6)}$$

δk merupakan unit kesalahan yang akan dipakai dalam perubahan bobot layar di

bawahnya (langkah 7)

Hitung suku perubahan bobot wkj (yang akan dipakai nanti untuk merubah bobot wkj) dengan laju percepatan α

$$\Delta w_{kj} = \alpha\, \delta_k\, z_j \quad ; \qquad k = 1,2, \ldots,m \; ; \; j = 0,1,\ldots,p \qquad \ldots\ldots\ldots\ldots (2\text{-}7)$$

**Langkah 7** : Hitung faktor δ unit tersembunyi berdasarkan kesalahan setiap unit tersembunyi zj (j = 1,2, ...,p)

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k\, w_{kj} \qquad \ldots\ldots\ldots\ldots (2\text{-}8)$$

Faktor δ unit tersembunyi:

$$\delta_j = \delta\_net_j\, f'(z\_net_j) = \delta\_net_j\, z_j (1\text{-}z_j) \qquad \ldots\ldots\ldots\ldots (2\text{-}9)$$

Hitung suku perubahan bobot $v_{ji}$ (yang akan dipakai nanti untuk merubah bobot $v_{ji}$)

$$\Delta vji = \alpha\, \delta j\, xi \quad ; \quad j = 1,2,\ldots,p \quad ; \quad i = 0,1,\ldots,n \qquad \ldots\ldots\ldots\ldots (2\text{-}10)$$

Fase III : Perubahan bobot

**Langkah 8** : Hitung semua perubahan bobot

Perubahan bobot garis yang menuju ke unit keluaran :

$$W_{kj}\,(baru) = w_{kj}\,(lama) + \Delta w_{kj} \quad (k=1,2,\ldots,m \; ; \; j=0,1,\ldots,p) \ldots\ldots (2\text{-}11)$$

Perubahan bobot garis yang menuju ke unit tersembunyi :

$$W_{ji}\,(baru) = v_{ji}\,(lama) + \Delta v_{ji} \quad (j=1,2,\ldots,p \; ; \; i=0,1,\ldots,n) \ldots\ldots\ldots (2\text{-}12)$$

Setelah pelatihan selesai dilakukan, jaringan dapat dipakai untuk pengenalan pola. Dalam hal ini, hanya propagasi maju (langkah 4 dan 5) saja yang dipakai unutk

menentukan keluaran jaringan

Apabila fungsi aktivasi yang dipakai bukan sigmoid biner, maka langkah 4 dan 5 harus disesuaikan. Demikian juga turunannya pada langkah 6 dan 7

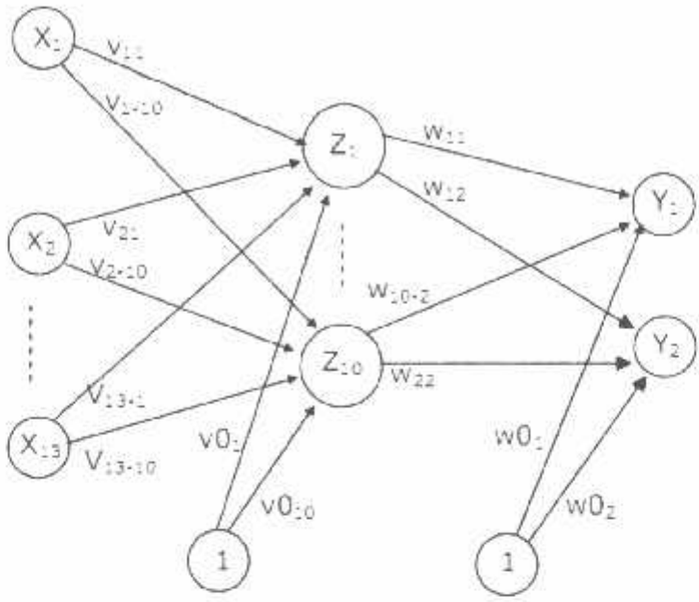
Perambatan galat mundur (Backpropagation) adalah sebuah metode sistematik untuk pelatihan multiplayer jaringan saraf tiruan. Metode ini memiliki dasar matematis yang kuat, obyektif dan algoritma ini mendapatkan bentuk persamaan dan nilai koefisien dalam formula dengan meminimalkan jumlah kuadrat galat error melalui model yang dikembangkan (training set).

1. Dimulai dengan lapisan masukan, hitung keluaran dari setiap elemen pemroses melalui lapisan luar.
2. Hitung kesalahan pada lapisan luar yang merupakan selisih antara data aktual dan target.
3. Transformasikan kesalahan tersebut pada kesalahan yang sesuai di sisi masukan elemen pemroses.
4. Propagasi balik kesalahan-kesalahan ini pada keluaran setiap elemen pemroses kekesalahan yang terdapat pada masukan. Ulangi proses ini sampai masukan tercapai.
5. Ubah seluruh bobot dengan menggunakan kesalahan pada sisi masukan elemen dan luaran elemen pemroses yang terhubung.

### 2.6.2 Arsitektur Jaringan Metode Backpropagation[6][9]

Backpropagation memiliki beberapa unit yang ada dalam satu atau lebih layar tersembunyi. Gambar 2.4 adalah arsitektur backpropagation dengan n buah masukkan (ditambah sebuah bias), serta m buah unit keluaran.

Jaringan saraf terdiri dari 3 lapisan, yaitu lapisan masukan/input terdiri atas variabel masukan unit sel darah merah, lapisan tersembunyi terdiri atas beberapa unit sel darah, dan lapisan keluaran/output terdiri atas 2 kemungkinan yaitu sel darah merah atau bukan sel darah merah. Lapisan masukan digunakan untuk menampung beberapa pixel yaitu X1 sampai dengan seterusnya, sedangkan 2 lapisan keluaran digunakan untuk mempresentasikan pengelompokan Sel darah merah atau bukan, nilai 1 adalah sel darah merah, nilai 0 adalah bukan sel darah merah.




**Gambar 2.5** *Arsitektur Jaringan Backpropagation*

Keterangan :

X = Masukan (input).

J = 1 s/d n (n = 10).

V = Bobot pada lapisan tersembunyi.

W = Bobot pada lapisan keluaran.

n = Jumlah unit pengolah pada lapisan tersembunyi.

b = Bias pada lapisan tersembunyi dan lapisan keluaran.

k = Jumlah unit pengolah pada lapisan keluaran.

Y = Keluaran hasil.

Tujuan dari perubahan bobot untuk setiap lapisan, bukan merupakan hal yang sangat penting. Perhitungan kesalahan merupakan pengukuran bagaimana jaringan dapat belajar dengan baik. Kesalahan pada keluaran dari jaringan merupakan selisih antara keluaran aktual (current output) dan keluaran target (desired output). Langkah berikutnya adalah menghitung nilai SSE (Sum Square Error) yang merupakan hasil penjumlahan nilai kuadrat error neuron1 dan neuron - neuron pada lapisan output tiap data, dimana hasil penjumlahan keseluruhan nilai SSE akan digunakan untuk menghitung nilai RMSE (Root Mean Square Error) tiap iterasi

## 2.7 Sekilas Delphi 7.0[4]

### 2.7.1 Kebutuhan Sistem

Untuk dapat menjalankan program aplikasi Borland Delphi 7.0 membutuhkan sistem komputer minimal :

a. Intel Pentium II kelas processor 450 MHz atau yang lebih tinggi
b. Sistem operasi Windows 98, Windows Server 2003, XP Profesional, 2000 Profesional atau 2000 Server.
c. RAM 128 MB atau yang lebih tinggi.
d. CD-ROM atau DVD-ROM Drive
e. Resolusi monitor 1024 x 768 atau yang lebih tinggi.

### 2.7.2 Elemen-Elemen Delphi 7.0

1. Menu File, Edit, Search, dst. Jika anda biasa memakai aplikasi windows maka anda akan terbiasa menggunakan menu ini. Tepat dibawahnya akan anda temui 2 elemen berikut yaitu :

2. Disebelah kiri, yang biasa disebut speed buttons. Ada 2 baris. Jika anda arahkan mouse di salah satu tombolnya akan anda lihat keterangan mengenai tombol tersebut. Sebagai

contoh New, Open, Save, dst. Sampai help contents pada baris pertama tombol-tombol ini akan menghemat waktu anda untuk mengerjakan tugas-tugas tertentu dibandingkan jika anda menggunakannya dari menu atasnya.

3. Disebelah kiri, terdapat sekumpulan palette categories, standard additional, dst. Sampai activeX. Pada start-up, kategori Standard yang terbuka.

4. Tepat dibawahnya adalah apa yang disebut component palette, dengan icon-icon yang mewakili komponen-komponen VCL (Visual Component Library). Komponen-komponen ini tergantung pada tab/page kategori palette yang anda pilih. Misalkan anda memilih tab standard, maka anda akan melihat sebuah icon A ()kompnen label), ikon dengan ab (komponen edit) dan sebuah tombol ok (komponen button). Anda dapat meletakkan

mouse diatas komponen-komponen tersebut untuk mengetahui komponen apakah itu.

5. Disebelah kiri terdapat object inspector dengan tab properties dan events. Secara singkat, object inspector mengijinkan anda mengeset property (atribut) dari sebuah komponen, misalnya nama, warna, tinggi, lebar, caption dst. Dan kejadian-kejadian yang terjadi pada komponen tersebut misalnya apa yang terjadi ketika suatu tombol ditekan.

6. Berikutnya lagi adalah form windows. Tempat dimana anda mengatur tampilan untuk program anda. Form inilah yang akan ditampilkan ketika program dijalankan. Titik-titik tersebut ada disana untuk membantu anda menempatkan dan mengubah posisi komponen-komponen yang anda masukkan.

7. Yang mungkin tidak tampak oleh anda adalah code editor window. Code editor adalah tempat anda menuliskan program-program. Ditempat ini anda bisa menyusun program untuk memberikan apa yang anda inginkan untuk setiap kejadian (event) yang terjadi pada form anda.

### 2.7.3 Citra Dalam Delphi

Delphi tidak menyediakan secara khusus rutin-rutin untuk pengolahan citra, olch karena itu perlu dibuat sendiri program untuk mengolah citra. Namun Delphi telah menyediakan sarana untuk menampilkan citra, yaitu melalaui Komponen Timage yang terdapat pada palet komponen Additional.

Komponen ini memiliki properti Picture yang digunakan untuk menyimpan fata citra. Citra yang akan ditampilkan diambil dari file gambar yang dapat ditentukan pada saat mendesain dengan cara mengisi nilai properti ini, atau pada saat program dijalankan dengan menggunakan prosedur LoadFromFile.

Subproperti yang penting pada picture pada Picture antara lain adalah :

a. Height, berisi nilai tinggi citra
b. Width, berisi nilai lebar citra
c. Bitmap, berisi data format dan piksel citra.

### 2.7.4 Komponen Timage

Delphi tidak menyediakan secara khusus rutin-rutin untuk pengolahan citra, oleh karena itu perlu dibuat sendiri program untuk mengolah citra. Namun Delphi telah menyediakan sarana untuk menampilkan citra, yaitu melalui komponen TImage yang terdapat pada palet komponen Additional.

Komponen ini memiliki properti picture yang digunakan untuk menyimpan data citra. Citra yang akan ditampilkan diambil dari file gambar yang dapat ditentukan pada saat mendesain dengan cara mengisi nilai property ini, atau pada saat program dijalankan dengan menggunakan prosedur LoadFromFile.

### 2.7.5 Program Penampil Citra

Untuk menampilkan sebuah citra yang diambil dari file berekstensi BMP, sebenarnya selain bitmap ( berkekstensi BMP), komponen Picture mendukung format gambar berupa vector, yaitu windows metafile (WMF) dan enhanced metafile (EMF), serta file icon (ICO). Pada skripsi ini penulis hanya membahas mengenai penggunaan file bitmap sebagai citra yang diolah. File citra berekstensi lain (GIF, JPG, PNG) harus dikonversi dahulu menggunakan aplikasi pengolahan gambar lain ( misalnya Adobe Photoshop atau Corel Photopaint ) menjadi file bitmap.

Pengambilan gambar dilakukan dengan menggunakan komponen TOpenPictureDialog yang terdapat pada palet komponen Dialogs.

## 2.8 Gambar Bitmap[2]

Gambar *Bitmap* sering disebut juga dengan gambar raster. Gambar *Bitmap* adalah gambar yang terbentuk dari *pixel*, dengan setiap *pixel*nya mempunyai warna tertentu. Jika gambar *bitmap* ini diperbesar, misalnya menjadi 4 kalinya, maka gambar akan menjadi kabur karena pixelnya juga bertambah besar menjadi 4 kalinya (kualitas gambar menurun). Format gambar *bitmap* sering dipakai dalam foto dan gambar. Dua istilah yang perlu dipahami ketika bekerja dengan gambar bitmap adalah resolusi dan kedalaman warna.

Setiap kotak kecil (*pixel*) mempunyai nilai (kecerahan atau warna) dan lokasi

masing-masing. Setiap *pixel* yang ditampilkan pada layar monitor, dipetakan sebagai salah satu atau lebih *bit* dalam memori komputer. Karena itu, gambar yang ditampilkan dengan cara ini disebut sebagai *bitmap* yang artinya peta *bit*. Cara ini sering digunakan karena lebih mudah digunakan, tanpa batas, dan dapat berlaku untuk semua gambar.

Gambar *bitmap* mempunyai keunggulan, yaitu kemudahannya untuk ditampilkan secara rinci dengan pola-pola yang komplcks atau gambar fotorealistik, yang tidak dapat dengan mudah direpresentasikan sebagai model matematika (garis, kurva, dan bidang).

### 2.8.1 Format File BMP

Struktur *file BMP* terdiri dari empat bagian seperti pada diagram di bawah ini. Bagian pertama adalah *header*, diikuti dengan bagian informasi, palet warna, dan data *pixel*.

**Tabel 2.1** *Struktur File BMP*

| **Header File** |
|---|
| **Info Header** |
| **Optional Pallet** |
| **Image Data** |

Berikut ini akan dibahas secara singkat masing-masing bagian struktur yang terdapat pada sebuah *file BMP*

### 2.8.2 Header BMP

Fungsi utama dari *header* pada *file BMP* adalah sebagai tanda/ciri yang mengidentifikasi format *file* tersebut. *Header* pada *file BMP* mempunyai beberapa *field* yang jarang digunakan. Hal ini dapat dilihat pada Tabel 2.1

**Tabel 2.2** Tabel *Header BMP*

| Offset | Nama Field | Ukuran | Keterangan |
|---|---|---|---|
| 0 | BfType | 2 byte | Berisi Karakter "BM" |

| | | | |
|---|---|---|---|
| 2 | BfSize | 4 byte | Ukuran file BMP dalam byte |
| 6 | BfReserved 1 | 2 byte | Tidak digunakan |
| 8 | BfReserved 2 | 2 byte | Tidak digunakan |
| 10 | BfOffbits | 4 byte | Offset kepada awal data pixel dalam byte |

### 2.8.3 Informasi BMP

Informasi *BMP* yang mengikuti*header BMP*, mempunyai ukuran minimal 40 *byte*. Beberapa *field* penting yang terdapat pada informasi *BMP*, yaitu panjang dan lebar, jumlah *bit* per *pixel*, dan jenis kompresi, seperti terlihat pada Tabel 2.3.

**Tabel 2.3** Tabel *Informasi BMP*

| Offset | Nama Field | Ukuran | Keterangan |
|---|---|---|---|
| 14 | BiSize | 4 byte | Ukuran header dalam byte |
| 18 | BiWidth | 4 byte | Lebar gambar |
| 22 | BiHeight | 4 byte | Panjang gambar |
| 26 | BiPlanes | 2 byte | Bernilai 1 |
| 28 | BiBitCount | 2 byte | Bits per pixel 1, 4, 8, 16, 24, atau 32 |
| 30 | BiCompression | 4 byte | Jenis Kompresi . RGB=0, RLE8=1, RLE4=2, atau BITFIELDS=3 |
| 34 | BiSizeImage | 4 byte | Ukuran gambar kompresi dalam byte |
| 38 | BixPelsPerMeter | 4 byte | Resolusi horizontal |
| 42 | BiyPelsPerMeter | 4 byte | Resolusi vertical |
| 46 | BiClrUsed | 4 byte | Jumlah warna yang digunakan |
| 50 | BiClrImportant | 4 byte | Jumlah warna yang penting |

### 2.8.4 Palet Warna BMP

Gambar yang menggunakan 1, 4, atau 8 *bits* per *pixel* pasti mempunyai palet warna. Pada umumnya palet berisi 2, 16, atau 256 macam warna, tetapi dapat lebih sedikit jika gambar yang bersangkutan tidak menggunakan semua warna yang tersedia. Untuk gambar 24 *bit*, tidak digunakan palet warna, melainkan langsung ditampilkan nilai *RGB* (*Red, Green, Blue*). Masing-masing warna dalam palet

memiliki ukuran sebesar 4 *byte*, seperti terlihat pada Tabel 2.4.

**Tabel 2.4** Tabel *Palet Warna BMP*

| *Offset* | Nama *Field* | Ukuran | Keterangan |
|---|---|---|---|
| 0 | RgbBlue | 1 byte | Nilai warna biru |
| 1 | RgbGreen | 1 byte | Nilai warna hijau |
| 2 | RgbRed | 1 byte | Nilai warna merah |
| 3 | RgbReserved | 1 byte | Bernilai 0 |

## 2.9 Pixel (Picture Elemen) [2]

Gambar yang bertipe *bitmap* tersusun dari *pixel-pixel*. *Pixel* disebut juga dengan dot. *Pixel* berbentuk bujur sangkar dengan ukuran relatif kecil yang merupakan penyusun/pembentuk gambar *bitmap*.

Banyaknya*pixel* tiap satuan luas tergantung pada resolusi yang digunakan. Keanekaragaman warna *pixel* tergantung pada *bit depth* (kedalaman warna) yang dipakai. Semakin banyak jumlah *pixel* tiap satu satuan luas, semakin baik kualitas gambar yang dihasilkan dan tentu akan semakin besar ukuran *file*nya.

## 2.10 Flowchart[10]

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. Flowchart menolong analis dan programmer untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. Flowchart biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut.

Simbol-simbol ini dapat dilihat pada Gambar 2.3 . Simbol Flowchart Standar berikut ini :

# BAB III
# ANALISA DAN PERANCANGAN SISTEM

## 3.1 Analisa Sistem

### 3.1.1 Analisa Desain Layout

Analisa desain layout tampilan pada program penghitung jumlah sel darah merah meliputi Input Training, Open Training, Open Test, Hitung Jumlah Sel.

### 3.1.2 Analisa Informasi

Analisa Informasi, mengenai informasi jumlah sel darah merah yang dihitung dari gambar citra sel darah merah.

### 3.1.3 Analisa User

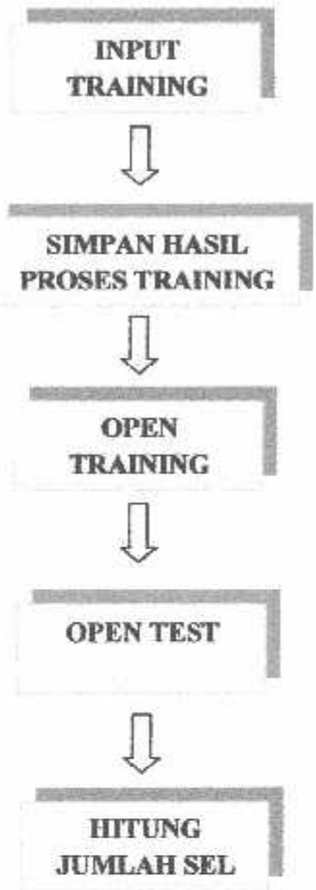
Analisa User, Kategori user yang digunakan dalam Sistem disini lebih ditujukan kepada para dokter dan analis kesehatan.

Langkah-langkah di dalam tahap analisis sistem hampir sama dengan langkah-langkah yang dilakukan dalam mengidefinisikan proyek-proyek sistem yang akan dikembangkan di tahap perencanaan sistem. Perbedaannya pada analisis sistem ruang lingkup tugasnya lebih terinci.

## 3.2 Gambaran Umum

Pada perancangan penghitung jumlah sel darah merah yang akan dibuat, secara umum meliputi beberapa proses yaitu proses Training citra, Input citra, Open test, kemudian yang terakhir yaitu proses penghitungan sel darah merah.

Blok diagram dari skripsi dapat dilihat pada gambar 3.1 dibawah ini.




**Gambar 3.1** *Blok diagram sistem penghitung sel darah merah*

Dari blok diagram diatas dapat terlihat bahwa dalam proses penghitung jumlah sel darah merah melalui beberapa proses yaitu input training, menyimpan hasil proses training, open training, open test, kemudian hitung jumlah sel darah merah. Proses input training yaitu proses pengambilan contoh salah satu bentuk citra sel darah merah pada gambar, setelah di training citra tersebut kemudian di simpan dalam format .DCR. Kemudian proses open training yaitu proses membuka file yang sudah di simpan dalam format .DCR kemudian proses open test yaitu proses pengambilan gambar citra sel darah merah yang akan di hitung. Setelah itu proses yang terakhir proses hitung jumlah sel yaitu prosesnya membandingkan bentuk citra

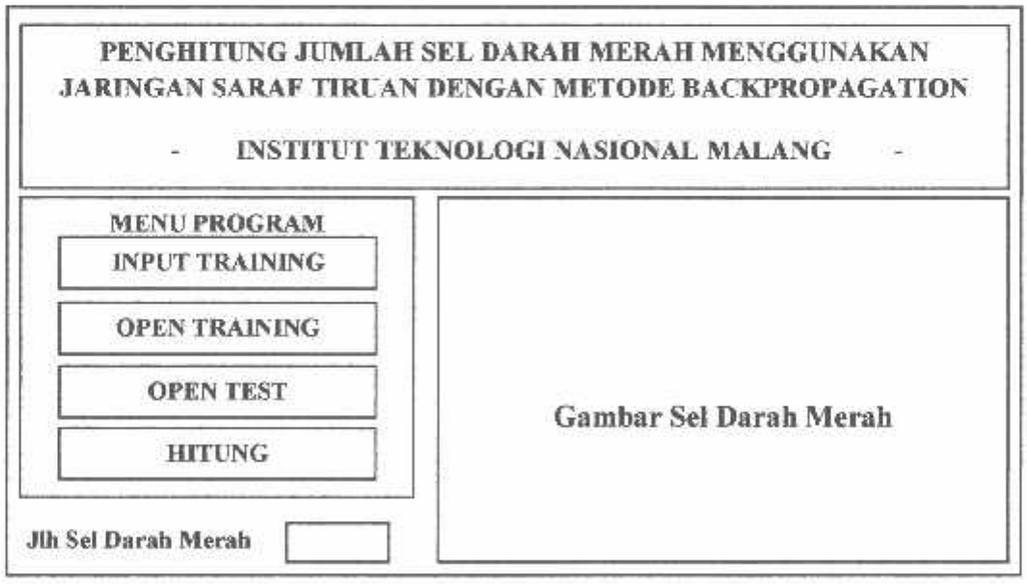
yang sudah di training tadi apabila bentuk citranya sama maka akan di anggap sel darah merah.

### 3.3 Perancangan Sistem

Untuk mengetahui seperti apa antar muka pengguna dari program penghitung sel darah merah ketika perangkat lunak digunakan. Dari rancangan ini akan terlihat bagaimana pengguna akan memasukkan data, melakukan pemilihan menu, maupun untuk mendapatkan output dari hasil pemprosesan penghitung jumlah sel darah merah. Rancangan desain layout program penghitung jumlah sel darah merah terdiri dari beberapa form sebegai berikut :

#### 3.3.1 Halaman Utama

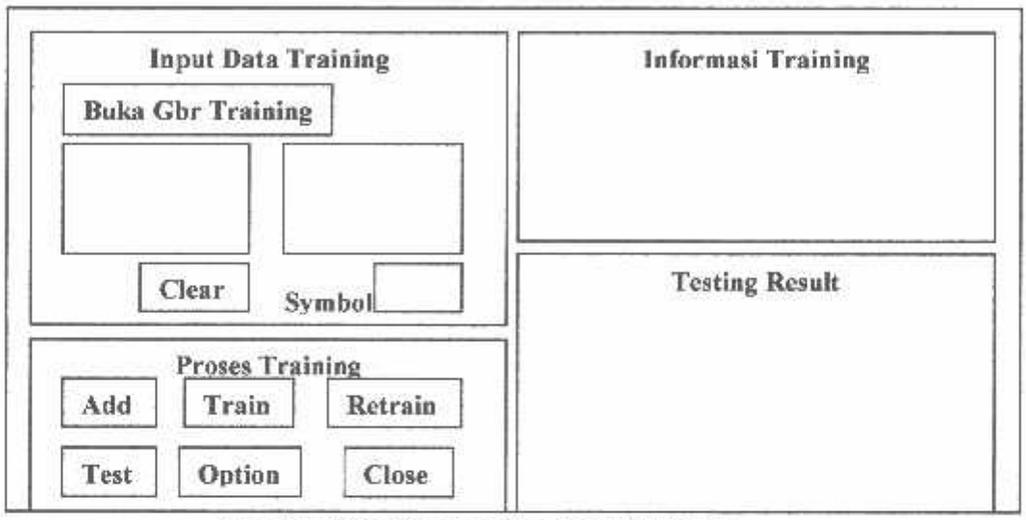
Halaman utama dari aplikasi penghitung jumlah sel darah merah menggunakan jaringan saraf tiruan dengan metode backpropagation untuk mengetahui jumlah sel darah merah pada tubuh. Pada halaman utama berisi uraian yang menampilkan tetang penjelasan dari aplikasi yang bersangkutan dan halaman ini dapat diakses oleh semua user. Berikut tampilan halaman utama dari aplikasi penghitung jumlah sel darah merah sebagai berikut :




**Gambar 3.2** *Desain Form Utama Interface*

### 3.3.2 Input Training

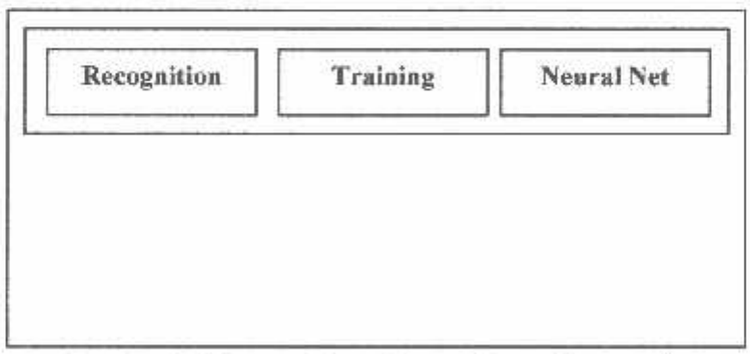
Desain form input training yang berfungsi untuk mengolah data dari sample gambar sel darah merah. Form input training ini dapat di akses oleh semua user yang meliputi input data training, informasi training, dan proses dari training tersebut dan setelah itu data di simpan dalam bentuk file dengan format .DCR. Berikut tampilan dari form input training penghitung jumlah sel darah merah :




**Gambar 3.3** *Desain Form Input Training*

### 3.3.3 Option (Form Input Training)

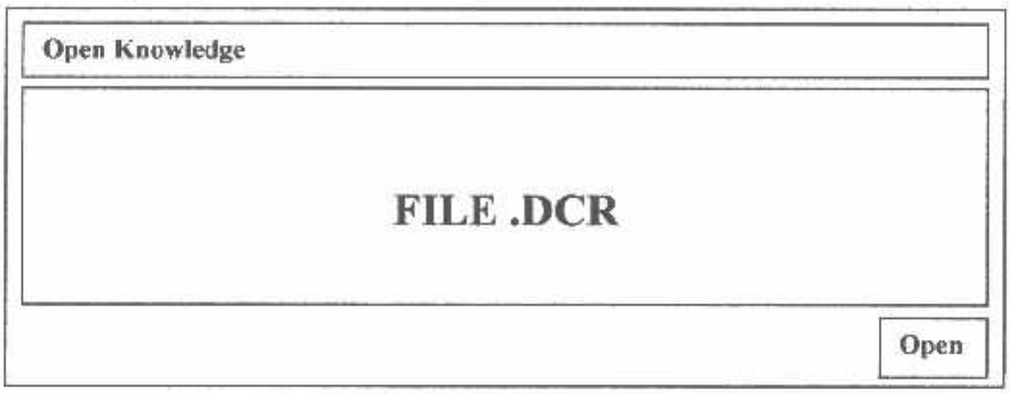
Desain form Option pada form input training yang berfungsi untuk mengatur training program dengan angka-angka tertentu yang bisa ditentukan dan di form option juga terdapat menu default. Pada form option ini meliputi Recognition, Training, Neural Net. Berikut tampilan desain form option pada input training :



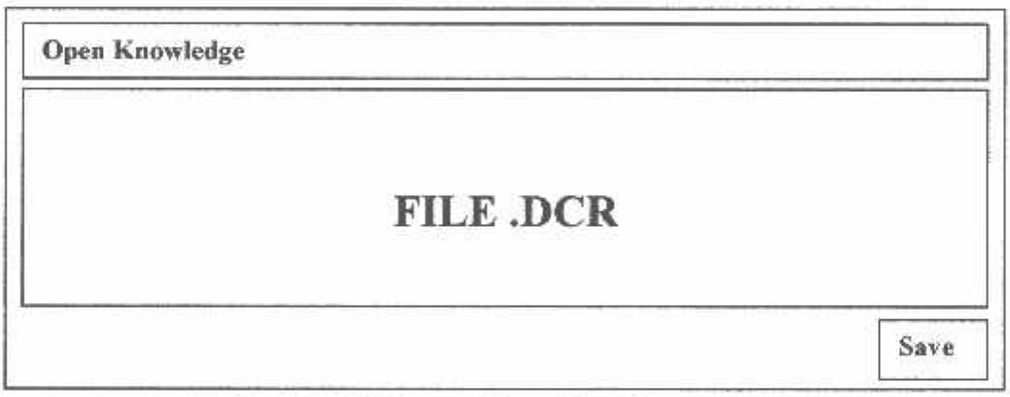

**Gambar 3.4** *Desain Form Option (Input Training)*

### 3.3.4 Form Open dan Save Training

Desain form open training yang berfungsi untuk membuka file .DCR yang sudah di training tadi sebagai perbandingan bentuk sel darah merah yang lain agar program dapat membedakan antara sel darah merah atau bukan sel darah merah. Berikut tampilan desain form open training :



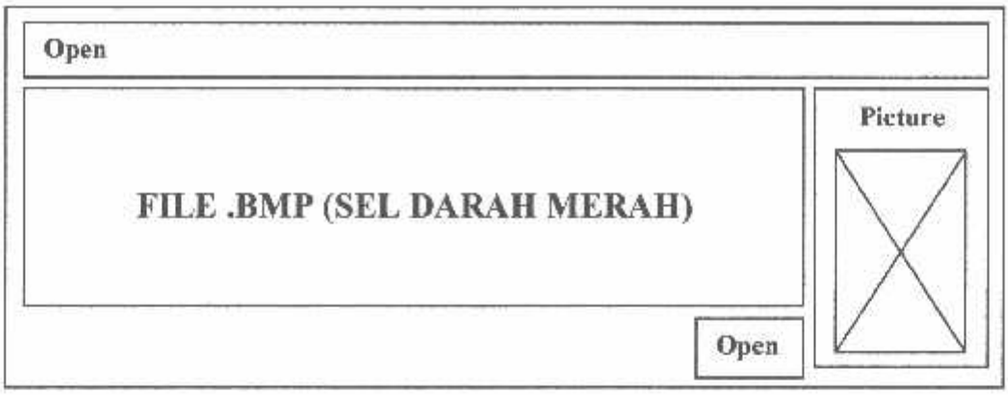

**Gambar 3.5** *Desain FormOpen Training*




**Gambar 3.6** *Desain Form Save Training*

### 3.3.5 Form Open Test

Desain form open test yang berfungsi untuk membuka file gambar citra sel darah merah .BMP yang akan dihitung jumlah sel darah merahnya pada gambar citra tersebut. Berikut tampilan desain form open test :




**Gambar 3.7** *Desain FormOpen Test*

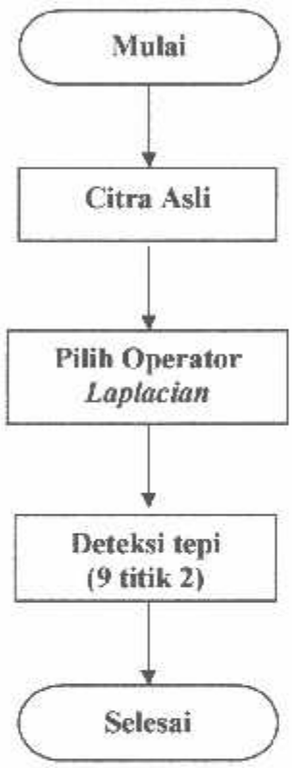
### 3.4 Flowchart Sistem


**Gambar 3.8** *Flowchart Utama*

Pada proses utama penghitungan jumlah sel darah merah memiliki beberapa proses yaitu yang pertama aplikasi akan membaca citra sel darah merah yang akan di hitung jumlahnya, setelah itu Proses selanjutnya yaitu proses perbandingan terhadap training citra dengan mengambil salah satu bentuk citra (sel darah merah) untuk dijadikan perbandingan pada pngenalan citra. Kemudian proses Preprocessing citra yaitu proses deteksi tepi agar mudah dikenali dan dihitung jumlah sel darah merah tersebut. Selanjutnya yang terakhir Deskripsi citra dan penentuan jumlah sel darah

merah apabila citra dikenali oleh program maka program akan melukan proses *counter*.

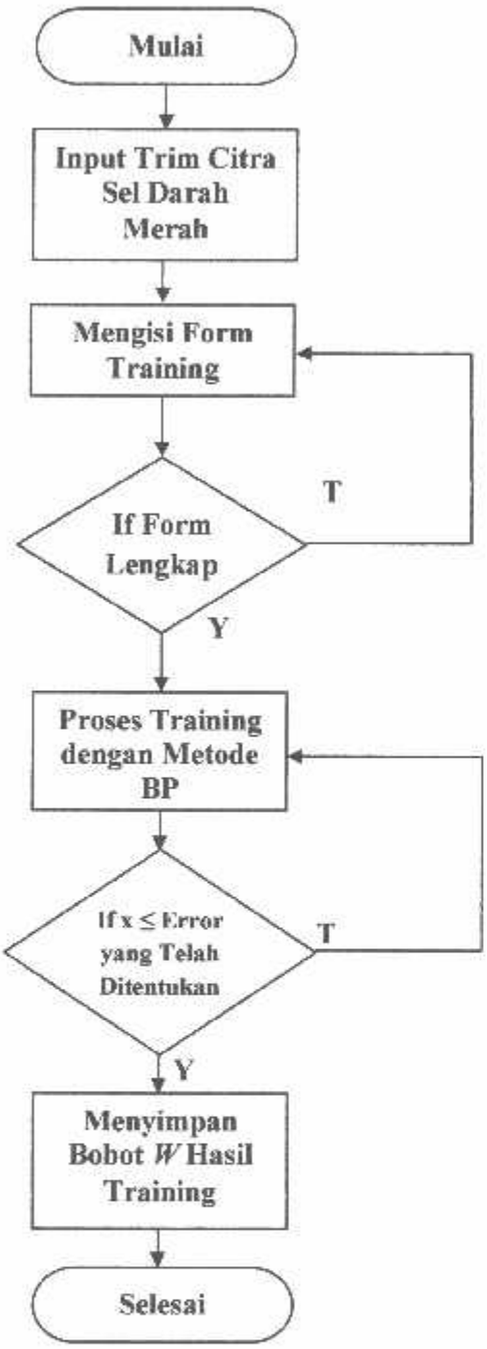
Proses pengolahan citra digital berakhir dengan tampilan deskripsi atas hasil pengolahan dalam bentuk tekstual. Karena program simulasi ini dibuat untuk menghitung jumlah sel darah khususnya sel darah merah, maka analisis yang diambil adalah jumlah sel darah merah.




**Gambar 3.9** *Flowchart Deteksi Tepi*

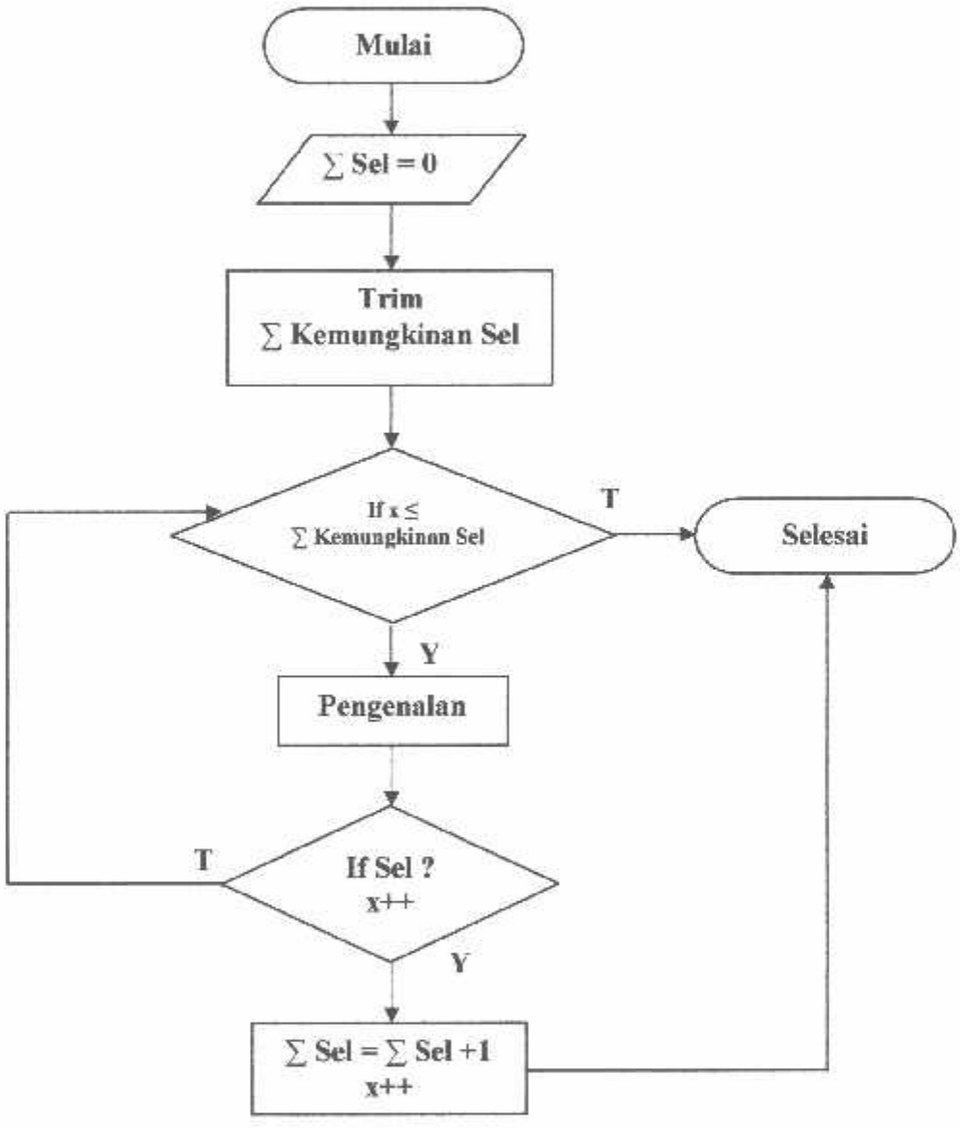
Sel darah merah memiliki bentuk umum menyerupai cakram dengan tengah yang cekung. Efek pencahayaan menyebabkan beberapa sel darah merah terlihat terang pada bagian tengahnya seperti donat. Hal ini dapat menimbulkan kesulitan dalam pengolahan. Oleh karena itu perlu dilakukan proses deteksi tepi menggunakan operator laplacian. Dalam proses deteksi tepi mempunyai pilihan laplacian 5 titik,

laplacian 9 titik 1, dan laplacian 9 titik 2. Setelah itu selesai dan citra sel darah tersebut disimpan dengan format .BMP 8 bit.



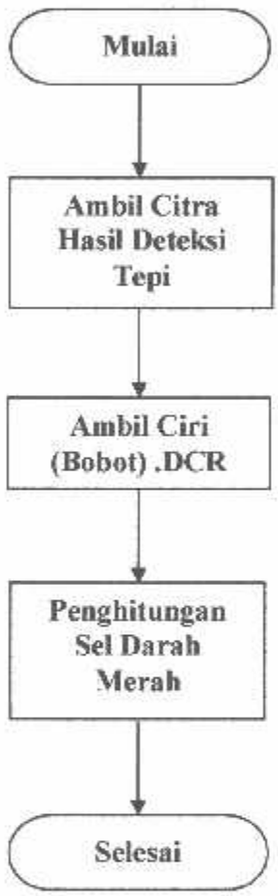

**Gambar 3.10** *Flowchart Pembelajaran (Backpropagation)*

Pada flowchart pembelajaran ini proses pertama yaitu mengambil sample citra sel darah merah kemudian mengisi form training misalnya huruf "A" setelah itu aplikasi melakukan proses training dengan metode backpropagation dan apabila nilai bobot "W" lebih kecil dari pada nilai error yang ditentukan maka nilai bobot akan disimpan dan apabila nilai errornya lebih besar dari nilai yang ditentukan maka aplikasi melakukan proses retrain sampai nilai error menjadi lebih kecil dari nilai error yang ditentukan.




**Gambar 3.11** *Flowchart Perhitungan*

Pada flowchart perhitungan di atas jumlah sel masih dalam keadaan 0. Kemudian ambil hasil trim dari sel darah merah. Setelah itu apabila jumlah sel lebih kecil dari pada kemungkinan sel maka akan melakukan proses pengenalan dan apabila sel darah merah maka aplikasi akan counter dan proses selesai. jika sel darah merah belum di kenali seluruhnya oleh aplikasi maka akan kembali ke proses pengenalan lagi sampai jumlah sel darah merah terhitung keseluruhan oleh aplikasi.




**Gambar 3.12** *Flowchart Pengujian*

Pada flowchart pengujian ini proses pertama yaitu mengambil citra hasil sel darah merah yang telah dideteksi tepi kemudian menginputkan ciri dari sel darah

merah yang telah ditraining dengan format .DCR. Setelah itu proses terakhir penghitungan jumlah sel sel darah merah.

# BAB IV
# PENGUJIAN DAN HASIL

Dalam bab ini dibahas mengenai hasil uji coba program yang telah dirancang dan dibuat, serta kontribusi program. Uji coba dilakukan untuk mengetahui apakah program dapat berjalan sebagaimana mestinya dengan lingkungan uji coba yang telah ditentukan serta dilakukan sesuai dengan skenario uji coba.

Ada beberapa hasil uji coba yang telah dilakukan terhadap data yang telah dipilih, antara lain: Merubah citra gambar sel darah merah menjadi grayscale, Training, Open Training, Open Test, dan Hitung jumlah sel darah merah.

## 4.1 Linkungan Uji Coba

Pada subbab ini dijelaskan mengenai lingkungan uji coba yang meliputi perangkat lunak dan perangkat keras yang digunakan. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam uji coba antara lain adalah:

**Tabel 4.1** Tabel *Lingkungan Uji Coba*

| | |
|---|---|
| Perangkat Keras | Prosesor : Intel Pentium dual-core 1.86 GHz<br>Memori : 120 GB<br>Piranti Masukan : - Citra dari Internet<br>- Mouse<br>- Keyboard |
| Perangkat Lunak | Sistem Operasi : Microsoft Windows XP Professional 2002 SP 3<br>Perangkat Pengembang : Borland Delphi 7.0 |

## 4.2 Data Uji Coba

Pada uji coba yang akan dilakukan, digunakan data yang berasal dari gambar citra sel darah merah dari internet yang telah ada dan betipe .bmp. Data gambar sel

darah merah yang dimasukkan kedalam program berupa gambar sel darah merah yang sudah di rubah menggunakan operator laplacian.

## 4.3 Hasil Penelitian

### 4.3.1 Hasil Pengujian Deteksi Tepi (*Laplacian*)

Gambar / citra dapat di konversi menggunakan laplacian 5 titik, laplacian 9 titik 1, dan laplacian 9 titik 2. Namun disini penulis menggunakan laplacian 9 titik 1 karena hasil Deteksi tepi citra tersebut lebih tajam dan lebih mudah d kenali oleh program penghitung sel darah merah. Hasil dari citra yangx di deteksi tepi menggunakan operator 5 titik, laplacian 9 titik 1 dan laplacian 9 titik 2 dapat dilihat pada gambar 4.1, 4.2, dan 4.3.

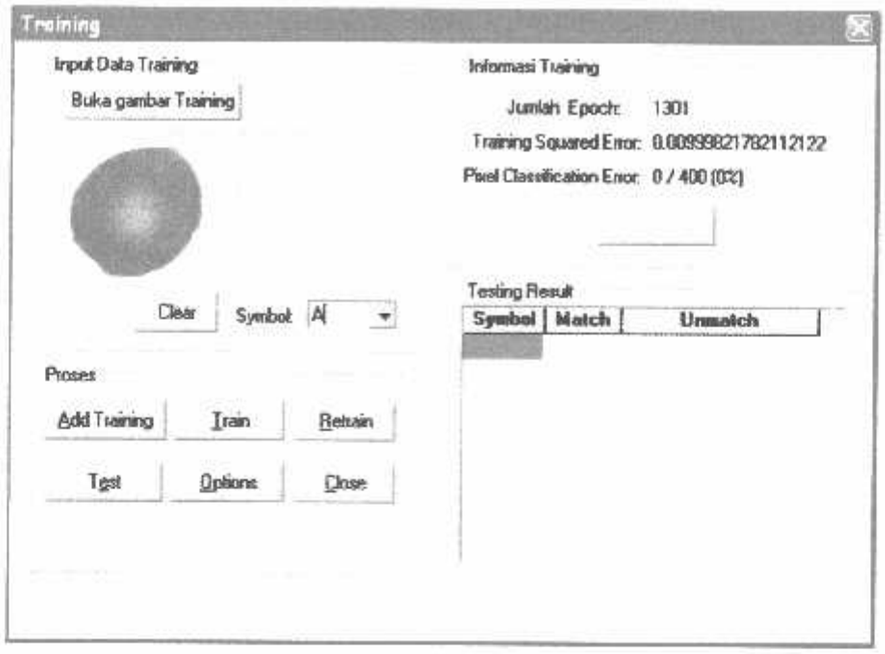

**Gambar 4.1** *Deteksi tepi menggunakan laplacian 5 titik*


**Gambar 4.2** *Deteksi tepi menggunakan laplacian 9 titik 1*

**Gambar 4.3** *Deteksi tepi menggunakan laplacian 9 titik 2*

### 4.3.2 Hasil Pengujian Training Program

Proses pengenalan citra sel darah merah pada program ini menggunakan jaringan saraf tiruan. Jadi sebelum melakukan proses penghitungan sel darah merah lebih dulu melakukan proses training pada sample sel darah merah tersebut. Setelah di training citra sel yang dijadikan sample tadi di simpan dalam bentuk file .DCR. Sel darah merah yang dijadikan sample dengan ukuran 100px x 100px. Proses tersebut dapat dilihat pada gambar 4.4.

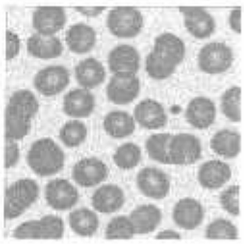


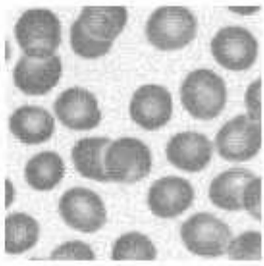
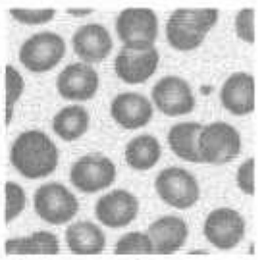
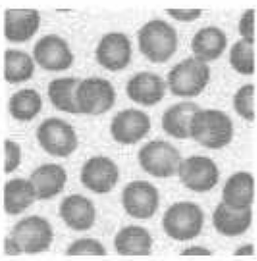
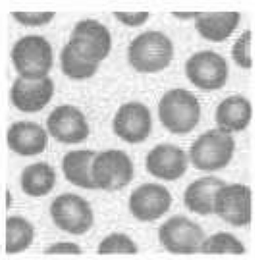

**Gambar 4.4** *Proses training*

### 4.3.3 Hasil Pengujian Penghitung Sel Darah Merah

Pengujian aplikasi program ini dilakukan menggunakan contoh citra sel darah merah yang telah ada setelah itu dirubah menjadi hitam dan putih menggunakan operator laplacian. Pengujian ini bertujuan untuk mengetahui kemampuan aplikasi yang digunakan untuk mengenali data sel darah merah. Pengujian ini dilakukan 10 data citra sel darah merah. Hasil pengenalan citra ditunjukkan pada tabel 4.1.

**Tabel 4.2** *Rekapitulasi Hasil Pengujian*

| No. | Citra Sel Darah Merah | Penghitungan Aplikasi | | | Penghitungan Manual | Kesalahan Relatif | | |
|---|---|---|---|---|---|---|---|---|
| 1. | | <196 | 196 | >196 | | <196 | 196 | >196 |
| |  | 4 | 27 | 15 | 27 | 85% | 0% | 40% |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7. |  | 2 | 30 | 14 | 27 | 92% | 11% | 48% |
| 8. |  | 2 | 31 | 12 | 31 | 93% | 0% | 61% |
| 9. |  | 2 | 33 | 17 | 33 | 93% | 0% | 48% |
| 10. |  | 3 | 27 | 14 | 27 | 88% | 0% | 48% |
| KR rata-rata = $\frac{\sum KR}{N}$ | | | | | | 85% | 3.8% | 54% |

Dari tabel 1 dijelaskan bahwa citra sel darah merah yang di uji menggunakan citra RGB dengan ukuran 280 x 280 pixel. Pengenalan citra oleh aplikasi dari sample no.1 dapat dilihat target sel darah merah berjumlah 27 dan dikenali oleh aplikasi dengan

jumlah sel yang sama yaitu dapat dihitung nilai kesalahan relatif ($KR=\frac{jumlah\ sel\ target\ -\ jumlah\ sel\ yg\ dikenali}{jumlah\ sel\ target}$ x 100%) yaitu 0%. Hasil nilai KR yang sama ditunjukkan pada sample no. 1,2,5,8,9 dan10 sedangkan pada sample sel darah yang lainnya untuk contoh no. 3 dari jumlah sel darah merah yang berjumlah 30 kemudian dikenali oleh program 33 hal ini terjadi karena masih adanya noise ( citra yang bukan sel darah merah) yang tidak mampu dinetralisasi oleh operator laplacian. Jadi dalam pengujian pengenalan citra di atas memiliki keberhasilan 96.2% dengan tresholding 196.

% Keberhasilan Sel Darah = 100% – KR

= 100% - 3.8%

= 96.2%

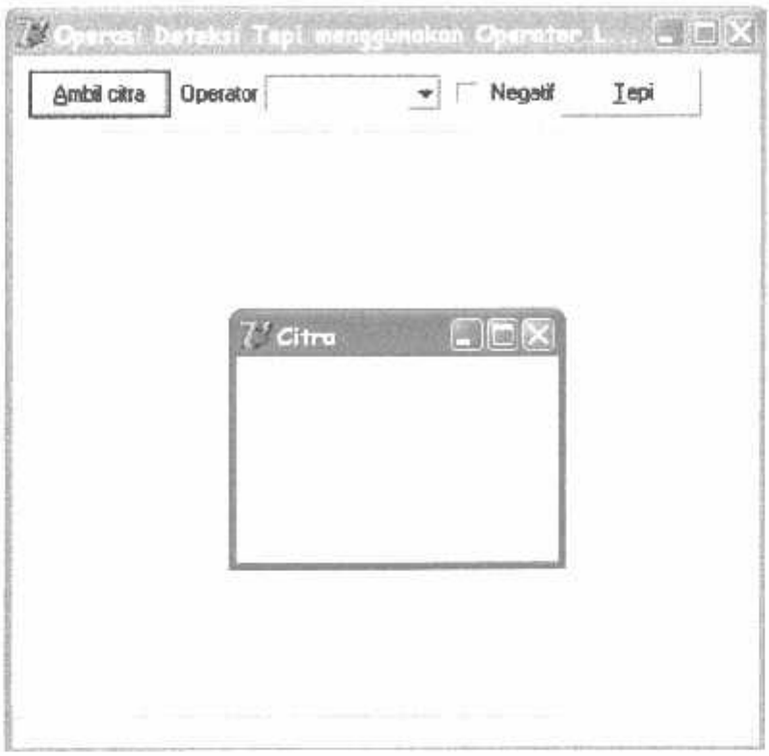
## 4.4 Hasil Tampilan Program

Aplikasi penghitungan jumlah sel darah merah ini, menggunakan bahasa pemrograman delphi 7, dan masih manual. Adapun aplikasinya memiliki 2 tampilan, yaitu tampilan aplikasi pengolahan citra digital dengan deteksi tepi menggunakan operator laplacian dan aplikasi penghitungan jumlah sel darah merah.

### 4.4.1 Aplikasi Deteksi Tepi

Aplikasi pengolahan citra terdiri dari menu "Ambil" yang berfungsi untuk membuka image dari drive tertentu , "Operator" berfungsi untuk pendeteksian tepi yaitu laplacian 5 titik, laplacian 9 titik 1, dan laplacian 9 titik 2, botton "Tepi" berfungsi untuk melakukan eksekusi deteksi tepi.

Adapun tampilan aplikasi pengolahan citra deteksi tepi dengan operator laplacian dapat di lihat pada gambar 4.5.

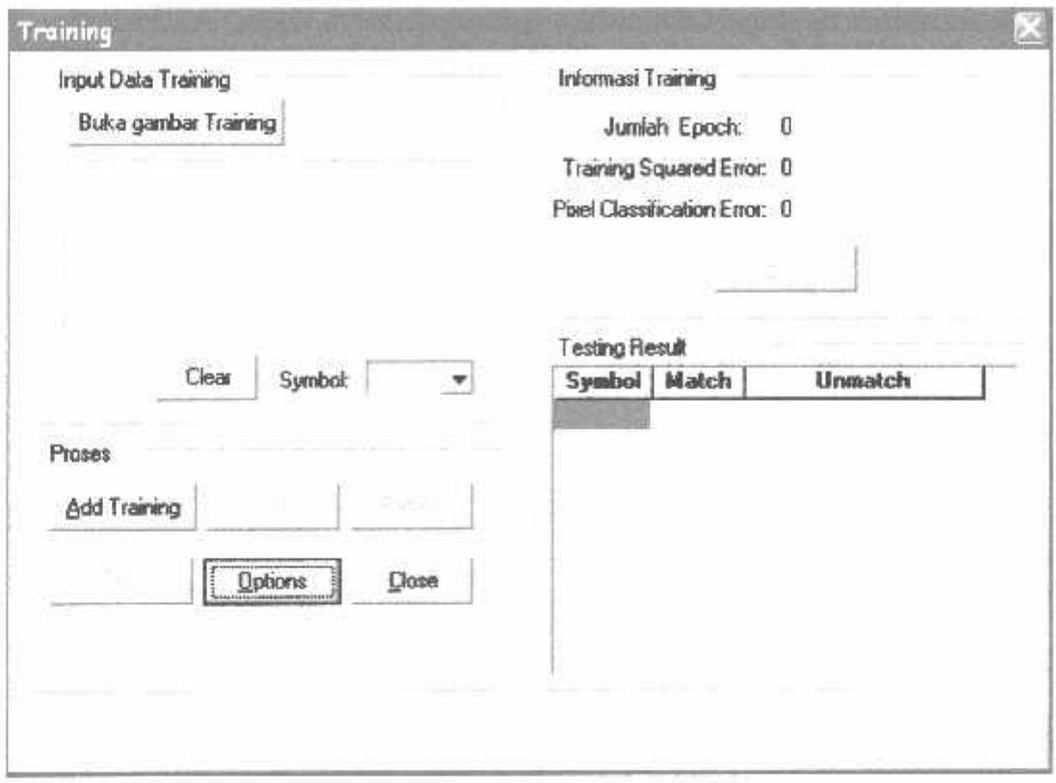



**Gambar 4.5** *Aplikasi pengolahan citra untuk deteksi tepi*

### 4.4.2 Aplikasi Training Sel Darah Merah (JST)

Pada proses training sel darah merah menggunakan jaringan saraf tiruan terdiri dari beberapa menu "Input" yang berfungsi untuk membuka image dari drive tertentu, setelah data diinput kemudian isi symbol tersebut dengan huruf A sebagai inisialisasi, "Add Training" botton ini berfungsi untuk menambahkan inputan sel darah kedalam aplikasi, "Train" berfungsi untuk memulai proses training sel darah merah, "Retrain" berfungsi untuk mengulang kembali proses training agar lebih detail lagi dalam prosesnya, "Test" berfungsi untuk menampilkan hasil akhir dari proses training, "Option" berfungsi untuk mengatur *recognation, training, neural net*, dan "Close" berfungsi untuk keluar dari proses training.

Adapun tampilan Training sel darah merah menggunakan jaringan saraf tiruan dapat di lihat pada gambar 4.6.




**Gambar 4.6** *Tampilan Training Sel darah merah*

### 4.4.3 Aplikasi Utama Penghitung Sel Darah Merah

Setelah itu sel citra yang sudah di deteksi tepi di ambil dan di masukkan ke aplikasi utama penghitung sel darah merah. Dan pada aplikasi utama penghitung jumlah sel darah merah terdapat menu "Input Training" berfungsi untuk melakukan proses pengenalan sample citra sel darah merah untuk di jadikan perbandingan dengan citra sel darah merah yang lain. Menu "Open Training" berfungsi untuk mengambil file .DCR yang sudah ditraining. Menu "Open Test" berfungsi membuka file citra sel darah merah yang sudah melakukan deteksi tepi untuk di hitung jumlahnya. Yang terakhir menu "Hitung" berfungsi untuk penghitungan jumlah sel darah merah.

Adapun tampilan aplikasi penghitung jumlah sel darah menggunakan jaringan saraf tiruan dengan metode backpropagation dapat di lihat pada gambar 4.7.


**Gambar 4.7** *Aplikasi penghitung jumlah sel darah merah*

Dari gambar 4.7 diatas dapat dilihat bahwa aplikasi ini merupakan sebuah perkembangan teknologi kedokteran yang kemudian dikembangkan secara terus menerus sehingga dapat ditemukannya pemaduan ilmu medis dengan teknologi komputerisasi yang semakin berkembang di masyarakat di zaman sekarang ini yaitu aplikasi penghitung jumlah sel darah merah menggunakan jaringan saraf tiruan dengan metode backpropagation.

# BAB V
# PENUTUP

## 5.1 Kesimpulan

Berdasarkan uraian yang telah diberikan pada bab-bab terdahulu dapat ditarik kesimpulan sebagai berikut :

1. Deteksi tepi terbaik menggunakan operator laplacian 9 titik 1 karena hasil deteksi tepi sel darah merah sangat baik.
2. Program simulasi ini dapat menghitung jumlah sel darah merah dalam citra memiliki nilai black & white treshold optimal pada level 196 pada range 0 – 255.
3. Proses training rate akan berhenti apabila jumlah epoch lebih dari 10000. Batasan jumlah epoch dapat dirubah sesuai keinginan.
4. Proses penghitungan jumlah sel darah merah setiap pengujian citra menggunakan ukuran pixel 210 x 210.
5. Informasi yang di hasilkan berupa jumlah sel darah merah dengan akurasi 96,2 %.

## 5.2 Saran

1. Penelitian dapat dilanjutkan untuk mengenali jenis sel darah yang lain seperti sel darah putih, keping darah, dan sebagainya sehingga dapat diketahui kemungkinan penyakit yang diderita pasien.
2. Pengambilan preparat sel darah merah sebaiknya dilakukan oleh tenaga medis yang berpengalaman karena tingkat ketelitian dan kualitas citra sangat dipengaruhi oleh proses pemotretan.

## DAFTAR PUSTAKA


[1] Ahmad, B., Firdausy, K., 2005, *Teknik Pengolahan Citra Digital Menggunakan DELPHI*, Yogyakarta, Ardi Publishing

[2] Binus, 2007. *Thesis*. http://library.binus.ac.id/eColls/eThesis/Bab2/2007-2-00459-MTIF-Bab%202.pdf

[3] Drdjebrut, 2011. *Gambaran Sel darah Normal*. Diakses tanggal 23 Desember 2012. Dari http://drdjebrut.wordpress.com/tag/sel-darah/

[4] Fatmaclass, 2009. *Pengantar Pemrograman delphi*. Diakses tanggal 19 November 2012. Dari fatmaclass.files.wordpress.com/2009/03/modul-delphi11.pdf

[5] Hartadi Diaz, 2011. *Simulasi Penghitungan Jumlah Sel Darah Merah*, Semarang, Universitas Diponorogo.

[6] Kiki, Kusumadewi, S., *Analisis Jaringan Saraf Tiruan dengan Metode Backpropagation Untuk Mendeteksi Gangguan Psikologi*, Yogyakarta, Universitas Islam Indonesia.

[7] Muliya, M., 2006, *Pengolahan Citra Digital Menggunakan Delphi*, Medan, Universitas Pembangunan Panca Budi Medan.

[8] Mulyanto, E., Suhartono, V., Wijanarto, *Teori Pengolahan Citra Digital*, Semarang, Universitas Dian Nuswantoro, Andi.

[9] Puspita, A., Eunike, 2007, *Penggunaan Jaringan Saraf Tiruan Metode Backpropagation Untuk Memprediksi Bibir Sumbing*, Yogyakarta, Sekolah Tinggi Teknik Surabaya.

[10] Rinoan, 2005. *Simbol-simbol Flowchart*, Jakarta, Universitas Guna Dharma

[11] Siang Jong Jek, Drs, M.Sc., 2009, *Jaringan Saraf tiruan dan Pemrogramannya Menggunakan MATLAB*, Yogyakarta, Andi.

# LAMPIRAN

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI
NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

# BERITA ACARA UJIAN SKRIPSI
# FAKULTAS TEKNOLOGI INDUSTRI

Nama : Ihsan

NIM : 11.18.908

Jurusan : Teknik Informatika S-1

Judul Skripsi : **PENGHITUNG JUMLAH SEL DARAH MERAH MENGGUNAKAN JARINGAN SARAF TIRUAN DENGAN METODE BACKPROPAGATION**

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :

Hari : Selasa

Tanggal : 19 Februari 2013

Nilai : 85,4 (A)

Panitia Ujian Skripsi
Ketua Majelis Penguji

Joseph Dedy Irawan, ST, MT
NIP. 197404162005011002

Anggota Penguji

Penguji I

Michael Ardita, ST, MT.
NIP.P. 1031000434

Penguji II

Yosep Agus Pranoto, ST.
NIP.P. 1031000432

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI
NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

# FORMULIR PERBAIKAN SKRIPSI

Nama : Ihsan
NIM : 11.18.908
Jurusan : Teknik Informatika S-1
Judul Skripsi : **PENGHITUNG JUMLAH SEL DARAH MERAH MENGGUNAKAN JARINGAN SARAF TIRUAN DENGAN METODE BACKPROPAGATION**

| TANGGAL | PENGUJI | URAIAN | PARAF |
|---|---|---|---|
| 19 Februari 2013 | I | 1. Melengkapi pengujian supaya mendukung kesimpulan point 3.→ 2<br>2. Menambahkan perancangan untuk backpropagasi pada BAB III.<br>3. Memperbaiki penulisan kesimpulan point 6/ | |
| | II | 1. Menambahkan ABSTRACT<br>2. Memperbaiki flowchart dan memberikan penjelasan tiap flowchart pada BAB III.<br>3. Memperbaiki Kesimpulan.<br>4. Menambahkan pengujian dengan threshold = 196, threshold > 196 dan threshold < 196. | |

Anggota Penguji

Penguji I

Michael Ardita, ST, MT.
NIP.P.1031000434

Penguji II

Yosep Agus Pranoto, ST.
NIP.P. 1031000432

Mengetahui,

Dosen Pembimbing I

Dr. Eng. Aryuanto Soetedjo, ST, MT.
NIP. P.1030800417

Dosen Pembimbing II

Nurlaily Vendyansyah, ST

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI
NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

# FORMULIR BIMBINGAN SKRIPSI

Nama : Ihsan

NIM : 11.18.908

Masa Bimbingan : 17 Oktober 2012 s/d 17 April 2013

Judul Skripsi : **PENGHITUNG JUMLAH SEL DARAH MERAH MENGGUNAKAN JARINGAN SARAF TIRUAN DENGAN METODE BACKPROPAGATION**

| NO | TANGGAL | URAIAN | PARAF |
|---|---|---|---|
| 1 | 20 Desember 2012 | Konsultasi Mengenai Konsep Program | |
| 2 | 10 Januari 2013 | Demo Program | |
| 3 | 13 Januari 2013 | Revisi Program | |
| 4 | 20 Januari 2013 | Pengajuan Laporan Skripsi | |
| 5 | 25 Januari 2013 | Revisi Laporan Skripsi | |
| 6 | 31 Januari 2013 | Pengajuan Makalah Seminar Hasil | |
| 7 | 2 Februari 2013 | Revisi Makalah Seminar Hasil | |
| 8 | 13 Februari 2013 | Revisi Laporan Skripsi untuk Ujian Komprehensif | |

Malang, 12 April 2013
Dosen Pembimbing I

Dr. Eng. Aryuanto Soetedjo, ST, MT.
NIP. P. 1030800417

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI
NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

# FORMULIR BIMBINGAN SKRIPSI

Nama : Ihsan

NIM : 11.18.908

Masa Bimbingan : 17 Oktober 2012 s/d 17 April 2013

Judul Skripsi : **PENGHITUNG JUMLAH SEL DARAH MERAH MENGGUNAKAN JARINGAN SARAF TIRUAN DENGAN METODE BACKPROPAGATION**

| NO | TANGGAL | URAIAN | PARAF |
|---|---|---|---|
| 1 | 19 Oktober 2012 | Pengajuan Laporan Skripsi Bab.I-Bab.III | |
| 2 | 30 Oktober 2012 | Revisi Laporan Skripsi Bab.I-Bab.III | |
| 3 | 10 Januari 2013 | Demo Program | |
| 4 | 12 Januari 2013 | Pengajuan Laporan Skripsi Bab.IV-Bab.V | |
| 5 | 15 Januari 2013 | Revisi Laporan Skripsi Bab.IV-Bab.V | |
| 6 | 31 Januari 2013 | Pengajuan Makalah Seminar Hasil | |
| 7 | 3 Februari 2013 | Revisi Makalah Seminar Hasil | |
| 8 | 12 Februari 2013 | Revisi Laporan Skripsi untuk Ujian Komprehensif | |

Malang, 12 April 2013
Dosen Pembimbing II

Nurlaily Vendyansyah, ST

**Lampiran 4 : Form Deteksi Tepi**

```
unit UnitUtama;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics,
 Controls, Forms, Dialogs, ExtDlgs, Menus, StdCtrls,
 ComCtrls, Clipbrd, ExtCtrls, Spin;
type
 TFormUtama = class(TForm)
  PanelAtas: TPanel;
  ButtonAmbilCitra: TButton;
  StatusBar: TStatusBar;
  OpenPictureDialog: TOpenPictureDialog;
  ButtonTepi: TButton;
  ComboBoxOperator: TComboBox;
  Label1: TLabel;
  CheckBoxNegatif: TCheckBox;
  procedure ButtonAmbilCitraClick(Sender: TObject);
  procedure Olah;
  procedure ButtonTepiClick(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;
var
 FormUtama: TFormUtama;
implementation
uses UnitCitra;
type
 Mask3x3 = array [-1..1,-1..1] of real;
var
 FormHasil: TFormCitra;
 Mask: Mask3x3;
const
 MaskLaplacian5: Mask3x3 =
 (( 0, -1,  0),
  (-1,  4, -1),
  ( 0, -1,  0));
MaskLaplacian91: Mask3x3 =
 ((-1, -1, -1),
 (-1,  8, -1),
 (-1, -1, -1));
```

```
    MaskLaplacian92: Mask3x3 =
     ((  1, -2,  1),
      ( -2,  4, -2),
      (  1, -2,  1));
  {$R * dfm}
  procedure TFormUtama.ButtonAmbilCitraClick(Sender: TObject);
  var
   fc: string;
  begin
   if (OpenPictureDialog.Execute) then
    begin
     if (FormCitra = nil) then
      Application.CreateForm(TFormCitra, FormCitra);
     FormCitra.Image.Picture.LoadFromFile(
      OpenPictureDialog.FileName);
     FormCitra.ClientHeight :=
      FormCitra.Image.Picture.Height;
     FormCitra.ClientWidth :=
      FormCitra.Image.Picture.Width;
     FormCitra.ClientHeight :=
      FormCitra.Image.Picture.Height;
     case (FormCitra.Image.Picture.Bitmap.PixelFormat) of
      pf1bit : fc := 'biner';
      pf8bit : fc := 'keabuan';
      pf24bit : fc := 'true color';
     end;
     StatusBar.SimpleText := OpenPictureDialog.FileName
      + ' (' + IntToStr(FormCitra.Image.Picture.Width)
      + 'x' + IntToStr(FormCitra.Image.Picture.Height)
      + ', ' + fc + ')';
    end;
  end;

  procedure TFormUtama.Olah;
  var
   x, y, w, h, u, v: integer;
   PC, PH: PByteArray;
   Ki, Ri, Gi, Bi, Ko, Ro, Go, Bo: array of array of byte;
   jumlah: real;
  begin
   if (ComboBoxOperator.Text = '') then
    begin
     ShowMessage('Pilih operator gradien deteksi tepi');
     exit;
```

```
    end
  else if (ComboBoxOperator.Text = 'Laplacian 5 titik') then
   Mask := MaskLaplacian5
  else if (ComboBoxOperator.Text = 'Laplacian 9 titik 1') then
   Mask := MaskLaplacian91
  else if (ComboBoxOperator.Text = 'Laplacian 9 titik 2') then
   Mask := MaskLaplacian92;
 w := FormCitra.Image.Picture.Width;
 h := FormCitra.Image.Picture.Height;
 if (FormCitra.Image.Picture.Bitmap.PixelFormat = pf8bit)
 then
  begin
   SetLength(Ki, w, h);
   SetLength(Ko, w, h);
   for y := 0 to h-1 do
    begin
     PC := FormCitra.Image.Picture.Bitmap.ScanLine[y];
     PH := FormHasil.Image.Picture.Bitmap.ScanLine[y];
     for x := 0 to w-1 do
      begin
       Ki[x, y] := PC[x];
       Ko[x, y] := PH[x];
      end;
    end;
   for x := 1 to w-2 do
    for y := 1 to h-2 do
     begin
      jumlah := 0;
      for u := -1 to 1 do
       for v := -1 to 1 do
        jumlah := jumlah+Mask[u,v]*Ki[x-u,y-v];
      if (Abs(jumlah)<255) then
       Ko[x,y] := Round(Abs(jumlah))
      else
       Ko[x,y] := 255;
      if (CheckBoxNegatif.Checked) then
       Ko[x,y] := 255-Ko[x,y];
     end;
 for y := 0 to h-1 do
  begin
   PH := FormHasil.Image.Picture.Bitmap.ScanLine[y];
   for x := 0 to w-1 do
    PH[x] := Ko[x, y];
 end;
```

```
    Ki := nil;
    Ko := nil;
  end;
if (FormCitra.Image.Picture.Bitmap.PixelFormat = pf24bit)
then
 begin
  SetLength(Ri, w, h);
  SetLength(Gi, w, h);
  SetLength(Bi, w, h);
  SetLength(Ro, w, h);
  SetLength(Go, w, h);
  SetLength(Bo, w, h);
  for y := 0 to h-1 do
   begin
    PC := FormCitra.Image.Picture.Bitmap.ScanLine[y];
    PH := FormHasil.Image.Picture.Bitmap.ScanLine[y];
    for x := 0 to w-1 do
     begin
      Bi[x, y] := PC[3*x];
      Gi[x, y] := PC[3*x+1];
      Ri[x, y] := PC[3*x+2];
      Bo[x, y] := PH[3*x];
      Go[x, y] := PH[3*x+1];
      Ro[x, y] := PH[3*x+2];
     end;
 end;
for x := 1 to w-2 do
 for y := 1 to h-2 do
  begin
   jumlah := 0;
   for u := -1 to 1 do
    for v := -1 to 1 do
     jumlah := jumlah+Mask[u,v]*Ri[x-u,y-v];
  if (Abs(jumlah)<255) then
   Ro[x,y] := Round(Abs(jumlah))
  else
   Ro[x,y] := 255;
  if (CheckBoxNegatif.Checked) then
   Ro[x,y] := 255-Ro[x,y];
 jumlah := 0;
 for u := -1 to 1 do
  for v := -1 to 1 do
   jumlah := jumlah+Mask[u,v]*Gi[x-u,y-v];
 if (Abs(jumlah)<255) then
```

```
            Go[x,y] := Round(Abs(jumlah))
           else
            Go[x,y] := 255;
           if (CheckBoxNegatif.Checked) then
            Go[x,y] := 255-Go[x,y];
           jumlah := 0;
           for u := -1 to 1 do
            for v := -1 to 1 do
             jumlah := jumlah+Mask[u,v]*Bi[x-u,y-v];
           if (Abs(jumlah)<255) then
            Bo[x,y] := Round(Abs(jumlah))
           else
            Bo[x,y] := 255;
           if (CheckBoxNegatif.Checked) then
            Bo[x,y] := 255-Bo[x,y];
          end;
        for y := 0 to h-1 do
         begin
          PH := FormHasil.Image.Picture.Bitmap.ScanLine[y];
          for x := 0 to w-1 do
           begin
            PH[3*x] := Bo[x, y];
            PH[3*x+1] := Go[x, y];
            PH[3*x+2] := Ro[x, y];
           end;
         end;
        Ri := nil;
        Gi := nil;
        Bi := nil;
        Ro := nil;
        Go := nil;
        Bo := nil;
    end;
end;

procedure TFormUtama.ButtonTepiClick(Sender: TObject);
begin
 if (FormCitra = nil) then
  begin
   ShowMessage('Ambil dulu citra yang akan diolah');
   exit;
  end;
if (FormHasil = nil) then
 Application.CreateForm(TFormCitra, FormHasil);
```

```
  FormHasil.Caption := 'Citra Hasil';
  FormHasil.Image.Picture := FormCitra.Image.Picture;
  FormHasil.Top := FormCitra.Top;
  FormHasil.Left := FormCitra.Left+FormCitra.Width;
  FormHasil.ClientHeight :=
   FormHasil.Image.Picture.Height;
  FormHasil.ClientWidth :=
   FormHasil.Image.Picture.Width;
  FormHasil.ClientHeight :=
   FormHasil.Image.Picture.Height;
  Olah;
 end;
end.
```

**Lampiran 5 : Form Utama**

```
unit MainFrm;
interface
uses
 ActnList, Classes, ComCtrls, Controls, Dialogs, ExtCtrls, ExtDlgs, Forms,
 ImgList, Menus, StdCtrls, SysUtils, ToolWin, Windows, Graphics,
 BackPropCls, DCRFra, GlobalMdl, TextFra, jpeg, Buttons, TeEngine, Series,
 TeeProcs, Chart, DBCtrls, ShellAPI, Grids;
type
 TMainForm = class(TForm)
  mmMain: TMainMenu;
  File1: TMenuItem;
  mmiExit: TMenuItem;
  mmiTraining: TMenuItem;
  SaveKnowledgeDialog: TSaveDialog;
  OpenKnowledgeDialog: TOpenDialog;
  mmiOpenPicture: TMenuItem;
  OpenPictureDialog: TOpenPictureDialog;
  SavePictureDialog: TSavePictureDialog;
  mmiRecognize: TMenuItem;
  StatusBar: TStatusBar;
  ToolbarImages: TImageList;
  ActionList1: TActionList;
  actFileNewKnowledge: TAction;
  actFileOpenKnowledge: TAction;
  actFileSaveKnowledge: TAction;
  actFileSaveKnowledgeAs: TAction;
  actFileNewPicture: TAction,
  actFileOpenPicture: TAction;
  actFileSavePicture: TAction;
  actFileSavePictureAs: TAction;
  actFileExit: TAction;
  actFileSaveResult: TAction;
  actFileSaveResultAs: TAction;
  N3: TMenuItem;
  actRecognize: TAction;
  actTraining: TAction;
  actAbout: TAction;
  actOptions: TAction;
  mmiOptions: TMenuItem;
  actShowHideToolBar: TAction;
  actShowHideStatusBar: TAction;
  actShowHidePictureTitle: TAction;
```

```
    actShowHideResultTitle: TAction;
    actPencil: TAction;
    actEraser: TAction;
    actClear: TAction;
    pmPicture: TPopupMenu;
    Pencil2: TMenuItem;
    Eraser2: TMenuItem;
    N6: TMenuItem;
    Clear2: TMenuItem;
    SaveResultDialog: TSaveDialog;
    OpenTraining1: TMenuItem;
    Panel5: TPanel;
    Panel6: TPanel;
    DCR: TDCRFrame;
    Panel7: TPanel;
    ImageList1: TImageList;
    Image3: TImage;
    SaveDialog1: TSaveDialog;
    save1: TMenuItem;
    OpenDialog1: TOpenDialog;
    Panel13: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    BitBtn2: TBitBtn;
    BitBtn1: TBitBtn;
    BitBtn3: TBitBtn;
    Image4: TImage;
    Panel14: TPanel;
    X1: TPanel;
    ResultText: TTextFrame;
    Panel2: TPanel;
    Panel3: TPanel;
    Panel8: TPanel;
    Image1: TImage;
    BitBtn4: TBitBtn;
    N1: TMenuItem;
    N2: TMenuItem;
    x3: TPanel;
    citra: TImage;
    x4: TPanel;
    Image: TImage;
    StringGrid1: TStringGrid;
    Label4: TLabel;
```

```
    procedure Pencil(Sender: TObject);
    procedure ShowHideResultTitle(Sender: TObject);
    procedure ShowHidePictureTitle(Sender: TObject);
    procedure ShowHideStatusBar(Sender: TObject);
    procedure Options(Sender: TObject);
    procedure About(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FileSaveResult(Sender: TObject);
    procedure FileSaveResultAs(Sender: TObject);
    procedure Recognize(Sender: TObject);
    procedure FileSavePictureAs(Sender: TObject);
    procedure FileSavePicture(Sender: TObject);
    procedure FileOpenPicture(Sender: TObject);
    procedure FileNewPicture(Sender: TObject);
    procedure Training(Sender: TObject);
    procedure FileNewKnowledge(Sender: TObject);
    procedure FileOpenKnowledge(Sender: TObject);
    procedure FileSaveKnowledgeAs(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FileExit(Sender: TObject);
    procedure FileSaveKnowledge(Sender: TObject);
    procedure OpenTraining1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure BitBtnClick(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure save1Click(Sender: TObject);
    procedure BitBtn4Click(Sender: TObject);
    procedure Panel8Click(Sender: TObject);
    procedure Image4Click(Sender: TObject);
private
    TB      : Array[0..255] of int64;
    DB      : Byte;
    TeTellenF : File Of Byte;
    EenPct : real;
    FKnowledgeFileName: string;
    FPictureFileName: string;
    FResultTextFileName: string;
    Procedure Init;
    procedure CheckKnowledgeSave;
    procedure CheckPictureSave;
    procedure ShowHint(Sender: TObject);
    Procedure WriteGraph;
```

```
    procedure ClearImage;
     Procedure TelFile,
   public
    CancelFlag : Boolean;
    { Public declarations }
    // Function CheckKnowledge(A:String).String;
   end;
  const
   APP_TITLE = 'DCR';
  var
   MainForm: TMainForm;
  implementation
  uses AboutFrm, OptionsFrm, TrainingFrm, UPrint, filehistU2, UnitUtama;
  {$R *.dfm}
  {$ASSERTIONS ON}
  resourcestring
   SAVE_KNOWLEDGE_CHANGES = 'Disimpan ke %s?';
   SAVE_PICTURE_CHANGES = 'Simpan ke  %s?';
   SAVE_RESULT_CHANGES = 'Simpan Training ke %s?';
   UNTITLED = '?';
   Procedure TMainForm.Init;
  Var I : integer;
  Begin
   For I := 0 To 255 Do
    TB[I] := 0;
  End;
   PROCEDURE TMainForm TelFile;
  Var
  // Kleinste: Char;
  // Grootste : Char;
   ByteMin,
   ByteMax,
      I : Integer;
  NumRead : Integer;
   Totaal : Int64;
    Step : Real;
    buf : array[1..2048] of Byte;
   FSize : Longword ; // Int64;
Begin
try
 {$I-}
 //AssignFile(TeTellenF, OpenDialog1.FileName);
 AssignFile(TeTellenF, OpenPictureDialog.FileName);
 FileMode := 0;  // { Set file access to read only }
```

```
      Reset(TeTellenF);
      FSize := FileSize(TeTellenF);
      {$I+}
      Form2 := TForm2.Create(Application);
      //Form1.Enabled := False;
      Form2.Show;
      Init;
      Step := Round(FSize / 100) + 0.0000001;
      Totaal := 0;
      Repeat
       BlockRead(TeTellenF,buf,1024,NumRead);
       Totaal := Totaal + NumRead;
       Application.Title := ''+ FloatToStrF(Totaal/Step,ffGeneral,2,0) + '%';
       Form2.Progressbar1.Position := Round(Totaal / Step);
       Form2.Label2.Caption :=  IntToStr(ToTaal) + ' Bytes';
       FOR I := 1 To NumRead Do Begin
        DB := buf[I];
        TB[DB] := TB[DB] + 1;
        Application.ProcessMessages;
        If CancelFlag = True then exit;
       End
      Until NumRead = 0;
      { Array is met bytes gevuld }
      ByteMax := TB[0];  { eerste is grootste }
      //Grootste := Chr(0);
      For I := 1 To 255 Do
       Begin
        IF TB[I] > ByteMax Then
         Begin
          ByteMax  := TB[I];
          //Grootste := Chr(I);
         End;
       End;
     // Kleinste := Chr(0);  { eerste is kleinste }
     ByteMin := TB[0];
     For I := 1 To 255 Do Begin
      IF TB[I] < ByteMin Then Begin
       ByteMin  := TB[I];
       // Kleinste := Chr(I);
      End;
    End;
    Label1.Caption := 'File : ' + OpenDialog1.FileName +
      ' Size : ' +
      IntToStr(FSize) + ' Bytes';
```

```
        Font.Name := 'Courier';
        Font.Size := 8;
        ClearImage;
        Pen.Color := clBlack;                    // color of the data line
        { maak assen stelsel }
        MoveTo(Links,(Onder - Trunc(VertDiv*10)) - 10 );
        LineTo(Links,Onder);                                  // links vert.
        MoveTo(Links,Onder);
        LineTo(Rechts +1,Onder);                               // basis lijn
        MoveTo(Rechts,(Onder - Trunc(VertDiv*10)) - 10 );
        LineTo(Rechts,Onder);                                 // rechts vert.
        { zet procenten by de linker Y schaal }
        for I := 0 To IMax do begin
         Str(I*10:3,S);
         S := S + '%_';
         TextOut(Links - 40,(Onder - Trunc( I * VertDiv) - 12) ,S)
        End;
        { zet getallen by de rechter Y schaal }
        For I := 0 To IMax Do Begin
         Str(Trunc(EenPct * 10 * I),S);
         S := '_' + S;
         if I = 0 then S := S + ' Bytes';
         TextOut(Rechts + 1,(Onder + 10) - Trunc( I * VertDiv) - 22,S)
        End;
        { maak grafiek }
        For I := 0 To 255 Do begin
         // Bar(Links+I*2, Onder-Trunc(TB[I]*EenDiv/EenPct), (Links+I*2)+2, Onder);
         MoveTo(Links+2+I*2, Onder-Trunc(TB[I]*EenDiv/EenPct));
         LineTo((Links+2+I*2), Onder);
        end;
      end;
    end;
    procedure TMainForm.About(Sender: TObject);
    begin
      AboutForm.ShowModal;
    end;
    procedure TMainForm.CheckKnowledgeSave;
    var
     SaveResp: Integer;
    begin
     if BackProp.Modified then
     begin
      SaveResp := MessageDlg(Format(SAVE_KNOWLEDGE_CHANGES,
    [FKnowledgeFileName]),
```

```
                    mtConfirmation, mbYesNoCancel, 0);
      case SaveResp of
       idYes: FileSaveKnowledge(Self);
       idNo: { Do nothing };
       idCancel: Abort;
      end;
     end;
    end;

    procedure TMainForm.CheckPictureSave;
    var
     SaveResp: Integer;
    begin
     if DCR.Modified then
     begin
      SaveResp := MessageDlg(Format(SAVE_PICTURE_CHANGES, [FPictureFileName]),
                    mtConfirmation, mbYesNoCancel, 0);
      case SaveResp of
       idYes: FileSavePicture(Self);
       idNo: { Do nothing };
       idCancel: Abort;
      end;
     end;
    end;
    procedure TMainForm.FormClose(Sender: TObject; var Action: TCloseAction);
    begin
     CheckKnowledgeSave;
    end;
    procedure TMainForm.FormCreate(Sender: TObject);
    begin
     FKnowledgeFileName := UNTITLED;
     FPictureFileName := UNTITLED;
     FResultTextFileName := UNTITLED;

     BackProp := TBackProp.Create(DEFAULT_INPUT_PATTERN_HEIGHT,
                    DEFAULT_INPUT_PATTERN_WIDTH,
                    DEFAULT_TARGET_PATTERN_HEIGHT,
                    DEFAULT_TARGET_PATTERN_WIDTH,
                    DEFAULT_NUMBER_OF_HIDDEN_NEURON);
     Application.OnHint := ShowHint;
     DCR.Init;
    end;
    procedure TMainForm.FormDestroy(Sender: TObject);
    begin
```

```
    BackProp.Free;
  end;
  procedure TMainForm.FileExit(Sender: TObject);
  begin
   Close;
  end;
  procedure TMainForm.FileNewKnowledge(Sender: TObject);
  begin
   CheckKnowledgeSave;
   BackProp.NewKnowledge;
   FKnowledgeFileName := UNTITLED;
  end;
  procedure TMainForm.FileOpenKnowledge(Sender: TObject);
  begin
   CheckKnowledgeSave;
   if OpenKnowledgeDialog.Execute then
   begin
    FKnowledgeFileName := OpenKnowledgeDialog.FileName;
    Screen.Cursor := crHourglass;
    BackProp.OpenKnowledge(FKnowledgeFileName);
    Screen.Cursor := crDefault;
   end;
  end;
  procedure TMainForm.FileSaveKnowledge(Sender: TObject);
  begin
   if FKnowledgeFileName = UNTITLED then
    FileSaveKnowledgeAs(Sender)
   else
   begin
    Screen.Cursor := crHourglass;
    BackProp.SaveKnowledge(FKnowledgeFileName);
    Screen.Cursor := crDefault;
   end;
  end;
  procedure TMainForm.FileSaveKnowledgeAs(Sender: TObject);
  begin
   if SaveKnowledgeDialog.Execute then
   begin
    FKnowledgeFileName := SaveKnowledgeDialog.FileName;
    Screen.Cursor := crHourglass;
    BackProp.SaveKnowledge(FKnowledgeFileName);
    Screen.Cursor := crDefault;
   end;
  end;
```

```
procedure TMainForm.Training(Sender: TObject);
begin
 TrainingForm.ShowModal;
end;
procedure TMainForm.FileNewPicture(Sender: TObject);
begin
 CheckPictureSave;
 DCR.Clear;
 FPictureFileName := UNTITLED;
end;
procedure TMainForm.FileOpenPicture(Sender: TObject);
Var
Hasil,Hasil1 : String;
Selesai : Boolean;
I : Integer;
begin
 CheckPictureSave;
 if OpenPictureDialog.Execute then
 begin
  FPictureFileName := OpenPictureDialog.FileName;
  Screen.Cursor := crHourglass;
  DCR.OpenPicture(FPictureFileName);
  Screen.Cursor := crDefault;
  Panel2.Caption:= FPictureFileName;
 Hasil:='';
 Selesai:=False;
 I:= Length(Panel2.Caption)-4;
 Repeat
    If Panel2.Caption[i]<>'\' Then
      Begin
       Hasil:=Hasil+Panel2.Caption[i];
      End Else
      Begin
       Selesai:=TRue;
      ENd;
    Dec(I);
Until Selesai=TRue;
Panel2.Caption:= Hasil;
Hasil1:='';

I:= Length(Hasil);
Repeat
 Hasil1:=Hasil1+Hasil[i];
 Dec(I);
```

```
    begin
     if SaveResultDialog.Execute then
     begin
      FResultTextFileName := SaveResultDialog.FileName;
      ResultText.SaveText(FResultTextFileName);
     end;
    end;
    procedure TMainForm.Options(Sender: TObject);
    begin
     OptionsForm.ShowModal;
    end;
    procedure TMainForm.Pencil(Sender: TObject);
    begin
     DCR.DrawingTool := dtPencil;
     actEraser.Checked := False;
     actPencil.Checked := True;
    end;
    procedure TMainForm.ShowHidePictureTitle(Sender: TObject);
    begin
     DCR.ShowHideTitle;
     actShowHidePictureTitle.Checked := not actShowHidePictureTitle.Checked;
    end;
    procedure TMainForm.ShowHideResultTitle(Sender: TObject);
    begin
     ResultText.ShowHideTitle;
     actShowHideResultTitle.Checked := not actShowHideResultTitle.Checked;
    end;
    procedure TMainForm.ShowHideStatusBar(Sender: TObject);
    begin
     StatusBar.Visible := not StatusBar.Visible;
     actShowHideStatusBar.Checked := StatusBar.Visible;
    end;
    procedure TMainForm.Recognize(Sender: TObject);
    begin
     DCR.Recognize;
     ResultText.SetText(DCR.ResultText);
     FResultTextFileName := UNTITLED;
    end;
    procedure TMainForm.OpenTraining1Click(Sender: TObject);
    begin
     CheckKnowledgeSave;
     if OpenKnowledgeDialog.Execute then
     begin
      FKnowledgeFileName := OpenKnowledgeDialog.FileName;
```

```
 // If OpenDialog1.execute then
  begin
   TelFile;
   If not cancelFlag then begin
     Save1Click(Self);
     WriteGraph;
   end;
  end;
 Image.Visible:=False;
end;
procedure TMainForm.BitBtn3Click(Sender: TObject);
Var
 XX,YY: String;
 I : Integer;
 kk,mm: INteger;
begin
DCR.Recognize;
ResultText.SetText(DCR.ResultText);
XX:='';
YY:='';
XX:=Trim(ResultText.RichEdit.Text);
kk:=0;
mm:=0;
For I:= 1 To Length(XX) do
 Begin
   if XX[i] in ['A'] Then kk:=KK+1;
   //if XX[i] in ['B'] Then mm:=mm+1;
 ENd;
Panel2.Caption:=Inttostr(kk);

//Panel1.Caption:=CheckKnowledge(Panel2.Caption);
//Panel1.Caption:=CheckKnowledge(Panel2.Caption);
end;
procedure TMainForm.save1Click(Sender: TObject);
Var I : Integer;
  StrL : TStringList;
    S : String;
 FileOut : Boolean;
Begin
 StrL := TStringList.Create;
 FileOut := false;
 If sender = save1 then
   FileOut := True;
 try
```

```
      For I := 0 To 255 Do Begin
       S := '$' + IntToHex(I,2) ;
       S := S + '=' + IntToStr(TB[I]) + ' ';
       StrL.Add(S);
       StringGrid1.Cells[(I div 16) + 1, (I mod 16) + 1] := IntToStr(TB[I]);
      End;
       If FileOut then begin
        SaveDialog1.Filename := ChangeFileExt(OpenPictureDialog.Filename,'.fhn');
        if SaveDialog1.Execute then
          StrL.SaveToFile(SaveDialog1.Filename);
       end;
    finally
     StrL.Destroy;
    end;
  End;
  procedure TMainForm.BitBtn4Click(Sender: TObject);
  begin
   CheckKnowledgeSave;
   if OpenKnowledgeDialog.Execute then
   begin
    FKnowledgeFileName := OpenKnowledgeDialog.FileName;
    Screen.Cursor := crHourglass;
    BackProp.OpenKnowledge(FKnowledgeFileName);
    Screen.Cursor := crDefault;
   end;
  end;
  procedure TMainForm.Panel8Click(Sender: TObject);
  begin
  If X1.Visible THen
    Begin
     X1.Visible:=False;
     X3.Visible:=TRue;
     X4.Visible:=TRue;
    ENd else
    Begin
     X1.Visible:=TRue;
     X3.Visible:=False;
     X4.Visible:=False;
    ENd;
  end;
  procedure TMainForm.Image4Click(Sender: TObject);
  begin
  FormUtama.Show;
  end;
```

```
    mmiPaste: TMenuItem;
    mmiCut: TMenuItem;
    mmiCopy: TMenuItem;
    Button1: TButton;
    Ole: TOpenPictureDialog;
    GroupBox2: TGroupBox;
    btnAdd: TButton;
    btnTrain: TButton;
    btnClose: TButton;
    btnTest: TButton;
    btnRetrain: TButton;
    btnOptions: TButton;
    procedure btnCloseClick(Sender: TObject);
    procedure mmiCutClick(Sender: TObject);
    procedure mmiPasteClick(Sender: TObject);
    procedure mmiCopyClick(Sender: TObject);
    procedure sgResultDrawCell(Sender: TObject; ACol, ARow: Integer;
      Rect: TRect; State: TGridDrawState);
    procedure btnStopClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure imgOrgSymbolMouseDown(Sender: TObject;
      Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
    procedure imgOrgSymbolMouseMove(Sender: TObject; Shift: TShiftState;
      X, Y: Integer);
    procedure imgOrgSymbolMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure btnClearClick(Sender: TObject);
    procedure lbCharsClick(Sender: TObject);
    procedure lbFontsClick(Sender: TObject);
    procedure btnAddClick(Sender: TObject);
    procedure btnOptionsClick(Sender: TObject);
    procedure btnTrainClick(Sender: TObject);
    procedure btnTestClick(Sender: TObject);
    procedure btnRetrainClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    FDrawing: Boolean;
    FInputPatternWidth: Integer;
    FInputPatternHeight: Integer;
    FOutputPatternWidth: Integer;
    FOutputPatternHeight: Integer;
    procedure BackPropTraining(Sender: TObject);
    procedure BackPropTrainingFinish(Sender: TObject);
```

```
    IxInput := 0;
    for f := 0 to lbFonts.Items.Count - 1 do
     if lbFonts.Selected[f] then
      for s := lbSizes.Items.Count - 1 downto 0 do
       if lbSizes.Selected[s] then
       begin
        with OrgBitmap.Canvas do
        begin
         FillRect(Rect(0, 0, OrgBitmap.Width, OrgBitmap.Height));
         Font.Name := lbFonts.Items.Strings[f];
         Font.Size := StrToInt(lbSizes.Items.Strings[s]);
         TextOut(0, 0, lbChars.Items.Strings[c]);
        end;
        AutoFitBitmap := GetAutoFitBitmap(OrgBitmap,
                                BackProp.InputPatternWidth,
                                BackProp.InputPatternHeight);
        imgNormalSymbol.Canvas.Draw(0, 0, AutoFitBitmap);
        TBackProp.BitmapToLayer(AutoFitBitmap,
                        TrainingPairs.InputPatterns[IxInput]);
        AutoFitBitmap.Free;
        if IxInput = 0 then
        begin
         AutoFitBitmap :=
          GetAutoFitBitmap(OrgBitmap, BackProp.TargetPatternWidth,
                    BackProp.TargetPatternHeight);
         TBackProp.BitmapToLayer(AutoFitBitmap,
                         TrainingPairs.TargetPattern);
         AutoFitBitmap.Free;
        end;
        IxInput := IxInput + 1;
        TrainingPairs.TargetPatternClass := lbChars.Items.Strings[c];
       end;
     BackProp.AddTrainingPairs(TrainingPairs);
    end;
  OrgBitmap.Free;
end
else if rbInputPattern2.Checked = True then
begin
 if Trim(cbSymbol.Text) = '' then
 begin
  MessageDlg('Pilih',
          mtWarning, [mbOk], 0);
  cbSymbol.SetFocus;
 end
```

```
    else
    begin
     if cbSymbol.Items.IndexOf(cbSymbol.Text) = -1 then
      cbSymbol.Items.Add(cbSymbol.Text);
     SetLength(TrainingPairs.InputPatterns, 1, BackProp.NInputNeuron + 1);
     SetLength(TrainingPairs.TargetPattern, BackProp.NOutputNeuron + 1);
     AutoFitBitmap := GetAutoFitBitmap(imgOrgSymbol.Picture.Bitmap,
                                BackProp.InputPatternWidth,
                                BackProp.InputPatternHeight);
     imgNormalSymbol.Canvas.Draw(0, 0, AutoFitBitmap);
     TBackProp.BitmapToLayer(AutoFitBitmap, TrainingPairs.InputPatterns[0]);
     AutoFitBitmap.Free;
     AutoFitBitmap := GetAutoFitBitmap(imgOrgSymbol.Picture.Bitmap,
                                BackProp.TargetPatternWidth,
                                BackProp.TargetPatternHeight);
     TBackProp.BitmapToLayer(AutoFitBitmap, TrainingPairs.TargetPattern);
     AutoFitBitmap.Free;
     TrainingPairs.TargetPatternClass := cbSymbol.Text;
     BackProp.AddTrainingPairs(TrainingPairs);
    end;
   end;
   UpdateForm;
   UpdateTrainingInfo;
   Screen.Cursor := crDefault;
end;
procedure TTrainingForm.btnClearClick(Sender: TObject);
begin
  imgOrgSymbol.Canvas.FillRect(Rect(0, 0, imgOrgSymbol.Width,
                              imgOrgSymbol.Height));
  imgNormalSymbol.Canvas.FillRect(Rect(0, 0, imgNormalSymbol.Width,
                                 imgNormalSymbol.Height));
end;
procedure TTrainingForm.btnCloseClick(Sender: TObject);
begin
  Close;
end;
procedure TTrainingForm.btnOptionsClick(Sender: TObject);
begin
  OptionsForm.ShowModal;
  UpdateForm;
end;
procedure TTrainingForm.btnRetrainClick(Sender: TObject);
begin
  UpdateForm(True);
```

```
  BackProp.Retrain;
  UpdateForm(False);
end;
procedure TTrainingForm.btnStopClick(Sender: TObject);
begin
  BackProp.StopTraining := True;
end;
procedure TTrainingForm.btnTestClick(Sender: TObject);
var
  Bitmap: TBitmap;
  i : Integer;
  Symbols, Matchs, Unmatchs: TStringList;
begin
  Bitmap := GetAutoFitBitmap(imgOrgSymbol.Picture.Bitmap,
                    BackProp.InputPatternWidth,
                    BackProp.InputPatternHeight);
  imgNormalSymbol.Canvas.Draw(0, 0, Bitmap);
  TBackProp.BitmapToLayer(Bitmap, BackProp.InputLayer);
  Bitmap.Free;
  BackProp.Apply;
  Symbols := TStringList.Create;
  Matchs := TStringList.Create;
  Unmatchs := TStringList.Create;
  BackProp.GetResult(Symbols, Matchs, Unmatchs);
  for i := sgResult.Cols[0].Count - 1 downto 1 do
   sgResult.Rows[i].Clear;
  sgResult.RowCount := Symbols.Count + 1;
  sgResult.Cols[0].AddStrings(Symbols);
  sgResult.Cols[1].AddStrings(Matchs);
  sgResult.Cols[2].AddStrings(Unmatchs);
  Unmatchs.Free;
  Matchs.Free;
  Symbols.Free;
end;

procedure TTrainingForm.btnTrainClick(Sender: TObject);
begin
  UpdateForm(True);
  BackProp.Train;
  UpdateForm(False);
end;
procedure TTrainingForm.FormCreate(Sender: TObject);
var
  i: Integer;
```

```
begin
  OptionsForm := TOptionsForm.Create(TrainingForm);
  lbFonts.Items := Screen.Fonts;
  for i := 33 to 126 do
    lbChars.Items.Add(Chr(i));
  for i := 6 to 50 do lbSizes.Items.Add(IntToStr(i));
  UpdateTrainingInfo;
  imgOrgSymbol.Canvas.Pen.Width := DEFAULT_DRAWING_WIDTH;
  btnClearClick(Sender);

  BackProp.OnTraining := BackPropTraining;
  BackProp.OnTrainingFinish := BackPropTrainingFinish;

  sgResult.Rows[0].Add('Symbol');
  sgResult.Rows[0].Add('Match');
  sgResult.Rows[0].Add('Unmatch');
end;
procedure TTrainingForm.FormShow(Sender: TObject);
var
  i: Integer;
begin
  lbFonts.Selected[0] := True;
  lbChars.Selected[0] := True;
  lbSizes.Selected[14] := True;
  lbCharsClick(Sender);

  cbSymbol.Clear;
  for i := 0 to BackProp.NTrainingPair - 1 do
    cbSymbol.Items.Add(BackProp.KnownSymbol[i]);

  UpdateForm;
  UpdateTrainingInfo;
  rbInputPattern2.Checked:=tRUE;
  btnClearClick(Sender);
end;
procedure TTrainingForm.imgOrgSymbolMouseDown(
        Sender: TObject; Button: TMouseButton;
        Shift: TShiftState; X, Y: Integer);
begin
  if (Button = mbLeft) then
  begin
    FDrawing := True;
    rbInputPattern2.Checked := True;
    imgOrgSymbol.Canvas.MoveTo(X, Y);
```

```
    imgOrgSymbolMouseMove(Sender, Shift, X, Y);
  end;
end;
procedure TTrainingForm.imgOrgSymbolMouseMove(
        Sender: TObject; Shift: TShiftState; X, Y: Integer);
var
  dPos: Integer;
begin
  if FDrawing then
    with imgOrgSymbol do
      if sbPencil.Down then
        Canvas.LineTo(X, Y)
      else if sbEraser.Down then
      begin
        dPos := (Canvas.Pen.Width div 2) + (Canvas.Pen.Width mod 2);
        Canvas.FillRect(Rect(X - dPos, Y - dPos, X + dPos, Y + dPos));
      end;
end;
procedure TTrainingForm.imgOrgSymbolMouseUp(
        Sender: TObject; Button: TMouseButton;
        Shift: TShiftState; X, Y: Integer);
begin
  FDrawing := False;
end;
procedure TTrainingForm.lbCharsClick(Sender: TObject);
var
  Bitmap: TBitmap;
begin
  btnClearClick(Sender);
  with imgOrgSymbol.Canvas do
  begin
    Font.Name := lbFonts.Items.Strings[lbFonts.ItemIndex];
    Font.Size := StrToInt(lbSizes.Items.Strings[lbSizes.ItemIndex]);
    TextOut(0, 0, lbChars.Items.Strings[lbChars.ItemIndex]);
  end;
  rbInputPattern1.Checked := True;
  Bitmap := GetAutoFitBitmap(imgOrgSymbol.Picture.Bitmap,
                    BackProp.InputPatternWidth,
                    BackProp.InputPatternHeight);
  imgNormalSymbol.Canvas.Draw(0, 0, Bitmap);
  Bitmap.Free;
end;
procedure TTrainingForm.lbFontsClick(Sender: TObject);
begin
```

```
 lbCharsClick(Sender);
end;
procedure TTrainingForm.mmiCopyClick(Sender: TObject);
begin
 Clipboard.Assign(imgOrgSymbol.Picture);
end;
procedure TTrainingForm.mmiCutClick(Sender: TObject);
begin
 btnClearClick(Sender);
 mmiCopyClick(Sender);
end;
procedure TTrainingForm.mmiPasteClick(Sender: TObject);
var
 Bitmap: TBitmap;
begin
 btnClearClick(Sender);
 if Clipboard.HasFormat(CF_BITMAP) then
 begin
  Bitmap := TBitmap.Create;
  try
   Bitmap.Assign(Clipboard);
   imgOrgSymbol.Canvas.Draw(0, 0, Bitmap);
  finally
   Bitmap.Free;
  end;
 end;
 rbInputPattern2.Checked := True;
end;
procedure TTrainingForm.sgResultDrawCell(Sender: TObject; ACol, ARow: Integer;
                          Rect: TRect; State: TGridDrawState);
const
 ALIGNFLAGS: array [TAlignment] of Integer =
  (DT_LEFT or DT_VCENTER or DT_WORDBREAK or DT_EXPANDTABS or
DT_NOPREFIX,
   DT_RIGHT or DT_VCENTER or DT_WORDBREAK or DT_EXPANDTABS or
DT_NOPREFIX,
   DT_CENTER or DT_VCENTER or DT_WORDBREAK or DT_EXPANDTABS or
DT_NOPREFIX);
var
 Alignment: TAlignment;
 Text: string;
begin
 inherited;
```

```
  Text := sgResult.Cells[ACol, ARow];
  if (ARow < sgResult.RowCount) and (ACol < sgResult.ColCount) then
  begin
   if (ARow = 0) then
   begin
    sgResult.Canvas.Font.Style := [fsBold];
    Alignment := taCenter;
   end
   else if (ACol in [0..2]) then
    Alignment := taCenter
   else
    Alignment := taRightJustify;
   sgResult.Canvas.FillRect(Rect);
   DrawText(sgResult.Canvas.Handle,
        PChar(Text), Length(Text), Rect, ALIGNFLAGS[Alignment]);
  end;
end;
procedure TTrainingForm.UpdateForm(TrainingInProgress: Boolean = False);
begin
  if TrainingInProgress then
   Screen.Cursor := crAppStart
  else
   Screen.Cursor := crDefault;
  btnStop.Enabled := TrainingInProgress;
  btnAdd.Enabled := not TrainingInProgress;
  btnTrain.Enabled := (BackProp.NTrainingPair > 0) and not TrainingInProgress;
  btnRetrain.Enabled := (BackProp.NTrainingPair > 0) and not TrainingInProgress;
  btnTest.Enabled := (BackProp.NTrainingPair > 0) and not TrainingInProgress;
  btnOptions.Enabled := not TrainingInProgress;
  btnClose.Enabled := not TrainingInProgress;
  if not (BackProp.NTrainingPair > 0) and (cbSymbol.Items.Count > 0) then
   cbSymbol.Items.Clear;
end;
procedure TTrainingForm.UpdateTrainingInfo;
begin
  with BackProp do
  begin
   lblCurrentEpoch.Caption := IntToStr(NTrainingEpoch);
   if NTrainingNeuron > 0 then
   begin
    lblTrainingError.Caption := FloatToStr(TrainingError / NTrainingNeuron);
    lblPixelError.Caption :=
     IntToStr(NNeuronError) + '/' + IntToStr(NTrainingNeuron) + ' ' +
     '(' + FloatToStrF((NNeuronError / NTrainingNeuron) * 100,
```

```
                    ffGeneral, 4, 0) + '%)';
    end
    else
    begin
     lblTrainingError.Caption := '0';
     lblPixelError.Caption := '0';
    end;
  end;
end;
procedure TTrainingForm.Button1Click(Sender: TObject);
begin
  If Ole.Execute then
    begin
      BG:=TJpegImage.Create;
      try
        imgOrgSymbol.Picture.LoadFromFile(Ole.FileName);
        BG.Assign(imgOrgSymbol.picture.Graphic),
      except
       beep,
       imgOrgSymbol.Picture :=Nil;
       ShowMessage('Format Grafik Tidak Support !');
      end;
      BG.Free ;
    end;
end;
end.
```

**Lampiran 7 : Form Option**

```
unit OptionsFrm;
interface
uses
 Classes, ComCtrls, Controls, DCRFra, Dialogs, Forms, MainFrm, Spin, SysUtils,
 StdCtrls, Windows;
type
 PageTabIndex = (ptRecognition = 0, ptTraining = 1, ptNeuralNet = 2);
 TOptionsForm = class(TForm)
  pgcOptions: TPageControl;
  TabSheet1: TTabSheet;
  GroupBox1: TGroupBox;
  Label1: TLabel;
  Label2: TLabel;
  speMaxEpochs: TSpinEdit;
  edtTargetSquaredError: TEdit;
  GroupBox2: TGroupBox;
  Label3: TLabel;
  edtLearningRate: TEdit;
  Label4: TLabel;
  edtWeightsInitFactor: TEdit;
  btnOK: TButton;
  btnCancel: TButton;
  btnDefault: TButton;
  TabSheet2: TTabSheet;
  GroupBox3: TGroupBox;
  Label5: TLabel;
  Label6: TLabel;
  speInputHeight: TSpinEdit;
  speInputWidth: TSpinEdit;
  GroupBox4: TGroupBox;
  Label7: TLabel;
  Label8: TLabel;
  speTargetHeight: TSpinEdit;
  speTargetWidth: TSpinEdit;
  Label9: TLabel;
  speNHiddenNeuron: TSpinEdit;
  speNOutputNeuron: TSpinEdit;
  Label10: TLabel;
  Label11: TLabel;
  speNInputNeuron: TSpinEdit;
  TabSheet3: TTabSheet;
  speBWThreshold: TSpinEdit;
```

```
    edtLearningRate.Text := FloatToStr(DEFAULT_LEARNING_RATE);
    edtWeightsInitFactor.Text := FloatToStr(DEFAULT_WEIGHTS_INIT_FACTOR);
     FloatToStr(DEFAULT_TARGET_CLASSIFICATION_ERROR);
    edtTargetSquaredError.Text :=
     FloatToStr(DEFAULT_TARGET_SQUARED_ERROR);
    speMaxEpochs.Value := DEFAULT_MAX_EPOCH;
   end;
   Ord(ptNeuralNet):
   begin
    speInputHeight.Value := DEFAULT_INPUT_PATTERN_HEIGHT;
    speInputWidth.Value := DEFAULT_INPUT_PATTERN_WIDTH;
    speTargetHeight.Value := DEFAULT_TARGET_PATTERN_HEIGHT;
    speTargetWidth.Value := DEFAULT_TARGET_PATTERN_WIDTH;
    speNHiddenNeuron.Value := DEFAULT_NUMBER_OF_HIDDEN_NEURON;
   end;
  end;
end;
// Purpose   : Menyimpan pilihan dan menutup form
// Event     : btnOK.OnClick
procedure TOptionsForm.btnOKClick(Sender: TObject);
var
 NewKnowledgeResp: Integer;
begin
 MainForm.DCR.BWThreshold := speBWThreshold.Value;
 MainForm.DCR.NoiseThreshold := speNoiseThreshold.Value;
 MainForm.DCR.SpaceWidth := speSpaceWidth.Value;
 with BackProp do
 begin
  LearningRate := StrToFloat(edtLearningRate.Text);
  WeightsInitFactor := StrToFloat(edtWeightsInitFactor.Text);
  TargetSquaredError := StrToFloat(edtTargetSquaredError.Text);
  MaxEpoch := speMaxEpochs.Value;
  if ((InputPatternHeight <> speInputHeight.Value) or
    (InputPatternWidth <> speInputWidth.Value) or
    (TargetPatternHeight <> speTargetHeight.Value) or
    (TargetPatternWidth <> speTargetWidth.Value) or
    (NHiddenNeuron <> speNHiddenNeuron.Value)) then
  begin
   NewKnowledgeResp := MessageDlg(NEW_KNOWLEDGE,
                mtConfirmation, [mbYes, mbNo], 0);
   case NewKnowledgeResp of
    idYes:
     BackProp.NewKnowledge(speInputHeight.Value, speInputWidth.Value,
              speTargetHeight.Value, speTargetWidth.Value,
```

```
                    speNHiddenNeuron.Value);
      idNo:
        Exit;
     end;
    end;
  end;
  Close;
end;
// Purpose    : Initilizes form
// Event      : OptionsForm.OnShow
procedure TOptionsForm.FormShow(Sender: TObject);
begin
  //--- DCR
  speBWThreshold.Value := MainForm.DCR.BWThreshold;
  speSpaceWidth.Value := MainForm.DCR.SpaceWidth;

  //--- Training parameter
  edtLearningRate.Text := FloatToStrF(BackProp.LearningRate, ffGeneral, 7, 4);
  edtWeightsInitFactor.Text :=
    FloatToStrF(BackProp.WeightsInitFactor, ffGeneral, 7, 4);
  //edtErrorThreshold.Text :=
    FloatToStrF(BackProp.ErrorThreshold, ffGeneral, 7, 4);
  //edtTargetClassificationError.Text :=
    FloatToStrF(BackProp.TargetClassificationError, ffGeneral, 7, 4);
  edtTargetSquaredError.Text :=
    FloatToStrF(BackProp.TargetSquaredError, ffGeneral, 7, 4);
  speMaxEpochs.Value := BackProp.MaxEpoch;
  //--- Network architecture
  speInputHeight.Value := BackProp.InputPatternHeight;
  speInputWidth.Value := BackProp.InputPatternWidth;
  speTargetHeight.Value := BackProp.TargetPatternHeight;
  speTargetWidth.Value := BackProp.TargetPatternWidth;
  speNHiddenNeuron.Value := BackProp.NHiddenNeuron;
end;
// Purpose    : Mengatur nilai dari jumlah input neuron
// Event      : speInputHeight.OnChange
procedure TOptionsForm.speInputHeightChange(Sender: TObject);
begin
  speNInputNeuron.Value := speInputHeight.Value * speInputWidth.Value;
end;

// Purpose    : Mengatur nilai dari jumlah input neuron
// Event      : speInputWidth.OnChange
procedure TOptionsForm.speInputWidthChange(Sender: TObject);
```

```
begin
  speNInputNeuron.Value := speInputHeight.Value * speInputWidth.Value;
end;

// Purpose    : Mengatur nilai dari jumlah input neuron
// Event      : speTargetHeight.OnChange
procedure TOptionsForm.speTargetHeightChange(Sender: TObject);
begin
  speNOutputNeuron.Value := speTargetHeight.Value * speTargetWidth.Value;
end;

// Purpose    : Mengatur nilai dari jumlah input neuron
// Event      : speTargetWidth.OnChange
procedure TOptionsForm.speTargetWidthChange(Sender: TObject);
begin
  speNOutputNeuron.Value := speTargetHeight.Value * speTargetWidth.Value;
end;
end.
```

**Lampiran 8 : BackPropCls**

```
unit BackPropCls;
interface
uses Classes, Graphics;
const
 //- Network architecture-
 DEFAULT_INPUT_PATTERN_HEIGHT: Integer = 20;
 DEFAULT_INPUT_PATTERN_WIDTH = 20;
 DEFAULT_TARGET_PATTERN_HEIGHT = 20;
 DEFAULT_TARGET_PATTERN_WIDTH = 20;
 DEFAULT_NUMBER_OF_HIDDEN_NEURON = 50;
 //- Training set parameters-
 DEFAULT_LEARNING_RATE = 0.001;
 DEFAULT_MAX_EPOCH = 10000;
 DEFAULT_TARGET_CLASSIFICATION_ERROR = -1;
 DEFAULT_TARGET_SQUARED_ERROR = 0.01;
 DEFAULT_ERROR_THRESHOLD = 1;
 DEFAULT_WEIGHTS_INIT_FACTOR = 0.5;
type
 //- Events -
 TTrainingEvent = procedure(Sender: TObject) of object;
 TTrainingFinishEvent = procedure(Sender: TObject) of object;
 //- Neurons and their weights -
 TNeuron = Single;
 TLayer = array of TNeuron;
 TWeight = Single;
 TLayerWeights = array of array of TWeight;
 //- Training set -
 TPatternClass = string;
 TTrainingPairs = record
  InputPatterns: array of TLayer;
  TargetPattern: TLayer;
  TargetPatternClass: TPatternClass;
 end;
 TTrainingSet = array of TTrainingPairs;
type
 TBackProp = class(TObject)
 //--- Construction/Destruction ---
 public
  constructor Create(NInputNeuron: Integer;
                     NHiddenNeuron: Integer; NOutputNeuron: Integer);
                     overload; virtual;
  constructor Create(InputPatternHeight: Integer; InputPatternWidth: Integer;
```

```
                TargetPatternHeight: Integer;
                TargetPatternWidth: Integer; NHiddenNeuron: Integer);
                overload; virtual;
  constructor Create(FileName: string); overload; virtual;
//--- Events ---
private
  FOnTraining: TTrainingEvent;
  FOnTrainingFinish: TTrainingFinishEvent;
public
  property OnTraining: TTrainingEvent read FOnTraining write FOnTraining;
  property OnTrainingFinish: TTrainingFinishEvent
          read FOnTrainingFinish write FOnTrainingFinish;
//--- Properties ---
private
  //- Neurons and their weights -
  FNInputNeuron: Integer;
  FNHiddenNeuron: Integer;
  FNOutputNeuron: Integer;
  FInputLayer: TLayer;
  FHiddenLayer: TLayer;
  FOutputLayer: TLayer;
  FHiddenLayerWeights: TLayerWeights;
  FOutputLayerWeights: TLayerWeights;
  //- Training set and its parameters -
  FTrainingSet: TTrainingSet;
  FErrorThreshold: Single;
  FLearningRate: Single;
  FMaxEpoch: Integer;          // maximum epoch for training stop condition
  FStopTraining: Boolean;
  FTargetClassificationError: Single;          // target error for training
  FTargetSquaredError: Single;                 //          stop condition
  FWeightsInitFactor: Single;          // for random weights initialization
  //- Patterns size
  FInputPatternHeight: Integer;
  FInputPatternWidth: Integer;
  FTargetPatternHeight: Integer;
  FTargetPatternWidth: Integer;
  //- Training info -
  FNNeuronError: Integer;
  FNTrainingEpoch: Integer;
  FNTrainingNeuron: Integer;
  FTrainingError: Single;
  //- Others
  FModified: Boolean;                  // indicates whether the weights or
```

```
                                // training set have been modified
  //- Properties procedure/functions
  function GetKnownSymbol(Index: Integer): TPatternClass; virtual;
  function GetNTrainingPair: Integer; virtual;
  function GetOutputLayer: TLayer; virtual;
  procedure SetLearningRate(Value: Single); virtual;
  procedure SetMaxEpoch(Value: Integer); virtual;
  procedure SetWeightsInitFactor(Value: Single); virtual;
public
  //- Backpropagation layers -
  property InputLayer: TLayer read FInputLayer write FInputLayer;
  property OutputLayer: TLayer read GetOutputLayer;
  property NInputNeuron: Integer read FNInputNeuron;
  property NHiddenNeuron: Integer read FNHiddenNeuron;
  property NOutputNeuron: Integer read FNOutputNeuron;
  //- Training set parameters -
  property ErrorThreshold: Single read FErrorThreshold write FErrorThreshold;
  property LearningRate: Single read FLearningRate write SetLearningRate;
  property MaxEpoch: Integer read FMaxEpoch write SetMaxEpoch;
  property StopTraining: Boolean read FStopTraining write FStopTraining;
  property TargetClassificationError: Single read FTargetClassificationError
                                write FTargetClassificationError;
  property TargetSquaredError: Single read FTargetSquaredError
                                write FTargetSquaredError;
  property WeightsInitFactor: Single read FWeightsInitFactor
                                write SetWeightsInitFactor;
  //- Patterns size -
  property InputPatternHeight: Integer read FInputPatternHeight;
  property InputPatternWidth: Integer read FInputPatternWidth;
  property TargetPatternHeight: Integer read FTargetPatternHeight;
  property TargetPatternWidth: Integer read FTargetPatternWidth;

  //- Training info -
  property KnownSymbol[Index: Integer]: TPatternClass read GetKnownSymbol;
  property NNeuronError: Integer read FNNeuronError;
  property NTrainingEpoch: Integer read FNTrainingEpoch;
  property NTrainingNeuron: Integer read FNTrainingNeuron;
  property NTrainingPair: Integer read GetNTrainingPair;
  property TrainingError: Single read FTrainingError;

  //- Others -
  property Modified: Boolean read FModified;

//--- Methods ---
```

```
  private
   function BipolarSigmoid(x: Single): Single;
   function BipolarSigmoidDerivation(Fx: Single): Single; overload;
   procedure InitLayers; virtual;
   procedure InitWeights; virtual;
   procedure UpdateNTrainingNeuron; virtual;

  public
   //- Training -
   procedure AddTrainingPairs(TrainingPairs: TTrainingPairs); virtual;
   procedure Retrain; virtual;
   procedure Train; virtual;

   //- Testing -
   procedure Apply; virtual;
   procedure GetResult(out Symbol: TPatternClass); overload, virtual;
   procedure GetResult(var Symbols: TStringList;
                       var Matchs: TStringList;
                       var Unmatchs: TStringList); overload; virtual;

   //- Knowledge -
   procedure NewKnowledge; overload; virtual;
   procedure NewKnowledge(
             InputPatternHeight: Integer; InputPatternWidth: Integer;
             TargetPatternHeight: Integer; TargetPatternWidth: Integer;
             NHiddenNeuron: Integer); overload; virtual;
   procedure OpenKnowledge(FileName: string); virtual;
   procedure SaveKnowledge(FileName: string); virtual;

  //--- Class Methods ---
  public
   class procedure BitmapToLayer(Bitmap: TBitmap; Layer: TLayer);
  end;

implementation

uses Forms, SysUtils;

//----------------------------
// Construction/Destruction
//----------------------------

// Purpose    : Creates and initializes a new backpropagation object by
//              defining the size of input layer, hidden layer and output layer
```

```
begin
  Assert(FileExists(FileName));

  inherited Create;
  OpenKnowledge(FileName);
end;

// Purpose    : Creates and initializes a new backpropagation object by
//              defining the size of input pattern, output pattern and hidden
//              layer
// Inputs     : * InputPatternHeight (>0)
//              * InputPatternWidth (>0)
//              * TargetPatternHeight (>0)
//              * TargetPatternWidth (>0)
//              * NHiddenNeuron (number of neurons, not including bias, in
//                      hidden layer; >0)
constructor TBackProp.Create(
          InputPatternHeight: Integer; InputPatternWidth: Integer;
          TargetPatternHeight: Integer; TargetPatternWidth: Integer;
          NHiddenNeuron: Integer);
begin
  Assert((InputPatternHeight > 0) and (InputPatternWidth > 0) and
      (TargetPatternHeight > 0) and (TargetPatternWidth > 0) and
      (NHiddenNeuron > 0));

  Create(InputPatternHeight * InputPatternWidth, NHiddenNeuron,
      TargetPatternHeight * TargetPatternWidth);
  FInputPatternHeight := InputPatternHeight;
  FInputPatternWidth := InputPatternWidth;
  FTargetPatternHeight := TargetPatternHeight;
  FTargetPatternWidth := TargetPatternWidth;
end;

//--------------
//  Properties
//--------------

// Purpose    : Returns the known symbol with certain index
// Input      : Index (0 <= Index <= number of symbol - 1)
function TBackProp.GetKnownSymbol(Index: Integer): TPatternClass;
begin
  Assert((0 <= Index) and (Index <= High(FTrainingSet)));

  Result := FTrainingSet[Index].TargetPatternClass;
```

```
end;

// Purpose    : Returns the number of training pairs (precisely the number of
//              known symbol)
function TBackProp.GetNTrainingPair: Integer;
begin
  Result := Length(FTrainingSet);
end;

// Purpose    : Returns the output layer without the bias neuron
function TBackProp.GetOutputLayer: TLayer;
var
  k: Integer;                       // looping index for output layer
begin
  SetLength(Result, FNOutputNeuron);
  for k := 1 to FNOutputNeuron do
    Result[k - 1] := FOutputLayer[k];
end;

// Purpose    : Sets the learning rate
// Input      : Value (0 < Value <= 1)
procedure TBackProp.SetLearningRate(Value: Single);
begin
  Assert((0 < Value) and (Value <= 1));

  FLearningRate := Value;
end;

// Purpose    : Sets the maximum number of epoch for the current training (not
//              for overall training)
// Input      : Value (>0)
procedure TBackProp.SetMaxEpoch(Value: Integer);
begin
  Assert(Value > 0);

  FMaxEpoch := Value;
end;

// Purpose    : Sets the weights initialize factor
// Input      : Value (>0)
procedure TBackProp.SetWeightsInitFactor(Value: Single);
begin
  Assert(Value > 0);
```

```
  FWeightsInitFactor := Value;
end;

//----------
// Methods
//----------

// Purpose    : Adds training pairs to the training set
// Input      : TrainingPairs (the training pairs that will be added; must has
//                             the same size with the input and output layer)
// Effect     : Property HasTrainingSet is set to true
procedure TBackProp.AddTrainingPairs(TrainingPairs: TTrainingPairs);
var
  i, k, l, m: Integer;      // looping index for input pattern, target pattern,
                            //      training set and training input patterns
  IxClass: Integer;                     // index of target pattern class
begin
  for i := 0 to High(TrainingPairs.InputPatterns) do
    Assert(Length(TrainingPairs.InputPatterns[i]) = FNInputNeuron + 1);
  Assert(Length(TrainingPairs.TargetPattern) = FNOutputNeuron + 1);

  //--- Initializes a new training pair slot in the training set
  //- Searches for the training pairs' class in the training set
  IxClass := -1;
  for l := 0 to High(FTrainingSet) do
    if FTrainingSet[l].TargetPatternClass =
       TrainingPairs.TargetPatternClass then
    begin
      IxClass := l;
      SetLength(FTrainingSet[IxClass].InputPatterns,
            Length(FTrainingSet[IxClass].InputPatterns) +
              Length(TrainingPairs.InputPatterns), FNInputNeuron + 1);
      Break;
    end;
  //- If the training pair's class doesn't exist in the training set, create
  //  a new class in the training set
  if IxClass = -1 then
  begin
    SetLength(FTrainingSet, Length(FTrainingSet) + 1);
    IxClass := High(FTrainingSet);
    SetLength(FTrainingSet[IxClass].InputPatterns,
          Length(TrainingPairs.InputPatterns), FNInputNeuron + 1);
    SetLength(FTrainingSet[IxClass].TargetPattern, FNOutputNeuron + 1);
  end;
```

```
  //-- Adds the training pair to the training set
  with FTrainingSet[IxClass] do
  begin
   for m := 0 to High(TrainingPairs.InputPatterns) do
    for i := 1 to High(TrainingPairs.InputPatterns[m]) do
     InputPatterns[Length(InputPatterns) -
                   Length(TrainingPairs.InputPatterns) + m,
                  i] := TrainingPairs.InputPatterns[m, i];
   if Length(FTrainingSet[IxClass].InputPatterns) =
     Length(TrainingPairs.InputPatterns) then
   begin
    for k := 1 to High(TrainingPairs.TargetPattern) do
     TargetPattern[k] := TrainingPairs.TargetPattern[k];
    TargetPatternClass := TrainingPairs.TargetPatternClass;
   end;
  end;

  //-- Finishing
  UpdateNTrainingNeuron;
  FModified := True;
 end;

 // Purpose   : Executes the feedforward phase of the backpropagation
 // Assumption : Input layer activation value has been set
 // Effect    : A continuous activation value for output layer has been computed
 procedure TBackProp.Apply;
 var
  i, j, k: Integer;        // looping index for input, hidden and output layer
 begin
  //--- Feedforwards from input layer to hidden layer
  for j := 1 to FNHiddenNeuron do
  begin
   //- Sums its weighted input signals from input layer
   FHiddenLayer[j] := 0;
   for i := 0 to FNInputNeuron do
    FHiddenLayer[j] := FHiddenLayer[j] +
                  (FInputLayer[i] * FHiddenLayerWeights[j, i]);
   //- Applies the bipolar sigmoid activation function
   FHiddenLayer[j] := BipolarSigmoid(FHiddenLayer[j]);
  end;

  //--- Feedforwards from hidden layer to output layer
  for k := 1 to FNOutputNeuron do
  begin
```

```
    //- Sums its weighted input signals from hidden layer
    FOutputLayer[k] := 0;
    for j := 0 to FNHiddenNeuron do
      FOutputLayer[k] := FOutputLayer[k] +
                  (FHiddenLayer[j] * FOutputLayerWeights[k, j]);
    //- Applies the bipolar sigmoid activation function
    FOutputLayer[k] := BipolarSigmoid(FOutputLayer[k]);
  end;
end;

// Purpose    : Gets the symbol with higest percentage of match with the pattern
//              in output layer
// Assumption : Output layer activation value has been computed
procedure TBackProp.GetResult(out Symbol: TPatternClass);
var
  Symbols, Matchs, Unmatchs: TStringList;
begin
  Symbols := TStringList.Create;
  Matchs := TStringList.Create;
  Unmatchs := TStringList.Create;
  GetResult(Symbols, Matchs, Unmatchs);
  if Symbols.Count > 0 then
    Symbol := Symbols.Strings[0]
  else
    Symbol := '';
  Unmatchs.Free;
  Matchs.Free;
  Symbols.Free;
end;

// Purpose    : Gets all symbols with their percentage of match and unmatch
//              information with the pattern in output layer
// Assumption : Output layer activation value has been computed
procedure TBackProp.GetResult(var Symbols: TStringList; var Matchs: TStringList;
                    var Unmatchs: TStringList);
var
  HammingDistance: Integer;
  k, l: Integer;  // looping index for training input patterns and training set
  MatchPercentage: Single;
  Pos: Integer;                 // position to insert the processed symbol
begin
  for l := 0 to High(FTrainingSet) do
  begin
    //--- Gets the hamming distance
```

```
    HammingDistance := 0;
    for k := 1 to FNOutputNeuron do
     if Abs(FTrainingSet[l].TargetPattern[k] - FOutputLayer[k])
       >= FErrorThreshold then
       HammingDistance := HammingDistance + 1;

    //-- Counts the match percentage
    MatchPercentage := (1 - (HammingDistance / FNOutputNeuron)) * 100;

    //--- Inserts to the output variables
    Pos := 0;
    while (Pos < l) do
     if (StrToFloat(Matchs[Pos]) < MatchPercentage) then
       Break
     else
       Pos := Pos + 1;
    Symbols.Insert(Pos, FTrainingSet[l].TargetPatternClass);
    Matchs.Insert(Pos, FloatToStrF(MatchPercentage, ffFixed, 4, 2));
    Unmatchs.Insert(Pos, IntToStr(HammingDistance) + '/' +
                    IntToStr(FNOutputNeuron) + ' ');
  end;
  for Pos := 0 to Matchs.Count - 1 do
    Matchs[Pos] := Matchs[Pos] + '%'
end;

// Purpose    : Creates a new knowledge and maintains the network architecture
// Effects    : * The object has no training set
//              * The training info has been reset
//              * The object is not modified
procedure TBackProp.NewKnowledge;
begin
  //--- New (no) training set
  SetLength(FTrainingSet, 0);

  //--- New (init) the training info
  FNNeuronError := 0;
  FNTrainingEpoch := 0;
  FTrainingError := 0;
  FNTrainingNeuron := 0;

  FModified := False;
end;

// Purpose    : Creates a new knowledge and new network architecture
```

```
// Inputs    : * InputPatternHeight (>0)
//             * InputPatternWidth (>0)
//             * TargetPatternHeight (>0)
//             * TargetPatternWidth (>0)
//             * NHiddenNeuron (>0)
// Effects   : * The object has a new network architecture
//             * The weights have been initialized
//             * The object has no training set
//             * The training info has been reset
//             * The object is not modified
procedure TBackProp.NewKnowledge(
        InputPatternHeight: Integer; InputPatternWidth: Integer;
        TargetPatternHeight: Integer; TargetPatternWidth: Integer;
        NHiddenNeuron: Integer);
begin
 Assert((InputPatternHeight > 0) and (InputPatternWidth > 0) and
     (TargetPatternHeight > 0) and (TargetPatternWidth > 0) and
     (NHiddenNeuron > 0));

 //--- New network architecture
 FInputPatternHeight := InputPatternHeight;
 FInputPatternWidth := InputPatternWidth;
 FTargetPatternHeight := TargetPatternHeight;
 FTargetPatternWidth := TargetPatternWidth;
 FNHiddenNeuron := NHiddenNeuron;
 FNInputNeuron := FInputPatternHeight * FInputPatternWidth;
 FNOutputNeuron := FTargetPatternHeight * FTargetPatternWidth;
 InitLayers;

 //--- New (init) weights
 InitWeights;

 NewKnowledge;
end;

// Purpose   : Opens a new knowledge from external file
// Input     : FileName (must exist)
procedure TBackProp.OpenKnowledge(FileName: string);
var
 DataSize: Integer;                          // temporary variable
 FileStream: TFileStream;
 i, j, k, l, m, n: Integer;    // looping index for input layer, hidden layer,
                        //              output layer, training set,
                        //         training input patterns and other
```

```
begin
 Assert(FileExists(FileName));

 FileStream := TFileStream.Create(FileName, fmOpenRead or fmShareDenyWrite);
 with FileStream do
 begin
  try
   //--- Opens the network architecture
   Read(FInputPatternHeight, SizeOf(Integer));
   Read(FInputPatternWidth, SizeOf(Integer));
   Read(FTargetPatternHeight, SizeOf(Integer));
   Read(FTargetPatternWidth, SizeOf(Integer));
   Read(FNHiddenNeuron, SizeOf(Integer));
   FNInputNeuron := FInputPatternHeight * FInputPatternWidth;
   FNOutputNeuron := FTargetPatternHeight * FTargetPatternWidth;
   InitLayers;

   //--- Opens the weights
   for j := 1 to FNHiddenNeuron do
    for i := 0 to FNInputNeuron do
     Read(FHiddenLayerWeights[j, i], SizeOf(TWeight));
   for k := 1 to FNOutputNeuron do
    for j := 0 to FNHiddenNeuron do
     Read(FOutputLayerWeights[k, j], SizeOf(TWeight));

   //--- Opens the training set
   Read(DataSize, SizeOf(Integer));
   SetLength(FTrainingSet, DataSize);
   for l := 0 to High(FTrainingSet) do
   begin
    Read(DataSize, SizeOf(Integer));
    SetLength(FTrainingSet[l].InputPatterns, DataSize, FNInputNeuron + 1);
    for m := 0 to High(FTrainingSet[l].InputPatterns) do
     for i := 1 to High(FTrainingSet[l].InputPatterns[m]) do
      Read(FTrainingSet[l].InputPatterns[m, i], SizeOf(TNeuron));
    SetLength(FTrainingSet[l].TargetPattern, FNOutputNeuron + 1);
    for k := 1 to High(FTrainingSet[l].TargetPattern) do
     Read(FTrainingSet[l].TargetPattern[k], SizeOf(TNeuron));
    Read(DataSize, SizeOf(Integer));
    SetLength(FTrainingSet[l].TargetPatternClass, DataSize);
    for n := 1 to DataSize do
     Read(FTrainingSet[l].TargetPatternClass[n], SizeOf(Char));
   end;
```

```
with FileStream do
 try
  //-- Saves the network architecture
  Write(FInputPatternHeight, SizeOf(Integer));
  Write(FInputPatternWidth, SizeOf(Integer));
  Write(FTargetPatternHeight, SizeOf(Integer));
  Write(FTargetPatternWidth, SizeOf(Integer));
  Write(FNHiddenNeuron, SizeOf(Integer));

  //--- Saves the weights
  for j := 1 to FNHiddenNeuron do
   for i := 0 to FNInputNeuron do
    Write(FHiddenLayerWeights[j, i], SizeOf(TWeight));
  for k := 1 to FNOutputNeuron do
   for j := 0 to FNHiddenNeuron do
    Write(FOutputLayerWeights[k, j], SizeOf(TWeight));

  //--- Saves the training set
  DataSize := Length(FTrainingSet);
  Write(DataSize, SizeOf(Integer));
  for l := 0 to High(FTrainingSet) do
  begin
   DataSize := Length(FTrainingSet[l].InputPatterns);
   Write(DataSize, SizeOf(Integer));
   for m := 0 to High(FTrainingSet[l].InputPatterns) do
    for i := 1 to High(FTrainingSet[l].InputPatterns[m]) do
     Write(FTrainingSet[l].InputPatterns[m, i], SizeOf(TNeuron));
   for k := 1 to High(FTrainingSet[l].TargetPattern) do
    Write(FTrainingSet[l].TargetPattern[k], SizeOf(TNeuron));
   DataSize := Length(FTrainingSet[l].TargetPatternClass);
   Write(DataSize, SizeOf(Integer));
   for n := 1 to DataSize do
    Write(FTrainingSet[l].TargetPatternClass[n], SizeOf(Char));
  end;

  //--- Saves the training parameters
  Write(FLearningRate, SizeOf(Single));
  Write(FWeightsInitFactor, SizeOf(Single));
  Write(FErrorThreshold, SizeOf(Single));
  Write(FTargetClassificationError, SizeOf(Single));
  Write(FTargetSquaredError, SizeOf(Single));
  Write(FMaxEpoch, SizeOf(Integer));

  //--- Saves the training info
```

```
    Write(FNNeuronError, SizeOf(Integer));
    Write(FNTrainingEpoch, SizeOf(Integer));
    Write(FTrainingError, SizeOf(Single));
   finally
    Free;
   end;
  FModified := False;
end;

// Purpose    : Trains the backpropagation neural net without weight
//              initialization process
// Assumptions: * There is at least one training pairs in the training set
//              * The training parameters have been set
procedure TBackProp.Train;
label
  Finish;
const
  INFINITE_ERROR = 2;
type
  TErrors = array of single;
  TWeightCorrection = array of array of Single;
var
  //--- Looping index for input layer, hidden layer, output layer, training set
  //    and training input patterns
  i, j, k, l, m: Integer;

  //--- Weights correction with momentum
  HiddenErrors: TErrors;
  HiddenWeightsCorrection: TWeightCorrection;
  OutputErrors: TErrors;
  OutputWeightsCorrection: TWeightCorrection;

  //--- Training stop condition
  Error: Single;
  StartEpoch: Integer;
begin
  Assert(Length(FTrainingSet) > 0);

  //--- Initializing
  SetLength(OutputErrors, FNOutputNeuron + 1);
  SetLength(HiddenErrors, FNHiddenNeuron + 1);
  SetLength(OutputWeightsCorrection, FNOutputNeuron + 1, FNHiddenNeuron + 1);
  SetLength(HiddenWeightsCorrection, FNHiddenNeuron - 1, FNInputNeuron + 1);
```

```
//--- Repeats the training until number of epoch is greater or equal than
//   maximum number of epoch defined in property MaxEpcoh or the average
//   error is less or equal than the target error minimum defined in property
//   MinError or the user stop the training
StartEpoch := FNTrainingEpoch,
FStopTraining := False;
repeat
 FNTrainingEpoch := FNTrainingEpoch + 1;
 FTrainingError := 0;
 FNNeuronError := 0,

 //--- Repeats for all training pair in the training set
 for l := 0 to High(FTrainingSet) do
  for m := 0 to High(FTrainingSet[l].InputPatterns) do
  begin
   //--- Sets input layer activation value
   for i := 1 to High(FTrainingSet[l].InputPatterns[m]) do
    FInputLayer[i] := FTrainingSet[l].InputPatterns[m, i];

   //--- The feedforward phase
   Apply;

   //--- The backpropagation of error phase
   //- Computes output layer weights error information
   for k := 1 to FNOutputNeuron do
   begin
    Error := FTrainingSet[l].TargetPattern[k] - FOutputLayer[k];
    if Abs(Error) >= FErrorThreshold then
     FNNeuronError := FNNeuronError - 1;
    FTrainingError := FTrainingError + (Error * Error);
    OutputErrors[k] := Error * BipolarSigmoidDerivation(FOutputLayer[k]);
    for j := 0 to FNHiddenNeuron do
     OutputWeightsCorrection[k, j] :=
      FLearningRate * OutputErrors[k] * FHiddenLayer[j];
   end;
   //- Computes hidden layer weights error information
   for j := 1 to FNHiddenNeuron do
   begin
    HiddenErrors[j] := 0;
    for k := 1 to FNOutputNeuron do
     HiddenErrors[j] := HiddenErrors[j] +
                (OutputErrors[k] * FOutputLayerWeights[k, j]);
    HiddenErrors[j] := HiddenErrors[j] *
                BipolarSigmoidDerivation(FHiddenLayer[j]);
```

```
        for i := 0 to FNInputNeuron do
          HiddenWeightsCorrection[j, i] :=
            FLearningRate * HiddenErrors[j] * FInputLayer[i]
      end;
      //- Updates output layer weights and bias
      for k := 1 to FNOutputNeuron do
        for j := 0 to FNHiddenNeuron do
          FOutputLayerWeights[k, j] := FOutputLayerWeights[k, j] +
                                       OutputWeightsCorrection[k, j];
      //- Updates hidden layer weights and bias
      for j := 1 to FNHiddenNeuron do
        for i := 0 to FNInputNeuron do
          FHiddenLayerWeights[j, i] := FHiddenLayerWeights[j, i] +
                                       HiddenWeightsCorrection[j, i];
      Application.ProcessMessages;
    end;
    if FStopTraining then goto Finish;
    if Assigned(OnTraining) then OnTraining(Self);
  until (FNNeuronError <= FTargetClassificationError) or
      (FTrainingError / FNTrainingNeuron <= FTargetSquaredError) or
      (FNTrainingEpoch - StartEpoch >= FMaxEpoch);

Finish:
  FModified := True;
  if Assigned(OnTrainingFinish) then OnTrainingFinish(Self);
end;

//----------------
//  Class Methods
//----------------

// Purpose    : Converts pattern in bitmap format to input layer
// Inputs     : * Bitmap
//              * Layer (layer's size = #pixel in Bitmap + 1)
class procedure TBackProp.BitmapToLayer(Bitmap: TBitmap; Layer: TLayer);
var
  i: Integer;                          // index for input layer
  x, y: Integer;                       // x and y coordinate in the bitmap
begin
  Assert(Length(Layer) = (Bitmap.Width * Bitmap.Height) + 1);

  i := 1;
  for y := 0 to Bitmap.Height - 1 do
    for x := 0 to Bitmap.Width - 1 do
```

```
    begin
      if Bitmap.Canvas.Pixels[x, y] = clBlack then
        Layer[i] := 1
      else
        Layer[i] := -1;
      i := i + 1;
    end;
  end;

//-----------------------------------
// Private Functions and Procedures
//-----------------------------------

// Purpose    : Returns the bipolar sigmoid function value of a x
function TBackProp.BipolarSigmoid(x: Single): Single;
begin
  try
    Result := (2 / (1 + Exp(-x))) - 1;
  except
    on EOverflow do Result := 0;
  end;
end;

// Purpose    : Returns the derivation of bipolar sigmoid function value using
//              its function value Fx
function TBackProp.BipolarSigmoidDerivation(Fx: Single): Single;
begin
  Result := ((1 + Fx) * (1 - Fx)) / 2;
end;

procedure TBackProp.InitLayers;
begin
  SetLength(FInputLayer, FNInputNeuron + 1);
  SetLength(FHiddenLayer, FNHiddenNeuron + 1);
  SetLength(FOutputLayer, FNOutputNeuron + 1);
  FInputLayer[0] := 1;                          // input layer's bias
  FHiddenLayer[0] := 1;                         // hidden layer's bias
  SetLength(FHiddenLayerWeights, FNHiddenNeuron + 1, FNInputNeuron + 1);
  SetLength(FOutputLayerWeights, FNOutputNeuron + 1, FNHiddenNeuron + 1);
end;

// Purpose    : Initilizes the backpropagation weights with random value
// Assumption : Parameter for weights random initialization has been set
procedure TBackProp.InitWeights;
```

```
var
 i, j, k: Integer;        // looping index for input, hidden and output layer
begin
 Randomize;
 //--- Initializes the hidden layer weights
 for j := 0 to FNHiddenNeuron do
  for i := 0 to FNInputNeuron do
   FHiddenLayerWeights[j, i] := (Random - 0.5) * (FWeightsInitFactor * 2);

 //--- Initializes the output layer weights
 for k := 1 to FNOutputNeuron do
  for j := 0 to FNHiddenNeuron do
   FOutputLayerWeights[k, j] := (Random - 0.5) * (FWeightsInitFactor * 2);
end;

// Purpose   : Updates the number of training neurons
procedure TBackProp.UpdateNTrainingNeuron;
var
 l: Integer;
begin
 FNTrainingNeuron := 0;
 for l := 0 to High(FTrainingSet) do
  FNTrainingNeuron :=
   FNTrainingNeuron +
   (Length(FTrainingSet[l].InputPatterns) * FNOutputNeuron);
end;
end.
```

**Lampiran 9 : Form .DCR Frame**

```
unit DCRFra;
interface
uses
  Classes, Controls, ExtCtrls, Forms, Windows, Graphics;
const
  DEFAULT_BW_THRESHOLD = 196;
  DEFAULT_NOISE_THRESHOLD = 10;
  DEFAULT_SPACE_WIDTH = 22;
type
  //- Text Segmentasi Karakter -
  TImageArray = array of array of Byte;
  TChar = record
    Pixels: array of TPoint;
    MinPixel: TPoint;
    MaxPixel: TPoint;
    Noise: Boolean;
    CharCode: Char;
  end;
  TWord = record
    Left: Integer;
    Right: Integer;
    Chars: array of TChar;
  end;
  TLine = record
    Top: Integer;
    Bottom: Integer;
    Words: array of TWord;
  end;
  TLines = array of TLine;
  //- Drawing tool -
  TDrawingTool = (dtPencil, dtEraser);
type
  TDCRFrame = class(TFrame)
    Image: TImage;
    pnlTitle: TPanel;
    procedure ImageMouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure ImageMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure ImageMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
```

```
    FDrawing: Boolean;
    FDrawingTool: TDrawingTool;
    FImageArray: TImageArray;
    FLines: TLines;
    FModified: Boolean;
    FResultText: string;
    FBWThreshold: Integer;
    FNoiseThreshold: Integer;
    FSpaceWidth: Integer;
    procedure FilterBnW;
    procedure SegmentLines;
    procedure SegmentWords;
    procedure SegmentChars;
    procedure GetRGBColor(Color: TColor;
                out R: Integer; out G: Integer; out B: Integer);
  public
    property BWThreshold: Integer read FBWThreshold write FBWThreshold;
    property DrawingTool: TDrawingTool read FDrawingTool write FDrawingTool;
    property NoiseThreshold: Integer read FNoiseThreshold write FNoiseThreshold;
    property Modified: Boolean read FModified;
    property SpaceWidth: Integer read FSpaceWidth write FSpaceWidth;
    property ResultText: string read FResultText;
    procedure Clear;
    procedure Init;
    procedure OpenPicture(FileName: string);
    procedure Recognize;
    procedure SavePicture(FileName: string);
    procedure ShowHideTitle;
  end;
implementation
uses
  SysUtils,
  GlobalMdl, BackPropCls;
{$R *.dfm}
// Purpose   : Deletes all drawing on the text picture area
procedure TDCRFrame.Clear;
begin
  Image.Canvas.FillRect(Rect(0, 0, Image.Width, Image.Height));
  Image.Picture.Bitmap.Width := ClientWidth ;
  Image.Picture.Bitmap.Height := ClientHeight - pnlTitle.Height;
  FModified := False;
end;

// Purpose   : Melakukan langkah preprocessing, yaitu Filter hitam & putih,
```

```
// Event     : DCRFrame.Image.OnMouseMove
procedure TDCRFrame.ImageMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
var
  dPos: Integer;
begin
  if FDrawing then
    with Image do
      if DrawingTool = dtPencil then
        Canvas.LineTo(X, Y)
      else if DrawingTool = dtEraser then
      begin
        dPos := (Canvas.Pen.Width div 2) + (Canvas.Pen.Width mod 2);
        Canvas.FillRect(Rect(X - dPos, Y - dPos, X + dPos, Y + dPos));
      end;
end;
// Purpose   : menggambar pada bidang gambar teks
// Event     : DCRFrame.Image.OnMouseUp
procedure TDCRFrame.ImageMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  FDrawing := False;
end;
// Purpose   : Membuka gambar dari file eksternal
// Input     : FileName (must exist)
procedure TDCRFrame.OpenPicture(FileName: string);
begin
  Assert(FileExists(FileName));
  Image.Picture.LoadFromFile(FileName);
  Image.Canvas.Pen.Width := DEFAULT_DRAWING_WIDTH;
  FModified := False;
end;
// Purpose   : Recognize gambar pada area gambar teks
procedure TDCRFrame.Recognize;
var
  IxLine, IxWord, IxChar: Integer;
  i: Integer;
  OrgBitmap: TBitmap;
  StretchBitmap: TBitmap;
  ResultSymbol: TPatternClass;
begin
  Screen.Cursor := crHourglass;
  //--- Preprocess
  FilterBnW;
```

```
  //--- Text segmentation
  SegmentLines;
  SegmentWords;
  SegmentChars;
  //--- Recognition and reconstruction
  FResultText := '';
  OrgBitmap := TBitmap.Create;
  for IxLine := 0 to High(FLines) do
   for IxWord := 0 to High(FLines[IxLine].Words) do
   begin
    for IxChar := 0 to High(FLines[IxLine].Words[IxWord].Chars) do
     with FLines[IxLine].Words[IxWord].Chars[IxChar] do
      if not Noise then
      begin
       OrgBitmap.Width := MaxPixel.x - MinPixel.x + 1;
       OrgBitmap.Height := MaxPixel.y - MinPixel.y + 1;
       OrgBitmap.Canvas.FillRect(
        Rect(0, 0, OrgBitmap.Width, OrgBitmap.Height));
       for i := 0 to High(Pixels) do
        OrgBitmap.Canvas.Pixels[
         Pixels[i].x - MinPixel.x,
         Pixels[i].y - MinPixel.y] := clBlack;
       StretchBitmap := GetAutoFitBitmap(
                  OrgBitmap, BackProp.InputPatternWidth,
                  BackProp.InputPatternHeight);
       TBackProp.BitmapToLayer(StretchBitmap, BackProp.InputLayer);
       StretchBitmap.Free;
       BackProp.Apply;
       BackProp.GetResult(ResultSymbol);
       FResultText := FResultText + ResultSymbol;
       if (ResultSymbol = '.') and
         (IxWord = High(FLines[IxLine].Words)) and
         (IxChar = High(FLines[IxLine].Words[IxWord].Chars)) then
         FResultText := FResultText + chr(13) + chr(10);
      end;
    FResultText := FResultText + ' ';
   end;
  OrgBitmap.Free;
  Screen.Cursor := crDefault;
end;
// Purpose   : Menyimpan gambar pada area gambar teks
procedure TDCRFrame.SavePicture(FileName: string);
begin
  Image.Picture.SaveToFile(FileName);
```

```
 FModified := False;
end;
// Purpose    : Melakukan segmentasi karakter (kata-kata untuk karakter)
procedure TDCRFrame.SegmentChars;
const
 UNLABELED = MaxInt;
type
 TPixelLabel = Integer;
 TPixelLabels = array of TPixelLabel;
 TDirection = (drTop, drTopLeft, drLeft, drBottomLeft);
var
 Dir: TDirection;
 IdChars: TPixelLabels;
 IxLine, IxWord, IxChar, IxPixel: Integer;
 LabeledPixels: array of TPixelLabels;
 NewLabel: TPixelLabel;
 PixelLabels: array[TDirection] of TPixelLabel;
 ReplacementIdChars: TPixelLabels;
 ReplacementLabel: TPixelLabel;
 x, y: Integer;
begin
 for IxLine := 0 to High(FLines) do
  for IxWord := 0 to High(FLines[IxLine].Words) do
  begin
   with FLines[IxLine] do
   begin
    NewLabel := 0;
    SetLength(LabeledPixels, Words[IxWord].Right - Words[IxWord].Left + 2,
                  Bottom - Top + 3);
    for x := 0 to High(LabeledPixels) do
    begin
     LabeledPixels[x, 0] := UNLABELED;
     LabeledPixels[x, High(LabeledPixels[0])] := UNLABELED;
    end;
    for y := 0 to High(LabeledPixels[0]) do
     LabeledPixels[0, y] := UNLABELED;
    for x := 1 to High(LabeledPixels) do
     for y := 1 to High(LabeledPixels[0]) - 1 do
     begin
      if FImageArray[x + Words[IxWord].Left - 1, y + Top - 1] <> 1 then
       LabeledPixels[x, y] := UNLABELED
      else
      begin
       PixelLabels[drTop] := LabeledPixels[x, y - 1];
```

```
      PixelLabels[drTopLeft] := LabeledPixels[x - 1, y - 1];
      PixelLabels[drLeft] := LabeledPixels[x - 1, y];
      PixelLabels[drBottomLeft] := LabeledPixels[x - 1, y + 1];
      if (PixelLabels[drTop] = UNLABELED) and
        (PixelLabels[drTopLeft] = UNLABELED) and
        (PixelLabels[drLeft] = UNLABELED) and
        (PixelLabels[drBottomLeft] = UNLABELED) then
      begin
        LabeledPixels[x, y] := NewLabel;
        SetLength(IdChars, NewLabel + 1);
        IdChars[NewLabel] := NewLabel;
        NewLabel := NewLabel + 1;
      end
      else
      begin
        ReplacementLabel := UNLABELED;
        for Dir := drTop to drBottomLeft do
          if PixelLabels[Dir] <> UNLABELED then
            if ReplacementLabel = UNLABELED then
            begin
              LabeledPixels[x, y] := IdChars[PixelLabels[Dir]];
              ReplacementLabel := IdChars[PixelLabels[Dir]];
            end
            else
              IdChars[PixelLabels[Dir]] := ReplacementLabel;
      end;
    end;
  end;
end;
for IxChar := 0 to High(IdChars) do
  while IdChars[IdChars[IxChar]] <> IdChars[IxChar] do
    IdChars[IxChar] := IdChars[IdChars[IxChar]];

SetLength(ReplacementIdChars, NewLabel);
for IxChar := 0 to High(ReplacementIdChars) do
  ReplacementIdChars[IxChar] := UNLABELED;
NewLabel := 0;
for IxChar := 0 to High(IdChars) do
begin
  if ReplacementIdChars[IdChars[IxChar]] = UNLABELED then
  begin
    ReplacementIdChars[IdChars[IxChar]] := NewLabel;
    NewLabel := NewLabel + 1;
  end;
```

```
 IdChars[IxChar] := ReplacementIdChars[IdChars[IxChar]];
end;
with FLines[IxLine].Words[IxWord] do
begin
 SetLength(Chars, NewLabel);
 for IxChar := 0 to High(Chars) do
 begin
  Chars[IxChar].MinPixel := Point(MaxInt, MaxInt);
  Chars[IxChar].MaxPixel := Point(0, 0);
  Chars[IxChar].Noise := False;
 end;
 for x := 1 to High(LabeledPixels) do
  for y := 1 to High(LabeledPixels[0]) - 1 do
   if LabeledPixels[x, y] <> UNLABELED then
    with Chars[IdChars[LabeledPixels[x, y]]] do
    begin
     SetLength(Pixels, Length(Pixels) + 1);
     Pixels[High(Pixels)] := Point(x, y);
     if x < MinPixel.x then MinPixel.x := x;
     if y < MinPixel.y then MinPixel.y := y;
     if x > MaxPixel.x then MaxPixel.x := x;
     if y > MaxPixel.Y then MaxPixel.y := y;
    end;
 for IxChar := 0 to High(Chars) - 1 do
  if ((Chars[IxChar].MinPixel.x >= Chars[IxChar + 1].MinPixel.x) and
     (Chars[IxChar].MaxPixel.x <= Chars[IxChar + 1].MaxPixel.x)) or
    ((Chars[IxChar + 1].MinPixel.x >= Chars[IxChar].MinPixel.x) and
     (Chars[IxChar + 1].MaxPixel.x <= Chars[IxChar].MaxPixel.x)) then
  begin
   SetLength(Chars[IxChar + 1].Pixels,
          Length(Chars[IxChar + 1].Pixels) +
           Length(Chars[IxChar].Pixels));
   for IxPixel := 0 to High(Chars[IxChar].Pixels) do
    Chars[IxChar + 1].
     Pixels[Length(Chars[IxChar + 1].Pixels) -
          Length(Chars[IxChar].Pixels) + IxPixel] :=
       Chars[IxChar].Pixels[IxPixel];
   if Chars[IxChar].MinPixel.x < Chars[IxChar + 1].MinPixel.x then
    Chars[IxChar + 1].MinPixel.x := Chars[IxChar].MinPixel.x;
   if Chars[IxChar].MinPixel.y < Chars[IxChar + 1].MinPixel.y then
    Chars[IxChar + 1].MinPixel.y := Chars[IxChar].MinPixel.y;
   if Chars[IxChar].MaxPixel.x > Chars[IxChar + 1].MaxPixel.x then
    Chars[IxChar + 1].MaxPixel.x := Chars[IxChar].MaxPixel.x;
   if Chars[IxChar].MaxPixel.y > Chars[IxChar + 1].MaxPixel.y then
```

```
          Chars[IxChar + 1].MaxPixel.y := Chars[IxChar].MaxPixel.y;
         Chars[IxChar].Noise := True;
        end;
       for IxChar := 0 to High(Chars) do
        if not Chars[IxChar].Noise then
         Chars[IxChar].Noise :=
          (Length(Chars[IxChar].Pixels) < FNoiseThreshold);
     end;
    end;
end;
// Purpose   : Melakukan segmentasi baris (teks ke baris) step
procedure TDCRFrame.SegmentLines;
var
 BlankRow: Boolean;
 SearchFor: (sfTop, sfBottom);
 x, y: Integer;
begin
 y := 0;
 SearchFor := sfTop;
 SetLength(FLines, 0);
 repeat
  BlankRow := True;
  for x := 0 to High(FImageArray) do
   if FImageArray[x, y] = 1 then
   begin
    BlankRow := False;
    if SearchFor = sfTop then
    begin
     SetLength(FLines, Length(FLines) + 1);
     FLines[High(FLines)].Top := y;
     SearchFor := sfBottom;
    end;
    Break;
   end;
  if (SearchFor = sfBottom) and BlankRow then
  begin
   FLines[High(FLines)].Bottom := y - 1;
   SearchFor := sfTop;
  end;
  y := y + 1;
 until y = Length(FImageArray[0]);
 if SearchFor = sfBottom then FLines[High(FLines)].Bottom := y - 1;
end;
```

```
// Purpose   : Conducts the word segmentation (lines to words) step
procedure TDCRFrame.SegmentWords;
var
 IxLine: Integer;
 NBlankColumn: Integer;
 SearchFor: (sfLeft, sfRight);
 x, y: Integer;
begin
 for IxLine := 0 to High(FLines) do
  with FLines[IxLine] do
  begin
   x := 0;
   SearchFor := sfLeft;
   NBlankColumn := 0;
   repeat
    NBlankColumn := NBlankColumn + 1;
    for y := Top to Bottom do
     if FImageArray[x, y] = 1 then
     begin
      NBlankColumn := 0;
      if SearchFor = sfLeft then
      begin
       SetLength(Words, Length(Words) + 1);
       Words[High(Words)].Left := x;
       SearchFor := sfRight;
      end;
      Break;
     end;
    if (SearchFor = sfRight) and (NBlankColumn >= FSpaceWidth) then
    begin
     Words[High(Words)].Right := x - NBlankColumn;
     SearchFor := sfLeft;
    end;
    x := x + 1;
   until x = Length(FImageArray);
   if SearchFor = sfRight then Words[High(Words)].Right := x - 1;
  end;
end;
// Purpose   : show/hide the title
procedure TDCRFrame.ShowHideTitle;
begin
 pnlTitle.Visible := not pnlTitle.Visible;
 if pnlTitle.Visible then
  Image.Top := pnlTitle.Height + 1
```

```
  else
    Image.Top := 0;
end;
// Purpose    : Initializes the frame
procedure TDCRFrame.Init;
begin
  FBWThreshold := DEFAULT_BW_THRESHOLD;
  FNoiseThreshold := DEFAULT_NOISE_THRESHOLD;
  FSpaceWidth := DEFAULT_SPACE_WIDTH;
  Image.Canvas.Pen.Width := DEFAULT_DRAWING_WIDTH;
  Clear;
end
end.
```