

IDENTIFIKASI OBJEK BERDASARKAN CITRA WARNA MENGUNAKAN MATLAB

Adhi Kusnadi

adhikusnadi@yahoo.com

Program Studi Teknik Informatika
Fakultas Teknik, Matematika dan Ilmu Pengetahuan Alam
Universitas Indraprasta PGRI

Abstract. *Dalam suatu citra, banyak informasi yang bisa didapat, dan informasi tersebut dapat digunakan untuk menyederhanakan analisis citra, misalkan identifikasi objek dan ekstraksi warna. Citra memiliki citra warna, salah satunya adalah citra warna model RGB. Selain itu sebuah citra juga memiliki citra skala keabuan (grayscale) memiliki unsur-unsur yang terdiri dari entropy, kontras, energi, homogenitas, skala keabuan dan standar deviasi. Unsur-unsur citra RGB dan skala keabuan tersebut dapat dijadikan masukan ke sistem jaringan syaraf tiruan, sehingga sistem jaringan syaraf tiruan (JST) dapat mengenali atau membedakan suatu objek berbentuk citra berdasarkan warnanya. Sebagai alat bantu dalam mengolah citra dan membuat sistem jaringan syaraf tiruan, digunakan software MATLAB versi 7.5.0.342, karena terdapat toolbox untuk mengolah JST yang sangat membantu dan mempermudah dalam pembuatan sistem ini.*

Kata kunci : citra, warna, keabuan, jaringan syaraf tiruan

PENDAHULUAN

Data atau informasi tidak hanya disajikan dalam bentuk teks, tetapi juga dapat berupa gambar, audio (bunyi, suara, musik), dan video. Untuk mengenali suatu data yang terdapat pada gambar atau citra yang memiliki bermacam-macam jenis, dapat digunakan metode pengolahan citra.

Sistem visual manusia dapat membedakan ratusan ribu shade warna dan intersitas, tetapi hanya 100 shade keabun. Oleh sebab itu, dalam suatu citra, masih banyak informasi lainnya yang ada pada warna, dan informasi tersebut juga dapat digunakan untuk menyederhanakan analisis citra, misalkan identifikasi objek dan ekstraksi warna

Citra memiliki citra warna, salah satunya adalah citra warna model RGB, yaitu citra yang memiliki unsur warna merah, warna hijau dan warna biru. Selain itu sebuah citra juga memiliki citra skalakeabuan (*grayscale*) memiliki unsur-unsur yang terdiri dari entropy, kontras, energi, homogenitas, skala keabuan dan standar deviasi (Mathwork, 1999).

Unsur citra warna ditambahkan dengan unsur citra skala keabuan yang berjumlah sembilan, dapat dijadikan masukan ke sistem jaringan syaraf tiruan (JST), sehingga sistem jaringan syaraf tiruan dapat mengenali atau membedakan suatu objek berbentuk citra berdasarkan warnanya.

Digunakannya JST karena kemampuannya untuk meniru sifat sistem yang diinputkan. Selain itu karena JST merupakan topik yang hangat dibicarakan dan mengundang banyak ketertarikan orang dalam menerapkannya dalam berbagai macam aplikasi.

Sebagai alat bantu dalam mengolah citra dan membuat sistem (JST) dalam tulisan ini, digunakan software MATLAB versi 7.5.0.342, karena dalam MATLAB terdapat toolbox untuk mengolah JST yang sangat membantu dan mempermudah dalam pembuatan sistem dan versi tersebut merupakan versi yang cukup terbaru.

TINJAUAN PUSTAKA

Komponen Citra Digital

Setiap citra digital memiliki beberapa karakteristik, antara lain ukuran citra, resolusi dan format nilainya. Umumnya citra digital berbentuk persegi panjang yang memiliki lebar dan tinggi tertentu. Ukuran ini biasanya dinyatakan dalam banyaknya titik atau pixel, sehingga ukuran citra selalu bernilai bulat.

Komputer dapat mengolah isyarat-isyarat elektronika digital yang merupakan kumpulan sinyal biner (bernilai 0 atau 1). Untuk itu, citra digital harus mempunyai format tertentu yang sesuai sehingga dapat mempresentasikan obyek pencitraan dalam bentuk kombinasi data biner.

Pada umumnya untuk penampilan secara visual, nilai data digital tersebut mempresentasikan warna dari citra yang diolah, dengan demikian format citra yang digital berhubungan erat dengan warna. Format citra digital yang banyak dipakai adalah citra biner, skala keabuan (*grey scale*), warna dan warna berindeks.

Citra Skala Keabuan

Bagian ini melibatkan matriks korelasi kejadian (*co-occurrence matrix*) dari sebuah citra. *Co-occurrence matrix* bertujuan menganalisa pasangan pixel yang bersebelahan secara horizontal. Pada objek citra *gray-level*, matriks ini disebut GLCM (*Gray-level co-occurrence*). Jika objek berupa citra biner, maka akan terbentuk matriks GLCM 2 levels (2 x 2). Sedangkan jika objek berupa citra intensitas, maka akan terbentuk matriks GLCM 8 levels (8 x 8) (Mathwork, 1999). Unsur-unsur *gray-level* dalam citra berikut ini dapat digunakan untuk membedakan suatu objek citra, yaitu :

1. Energi
Digunakan untuk mengukur konsentrasi pasangan *gray level*. Nilai ini didapat dengan mengangkatkan setiap elemen dalam GLCM, kemudian dijumlahkan.
2. Kontras

Menyatakan sebarang terang (*lightness*) dan gelap (*darkness*) di dalam sebuah gambar. Berfungsi untuk mengukur perbedaan lokal dalam citra.

3. Homogenitas
Berfungsi untuk mengukur kehomogenan variasi *gray level* lokal dalam citra.
4. Entropy
Berfungsi untuk mengukur keteracakan dari distribusi perbedaan lokal dalam citra.
5. Derajat keabuan.
Merupakan nilai ambang global dari *grayscale* (250 x 250).
6. Standar deviasi
Merupakan nilai standar deviasi dari citra *grayscale*.

Citra Warna (True Color)

Pada citra warna, setiap titik mempunyai warna yang spesifik yang merupakan kombinasi dari 3 warna dasar, masing-masing warna diarahkan ke salah satu *standard hardware* tertentu (RGB, CMY, YIQ), atau aplikasi pengolahan citra (HSI). Penelitian memperlihatkan bahwa kombinasi warna yang memberikan rentang warna yang paling lebar adalah model warna RGB, yaitu merah, hijau dan biru (Munir, 2004). Format citra ini sering disebut sebagai citra RGB (*red-green-blue*). Setiap warna dasar memiliki intensitas sendiri dengan nilai maksimum 225 (8 bit), misalnya warna kuning merupakan kombinasi warna merah dan hijau sehingga nilai RGB-nya adalah 255 255 0, sedangkan warna ungu muda memiliki nilai RGB 150 0 150. Dengan demikian, setiap titik pada citra warna membutuhkan data 3 byte.

Jumlah kombinasi warna yang mungkin untuk citra adalah 2^{24} atau lebih dari 16 juta warna atau bisa dianggap mencakup semua warna yang ada.

Jaringan Syaraf Tiruan

Jaringan syaraf tiruan (JST) diinspirasi oleh struktur jaringan sel-sel

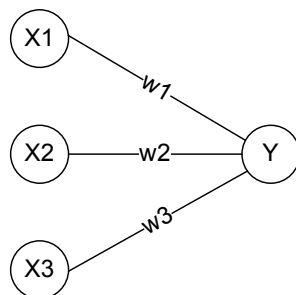
syaraf di dalam otak. Menurut Haykin, S. (1994), "Sebuah jaringan saraf adalah sebuah prosesor yang terdistribusi paralel dan mempunyai kecenderungan untuk menyimpan pengetahuan yang didapatkannya dari pengalaman dan membuatnya tetap tersedia untuk digunakan. Hal ini menyerupai kerja otak dalam dua hal yaitu:

1. Pengetahuan diperoleh oleh jaringan melalui suatu proses belajar.
2. Kekuatan hubungan antar sel saraf yang dikenal dengan bobot sinapsis digunakan untuk menyimpan pengetahuan.

JST merupakan suatu model komputasi yang meniru cara kerja sistem otak manusia. JST merupakan sebuah model komputasi dari otak manusia yang mampu melakukan perhitungan, pengenalan, pengamatan serta pengambilan keputusan. Jaringan syaraf tiruan memanfaatkan struktur pengolahan paralel atas sejumlah pengolah sederhana dan hubungan antarpengolah tersebut.

Dalam jaringan syaraf tiruan, terdapat elemen pengolah yang merupakan model dari neuron. Setiap hubungan elemen pengolah menentukan kemampuan JST. Seperti halnya jaringan syaraf biologis, JST juga memiliki kemampuan untuk belajar dan beradaptasi terhadap masukan-masukannya. JST tidak perlu diprogram secara eksplisit, karena JST dapat belajar dari beberapa contoh pelatihan.

Sebagai contoh, perhatikan neuron Y pada gambar berikut :



Gambar 1. Contoh Arsitektur

JST ditentukan oleh 3 hal :

1. Pola hubungan antar neuron (arsitektur).
2. Metode menentukan bobot penghubung (metode *training/learning/algorithm*).
3. Fungsi aktivasi

Y menerima input dari neuron $x1$, $x2$ dan $x3$ dengan bobot hubungan masing-masing adalah $w1$, $w2$ dan $w3$. Ketiga impuls neuron yang ada dijumlahkan:

$$net = x1w1 + x2w2 + x3w3$$

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. Beberapa arsitektur jaringan yang sering dipakai dalam jaringan syaraf tiruan antara lain :

1. Jaringan Layar Tunggal (*single layer network*).

Dalam jaringan ini, sekumpulan input neuron dihubungkan langsung dengan sekumpulan outputnya. Dalam beberapa model hanya ada sebuah unit neuron output. Contoh dapat dilihat pada gambar 1.

2. Jaringan Layar Jamak (*multi layer network*)

Jaringan ini merupakan perluasan dari layar tunggal. Dalam jaringan ini, selain unit input dan output, ada unit-unit lain (layar tersembunyi). Dimungkinkan pula ada beberapa layar tersembunyi. Sama seperti pada unit input dan output, unit-unit dalam satu layar tidak saling berhubungan. Contoh dapat dilihat pada gambar 4, simbol z merupakan layar tersembunyi.

Ada banyak model arsitektur JST dimulai dari yang sangat sederhana dan

analitik yang ditemukan oleh McCulloch-Pitts, Jaringan Hebb hingga jaringan dengan supervisi yang umum dipakai seperti Perceptron, ADALINE serta Back Propagation.

Jaringan Saraf Tiruan BackPropagation

Secara umum, jaringan seperti ini terdiri dari sejumlah unit neuron sebagai lapisan *input*, satu atau lebih lapisan simpul-simpul neuron komputasi *hidden* (lapisan tersembunyi), dan sebuah lapisan simpul-simpul neuron komputasi *output*. Sinyal *input* dipropagasikan ke arah depan (arah lapisan *output*), lapisan demi lapisan. Jenis jaringan ini adalah hasil generalisasi dari arsitektur *perceptron* satu lapisan, jadi biasa disebut sebagai *multilayer perceptron* (MLPs).

Error back propagation adalah algoritma MLPs yang menggunakan prinsip *supervised learning*. Propagasi balik (ke arah lapisan *input*) terjadi setelah jaringan menghasilkan *output* yang mengandung *error*. Pada fase ini seluruh bobot *synaptic* (yang tidak memiliki aktivasi nol) dalam jaringan akan disesuaikan untuk mengoreksi/memperkecil *error* yang terjadi (*error correction rule*). Untuk pelatihan jaringan, pasangan fase propagasi ke depan dan balik dilakukan secara berulang untuk satu set data latihan, kemudian diulangi untuk sejumlah *epoch* (satu sesi lewat untuk seluruh data latihan dalam sebuah proses pelatihan jaringan) sampai *error* yang terjadi mencapai batas kecil toleransi tertentu atau nol.

Fungsi aktivasi yang digunakan pada arsitektur ini adalah yang menghasilkan nilai kontinu, jadi *output* jaringan juga akan bernilai kontinu. Fungsi aktivasi yang umum digunakan disini adalah *sigmoidal* (biner atau *bipolar*, mengacu pada pembuatan program, akan ditunjukkan yang bipolar, *range*:[-1,1]) :

$$f(v_n) = y_n = \frac{2}{1 + \exp(-v_n)} - 1$$

y_j adalah nilai aktivasi setiap neuron/*node*, dan v_j adalah fungsi integrasi untuk setiap neuron, dimana :

$$v_n = \sum_m w_{mn} y_{mn} - \theta_n$$

w_{ij} adalah bobot *synaptic link*-link antarneuron, dan θ adalah *threshold* (nilai ambang) yang dimiliki setiap neuron. Untuk menyederhanakan perhitungan, pada setiap lapisan (*input*, *hidden*, dan *output*) diberikan neuron tambahan yang selalu memiliki nilai aktivasi 1, dan bobot *synaptic* setiap *link*-nya dengan neuron tertentu sama dengan nilai *threshold* neuron tersebut. Sehingga v_n menjadi :

$$v_n = \sum_m w_{mn} y_{mn}$$

Untuk pelatihan, diberikan satu set data latihan ($s \ t$) dimana s adalah *input* dan t adalah target (*output*) yang valid untuk s . *Error* terjadi bila *output* yang dihasilkan jaringan dengan bobot *synaptic* yang digunakan saat itu tidak sama dengan target (pada satu atau lebih neuron *output*). Sinyal *error* pada sebuah neuron *output* adalah :

$$e_n = t_n - y_n$$

Jumlahan kuadrat sinyal *error* digunakan sebagai acuan untuk melihat apakah jaringan sudah terlatih dengan baik atau tidak. Semakin kecil nilainya menunjukkan bobot *synaptic* setiap *link* semakin menyesuaikan untuk menghasilkan *output* yang benar. (n = jumlah *output*).

$$E = \frac{1}{2} \sum_n e_n^2$$

Error kuadrat rerata (*average squared error*) adalah rata-rata E terhadap cacah data latihan (P).

$$E_v = \frac{1}{P} \sum_P E_P$$

Algoritma *Backpropagation*

Menurut Haykin (1994), prosedur algoritma *Backpropagation* akan mengikuti langkah-langkah sebagai berikut :

1. Inisialisasi ; menentukan konfigurasi jaringan, kemudian menetapkan seluruh bobot *synaptic* dan *threshold* dengan nilai acak kecil yang terdistribusi secara seragam.
2. Menyiapkan data pelatihan ; untuk setiap pasangan (s, t) dilakukan langkah 3 (komputasi ke depan) dan langkah 4 (komputasi balik) secara berurutan.
3. Komputasi ke depan ; *input* yang tersedia adalah nilai aktivasi bagi neuron-neuron sesudah lapisan *input*. Kemudian untuk lapisan-lapisan berikutnya nilai aktivasi dihasilkan kemudian di propagasikan dengan memakai fungsi aktivasi *sigmoid* (rumus 1). Pada lapisan *output* nilai aktivasinya adalah sebagai keluaran.
4. Komputasi balik ; seluruh bobot *synaptic* disesuaikan untuk memperkecil *error*. Mulai dari *link-link* yang menuju lapisan *output*, sampai *link-link* yang menuju lapisan *hidden* pertama. Untuk penyesuaian bobot ini antara lain bisa menggunakan rumus 7 – rumus 10, tergantung posisi *link*. Untuk mempercepat konvergensi, bisa ditambahkan parameter momentum (α). Untuk data latihan ke- p :

$$\begin{aligned} w_{mn}(p) &= w_{mn}(p-1) + \Delta w_{mn}(p) + \alpha(w_{mn}(p-1) - w_{mn}(p-2)) \\ &= w_{mn}(p-1) + \Delta w_{mn}(p) + \alpha \Delta w_{mn}(p-1) \end{aligned}$$

5. Iterasi ; iterasi dilakukan untuk sejumlah *epoch*, jika masih terjadi *error*, sampai *average squared error*

(rumus 5) mencapai batas kecil toleransi tertentu atau nol.

6. Langkah 2 s/d 4 adalah algoritma untuk melatih jaringan. Untuk pengujian pola, cukup dilakukan komputasi ke depan satu lewatan (langkah 3).

Pelatihan Dengan dan Tanpa Supervisi

Berdasarkan cara memodifikasi bobotnya, ada 2 macam pelatihan yang dikenal yaitu dengan supervisi (*supervised*) dan tanpa supervisi (*unsupervised*).

Dalam pelatihan dengan supervisi, terdapat sejumlah pasangan data (masukan-target keluaran) yang dipakai untuk melatih jaringan hingga diperoleh bobot yang diinginkan. Sebaliknya dalam pelatihan tanpa supervisi tidak ada pelatihan yang akan mengarahkan proses, sehingga tidak ada proses pelatihan untuk mengenali objek. Perubahan bobot jaringan dilakukan berdasarkan parameter tertentu dan jaringan dimodifikasi menurut ukuran parameter tersebut. Berdasarkan hasil yang pernah dilaporkan, model pelatihan dengan supervisi lebih banyak digunakan dan terbukti cocok dipakai dalam berbagai aplikasi (Siang, 2005). Dalam pembuatan sistem ini juga digunakan pelatihan dengan supervisi.

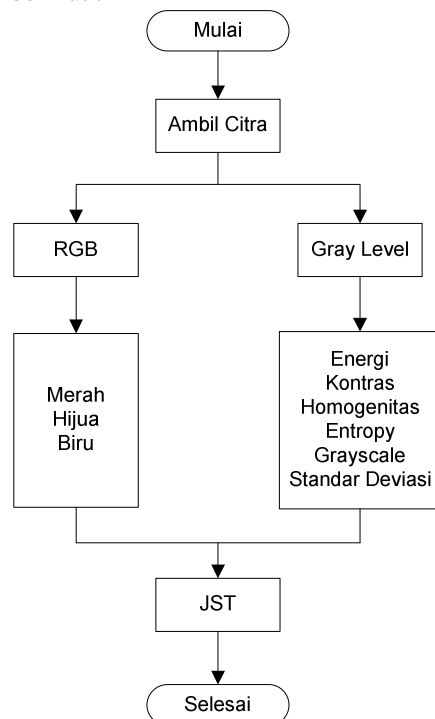
Matlab

MATLAB merupakan singkatan dari *matrix laboratory*. MATLAB pada awalnya ditulis untuk memudahkan akses perangkat lunak matrik yang telah dibentuk oleh LINPACK dan EISPACK. Saat ini perangkat MATLAB telah menggabung dengan LAPACK dan BLAS library, yang merupakan satu kesatuan dari sebuah seni tersendiri dalam perangkat lunak untuk komputasi matrix. Dalam lingkungan perguruan tinggi teknik, Matlab merupakan perangkat standar untuk memperkenalkan dan mengembangkan penyajian materi matematika, rekayasa dan kelimuan. Di industri, MATLAB merupakan perangkat pilihan untuk penelitian dengan produktifitas yang tinggi, pengembangan dan analisisnya.

Fitur-fitur MATLAB sudah banyak dikembangkan, dan lebih kita kenal dengan nama *toolbox*. Sangat penting bagi seorang pengguna MATLAB, *toolbox* mana yang mendukung untuk *learn* dan *apply* teknologi yang sedang dipelajarinya. *Toolbox-toolbox* ini merupakan kumpulan dari fungsi-fungsi MATLAB (*M-files*) yang telah dikembangkan ke suatu lingkungan kerja MATLAB untuk memecahkan masalah dalam kelas khusus. Area-area yang sudah bisa dipecahkan dengan *toolbox* saat ini meliputi pengolahan sinyal, system kontrol, neural networks, fuzzy logic, wavelets, dan lain-lain.

HASIL DAN PEMBAHASAN

Dari hasil pembahasan teori-teori diatas, diimplementasikan dalam sebuah program aplikasi dengan menggunakan MATLAB. Langkah-langkah pembuatan dibuat dalam bentuk *flow chart* sebagai berikut :



Gambar 2. Langkah Ekstraksi Citra

Langkah-langkah tersebut diuraikan dan diimplementasikan dalam program MATLAB sebagai berikut :

- Langkah Pertama

Untuk pengambilan citra :

Dalam mengolah citra disini digunakan sebuah sampel citra yang berasal dari *sample pictures windows seven ultimate* dengan nama *penguins.jpg* (gambar.3). Sebelum diolah sebuah citra di simpan terlebih dahulu di direktori yang akan dipakai sebagai tempat bekerja, jika menggunakan *default* maka matlab akan menggunakan direktori dengan nama *work*. Atau jika tidak, dapat dibuat direktori baru dengan nama disesuaikan dengan keinginan, citra dapat dibaca dengan menggunakan perintah '*imread*'.

Citra yang dipakai disini, hanya digunakan sebagai contoh pengolahan citra, tidak dapat digunakan dalam aplikasi sistem untuk mengidentifikasi suatu objek, karena untuk mengidentifikasi objek diperlukan dua buah objek citra yang mirip tapi memiliki perbedaan warna, misal membedakan buah mangga yang matang dengan buah mangga yang mentah. Kedua objek sama, yaitu mangga, akan tetapi memiliki perbedaan warna karena yang satu matang, yang satu lagi mentah. Aplikasi sistem tersebut akan dibuat dalam tulisan lanjutan oleh penulis. Perintah MATLAB pengambilan gambar sebagai berikut :

```
citraEdit=imread('nama_file.ext')
```

Implementasi :

```
citraEdit=imread('Penguins.jpg')
```

Untuk menampilkan gambar:

```
imshow(citraEdit)
```

Gambar yang dihasilkan :



Gambar 3. Penguins.jpg

Instruksi tersebut berlaku untuk format image BMP, JPE dan TIFF, sedangkan untuk format gambar Dicom proses pembacaannya agak berbeda.

- Langkah Kedua
Ambil RGB dari citra dengan perintah 'mean' :

Implementasi :

```
rgb = mean(mean(citraEdit))
```

output yang didapat :

```
rgb(:, :, 1); % matriks gambar merah
```

```
rgb(:, :, 2); % matriks gambar hijau
```

```
rgb(:, :, 3); % matriks gambar biru
```

Implementasi output yang dihasilkan :

```
rgb(:, :, 1) = 127.8215
```

```
rgb(:, :, 2) = 148.0427
```

```
rgb(:, :, 3) = 168.1294
```

- Langkah Ketiga

Ubah gambar RGB ke gambar grayscale

Dengan perintah:

```
citraAbu=rgb2gray(citraEdit);
```

untuk mengetahui besarnya digunakan :

```
size (citraAbu);
```

Implementasi output yang dihasilkan :

```
ans = 768 1024
```

- Langkah Keempat :

Ambil standar deviasi dengan perintah :

```
std = std2(citraAbu);
```

Implementasi Output :

```
std = 77.3130
```

- Langkah Kelima :

Ambil *treshold* (level) dengan perintah :

```
level = graythresh(citraAbu);
```

Output :

```
level = 0.4353
```

- Langkah Keenam :

Ambil entropy :

```
en=entropy(citraAbu);
```

Output :

```
en = 7.7041
```

- Langkah Ketujuh :

Ambil kontras, homogeneity, energy :

```
glcm = graycomatrix(citraAbu);
```

```
che=graycoprops(glcm,{'contrast','homogeneity','energy'})
```

Output :

che =

```
Contrast: 0.2323
```

```
Energy: 0.1320
```

```
Homogeneity: 0.9209
```

Disain Model

Pembentukan model dapat menggunakan algoritma dalam jaringan syaraf tiruan seperti *perceptron*, *adeline*, *backpropagation* atau model lainnya. Dari hasil pengambilan unsur-unsur citra diatas, disusun menjadi unit-unit input :

Red dijadikan $x1$

Green dijadikan $x2$

Blue dijadikan $x3$

Standar deviasi dijadikan $x4$

Level dijadikan $x5$

Entropy dijadikan $x6$

Kontras dijadikan $x7$

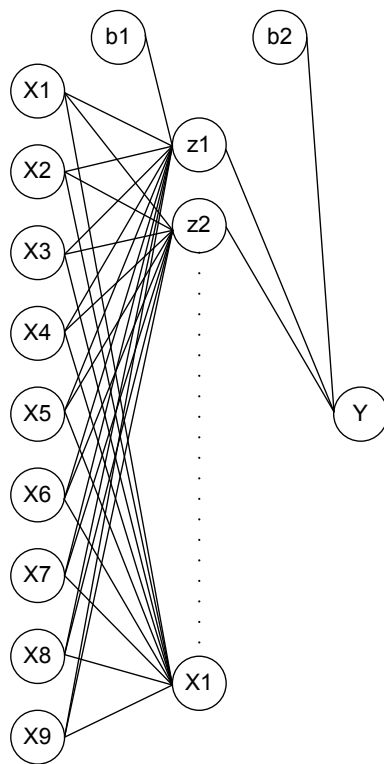
Energi dijadikan $x8$

Homogenitas dijadikan $x9$

$x1, x2, \dots, x9$ dijadikan sebagai inputan kedalam JST, sehingga membentuk sebuah arsitektur. Arsitektur tersebut hanya sebagai contoh, untuk arsitektur yang real tergantung dari model algoritma JST yang digunakan. Seperti model *Perceptron* tentu berbeda dengan model *Backpropagation*. Akan tetapi jika menghendaki sistem

aplikasi yang baik, yang dapat langsung mengenali objek, disarankan menggunakan model *Backpropagation*, karena *Backpropagation* adalah model terbaru yang dikembangkan karena model-model terdahulu memiliki kelemahan, yaitu tidak semua objek dapat dikenali.

Model arsitektur *Backpropagation* dapat dibuat seperti pada gambar berikut ini :



Gambar 4. Arsitektur JST

Pemograman Perceptron

- Pembentukan Jaringan

Penyelesaian berbagai model JST tidak jauh berbeda, dalam perceptron pertama-tama dibentuk :

$$net = newp (PR, S, TF, LF)$$

Dimana :

newp = membentuk jaringan Perceptron.

PR = matrik ordo $R \times 2$ yang menyatakan nilai minimum dan maksimum tiap unit masukan.

R = jumlah unit masukan

S = jumlah neuron target

TF = fungsi aktivasi biner. Defaultnya adalah fungsi treshold (*hardlim*).

LF = fungsi pelatihan. Defaultnya 'learnp'.

Untuk menset bobot dan bias awal ke 0. Untuk mengubahnya digunakan penugasan:

$$net.IW\{i,j\} \text{ dan } net.b\{i\}$$

indek *i* dan *j* menunjukkan bobot awal dari layar-*j* ke layar-*i*. Dalam perceptron hanya dikenal layar masukan dan keluaran, sehingga *net.IW* menunjukkan bobot awal dari layar masukan ke neuron targetnya.

- Menghitung Keluaran Perceptron

Setelah pola masukan diberikan, kita dapat menghitung keluaran jaringannya :

$$[Y, Pf, Af, E, perf] = sim(net, P, Pi, Ai, T)$$

Dimana :

Parameter masukan

net = nama jaringan

P = vektor masukan jaringan

Pi = kondisi delay awal masukan. Default = zeros.

Ai = kondisi delay layar. Default = zeros.

T = vektor target jaringan. Default = zeros.

Parameter hasil

Y = keluaran jaringan

Pf = kondisi akhir delay masukan

Af = Kondisi akhir delay layar

E = Error jaringan = $T - Y$.

perf = unjuk kerja jaringan

Untuk sekedar menghitung keluaran jaringan dapat dipakai statemen sederhana:

$$y = \text{sim}(\text{net}, p)$$

Untuk menghitung keluaran jaringan, tidak diperlukan target. Akan tetapi jika ingin dihitung kesalahan yang terjadi (selisih antara target dengan keluaran jaringan) maka harus diketahui targetnya.

- Modifikasi Bobot dan Bias

Setelah menghitung keluaran jaringan, langkah berikutnya adalah mengubah bobot berdasarkan selisih antara keluaran jaringan dengan target yang diinginkan.

$\text{dw} = \text{learnp}(W, P, Z, N, A, T, E, gW, gA, D, LP, LS)$

Dimana :

W = matriks bobot

P = vektor masukan

Z = vektor masukan dengan bobot

N = vektor masukan net

A = vektor keluaran

T = vektor layar target

E = vektor layar error

gW = gradien bobot terhadap unjuk kerja

gA = gradien keluaran terhadap unjuk kerja

D = jarak neuron

LP = parameter pemahaman

LS = state pemahaman

Untuk menyingkat keseluruhan proses pelatihan (mulai dari perhitungan keluaran jaringan hingga modifikasi bobot) MATLAB menyediakan perintah *train* :

$[\text{net}, \text{tr}, Y, E, Pf, Af] = \text{train}(\text{net}, P, T, Pi, Ai, VV, T, T)$

Dimana :

VV = struktur validasi vektor. Default = [].

TV = struktur vektor uji. Default = [].

Pemograman Backpropagation

- Membentuk Jaringan

$\text{net} = \text{newff}(PR, [s1 \ s2 \dots SN], \{TF1 \ TF2 \dots TFN\}, BTF, BLF, PF)$

Dimana :

Si ($i=1,2,\dots,n$) = jumlah unit pada layar ke- i ($i=1,2,\dots,n$)

Tfi ($i=1,2,\dots,n$) = fungsi aktivasi yang dipakai pada layar ke- i ($i=1,2,\dots,n$). Default = *tansig(sigmoid bipolar)*

BTF = fungsi pelatihan jaringan. Default = *traingdx*.

BLF = fungsi perubahan bobot/bias. Default = *learnqdm*.

PF = fungsi perhitungan error. Default = *mse*.

- Inisialisasi Bobot

Setiap kali membentuk jaringan, MATLAB akan memberi nilai bobot dan bias awal dengan bilangan acak kecil. Bobot dan bias ini akan berubah setiap kali membentuk jaringan. Akan tetapi jika diinginkan memberi bobot tertentu, kita bisa melakukan dengan memberi nilai pada *net.IW*, *net.LW* dan *net.b*.

Perhatikan perbedaan antara *net.IW* dan *net.LW*. *net.IW{,i}* digunakan sebagai variabel untuk menyimpan bobot dari unit masukan layar i ke unit tersembunyi (atau unit keluaran) layar j . Karena dalam Backpropagation, unit masukan hanya terhubung dengan layar tersembunyi paling bawah, maka bobotnya disimpan dalam *net.IW {1,1}*.

Sebaliknya *net.LW{k,j}* dipakai untuk menyimpan bobot dari unit di layar tersembunyi ke- j ke unit layar tersembunyi ke- k .

PENUTUP

Penggunaan unsur-unsur citra warna model RGB yang terdiri dari warna merah, hijau dan biru ditambah citra skala keabuan (grayscale) memiliki unsur-unsur yang terdiri dari entropy, kontras, energi, homogenitas, skala keabuan dan standar deviasi. Dapat dapat digunakan untuk mengenali atau membedakan suatu objek

berbentuk citra berdasarkan warnanya. Unsur-unsur tersebut digunakan sebagai masukan jaringan syaraf tiruan. Penggunaan software MATLAB sangat membantu dan mempermudah dalam pembuatan sistem ini. Untuk mempercantik program aplikasi, dapat digunakan GUI (*Graphic User Interface*) yang sudah disediakan oleh MATLAB.

DAFTAR PUSTAKA

- Gasim. 2006. **Jaringan Syaraf Tiruan Untuk Pengenalan Jenis Kayu Berbasis Citra**. Thesis, IPB.
- Iqbal, Muhammad. 2005. **Dasar Pengolahan Citra Menggunakan MATLAB**. IPB 2005.
- Krisnawati. 2009. **Transformasi Citra Menggunakan MATLAB**. Jurnal.
- Siang, Jong Jek. 2005. **Jaringan Syaraf Tiruan dan Pemograman Menggunakan MATLAB.**, Penerbit Andi Yogyakarta.
- Suteja, Bernard Suteja. 2005. **Penerapan Jaringan Saraf Tiruan Propagasi Balik Studi Kasus Pengenalan Jenis Kopi**. Jurnal.