JURNAL TEKNOLOGI ACADEMIA ISTA ISSN: 1410-5829
Vol. 12 No. 2 Februari 2008

OPTICAL CHARACTER RECOGNITION DENGAN METODE JARINGAN SARAF TIRUAN *BACKPROPAGATION*

Uning Lestari[1]

ABSTRACT

Artificial Neural Network is one of algorithms that is often used in intelligent system. These algorithms build a system based on biology Artificial Neural Network which is similar to human brain in solving a problem. Many applications based on Artificial Neural Network (ANN) have been developed in all areas especially in Pattern recognition. One of ANN is back propagation that is supervised learning. This training process is done by upgrading also achieving some targets. The implementation of character pattern recognition is as an application for pattern character recognition from A-Z which function to analyze sample data in image form (BMP file) that will be changed into grayscale. These conversion will produce character in document that gives benefit for process information in text that has already saved in images so that it can be edited or erased based on user needs.

Keywords*: Artificial Neural Network, Back propagation Algorithm, Optical Character Recognition*

INTISARI

Jaringan Syaraf Tiruan (*Artificial Neural Network*) adalah salah satu algoritma yang sering digunakan dalam pembuatan sistem cerdas. Algoritma ini membangun suatu sistem berdasarkan model jaringan syaraf biologi yang meniru cara kerja otak manusia dalam menyelesaikan suatu permasalahan. Banyak aplikasi berbasis Jaringan Syaraf Tiruan (JST) telah di kembangkan di berbagai bidang khususnya di bidang pengenalan pola (*pattern recognition*). Salah satu metode JST yaitu algoritma Propagasi Balik (*backpropagation*) yang merupakan proses terawasi (s*upervised learning*). Pada proses ini tahap pelatihan dilakukan dengan perbaikan kesalahan serta pencapaian suatu target tertentu. Implementasi pengenalan pola karakter ini sebagai aplikasi untuk pengenalan suatu pola karakter dari A-Z yang fungsi utamanya menganalisa data yang telah ada (sampel) yang berbentuk image (file BMP) yang akan di ubah menjadi bentuk *gray_scale.* Hasil konversi ini akan menghasilkan karakter dalam bentuk dokumen yang bermanfaat untuk mengolah informasi berupa teks yang tersimpan dalam suatu image sehingga dapat diedit, dihapus dan lain-lain sesuai dengan kebutuhan pengguna.

Kata kunci: *Artificial Neural Network, Algoritma Backpropagation, Optical Character Recognition*

PENDAHULUAN

Kemajuan teknologi membuat sebuah perangkat komputer memiliki kemampuan komputasi yang tinggi untuk meningkatkan kinerja dalam pengolahan data menjadi informasi. Namun hal ini tidak dapat diimbangi dengan kemampuan manusia dalam memasukkan data secara manual kedalam komputer agar dapat diolah lebih lanjut.

Hal yang dapat dilakukan oleh manusia untuk mengimbangi kemajuan teknologi adalah dengan mengembangkan sistem yang dapat memanfaatkan teknologi tersebut untuk dapat membantu memasukkan data kedalam komputer. Salah satu dari sistem tersebut adalah sistem pengenalan huruf atau *Optical Character Recognition* ***(OCR).*** Pengenalan huruf adalah suatu sistem dimana data yang berupa lembaran kertas dapat discan menggunakan scanner yang akan menghasilkan gambar yang pada komputer akan dikenali sebagai titik-titik yang mempunyai format (*bitmap*). *Bitmap* inilah yang kemudian akan diproses lebih lanjut dengan menggunakan algoritma tertentu menjadi karakter. Software ini berfungsi untuk mengubah *file Image* menjadi *file* teks.

Bidang ilmu yang melandasi aplikasi ini adalah kecerdasan buatan (*Arti-*

[1] Jurusan Teknik Informatika, ISTA Yogyakarta, uningl@yahoo.com

ficial Intellegence). Dimana jaringan syaraf tiruan adalah salah satu daerah kerja kecerdasan buatan, dan konsep yang dipakai dalam pembuatan sistem pengenalan karakter ini menggunakan jaringan syaraf tiruan metode *backpropagation*. *Backpropagation* merupakan algoritma yang terawasi dan biasanya banyak digunakan oleh *perceptron* dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan *neuron-neuron* yang ada pada lapisan tersembunyi. *Backpropagation* terdiri dari tiga *layer* atau lapisan diantaranya adalah lapisan input, lapisan tersembunyi, dan lapisan *output*. Sehingga nantinya data yang berbentuk bitmap tersebut dapat dikenali dan dapat dilakukan pengeditan.

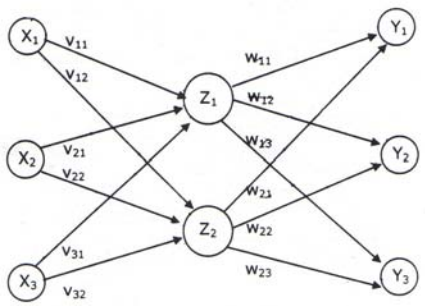
Konsep dari pembuatan *character recognition* ini menggunakan jaringan *backpropagation*. *Backpropagation* merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan oleh perceptron dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan *neuron-neuron* yang ada pada lapisan tersembunyinya. *Backpropagation* terdiri dari 3 *layer* atau lapisan diantaranya adalah lapisan *input*, lapisan tersembunyi, dan lapisan *output*. Algoritma *backpropagation* menggunakan *error output* untuk mengubah nilai bobotnya dalam arah mundur (*backward*). Untuk mendapatkan *error* ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, *neuron-neuron* diaktifkan dengan menggunakan fungsi aktifasi sigmoid, yaitu :

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

Arsitektur jaringan *backpropagation* seperti terlihat pada gambar 1.

Algoritma *Backpropagation* :

1. Langkah 0: Inisialisasi bobot (ambil bobot awal dengan nilai random yang cukup kecil).
2. Langkah 1: Kerjakan langkah 2 s/d 9 berikut, jika kondisi berhenti bernilai False.
3. Langkah 2: Kerjakan langkah 3 s/d 8 berikut untuk tiap-tiap pasangan elemen yang akan dilakukan pembelajaran.



Gambar 1. Arsitektur Jaringan Backpropagation

Untuk setiap pasangan elemen yang akan dilakukan pembelajaran, maka menggunakan *feed forward* terlebih dahulu:

Feedforward:

4. Langkah 3: Tiap-tiap unit input (X_i, i=1,2,3,…,n) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit pada lapisan yang ada di atasnya (lapisan tersembunyi).
5. Langkah 4: Tiap-tiap unit tersembunyi (Z_i, j=1,2,3,…,p) menjumlahkan sinyal-sinyal *input* terbobot:

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

…….(2)

gunakan fungsi aktivasi untuk menghitung sinyal *output*nya: $z_j = f(z\_in_j)$ dan kirimkan sinyal tersebut ke semua unit di lapisan atasnya (unit-unit *output*).

6. Langkah 5: Tiap-tiap unit *output* (Yk, k=1,2,3,…,m) menjumlahkan sinyal-sinyal *input* terbobot.

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

…….(3)

gunakan fungsi aktifasi untuk menghitung sinyal *output*nya: yk = f(y_ink) dan kirimkan sinyal tersebut ke semua unit di lapisan atasnya (unit-unit *output*).

7. Langkah 6: Tiap-tiap unit *output* (Y_k, k=1,2,3,…,m) menerima target pola yang berhubungan dengan pola input pembelajaran, hitung informasi *error*nya: $\delta_k = (t_k$

– y_k) f(y_in_k) kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai w_{jk}): $\Delta w_{jk} = \alpha\ \delta_k\ z_j$ hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai w_{0k}): $\Delta w_{0k} = \alpha\ \delta_k$ kirimkan δ_k ini ke unit-unit yang ada di lapisan bawahnya.

8. Langkah 7: Tiap-tiap unit tersembunyi (Z_j, j=1,2,3,…,p) menjumlahkan *delta input*nya (dari unit-unit yang berada pada lapisan di atasnya):

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad ....(4)$$

kalikan nilai ini dengan turunan dari fungsi aktivasinya untuk menghitung informasi *error*: $\delta_j = \delta\_in_j$ f(z_in_j) kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai v_{ij}): $\Delta v_{jk} = \alpha\ \delta_j\ x_i$ hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai v_{0j}): $\Delta v_{0j} = \alpha\ \delta_j$ Perbaharui bobot dan biasnya:

9. Langkah 8: Tiap-tiap unit *output* (Yk, k=1,2,3,…,m) memperbaiki bias dan bobotnya (j=0,1,2,3, …,p):
   W_{jk}(baru) = w_{jk}(lama) + Δw_{jk} (bobot)
   W_{0k}(baru) = w_{0k}(lama) + Δw_{0k} (bias)
   Tiap-tiap unit tersembunyi (Zj, j=1,2,3,…,p) memperbaiki bias dan bobotnya (i=0,1,2,3,…,n):
   V_{ij}(baru) = v_{ij}(lama) + Δv_{ij} (bobot)
   V_{0j}(baru) = v_{0j}(lama) + Δv_{0j} (bias)
10. Langkah 9: Tes kondisi berhenti, kondisi berhenti dinyatakan dengan bila epoch lebih besar atau samadengan maksimal epoch atau informasi error yang didapatkan kurang dari atau sama dengan target errornya. Epoch merupakan istilah untuk tiap satukali proses perambatan maju dan satu kali proses perambatan mundur, di hitung sebagai satu epoch (iterasi). Sedangkan nilai epoch adalah nilai RMSE (*random mean square error*)

Tujuan dari perubahan bobot untuk setiap lapisan, bukan merupakan hal yang sangat penting. Perhitungan kesalahan merupakan pengukuran bagaimana jaringan dapat belajar dengan baik. Rumus untuk menghitung kesalahan tiap iterasi *Sum Square Error (*SSE) adalah sebagai berikut.

1. Hitung lapisan prediksi atau luaran model untuk masukan pertama.
2. Hitung selisih antara nilai luar prediksi dan nilai target atau sinyal latihan untuk setiap luaran.
3. Kuadratkan setiap luaran kemudian hitung seluruhnya. Ini merupakan kuadrat kesalahan untuk contoh lain

$$SSE = \sum_{i-1}^{N} \sum_{i-1}^{N} (D_{ij} - f_i(x_i))$$

.....(5)

Roat Mean Square Error (*RMS Error*) dihitung sebagai berilkut:

1. Hitung SSE
2. Hasilnya dibagi dengan perkalian antara banyaknya data pada latihan dan banyaknya keluaran kemudian diakarkan.

$$RMSE = \sqrt{SSE / N.K} \quad ..(6)$$

RMSE = Roat Mean Square Error
SSE = Sum Square Error
N = Banyaknya data pada latihan
K = Banyaknya luaran

Dasar pengolahan citra adalah proses pengolahan warna RGB *(Red, Green, Blue)* pada posisi tertentu. Dalam pengolahan citra, warna dipresentasikan dengan nilai heksadesimal, dimana nilainya berada dalam *range* h000000 menyatakan warna hitam sampai dengan hffffff menyatakan warna putih. Untuk memberikan nilai warna tertentu digunakan kombinasi dari setiap warna *Red-Green-Blue* mempunyai range nilai 00 (angka desimalnya adalah 0) dan ff (angka desimalnya 255) atau mempunyai nilai derajat keabuan 256 = 2^8. dengan demikian range warna yang digunakan adlah (2^8) (2^8) (2^8) = 2^{24} (atau yang dikenal dengan istilah *True-Color* pada Windows). Nilai warna yang digunakan di atas merupakan gabungan warna cahaya Merah, Hijau dan Biru. Dengan demikan

untuk menentukan nilai dari suatu warna yang bukan warna dasar digunakan gabungan skala kecerahan dari setiap warnanya. Berikut adalah gabungan warna RGB.

Image Processing atau sering disebut pengolahan citra digital merupakan suatu proses filter gambar asli menjadi gambar lain sesuai dengan keinginan kita (Riyanto, 2005). Misalnya ada suatu gambar yang terlalu gelap. Dengan *image processing*, maka gambar tersebut bisa diproses agar mendapatkan gambar yang jelas. Proses awal yang banyak dilakukan dalam *image processing* adalah mengubah citra berwarna menjadi citra *gray-scale* untuk menyederhanakan model citra. Citra berwarna terdiri atas 3 *layer* matriks, yaitu *R-layer, G-layer*, dan *B-layer*. Oleh karena itu, agar dapat melakukan proses-proses selanjutnya, maka ketiga layer tersebut tetap diperhatikan. Bila setiap proses perhitungan menggunakan 3 *layer*, maka dilakukan 3 perhitungan yang sama pula. Jadi, konsep diubah dengan mengubah ketiga *layer* menjadi sebuah *layer* matriks *gray-scale* dan hasilnya adalah citra *gray-scale*. Dalam citra *gray-scale*, tidak ada lagi warna, tetapi hanya derajat keabuan. Agar dapat mengubah citra berwarna yang mempunyai nilai matriks, masing-masing r, g, dan b, menjadi citra *gray-scale* dengan nilai s, konversi dapat dilakukan dengan mengambil rata-rata nilai r, g, dan b sehingga dapat dituliskan menjadi:

$$s = \frac{r + g + b}{3} \quad \text{.....(7)}$$

Komponen-komponen grafis dipakai untuk menampilkan bentuk-bentuk dan objek-objek grafik. Beberapa diantaranya merupakan komponen sederhana, sedang yang lain dapat berupa komponen yang sangat kompleks di mana masing-masing memiliki fungsi yang berbeda.

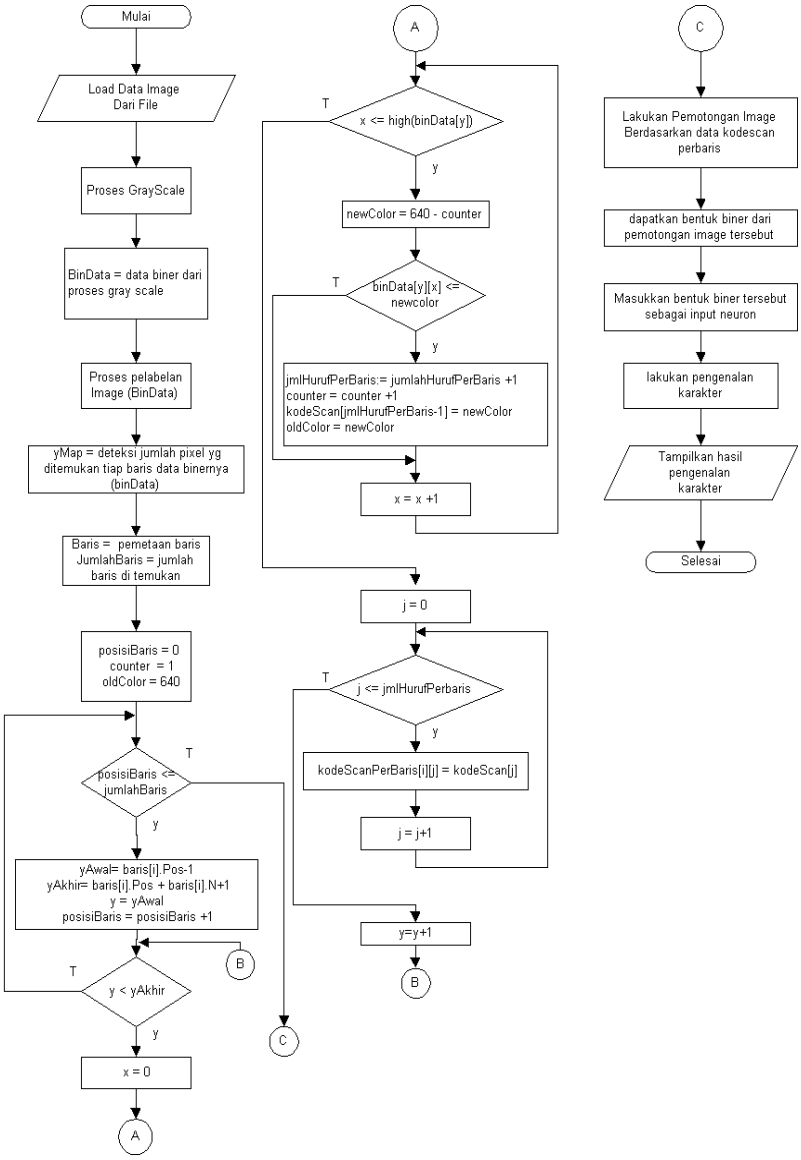
Komponen *Image* dipakai untuk meletakkan sebuah gambar dalam sebuah form. Nama file yang akan ditampilkan ditetapkan pada properti *picture*. Jenis file yang dapat ditampilkan meliputi file Bitmap (BMP), file Icon (ICO), atau Metafile milik Windows (WMF).

Properti-properti yang penting untuk diketahui adalah *Stretch* dan *AutoSize* yang mengatur bagaimana gambar ditampilkan. Bila gambar ingin ditampilkan dengan ukuran asli, beri properti *AutoSize* nilai *True* sebelum file gambar dibaca. Jika gambar sudah tampil pada form lalu nilai properti diubah dari *False* menjadi *True*, maka gambar langsung diubah ke dalam ukuran yang sebenarnya. Bila properti *Stretch* diberi nilai *True* maka akibatnya gambar tersebut dapat diubah ukurannya secara manual.

Komponen *outline* berguna untuk menampilkan data hirarki seperti direktori. Komponen ini sangat fleksibel karena dapat ditampilkan secara sederhana maupun kompleks. Komponen *outline* terdiri dari data-data string yang diakses melalui properti *Lines* dan properti *Items*. Properti *Lines* berupa kelas *TString*. Semua pilohan dalam komponen ini dapat dimasukkan melalui *Editor String*. Penampilan citra bitmap dalam delphi menggunakan komponen TBItmap, dan untuk menggambarkan bitmap dalam canvas, menggunakan pemanggilan prosedur draw atau stretchdraw. Citra bitmap yang ditampilkan menggunakan komponen tersebut dapat dilakukan proses scanning perbaris pixel menggunakan perintah *ScanLine* yang digunakan pada *Device Independent bitmap* untuk image editing yang dilakukan pada tingkat pixel.

PEMBAHASAN

Aplikasi pengenalan pola karakter ini dirancang untuk dapat mengenali berbagai pola karakter (ASCII). Aplikasi dapat menerima *input* dan mengolah *input* dari *user* melalui penekanan tombol *mouse*, seperti halnya membuka gambar, melakukan pencocokan karakter. Cara untuk membuat aplikasi pengenal karakter ini untuk membantu proses pemindahan informasi data karakter yang berupa image (*bitmap*) menjadi dokumen sehingga dapat dilakukan pengeditan terhadap dokumen tersebut. Algoritma sistem ini dapat dilihat pada gambar 2.



Gambar 2 . Flowchart Sistem Pengenalan Karakter

Pertama yang dilakukan yaitu memasukkan gambar, gambar tersebut diubah kebentuk skala abu-abu (*gray-scale*), pembentukan skala abu-abu diberikan untuk memberi batasan bahwa warna yang dijangkau hanya antara 0 hingga 255. Dari bentuk abu-abu tersebut telah didapatkan bentuk data binernya. Langkah berikutnya yaitu proses pelabelan, proses ini bermaksud untuk mencari obyek dalam gambar. Langkah ini akan dijelaskan pada proses berikutnya.

Setelah proses pelabelan selesai maka dilanjutkan dengan mendeteksi jumlah piksel yang ditemukan tiap baris data binernya yang telah dilakukan pelabelan tersebut. Deteksi ini disimpan dalam variabel yMap. Dari pendeteksian ini selanjutnya bisa dilakukan pemetaan baris atau mencari jumlah baris yang ada dalam gambar tersebut. Untuk mengetahui jumlah baris harus mengetahui nilai yAwal dan yAkhir tiap baris, yAwal = baris[i].pos – 1 dan yAkhir = baris[i].pos + baris[i].N + 1. Proses ini dilakukan seba-

nyak jumlah baris yang ada dalam gambar tersebut.

Selanjutnya proses pemetaan kolom perbaris, pemetaan kolom ini dimulai dengan memberi nilai y = yAwal dan x=0. Dalam proses ini sebenarnya sama dengan proses pemetaan baris. Pemetaan kolom dimulai pada posisi baris pertama, jika pada posisi x=0 menemukan obyek yang bernilai 639 maka diberi warna baru *(newcolor)*, kemudian dilakukan pengecekan lagi untuk nilai obyek yang sama yaitu 639 sampai tidak ditemukan lagi, maka jumlah huruf perbaris =1. Jika dalam baris pertama ditemukan nilai yang berbeda maka dalam obyek tersebut diberi warna yang lain untuk mempermudah perhitungan jumlah huruf perbaris. Proses ini dilakukan sampai jumlah baris terpenuhi. Untuk pewarnaan tiap obyek disimpan dalam variabel kodeScan[jumlahHurufPerBaris-1] = *newcolor*. Setelah terpenuhi maka bisa dilanjutkan proses berikutnya yaitu pemotongan gambar perkolom. Proses pemotongan gambar perkolom berdasarkan data kodescan yang telah dilakukan pada proses pemetaan kolom perbaris. Dan yang terakhir akan tampil hasil dari pengenalan karakter ini.

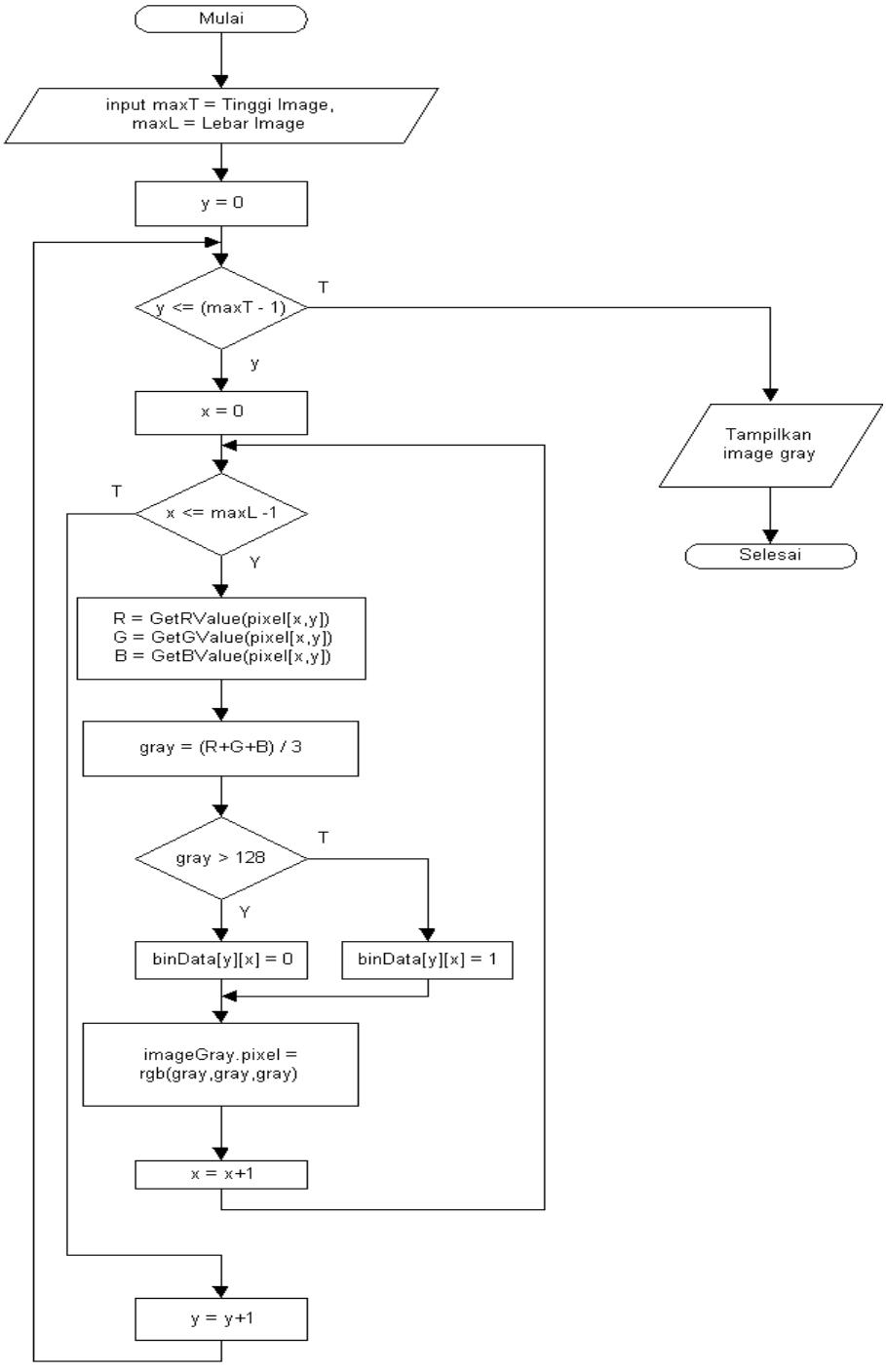
Pada gambar 3 flowchart *gray-scale* menjelaskan tentang proses dari gambar (*image)* asli diubah keimage *gray-scale* atau skala abu-abu untuk mendapatkan bentuk binernya. Bentuk biner ini berfungsi untuk membedakan antara latar belakang gambar dan gambar itu sendiri. Tiap piksel yang ditemukan mempunyai nilai R,G,B. Nilai tiap R,G,B tersebut dijumlahkan dan dibagi dengan 3 maka didapatkan nilai abu-abu. Jika nilai abu-abu (*gray*) lebih dari 128 maka bindata pada posisi yang dihitung bernilai 0. Jika nilai abu-abu (*gray*) kurang dari 128 maka bindata pada posisi yang dihitung bernilai 1. Nilai tersebut disimpan pada variabel *imageGray.pixel*=rgb(*gray,gray,gray*). Proses ini dilakukan sampai tinggi dan lebar pada gambar.

Pada gambar 4, *flowchart* pelabelan *image* menjelaskan tentang pencarian obyek dalam gambar. Pelabelan gambar merupakan pengenalan obyek dari image yang telah dibentuk data binernya, dengan cara melakukan deteksi keberadaan nilai piksel tetangganya, bila keberadaan piksel tetangganya tidak ada, hal ini akan menjadi tanda pemisah obyek satu dengan lainnya. Penomoran (pelabelan) digunakan untuk memudahkan melakukan pemetaan kolom dan mengetahui posisi koordinat yang tepat dari sebuah obyek yang akan dilakukan deteksi karakternya.

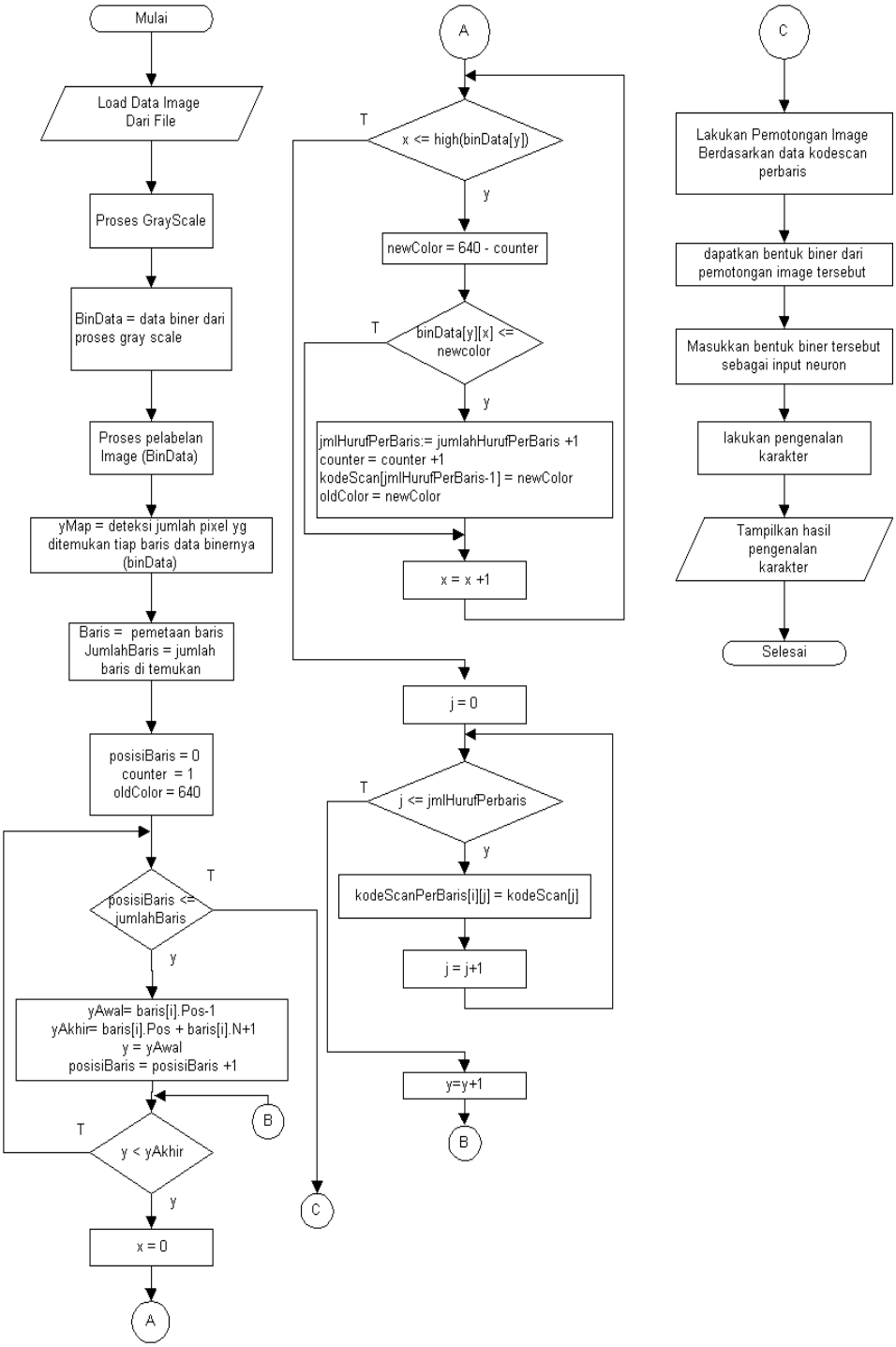
Untuk pencarian posisi dimulai dari y=0 dan x=0 sampai jumlah tinggi dan lebar *image*. Jika pada posisi tertentu ditemukan obyek maka obyek tersebut disimpan dalam variabel yaitu variabel binData [x] [y] yang bernilai 640. Obyek pertama yang ditemukan akan diganti intensitasnya dengan 639, sebab variabel lnum yang digunakan sebagai pengurang adalah 1. Mekanismenya sebagai berikut, bila satu piksel atau titik dari obyek pertama tadi ditemukan, nilai titik tersebut akan diubah intensitasnya menjadi 639. Selanjutnya piksel-piksel disekitar titik tersebut akan diperiksa, bila ada yang bernilai 640, yang berarti terhubung dengan piksel yang diubah (dengan demikian keduanya berarti milik obyek yang sama), nilainya juga diubah menjadi 639. Pencarian ini berulang terus sampai tidak ada piksel yang berintensitas 640, yang artinya pelacakan telah sampai pada tepi obyek dan tidak terjadi penambahan nilai sehingga variabel npx tetap bernilai nol.

Pada kondisi inilah perulangan harus dihentikan. Artinya sebuah obyek telah selesai diberi tanda. Obyek selanjutnya dicari dan diubah intensitasnya menjadi 638, karena kali ini variabel lnum bernilai 2, dengan mekanisme yang sama seperti pada pelacakan obyek pertama.

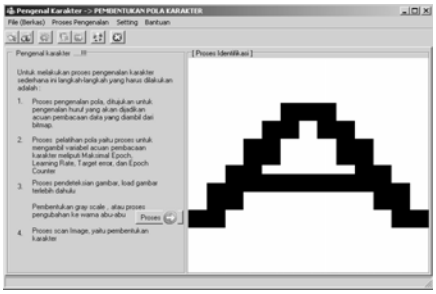
Hasil sistem pengenalan pola karakter optik ini dapat dilihat pada gambar 1.5. Pada menu ini pengguna diminta untuk memilih atau menentukan jenis huruf yang sesuai dengan jenis huruf dengan gambar yang akan dikenali. Proses diatas dilakukan untuk membentuk *image* (bitmap) huruf dari tiap *font* yang terpilih, sebanyak jumlah huruf yaitu dari huruf ‘A’ hingga ‘Z’ dan ‘a’ hingga ‘z’ .



Gambar 3. Flowchart Gray-Scale

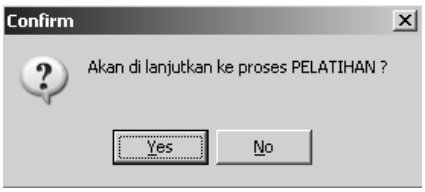


Gambar 4. Pelabelan Image



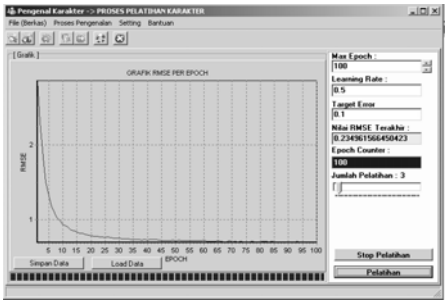
Gambar 5. Tampilan Menu Pembentukan Pola Karakter

Setelah penciptaan pola selesai maka muncul dialog konfirmasi yang menyatakan apakah akan dilanjutkan ke proses pelatihan karakter tersebut (gambar 6)



Gambar 6. Tampilan Dialog, Proses Pelatihan Karakter

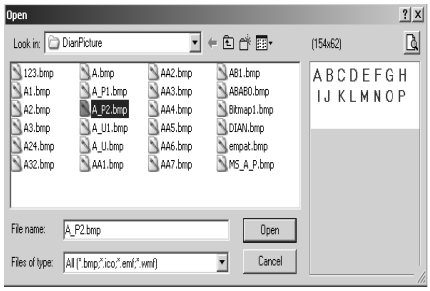
Proses pelatihan dilakukan dengan memasukkan data huruf menjadi masukkan (*input neuron*) bagi jaringan dan penentuan karakter tersebut menjadi target keluaran (*output neuron*). Dalam tampilan proses pelatihan terdapat grafik nilai rata-rata kesalahan (RMSE) per *epoch* serta nilai-nilai yang digunakan untuk pelatihan dengan menggunakan metode *backpropagation* (gambar 7)



Gambar 7. Tampilan Menu Grafik Pelatihan

Pada proses pelatihan, input yang dapat diberikan adalah maksimal *epoch* (iterasi), *learning rate* dan penentuan target *error*. Penentuan target *error* dengan catatan, bila nilai target *error* semakin kecil dapat dikatakan bahwa tingkat akurasinya juga semakin tepat (mendekati 100%), dengan konsekuensi *epoch* (iterasi) dapat lebih besar dari maksimal *epoch* yang di tentukan. *Learning rate* dapat dimasukan nilai sembarang contoh dalam sismtem ini memakai 0.5 karena semakin kecil learning ratenya maka dapat dikatakan bahwa tingkat akurasinya semakn tepat (mendekati 100 %), dengan penggunaan target error 0.1 semakin kecil kita menggunakan target error maka dalam langkah pengenalan karakter akan mendekati kebenaran.

Menu buka *file* (gambar 8) dapat dilakukan setelah proses pelatihan atau minimal pernah satu kali dilakukan pelatihan untuk seluruh karakternya. Pada menu ini pengguna bisa membuka *file* yang akan dilakukan proses pengenalan karakter. Menu buka *file* berfungsi untuk membuka *file bmp* yang tersimpan dan ditampilkan pada kotak image. Bila menu 'buka *file*' diklik, maka kotak dialog untuk membuka *file bitmap* tersimpan akan ditampilkan seperti pada gambar berikut ini:



Gambar 8. Tampilan Menu Buka File

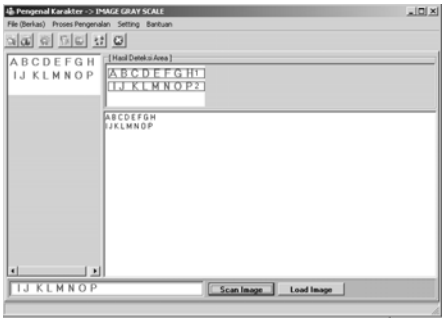
Setelah membuka dan memilih *file*, maka akan ditampilkan dalam menu skala abu-abu *(gray-scale)* yang dapat dilihat pada gambar 9. Setelah file *image* (bitmap) dipilih maka proses berikutnya adalah pembentukan *gray-scale*. Prosedur ini merupakan kode untuk tombol "skala abu-abu". Dalam proses ini, dari gambar asli diubah menjadi gambar skala abu-abu. Untuk mendapatkan skala a-

 ISSN: 1410-5829

bu-abu atau nilai *gray* ini menggunakan rumus (R+G+B)/3. Setelah memperoleh hasil dari perhitungan tersebut kemudian dilakukan pembentukan *image*. Bila nilai *gray* > 128 maka dataBiner[y][x] = 0 bila tidak maka dataBiner[y][x] = 1. Dari proses tersebut *image* sudah terbentuk. Tampilan dari menu *gray-scale* dapat dilihat pada berikut ini:

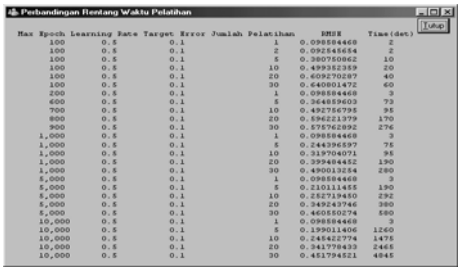
Gambar 9. Tampilan Menu Gray-Scale

Setelah proses pembentukan *grayscale* selesai dilanjutkan dengan proses pengenalan dari sebuah image ke bentuk karakter yang dapat dilihat pada gambar 10. Proses ini membutuhkan empat proses lagi untuk mendapatkan pola karakter dari file *image* yang dipilih. Empat proses tersebut adalah proses pelabelan, proses pemetaan baris, proses pemetaan kolom perbarisnya, dan proses pengecekan posisi untuk melakukan pemotongan *image* per kolom (per obyeknya).



Gambar 10. Tampilan Menu Proses Pengenalan

Hasil perbandingan waktu dari beberapa kali pelatihan yang dilakukan dapat dilihat pada gambar 11. Pada tabel ini hanya sebagai keterangan rentang waktu pelatihan dimana bila max epochnya berbeda, jumlah pelatihanya berbeda, maka akan menghasilkan waktu yang berbeda. perbandingan rentang waktu ini diambil secara acak.



Gambar 11. Tabel Perbandingan Rentang Waktu Pelatihan

KESIMPULAN

Pengenalan karakter optikal dengan menggunakan Jaringan Syaraf Tiruan metode *backpropagation* sangat bermanfaat untuk mengolah informasi berupa teks yang tersimpan dalam suatu image sehingga dapat diedit, dihapus dan lain-lain sesuai dengan kebutuhan pengguna. Sistem yang dibuat ini masih bisa untuk dikembangkan ke pengenalan pola selain karakter, dikembangkan untuk image selain BMP, dan dikembangan de-ngan metode lain selain Backpropaga-tion.

DAFTAR PUSTAKA

Mohamed A. Shahin, Mark B. Jaksa, Holger R. Maier, 2004. “Applications Of ANN In Foundation Engineering”, http://www.ecms.adelaide.edu.au/civeng/staff/mjaksa01/pdf/eConf_2004df

Sigit Riyanto, 2005, “Step by Step Pengolahan Citra Digital”, Andi Offset, Yogyakarta

Schalkoff, RJ, 2003, “*Pattern Recognition: Statistical, Structural and Neural Approach*”, Clemson University, John Willey & Sons, Inc., New York

Wisneski, R, 2004, “Digital Image Processing: Optical Character Recognition (OCR)”. http://www.personal.kent.edu/~rwisnesk/jstor/jstor.htm