BAB 2
LANDASAN TEORI

2.1. Citra Digital

Citra atau gambar yang didefinisikan sebagai sebuah fungsi dua dimensi, f(x,y), dimana x dan y adalah koordinat bidang datar, dan harga fungsi f disetiap pasangan koordinat (x,y) disebut intensitas atau level keabuan (*grey level*) dari gambar dititik itu. Jika x,y dan f semuanya berhingga (*finite*) dan nilainya diskrit maka gambarnya disebut citra digital.

2.1.1. Citra

Secara harfiah, citra (*image*) adalah gambar pada bidang dwimatra (dua dimensi) Gambar 2.1 adalah citra gadis model yang bernama lena. Ditinjau dari sudut pandang matematis, citra merupakan fungsi terus menerus (*continue*) dari intensitas bidang dwimatra. Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat – alat optik, misalnya mata pada manusia, kamera, pemindai (*scanner*), dan sebagainya, sehingga bayangan objek yang disebut citra tersebut terekam.

Citra sebagai keluaran dari suatu sistem perekaman data dapat bersifat : optik berupa foto, analog berupa sinyal video seperti gambar pada monitor televisi, digital yang dapat langsung disimpan pada suatu pita magnetik. Citra mempunyai dua jenis kategori yaitu : citra diam (*still image*) adalah citra tunggal yang tidak bergerak. Gambar 2.1 adalah contoh dari citra diam. Citra bergerak (*moving images*) adalah rangkaian citra diam yang ditampilkan secara beruntun (sekuensial) sehingga memberi kesan pada mata kita sebagai gambar yang bergerak. Setiap citra di dalam rangkaian itu disebut *frame*. Gambar-gambar yang tampak pada film layar lebar atau televisi pada hakikatnya terdiri atas ratusan sampai ribuan frame.



Gambar 2.1 Lena [5]

2.1.2. Definisi Pengolahan Citra

Meskipun sebuah citra kaya informasi, namun seringkali citra yang dimiliki mengalami penurunan mutu (degradasi) misalnya mengandung cacat atau derau (*noise*), warnanya terlalu kontras, kurang tajam, kabur (*blurring*), dan sebagainya. Tentu saja citra semacam ini menjadi lebih sulit diinterpretasi karena informasi yang disampaikan oleh citra tersebut menjadi berkurang. Agar citra yang mengalami gangguan mudah diinterpretasi (baik oleh manusia maupun mesin), maka citra tersebut perlu dimanipulasi menjadi citra lain yang kualitasnya lebih baik. Umumnya, operasi – operasi pada pengolahan citra diterapkan pada citra bila :

1. Perbaikan atau memodifikasi citra perlu dilakukan untuk meningkatkan kualitas penampakan atau untuk menonjolkan beberapa aspek informasi yang terkandung di dalam citra.
2. Elemen di dalam citra perlu dikelompokan, dicocokan atau diukur.
3. Sebagian citra perlu digabung dengan bagian citra yang lain.

2.1.3. Model Citra

Citra ada dua macam yaitu citra kontinu dan citra diskrit. Citra kontinu dihasilkan dari sistem optik yang menerima sinyal analog, misalnya mata manusia dan kamera analog. Citra diskrit dihasilkan melalui proses digitalisasi terhadap citra kontinu. Beberapa sistem optik dilengkapi dengan fungsi digitalisasi sehingga mampu menghasilkan citra diskrit, misalnya kamera digital dan *scanner*. Citra diskrit disebut juga citra digital. Komputer digital saat ini hanya dapat mengelola citra digital. Citra merupakan fungsi malar (kontinu) dari intensitas

cahaya pada bidang dwimatra. Secara matematis fungsi intensitas cahaya pada bidang dwimatra disimbolkan dengan f (x,y) yang dalam hal ini :

(x,y) : koordinat pada bidang dwimatra.

f (x,y) : intensitas cahaya (*brightness*) pada bidang dwimatra.

Gambar 2.2 memperlihatkan posisi koordinat pada bidang citra. Sistem koordinat yang diacu adalah sistem koordinat kartesian, yang dalam hal ini sumbu mendatar menyatakan sumbu x, dan sumbu tegak menyatakan sumbu y.



Gambar 2.2 Cara menentukan koordinat tititk di dalam citra [5]

Karena cahaya merupakan bentuk energi, maka intensitas cahaya bernilai antara 0 sampai tidak berhingga, sehingga di dapatkan persamaan seperti pada persamaan (2.1).

$$0 \leq f(x,y) < \infty \quad \text{.....} (2.1)$$

Nilai f (x,y) sebenarnya adalah hasil kali dari :

1. i (x,y) = jumlah cahaya yang berasal dari sumbernya (*illumination*), nilainya antara 0 sampai tidak berhingga.
2. r (x,y) = derajat kemampuan objek memantulkan cahaya (*reflection*), nilainya antara 0 dan 1.

Gambar 2.3 memperlihatkan proses pembentukan intensitas cahaya. Sumber cahaya menyinari permukaan objek. Jumlah pancaran (iluminasi) cahaya yang diterima objek pada koordinat (x,y) adalah i (x,y). Objek memantulkan cahaya yang diterimanya dengan derajat pantulan r (x,y). Hasil kali antara i (x,y) dan r (x,y) menyatakan intensitas cahaya pada koordinat (x,y) yang ditangkap

oleh sensor visual pada sistem optik seperti pada persamaan (2.2) kemudian penjelasannya penguraian rumus berasal dari persamaan (2.3) dan (2.4) sehingga penggabungan keduanya akan menjadi persamaan (2.5).

$$f(x,y) = i(x,y).r(x,y) \quad \text{.....(2.2)}$$

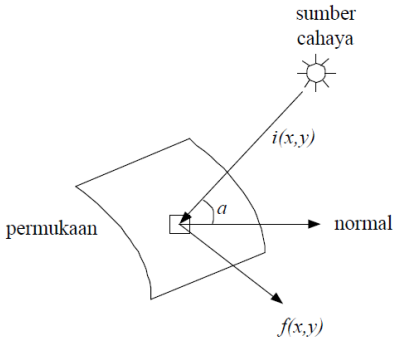
yang dalam hal ini

$$0 \leq i(x,y) < \infty \quad \text{.....(2.3)}$$

$$0 \leq r(x,y) \leq 1 \text{.....(2.4)}$$

Sehingga

$$0 \leq f(x,y) < \infty \text{.....(2.5)}$$



Gambar 2.3 Pembentukan Citra [5]

Nilai *i*(*x*, *y*) ditentukan oleh sumber cahaya, sedangkan *r*(*x*, *y*) ditentukan oleh karakteristik objek di dalam gambar. Nilai *r*(*x*,*y*) = 0 mengindikasikan penerapan total, sedangkan *r*(*x*,*y*) = 1 menyatakan pemantulan total. Jika permukaan mempunyai derajat pemantulan nol, maka fungsi intensitas cahaya, *f*(*x*, *y*), juga nol. Sebaliknya, jika permukaan mempunyai derajat pemantulan 1, maka fungsi intensitas cahaya sama dengan iluminasi yang diterima oleh permukaan tersebut. Intensitas *f* dari gambar hitam putih pada titik (*x*, *y*) disebut *grey level* (derajat keabuan), yang dalam hal ini derajat keabuannya bergerak dari hitam ke putih, sedangkan citranya disebut *greyscale image* (citra hitam putih)

atau *monochrome image* (citra *monochrome*). Derajat keabuan memiliki rentang nilai dari l_{min} sampai $l_{\max}$ atau $l_{min} < f < l_{\max.}$ Selang ($l_{min,}$ $l_{\max}$) disebut skala keabuan.

Biasanya selang ($l_{\min}$, $l_{\max}$) sering digeser untuk alasan-alasan praktis menjadi selang [0, L], yang dalam hal ini nilai intensitas 0 menyatakan hitam, nilai intensitas L menyatakan putih, sedangkan nilai intensitas antara 0 sampai L bergeser dari hitam ke putih. Sebagai contoh, citra hitam-putih dengan 256 *level* artinya mempunyai skala abu dari 0 sampai 255 atau [0, 255], yang dalam hal ini nilai intensitas 0 menyatakan hitam, nilai intensitas 255 menyatakan putih, dan nilai antara 0 sampai 255 menyatakan warna keabuan yang terletak antara hitam dan putih. Citra hitam-putih disebut juga citra satu kanal, karena warnanya hanya ditentukan oleh satu fungsi intensitas saja. Citra berwarna (*color images*) dikenal dengan nama citra spektral, karena warna pada citra disusun oleh tiga komponen warna yang disebut komponen *RGB*, yaitu merah (*red*), hijau (*green*), dan biru (*blue*). Intensitas suatu titik pada citra berwarna merupakan kombinasi dari tiga intensitas: derajat keabuan merah ($f_{\text{merah}}(x,y)$), hijau ($f_{\text{hijau}}(x,y)$), dan biru ($f_{\text{biru}}(x,y)$).

2.1.4 Digitalisasi Citra

Agar dapat diolah dengan komputer digital, maka suatu citra harus direpresentasikan secara numerik dengan nilai-nilai *diskrit*. Representasi citra dari fungsi malar (kontinu) menjadi nilai-nilai diskrit disebut *digitalisasi* . Citra yang dihasilkan inilah yang disebut citra digital (*digital image*). Pada umumnya citra digital berbentuk empat persegi panjang, dan dimensi ukurannya dinyatakan sebagai tinggi x lebar (atau lebar x panjang).

Citra digital yang tingginya N, lebarnya M, dan memiliki L derajat keabuan dapat dianggap sebagai fungsi seperti pada persamaan (2.6).

$$f(x,y)\begin{cases}0 \le x \le M\\ 0 \le y \le N\\ 0 \le f \le L\end{cases} \quad \text{.....(2.6)}$$

Citra digital yang berukuran N x M lazim dinyatakan dengan matriks yang berukuran N baris dan M kolom seperti pada persamaan (2.7).

$$f(x,y) \approx \begin{bmatrix} f(0,0) & \cdots & f(0,M) \\ \vdots & \ddots & \vdots \\ f(N-1,0) & \cdots & f(N-1,M-1) \end{bmatrix} \quad \text{.....(2.7)}$$

Indeks baris (*i*) dan indeks kolom (*j*) menyatakan suatu koordinat titik pada citra, sedangkan *f* (*i, j*) merupakan intensitas (derajat keabuan) pada titik (*i, j*). Masing-masing elemen pada citra digital (berarti elemen matriks) disebut *image element*, *picture element* atau *pixel* atau *pel*. Jadi, citra yang berukuran *N* x *M* mempunyai *NM* buah *pixel*. Sebagai contoh, misalkan sebuah berukuran 256 x 256 *pixel* dan direpresentasikan secara numerik dengan matriks yang terdiri dari 256 buah baris (di-indeks dari 0 sampai 255) dan 256 buah kolom (di-indeks dari 0 sampai 255) seperti contoh berikut pada persamaan (2.8).

$$\begin{bmatrix} 0 & 134 & 231 \\ \vdots & \ddots & \vdots \\ 221 & \cdots & 156 \end{bmatrix} \quad \text{.....(2.8)}$$

Pada matriks persamaan 2.7 *pixel* pertama pada koordinat (0, 0) mempunyai nilai intensitas 0 yang berarti warna *pixel* tersebut hitam, *pixel* kedua pada koordinat (0, 1) mempunyai intensitas 134 yang berarti warnanya antara hitam dan putih, dan seterusnya. Proses digitalisasi citra ada dua macam:

1. Digitalisasi spasial (*x, y*), sering disebut sebagai penerokan (*sampling*). *Sampling* menyatakan besarnya kotak-kotak yang disusun dalam baris dan kolom. Dengan kata lain sampling pada citra menyatakan besar kecilnya ukuran piksel pada citra. Untuk memudahkan implementasi, jumlah sampling biasanya diasumsikan perpangkatan dari dua seperti pada persamaan (2.9).

   $$N = 2^n \quad \text{.....(2.9)}$$

   Dimana,

   N = jumlah sampling pada suatu baris atau kolom

   n = bilangan bulat positif

   Pembagian gambar menjadi ukuran tertentu menentukan resolusi spasial yang diperoleh. Semakin tinggi resolusinya, yang berarti semakin kecil ukuran piksel (atau semakin banyak jumlah pikselnya), semakin halus gambar yang diperoleh karena informasi yang hilang akibat pengelompokkan derajat keabuan pada pen-*sampling*-an semakin kecil.

2. Digitalisasi intensitas *f* (*x*, *y*), sering disebut sebagai *kuantisasi*.

   Setelah proses *sampling* pada citra maka proses selanjutnya adalah kuantisasi. Kuantisasi menyatakan besarnya nilai tingkat kecerahan yang dinyatakan dalam nilai tingkat keabuan (*grayscale*) sesuai dengan jumlah bit biner yang digunakan, dengan kata lain kuantisasi pada citra menyatakan jumlah warna yang ada pada citra. Proses kuantisasi membagi skala keabuan (0, L) menjadi G buah level yang dinyatakan dengan suatu harga bilangan bulat (integer), biasanya G diambil perpangkatan dari 2, seperti pada persamaan (2.10). Pada tabel 2.1 ditunjukkan hasil kuantisasi citra dengan skala keabuan yang berbeda-beda.

$$G = 2^m \quad \text{.....(2.10)}$$

   Dimana,

   G = derajat keabuan

   m = bilangan bulat positif

Tabel 2.1 Tabel Kuantisasi Citra dengan Skala Keabuan yang Berbeda

Skala Keabuan	Rentang Nilai Keabuan	Pixel Depth
2^1 (2 nilai)	0,1	1 bit
2^2 (4 nilai)	0 sampai 3	2 bit
2^4 (16 nilai)	0 sampai 15	4 bit
2^8 (256 nilai)	0 sampai 255	8 bit

   Jumlah bit yang dibutuhkan untuk mempresentasikan nilai keabuan piksel disebut kedalaman piksel (*pixel depth*). Citra sering diasosiasikan dengan kedalaman pikselnya. Jadi, citra dengan kedalaman 8 bit disebut juga citra 8-bit (atau citra 256 warna, $G = 256 = 2^8$). Semakin banyak jumlah derajat keabuan (berarti jumlah bit kuantisasinya makin banyak), semakin bagus gambar yang diperoleh.

2.1.5. Kategori Pengolahan Citra

Pengolahan citra dapat dibagi ke dalam tiga kategori yakni, kategori rendah, menengah, dan tinggi.

1. Kategori rendah, melibatkan operasi-operasi sederhana seperti prapengolahan citra untuk mengurangi derau, pengaturan kontras, dan pengaturan ketajaman citra. Pengolahan kategori rendah ini memiliki input dan output berupa citra.
2. Kategori menengah, melibatkan operasi-operasi seperti segmentasi dan klasifikasi citra. Proses pengolahan citra menengah ini melibatkan input berupa citra dan output berupa atribut (fitur) citra yang dipisahkan dari citra input.
3. Kategori tinggi, melibatkan proses pengenalan dan deskripsi citra.

Penelitian tugas akhir ini proses pengolahan citra masuk tergolong kedalam kategori tinggi karena melibatkan proses pengenalan dan deskripsi citra.

2.1.6. Pengolahan Citra Digital

Pengolahan citra adalah kegiatan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia/mesin (komputer). Data masukan adalah citra dan keluarannya juga citra tapi dengan kualitas lebih baik dari pada citra masukan. Proses pengolahan citra yang digunakan pada penelitian ini terdiri dari proses *scaling,grayscale,edge detection (sobel),tresholding.*

2.1.6.1. Scaling

Scaling merupakan proses pengubahan ukuran gambar digital, hal ini perlu dilakukan agar semua gambar digital data masukan memiliki ukuran yang sama dengan memperkecil citra[11]. Menghitung skala perubahan panjang dan lebar dengan terlebih dahulu menentukan skala baru dihitung menggunakan persamaan (2.11) dan (2.12).

$$nwidthfactor = \frac{width}{nwidth} \quad \text{.....(2.11)}$$

$$nheightfactor = \frac{height}{nheight} \quad \text{.....(2.12)}$$

Menentukan koordinat pixel disekitar matriks yang sedang diulang, x dan y dengan menggunakan persamaan (2.13) – (2.20).

$$fr_x = x * nwidthfactor \quad \text{.....(2.13)}$$

$$fr_y = y * nheigthfactor \quad \text{.....(2.14)}$$

$$C_x = fr_x + 1 \quad \text{.....(2.15)}$$

$$C_y = fr_y + 1 \quad \text{.....(2.16)}$$

$$f_x = x * nwidthfactor - fr_x \quad (2.17)$$

$$f_y = y * nheightfactor - fr_y \quad (2.18)$$

$$n_x = 1{,}0 - f_x \quad (2.19)$$

$$n_y = 1{,}0 - f_y \quad (2.20)$$

Mengambil nilai pixel pada 4 koordinat menggunakan persamaan (2.21) – (2.24) .

$$C_1 = pixel\ (fr_x, fr_y) \quad (2.21)$$

$$C_2 = pixel\ (C_x, fr_y) \quad (2.22)$$

$$C_3 = pixel\ (fr_x, C_y) \quad (2.23)$$

$$C_4 = pixel\ (C_x, C_y) \quad (2.24)$$

Menghitung nilai RGB baru dengan menggunakan nilai RGB dari keempat koordinat yang sudah diambil dengan menggunakan persamaan (2.25) – (2.33).

$$b_1 = (n_x * C_1.R + f_x * C_2.R) \quad (2.25)$$

$$b_2 = (n_x * C_3.R + f_x * C_4.R) \quad (2.26)$$

$$R = (n_x * b_1 + f_y * b_2) \quad (2.27)$$

$$b_1 = (n_x * C_1.G + f_x * C_2.G) \quad (2.28)$$

$$b_2 = (n_x * C_3.G + f_x * C_4.G) \quad (2.29)$$

$$G = (n_x * b_1 + f_y * b_2) \quad (2.30)$$

$$b_1 = (n_x * C_1.B + f_x * C_2.B) \quad (2.31)$$

$$b_2 = (n_x * C_3.B + f_x * C_4.B) \quad (2.32)$$

$$B = \left(n_x * b_1 + f_y * b_2\right) \quad (2.33)$$

2.1.6.2. Grayscale

Citra *grayscale* merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pixelnya, dengan kata lain nilai bagian *RED = GREEN = BLUE*. Nilai tersebut digunakan untuk menunjukan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan, dan putih. Tingkatan kebuan disini merupakan warna abu dengan berbagai dari hitam hingga mendekati putih. Citra *grayscale* memiliki kedalaman warna 8 bit (256 kombinasi warna keabuan). Rentang nilai warna – warna pada RGB untuk Red (merah) rentangnya (255,0,0),

Green (hijau) rentangnya (0,255,0), Blue (biru) rentangnya (0,0,255), dan Mediun Gray (abu) rentangnya (127,127,127). Misalkan nilai RGB warna adalah (r, g, b), dimana r, g, dan b adalah bilangan bulat antara 0 dan 255. Rata-rata tertimbang grayscale, x, diberikan oleh persamaan (2.34).

$$x = (r + g + b)/3 \quad \text{.....(2.34)}$$

Contoh, sebuah bayangan gelap ungu memiliki nilai RGB (100, 0, 150). Rata-rata tertimbang adalah x = (100 + 0 +150)/3 = 83 , Jadi (83,83,83) hasilnya adalah abu – abu.

Tabel 2.2 Tabel Warna RGB [8]

Colorname	HEX	RGB
Black	000000	0,0,0
Blue	0000FF	0,0,255
Green	008000	0,128,0
Red	FF0000	255,0,0

2.1.6.3. *Edge Detection* (*Sobel*)

Metode ini merupakan pengembangan metode robert. Kelebihan dari metode sobel ini adalah kemampuan untuk mengurangi *noise* sebelum melakukan perhitungan deteksi tepi sehingga tepi-tepi yang dihasilkan lebih banyak dibanding dua metode sebelumnya. Operator terdiri dari sepasang kernel 3 × 3 konvolusi seperti yang ditunjukkan pada Gambar 2.4 Satu kernel yang lain hanya diputar oleh 90 °.

1	0	-1
2	0	-2
1	0	-1

a. Gx

1	2	1
0	0	0
-1	-2	-1

b. Gy

Gambar 2.4 Matriks Sobel 3 x 3

Kernel ini dirancang untuk merespon secara maksimal untuk tepi berjalan secara vertikal dan horizontal relatif terhadap grid pixel, satu kernel untuk

masing-masing dua orientasi tegak lurus. Kernel dapat diterapkan secara terpisah dengan citra masukan, untuk menghasilkan pengukuran yang terpisah dari komponen gradien di setiap orientasi (menyebut Gx dan Gy). Ini kemudian dapat digabungkan bersama-sama untuk mencari besarnya nilai mutlak gradien pada setiap titik dan orientasi gradien itu [6]. Besarnya gradien diberikan persamaan (2.35).

$$|G| = \sqrt{Gx^2 + Gy^2} \quad \text{.....(2.35)}$$

Dengan, G = besar gradien operator sobel,

G_x = Gradien sobel arah horizontal

G_y = Gradien sobel arah vertikal

Biasanya, sebuah perkiraan besarnya dihitung dengan menggunakan persamaan (2.36).

$$|G| = |Gx| + |Gy| \text{.....(2.36)}$$

yang lebih cepat untuk menghitung.

Sudut orientasi tepi (relatif ke grid pixel) menimbulkan gradien spasial diberikan oleh persamaan (2.37).

$$\theta = \arctan\left(\frac{Gy}{Gx}\right) \text{.....(2.37)}$$

2.1.6.4. *Tresholding*

Merupakan tahap untuk membuat warna citra menjadi hanya berwarna hitam atau putih. Untuk melakukan *thresholding* terlebih dahulu harus dicari suatu nilai yang digunakan untuk batas toleransi suatu warna keabuan dapat dijadikan warna hitam atau warna putih, nilai tersebut biasa disebut nilai ambang. Setelah semua piksel diubah menjadi nilai keabuan, pencarian nilai ambangpun dilakukan. Ada beberapa cara menentukan nilai ambang yaitu pertama, dengan menentukan langsung berdasarkan nilai tengah dari level keabuan yaitu 128. Kedua, menentukan suatu nilai tertentu sebagai nilai ambang. Ketiga, menentukan nilai rata – rata dari total piksel keabuan dalam citra dan menjadikannya sebagai nilai ambang rata – rata. Kemudian setelah didapat nilai ambang tersebut maka akan dilakukan perbandingan setiap nilai piksel dari citra keabuan tersebut dengan nilai ambangnya. Jika nilai piksel lebih besar dari nilai ambangnya maka nilai piksel

tersebut akan diubah menjadi 0 (piksel berwarna hitam). Sedangkan jika yang ditemukan sebaliknya yaitu nilai pikselnya lebih kecil dari nilai ambang yang telah ditentukan maka nilai piksel tersebut diubah menjadi 1 (piksel berwarna putih) [6]. Cara membuat nilai pixel menjadi *tresholding* menggunakan persamaan (2.38).

$$f(x,y) = \begin{cases} 0 \; if \; f(x,y) < T \\ 1 \; if \; f(x,y) \geq T \end{cases} \quad \text{.....(2.38)}$$

Keterangan :

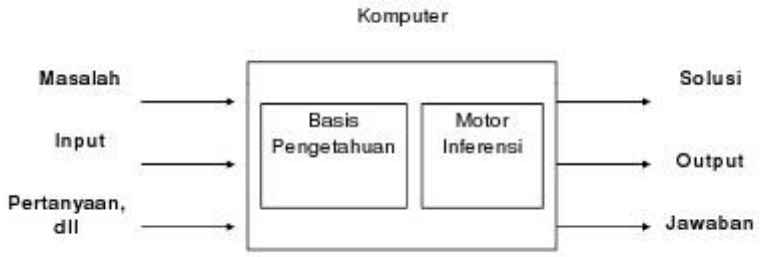
f (x,y) : citra biner dari citra *grayscale*

T : nilai ambang

2.2. Kecerdasan Buatan

Kecerdasan buatan *(Artificial Intelligence)* merupakan salah satu ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia. Kecerdasan buatan mempunyai dua bagian utama yang sangat dibutuhkan dalam melakukan aplikasi, seperti terlihat pada gambar 2.5 komponen yang dibutuhkan adalah :

a. Basis Pengetahuan *(Knowledge Base)*, berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya.
b. Motor Inferensi *(Inference Engine),* yaitu kemampuan menarik kesimpulan berdasarkan pengalaman.



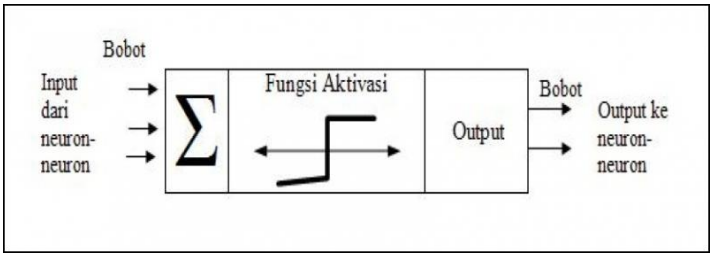
Gambar 2.5 Penerapan Kecerdasan Buatan [7]

Kecerdasan buatan terdiri dari beberapa kajian diantaranya adalah sistem pakar, jaringan syaraf tiruan, *computer vision*, simulasi *crowd*, logika *fuzzy*, algoritma genetika, dokumen *minning.*

2.3. Jaringan Syaraf Tiruan

Jaringan syaraf tiruan bisa dibayangkan seperti otak buatan di dalam cerita – cerita fiksi ilmiah. Otak buatan ini dapat berpikir seperti manusia, dan juga sepandai manusia dalam menyimpulkan sesuatu dari potongan – potongan informasi yang diterima. Khayalan manusia tersebut mendorong para peneliti untuk mewujudkannya. Komputer diusahakan agar bisa berpikir sama seperti cara berpikir manusia. Caranya adalah dengan melakukan peniruan terhadap aktivitas – aktivitas yang terjadi di dalam sebuah jaringan saraf biologis [8].

Komponen JST terdiri dari neuron – neuron dan neuron – neuron tersebut saling berhubungan. Neuron – neuron tersebut akan mentransformasikan informasi yang diterima melalui sambungan keluarnya menuju neuron – neuron yang lain, pada JST hubungan neuron - neuron tersebut dikenal dengan nama bobot. Komponen JST seperti terlihat pada Gambar 2.6 terdiri dari input (informasi) bobot (nilai- nilai tertentu), fungsi aktivasi berfungsi apabila input yang dimasukan sesuai dengan nilai ambang yang ditentukan jika tidak sesuai maka fungsi aktivasi tidak diaktifkan, dan apabila neuron – neuron tersebut diaktifkan maka neuron akan mengirimkan output melalui bobot – bobot.



Gambar 2.6 Komponen Jaringan Syaraf Tiruan [6]

Jaringan Syaraf tiruan mempunyai dua macam proses pembelajaran yaitu pembelajaran terawasi dan pembelajaran tidak terawasi. Pembelajaran terawasi adalah jika output yang diharapkan telah diketahui sebelumnya, sedangkan pembelajaran tidak terawasi adalah proses pembelajaran yang tidak memerlukan target output.

1. Pembelajaran Terawasi (*Supervised Learning*)

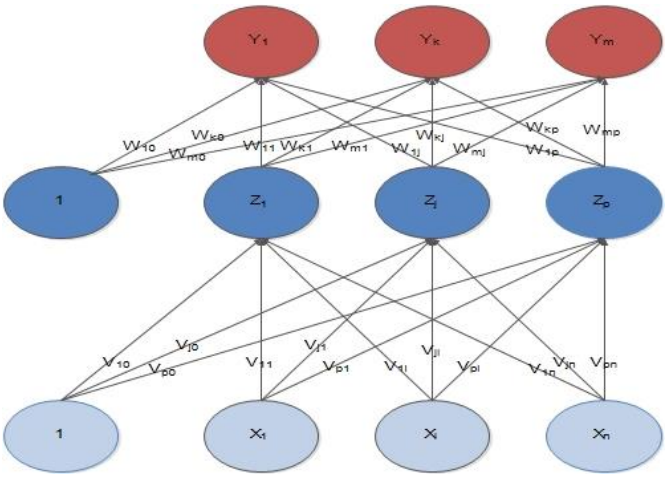
   Metode pembelajaran pada jaringan syaraf disebut terawasi jika output yang diharapkan telah diketahui sebelumnya. Pada proses pembelajaran,

satu pola input akan diberikan ke satu neuron pada lapisan input. Pola ini akan dirambatkan di sepanjang jaringan syaraf hingga sampai ke neuron pada lapisan output. Lapisan output ini akan membangkitkan pola output yang nantinya akan dicocokkan dengan pola output targetnya. Apabila terjadi perbedaan antara pola output hasil pembelajaran dengan pola target, maka disini akan muncul error. Apabila nilai error ini masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi. Metode pembelajaran terawasi diantaranya *Hebb Rule, Perceptron, Delta Rule, Backpropagation, Learning Vektor Quantization, Heteroassociative Memory*.

2. Pembelajaran Tak Terawasi (*Unsupervised Learning*) pada metode pembelajaran yang tak terawasi ini tidak memerlukan target output. Pada metode ini, tidak dapat ditentukan hasil yang seperti apakah yang diharapkan selama proses pembelajaran. Metode pembelajaran tidak terawasi salah satunya adalah jaringan kohonen.

2.4. Arsitektur *Backpropagation*

Backpropagation memiliki beberapa unit yang ada dalam satu atau lebih layar tersembunyi. Gambar 2.7 adalah arsitektur *backpropagation* dengan x buah masukan (ditambah sebuah bias), sebuah layar tersembunyi yang terdiri dari z unit (ditambah sebuah bias), serta m buah unit keluaran. V_{ji} merupakan bobot garis dari unit masukan X_i ke unit layar tersembunyi Z_j (V_{j0} merupakan bobot garis yang menghubungkan bias di unit masukan ke unit layar tersembunyi Z_j). W_{kj} merupakan bobot dari unit layar tersembunyi Z_j ke unit keluaran Y_k (W_{k0} merupakan bobot dari bias dilayar tersembunyi ke unit keluaran Z_k).



Gambar 2.7 Arsitektur Jaringan *Backpropagation*

2.5. Backpropagation

Backpropagetion atau propagasi balik suatu algoritma pembelajaran dan digunakan oleh perceptron dengan banyak lapisan untuk mengubah bobot – bobot yang terhubung dengan neuron – neuron yang terdapat pada lapisan tersembunyi [7]. Algoritma propagasi balik menggunakan *error output* agar dapat mengubah nilai bobot – bobotnya dalam proses mundur *(backward)* untuk mendapatkan error ini, tahap perambatan maju, neuron – neuron diaktifkan dengan menggunakan fungsi aktivasi sigmoid, sedangkan dalam melakukan sebuah proses perambatan maju dan mundur disebut *epoch* atau iterasi, aktivasi sigmoid biner yang memiliki rentang nilai (0,1), yaitu seperti pada persamaan (2.39).

$$f(x) = \frac{1}{1+e^{-x}} \quad \text{.....(2.39)}$$

dengan turunan pada persamaan (2.40)

$$f'(x) = f(x)\big(1 - f(x)\big) \quad \text{.....(2.40)}$$

Fungsi lain yang sering dipakai adalah fungsi sigmoid bipolar yang bentuk fungsinya mirip dengan fungsi sigmoid biner, tapi dengan range (-1,1) seperti pada persamaan (2.41).

$$f(x) = \frac{2}{1+e^{-x}} - 1 \quad \text{.....(2.41)}$$

dengan turunan pada persamaan (2.42).

$$f'(x) = \frac{(1+f(x))(1-f(x))}{2} \quad \text{.....(2.42)}$$

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_i) tersebut selanjutnya dipropagasikan maju lagi ke layar tersembunyi di atasnya menggunakan fungsi aktivasi yang ditentukan. Demikian seterusnya hingga menghasilkan keluaran jaringan (= y_k). Berikutnya, keluaran jaringan (= y_k) dibandingkan dengan target yang harus dicapai (= 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 (w dan v) setiap garis dalam jaringan akan dimodifikasi untuk mengurangi kesalahan yang terjadi.

Fase II : Propagasi mundur

Berdasarkan kesalahan t_k - y_k dihitung faktor δ_k (k = 1,2,...,m) yang dipakai untuk mendistribusikan kesalahan di unit y_k ke semua unit tersembunyi yang terhubung langsung dengan y_k. δ_k juga dipakai untuk mengubah bobot garis yang berhubungan langsung dengan unit keluaran. Dengan cara yang sama, dihitung faktor δ_j di setiap unit di layar tersembunyi sebagai dasar perubahan bobot semua garis yang berasal dari unit tersembunyi di layar 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.

Berdasarkan penjelasan terhadap fase yang ada didalam *backpropagation* maka algoritma pelatihan untuk jaringan 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 pasangan data pelatihan, lakukan langkah 3-8

Fase I : Propagasi maju

Langkah 3 : Tiap unit masukan menerima sinyal dan meneruskannya ke unit tersembunyi diatasnya.

Langkah 4 : Hitung semua keluaran di unit tersembunyi z_j (j=1,2,..p) dengan menggunakan persamaan (2.43) dan (2.44).

$$z_net_j = \dot{V_{jo}} + \sum_{i=1}^{n} x_i v_{ji} \quad \text{.....(2.43)}$$

$$z_j = f(z_net_j) = \frac{1}{1+e^{-z_{net_j}}} \quad \text{.....(2.44)}$$

Langkah 5 : Hitung semua keluaran jaringan di unit y_k (k=1,2,..m) dengan menggunakan persamaan (2.45) dan (2.46).

$$y_net_k = w_{ko} + \sum_{j=1}^{p} z_i w_{kj} \quad \text{.....(2.45)}$$

$$y_k = f(y_net_k) = \frac{1}{1+e^{-y_{net_k}}} \quad \text{.....(2.46)}$$

Fase II : Propagasi mundur

Langkah 6 : Hitung faktor δ unit keluaran berdasarkan kesalahan di setiap unit keluaran y_k (k = 1,2,...,m) dengan menggunakan persamaan (2.47).

$$\delta_k = (t_k - y_k)\dot{f}'(y_net_k) = (t_k - y_k)y_k(1 - y_k) \quad \text{.....(2.47)}$$

δ_k merupakan unit kesalahan yang akan dipakai dalam perubahan bobot layar dibawahnya (langkah 7). Hitung suku perubahan bobot w_{kj} (yang akan dipakai nanti untuk merubah bobot w_{kj} dengan laju percepatan α dengan menggunakan persamaan (2.48).

$$\Delta w_{kj} = \alpha \delta_k z_j \quad ; k = 1,2,..,m; j = 0,1,\dots,p \quad \text{.....(2.48)}$$

Langkah 7 : Hitung faktor δ unit tersembunyi berdasarkan kesalahan disetiap unit tersembunyi z_j (j = 1,2,...,p) dengan menggunakan persamaan (2.49).

$$\delta_net_j = \sum_{k=1}^{m} \delta_k w_{kj} \quad \text{.....(2.49)}$$

Hitung faktor δ unit tersembunyi dengan menggunakan persamaan (2.50).

$$\delta_j = \delta_net_j \dot{f}'(z_net_j) = \delta_net_j z_j(1 - z_j) \quad \text{.....(2.50)}$$

Hitung suku perubahan bobot v_{ji} (yang akan dipakai untuk merubah bobot v_{ji}) dengan menggunakan persamaan (2.51).

$$\Delta v_{ji} = \alpha \delta_j x_i \;\; ; j = 1,2,..p \; ; i = 0,1,..n \quad \text{.....(2.51)}$$

Fase III : Perubahan bobot

Langkah 8 : Hitung semua perubahan bobot dengan menggunakan persamaan (2.52) dan (2.53).

Perubahan bobot garis yang menuju ke unit keluaran :

$$w_{kj}(baru) = w_{kj}(lama) + \Delta w_{kj} \; (k = 1,2,\dots,m \; ; j = 0,1,\dots,p) \quad \text{....(2.52)}$$

Perubahan bobot garis menuju ke unit tersembunyi :

$$v_{ji}(baru) = v_{ji}(lama) + \Delta v_{ji} \; (j = 1,2,\dots,p \; ; i = 0,1,\dots,n) \quad \text{.....(2.53)}$$

Setelah pelatihan selesai dilakukan, jaringan dapat dipakai untuk pengenalan pola.

2.6. Pemograman Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara matematis. Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana perangkat lunak di bangunmelalui pendekatan objek secara sistematis [1].

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut :

1. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih bisa dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).
2. Kecepatan pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan dalam pengkodean.
3. Kemudahan pemeliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.
4. Adanya konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analis, perancangan maupun pengkodean.
5. Meningkatkan kualitas perangkat lunak

   Karena pendekatan pembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek :

a. *Class*

Class adalah kumpulan objek – objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama mungkin lahir atau diciptakan dengan kelas tersebut. Sebuah kelas mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (relationship) dan arti.

b. Objek

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, suatu organisasi, tempat, kejadian, struktur, status atau hal – hal lain yang bersifat abstrak.

c. Metode

Operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi.

d. Atribut

Atribut adalah sebuah kelas variabel global yang dimiliki sebuah kelas, atribut dapat berupa nilai atau elemen – elemen data yang dimiliki oleh objek dalam kelas objek.

e. Abstraksi

Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek – aspek lain yang tidak sesuai dengan permasalahan.

f. Enkapsulasi

Pembungkusan atribut data dan layanan yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

g. Pewarisan

Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.

h. Antarmuka

Antarmuka sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi.

i. Reusability

Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

j. Generalisasi dan Spesialisasi

Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

k. Komunikasi Antar Objek

Komunikasi antar – objek dilakukan lewat pesan yang dikirim dari satu objek

ke objek lainnya.

l. Polimorfisme

Kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

m. Package

Package adalah sebuah komentar atau kemasan yang dapat digunakan untuk mengelompokkan kelas – kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda.

2.7. UML *(Unified Modelling Language)*

UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram teks – teks pendukung. Berbagai diagram pada UML antara lain [1] :

a. Class Diagram

Diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas – kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

b. Object Diagram

Diagram object menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem.

c. Component Diagram

Diagram komponen dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem.

d. Package Diagram

Menyediakan cara mengumpulkan elemen – elemen yang saling terkait dalam diagram UML. Hampir semua diagram dalam uml dapat dikelompokkan menggunakan package diagram.

e. Deployment Diagram

Diagram ini menunjukkan konfigurasi komponen dalam proses eksekusi

aplikasi.

f. Use Case Diagram

Merupakan pemodelan kelakuan sistem informasi yang akan dibuat. Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.

g. Activity Diagram

Diagram aktivitas menggambarkan workflow atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.

h. Sequence Diagram

Diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek.

2.8. IDE yang Digunakan

IDE atau singkatan dari (Integrated Development Environment adalah program komputer yang memiliki beberapa fasilitas yang diperlukan dalam pembangunan perangkat lunak. Tujuan dari IDE adalah untuk menyediakan semua utilitas yang diperlukan dalam membangun perangkat lunak[10].

Sebuah IDE, atau secara bebas dapat diterjemahkan sebagai Lingkungan Pengembangan Terpadu, setidaknya memiliki fasilitas:

a. Editor, yaitu fasilitas untuk menuliskan kode sumber dari perangkat lunak.
b. Compiler, yaitu fasilitas untuk mengecek sintaks dari kode sumber kemudian mengubah dalam bentuk binari yang sesuai dengan bahasa mesin.
c. Linker, yaitu fasilitas untuk menyatukan data binari yang beberapa kode sumber yang dihasilkan compiler sehingga data-data binari tersebut menjadi satu kesatuan dan menjadi suatu program komputer yang siap dieksekusi.
d. Debuger, yaitu fasilitas untuk mengetes jalannya program, untuk mencari bug/kesalahan yang terdapat dalam program.

Sampai tahap tertentu IDE modern dapat membantu memberikan saran yang mempercepat penulisan. Pada saat penulisan kode, IDE juga dapat menunjukan bagian-bagian yang jelas mengandung kesalahan atau keraguan.

2.8.1. Microsoft Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup kompiler, SDK, Integrated Development Environment (IDE), dan dokumentasi (umumnya berupa MSDN Library). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe.

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam native code (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun managed code (dalam bentuk Microsoft Intermediate Language di atas .NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework).

Visual Studio kini telah menginjak versi Visual Studio 9.0.21022.08, atau dikenal dengan sebutan Microsoft Visual Studio 2008 yang diluncurkan pada 19 November 2007, yang ditujukan untuk platform Microsoft .NET Framework 3.5. Versi sebelumnya, Visual Studio 2005 ditujukan untuk platform .NET Framework 2.0 dan 3.0. Visual Studio 2003 ditujukan untuk .NET Framework 1.1, dan Visual Studio 2002 ditujukan untuk .NET Framework 1.0. Versi-versi tersebut di atas kini dikenal dengan sebutan Visual Studio .NET, karena memang membutuhkan Microsoft .NET Framework. Sementara itu, sebelum muncul Visual Studio .NET, terdapat Microsoft Visual Studio 6.0 (VS1998). Dan setelah Visual Studio 2008 muncul versi Visual Studio 2010 dan sekarang yang digunakan adalah Visual Studio 2012[10].

{Halaman ini dikosongkan}