BAB II
LANDASAN TEORI

2.1 CITRA DIJITAL

Istilah "citra" yang digunakan dalam pengolahan citra dapat diartikan sebagai suatu fungsi kontinu dari intensitas cahaya dalam bidang dua dimensi. Pemrosesan citra dengan komputer dijital membutuhkan citra dijital sebagai masukkannya. Citra dijital adalah citra kontinu yang diubah dalam bentuk diskrit, baik koordinat ruang maupun intensitas cahayanya. Pengolahan dijitalisasi terdiri dari dua proses yaitu pencuplikan (*sampling*) posisi, dan kuantisasi intensitas. Citra dijital dapat dinyatakan dalam matriks dua dimensi *f* (x,y) dimana 'x' dan 'y' merupakan koordinat piksel dalam matriks dan '*f*' merupakan derajat intensitas tersebut. Citra dijital berbentuk matriks dengan ukuran M × N, susunannya sebagai berikut [2]:

$$f(x,y)=\begin{Bmatrix} f(0,0) & f(0,1) & f(0,2) & .... & f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2) & ..... & f(1,N-1) \\ f(2,0) & f(2,1) & f(2,2) & ..... & f(2,N-1) \\ f(M-1,0) & f(M-1,1) & f(M-1,2) & ..... & f(M-1,N-1) \end{Bmatrix} \quad \text{..... (2.1)}$$

Suatu citra *f*(x,y) dalam fungsi matematis dapat dituliskan sebagai berikut:

Dimana: M = banyaknya baris pada array citra

N = banyaknya kolom pada array citra

G = banyaknya skala keabuan (*graylevel*)

Interval (0, *G*) disebut skala keabuan (*grayscale*). Besar *G* tergantung pada proses dijitalisasinya. Biasanya keabuan 0 (nol) menyatakan intensitas hitam dan *G* menyatakan intensitas putih. Untuk cira 8 bit, nilai *G* sama dengan $2^8 = 256$ warna (derajat keabuan).

Jika citra dijital diperhatikan secara seksama, akan melihat titik – titik kecil berbentuk segiempat yang membentuk citra tersebut. Titik – titik tersebut merupakan satuan terkecil dari suatu citra dijital disebut sebagai "*picture element*", "*pixel*", piksel, atau "*pel*". Jumlah piksel per satuan panjang akan

menentukan resolusi citra tersebut. Makin banyak piksel yang mewakili suatu citra, maka makin tinggi nilai resolusinya dan makin halus gambarnya. Pada sistem dengan tampilan citra dijital yang dirancang dengan baik (beresolusi tinggi), titik – titik kecil tersebut tidak dapat terlihat karena ukurannya yang terlalu kecil.

Untuk aplikasi 3 (tiga) dimensi dikenal satuan *voxel* yang memiliki dimensi ketebalan dan berbentuk kubus. Beberapa sistem peralatan *medical imaging* seperti *CT-Scan* atau *ultrasonography* menggunakan bentuk citra dijital 3 (tiga) dimensi sebagai keakuratan pengamatan

Citra bisa digolongkan berdasarkan terdefinisi atau tidaknya citra tersebut pada setiap titik spasial (x,y) dan terhingga atau tidak terhinggganya nilai kecerahan citra. Berdasarkan penggolongan ini, citra dibagi menjadi kontinu-kontinu, kontinu-diskrit, diskrit-kontinu, dan diskrit-diskrit .

Citra kontinu-kontinu adalah citra dengan presisi yang tak terhingga, terdefinisi pada setiap titik spasial. Citra ini sering disebut citra analog atau citra kontinu. Citra ini dapat direpresentasikan dengan presisi yang terhingga untuk meghasilkan citra kontinu diskrit. Citra jenis ini dihasilkan dari proses kuantisasi yang memetakan nilai kecerahan citra yang sebenarnya (tak berhingga) menuju nilai kecerahan yang berhingga yang mampu diakomodasi oleh proses komputer. Citra juga bisa tetap memiliki tingkat kecerahan yang tak berhingga namun hanya terdefinisi pada kumpulan titik spasial yang terbatas. Citra jenis ini dihasilkan dari proses sampling yang menghasilkan citra diskrit kontinu. Karena komputer dijital beroperasi pada presisi yang terbatas, maka komputer dijital cocok untuk citra diskrit-diskrit. Pada citra jenis ini koordinat spasial dan tingkat kecerahan dikuantisasi menuju presisi numerik yang mampu ditangani komputer dijital. Citra jenis ini biasa disebut citra dijital atau citra diskrit.

Citra monokrom atau citra hitam putih ialah citra yang memiliki nilai $f(x,y)$ untuk satu warna yang merepresentasikan *graylevel*. Selain itu ada pula citra berwarna dimana nilai $f(x,y)$ merepresentasikan tiga komponen warna pada setiap titik pada koordinat spasial. Tiga komponen warna tersebut adalah *Red Green Blue* (RGB – merah hijau biru). Sehingga untuk citra berwarna $f(x,y)$ menjadi [2]:

$$f(x,y) = \{f_{merah}(x,y), f_{hijau}(x,y) f_{biru}(x,y)\} \quad \text{.....(2.2)}$$

Citra berwarna tersebut bisa diinterpretasikan sebagai tumpukan (*stack*) dari tiga citra. Tiga komponen warna tersebut masing-masing disebut kanal (*channel*). Dengan demikian citra monokrom hanya memiliki satu kanal, sedangkan citra RGB memiliki tiga kanal.

2.1.1 *Sampling* dan *Graylevel Quantization*

Untuk prosesi komputer, citra kontinu harus diubah menjadi citra diskrit dengan melakukan dua proses yaitu *sampling* dan *graylevel quantization*. Citra kontinu yang di-sampling dan dikuantisasi akan menghasilkan N × M array dimana setiap elemen array (disebut *picture element / pixel*) bersifat diskrit dan merupakan representasi dari tingkat kecerahan (*graylevel*) citra pada titik koordinat spasialnya.

Tingkat keabuan yang ada menentukan storage bits dengan perhitungan suatu citra yang disampling menjadi N x M array [2] :

$$N = 2^n, M = 2^k, Maka \;\; G = 2^m \quad \text{..... (2.3)}$$

Dengan G merupakan nilai *graylevel*. Maka *storage bit* yang dibutuhkan adalah B= N × M × m Contohnya *storage bit* yang dibutuhkan untuk gambar dengan ukuran piksel 256 x 256 dengan *graylevel* 64 membutuhkan 393.216 *bit* untuk penyimpanan.

2.1.2 *Reshaping*

Reshaping yaitu pembentukan matriks menjadi suatu baris, misalkan matriks N × M menjadi matriks 1 × NM. Tujuan *reshaping* adalah untuk memudahkan dalam perhitungan rata-rata parameter karakteristik matriks citra yaitu rata-rata nilai real FFT dan rata-rata nilai DCT.

Citra berwarna dapat dinyatakan dengan banyak cara, salah satunya adalah dengan menggunakan sinyal RGB. Pada cara ini, sebuah citra berwarna dinyatakan sebagai gabungan dari tiga buah citra *monochrome* merah, hijau, dan biru yang berukuran sama. Warna untuk setiap pikselnya tergantung dari komposisi ketiga komponen pada koordinat tersebut. Konsep ini digunakan luas untuk berbagai aplikasi citra berwarna.

Untuk menyederhanakan perhitungan, semua citra pada laporan ini akan diolah ke dalam bentuk graylevel, di mana pada citra berwarna direprentasikan dengan nilai yang sama pada ketiga komponen RGB-nya. Penyederhanaan ini akan mengurangi waktu yang dibutuhka1n untuk melakukan pengolahan citra.

2.2 *IMAGE RECOGNITION*

Pengolahan citra (*image processing*) merupakan proses mengolah piksel-piksel dalam citra dijital untuk suatu tujuan tertentu. Beberapa alasan dilakukannya pengolahan citra pada citra dijital antara lain yaitu [2]:

1. Untuk mendapatkan citra asli dari suatu citra yang sudah buruk karena pengaruh derau. Proses pengolahan bertujuan mendapatkan citra yang diperkirakan mendekati citra sesungguhnya.
2. Untuk memperoleh citra dengan karakteristik tertentu dan cocok secara visual yang dibutuhkan untuk tahap yang lebih lanjut dalam pemrosesan analisis citra

Dalam proses akuisisi, citra yang akan diolah ditransformasikan dalam suatu representasi numerik. Pada proses selanjutnya reprentrasi numerik tersebutlah yang akan diolah secara dijital oleh komputer.

Pengolahan citra pada umumnya dapat dikelompokan dalam dua jenis kegiatan, yaitu:

1. Memperbaiki kualitas citra sesuai kebutuhan
2. Mengolah informasi yang terdapat pada citra

Bidang aplikasi yang kedua ini sangat erat kaitannya dengan komputer aided analysis yang umumnya bertujuan untuk mengolah suatu objek citra dengan cara mengekstraksi informasi penting yang terdapat di dalamnya. Dari informasi tersebut dapat dilakukan proses analisis dan klasifikasi secara cepat memanfaatkan algoritma perhitungan komputer.

Dari pengolahan citra diharapkan terbentuk suatu sistem yang dapat memproses citra masukan hingga citra tersebut dapat dikenali cirinya. Pengenalan ciri inilah yang sering diaplikasikan dalam kehidupan sehari hari. Aplikasi yang dibahas pada laporan ini adalah aplikasi di bidang kedokteran, yaitu untuk

aplikasi analisis Dalam pengolahan citra dijital terdapat lima proses secara umum, yaitu [2]:

1. *image restoration*
2. *image enhancement*
3. *image data compaction*
4. *image analysis*
5. *image reconstruction*

2.2.1 Pemisahan Objek Iris dari Citra Mata

Citra mata mengandung objek-objek selain iris yang dalam proses analisis biometrik tidak dibutuhkan, bahkan terkadang mengganggu. Objek tersebut misalnya kelopak mata, alis mata dan bagian lainnya. Untuk itu diperlukan sebuah teknik pemisahan objek iris dari citra mata keseluruhan. Metode yang dikembangkan untuk pemisahan ROI iris mata ini dilakukan dengan 3 (tiga) tahap : Binerisasi, Operasi Morfologi dan rekonstruksi *Look Up* .

2.2.2 Binerisasi

Dalam hal ini citra mata akan diubah dari citra warna menjadi citra *grayscale* lalu menjadi citra biner dengan *threshold* nilai *graylevel* 128. Hasil dari proses ini adalah bentuk iris mata yang berwarna hitam (pixel 0) dan selainnya berwarna putih (pixel 1). Proses ini bisa disebut juga tahap segmentasi yang memisahkan roi iris yang diinginkan dengan bagian lain mata yang tidak dibutuhkan secara global.

2.2.3 Operasi Morphologi

Bentuk iris pada mata kadangkala terganggu oleh kehadiran kilatan blitz atau pantulan cahaya yang akan menimbulkan spot putih (*glare*) di dalam bagian iris. Dalam citra biner hal ini dapat dilihat dari kehadiran lubang putih pada bagian iris yang berwarna hitam. Untuk menghilangkan noise seperti ini dapat dilakukan operasi morphologi untuk menutup lubang *glare* yang terjadi. Proses yang dilakukan adalah proses *closing*, yaitu dengan melakukan operasi dilasi dan erosi secara berurutan.

2.2.4 Rekonstruksi *Look Up*

Dari proses segmentasi sebelumnya, akan diperoleh bagian iris sebagai bentuk bulatan hitam dengan latar putih. Langkah selanjutnya adalah untuk mengambil hanya bagian iris dengan melakukan rekonstruksi ulang citra mata asli.

2.2.5 Transformasi Koordinat Polar [3]

Bentuk iris yang berupa lingkaran akan sangat menyulitkan untuk dianalisis dan diolah lebih lanjut. Pola susunan piksel yang dianalisa harus mengikuti algoritma tertentu yang memungkinkan pengambilan piksel dengan bentuk geometri lingkaran. Hal ini akan sangat merepotkan dan tidak efisien, untuk mengatasinya citra iris terlebih dahulu di ubah ke dalam bentuk yang dapat di rekayasa oleh sistem. Pengubahan bentuk ini dapat dilakukan dengan melakukan transformasi koordinat polar dari citra iris. Untuk dapat melakukan transformasi tersebut, pertama harus dilakukan deteksi tepian, perhitungan parameter koordinat polar, baru kemudian dilakukan pembentukan citra koordinat polar itu sendiri.

2.2.5.1 Perhitungan Parameter Koordinat Polar [3]

Parameter lingkaran dibutuhkan untuk pembentukan citra polar. Titik pusat dan jari-jari diperlukan sebagai referensi dalam mentransformasikan citra berbentuk lingkaran ke dalam bentuk koordinat polar.

Untuk dapat menghitung parameter-parameter tersebut, terlebih dahulu diambil beberapa titik sampel pada tepian citra yang membentuk pola lingkaran. Jumlah titik sampel yang diambil sangat berpengaruh pada ketelitian pengukuran. Semakin banyak titik sampel yang diambil maka perhitungan yang dilakukan akan semakin teliti. Namun di sisi lain perhitungan akan berjalan sangat rumit dan memakan waktu cukup lama. Untuk itu, pemilihan titik sampel harus dipertimbangkan secara optimum karena akan sangat berpengaruh pada unjuk kerja sistem nantinya.

2.2.5.2 Perhitungan Titik Pusat [3]

Rumus dasar lingkaran digunakan untuk menghitung titik pusat dari lingkaran.

$$R^2 = (x-a)^2 + (y-b)^2 \quad \text{..... (2.4)}$$

$$R^2 = (x^2 - a^2) - (2x)a - (2y)b + a^2 + b^2 \quad \text{.....(2.5)}$$

Dimana: R = Jari – jari yang di cari

a dan b = Titik pusat citra

Dari persamaan di atas untuk mencari 2 (dua) buah variabel absis dan ordinat titik pusat dibutuhkan sedikitnya minimal 3 (tiga) buah persamaan yang berasal dari 3 (tiga) titik sampel. Oleh karena itu pemilihan sampel harus lebih dari 3 (tiga) buah titik pada tepian lingkaran. Jumlah perhitungan yang akan dilakukan merupakan hasil kombinasi semua titik sampel pada tiap perhitungan 3 (tiga) titik yang dibutuhkan. Jumlah perhitungan yang dilakukan adalah sebanyak 3xN proses dengan N adalah banyaknya titik sampel yang diambil.

2.2.5.3 Perhitungan Jari-jari Lingkaran [3]

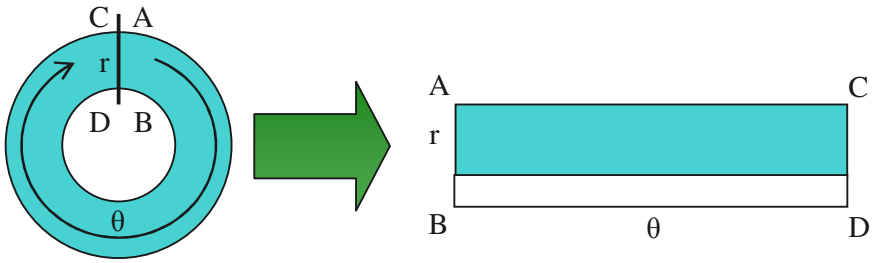
Setelah mengetahui titik pusat, jari-jari lingkaran dapat dengan mudah dihitung menurut persamaan berikut :

$$R_{nn}^2 = \left((x_n^2 + y_n^2) - (2x_n)a - (2y_n)b + a^2 + b^2\right) \quad \text{n = 0,1,….N.....(2.6)}$$

Hasil perhitungan akhir diperoleh dengan mengambil nilai rata-rata dari N buah perhitungan jari-jari yang diperoleh dari N buah titik sampel yang berbeda.

2.2.5.4 Pembentukan Citra Polar [3]

Parameter-paramter yang telah dihitung di pakai untuk mengubah bentuk citra iris yang berbentuk lingkaran ke dalam koordinat polar. Citra ditransformasikan ke dalam bentuk polar dengan titik pusat sebagai acuannya. Pada gambar di bawah, lebar data citra (r) hasil transformasi adalah sebesar a-b dan panjang data citra (θ) sebesar c-d. Lebar citra sangat tergantung kepada besar jari-jari dalam dan jari-jari luar lingkaran. Sedangkan panjang data citra tergantung kepada besarnya pengambilan piksel tiap derajat lingkaran. Proses transformasi dapat diilustrasikan pada Gambar 2.1.



Gambar 2.1. Ilustrasi Transformasi Koordinat Polar [3]

2.3 METODE JARINGAN SYARAF TIRUAN

Jaringan Syaraf Tiruan (JST) atau *Artificial Neural Network* merupakan teknik komputasi yang digunakan untuk mengenali suatu pola atau objek dengan mengadaptasi sistem jaringan syaraf pada manusia. Beberapa definisi mengenai JST dikemukakan oleh beberapa ahli, antara lain [4]:

1. Haykin, S. (1994), Neural Networks: A Comprehensive Foundation, NY, Macmillan, "Sebuah jaringan saraf adalah sebuah prosesor yang terdistribusi paralel dan mempuyai kecenderungan untuk menyimpan pengetahuan yang didapatkannya dari pengalaman dan membuatnya tetap tersedia untuk digunakan. Hal ini menyerupai kerja otak dalam dua hal yaitu:
    a. Pengetahuan diperoleh oleh jaringan melalui suatu proses belajar.
    b. Kekuatan hubungan antar sel saraf yang dikenal dengan bobot sinapsis digunakan untuk menyimpan pengetahuan.
2. Zurada, J.M. (1992), Introduction To Artificial Neural Systems, Boston: PWS Publishing Company, mendefinisikan JST sebagai "Sistem saraf tiruan atau jaringan saraf tiruan adalah sistem selular fisik yang dapat memperoleh, menyimpan dan menggunakan pengetahuan yang didapatkan dari pengalaman".
3. DARPA Neural Network Study (1988, AFCEA International Press, p. 60) "Sebuah jaringan syaraf adalah sebuah sistem yang dibentuk dari sejumlah elemen pemroses sederhana yang bekerja secara paralel dimana fungsinya ditentukan oleh stuktur jaringan, kekuatan hubungan, dan pegolahan dilakukan pada komputasi elemen atau nodes

Konsep JST adalah mengambil ide dari *Parallel Distributed Processing* (PDP). Meniru jaringan syaraf manusia, jaringan buatan ini terdiri dari kelompok-kelompok unit proses sederhana yang berkomunikasi dengan mengirimkan sinyal satu dengan lainnya.

Aspek utama dari model distribusi parallel dapat dibedakan:

1. Kumpulan unit proses (*neuron*, sel)
2. Keadaan bergerak y_k untuk setiap unit yang sama pada keluaran tiap unit.
3. Hubungan antar unit. Secara umum diwakili w_{jk} yang menjelaskan efek sinyal dari unit j terhadap unit k.
4. Fungsi pergerakan F_k, yang menentukan level pergerakan berikutnya pada input efektif $s_k(t)$ dan pergerakan pada saat itu y_k (t).
5. Input eksternal (bias, *offset*)
6. Metode pengambilan informasi.
7. Lingkungan yang mendukung, menyediakan masukan, bahkan sinyal eror.

2.3.1. Unit Proses

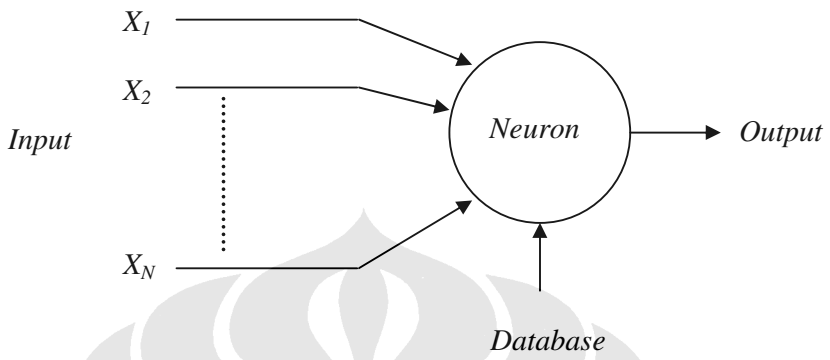
Tiap unit memiliki tugas sederhana, menerima sinyal masukan dari sumber dan menggunakannya untuk memperoleh sinyal keluaran yang diteruskan ke unit selanjutnya. Selanjutnya adalah memperkirakan bobot, yaitu karena sistem bekerja secara paralel dan perhitungan sinyal keluaran dapat terjadi bersamaan.

Selama beroperasi, unit dapat bekerja secara sinkron dan asinkron. Sinkron berarti unit selalu memperbarui pergerakannya serempak dan asinkron berarti memperbarui pergerakan tiap unti pada tiap waktu *t*, dan hanya bisa dilakukan satu unit pada satu waktu.

2.3.2. Pengenalan Pola

Hal yang paling penting pada metode jaringan syaraf adalah pengenalan pola. Ini dapat dilakukan seperti yang ditunjukan Gambar 2.2, dimana jaringan syaraf telah dilatih pada pola tertentu. Saat mempelajari, jaringan dapat mengasosiasikan keluaran dengan pola input. Setelah terbiasa, jaringan dapat

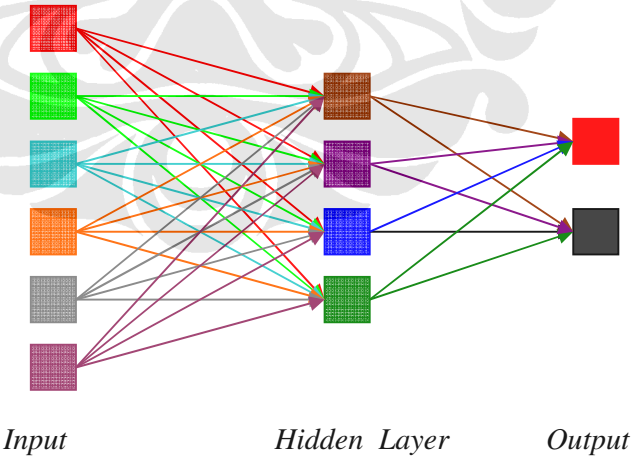
mengidentifikasi masukan untuk mengetahui keluaran sesuai pola keluaran yang dipelajari. Sehingga ketika diberikan suatu pola baru pada masukan, jaringan syaraf akan memperkirakan keluaran sesuai dengan pola keluaran yang dipelajari.



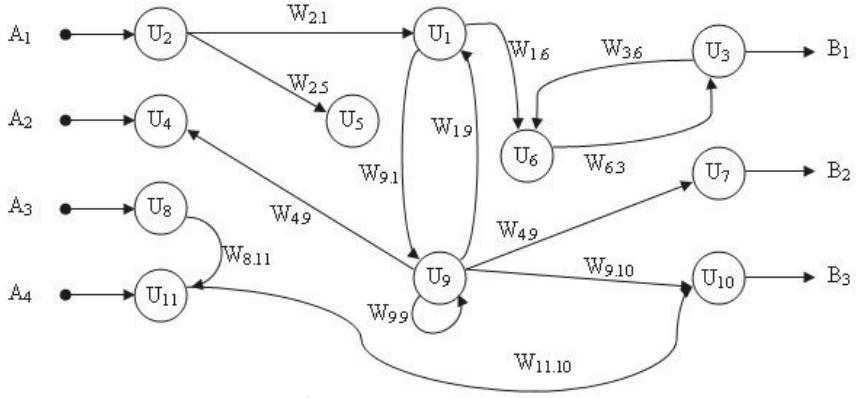
Gambar 2.2. Pola pengenalan pada JST [5]

2.3.3 Arsitektur JST

Ada tiga unit yang dapat dibedakan, yaitu: unit masukan (*input*) yang menerima data dari jaringan syaraf, unit keluaran (output) yang mengirimkan data keluar dari jaringan syaraf, dan unit tersembunyi (*hidden*) yang masukan dan keluarannya tetap berada di jaringan syaraf seperti yang ditunjukkan pada Gambar 2.3 dan Gambar 2.4 dibawah ini.



Gambar 2.3. Contoh arsitektur sederhana jaringan syaraf sederhana [5]



Gambar 2.4. Contoh arsitektur jaringan syaraf kompleks [5]

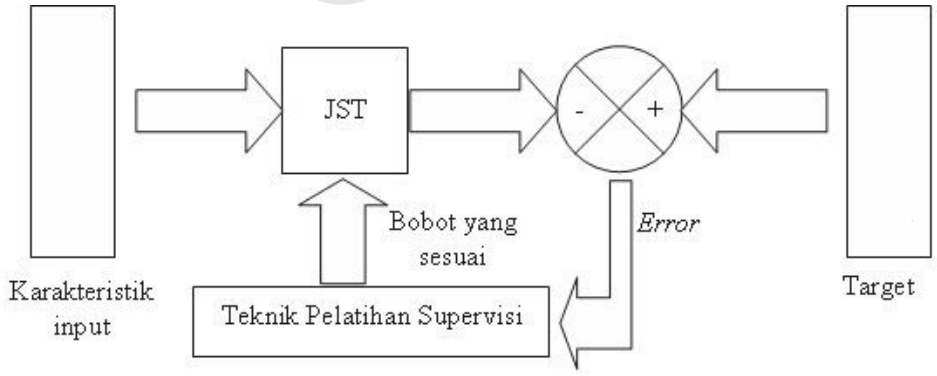
2.3.4 Melatih JST

Jaringan syaraf tiruan harus dikondisikan untuk menerima input yang memiliki karakteristik berbeda untuk menghasilkan keluaran yang diharapkan. Beberapa metode digunakan untuk memperkuat koneksi masukan keluaran ini. Salah satunya dengan mengatur banyaknya beban. Lainnya dengan mengajari JST dengan banyak pola dengan karakteristik berbeda, sehingga bebabn yang ada nantinya akan disesuaikan dengan pola yang mendekatinya.

Pola pembelajaran ini dapat dibedakan menjadi dua, yaitu:

1. Pengajaran dengan Supervisi

   Jaringan dilatih dengan menyediakan masukan dan membandingkannya dengan pola keluaran. Pasangan masukan – keluaran ini dapat diajari secara eksternal atau disediakan oleh sistem jaringan syaraf itu sendiri seperti yang ditunjukkan pada Gambar 2.5 .



Gambar 2.5. Pengajaran dengan Supervisi [6]

2. Pengajaran tanpa Supervisi

   Dimana keluaran dilatih untuk merespon terhadap kluster dan pola pada masukan. Sistem akan menemukan ciri-ciri tertentu pada pola masukan.

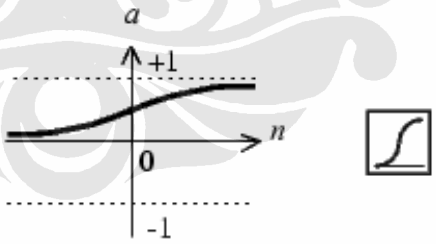
2.3.5 *Backpropagation*

Backpropagation merupakan salah satu algoritma pelatihan terarah. Algoritma *backpropagation* biasa digunakan oleh *perceptron* dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang ada pada lapisan tersembunyinya. Algoritma *Backpropagation* menggunakan eror *output* untuk mengubah nilai bobot-bobotnya dalam arah mundur *(backward).* Untuk mendapatkan eror tersebut, tahap perambatan maju *(forward propagation)* harus dilakukan terlebih dahulu. Pada perambatan maju *neuron-neuron* akan diaktifkan dengan menggunakan fungsi aktivasi yang dapat didiferensiasikan, seperti :

1. Sigmoid

   $$y = f(x) = \frac{1}{1+e^{-\sigma x}} \quad \text{.....(2.7)}$$

   dimana : $f'(x) = \sigma f(x)[1 - f(x)]$ dan fungsinya dapat dilihat pada Gambar 2.6:



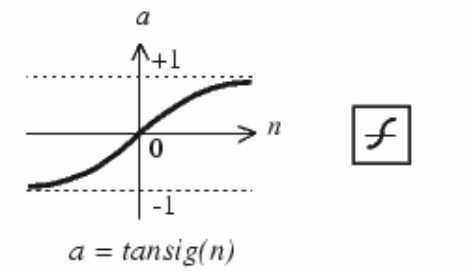
Gambar 2.6. Fungsi aktivasi sigmoid [6]

2. Tansig :

   $$y = f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e} \quad \text{.....(2.8)}$$

   atau $$y = f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \quad \text{.....(2.9)}$$

   dengan : $f'(x) = [1+f(x)][1-f(x)]$ dan fungsinya dapat dilihat pada Gambar 2.7 :

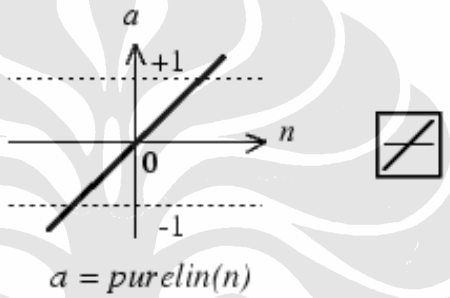


Gambar 2.7. Fungsi aktivasi tansig [6]

3. Purelin

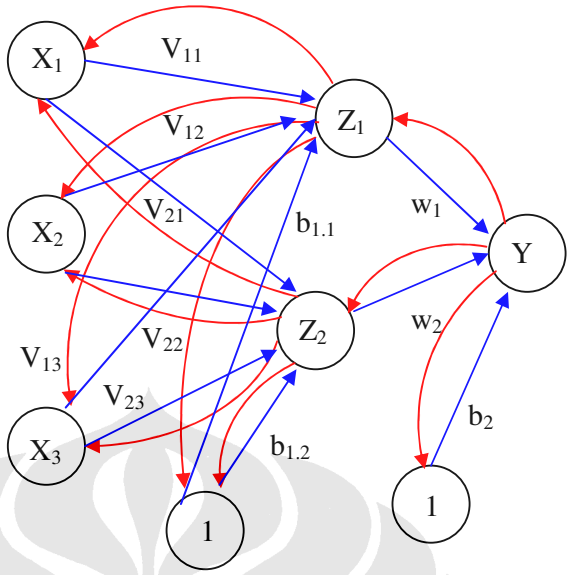
   $y = f(x) = x$ .....(2.10)

   dengan $f'(x) = 1$ dan fungsinya dapat dilihat pada Gambar 2.8 :



Gambar 2.8. Fungsi aktivasi purelin [6]

Pada Gambar 2.9 dapat dilihat arsitektur jaringan *backpropagation* yang terdiri dari 3 unit (*neuron*) pada lapisan input yaitu x_1, x_2,dan x_3; 1 lapisan tersembunyi dengan 2 *neuron* yaitu z_1 dan z_2; serta 1 unit pada lapisan *output*, yaitu y. Bobot yang menghubungkan x_1, x_2,dan x_3 dengan *neuron* pertama pada lapisan tersembunyi adalah v_{11}, v_{21} dan v_{31}. (v_{ij}; bobot yang menghubungkan *neuron input* ke-*j* pada suatu lapisan *neuron* ke-*i* pada lapisan sesudahnya). Bobot bias yang menuju menuju ke *neuron* pertama dan kedua pada lapisan tersembunyi adalah b_{11} dan b_{12} Bobot yang menghubungkan bobot z_1 dan z_2 dengan *neuron* lapisan *output* adalah w_1 dan w_2. bobot bias b_2 menghubungkan lapisan tersembunyi dengan lapisan *output*. Fungsi aktivasi digunakan antar lapisan *input* dengan lapisan tersembunyi dan lapisan tersembunyi dengan lapisan *output*.



Gambar 2.9. Contoh arsitektur jaringan *backpropagation* [7]

Algoritma backpropagation adalah :

a. Inisialisasi bobot (ambil awal dengan nilai *random* yang cukup kecil)
b. Tetapkan : Maksimum Epoh, Target *error*, dan *learning rate* (α)
c. Inisialisasi : Epoh = 0, MSE = 1.
d. Kerjakan langkah-langkah berikut selama (Epoh < Maksimum Epoh) dan (MSE>Target Error) :
   1. Epoh = Epoh + 1
   2. Untuk tiap-tiap pasangan elemen yang akan dilakukan pembelajaran, kerjakan :

      Feedforward :

      a. tiap-tiap unit input (x_i= 1,2,3,…,n) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit pada lapisan yang ada diatasnya (lapisan tersembunyi).
      b. Tiap-tiap unit pada lapisan tersembunyi (Z_j, j = 1,2,3,..,p) menjumlahkan sinyal-sinyal input berbobot [6]:

$$z\_in_j = b_{1j} + \sum_{i=1}^{n} x_i v_{ij} \quad \text{.....(2.11)}$$

Gunakan fungsi aktivasi untuk menghitung sinyal outputnya :

$z_j = f(z\_in_j)$ ……..(2.12)

Dan kirimkan sinyal tersebut ke semua unit di lapisan atasnya (unit-unit output).

c. Tiap-tiap unit output (Y_k, k=1,2,3,…m) menjumlahkan sinyal-sinyal input terbobot [7].

$$y\_in_k = b2_k + \sum_{i=1}^{p} z_j w_{jk}$$ ……..……..……..(2.13)

Gunakan fungsi aktivasi untuk menghitung sinyal outputnya :

$y_k = f(y\_in_k)$...……..……...(2.14)

Dan kirimkan sinyal output tersebut ke semua unit di lapisan atasnya (unit-unit output).

Langkah (b) dilakukan sebanyak jumlah lapisan tersembunyi.

tiap-tiap unit output (Y_k= 1,2,3,…m) menerima target pola yang berhubungan dengan pola input pembelajaran, hitung informasi errornya [7]:

$\delta 1_j = \delta\_in_j f'(z\_in_j)$ ……..……....(2.15)

$\varphi 1_{ij} = \delta 1_j x_j$ ……..……...(2.16)

$\beta 1_j = \delta 1_j$ ……..……...(2.17)

Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk menghitung nilai w_{jk}):

$\Delta w_{jk} = \alpha \varphi 2_{jk}$ .....(2.18)

Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai $b2_k$) :

$\Delta b2_k = \alpha \beta 1_k$ .....(2.19)

Langkah (d) ini juga dilakukan sebanyak jumlah lapisan tersembunyi, yaitu menghitung informasi eror dari suatu lapisan tersembunyi ke lapisan tersembunyi sebelumnya.

d. tiap-tiap unit tersembunyi (Z_j, j=1,2,3,…,p) menjumlahkan delta *input*nya (dan unit-unit yang berada pada lapisan yang ada diatasnya):

$$\delta\_in_j = \sum_{k=1}^{m} \delta 2_k w_{jk} \quad \text{.....(2.20)}$$

Kalikan nilai ini dengan turunan dari fungsi aktivasinya untuk menghitung informasi error :

$$\delta 1_j = \delta\_in_j f'(z\_in_j) \quad \text{.....(2.21)}$$

$$\varphi 1_{ij} = \delta 1_j x_j \quad \text{.....(2.22)}$$

$$\beta 1_j = \delta 1_j \quad \text{.....(2.23)}$$

Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai v_{jk}):

$$\Delta v_{ij} = \alpha \varphi 1_{ij} \quad \text{.....(2.24)}$$

Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai $b1_j$)

$$\Delta b1_j = \alpha \beta 1_j \quad \text{.....(2.25)}$$

Tiap-tiap unit output (Y_k, k = 1,2,3,…,m) memperbaiki bias dan bobotnya (j=0,1,2,…,p):

$$w_{jk}\,(baru) = w_{jk}\,(lama) + \Delta w_{jk} \quad \text{.....(2.25)}$$

$$b2_k\,(baru) = b2_k\,(lama) + \Delta b1_k \quad \text{.....(2.26)}$$

f. Tiap-tiap unit tersembunyi (Z_j= j=1,2,3,…p) memperbaiki bias dan bobotnya(i=0,1,2,…,n):

$$v_{ij}\,(baru) = v_{ij}\,(lama) + \Delta v_{ij} \quad \text{.....(2.27)}$$

$$b1_j\,(baru) = b1_j\,(lama) + \Delta b1_j \quad \text{.....(2.28)}$$

3. Hitung MSE [7]

Setelah dilakukan algoritma tersebut pada jaringan maka akan di dapat jaringan yang sudah di latih. Sehingga untuk melakukan indentifikasi, dapat dilakukan dengan langsung memberikan input dan jaringan akan mengklasifikasinya sesuai dengan bobot-bobot yang diperoleh dari proses *training* sebelumnya.